
E g P o s t (イージーポスト)

取扱説明書

改訂 15 版 2008/6/25

EgPost Rev4.25 用に改訂

発行：著作権

(株)セイロジャパン・システム技術部

前書き

EgPost(本ソフト)は、STATION、CimatronEで作成された加工データを各種工作機械で利用可能にするためのデータ変換ソフトウェアであり、本ソフトにより現在市場に出ている主要工作機械(フライス・マシニングセンター)へのNCプログラム作成が可能となります。

また、本ソフトを利用することにより、機械や利用者ごとに異なるNCプログラムへの出力が1つのソフトで可能となり、随時NCプログラムの記述方法が変更になった場合でもユーザサイドで変更が自由に行えるという特徴を持っています。

このような特徴を持った EgPost について、その利用方法およびNCデータの定義方法に関して本書は説明していきます。

尚、本ソフトウェアは動作環境として、MicrosoftのWindows2000、X Pを対象としています。そのため、Windows2000、X Pの基本的な操作が可能なことを前提として、説明を行っておりますので、未だ不慣れな方は別途市販の参考書等で予習しておかれることをお勧めします。

注意事項

EgPostが対象とする工作機械は、動作軸が3軸以内のものであり、回転軸等のインデックス割付には対応していません。

また、4軸ワイヤー、旋盤での利用には不向きですのであらかじめご注意ください。

目次

第1章 インストール	1
1.1 動作環境	1
1.2 インストール方法	1
1.3 システムの構成	3
第2章 操作方法	4
2.1 NCデータ作成までの流れ	4
2.2 NCデータ作成方法	6
2.3 ログファイルについて	14
2.4 EGP ファイルについて	14
第3章 EGPファイルの定義方法	15
3.1 EgPostについて	15
3.1.1 <i>EgPost</i> でできること、できないこと	15
3.1.2 <i>GPP1</i> ベースと <i>GPP2</i> ベース	16
3.2 EgPost編集ソフトの操作方法	17
3.2.1 ファイルの読み込み&保存	17
3.2.2 文字の入力と編集	19
3.2.3 セクションの切り替え	20
3.2.4 変数の入力方法	21
3.2.5 表示フォントの指定	21
3.2.6 表示色の指定	22
3.3 NCプログラムの記述	23
3.3.1 セクション	23
3.3.2 セクションの記述	24
3.3.3 変数の利用	25
3.3.4 モーダルと非モーダル	26
3.3.5 モーダルブロックの利用	28
3.3.6 ユーザ変数の利用1	29
3.3.7 ユーザ変数の利用2	30
3.3.8 条件変数の使い方	31
3.3.9 条件構文の使い方	32
3.3.10 コメントの付記	33
3.4 NCプログラムの書式(フォーマット)	34
3.4.1 数値変数のフォーマット指定	34
3.4.2 変数のモーダル設定	37
3.4.3 <i>G90</i> / <i>G91</i> / <i>G91</i> の指定	38
3.4.4 <i>G</i> 、 <i>M</i> コードの指定	40
3.4.5 固定サイクルコードの指定	40
3.4.6 ユーザ変数1の指定	41
3.4.7 ユーザ変数2の指定	42
3.4.8 ファイル分割の指定	43
3.4.9 工程表出力の指定	47
3.4.10 工程表出力のセクション	48

第4章 記述例(Tutorial)	49
4.1 例ー1 (G 9 0 基本形)	49
4.1.1 はじめの一步	49
4.1.2 工具を動かす	50
4.1.3 クーラントとスピンドル	51
4.2 例ー2 (G 9 0、工具交換あり)	52
4.2.1 はじめの一步	52
4.2.2 工具交換	52
4.2.3 クーラントとスピンドル	53
4.3 例ー3 (G 9 1 工具交換なし)	54
4.3.1 EGPファイルの設定	54
4.3.2 各手続きを安全共通点から開始する例	55
4.3.3 各手続きを手続き開始点から始める例	56
4.3.4 工具を動かす	56
4.4 例ー4 (G 9 2)	57
4.5 例ー5 (手続きごとのファイル分割)	58
4.5.1 どこで、ファイルを切るのか？	58
4.5.2 ファイル分割の記述	59
4.6 例ー6 (条件文の記述)	60
4.6.1 R指令円弧補間の記述	60
4.6.2 条件文について	61
4.7 使用可能変数一覧	62
4.8 座標変数について	70
第5章 補足	71
5.1 のポストと同じ名前で、NCプログラムを出力するには？(その1)	71
5.2 のポストと同じ名前で、NCプログラムを出力するには？(その2)	72
5.3 テープ長さ/加工長さで出力ファイルを分割するときの注意点	73
5.4 テープ長さでファイルを分割するには？	73
5.5 工具毎にファイルを分割するには？	74
5.6 設定内容を、テキストファイルとして保存するには？	75
5.7 工具中心座標で、NCプログラムを作成するには？	75
5.8 シーケンス番号を付けるには？	76
5.9 円弧近似をRで出すには？	76
5.10 MODE()関数の応用例	77
5.11 G 9 2 モードの考え方	78
5.12 N U R B S 補間の入力方法	79
5.13 計算式の入力あれこれ	80
5.14 工程表作成時の注意	81
第6章 レジストリエントリ	82
第7章 終わりに	84

第1章 インストール

§ 1.1 動作環境

本ソフトウェアは、その動作に関して以下の環境を前提としています。それ以外の環境に関しては、巻末の問い合わせ先までお問い合わせください。

コンピュータ： DOS / V
OS： Windows 2000, XP
メモリ： 128 Mバイト 以上
ハードディスク： 最低200～300 Mバイトの空き領域を有すること
STATION： バージョン9以上
CimatronE： バージョン5以上
(STATION、CimatronEはどちらか1つで可)

STATION(以下 St.)、CimatronE(以下CimE)のモジュール構成に関しては、NCプログラムが作成可能な環境を有することが最低条件となります。

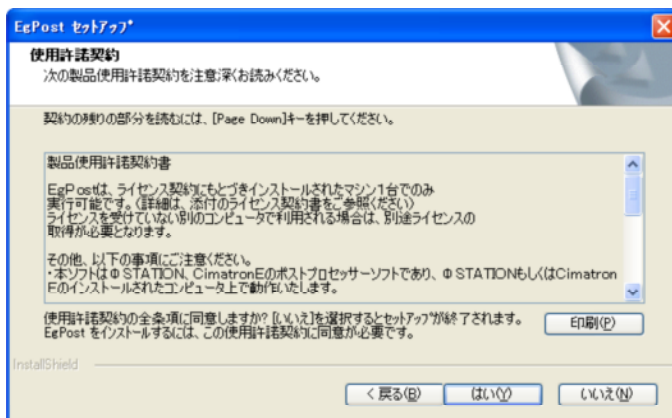
§ 1.2 インストール方法

以下の通り、インストールを行います。

1、EgPost SetupDisk をフロッピーに挿入し、a:\%setup.exe を実行します。

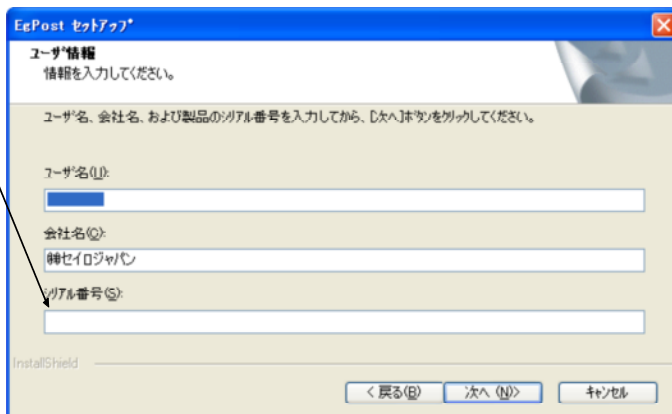
2、インストールが開始されますので、指示に従って操作を行ってください。

3、使用許諾契約書が表示されたら内容を読んだ上で[はい]を押してください。[はい]を押さないで先へは進めません。また[はい]を押したと云うことは使用許諾契約書の内容に同意したと云うことになります。

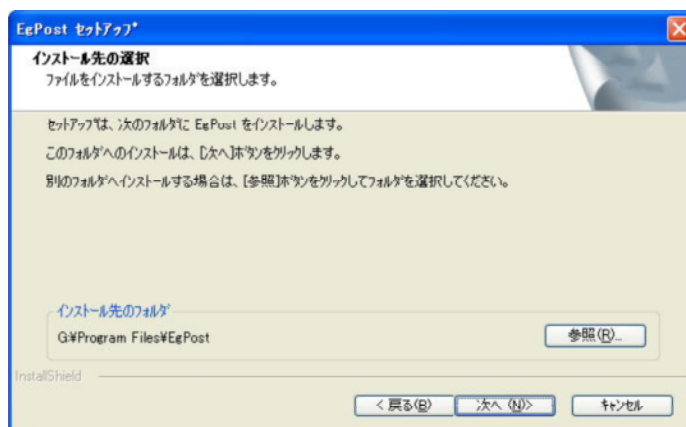


4、次に、以下のような画面が表示されます。

ここで、添付してある Serial 番号 を間違えずに入力してください。



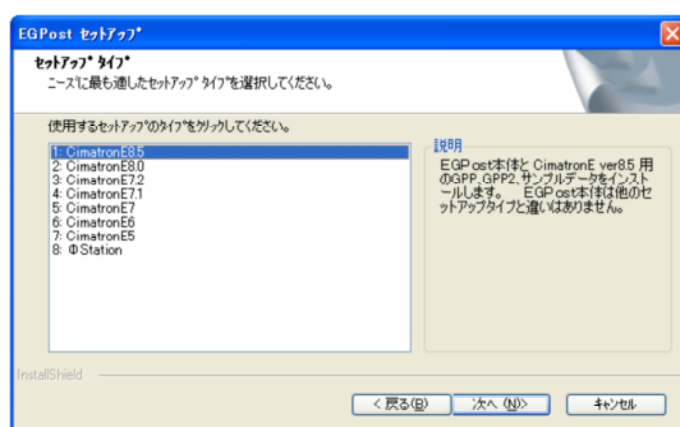
5、EGPostエディタのインストール先を選択します。インストール先は初期値が表示されますが、変更したい場合は[参照]ボタンを押すとフォルダ選択画面が表示されますので、フォルダを選択してください



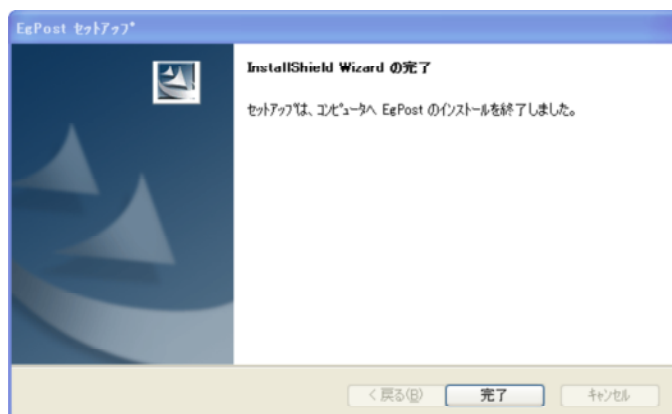
6、EGPostを動作させるCAD/CAMシステムを選択します。

[次へ]を押すとインストールを開始します。

CimatronE ver8.0とver8.5では従来のGPPをベースにしたシステムと、GPP2をベースにしたシステムの両方をインストールします



7、右の画面が表示されればインストールは完了です。[完了]ボタンを押してインストール画面を閉じてください。



注意

事前に St.またはCimEがインストールされていないと正しくインストールされません
必ず St.またはCimEを先にインストールしてください

§ 1.3 システムの構成

インストールを行うと以下のシステムがディスク上にコピーされます。

システム名	ファイル	機能
変換ソフト	EGPP.exe	St.またはCimEの中間ファイルをNCプログラムに変換します。
編集ソフト	EgPost.exe EgPost.ini	EGPファイルを作成、編集します。
St.またはCimE のポスト	EgPost.dex EgPost.def EgPost.cmd EgPost.bat	St.またはCimEに中間ファイルを作成させて、変換ソフトを起動させます。 標準のポストです。
	EgPstn.dex EgPstn.def EgPstn.cmd EgPstn.bat	Nurbsに対応したポストです。
	EGPost.df2 EGPost.dx2	CimE8.0用のGPP2ポストです。
	EGPost_gpp2.df2 EGPost_gpp2.dx2	CimE8.5用のGPP2ポストです。
サンプルデータ	サブフォルダEGPData 内のファイル	EGPファイルのサンプル

上記のシステム全体を総称して「本ソフト」と呼びます。

第2章 操作方法

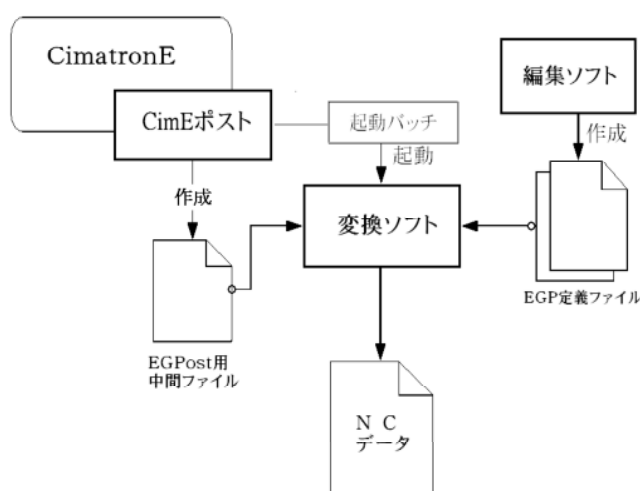
§ 2.1 NCデータ作成までの流れ

まず、本節では具体的な操作に入る前に E g P o s t 動作の全体像について説明することになります。全体像を確認することで、今何をすればいいのかわかりやすくなることでしょう。

A G P Pベース(従来のシステム)

G P PベースのE g P o s t とCimatronEは下図のような関連性を持っています。

EGPost 動作関連図



簡単に解説すると、以下のような動作手順で作業が進められていきます。

C A Mシステム(S T A T I O N、C i m a t r o n E)でポストプロセッサー
(E G P O S T、E G P S T N) を実行

これにより、C A Mシステム上のデータが中間コードで出力されます。

E G P S T NはN U R B S補間に対応したポストです。従来は標準のG P PとN U R B S
対応G P Pを切り替えるには再インストールが必要でしたが、標準とN U R B S対応を使
い分けできるようにしました。

外部プログラム E g P o s t が自動的に起動

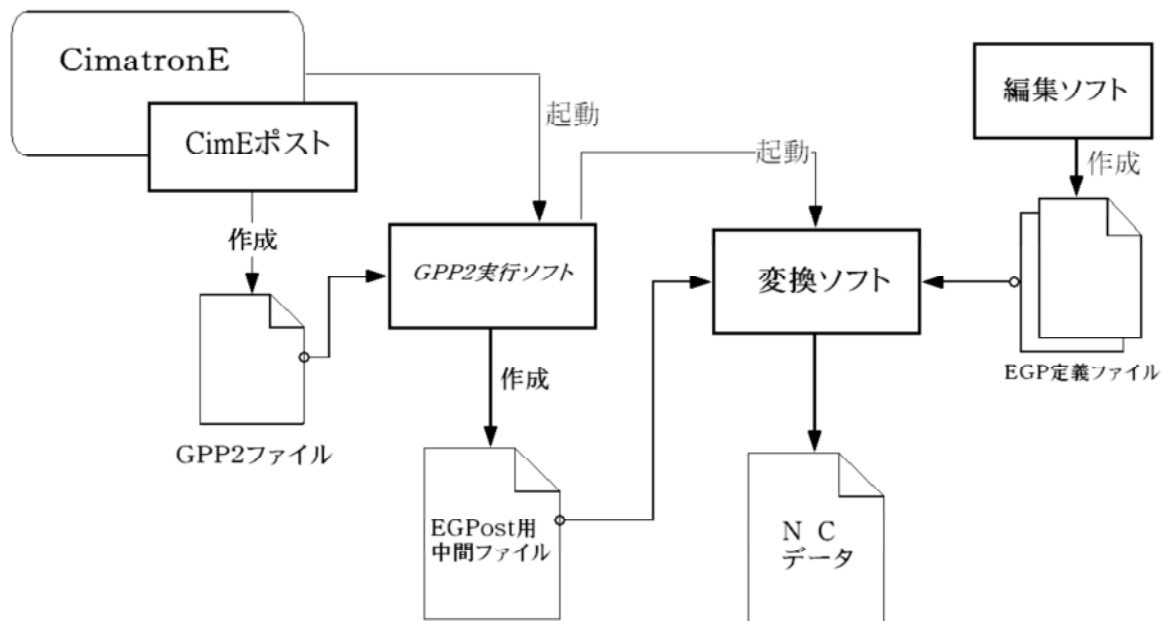
C A Mシステムのポストが終了すると、自動的に E g P o s t (変換ソフト) が起動します。

“EGP定義ファイル”を読み込み、中間コードをNCデータ(プログラム)に変換
あらかじめ作成しておいた、EGP定義ファイル(後の章で説明します)を読み込み
先ほど出力された中間コードを解釈し、具体的なNCデータ(プログラム)へと変換
していきます。

B G P P 2 ベース (R e v 4 . 2 2 より追加されたシステム)

G P P 2 ベースの E g P o s t と CimatronE は下図のような関連性を持っています。

EGPost 動作関連図



簡単に解説すると、以下のような動作手順で作業が進められていきます。

C A Mシステム (C i m a t r o n E) でポストプロセッサ (G P P 2) を実行
これにより、C A Mシステム上のデータを G P P 2 ファイルに出力し、G P P 2 実行ソフトを起動させます。

G P P 2 実行ソフト上でポストプロセッサ (E G P O S T) を実行
これにより、C A Mシステム上のデータが中間コードで出力されます。

G P P 2 ベースの E G P o s t は N U R B S 補間に対応していません。N U R B S 補間を使う場合は G P P ベースで起動させてください。

外部プログラム E g P o s t が自動的に起動
C A Mシステムのポストが終了すると、自動的に E g P o s t (変換ソフト) が起動します。

“EGP定義ファイル”を読み込み、中間コードをN Cデータ (プログラム) に変換
あらかじめ作成しておいた、EGP定義ファイル (後の章で説明します) を読み込み
先ほど出力された中間コードを解釈し、具体的なN Cデータ (プログラム) へと変換していきます。

以上の流れで、動作していきます。

§ 2.2 NCデータ作成方法

ここでは、E g P o s t を操作して、NCデータを作成するまでを具体的な例で示します。

注意

ここで紹介する例は、あくまでも1例です。

実際には、これ以外の操作も必要になることがあります。全体的な流れを理解していただければそれで結構です。

A) G P Pからの起動(従来の起動方法)

C A Mシステム(S T A T I O N、C i m a t r o n E)でポストを実行します。
ポストの種類として、EGPOST あるいはEGPSTNを選んでください。

Gコードパラメータの設定(R e v 4.1 2より追加)

EGPOST、EGPSTNを実行させる際にGコードパラメータを設定することにより
円弧を直線で近似して出力させることができるようになりました。

項目	初期値	説明
円弧を直線近似.Y/N	N	この項目を"Y"にすると円弧を直線に分割したNCプログラムを作成します。
直線近似の公差	0.005	円弧を直線に分割する際の公差を設定します

EGPost の操作については C) へ進んでください

B) GPP2からの起動

(Rev4.22より追加された起動方法、CimatronEのみにしか対応していません)

注意

GPP2から起動させる場合は、CimatronEのポスト画面からGPP2の実行環境を立ち上げ、さらにそこからEGPostを立ち上げます。従って従来の起動方法(GPPからの起動)より手順が増えることになります。

1 GPP2の起動 - CimatronE ver8.0の場合

起動させる前に

CimatronE本体のメニューの[ツール]、[環境設定]を選び、[環境設定エディタ]を開いてください。[環境設定エディタ]の左側の「一般」、「一般NC」を選ぶと右側に一般NCの項目が表示されますのでその中の「ポストファイル用のエディタ」にGPP2実行ファイル(GPP2.EXE)のパス名を設定してください
(欄の右側のフォルダボタンを押してファイルを選択してください。)

この操作はCimatronEのインストール直後に一度行えば、それ以降は操作を行う必要はありません。

CAMシステム(CimatronE)で

ポストを実行します。

ポストプロセッサ名の"GPP2"を選んでください。

ファイル拡張子を"GPP2"から変更しないでください。

"実行後に出力ファイル表示"に必ずチェックを入れてください。



機械原点パラメータ

ポストプロセッサ名で"GPP2"を選択すると座標系についての設定ができるようになります

a) 座標系リスト

座標系のリストです。"出力ファイル"欄の数値は出力ファイルでの各座標系の名前(番号)です。

b) ゼロ位置のセットアップ

機械原点の位置を設定します。(通常は設定する必要はありません)

c) GPP2参照座標系タイプ

NCプログラムを通して基準となる座標系を設定します

・座標系セットアップ

MACSYSを基準とします

・最初の手続きの座標系

最初の手続きの座標系を基準とします

・座標系選択

基準となる座標系を選択します

座標系のリストを表示し選択できるようになります。

座標系リスト	出力ファイル名
MODEL(A)	1
UCS12.1(R)	2
UCS13.1(D)	3

ゼロ位置のセットアップ*

NCセットアップに関連した機械座標ゼロ

X: 0
Y: 0
Z: 0

GPP2参照座標系タイプ:

座標系セットアップ*

目標フォルダ: E:\Cimatron\CimatronE8.0\Data\NcData

ファイル名変更: なし

ファイル名: NewPostFile

ファイル拡張子: GPP2

☒ 実行直に出力ファイル表示 ☐ 自動ポスト処理を複数手続きとしてポスト処理



をクリックするとGPP2が起動します

2 GPP2の起動 - CmatronE ver8.5の場合

CAMシステム(CimatronE)で
ポストを実行します。
ポストプロセッサ名の"egpost_gpp2"を
選んでください。

参照座標系

NCプログラムを通して基準となる座標系を
設定します

・座標系セットアップ

MACSYSを基準とします

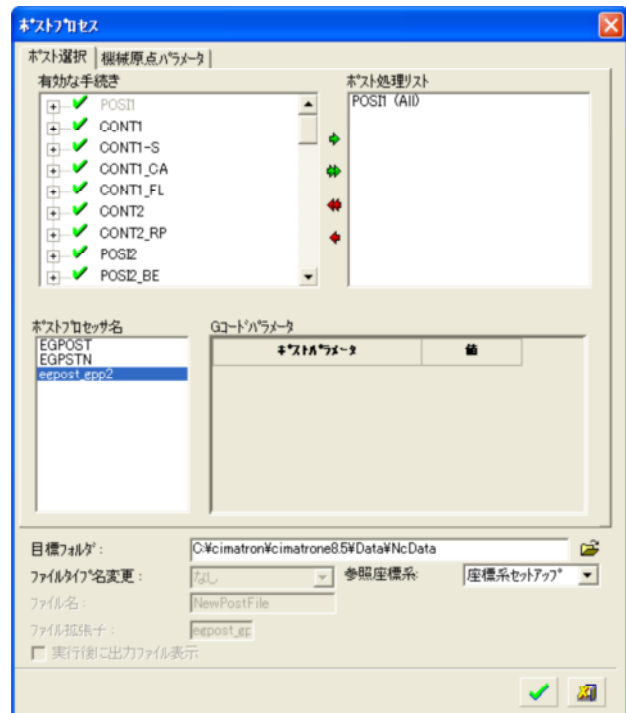
・最初の手続きの座標系

最初の手続きの座標系を基準とします

・座標系選択

基準となる座標系を選択します

座標系選択を選択すると項目の下に座
標系名が表示されます。その座標系名をク
リックするとリストが表示され選択できるよ
うになります。



機械原点パラメータ

ポストプロセッサ名で"GPP2"を選択すると座標系についての設定ができるようになります

a) 座標系リスト

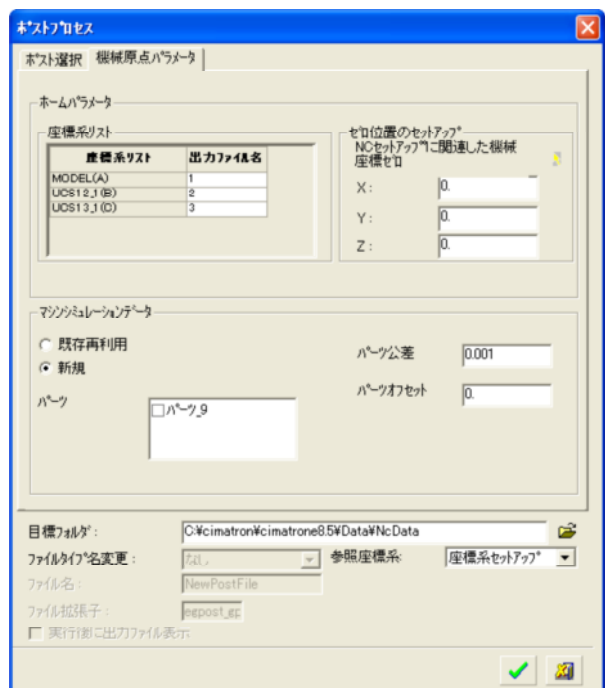
座標系のリストです。"出力ファイル"欄の
数値は出力ファイルでの各座標系の名
前(番号)です。


b) ゼロ位置のセットアップ

機械原点の位置を設定します。(通常
は設定する必要はありません)

c) マシンシミュレーションデータ

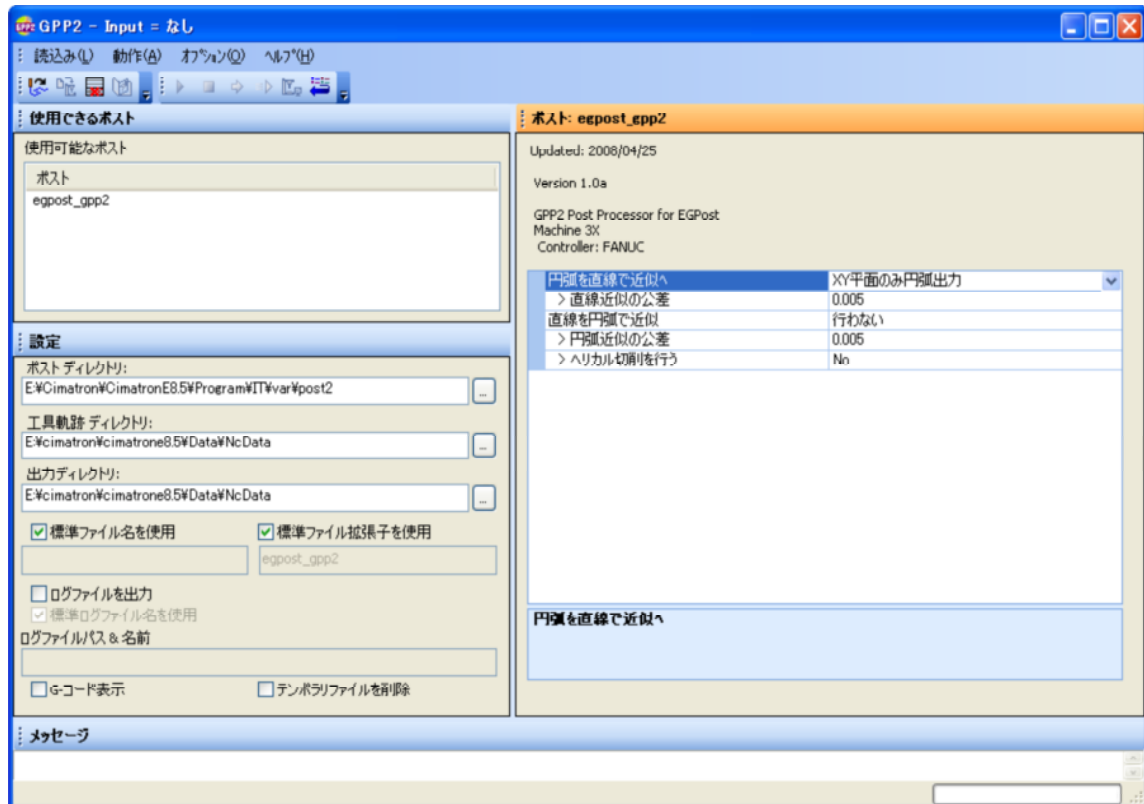
EGPostはマシンシミュレーションに対応し
ていません



 をクリックするとGPP2が起動します

3 GPP2の操作(CimatronEの各バージョン共通です)

以下のようなGPP2の実行環境が立ち上がりましたら、以下の説明に従って操作してください



a) .GPP2ファイル

実行画面の最上部、"GPP2 - Input = "に続いて e)で作成したGPP2ファイルのファイル名が表示されていることを確認してください

ファイル名が"なし"になっている場合はメニューの"読み込み(L)"、"工具軌跡(I)"からGPP2ファイルを選択してください

GPP2ファイルとはCimatronEがGPP2にデータを渡すための中間ファイルです
CimatronEからGPP2を起動させたときに作成されます。

b) ポストを選択

"使用可能なポスト(Available posts)"欄からポストを選択します

"egpost"(あるいは"egpost_gpp2")をダブルクリックしてください。ポストが選択されると右の"ポスト:"に"egpost"と表示されます

c) 設定 GPP2を実行する際の設定内容です。

全てを操作する必要はありません、必要な設定のみ変更してください。

イ) ポストフォルダ (Post Directory:)

ポストがあるフォルダを指定します。

"使用可能なポスト(Available posts)"欄に"egpost"あるいは"egpost_gpp2"がない場合は、
ここで"EGPost.dx2"、"EGPost.df2"ファイル、あるいは"EGPost_gpp2.dx2"、"EGPost_gpp2.df2"
ファイルがあるフォルダを選択してください。

ロ) 軌跡フォルダ (Toolpath Directory:)

.GPP2 ファイルがあるフォルダを指定します

ハ) 出力フォルダ (Output Directory:)

EGPost用の中間ファイルを作成するフォルダを指定します。

通常はロ)の軌跡フォルダと同じフォルダを指定してください。

ニ) 標準ファイル名を使用 (Use Standard File name)

標準のファイル名(で表示されているGPP2ファイルと同じファイル名)を使うかどうかを指定します
チェックをはずすと下の入力欄が有効になりますのでファイル名を入力します
この設定はチェックを入れた状態で実行させてください。

ホ) 標準ファイル拡張子を使用 (Use Standard File Extension)

標準のファイル拡張子(で選択したポスト名)を使うかどうかを指定します。

チェックをはずすと下の入力欄が有効になりますのでファイル拡張子を入力します。

この設定はチェックを入れた状態で実行させてください。

ヘ) ログファイルを出力 (Output Log File)

ログファイルを出力するかどうかを指定します

チェックを入れると"標準ログファイル名を使用(Use Standard File Name)"が有効になります

"標準ログファイル名を使用"にチェックが入っているとログファイル名に標準のファイル名

(NCプログラムのファイル名+".log")が使われます

"標準ログファイル名を使用"のチェックをはずすと"ログファイルパス & 名前

(Log File Path & Name)"が有効になりますのでファイル名を入力します

ト) Gコード表示 (Display G-Code)

作成したNCプログラムを表示するかどうかを指定します。

この設定はチェックを外した状態で実行させてください。

チ) 中間ファイルを削除>Delete temporary Files)

中間ファイル(.GPP2ファイル)を削除するかどうかを指定します

d) パラメータ設定

各項目の内容は次の通りです。変更する場合はその欄をマウスで指定し、新しい値を入力してください。

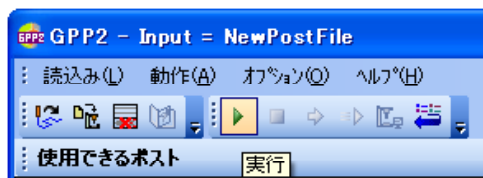
パラメータ名	初期値	詳細
円弧を直線で近似	XY平面のみ円弧出力	円弧を直線で近似する際の設定を行います 全て直線近似 : 全ての円弧を直線で近似します XY平面のみ円弧出力 : XY平面の円弧を直線で近似します 主要平面のみ円弧出力 : XY、YZ、ZX平面の円弧を直線で近似します 全て円弧出力" : 直線での近似を行いません
> 直線近似の公差	0.005	近似の際の公差を指定します
直線を円弧で近似	行わない	直線を円弧で近似する際の設定を行います CimatronEver8.5以降で使用可能です 8.0では使用できません 行わない : 近似を行いません XY平面のみ : XY平面の円弧にのみ近似します 主要平面のみ : XY、YZ、ZX平面の円弧に近似します
> 円弧近似の公差	0.005	近似の際の公差を指定します
> ヘリカル切削を行う	NO	ヘリカル切削を行うかどうかを指定します

e) 実行(EGPostを起動)

メニューの実行ボタン(画像の赤枠内)を押すEGPostが起動します



ボタンにマウスカーソルを合わせ得るとボタンが下図のように変化します

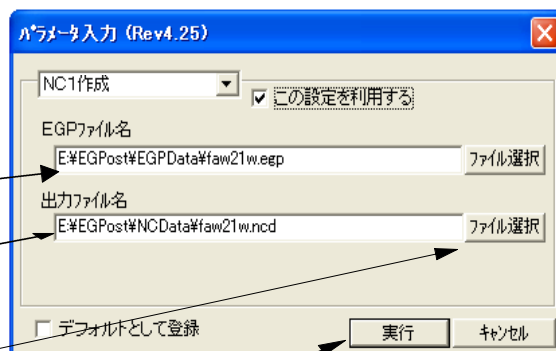


ボタンを押してEGpostを起動させます。

C) EGPPostの操作

EGPPostが起動すると、以下のようなウィンドウがCAMシステム上に現れます。

上の枠に、EGP定義ファイル名を入力します。
下の枠には、NCデータを出力するファイル名を入力します。



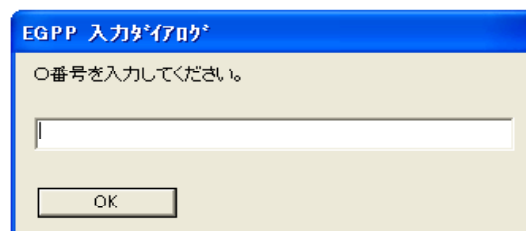
「ファイル選択」ボタンを押すとファイル選択のダイアログを開いてファイルを選択指定することもできます。

最初に表示される場面はRev3.8ではレジストリに登録してあるフォルダの内容でしたが、Rev4.03以降では入力枠に表示してあるパスの内容になります。入力枠に何もなければレジストリに登録してあるフォルダを表示します。

実行ボタンをマウスでクリックします。

セクションの記述にユーザー変数(USER1～USER10)を使用していると、以下のような入力ウィンドウが出てきます。
メッセージにしたがって、枠に入力します。

変数が「都度入力」に設定されていると(3.4.6 ユーザ変数1の指定 参照)変数が出力されるたびに右のウィンドウが表示されます。



ファイルの分割を使用する場合に、ファイル名を「都度入力」に設定していると(3.4.8 ファイル分割の指定 参照)、新しいファイルを作成するたびにファイル名入力のウィンドウが表示されますのでファイル名を入力します。

再び、CAMシステムの画面に戻ります。

以上で、操作は終了です。で指定した名前で作成されていることを確認し、エディタソフト等でNCプログラムを確認してみてください。

§ 2.3 ログファイルについて

EGPostは起動後にログファイル"EGPost.log"を作成し、実行中の情報を記録します
特に不自然な動作をした場合にはエラー、警告情報を書き込まれている可能性がありますので
このファイルを確認してみることをお勧めします

ログファイルはEGPostのインストール先フォルダに作成されます

以下はEGPostが正常に動作、終了した場合のログファイルの内容の一例です

```
EGPP Ver.4.25 (2008/6/25)
Copyright (C) 1999-2008 SAEILO JAPAN Inc.

認証OK
>> NC1作成 の処理を始めます。
XMLファイル読込OK。
EGP   File: C:\Program Files\EGPost\EGPData\aw21w.egp
Out   File: C:\Program Files\EGPost\NCData\aw21w.ncd
Execution Successfully Finished.
```

§ 2.4 EGP ファイルについて

EGPファイルとは、St.、CimEで作成された中間コードをNCデータに変換する方法を記述したもので、このファイルに従ってSt.、CimEから出力された中間コードを実際に利用するNCプログラムへ変換するために用いられます。尚、このファイルは、次の章で扱う編集ソフトを用いて編集が可能であり、編集ソフトを用いてさまざまな形に編集・作成することで自由な形のNCプログラムを作成することが可能になります。

3章以降では、このEGPファイルの作成方法を具体的に紹介していきます。

第3章 EGPファイルの定義方法

本章では、

EgPost で出来ること出来ないこと (§3.1) を明らかにした上で、EGPファイルを作成していく一通りの流れ (§3.2) を説明します。その後、具体的なEGPファイルの記述方法 (§3.3, §3.4) について紹介しています。

初めて本書を読まれる方 (初めて、EgPostを利用してNCプログラムを作成する方) は、必ず §2.1, §2.2 を読んで基本的な考え方・操作方法を理解してください。

その後、必要に応じて §3.3, §3.4 を読んでいただければ理解が深まることと思います。

さらに、次の4章で具体的な例を紹介していますから、記述方法等で困ったときに一読していただければよいかなと思います。

§ 3.1 EgPostについて

§ 3.1.1 EgPostで出来ること、出来ないこと

具体的な説明に入る前に、本ソフトで出来ることと出来ないことをあらかじめしておきます。

本ソフトで出来ること

- ・ CimatronE から出力される NC (3 軸) データを任意形式の NC プログラムに変換
- ・ 出力座標系 (G90 / G91 / G92) の任意切り替え
- ・ 数値フォーマット (小数点あり・なし等) の切り替え
- ・ 任意の倍率をかけた数値の出力
- ・ 工具交換の指定
- ・ 工作機械 (メーカー) 毎に異なる M、G コードの切り替え
- ・ 加工長、軌跡による任意ファイル分割
- ・ 任意の形式の工程表出力

× 本ソフトでは出来ないこと

- ・ 4 軸以上の工作機械 (4 軸、5 軸...) への出力
- ・ A, B, C 軸等の任意軸を持つ機械への出力
- ・ 旋盤および、ターニングセンタへの出力
- ・ ワイヤへの出力
- ・ 出力に複雑な計算が必要なもの

本ソフトの特徴を理解された上で、あらかじめ想定された NC プログラムが本ソフトで作成可能かどうかを判断してください。

§ 3.1.2 GPP1ベースとGPP2ベース

Rev4.22より従来のGPP1ベースでの使用法に加えてGPP2ベースでの使用が可能になりました
GPP2ベースの場合GPP2でEGPostの中間ファイルを作成し、また、EGPost本体もGPP2から
起動させます。

GPP2ベースの場合GPP2の機能を利用して以下のことができるようになります

- ・基準の座標系を選ぶことができる(GPPベースでは最初の手続きの座標系に固定)
- ・直線を円弧で近似することができる
- ・GPP2が作成するXMLファイルを利用して、工程表のデータをより正確にできる

ただし、デメリットとなる点もありますので注意してください

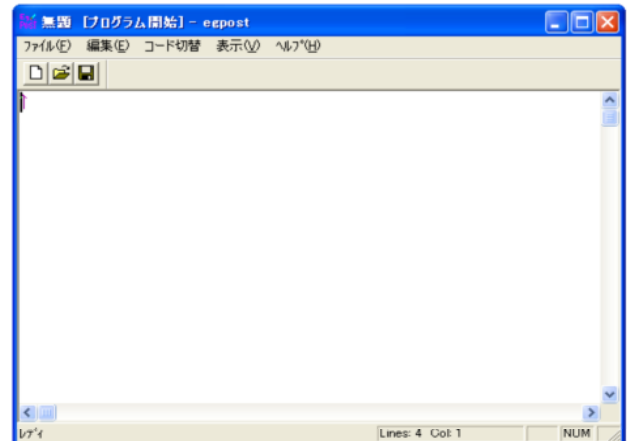
- ・ドキュメント名に日本語(全角文字)が使われているドキュメントは処理できません
- ・NURBS補間に対応できない
- ・起動時の手順が1ステップ増える

§ 3.2 EgPost編集ソフトの操作方法

本ソフトをインストールすると、次のようなアイコンがスタートメニューに表示されます。
このアイコンをマウスでクリックします。



そうすれば、以下のようなウィンドウが画面上に表示されます。
このウィンドウを用いて、様々なNCプログラムの出力形式を定義していきます。



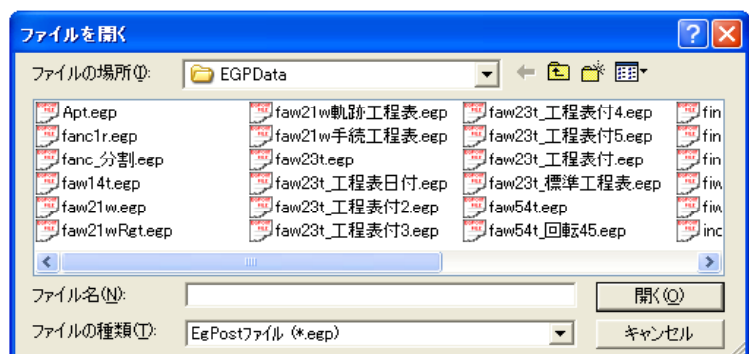
§ 3.2.1 ファイルの読み込み&保存

編集ソフトを用いた最も簡単な利用方法は、既に作ってあるファイルを読み、その意味を解釈し、必要に応じて出力形式を編集し、再びファイルとして保存することでしょう。
(例外的に、何もない状態から作成するほうが簡単なこともあります。)

そこで最初に、ファイルの読み込み、保存方法を紹介します。

ファイルの読み込み

編集ソフトのメニューから、ファイル(F) / 開く(O) を選択します。
以下のような、ファイルを開くウィンドウが表示されます。



ここで、ファイルをマウスで選択し、開くボタンをマウスでクリックすれば、ウィンドウ上になんらかの文字（NCプログラム）が表示されます。

（インストール時に、サンプルとして数種類のEGPファイルがハードディスクにコピーされているので、最初はこのファイルを利用すると良いでしょう。）

指定したファイル名は、タイトルバーに表示されます



ファイルの保存

編集ソフトで、さまざまな操作（編集、削除、）を行った後には、その操作内容が失われないように保存をする必要があります。

編集ソフトのメニューから、ファイル(F) / 上書き保存 (S) を選択します。

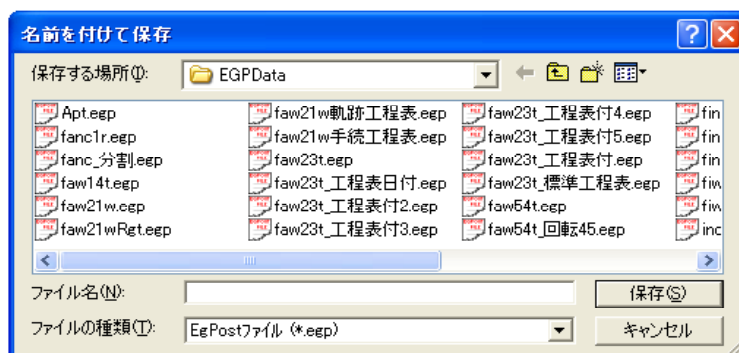
以上で、現在読み込んでいるファイルと同一の名前でEGPファイルが保存されます。

ファイルの別名保存

場合によっては、現在読み込んでいるファイルとは別の名前で保存したいことがあります。これは、以下の方法で行うことができます。

編集ソフトのメニューから、ファイル(F) / 名前を付けて保存 (A)... を選択します。

以下のようなウィンドウが表示されます。



ここで、保存したいファイル名をキーボードから入力します。

最後に、保存ボタンをマウスでクリックすれば、指定した名前で保存されます。

§ 3.2.2 文字の入力と編集

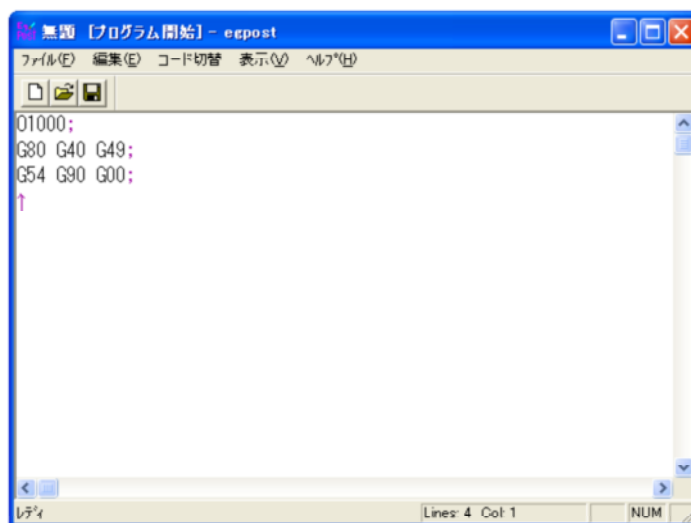
具体的な操作説明に入る前に、編集ソフトで入力した内容と最終的に出力されるNCプログラムの関連性について説明します。(より詳しい内容は、次節でも紹介します。)

編集ソフトでは、キーボードから入力された文字列がそのまま登録出来るようになっています。例えば、キーボードから、A B C と入力すれば、画面上にも A B C と表示されます。そして、最終的に出力される NC プログラム上にもそのとおりに A B C と出力されます。ですから、最終的なNCプログラムがイメージできるならば、NCプログラム通りにこの編集ソフト上に入力しておけば、入力した通りにNCプログラムに出力されます。

たとえば、欲しいNCデータに以下のような記述があれば、この編集ソフトでも同様に以下の通り入力します。

```
O1000  
G80 G40 G49  
G54 G90 G00
```

その様子を右の画面に表示します。



《凡例》

入力した文字以外に、行末に ;、最後の行に が表示されています。
これは、それぞれ改行と、文字列(セクション)の終わりを意味していて、NCプログラムとしては出力されません。
; はENTERキーを押すと自動で入力されます。

文字を入力する以外の操作としては、以下のような処理が可能となっています。

編集場所の移動	カーソルキー () または マウスで文字の近くをクリック
1文字削除	Delete キー (カーソルの後の文字) または Back Space キー (カーソルの前の文字)
文字のコピー	Shift + カーソルキー で範囲を選択し、メニューの 編集(E) / コピー(C) をマウスでクリック
文字の貼りつけ	メニューから 編集(E) / 貼りつけ(P) をマウスでクリック

§ 3.2.3 セクションの切り替え

Cimatron E 及びEGPostでは、NCプログラムをその動作の性格によって分割し、それらの部品を以下のように分類して考えています。

	セクション名	概 要
1	プログラム開始	各種の初期化、座標系の設定等を行う
2	プログラム終了	クーラント停止、スピンドル停止等のプログラム終了時の処理
3	工程開始	1つの工具動作が始まる前の処理
4	工程終了	1つの工具動作が終了するときの処理
5	工具交換	工具交換時の動作
6	X Y 早送り	工具が早送り (G 0 0) で動作
7	Z 早送り	工具がZ 移動を伴い早送り (G 0 0) で動作
8	直線送り	工具が直線送り (G 0 1) で動作
9	円弧送り	工具が円弧送り (G 0 2 , G 0 3) で動作
10	NURBS補間	工具がNURBS補間で動作
11	NURBS補間終了	NURBS補間を終了するときの動作
12	サイクル	工具が固定サイクルで動作
13	サイクルオフ	固定サイクルが終了するときの動作
14	アプローチ	径補正オンとなる直線送り動作
15	リトラクト	径補正オフとなる直線送り動作
16	メッセージ	メッセージを記述
17	分割ファイル先頭	NC ファイルが分割されたときの先頭部分
18	分割ファイル終了	NC ファイルが分割されたときの終了部分
19	工程表先頭	工程表の先頭部分
20	工程表項目	工程表の情報
21	工程表終了	工程表の終了部分

本ソフトでは、分類名をセクション名と呼びます。

19,20,21はNCプログラムとは直接的な関係がない特殊なセクションです。

EGPostでは、これらのセクション毎にコード（文字列）を記述するようになっています。
詳細は § 3.3.1 セクションを参照してください。

操作方法としては、例えば"プログラム終了"のセクションにコードを記述する場合
編集ソフトのメニューから、コード切替 / プログラム終了 をマウスでクリックします。

タイトルバーに、現在のファイル名とともに [プログラム終了] と表示され、すでにコードが
入力してあればそのコードが表示されます。

ここに記述するコード（文字列）は [プログラム終了] のコードですから、NCプログラムの
最後で出力されます。

他のセクションも同様の操作で変更することが出来ます

§ 3.2.4 変数の入力方法

変数の入力も当然キー入力で入力することもできますが、以下の方法を使う方が入力ミスがなく確実に入力できます。

コード表を表示させます。

編集ソフトの「表示」メニューの「コード一覧」を実行して「コード表」ウィンドウを表示させます。

編集画面の変数を入力したい位置に文字カーソルを合わせます。

コード表の中から入力したい変数を選択します。

変数を選択すると一覧表の下に選択した変数の説明が表示されます。

「貼り付け」ボタンを押します。

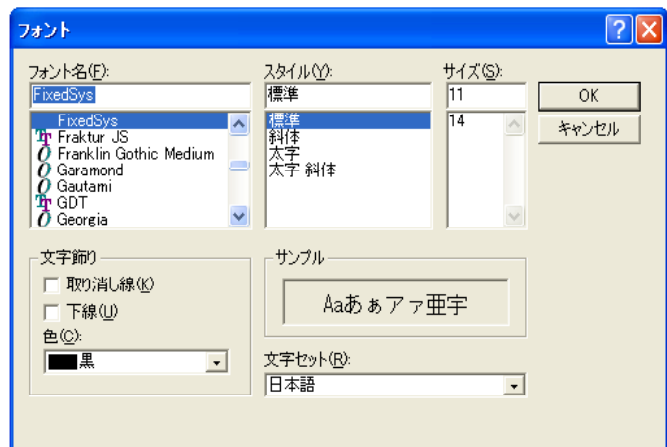
「貼り付け」ボタンを押すと編集画面に変数が入力されます。

コード表を表示させたままでも編集の作業は行えます。コード表を消したい場合は「キャンセル」ボタンを押してください。

§ 3.2.5 表示フォントの指定

最初表示される文字（フォント）は、小さくて見づらいかもしれません。編集ソフトでは、表示文字の大きさ・種類を自由に変更することが可能です。

編集ソフトのメニューから、表示(V) / フォント... をマウスでクリックします。以下のようなウィンドウが表示されます。



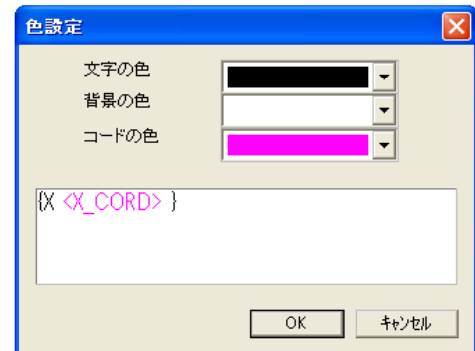
表示したいフォント名とサイズを選択し、OKボタンをクリックすれば、指定したフォントで（コード）文字列が表示されます。

§ 3.2.6 表示色の指定

編集作業になれてくると、自分の好きな色で画面を表示したいという要求が出てきます。編集ソフトでは、以下の3項目について好きな色で表示できるように設定できます。

- ・ 文字列の色
- ・ 背景の色
- ・ 変数の色

編集ソフトのメニューから、表示(V) / 色の設定 をマウスでクリックします。
以下のようなウィンドウが表示されます。



ここで、好きな色をマウスで選択して、OK ボタンをクリックすれば、編集ソフトの色が変更されます。

§ 3.3 NCプログラムの記述

本節では、前節で紹介したコードの記述方法を踏まえて、より具体的な記述方法を説明します。

§ 3.3.1 セクション

プログラムを記述していく上で、まず最初にEGPostがNCプログラムを作成する仕組みをある程度理解していただく必要があります。

まず、最初に理解していただきたいのがこの節で説明する“セクション”という考え方です。

STATION、CimatronE、EgPost では長いNCプログラムをその一つ一つの動作の性格から多くの部品に分割し、それらを16のセクションに分類しています。

同種のセクションはどの部分でも同じような動作をしますので一つのセクションに対しては一つのコードを記述することで対応できます。

従って、セクションが複数回登場しても、コードを2度以上記述する必要はありません。

座標値等の細かな違いは変数で吸収します。変数についての細かな説明は

§ 3.3.2 セクションの記述 以降を参照してください

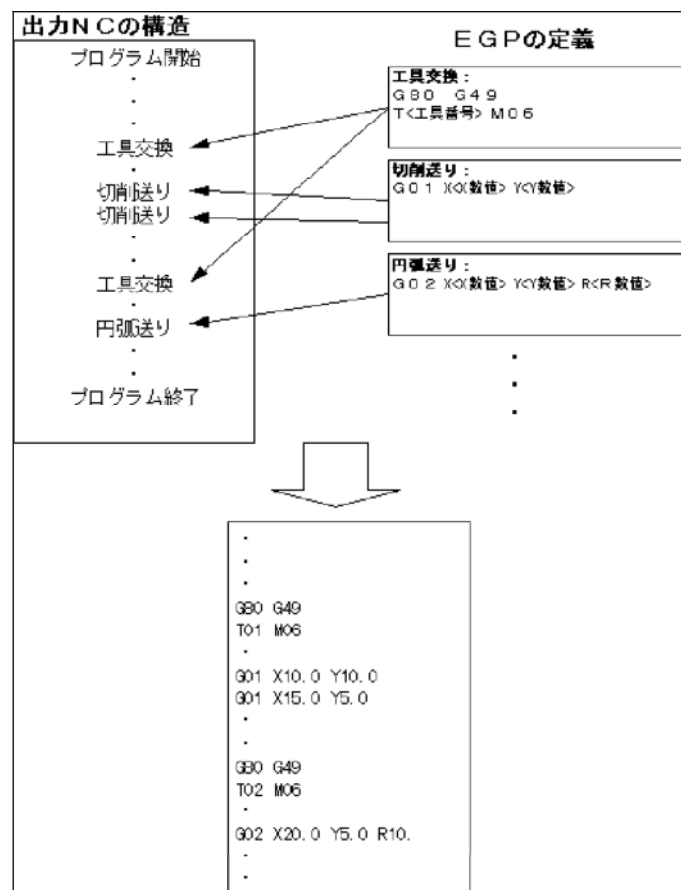
理解を深めるために以下の図を参照してください。

図のように、工具交換や、送り動作等はNCプログラムの中で何度（場合によっては何千回）も利用されるコードです。

もし、セクションと云うような考えを使用しないと、同じようなコードを何十回（何百回）も記述しなければならなくなります。

しかし、NCプログラムをセクション毎に分けておくことで、必要なときに必要なセクションが何度も利用され、（まるで積み木のように）最終的には巨大なNCデータが作成されていくようになります。

EGPostは中間ファイルに記述してあるNCプログラムの構造に従って対応するセクションを呼び出し、変数を具体的な数値、文字に変換して、ファイルにNCプログラムとして出力します。



§ 3.3.2 セクションの記述

§ 3.2.3 でも扱いましたが、再度セクション一覧を以下に記述します。

	セクション名	概 要
1	プログラム開始	各種の初期化、座標系の設定等を行う
2	プログラム終了	クーラント停止、スピンドル停止等のプログラム終了時の処理
3	工程開始	1つの工具動作が始まる前の処理
4	工程終了	1つの工具動作が終了するときの処理
5	工具交換	工具交換時の動作
6	X Y 早送り	工具が早送り (G 0 0) で動作
7	Z 早送り	工具がZ移動を伴い早送り (G 0 0) で動作
8	直線送り	工具が直線送り (G 0 1) で動作
9	円弧送り	工具が円弧送り (G 0 2 , G 0 3) で動作
10	NURBS補間	工具がNURBS補間で動作
11	NURBS補間終了	NURBS補間を終了するときの動作
12	サイクル	工具が固定サイクルで動作
13	サイクルオフ	固定サイクルが終了するときの動作
14	アプローチ	径補正オンとなる直線送り動作
15	リトラクト	径補正オフとなる直線送り動作
16	メッセージ	メッセージを記述
17	分割ファイル先頭	NCファイルが分割されたときの先頭部分
18	分割ファイル終了	NCファイルが分割されたときの終了部分
19	工程表先頭	工程表の先頭部分
20	工程表項目	工程表の情報
21	工程表終了	工程表の終了部分

セクションの種類に基づいて、必要な部分に必要なコード（文字列）を記述していきます。

（もちろん必要のないセクションには何も記入しなくても結構です。）

例として、 プログラム開始、 プログラム終了には以下のような文字を記入することになるかと思います。

プログラム開始

```
%
O1000
G80 G40 G49
G54 G90 G00
```

プログラム終了

```
M05
M30
%
```

以下のセクションも同様に記述していくことで、最終的には完成されたNCプログラムを作成することが可能になります。

各セクションの具体的な内容については各NCプログラムを運転させる機械、またユーザーそれぞれの仕様に合わせて作成してください。

尚、 ~ のセクションは、特殊なセクションです。このセクションの扱いについては §3.4.9以降で説明します。また、サンプルデータ" faw23t_工程表付.egp"も参考にしてください。

§ 3.3.3 変数の利用

この節では、変数の利用について説明を行います。

変数とは？ 字のごとく常に変化する数値のことを意味します。

(コンピューターの世界では文字も数値で管理していますので、文章(文字列)も変数として扱うことができます。本マニュアルでは文章を扱う変数を文字変数、数値を扱う変数を数値変数と呼びます。)

たとえば、NCプログラムを作成するときいつも同じところを加工するわけではありません。

加工するもの毎にNCプログラムに出てくるX座標やY座標は異なります。

この常に変化する数値や文字を、変数を利用してNCプログラムの記述を行います。

本ソフトで利用できる変数は、巻末の§4.5に一覧を記載していますが、このすべてを利用しなければならないわけではなく、この中で必要なものだけを利用すればよいのです。

たとえば、直線送りの動作をする場合は、X_CORD, Y_CORD, Z_CORD などの変数が利用できます。このX_CORD, Y_CORD, Z_CORD は、現在の工具の座標値を出力するための変数です。

ですから、一般的なNCプログラムでは、以下のように記述できます。

```
G01 X<X_CORD> Y<Y_CORD> Z<Z_CORD>
```

ちなみに、<X_CORD> のように<>で括られた文字が変数として扱われ、都度異なる数字(または文字)が出力されます。

もうひとつ例を示しましょう。

これは、円弧送りのときの記述例です。

```
<GMOVE> X<X_CORD> Y<Y_CORD> R<R_CORD>
```

<X_CORD>, <Y_CORD>, <Z_CORD> までは今までの説明でOKだと思いますが、<GMOVE> がなぜ必要なのでしょう？

それは、円弧送りには、右周りと左周りの2種類があり、それは出力してみるまでどちらの動作になるかわからないからです。

このような動作指令(G00, G01, G02, G03) は<GMOVE>で代用でき、そのときの動作によって異なる文字が出力されます。

これは、数字ではなく文字が出力される変数の例です。

注意事項

最後に気を付けないといけないことを1つ説明しましょう。

それは、前述のセクションと変数には大きな関係が存在するということです。

たとえば、プログラム開始セクションで、<GMOVE>などの動作指令を利用しようとしてもそれはできません。なぜならば、プログラム開始セクションはプログラムの先頭に行う各種の初期化処理を行うための場所とされているため、後続にくる様々な動作までは面倒をみてくれないからです。

同じように、工具交換で<GMOVE>を利用しても同じことが言えるでしょう。

その逆に、特定のセクションで数値(文字)が定義された変数については、その後のセクションにおいても利用可能で、たとえば、プログラム終了セクションのようにすべての動作の最後にくるようなセクションでは、すべての変数が利用可能となります。

そのように、変数は必要なところでしか利用できませんから、気を付けてください。

§ 3.3.4 モーダルと非モーダル

モーダルと非モーダルに関して説明を行いますが、非常に聞きなれない言葉です。又言葉の意味を説明すると余計に意味がつかみにくくなってしまいますので、ここでは具体的な例を挙げて説明を行いたいと思います。

たとえば、直線送りで以下のように定義したとします。

<GMOVE> X<X_CORD> Y<Y_CORD>

そして、出力されたNCデータが以下ようになったとします。

G01 X10.00 Y20.00

G01 X10.00 Y21.00

G01 X10.00 Y22.00

G01 X10.00 Y23.00

これでもNCプログラムとしては正しく動作するはずです。

しかし、熟練したNCプログラム作成者は、このようなコードを忌み嫌うことでしょう。

なぜならば、どの行にも G01 と X10.00 のコードが出力されているからです。

このような場合、一般的には以下のように出力すると、とてもすっきりします。

G01 X10.00 Y20.00

Y21.00

Y22.00

Y23.00

このように、変化のない数字(文字)は出力しないようにすれば、NCプログラム(ファイル)の大きさを小さくできるばかりか、読みやすくなります。

このような出力を可能とするのが、変数のモーダル・非モーダルという設定です。

変数のモーダル・非モーダルの切り替え方法は、後述の§ 3.4.2 で扱います。

基本的に、X_CORD, Y_CORD, Z_CORD の変数はモーダル変数になっています。

ですから、以前出力された数値と同じ数値を各変数が保持している場合は、その変数は出力として何も出力しないという設定になります。

すなわち、<GMOVE> X<X_CORD> Y<Y_CORD> という記述をしておけば、前述のような長たらしい出力ではなく、後述のようなすっきりした出力がなされます。

このようなモーダル・非モーダルを利用することでとてもすっきりしたNCプログラムの記述を行うことができますが、反面モーダルを使いすぎると必要な変数が出力されなくなる可能性も出てきますので、注意が必要です。

モーダル変数の強制出力

モーダル変数は、その状態によって出力されたりされなかったりするのですが、記述上どうしてもここは出力しなければならないという個所が出てきます。このような場合は、強制出力という方法を取れます。

記述方法は以下の通りです。

X!<X_CORD> Y!<Y_CORD>

このように<>の前に!マークを付けることで、モーダル変数が出力すべき状態にない場合でも強制的に出力します。

モーダル状態の強制変更

モーダル状態とは？

これまでに説明したように、モーダル変数は値が変更されると出力されるようになり、また一旦出力されると、次に値が更新されるまで出力されなくなります。このように変数が出力されるのか、されないのかの状態をモーダル状態と呼びます。

また、モーダル変数が出力される状態にあることを**モーダル状態がON**、モーダル変数が出力されない状態にあるとき**モーダル状態がOFF**と表現します。

EGPostでは変数のモーダル状態を強制的に変更することができます。

モーダル状態をONにする

&<*****>

モーダル状態をOFFにする

\$<*****>

編集画面に上記のように記述するとモーダル状態が変更されます。この場合&\$に続く変数はNCプログラムには出力されません。

またこの動作は変数の出力と同様に他の変数で制御することもできます。

例

```
{ <SPINDIR>  
&<SPINSPEED> }
```

<SPINDIR>が出力されると<SPINSPEED>のモーダル状態がONになり、次回<SPINSPEED>が呼び出されたときには<SPINSPEED>の値が出力されます。

§ 3.3.5 モーダルブロックの利用

§3.3.4のモーダルの説明には、実は嘘の部分があります。

実際に出力されるNCプログラムは、以下のようになってしまいます。

```
G01 X10.00 Y20.00
      X Y21.00
      X Y22.00
      X Y23.00
```

このNCプログラムを機械にかけると、たちどころにエラーが発生することでしょう。

モーダルにしてあるのになぜXの文字だけが出力されたのでしょうか？

それは、モーダルは変数のみに適用されるため、Xという文字には適用されないからです。

しかし、変数のみにモーダルが適用されても、意味がありませんから、変数のモーダルに合わせて、周囲の文字もモーダルにするという方法を紹介します。

具体的には、直線送り等のセクションに、以下のようなコードを記述します。

```
<GMOVE>{X<X_CORD>}{Y<Y_CORD>}
```

{ } で括られた部分は、1つの変数のように出力され、{ }内の変数のモーダル状態に応じて出力されたり、されなかったりします。

この場合は、<X_CORD>のモーダル状態に応じてXの文字が、<Y_CORD>のモーダル状態に応じてYの文字が出力されたり、されなかったりします。

その結果、以下のような出力が得られます。

```
G01 X10.00 Y20.00
      Y21.00
      Y22.00
      Y23.00
```

注記

{ }内の変数は、普通の変数のように扱われ1度出力されると、変数の数値が変化するまでその変数は、出力されない変数となります。

{ }内の2つめ以降の変数は、1つめの変数のモーダル状態に応じて出力が制御されますが、その変数自身が持つモーダル状態によって出力されたり、されなかったりします。

§ 3.3.6 ユーザ変数 1 の利用

変数には、モーダル・非モーダル以外に特殊な動作をする変数が存在します。

例えばO番号やG 5 4 , G 5 5 などの座標設定はNCプログラム作成時に動かす機械によって異なり、一意に決められないコードがどうしても出てきます。

これらの部分を、作成時にユーザが随時入力することによりNCプログラムとして出力させるような変数のことを、ユーザ変数 1 といいます。

具体的には、以下のような変数が利用できます。

<USER_01>, <USER_02>, <USER_03>, <USER_04>, <USER_05>

<USER_06>, <USER_07>, <USER_08>, <USER_09>, <USER_10>

の10個がユーザ変数 1 として利用可能です。

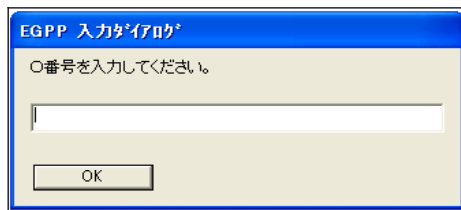
実際の利用方法は、通常の変数と同様に利用することが可能です。

記述例)

O<USER_01>

利用時のユーザ変数 1 入力

EgPost が<USER_n>を発見すると、画面上に以下のようなウィンドウが表示されます。



ユーザー変数 1 は文字変数ですので、ここでキーボード入力した文字(数字)がそのまま変数に登録され、また同時にNCプログラム内に出力されます。

§ 3.3.7 ユーザ変数 2 の利用

EgPost で利用可能なユーザ変数には、別の用途で利用するものが存在します。

たとえば、シーケンス番号のように “ 前に出力した数値に + 1 してその数値を出力する ” というような計算を必要とするものがあります。

このような計算結果を出力するための変数が、ユーザ変数 2 であり、以下の20個の変数が利用可能です。

<USER_C01>, <USER_C02>, <USER_C03>, <USER_C04>, <USER_C05>
<USER_C06>, <USER_C07>, <USER_C08>, <USER_C09>, <USER_C10>
<USER_C11>, <USER_C12>, <USER_C13>, <USER_C14>, <USER_C15>
<USER_C16>, <USER_C17>, <USER_C18>, <USER_C19>, <USER_C20>

当然のことながら、計算式は EgPost 上に任意の計算式を定義することが可能です。

たとえば、EgPost 上に <USER_C01> = <USER_C01> +10; といった計算式を登録しておけば、ユーザ変数を出力するごとに <USER_C01> は、10 だけ数値がプラスされますから結果として 10、20、30、・・・といった数字が出力されることになります。

ユーザー変数 2 は数値変数です。具体的に NC プログラムに出力されるときの書式はフォーマット設定された内容によります。(「§ 3.4.1 数値変数のフォーマット指定」参照)

計算式の登録方法は、§ 3.4.7 で紹介します。

§ 3.3.8 条件変数の使い方

コード表の先頭に表示される 5 つのコード(_CYCNREGIT、_CYCREGIT、_NTLCPRC、_TL~LAST、_TLFAST、_TLLAST、_TLNEXT)は条件変数です。

これらの変数はONかOFFかの2つの値(状態)を持ち、その値によってNCプログラムへの出力を制御することができます。

たとえば、以下のように使用します。(サンプルファイル faw14t.egp を参照)
(「工具交換」セクション)

```
[<_TLNEXT>M09;  
M05;  
]
```

_TLNEXT は「2 番目以降の工具交換時にON」になる変数です。

従ってこの場合、このセクションが最初の工具交換の場合は条件変数<_TLNEXT>はOFFになり、[] 中の条件変数以降の部分

```
M09;  
M05;
```

は出力されません。このセクションが2 番目以降の工具交換の場合は条件変数<_TLNEXT>がONになり、[] 中の条件変数以降の部分は出力されます。

[] の中には変数を使用することも出来ます。ただし {} を使用することはできません。

条件変数の内容は以下の通りです。

_CYCNREGIT	リジットタップを使用しない時にON
_CYCREGIT	リジットタップ使用する時にON
_NTLCPRC	工具交換のない手続き時にON
_TL~LAST	最終の工具交換時にOFF
_TLFAST	最初の工具交換時にON
_TLLAST	最終の工具交換時にON
_TLNEXT	2 番目以降の工具交換時にON

§ 3.3.9 条件構文の使い方

"§ 3.3.8 条件変数の使い方"で説明した変数出力の外にも条件構文を使うとより細かくNCプログラムの出力を制御することができます。

条件構文では以下の5つの構文変数を使います。

<IF>
<THEN>
<ELSEIF>
<ELSE>
<ENDIF>

条件変数の解説

<IF>: 条件構文の先頭に必ずつけ、条件構文の開始を指示します。

これは条件構文には必ず必要です。

<THEN>: 直前の<IF>(あるいは<ELSEIF>)変数の後に記述された条件文が成立する場合はこれ以降のNC文を出力します。これは条件構文には必ず必要です。

<ELSE>: 直前の<IF>(あるいは<ELSEIF>)変数の後に記述された条件文が成立しない場合はこれ以降のNC文を出力します。これは条件構文には必ずしも必要ではありません。

<ENDIF>: 条件構文の最後に必ずつけ、条件構文の終了を指示します。

これは条件構文には必ず必要です。

<ELSEIF>: この変数は<ELSE>と<IF>を合わせたような機能を持っています。

直前の<IF>(あるいは<ELSEIF>)変数の後に記述された条件文が成立しない場合に、さらにまた別の条件でNC文を制御したい場合に使います。

これは条件構文には必ずしも必要ではありません。

<ELSEIF>変数を使った場合と、同じ構文を<ELSEIF>を使わずに記述した場合の比較
以下の左右の構文は同じ意味になります。

<IF>	<IF>
<THEN>	<THEN>
<ELSEIF>	<ELSE>
<THEN>	<IF>
<ELSE>	<THEN>
<ENDIF>	<ELSE>
	<ENDIF>
	<ENDIF>

条件構文については "§ 4.6 例－6(条件構文の記述)"で具体例を挙げてより詳しく説明しています。

§ 3.3.10 コメントの付記

EgPost 3.3 からは、NC プログラム内にコメントを記述することが可能です。
コメント文字とは、出力するNC プログラム中には出力したくないが、プログラム記述上
付記しておいた方がよい場合に利用します。
たとえば、以下のように利用します。

G80G90G00 ...機械の初期化

この場合、NC プログラムに出力したいのは G80G90G00 ですが、ここで何をしたのか書
いておけば後々プログラムがわかりやすくなります。
このように、...で続く文字はコメント文字となります。
コメント文字は深緑色で表示されます。

コメントの記述方法は以下の通りです。

...コメント文字列	改行付きのコメント文字列です。
...,コメント文字列	改行無しのコメント文字列です。

§ 3.4 NCプログラムの書式（フォーマット）

§ 3.4 では、§ 3.3 までで記述してきたNCプログラムの書式を変更するための設定方法を紹介しします。ここで書式とは、以下に示すような内容を指します。

しかし、必ずしも設定が必要とは限りません。（設定の必要ないEGPファイルを定義する方が多いことでしょう）

（本節で扱う書式編集の内容）

- ・ G 9 1 出力のNCプログラムを出力したい。
- ・ 座標出力の形式として、少数点なしの形式にしたい。
- ・ §3.3.3 で説明した変数のモーダル・非モーダルを切り替えたい。
- ・ §3.3.5-6で説明したユーザ変数の設定を行いたい。
- ・ 固定サイクル等の特殊なGコード、Mコードを指定したいとき。
- ・ ファイルを加工長さで自動分割したい時、あるいは分割したファイル名を自動的に指定するとき

これらの項目について、以下の節で具体的な操作方法を交えて紹介します。

§ 3.4.1 数値変数のフォーマット指定

a) 基本操作

座標値の基本フォーマットは、少数点付き（少数点以下3桁）で出力されます。
また、工具番号等の整数数値に関しては、標準で少数点なし数値で出力されます。
これらの出力フォーマットを変更する場合、以下のように行います。

編集ソフトのメニューから、編集・NC書式編集をマウスでクリックします。
フォーマットと書かれたタブをマウスでクリックします。

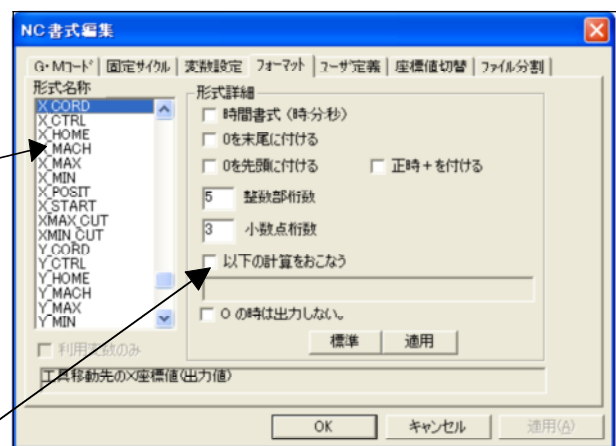
右図のようなウィンドウ表示が出てきます。

以下の手順で、設定を行います。

設定したい変数名を、選択します。

選択すると、現在この変数に定義されているフォーマットが表示されます。

「利用変数のみ」をチェックすると
現在使用されている（編集画面に記述されている）変数のみが表示されます。



想定する書式に各項目を設定します。

各項目の意味は、以下の通りです。

時間書式（時分秒）

数値を秒単位として、時間書式（時:分:秒）で出力するように設定します。

時間書式をONにすると以下の書式設定は無効になります。

例) 10000 を時間書式にしたとき

2:46:40

0 を末尾に付ける

出力しようとする変数の小数点桁数が、指定桁数に足りないときは0を付加します。

例) 5.12 を小数点3桁で0末尾をONにしたとき

5.120

0 を先頭に付ける

出力しようとする変数の整数部桁数が、指定桁数に足りないときは先頭に0を付加します。

例) 4.32 を整数部5桁で0先頭をONにしたとき

00004.32

整数部桁数

出力しようとする変数の整数部桁数を指定します。

表示しようとする変数の整数部桁数が、指定より大きい場合は変数に合わされます。

小数部桁数

出力しようとする変数の小数点以下の桁数を指定します。

(指定桁数+1の桁で四捨五入がされます。)

正時+を付ける

出力しようとする変数が0以上の場合、先頭に+が付加されます。

以下の計算を行う

出力する変数に、指定の計算をおこなったものを実際に出力します。

例) 1.23 を 1000倍した数値を出力する場合

*1000 を入力する。

0の時は出力しない

値が0のときは出力しません。

最後に適用ボタンを押します。

以下、 ~ の指定を繰り返し全ての変数の設定が終わったら、OKボタンを押します。

以上で、変数のフォーマット指定が出来ます。

注意

ここで設定する数値の書式は出力の形式だけでなく、それぞれの変数の精度を表すという意味も持ちます。インクリメンタルの計算、モーダル状態の設定は、書式設定された数値で行われます。

参考

NCプログラムでよく利用される変数のフォーマット指定例を紹介します。

) G02, G01のように2桁の整数で0を先頭に付ける

0先頭をON、整数部桁数2、小数点桁数0

) 小数点なしの座標値出力

小数点桁数0、以下の計算を行う*1000

b) 変数の選択方法の詳細

使用する工作機械を変更したとき等には、変数のフォーマットを大幅に変更する必要が出てきます。このようなときは変数を一つ一つ変更するのではなく、変数を複数選択し、一括してフォーマットを修正した方が間違い等が少なく確実に修正できます。

そこで変数を一括して選択する方法と、それらのフォーマットを修正する際の注意事項を説明します。(ここで説明する方法は、次の § 3.4.2 変数のモーダル設定 でも使用できます。)

選択

変数を複数同時に選択する方法を説明します。

(以下の内容は、説明だけでなく実際に操作を行って感覚的に覚えた方が理解が早いと思います)

-)ドラッグを使う方法(“ドラッグ”とはマウスのボタンを押したまま
マウスを動かす事です)

選択したい変数の上でマウスの右ボタンを押します。

ボタンを押したままカーソルを上下に動かすとカーソルが通過した変数が選択状態になります。

-)シフトキーを使う方法

変数を選択します。

シフトキーを押したまま他の変数を選択すると直前に選択した変数とその間の変数が全て選択されます。

-)コントロールキーを使う方法

コントロールキーを押しながら選択すると、それ以前に選択してある変数の選択状態は変えずに選択を行います。

また、コントロールキーを押しながら既に選択してある変数を再度選択すると、その変数の選択を解除します。)、)で一括して選択した項目の中から余分な項目の選択を解除したいときなどに使用することもできます。

フォーマットを修正する際の注意事項

項目を複数選択するとフォーマットの設定項目に入力ウィンドウが灰色に表示される項目がある場合があります。これは、その項目については、選択変数の設定が全て同じではないことを示しています。

まず、この場合は選択した変数が正しいかどうかを確認してください。

次に、修正をする際の注意点をあげます。

-)チェックボックスは灰色の状態でもON、OFFを選択することはできます。
-)桁数の入力に灰色の状態になっているときは入力できません。
-)計算式は、まず「以下の計算を行う」をチェックします。するとウィンドウが入力可能になり、選択された変数のなかで最初の変数に設定された数式が表示されます。

[適用]ボタンを押すと選択した変数全てに表示されているフォーマットの内容が設定されます。

§ 3.4.2 変数のモーダル設定

変数に対してモーダル変数の設定を行います。(全ての変数に対して設定が可能です。)

編集ソフトのメニューから、編集・NC書式編集をマウスでクリックします。
変数設定と書かれたタブをマウスでクリックします。

右図のようなウィンドウ表示が出てきます。

以下の手順で、設定を行います。

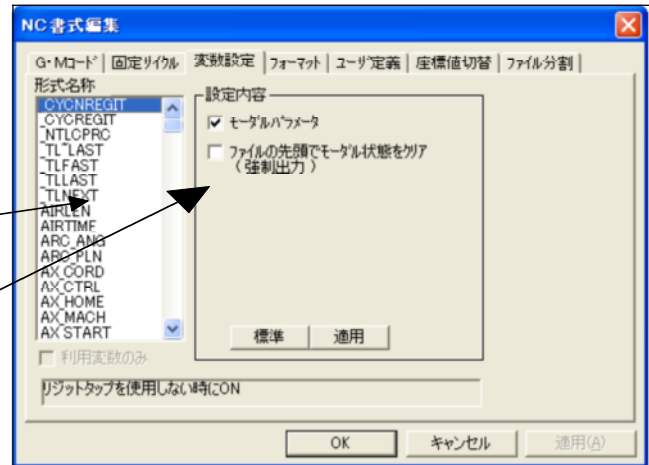
設定したい変数名を、選択します。

選択すると、現在この変数の設定状態が表示されます。

§ 3.4.1 b)で説明した選択方法を使用することもできます。

各項目を設定します。

各項目の意味は、以下の通りです。



モーダルパラメータ

この変数がモーダル変数の振る舞いを行うかどうかの指定を行います。

ファイルの先頭でモーダル状態をクリア (強制出力)

モーダル変数は値が変更されない限り出力されませんが、この設定をONにするとファイルを変更した時点で強制的にモーダル状態をONの状態にして、値が変更されていなくても変数を出力するようになります。

最後に適用ボタンを押します。

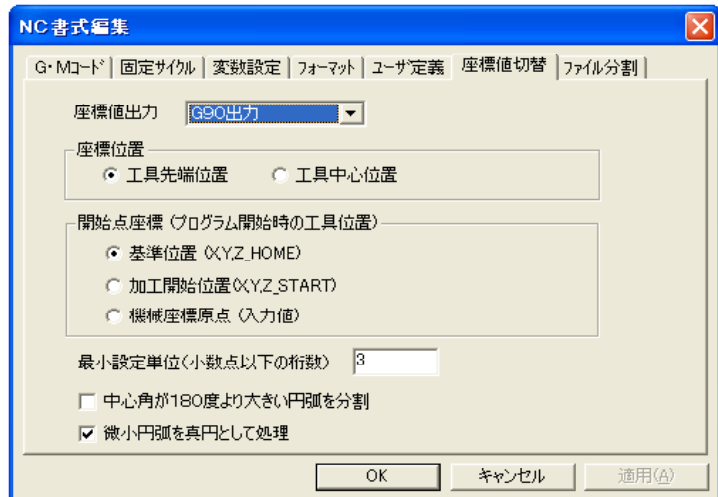
以下、～の指定を繰り返し全ての変数の設定が終わったら、OKボタンを押します。

§ 3.4.3 G 9 0 / G 9 1 / G 9 2 の指定

本ソフトでは、座標値の出力として G 9 0 を標準としています。
設定を行うことで、G 9 1 出力にも対応することが可能です。
(G 9 1 出力にした場合のプログラム記述上の注意点は、4 章の記述例で紹介します。)

G 9 0 / G 9 1 の切り替えは、以下の方法で行います。
編集ソフトのメニューから、編集 / N C 書式編集をマウスでクリックします。
座標値切り替えと書かれたタブをマウスでクリックします。

右のようなウィンドウが表示されます。



座標値出力をマウスでクリックすることにより、座標の出力を以下の通りに切り替えることが可能となります。

選択メニュー	概 要
G 9 0 出力	絶対値座標で出力します。
G 9 1 出力	開始点からの増分座標で出力します。
G 9 2 出力	G 9 2 で設定されたワーク座標で出力します。

ここで設定するのはあくまでもプログラムの初期状態です。N C プログラムの途中で G 9 0、G 9 1 が記述してあると座標値出力も切り替わります。(途中で G 9 2 へ切り替えることはできません。)

座標位置

CAMの出力値を工具先端の座標として、座標値を工具中心の座標に変換して出力するかどうかを選択します。

工具先端位置：工具先端の座標を出力します。(CAMの出力値をそのまま出力します。)

工具中心位置：工具の半径分高い位置の座標を出力します。

開始点座標

プログラムを起動させた時点での工具の位置を選択します。

基準座標：基準位置(安全共通点)を開始位置とします。

開始位置：手続き開始時の工具位置を開始位置とします。

機械座標原点：機械座標の原点位置(ポスト実行時に設定)を開始位置とします

G91出力の時はこの座標から増分位置を計算します。またG92をプログラムの冒頭で使う場合にはこの位置がG92以下の座標値に設定されます。

(G92の出力については 第4章 記述例 の § 4.4 例4 (G 9 2) を参照してください。)

- 1 開始点座標と開始位置は言葉としては同じ意味ですが、EGPost上では別の意味を持ちます。開始点座標はプログラム全体の最初の位置で、ここでの設定によって基準位置、開始位置、機械座標原点のどれかをとることになります。
開始位置は各手続きの最初の位置で、<*_START>変数の中身です。
- 2 機械座標の原点位置は STATIONではポスト起動後の対話部で、CiamtronEでは"ポストプロセス"ダイアログの"マシン原点パラメータ"で設定します。

最小設定単位(小数点以下の桁数)

最小設定単位とは小数点が付かない数値の単位です。EgPost内部で小数点が付かない座標値の計算を行うとき、座標値をこの値を単位とする値として解釈します。

通常は小数点が付かない座標値は"1"が"0.001"(千分台)を表すので小数点以下の桁数は"3"になります。よって"3"が初期値になっています。例えば"1"を"0.0001"(万分台)として処理したいときはこの値を"4"にしてください。またこの値は書式設定での座標値の精度と整合がとれる値にしてください。

中心角が180度より大きい円弧を分割

EgPost ver3.* では、どのような場合でも円弧角度が180度を超える場合には自動的に2つの円弧に自動的に分割し、2行に分けて出力していました。しかし、これはIJK指令の時は不要な処理であり、またNCプログラムが長くなってしまうためEgPost ver4.* では、円弧を分割する、しないを選択できるようにしました。

ここにチェックを入れた場合のみ中心角が180度より大きい円弧を分割します。

チェックを入れない状態でR指令の円弧補間をつかうときは、円弧補間のセクションに特殊な記述が必要になります。詳細は§4.6 例6(条件構文の記述)で説明してありますので、参照してください。

微小円弧を真円として出力

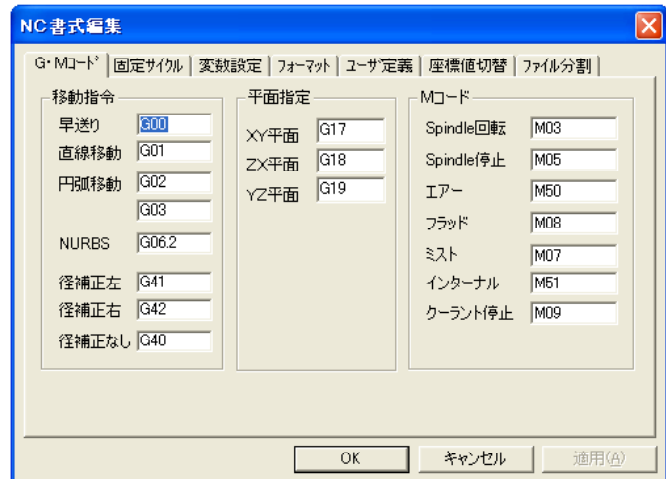
ここにチェックを入れる(ONにする)と微小円弧(中心角が0度の円弧)を真円として出力します。ONが初期値です。チェックを外すと本来真円(中心角が360度の円弧)として出力されるデータが出力されない場合がでてきます。

最後にOKをマウスでクリックします。

§ 3.4.4 G、Mコードの指定

本ソフトのGコード、MコードはFANUCの仕様に合わせてあります。
このため、他の制御機で利用する場合には一部コードの修正が必要となることがあります。
この設定（変更）方法は以下の通りです。

編集ソフトのメニューから、編集 / NC書式編集をマウスでクリックします。
G・Mコードと書かれたタブをマウスでクリックします。
右図のようなウィンドウ表示が出てきます。



NC書式編集

G・Mコード | 固定サイクル | 変数設定 | フォーマット | ユーザ定義 | 座標値切替 | ファイル分割

移動指令	平面指定	Mコード
早送り [G00]	XY平面 [G17]	Spindle回転 [M03]
直線移動 [G01]	ZX平面 [G18]	Spindle停止 [M05]
円弧移動 [G02]	YZ平面 [G19]	エア [M50]
[G03]		フラッド [M08]
NURBS [G06.2]		ミスト [M07]
径補正左 [G41]		インターナル [M51]
径補正右 [G42]		クーラント停止 [M09]
径補正なし [G40]		

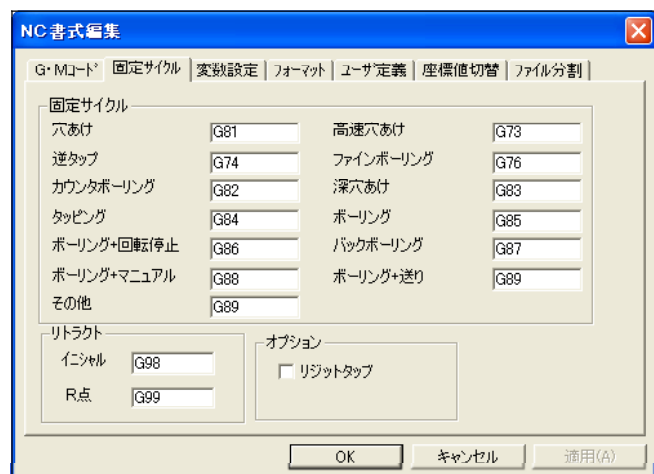
OK キャンセル 適用(A)

ここで、各項目に利用されるコードを直接キーボードから入力してください。
最後にOKをマウスでクリックします。

§ 3.4.5 固定サイクルコードの指定

本ソフトの固定サイクルコードはFANUCの仕様に合わせてあります。
このため、他の制御機で利用する場合には一部コードの修正が必要となることがあります。
この設定（変更）方法は以下の通りです。

編集ソフトのメニューから、編集 / NC書式編集をマウスでクリックします。
固定サイクルと書かれたタブをマウスでクリックします。
右図のようなウィンドウ表示が出てきます。



NC書式編集

G・Mコード | 固定サイクル | 変数設定 | フォーマット | ユーザ定義 | 座標値切替 | ファイル分割

固定サイクル	オプション
穴あけ [G81]	高速穴あけ [G73]
逆タップ [G74]	ファインボーリング [G76]
カウンタボーリング [G82]	深穴あけ [G83]
タッピング [G84]	ボーリング [G85]
ボーリング+回転停止 [G86]	バックボーリング [G87]
ボーリング+マニュアル [G88]	ボーリング+送り [G89]
その他 [G89]	

リトラクト
イニシャル [G98]
R点 [G99]

オプション
☐ リジットタップ

OK キャンセル 適用(A)

ここで、各項目に利用されるコードを直接キーボードから入力してください。
「オプション」の「リジットタップ」にチェックを入れるとタッピングサイクルの時にリジットタップを使用します。
リジットタップの設定はサイクル、サイクルオフセクションで、条件変数<_CYCREGIT>、<_CYCNREGIT>を使って記述してください。

最後にOKをマウスでクリックします。

§ 3.4.6 ユーザ変数 1 の指定

ユーザ変数はNC作成時に利用される変数で、作成時ごとに異なる数値（文字）を利用するときの変数です（詳細は、§3.3.5を参照ください）

ユーザ変数を利用するように設定すると、§ 3.3.5のようなウィンドウがNCプログラム作成時に表示されますが、この注釈文字の指定がここで行えます。

以下の通り操作を行います。

編集ソフトのメニューから、編集 / NC 書式編集をマウスでクリックします。

ユーザ定義と書かれたタブをマウスでクリックします。

以下のようなウィンドウ表示が出てきます。

形式名称から、利用する変数をマウスで選択します。
（ただし、USER_01..10）

表示メッセージをキーボード入力します。

§ 3.3.5の例では
「O番号を入力してください。」
と入力されています

都度入力が必要かどうかをマウスで選択します。

「都度入力が必要」をONにすると、この変数が利用される度に入力のウィンドウを表示します。

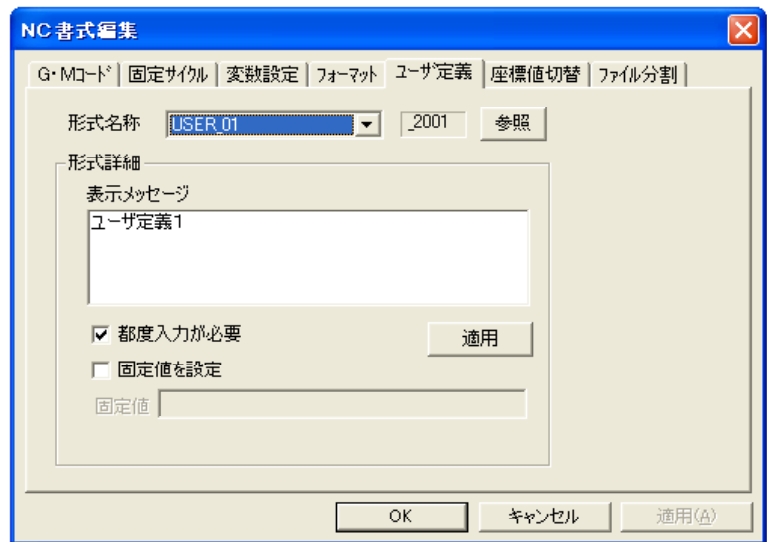
固定値を設定するかどうかを設定します。

ここにチェックを入れると、変数には必ず固定値に設定された文字列が入るようになります。

適用ボタンをマウスでクリックします。

他の変数についても、上記 ～ を繰り返し行います。

最後にOKボタンを押します。



§ 3.4.7 ユーザ変数 2 の指定

ユーザ変数 2 は任意の計算式で計算された結果を格納する変数です。
この変数への計算式登録の方法を紹介します。

以下の通り操作を行います。

編集ソフトのメニューから、編集 / NC 書式編集をマウスでクリックします。
ユーザ定義と書かれたタブをマウスでクリックします。
前ページと同じウィンドウが出てきます。

形式名称から、利用する変数をマウスで選択します。(ただし、USER_C01..C20)

計算式をキーボード入力します。
計算式の入力方法

以下の要領で計算式を作成し、それをキーボードから入力します。
) 具体的な計算式を考え、紙上等にそれを記述します。

たとえば、
<USER_C01> = <USER_C01> + 10;
といった具合です。

) 次に、< > で囲まれた変数を独自の記述で置き換えていきます。

現在のユーザー変数の内部コードは形式名称の右側に表示されています。他の変数の内部コードはその右の[参照]ボタンで一覧を表示させて確認するか、巻末の変数一覧の内部コードで確認することができます。

この内部コードを、で記述した計算式の変数名に置き換えていきます。

すなわち、<USER_C01> は内部コード _2101 ですから、計算式は以下の通りになります。

_2101 = _2101 + 10; (最後の ; は必ず入力してください)

都度計算が必要かどうかをマウスで選択します。

「都度計算を実行」を ON にすると、この変数が利用される度に計算を行います。

初期値入力が必要かどうかをマウスで選択します。

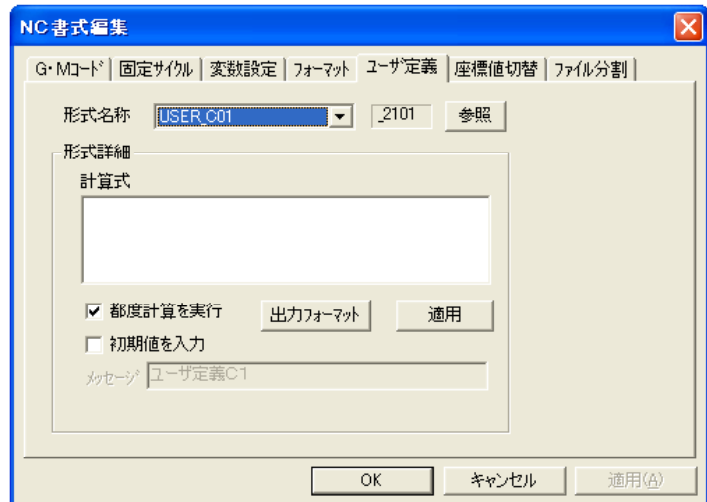
また、ON にすると「メッセージ」欄が入力可能になりますので初期値入力の入力ウィンドウに表示するメッセージを入力します。

初期値入力を ON にするとポスト実行時に初期値を入力するウィンドウを開き初期値を設定できます。OFF の場合はウィンドウは開きません。初期値は 0 となり、例の計算式では出力される数値は 10 からになります。

適用ボタンをマウスでクリックします。

[出力フォーマット] ボタンを押すとフォーマットの設定が出来ます。ユーザー変数 2 は数値変数ですので出力したい数値の書式に合わせてフォーマットを設定してください。

フォーマットの設定は「3.4.1 数値変数のフォーマット指定」の方法でも行うことができます。



他の変数を使用する場合にも、上記 ～ を繰り返し行います。

最後にOKボタンを押します。

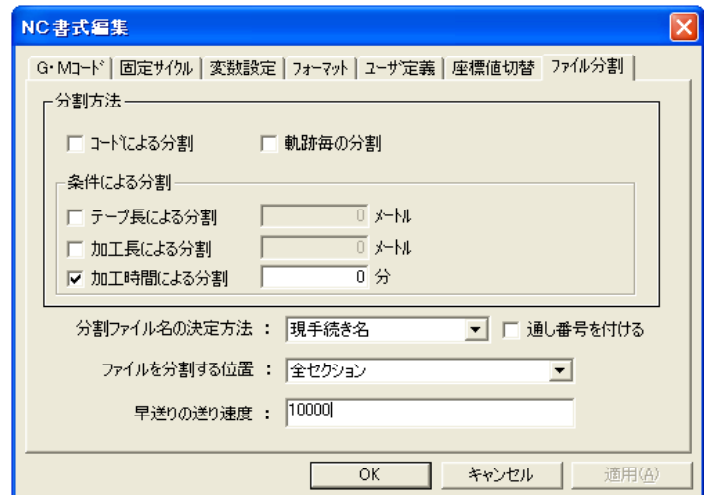
§ 3.4.8 ファイル分割の指定

本ソフトでは、作成するNCプログラムを複数のファイルに分割して出力することが可能です。この設定方法を以下に紹介します。

編集ソフトのメニューから、編集・NC書式編集をマウスでクリックします。
ファイル分割と書かれたタブをマウスでクリックします。
以下のようなウィンドウ表示が出てきます。

分割方法を以下の3つから選択します。

"条件による分割"の3つの条件は同時に設定することができます。複数の条件が設定されている場合はどれか1つでも満たせばファイルが分割されます。



コードによる分割

プログラム上に変数
<NEW_FILE>が存在すると、
ここから先は別のファイルに出

力します。<NEW_FILE>が使用されていないときはファイル分割は行いません。ファイル分割を行わないときはこの設定を選択してください。

コードによる分割と他の条件による分割を同時に設定することはできません。コードによる分割を設定すると他の条件は自動的に解除され、設定できなくなります。

条件による分割

出力したデータが以下の条件を満たすようになるとファイルを分割します。

条件は複数設定することができます。

加工長による分割

作成されたNCプログラムの加工長さが、指定値以上になると自動的にファイル分割します。

テープ長による分割

作成されたNCプログラムのテープ長さが指定値以上になると自動的にファイル分割します。

加工時間による分割

作成されたNCプログラムの加工時間が、指定値以上になると自動的にファイル分割します。

チェックを入れた条件の指定値に0が設定されている場合はポスト処理の実行時に値を設定するようになります。ポスト処理を開始した直後に値を入力するダイアログが開きますので値を入力してください。

軌跡毎の分割

工具軌跡毎に分割します。

分割ファイル名の決定方法を指定します。

- ・ 都度入力

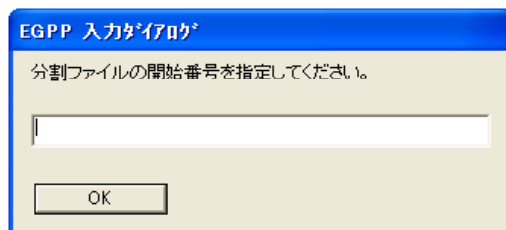
ファイル分割の必要が出たときに、都度入力ウィンドウを表示して保存ファイル名を入力するようにします。

- ・ 元ファイル + 数値

ファイル分割の必要が出たときに、最初に指定したファイル名に数値が付いた形式のファイル名を自動的に作成し保存します。

(例) 最初のファイルが、abc.ncd であった場合は abc001.ncd のような形式となります。

数値の初期値はレジストリに登録した値を使うか、実行時に指定するかを選択できます。実行時に指定するように設定してある場合は、変換ソフトの起動後、最初にファイルを分割する時に数値の初期値を入力するダイアログボックスを開きます。



詳細は「第6章 レジストリエントリ」を参照してください。

- ・ 現手続き名

ファイル分割の必要が出たときに、現在のCimEの手続き名をそのままファイル名として保存します。

- ・ 現手続きコメント

ファイル分割の必要が出たときに、現在のCimEの手続きのコメントをそのままファイル名として保存します。

- ・ 現工具軌跡名

ファイル分割の必要が出たときに、現在のCimEの工具軌跡名をそのままファイル名として保存します。

- ・ 現工具軌跡コメント

ファイル分割の必要が出たときに、現在のCimEの工具軌跡コメントをそのままファイル名として保存します。

- ・ 通し番号を付ける

現手続き名、現手続きコメント、現工具軌跡名、現工具軌跡コメントを選んだ場合にはファイル名に通し番号を付けることができます。番号の設定は元ファイル + 番号と同じです。

*番号は通し番号です、軌跡毎、手続き毎の番号ではありません。

*都度入力と元ファイル + 番号についてはこの設定は関係ありません。

ファイルを分割する位置を指定します。

加工長、テープ長、加工時間で分割の時のみに有効です。

- ・全セクション

設定した条件のどれかが指定値に達したとき無条件でファイルを分割します。

- ・早送りセクション

設定した条件のどれかが指定値に達した後で、次の早送りのセクションを出力した後にファイルを分割します。

- ・Z方向の早送りセクション

設定した条件のどれかが指定値に達した後で、次のZ方向の早送りセクションを出力した後にファイルを分割します。

早送りの送り速度を設定します。

この値は空切削時間の計算等に使用されます。

また、G01指令の動作でも送り速度がこの速度より速い場合は早送り動作と見なします。

最後にOKを押します。

§ 3.4.9 工程表出力の指定

本ソフトでは、NCプログラムと同時に工程表を出力することが可能です。
この設定方法を以下に紹介します。

編集ソフトのメニューから、編集・工程表設定をマウスでクリックします。
以下のようなウィンドウ表示が出てきます。

ここで、工程表出力のOn / Offの切り替え
工程表タイプ、出力ファイルの拡張子の指定を
行います。

工程表タイプは工程表を手続き毎に出す、工具
毎に出す、工具軌跡毎に出す、ファイル毎に出
すのいずれかを選択します。

ファイル毎の場合はファイル分割を使って
ください。ファイル分割を使わないとプロ
グラム全体についての1行だけのデータに
なります。

工程表ファイルのファイル名にはファイル名の
設定方法を以下の3通りから、また拡張子を以
下の8種類から選択できます。

ファイル名の設定方法

- ・NCプログラム名
EgPostで設定されているNCプログラム
のファイル名を使用します
- ・手続き名
中間ファイルに最初に記述されている手続きの名前を使用します。
- ・工具軌跡名
中間ファイルに最初に記述されている工具軌跡の名前を使用します。

拡張子

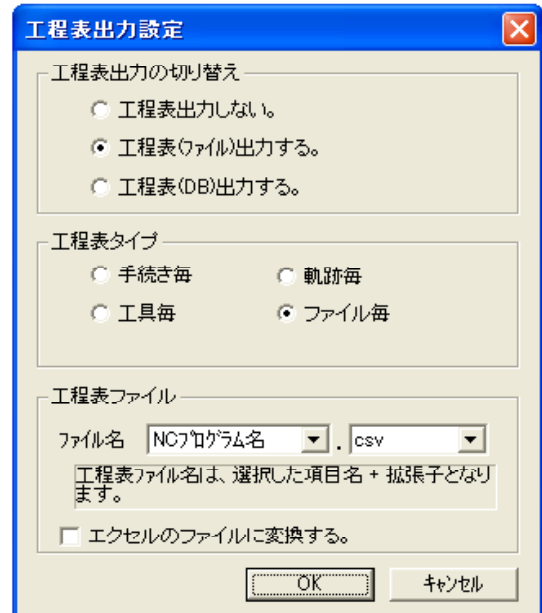
csv、log、inf、nci、kty、kti、txt、dat

工程表をエクセルファイルに変換するプログラムの実行を指定します。

「エクセルのファイルに変換する。」にチェックを入れると実行後にエクセルファイル変換プログラムを実行します。(ver 3.8まではインストール時に設定していましたがver 3.9よりEGPファイル毎に設定することができるようになりました。)

最後に、OKボタンを押せば設定は完了です。

工程表 (DB) 出力は、有償オプションです。
別冊 (工程情報DB出力マニュアル) をご参照ください。



§ 3.4.10 工程表出力のセクション

工程表の内容もNCプログラムと同様に、セクション毎に記述します。

工程表のセクションは以下の3つのセクションです。

(セクションの番号は § 3.2.3、 § 3.3.2 で紹介した全セクションの一覧での番号です。)

工程表先頭	工程表の先頭部分
工程表項目	工程表の情報
工程表終了	工程表の終了部分

) 工程表先頭

工程表の先頭部分です。ここには工程表の名前、ファイル名、工程表に出力する各項目の項目名等を記述してください。このセクションは1回しか出力されません。

) 工程表項目

工程表の具体的な情報の部分です。このセクションには工程表に出力させたい情報を変数で記述してください。モーダル変数は強制出力にしておく(!を付ける)ことをお勧めします。

このセクションは工程表タイプが手続き毎の場合は手続きの数、工程表タイプが工具毎の場合は工具交換の回数分出力されます。

) 工程表終了

工程表の終了部分を記述します。このセクションに、切削時間、切削長さ、空切削時間、テーブル長の変数を記述するとそれぞれの変数の合計を出力します。

このセクションも1回しか出力されません。

第4章 記述例(Tutorial)

第4章では、出力NCプログラムごとに記述例を紹介します。

第3章までは、基本的な記述方法については紹介してきましたが、紹介できなかったより細かな内容を示し、また具体的な例を紹介することにより、記述方法がより鮮明になるかと思えます。

また、ここに紹介した具体例はそのまま利用可能なコードですので、必要に応じてご自分で作成されるEGPファイルにそのままコピーしていただいても結構です。

§ 4.1 例1 (G90基本形)

最初に扱うNCプログラムの記述例は、以下のようなマシンを基準としています。

座標系：G90、原点指示：G54

工具交換：なし

§ 4.1.1 はじめの一步

EGPファイルを作成していく上で最も最初に記述すべきところはどこでしょうか？

どこから記述しても問題はないのですが、きっと1番簡単なところから記述していくのが賢明な方法でしょう。そして、ケースバイケースで次第に様々なコードを積み重ね、最終的な完成形を作成させていけば良いと思います。

さらに、1回のトライでは十分な出力結果が得られないかもしれませんが、あきらめずに出力結果を見ておかしい所を修正していけば、最終的には完全なEGPファイルが出来上がるはずです。

それでは、前置きはこの辺にして、具体的な第1歩を記すことにしましょう。

プログラムで誰でも分かる簡単な場所といえば、プログラムの開始と終了セクションでしょう。編集ソフトのメニューから、コード切り替え / プログラム開始 をマウスでクリックします。若干、利用される機械のそれとは異なるかもしれませんが、以下のようなコードを記述します。

```
%;  
O1000;  
G80 G49;  
G54 G90;
```

次に、編集ソフトのメニューから、コード切り替え / プログラム終了 をマウスでクリックし、プログラム終了を記述します。

```
G80G49M05;  
M30;  
%;
```

これで、プログラムの先頭と終了部分の記述は終了です。

念のために、メニューのファイル / 保存でファイルを保存しておいてください。

§ 4.1.2 工具を動かす

次に、工具を動かすコードを記述していきましょう。

工具を動かすセクションは、XY早送り、Z早送り、直線送り、円弧送りの4つです。

(固定サイクル、アプローチ、リトラクトはここではあつかいません。)

各セクションについて、以下のようなコードを記述します。

・XY早送りのセクション：

```
<GMOVE>{X<X_CORD>}{Y<Y_CORD>;
```

・Z早送りのセクション：

```
<GMOVE>{X<X_CORD>}{Y<Y_CORD>}{Z<Z_CORD>;
```

・直線送り：

```
<GMOVE>{X<X_CORD>}{Y<Y_CORD>}{Z<Z_CORD>}{F<FEED>;
```

・円弧送り：

```
<GMOVE>{X<X_CORD>}{Y<Y_CORD>}{I<I_CORD>}{J<J_CORD>}{F<FEED>;
```

これで、OKです。

ここまで出来たら、一回保存し、出来たEGPファイルを元にCimatronEからポスト出力してみてください。

かなり、目的のNCプログラムに近い形のNCデータが作成できるはずです。

§ 4.1.3 クーラントとスピンドル

E G P ファイルの最後にクーラントとスピンドル指令を記述することにしましょう。
実際の N C プログラムを直接 E G P ファイルの中に記述してもいいのですが、ここでは C i m a t r o n E で設定されたクーラント、スピンドル（回転数）に合わせた出力をするようにしましょう。

クーラントとスピンドルの指令をどこで O N / O F F させるかは、様々な考え方がありますが、最も簡単な方法をここで紹介します。

最も簡単な指定は、工程開始で O N、工程終了時に O F F させることです。

実際に、クーラントとスピンドルの指示を記述しましょう。

工程開始のセクションに移動し、以下のコードを記述します。

```
{<SPINDIR>S<SPINDLE>;  
}{<COOLANT>;  
}
```

N C 書式編集で、変数：SPINDLE をモーダル解除にします。
工程終了のセクションに移動し、以下のコードを記述します。

```
{<SPINDIR>;  
}{<COOLANT>;  
}
```

すこし変わった記述と思われるかもしれませんが、{} ブロックの中に改行文字が入っていると考えればご理解いただけるかと思います。すなわち、指定変数のモーダル状態に応じて改行する、しないを指定できるのです。例えば以下のような記述だと

```
{<SPINDIR>;  
{<COOLANT>;  
<SPINDIR>変数や<COOLANT>変数が出力されないときは改行のみが出力されて空白行が  
出来てしまいます。前述のような記述にすることによって空白行が出力されるのを防ぐことが  
できます。
```

以上で、基本的な E G P ファイルが完成です。

実際に C i m a t r o n E から、ポストを実行して結果を試してみてください。

出力結果を見て、中には物足りないなと感じられるかもしれません。

しかし、本例は N C プログラムに最低限必要なコードのみを記述してあります。

このように直したいという具体例があれば、編集ソフトを利用して E G P ファイルを編集してみてください。次回 N C プログラムを作成するときにはそのとおり出力されることでしょう。

§ 4.2 例 2 (G 9 0、工具交換あり)

本例で扱う N C プログラムの記述例は、以下のようなマシンを基準としています。

座標系 : G 9 0、原点指示 : G 5 4

工具交換 : T コードによる

尚、本例は § 4.1 の基本型を元にして説明を行いますので、あらかじめ § 4.1 の説明をご参照ください。

§ 4.2.1 はじめの一步

本例で出力される N C データは、基本的には例-1 で扱ったものに工具交換が付いた場合の例です。

ゆえに、以下のセクションには変更がありませんので、例-1 を参照の上そのまま記述してください。

- ・ プログラム開始
- ・ プログラム終了
- ・ 直線送り
- ・ 円弧送り

§ 4.2.2 工具交換

次に、工具交換のコードを付加したいと思います。

基本的に、工具交換のセクションは、手続きが変わっても工具交換が必要でない場合には呼び出されません。そのことから、工具交換が必要な時の N C プログラムだけを記述すれば O K です。

簡単な例として、以下のようなコードを記述すれば大丈夫だと思います。

```
(** TOOL CHANGE **);  
T<CURR_TOOL>;  
G90G00X<X_HOME>Y<Y_HOME>;  
G43Z<Z_HOME>H<CURR_TOOL>;
```

1 行目は、注釈です。

2 行目で工具交換を行い、3 行目で安全共通点の X , Y 座標に水平移動し、

4 行目で安全共通点まで、工具長補正を利用し移動します。

§ 4.2.3 クーラントとスピンドル

本例の最後として、クーラントとスピンドルのコードを付加していきます。

クーラントとスピンドルのコードに関しては、例-1でも述べましたが、例-1のままでは若干不具合が発生してしまいます。なぜならば、例-1ではクーラントとスピンドルの回転指令を工程の開始終了でON/OFFさせています。

しかし、工具交換の指令は工程開始セクションの後に呼び出されるため、結果としてクーラントとスピンドルがONのまま、工具交換をしてしまうようになります。

これでは、大部分の機械ではトラブルが発生してしまいます。

そこで着眼点を変えて、クーラントとスピンドルのONは工具交換後の第1動作の前に行うようにします。

工具交換後の第1動作は、基本的にXY早送りかZ早送りのいずれかのセクションとなります。そこで、以下のようなコードを記述します。

XY早送り：

```
{<SPINDIR>S<SPINDLE>;  
}{<COOLANT>;  
>GMOVE>{X<X_CORD>}{Y<Y_CORD>}
```

Z早送り：

```
{<SPINDIR>S<SPINDLE>;  
}{<COOLANT>;  
>GMOVE>{X<X_CORD>}{Y<Y_CORD>}{Z<Z_CORD>}
```

これで、どちらのセクションがきてもスピンドルと、クーラントのコードが出力されます。

変数 <SPINDLE> をモーダルOFFにしておいてください。<SPINDLE>の出力は同じ {} 内の変数<SPINDIR>によって制御されます。

次に、クーラント・スピンドルOFFは、工程終了のセクションで良いので、以下のコードを工程終了に記述します。

```
{<SPINDIR>;  
}{<COOLANT>;  
}
```

以上で、プログラム作成は終了です。

後は、数値等のフォーマットを決定してあげれば、立派なEGPファイルとして完成できます。

§ 4.3 例3 (G91 工具交換なし)

本例では、G91 (インクリメント) 座標を持つNCプログラムの作成方法について説明を行います。G91 座標のNCプログラムを記述するには、CimatronE から出力される中間コードの構造を考慮すべき必要があります。このため、必ず § 4.3.1 を参照してください。尚、説明を簡略化するために工具交換、クーラント・スピンドル指令は省略しました。

§ 4.3.1 EGPファイルの設定

G91 (インクリメント) 座標を持つNCプログラムを作成する際の注意点を説明します。EGPost ver4.0 からインクリメンタル指令の場合も比較的自由的なNCプログラムを作成できるようになりました。

§ 4.3.2、§ 4.3.3 で設定の例を提示していますが、それに拘る必要はありません。自由な記述をしても、記述の内容を或る程度判別して正常な動作を行うようなインクリメンタル座標を出力します。

プログラム開始からの工具の動き

プログラムの先頭からインクリメンタル指令で工具を動かす場合、プログラムを開始する以前の工具の位置が何処なのかを設定しておく必要があります。

基準位置 : *_HOME の位置から加工を開始します。
加工開始位置 : *_START の位置から加工を開始します。
機械座標原点 : *_MACH の位置から加工を開始します。

座標値の持ち方

G91 モード (3.4.2 参照) では、座標値変数 (*_CORD、*_HOME、*_MACH、*_START、*_CTRL) が G91 座標としての座標値を持つようになります。

また、で HOME、START、MACH を開始点にするのでは、それぞれ最初に出力される座標値が変わります。

工程開始と工程終了

G91 の場合は工程終了セクションで <*_CORD> 変数に開始点座標の値が入るようになっていましたが、これを選択できるようにしました。

「レジストリ設定」ダイアログに「インクリメンタルモードでは工程終了セクションの X,Y,Z_CORD 変数の開始点座標を入れる。」という項目を追加しました。この項目にチェックを入れると *_CORD 変数には開始点座標が入るようになります。(デフォルトではチェックが入った状態になっています。)

Rev3.8 ではインクリメンタルの場合、手続きの先頭は開始位置から計算するようになっていたので、前の手続き終了で工具を開始位置に戻しておく必要がありました。

EGPost Rev4.0 以降ではインクリメンタルの座標値は最後に出力した座標値から計算する様にしましたので、工程終了で開始位置に戻さなくとも 増分値を正確に計算できるようになりました。EGPost Rev4.0 で加工開始点に戻る動作を入れたくない場合は上記のチェックを外してください。またチェックを外した状態で、工程開始、工程終了で基準位置や加工開始点に移動させたい場合はそのような動作を記述しておかなければなりません。

§ 4.3.2 各手続きを安全共通点から開始する例

安全共通点からプログラム開始する場合は、以下のようなNCプログラムを作成します。

- ・ <X,Y,Z_HOME>を動作原点にするようにする。
- ・ 工程開始（工具交換）セクションで、<X,Y,Z_CORD>による移動を行う または、<X,Y,Z_START>による G 9 0 移動を行う。
- ・ 工程終了セクションで、<X,Y,Z_CORD>による移動を行うまたは、 <X,Y,Z_HOME>による G 9 0 移動を行う。

以下に、E G P の各セクションの記述例を示します。

（本例では、安全共通点をX=0, Y=0, Z=0として座標系設定を行います）

プログラム開始：

```
%;  
O1000;  
G80 G49;  
G92 X0 Y0 Z0;  
G91;
```

プログラム終了：

```
M30;
```

工程開始：

```
G91;  
G00 {X<X_CORD>} {Y<Y_CORD>} {Z<Z_CORD>};
```

工程終了

```
G90;  
G00 X0 Y0 Z0;
```

§ 4.3.3 各手続きを開始位置から始める例

手続き開始からプログラム開始する場合は、以下の要領でNCプログラムを作成します。

- ・ <X,Y,Z_HOME>を動作原点にするようにする。
- ・ 工程開始（工具交換）セクションで、<X,Y,Z_START>による移動を行う。
(Rev4. * より<X,Y,Z_START>変数もインクリメンタルで出力されますので、G90を使う必要はありません。)
- ・ 工程終了セクションで、<X,Y,Z_HOME>による移動を行う。
<X,Y,Z_START>は手続き毎に違う値ですので基準位置に戻し、次の手続き開始で開始位置に移動させた方が安全です。

以下に、EGPの各セクションの記述例を示します。

プログラム開始：

```
%;  
O1000;  
G80 G49;  
G92 X0 Y0 Z0;  
G91;
```

プログラム終了：

```
M30;
```

工程開始：

```
G00X<X_START>Y<Y_START>Z<Z_START>;
```

工程終了：

```
G00X<X_HOME>Y<Y_HOME>Z<Z_HOME>;
```

§ 4.3.4 工具を動かす

G91で考慮すべきコードは、§ 4.3.2 § 4.3.3 で説明した通りです。

直線送り、円弧送り等の工具動作はシステムが自動的にG91座標系で出力しますのでG90のそれと同等でOKです。

ゆえに、§ 4.1で紹介したコードをそのまま利用していただければ結構です。

以上で、G91出力のEGPファイル作成方法の説明を終わります。

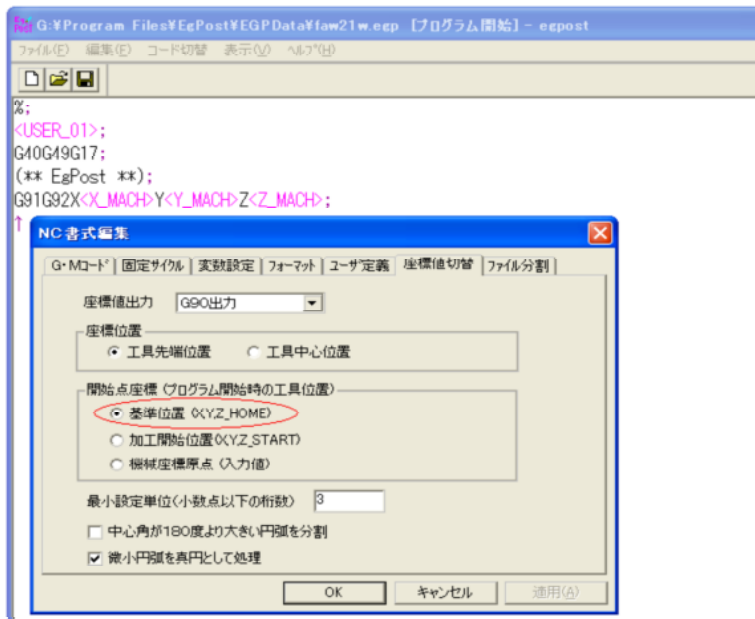
§ 4.4 例4 (G 9 2)

本例では、G 9 2 座標系のN Cプログラムを出力することを目的とします。

G 9 2 P(Pは座標値)を出力すると現在の工具先端の位置が Pの座標になるようにワーク座標系を設定します。

"G92"がN Cプログラムに記述されていないとG 9 0 モードと全く同じ座標値を出力します。

例：プログラム開始：



N Cプログラム：プログラム開始

%

O1000

G80 G49

G92 X<X_MACH> Y<Y_MACH> Z<Z_MACH>

G90

例の場合、加工開始点が基準位置に設定してあるので工具は基準位置にあるあることとなります。そのままG 9 2を指令しますので、基準位置がG92以下の座標値 *_MACH の位置になります。例えば基準位置が(0 , 0 , 1 0 0)で、機械座標原点(*_MACH)を(0 , 0 , 1 5 0)にした場合、G92以後の行ではG92を使用しない場合よりZの値が5 0 大きくなります。

加工開始点が機械座標原点の場合は機械座標原点の位置をそのままG 9 2で指令しますので座標値の変更はありません。

G92以下の座標値には数字をそのまま書き込むこともできます。(例：G92X0.Y0.Z100.)
またG92の指令に座標値変数以外の変数は使わないでください。

座標値変数とは以下の変数です

*_CORD、*_HOME、*_MACH、*_START、*_CTRL (*はX、Y、Z)

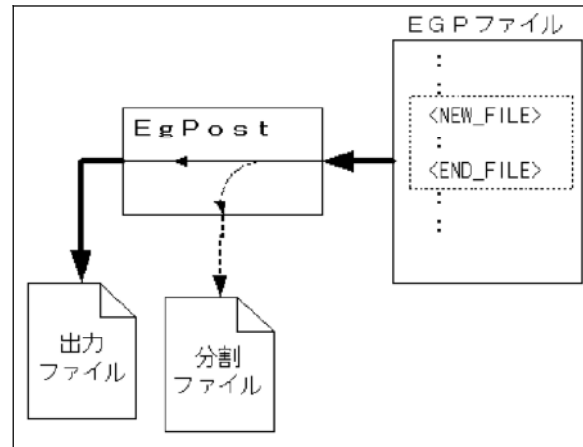
§ 4.5 例5（手続きごとのファイル分割）

記述例の最後として、ファイル分割の例を紹介します。
ここでは、§ 4.1 の基本形をファイル分割する例を示します。

§ 4.5.1 どこで、ファイルを切るのか？

EgPost では、どこでもファイルを切ることができます。
たとえば、工具交換のたびにファイルを切することもできますし、1 工程ごと別のファイルへ出力することも可能です。
それは、どのようにして行うのでしょうか？

右図のように、記述出来る変数の中でファイル分割特殊変数というものが存在します。
ここで、<NEW_FILE> 変数が存在したところから、別のファイルへ出力が渡されます。
さらに、<END_FILE> 変数が存在したところで、別ファイルへの出力を終了し、再び以前のファイルへ出力を戻します。
この特殊変数を、様々な場所に記述することで、いろいろな方法でファイル分割することができます。



たとえば、工具交換のところに記述してあれば、工具交換のたびに別ファイル出力という具合です。

さらに、<NEW_FILE> が呼ばれた場合には、必ず分割ファイル先頭セクションが呼ばれますので、分割したファイルの先頭に特別のNCデータを記述したいときには、このセクションに記述すればよいでしょう。

同様に、<END_FILE> が呼ばれた場合には、必ず分割ファイル終了セクションが呼ばれますので、分割したファイルの末尾に特別のNCデータを記述したいときには、このセクションに記述すればよいでしょう。

§ 4.5.2 ファイル分割の記述

本例では、工程毎にファイル分割するようにしましょう。

(コードによる でファイル分割するように、§3.4.8 を参照し、あらかじめ設定してください)

前節の説明で、工程毎にファイル分割する場合は、工程開始・工程終了セクションにこの特殊変数を記述すればよいと分かります。

そこで、以下のように記述していきます。

工程開始：

<NEW_FILE>;

工程終了：

<END_FILE>;

たった、これだけでOKです。

後のセクションは、§ 4.1 のそれと同等で結構です。

注意事項

ただし、<NEW_FILE> と <END_FILE> はセットで動作しますので、<NEW_FILE>だけあって<END_FILE>が無い場合とか、<END_FILE>だけあって<NEW_FILE>が無い場合の動作は保証されていませんので、あらかじめご注意ください。

§ 4.6 例6（条件構文の記述）

バージョン4.0以降、NC文の記述に条件構文を使うことができるようになりました。
条件構文について具体例を挙げて説明します。

§ 4.6.1 R指令円弧補間の記述

例は円弧補間をR指令で出力する場合の記述方法です。

円弧補間（サンプルファイルFAW14T.egp）:

```
<IF>abs(<ARC_ANG>)=360.0 <THEN><PLANE><GMOVE>{X<X_CORD>}{Y<Y_CORD>}{Z<Z_CORD>}{I<I_CORD>}{J<J_CORD>}{K<K_CORD>}{F<FEED>};  
<ELSEIF>abs(<ARC_ANG>)>180.0 <THEN><PLANE><GMOVE>{X<X_CORD>}{Y<Y_CORD>}{Z<Z_CORD>}{R<R_CORD>}{F<FEED>};  
<ELSE><PLANE><GMOVE>{X<X_CORD>}{Y<Y_CORD>}{Z<Z_CORD>}{R<R_CORD>}{F<FEED>};  
<ENDIF>
```

サンプルを構成文毎に分割して記述します。

太字の部分が条件変数、斜体の部分が条件文です。

<IF>

abs(<ARC_ANG>)=360.0 ----- 条件1

<THEN>

<PLANE><GMOVE>{X<X_CORD>}{Y<Y_CORD>}{Z<Z_CORD>}{I<I_CORD>}{J<J_CORD>}{K<K_CORD>}{F<FEED>};
----- NC文1

<ELSEIF>

abs(<ARC_ANG>)>180.0 ----- 条件2

<THEN>

<PLANE><GMOVE>{X<X_CORD>}{Y<Y_CORD>}{Z<Z_CORD>}{R<R_CORD>}{F<FEED>};
----- NC文2

<ELSE>

<PLANE><GMOVE>{X<X_CORD>}{Y<Y_CORD>}{Z<Z_CORD>}{R<R_CORD>}{F<FEED>};
----- NC文3

<ENDIF>

条件文の解説

<IF>変数、<ELSEIF>変数のすぐ後続く文が条件文です。

条件が成立する場合を"真"と云い、条件が真の場合に<THEN>以降が出力されます。

条件が成立しない場合を"偽"と云い、条件が偽の場合は<ELSE>以降が出力されます。

例では変数<ARC_ANG>変数(円弧の中心角の変数)の絶対値の大きさを検証しています。

条件1では中心角が360度かどうかを検証しています。

中心角が360度だと真円になりますのでR指令は使えません。そこで<THEN>の後にはIJK指令での円弧補間のNC文を記述してあり、<ELSEIF>以下で中心角が360度ではない場合のR指令出力が実行されるということになります。

条件2では中心角が180度より大きいかどうかを検証しています。

中心角が180度より大きいとRの値にマイナス記号(-)をつけなければなりませんのでNC文2ではRと半径の変数<R_CORD>の間にマイナス記号をつけてあります。

それ以外(中心角が180度以下)の場合にはNC文3が出力されます。

§ 4.6.2 条件文について

条件文は数値、文字列、変数、演算子、関数、定数で構成されます。

ここでは演算子、関数、定数について説明します。

なお、演算子、関数名、定数名については、大文字と小文字を区別しません。

演算子

条件文で使える演算子は次のものです。

実数の演算 (+ - * / ^)

^ はべき乗の計算を行う演算子です。

例) 2^3 は 2 の 3 乗となり、演算結果は 8 になります。

数値の比較 (= 、 < > 、 < 、 > 、 < = 、 > =)

文字列の比較 (= 、 < > 、 < 、 >)

文字列比較の < 、 > は文字が含まれる場合を検証します。

> は最初の文字列に 2 つ目の文字列が含まれるかどうか、< は 2 つ目の文字列に最初の文字列が含まれるかどうかを検証します。

例) "面輪郭加工" > "輪郭" 最初の文字列に第 2 の文字列が含まれますので結果は"真"になります。

"面輪郭加工" < "輪郭" 2 つ目の文字列に最初の文字列は含まれませんので結果は"偽"になります。

論理演算 (NOT、AND、OR、XOR)

X O R は前後の条件が片方のみ成立する場合に"真"になります。

関数

条件文で使える関数は次のものです。

関数：実数値関数

単引数の関数

三角関数：SIN、COS、TAN、ASIN、ACOS、ATAN

三角関数で使用する角度の単位は度です。

NCプログラム上では角度の単位は度ですので、三角関数を使う場合には変数をそのまま使います。

例 SIN(<ARC_ANG>)

平方根：SQRT

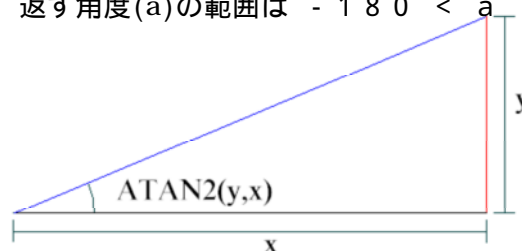
絶対値：ABS

2引数の関数

三角関数：ATAN2 { ATAN2(y,x) }

X方向の距離(x)、Y方向の距離(y)から角度を計算します。

返す角度(a)の範囲は $-180 < a \leq 180$ です。



文字列 数値変換：VAL

数値 文字列変換：STR

特殊な関数

MODE (<変数>) 変数のモーダル状態を調べます。

変数のモーダル状態がONの時に真を返し、

OFFの時には偽を返します。

括弧内には一つの変数だけしか記入できません。

定数

条件文では次のものを定数として扱います。

円周率 (PI)

文字列

条件文中の " ~ " で括られた部分は文字列データとして扱います。

文字列に対しては 比較演算子 (= =、< >、<、>) による比較演算と + 演算子で結合の演算を行うことができます。

例 1 + 演算子

<USER_1> が"輪郭加工"、<USER_2> が"輪郭" の場合

条件文 <USER_01>==<USER_02> + "加工" は真になります。

例 2 <、> 演算子

A<BはAがBに含まれる場合、A>BはBがAに含まれる場合に真になります。

<USER_1> が"輪郭加工"、<USER_2> が"輪郭" の場合

条件文 <USER_01>><USER_02> は真になります。

§ 4.7 使用可能変数一覧

EGPostで利用可能な変数の一覧表を掲示します。

凡例

変数種類 R: 実数、I : 整数、S : 文字列、

条件 : 変数のモーダル状態で出力を制御する変数、

構文 : 条件構文の動作を制御する変数を意味します。

名称	概要	種類	内部コード
_CYCNREGIT	リジットタップを使用しない時にON	条件	3006
_CYCREGIT	リジットタップ使用する時にON	条件	3005
_NTLCPRC	工具交換のない手続き時にON	条件	3004
_TL~LAST	最終の工具交換時にOFF	条件	3003
_TLFAST	最初の工具交換時にON	条件	3000
_TLLAST	最終の工具交換時にON	条件	3002
_TLNEXT	2番目以降の工具交換時にON	条件	3001
AIRLEN	空切削長さ	R	47
AIRTIME	空切削時間	R	44
ARC_ANG	円弧の中心角	R	301
ARC_PLN	円弧平面	I	313
AX_CORD	工具移動先のX座標値(絶対値)	R	51
AX_CTRL	制御点X座標(絶対値)	R	1723
AX_HOME	基準点X座標(絶対値)	R	123
AX_MACH	機械原点X座標(絶対値)	R	126
AX_START	開始点X座標(絶対値)	R	227
AY_CORD	工具移動先のY座標値(絶対値)	R	52
AY_CTRL	制御点Y座標(絶対値)	R	1724
AY_HOME	基準点Y座標(絶対値)	R	124
AY_MACH	機械原点Y座標(絶対値)	R	127
AY_START	開始点Y座標(絶対値)	R	228
AZ_CORD	工具移動先のZ座標値(絶対値)	R	53
AZ_CTRL	制御点Z座標(絶対値)	R	1725
AZ_HOME	基準点Z座標(絶対値)	R	125
AZ_MACH	機械原点Z座標(絶対値)	R	128
AZ_START	開始点Z座標(絶対値)	R	229
C_FILENAME	現在のファイル名	S	43
CHECK_OFST	チェックオフセット量	R	201
CHECK_TOL	チェックトレランス量	R	202
CIRMOVE	円弧動作指令	S	20
CLEAR LENG	有効長	R	1512
CMPMOVE	径補正動作指令	S	23
CON_ANG	工具の勾配角度	R	1501
CONT_OFST	輪郭オフセット量	R	203
CONT_TOL	輪郭トレランス	R	204
COOLANT	クーラント	S	17

名称	概要	種類	内部コード
CTRL_NUM	制御点数	I	1710
CURR_NAME	現在の工具名称	S	1502
CURR_ORIG	現在の原点に対応するNCコントローラの原点の番号	I	1910
CURR_TOOL	現在の工具番号	I	1504
CUT_LENGTH	刃長	R	1513
CUTLEN	切削長さ(Proc開始時にクリア)	R	35
CUTTIME	切削時間(Proc開始時にクリア)	R	36
CYC_DWELL	ドウェル値	R	504
CYC_PECK	ペック値	R	506
CYC_R	出力固定サイクルR値	R	25
CYC_REduc	固定サイクル戻り量	R	507
CYC_SZ	固定サイクル前Z値	R	26
CYC_TIMES	断続回数	I	509
CYC_XSHFT	Xシフト量	R	510
CYC_YSHFT	Yシフト量	R	511
CYC_Z	出力固定サイクルZ値	R	24
CYCMOVE	固定サイクル動作指令	S	21
CYCRET	復帰動作指令	S	22
DATE_DD	現在の日	I	111
DATE_MM	現在の月	I	112
DATE_YY	現在の年	I	113
DEL_Z_UP	クリアランス量	R	205
DIA_COMP	径補正番号	R	1514
DIAMETER_	工具径	R	1515
DOWN_STEP	ダウンステップ量	R	206
DVAL	径補正值	I	28
EGP_NAME	EGPファイルのパス名	S	140
ELSE	条件式が偽(= 0)のときの処理	構文	64004
ELSEIF	2番目以降の条件式	構文	64003
END_FILE	ファイル分割終了指令	S	5001
ENDIF	条件文の終了	構文	64005
FEED	送り速度	I	14
FILENUM_INI	分割ファイル番号の初期値	I	130
FILENUM_CRR	現在の分割ファイル番号	I	131
FIXT_COMP	治具補正值	R	1516
GAUGE_LEN	ゲージ長	R	1517
GMOVE	出力動作指令	S	18
HOLD_BOT01	ホルダー1底面径	R	1518
HOLD_BOT02	ホルダー2底面径	R	1521
HOLD_BOT03	ホルダー3底面径	R	1538
HOLD_BOT04	ホルダー4底面径	R	1542
HOLD_BOT05	ホルダー5底面径	R	1546

名称	概要	種類	内部コード
HOLD_CON01	ホルダー1テーパ高さ	R	1519
HOLD_CON02	ホルダー2テーパ高さ	R	1522
HOLD_CON03	ホルダー3テーパ高さ	R	1539
HOLD_CON04	ホルダー4テーパ高さ	R	1543
HOLD_CON05	ホルダー5テーパ高さ	R	1547
HOLD_NUM	ホルダー定義数	I	1524
HOLD_TOP01	ホルダー1上面径	R	1520
HOLD_TOP02	ホルダー2上面径	R	1523
HOLD_TOP03	ホルダー3上面径	R	1540
HOLD_TOP04	ホルダー4上面径	R	1544
HOLD_TOP05	ホルダー5上面径	R	1548
HOLD_TOT01	ホルダー1全体高さ	R	1536
HOLD_TOT02	ホルダー2全体高さ	R	1537
HOLD_TOT03	ホルダー3全体高さ	R	1541
HOLD_TOT04	ホルダー4全体高さ	R	1545
HOLD_TOT05	ホルダー5全体高さ	R	1549
HOLDER_DIA	ホルダー径	R	1506
HVAL	長補正值	I	29
I_CORD	出力I座標値	R	30
IF	条件文の開始・最初の条件式	構文	64001
INS_STR	出力メッセージ	S	701
J_CORD	出力J座標値	R	31
K_CORD	出力K座標値	R	50
KNOT	制御点の節点(ノット)	R	1730
LENG_COMP	長補正番号	R	1525
LINMOVE	直線動作指令	S	19
MOVELEN	総動作長さ(空動作長さ+切削長さ)	R	48
MOVETIME	総動作時間(空動作時間+切削時間)	R	49
MOVMNT_NUM	動作数	I	208
NEW_FILE	ファイル分割開始指令	S	5000
NEXT_NAME	次の工具名称	S	1503
NEXT_TOOL	次の工具番号	I	1505
NUM_LAYERS	レイヤ数	I	207
NURBS_DEG	NURBS曲線の階数	I	1712
NURBSMOVE	NURBS補間指令	S	1705
OSIDE_STEP	レイヤ間/サイドステップ	R	230
PART_NAME	ファイル名	S	101
PART_OFST	面オフセット	R	209
PART_TOL	面公差	R	210
PLANE	平面指令	S	46
LATFORM_	プラットフォーム	S	211
PRCD_FED_TIME	手続きの切削加工時間	R	250
PRCD_FST_TIME	手続きの空動作時間	R	251
PRCD_GEN_TIME	手続きの総加工時間	R	252

名称	概要	種類	内部コード
PROC_CMNT	手続き注釈	S	212
PROC_FULLNAME	長い手続き名 [メイン選択]-[サブ選択]	S	232
PROC_NAME	手続き名	S	213
PROC_NUM	手続き番号	I	214
R_CORD	出力R値	R	13
RADIUS_	円弧半径	R	305
SHANK_BOT	シャンクの底面径	R	1526
SHANK_CON	シャンクのテーパ－高さ	R	1527
SHANK_TOP	シャンクの上面径	R	1528
SHANK_TOT	シャンクの全体高さ	R	1529
SIDE_STEP	サイドステップ	R	215
SPINDIR	回転指令	S	16
SPINDLE	回転数	I	15
SPNDL_BOT	主軸の底面径	R	1530
SPNDL_CON	主軸のテーパ－高さ	R	1531
SPNDL_TOP	主軸の上面径	R	1532
SPNDL_TOT	主軸の全体高さ	R	1533
TAPELEN	テープ長	R	45
TEETH_NUM	工具の刃数	I	1534
THEN	条件式が真(0)のときの処理	構文	64002
TOOL_CMNT	工具注釈	S	1507
TOOL_MAT	工具材質	S	1508
TOOL_RAD	工具のコーナーR	R	1535
TP_CMNT	軌跡注釈	S	118
TP_NAME	工具軌跡名	S	102
TRANSF_NUM	変換番号	I	216
USER_01	ユーザ定義1	S	2001
USER_02	ユーザ定義2	S	2002
USER_03	ユーザ定義3	S	2003
USER_04	ユーザ定義4	S	2004
USER_05	ユーザ定義5	S	2005
USER_06	ユーザ定義6	S	2006
USER_07	ユーザ定義7	S	2007
USER_08	ユーザ定義8	S	2008
USER_09	ユーザ定義9	S	2009
USER_10	ユーザ定義10	S	2010
USER_C01	ユーザ定義C1	R	2101
USER_C02	ユーザ定義C2	R	2102
USER_C03	ユーザ定義C3	R	2103
USER_C04	ユーザ定義C4	R	2104
USER_C05	ユーザ定義C5	R	2105
USER_C06	ユーザ定義C6	R	2106
USER_C07	ユーザ定義C7	R	2107
USER_C08	ユーザ定義C8	R	2108
USER_C09	ユーザ定義C9	R	2109

名称	概要	種類	内部コード
USER_C10	ユーザ定義C10	R	2110
USER_C11	ユーザ定義C11	R	2111
USER_C12	ユーザ定義C12	R	2112
USER_C13	ユーザ定義C13	R	2113
USER_C14	ユーザ定義C14	R	2114
USER_C15	ユーザ定義C15	R	2115
USER_C16	ユーザ定義C16	R	2116
USER_C17	ユーザ定義C17	R	2117
USER_C18	ユーザ定義C18	R	2118
USER_C19	ユーザ定義C19	R	2119
USER_C20	ユーザ定義C20	R	2120
USER_NAME	ログイン名	S	117
WEIGHT	制御点の重み(ウェイト)	R	1740
X_CORD	工具移動先のX座標値(出力値)	R	10
X_CTRL	制御点X座標(出力値)	R	1720
X_HOME	基準点X座標(出力値)	R	103
X_MACH	機械原点X座標(出力値)	R	106
X_MAX	X_最大値(Proc開始時にクリア)	R	38
X_MIN	X_最小値(Proc開始時にクリア)	R	37
X_POSIT	現在の絶対X値	R	32
X_START	開始点X座標(出力値)	R	217
XMIN_CUT	加工動作でのX_最小値(Proc開始時にクリア)	R	57
XMAX_CUT	加工動作でのX_最大値(Proc開始時にクリア)	R	58
Y_CORD	工具移動先のY座標値(出力値)	R	11
Y_CTRL	制御点Y座標(出力値)	R	1721
Y_HOME	基準点Y座標(出力値)	R	104
Y_MACH	機械原点Y座標(出力値)	R	107
Y_MAX	Y_最大値(Proc開始時にクリア)	R	40
Y_MIN	Y_最小値(Proc開始時にクリア)	R	39
Y_POSIT	現在の絶対Y値	R	33
Y_START	開始点Y座標(出力値)	R	218
YMIN_CUT	加工動作でのY_最小値(Proc開始時にクリア)	R	59
YMAX_CUT	加工動作でのY_最大値(Proc開始時にクリア)	R	60
Z_CORD	工具移動先のZ座標値(出力値)	R	12
Z_CTRL	制御点Z座標(出力値)	R	1722
Z_DOWN	加工最低点	R	220
Z_HOME	基準点Z座標(出力値)	R	105
Z_MACH	機械原点Z座標(出力値)	R	108
Z_MAX	Z_最大値(Proc開始時にクリア)	R	42
Z_MIN	Z_最小値(Proc開始時にクリア)	R	41
Z_POSIT	現在の絶対Z値	R	34
Z_START	開始点Z座標(出力値)	R	219
Z_UP	加工最上点	R	221
ZMIN_CUT	加工動作でのZ_最小値(Proc開始時にクリア)	R	61
ZMAX_CUT	加工動作でのZ_最大値(Proc開始時にクリア)	R	62

GPP2ベースで使用する場合のみ使える変数

名称	概要	種類	内部コード	XMLコード
INTR_CLEARANCE	内部安全共通点 (Z 絶対値)	R	2201	3303
FLOOR_OFST	加工面底面オフセット	R	2202	20391
BASE_FEED	送り速度の設定値	I	2203	3201

§ 4.8 座標値変数について

座標値の変数は1つの数値に対して2つの変数(内部変数、出力変数)を使用します。それぞれの使い方は以下の点に注意してください。

	内部変数	出力変数
数値	モデル座標系での 絶対値	設定によって変更 (絶対値、増分値、座標の移動)
使用箇所	出力変数は条件文には使えません 条件文 に座標値を使うときは 内部変数を使ってください	一般の座標値 の出力には 出力変数を使ってください。
書式	実数としての書式 にしてください。 内部変数には 計算式を設定しない てください。	出力時の書式 にしてください。 内部変数と出力変数は精度が同じに なるようにしてください。

内部変数と出力変数の使い分けが必要な変数

内部変数	出力変数	概要
AX_CORD	X_CORD	工具移動先のX座標値
AX_CTRL	X_CTRL	制御点X座標
AX_HOME	X_HOME	基準点X座標
AX_MACH	X_MACH	機械原点X座標
AX_START	X_START	開始点X座標
AY_CORD	Y_CORD	工具移動先のY座標値
AY_CTRL	Y_CTRL	制御点Y座標
AY_HOME	Y_HOME	基準点Y座標
AY_MACH	Y_MACH	機械原点Y座標
AY_START	Y_START	開始点Y座標
AZ_CORD	Z_CORD	工具移動先のZ座標値
AZ_CTRL	Z_CTRL	制御点Z座標
AZ_HOME	Z_HOME	基準点Z座標
AZ_MACH	Z_MACH	機械原点Z座標
AZ_START	Z_START	開始点Z座標

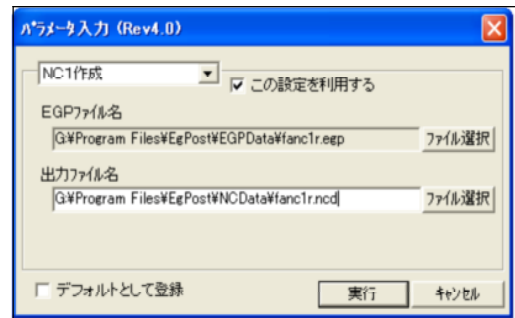
第5章 補足

この章では、EgPostを利用して特殊ポストを作成する場合の方法を説明します。
EgPostの利用方法そのものは、以前の章に記載されていますので、重複する内容は省いてあります。

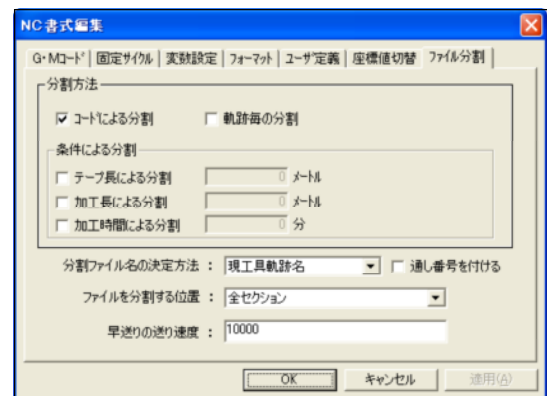
本章では、”このような場合どうしたらいいのか？”といったケースに沿って、考え方・設定例を紹介し、より詳細なポストの作成を説明します。

§ 5.1 のポストと同じ名前で、NCプログラムを出力するには？（その１）

通常、EgPostからのNCプログラム出力は、図に示すウィンドウで指示するようになっています。しかし、出力ファイルを自動的に工具軌跡名と同じにするにはどうすればいいのでしょうか。



これは、メニューの編集 / NC書式設定で、ファイル分割を利用し、分割方法：コードによる分割 分割ファイル名：工具軌跡名として設定します。（下図参照）



さらに、分割のためのコード<NEW_FILE>を、プログラム開始セクションの先頭行に挿入します。さらに、<END_FILE>を、プログラム終了セクションの最終行に挿入します。以上で、プログラム全体が工具軌跡名の名前のファイルに保存されることになります。（指示したファイル名は、作成されてしまいますが、何も入っていないファイルになっているはずです）

注）ここでは、ファイル分割方法として、コードによる分割にしています。

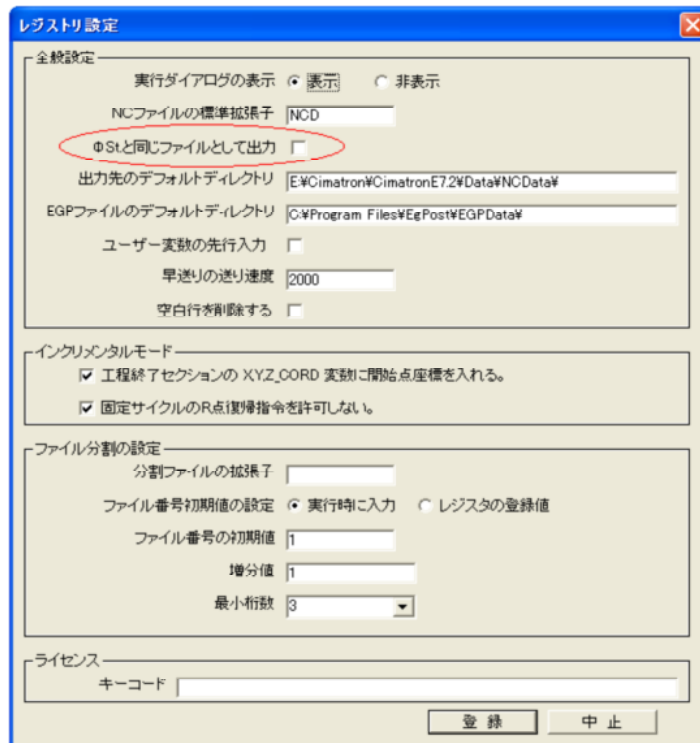
そのため、テープ長さによる分割を利用する場合にはこの方法を利用することはできません。

§ 5.2 のポストと同じ名前で、NCプログラムを出力するには？（その2）

EgPost のオプション（隠し）として、STATION、CimatronEのポストで出力されるファイルと同一ファイル名で出力するための設定があります。

ここでは、そのオプションの設定方法を紹介します。

「ファイル」メニューの「レジストリ設定」を実行します。
以下のダイアログボックスが表示されます。

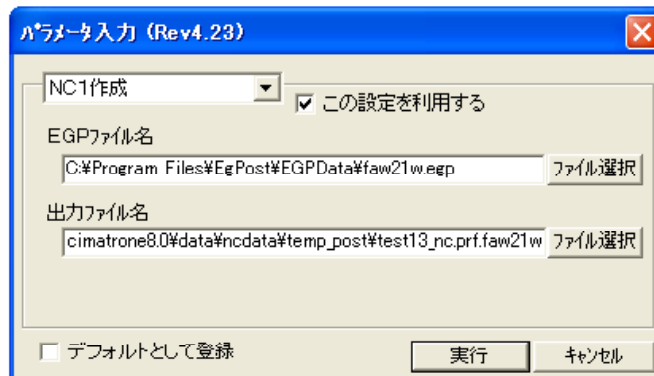


「St」と同じファイルとして出力」をチェックします。

「登録」ボタンを押すと、確認のダイアログボックスが表示されるので「はい」ボタンを押します。

以上の設定を行った状態で、EGPostを実行すると以下のようなダイアログを表示します。

出力ファイル名がCimatronE、Stationの標準のNCプログラムファイルのように
"ドキュメント名"."最初の軌跡名"."ポスト名"になります。ファイル選択でEGPファイル名を変更すると出力ファイル名の"ポスト名"の部分も自動で変更されます



手入力でEGPファイルのファイル名を変更しても出力ファイルには反映されません。

NCプログラムファイルの保存先はCimatronE本体からEGPostを起動させたときに"目標フォルダ"に設定されていたフォルダに設定されます。

§ 5.3 テープ長さ / 加工長さで出力ファイルを分割するときの注意点

EgPost では、コードによるファイル分割以外にテープ長さ / 加工長さによる分割も可能となります。この時は、コードによるファイル分割とは分割手法が若干異なりますので注意が必要です。

この時の注意点について、以下に記載します。

1、テープ長さ、加工長さによる分割を選択すると、コードの中に <NEW_FILE>, <END_FILE> が存在してもこのコードは無視されます。

2、コードによる分割の場合、<END_FILE>が参照された時点で分割する前のファイルが再び出力ファイルとして設定されます。

しかし、テープ長さ / 加工長さによる分割では以前のファイルに戻るということは出来ません。すなわち、常に新しいファイルへ出力先が移動していきます。

3、テープ長さ / 加工長さによる分割時のセクションの呼ばれ方
分割時には、以下のような順序で作業が行われます

分割長さに到達 分割終了セクションを実行 現在のファイルを保存
新しいファイルに出力先を変更 分割先頭セクション実行 次のセクションを実行

§ 5.4 テープ長さでファイルを分割するには？

EgPost では、出力されるNCプログラムの大きさによって自動的にファイルを複数のファイルに分割出力することが可能です。

これを行うには、以下のように行います。

編集 / NC 書式設定を選択し、ファイル分割をマウスでクリックします。

(前ページの図参照)

分割方法として、テープ長さによる分割を選択します。

分割したときのファイル名を適宜選択します。(元ファイル+数値)が最適です。

分割するテープ長さを指定します。

以上だけの設定で、テープ長さで分割できるようになります。

具体的な、コードの設定方法は、§ 2 の注意点を参照してください。

§ 5.5 工具毎にファイルを分割するには？

EgPostで工具毎にNCプログラムを作成し、プログラム毎にファイルを分けて出力する方法を説明します。
工具毎にファイルを分割するために、工具交換セクションの先頭に以下のコードを入れます。

[<_TLNEXT><END_FILE>]<NEW_FILE>;

<NEW_FILE>コードでファイルを分割します。

[<_TLNEXT><END_FILE>] コードで工具交換の時、現在のファイルに分割ファイル終了のコードを書き込みます。<_TLNEXT>コードを使って、最初の工具交換では分割ファイル終了のコードを出力しないようにしています。

各ファイルの内容を完成したNCプログラムにするために分割ファイル先頭にNCプログラムの開始部分を、分割ファイル終了にNCプログラムの終了部分を書き込みます。

最後の工程の後では工具交換は行いませんので分割ファイル終了セクションが出力されません。最後の工程の後にはプログラム終了セクションが出力されますので、プログラム終了セクションにも<END_FILE>コードを記述する必要があります。

この場合、工具交換の回数分のNCプログラムが一度に作成されますが、そのNCプログラムのプログラム番号を全て同じにしたい、また、一回一回入力するのは面倒だという場合には以下のようにしてください。

- ・プログラム番号を

"0<USER_C01>;"

とします。ユーザー変数は数値変数の1から20までどの変数を使ってもかまいません。

- ・NC書式編集でユーザー変数を右図のように設定します。

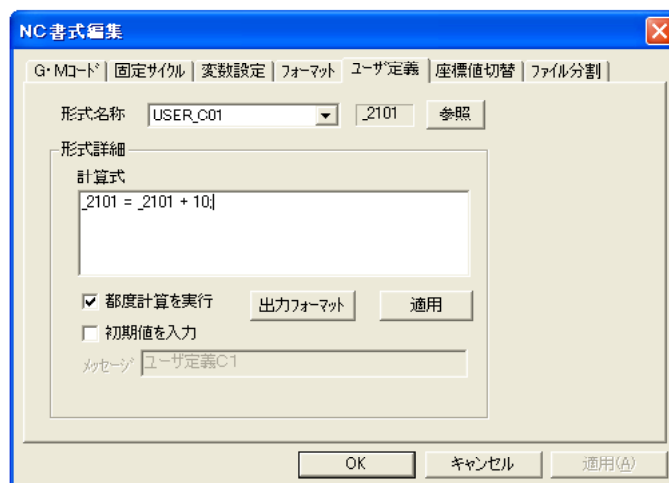
右図の設定の場合、変数は呼び出される度に10ずつ増加することになります。

出力フォーマットは

「0を先頭に着ける」をチェック

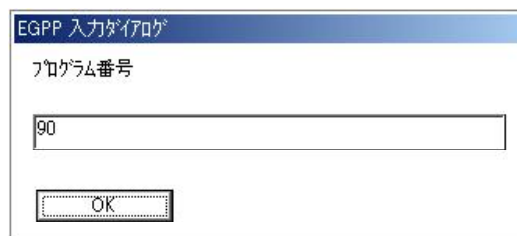
「整数部桁数」を 4、

「小数点桁数」を 0 にします。



「初期値を入力」をチェックすると
EgPostの実行時に右図のような、初期値を入力する
ダイアログボックスを開きます。

プログラム番号は初期値90で計算を実行した
100からになります。結果としてプログラム番号は
00100、00110、00120、・・・となります。



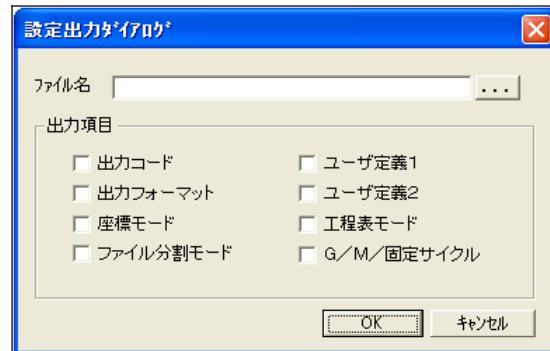
「初期値を入力」をチェックしないと 初期値は0となり、
プログラム番号は10からになり、プログラム番号は
00010、00020、00030、・・・ となります。

§ 5.6 設定内容を、テキストファイルとして保存するには？

EgPost 内に設定した全ての内容を一覧にして出力することが可能です。
これは、以下のようにして行います。

メニューから、ファイル / 設定のファイル出力を選択します。

以下のような画面が表示されます。



ここで、内容一覧を出力するファイル名を入力します。

出力する内容をマウスでクリックします。

OK ボタンを押せば、一覧形式で出力されます。

§ 5.7 工具中心座標で、NC プログラムを作成するには？

EgPost で出力される座標値は、G90,91,92を問わずすべて工具の先端座標として出力されます。
これを、工具中心座標として出力する場合の方法（および注意点）を説明します。

メニューの編集 / NC 書式編集を選択し、座標値切替えを選択します。

座標値から、工具中心位置をマウスで選択します。

以上で、工具中心位置座標でNCプログラムが出力されます。

尚、工具中心座標はボールエンドミル（工具 $R \times 2$ = 工具直径）の時のみ出力され、フラットエンドミルやブルノーズ（R 付きフラット工具）では設定に関わらず、工具先端で出力されます。

§ 5.8 シーケンス番号を付けるには？

NC プログラム内にシーケンス番号 (cf., N0002) を付ける場合の方法を紹介します。
シーケンス番号は、出力するたびにその数値を一定間隔でインクリメントするとします。
このような数値は、ユーザ変数 (計算変数) を利用し出力することが可能です。
計算式の詳細な件については、後節で紹介します。

コードの記述

シーケンス番号を記述したいセクションに、

N<USER_C01>;

のように記述しておきます。

USER_C01...C20のいずれを利用してもOKです。USER_C01以外の変数を使う場合には以下の説明のユーザー変数、内部コードもその変数に読み替えてください。

メニューの編集 / NC 書式編集を選択し、ユーザ定義を選択します。

変数名として、USER_C01 を選択します。

この変数が、出力のたびに + 2 する場合は、以下のような計算式を入力します。

_2101 = _2101 + 2; (_2101はユーザー変数 USER_C01 の内部コードです。)

都度計算を選択状態にしておきます。

必要に応じて、USER_C01 のフォーマットを選択します。

0002, 0004 であればフォーマットは以下の通りです。

O を先頭に付ける、整数部 = 4、小数部 = 0

Modal は必ず Off にしておいてください。

§ 5.9 円弧補間を R で出すには？

EgPost では、円弧補間の方法として、I J / R のいずれでも出力が可能です。
ここで、R の場合で円弧角度が 180 度以上の場合はどうしたらよいでしょうか？

EgPost ver3.* では、どのような場合でも円弧角度が 180 を超える場合には自動的に 2 つの円弧に自動的に分割してしまいます。このため、最大 360 の場合には、円弧角度 = 180 の円弧が 2 つ出力されることになっていました。

しかし、これでは NC プログラムが長くなってしまったため EgPost ver4.* では、円弧を分割せずに、条件構文を使って R 指令を出力できるようにしました。

R 指令の円弧補間の記述は以下ようになります。詳細は § 4.6 例 6 (条件構文の記述) で説明してありますので、ここでは省略します。

```
<IF>abs(<ARC_ANG>)>=360.0 <THEN><PLANE><GMOVE>{X<X_CORD>}{Y<Y_CORD>}{Z<Z_CORD>}{I<I_CORD>}{J<J_CORD>}{K<K_CORD>}{F<FEED>};  
<ELSEIF>abs(<ARC_ANG>)>180.0 <THEN><PLANE><GMOVE>{X<X_CORD>}{Y<Y_CORD>}{Z<Z_CORD>}{R<R_CORD>}{F<FEED>};  
<ELSE><PLANE><GMOVE>{X<X_CORD>}{Y<Y_CORD>}{Z<Z_CORD>}{R<R_CORD>}{F<FEED>};  
<ENDIF>
```


§ 5.1 0 MODE()関数の応用例

MODE()関数を使うことで変数をスイッチとして使うことができます。

例えば、工程表に径補正番号を出力するときに、径補正を行う手続きだけに径補正番号を出力したい場合は、以下のような方法があります。

工程先頭セクションに以下のコードを追加します。(＜USER_10＞のモーダル状態をOFFにします。)

```
$＜USER_10＞
```

アプローチセクションに以下のコードを追加します。(＜USER_10＞のモーダル状態をONにします。アプローチセクションを出力した場合のみ＜USER_10＞がONになります。)

```
&＜USER_10＞
```

工程表の径補正番号の欄の設定を以下のようにします。(＜USER_10＞がONの場合、つまりアプローチセクションを出力した場合のみ＜DIA_COMP＞を出力します。)

```
<IF>MODE(＜USER_10＞)<THEN><DIA_COMP><ENDIF>
```

<DIA_COMP>はモーダル設定をOFF

<USER_10>はモーダル設定をON、「都度入力が必要」のチェックを外し、「固定値を設定」にチェックを入れます。

§ 5.1 1 G 9 2 モードの考え方

G 9 2 を座標設定に利用する場合は、ケースに応じて作成方法を変えなければなりません。

- 、工具スタート位置を安全共通点に合わせて、そこに独自の座標系として G 9 2 を利用する場合

この場合は、EgPostのG92モードを利用します。

独自座標系は、EgPostの機械座標入力時に入力した数値が利用されますから、EgPost のプログラム開始セクションで以下のようなコードを記述するだけでO kです。

```
G92 X<X_MACH> Y<Y_MACH> Z<Z_MACH>
```

- 、工具スタート位置を任意の場所に合わせて、そこから G 9 0 で動作させる場合

この場合は、通常のG90出力と何ら変わりありません。

- 、工具スタート位置を任意の場所に合わせて、そこから G 9 1 で動作させる場合

この場合は、通常のG91出力方法と変わりありませんが、プログラム先頭で G 9 0 を利用し、安全共通点 / 手続き開始点に移動させるコードを入力しておく必要があります。

example

プログラム開始セクション

```
G92 X0 Y0 Z50.;
```

```
G90 X<X_HOME> Y<Y_HOME> Z<Z_HOME>;
```

```
G91
```

```
...
```

§ 5.1 2 N U R B S 補間の入力方法

EGPost には Rev3.5 より N U R B S 補間の機能が加わりました。

NURBS補間は以下のような形式になります。

```
G06.2 K0. X0. Z0. ;  
      K0. X300. Z100. ;  
      K0. X700. Z100. ;  
      K0. X1300. Z-100. ;  
      K0.5 X1700. Z-100. ;  
      K0.5 X2000. Z0. ;  
      K1.0. ;  
      K1.0. ;  
      K1.0. ;  
      K1.0. ;  
G01 Y0.5 ;
```

} a
}
} b

a の部分を " N U R B S 補間 " セクションに、b の部分を " N U R B S 補間終了 " セクションに記述します。

通常のセクションの考え方と同じように a) も b) も全てを記述する必要はありません。以下のように 1 行だけ記述しておけばセクションが繰り返し呼び出されて上記のような N C プログラムが作成されます。

" N U R B S 補間 " セクションの例

```
<NURBSMOVE>{P<NURBS_DEG>}{K<KNOT>}{X<X_CTRL>}{Y<Y_CTRL>}{Z<Z_CTRL>}  
{R<WEIGHT>;}
```

" N U R B S 補間終了 " セクションの例

```
{K<KNOT>;  
}
```

N U R B S 補間で使用する変数の説明

<NURBSMOVE>	N U R B S 補間モードを ON にする G コード (NURBSMOVE 変数は N U R B S 補間終了のセクションが出力されるとモーダルを解除します。従ってこの変数は N U R B S 補間の先頭で必ず出力されるようになります。)
P<NURBS_DEG>	N U R B S 曲線の階数
K<KNOT>	ノット
X<X_CTRL>	制御点の X 座標
Y<Y_CTRL>	" Y "
Z<Z_CTRL>	" Z "
R<WEIGHT>	ウェイト

§ 5.1 3 計算式の入力あれこれ

Egpost 内の変数<USER_C01>...<USER_C20>は、計算によって得られる数値を出力することが出来ます。

この計算式について説明します。

、どのような演算が可能か？

EgPost内で出来る演算は、以下の通りです。

- ・ $\times \div + -$ の四則演算
- ・ sqrt, sin, cos, tan, log の数学関数
- ・ $>, <, >=, <=$ の比較演算(真ならば 1、偽ならば 0)
- ・ 文字列を利用する演算はできません。

、計算式の例

次に、上記の計算式を利用して出力する例を紹介します。

・ X , Y の座標を 4 5 度傾けて出力する。

この場合の計算式は、4 5 度傾けた X , Y 座標値をそれぞれ変数<USER_C01>, <USER_C02>とすれば、

$\text{<USER_C01>} = \text{<X_CORD>} * \cos(45) + \text{<Y_CORD>} * \sin(45);$

$\text{<USER_C02>} = \text{<X_CORD>} * (-\sin(45)) + \text{<Y_CORD>} * \cos(45);$ となります。

そこで、実際の計算式の入力には内部変数を利用することになりますから、以下のようになります。

$_2101 = _10 * \cos(45) + _11 * \sin(45); \quad \dots(1)$

$_2102 = _10 * (-\sin(45)) + _11 * \cos(45); \quad \dots(2)$

すなわち、(1)の式を<USER_C01>の定義に、(2)の式を<USER_C02>の定義に利用します。

具体的な出力コードは以下のようにすればいいでしょう。

X<USER_C01> Y<USER_C02>

、計算式入力時の注意事項

上記の例では、計算結果を<USER_C01>, <USER_C02> の変数を利用して出力する例を紹介しましたが、一般変数の数値を置き換えることも可能です。

たとえば、<USER_C01>の計算式として $_10 = 1.0;$ とすれば、コードの中で<USER_C01>が参照されたときに、自動的にこの計算式が処理されますから $_10$ <X_CORD> の数値が 1.0 になってしまいます。このように、全ての数値変数は置き換えが可能となっていますので、注意が必要です。

仮に、計算途中で任意の変数が利用したい場合には、 $_$ で始まらない文字列を任意変数として利用してください。

cf., $A = _10 + 1; B = _11 + 2; _2101 = A * B;$

§ 5.1 4 工程表作成時の注意

EgPost を利用すれば、工程表（NCプログラム）内にCimatronEで設定した加工パラメータ（面オフセット、加工最上点、）を出力することが可能です。

しかし、一部変数によっては加工手続きの種類により出力されない変数もありますから注意が必要です。

加工パラメータの代表的変数と、利用可能な加工手続きの一覧表を以下に掲載します。

（ただし、利用できる手続きであってもその変数が利用されていない状態の場合には同様に出力されなくなります）

	輪郭加工	ポケット	面ポケット	等高切削	Z切削	面沿加工	面輪郭	面領域沿	R M X
CHECK_OFST									
CHECK_TOL									
CONT_OFST									
CONT_TOL									
DEL_Z_UP									
DOWN_STEP									
PART_OFST									
PART_TOL									
SIDE_STEP									
Z_DOWN									
Z_UP									

工程表出力時の変数の扱いとしては、NCの場合と同様にモータル状態のON、OFFによって出力・非出力の切り替えが行われます。

このため、出力する場合には !<CHECK_OFST>のように、必ず !を付加した変数を利用するようにしてください。

第6章 レジストリ設定

EgPost は、動作の決定にレジストリ情報を利用します。

このレジストリ情報は、インストール時に決定され EGP ファイルの所在等が記録されています。この情報を必要に応じて変更することで動作のカスタマイズを行うことができます。

変更は通常、レジストリ設定のダイアログボックス内で行います。

レジストリ設定の項目一覧を記載します。

全般設定

項目名	説明
実行ダイアログの表示 (表示、非表示)	非表示を選択する(レジストリエントリを1にする)と実行のためのダイアログが表示しなくなります。ただし、出力ファイルとEGPファイルがともにデフォルトとして保存されていなければなりません。
NCプログラムの標準拡張子	NCプログラムファイルの拡張子です。NCプログラムファイルの選択ダイアログを開くと最初にこの拡張子のファイルのみが表示されます。また拡張子のないファイルを指定すると自動でこの拡張子を付けます。
Stと同じファイルとして出力	チェックを入れると、STATION、CimatronEと同じファイル形式で出力します。Station、CimatronEのNCプログラムは通常[ドキュメント名].[軌跡名].[ポスト名]になりますのでこれと同じような形式にします。
標準拡張子を使用	出力先もCAMで指定した出力先になります 出力ファイルの拡張子を"NCプログラムの標準拡張子"で指定した拡張子にします ファイル名:[ドキュメント名].[軌跡名].[標準拡張子]
EGPファイル名を使用	出力ファイルの拡張子をEGPファイルのファイル名(拡張子を除く)にします ファイル名:[ドキュメント名].[軌跡名].[EGPファイル名]
ユーザー変数の先行入力	チェックを入れるとユーザー変数の先行入力を行います。
早送りの送り速度	この送り速度値は空切削時間の計算に使われます。空切削時間が0になる時はこの値を確認してください。 またRev3.91以降は送り速度がこの値より大きい場合は切削送りも早送りで見なします。
空白行を削除する	プログラム中に空白行を出力しない様にします。
NCプログラム作成完了のメッセージを表示	NCプログラムを作成した後に作成完了のメッセージを出すようにします
デフォルトフォルダ設定	
出力先	出力先のデフォルトディレクトリ
EGPファイル	EGPファイルのデフォルトディレクトリ

インクリメンタルモード

工程終了セクションのX,Y,Z_CORD変数の開始点座標を入れる。	この項目にチェックを入れると、インクリメンタルモードでは工程終了セクションのX,Y,Z_CORD変数に開始点座標が入るようになります。(デフォルトではチェックが入った状態になっています。チェックが入った状態がRev3.8と同じ仕様になります。)
固定サイクルのR点復帰を許可しない。	Rev4.03ではインクリメンタル指令の場合、固定サイクルの復帰高さをR点高さに設定してもG99のコードは出力されずに、強制的にG98を出力し、G99は使用できないようになっていましたが、これを使用できるように選択できるようにしました。このチェックを外すとG99が出力されるようになります。初期設定(インストール直後)ではチェックが入っています。

ファイル分割の設定

項目名	説明
ファイル分割の拡張子	ファイル分割したときの拡張子
ファイル番号初期値の設定	ファイル名を元ファイル + 番号にしてファイル分割を行う場合、あるいは「通し番号を付ける」にチェックを入れたときの番号の初期値の設定方法を選択します。 実行時に入力 : 最初にファイルを分割する際にウィンドウが表示されるので入力します。 レジスタの登録値: ファイル番号の初期値を使用します。
ファイル番号の初期値	ファイル分割を行う場合のファイル番号の初期値。
増分値	ファイル分割を行う場合のファイル番号の番号の増分値。
最小桁数	ファイル分割を行う場合のファイル番号の桁数。

ライセンスに関する設定

キーコード	EGPostを動作させるためのキーコードです。 これが入っていないとき、あるいは正しいコードが入っていないときはNCプログラムの行数に制限が付きます。
-------	--

レジストリエディタ等で直接レジストリを操作することはなるべく行わないようにしてください。やむなく操作を行うときは、細心の注意を払い行ってください。不用意に書き換えると、EgPostだけでなくシステム全体が不安定になることもあります。

上記のエントリは、HKEY_LOCAL_MACHINE¥Software¥SAEILO¥EGPの下に存在します。

上記以外のエントリは操作しないようにしてください。

第7章 終わりに

最後に、本プログラム利用上の注意点を掲載します。

- ✓ EGP Postで作成されたNCプログラムは、EGPファイルの作り方で大きく変わってきます。このため、本ソフトで作成されたNCプログラムを利用し、機械が正常動作しなくなる可能性も生じます。ゆえに、作成したNCプログラムは、検証ソフト等で確認をとっていただくようお願いいたします。
- ✓ EGP Postで作成可能なNCプログラムサイズに限度はありませんが、数 + メガ単位のNCプログラムを作成すると、非常に時間がかかりますのであらかじめご了承ください。
- ✓ EGP Postで作成されたNCプログラムによる工具等の破損などのいかなる引責にも、弊社はこれを負わないものといたします。
また、本ソフトを利用して生じたいかなる引責に対しても、弊社はその責を負いません。
- ✓ 本ソフトは、コンピュータ単位でライセンスされます。ライセンスされたコンピュータ以外で利用する場合は、登録が必要です。詳細は、担当営業までお問い合わせください。
- ✓ いかなる場合であっても、本ソフトの改変・無許可複製・逆アセンブルを行うことは禁止しております。

問い合わせ先

本プログラムおよび取り扱い説明書に関する、お問い合わせは以下までお願いいたします。

TEL : 043-350-4820 FAX : 043-350-4821

EMAIL : callcenter@saeilo.co.jp

発行・文責：(株)セイロジャパン

2008 / 6 / 25