オープンＣＮＣマシニングセンタを用いた
知能化加工システムの研究

指導教員　　長尾高明　教授

平成１３年2月２８日提出

高知工科大学　知能機械システム工学科

１０１０１４３　　木崎　剛

目次

目次

第１章

序章

１．１　本研究の背景

コンピュータ技術の驚異的な発展により、工業製品の多くがメカトロニクス化され、多機能化、高機能化している。身近な日用品、電気器具、光学機械、カメラ、計測器などの他、機械、自動車、航空機に至るまで、およそ軽化思想の盛り込まれていないものはない。いずれもこれらの軽量化は、精度向上に結びつく。このような現状では、加工される部品の軽量化及び高精度化の要求は高まるばかりである。また、市場ニーズの激しい変化に常に適応するために、すばやく消費者の要求に対応することができる生産システムが求められている。

生産工場では今まで、高い生産性と安定した製品品質を維持するために、また人間を過酷な作業から解放させることを目的として、生産活動の自動化が追及されてきた。しかし近年の技術革新による、通信ネットワーク技術とコンピュータ技術の急速な向上により、その技術を利用して生産活動もさらなる発展が可能となると思われる。つまり、①ネットワークを利用した遠隔地から操作できるシステムと、製品の高性能化に欠かせない②精密加工を行なうシステム技術である。

①のメリットとして加工の現場と操作室を別にすることは、操作者の環境を快適で安全なものにするばかりか、すばやい加工データのやりとりにより製品設計から加工開始までの時間ロスを抑えることができ、市場ニーズに応じたすばやい対応ができる。

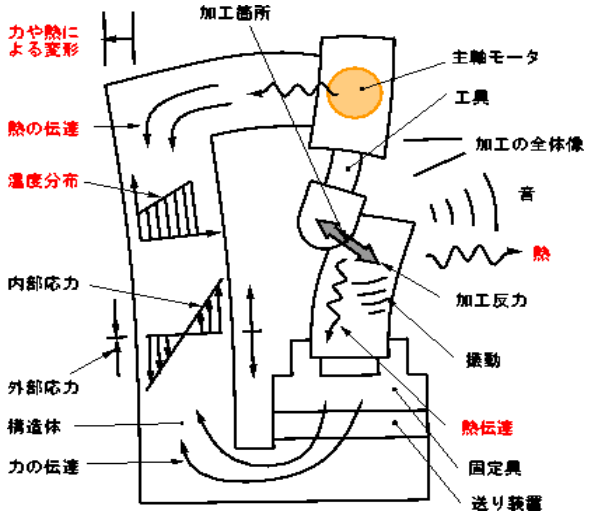
また②の技術は、製品の高性能化に向かう最近の工業製品を作ることにおいて欠くことのできない技術になるであろう。

このような２つの点に注目することによって、工業製品の高性能化と市場ニーズにすばやく対応することができ、かつそれを使う人間にとっても安全で快適に生産活動を行なうことができるのではないだろうか。それらを実現するため加工のネットワーク化と高精度加工化を目的とした知能化加工システムの構築が求められてきている。

１．２　知能化加工システムとは

工作機械が加工を行なうとき、機械および被加工物（ワーク）を含む系全体に（図１．１）に示すような種々の物理現象が生じている。これらの原因により、より精密な加工をする上の障害になっている。知能化加工システムにおいてはこれらの加工現象からの情報を取得し、その情報をもとに加工自体が制御される結果、所望の製品を作り出すことができる。また、生産拠点内で工作機械などがこのような情報を得る各種センサを装備しており、かつネットワークにつながっていて、それらの機械自身が知能化されているとともに、それら自身も情報を外部からうけ、また外部へ情報を発信することができる。これらのことによってあらゆる種類の制御が可能となり、また外部から制御することはもちろん、地球の裏側など遠隔地からのテレオペレーションすら可能となる。

多数の機械、多くの事業所が一つのコンピュータシステムの基に統合され、遠隔地から操作できる工作機械群よりなるシステム統合により、集中管理が容易になる。



（図１．１）加工で生じる種々の物理現象

１．３　本研究の目的

本研究では、まずオープンＣＮＣマシニングセンタを用い、加工中の加工力をリアルタイムで測定するシステムを開発する。それにより、加工力の変化の状況を知ることができる。次にマシニングセンタの遠隔操作システムを開発する。インターネットにつながったコンピュータなら、このソフトを使うことによって、どんなに遠く離れた場所でもマシニングセンタの起動、制御、位置データなどのやりとりをリアルタイムで行なえる。

また、ＮＣプログラムによってマシニングセンタが指定されたプログラム通りに起動している最中に、外部よりハンドルパルス信号を割りこますことによって、加工制御をおこなうプログラムを作成する。これを利用しリアルタイムに加工力を測定し、必然的に発生する加工誤差を修正する機能を構築したいところだが、今回はその第一歩として、外部ハンドルパルス割り込み機能を使い、簡単な加工制御をおこなうプログラムの作成と、実験を行なうことにより、この機能の有効性を証明する。

これらを組み合わせることにより、知能化加工システムの構築を目的とする。

１．４　本論文の構成

本論文は、以下の各章から構成されている。

第１章、研究の背景及びその目的。
第２章、本研究で利用したハードウェア、ソフトウェア、全体の構成。
第３章、力センサを用いた加工力モニタリングシステムの開発と構成。
第４章、オープンＣＮＣマシニングセンタの遠隔操作システムの開発と構成。
第５章、オープンＣＮＣマシニングセンタを用い、センサ情報による加工誤差修正機能を構築するための第一歩として、外部ハンドルパルス割り込み機能の構成とシステム構築、それを用いた実験。
第６章では、本論文の総合的考察と結論、及び今後の展望について述べる。

第２章

本研究のシステム構成

２．１　ハードウェアのシステム構成

ここでは本研究で用いたハードウェアの構成について述べる。

本研究で用いたマシニングセンタには、工作機械を自動制御し、外部コンピュータから制御可能なオープンＣＮＣ(Open Computer Numerical Control)装置はついており、ＨＳＳＢ回線(High Speed Serial Bus)を介し、制御用コンピュータより制御する。

知能化加工システムを構築するためには、加工力を測定することが不可欠である。そのため力センサが必要になる。本研究ではキスラー社製３成分力動力計を利用した。力センサの出力は、ストレイン・アンプを介してアナログ電圧信号に変換され、Ａ／Ｄ変換ボードに入力されてデジタル信号となり、制御用コンピュータに取り込まれる。

また、ＮＣプログラムで加工中に、ハンドル機能で割り込ませるためのパルス発生機能をもつＰＣＰＧボードも利用する。

２．１．１　オープンＣＮＣマシニングセンタ

マシニングセンタに外部コンピュータから制御可能な自動制御装置、オープンＣＮＣ(Open Computer Numerical Control)を取り付けたものである。

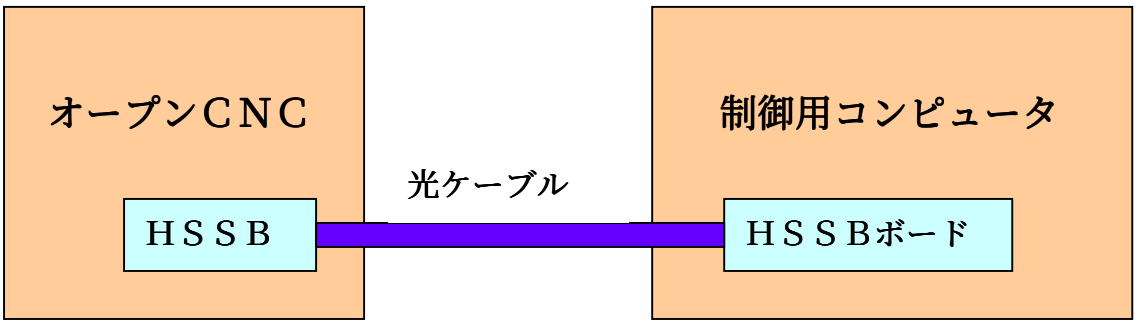
今回、利用したマシニングセンタは、大阪機工株式会社（ＯＫＫ）社製のＶＭ４Ⅱ（図２．１）を、またオープンＣＮＣはＦＡＮＡＣ社製のＳｅｒｉｅｓ１６０ｉ―Ｍ（図２．２）を使用した。

（図２．１）オープンＣＮＣマシニングセンタ＜ＯＫＫ　ＶＭ４Ⅱ＞

（図２．２）FANUC Series 160I-M

２．１．２　ＨＳＳＢ（High Speed Serial Bus）

制御用コンピュータとオープンＣＮＣとの間を結ぶ、光ファイバーケーブルである。ノイズの影響を受けない高速・広範囲のデータ転送が可能。コンピュータ側にはＨＳＳＢボードを組み込み、常にデータをやりとりすることで制御可能となる。



（図２．３）ＨＳＳＢ接続の概念図

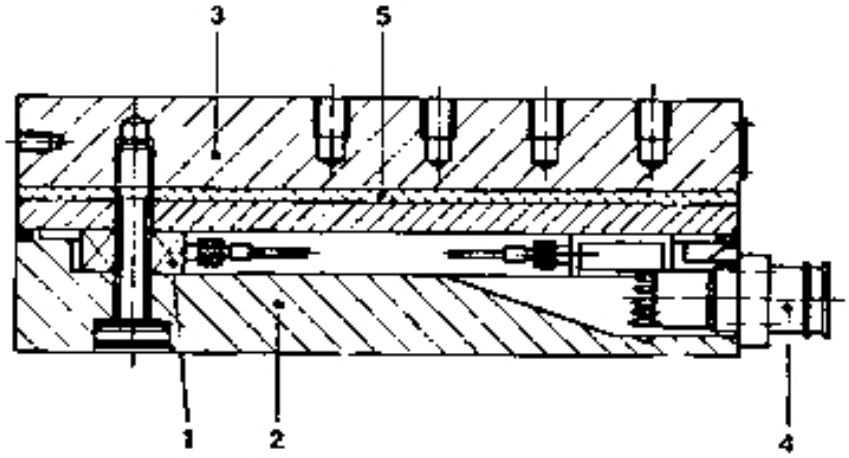
２．１．３　力センサ

力センサには、日本キスラー株式会社製の３成分動力計（型式９２５７Ｂ）を利用した。測定される力は、トップ・プレートを通して伝えられ、ベース・プレートとトップ・プレート間に配置された、４個の力変換器に分配される。測定する場合、個々の信号は接続ケーブルによって３成分にまとめられる。

動力計には防錆材が使用されており、水および冷却液の飛沫程度の進入に対しては保護されている。また、トッププレートには温度の影響を極力押さえるために、断熱コーティングを施している。

動力計の構造を（図２．５）に示す。同じく仕様を（表２．１）に示す。

（図２．４）　力センサの概観



1　力変換器
2　ベース・プレート
3　トップ・プレート
4　コネクタ
5　断熱コーティング

（図２．５）動力計の構造

寸法		mm	１４０×１７０
重量		ｋｇ	７．３
測定範囲	Ｆｘ、Ｆｙ、Ｆｚ	ｋＮ	－５～５
感度	Ｆｘ、Ｆｙ	ｐＣ／Ｎ	－７、５
	Ｆｚ	ｐＣ／Ｎ	－３、５
許容過負荷	Ｆｘ、Ｆｙ	ｋＮ	－７、５/７、５
	Ｆｚ	ｋＮ	－７、５/１５
使用温度範囲		℃	０～７０
静電容量		ｐＦ	２２０
設置絶縁		Ω	＞１０

（表２．１）３成分動力計の仕様

２．１．４ チャージアンプ

チャージアンプは力センサで発生した電荷をそれに比例した電圧に変換する装置である。本研究では日本キスラー社のマルチチャンネル・チャージアンプ（型式５０１９Ａ）を使用した。

以下、チャージアンプの概観（図２．６）と仕様（表２．２）を示す。

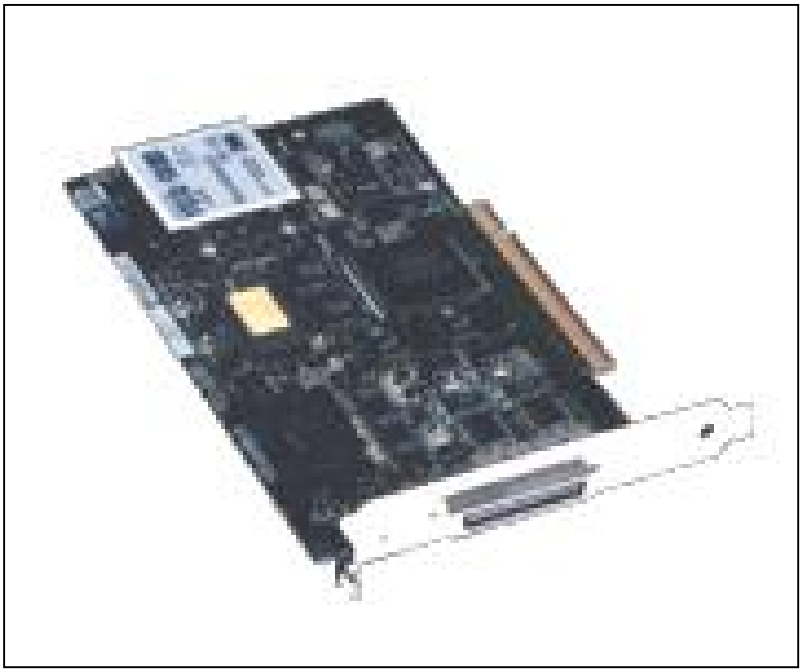
（図２．６）チャージアンプの概観

チャンネル数			１～４
測定範囲		pC	±10~999,000
センサ感度[TS]		pC/M.U.	0.01~9,990
スケール[SC]		M.U./V	0.001~9,990,000
精度	±99.9pCFS まで	%	≦±3
	±100pCFS 以上	%	≦±1
出力	フルスケール出力電圧	V	±10
	最大出力電流	mA	±5
	出力インピーダンス	Ω	10
入力電圧（最大許容）		V	±125

（表２．２）チャージアンプの仕様

２．１．５ A／Dボード

チャージアンプを通し、電圧に変換された力センサの出力をコンピュータで読み込むために利用する。本研究では株式会社インターフェイスのA／D変換ボードＰＣＩ－３１５５（図２．７）を使用した。A／Dボードの仕様は（表２．３）に示す。



（図２．７）A／D変換ボード

入力チャンネル数	シングルエンド入力 16ＣＨ
	差動入力　8ＣＨ
入力形式	マルチプレクサ方式
入力レンジ	バイポーラ：±1V,±2.5V,±5V, ±10V
入力インピーダンス	10MΩ
入力保護	POWER ON 時　±35Ｖ
	POWER OFF 時　±20Ｖ
分解能	16bit
変換時間	10μsec(チャンネル固定時)
	10μsec(チャンネル切り替え高速時)
	20μsec(チャンネル切り替え通常時)
相対精度	±2LSBmax(at25℃)
誤差	±0.10%max(0～50℃): ±10Ｖ
	±0.15%max(0～50℃): ±1Ｖ, ±2.5Ｖ, ±5Ｖ
絶縁方式	非絶縁

（表２．３）Ａ／Ｄボードの仕様

２．１．６　モーターコントロールボード（ＰＣＰＧ－４６）

外部ハンドルパルス割り込み機能（第５章参照）を利用するときに、操作コンピュータ側に取り付ける。制御プログラムによりパルスを発生させマシニングセンタのハンドル機能に割り込む。

以下、概観（図２．８）と仕様（表２．４）である。

（図２．８）モータコントロールボード（PCPG－46）

外形寸法	106.68×174.63mm(コネクタ部、パネル部含まず)
入力電源	DC+5V±0.25V
消費電流	200mA~1000mA
制御機能	最高出力周波数 4.096MPPS 位置決め/連続/機械信号検出の各ドライブ 速度オーバーライド/移動量オーバーライド機能 減速開始ポイント自動検出／残パルス数指定切り替え機能 直線/疑似Ｓ字/完全自動Ｓ字加減速機能 非対称加減速機能 三角駆動回避機能 急速／減速リミット停止機能(アクティブ指定可能) リミット停止機能無効指定可能 急速／減速コマンド停止機能 制御軸数　最大４軸
出力信号	パルス出力等…ラインドライバによる差動出力 ドライバ制御出力…フォトカプラによるオープンコレクタ出力
入力信号	機械系入力及び汎用入力…+12~24V フォトカプラ入力 ドライバステータス入力…+12~24V フォトカプラ入力 フィードバックパルス入力…ラインレシーバによる差動入力

（表２．４）モータコントロールボード（PCPG－46）の仕様

２．２　ソフトウェアシステム

本研究で制御システム開発などで用いたプログラム言語は、全て　Microsoft Visual C++6.0 を用いた。

Visual C++は Windows のアプリケーション開発のための総合開発環境という位置付けにある言語。プログラムのスケルトン作成や多くの開発支援ツールを備えている。また、Visual C++で作成された、別々のプログラム同士を統合するのが容易にできるため、本研究のシステム開発には、これを採用した。

第３章

力センサを用いた加工力モニタリングシステムの開発

３．１　目的

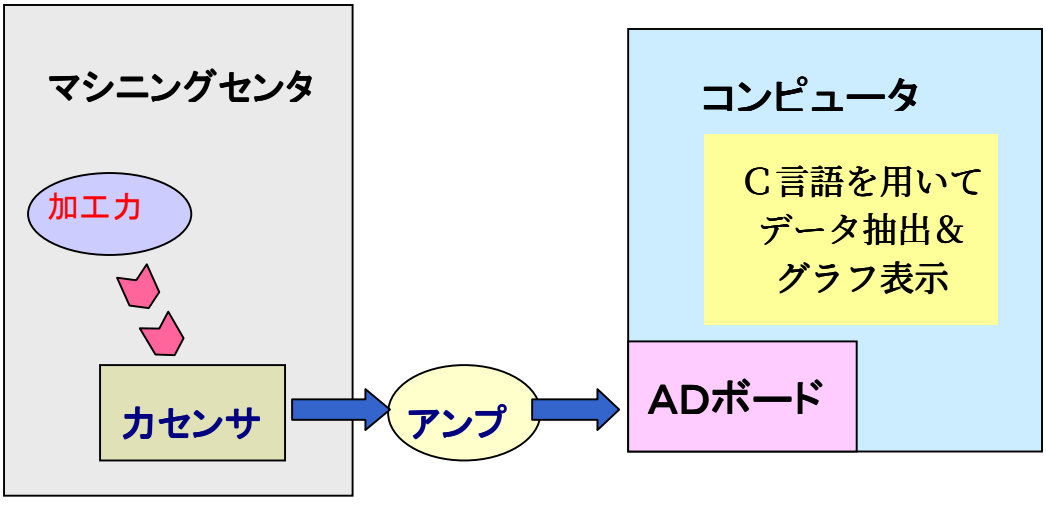
加工中の加工力をリアルタイムで知ることは、加工物の状態を知る上において大変重要な情報である。例えば、工具の磨耗による破損や機械そのものの異常など、加工異常が起これば、それに反映されるからである。また、もっと細かく加工力を調べていくならば、正常な状態でどのような力がかかっているかを知ることができ、それを解析することにより、精密加工に応用できるようになる。

今回、力センサを利用し、３成分の力（Ｆｘ，Ｆｙ，Ｆｚ）の力をリアルタイムで測定、抽出し、制御コンピュータに表示、また一目でどれほどの力がかかっているか分かるよう、グラフ化した。

なお、プログラム作成には、ＶｉｓｕａｌＣ＋＋を利用した。本研究において、すべてのプログラムはこの言語を利用しているため、今後の研究においても容易にプログラム結合ができるようになる。

３．２　システム構成

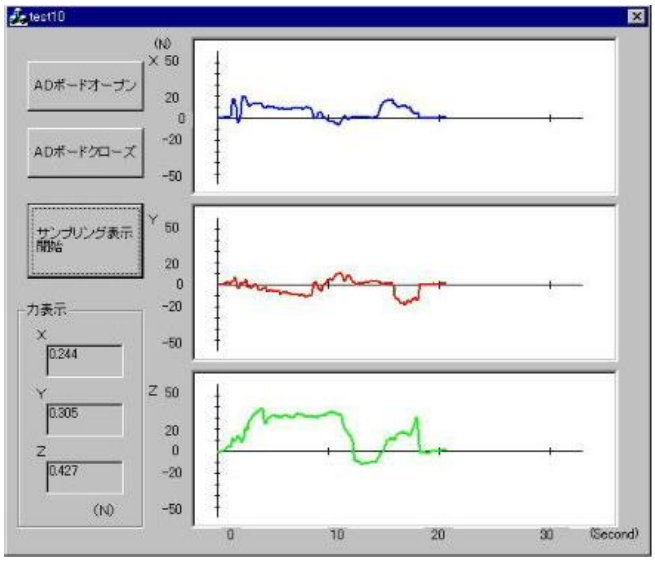
加工力が工作物にかかると、力センサが感知し、微小な電圧を出す。力センサから出る微小な電圧の変化をを増幅器（アンプ）により増幅し、制御コンピュータに伝える。しかしコンピュータ自体は電圧の変化は受け取れないため、ＡＤボードにより電気信号に変換する。これにより力センサからの情報は制御コンピュータに伝わるわけである。この情報を基に制御コンピュータ側ではリアルタイムにデータを抽出し、電圧信号を力データに変換、同時にグラフ化する。（図３．１）に構成図を示す。



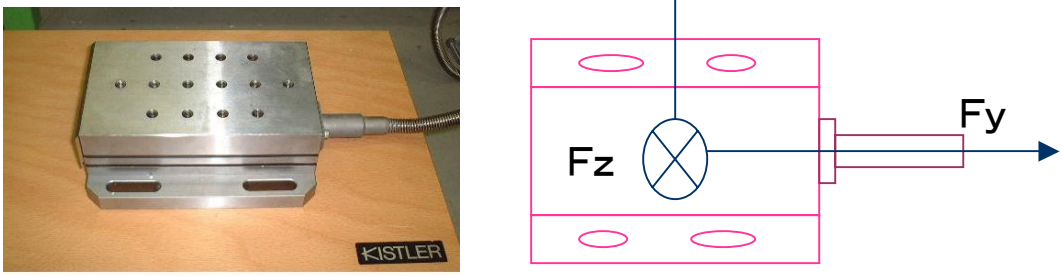
（図３．１）データ抽出プログラムの構成図

３．３　操作画面

下図（図３．２）が今回作成した操作画面である。３成分（Ｆｘ，Ｆｙ，Ｆｚ）の力を抽出し、単位をＮ（ニュートン）に変換、リアルタイムにグラフまたは数字で表示している。これにより一目でどのぐらいの力がどの方向にかかっているかを知ることができる。



（図３．２）力センサデータの表示画面



（図３．３）力センサ概観と分力図

３．４　機能説明

力センサからのデータをリアルタイムに抽出するためには、ある一定の短い時間間隔でデータを連続してとらなければならない。そのため、データを抽出するプログラム（Ｃ＋＋で言えば関数）を一定の時間おきに起動させる。それをする機能としてＶｉｓｕａｌＣ＋＋に標準でついている「Ｏｎｔｉｍｅｒ」関数を利用する。これは指定したある一定の時間ごとに指定したプログラムを動かす機能である。これを利用することにより、リアルタイムに新しいデータを連続してとることができるようになった。この機能は次章以降でもたびたび利用する。また、抽出するプログラムで時間間隔を細かくすることによって、より精密なデータを取ることもであろう。内部プログラムは付録参照。

第４章

オープンＣＮＣマシニングセンタの遠隔操作

４．１　工作機械の遠隔操作のメリット

工作機械を扱うといえは、今までは汚い、危険というイメージがあったのではないだろうか。また昨今の機械はより複雑化、高性能化し、初心者がこの複雑な動作を覚えるにはかなりの時間を要する。また、何かトラブルが起きた場合、人間が動いてトラブルを解決しなければならない。このような現実の中で工作機械の遠隔操作は多くのメリットがある。人間を危険な現場からできるだけ遠ざけ、また世界のどこの場所にいようとも、使い方を教えることや、トラブルシューティングを行なうことができる。とにかく人間自体が動かなくてもあたかもその現場にいるかのようにできるわけである。

また他のメリットとして、ネットワークを介して工場などの生産拠点と設計を行なう拠点、研究所などを結ぶことにより、新しい製品をより迅速に作り出すことができるというわけである。

人や物の移動を伴うことは不利であり、技術開発において開発時間、開発コストの低減はとても重要である。よってこのような情報化された生産システムが今後求められてくるであろう。

これらのことから考えると工作機械の遠隔操作システムは十分なメリットがあると考える。

本研究ではこのシステムを知能化加工システムの一部としてとりあげる。

４．２　遠隔操作の構成

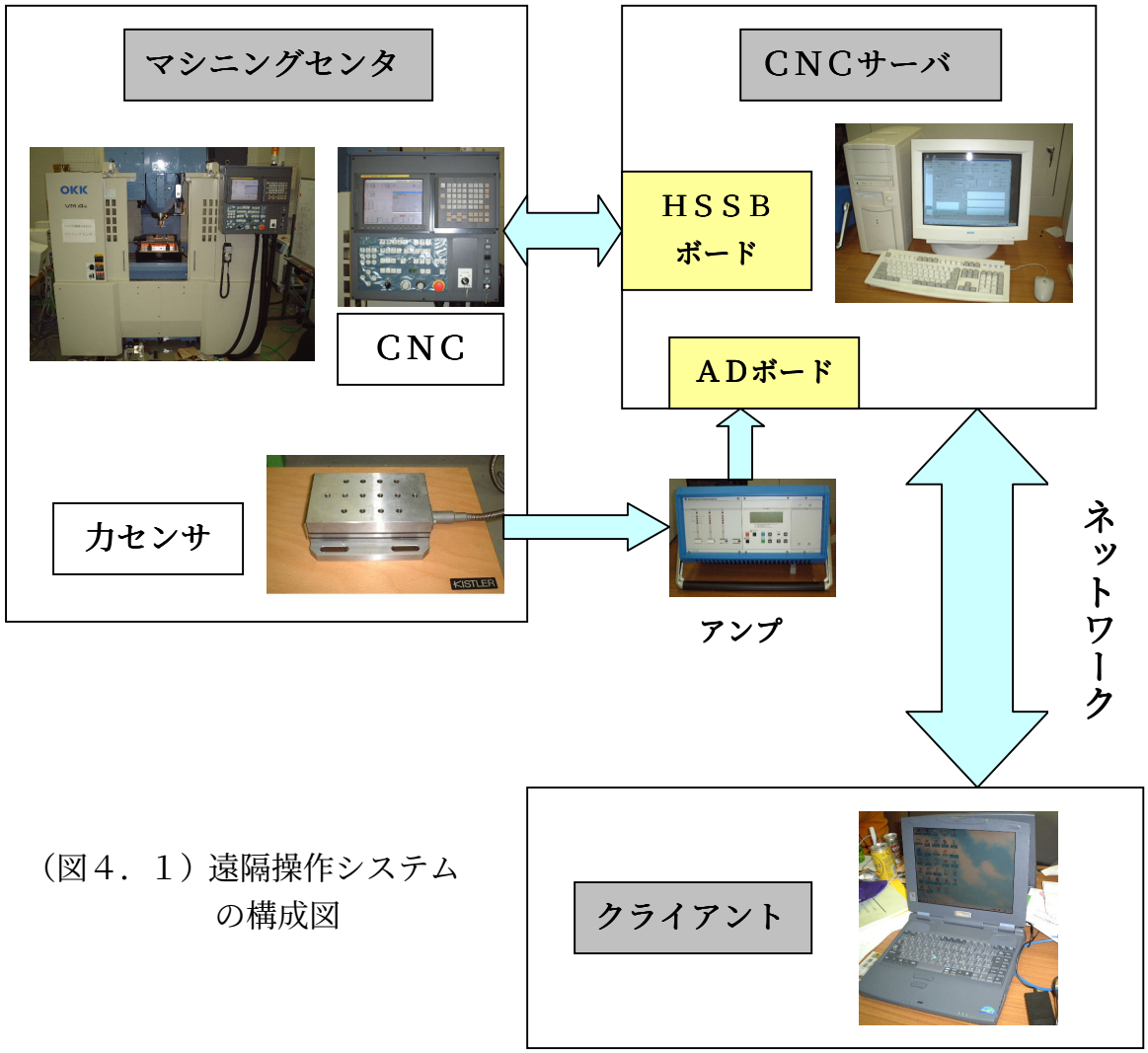
まず、オープンＣＮＣマシニングセンタをＨＳＳＢ回線(High Speed Serial Bus)を介してＣＮＣサーバーコンピュータ（以下サーバ）より制御ができるプログラムを作成する。また、前章で扱った力センサからのデータ抽出プログラムも結合させる。ＣＮＣからの位置情報、主軸回転数、送り速度などと、力センサからの加工力データをサーバー側でリアルタイムで表示。また起動、停止命令などもサーバから瞬時にＣＮＣに伝えるプログラムを作成する。

サーバはインターネットと接続されており、遠隔地から操作するクライアントとなるコンピュータ（以下クライアント）と常にデータをやりとりすることにより、遠隔操作を実現させる。

遠隔操作を行なうには２つのプログラムが必要である。つまりサーバー側とクライアント側である。

本研究では特に著者はサーバのシステム構築を行なったため、そちらを中心に述べる。

（図４．１）では全体の構成図を示す。

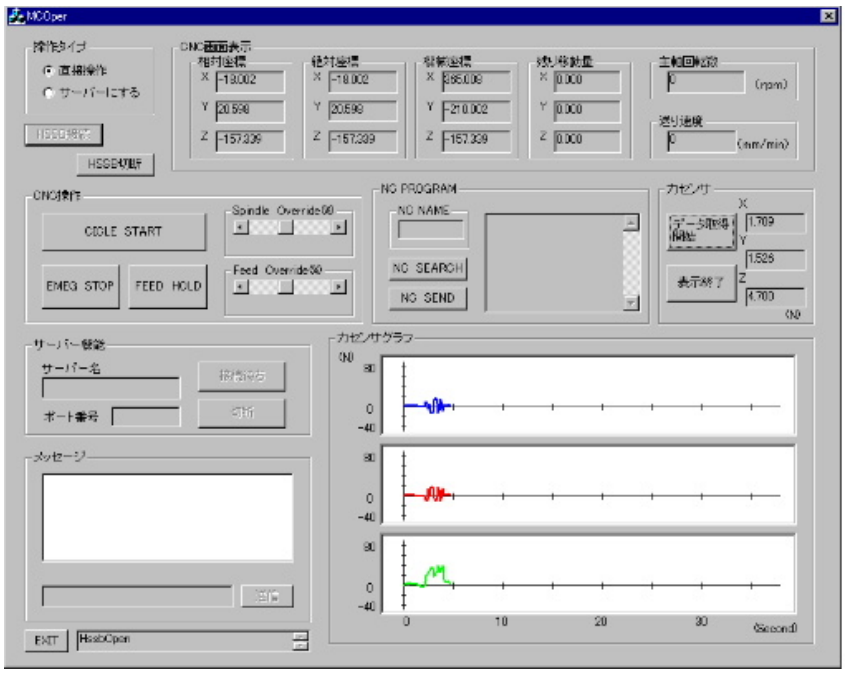


（図４．１）遠隔操作システム
の構成図

４．３　遠隔操作画面の構成

以下の図（４．２）は、ＣＮＣサーバ側の操作画面である。サーバ側からも直接ＣＮＣの制御が行なえる。クライアントとなるコンピュータと接続すると、サーバ側からの操作は、接続を切ること以外はできなくなる。

サーバ側からＨＳＳＢ接続をし、接続待ちボタンを押すことにより、接続待機状態になる。



（図４．２）サーバ側の操作画面

４．４　機能説明

遠隔操作をするにあたって、サーバ側に求められる機能は、

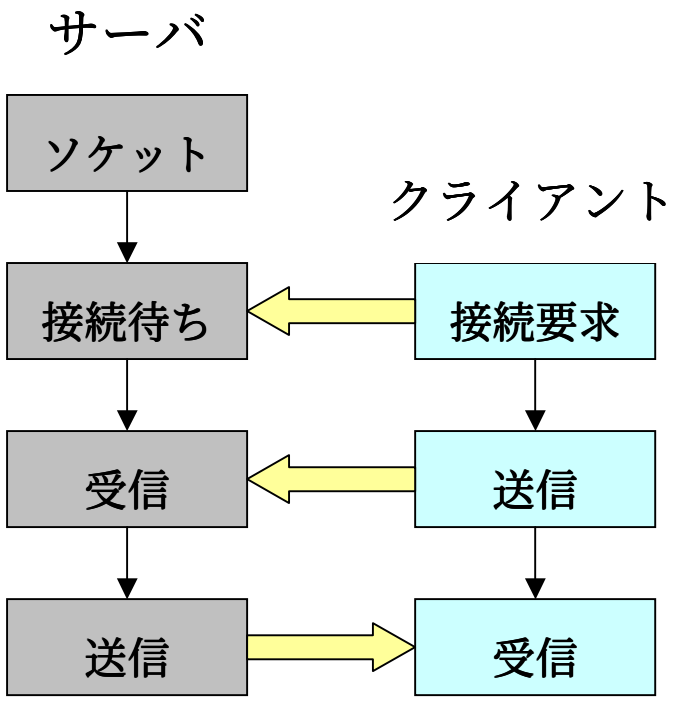
（１）ＨＳＳＢ接続により、ＣＮＣとサーバコンピュータと接続。
（２）ＣＮＣから座標位置情報や回転数などの情報をリアルタイムで取得し、サーバー側からの命令も瞬時に受け入れる状態にする。また、力センサからの情報もリアルタイムに受け入れる。つまりサーバによって全ての情報が一括管理できる状態におく。
（３）ネットワークにつなげ、外部コンピュータ（クライアント）からの接続ができる状態をつくる。
（４）クライアントと接続すると、リアルタイムにデータをやり取りし、外部からの命令によってＣＮＣを通して機械を制御する。

これらの機能を構築することにより、遠隔操作が可能になる。
プログラムの内容は付録参照。

４．４．１　データの送受信

サーバとＣＮＣ、またクライアントとのデータの送受信はリアルタイム性が要求される。よって前章で利用したＣ＋＋の「Ｏｎｔｉｍｅｒ」関数を利用する。サーバはＣＮＣと力センサからの最新の情報を設定した時間毎に受け取り、またクライアント側に送信する。

また、サーバとクライアントはネットワークでつながっており、専用線でつながっているわけではない。したがってネットワークのルールに応じた通信システムが必要である。今回はデータのやり取りに一般的によく利用されている「ソケット通信」を利用した。ソケット通信の構成は以下の（図４．３）に示す



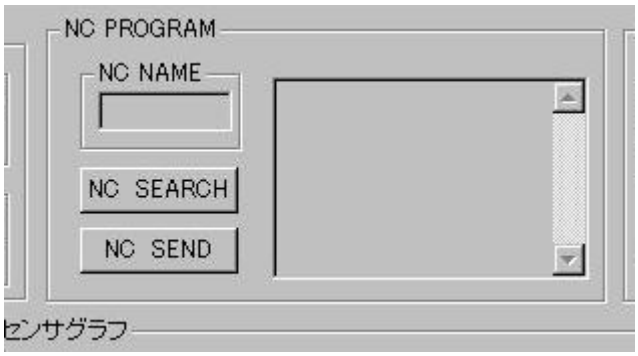
（図４．３）ソケット通信の構成図

４．４．２　ＮＣプログラム転送機能

ＮＣプログラム転送機能とは、クライアントからサーバへＮＣプログラムを送る機能である。離れた場所にあるクライアントコンピュータからでも、ＮＣプログラムをサーバを介してＣＮＣに直接送ることができ、遠隔操作でそのプログラムをすぐに動かすことが可能である。つまり、設計から加工開始までのロスを抑えるための利用が有効的な機能である。

ＮＣ　ＳＥＡＲＣＨ　ボタンでＮＣプログラムを検索、ＮＣ　ＳＥＮＤ　ボタンで送ることができる。

以下の図は操作画面である。サーバの操作画面の一部になる。



（図４．４）ＮＣプログラム操作画面

４．５　考察と問題点

今回作成した遠隔操作プログラムでは、ＣＮＣとサーバとのデータの送受信は問題なく行なえたが、サーバとクライアントとのやりとり、特にサーバからクライアントにデータを送る場合に、時間の遅れが生じた。この間は、ネットワークを利用したため、回線の状態により遅れが生じたと思われる。遠隔操作のリアルタイム性を欠くことは問題である。この点は今後の課題とする。

第５章

外部ハンドルパルス割り込み機能

５．１　精密加工の重要性

加工技術は、（１）精度のよい加工、（２）効率のよい加工、（３）コストの低い加工をめざした発展してきた。さらに最近では、コンピュータおよびセンサ技術の発達に伴い、（４）加工の無人化、（５）加工の適応制御も盛んに行なわれてきている。

加工で生じる物理現象を考えれば、これは工作物と工具の相互作用である。この２つのものが接触し、その結果として必ず「力」が生じ、その力の発生の仕方によって工作物の加工精度が決まる。従来の剛性によって精度を上げようとするやり方は、工具、工作物、工作機械など、力の伝達経路に含まれる全ての構成要素の変形をできるだけ小さくしようとするものであった。そしてそれは、発生する力による変形は避け得ぬものとして受動的に受け止め、その変形ができるだけ小さくなるように構造の剛性を上げるやり方であった。しかしながら、精度に影響を与えるのは力だけでなく、熱（切削熱、環境温度の変化）によっても大きく変化することが分かっている。またびびり振動などの因子がある。これにより、所望の寸法の部品を正確に効率よく作り出すことができない。

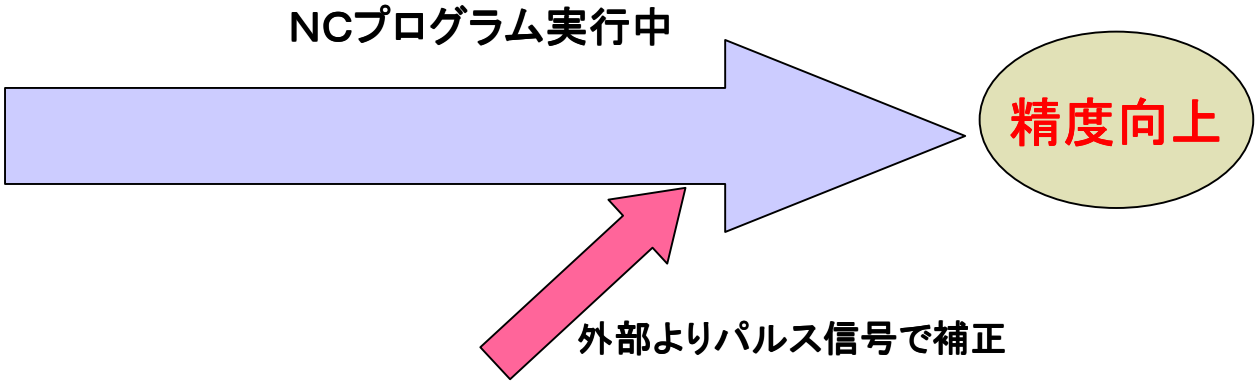
このような状況にあっては、単に剛性をあげることによる高精度では種々も問題を解決することは難しい。より高精度な加工を実現するためには、少なくともセンシング機能を加工システムにもたせる必要がある。種々のセンサをシステムにとりつけ、その情報をもとに物理現象による加工精度のずれを常に修正しながら加工することが必要である。そのためには、それを行なうためのシステムの構築が不可欠である。

５．２　外部ハンドルパルス割り込み機能

外部ハンドルパルス割り込み機能とは、ＮＣプログラム実行中に外部制御用コンピュータからパルス信号を送ることにより、１μｍ（1/1000mm)単位で補正プログラムを割り込ますことのできる最新機能である。

ＣＮＣに設置されている手動ハンドルと同じ機能を、パルス信号を外部より送ることにより実現する。また固定されたＮＣプログラム稼動中であっても利用可能で微小な補正を行なえるため、精密加工を行なう上での利用価値は高い。

しかし、この機能は最新機能であるため、有効性はまだ証明されていない。したがって、本研究では、この機能を利用するための制御プログラムの構築と、有効性を実証すための実験を行なった。



（図５．1）外部ハンドルパルス割り込み機能の概念図

５．３　制御プログラム

割り込みパルスを発生させ、それを機能に応じて制御するためにプログラムを作成した。制御のポイントとして、

１、制御軸（Ｘ，Ｙ，Ｚ（今回、Ｚ軸は使用しない））

２、＋方向と－方向

３、総発生パルス数

４、パルス発生速度（１秒間に何パルス発生させるか）

という４点が状況によって設定しなおされる部分である。

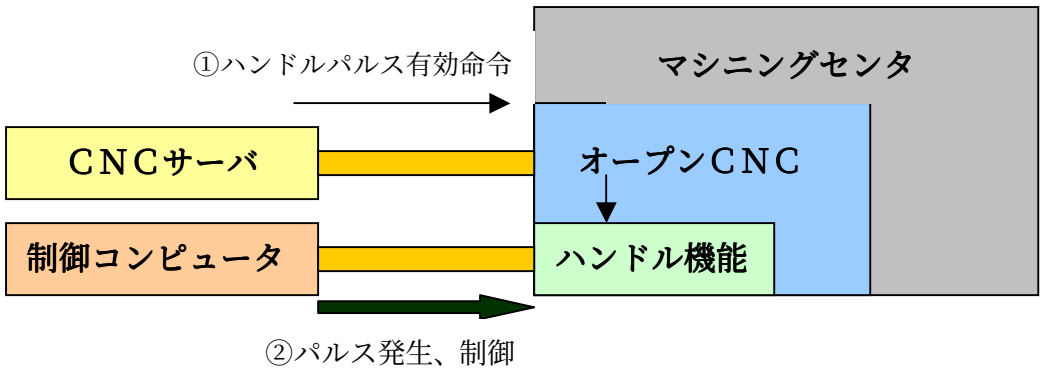
プログラムを状況に応じて簡単に利用するためには、これらを最初に入力する画面を作ることが必要である。したがって上の初期設定を入力する画面を作成し、それを基にプログラムが動く様に作成した。

（図５．3）はＣＮＣサーバ側のシステムの画面である。マシニングセンタの数値制御はすべてオープンＣＮＣを通してＣＮＣサーバによって制御されるので、自動割込みが有効か無効かを決めるスイッチとなるプログラムはここに作成する。倍率と制御軸も設定することができる。

（図５．4）は制御コンピュータ側に作成したシステム画面である。総パルス数と発生速度、＋、－方向の制御をすることができる。

本来ＣＮＣサーバ側で制御するのが好ましいが、パルス発生させるボードがＣＮＣサーバ側にはないため、パルス発生ボードがある、別のコンピュータを使用した。よって、やもなくプログラムを２つに分けることにした。

なお、（図５．2）に全体の構成図を示す。



（図５．2）外部ハンドルパルス割り込み機能　構成図

（図５．3）ＣＮＣサーバ側のシステム画面

（図５．4）制御コンピュータ側のシステム画面

５．４　実験

５．４．１　実験目的及び方法

外部ハンドル割り込み機能の有効性を確かめるための実験を行なった。
実験方法として、

１、 主軸が１秒間に３mm(3mm/sec)Ｘ＋方向に３０秒間動くＮＣプログラムを作成。
２、 １のプログラム起動中にＹ方向に１秒間に１mm(1mm/sec)動くようなパルスを同じく３０秒間割り込ませる。
３、 パルス割り込みなしの、１のＮＣプログラムのみで動いた場合と比較。

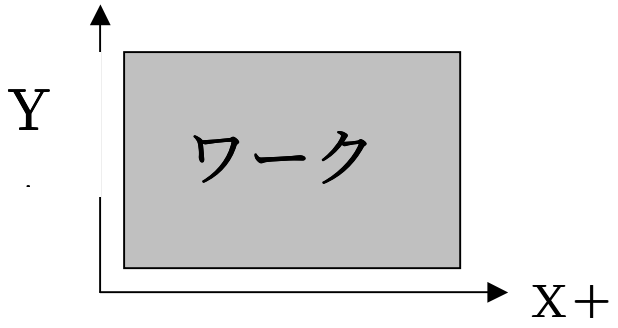
このような順序で実験を行なった。軸の方向は（図５．5）に示す。

５．４．２　割り込みパルスの設定値

まず、１軸に対して４パルス発生で１μm動くということは最初から決められている。今回の実験では、Ｙ＋軸のみ 1mm/sec で３０秒間動かすわけだから、設定値は

Ｙ＋軸に対して、
パルス発生速度　＝　４０００（パルス／秒）
総パルス数　　＝　１２００００（パルス）

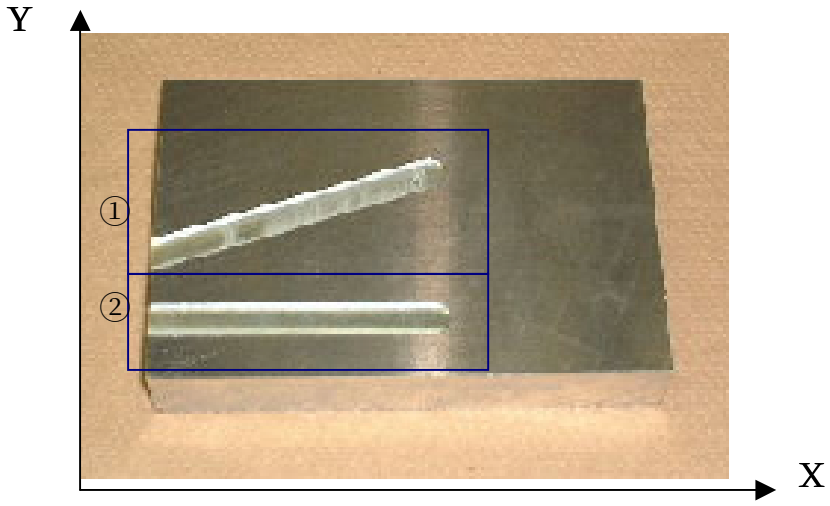
である。



（図５．5）軸の方向

５．４．３　実験結果

結果は下図（図５．6）のようになった。両方とも同一のＮＣプログラムで加工し、①はＹ＋方向にパルスを割り込んだもの、②は割り込みなしである。



（図５．6）

上の加工はスタート地点を基準に水平線を取ると、Ｘ軸に対して３０ｍｍＹ＋方向にずれていた。下の割り込みなしの加工と比較すると一目瞭然だろう。

つまりこの実験ではハンドルパルス割り込みが有効であることが分かった。

５．５　考察

今回の実験ではハンドルパルス割り込み機能は実際に有効であることが分かったが、１軸に対してのみである。今後、精密加工に応用するためには、Ｘ，Ｙ，Ｚの３軸同時に有効でなければならない。しかし今回この機能をテスト的に扱ったが、問題点もあった。ＮＣプログラムで工具が動く方向と同じ方向に割り込みができないというエラーが発見されたのである。つまりＸ＋方向に工具が動いていたとして、同じＸ＋方向にパルスを割り込みさせても正常に動かないのである。これは、ハンドルのシステム自体（ＣＮＣ）の根本的なエラーが原因と考えられる。このため２軸以上同時のパルス割り込み実験はすることができなかった。今後この異常が改善され、完全にこの機能が使えるようになれば、力センサなどの各種センサと組み合わせ、加工誤差を常に修正しながら加工するシステムを構築できるであろう。

第６章

結論と展望

今回の研究では３つの点でシステム開発、実験を行なった。加工力の測定とグラフ化を目指した、（１）加工力モニタリングシステムの開発。ネットワークにつながった遠隔地からでも操作が可能となる（２）遠隔操作システムの開発。そして高精度加工を実現する為に用いられると考えられる、（３）外部ハンドルパルス割り込み機能のシステム開発と有効性を調べるための実験である。

個々のシステムにおいて、基本的な機能は構築できたが、まだまだ問題もたくさんあるため実用性には程遠い。まず、遠隔操作におけるデータ送受信のリアルタイム性の問題である。これは、回線の状態によりデータの送信量が不安定になる為と考えられるが、最適な送信量と回線をうまく利用することにより、改善が可能であるのではないだろうか。次に、ハンドルパルス割り込み機能が正常に動作しないという問題である。これはシステムの根本的欠陥と考えられるが、これを完全に機能させなければ高精度加工のシステム作りにもっていくことはできない。また、それぞれのプログラムを統合し、総合的なシステムを作る必要もある。そうすることにより、実用的な知能化加工システムが構築できるのではないだろうか。

今後研究を進めるにあたり、これら諸問題の解決と、力センサからの情報を利用し、加工の知能化を行なう為に自動的に加工誤差を修正する機能を作成することがこれからの課題となる。

このようにまだまだ課題は山積みされているが、本研究では高精度加工を実現する為の、基礎的な機能は構築と、方向性が示せたのではないだろうか。

参考文献

[1] 林晴比古 “新ＶｉｓｕａｌＣ＋＋６．０入門 ビギナー編” ソフトバンク株式会社、１９９９

[2] 林晴比古 “新Ｃ＋＋入門” ソフトバンク株式会社、１９９８

[3] 立花厚子、山内美恵子、木村総司 [共著]“ＶｉｓｕａｌＣ＋＋で学ぶプログラミング” 株式会社培風館、１９９９

[4] 清水康晶 “作ってわかるＶｉｓｕａｌＣ＋＋６，０” 株式会社秀和システム、２０００

[5] 田中ひろゆき “ＶｉｓｕａｌＣ＋＋６，０の応用５０例” ソフトバンク株式会社、２０００

[6] 山岸正謙 “図解ＮＣ工作機械の入門” 東京電気大学出版局、１９８７

[7] 長尾高明、畑村洋太郎、光石 衛、中尾政之 “知能化生産システム”株式会社朝倉書店、２０００

[8] 李 軍旗 “オープンアーキテクチャＣＮＣを用いたセンサ情報に基づく知能化加工システムの研究” 平成１０年度 学位論文

[9] 王 衛珍 “薄物部品の知能化加工における物理モデルの研究” 平成９年度 修士論文

[10] 東豊一郎 “加工の臨場感通信システムにおける操作システムの研究”平成４年度 卒業論文

謝辞

本研究を進めるにあたり、多くの方々に御指導、ご協力をいただきました。

指導教員である　長尾高明教授、李軍旗助教授には、研究全般で御助言、御指導をいただきました。ありがとうございました。

また、同じ研究室の同士として、更谷さん、増吉さん、福見さん、田中さん、現田さん、大谷さんには、いろいろな面でお世話になりました。

以上の方々に深く感謝し、厚くお礼申し上げます。

付録

本研究で作成したプログラムを付録といたします。ただしプログラムの主要部分のみ掲載いたします。

Test10Dlg.cpp ---------- 力センサモニタリングプログラム
MCOperDlg.cpp ---------- 遠隔操作サーバプログラム
PcpgDlg.cpp ---------- 外部ハンドルパルス割り込み機能プログラム

```
// test10Dlg.cpp : インプリメンテーション ファイル
//

#include "stdafx.h"
#include "test10.h"
#include "test10Dlg.h"

#include"windows.h"
#include"stdio.h"
#include"FbiAd.h"

HANDLE hDeviceHandle;
ADSMPLCHREQ SmplChReq[3];
WORD wSmpData[3];
double x1,y1,y2,y3;
static char                     dummy[256];
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
///
// アプリケーションのバージョン情報で使われている CAboutDlg ダイアログ

class CAboutDlg : public CDialog
{
public:
        CAboutDlg();

// ダイアログ データ
        //{{AFX_DATA(CAboutDlg)
        enum { IDD = IDD_ABOUTBOX };
        //}}AFX_DATA
```

```
	// ClassWizard は仮想関数のオーバーライドを生成します
	//{{AFX_VIRTUAL(CAboutDlg)
	protected:
	virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV のサポート
	//}}AFX_VIRTUAL

// インプリメンテーション
protected:
	//{{AFX_MSG(CAboutDlg)
	//}}AFX_MSG
	DECLARE_MESSAGE_MAP()
};

CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
	//{{AFX_DATA_INIT(CAboutDlg)
	//}}AFX_DATA_INIT
}

void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CAboutDlg)
	//}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
	//{{AFX_MSG_MAP(CAboutDlg)
		// メッセージ ハンドラがありません。
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
///
// CTest10Dlg ダイアログ
```

```
CTest10Dlg::CTest10Dlg(CWnd* pParent /*=NULL*/)
	: CDialog(CTest10Dlg::IDD, pParent)
{
	//{{AFX_DATA_INIT(CTest10Dlg)
	m_tuyox = _T("");
	m_tuyoy = _T("");
	m_tuyoz = _T("");
	//}}AFX_DATA_INIT
	// メモ: LoadIcon は Win32 の DestroyIcon のサブシーケンスを要求しません。
	m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
}

void CTest10Dlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CTest10Dlg)
	DDX_Control(pDX, IDC_PICT3, m_pict3);
	DDX_Control(pDX, IDC_PICT2, m_pict2);
	DDX_Control(pDX, IDC_PICT, m_pict);
	DDX_Text(pDX, IDC_EDIT1, m_tuyox);
	DDX_Text(pDX, IDC_EDIT2, m_tuyoy);
	DDX_Text(pDX, IDC_EDIT3, m_tuyoz);
	//}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CTest10Dlg, CDialog)
	//{{AFX_MSG_MAP(CTest10Dlg)
	ON_WM_SYSCOMMAND()
	ON_WM_PAINT()
	ON_WM_QUERYDRAGICON()
	ON_BN_CLICKED(IDC_BUTTON1, OnButton1)
	ON_BN_CLICKED(IDC_BUTTON2, OnButton2)
	ON_BN_CLICKED(IDC_BUTTON3, OnButton3)
	ON_WM_TIMER()
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()
```

```
////////////////////////////////////////////////////////////////////////////
///
// CTest10Dlg メッセージ ハンドラ

BOOL CTest10Dlg::OnInitDialog()
{
	CDialog::OnInitDialog();

	// "バージョン情報..." メニュー項目をシステム メニューへ追加します。

	// IDM_ABOUTBOX はコマンド メニューの範囲でなければなりません。
	ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
	ASSERT(IDM_ABOUTBOX < 0xF000);

	CMenu* pSysMenu = GetSystemMenu(FALSE);
	if (pSysMenu != NULL)
	{
		CString strAboutMenu;
		strAboutMenu.LoadString(IDS_ABOUTBOX);
		if (!strAboutMenu.IsEmpty())
		{
			pSysMenu->AppendMenu(MF_SEPARATOR);
			pSysMenu->AppendMenu(MF_STRING,		IDM_ABOUTBOX,
strAboutMenu);
		}
	}

	// このダイアログ用のアイコンを設定します。フレームワークはアプリケーションのメイ
ン
	// ウィンドウがダイアログでない時は自動的に設定しません。
	SetIcon(m_hIcon, TRUE);			// 大きいアイコンを設定
	SetIcon(m_hIcon, FALSE);		// 小さいアイコンを設定

	// TODO: 特別な初期化を行う時はこの場所に追加してください。
```

```
	return TRUE;  // TRUE を返すとコントロールに設定したフォーカスは失われません。
}

void CTest10Dlg::OnSysCommand(UINT nID, LPARAM lParam)
{
	if ((nID & 0xFFF0) == IDM_ABOUTBOX)
	{
		CAboutDlg dlgAbout;
		dlgAbout.DoModal();
	}
	else
	{
		CDialog::OnSysCommand(nID, lParam);
	}
}

// もしダイアログボックスに最小化ボタンを追加するならば、アイコンを描画する
// コードを以下に記述する必要があります。MFC アプリケーションは document/view
// モデルを使っているので、この処理はフレームワークにより自動的に処理されます。

void CTest10Dlg::OnPaint()
{
	if (IsIconic())
	{
		CPaintDC dc(this); // 描画用のデバイス コンテキスト

		SendMessage(WM_ICONERASEBKGND, (WPARAM) dc.GetSafeHdc(), 0);

		// クライアントの矩形領域内の中央
		int cxIcon = GetSystemMetrics(SM_CXICON);
		int cyIcon = GetSystemMetrics(SM_CYICON);
		CRect rect;
		GetClientRect(&rect);
		int x = (rect.Width() - cxIcon + 1) / 2;
		int y = (rect.Height() - cyIcon + 1) / 2;
```

```
                // アイコンを描画します。
                dc.DrawIcon(x, y, m_hIcon);
        }
        else
        {
                CDialog::OnPaint();
        }
}

// システムは、ユーザーが最小化ウィンドウをドラッグしている間、
// カーソルを表示するためにここを呼び出します。
HCURSOR CTest10Dlg::OnQueryDragIcon()
{
        return (HCURSOR) m_hIcon;
}

void CTest10Dlg::OnButton1()
{
        // TODO: この位置にコントロール通知ハンドラ用のコードを追加してください


    hDeviceHandle = AdOpen("FBIAD1");
        if (hDeviceHandle == INVALID_HANDLE_VALUE){
                 MessageBox("Device FBIAD1 は使用できません!");
                }
        else{
                MessageBox("ADボードオープンしました");
        }
//初期設定
//X
   //グラフエリアを白で塗りつぶす
        CDC* pDC=m_pict.GetDC();
        CRect myRECT;
        m_pict.GetClientRect(myRECT);
        pDC->FillSolidRect(myRECT, RGB(255,255,255));
```

```
        pDC->MoveTo(20,70);
        pDC->LineTo(350,70);
        pDC->MoveTo(20,10);
        pDC->LineTo(20,130);

        pDC->MoveTo(120,67);
        pDC->LineTo(120,74);
        pDC->MoveTo(220,67);
        pDC->LineTo(220,74);
        pDC->MoveTo(320,67);
        pDC->LineTo(320,74);

        pDC->MoveTo(18,20);
        pDC->LineTo(23,20);
    pDC->MoveTo(18,30);
        pDC->LineTo(23,30);
        pDC->MoveTo(18,40);
        pDC->LineTo(23,40);
    pDC->MoveTo(18,50);
        pDC->LineTo(23,50);
        pDC->MoveTo(18,60);
        pDC->LineTo(23,60);
    pDC->MoveTo(18,80);
        pDC->LineTo(23,80);
    pDC->MoveTo(18,90);
        pDC->LineTo(23,90);
    pDC->MoveTo(18,100);
        pDC->LineTo(23,100);
    pDC->MoveTo(18,110);
        pDC->LineTo(23,110);
    pDC->MoveTo(18,120);
        pDC->LineTo(23,120);
//Y
    CDC* pDC2=m_pict2.GetDC();
        CRect myRECT2;
        m_pict2.GetClientRect(myRECT2);
```

```
        pDC2->FillSolidRect(myRECT, RGB(255,255,255));

    pDC->MoveTo(20,220);
        pDC->LineTo(350,220);
        pDC->MoveTo(20,160);
        pDC->LineTo(20,280);

    pDC->MoveTo(120,217);
        pDC->LineTo(120,224);
        pDC->MoveTo(220,217);
        pDC->LineTo(220,224);
        pDC->MoveTo(320,217);
        pDC->LineTo(320,224);

    pDC->MoveTo(18,170);
        pDC->LineTo(23,170);
    pDC->MoveTo(18,180);
        pDC->LineTo(23,180);
        pDC->MoveTo(18,190);
        pDC->LineTo(23,190);
    pDC->MoveTo(18,200);
        pDC->LineTo(23,200);
        pDC->MoveTo(18,210);
        pDC->LineTo(23,210);
    pDC->MoveTo(18,230);
        pDC->LineTo(23,230);
    pDC->MoveTo(18,240);
        pDC->LineTo(23,240);
    pDC->MoveTo(18,250);
        pDC->LineTo(23,250);
    pDC->MoveTo(18,260);
        pDC->LineTo(23,260);
    pDC->MoveTo(18,270);
        pDC->LineTo(23,270);
```

```
//Z
    CDC* pDC3=m_pict3.GetDC();
        CRect myRECT3;
        m_pict3.GetClientRect(myRECT2);
        pDC3->FillSolidRect(myRECT, RGB(255,255,255));

    pDC->MoveTo(20,370);
        pDC->LineTo(350,370);
        pDC->MoveTo(20,310);
        pDC->LineTo(20,430);

    pDC->MoveTo(120,367);
        pDC->LineTo(120,374);
        pDC->MoveTo(220,367);
        pDC->LineTo(220,374);
        pDC->MoveTo(320,367);
        pDC->LineTo(320,374);

    pDC->MoveTo(18,320);
        pDC->LineTo(23,320);
    pDC->MoveTo(18,330);
        pDC->LineTo(23,330);
        pDC->MoveTo(18,340);
        pDC->LineTo(23,340);
    pDC->MoveTo(18,350);
        pDC->LineTo(23,350);
        pDC->MoveTo(18,360);
        pDC->LineTo(23,360);
    pDC->MoveTo(18,380);
        pDC->LineTo(23,380);
    pDC->MoveTo(18,390);
        pDC->LineTo(23,390);
    pDC->MoveTo(18,400);
        pDC->LineTo(23,400);
    pDC->MoveTo(18,410);
        pDC->LineTo(23,410);
```

```
	pDC->MoveTo(18,420);
		pDC->LineTo(23,420);
}

void CTest10Dlg::OnButton2()
{
		// TODO: この位置にコントロール通知ハンドラ用のコードを追加してください

		SetTimer(1,100,NULL);
		tu=20;
	x1=20;
		y1=70; y2=220; y3=370;
}

void CTest10Dlg::OnButton3()
{
		// TODO: この位置にコントロール通知ハンドラ用のコ加ードを追してください
	KillTimer(1);

	CDC* pDC=m_pict.GetDC();
	CRect myRECT;
			m_pict.GetClientRect(myRECT);
			pDC->FillSolidRect(myRECT, RGB(255,255,255));

	CDC* pDC2=m_pict2.GetDC();
	CRect myRECT2;
			m_pict2.GetClientRect(myRECT2);
			pDC2->FillSolidRect(myRECT2, RGB(255,255,255));

	CDC* pDC3=m_pict3.GetDC();
	CRect myRECT3;
			m_pict3.GetClientRect(myRECT3);
			pDC3->FillSolidRect(myRECT3, RGB(255,255,255));

		AdClose(hDeviceHandle);
	AfxMessageBox(″ADボードクローズ¥n プログラムを終了します。″);
```

```
	CDialog::OnCancel();
}

void CTest10Dlg::OnTimer(UINT nIDEvent)
{
	// TODO: この位置にメッセージ ハンドラ用のコードを追加するかまたはデフォルトの
処理を呼び出してください
    double data_x,data_y,data_z;
                data_x=data_y=data_z=0;
    tu=tu+1;

        SmplChReq[0].ulChNo = 1;
        SmplChReq[0].ulRange = AD_10V;
    SmplChReq[1].ulChNo = 2;
        SmplChReq[1].ulRange = AD_10V;
    SmplChReq[2].ulChNo = 3;
        SmplChReq[2].ulRange = AD_10V;

    AdInputAD(hDeviceHandle, 3, AD_INPUT_SINGLE, &SmplChReq[0], &wSmpData);

                data_x = -2000.0/65536*(wSmpData[0]-32768);
                data_y = -2000.0/65536*(wSmpData[1]-32768);
                data_z = -2000.0/65536*(wSmpData[2]-32768);

            sprintf(dummy,"%.3f",2000.0/65536*(wSmpData[0]-32768));
                m_tuyox=dummy;
                sprintf(dummy,"%.3f",2000.0/65536*(wSmpData[1]-32768));
                m_tuyoy=dummy;
          sprintf(dummy,"%.3f",2000.0/65536*(wSmpData[2]-32768));
                m_tuyoz=dummy;
      UpdateData(false);
      double x, yx,yy,yz;

          x=tu;
          yx= data_x + 70;
```

```
            yy= data_y + 220;
            yz= data_z + 370;
//Xのデータを表示
            CDC* pDC=m_pict.GetDC();
             //pDC->SetPixel(x,y, RGB(0,0,255));
            CPen BluePen,RedPen,GreenPen,BrackPen;
            BluePen.CreatePen(PS_SOLID,2,RGB(0,0,255));
        //RedPen.CreatePen(PS_SOLID,2,RGB(255,0,0));
        //GreenPen.CreatePen(PS_SOLID,2,RGB(0,255,0));
        //BrackPen.CreatePen(PS_SOLID,1,RGB(0,0,0));

            pDC->SelectObject(&BluePen);
            pDC->MoveTo(x1,y1);
            pDC->LineTo(x,yx);

            //x1=x;
            y1=yx;

//Yのデータを表示
        CDC* pDC2=m_pict2.GetDC();

        RedPen.CreatePen(PS_SOLID,2,RGB(255,0,0));

        pDC->SelectObject(&RedPen);
            pDC->MoveTo(x1,y2);
            pDC->LineTo(x,yy);

            //x1=x;
            y2=yy;

//Zのデータを表示
        CDC* pDC3=m_pict3.GetDC();
        GreenPen.CreatePen(PS_SOLID,2,RGB(0,255,0));

            pDC->SelectObject(&GreenPen);
            pDC->MoveTo(x1,y3);
```

```
            pDC->LineTo(x,yz);

            x1=x;
            y3=yz;


            if (tu > 320){
//グラフエリアを白で塗りつぶす
          //X
          CDC* pDC=m_pict.GetDC();
          CRect myRECT;
          m_pict.GetClientRect(myRECT);
          pDC->FillSolidRect(myRECT, RGB(255,255,255));
     //Y
          CDC* pDC2=m_pict2.GetDC();
          CRect myRECT2;
          m_pict2.GetClientRect(myRECT2);
          pDC2->FillSolidRect(myRECT2, RGB(255,255,255));
     //Z
          CDC* pDC3=m_pict3.GetDC();
          CRect myRECT3;
          m_pict3.GetClientRect(myRECT3);
          pDC3->FillSolidRect(myRECT3, RGB(255,255,255));

//x軸、y軸の初期設定
     pDC->SelectObject(&BrackPen);
   //X軸
          pDC->MoveTo(20,70);
          pDC->LineTo(350,70);
     pDC->MoveTo(20,10);
          pDC->LineTo(20,130);

     pDC->MoveTo(120,67);
          pDC->LineTo(120,74);
          pDC->MoveTo(220,67);
          pDC->LineTo(220,74);
```

```
        pDC->MoveTo(320,67);
        pDC->LineTo(320,74);

        pDC->MoveTo(18,20);
        pDC->LineTo(23,20);
    pDC->MoveTo(18,30);
        pDC->LineTo(23,30);
        pDC->MoveTo(18,40);
        pDC->LineTo(23,40);
    pDC->MoveTo(18,50);
        pDC->LineTo(23,50);
        pDC->MoveTo(18,60);
        pDC->LineTo(23,60);
    pDC->MoveTo(18,80);
        pDC->LineTo(23,80);
    pDC->MoveTo(18,90);
        pDC->LineTo(23,90);
    pDC->MoveTo(18,100);
        pDC->LineTo(23,100);
    pDC->MoveTo(18,110);
        pDC->LineTo(23,110);
    pDC->MoveTo(18,120);
        pDC->LineTo(23,120);
  //Y軸
    pDC->MoveTo(20,220);
        pDC->LineTo(350,220);
        pDC->MoveTo(20,160);
        pDC->LineTo(20,280);

    pDC->MoveTo(120,217);
        pDC->LineTo(120,224);
        pDC->MoveTo(220,217);
        pDC->LineTo(220,224);
        pDC->MoveTo(320,217);
        pDC->LineTo(320,224);
```

```
    pDC->MoveTo(18,170);
        pDC->LineTo(23,170);
    pDC->MoveTo(18,180);
        pDC->LineTo(23,180);
        pDC->MoveTo(18,190);
        pDC->LineTo(23,190);
    pDC->MoveTo(18,200);
        pDC->LineTo(23,200);
        pDC->MoveTo(18,210);
        pDC->LineTo(23,210);
    pDC->MoveTo(18,230);
        pDC->LineTo(23,230);
    pDC->MoveTo(18,240);
        pDC->LineTo(23,240);
    pDC->MoveTo(18,250);
        pDC->LineTo(23,250);
    pDC->MoveTo(18,260);
        pDC->LineTo(23,260);
    pDC->MoveTo(18,270);
        pDC->LineTo(23,270);

//Z軸
        pDC->MoveTo(20,370);
        pDC->LineTo(350,370);
        pDC->MoveTo(20,310);
        pDC->LineTo(20,430);

    pDC->MoveTo(120,367);
        pDC->LineTo(120,374);
        pDC->MoveTo(220,367);
        pDC->LineTo(220,374);
        pDC->MoveTo(320,367);
        pDC->LineTo(320,374);

    pDC->MoveTo(18,320);
        pDC->LineTo(23,320);
```

```
        pDC->MoveTo(18,330);
            pDC->LineTo(23,330);
            pDC->MoveTo(18,340);
            pDC->LineTo(23,340);
        pDC->MoveTo(18,350);
            pDC->LineTo(23,350);
            pDC->MoveTo(18,360);
            pDC->LineTo(23,360);
        pDC->MoveTo(18,380);
            pDC->LineTo(23,380);
        pDC->MoveTo(18,390);
            pDC->LineTo(23,390);
        pDC->MoveTo(18,400);
            pDC->LineTo(23,400);
        pDC->MoveTo(18,410);
            pDC->LineTo(23,410);
        pDC->MoveTo(18,420);
            pDC->LineTo(23,420);

            tu=20;
        x1=20;
            y1=70; y2=220; y3=370;
              }



            CDialog::OnTimer(nIDEvent);
}
```

```
// MCOperDlg.cpp : インプリメンテーション ファイル
//

#include "stdafx.h"
#include "MCOper.h"
#include "MCOperDlg.h"
#include "FbiAd.h"
HANDLE hDeviceHandle;
ADSMPLCHREQ SmplChReq[3];
WORD wSmpData[3];
    double adx0,ady1,ady2,ady3;
        int gf_x=20;
        int gf_x_y=50;
        int gf_y_y=140;
        int gf_z_y=230;

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif


#define READTIMER_ID    0

static char                 dummy[256];
static bool StartFlag = false;
static S_DATA    sData,rData;
static int NCProgFlag = 0, overvalue;
static int ProgNum;


/////////////////////////////////////////////////////////////////////////////
///
```

```
// アプリケーションのバージョン情報で使われている CAboutDlg ダイアログ

class CAboutDlg : public CDialog
{
public:
	CAboutDlg();



// ダイアログ データ
	//{{AFX_DATA(CAboutDlg)
	enum { IDD = IDD_ABOUTBOX };
	//}}AFX_DATA

	// ClassWizard は仮想関数のオーバーライドを生成します
	//{{AFX_VIRTUAL(CAboutDlg)
	protected:
	virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV のサポート
	//}}AFX_VIRTUAL

// インプリメンテーション
protected:
	//{{AFX_MSG(CAboutDlg)
	//}}AFX_MSG
	DECLARE_MESSAGE_MAP()
};

CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
	//{{AFX_DATA_INIT(CAboutDlg)
	//}}AFX_DATA_INIT
}

void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
```

```
	//{{AFX_DATA_MAP(CAboutDlg)
	//}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
	//{{AFX_MSG_MAP(CAboutDlg)
		// メッセージ ハンドラがありません。
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
///
// CMCOperDlg ダイアログ

CMCOperDlg::CMCOperDlg(CWnd* pParent /*=NULL*/)
	: CDialog(CMCOperDlg::IDD, pParent)
{
	//{{AFX_DATA_INIT(CMCOperDlg)
	m_rely = _T("");
	m_relz = _T("");
	m_absx = _T("");
	m_absy = _T("");
	m_absz = _T("");
	m_spindle = _T("");
	m_feed = _T("");
	m_message = _T("");
	m_ncprog = _T("");
	NCNAME = _T("");
	m_macx = _T("");
	m_macy = _T("");
	m_macz = _T("");
	m_relx = _T("");
	m_remx = _T("");
	m_remy = _T("");
	m_remz = _T("");
	m_feedover = 100;
```

```
	m_spindleover = 100;
	m_forcey = _T("");
	m_forcez = _T("");
	m_forcex = _T("");
	//}}AFX_DATA_INIT
	// メモ: LoadIcon は Win32 の DestroyIcon のサブシーケンスを要求しません。
	m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
}

void CMCOperDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CMCOperDlg)
	DDX_Control(pDX, IDC_FORCE_OPEN, m_fo);
	DDX_Control(pDX, IDC_FORCE_CUT, m_fc);
	DDX_Control(pDX, IDC_PICTZ, m_pictz);
	DDX_Control(pDX, IDC_PICTY, m_picty);
	DDX_Control(pDX, IDC_PICTX, m_pictx);
	DDX_Control(pDX, IDC_SCROLLBAR_SPINDLE, m_spindlescrol);
	DDX_Control(pDX, IDC_SCROLLBAR_FEED, m_feedscrol);
	DDX_Control(pDX, IDC_NCSEND, m_ncsend);
	DDX_Control(pDX, IDC_HSSBOPEN, m_hssbopen);
	DDX_Control(pDX, IDC_FEEDHOLD, m_feedhold);
	DDX_Control(pDX, IDC_CICLESTART, m_ciclestart);
	DDX_Control(pDX, IDC_EMEGSTOP, m_emegstop);
	DDX_Control(pDX, IDC_HSSBCOLSE, m_hssbcolse);
	DDX_Text(pDX, IDC_RELY, m_rely);
	DDX_Text(pDX, IDC_RELZ, m_relz);
	DDX_Text(pDX, IDC_ABSX, m_absx);
	DDX_Text(pDX, IDC_ABSY, m_absy);
	DDX_Text(pDX, IDC_ABSZ, m_absz);
	DDX_Text(pDX, IDC_EDIT_SPINDLE, m_spindle);
	DDX_Text(pDX, IDC_EDIT_FEED, m_feed);
	DDX_Text(pDX, IDC_EDIT_MESSA, m_message);
	DDX_Text(pDX, IDC_EDIT_NCPROG, m_ncprog);
	DDX_Text(pDX, IDC_NCNAME, NCNAME);
```

```
	DDX_Text(pDX, IDC_MACX, m_macx);
	DDX_Text(pDX, IDC_MACY, m_macy);
	DDX_Text(pDX, IDC_MACZ, m_macz);
	DDX_Text(pDX, IDC_RELX, m_relx);
	DDX_Text(pDX, IDC_REMX, m_remx);
	DDX_Text(pDX, IDC_REMY, m_remy);
	DDX_Text(pDX, IDC_REMZ, m_remz);
	DDX_Scroll(pDX, IDC_SCROLLBAR_FEED, m_feedover);
	DDX_Scroll(pDX, IDC_SCROLLBAR_SPINDLE, m_spindleover);
	DDX_Text(pDX, IDC_FORCE_Y, m_forcey);
	DDX_Text(pDX, IDC_FORCE_Z, m_forcez);
	DDX_Text(pDX, IDC_FORCE_X, m_forcex);
	//}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CMCOperDlg, CDialog)
	//{{AFX_MSG_MAP(CMCOperDlg)
	ON_WM_SYSCOMMAND()
	ON_WM_PAINT()
	ON_WM_QUERYDRAGICON()
	ON_BN_CLICKED(IDC_HSSBOPEN, OnHssbopen)
	ON_BN_CLICKED(IDC_NCSELECT, OnNcselect)
	ON_BN_CLICKED(IDC_NCSEND, OnNcsend)
	ON_BN_CLICKED(IDC_CICLESTART, OnCiclestart)
	ON_BN_CLICKED(IDC_FEEDHOLD, OnFeedhold)
	ON_BN_CLICKED(IDC_EMEGSTOP, OnEmegstop)
	ON_BN_CLICKED(IDC_HSSBCOLSE, OnHssbcolse)
	ON_WM_TIMER()
	ON_WM_HSCROLL()
	ON_BN_CLICKED(IDC_EXIT, OnExit)
	ON_BN_CLICKED(IDC_RADIO_SAVER, OnRadioSaver)
	ON_BN_CLICKED(IDC_RADIO_SOUSA, OnRadioSousa)
	ON_BN_CLICKED(IDC_SERVER_CONNECT, OnServerConnect)
	ON_BN_CLICKED(IDC_FORCE_OPEN, OnForceOpen)
	ON_BN_CLICKED(IDC_FORCE_CUT, OnForceCut)
	ON_BN_CLICKED(IDC_SERVER_CUT, OnServerCut)
```

```
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
///
// CMCOperDlg メッセージ ハンドラ

BOOL CMCOperDlg::OnInitDialog()
{
	CDialog::OnInitDialog();

	// "バージョン情報..." メニュー項目をシステム メニューへ追加します。

	// IDM_ABOUTBOX はコマンド メニューの範囲でなければなりません。
	ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
	ASSERT(IDM_ABOUTBOX < 0xF000);

	CMenu* pSysMenu = GetSystemMenu(FALSE);
	if (pSysMenu != NULL)
	{
		CString strAboutMenu;
		strAboutMenu.LoadString(IDS_ABOUTBOX);
		if (!strAboutMenu.IsEmpty())
		{
			pSysMenu->AppendMenu(MF_SEPARATOR);
			pSysMenu->AppendMenu(MF_STRING, IDM_ABOUTBOX,
strAboutMenu);
		}
	}

	// このダイアログ用のアイコンを設定します。フレームワークはアプリケーションのメイ
ン
	// ウィンドウがダイアログでない時は自動的に設定しません。
	SetIcon(m_hIcon, TRUE);			// 大きいアイコンを設定
	SetIcon(m_hIcon, FALSE);		// 小さいアイコンを設定
```

```
        // TODO: 特別な初期化を行う時はこの場所に追加してください。


        m_hsfeedover = (CScrollBar* )GetDlgItem(IDC_SCROLLBAR_FEED);
        m_hsfeedover ->SetScrollRange(20,200);
        m_hsfeedover ->SetScrollPos(100);

        m_hsspindleover = (CScrollBar* )GetDlgItem(IDC_SCROLLBAR_SPINDLE);
        m_hsspindleover ->SetScrollRange(20,200);
        m_hsspindleover ->SetScrollPos(100);
        m_fc.EnableWindow(FALSE);
        m_fo.EnableWindow(FALSE);

    // m_hssbcolse.ShowWindow(SW_HIDE);
        return TRUE;  // TRUE を返すとコントロールに設定したフォーカスは失われません。
}

void CMCOperDlg::OnSysCommand(UINT nID, LPARAM lParam)
{
        if ((nID & 0xFFF0) == IDM_ABOUTBOX)
        {
                CAboutDlg dlgAbout;
                dlgAbout.DoModal();
        }
        else
        {
                CDialog::OnSysCommand(nID, lParam);
        }
}

// もしダイアログボックスに最小化ボタンを追加するならば、アイコンを描画する
// コードを以下に記述する必要があります。MFC アプリケーションは document/view
// モデルを使っているので、この処理はフレームワークにより自動的に処理されます。

void CMCOperDlg::OnPaint()
{
```

```
	if (IsIconic())
	{
		CPaintDC dc(this); // 描画用のデバイス コンテキスト

		SendMessage(WM_ICONERASEBKGND, (WPARAM) dc.GetSafeHdc(), 0);

		// クライアントの矩形領域内の中央
		int cxIcon = GetSystemMetrics(SM_CXICON);
		int cyIcon = GetSystemMetrics(SM_CYICON);
		CRect rect;
		GetClientRect(&rect);
		int x = (rect.Width() - cxIcon + 1) / 2;
		int y = (rect.Height() - cyIcon + 1) / 2;

		// アイコンを描画します。
		dc.DrawIcon(x, y, m_hIcon);
	}
	else
	{
		CDialog::OnPaint();
	}
}

// システムは、ユーザーが最小化ウィンドウをドラッグしている間、
// カーソルを表示するためにここを呼び出します。
HCURSOR CMCOperDlg::OnQueryDragIcon()
{
	return (HCURSOR) m_hIcon;
}




volatile unsigned int counter ;
int pos_x=0;
```

```
void CMCOperDlg::OnHssbopen()
{
        // TODO: この位置にコントロール通知ハンドラ用のコードを追加してください
  //    char Disptmp[16];
        //CString CIntIP,Disp;
        //UINT      CIntPN;

        okk.HssbOpen();
    m_message ="HssbOpen";
    StartFlag = true;
        UpdateData(false);

        // timer
        SetTimer(READTIMER_ID,500,NULL);

        m_hssbopen.EnableWindow(FALSE);
    m_hssbcolse.EnableWindow(TRUE);
        m_fc.EnableWindow(FALSE);
        m_fo.EnableWindow(FALSE);
    GetDlgItem(IDC_RADIO_SOUSA)->EnableWindow(TRUE);
        GetDlgItem(IDC_RADIO_SAVER)->EnableWindow(TRUE);

        if(m_con==1)
        {
            m_ciclestart.EnableWindow(TRUE);
        m_feedhold.EnableWindow(TRUE);
        m_feedscrol.EnableWindow(TRUE);
        m_spindlescrol.EnableWindow(TRUE);
                m_emegstop.EnableWindow(TRUE);
                GetDlgItem(IDC_NCSELECT)->EnableWindow(TRUE);
                GetDlgItem(IDC_NCSEND)->EnableWindow(TRUE);
                GetDlgItem(IDC_FORCE_OPEN)->EnableWindow(TRUE);
                GetDlgItem(IDC_FORCE_CUT)->EnableWindow(TRUE);

        }
```

```
	if(m_server==1)
	{
		GetDlgItem(IDC_SERVER_CONNECT)->EnableWindow(TRUE);
	}
}

void CMCOperDlg::OnNcselect()
{
	// TODO: この位置にコントロール通知ハンドラ用のコードを追加してください

	char	buf[256];
	int		len,line_num=0;

	CFileDialog FileDLG(
		true,			//[ファイルを開く]ダイアログボックス
		"out",			//デフォルト拡張子
		"*.out",  //デフォルトファイル名
		OFN_FILEMUSTEXIST | OFN_HIDEREADONLY,  //ダイアログのカスタマイズ
		"NC プログラム(*.out)|*.out|すべて(*.*)|*.*||"  //フィルタ
		);

	// !! 初期ディレクトリ 要変更 !!
	FileDLG.m_ofn.lpstrInitialDir = "D:¥¥ljq¥¥teleokk¥¥MCOper";

	if(FileDLG.DoModal()==IDOK){
		CStdioFile fin(FileDLG.GetPathName(),CFile::modeRead);
		m_ncprog.Empty();
		while(fin.ReadString(buf,255)!=NULL){
			len = strlen(buf);
			buf[len-1] = '¥0';
			m_ncprog += buf;
			m_ncprog += "¥r¥n";
			line_num++;
		}
		NCNAME= FileDLG.GetFileName();
	}
```

```
    m_ncsend.EnableWindow(TRUE);
        UpdateData(false);

}

void CMCOperDlg::OnNcsend()
{


        char buf[255];
        int   Datlen, len, line_num;
        int   ProgNumOrg;

        m_message =″NC Program Send.″;
        UpdateData(false);

        // NC のモードを編集モードにする。
        okk.SetMode(NCMODE_EDIT);
        m_message += ″Change to EDIT-Mode.¥r¥n″;
        UpdateData(false);

        // NC プログラム転送準備
        okk.PutNCProgStart();
        m_message += ″Ready to trans.(PC->MC)¥r¥n″;
        UpdateData(false);

        // NC プログラムを受信し、MC へ転送
        int j = 0;

                CStdioFile fin(NCNAME,CFile::modeRead|CFile::typeText);
                m_ncprog.Empty();
                while(fin.ReadString( buf,255)!=NULL){
                        len = strlen(buf);
            okk.PutNCProgLine(buf);
                        buf[len-1] = '¥0';
                        m_ncprog += buf;
```

```
                    m_ncprog += "¥r¥n";
                    line_num++;
                UpdateData(false);
                }

        // NC プログラム転送終了
        okk.PutNCProgEnd();
        m_message = "FileTrans Done.¥r¥n";
        UpdateData(false);

        // NC プログラム呼び出し
        ProgNumOrg = okk.GetNCProgNum();
        okk.SetNCProgNum(112);

        // NC のモードをメモリ運転モードにする。
        okk.SetMode(NCMODE_MEMORY);
        m_message = "Change to MEMORY-Mode.¥r¥n";
        UpdateData(false);

}

void CMCOperDlg::OnCiclestart()
{
  okk.SetSpinOvr(int(100));
  okk.SetFeedOvr1(int(100));
  okk.CycleStart();
  m_message ="CICLE START¥r¥n";
  UpdateData(false);
}

void CMCOperDlg::OnFeedhold()
{
  okk.FeedHold();
  m_message ="FEED HOLD";
  UpdateData(false);
}
```

```
void CMCOperDlg::OnEmegstop()
{

   KillTimer(READTIMER_ID);
   okk.Emgstp();

   m_hssbopen.EnableWindow(TRUE);
   m_hssbcolse.EnableWindow(FALSE);
   m_ciclestart.EnableWindow(FALSE);
   m_feedhold.EnableWindow(FALSE);
   m_ciclestart.EnableWindow(FALSE);
   m_ncsend.EnableWindow(FALSE);
   m_feedscrol.EnableWindow(FALSE);
   m_spindlescrol.EnableWindow(FALSE);
   m_emegstop.EnableWindow(FALSE);

   m_message ="EMEG. STOP";
   UpdateData(false);
}

void CMCOperDlg::OnHssbcolse()
{
          counter = 0 ;

     if(StartFlag == true)
          {
              KillTimer(READTIMER_ID);
         SockServ.Close();
         StartFlag = false;
     }

          OnForceCut();
     okk.HssbClose();

   m_hssbopen.EnableWindow(TRUE);
```

```
    m_hssbcolse.EnableWindow(FALSE);
    m_ciclestart.EnableWindow(FALSE);
    m_feedhold.EnableWindow(FALSE);
    m_emegstop.EnableWindow(FALSE);
    m_ncsend.EnableWindow(FALSE);
    m_feedscrol.EnableWindow(FALSE);
    m_spindlescrol.EnableWindow(FALSE);
    GetDlgItem(IDC_NCSELECT)->EnableWindow(FALSE);
    GetDlgItem(IDC_SERVER_CONNECT)->EnableWindow(FALSE);
    GetDlgItem(IDC_SERVER_CUT)->EnableWindow(FALSE);
    GetDlgItem(IDC_FORCE_OPEN)->EnableWindow(FALSE);
    GetDlgItem(IDC_FORCE_CUT)->EnableWindow(FALSE);
    GetDlgItem(IDC_RADIO_SOUSA)->EnableWindow(TRUE);
    GetDlgItem(IDC_RADIO_SAVER)->EnableWindow(TRUE);

    m_message ="HSSB CLOSE";
    UpdateData(false);

}


void CMCOperDlg::OnHScroll(UINT nSBCode, UINT nPos, CScrollBar *pScrollBar)
{
            int pos, x, xmin, xmax;
            char text[20];

            pScrollBar->GetScrollRange(&xmin, &xmax);
            pos = pScrollBar->GetScrollPos();
            x = pos;

            switch(nSBCode)
            {
                 case SB_LINELEFT:
                          x-=2;
                    break;
            case SB_LINERIGHT:
```

```
                x+=2;
            break;
        case SB_PAGELEFT:
                x-=10;
            break;
        case SB_PAGERIGHT:
                x+=10;
                break;
        case SB_THUMBTRACK:
                x=nPos;
            break;
             default:
                    return;
        }
        if(x<xmin) x=xmin;
        else if(x>xmax) x=xmax;
        if(x!=pos)
        {
            pScrollBar->SetScrollPos(x);

            if(pScrollBar == m_hsfeedover){
            //      sprintf(text, "%d", x);
                    m_feedover = x;
      okk.SetFeedOvr1(m_feedover);
                    m_message ="FEED OVERRIDE";
                }
                else if(pScrollBar ==m_hsspindleover){
      //    sprintf(text, "%d", x);
                    m_spindleover = x;
      okk.SetSpinOvr(m_spindleover);
                    m_message ="SPINDLE OVERRIDE";
                }
                UpdateData(false);

        }
}
```

```
void CMCOperDlg::OnTimer(UINT nIDEvent)
{
	if(nIDEvent == READTIMER_ID)
	{
	//Get Relative axis position.
		okk.GetAxisPosition(POS_RELATIVE);
		sprintf(dummy,"%.3f",(okk.axisPosition[0])*0.001);
		m_relx = dummy;
		sprintf(dummy,"%.3f",(okk.axisPosition[1])*0.001);
		m_rely = dummy;
		sprintf(dummy,"%.3f",(okk.axisPosition[2])*0.001);
		m_relz = dummy;

		if(m_server==1)
		{
		//send to client
		    sprintf(sData.command, "Relative axis pos %d", counter);
		    sData.fv[0] = double((okk.axisPosition[0])*0.001);
		    sData.fv[1] = double((okk.axisPosition[1])*0.001);
		    sData.fv[2] = double((okk.axisPosition[2])*0.001);
		    SockServ.SockWrite(sData);
		}

		// Get machine axis position.
		okk.GetAxisPosition(POS_MACHINE);
		sprintf(dummy,"%.3f",(okk.axisPosition[0])*0.001);
		m_macx = dummy;
		sprintf(dummy,"%.3f",(okk.axisPosition[1])*0.001);
		m_macy = dummy;
		sprintf(dummy,"%.3f",(okk.axisPosition[2])*0.001);
		m_macz = dummy;

		if(m_server==1)
		{
```

```
        //send to client
        sprintf(sData.command, "Machine axis pos. %d", counter);
        sData.fv[0] = double((okk.axisPosition[0])*0.001);
        sData.fv[1] = double((okk.axisPosition[1])*0.001);
        sData.fv[2] = double((okk.axisPosition[2])*0.001);
        SockServ.SockWrite(sData);
    }


    // Get absolute axis position.
    okk.GetAxisPosition(POS_ABSOLUTE);
    sprintf(dummy,"%.3f",(okk.axisPosition[0])*0.001);
    m_absx = dummy;
    sprintf(dummy,"%.3f",(okk.axisPosition[1])*0.001);
    m_absy = dummy;
    sprintf(dummy,"%.3f",(okk.axisPosition[2])*0.001);
    m_absz = dummy;

    if(m_server==1)
    {
        //send to client
        sprintf(sData.command, "Absolute axis post. %d", counter);
        sData.fv[0] = double((okk.axisPosition[0])*0.001);
        sData.fv[1] = double((okk.axisPosition[1])*0.001);
        sData.fv[2] = double((okk.axisPosition[2])*0.001);
        SockServ.SockWrite(sData);
    }

// Get remain moving   distance .
    okk.GetAxisPosition(POS_DISTANCE);
    sprintf(dummy,"%.3f",(okk.axisPosition[0])*0.001);
    m_remx = dummy;
    sprintf(dummy,"%.3f",(okk.axisPosition[1])*0.001);
    m_remy = dummy;
    sprintf(dummy,"%.3f",(okk.axisPosition[2])*0.001);
    m_remz = dummy;
```

```
        if(m_server==1)
        {
                //send to client
            sprintf(sData.command, "No moving distance %d", counter);
            sData.fv[0] = double((okk.axisPosition[0])*0.001);
            sData.fv[1] = double((okk.axisPosition[1])*0.001);
            sData.fv[2] = double((okk.axisPosition[2])*0.001);
            SockServ.SockWrite(sData);
        }

        // Get feed rate and spindle speed
        int feed, spindle;
        feed = okk.GetFeedAct();
        sprintf(dummy,"%d",feed);
        m_feed = dummy;
        spindle = okk.GetSpinAct();
        sprintf(dummy,"%d",spindle);
        m_spindle = dummy;

        if(m_server==1)
        {
            //send to client
            sprintf(sData.command, "Spindle speed and feed rate %d", counter);
            sData.fv[0] = double(feed);
            sData.fv[1] = double(spindle);
            sData.fv[2] = double(0.0);
            SockServ.SockWrite(sData);
        }

        UpdateData(false);
}


if(m_server==1)
{
```

```
  if(NCProgFlag == 1)
  {
            // NC prog accept
SockServ.SockRead(&rData);
okk.PutNCProgLine(rData.command);
            if(rData.command[0]=='O'){
                ProgNum = int (rData.fv[0]);
            }
    if(rData.command[0]=='%')  {
      NCProgFlag = 0;
    // NC プログラム転送終了
    okk.PutNCProgEnd();
                sprintf(dummy, "FileTrans Done, NC prog. Num %d",
ProgNum);
    m_message = dummy;
                m_message += "¥r¥n";
    UpdateData(false);

    // NC プログラム呼び出し
          // int ProgNumOrg;
   // ProgNumOrg = okk.GetNCProgNum();
    okk.SetNCProgNum(ProgNum);

    // NC のモードをメモリ運転モードにする。
    okk.SetMode(NCMODE_MEMORY);
    // m_message = "Change to MEMORY-Mode.¥r¥n";
   // UpdateData(false);
       sprintf(sData.command, "Nc prog sending to machine has end. %d",
counter);
        sData.fv[0] = double(0.0);
        sData.fv[1] = double(0.0);
        sData.fv[2] = double(0.0);
        SockServ.SockWrite(sData);
            return;
      }
    m_ncprog  += rData.command;
```

```
                m_ncprog  += "¥r¥n";
                UpdateData(false);
                return;
            }
            rData.command[0] = NULL;
            SockServ.SockRead(&rData);
if(rData.command[0]!=NULL){
              m_message = rData.command;
            //   overvalue = int (rData.fv[0]);
           //    sprintf(dummy,"     %d", overvalue);
  //      m_message += dummy;
        m_message += "¥r¥n";
                    if(rData.command[0] =='N'){
                            NCProgFlag = 1;

                            // NC のモードを編集モードにする。
                m_message = "Change to EDIT-Mode.¥r¥n";
                UpdateData(false);

                // NC プログラム転送準備
                okk.PutNCProgStart();
                m_message = "Ready to trans.(PC->MC)";
                            m_message += "¥r¥n";
                UpdateData(false);

                    }else if (rData.command[0] =='C'){
              okk.SetSpinOvr(int(100));
              okk.SetFeedOvr1(int(100));
              okk.CycleStart();
                    }else if (rData.command[0] =='E'){
              KillTimer(READTIMER_ID);
              okk.Emgstp();
                    }else if (rData.command[0] =='H'){
              okk.FeedHold();
                    }else if (rData.command[0] =='S'){
```

```
                m_spindleover = int (rData.fv[0]);
                okk.SetSpinOvr(m_spindleover);
                        }else if (rData.command[0] =='F'){
                m_feedover = int (rData.fv[0]);
                okk.SetFeedOvr1(m_feedover);
              }else if (rData.command[0] =='A'){
                   okk.DncStart();
                   m_message="DNC 運転をスタートします";
              }else if (rData.command[0] =='B'){
                   okk.DncEnd();
                   m_message="DNC 運転を終了します";
                        }else if (rData.command[0] =='Y'){
                                sprintf(dummy,"S %.0f rpm F %.0f mm/min Y 方向
へ %.3f mm 動かします",rData.fv[1],rData.fv[2],rData.fv[0]);
                                  m_message=dummy;

sprintf(dummy,"S%.0fM03F%.0fG91G01Y%.3f¥n",rData.fv[1],rData.fv[2],rData.fv[0]);
                   okk.SetNCCodeDNC(dummy);
                                okk.SetNCCodeDNC("%");

                        }else if (rData.command[0] =='X'){
                                sprintf(dummy,"S %.0f rpm F %.0f mm/min X 方向
へ %.3f mm 動かします",rData.fv[1],rData.fv[2],rData.fv[0]);
                                  m_message=dummy;

        sprintf(dummy,"S%.0fM03F%.0fG91G01X%.3f¥n",rData.fv[1],rData.fv[2],rData.fv[0]);
                   okk.SetNCCodeDNC(dummy);
                                okk.SetNCCodeDNC("%");
                        }else if (rData.command[0] =='Z'){
                                sprintf(dummy,"S %.0f rpm F %.0f mm/min Z 方向
へ %.3f mm 動かします",rData.fv[1],rData.fv[2],rData.fv[0]);
                                  m_message=dummy;

        sprintf(dummy,"S%.0fM03F%.0fG91G01Z%.3f¥n",rData.fv[1],rData.fv[2],rData.fv[0]);
                   okk.SetNCCodeDNC(dummy);
                                okk.SetNCCodeDNC("%");
```

```
                    }else if(rData.command[0] =='P'){
                            OnForceOpen();
                    }else if(rData.command[0] =='O'){
                            OnForceCut();
                    }
            UpdateData(false);
                return;
              }
              }


    if(nIDEvent == 1)
    {
  double data_x,data_y,data_z;
            data_x=data_y=data_z=0;
  tu=tu+1;

    SmplChReq[0].ulChNo = 1;
    SmplChReq[0].ulRange = AD_10V;
  SmplChReq[1].ulChNo = 2;
    SmplChReq[1].ulRange = AD_10V;
  SmplChReq[2].ulChNo = 3;
    SmplChReq[2].ulRange = AD_10V;

  AdInputAD(hDeviceHandle, 3, AD_INPUT_SINGLE, &SmplChReq[0], &wSmpData);

              data_x = 2000.0/65536*(wSmpData[0]-32768);
              data_y = 2000.0/65536*(wSmpData[1]-32768);
              data_z = 2000.0/65536*(wSmpData[2]-32768);


//画面に表示
              sprintf(dummy,"%.3f",data_x);
              m_forcex = dummy;
              sprintf(dummy,"%.3f",data_y);
              m_forcey = dummy;
```

```
                sprintf(dummy,"%.3f",data_z);
                m_forcez = dummy;

                if(m_server==1)
                {
                    //send to client
                    sprintf(sData.command, "Kisaki force. %d", counter);
                    sData.fv[0] = double(data_x);
                    sData.fv[1] = double(data_y);
                    sData.fv[2] = double(data_z);
                    SockServ.SockWrite(sData);
                }

        double adx, adyx,adyy,adyz;

            if(data_x>100)
            { data_x=100; }
            if(data_x<-50)
            { data_x=-50; }
            if(data_y>100)
            { data_y=100; }
            if(data_y<-50)
            { data_y=-50; }
            if(data_z>100)
            { data_z=100; }
            if(data_z<-50)
            { data_z=-50; }

            adx   = tu;
            adyx = 0.5 * -data_x + gf_x_y;
            adyy = 0.5 * -data_y + gf_y_y;
            adyz = 0.5 * -data_z + gf_z_y;
//Xのデータを表示
            CDC* pDC=m_pictx.GetDC();
             //pDC->SetPixel(x,y, RGB(0,0,255));
            CPen BluePen,RedPen,GreenPen,BrackPen;
```

```
            BluePen.CreatePen(PS_SOLID,2,RGB(0,0,255));


            pDC->SelectObject(&BluePen);
            pDC->MoveTo(adx0,ady1);
            pDC->LineTo(adx,adyx);

            ady1=adyx;

//Yのデータを表示
        CDC* pDC2=m_picty.GetDC();

        RedPen.CreatePen(PS_SOLID,2,RGB(255,0,0));

        pDC->SelectObject(&RedPen);
            pDC->MoveTo(adx0,ady2);
            pDC->LineTo(adx,adyy);

            ady2=adyy;

//Zのデータを表示
        CDC* pDC3=m_pictz.GetDC();
        GreenPen.CreatePen(PS_SOLID,2,RGB(0,255,0));

            pDC->SelectObject(&GreenPen);
            pDC->MoveTo(adx0,ady3);
            pDC->LineTo(adx,adyz);

            adx0=adx;
            ady3=adyz;

            if (tu > 370)
            {
//グラフエリアを白で塗りつぶす
          //X
          CDC* pDC=m_pictx.GetDC();
```

```
        CRect myRECT;
        m_pictx.GetClientRect(myRECT);
        pDC->FillSolidRect(myRECT, RGB(255,255,255));
    //Y
        CDC* pDC2=m_picty.GetDC();
        CRect myRECT2;
        m_picty.GetClientRect(myRECT2);
        pDC2->FillSolidRect(myRECT2, RGB(255,255,255));
    //Z
        CDC* pDC3=m_pictz.GetDC();
        CRect myRECT3;
        m_pictz.GetClientRect(myRECT3);
        pDC3->FillSolidRect(myRECT3, RGB(255,255,255));

//x軸、y軸の初期設定
    pDC->SelectObject(&BrackPen);
//X
        pDC->MoveTo( gf_x       , gf_x_y     ); //X軸
        pDC->LineTo( gf_x +380 , gf_x_y     );
        pDC->MoveTo( gf_x       , gf_x_y -45); //Y軸
        pDC->LineTo( gf_x       , gf_x_y +25);

        pDC->MoveTo( gf_x  +50 , gf_x_y  -3); //X軸目盛
        pDC->LineTo( gf_x  +50 , gf_x_y  +4);
        pDC->MoveTo( gf_x +100 , gf_x_y  -3);
        pDC->LineTo( gf_x +100 , gf_x_y  +4);
        pDC->MoveTo( gf_x +150 , gf_x_y  -3);
        pDC->LineTo( gf_x +150 , gf_x_y  +4);
        pDC->MoveTo( gf_x +200 , gf_x_y  -3);
        pDC->LineTo( gf_x +200 , gf_x_y  +4);
        pDC->MoveTo( gf_x +250 , gf_x_y  -3);
        pDC->LineTo( gf_x +250 , gf_x_y  +4);
        pDC->MoveTo( gf_x +300 , gf_x_y  -3);
        pDC->LineTo( gf_x +300 , gf_x_y  +4);
        pDC->MoveTo( gf_x +350 , gf_x_y  -3);
        pDC->LineTo( gf_x +350 , gf_x_y  +4);
```

```
        pDC->MoveTo( gf_x    -2 , gf_x_y -40); //Y軸目盛
        pDC->LineTo( gf_x    +3 , gf_x_y -40);
        pDC->MoveTo( gf_x    -2 , gf_x_y -30);
        pDC->LineTo( gf_x    +3 , gf_x_y -30);
        pDC->MoveTo( gf_x    -2 , gf_x_y -20);
        pDC->LineTo( gf_x    +3 , gf_x_y -20);
        pDC->MoveTo( gf_x    -2 , gf_x_y -10);
        pDC->LineTo( gf_x    +3 , gf_x_y -10);
        pDC->MoveTo( gf_x    -2 , gf_x_y +10);
        pDC->LineTo( gf_x    +3 , gf_x_y +10);
        pDC->MoveTo( gf_x    -2 , gf_x_y +20);
        pDC->LineTo( gf_x    +3 , gf_x_y +20);

//Y
        pDC->MoveTo( gf_x        , gf_y_y     ); //X軸
        pDC->LineTo( gf_x +380 , gf_y_y     );
        pDC->MoveTo( gf_x        , gf_y_y -45); //Y軸
        pDC->LineTo( gf_x        , gf_y_y +25);

        pDC->MoveTo( gf_x   +50 , gf_y_y   -3); //X軸目盛
        pDC->LineTo( gf_x   +50 , gf_y_y   +4);
        pDC->MoveTo( gf_x +100 , gf_y_y   -3);
        pDC->LineTo( gf_x +100 , gf_y_y   +4);
        pDC->MoveTo( gf_x +150 , gf_y_y   -3);
        pDC->LineTo( gf_x +150 , gf_y_y   +4);
        pDC->MoveTo( gf_x +200 , gf_y_y   -3);
        pDC->LineTo( gf_x +200 , gf_y_y   +4);
        pDC->MoveTo( gf_x +250 , gf_y_y   -3);
        pDC->LineTo( gf_x +250 , gf_y_y   +4);
        pDC->MoveTo( gf_x +300 , gf_y_y   -3);
        pDC->LineTo( gf_x +300 , gf_y_y   +4);
        pDC->MoveTo( gf_x +350 , gf_y_y   -3);
        pDC->LineTo( gf_x +350 , gf_y_y   +4);

        pDC->MoveTo( gf_x    -2 , gf_y_y -40); //Y軸目盛
```

```
        pDC->LineTo( gf_x    +3 , gf_y_y -40);
        pDC->MoveTo( gf_x    -2 , gf_y_y -30);
        pDC->LineTo( gf_x    +3 , gf_y_y -30);
        pDC->MoveTo( gf_x    -2 , gf_y_y -20);
        pDC->LineTo( gf_x    +3 , gf_y_y -20);
        pDC->MoveTo( gf_x    -2 , gf_y_y -10);
        pDC->LineTo( gf_x    +3 , gf_y_y -10);
        pDC->MoveTo( gf_x    -2 , gf_y_y +10);
        pDC->LineTo( gf_x    +3 , gf_y_y +10);
        pDC->MoveTo( gf_x    -2 , gf_y_y +20);
        pDC->LineTo( gf_x    +3 , gf_y_y +20);

//Z
        pDC->MoveTo( gf_x        , gf_z_y     ); //X軸
        pDC->LineTo( gf_x +380 , gf_z_y     );
        pDC->MoveTo( gf_x        , gf_z_y -45); //Y軸
        pDC->LineTo( gf_x        , gf_z_y +25);

        pDC->MoveTo( gf_x   +50 , gf_z_y   -3); //X軸目盛
        pDC->LineTo( gf_x   +50 , gf_z_y   +4);
        pDC->MoveTo( gf_x +100 , gf_z_y   -3);
        pDC->LineTo( gf_x +100 , gf_z_y   +4);
        pDC->MoveTo( gf_x +150 , gf_z_y   -3);
        pDC->LineTo( gf_x +150 , gf_z_y   +4);
        pDC->MoveTo( gf_x +200 , gf_z_y   -3);
        pDC->LineTo( gf_x +200 , gf_z_y   +4);
        pDC->MoveTo( gf_x +250 , gf_z_y   -3);
        pDC->LineTo( gf_x +250 , gf_z_y   +4);
        pDC->MoveTo( gf_x +300 , gf_z_y   -3);
        pDC->LineTo( gf_x +300 , gf_z_y   +4);
        pDC->MoveTo( gf_x +350 , gf_z_y   -3);
        pDC->LineTo( gf_x +350 , gf_z_y   +4);

        pDC->MoveTo( gf_x    -2 , gf_z_y -40); //Y軸目盛
        pDC->LineTo( gf_x    +3 , gf_z_y -40);
        pDC->MoveTo( gf_x    -2 , gf_z_y -30);
```

```
            pDC->LineTo( gf_x    +3 , gf_z_y -30);
            pDC->MoveTo( gf_x    -2 , gf_z_y -20);
            pDC->LineTo( gf_x    +3 , gf_z_y -20);
            pDC->MoveTo( gf_x    -2 , gf_z_y -10);
            pDC->LineTo( gf_x    +3 , gf_z_y -10);
            pDC->MoveTo( gf_x    -2 , gf_z_y +10);
            pDC->LineTo( gf_x    +3 , gf_z_y +10);
            pDC->MoveTo( gf_x    -2 , gf_z_y +20);
            pDC->LineTo( gf_x    +3 , gf_z_y +20);

            tu    = gf_x;
        adx0 = gf_x;
            ady1 = gf_x_y;   ady2 = gf_y_y;   ady3 = gf_z_y;
            }
            }

                    CDialog::OnTimer(nIDEvent);
}


void CMCOperDlg::OnExit()
{
        if(StartFlag == true) {
                KillTimer(READTIMER_ID);
                okk.HssbClose();
        }
            CDialog::OnClose();
            PostMessage(WM_CLOSE, 0, 0);
}

void CMCOperDlg::OnRadioSaver()
{
            m_server=1;
            m_con=0;
            m_fc.EnableWindow(FALSE);
            m_fo.EnableWindow(FALSE);
```

```
        GetDlgItem(IDC_HSSBOPEN)->EnableWindow(TRUE);
}

void CMCOperDlg::OnRadioSousa()
{
    m_server=0;
        m_con=1;
                m_fc.EnableWindow(TRUE);
        m_fo.EnableWindow(TRUE);
        GetDlgItem(IDC_HSSBOPEN)->EnableWindow(TRUE);
}

void CMCOperDlg::OnServerConnect()
{
// サーバ起動、ソケット待ち受け
        char Disptmp[16];
        CString ClntIP,Disp;
        UINT    ClntPN;

        m_message = "Waiting for request to connect.¥r¥n";
        UpdateData(false);
        SockServ.Initialize(10000);
        m_message = "Connection Established.¥r¥n";
        UpdateData(false);

        Disp = "Connect : ";
        SockServ.GetPeerName(ClntIP,ClntPN);
        Disp += ClntIP;
        sprintf(Disptmp," : %d",ClntPN);
        Disp += Disptmp;
        AfxMessageBox(Disp);

        GetDlgItem(IDC_SERVER_CONNECT)->EnableWindow(FALSE);
        GetDlgItem(IDC_SERVER_CUT)->EnableWindow(TRUE);
}
```

```
void CMCOperDlg::OnForceOpen()
{


//初期設定
        hDeviceHandle = AdOpen("FBIAD1");
        if (hDeviceHandle == INVALID_HANDLE_VALUE){
                MessageBox("Device FBIAD1 は使用できません!");
               }

//X
        CDC* pDC=m_pictx.GetDC();
        CRect myRECT;
        m_pictx.GetClientRect(myRECT);
        pDC->FillSolidRect(myRECT, RGB(255,255,255));

        pDC->MoveTo( gf_x       , gf_x_y     ); //X軸
        pDC->LineTo( gf_x +380 , gf_x_y     );
        pDC->MoveTo( gf_x       , gf_x_y -45); //Y軸
        pDC->LineTo( gf_x       , gf_x_y +25);

        pDC->MoveTo( gf_x  +50 , gf_x_y  -3); //X軸目盛
        pDC->LineTo( gf_x  +50 , gf_x_y  +4);
        pDC->MoveTo( gf_x +100 , gf_x_y  -3);
        pDC->LineTo( gf_x +100 , gf_x_y  +4);
        pDC->MoveTo( gf_x +150 , gf_x_y  -3);
        pDC->LineTo( gf_x +150 , gf_x_y  +4);
        pDC->MoveTo( gf_x +200 , gf_x_y  -3);
        pDC->LineTo( gf_x +200 , gf_x_y  +4);
        pDC->MoveTo( gf_x +250 , gf_x_y  -3);
        pDC->LineTo( gf_x +250 , gf_x_y  +4);
        pDC->MoveTo( gf_x +300 , gf_x_y  -3);
        pDC->LineTo( gf_x +300 , gf_x_y  +4);
        pDC->MoveTo( gf_x +350 , gf_x_y  -3);
        pDC->LineTo( gf_x +350 , gf_x_y  +4);
```

```
        pDC->MoveTo( gf_x    -2 , gf_x_y -40); //Y軸目盛
        pDC->LineTo( gf_x    +3 , gf_x_y -40);
        pDC->MoveTo( gf_x    -2 , gf_x_y -30);
        pDC->LineTo( gf_x    +3 , gf_x_y -30);
        pDC->MoveTo( gf_x    -2 , gf_x_y -20);
        pDC->LineTo( gf_x    +3 , gf_x_y -20);
        pDC->MoveTo( gf_x    -2 , gf_x_y -10);
        pDC->LineTo( gf_x    +3 , gf_x_y -10);
        pDC->MoveTo( gf_x    -2 , gf_x_y +10);
        pDC->LineTo( gf_x    +3 , gf_x_y +10);
        pDC->MoveTo( gf_x    -2 , gf_x_y +20);
        pDC->LineTo( gf_x    +3 , gf_x_y +20);

//Y
    CDC* pDC2=m_picty.GetDC();
        CRect myRECT2;
        m_picty.GetClientRect(myRECT2);
        pDC2->FillSolidRect(myRECT2, RGB(255,255,255));

        pDC->MoveTo( gf_x        , gf_y_y      ); //X軸
        pDC->LineTo( gf_x +380 , gf_y_y      );
        pDC->MoveTo( gf_x        , gf_y_y -45); //Y軸
        pDC->LineTo( gf_x        , gf_y_y +25);

        pDC->MoveTo( gf_x   +50 , gf_y_y   -3); //X軸目盛
        pDC->LineTo( gf_x   +50 , gf_y_y   +4);
        pDC->MoveTo( gf_x +100 , gf_y_y   -3);
        pDC->LineTo( gf_x +100 , gf_y_y   +4);
        pDC->MoveTo( gf_x +150 , gf_y_y   -3);
        pDC->LineTo( gf_x +150 , gf_y_y   +4);
        pDC->MoveTo( gf_x +200 , gf_y_y   -3);
        pDC->LineTo( gf_x +200 , gf_y_y   +4);
        pDC->MoveTo( gf_x +250 , gf_y_y   -3);
        pDC->LineTo( gf_x +250 , gf_y_y   +4);
        pDC->MoveTo( gf_x +300 , gf_y_y   -3);
        pDC->LineTo( gf_x +300 , gf_y_y   +4);
```

```
        pDC->MoveTo( gf_x +350 , gf_y_y   -3);
        pDC->LineTo( gf_x +350 , gf_y_y   +4);

        pDC->MoveTo( gf_x    -2 , gf_y_y -40); //Y軸目盛
        pDC->LineTo( gf_x    +3 , gf_y_y -40);
        pDC->MoveTo( gf_x    -2 , gf_y_y -30);
        pDC->LineTo( gf_x    +3 , gf_y_y -30);
        pDC->MoveTo( gf_x    -2 , gf_y_y -20);
        pDC->LineTo( gf_x    +3 , gf_y_y -20);
        pDC->MoveTo( gf_x    -2 , gf_y_y -10);
        pDC->LineTo( gf_x    +3 , gf_y_y -10);
        pDC->MoveTo( gf_x    -2 , gf_y_y +10);
        pDC->LineTo( gf_x    +3 , gf_y_y +10);
        pDC->MoveTo( gf_x    -2 , gf_y_y +20);
        pDC->LineTo( gf_x    +3 , gf_y_y +20);

//Z
    CDC* pDC3=m_pictz.GetDC();
        CRect myRECT3;
        m_pictz.GetClientRect(myRECT2);
        pDC3->FillSolidRect(myRECT, RGB(255,255,255));

        pDC->MoveTo( gf_x        , gf_z_y      ); //X軸
        pDC->LineTo( gf_x +380 , gf_z_y      );
        pDC->MoveTo( gf_x        , gf_z_y -45); //Y軸
        pDC->LineTo( gf_x        , gf_z_y +25);

        pDC->MoveTo( gf_x   +50 , gf_z_y   -3); //X軸目盛
        pDC->LineTo( gf_x   +50 , gf_z_y   +4);
        pDC->MoveTo( gf_x +100 , gf_z_y   -3);
        pDC->LineTo( gf_x +100 , gf_z_y   +4);
        pDC->MoveTo( gf_x +150 , gf_z_y   -3);
        pDC->LineTo( gf_x +150 , gf_z_y   +4);
        pDC->MoveTo( gf_x +200 , gf_z_y   -3);
        pDC->LineTo( gf_x +200 , gf_z_y   +4);
        pDC->MoveTo( gf_x +250 , gf_z_y   -3);
```

```
        pDC->LineTo( gf_x +250 , gf_z_y   +4);
        pDC->MoveTo( gf_x +300 , gf_z_y   -3);
        pDC->LineTo( gf_x +300 , gf_z_y   +4);
        pDC->MoveTo( gf_x +350 , gf_z_y   -3);
        pDC->LineTo( gf_x +350 , gf_z_y   +4);

        pDC->MoveTo( gf_x    -2 , gf_z_y -40); //Y軸目盛
        pDC->LineTo( gf_x    +3 , gf_z_y -40);
        pDC->MoveTo( gf_x    -2 , gf_z_y -30);
        pDC->LineTo( gf_x    +3 , gf_z_y -30);
        pDC->MoveTo( gf_x    -2 , gf_z_y -20);
        pDC->LineTo( gf_x    +3 , gf_z_y -20);
        pDC->MoveTo( gf_x    -2 , gf_z_y -10);
        pDC->LineTo( gf_x    +3 , gf_z_y -10);
        pDC->MoveTo( gf_x    -2 , gf_z_y +10);
        pDC->LineTo( gf_x    +3 , gf_z_y +10);
        pDC->MoveTo( gf_x    -2 , gf_z_y +20);
        pDC->LineTo( gf_x    +3 , gf_z_y +20);

        //タイマーセット
        SetTimer(1,100,NULL);

        tu    = gf_x;
    adx0 = gf_x;
        ady1 = gf_x_y;  ady2 = gf_y_y;  ady3 = gf_z_y;

}

void CMCOperDlg::OnForceCut()
{
            KillTimer(1);
            AdClose(hDeviceHandle);

//グラフエリアを白で塗りつぶす
        //X
            CDC* pDC=m_pictx.GetDC();
```

```
            CRect myRECT;
            m_pictx.GetClientRect(myRECT);
            pDC->FillSolidRect(myRECT, RGB(255,255,255));
    //Y
            CDC* pDC2=m_picty.GetDC();
            CRect myRECT2;
            m_picty.GetClientRect(myRECT2);
            pDC2->FillSolidRect(myRECT2, RGB(255,255,255));
    //Z
            CDC* pDC3=m_pictz.GetDC();
            CRect myRECT3;
            m_pictz.GetClientRect(myRECT3);
            pDC3->FillSolidRect(myRECT3, RGB(255,255,255));
}

void CMCOperDlg::OnServerCut()
{

        KillTimer(READTIMER_ID);
    SockServ.Close();
    StartFlag = false;

        GetDlgItem(IDC_SERVER_CONNECT)->EnableWindow(TRUE);
        GetDlgItem(IDC_SERVER_CUT)->EnableWindow(FALSE);
}
```

```
// pcpgDlg.cpp : インプリメンテーション ファイル
//

#include "stdafx.h"
#include "pcpg.h"
#include "pcpgDlg.h"
#include "pcpg46.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
int A=0;
#endif

pcpg46 pcpg;

        TCHAR                   szBuf[1024];
        BYTE                    bData[256];
        WORD                    wBsn, wAxis;
        PCPG46RESOURCE      ri;
        BYTE                    b;
        DWORD                   dwCount;

/////////////////////////////////////////////////////////////////////////////
///
// アプリケーションのバージョン情報で使われている CAboutDlg ダイアログ

class CAboutDlg : public CDialog
{
public:
        CAboutDlg();

// ダイアログ データ
        //{{AFX_DATA(CAboutDlg)
        enum { IDD = IDD_ABOUTBOX };
```

```
	//}}AFX_DATA

	// ClassWizard は仮想関数のオーバーライドを生成します
	//{{AFX_VIRTUAL(CAboutDlg)
	protected:
	virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV のサポート
	//}}AFX_VIRTUAL

// インプリメンテーション
protected:
	//{{AFX_MSG(CAboutDlg)
	//}}AFX_MSG
	DECLARE_MESSAGE_MAP()
};

CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
	//{{AFX_DATA_INIT(CAboutDlg)
	//}}AFX_DATA_INIT
}

void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CAboutDlg)
	//}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
	//{{AFX_MSG_MAP(CAboutDlg)
		// メッセージ ハンドラがありません。
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
///
```

```
// CPcpgDlg ダイアログ

CPcpgDlg::CPcpgDlg(CWnd* pParent /*=NULL*/)
	: CDialog(CPcpgDlg::IDD, pParent)
{
	//{{AFX_DATA_INIT(CPcpgDlg)
	m_distance = 0;
	m_feedrate = 0;
	m_massage = _T("");
	//}}AFX_DATA_INIT
	// メモ: LoadIcon は Win32 の DestroyIcon のサブシーケンスを要求しません。
	m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
}

void CPcpgDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CPcpgDlg)
	DDX_Control(pDX, IDC_Xjiku, m_chkX);
	DDX_Control(pDX, IDC_Zjiku, m_chkZ);
	DDX_Control(pDX, IDC_Yjiku, m_chkY);
	DDX_Text(pDX, IDC_EDIT_DISTANCE, m_distance);
	DDX_Text(pDX, IDC_EDIT_FEED, m_feedrate);
	DDX_Text(pDX, IDC_EDIT_MESSAGE, m_massage);
	//}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CPcpgDlg, CDialog)
	//{{AFX_MSG_MAP(CPcpgDlg)
	ON_WM_SYSCOMMAND()
	ON_WM_PAINT()
	ON_WM_QUERYDRAGICON()
	ON_BN_CLICKED(IDC_PCPGINIT, OnPcpginit)
	ON_BN_CLICKED(IDC_MINUS, OnMinus)
	ON_BN_CLICKED(IDC_PLUS, OnPlus)
	ON_BN_CLICKED(IDC_PCPGCLOSE, OnPcpgclose)
```

```
	ON_EN_CHANGE(IDC_EDIT_DISTANCE, OnChangeEditDistance)
	ON_EN_CHANGE(IDC_EDIT_FEED, OnChangeEditFeed)
	ON_BN_CLICKED(IDC_Xjiku, OnXjiku)
	ON_BN_CLICKED(IDC_Yjiku, OnYjiku)
	ON_BN_CLICKED(IDC_Zjiku, OnZjiku)
	ON_EN_CHANGE(IDC_EDIT_MESSAGE, OnChangeEditMessage)
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
///
// CPcpgDlg メッセージ ハンドラ

BOOL CPcpgDlg::OnInitDialog()
{
	CDialog::OnInitDialog();

	// "バージョン情報..." メニュー項目をシステム メニューへ追加します。

	// IDM_ABOUTBOX はコマンド メニューの範囲でなければなりません。
	ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
	ASSERT(IDM_ABOUTBOX < 0xF000);

	CMenu* pSysMenu = GetSystemMenu(FALSE);
	if (pSysMenu != NULL)
	{
		CString strAboutMenu;
		strAboutMenu.LoadString(IDS_ABOUTBOX);
		if (!strAboutMenu.IsEmpty())
		{
			pSysMenu->AppendMenu(MF_SEPARATOR);
			pSysMenu->AppendMenu(MF_STRING,		IDM_ABOUTBOX,
strAboutMenu);
		}
	}
```

```
	// このダイアログ用のアイコンを設定します。フレームワークはアプリケーションのメイ
ン
	// ウィンドウがダイアログでない時は自動的に設定しません。
	SetIcon(m_hIcon, TRUE);			// 大きいアイコンを設定
	SetIcon(m_hIcon, FALSE);		// 小さいアイコンを設定

	// TODO: 特別な初期化を行う時はこの場所に追加してください。

	return TRUE;  // TRUE を返すとコントロールに設定したフォーカスは失われません。
}

void CPcpgDlg::OnSysCommand(UINT nID, LPARAM lParam)
{
	if ((nID & 0xFFF0) == IDM_ABOUTBOX)
	{
		CAboutDlg dlgAbout;
		dlgAbout.DoModal();
	}
	else
	{
		CDialog::OnSysCommand(nID, lParam);
	}
}

// もしダイアログボックスに最小化ボタンを追加するならば、アイコンを描画する
// コードを以下に記述する必要があります。MFC アプリケーションは document/view
// モデルを使っているので、この処理はフレームワークにより自動的に処理されます。

void CPcpgDlg::OnPaint()
{
	if (IsIconic())
	{
		CPaintDC dc(this); // 描画用のデバイス コンテキスト

		SendMessage(WM_ICONERASEBKGND, (WPARAM) dc.GetSafeHdc(), 0);
```

```
                // クライアントの矩形領域内の中央
                int cxIcon = GetSystemMetrics(SM_CXICON);
                int cyIcon = GetSystemMetrics(SM_CYICON);
                CRect rect;
                GetClientRect(&rect);
                int x = (rect.Width() - cxIcon + 1) / 2;
                int y = (rect.Height() - cyIcon + 1) / 2;

                // アイコンを描画します。
                dc.DrawIcon(x, y, m_hIcon);
        }
        else
        {
                CDialog::OnPaint();
        }
}

// システムは、ユーザーが最小化ウィンドウをドラッグしている間、
// カーソルを表示するためにここを呼び出します。
HCURSOR CPcpgDlg::OnQueryDragIcon()
{
        return (HCURSOR) m_hIcon;
}

void CPcpgDlg::OnPcpginit()
{

        //--------------------------------------------------
        // Pcpg46.Dll を開く
        //--------------------------------------------------
        if ( FALSE == pcpg.Pcpg46wDllOpen() ) {
                pcpg.ErrorMessage(PCPG46_BSN_AUTO);
                return ;                // DLL のロードまたはドライバのオープンに失敗しました
        }
        //--------------------------------------------------
        // DLLバージョン情報の表示
```

```
//-----------------------------------------------------
if ( FALSE == pcpg.Pcpg46wGetLibVersion(bData)) {
        pcpg.ErrorMessage(PCPG46_BSN_AUTO);
        return ;
}
wsprintf( szBuf,
        _T("PCPG46¥n")
        _T("Dll Version %s"),
        bData );
pcpg.ResultMessage( szBuf );
//-----------------------------------------------------
// ドライババージョン情報の表示
//-----------------------------------------------------
if ( FALSE == pcpg.Pcpg46wGetDrvVersion(bData)) {
        pcpg.ErrorMessage(PCPG46_BSN_AUTO);
        return ;
}
wsprintf( szBuf,
        _T("PCPG46¥n")
        _T("Drv Version %s"),
        bData );
pcpg.ResultMessage( szBuf );
//-----------------------------------------------------
// デバイスの使用を宣言する
//-----------------------------------------------------
wBsn = PCPG46_BSN_AUTO;          // 空きデバイス自動検索を指定
if ( FALSE == pcpg.Pcpg46wCreate(&wBsn) ){
        pcpg.ErrorMessage(PCPG46_BSN_AUTO);
        return ;
}
wsprintf( szBuf,
        _T("PCPG46 Created¥n")
        _T("BSN Number %d"), wBsn );
pcpg.ResultMessage( szBuf );
//-----------------------------------------------------
// リソース情報を表示
```

```
        //-------------------------------------------------
        if ( FALSE == pcpg.Pcpg46wGetResource(wBsn,&ri)){
                pcpg.ErrorMessage(wBsn);
                return ;
        }
        wsprintf(szBuf, _T("Board   Name: %s¥n")
                                        _T("IO Address : %04Xh-%04Xh¥n"),
                                        szPcpg46,
                                        ri.dwIOPortBase[1],ri.dwIOPortBase[1]        +
ri.dwIOPortLength[1] - 1 );
        pcpg.ResultMessage(szBuf);


    //-------------------------------------------------
        // 第1軸を選択
        //-------------------------------------------------
          wAxis = PCPG46_AXIS_1;
          //-------------------------------------------------
          // 制御軸の初期化
          //-------------------------------------------------
          if ( FALSE == pcpg.Pcpg46_InitAxis(wBsn,wAxis)){
                pcpg.ErrorMessage(wBsn);
                return ;
          }
          wsprintf( szBuf,
                _T("PCPG46¥n")
                _T("制御軸 %d を初期化しました"), wAxis );
          pcpg.ResultMessage( szBuf );

    //-------------------------------------------------
        // 第 2 軸を選択
        //-------------------------------------------------
      wAxis = PCPG46_AXIS_2;
         //-------------------------------------------------
         // 制御軸の初期化
         //-------------------------------------------------
```

```
        if ( FALSE == pcpg.Pcpg46_InitAxis(wBsn,wAxis)){
                pcpg.ErrorMessage(wBsn);
                return ;
        }
        wsprintf( szBuf,
                _T("PCPG46¥n")
                _T("制御軸 %d を初期化しました"), wAxis );
        pcpg.ResultMessage( szBuf );

    //-----------------------------------------------------
        // 第 3 軸を選択
        //-----------------------------------------------------
     wAxis = PCPG46_AXIS_3;
        //-----------------------------------------------------
        // 制御軸の初期化
        //-----------------------------------------------------
        if ( FALSE == pcpg.Pcpg46_InitAxis(wBsn,wAxis)){
                pcpg.ErrorMessage(wBsn);
                return ;
        }
        wsprintf( szBuf,
                _T("PCPG46¥n")
                _T("制御軸 %d を初期化しました"), wAxis );
        pcpg.ResultMessage( szBuf );

}

void CPcpgDlg::OnMinus()
{
        // TODO: この位置にコントロール通知ハンドラ用のコードを追加してください

        //-----------------------------------------------------
        // 一方向に 100*m_distance　パルス出力
        //-----------------------------------------------------
        if(m_chkX.GetCheck()){
       //-----------------------------------------------------
```

```
    // 第1軸を選択
    //-----------------------------------------------------
    wAxis = PCPG46_AXIS_1;
    //-----------------------------------------------------
    if ( FALSE == pcpg.Pcpg46wDataFullWrite(
          wBsn, wAxis, PCPG46_MINUS_PRESET_PULSE_DRIVE, 1*m_distance ) ) {
          pcpg.ErrorMessage(wBsn);
          return ;
    }

 }
 if(m_chkY.GetCheck()){
//-----------------------------------------------------
    // 第 2 軸を選択
    //-----------------------------------------------------
    wAxis = PCPG46_AXIS_2;
    //-----------------------------------------------------
    if ( FALSE == pcpg.Pcpg46wDataFullWrite(
          wBsn, wAxis, PCPG46_MINUS_PRESET_PULSE_DRIVE, 1*m_distance ) ) {
          pcpg.ErrorMessage(wBsn);
          return ;
    }

 }
 if(m_chkZ.GetCheck()){
//-----------------------------------------------------
    // 第 3 軸を選択
    //-----------------------------------------------------
    wAxis = PCPG46_AXIS_3;
    //-----------------------------------------------------
    if ( FALSE == pcpg.Pcpg46wDataFullWrite(
          wBsn, wAxis, PCPG46_MINUS_PRESET_PULSE_DRIVE, 1*m_distance ) ) {
          pcpg.ErrorMessage(wBsn);
          return ;
    }
```

```
        }

}

void CPcpgDlg::OnPlus()
{


            //-------------------------------------------------
        // ＋方向に m_distance パルス出力
        //-------------------------------------------------

        if(m_chkX.GetCheck()){
       //-------------------------------------------------
            // 第1軸を選択
            //-------------------------------------------------
            wAxis = PCPG46_AXIS_1;
            //-------------------------------------------------
            if ( FALSE == pcpg.Pcpg46wDataFullWrite(
                    wBsn, wAxis, PCPG46_PLUS_PRESET_PULSE_DRIVE, m_distance ) ) {
                    pcpg.ErrorMessage(wBsn);
                    return ;
            }

        }
        if(m_chkY.GetCheck()){
       //-------------------------------------------------
            // 第 2 軸を選択
            //-------------------------------------------------
            wAxis = PCPG46_AXIS_2;
            //-------------------------------------------------
            if ( FALSE == pcpg.Pcpg46wDataFullWrite(
                    wBsn, wAxis, PCPG46_PLUS_PRESET_PULSE_DRIVE, m_distance ) ) {
                    pcpg.ErrorMessage(wBsn);
                    return ;
            }
```

```
        }
        if(m_chkZ.GetCheck()){
        //-------------------------------------------------
            // 第 3 軸を選択
            //-------------------------------------------------
            wAxis = PCPG46_AXIS_3;
            //-------------------------------------------------
            if ( FALSE == pcpg.Pcpg46wDataFullWrite(
                    wBsn, wAxis, PCPG46_PLUS_PRESET_PULSE_DRIVE, m_distance ) ) {
                    pcpg.ErrorMessage(wBsn);
                    return ;
            }

        }

}

void CPcpgDlg::OnPcpgclose()
{


                //-------------------------------------------------
        // デバイスを解放する
        //-------------------------------------------------
        if ( FALSE == pcpg.Pcpg46wClose(wBsn)){
                pcpg.ErrorMessage(wBsn);
                return ;
        }
        wsprintf( szBuf,
                _T("PCPG46 Close¥n")
                _T("Bsn %d"), wBsn );
        pcpg.ResultMessage( szBuf );
        //-------------------------------------------------
        // Pcpg46.Dll を閉じる
        //-------------------------------------------------
```

```
	if ( FALSE == pcpg.Pcpg46wDllClose()) {
		pcpg.ErrorMessage(PCPG46_BSN_AUTO);
		return ;                                                    // DLL のアンロー
ドに失敗しました
	}
	return ;

}

void CPcpgDlg::OnChangeEditDistance()
{
	// TODO: これが RICHEDIT コントロールの場合、コントロールは、 lParam マスク
	// 内での論理和の ENM_CHANGE フラグ付きで CRichEditCrtl().SetEventMask()
	// メッセージをコントロールへ送るために CDialog::OnInitDialog() 関数をオーバー
	// ライドしない限りこの通知を送りません。

	CEdit * str = (CEdit *) GetDlgItem(IDC_EDIT_DISTANCE);
	char count[10];
	int n;
	n=str->GetLine(0,count,10);
	count[n] = 0;
	m_distance = atoi(count);
}



void CPcpgDlg::OnChangeEditFeed()
{
	// TODO: これが RICHEDIT コントロールの場合、コントロールは、 lParam マスク
	// 内での論理和の ENM_CHANGE フラグ付きで CRichEditCrtl().SetEventMask()
	// メッセージをコントロールへ送るために CDialog::OnInitDialog() 関数をオーバー
	// ライドしない限りこの通知を送りません。

	CEdit * str = (CEdit *) GetDlgItem(IDC_EDIT_FEED);
	char count[10];
	int n;
```

```
        n=str->GetLine(0,count,10);
        count[n] = 0;
        m_feedrate = atoi(count);
        if(m_chkX.GetCheck()){
       //-------------------------------------------------------
            // 第1軸を選択
            //-------------------------------------------------------
            wAxis = PCPG46_AXIS_1;
            //-------------------------------------------------------
            if ( FALSE == pcpg.Pcpg46_Change_Feed(wBsn,wAxis,m_feedrate)){
                  pcpg.ErrorMessage(wBsn);
                  return ;
            }
            wsprintf( szBuf,
                  _T("PCPG46¥n")
                  _T("制御軸 %d のパルス発生速度を%d PPS に変換しました"),
wAxis,m_feedrate);
            pcpg.ResultMessage( szBuf );
        }else
        if(m_chkY.GetCheck()){
       //-------------------------------------------------------
            // 第2軸を選択
            //-------------------------------------------------------
            wAxis = PCPG46_AXIS_2;
            //-------------------------------------------------------
            if ( FALSE == pcpg.Pcpg46_Change_Feed(wBsn,wAxis,m_feedrate)){
                  pcpg.ErrorMessage(wBsn);
                  return ;
            }
            wsprintf( szBuf,
                  _T("PCPG46¥n")
                  _T("制御軸 %d の送り速度を%d mm/min に変換しました"),
wAxis,m_feedrate);
            pcpg.ResultMessage( szBuf );
        }else
        if(m_chkY.GetCheck()){
```

```
        //-----------------------------------------------------
            // 第 3 軸を選択
            //-----------------------------------------------------
            wAxis = PCPG46_AXIS_3;
            //-----------------------------------------------------
            if ( FALSE == pcpg.Pcpg46_Change_Feed(wBsn,wAxis,m_feedrate)){
                pcpg.ErrorMessage(wBsn);
                return ;
            }
            wsprintf( szBuf,
                _T("PCPG46¥n")
                _T("制御軸　%d の送り速度を%d　mm/min に変換しました"),
wAxis,m_feedrate);
            pcpg.ResultMessage( szBuf );
        }else
        {
            wsprintf( szBuf,
                _T("PCPG46¥n")
                _T("制御軸選択されません"));
            pcpg.ResultMessage( szBuf );
            return;
        }

}



void CPcpgDlg::OnXjiku()
{

    A=1;
}

void CPcpgDlg::OnYjiku()
{
    A=2;
```

```
}

void CPcpgDlg::OnZjiku()
{
    A=3;
}
```