Ductpost User Guide

8th September 2006

Version 1490

Introduction

Introduction

(Up-dated 27/12/2000)

- DUCTpost is a fully flexible configurable post processor system. It allows a user with a single machine control licence to make small changes very simply, such as the number of decimal places, modifications to the start and end of tape output, for a similar control that requires minor variations of machine tape input.

- It is not necessary to spend time writing all the information for a new control, DUCTpost uses stored information for fifty of the most common machine controls, and these may be modified by writing an ***option*** file.

By typing: ***ductpost -w [control type] > [control type] . dmp*** on the command line of a Shell Window, DUCTpost will produce a dump to a file of the current settings for that particular control. Using this file as a base any of the lines may be modified and put in to an ***option*** file. It is ***only necessary*** to include in the ***option*** file the modified sections of the listing.

Note:- From Version DP1205 it is possible to obtain a list of the machine controls supported, by typing: ***ductpost -l*** (small L) and from this select the nearest base control to modify for your type of control output. (see Controls for details)

- The ***option*** file is written in simple English, and lines of the file may be given in any order, but it is sensible to try and maintain a logical progression of events. It is independent of the version of DUCTpost, so once an ***option*** file has been set up it will not normally need to be changed if a new version of DUCTpost becomes available. (From **Ductpost 1205** this will not be true in all cases, for if any internal change has been made that will affect an ***option*** the Post Processing will **halt**, and an *****error message***** produced, highlighting the change that will need to made to the option) It can also be transferred to any computer. (However, it is recommended that a check on the output is made between **OLD** and **NEW** to ensure compatibility has been maintained. Although every effort is made to ensure this is so, with the multitude of variations out in the field this cannot always be guaranteed.)

- Using the ***option*** file, any word or group of words, in the machining tape file may be given any format to the lines output for each part of the cutfile. The following parts (tape start, tool change, linear moves etc) may be defined, and global details such as whether or not block numbers, or messages are output can be set. It is even possible to segment long files of machining data into separate sequencial files of any length, or to insert special blocks or characters into the machining file after a certain number of blocks or tool travel distance.

- DUCTpost is also flexible in terms of file names. The NC program produced by DUCTpost by default is called ***jobname . tap***, but it can be ***jobname . nc*** or ***jobname . 01.*** It is also possible to write a setup sheet containing tooling information, though a standard one is provided, and optionally, a full print file containing absolute positions for incremental tapes. These facilities are controlled by a file called ***startup.*** (Versions of Ductpost earlier than DP1325 ignore the " ***startup*** " file from PowerMILL ***tape file output*** and a different proceedure is needed)

- The old format of *xyz..cfg* files are no longer supported and will need to be converted to use the " ***standard*** " format, or one of the other type of controls by converting them to an ***option*** file.

Ductpost tree structure

(Up-dated 27/12/2000)

Ductpost files are arranged in this form for versions prior to and including DP1100 series :-

```
                                    dcam
                      ______________|________________
                      |                              |
                   product                         config
                      |                              |
                 ductpostNNNN                     ductpost
                      |                              |
                      |                              |
            __________|_______________               |
            |                         |            . opt
           sys                       lib
     _______|_______________          |
     |          |           |        / \
   exec        misc        hci   . mcd  . cfg
     |          |           |
ductpost.exe  startup   ductpost.msg
```

Where NNNN is the DUCTpost version number.

Ductpost files are arranged in this form for versions DP1200 series and later :-

```
                                    dcam
                      ______________|______________
                      |                            |
                   product                       config
                      |                            |
                 ductpostNNNN                   ductpost
                      |                            |
                      |                            |
            __________|_______________             |
            |                         |           / \
           sys                       lib   startup  . opt
     _________|_____________          |
     |                      |        / \
   exec                    hci   . mcd  . cfg
     |                      |
ductpost.exe            ductpost.msg
```

Where NNNN is the DUCTpost version number.

The lib directory will need to be created for these, and any " cfg " or " mcd " files transfered from the older version into it.

NOTE : - This is applicable to UNIX platforms only.
NT platforms will require the " cfg " or " mcd " files to be placed in the WORKING directory until DP1320 release which will fix a bug and allow them to be inserted in the correct place.
It is essential to check that the customer is PAF'd correctly for DUCTPOST - XXXX - CONFIG for these files to work. [XXXX is year of issue]

The startup file and option file are independent of the version of ductpost and are set up to suit an individual customer so they are not part of the ductpost tree. Once a suitable version of the option file and startup file have been created they will not be overwritten if a new version of ductpost is installed.

If a startup file or option file exist in the working directory they will be used instead of the files in /dcam/config/ductpost

The Startup file

(Up-dated 27/12/2000)

The Startup file will normally reside in the following directory (**/dcam/config/ductpost/startup**).
However, it does not need to be present, and from DP1203 the Startup file is **not included**.

If it is required, a Startup file may have to be created, but if present, then it may contain the configuration details for default names and extensions for the tape, set-up and print files.

The order of lines in the Startup configuration file does not normally matter, nor do all of the files need to be mentioned if not required to be altered.   Normally no heading line is required if it is for one control output format, but it can be configured for several control options.

If **NO** Startup file is present then the default output will be as follows :-

Tape file = [***Output Cut-file name***] **. tap**
Set-up file = [***Output Cut-file name***] **. inf**
Print file = [***Output Cut-file name***] **. prt** (Depending on how it is envoked - see Running Ductpost)

The print file will not be created unless specifically envoked.

NOTE :- PowerMILL NC Tape output ignores the Startup file (See notes at end)

Startup file format :-

The general form of a line specifying a file default is

file part **=** ***output required***

file can be one of " **tape** " , " **setup** " , " **print** "

part can be " **file** " for the file name
" **extension**" for the file extension
" **dot** " for the separator

output required **none** (NO output produced) ,
or a string such as " **tty** " for an extension or name.
(" not usually required except as below)

Thus typical lines could be

setupfile = **" information "**
setupextension = **" "** (Must be " space ")
(NOTE **none** will output the extension as "**. none** ")
setup dot = **none**

This will give a ***setup file*** called **information** with no ***dot*** or ***extension***

The Startup file is independent of machine control or option file unless set with a machine control or option definition.

Example of a startup file :-

```
setup file = none            ( see note 1. )

machine f11m4x_AUT3   ( see note 2. )
tape extension  = "f11"
tape dot          = "*"
setup dot         = "*"
print extension = "pxt"      ( see note 3. )

machine heid                 ( see note 4. )
tape extension   = H
setup extension = " "        ( see note 5. )
setup dot          = *
setup file          = information
tape file           = HEID
cr                             ( see note 6. )
```

Explanation of above configuration :-

1/ This line is general and will overide anything similar below.
Its setting will suppress output of the setup file.
(# comment out, or delete if setup sheet required)

2/ This line specifically relates to a particular control, in this instance an Option File which can have any machine control as its operator. The following four lines will effect the output for this. (except if Line 1 is set, see note 1)
use :- ***ductpost f11m4x_AUT3 ltrbotl1***

3/ The print file is the CLdata output of the cut-file and the default extension is ***. prt*** To obtain a print file :-
use :- ***ductpost -p heid ltrbotl1*** ***(Output heid . prt)***
or :- ***ductpost heid ltrbotl1 -p*** ***(Output ltrbotl1 . prt)***

4/ This line specifically relates to the " ***heid***" machine control and will not react to any option file with a " **heid** " control.
use :- ***ductpost heid ltrbotl1***

5/ The use of a double quoted space string is essential in this and other extension suppression. Using " **none** " will produce the extension " ***.none*** ".

6/ The last line must be a blank produced by a carriage return from the proceeding entry.

(Return to Top)

PowerMILL and Startup :-

The Startup file was ignored by Post Processing for a **Tape output** from PowerMILL and all settings had to be made in the Selected Toolpath Output Form.

To set the tape extension to anything different to the default " **. tap** " it is neccessary to define the required extension in the Output File name box. e.g. ***ltrbotl1 . hd***

No set-up information file is produced unless the "**Produce info file** " flag is checked in the OptionFile box and will have the output as ***ltrbotl1 . inf***

To obtain a CLdata print file, (Only available from PM2507 under " **set preview**", and **enabled** from DP2510) set the Output for ***Cutter Location***, and select ***Print*** in the Format Box of Output Options box. The output file will reflect what is typed in the Output File name box. e.g. ***ltrbotl1 . pcl*** ***or*** ***ltrbotl1 . prt***

From **DP1320** the **Startup** file is now recognised and the configurations defined within it will be implemented for the Post Processor option.

Running DUCTpost

(Up-date 23/01/2001)

DUCTpost may be run in either of two modes :-

as a **standalone** programme by typing in a working Shell Window

ductpost [control name] [name cut-file]

where " **name** ". **cut** is the name of a cut-file produced by DUCT, or POWERMILL.

Ductpost will then produce an NC programme " **name**" **. tap** for the control.

Example

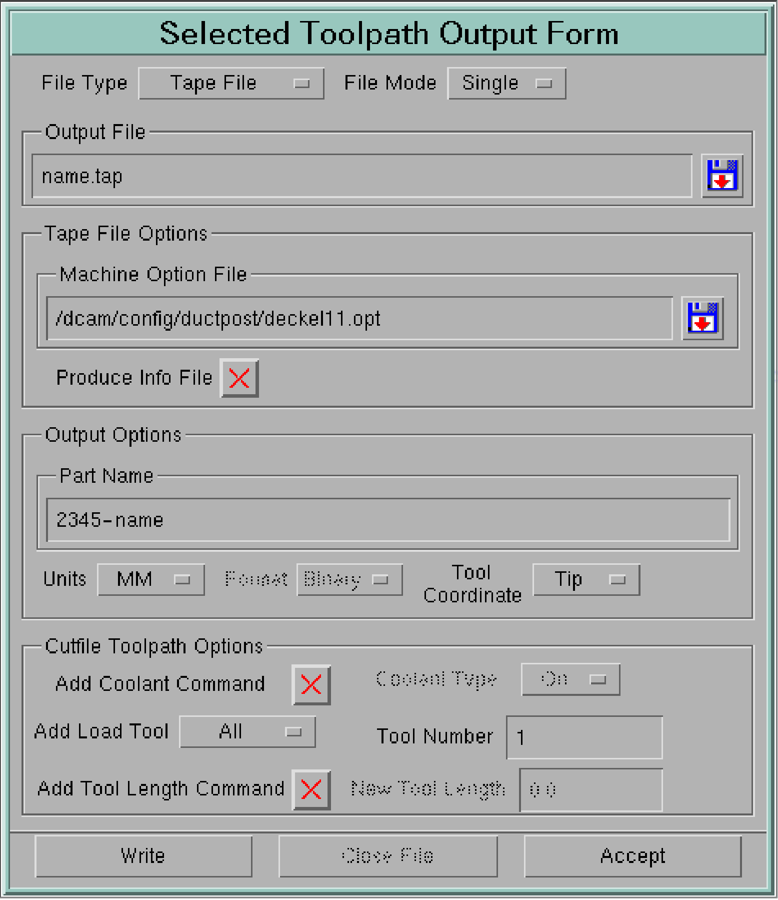
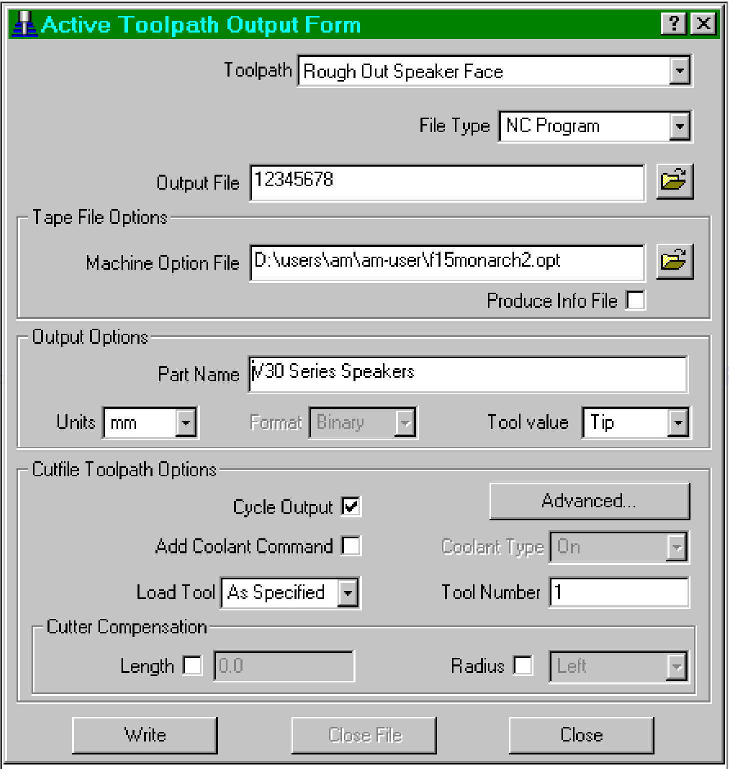
ductpost fanuc test

will post process the file *test.cut* , for the *fanuc* control, writing its output in *test.tap*

(see additional switches)

---

or, from within :-

PowerMill2XXX	PowerMill3XXX
	

Selected Toolpath Output Form

File Type: Tape File   File Mode: Single

Output File: name.tap

Tape File Options

Machine Option File: /dcam/config/ductpost/deckel11.opt

Produce Info File ☒

Output Options

Part Name: 2345-name

Units: MM   Format: Binary   Tool Coordinate: Tip

Cutfile Toolpath Options

Add Coolant Command ☒   Coolant Type: On

Add Load Tool: All   Tool Number: 1

Add Tool Length Command ☒   New Tool Length: 0.0

Active Toolpath Output Form

Toolpath: Rough Out Speaker Face

File Type: NC Program

Output File: 12345678

Tape File Options

Machine Option File: D:\users\am\am-user\f15monarch2.opt

Produce Info File □

Output Options

Part Name: V30 Series Speakers

Units: mm   Format: Binary   Tool value: Tip

Cutfile Toolpath Options

Cycle Output ☑

Add Coolant Command □   Coolant Type: On

Load Tool: As Specified   Tool Number: 1

Cutter Compensation

Length □ 0.0   Radius □ Left

PowerMill2XXX

Here, " **name** ". **tap** is entered (or just " **name** ") in the ***Output File Box***, the machine control Post Processor selection is entered in the ***Machine Option File Box***, and on completion of the rest of the parameters required, the file is ***Writen*****,** and Ductpost will then produce an NC programme " ***[name]*** " **. tap** . If a different extension to **. tap** is required then type this with the name. e.g. **name .** " **hd** "

No set-up information file is produced unless the " **Produce info file** " flag is checked in the OptionFile box and will have the output as *ltrbotl1* *. inf*

To obtain a CLdata print file, (Only available from PM2507 under " set preview ", and enabled from DP1208) set the Output for ***Cutter Location***, and select ***Print*** in the Format Box of Output Options box. The output file will reflect what is typed in the Output File name box. e.g. *ltrbotl1* *. pcl*

PowerMill3XXX

With the later releases of PowerMill the Output form has changed slightly as can be seen in the 2nd. Column.
However, the same principles apply as as before though this may change with later releases.

A series of default Post Processor control names are built into DUCTpost (for a full list see available controls) .

If a file named ***[control]*. opt** is present in the ***current*** directory, or **/dcam/config/ductpost** directory, then this will be read **first**, and the ***machine name*** at the head of this option file will be used as the **controller.** **However**, anything defined in this ***option*** file will be used instead of the built in default control configuration.

For details of this file see The Option File.

DUCTpost also reads a **message** file :

/dcam/product/ductpostxxxx/sys/hci/ductpost . msg

(where " **ductpostxxxx** " represents the DUCTpost version number). This contains all the messages which are used by DUCTpost, and may be modified if desired. (**Not recommended**)

DUCTpost may also use a startup file : (**/dcam/config/ductpost/startup**) to change the default tap-file names and extensions. This file will need to be created if it is required, but when using PowerMILL, especially with NT, the extension changes will probably be ignored. (This is has been fixed with DP1331)

DUCTpost will also produce a file called " **name** " **. inf** , which contains tooling details, and any comments that are in the cutfile. This output can be suppressed by means of the **startup** file if not required.

(Review this section Top)

Switches :-

Ductpost may be run in the **stand-alone** mode within an NT MSDos Window with some extra command line arguments.

ductpost fanuc test -t new.hnc -a

The possible switches are:

-a used if the cutfile is **ascii**.
-t ***name*** used to alter the NC programme name.
-i ***name*** used to alter the ***info*** file name.
-p Output a ***CLdata*** print file

ductpost -v

-v used to print out the DUCTpost version number.
-l (lower case L) used to print out a ***list*** of available controls
-w [*control*] used to write out the full default configuration for that control.

(Return to Top)

The Option File

(Up-dated 28/12/2000)

An optional file is used to re-format the standard in-built structure of the control post processor.
More than one option file may be used for each control.
The first line must contain the name of the control.
The option file will normally reside in the following directory :- **/dcam/config/ductpost/**

Examples

The standard Fidia control has NC programs with no coordinate decimal point, two decimal places accuracy, and no spaces.
The output looks like this :-

```
N10G00X1234Y3456Z2345
N20G01X2000Y3000
N30Z-2000
....
```

Another Fidia machine may require a decimal point, three decimal places and spaces.

```
N10 G00 X12.342 Y34.561 Z23.45
N20 G01 X20. Y30.
N30 Z-20.
....
```

To make these changes an option file would require the following :-

```
machine fidia

  define format (  X Y Z  )
    metric formats
    decimal point     = true
    decimal places  = 3
    trailing zeros     = false
  end define

 define format all
   tape position  = 1
 end define

end
```

Notice:- The first line is ' **machine control**' and the last line is ' **end** '. **These are the only fixed lines.**
One or more spaces may be used as indents in the option file, but **NO TABS.**
There must be a blank line after ' **end** '

Option file contents

The description consists of four sections

The format of each word and the word order.
The link between words and their meanings (keys and codes).
Integer, real, character and logical flags.
Description of blocks which correspond to tape start, tape end, tool change, circles, moves etc.

A typical option file might contain the lines

```
machine fanucom
```

```
   define format ( G1 M1 )
     leading zeros = true
   end define

   define format ( X Y Z I J K )
     metric formats
     decimal places = 4
   end define

   define keys
     spindle not used
     feedrate  = F
   end define

   block start        = 5
   block increment = 5

   define block tool change
     N ; T ; "M6"
     N ; Z FromZ
   end define

end
```

Where the first define block set the G and M codes to have leading zeros.
The coordinates to have 4 figures after the decimal point.
The third part means no S is output for a spindle speed but F is used for the feedrate.
Next the block line numbers start at 5 and go up in 5's,
The last part gives a tool change output of the form :-

```
N10T2M6
N15Z50.
```

For further details on option construction refere to Option Construction Tutorial.   (return to TOP)

The NC program

(Up-dated 10/10/2001)

An NC program normally consists of a series of blocks, each of which is an instruction to the machine tool.

Example

N30 X2.3 Y4.56

is :- **block number 30, move to point X2.3 Y4.56.**

A block consists of words, usually a letter followed by a number, and although most tapes have a single letter for each word, it is possible to have a double or more letters to a word. (e.g. I J and K are the standard circle centre letters, but one machine tool uses CX CY and CK).   The numbers have various formats, most commonly, coordinates are written with decimal points, whereas other words do not.

As well as coordinates, other information is given in NC output, typically :-

Block number	N.....e.g. N25
Feedrate	F.... e.g. F300 or F300.
Spindle speed	S.....e.g. S850
Tool Number	T.....e.g. T2

Machine functions (aux functions) :-

Machine stop	M0
Spindle on clockwise	M3
Spindle on anticlockw.	M4
Spindle off	M5
Tool change	M6
Coolant on	M8
Coolant off	M9
End of tape	M2
End of prog	M30

Preparatory functions :-

Rapid linear move	G0
Feedrate linear move	G1
Circle clockwise	G2
Circle anticlockwise	G3
Compensation on	G41
Compensation off	G40
Metric coordinates	G20 or G70
Inch coordinates	G21 or G71
Absolute coordinates	G90
Incremental coordinates	G91

Formats

Words with decimal points usually have a maximum number of figures after the decimal point (usually 3 for metric, 4 for inch). Words may have leading or trailing zeros.

e.g. X3.45   X3450   X000345   X3.450   X0003.45

may all be used for 3.45

M and **G** codes generally have a maximum of two figures (e.g. **G01** or **G1** ; **M03** or **M3**)

Tape Example

```
%                              tape start
:0001                          program number
N10G28G91X0Y0Z0                Move to start position (this is special for Fanuc )
N30T1M6                        tool change
N40G0G90X-25.Y-40.S800 M3   Rapid move to X-25 Y-40  absolute coordinates, spindle on
N50G43Z3.H1M8                  tool length compensation, coolant on , move down
N60G1Z-5.F100                  feedrate move to workplane
N70G41Y40.D1F200               turn on cutter compensation
N80X-5.
N90G2G17X25.Y10.I0J-30.    circular move to X25  Y10, centre point I0  J-30  from start of arc
N100G1Y-10.
N110G2X-5.Y-40.I-30.J0
N120G1X-25.
N130G0G40Z20.                  cutter compensation off

N140G28G91Z0                  return to reference point
N150G49H0                     tool length compensation off
N300G28X0Y0                   return to reference point in XY
N310T3M6                      tool change
N320G0G54G90X100.Y-35.S1500M3
N330G43Z3.H3
N340G1Z1.F40
N350G83G99Z-10.R1.Q3.66F150    deep drilling cycle
N360G28G91Z0                  return to reference point in Z
N370G49H0                     turn off tool length compensation
N380G28X0Y0                   return to reference point in XY
N390M30                       end of prog
```

(Return to TOP)

Tape start and end

(Up-dated 23/01/2001)

The characters and codes at the **start** and **end** of an NC tape can vary between different machines, even with the same type of control.

Many controls require a program number, or program name, at the beginning of the tape.
This can be ***requested*** at post processor run time when the **name,** or **number,** will be entered by the programmer in response to a **prompt**, this will usually be done outside of PowerMill in a DOS Window using a cut file. (Prompts are ignored by PowerMill NC Post Processing and tend to cause the process to hang)

This data can come from the Duct *APT PARTNO* statement, ***Output File Name*** or ***Part Name*** from PowerMill's **Active Tool Path Output Form,** and a fixed ***Part Number*** which can be defined in the Post Processor Option file, other than the default of **1**.

Codes and data required for **start** and **end** of an NC tape are defined in the blocks ***tape start*** and ***tape end***.
Anything which appears in these blocks will be output once only.
Characters and Codes which are unique and not supported via the Post Processor are written as a string in quotes :-

```
"%"
"( MAZAK VC200 POST )"
```

Other Codes and Data can usually be handled using a variety of Variables such as these listed here :-

Variable	Function
FromX, FromY, FromZ	To output the X,Y,Z values of "***from***" (safe start) position
JobName	To output the Tool Path File Name
TapeUnits	To output code for metric / imperial (usually G70/G71 or G20/G21)
TapeCoords	To output code for absolute / incremental (usually G90/G91)
PartID	To output Program name (Part Name from PowerMill)
ProgID	To output Program number (Usually 1 - default, or defined from Option)

or the relative Code word from the define codes list.

Example :-

```
#
                define block tape start
                  "%"
                  "( MAZAK VC200 POST )"
                  ID  ProgID          or    ID PartID       or    ID JobName
                  N  ; G5 TapeCoords  ;  G4 TapeUnits  ;  xy plane ; compensation off
                end define
#
                define block tape end
                  N  ;  end of prog
                  N  ;  program end
                  "%"
                end define
```

This would produce an NC output something like this :-

```
%
( MAZAK VC200 POST )
 O0001   ## ProgID   or   Oplan    ## PartID   or    Oxyzx    ## JobName
 N10 G40 G17 G21 G90       ## Output as defined by Word order list
 ::::::::::::::::::::::::::::::::::::::::::::::::::
```

```
N7620 G0 Z10.
N7630 M02       ## end of tape
N7640 M30       ## end of prog
%
```

Message File

(Up dated 28/12/2000)

Ductpost contains **messages** which are written in files and printed while ductpost is running. These messages may be modified or translated as required.

This may be useful if you wish to translate messages into another language which are output as the program is running (e.g. " Enter program number " to " Entrer le numero du programme ") or, to format the **.inf** file.

The Most used strings from message file which are most likely to need translating :-

```
102 "Enter program number :"
103 "Enter cutfile name :"
109 "Incomplete cldata file"
111 "Enter part number :"
127 "Problem opening files"
172 "Error(getrec): Could not determine Cldata precision"
186 " x too small in block %(I10)"
187 " x too large in block %(I10)"
188 " y too small in block %(I10)"
189 " y too large in block %(I10)"
190 " z too small in block %(I10)"
191 " z too large in block %(I10)"
192 " Control not available "
```

In order to make any modifications the file **/dcam/product/ductpostNNNN/sys/hci/english.msg** should be copied to **/dcam/product/ductpostNNNN/sys/hci/ductpost.msg**.

The file **ductpost.msg** will then be used for messages instead of the default messages.

Within the file each message is set out as a number followed by the **formatted message string in double quotes**. If the message needs more than one line subsequent lines may be added using **+ followed again by a string in double quotes :-**

Example :-

```
176 "---------------------------------------------------------------------------"
+ "------------------------------ Setup Sheet ------------------------------"
+ "---------------------------------------------------------------------------"
```

The format of each CLdata variable is controlled by the message file :-

%(I10) means a 10 digit integer.
%(F5.1) a five digit real with one figure after the decimal point.
%(A10) a ten character string.
%(A-10) Strings may be left justified by writing.

It is **not possible** to alter the variables which are written, only their format.

An Example of the Strings for the information file output are :-

```
175 " ---------------------------------------------------------------------------"
+ " -----------------------        Setup Sheet        -----------------------"
+ " ---------------------------------------------------------------------------"
+ " "
+ " ---------------------------------------------------------------------------"
+ " |   Date : %(A-11)                                                        |"
+ " |   Time : %(A-11)                                                        |"
+ " |   Programmer : %(A-11)                                                  |"
+ " |    Machine : %(A-11)                                      |"
+ " |--------------------------------------------------------------------------|"
+ " | Block Tool Number  Length     Radius     Comments                  |"
+ " |--------------------------------------------------------------------------|"
```

(where Date and Time are followed by the date and time, Programmer by the current login name, Machine by the machine name from the option file)

```
176 " |  |         |         |         |%(A-35)| "       (Comment from DUCT pprint statement)
177 " |  |         |         |         |                                |"
178 " |%(I5)|%(I2) (%(I2) %(I2))|  %(F5.1)   |  %(F5.1)  |                              |"
       ( blocknumber, tool number, radius offset, length offset, length, radius)
179 " |----------------------------------------------------------------------|"
180 " ==================================================================="
+ " |                                                       |"
+ " |        Maximum           Minimum                               |"
+ " |                                                       |"
+ " |    X  %     (F10.3)        %(F10.3)                        |"
+ " |                                                       |"
+ " |    Y  %     (F10.3)        %(F10.3)                        |"
+ " |                                                       |"
+ " |    Z  %     (F10.3)        %(F10.3)                        |"
+ " |                                                       |"
+ " ===================================================================="
     ( Maximum and minimum values of X Y and Z from the cutfile )
193 " |     |         |         |         | Tip radius    %     (F10.3)          |"      (Tip radius)
194 " |     |         |         |         | Feedrate      %     (F10.3)           |"    (Feedrate)
195 " |     |         |         |         | Plunge Rate   %     (F10.3)           |"  (Plunge rate)
196 " |     |         |         |         | Safe Z        %     (F10.3)          |"     (Initial Z position))
197 " |     |         |         |         | Rapid Height  %     (F10.3)          |"   (Z value after first Z move))
198 " |     |         |         |         | Spindle speed %     (F10.3)          |"  (Spindle speed)
199 " |    Part number                   |%     (A-35)  | "                    (Part Number)
```

Word formats

(Up-dated 10/10/2001)

- A line of an NC program contains a set of words : e.g **G1 X... Y... Z... F...** etc.

Each of these words may be described in detail using the formating available in an option file. Words may be individually described, or the formats of a group of several words may be described together.

It is only necessary to list the formats which are to be changed from the original definition.

The formats which can be altered are listed below:

- ***Each format description must start with***

define format (....)

and end with

end define

The statement ***define format*** should be followed by the name(s) of one, or more, words in the brackets, separated by spaces.

Examples :-

```
define format (  X  Y  Z  )
     field width      = 8
     leading zeros = false
    decimal point = true
 end define

 define format (  G  M  )
    leading zeros = true
    decimal point = false
 end define
```

This will give the **X Y Z** coordinates output in the form :-

X3.123 Y78.9 Z400.

the G and **M** codes in the form :-

G01 G90 M03 M10

If all words are to have a particular format, the following syntax is used :-

```
 define format all
     tape postion = 0
  end define
```

This will produce **NO** spaces between words on the tape. (For a further information on this see the document on spacing.)
(It is common to have one space between each word, as the tape output is easier to read, but this will increase the tape length, or file size.)

Possible format definitions

Format given to word	Example of use	Brief Explanation
	address letter =	

address letter	**"MSG, "**	The word begins **MSG,**
address width	address width = **5**	4 characters, including a space, ***or*** 5 characters
field width	field width = **8**	Up to 8 digits **ddddd.ddd**including point, ***or*** 8 characters. (**125 max.**)
tape position	tape position = **1**	1 space before the word
scale factor, ***or*** scale divisor	scale factor = **2** ***or*** scale divisor = **4**	Multiply value by 2, ***or*** Divide value by 4
sign	sign = **if negative**	negative sign output only
modal, ***or*** not modal	**modal**	see notes below
permanent, ***or*** not permanent	**not permanent**	see notes below
metric formats, ***and/or*** imperial formats	**metric formats**	The following formats are metric ***and/or*** imperial
decimal places	decimal places = **3**	3 figures after point
decimal point	decimal point = **false**	no decimal point required
leading zeros	leading zeros = **true**	see notes below
trailing zeros	trailing zeros = **true**	see notes below
exponent width	exponent width = **2**	**X3.45E+04**

Notes :-

address letter	Normally the value is one letter, "**X**" or "**G**" , but can be up to a maximum of **19 characters**.
address width	Can be anything from **0** to **19** maximum
field width	Can be anything from **0** to **125** maximum
tape position = 1	This will put one space before each word on the tape:- e.g. N10 G01 X2.345 Y4.56 Z0.567 F600 (Default could be ***tape position = 0*** *NO* ***spaces)***
scale factor or ***scale divisor***	Normally the scale factor and divisor are both **= 1** (one), but a requirement may be needed to multiply, or divide the value of the word, or even reverse the sign. (**Remember all values are integer**) (***scale factor = -1***)
sign	**Use:-sign = if negative** for a coordinate where only negative values are signed, **sign = always** if **+** / **-** signs are required. **sign = none** for G codes and feedrates where a sign is not required.
modal *or* ***not modal***	A word is **modal** if it only needs to be repeated when it has changed. Normally G codes and X, Y and Z coordinates are **modal**, but I, J, K codes for circle centres are usually not, and are therefore **not modal**
permanent	Permanent needs to be used with care as the word will only be output if there is any other change in the values of words on the same line it appears on.

	The most common example is the block number (**N**). The word **must be defined** in all the block definition lines it is to appear in the NC tape output. Note :- For " heidiso " ***every block*** is used for the " * " for the EOB.
metric formats ***or*** ***imperial formats***	It is sometimes necessary to have different formats for metric and inch coordinates (decimal places for example). The default is **metric formats** but it will depend on the input CLdata. as to the output See **Integers, reals and characters**
leading zeros ***and*** ***trailing zeros***	Leading or trailing zeros may be used to imply the position of the decimal point if no point is used. In this case check that the field width and decimal places value are correctly set. (e.g. the output of **345.1** would be **00345100**)
exponent width	The default of this is zero, and therefore it has no effect. However, if a value is defined, then the word is output using exponential formatting. (see above)

Full Example:-

(NOTE : Indents must be spaces, **NOT tabs**)

```
define format ( X )
    address letter    = "X"
    address width     = 1
    field width       = 8
    tape position     = 1
    scale factor      = 1
    scale divisor     = 1
    sign              = always
    modal
    metric formats
    decimal places = 3
    decimal point   = true
    leading zeros   = false
    trailing zeros  = false
    imperial formats
    decimal places = 4
    decimal point   = true
    leading zeros   = false
    trailing zeros  = false
  end define
```

In this case **X** will appear as **X+3.456** on a metric tape, and as **X+3.4562** on an inch (imperial) tape.

```
define format ( G1 )
    address letter   = "G"
    address width    = 1
    field width      = 2
    sign             = none
    not modal
    metric formats
    decimal places = 0
    decimal point   = false
    leading zeros   = true
    trailing zeros  = true
```

```
      imperial formats = metric formats
    end define
```

In this case **G1** will appear as **G01** on both metric, and inch (imperial) tapes.

(Return to Top)

Spaces on tape

(Up dated 31/01/2001)

Most controls will allow spaces on tapes although it may not be desirable since it increases tape length, it is however, more readable.

The number of spaces between words is defined using the format statement " ***tape position*** " for NC tape file, and " ***print position*** " for the print file.

If a **fixed** position is required a ***negative*** value may be used, however, this must be used cautiously to avoid overwriting previous words.

Example

```
define format ( X )
   tape position  = 2          ( e.g.   N123^^X345.678 )
   print position = -20        ( e.g.   N123^^^^^^^^^^^^^^^X345.678 )
end define
```

This will give **two** spaces between the word **X** and the previous word on the tape, and the word X will start in **column 20** in the print file.
Conversely, if the tape position is defined ***negative 20*** the word will start in **column 20** in the NC tape file output, and subsiquent words following will be normally spaced.

```
define format all
   tape position = 1
end define
```

This will give **one space** between all words on the tape.

```
define format all
   tape position = 0
end define
```

This will give **no spaces** on the the tape.

See also Formats

Some Typical Integers, Reals and Characters

(Up-dated 08/09/2006)

A number of flags may be set within the configuration file which either provide values the post processor uses, or control the way in which the post processor generates output. The general form for setting a flag is :-

flag name = value

The value should match the flag type, otherwise an **error message** is produced and the post processor will **STOP**, or the assignment may be ignored.

For example :-

```
block increment      = 10
maximum feedrate     = 1000.    # note the point
message output       = true
```

are correct, but **block increment = 10.** *or* **maximum feedrate = 1000** are not, and will produce an **error message**.

The flags of each type and their function are :-

Floating point :-

arc radius limit = n.	**Limits the Maximum Arc Radius** output to the NC tape compatible with the machine tool maximum capacity. (Was **real 16** prior to DP1335)
arc minimum radius = n.	**Limits the Minimum Arc Radius** output to the NC tape compatible with the machine tool minimum capacity. (New DP1357)
diameter = n.	This is no longer used and was defined for DUCT Lathe operation. M/c specific parameter (Linear = Pi * dia * A / 360)
maximum feedrate = n.	**Maximum allowable feedrate** - this is the Linear (G1) feedrate and any Plunge/Cutting/Rapid Skim feedrate set in excess of this is reduced to this value.
minimum feedrate = n.	**Minimum allowable feedrate** - Any feedrate set less than this value is reset to this minimum.
maximum segment = n.	**Defines where an insertion is to be made in an NC tape file** in feet. (See Segmentation for more information)
maximum tape length = n.	**Maximum length of a single section of an NC program** Default is **unlimited** (**set to 0**) - if the program output size is too large and needs to be limited to manageable sized files, then the value set here will split the program in to separate files.
rapid feedrate = n.	**Maximum rapid feedrate** - this the NON-Linear (G0) feedrate and is used in calculating the machining time. Should be set to the Machine Maximum.
retraction threshhold angle = n.	**Used for test purposes only** (Default is 360.0 - ***DO NOT ALTER AT ALL***)
plunge threshold angle = n.	**The inclined angle** at which a plunge move can be defined and be considered as a plunge move. (Default is 0.0 Vertical - *USE WITH EXTREME CARE*)
withdrawal amount = n.	**Distance tool moves off the job** at a reconfiguration of the rotary axes

Note :- All the above require a ' **.** ' (point)

(Back to Logical Flags)

Integer Variables :-

block increment	The NC block (line) number increment
block start	The value used for the first NC block (line) number
counter increment	The amount by which the variable " **counter**" is incremented.
counter start	The starting value for the variable " **counter**"
cycle output	Used to determine the type of output for canned cycles. (*Not very reliable - use PowerMill selection*)
integer 69	Used to provide plunge feed rate for tapping (default = **0** is 85% approx. = **2** actual defined plunge feed rate)

maximum block number *	The maximum block (line) number that will be output before resetting to 0.
maximum tape blocks *	The maximum number of blocks (lines) on the tape.
minimum tape blocks *	The minimum number of blocks (lines) on the tape.
program id start	The default value of the **progid**. (Usual default = 1)
segment type	Criteria for splitting, or segmenting a tape 0 -- length in feet of tape 1 -- distance tool has traveled 2 -- number of blocks 3 -- number of characters
special zero	If this is set to **2** , then output a value of zero using the special zero defined by ' **zero = "0.0"** ' To all intents and purposes this can be ignored and just redefine ' **zero = "0.0"** ' to what's wanted
split move	Criteria for splitting move 0 -- do not split XYZ moves (**Always for multi-axes**) 1 -- split moves into "XY" and "Z" components
tape split retract distance *	The distance the tool retracts from **off** the job when the tape splits.
tool reset coordinates	Sets criteria for moves following tool change (**Should be 0 for Multi-axes Tool Paths**) 0 -- No special action. Output depends on CLdata input 1 -- Output X and Y co-ordinates. 2 -- Output X, Y and Z co-ordinates. 3 -- Output X, Y on one line, Z on next 4 -- Output Z on one line, then X Y on next
workplane angle convention	1 (Default) to 24 Determines which Workplane angular rotation convention is used by the post processor. (See What's New in Ductpost 1430)

For those items marked *, see the section on tape splitting for a fuller discussion.

(BackTo : Top Integer Variables Strings)

Logical flags :-

block order	**= false** Output words etc. in the order they are defined by " *word order* " listing in the Post source file. **= *true*** Output words etc. in the order they appear in the block definition of the Post Option file.
full circle	**= true** Output full circle definition in circular moves. **= *false*** Output linear moves for circles
go home output	**= true** Output go home moves. **= *false*** Suppress go home moves.
incremental centre +	**= false** Output I, J, K coordinates representing ***absolute*** centre of circle. **= *true*** Output I, J, K coordintes representing ***incremental*** distance between the start point and the centre.
message output	**= true** Output cldata messages to the tape file. **= *false*** Suppress cldata messages to the tape file.
retract at angular limit ++	**= false** Multi-axes retraction and reconfiguration will not be implemented and posting will stop with an error message if angular and linear limits are exceeded in the tool path. **= true** Retraction and reconfiguration will be enabled if possible by inserting extra moves over and above those shown in the PowerMill tool path.
spindle w motion **spindle x motion** **spindle y motion** **spindle z motion**	**= false** Spindle aligns with the W axis **= false** Spindle aligns with the X axis **= false** Spindle aligns with the Y axis **= true** Spindle aligns with the Z axis

spindle azimuth rotation ++ spindle elevation rotation ++	= **false** Rotary axis is a **table** unit = *true* Rotary axis is **tool head** unit
tlo output	= **false** Suppress Tool length offset function code on a tool change. = *true* Output Tool length offset function code on a tool change.
use partid	= **true** The part name from the cut file is used. = *false* Prompt for a part id from the keyboard. (NOT from P'Mill)
use progid	= **true** Then use the default (1), or defined program number. = *false* Prompt for the prog id from the keyboard. (NOT from P'Mill)
tape split on tool change +++	= **false** Tape output normal = *true* Tape output with a series of tool changes will split at the tool change position. (Use for sub-routine programme output)
	(usually the default) : *(Alternative setting)*

For those items marked with a " +", see the section on circle output for more detail.
For those items marked with a " ++", see the section on multi-axes files for more detail. (4 axes - 5 axes)
For those items marked with a " +++", see the section on Sub-routines for more detail. (Still to be writen)

(Back to Top ; Integer Variables ; Logical Flags)

Strings :-

machine name	Example " Fanuc6m version 1.2 " Printed as postprocessing starts to Punched Tape
print header	Example " Delcam Postprocessor " Printed as postprocessing starts to Punched Tape
point	Example " **.** " Character to use as decimal point
zero	Example " **0.0** " Character string to use for outputting a floating point zero

Others :-

azimuth axis direction *and* **elevation axis direction**	= **positive** (Default) Direction of rotation clockwise as viewed from the positive end about axis of rotation. = ***negative*** Reversal of rotation (No longer viable, MUST always be ***positive,*** use the `“azimuth/elevation axis parameters ” by negating the axis 1 to -1`)
azimuth axis units *and* **elevation axis units**	= **degrees** Measure the rotary angle in degrees. = ***linear units*** Measure the rotary angle in linear units. (Used in conjunction with **diameter** parameter)
coordinates	= **absolute** The coordinates are Output in absolute values. = ***incremental*** The coordinates are Output in incremental values.
spline type +	= **bspline** Output spline data in B-spline format. = **polynomial** Output spline data in polynomial format.
units	= **metric** The tape is ouput in metric units (even if the cutfile is in inches). = ***imperial*** The tape is output in inch units (even if the cutfile is metric). = ***input*** The tape units are the same as the cutfile units. (**USE this setting for preference**) **(Warning some constants may not be converted correctly if input is different to default setting)**
	(usually the default) : ***(Alternative setting)*** : **(+ M/c specific)**

NOTE :- If the machine code for **absolute (normally G90)** is written on the tape by a block definition, the tape will be read as ***absolute*** by the machine control even if coordinates are output as ***incremental***.
For further information refer also to Integer List .

(Back to : Top ; Integer Variables ; Logical Flags ; Strings etc.)

Keys and Codes

(Up-dated 13/08/2002)

(Goto Codes)

Keys :-

The keys provide the link between the formats which have been defined and the words which are used by the postprocessor.

For example the " **x coordinate** " is usually associated with the word X. This is defined using the syntax

```
define keys
    x coordinate  =  X
end define
```

Normally keys rarely need to be changed from their default settings, so " **define keys** " is not often used in the option file.

However, in some circumstances it may be neccesary to change the ***function*** of a key(s), such as switching the axes (e.g. Y to Z and Z to Y for example).

It may also be useful to indicate that a word is **not used** for a particular function. (e.g. **spindle = not used**)

Important :- The **group name** is used to define the key, ***not the letter*** (although these are often the same). Thus **auxfun = M1** ***not*** **auxfun = M**

The group names can be found in the source file.

Some key and code names may be shortened, so " **x coordinate** " may be written " **x coord** "

It is **recomended** that the key, or code, definition is used in the option instead of the key, or code, letter (e.g. ***key i*** instead of **I** : ***tool length offset*** instead of **G6 43**)

Example :-

```
define keys
    z coordinate   =   Y
    y coordinate   =   X
    x coordinate   =   Z
    feedrate           not used
  end define
```

This exchanges the normal words for the coordinates and states that **NO** feedrate is output.

Full list :-

aux function	**azimuth axis**	**blocknumber**	**clearplane**
cycle dwell	**circle angle**	**drill hole depth**	**drill peck depth**
dwell	**elevation axis**	**feedrate**	**feed per rev**
key i	**key j**	**key k**	**leader**
message end	**message start**	**opskip**	**preparatory function**
program id	**radius**	**spindle**	**tool length**
tool length offset	**tool number**	**tool radius offset**	**x coordinate**
y coordinate	**z coordinate**	**x feedrate**	**y feedrate**

z feedrate	**x vector**	**y vector**	**z vector**
3rd rotation axis	**error**		

(Return to TOP)

Codes

A **code** is a predefined item **whose value** does not change and resides in the ***codes definition section*** (which is similar to the **keys definition section** above).

They are used to output machine control codes onto the tape, which usually have the standard G and M address letters.

Each code needs to be associated with an output word and value. (e.g. **G1 0** for a ***rapid*** move code output (**G0**).

These are defined as follows :-

```
  define codes
      function name  = word format label   function code value
   end define
```

Example :-

```
   define codes
      rapid             = G1 0
      linear            = G1 1
      comp on left      = G2 41
      comp on right      = G2 42
      comp off          = G2 40
      spindle on cw     = M1 3
      coolant on        = M2 8
   end define
```

G1 and *G2* are standard group names for **G codes**, *M1* and *M2* for **M codes**.

There may be more than one G or M code per line, so several groups are required for **G and M codes**, hence *G1 G2*.

This will give codes *G0*, *G1* for **rapid** and **linear** moves, and *G41*, *G42*, *G40* for cutter compensation.

(**Note :-** It is not permissable to have ***two G codes*** of the same group on a line [e.g. *G2 40* ; G3 17 ; *G2 80*] as G2 80 will overwrite G2 40)

Full list :-

The following function *code names* are understood by the postprocessor and will output the appropriate *G code* if set in the source code, or defined in an option file :-

absolute data	**bore 1**	**bore 2**	**bore 3**
bore 4	**bore 5**	**break chip**	**change tool**
circle ccw	**circle cw**	**clamp off**	**clamp on**
compensation off	**compensation on left**	**compensation on right**	**constant surface speed**
coolant off	**coolant on**	**coolant on flood**	**coolant on mist**

coolant on tap

cycle retract

deep drill

drill

dwell

end of drill

end of prog

end of tape

feedrate per minute

feedrate per rev

from

gear range 1

gear range 2

gear range 3

imperial data

incremental data

linear

metric data

opt stop

rapid

spin coolant off

spin cool on ccw

spin cool on cw

spindle on cc

spindle on cw

spindle off

spindle rpm

spline

stop

tap

tool length offset

xy plane

xz plane

zy plane

(Return to TOP)

Block definitions

(Up-dated 29/12/2000)

A block is a line or set of lines appearing in the NC program for a particular part of the cut file, for example the tape start, a linear move, a drilling cycle or the tape end. (See list of block names)

For each part of the NC program it is possible to **define a block**, consisting of several words that relate to their normal values (e.g. " **X**" will normally be the **x coordinate**, " **F** " the **feedrate** etc.).   These can be set a fixed value (e.g. F 9999) or to use a variable (e.g. X OldX which will output X with its **last value**).

The define block ..... consists of a list of possible words for each output NC line, each word separated by a semi-colon.

If more than one line is used the words will come on separate lines in the NC program unless the line has a semi-colon at the end.

Lines, or parts of lines, may also be defined using strings in double quotes. (e.g. "G90G70")

All blocks have the form :-

```
define block .....
    .......
 end define
```

Example :-

```
define block tape start
    "%"
    N  ;  ID PartID  ;  ")"
end define

define block tool change
    N  ;  T  ;  M1 6
    N  ;  S ToolSpeed  ;  M1 3
   N  ;  G5 90  ;  G6 54  ;
 end define

define block move rapid
    N  ; G1  ; X  ; Y  ; Z  ; F 5000
end define

define block move linear
    N  ; G1  ; X  ; Y  ; Z  ; F
end define
```

These give an NC program output of the form :-

```
%
N1 (PART NAME- Example Output)
N2 T1 M6
N3 S2500 M3
N3 G0 G90  G54  X... Y... Z... F5000
N4 G1  X....  Y....  Z..... F250
```

The blocks may be defined in any order, but it is recommended that a logical progression of functions is attempted as this makes the job of modification and debugging easier.

Words :-

The majority of the **words** used are pre-defined in the particular control source file and will have a designated output

letter and format.
To view these for a control it will be necessary to dump them to a Shell Window, or to a text file, by typing ***ductpost - w [control name]*** to dump to the shell, or ***ductpost - w [control name] > [control name] . dmp*** to dump to a text file. (*View using any plain text editor*)
(A list is being built here but only indicates the words and designated letter used for each control but no format data, see Word List)
(Word formatting can be seen here, indicating how the word is formatted.)

Words such as " **X Y Z I J K F** " etc. will have their own **individual** output function, (e.g. X1.234 Y5.678 Z9.0 F250) whereas **G** and **M** are **group** words having the same letter. (e.g. G1 G2 G3 all have "G", M1 M2 have "M" - Heidenhain are an exception)

The above example illustrates this where **G5 90 ; G6 54** output **G90 G54** and **G1** can output **G1 G0 G2 G3** depending whether or not it is a rapid, linear or circular move. Similarly **M1** can output **M0 M1 M3 M4 M54** etc., as can **M2** depending on their defined output function in the **define codes** list

(Back to Example)

Values :-

If a fixed output value is required this can be put in the block with a word :-

F 9999 will be output as **F9999**

NOTE :- NO decimal points are possible with fixed values so if **X1.234** is required, the entry has to be **X 1234** for metric, and **X 12340** for imperial

If a value needs to be repeated in subsequent output lines, even if it is the same as the previous value of that word, it is written as :-

F =C

Variables :-

Words may also be output with a ***variable***. This is may be used if the word is not to appear with its **usual** value, for example in the tool change block the usual form would be :-

```
define block tool change
   T  ;  M1 6
end define
```

to give an output of **T2 M6** for the second tool.
However another machine may require to preselect the next tool in which case we can use :-

```
define block tool change
   T  ;  M1 6
   T  NextTool
end define
```

giving an output of **T2 M6** for the 2nd. tool loaded
and **T3** for the NEXT tool to be loaded
(This is often used to load the next tool into an automatic tool changer while machining continues with the current tool).

Some Examples :-

These blocks are used to head a second, and subsequent, files when a long NC program is split into separate files :-

```
define block tape split start
   ID ProgID
   N ; " G90 G70"
end define

define block tape split move
   N  ;  G1 0  ;  X OldX  ;  Y OldY
   N  ;  G1 =C  ;  Z StartZ
   N  ;  G1 1  ;  Z OldZ
end define
```

ProgID is used to give the original program number. (e.g. MPF0001)
OldX, OldY and **OldZ** the **previous** **X, Y** and **Z** positions.
StartZ the clear plane position.

(For other variables see Block variables.)

(Back to TOP)

Obtaining a dump of a DUCTpost control configurations

(Up-dated 13/08/2002)

It is sometimes necessary to examine the full configuration for a control used by DUCTpost. DUCTpost will provide a ***dump*** of the full configuration when invoked using the " **-w** " switch option. (The dump is output to a MSDos window, but may be redirected to a file which is probably more useful for reference purposes, and can be retained in a directory for future use).

For example, to obtain a ***dump*** of the internal configuration of the built-in control " **heid400** ", type in a MS Command Window :-

```
ductpost^-w^heid400^>^heid400.dmp ( ^ = space )
```

This will place the file in your working directory.

A ***dump*** can also be obtained from an ***option*** file

```
ductpost^-w^DMU60PT-EMetV2.opt^>^DMU60PT-EMetV2.dmp ( ^ = space )
```

(the ***full*** configuration **incorporating** the *option* changes will be dumped), just as for any built-in configuration. (This is useful if you have added new words and require to know there list order, or to compare against the original source control to see what changes have been made.)

To obtain a list of machine controls supported by DUCTpost, use the **" -l "** (lowercase L) switch to list the controls.
i.e. type :-

```
ductpost -l    ( lowercase L )
```

Block Numbers

(Up-dated 13/08/2002)

Blocknumbers are used to define the number on a line of tape, they could start at 1 and increment by 1, as indicated below :-

```
%  O1234
N1 G99 M6 T1
N2 G0 X0 Y0 S1850 M3 PA10
N3 Z10.
N4 G30
N5 G0 X10.118 Y-36.377
```

However, this is not always required, and various exceptions with the method to change the output is given below:-

---

Example 1

If NO block numbers are required, then redefine the block number key " **N** ".

```
define format ( N )
     not permanent
  end define
```

This will remove block numbers from the tape :-

```
% O1234
G99 M6 T1
G0 X0 Y0 S1850 M3 PA10
Z10.
G30
G0 X10.118 Y-36.377
```

If the **inbuilt** post processor produces NO line number output, then format " **N** " to be " **permanent.**"

Example 2

If the numbering sequence requires to be changed, then the following needs to be inserted in the option file :-

```
block start          = 10
block increment  = 5
```

This will give a start block of 10, the block numbers will go up in 5 's.

Example 3

If the maximum number of lines a machine control can handle is limited, then insert in the option file :-

maximum block number = 5999 *(or whatever value required)*

and the tape line numbers will restart at the block start number after block **5999**.

Example 4

If a special block number is required to perhaps indicate a particular function,

```
N1000 BEGIN PGM SPECIAL MM
```

```
N1000 TOOL CALL 0 Z S3000
N1000 M55
N1000 M3
N1000 CYCL DEF 19.0 BEARBEITUNGSEBENE
N1000 CYCL DEF 19.1 A0 B0 C0
N10 L X0.0 Y0.0 Z150.0 B0 C0 FMAX
N12 L X254.345 Y146.780 B90.0 C35.250 FMAX
N14 L Z-55.70 FMAX
N16 L ..........................
```

then the following is a suggestion.

```
define word NF
    address letter  = "N1000"
    address width = 5
    field width      = 0
    permanent
  end define

  word order = ( + NF )

  define block tape start
    NF  ;  " BEGIN PGM"  ;  ID PartID  ;  metric data
  end define

  define block tool change first
    NF  ;  T2  0  ;  "  Z "  ;  S 3000
    NF  ;  M1 55
    NF  ;  M1 3
    NF  ;  G4 190  ;  " BEARBEITUNGSEBENE "
    NF  ;  G4 191  ;  A  0  ;  B  0  ;  C  0
    N  ;  G1  ;  X FromX  ;  Y  FromY  ;  Z  FromZ  ;  B =C  ;  C =C  ;  FMAX
  end define
```

See also Formats and Integer flags

(Return to TOP)

Linear moves

(Up-dated 13/08/2002)

The normal minimum output format for a linear move is

G01 X... Y... Z... F...

where **G01** (or G1) is the code for linear.
X Y and **Z** are the absolute **or** incremental positions.
F is the feed rate
The spindle speed and tool length offset may also appear on a linear move, this is more likely with multi-axes moves.
Also cutter compensation codes could be set if required, this is also more likey with PowerMill 3.0 2D machining..

A ***define block move linear*** is very rarely defined in the source files..

However, it is recomended that if the block is not set up in the Source Post Processor control file, the example below would be better defined in the option. . (In very rare circumstances it has been found that possible errors could occur if not set up)

The option file " ***define block move linear*** " could look like this :-

```
define block move linear
   N ; G1 ; G2 ; X ; Y ; Z ; D ;  F ; M1 ; M2
end define
```

(**G1** is the linear code (output normally G1), **G2** is the cutter compensation code (normally G41 or G42), **D** is the tool radius offset, **F** is the feed rate,.and **M1, M2** will be M function codes. [For multi-axes options it would be prudent to add G6, S and H]

- We **RECOMEND** that the format should be as follows instead of the above :-

```
define block move linear
   N ; linear ; G2 ; x coord ; y coord ; z coord ; tool radius ;  feedrate ; M1 ; M2
end define
```

- The radius compensation will need to be added to the majority of options as this is missing in the source files and is specifically required for PowerMill 2D radial leads in/out linear extension compensation setting. (Many machine tool controls cannot apply a radial compensation on an arc, only on a straight line)
  A feedrate is usually required to be output for a linear move.

  The first move after a tool change is always treated as a special case. (This is normally a Rapid Move function but can overlap, especially in multi-axes options. In these cases the Linear Block will need special attention, see tool change)

  Linear travel limits may be set and a warning message will appear if the X, Y or Z values exceed these limits. (The default limits in DUCTpost are usually set at -999999. and 999999. for all axes. See Limits)

  A check is made on the change in angle between each move and the next so that it is possible to output a constant contour speed code (e.g. **M90** on the Heidenhain) for moves where the change in angle is small.

- **NOTE :** All multi-axes moves are treated as Linear so it will be necessary to add the angular axes :- **azimuth axis ; elevation axis** to the above.

Rapid moves

(Up-dated 13/08/2002)

The typical outputl format for a rapid move could be as follows :-

G00 G6 X... Y... Z... S... H.... M3

where G00 (or G0) is the code for linear movement.
X Y and Z are the **absolute or incremental** positions
The spindle speed and tool length offset may also appear on a rapid move, (S2500 M3 and G43 H1.).
Radial Cutter Compensation codes are more likely to be **linear** output..

A rapid move block is usually defined as a default in the source file.

The " ***define block move rapid*** " could look like this :-

```
define block move rapid
   N ; G1 ; G2 ; G3 ; G6 ; X ; Y ; Z ; H ; S ; M1 ; M2
end define
```

(**G1** is the rapid code (normally G0)),
(**G2** is the radial cutter compensation code (normally G41 or G42))
(**G3** is the working plane code (G17, G18 or G19))
(**G6** is the Tool Length Offset code (possibly G43))
(**S** is the spindle speed)
(**F** is the rapid feed rate (not normaly required), but if needed then insert in the option " ***rapid feed code = 1*** ")
(**H** is the tool length offset. (This will require " ***tlo output = true*** " and " ***tool reset coordinates = 3*** set)
(**M1, M2** will be M function codes for Spindle ON (M3) and / or Coolant ON (M8))

We **RECOMEND** that the format should be as follows instead of the above :-

```
define block move rapid
   N ; rapid ; G3 ; tool length offset ; x coord ; y coord ; z coord ; tool length ; spindle  ; M1 ; M2
end define
```

It may be necessary to split all X Y Z moves into two moves as most machines do not guarantee linear interpolation on rapid moves. If this is required set :

```
split move = 1
```

In this case moves will be split so that downward moves are made with XY then Z, upward moves with Z then XY.
This is not set by default as 3D moves at rapid are normally made in safe positions.

For **Multi-axes** post processors it is **ESSENTIAL** that this is set to " **0** "

The first move after a tool change could well be treated as a special case, and whereas this used to be handled in the " ***define block move from*** " will now be handled in either the rapid, or linear blocks depending on the type of tool path encountered.
Linear travel limits may be set and a warning message will appear if the X, Y or Z values do not fall within these limits.
(The default limits in DUCTpost are -999999. and 999999. for all axes. See Limits)

Circular moves

(Up-dated 21/08/2003)

The normal form for a circular move is

```
  G02 X... Y... I... J... F...)
 or                              } xy Plane ( G17 )
 G03 X... Y... I... J... F...)

  G02 X... Z... I... K... F...)
 or                              } zx Plane ( G18 )
  G03 X... Z... I... K... F...)

  G02 Y... Z... J... K... F...)
or                              } yz Plane ( G19 )
 G03 Y... Z... J... K... F...)
```

where G02/G03 (or G2,G3) is the code for clockwise and anticlockwise arcs, X, Y or Z are the end points of the arc, I, J or K represent the circle centre. The G codes for circle and planes are set in the define codes block:

```
define codes
   circle cw    = G1 2
   circle ccw  = G1 3
   xy plane     = G3 17
   xz plane     = G3 18
   zy plane     = G3 19
end define
```

where G1 is the first modal group of G codes, and G3 the third group. They do not normally need to be changed.

The I, J and K coordinates may represent the actual **absolute** circle centre, in which case set

```
incremental centre = false
```

or they may represent the **incremental** distance between the start point and the centre, in this case set

```
incremental centre = true
```

If the I, J or K coordinates have the wrong sign (***it is important to check this if the centre type has been changed***) then the sign can be changed using

```
define format ( I J K )
  scale factor = -1  or  1
end define
```

Circle centre position I J, J K or I K appear for the appropriate plane output. If the plane code is also required (e.g **G17**, **G18** and **G19**) make sure that G3 is defined in the ***define block move circle***.

If the plane code has to appear on a line before the circle, put **G3** on a separate line in the " ***define block move circle*** ".

On a few machines it is necessary to split circular arcs into separate arcs for each quadrant. This is achieved by setting :-

```
single quadrant = true
```

The option file " ***define block move circle***" should look like this:

```
define block move circle
   N ; G1 ; G3 ; x coord ; y coord ; z coord ; key i ; key j ; key k ; feedrate ; M1 ; M2
```

```
          end define
```

G1 is the circle code (normally G2/G3), G3 is the plane code (G17, G18, G19).

Circle arc output is prevented :

Circular arcs can be suppressed and output as a set of small straight lines if the following flag is defined :

integer 26 = 0 is the prefered use **or circle output = (0 1 1 1)** (see Array data)

Preventing circle arc output in the individual major planes :

Circular arcs can now be suppressed and output as a set of small straight lines in individual planes by defining: the following flags :-

```
suppress xy arc = true default = false
suppress zx arc = true default = false
suppress yz arc = true default = false
```

Circle arc output exceeds machine maximum limit :

If an arc radius is generated that is larger than the maximum allowable machine radius, a machine error occurs.

The value " **arc radius limit** " represents the maximum radius setting, the default being **10000.0 mm.**
If the radius is to be limited to the appropriate machine maximum this will need to be inserted in the option file :-

arc radius limit = 5450.0 (example)

Circle arc output exceeds machine minimum limit :

If an arc radius is generated that is smaller than the machine can handle, a machine error occurs.

The value " **arc minimum radius** " represents the minimum radius setting, the default being **0.0 mm.**
If the radius is to be limited to the appropriate machine minimum this will need to be inserted in the option file :-

arc minimum radius = 0.016 (example)

The circle is not in a major plane :

DUCTpost prior to DP1335 will not support circular G2/G3 arc leads output for angular toolpaths offset in the **XY, YZ** or **ZX planes *greater than 0.1 degrees.*** Arcs are output as straight line moves, anything **less** than this, and the circle was ***snapped to the plane*** and a G02 / G03 output.
From **DP1335** this problem has been rectified to produce sensible output - however G02 / G03 is still not possible.

Reversal of G2/G3 for Arcs in G18/G19 Planes :

In some instances it has been found necessary to effect a reversal of G2/G3 in one, or both, of the **G18** and/or **G19** Plane.
There is no clear indication why this should be other than a machine tool configuration setting.
It is very unusual to change the **XY** Plane (**G17**)

The following needs to be added to the ***define block move circle*** :-

```
define block move circle
   if ( Word{G3} = = 18 )
     N ; G1 ( 5 - Word{G1} ) ; G3 ; x coord ; y coord ; z coord ;
          key i ; key j ; key k ; feedrate ; M1 ; M2
  else
     N ; G1 ; G3 ; x coord ; y coord ; z coord ;
          key i ; key j ; key k ; feedrate ; M1 ; M2
```

```
    end if
end define
```

(Word{G3} = = 18 or Word{G3} = = 19)   if both Planes are effected.

(Return to TOP)

Message Output

(Up dated 02/01/2001)

Messages (the APT PPRINT commands from DUCT and the CLdata 1044 PPrint from PowerMill) may be output on tape by use of the logical " **message output = true** " in the option file. (Usually the ***default*** setting in the source code though not in all cases)
The alternative if messages are not required is :- **message output = false**

However, it may be necessary to re-define the output words for the **start** and **end** of the messages. An example of how this is achieved is shown below :-

```
Output format required :-    N100 (MSG, ................... ET )

machine ....
#
# The output format required for the messages start,  in this case is ( MSG,
#
    define format ( MS )
        address letter     =  "( MSG, "   ## start address
        address width      =  7           ## include spaces
        field width        =  0           ## no output ( this is automatically handled )
    end define
#
# The output format required for the end of the message is  ET )
#
    define format ( EM )
        address letter     =  " ET )"
        address width      =  4
        field width        =  0
    end define
#
```

There should be no need to provide New Words as MS and EM are normally defined in the source file and therefore it will only be necessary to reformatting the existing word.
The same should apply to the inclusion in the word order listing and key definitions as illustrated below :-

```
#
    word order = (    OP   N    G1   G2   G3   G4   G5   )
    word order = ( +  G6   G7   X    Y    Z    B    C    )
    word order = ( +  I    J    K    R    D    S    T    )
    word order = ( +  H    M1   M2   MS   msg  EM   Q    )
    word order = ( +  Q1   Z2   R2   ID   F    )
#
    define keys
        message start   =  MS
        message end     =  EM
    end define
#
#   If messages are not output then ensure that the setting below
#   is included in the option file
#
     message output = true
#
    end
```

This will produce an output as below :-

```
%
:0001
N10G91G28X0Y0Z0
```

```
N20G40G17G80G49
N30G0G90Z10.
N40T10M6
N50G54G90M3
N60( MSG,  Toolpath Name: ET)
N70( MSG,    xyzxyz_cut_1 ET)
N80( MSG,  Output: ET)
N90( MSG,    UNITS: MILLIMETRES ET)
N100( MSG,    TOOL COORDINATES: TIP ET)
N110( MSG,    LOAD TOOL ET)
```

NOTE : Always check first whether or not these **words** are already defined in the source and are listed in the word order.

Do not re-create them if they are, **only re-format** them as shown above.

(Return to TOP)

Limits

(Up dated 03/01/2001)

The **Linear** axes limits can be set using the line

```
linear axis limits = ( -99999.  99999.  -99999.  99999.  -99999.  99999. )
```

where the first pair are the Minimum / Maximum limits of travel for the **X**, the second pair for **Y,** the third for **Z**.

(**Note :** The " **point** " after the value, although this can be omitted it is recommended that it be used, and these values are the default settings)

Practical Example :-

```
linear axis limits = (  -1235.0  1235.0   -540.0  540.0   -250.0  250.0  )
```

If these limits are exceeded a warning message will be printed, and the Post Processing will output " **x too small (or large) in block nn** " but will continue to output the actual value.

The **Rotary** axes limits can be set using the line

```
rotary axis limits = (  -99999  999999  -99999  999999    999     1 )   ## Default settings
```

where the first pair are the Minimum / Maximum limits of travel for the **Azimuth** rotation, the second pair for the **Elevation** rotation, the last pair are for the **angular tolerance** value to be maintained and the **number of moves** required to execute the angular change. This latter setting has, from DP1331 been superceded by " ***linearise multiaxis moves*** **= true / false** ", but will be retained to maintain compatibility with older option files.

Practical Example :-

4 axes m/c

```
                         Azimuth     Elevation
rotary axis limits = (  0.0   0.0  -360.0   360.0   0.1    4  )
```

If these rotational limits are exceeded the tool will be retracted to a pre-determined hieght from the surface of the job and the rotation angle set back 360 degrees, this will occur each time the limit is reached until the rotational motion is complete.
(e.g. If the tool path rotaion is say 1050 degrees the sequence will be 0 - 360 [360] ; 0 - 360 [720] ; 0 - 330 [1050])
It is likely that in circumstances such as this the machine control may well have a special code that requires outputing to ensure the movement is continuous in the same direction preventing a rewind back to the zero position.

5 axes m/c

```
                         Azimuth       Elevation
rotary axis limits = (  -20.0   110.0  -3600.0   3600.0   0.1    4  )
```

In the case above it is possible in certain circumstances where if the limits are exceeded, and a suitable alternative combination of Azimuth / Elevation angles are not possible, a warning message will be printed and the Post Processing will STOP.
With the latest version of Ductpost it is recommended that the Linear axis limits are accurately defined as these will have a bearing on the rotational axes movement.

(Return to TOP)

Explanation of Word[x] and its possible uses

(Up dated 29/12/2000)

In the inbuilt machine source files of Ductpost there is an initial list of defined words, an example is given below :-

```
     machine tiger

        define word /
         address letter = "/"
        end define

        define word N
         address letter = "N"
        end define

        define word G1
         address letter = "G"
        end define

        define word G2
         address letter = "G"
        end define

       etc. .......................
```

Each of the above defined words has a value i.e. **/** = **word[1]** ; **N** = **word[2]** ; **G1** = **word[3]** ; etc., and so on down the list.

In certain instances these can be helpful to use as a means of getting at a function that has no defined variable name in Ductpost.

An example may help.

Say that a particular output is required in the ' ***xz plane*** ', the rotation has to be reversed from **G2** to **G3** for this plane only. There is no variable defined word for the ' ***xz plane*** '. How then can we check for it?

Well we know that the planes are usually defined by the codes **G3 17** (***xy plane***), **G3 18** (***xz plane***), and **G3 19** (***zy plane***). Fortunately, **G3** is the group function code and can be used to evaluate which plane has been called by the use of the ***word[x]*** ploy, as is shown below :-

```
    define block move circle
       if ( word[5] = 18 )
          N ; G1 ( 5 - word[3] ) ; G3 ; G4 ; x coordinate ; y coordinate ;
             z coordinate ; B ; C ; R =C ; feedrate
       else
          N ; G1 ; G3 ; G4 ; x coordinate ; y coordinate ;
             z coordinate ; B ; C ; R =C ; feedrate
       end if
    end define
```

Another little trick that has been used in this example is the reversal of the rotation **G2 / G3**

G1 2 (**circle cw**) , **G1 3** (**circle ccw**) , 2 and 3 being the values assigned to **G1** (**word[3]**), so if we use the expression **G1 (5 - word[3])** we can effectively reverse the value output of **G1**. (i.e. **5 - 2 = 3** **or 5 - 3 = 2**)

This is useful to get you out of some problems, but it has limitations, and you will need to experiment to find if it will work for the case you may want to use it for.

NOTE :- If you define **additional** words in an option file, and wish to use these in the above method, then their values are **added** to the end of the in-built list of words in the order they are defined.

To check this order if you are not sure, do the following :-

type: ***ductpost -w [option file name] > [option file name].dmp***   to obtain a listing, and from this define the new **word[x]** value.

For a list of word values see :-   Word[x] List

(Return to TOP)

Decimal output

(Up-dated 03/01/2001)

The format of the co-ordinate output is determined by the following integer setting :-

Add the line ***integer 51 = n*** to the option file

- **integer 51** = (n) ***decimal output formats***

= 1 numbers less than 1 written as **.xxxx** (e.g. **.871**)
integers written as **xxxx.** (e.g. **34.**)

= 2 numbers less than 1 written as **0.xxxx** (e.g. **0.871**)
integers written as **xxxx.** (e.g. **34.**)

= 3 numbers less than 1 written as **.xxxx** (e.g. **.871**)
integers written as **xxxx.0** (e.g. **34.0**)

= 4 numbers less than 1 written as **0.xxxx** (e.g. **0.871**)
integers written as **xxxx.0** (e.g. **34.0**)

= 5 same as 3

= 6 numbers less than 1 written as **.xxxx** (e.g. **.871**)
integers written as **xxxx** (e.g. **34**)

Note :- These can affect the format of any decimal output

Explanation of Rapid Skim Feed

(Up-dated 10/10/2001)

PowerMill Skim moves are output as rapid **Linear** moves at the feed rate set in the Rapid Box of the **Feed Rates Form**, and is treated by Ductpost in the normal way. The default setting is ***3000 units/min***

PowerMill outputs the Feed Rates Form, feed rates in a " ***ppfun fedrat statement*** ".

An example of the cut file CLdata output illustrating this is given below :-

```
-----------------------------------------------------
   3 ( 25)     2000     1079  PPFUN
                   99
                          FEDRAT 6000 950 1100
-----------------------------------------------------
```

i.e. **Rapid Skim = 6000 Plunge = 950 Cutting = 1100**

If Skim moves are used in the tool path output then cutting times will be based on the the above rate settings, and the output feed rate will reflect this in the tape file as shown below :-

```
1538 G0 Z84.                       ## Normal Rapid move ( Feed rate not usually specified )
1539 G1 X15.653 Y-37.059 F6000     ## Skim move
1540 G0 Z69.667                    ## Normal Rapid move
1541 G1 Z66.667 F950                ## Plunge move
1542 X14.684 Y-37.456 F1100       ## Cutting move
```

The normal **NON Linear rapid G0** feed rate is not effected by this rapid value, being set by the internal " ***rapid feedrate = 9999.0***" default setting. (Or what ever value set in an option file)

The **Linear** maximum feed rate is also controlled internally by the " ***maximum feedrate = 9999.0*** " default setting (Or what ever value set in the option file)

If these rates are set too low then it will be neccessary to include these settings in the option file defined to the appropriate maximum values. (Conversely for imperial feed rates they may be set too high)

To obtain a feedrate output in the Rapid G0 block it will be necessary to add / modify the option " ***define block move rapid***" to include an " **; F** ", and to insure that " **rapid feed code = 1** " is include in the option

----

From **DUCTpost1300** version there will be a variable defined for this feedrate, " **Srat**", which will allow a little more flexibility in manipulating this function.

An example of this is if the skim move is to be classed as a **Rapid Non Linear G0** move instead of a **G1** move.
The following illustrates how this may be achieved.

```
define block move linear
  if (  feedrate => srat  )
    N ; rapid ; x coord ; y coord ; z coord  ; M1 ; M2   ## Rapid Skim as RAPID
  else
    N ; linear  ; x coord ; y coord ; z coord ; feedrate ; M1 ; M2
  end if
end define
```

So the above example tape output would now look as follows :-

```
1538 G0 Z84.                       ## Normal Rapid move
1539 X15.653 Y-37.059             ## Skim move ( At Rapid Non Linear rate )
```

```
1540 Z69.667                         ## Normal Rapid move
1541 G1 Z66.667 F950                  ## Plunge move
1542 X14.684 Y-37.456 F1100          ## Cutting move
```

Warning : It is recomended that this method be employed with care as it is possible that gouging may occure due to the move not being implemented in a direct line because the machine GO could move the axes in a zig zag.

NOTE :- If the machine control can accept **feed rate parameters** then to obtain the correct functional operation for a Skim move parameter feed rate output see :- Feedrate Parameters (To be writen)

Explanation of Angular Feed Rates

(Up-dated 03/01/2001)

It is possible that some machines require rotary movements to have an individual feed rate assigned, based on the nominal defined feed rate. (Usually defined as ***Inverse Time Feedrate***)

An example is shown below :-

```
%
O0100 ( 0100-Feedrate        )
N2G0G40G17G80G90G49
N3T04M6
N4G0G90X0.Y0.A0.B0.S9500M3
N5G0X0.Y0.A0.B0.
N6G43Z15.H04
N7G1Z12.F1.33
N8X-7.F0.56
N9Z5.F0.56
N10Z.9519F0.24
N11Z.9275A-1.0169F133.88
N12Z.9033A-2.0339F137.47
N13Z.8795A-3.0508F138.57
N14Z.8567A-4.0678F143.12
N15Z.835A-5.0847F130.66
N16Z.8145A-6.1017F156.34
```

To obtain this output it is necessary to include the following in an option file :=

```
integer 71 = 1
integer 72 = 1
```

It will probably be required to modify the "**F**" definition for the feed rate output as indicated below.

```
define format ( F )          [ This example and above is for an Imperial option ]
  not modal
   imperial formats
 decimal places = 2
end define
```

NOTE :- As this function is very rare it is not entirely sure that the output is effective and will need to be carefully tested to ensure there is no detrimental effect on the machining operation.

Tape Splitting and Segmenting

(Up dated 03/01/2001)

NC programmes may need to be **Split** into separate files, usually because the machine can only handle a programme of a fixed size, or possibly with the DP1321 version of Ductpost to provide sub routine output.
Segmentiation of an NC programme is where a **specific insert** is added to the tape at pre-determined intervals within a single file.

Tape Splitting

By setting either, the real flag " ***maximum tape length*** ", which will break the overall programme in to small sections of a size set by the value specified. (see below)

The real " ***maximum tape length***" is in **feet** and must have a **decimal point**with the length required.
A length of **100.** **(feet)** **= 12000 bytes approximately**.
A length of **0.** **(zero)** = **do not split** tape. (Usually the default setting for most controls)

or,

the integer flag " ***maximum tape blocks = 60000*** ", will do the same function, but after the predefined number of 60,000 **line** blocks, or what ever number is defined.

It is **recommended** that only one, or the other is used, **not both together.**

Tape **Splitting** for Sub-Routinr operation see Sub-Routines (To be writen)

Tool Movement

The next most important factor is the distance for the tool to withdraw on splitting, or at the segmentation point.

This is set by the integer flag " ***tape split retract distance*** ". (This replaces the older keyword " ***retract distance*** " which has been made **obsolete** from DP1203 version, and if encountered in an option file, post processing will stop, flagging an **error message**).

The way the tool moves back to the work piece **IS NOT** controlled by defining " ***split move*** " , this has no influence at all on this operation.

The following flag settings indicate the action to be expected with **NO** **"** ***define block tape split move*** **"or** " ***define block move linear*** " set :-

DUCTPost 1100 and earlier	**Lift Action at Tape Split**	**Return Action after Tape Split**
retract distance = 0	**retracts** tool to **FromZ** position at **cutting** feed rate	**returns** to a **Safe Z** height at **rapid**, then to **Lift Off** position at **cutting** feed rate
***retract distance =* -999**	**NO** retraction	**retracts** to a **StartZ** height at **rapid**, then to **Lift Off** position at **cutting** feed rate
retract distance* = -998** (***or any negative value other than -999)	******* **DISASTER PLUNGE** *******	As above
retract distance = 100	**retracts** tool to **100 mm** above **Lift Off** position at **cutting** feed rate	**returns** to a **StartZ** height at **rapid**, then to **Lift Off** position at **cutting** feed rate
DUCTPost 1205	**Lift Action at Tape Split**	**Return Action after Tape Split**
		retracts to **SafeZ**

tape split retract distance = 0	**NO** retraction	at **rapid,** moves **X0,Y0,Z0** at **cutting**feed rate **(** *****DISASTER***** **)**
tape split retract distance = -999 *(or any negative value)*	**retracts**tool to **FromZ** position at **rapid** rate	**returns** to **SafeZ** at **rapid**, then to **Lift Off** position at **cutting** feed rate
tape split retract distance = 100	**retracts** tool to **100 mm** above **Lift Off** position at **rapid** feed rate	As above

The following define blocks are used with tape splitting :-

define block tape split start ; define block tape split move ; define block tape split end

Example Control settings :-

Ductpost1100 or earlier **Ductpost1203 onwards**

```
   maximum tape length   = 70.
              or
  maximum tape blocks  = 500
  minimum tape blocks   = 0     ( Default, not usually required )
  retract distance          = 100          tape split retract distance = 100
```

Example of block definitions :-

```
 define block tape split start
    N ; "%  ( Tape Split Start )"
    N ; ID ProgID
  end define

  define block tape split move
    N ; G1 0 =C ; X OldX ; Y OldY ; Z SafZ ; S ; M1
    N ; G1 1 ; Z OldZ ; F Prat
  end define

  define block tape split end
   N ; M1 30 ; " ( Tape Split end )"
  end define
```

It is also advisable to verify if a *" **define block move linear** "* has been set up in the controls source file, and if not, then also include the following minimum definition :-

```
  define block move linear
   N ; G1 ; X ; Y ; Z ; F ; M1 ; M2
  end define
```

(Review this section Tape Splitting)

(Back to Ductpost1206 Revisions)

Tape Segmentation

This allows some special block to be inserted in the NC programme at pre-determined points, and is controlled by the flag *" **segment type** "*, in conjunction with the following :- " ***maximum segment*** " , and " ***max tape blocks*** " .
The recognised settings defining which action triggers the insertion, are :-

segment type = 0	after a fixed NC programme length in feet
segment type = 1	after a fixed tool travel distance.
segment type = 2	after a fixed number of blocks

The insertion data is handled by *" **define block tape segment** "*, and the way the tool behaves prior to segmentation is controlled by the setting of *" **tape split retract distance** "*.   (" **retract distance** " prior to DP1203)

1/ Insert by block number example

```
    segment type                 = 2
    max block number              = 11500
    tape split retract distance = 100      ## Lift Off 100 mm.

    define block tape segment
       N ; " ( Tape segment comment )"
       N ; M1 05 ; M2 09
       N ; M1 01
       N ; G1 00 =C ; X OldX ; X =C ; Y OldY ; Y =C ; Z SafeZ ; M1 03 ; M2 50
       N ; G1 01 ; Z OldZ ; F Prat
    end define
```

2/ Insert by tape length, or tool travel distance example

```
    segment type                 =  0    or   = 1
    max segment                  = 1100.    ## Insert at 1100 feet.
    tape split retract distance = 100       ## Lift Off 100 mm
    split move                   = 0        ## No specific XYZ moves
```

Resultant moves to be expected following segmentation insert

With minimum defined " ***block tape segment*** " and NO " ***define block move linear*** " :-

```
  define block tape segment
     N ; " ( Tape segment comment )"
  end define
```

DUCTPost 1100 and earlier	Lift Action at Segmentation	Return Action after Segmentation
retract distance = 0	**NO** retraction	**moves to next point** after inserting segment block at **cutting feed rate**
retract distance = -999 ***(Or any negative number)***	**retracts** to **(value defined) mm** above **Lift Off** point at **rapid**	**moves to next point** after inserting segment block but **doesn't return** to **Lift Off** point. (*****Potential Disaster*****)
retract distance = 100	**plunges100 mm** from **Lift Off** point at **cutting** feed rate (*****Disaster*****)	**As above** (*****Already a Disaster*****)

DUCTPost 1205	Lift Action at Segmentation	Return Action after Segmentation
tape split retract distance = 0	**NO** retraction	**moves to next point** after inserting segment block at **cutting** feed rate
tape split retract distance = -999 ***(Or any negative number)***	**retracts** tool to **FromZ** position at **rapid** rate	**returns** to **Lift Off** point after inserting segment block at **cutting** feed rate
tape split retract distance = 100	**retracts** tool **100 mm** from **Lift Off** point at **rapid** feed rate	As above

NOTE :- in the above operation **FromZ** and **SafZ** may be treated as being the same, even if different values, **StaZ** is treated as **SafZ** in DP1203-1205, and is corrupted in DP1100.

Recomendation:-

It is advisable to have the following format for the " ***define block move segment*** ", and to ensure that " ***define block move linear*** " has a movement defined.

```
 define block tape segment
    N ; " ( Tape segment )"    ## Insertion Data
    N ; G1 00 =C ; X OldX ; X =C ; Y OldY ; Y =C ; Z SafeZ ; M1  ; M2
    N ; G1 01 ; abs data ; Z OldZ ; F Prat
  end define
```

(Review this section Segmentation)
(Review Tape Splitting)

DUCTpost 1206 Revisions

Tape Splitting

Settings same as previous.

DUCTPost1206	Lift Action at Tape Split	Return Action after Tape Split
tape split retract distance = 0	**NO** retraction	**retracts** to **StartZ** at **rapid**, then to **Lift Off** point at **cutting** feed rate
tape split retract distance = -999 ***(Or any negative number)***	**retracts** tool to **FromZ** position at **rapid** feed rate	As above
tape split retract distance = 100	**retracts** tool **100 mm** from **Lift Off** point at **rapid** feed rate	**returns** to **SartZ** at **rapid**, then to **Lift Off** point a **cutting** feed rate

Tape Segmentation

DUCTPost1206	Lift Action at Segmentation	Return Action after Tape Segment
tape split retract distance = 0	**NO** retraction	**moves to next point** after inserting segment block at **cutting** feed rate
tape split retract distance = -999 ***(Or any negative number)***	**retracts** tool to **FromZ** position at **rapid** feed rate	**returns** to **Lift Off** point after inserting segment block at **cutting** feed rate
ape split retract distance = 100	**retracts** tool **100 mm** from **Lift Off** point at **rapid**feed rate	As above

(Back to DP1206 , Segmentation , Top)

Constant contour speed

(Up-dated 28/12/2000)

On each rapid and feedrate move the anglular deviation between the previous move and the next is checked, so that a code can be output where the angle change if small prevents deceleration at the end of a move.

Example :-

```
move safe angles = (  0  5  -999  -999  )

define codes
   constant contour speed = M2 90
end define
```

This will put M90 on the end of each line where the angle change is less than 5 degrees.

It is possible (although not usual) to have two separate codes for different ranges of angles.

Example :-

```
move safe angles = (  0  20  180  360  )

define codes
   constant contour speed     = G6 11
   constant contour speed 2  = G6 12
end define
```

This will put G11 on each line where the angle change is less than 20 degrees, and G12 where the change is more than 180.

Important :-

The default angle checked is that between the line being read and the previous line which will place the M90 on the correct line for the Heidenhain.

For other machines set

```
integer 77 = 2
```

which causes the the code to be output on the line ***before***the small angle change.

Radial Cutter Compensation

(Up dated 10/10/2001)

From Ductpost Version 1331 and PowerMill 3.0 the RADIAL cutter compensation function has been revised and two of the array settings are now obsolete. (There may be some problems with old DUCT cut files but it is not thought likely)

The old array setting has been dispensed with and replaced with the integer values 34 , 35 , 36, 37 (see Array Data) and the integers 34 and 36 are now obsolete. It is still necessary to define the keys and codes for compensation output, and also to ensure that a ***Linear Block*** is defined :-

Example ;-

```
        integer        34  35  36  37
  comp output     = ( 0   1    0    1 )    ## If this format used then all four elements must be entered.
               or
    integer  34    =   0    ## Obsolete
    integer  35    =   1    ## Output Radial Compensation D ( Usually D ToolNum ) [Default]
    integer  36    =   0    ## Obsolete
    integer  37    =   1    ## Output Radial Compensation codes G41 G42 G40 [Default]

  define codes
    comp off        = G2  40
    comp on left    = G2  41
    comp on right   = G2  42
  end define

  define keys
    tool radius  =  D
  end define

 define block move linear
    N ; linear ; G2 ; tool radius ; x coordinate ; y coordinate ; z coordinate ; feedrate ; M1
 end define
```

tool radius can be placed as shown above, or before " **feedrate** " but **NOT AFTER** as the majority of machine controls will object.

Maho Special Case.

Maho in some cases require a **G43** code output with the compensation G41 or G42, this can be obtained in the following manner.

```
      define block move rapid
         N ; G1 ; x coord ; y coord ; z coord
         set swa
      end define

     define block move linear
        if ( ToolComp = 41 or ToolComp = 42 and swa )
         N ; linear ; z coord ; feedrate
         N ; G2 43 ; x coord ; y coord ; z coord ; feedrate ; M1 ; M2
         unset swa
       else
         N ; linear ; G2 ; x coord ; y coord ; z coord ; tool radius ; feedrate ; M1 ; M2
       end if
    end define
```

Should provide the following output :-

```
N26 G01 Z12. F750
N28 G43
N30 G01 G41 X56.779 D12 F1000
```

(Return to TOP)

Drilling -Tapping Cycles

(Up-dated 17/10/2003)

Ductpost Canned cycles were originally set up for **DUCT,** unfortunately ***PowerMill*** works to slightly different rules and could utterly confuse Ductpost in some instances. Work is on going with Ductpost to resolve the difference in operation but as PowerMill is also evolving with the canned cycle routines this may take a little time to settle down so care is needed.

The **Basic** structure of the Canned Cycle operation is described below and deals in the first part with **3 axes** output.

There are four blocks available to define canned cycle output :-

```
define block cycle start
......
end define

define block move cycle
......
end define

define block move tap
......
end define

define block cycle end
......
end define
```

In most cases the ***cycle start*** is not used in the inbuilt source codes as the cycle parameters are normally output on the first move. The ***cycle end*** output is usually just *G80*. (**end of drill**)

Goto (Heidenhain , Siemens , Cincinnatti-Acramatic)

The Standard In-built Cycles

The **codes** for different canned cycle types are normally set up using the ***G4*** group name as indicated below, however exceptions to this can occur, and Heidenhain is a classic example as shown in the second column :-

Fagor Control		Heid400 Control	
define codes		define codes	
dwell	= G1 4	dwell	= G1 4
drill	= G4 81	drill	= G4 1
break chip	= G4 82	break chip	= G4 1
deep drill	= G4 83	deep drill	= G4 1
tap	= G4 84	tap	= G4 2
bore 1	= G4 85	bore 1	= G4 1

bore 2 = G4 86	bore 2 = G4 1
bore 3 = G4 87	bore 3 = G4 1
bore 4 = G4 88	bore 4 = G4 1
bore 5 = G4 89	bore 5 = G4 1
end of drill = G4 80	end of drill not used
cycle retract not used	cycle retract not used
end define	end define

The various cycle ***parameters*** required are set using keys, or by the use of ***variable names*** in the blocks.

Using keys

Fagor Control	Heid400 Control
define keys	define keys
clearplane = R2	clearplane = R2
drill peck depth not used	drill peck depth = Z3
drill hole depth = Z2	drill hole depth = Z2
cycle dwell not used ++	cycle dwell not used ++
dwell = X **	dwell = DW **
end define	end define
** Use for DUCT and PM2.5	** Use for DUCT and PM2.5
++ Use for PM3.0 > (see notes below)	++ Use for PM3.0 > (see notes below)

An example of two typical inbuilt Canned Cycle set up is shown here :-

Fagor Control	Heid400 Control
define block cycle start	define block cycle start
end define	if (cycle != 4.)
	N ; " CYCL DEF 1.0 PECKING"
	N ; G4 =C ; clearplane =C
	N ; G4 =C ; drill hole depth =C
	N ; G4 =C ; drill peck depth =C
	N ; ".4 DWELL 0,000" ; G4 =C
	N ; G4 =C ; ".5" ; FF =C
	else
	N ; " CYCL DEF 2.0 TAPPING"
	N ; G4 =C ; clearplane =C
	N ; G4 =C ; drill hole depth =C
	N ; ".3 DWELL 0,000" ; G4 =C
	N ; G4 =C ; ".4" ; FF =C
	end if
	end define
define block move cycle	define block move cycle
N ; X ; Y ; G4 ; R2 ; Z2	N ; G1 =C ; x coordinate =C ;
end define	y coordinate =C ; RR ; FF 9999
	N ; " CYCL CALL M"
	N ; G1 =C ; z coordinate =C ; RR =C ;

```
                                                  FMAX =C
                                              end define
 define block move tap                        define block move tap
   N  ;  X   ;  Y   ; G4  ; R2  ; Z2          end define
 end define
 define block cycle end                       define block cycle end
   N  ; G4  80                                end define
 end define
```

(Return to Codes Variables Keys TOP)

Using variables :-

The **variables** available to use within the block definition are :-

ClearPlane
HoleDepth
HoleDiameter
Cycfed

ClearPlaneInc
HoleDepthInc
PeckDepth

Cycledwell
HoleTop
PeckDepthInc

Example :-

```
  define block cycle start
     N ; G4 ; G6 ; Z2 HoleDepth ; R2 ClearPlane ; Q PeckDepth ; feedrate
  end define
```

Note :- **U**pper & **l**ower case letters for ***variables*** - used to differentiate from ***key*** or ***code*** use see below.

```
  define block move cycle
     N ; G4 ; x coord ; y coord ; z coord ; feedrate
  end define
```

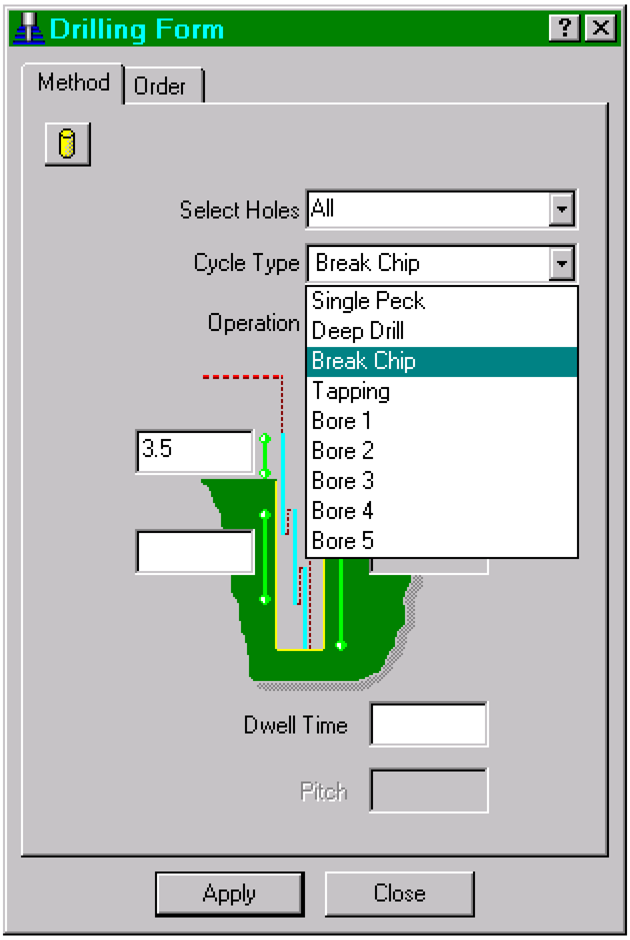
However, we recomend the following key format :-

```
  define block cycle start
     N ; G4 ; G6 ; drill hole depth ; clearplane ; drill peck depth ; cycle dwell ; feedrate Prat
  end define
```

Note :- **lower** case letters only for ***key*** or ***code*** use.

Cycle Types

The PowerMill Drill Form provides a selection of cycle output types which Ductpost refers to as indicated below :-



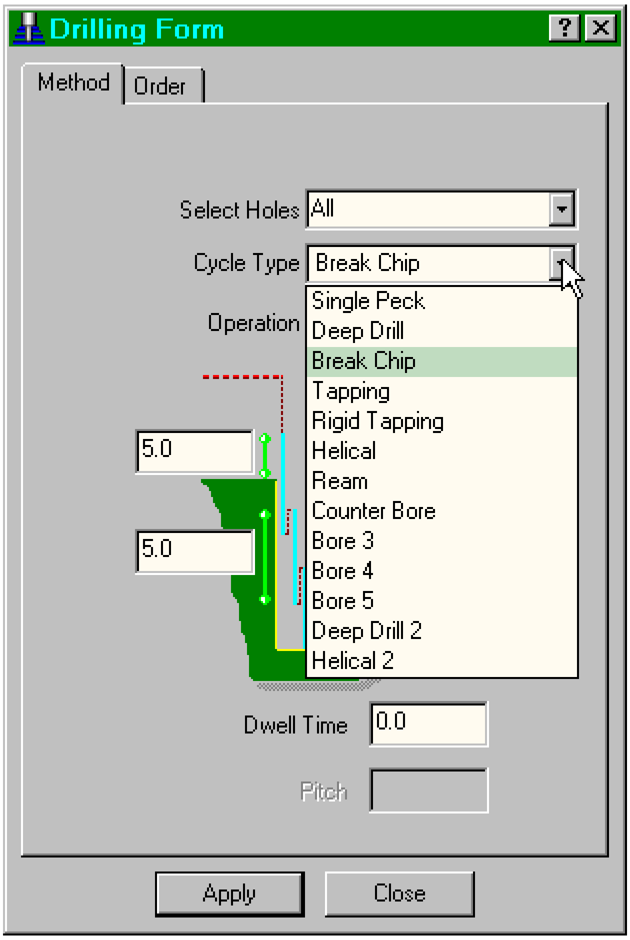
Select Holes: All

Cycle Type: Break Chip

3.5

<u>Prior to PM4.0</u>

Cycle Type		**DP Cycle Ref.**
Single Peck	=	cycle 1
Break Chip	=	cycle 2
Deep Drill	=	cycle 3
Tapping	=	cycle 4
Bore 1	=	cycle 5
Bore 2	=	cycle 6
Bore 3	=	cycle 7
Bore 4	=	cycle 8
Bore 5	=	cycle 9



From PM4.0 onwards

Cycle Type **DP Cycle Ref.**

Single Peck	=	cycle 1
Break Chip	=	cycle 2
Deep Drill	=	cycle 3
Tapping	=	cycle 4
Rigid Tapping	=	cycle 10
Helical	=	cycle 11
Ream	=	cycle 5
Counter Bore	=	cycle 6
Bore 3	=	cycle 7
Bore 4	=	cycle 8
Bore 5	=	cycle 9
Deep Drill 2	=	cycle 13
Helical 2	=	cycle 12

These cycle references can be used in " ***if*** " statements as indicated in the above Heid400 example where " ***if (cycle != 4)*** " is checking to see if the cycle is other than a Tapping Cycle, in which case it will action the first block of code.

Ductpost cycle routines were very simply defined and can no longer cope with the more versatile output from PowerMill, and in many cases the requirements of the more modern machine tool controls.
It will be found that recourse to the use of " ***if*** " statements to ensure the required cycle output format will become a necessity in a lot of cases.

(Return to Complex Cycles Variables TOP)

Cycle Output Format (**Warning**)

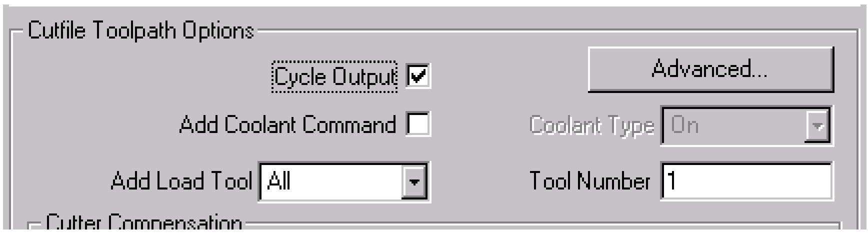
The **integer flag** *cycle output* was used to define whether or not the output from DUCT appeared as G code, or linear moves for the cycles.

.

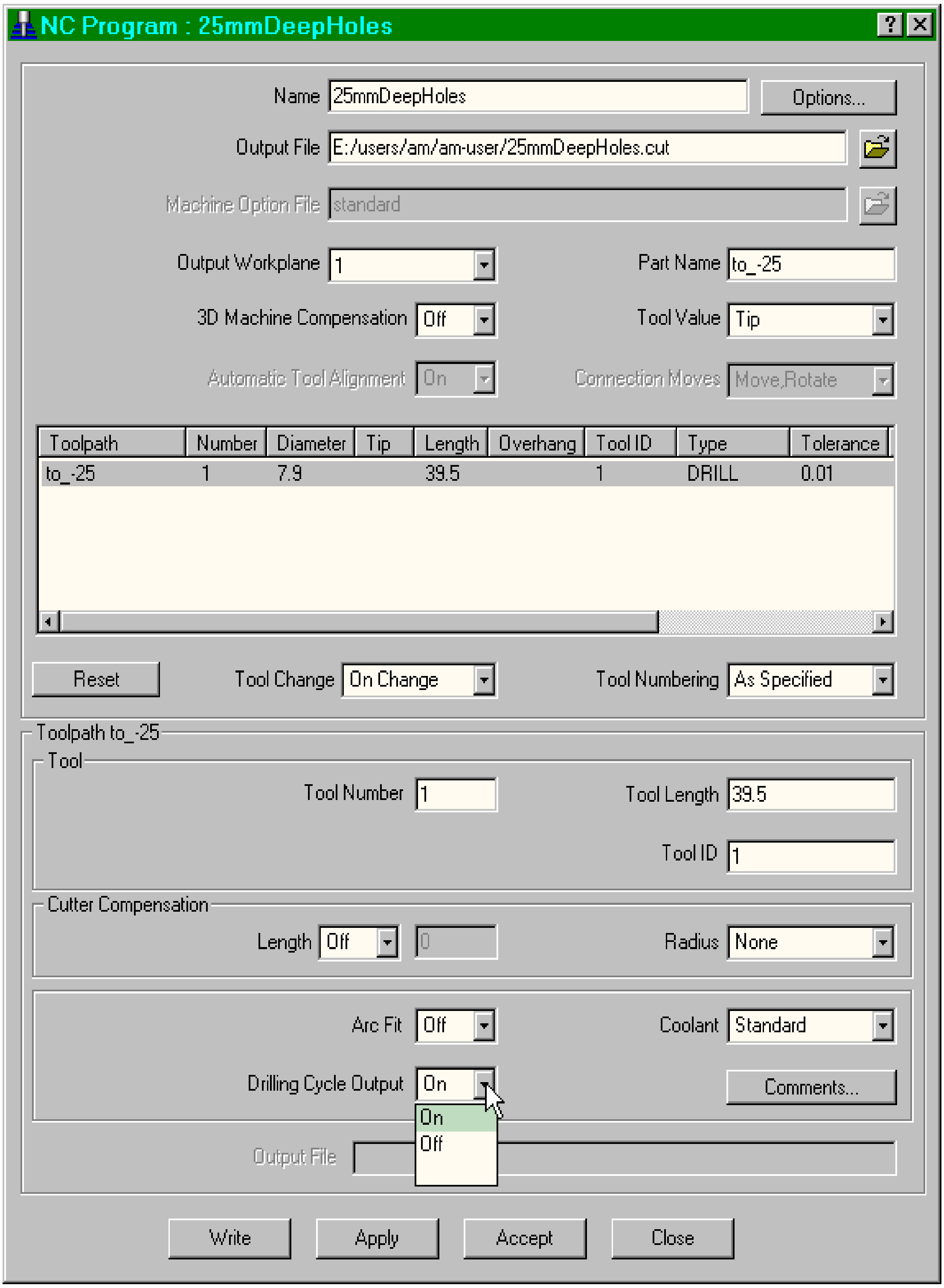
The **default** setting *cycle output = 1* will output the G code format.
The setting *cycle output = 0* still outputs the G code, but alters the cycle sequence after the first cycle (see below)

cycle output = 1	*cycle output = 0*
N50X0Y0 N60X103.Y17.33 N70G81Z-17.456R-10.456 N80G80 N90G81*X179.*Z-17.448R-10.448 N100G80	N50X0Y0 N60X103.Y17.33 N70G81***F100***Z-17.456R-10.456 N80G80 N90G81F100Z-17.448R-10.448 N100*X179.* N110G80

As this is contrary to what it is supposed to do it is recommended that this method of control is ignored and if Linear output is required for cycle output then use PowerMill's - Tool Path Output Form - Cycle Output check box to select what output is preferred.


Cutfile Toolpath Options
Cycle Output ☑
Advanced...
Add Coolant Command □
Coolant Type On
Add Load Tool All
Tool Number 1
Cutter Compensation

Prior to PM4.0



Name 25mmDeepHoles

Output File E:/users/am/am-user/25mmDeepHoles.cut

Machine Option File standard

Output Workplane 1

Part Name to_-25

3D Machine Compensation Off

Tool Value Tip

Automatic Tool Alignment On

Connection Moves Move,Rotate

Toolpath	Number	Diameter	Tip	Length	Overhang	Tool ID	Type	Tolerance
to_-25	1	7.9		39.5		1	DRILL	0.01

Tool Change On Change

Tool Numbering As Specified

Toolpath to_-25

Tool

Tool Number 1

Tool Length 39.5

Tool ID 1

Cutter Compensation

Length Off 0

Radius None

Arc Fit Off

Coolant Standard

Drilling Cycle Output On

Output File

From PM4.0

Un-check **Cycle Output** if Linear format required for Cycle moves.

This must **ALWAYS** be un-checked for 3 + 2 angular drilling, unless the machine control can handle canned cycles in this situation, such as the Heidenhain 430 control via the CYCL DEF 19.0 function.

(***However, it is not possible to implement this feature with PowerMill 3.0 and versions of Ductpost1331 and earlier***)

PowerMill Drilling, Tapping etc. DP1335 & PM4.0 onwards.

As PowerMill is now the main output for this form of machining, **DUCT** output will **not** feature in the following descriptions.
(For help with Duct problems refer to Support)

Two examples are given below to illustrate probable changes required to the in-built cycle definitions.

ISO Code (Fanuc11m) Inbuilt Source file

```
    define keys
       cycle dwell          not used
       dwell               = X     ## Old DUCT dwell
       drill peck depth   = Q1
       drill hole depth    = Z2
       clearplane          = R2
   end define
#
   define codes
       drill                = G4  81
       break chip         = G4  82
      deep drill          = G4  83
       tap                 = G4  84
       bore 1              = G4  85
       bore 2              = G4  86
       bore 3              = G4  87
       bore 4              = G4  88
       bore 5              = G4  89
       end of drill        = G4  80
       cycle retract     = G6  99
   end define
#
      cycle output         =  1
      integer 69            =  0   ## Default tapping
feedrate approx. 85% Normal Plunge feed rate.
(Not shown )
#
   define block cycle start
      N ; S ; M1
      N ; G3 ; G5 ; M2
      F =C ; G4 =C ; Z2 ; R2 ; Q ; Q1 ; G6 ;
   end define
#
```

ISO Code (Fanuc11m) Option file [changes]

```
   define format ( P )
       field width        = 3
       tape position     = 1
       modal
       metric formats
      decimal point     = true
      decimal places   = 2
       trailing zeros     = false
       imperial formats = metric formats
   end define
#
     word order  = ( + P )
#
   define keys
      cycle dwell      = P
      dwell              not used
   end define
#
   define codes
       cycle retract    = G6 98  ## Can be either 98 or 99
   end define
#
     block order       =  true
     integer 69         = 2    ## Reset tapping feedrate
                   to be Normal Plunge feedrate.( cycfed or
Prat )
#
  define block cycle start
     if ( cycle <= 2 or cycle => 5 )
       N ; G4 ; G6 ; x coord  =C ; y coord  =C ; drill hole
depth ; clearplane ;
              cycle dwell ; feedrate Prat
     end if
     if ( cycle = = 3 )
       N ; G4 ; G6 ; x coord =C ; y coord =C ; drill hole
  define block move cycle
     N ; G4 ; G6 ; X ; Y ; Z2 ; R2 ; Q ; Q1 ; F ; M2
  end define
#
  define block move tap
     N ; G6 ; G4 ; X ; Y ; Z2 ; R2 ; F ; M1 ; M2
  end define
#
  define block cycle end
     G4  80
  end define
```

```
depth ;  clearplane ;
             drill peck depth ; cycled well ; feedrate Prat
     end if
     if ( cycle = = 4  )
       N ; G4 ; G6 ; x coord =C ; y coord =C ; drill hole
depth ; clearplane ;
             cycle dwell ; feedrate cycfed ;
     end if
  end define
#
    define block move cycle
      N ; x coord ; y coord ; M1 ; M2
    end define
#
    define block move tap
      N ; x coord ; y coord ; M1 ; M2
    end define
#
    define block cycle end
      N ; end of drill
    end define
```

The following is the outputs from both examples using the same cut file which illustrates the NC formats for Single Peck Drilling ***T2*** **G81** ; Break Chip ***T3*** **G82** ; Tapping ***T4*** **G84** ; Bore 1 (Reamer) ***T6*** **G85** and Deep Drill ***T7*** **G83** for a pair of holes in each case :-

NC Output for Source file	NC Output for the Option file
T2 M6	***T2 M06***
X0 Y0 S1500 M3	(10.000mm Drill)
Z60.	G0 G90 X0 Y0 B0 S1500 M03
Z-40.	G43 Z60. H2
X-24.998 Y24.998	X-24.998 Y24.998
G81 G99 Z-84. R-60.5 F100	**G81** G98 X-24.998 Y24.998 Z-84. R-60.5 P0
X-74.998	F100
G80	X-74.998
T3 M6	G80
G43 Z60. H3	***T3 M06***
G0 X-25.001 Y44.999 S1500 M3	(6.800mm TapDrill)
Z-40.	G0 G90 X-25.001 Y44.999 B0 S1500 M03
G82 G99 Z-74. R-46.5 F100	G43 Z60. H3
G80	**G82** G98 X-25.001 Y44.999 Z-74. R-46.5 P0
G82 G99 X-75.001 Z-84. R-46.5 F100	F100
G80	G80
T4 M6	**G82** G98 X-75.001 Y44.999 Z-84. R-46.5 P0
G43 Z60. H4	F100
G0 X-25.001 Y44.999 S100 M3	G80
Z-40.	***T4 M06***
G84 G99 Z-74. R-46.5 **F118**	(8.000mm Tap7)
G80	G0 G90 X-25.001 Y44.999 B0 S100 M03

```
G84 G99 X-75.001 Z-84. R-46.5 F118
G80
T6 M6
G43 Z60. H6
G0 X-24.999 Y64.998 S800 M3
Z-40.
G85 G99 Q5. Z-74. R-50.5 F110
G80
G85 G99 X-74.999 Q5. Z-84. R-46.5 F110
G80
T7 M6
G43 Z60. H7
G0 X-24.999 Y84.997 S1250 M3
Z-40.
G83 G99 Q4. Z-94. R-60.5 F120
X-74.999
G80
```

Points to Note

Missing from the above output is the dwell call " **G4 Xn.nn** " as PowerMill does not use this in the New Drilling function, using instead " ***cycle dwell*** ".

#

Particular attention needs to be paid to the first hole positioning output and subsequent holes, as it is possible that the default settings can in some instances cause the 1st. hole to be drilled twice, or missed all together.

#

Note the feed rate for the **G84** Tapping cycle and compare with the output opposite. This is due to " ***integer 69*** "

```
G43 Z60. H4
G84 G98 X-25.001 Y44.999 Z-74. R-46.5 P2.75
F125
G80
G84 G98 X-75.001 Y44.999 Z-84. R-46.5 P2.75
F125
G80
T6 M06
( 8.000mm Reamer          )
G0 G90 X-24.999 Y64.998 B0 S800 M03
G43 Z60. H6
G85 G98 X-24.999 Y64.998 Z-74. R-50.5 P1.5
F110
G80
G85 G98 X-74.999 Y64.998 Z-84. R-46.5 P1.5
F110
G80
T7 M06
( 10.000mm Drill          )
G0 G90 X-24.999 Y84.997 B0 S1250 M03
G43 Z60. H7
G83 G98 X-24.999 Y84.997 Z-94. R-60.5 P2.25
Q4. F120
X-74.999
G80
```

Points to Note

Dwell has been re-defined with a new word **P** and is assigned to the **key** " ***cycle dwell*** " with the address letter **P**.

#

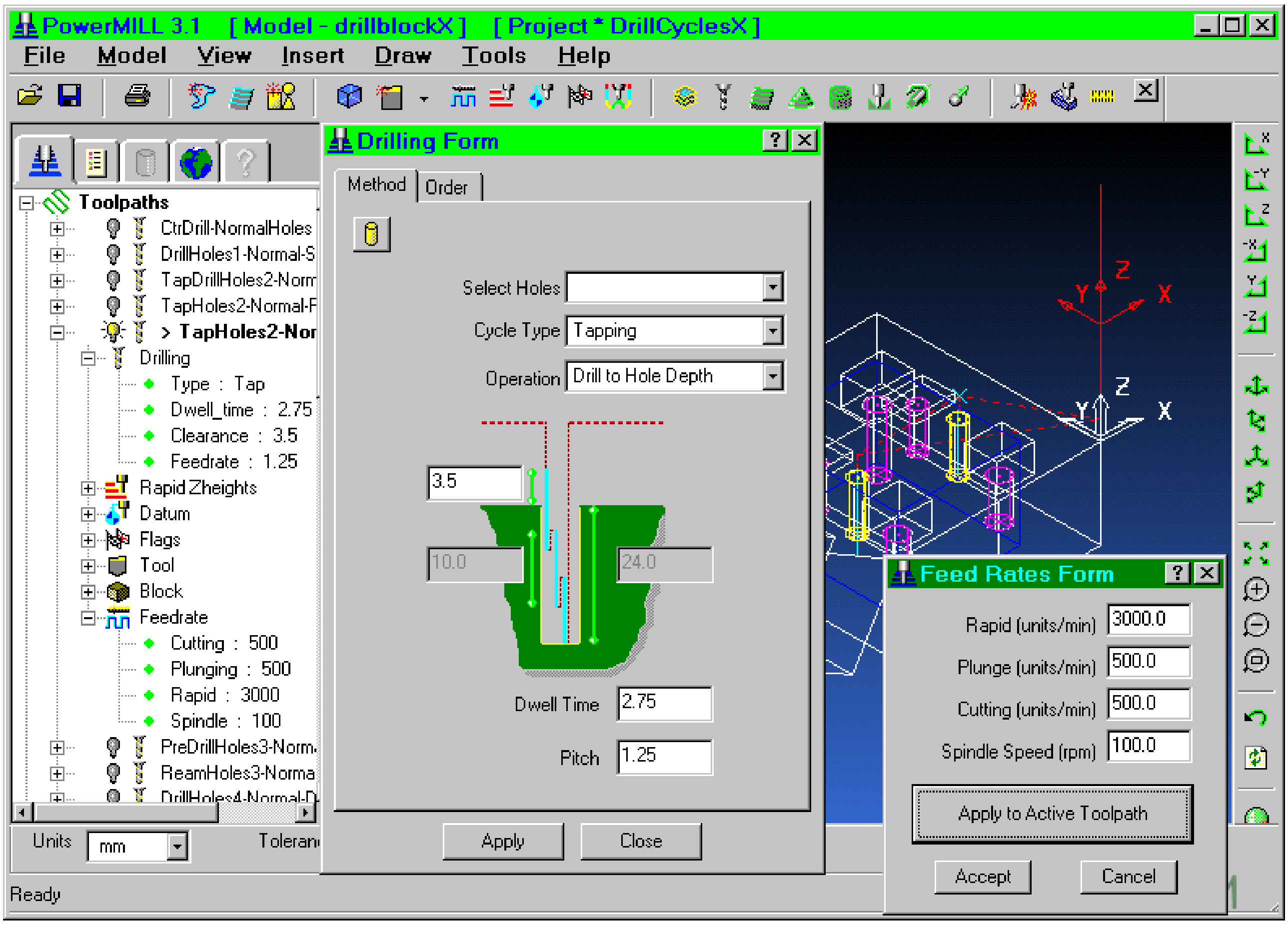
It will be noted in the above output that the first hole position is always output with the **G8X** cycle data line, even though the move to hole position proceeds this. It is entirely dependent on the machine control requirement whether or not this format is correct - some controls need the 1st hole position stated after the **G8X** line.

#

Note the difference of the **G84** Tapping feed rate compared to the previous, and the " ***integer 69*** " setting.

PowerMill Tapping Pitch Output PM3.0

Some machine controls require the Tapping feedrate to be output as a **F *Pitch*** and to obtain the Pitch setting from PowerMill Drill Form (see below)



PowerMILL 3.1   [Model - drillblockX]   [Project * DrillCyclesX]

Toolpaths
- CtrDrill-NormalHoles
- DrillHoles1-Normal-S
- TapDrillHoles2-Norm
- TapHoles2-Normal-F
- > TapHoles2-Nor
  - Drilling
    - Type : Tap
    - Dwell_time : 2.75
    - Clearance : 3.5
    - Feedrate : 1.25
  - Rapid Zheights
  - Datum
  - Flags
  - Tool
  - Block
  - Feedrate
    - Cutting : 500
    - Plunging : 500
    - Rapid : 3000
    - Spindle : 100
- PreDrillHoles3-Norm
- ReamHoles3-Norma
- DrillHoles4-Normal-D

Units mm

Drilling Form

Select Holes

Cycle Type Tapping

Operation Drill to Hole Depth

3.5

10.0

24.0

Dwell Time 2.75

Pitch 1.25

Feed Rates Form

Rapid (units/min) 3000.0

Plunge (units/min) 500.0

Cutting (units/min) 500.0

Spindle Speed (rpm) 100.0

Ready

The above example indicates the **Pitch = 1.25** (8mm Coarse Thread) - **Plunge rate 500 units/min** - **Spindle Speed 100 rpm,** of **the two latter** settings the ***speed*** for the size of Tap and Pitch should be carefully chosen as the **Feed Rate (cycfed)** is calculated by PowerMill using the formula ***Feed = Pitch x Spindle Speed***.

If we use the right hand example option file from above the following alterations will be needed :-

```
#
   define block cycle start
     :::::::::::::::::::::
     if ( cycle = = 4  )
       N ; G4 ; G6 ; X =C ; Y =C ; R2 ; Z2 ; P
cycledwell ;
    F cycfed ;
     end if
  end define
#
```

```
T4 M06
( 8.000mm Tap            )
G0 G90 X0 Y0 B0 S100 M03
G43 Z60. H4
X-25.001 Y44.999
G84 G98 X-25.001 Y44.999 Z-74. R-46.5 P2.75
F125
G80
G84 G98 X-75.001 Y44.999 Z-84. R-46.5 P2.75
F125
G80
```

```
  define word PF
     address letter  = "F"
    address width   = 1
     field width       = 3
     tape position    = 1
     modal
     metric formats
     decimal point     = true
     decimal places  = 2
     trailing zeros     = false
     imperial formats = metric formats
  end define
#
    word order  =  ( + P  PF )
#
  define block cycle start
    :::::::::::::::::::::
    if ( cycle = = 4  )
      N ; G4 ; G6 ; x coord =C ; y coord =C ;
           drill hole depth ; clearplane ; cycle dwell
;
           PF ( cycfed / ToolSpeed )
    end if
  end define
#
  define block move tap
     N ; x coord ; y coord ; M1 ; M2
   end define
#
```

```
T4 M06
( 8.000mm Tap            )
G0 G90 X-25.001 Y44.999 B0 S100 M03
G43 Z60. H4
G84 G98 X-25.001 Y44.999 Z-74. R-46.5 P2.75
F1.25
G80
G84 G98 X-75.001 Y44.999 Z-84. R-46.5 P2.75
F1.25
G80
```

Note :- We can't use the normal " **F** " word for the Tapping feed rate as a different format is required for the ***Pitch***, hence the need to create a new word for the **Pitch** calculation.

(Return to Pitch Variables Codes PM-Drilling TOP)

Drill Cycles are becoming more complex with later Machine controls.

Heidenhain (Controls TNC430 onward) & Ductpost1400 onwards

Heidenhain, and other machine tool control manufactures, have started to introduce more complex cycles, mainly as Manual programing functions, but customers expect CAM systems to output the format so that machine operators can modify the program at the machine without recourse to going back to the programmer.

To achieve these required formats is proving taxing, and it is becoming essential that all option file construction should adhere to certain fixed rules to ensure some conformity is maintained no matter, who is building them.

We have included the **OLD,** simpler type of PECKING cycle with the NEW which adds additional complexity that we consider maybe unjustifiable. (However, specific **exceptions** are made to this with CYL DEF 1.0 PECKING and the CYCL DEF 5.0 CIRCULAR POCKET format) (See Word document cycle-records.rtf)

Heidenhain CYCL DEF 2XX Cycles

New Words are required to be defined for these, and some of the old ones re-defined, however, it is possible that if the option includes Feedrate Parameter and Spline function then there is a likelihood of running out of available word capacity with Ductpost versions earlier than DP1414.
If this occurs it is suggested that the Spline function is dropped as Heidenhain do not recommend its use with their controls.
Ductpost1414 will have a **95** word capacity.
The Cycles used are the ones indicated in the PowerMill4> Drill Form (See Cycle Types)
We recommend the following new words and formatting in order to have some standardisation of use :-

(## Comments **only** - relate to PM Drill Form)

Words, formats, keys, codes, settings	Cycle Block definitions
# vvvvvvv CYCLE WORDS vvvvvvv # define word CQ address letter = "Q" address width = 1 field width = 3 not modal tape position = -3 metric formats leading zeros = false trailing zeros = true decimal point = false decimal places = 0 imperial formats = metric formats end define	define block cycle start if (swa) N ; " ;" N ; TPN ToolPathName N ; " ; TOOLPATH" ; WPN ToolPathWorkPlaneName N ; " ;" unset swa end if # N ; CYC Cycle ; ")" if (cycle == 1) N ; " CYCL DEF 1.0 PECKING" set swj else if (cycle == 2) N ; G4 0 ; " DRILLING~"

```
#
  define word CF
    address letter   = "="
    address width  = 1
    field width       = 5
    tape position    = 0
    print position    = 0
    not modal
    metric formats
    leading zeros    = false
    trailing zeros     = true
    decimal point    = false
    decimal places   = 0
    imperial formats = metric formats
  end define
#
  define word C1
    address letter = "="
    sign               = always
  end define
#
  define word C2
    address letter  = "="
    sign                 = none
  end define
#
  define word C3
    address letter   = "="
    address width  = 1
    field width       = 2
    tape position   = 0
    sign                 = none
    decimal point  = false
    decimal places = 0
    trailing zeros    = false
    imperial formats = metric formats
  end define
#
  define word C4
    address letter  = " "
    address width  = 1
    sign                  = if negative
  end define
#
   define word CYC
    address letter = "; (Cycle= "
    address width  = 10
    field width    = 2
    tape position  = 1
   end define
###   This last word is not normally required
```

```
      unset swj
    else if ( cycle == 3 )
      N ; G4 3 ; " UNIVERSAL DRILLING~"
      unset swj
    else if ( cycle == 4 )
      N ; G4 6 ; " TAPPING NEW~"
      unset swj
    else if ( cycle == 5 )
      N ; G4 1 ; " REAMING~"
      unset swj
    else if ( cycle == 6 )
      N ; G4 8 ; " BORE MILLING~"
      unset swj
    else if ( cycle == 7 )
      N ; " CYCL DEF 5.0 CIRCULAR POCKET"
      set swj
    else if ( cycle == 8 )
      N ; G4 14 ; " CIRCULAR POCKET FINISHING~"
      unset swj
    else if ( cycle == 9 )
      N ; G4 2 ; " BORING~"
      unset swj
    else if ( cycle == 10 )
      N ; G4 7 ; " RIGID TAPPING NEW~"
      set swi
      unset swj
    else if ( cycle == 11 or cycle == 12 )
      error "THIS IS NOT A SUPPORTED CYCLE-TURN
CYCLES OFF IN POWERMILL"
    else if ( cycle == 13 )
      N ; G4 5 ; " UNIVERSAL PECKING"
      unset swj
    end if
##-0
    if ( not swj )
      CQ 200 ; clearplane       ; E ; " CLEARPLANE~"
      CQ 201 ; drill hole depth ; E ; " HOLE DEPTH~"
      if ( not swi )
        CQ 206 ; CF Prat        ; E ; " PLUNGE FEEDRATE~"
      end if
    end if
#
##-1
    if ( cycle == 1 )
      N ; " CYCL DEF 1.1 SET UP"  ; C4 ( Clearplane -
HoleTop )
      N ; " CYCL DEF 1.2 DEPTH"   ; C4 ( HoleDepth -
HoleTop )
      N ; " CYCL DEF 1.3 PECKG"  ; C4 PeckDepth
      N ; " CYCL DEF 1.4 DWELL"  ; TL CycleDwell
      N ; " CYCL DEF 1.5" ; FF Prat
      but may be of help in debugging options.
#
#  vvvvvvv CYCLE WORDS Format vvvvvvv
#
  define format ( C1 C2 C4 )
    address width  = 1
    field width      = 8
    tape position   = 0
    print position   = 0
    not modal
    metric formats
    leading zeros   = false
    trailing zeros    = true
    decimal point   = true
    decimal places  = 3
    imperial formats = metric formats
  end define
#
  define format ( G4 )
    address letter   = "CYCL DEF 2"
    address width  = 11
    field width      = 2
    not modal
    leading zeros  = true
    imperial formats = metric formats
  end define
#
  define format ( G6 )
    field width      = 3
    decimal places = 1
    decimal point   = true
    not modal
    imperial formats = metric formats
  end define
#
  define format ( TL )
    address letter = " "
    address width  = 0
    field width    = 5
    sign          = none
    tape position  = 1
    metric formats
    decimal point  = true
    decimal places = 3
    trailing zeros = false
    imperial formats = metric formats
  end define
#
  define format ( Z2 )
    address letter  = "=-"
```

```
     edit intd 1
##-2
   else if ( cycle == 2 )
     CQ 202 ; Z3 PeckDepth           ; E ; " PECKING DEPTH~"
     CQ 210 ; DW 0                    ; E ; " DWELL TIME AT TOP~"
     CQ 203 ; C1 HoleTop             ; E ; " SURFACE COORDINATE~"
     CQ 204 ; C2 ( SafZ - Holetop ) ; E ; " 2ND SET-UP CLEARANCE~"
     CQ 211 ; DW CycleDwell =C   ; E ; " DWELL TIME AT BOTTOM "
##-3
   else if ( cycle == 3 )
     CQ 202 ; Z3 PeckDepth           ; E ; " PECKING DEPTH~"
     CQ 210 ; DW 0                    ; E ; " DWELL TIME AT TOP~"
     CQ 203 ; C1 Holetop             ; E ; " SURFACE COORDINATE~"
     CQ 204 ; C2 ( SafZ - Holetop ) ; E ; " 2ND SET-UP CLEARANCE~"
     CQ 212 ; C2 ( HoleDepth / NumberPecks ) ; E ; " PECK DECREMENT~"
     CQ 213 ; C3 ( NumberPecks / 2 )              ; E ; " NUMBER OF PECKS~"
     CQ 205 ; C2 ( NumberPecks / Word{TD} ) ; E ; " MIN. PECK DEPTH~"
     CQ 211 ; DW CycleDwell =C   ; E ; " DWELL TIME AT BOTTOM~"
     CQ 208 ; CF Srat                  ; E ; " RETRACT FEEDRATE~"
     CQ 256 ; C2 200                  ; E ; " CHIPBREAK DIST. "
##-4
   else if ( cycle == 4 )
     CQ 211 ; DW CycleDwell        ; E ; " DWELL TIME AT BOTTOM~"
     CQ 203 ; C1 Holetop             ; E ; " SURFACE COORDINATE~"
     CQ 204 ; C2 ( SafZ - Holetop ) ; E ; " 2ND SET-UP CLEARANCE"
##-5
   else if ( cycle == 5 )
     CQ 211 ; DW CycleDwell        ; E ; " DWELL TIME AT BOTTOM~"
     CQ 208 ; CF Srat                  ; E ; " RETRACT FEEDRATE~"
     CQ 203 ; C1 Holetop             ; E ; " SURFACE COORDINATE~"
```

```
    address width  = 2
    sign              = none
    tape position   = 0
    print position  = 0
  end define
#
  define format ( R2 Z3 )
    address letter  = "="
    address width  = 1
    sign               = none
    tape position   = 0
    print position  = 0
  end define
#
  define format ( R2 )
   scale factor   = 1
  end define
#
  define format ( DW )
    address letter   = "="
    address width  = 1
    field width      = 3
    sign               = none
    decimal places = 2
    print position   = 0
  end define
#
 define format ( E )
   address letter = ";"
   field width      = 0
   permanent
   not modal
   tape position  = -18
  end define
#
   word order = ( + CQ CF C1 C2 C3 C4 Q E DW )
#
  define keys
    cycle dwell    = DW
    dwell            not used
  end define
#
   block order              = true
   use hole top in cycles = true
   integer 69                 = 2
```

```
      CQ 204 ; C2 ( SafZ - Holetop ) ; E ; " 2ND SET-UP CLEARANCE"
##-6
    else if ( cycle == 6 )
      CQ 334 ; C2 ( Word{TD} / 3 )  ; E ; "DWELL TIME AT BOTTOM~"
      CQ 203 ; C1 Holetop               ; E ; "SURFACE COORDINATE~"
      CQ 204 ; C2 ( SafZ - Holetop ) ; E ; "2ND SET-UP CLEARANCE~"
      CQ 335 ; CF HoleDiameter       ; E ; "NOMINAL DIAMETER"
      CQ 342 ; CF 0                         ; E ; "ROUGHING DIAMETER"
##-7
    else if ( cycle == 7 )
      N ; " CYCL DEF 5.1 SET UP"       ; C4 ( Clearplane - HoleTop )
      N ; " CYCL DEF 5.2 DEPTH"        ; C4 ( HoleDepth - HoleTop )
      N ; " CYCL DEF 5.3 PLUNGING" ; C4 ( Prat / ToolSpeed ) ; FF Prat
      N ; " CYCL DEF 5.4 RADIUS"       ; C4 ( HoleDiameter / 2 )
      N ; " CYCL DEF 5.5" ; FF Frat ; " DR+"
      edit intd 1
##-8
    else if ( cycle == 8 )
      CQ 202 ; Z3 ( Word{TD} / 3 )  ; E ; "PLUNGING DEPTH~"
      CQ 207 ; CF Frat                    ; E ; "FEED RATE FOR MILLING~"
      CQ 203 ; C1 Holetop              ; E ; "SURFACE COORDINATE~"
      CQ 204 ; C2 ( SafZ - Holetop ) ; E ; "2ND SET-UP CLEARANCE~"
      CQ 216 ; C2 xcoord =C          ; E ; "CENTER IN 1st AXIS~"
      CQ 216 ; C2 ycoord =C          ; E ; "CENTER IN 2nd AXIS~"
      CQ 222 ; C2 Word{TD}          ; E ; "WORKPIECE BLANK DIA.~"
      CQ 223 ; C2 HoleDiameter       ; E ; "FINISHED PART DIA."
##-9
    else if ( cycle == 9 )
      CQ 211 ; DW cycledwell          ; E ; "DWELL TIME AT BOTTOM~"
      CQ 208 ; CF Srat                    ; E ; "RETRACT FEEDRATE~"
      CQ 203 ; C1 Holetop              ; E ; "SURFACE
```

```
COORDINATE~"
       CQ 204 ; C2 ( SafZ - Holetop ) ; E ; "2ND SET-UP CLEARANCE~"
       CQ 214 ; C3 1                          ; E ; "DISENGAGING DIRECTN~"
       CQ 336 ; C3 0                          ; E ; "ANGLE OF SPINDLE"
##-10
     else if ( cycle == 10 )
       CQ 239 ; C2 ( Cycfed / ToolSpeed[ToolNum] ) ; E ; " THREAD PITCH~"
       CQ 203 ; C1 Holetop                        ; E ; " SURFACE COORDINATE~"
       CQ 204 ; C2 ( SafZ - Holetop )        ; E ; " 2ND SET-UP CLEARANCE"
       unset swi
##-13
     else if ( cycle == 13 )
       CQ 202 ; Z3 PeckDepth            ; E ; " PECKING DEPTH~"
       CQ 203 ; C1 Holetop                 ; E ; " SURFACE COORDINATE~"
       CQ 204 ; C2 ( SafZ - Holetop ) ; E ; " 2ND SET-UP CLEARANCE~"
       CQ 212 ; C2 ( HoleDepth / NumberPecks )  ; E ; " PECK DECREMENT~"
       CQ 205 ; C2 ( NumberPecks / Word{TD} ) ; E ; " MIN. PECK DEPTH~"
       CQ 258 ; C2 500                        ; E ; " UPPER ADV STOP DIST~"
       CQ 259 ; C2 1000                      ; E ; " LOWER ADV STOP DIST~"
       CQ 257 ; C2 ( PeckDepth / 2 )  ; E ; " DEPTH FOR CHIP BRKNG~"
       CQ 256 ; C2 200                         ; E ; " CHIPBREAK DIST.~"
       CQ 211 ; DW CycleDwell         ; E ; " DWELL TIME AT BOTTOM~"
     end if
   end define
#
    define block move cycle
      if ( Cycle = = 1 or Cycle = = 7 and intd = = 1 )
        N ; G1 ; x coord ; y coord ; FMAX
        N ; G1 ; z coord Clearplane ; FMAX
        N ; G1 ; x coord =C ; y coord =C ; RR 0 ; FMAX ; " M99"
        edit intd 0
      else
        N ; G1 ; x coord =C ; y coord =C ; RR ; FMAX ; " M99"
      end if
```

```
 end define
#
  define block move tap
    N ; G1 ; x coord =C ; y coord =C ; RR ; FMAX ; " M99"
  end define
#
  define block cycle end
    N ; G1 ; z coord Saf Z =C ; RR ; FMAX
  end define
#
 Note :-  M99 must be a string, not M1 99 or M2 99
```

The above provides the following output :- (See illustrated pictures as a reference) **NOTE:** This is a TIF file and will issue a warning - it is safe to use.

These are ChipBreak Drilling and Deep Drill 2 Cycles

```
79 L X-3.188 Y-10.79 FMAX
80 ; CYCLE=( 2 ) ## Indicator of cycle only, can be removed
81 CYCL DEF 200 DRILLING~
  Q200=5.000       ; CLEARPLANE~
  Q201=-36.949     ; HOLE DEPTH~
  Q206=333         ; PLUNGE FEEDRATE~
  Q202=36.950      ; PECKING DEPTH~
  Q210=0.0         ; DWELL TIME AT TOP~
  Q203=+0.0        ; SURFACE COORDINATE~
  Q204=55.290      ; 2ND SET-UP CLEARANCE~
  Q211=3.25        ; DWELL TIME AT BOTTOM
82 L X-3.188 Y-10.79 FMAX M99
83 L Z55.29 FMAX
:::::::::::::::::::::::::::
:::::::::::::::::::::::::::
98 L X10.0 Y16.5 FMAX M03
99 L Z20.0 FMAX M08
100 L Y12.5 FMAX
101 ; CYCLE=( 13 )## Indicator of cycle only, can be removed
102 CYCL DEF 205 UNIVERSAL PECKING
  Q200=4.999       ; CLEARPLANE~
  Q201=-5.000      ; HOLE DEPTH~
  Q206=750         ; PLUNGE FEEDRATE~
  Q202=1.500       ; PECKING DEPTH~
  Q203=+0.0        ; SURFACE COORDINATE~
```

These are the *EXCEPTIONS* - and use the Single Peck and Bore 3 Cycle

```
53 L X18.0 Y2.34 FMAX
54 ; CYCLE=( 1 )## Indicator of cycle only, can be removed
55 CYCL DEF 1.0 PECKING
56 CYCL DEF 1.1 SET UP 5.000
57 CYCL DEF 1.2 DEPTH -36.950
58 CYCL DEF 1.3 PECKG 36.950
59 CYCL DEF 1.4 DWELL 1.5
60 CYCL DEF 1.5 F333
61 L Z5.0 FMAX
62 L X18.0 Y2.34 R0 FMAX M99
63 L Z55.29 FMAX
::::::::::::::::::::
::::::::::::::::::::
68 L Y16.5 FMAX
69 L X65.0 Y12.5 FMAX M08
70 ; CYCLE=( 7 )## Indicator of cycle only, can be removed
71 CYCL DEF 5.0 CIRCULAR POCKET
72 CYCL DEF 5.1 SET UP 5.000
73 CYCL DEF 5.2 DEPTH -5.000
74 CYCL DEF 5.3 PLUNGING 1.500 F750
75 CYCL DEF 5.4 RADIUS 5.000
76 CYCL DEF 5.5 F2000 DR+
77 L Z5.0 FMAX
78 L X65.0 Y12.5 R0 FMAX M99
79 L Z20.0 FMAX
```

```
  Q204=20.000       ; 2ND SET-UP
CLEARANCE~
  Q212=0.132        ; PECK DECREMENT~
  Q205=6.333        ; MIN. PECK DEPTH~
  Q258=0.500        ; UPPER ADV STOP
DIST~
  Q259=1.000        ; LOWER ADV STOP
DIST~
  Q257=0.750        ; DEPTH FOR CHIP
BRKNG~
  Q256=0.200        ; CHIPBREAK DIST.~
  Q211=0.0          ; DWELL TIME AT
BOTTOM~
105 L X10.0 Y12.5 FMAX M99
106 L Z20.0 FMAX
```

NOTES :-

1/ **Helical** and **Helical 2** are not strictly cycles and are treated as Linear movement by PowerMill and Ductpost, therefore the ***Drilling Cycle Output*** check box in the NC edit form must be **OFF** for these two functions.

2/ **Word{TD}** = ToolDiameter and is extracted from the Tool Change block data. It **MUST** be defined as **TD (ToolRadius[ToolNum] * 2)** and **NOT** as previously by the **word TD scaled by 2**.

3/ It is probable that additional cycle functions will be added later and will follow on from **cycle 13**

4/ The defined cycles are not cast in stone and can be changed as felt neccessary, and will be refined as required from feedback from users. However, we would advise that words given above are used wherever possible if and when a change is made to try and avoid confusion.

Return to (Complex Cycle Types TOP)

Siemens Drilling Cycles

Siemens drilling cycles are nearly as complex as Hiedenhain though not as informative friendly in format output.

The following is a breakdown of the codes applicable to the various cycle functions and they must be placed in the order indicated for each particular cycle for the cycle to work correctly. Siemens do not use a "**Q**" values as Hiedenhain, but a " ***comma*** " to separate each function, e.g. **,** clearplane **,** which can be a problem if a function is a zero value as then **, ,** is acceptable and **must** still be included in the sequence.
The following is an example output which has been artificially spaced to correspond to the individual sector. :-
(It is suggested that the MSG line is output to the NC program as a guide for the operator, it can always be suppressed if not wanted.)

Example of NC output :-

```
N12           MSG(RTP, RFP,SDIS, DP   ,DPR   , FDEP,FDPR,DAM ,DTB ,DTS,FRF,VARI)
N13 MCALL CYCLE83(10.,-20.,5. ,-81.322,-61.322,-30.5,10.5,1.75,3.55,   , 1 ,  1 )
N141           MSG(RTP,RFP,SDIS,DP ,DPR,DTB,DTS,, PIT,SST ,SST1")
N142 MCALL CYCLE84(10.,0. ,3.  ,-45.,45.,1.5,  , , 1.5,3000,4500 )
```

Note : CYCLE84 is for Rigid Tapping (Cycle10) : CYCLE840 is for Floating Tap holder (Cycle 4)

Reference Table to codes used :-

Code	=	Reference	Code	=	Reference	Code	=	Reference
RTP	=	Absolute SafeZ Height	RFP	=	Absolute Holetop Surface Coordinate	SDIS	=	Incremental Clearplane Height
DP	=	Drilled Hole Point Depth	DPR	=	Full Hole Depth	FDEP	=	1st.Abs.Peck Coordinate
FDPR	=	Incremental Peck Depth	DAM	=	Degression distance	DTB	=	Cycle Dwell at Hole Bottom
DTS	=	Cycle Dwell at Hole Top	FRF	=	1st.Peck Feedrate Variation Factor	VARI	=	0 = ChipBreak. 1 = Swarf Out
		Tapping Codes :-						
PIT	=	Thread Pitch	SST	=	Tapping Tool Speed (RPM)	SST1	=	Retract Tool Speed (RPM)
			SDAC	=	Spindle Rotation Direction after Cycle			

Words, formats, keys, codes, settings	**Cycle Block definitions**
# vvvvvvvv CYCLE Words vvvvvvvv # define word RFP address letter = "," address width = 1 field width = 8 sign = if negative metric formats decimal point = true decimal places = 3 leading zeros = false	# define block cycle start if (swa) N ; TPN ToolPathName ; " \")" unset swa end if N ; G3 ; feedrate Cycfed ; CYC Cycle ; ")" if (cycle = = 1 or cycle = = 5) N ; " MSG (RTP , RFP , SDIS , DP , DPR)"

```
      trailing zeros = false
    end define
#
    define word DP
       address letter = ","
       address width  = 1
       sign           = if negative
    end define
#
    define word Q1
       address letter = ","
       address width  = 1
    end define
#
   define word PT
      address letter = " , "
      address width  = 3
      field width    = 5
      sign           = none
      metric formats
      decimal point  = true
      decimal places = 3
      leading zeros  = false
      trailing zeros = false
   end define
#
   define word ST
      address letter = ", "
      address width  = 2
      field width    = 5
      sign           = none
      metric formats
      decimal point  = false
      decimal places = 0
      leading zeros  = false
      trailing zeros = true
   end define
#
   define word ST1
      address letter = ", "
      address width  = 2
      scale factor   = 3
      scale divisor  = 2
      sign           = none
   end define
#
   define word PF
      address letter   = "F=R10"
      address width  = 5
      field width      = 1
      modal
```

```
**
       N ; G4 ; R11 SafeZ ; RFP HoleTop ; clearplane ; DP
HoleDepth ; drill hole depth ; ")"
     else if ( cycle = = 2 )
       N ;  MSG (RTP   ,RFP   , SDIS  , DP  , DPR , DTB  )" **
       N ; G4 ; R11 SafeZ ; RFP HoleTop ; clearplane ; DP
HoleDepth ; drill hole depth ;
             cycle dwell ; ")"
     else if ( cycle = = 3 )
       N ; "  MSG (RTP  ,RFP  ,SDIS  ,DP  ,DPR  ,FDEP  ,FDPR  ,DAM  ,DTB
,DTS ,FRF ,VAR )" **
       N ; G4 ; R11 SafeZ ; RFP HoleTop ; clearplane ; DP
HoleDepth ; drill hole depth ;
            Q ( HoleTop - PeckDepth ) ; Q1 PeckDepth ; ", 1" ; cycle
dwell ; ", , 1, 0 )"
     else if ( cycle = = 4 )
       N ; "  MSG (RTP  ,RFP  ,SDIS  ,DP  ,DPR  ,DTB  ,DTS,  , )" **
       N ; G4 840 ; R11 SafeZ ; RFP HoleTop ; clearplane ; DP
HoleDepth ; drill hole depth ;
              cycle dwell ; ", ," ; PT ( Cycfed / ToolSpeed ) ; ST
ToolSpeed[ToolNum] ;
              ST1 ToolSpeed[ToolNum] ; " )"
     else if ( cycle = 10 )
       N ; "    MSG (RTP  ,RFP  ,SDIS  ,DP  ,DPR  ,DTB  ,SDAC,  ,PIT,  , SST
, SST1 )" **
       N ; G4 84 ; R11 SafeZ ; RFP HoleTop ; clearplane ; DP
HoleDepth ; drill hole depth ;
              cycle dwell ; " 3, ," ; PT ( Cycfed / ToolSpeed ) ; ST
ToolSpeed[ToolNum] ;
              ST1 ToolSpeed[ToolNum] ; " )"
     else if ( cycle = 13 )
       N ; "   MSG
(RTP  ,RFP  ,SDIS  ,DP  ,DPR  ,FDEP  ,FDPR  ,DAM  ,DTB ,DTS ,FRF ,VAR
)" **
       N ; G4 73 ; R11 SafeZ ; RFP HoleTop ; clearplane ; DP
HoleDepth ; drill hole depth ;
             Q ( HoleTop - PeckDepth ) ; Q1 PeckDepth ; ", 1" ; cycle
dwell ; ", , 1, 0)"
     end if
   end define
#
### These lines can be omitted  **
#
 define block move cycle
     N ; x coord =C ; y coord =C
    end define
#
   define block move tap
     N ; x coord =C ; y coord =C
   end define
#
```

```
   end define
#
    define word CYC
      address letter   = "MSG
(Cycle= "
      address width  = 11
      field width       = 2
      tape position   = 1
    end define
###   This last word is not
normally
      required but may be of help in
      debugging options.
#
# vvvvv CYCLE WORD Formats
vvvv
#
   define format ( G4 )
      address letter  = "MCALL
CYCLE"
      address width   = 11
   end define
#
   define format ( ST1 )
      field width        = 5
      metric formats
      decimal point   = false
      decimal places = 0
      leading zeros   = false
      trailing zeros    = true
   end define
#
   define format ( R11 )
      address letter   = "("
      address width  = 1
      sign                 = if negative
   end define
#
   define format ( Z2 R2 Q P )
      address letter = ", "
      address width  = 1
   end define
#
   define format ( R11 DP P Q Q1
TD )
      field width    = 5
      metric formats
      decimal point  = true
      decimal places = 3
      trailing zeros = false
      imperial formats = metric
```

```
   define block cycle end
     N ; " MCALL"
     N ; rapid =C ; z coord SafZ ; coolant off
   end define
#
```

```
formats
  end define
#
word order =  ( + RFP DP Q1 CYC
PT ST ST1 )
#
  define keys
    cycle dwell    = P
  end define
#
   use hole top in cycles = true
   integer 69                = 2
#
```

The above Siemens 840D cycles are still in proving trials and may well require ajustment for the particular machine control so be careful!

Cincinnatti Acramatic control Cycles

These Cycles have been developed in conjunction with Cincinnatti Milacron in the UK but again we have had little feed back from them to say they work so use them as a basis for development.

Words, formats, keys, codes, settings

```
  define format ( P )
    address letter =
"[$CYCLE_PARAMS(2)G8"
    address width  = 19
    field width    = 1
  end define
#82
  define format ( null )
    field width    = 3
    metric formats
    decimal point  = true
    decimal places = 1
    trailing zeros = true
  end define
#90
  define format ( L )
    address letter = "W"
    not modal
  end define
#95
  define format ( Q )
    address letter = "K"
  end define
#99
```

Cycle Block definitions

```
  define block cycle start
    if ( swa )
      N ; TPN ToolPathName ; " )"
      unset swa
    end if

    if ( ToolSpeed[ToolNum] > 2000 )
      N ; S 2000
    end if

    if ( cycle == 1 )
      N ; G4 ; x coord =C ; y coord =C ; drill hole
depth HoleDepth ;
            clearplane HoleTop ; L SafeZ ; feedrate Prat
; M1 ; M2
    else if ( cycle == 2 )
      N ; P cycle ; "_DWELL]=" ; null cycledwell
      N ; G4 ; x coord =C ; y coord =C ; drill hole
depth HoleDepth ;
            clearplane HoleTop ; L SafeZ ; feedrate Prat
; M1 ; M2
    else if ( cycle == 3 )
      N ; P cycle ; "_DWELL]=" ; null cycledwell
      N ; G4 ; x coord =C ; y coord =C ; drill hole
  define format ( O )
   address letter = "J"
   field width    = 2
  end define
#104
  define format ( L Q )
    field width    = 8
    tape position  = 1
    metric formats
    decimal point  = true
    decimal places = 3
    trailing zeros = false
  end define
#113
   word order = ( + P L O null )
#
  define keys
    cycle dwell    = P
    dwell           not used
  end define
#
   use hole top in cycles = true
   integer 69                  = 2
#
```

```
depth HoleDepth ;
            clearplane HoleTop ; L SafeZ ; drill peck
depth ; O 13 ;
            feedrate Prat ; M1 ; M2
    else if ( cycle == 4 )
      N ; P cycle ; "_DWELL]=" ; null cycledwell
      N ; G4 ; x coord =C ; y coord =C ; drill hole
depth HoleDepth ;
            clearplane HoleTop ;  L SafeZ ; O 2 ;
feedrate Prat ; M1 ; M2
    else if ( cycle == 5 )
      N ; G4 ; x coord =C ; y coord =C ; drill hole
depth HoleDepth ;
            clearplane HoleTop ; L SafeZ ; O 2 ;
feedrate Prat ; M1 ; M2
    else if ( cycle == 6 )
      N ; P cycle ; "_BOT_RET]=" ; null 0
      N ; G4 ; x coord =C ; y coord =C ; drill hole
depth HoleDepth ;
            clearplane HoleTop ; L SafeZ ; "U-0.5V-
0.5" ; O 0 ; feedrate Prat ; M1 ; M2
    else if ( cycle == 7  or cycle == 8 or cycle == 9 )
      error "THESE CYCLES G87 G88 G89 NOT
SUPPORTED"
    else if ( cycle == 10 )
      N ; P 4 ; ".1_DWELL]=" ; null cycledwell
      N ; G4 84 ; ".1" ; x coord =C ; y coord =C ; drill
hole depth HoleDepth ;
            clearplane HoleTop ; L SafeZ ; O 2 ; drill
peck depth ; "P1" ;
            feedrate Prat ; M1 ; M2
    else if ( cycle == 11 or cycle == 12 )
      error "HELICAL MILLING CYCLES NOT
SUPPORTED"
    else if ( cycle == 13 )
      N ; P 3 ; "_SHRT_RET]=" ; null ( PeckDepth /
2 )
      N ; P =C ; "_DWELL]=" ; null cycledwell
      N ; G4 83 ; x coord =C ; y coord =C ; drill hole
depth HoleDepth ;
            clearplane HoleTop ; L SafeZ ; drill peck
depth ; O 2 ; feedrate Prat ; M1 ; M2
    end if
  end define
#317
  define block move cycle
    if ( xcoord == OldX and ycoord == OldY )
      N ;
    else
      N ; x coord ; y coord ; L SafeZ
    end if
  end define
#325
  define block move tap
    if ( xcoord == OldX and ycoord == OldY )
      N ;
    else
      N ; x coord ; y coord ; L SafeZ
    end if
  end define
#333
  define block cycle end
    N ; end of drill ; z coord
  end define
#337
```

Return to :- Hiedenhan , Siemens ; TOP

Tool length offset

(Up-dated 29/12/2000)

The ***tool length offset code*** is initiated by the APT LOADTL command in DUCT or POWERMILL. It normally appears in the NC program as **H**, (or occasionaly D) on the first move after a tool change, usually required by most controls to be on the Z move along with an appropriate G code. (e.g. G43 Z12. H1)
(This means that at the end of the first Z move the machine is correctly positioned with tool length compensation in force)

To set this up simply define a word with a suitable format and set the key. (This is normally already done in the source file but worth a check)

```
define keys
  tool length = H
end define
```

The word **tool length (H),** along with G6, must also be included in the *define block move rapid* definition.

```
define block move rapid
   N ; rapid ; G6 ; x coord ; y coord ; z coord ; tool length ; spindle ; M1
end define
```

It will also require the following flags to be defined to obtain the desired output :-

```
tool reset coordinates   =  3
split move               =  0
tlo output               = true
```

with the above settings the following output can be expected :-

```
%
:0001
G91 G28 X0 Y0 Z0
G40 G17 G80 G49
G0 G90 Z49.5
T1 M6
X81. Y48.5 S1500 M3
G43 Z49.5 H1
X12.446 Y10.931
Z44.5
G1 Z44.4 F2500
```

If we change the **tool reset coordinates = 3** to **= 4**

we get the following :-

```
T1 M6
G43 Z49.5 S1500 H1 M3
X81. Y48.5
X12.446 Y10.931
Z44.5
G1 Z44.4 F2500
```

so by adjusting the value, differing output format can be obtained. (This is not always guaranteed especially with multi-axes options and other subterfuges may well have to be employed to obtain the required format)

To obtain the G43 code output, these additional settings are required, though they may already be defined in the source file.

```
    define codes
      tool len off  = G6 43
    end define
```

and **G6** must be in the *define block move rapid* as shown above.

Variations :-

If the **H** word is required on the tool change line it should be defined in the tool definition block with the appropriate variable :

1/

```
    define block tool change
      N ; tool number ; tool length ToolNum ; change tool
    end define

   tlo output   = false    ## This prevents the output of the normal function
```

provides the following :-

```
     %
     :0001
     G91 G28 X0 Y0 Z0
     G40 G17 G80 G49
     G0 G90 Z49.5
    T1 H1 M6
     Z49.5 S1500 M3
     X81. Y48.5
```

Sometimes the T word is used as the tool number and offset number (e.g. T03.50 means tool 3 offset length 50 mm). This can be achieved by using the tool change definition below and setting the format of **H** as follows:

2/

```
   define format ( H )
     address letter = "."
     tape position  = 0
    end define

    define block tool change
      N ; tool number ;  tool length ToolLength ; change tool
    end define
```

which produces :-

```
     %
     :0001
     G91 G28 X0 Y0 Z0
     G40 G17 G80 G49
     G0 G90 Z49.5
    T1.50 M6
     Z49.5 S1500 M3
     X81. Y48.5
```

If it is necessary to remove the offset before a tool change this can also be done in the tool change definition.

```
define block tool change
  N ; G6 49 ; tool length  0
  N ; tool number ; change tool
  N ; tool length offset ; z coord FromZ ; tool length ToolNum
end define
```

this gives:

```
 G0 Z49.5
G49 H0
 T3 M6
G43 Z49.5 H3
 Z49.5 S1500 M3
 X12.446 Y10.931
```

There are many other variations that can be used, and in multi-axes options the *define block move linear* is also involved, but this should at least give some idea how to tackle this requirement in general.

(Return to TOP)

Setting up a 4 axes option file

(Up dated 31/01/2001)

Some 4 axes examples to illustrate the various possibilities are given here :- (4th Rotary Axis)

Requirements for 4th Rotary axis

The following inclusions are required to be defined in an option file for Rotary Multi-axes working.

(The three Main rotary axis definitions are being defined in this example, A , B , ? C , but only **one** will actually be used for a single rotary axis.)

```
define format ( A B C )    ## May well be defined already in the inbuilt source
    metric formats
    leading zeros      = false
    trailing zeros      = true
    decimal point      = true
    decimal places     =   3
    imperial formats
    leading zeros       = false
    trailing zeros      = true
    decimal point      = true
    decimal places     =   4
end define

    word order  =  ( + A B C )   ## Only needed if not in inbuilt word order list

    block order = true              ## Override for inbuilt word order listing, uses "define block xxx " order

define keys
   azimuth axis      not used    ##  Not required for single rotary axis
   elevation axis    =  A        ##  4th rotary axis normally an elevation ( e.g. A rotates about X )
end define
```

## " **elevation axis** ", will cover A or B, or C " requires to be inserted in both Rapid, and Linear, blocks

```
define block move rapid
    N ; rapid ; G2 ; G3 ; G6 ; x coord ; y coord ; z coord ; elevation axis ; spindle ; tool length ; M1 ; M2
end define

define block move linear
    N ; linear ; G2 ; x coord ; y coord ; z coord ; elevation axis ; feedrate ; M1 ; M2
end define
```

Rotary axes parameter settings

The following parameters will need to be included in ***any*** rotary axes option.

```
azimuth axis parameters    = ( 0.  0.  0.   1.  0.  0. )
elevation axis parameters  = ( 0.  0.  0.   0.  1.  0. )
```

The first three figures in each case give the *centre of rotation* (usually **0. 0. 0.** for ***single*** rotary tables).

The second set of three figures define the *axis of rotation* about either X, Y, or Z axis.
(It is essential that both axes are not aligned to the same axis. The **elevation** is the ***main*** alignment for the **4th** axis)

Thus the example shown above represents a machine where the ***azimuth*** rotation is set to be round the X axis (vector **1. 0. 0.**), normaly for 4 axes these can remain at **zero**.

The ***main elevation*** rotation is round the Y axis (vector **0. 1. 0.**).

4th Rotary Axis is a swivel tool head unit.

The following will need to be included for this type of rotary axis.

```
spindle elevation rotation   =  true     ## true = Spindle rotation - false = Table rotary unit ( default )
elevation centre             =  (  0.   0.   180.5  )    ## Only used for Spindle offset from centre of rotation,
usually in the Z axis
```

(NOTE :- These figures must be written with points. (**0.** not 0)

The distance entered should be the difference between the the ***centre of rotation*** and the face of the tool holder. (In the example above the distance is **180.5** mm in the Z axis)
The distance between the top of the tool holder and the tool tip can then be entered using the tool length setting in DUCT, or PowerMILL.

NOTE :- Many machines with tilting tools have some form of code on the machine which will allow the tool to be run as though the ***centre of rotation*** were the centre of the tool tip. In this case set the centre offset to zero.

==

Additional parameters

```
elevation axis units          =  degrees    ## Usual units ( Default could be none )
elevation axis direction      =  positive    ## Normal rotational direction required. ( Default could be none )
pcs origin                    =  ( 0   0   0    0   0   0 )  ## ( Default, rarely used, mainly for 5 axes )

                   ## Azimuth Min   Max    Elev'n Min    Max      Tol.   Moves
rotary axis limits  = ( -99999  99999       -99999     99999     0.1      1  )    ## Default values, virtually unlimited
rotation
```

EXAMPLE ;-

```
rotary axis limits  = (  0    0    -720.0    720.0     0.1      1  )    ## Elevation limited to two table revolutions
```

Tolerance value (Tol.) is the tolerance to which the angular displacement is maintained by Ductpost in the number of Moves set.   Keep tolerance and moves to the highest and lowest values respectively ,conducive to good machine performance.   (FROM DP1331 the move fuction (last digit) has been superceeded by " ***linearise multiaxis moves = true*** " which automatically adjusts the number of moves needed to achive the tolerance setting)

See also 4-axes examples

Back to :- Top : Parameters

Setting up a 5 axes Table option file

(Up dated 10/10/2001)

An example to illustrate a type of 5 axes Table is given here :- (Rotary Axes Table)

Requirements for 4th and 5th Rotary axes

The following inclusions are probably required to be defined in an option file for Rotary Multi-axes working :-

(The three Main rotary axis definitions are being defined in this example, **A** , **B** , and **C** , but **only two** will actually be used for any multi-rotary axes machine.system)

```
        define format ( A B C )    ## May well be defined already in the inbuilt source
            metric formats
            leading zeros      = false
            trailing zeros     = true
            decimal point      = true
            decimal places     =   3
            imperial formats
            leading zeros      = false
            trailing zeros     = true
            decimal point      = true
            decimal places     =   4
          end define
#
           word order  =  ( + A B C )   ## Only needed if not in inbuilt word order list
#
           block order =  true          ## Override for word order listing, uses "define block .... " order
#
          define keys
            azimuth axis     =  A    ## 4th rotary axis normally an azimuth  ( Table tilt axis )
            elevation axis   =  C    ## 5th rotary axis normally an elevation  ( Rotary table on tilt table )
          end define
```

## " azimuth axis and elevation axis (**A**, **and /or B**, **and /or C** " requires to be inserted in both **Rapid**, and **Linear**, blocks depending upon axis alignments.

(Using the pictorial example No.1, **A** tilts about **X**, **C** rotates about **Z**)

```
#
          define block move rapid
            N ; rapid ; G2 ; G3 ; G6 ; x coord ; y coord ; z coord ; azimuth axis ; elevation axis ; spindle ; tool length
 ; M1 ; M2
          end define
#
          define block move linear
            N ; linear ; G2 ; x coord ; y coord ; z coord ; azimuth axis ; elevation axis ; tool radius ; feedrate ; M1 ;
 M2
          end define
```

Rotary axes parameter settings

The following parameters will need to be included in the rotary axes option.

```
          azimuth axis parameters   = (  0.  0.  0.   1.  0.  0. )
          elevation axis parameters = (  0.  0.  0.   0.  0.  1. )
```

The first three figures in each case position the ***centre of rotation***, the ***azimuth tilt table*** usually has no offset from the centre line of the elevation axis, (therefore these will be set at **0. 0. 0.**) but the ***elevation table surface centre*** could be either **on**, **above**, or **below**, the ***centre line of the tilt table*** axis.

In the pictorial example No.1 the rotary ***elevation table surface*** is **below** the ***azimuth tilt table*** **axis centre line**, and for the post processor to account for this **offset**, the difference is set in the appropriate ***elevation axis parameter***. The sign of this value will depend upon **where** the ***Machine Centre of Rotation*** has been defined by the manufacturer, and is usually on the ***Centre Line of the Tilt Table Rotary axis.***

The settings for an **offset** of say, 80.56 mm. would be as follows :-

elevation axis parameter = (**0. 0. 80.56** [***positive*** value for **below**, ***negative*** value for **above**, ***zero*** if surface **on** the Centre Line of azimuth rotation])

If the *Machine Centre of Rotation* is defined at the **Surface** of the the ***elevation table*** then the **signs are reversed**.

The second set of three figures define the axis of rotation about which the table rotates, either X, Y, or Z axis. (It is essential that both axes are not aligned to the same axis. The azimuth is nominally the **main** alignment)

Thus the example shown represents a machine where the **azimuth** rotation is rotating around the **X** axis (vector **1. 0. 0.**), and the **elevation** rotation is rotating around the **Z** axis. (vector **0. 0. 1.**)

(NOTE 1 :- These figures can be written with, or without points. (**0.**) or (**0**)

(NOTE 2 :- Some machine controls may have some form of **code** that will make the machine control work out it's own Rotational Transformations which allows the post processor to ignore any offset. In this case set the centre offsets to ***zero***.

Additional parameters

```
     spindle azimuth rotation   = false      ## Default for table unit
     azimuth axis units         = degrees    ## Usual units ( Default could be none )
     azimuth axis direction     = positive   ## Normal rotational direction required. ( Default could be none )
#
     spindle elevation rotation  = false     ## Default for table unit
     elevation axis units        = degrees   ## Usual units ( Default could be none )
     elevation axis direction    = positive  ## Normal rotational direction required. ( Default could be none )
#
     pcs origin                 = ( 0  0  0   0  0  0 )  ## ( Default, rarely used, sometimes in 5 axes spindle alignments )
#
                    ## Azimuth Min  Max.   Elev'n Min   Max.    Tol.  Moves
     rotary axis limits  = ( -99999  99999    -99999    99999    0.1    1 )   ## Default values, virtually unlimited rotation
#
 ( Example settings ) :-

   rotary axis limits  = (  -95.0   10.0    -3600.0    3600.0    0.1    4 )   ## Azimuth Tilt Table limited swing values, and Elevation rotary table +/- 10 revolutions)
#
     linearise multiaxis moves        = true    ## Default setting ( From DP1321 see What's New 5-axes improvements )
     multiaxis coordinate transform  = true    ## Default setting
#
```

Tolerance value (Tol.) is the tolerance to which the angular displacement is maintained by Ductpost within the number of Moves set, to achieve a good linearisation. Keep tolerance and moves to the ***highest*** and ***lowest*** values respectively conducive to good machine performance.

(**FROM DP1321** the move fuction (last digit) has been superceeded by " ***linearise multiaxis moves = true*** " which automatically adjusts the number of moves needed to achive the tolerance setting)

See also 5-axes Table examples

(Back to :- Top : Parameters)

Setting up a 5 axes Head/Table option file

(Up dated 31/01/2001)

An example to illustrate a type of 5 axes Head/Table is given here :- (Rotary Axes Head/Table)

Requirements for 4th and 5th Rotary axes

The following inclusions are probably required to be defined in an option file for Rotary Multi-axes working.
(The three Main rotary axis definitions are being defined in this example, **A** , **B** , **C** , but **only two** will actually be used for multi-rotary axes systems.)

```
define format ( A B C )    ## May well be defined already in the inbuilt source
    metric formats
    leading zeros      = false
    trailing zeros     = true
    decimal point      = true
    decimal places   =   3
    imperial formats
    leading zeros      = false
    trailing zeros     = true
    decimal point      = true
    decimal places   =   4
end define

word order  =  ( + A B C )   ## Only needed if not in inbuilt word order list
block order =  true          ## Override for word order listing, uses "define block xxx " order

define keys
   azimuth axis     =  A      ## 4th rotary axis normally an azimuth  ( Spindle Rotary )
   elevation axis   =  C     ## 5th rotary axis normally an elevation ( Rotary table  )
end define
```

## " **A** , **and/or B**, **and/or C** " requires to be inserted in both **Rapid**, and **Linear**, blocks depending upon axis alignments. (Using the pictorial example, **A** rotates about **X**, **C** rotates about **Z**)

```
define block move rapid
    N ; G1 ; G2 ; G3 ; G6 ; X ; Y ; Z ; A ; C ; S ; H ; M1 ; M2
end define

define block move linear
    N ; G1 ; G2 ; X ; Y ; Z ; A ; C ; F ; M1 ; M2
end define
```

Rotary axes parameter settings

The following parameters will need to be included in the rotary axes option.

```
spindle azimuth rotation  = true   ## Default false for rotary table
```

This defines that the spindle is a rotary axis head.

```
azimuth axis parameters     = ( 0.  0.  0.   1.  0.  0. )
elevation axis parameters  =  ( 0.  0.  0.   0.  0.  1. )
```

The ***first three figures*** in this case will be set at **0. 0. 0.** for both axis, there being no likelyhood of any offset in the **elevation**, and the **spindle azimuth** offset is accounted for by the ***azimuth centre*** parameter.

```
azimuth centre              = ( 0.  0.  180.5 )  ##  Offset of 180.5mm in Z
```

The ***second set of three figures*** give the axis of rotation about either X, Y, or Z axis.
(It is essential that both axes are not aligned to the same axis.   The **azimuth** is the **main** alignment)

Thus the example shown represents a machine where the **azimuth** rotation is set to be about the **X** axis (vector **1.** 0. 0.), and the **elevation** rotation is about the **Z** axis. (vector 0. 0. **1.**)

(NOTE 1 :- These figures must be written with points. (**0.** not **0**)

(NOTE 2 :- Some machine controls may have a form of code that will allow the tool to be run as though the ***centre of rotation*** were the centre of the tool tip.   In this case set the ***azimuth centre offset*** to **zero**.

Additional parameters

azimuth axis units   = degrees   ## Usual units (***Default could be none***)
azimuth axis direction   = positive   ## Normal rotational direction required. (***Default could be none***)

elevation axis units   = degrees   ## Usual units (***Default could be none***)
elevation axis direction   = positive   ## Normal rotational direction required. (***Default could be none***)

pcs origin   = (0   0   0   0   0   0) ## (***Default***, used only in some 5 axes horizontal spindle alignments)

## Azimuth Min   Max.   Elev'n Min   Max.   Tol.   Moves
rotary axis limits = (-99999 99999   -99999   99999   0.1   1)   ## ***Default*** values, virtually unlimited rotation

(Example Settings) :-

rotary axis limits = (**-190.0**   **10.0**   **-360.0**   **360.0**   0.1   4)   ## **Spindle Tilt Azimuth** limited swing values, and **Elevation Rotary Table** limited to one revolution either way)

Tolerance value (Tol.) is the tolerance to which the angular displacement is maintained by Ductpost in the number of Moves set.   Keep tolerance and moves to the ***highest*** and ***lowest*** values respectively, conducive to good machine performance.
(FROM DP1331 the move fuction (last digit) has been superceeded by " ***linearise multiaxis moves = true*** " which automatically adjusts the number of moves needed to achive the tolerance setting)

(NOTE 3 :- It is assumed that the Tool points vertically down at **A 0** for these settings.
If A 0 is either horizontal, or vertical, then a special case applies and should be refered to Delcam Support.

See also 5-axes Table examples

Back to :- Top : Parameters

Setting up a 5 axes Spindle Head option file

(Up dated 31/01/2001)

An example to illustrate a type of 5 axes Spindle Head is given here :- (Rotary Axes Head)

Requirements for 4th / 5th Rotary axes

The following inclusions are probably required to be defined in an option file for Rotary Multi-axes working.
(The three Main rotary axis definitions are being defined in this example, **A , B, C,** but **only two** will actually be used for any multi-rotary axes system.)

```
        define format ( A B C )    ## May well be defined already in the inbuilt source
            metric formats
            leading zeros      = false
            trailing zeros      = true
            decimal point      = true
            decimal places    =   3
            imperial formats
            leading zeros      = false
            trailing zeros      = true
            decimal point      = true
            decimal places    =   4
        end define

            word order  =  ( + A B C )   ## Only needed if not in inbuilt word order list
            block order = true             ## Override for word order listing, uses "define block xxx " order

        define keys
            azimuth axis      =  C           ##  4th rotary axis normally an azimuth  ( Column Rotary )
            elevation axis    =  B           ## 5th rotary axis normally an elevation ( Spindle Rotary  )
        end define
```

## " **A** , **and/or B**, **and/or C** " requires to be inserted in both **Rapid**, and **Linear**, blocks depending upon axis alignments. (Using the pictorial example, **B** rotates about **Y**, **C** rotates about **Z**)

```
        define block move rapid
            N ; G1 ; G2 ; G3 ; G6 ; X ; Y ; Z ; B ; C ; S ; H ; M1 ; M2
        end define

        define block move linear
            N ; G1 ; G2 ; X ; Y ; Z ; B ; C ; F ; M1 ; M2
        end define
```

Rotary axes parameter settings

The following parameters will need to be included in the rotary axes option.

```
    spindle azimuth rotation     = true   ## Default false for rotary table
    spindle elevation rotation  = true   ## Default false for rotary table
```

These define that the column / spindle is a rotary axis.

```
    azimuth axis parameters    = ( 0   0   0    0   0   1  )
    elevation axis parameters =  ( 0   0   0    0   1   0  )
```

The first three figures in this case will be set at **0 0 0** for ***both*** axis, and are normally reserved for table units.
Whereas ***spindle*** *azimuth* and *elevation* **offsets** are accounted for by the *azimuth and elevation centre* parameters.

azimuth centre = (0. 0. 0.) *## Unusual to have an Offset in the azimuth)*
elevation centre = (0. 0. **180.5**) *## Offset of 180.5mm in Z*

The second set of three figures give the ***axis of rotation*** about either X, Y, or Z axis.
(It is essential that both axes are not aligned to the same axis. The **azimuth** is the ***main*** alignment)

Thus the example shown represents a machine where the ***azimuth*** rotation is set to be round the **Z** axis (vector 0. 0. 1.), and the ***elevation*** rotation is round the **Y** axis. (vector 0. **1.** 0.)

(NOTE 1 :- These figures can be written with, or without, points. (**0.**) or (**0**)

(NOTE 2 :- Some machine controls may have a form of code that will allow the tool to be run as though the ***centre of rotation*** were the centre of the tool tip. In this case set the ***elevation centre*** offset to **zero**.

Additional parameters

azimuth axis units = degrees ## Usual units (***Default could be none***)
azimuth axis direction = positive ## Normal rotational direction required. (***Default could be none***)

elevation axis units = degrees ## Usual units (***Default could be none***)
elevation axis direction = positive ## Normal rotational direction required. (***Default could be none***)

pcs origin = (0 0 0 0 0 0) ## (***Default***, used only in some 5 axes horizontal spindle alignments)

	## Azimuth Min	Max.	Elev'n Min	Max.	Tol.	Moves	
rotary axis limits = (	-99999	99999	-99999	99999	0.1	1)	## ***Default*** values, virtually unlimited rotation

(EXAMPLE SETTINGS)

rotary axis limits = (**-360.0** **360.0** **-110.0** **110.0** 0.1 4) ## Spindle Azimuth Column rotation limited to one revolution either way, and Spindle Tilt Elevation rotation limited to **+/- 110** degs. either side of **zero** [***Vertical***])

Tolerance value (Tol.) is the tolerance to which the angular displacement is maintained by Ductpost within the number of Moves set. Keep tolerance and moves to the ***highest*** and ***lowest*** values respectively, conducive to good machine performance.
(FROM DP1331 the move fuction (last digit) has been superceeded by " ***linearise multiaxis moves = true*** " which automatically adjusts the number of moves needed to achive the tolerance setting)

(NOTE 3 :- It is assumed that the Column is vertical and the Tool points ***vertically*** down at **C 0** , **B 0** ,
or the Column is horizontal and the Tool points ***horizontally*** at **C 0**, **B 0** for these settings.
If the Column is horizontal and B 0 is vertically UP or DOWN, then a special case applies and should be refered to Delcam Support.)

See also 5-axes Spindle examples

Back to :- Top : Parameters

Spline Machining

(Up-dated 19/10/2000)

Many of the latest machine tool control units have the capability of spline interpolation. This means that rather than taking as input a series of small straight line moves, the control unit will accept a mathematical definition for a curve to control the movement of the machine tool.

However, most 3D CAM systems are only capable of producing straight line vectors, even when the shapes they are machining are designed with a type of spline curve.

What is spline machining?

Spline machining is using spline curves to define the path a machine tool should take.

This has two major advantages.

1/ The amount of data that is needed to be passed to the controller can be dramatically reduced.

2/ On some controllers with higher feed rates being used, the machine can traverse the spline tool path faster than a conventional one.

When people talk about spline tool paths, there are three main forms of them.

NURBS tool paths, (**N**on **U**niform **R**ational **BS**plines)
POLYNOMIAL tool paths, (Two types :- **Polynomial** parametric curves
and :- **Hermite** polynomials.)

All three curve types have their differing advantages, and disadvantages, so there is no one standard.
All can be made to work in similar ways.

With the splining method developed by Delcam, we offer tangent continuity between the curve spans when it is available.

What you will need :-

SplineMILL:- This is the program that takes the normal CLdata cut file, and splines the result.
These splines are then Post Processed using a special (or modified) Option file.

DuctPost ****:- This is a Post Processor that can interpret the Splined 5000 cut file record and change it into the format to produce the tape file that is needed by the controller.
The ******** is the Post Processor version that you will need and will change with time but should be **1203, or later.**

DUCTPost and SplineMILL will accept cut files produced by both DUCT, and PowerMILL.

However, cut files produced on a PC in native NT format, are not portable to a UNIX system and will therefore have to be processed on an NT system.

Which controllers are supported:-

Siemens

Controls	**Sinumerik 810D and 840D**
Output type	**Parametric curves (Version 1)** **Nurbs (Version 2)**
Tested	**Bridgeport -** **with Siemens approval**
Basic Option File	**spl_s840.opt**
Shipped	**PowerMILL 2475**

Post Processor	DUCTPost 1203 (or later)

Fanuc

Controls	15m and 16m controllers. Needs the 64 bit Risc option
Output type	Nurbs curves
Tested	Fanuc 16m Sidel of France.
Option File	spl_f16m.opt
Shipped	PowerMILL 2475
Post Processor	DUCTPost 1205 (or later)

Heidenhain can be supported but we have not found any material advantages over the convential Heidenhain output.

Okuma spline can be provided but has not had extensive testing and therefore cannot be guarenteed to be effective.

There are no plans to support any other controls at the moment.

For Licence and operational details refer to :- **Spline Licence ? Post Processor details**

(Back to :- TOP)

Splining Tolerances

As more customers become interested in what splining has to offer, it is becoming apparent that the use of tolerances needs explaining.

When Delcam did its original development work it was realised that the tolerances set during tool path generation could have a drastic effect on surface finish.

With the release of **SplineMill 1101** it was recognized that this could cause problems and the following was added to the documentation :-

A note about tolerances

In order for the splining process to be effective, it must usually make some approximations to the input data. The splining process is controlled by a tolerance value which is input to the program, and the fitted spline is guaranteed to be accurate to within the given tolerance at the co-ordinate positions given in the input file.

A second tolerance is used internally within SplineMILL to control the deviation of the fitted spline from the straight lines which join together the co-ordinate positions in the input file. This tolerance is initially set to the same tolerance which is used for the positions themselves, but may be increased automatically if it cannot be achieved. A warning will be printed if this is done, but this is only likely to occur where the tolerance used to generate the CLdata in the CAM system, is larger than the fitting tolerance input to SplineMill.

Since the generation of the original tool path is itself an approximation to the underlying geometry of the surface being produced, great care should be taken to avoid the build up of error which could affect the accuracy of the finished component. In order to achieve a given tolerance it is recommended that half the required value is used in the CLdata generation process within the CAM system, and the same value is also used in the splining process, in this way the overall tolerance should be as required.

Should the program warn that it can not maintain the surface tolerance and automatically increase the value, the new value should be taken into consideration when accessing the overall accuracy of the Splined tool path. If the result is unacceptable the input file should be recalculated using a finer tolerance value and the splining process repeated.

Whenever this program is used it is advisable to check the resulting CLdata file to make sure it is acceptable and has not produced splines which deviate significantly from the original tool path.

A Rule of Thumb for Tolerances

Delcam development have made various tests, and have came up with a rule of thumb for the relationship between the various tolerances. This is simply an empirical rule, but has been found to work well in a wide variety of circumstances.

TOL = SPL	SPL = Splining tolerance
M/C = 1/2 TOL	M/C = Machining tol. in PowerMill
TRI = 1/4 TOL	TRI = Triangulation tolerance

Why are these recommended?

These guidelines were developed during the machining trials of **SplineMill 3001**, and the reason for the factor being set to **2** is as follows.

In PowerMill you are generating a tool path over an approximation of the data, (i.e. A triangle set). If the value of the machining tol. is close to the triangular tolerance, a faceted surface will result. (This will show up as surface effects of the triangle boundaries)

If the machining tolerance is too high, (i.e. close to the splining tolerance) the splines have no room to move, and the amount of data reduction is restricted . This gives a longer tape and lessens the benefits that splining can provide.

The factor of **2** gave the best results and was the simplest description method.
This does not mean that the values can not be altered, but you must be aware of the adverse effects that can result.

==

For example, if you wanted to achieve a tolerance of 0.01

TOL = 0.01

SPL = 0.01

M/C = 0.005

TRI = 0.0025

This allows the spline maximum movement and to remain inside the tolerance. As a result the data is reduced.

The reasoning behind the last statement should probably be explained further ... This assumes that the linear tool path will under cut a convex shape, and leave additional material on a concave shape. In both cases the effect of splining will be to reverse the linearisation effect.

In many examples we have reduced the data to a sixth, or better, than its original linear ISO tape size.

(Back to :- Spline Tolerances ; Controls ; Top)

Written 11/11/1998 By MR

SplineMill Licence and Useage Details

(Up-dated 20/10/2000)

Splinepost has now been integrated into Ductpost from version Ductpost1203.

Ductpost1203 was supplied with PowerMill2475, and has recently been superceeded by **Ductpost1205** which is now the recommended version (and any later versions) to use with **SplineMill1101**.
As development proceeds further, up-dates will be inevitable and will usually be incorporated with the latest release of PowerMILL CDs.

SplineMill is a separate utility that is provided on the PowerMill CD.

To run spline machining, you need the following options in your license file:

To run **SplineMill** : SPLINEMILL--SPLINEMILL

To run **Ductpost** with spline machining, you need:

DUCTPOST--SPLINE DUCTPOST--DUCTPOST

You will also need a machine licence, and the only **two proven** available option files for spline machining at the present moment are :-

spl_f16m for which you need the machine licence : DUCTPOST--FANUC11M

spl_s840 for which you need the machine licence : DUCTPOST--SIEM850

These are the base configuration files and will run as stand alone, but normally will be incorporated within a Customer's Option file.

An **Okuma** Spline option has been developed but at the present moment we do not have sound machine testing to be assured that it is viable.

(Back to Top)

Useage

To generate a spline machining tape file, it is first necessary to output a cutfile as normal from PowerMILL.

e.g. :- **normal.cut**

Then use the product " **splinemill** ".

You will be prompted for the input cut file name :- **normal.cut**

Then for an output cut file name e.g. :- **normal-spl.cut**

Followed by a request for a tolerance for splining :- e.g **0.01 (say)** **{ This will not be asked for in later versions }**

The output cutfile will be the splined version of the original cutfile.

A sensible approach to naming cut / tape files that have been Spline processed is to use a naming convention similar to the above **[filename-spl]** (as used by Delcam) to signify files which have been splined.

The next step is to use the appropriate spline option with ductpost to process the splined cutfile,

e.g. :- ductpost spl_f16m.opt * normal-spl.cut

which will produce a spline tape file :- normal-spl.tap

The tape file produced will after the first few lines start to spline. The normal " **X,Y,Z** " output will not be evident.

*** Note** :*This is the stand alone option, if it has been incorporated in a customised option then use that instead.*

For further information on **Splinemilling** refer to :- Spline Machining

(Back to Useage ; Top)

Spline option files

(Up dated 29/09/2000)

This document is an overview of the necessary steps to writing an option file that allows spline machining. For a more detailed explanation of spline machining itself see the introduction to Spline Machining, and also Using Spline Machining guide.

Spline types

DUCTpost supports two types of spline representations : **polynomial** representation, and **B-spline** representation. The choice is determined by what is supported for a given machine tool, and the representation effects the way the option file is written, since each makes use of different data provided by DUCTpost.

To choose **polynomial** representation for the spline data, use the flag:

spline type = polynomial

To use **B-spline** representation for the spline data, use the flag:

spline type = bspline

Code "spline"

The code " ***spline*** " is used in the same ways as other codes, (for example the code " ***linear***"). It is usually used at the start of a spline move block, to signify that spline mode has been entered.

```
define codes
    spline   =  G1  "6.2"    ( For B-spline )
end define
```

Polynomial representation

This type of representation presents the spline data as polynomials in each of X, Y and Z co-ordinates in terms of a parameter form. The machine tool then requires the additional coefficients of the polynomials and these are accessed via certain numbered keys :-

```
 define keys
    key 58      = K0X
    key 59      = K1X
    key 60      = K2X
    key 61      = K3X
#
    key 62      = K0Y
    key 63      = K1Y
    key 64      = K2Y
    key 65      = K3Y
#
    key 66      = K0Z
    key 67      = K1Z
    key 68      = K2Z
    key 69      = K3Z
  end define
```

K0X will refer to the **constant** term in the **X polynomial**
K1X will refer to the **linear** term in the **X polynomial**
K2X will refer to the **quadratic** term in the **X polynomial**
K3X will refer to the **cubic** term in the **X polynomial**
Similarly for the **Y and Z polynomials**.

It is necessary to refer to the machine tool documentation to see precisely how these coefficients are to be presented to the machine. (For example, some machine tools may not require the **linear** coefficient, since this can be determined by other means).

Sample " ***define block move spline*** " for " ***spline type = polynomial*** "

```
 define block move spline
    N  ; G5 ; G7 ; G1 "POLY " ;
    if ( swx )
       K0X xcoord ;  K2X ; K3X ; ") " ;
   end if
   if ( swy )
       K0Y ycoord ;  K2Y ; K3Y ; ") " ;
   end if
   if ( swz )
      K0Z zcoord ;  K2Z ; K3Z ; ") " ;
   end if
   F
end define
```

The movement switches, " **swx**" etc., are used to ensure that the **polynomial** for a particular coordinate is only output if necessary.

B-spline representation

A full discussion of the mathematics of the **B-spline** representation is beyond the scope of this document. For our purposes it suffices to say that a **B-spline** is defined by a sequence of control points, and also by a **knot vector**. The **knot vector** is accessed through a particular numbered key :-

```
define keys
  key 70 = KNOT
end define
```

The control points are simply considered as any other points, and are accessed through X, Y and Z.

Sample " ***define block move spline***" for " ***spline type = bspline***"

```
define block move spline
     N ; G1 ; KNOT ; X ; Y ; Z
 end define
```

The above example will output each control point of the **B-spline** on a separate line, together with an associated **KNOT** value.

(Return to TOP)

List of Controls

(Revised 31/01/2001)

Ductpost Ref.	**Manufacturer** (Base control)
ab84	Allen Bradley
acra8	Acramatic 850 (Cincinnati Milacron)
agie	Agie wire spark
anilam	Anilam Crusader
boss	Bridgeport (Series 1, Series 2)
bosch	Bosch 200
bostom	Bostomatic
charm	Charmille Robofil wire spark
deckel3	Deckel Dialogue 3
deckel4	Deckel Dialogue 4
deckel11	Deckel Dialogue 11
dyna	Dynapath
dm	Dynamite
eberle	Eberle
elexa	Elexa
fadal	Fadal
fanuc	Fanuc (General)
fanucom	Fanuc OM
fanuc6m	Fanuc 6M
fanuc10m	Fanuc 10M **(No longer supported)**
fanuc11m	Fanuc 11M
fanuc15m	Fanuc 15M
fanucw	Fanuc wire spark
fanuc6t	Fanuc6t lathe **(No longer supported)**
fanuc10t	Fanuc10t lathe **(No longer supported)**
fagor	Fagor
fidia	Fidia
ge2000m	GE2000 General Electrics
hauser	Hauser 311,314 **(No longer supported)**
heid	Heidenhain 150,355
h33	Heidenhain H33 **(Russia only)**
h155	Heidenhain 155
heid400	Heidenhain 400 series
heidiso	Heidenhain Iso
hurco	Hurco
incon	Incon **(No longer supported)**
japtw	Japt wire spark **(No longer supported)**
kryle	Kryle

maho	Maho(Philips 432, 532, and DMG MillPlus controls)
mazak	Yamazaki Machine Tools (Mazak)
matsura	Matsura
mitsu	Mitsubishi
mitsuw	Mitsubishi wire spark
num	Num760F
okuma	Okuma
p28m	Posidata 2800 **(No longer supported)**
ph432	Philips 432
r2e3	Bridgeport R2E3
roland	Roland
siem850	Siemens 800,810,850
selca	Selca
servo	Servo **(No longer supported)**
sodick	Sodick wire spark
standard	***Base Post Processor***
tcomp	Tcomp **(No longer supported)**
tiger	Tiger

The above list of controls is currently included in Ductpost, with the supported controls in **bold** type.
Post Processors for later versions, or other types of control, will be based on an appropriate base control from the above list by means of an ***option*** file.

Block definitions

(Revised 10/06/2005)

A block definition consists of one or more lines which can appear in the NC program output in response to a particular part of the CLDATA input.

These take the format :-

define block …………….

end define

(e.g. **move linear**, or **tool change**).

The blocks which may be defined are as follows, and are listed in their logical order of use :-

define block ………	**Function**
tape start	Output standard data at the **beginning** of each nc program.
tool change first	If defined this block is output on the **first** tool change.
tool change	This block is output on the **second** and **subsequent** tool changes if the ***tool change first*** block is defined. It is output for **all** tool changes otherwise.
tool change clear	If this block is defined the stored values for **all modal** output words are cleared, and then the block is output with the current values.
move from	Output for each of the XYZ co-ordinate triplets in a record corresponding to the **from (datum)** point.
move rapid	Output for each of the XYZ co-ordinate triplets in a record corresponding to a **rapid** move.
datum shift	This handles any 3+2 **datumshift/workplane** changes and data related to them.
move linear	Output for each of the XYZ co-ordinate triplets in a record corresponding to a **linear** move.
multiaxis transition	This will handle the required ouput data for **Intermediate Toolpath moves** as specified in the block.
move spline	Output for a **spline** move. The output depends on the " *spline type* " and is either a sequence of piecewise-continuous polynomials, given in coefficient form, or a B-spline representation. The " **move spline** " block is used to output either each individual polynomial (in terms of its coeffiecients), or alternatively, one line of the standard B-spline representation.
move circle	Output for each of the XYZ co-ordinate triplets , and either I J K or R radius co-ordinates in a record corresponding to a **circle** output.
cycle start	Output at the **beginning** of each ***canned*** cycle.
move cycle	Output for each of the XYZ co-ordinate triplets in a record during a **canned drill / tapping** or **boring** cycle.
move tap	Output for each of the XYZ co-ordinate triplets in a record during a **canned tapping** cycle.
cycle end	Output at the **end** of each ***canned*** cycle.
tape split start	Output at the **beginning** of each file when the nc program is ***split*** into several files.
tape split move	Define **First** moves on consecutive ***split*** tape(s). (e.g. Old coordinates followed by Z move, then XY move)
tape split end	Output at the **end** of each ***split*** program file.
segment	Insert into tape file included data when prescribed limit of distance traveled by tool, or number of line blocks reached. (set by " segment type "

go home preamble	Output at the **start** of the go home sequence. This block is ***always*** output, even if the go home output flag is ***false***.
go home z move	Output during a go home move if the Z coordinate value has **changed**.
go home xy move	Output during a go home move if the X and Y coordinate values have **changed**.
tape end	Output standard data at the **end** of each nc program.

Other blocks that can be defined for specific requirements :-

define block ………	**Function**
cldat (xxxx)	Output included data, or set action, when **cldata** record called by CLDATA. (xxxx is the appropriate **cldat** number – brackets not required)
ppfun (abcd)	Output included data, or set action, when **ppfun** record called by CLDATA. (abcd is the appropriate **ppfun** name relating to the one in the PM Command Setting – brackets not required)
user (efgh)	A **user** defined block can be set up to encompass standard data that is used several times in the option file and can prevent mistakes in missed edits. “ define block user efgh ” Note:- “ efgh ” is case sensitive “ eFgH ” is not the same To call the block use “ call block efgh ” in a defined block where “ efgh ” or “ eFgH ” is the defined name of the user block.

If the output of a default defined block is not required, then in the option re-define as follows :-

```
          define block ..............
             N ;
          end define
or as
          define block ............
          end define
```

Variables used in block definitions

(Up dated 20/01/2006)

An NC tape block/line definition consists of one, or more, lines of data that appear in response to a particular part of a Postprocessed Cut file, or NC program. (e.g. a linear move or a tool change, etc., such as " N0 G71 ").
The words, and values, which appear in these block/lines can be defined in the option file by utelising defined Words and associated Variables within the various block forms :-

```
define block tape start
    N BlockNumber ; G4 TapeUnits
    ……………….
end define
```

The ***variables*** that may be used in block definitions with Words are given below.

(e.g. **T NextTool** gives the Word **T** the value of the **next load tool number**. They may be written in upper or lower case, or **Preferably As Indicated**)

Variable Name	Use and Function
Angle	The defined output **angle**
Argument	The **argument** from an PPFUN record in PowerMILL **NCPrograms/Preferences/Commands** Dialog box Use :- FEEDRATE **3000** define block **ppfun feedrate** N ; "FN 0 Q3= " ; TN Prat (150) N ; "FN 0 Q4= " ; TN Frat (1200) N ; "FN 0 Q5= " ; TN **Argument** (3000) ……………… end define N20 FN 0 Q3=150 N25 FN 0 Q4=1200 N30 FN 0 Q5=**3000** (The ***argument*** can be used in other blocks with care but only **ONE argument** is valid in the ppfun block. To use elsewhere in the option use ***Word{TN}***, relating to **TN** in the above example can be utelised)
Azmove	The **difference** between the current and previous **azimuth** angle.
Azimuth	The **Azimuth** angle
Azpos	The **current value** of the **azimuth** angle.
BlockMaxX **BlockMaxY** **BlockMaxZ**	The **Maximum** values of X, Y and Z of the **Block Size**, usually used in comments at tape start.
BlockMinX **BlockMinY** **BlockMinZ**	The **Minimum** values of X, Y and Z of the **Block Size**, usually used in comments at tape start.
BlockNumber	The **actual** block number at that point of output

Variable List Continued (2)

Variable Name	Use and Function
Character(n)	Depending on the **value**, outputs an ASCII character. (e.g. **character 8** = "**;**" will place this **string** at the end of every line)
CheckCounter	This will return the **current value** of the counter without incrementing its value.
ClearPlane	Drilling/Tapping **clear plane** above the **work surface** (Absolute position)
ClearPlaneInc	Drilling/Tapping **clear plane** above the **work surface** (Incremental position)
CompensationToolLength	The Cutter Compensation **Length** as defined in the PowerMill NC Preferences Toolpath dialog box
CoolantCode	**M code value** for coolant setting (e.g. **8** if coolant on)
ContactNormalX **ContactNormalY** **ContactNormalZ**	The current tool path **Contact Normal Vector**
Counter	This will be **incremented** each time it is used. The ***start value*** and ***increment*** can be set, using **counter start** and **counter increment**
Cycfed	Canned cycle Tapping **feed rate**
Cycle	Number of **current** drilling/tapping cycles (e.g. drill cycle = 1)
CycleCodeNumber	Provides access to **Cycle Function Codes**
CycleDwellTime	The **current** cycle dwell time
CycleSafeZ	Returns the previous behavioral **SafeZ** in drill cycles.
Day	**Day** of the week (numerical e.g. **01** for first day of the Month)
Distazi	The **Azimuth** angle feedrate code for **MillPlus** multiaxis controls
Distele	The **Elevation** angle feed rate code for **MillPlus** multiaxis controls
DPversion	Provides the **version** of **Ductpost** used to produce the NC tape output. (Use in conjunction with OptionFileName)
Elmove	The **difference** between the **current** and **previous** elevation angle.
Elpos	The **current** elevation angle position
FeedRate	The **current** feed rate
Frat	The **current cutting** feed rate
FromX ; FromY ; FromZ	The X , Y , Z co-ordinates of the **Datum** (FROM) point
GotContactNormals	The variable to ***check*** if the tool path has **Contact Normal Vectors enabled**
HoleDepth	Drilling/Tapping **hole depth** (Absolute bottom position from clear plane)
HoleDepthIncremental	Drilling/Tapping **hole depth** (Incremental distance from clear plane to bottom of hole)
HoleDiameter	This provides the actual **Hole Diameter** - not the Drill diameter
HoleTop	Provides the absolute position of the **top of the hole.**
Hours	**Time of Day** **Hour**
inta, intb, intc -> intf	Integers which can be **set** with an **integer value** (e.g. edit inta 25)
JobName	Cut or NC Tape file **name**

Variable List Continued (3)

Variable Name	Use and Function

LastTool	Last Load Tool **number** (Last tool in the series to be used)
LoopCounter	Returns the **number of passes** in the " **repeat** " loop function
MaxX ; MaxY ; MaxZ	The overall Maximum limit values of **travel** for X, Y and Z (usually used in comments at tape start, or for **DUCT** block size)
MinX : MinY : MinZ	The overall Minimum limit values of **travel** for X, Y and Z (usually used in comments at tape start, or for **DUCT** block size)
Minutes	**Time of Day Minutes**
MinorWord	**Values** from CLdata associated with a particular function (Rarely used now)
ModelX : ModelY : ModelZ	Give access to the coordinates in the model system in all circumstances. These were introduced for a specific operational debug and as such will not normally be of general use
Month	The **current month** (numerical e.g. **04** for April)
MoveComponentType	*This is still under development*
MoveMinorType	*This is still under development*
MoveType	This checks the **type** of tool path movement (MoveType = **11** connection moves ; MoveType = **10** 5axis moves) For full list see What's New
NextAngle	The **next** angle defined
NextTool	The tool number of the **next load tool** to be used (Used for preselection)
NextToolPath	The **next tool path** to the current one.
NumberPecks	Number of **pecks** in a drilling cycle
NumberToolPaths	Returns the **number of toolpaths** in the CLdata file (used in " **repeat** " function)
NumberTools	Returns the **number of tools** in the CLdata file (used in " **repeat** " function)
OldX ; OldY ; OldZ ; OldAz ; OldEl	The **last positional** axis move
OptionFileName	This should be used in conjunction with DPversion as a verification that the correct Option has been used to produce the NC output.
OutputWorkPlaneName	The PowerMill NC program **Output WorkPlane**
PartID	A **part identification** string. This may be obtained from the **DUCT PARTNO** cldata record, **PowerMILL Part Name**, the configuration file, or **prompted** for at run time (*Except prompts are ignored if posted via PowerMILL, or will cause PowerMill to hang*)
PartID[5]	This is similar to above, but has the added flexibility to select the starting point in the **Part Identification** string via the **[value]**. In the example the **5th character** in the **Part Name** will be the start point, and the number of characters output will be limited by the field width of the carrier word. (***ID PartID[5]***)
PeckDepth	Drilling cycle **peck depth** (Incremental)
Pitch	Returns the **Pitch** setting in PowerMill Tap Cycles
Plunging	Check for **Plunge** move.

Variable List Continued (4)

Variable Name	**Use and Function**
PMCodeBase	Provides the first **5 digits** of the PowerMill Codebase number

PMCodeBaseRevison	Provides the **2 digit** PowerMill Codebase Revision number
Prat	The **current plunge** feed rate
PrevTool	The tool number of the **previous tool** used
PrevToolPath	The **previous tool path** to the current one.
ProgID	A program **ID number**. (This defaults to " **1** ", or can be pre-set in the option by " **program id start** ", or prompted for at run time. (*Except prompts are ignored if posted via PowerMILL-see PartID*)
Programmer	The **current** owner of the programme
Reconfiguring	Checks to see if a **rotary axis reconfiguration** is happening.
RepeatedLoadTL	Check to verify if **same tool** is being used in current tool path as previous tool path.
RFrat	The **maximum** rapid feed rate setting (**rapid feedrate = 30000.0**)
rna, rnb -> rnf	Defined **Real** Numbers (**edit rna = 1.234**)
SafeZ (SafZ)	A **safe Z positional hieght** above the job
Seconds	**Time of Day Seconds**
SpecialRecordCode	Returns the **Integer** Code used to identify the particular special record in question. (see DP1460 **A new “ *define block special record* ” has been added**
SpecialRecordCode[n]	Returns the “ **Real** ” value defined in the particular special record.
SpecialRecordString[n]	Returns the “ **String** ” value defined in the particular special record.
SpindleCode	The **current M code** for spindle state (e.g. **3** if spindle ON)
SpindleSpeed	The **current spindle speed**
StartZ (StaZ)	The **Z position** at which **Plunge feed rate** starts
Srat	The **rapid skim feed rate** as defined in PowerMill Feed Rate Form "**Rapid**"
swa, swb , swc -> swl	**Switches** which may be set and tested for in logic " **if (statements)** " *(swx, swy, swz are used solely for Spline applications)*
TapeCoords	The **function code** for the current co-ordinate system (usually **90** for **absolute** or **91** for **incremental**)
TapeUnits	The **function code** for the current tape units (**21 or 71** for **metric** or **20 or 70** for **imperial,** except for ***Heidenhain*** unless defined)
Thickness	The **thickness** of material left on a defined cutter path
TipCoordinates	The Tool Path output coordinates as set for Tool **Tip** (***true***) or Tool **Center** (***false***)
TipRadius[ToolNum]	The **Tip radius** of **current** tool
Tolerance	The **current** tool path tolerance setting
ToolCompensation	The **current compensation** G code (e.g. 41, 42 or 40)
ToolCount	The current tool **sequence number** as aposed to given tool number
ToolFeed	The **current feed rate** for tool
ToolLength[ToolNum]	The **current** created tool length

Variable List Continued (5)

Variable Name	Use and Function
ToolLenOffset	The **Tool length offset** (Usually tool number)
ToolName	The **Tool Name** from PowerMill 2.5 Tool Form or **Tool Identifier** from PM3.0 (Usually a reference Number, or Name)

ToolNumber	The **current** tool number
ToolNumber[3]	The **Tool number** of the **3rd** in the list of tools programmed
ToolPath3axis	Checks for **3 axes type of tool path**
ToolPath3Plus2	Checks for **3+2 type of tool path**
ToolPath5axis	Checks for **5 axes type of tool path**
ToolPathFromTool	Maps **toolnumbers** to a **toolpath number** (used in " **repeat** " function)
ToolPathLength	The **length of tool path cutting metal** in **Output Units .**
ToolpathMaxX , Y , Z	The current tool path **Maximum travel** distance.
ToolpathMinX , Y , Z	The current tool path **Minimum travel** distance
ToolPathName	The **current tool path name.**
ToolPathTime	The **time of tool path cutting metal** in **Seconds**
ToolpathWorkplaneName	The **WorkPlane Name** used to generate the tool path
ToolPathWorkplaneA , B , C	Povide access to the **Euler** angles for a **Toolpath Workplane**
ToolPathWorkplaneX , Y , Z	Povide access to the **Shift** position for a **Toolpath Workplane**
ToolRadius[ToolNum]	The **current** tool radius
ToolRadOff	The **Tool radius offset** (Usually G43 coupled with H tool number)
ToolSpeed[ToolNum]	The **current spindle speed** for tool
ToolType	The defined PowerMill **Tool definition (e.g. EndMill)**
ToolVectorX **ToolVectorY** **ToolVectorZ**	The current linearised **Tool Vector**
UsingHoleTop	Check to see if **Hole Top coordinates** is defined for drilling cycles.
Word[n] **Word{ N }**	The **"value"** of the **[n]** word in the list of defined words. (e.g.**Word[2]** is the current value of the **N** word) This is new to DP1361 and serves the same function as above but uses the **Word** itself. (NOTE: **curly** brackets are used)
WordString[n] **WordString{ TPN }**	This is new to DP1361 and is similar to the above but will return the **"STRING"** content of the word.
WorkPlaneA, B, C	The **current** 3+2 tool path **Work Plane Coordinates** (Used with setting " workplane angles = apparent ")
WorkPlaneX, Y, Z	The **current** 3+2 tool path Datum Shift **Work Plane Coordinates** (Used with setting " **workplane angles = apparent** ")
WorkPlaneAz **WorkPlaneEl**	The **current** 3+2 tool path **Work Plane Angles** (Used with setting " **workplane angles = machine tool** ")
WorldFromX, Y, Z	The **current** 3+2 tool path Work Plane **From World Coordinates**

Variable List Continued (6)

Variable Name	Use and Function
Xcentre ; Ycentre ; Zcentre	The **stored** X, Y, Z co-ordinates
Xcoord ; Ycoord ; Zcoord	The **current** X, Y, Z co-ordinates

Xmove ; Ymove ; Zmove	The **distance moved** between **current** and **last** position of X ,Y, Z
Year	The **current** year (numerical e.g. **01** for 2001)

(Return to TOP)

Tool change

(Up-dated 29/12/2000)
(Under construction)

The **tool change** procedures for different machines can vary enormously, and it would therefore not be appropriate to try and provide innumerable examples here. The tool change output format is normally handled by three block definitions which are as follows :-

1/ *define block tool change first*
2/ *define block tool change*
3/ *define block tool change clear*

1/ *define block tool change first*

This block will handle the output formatting requirements for the FIRST tool change and may, or may not, been defined in the source file.

2/ *define block tool change*

This block will handle the output formatting requirements for the SUBSEQUENT tool changes and may, or may not, been defined in the source file. It will also handle the **First** tool change if that is not defined.

3/ *define block tool change clear*

This block, if defined, will clear down any stored values in previous two tool change blocks and may, or may not, been defined in the source file.
If defined and the other two blocks are not defined then it will act as a tool change function.
(It is recommended that this block is not used as its function is not sound, and should be changed if set up in the source file to a ***define block tool change*** as indicated below)

Source File Example	**Recommended Change**
define block tool change first end define define block tool change end define **define block tool change clear** **N ; change tool ; T ToolNum** **absolute data ;** **end define**	define block tool change first end define **define block tool change** **N ; change tool ; T ToolNum** **absolute data ;** **end define** define block tool change clear end define

---

Variables

ToolNumber	Tool number from duct APT LOADTL
ToolLenOff	Length offset switch number
ToolRadOff	Radius offset switch number
NextTool	Number of next tool to be loaded (for use on machines with a tool preselect)
ToolLength	Actual tool length programmed
ToolRadius	Actual tool radius programmed

ToolFeed Feedrate associated with current tool

ToolSpeed Spindle speed associated with current tool

ToolName Name / Identifier of Tool from PowerMill

Words used are usually :-

T	(Toolnumber)
H	(Length Offset Switch)
D	(Radius Offset Switch)

There are special rules for the output of tool length offset and tool radius offset
The blocks for the ***first tool change*** may be set separately from the blocks for ***subsequent tool changes*** to allow extra coding applicable to the start.

```
define block tool change first
   N  ; T  ; M1 6
   G6 90 ;
end define

define block tool change
   N  ; H  0 ; M1 9
   N  ; T  ; M1 6
end define
```

This gives output like this

```
N120T1M6
N130G90G0X..Y..
N................
N...............
N200H0M9
N210T2M6
N220........
```

## First Move The first move after a tool change can be given in various different ways, this is done by the flag "***tool reset coordinate***".

tool reset coord = 1 (output current X, Y and Z after the tool change)
tool reset coord = 2 (output the current X and Y after the tool change)
tool reset coord = 3 (output XY on first line after tool change, then Z on next line)
tool reset coord = 4 (output Z on first line after tool change, then XY on next line)

(Return to Variables TOP)

Explanation of Array / Integer Switches

(Up-dated 10/10/2001)

The following is the basic structure of the arrays, and the function of the switches.

Circles

```
          integer      26  20  27  28
 Circle output  =    (  n    n    n    n  )
                        !    !    !    !
                        !    !    !  0  =  Absolute Centre position
                        !    !    !  1  =  Incremental Centre from start point
                        !    !    !
                        !    !  0  =  Single Quadrant
                        !    !  1  =  Full Circular Interpolation
                        !    !
                        !  0  =  X,Y,Z coordinates not required
                        !  1  =  X,Y,Z coordinates required
                        !
                      0  =  NO Circular Output
                      1  =  Circular Output
 -------------------------------------------
```

Spindle

```
          integer      10   11  12   13
 Spindle output  =    (  n    n    n    n  )
                         !    !    !    !
                         !    !    !  0  =  Don't link Coolant to spindle
                         !    !    !  1  =  Link Coolant to spindle ( i.e. M13 )
                         !    !    !
                         !    !  0  =  No M code output
                         !    !  1  =  M code output for spindle ( i.e. M03 )
                         !    !
                         !  0  =  No S output for spindle OFF
                         !  1  =  Output S0 for spindle OFF
                         !
                       0  =  Output spindle data when called by CLdata
                       1  =  Output spindle data in next block
                       2  =  Output spindle data with next move
```

Note :- It is possible that the " **define keys** ", **spindle = S**, **or spindle not used**,
may have to be changed to get a particular output required. **Experiment !**

```
 -----------------------------------------------
```

Coolant

```
          integer      6    7
 Coolant output  =    (  n    n  )
                         !    !
                         !  0  =  Output cool't OFF when called by CLdata
                         !  1  =  Output cool't OFF in next block
                         !  2  =  Output cool't OFF with next move
                         !
                       0  =  Output cool't ON when called by CLdata
                       1  =  Output cool't ON in next block
                       2  =  Output cool't ON with next move
 ----------------------------------------------
```

Radial Compensation

(Applicable for PowerMill 3.0 and Ductpost 1331 onwards see Radial Compensation)

```
                        integer       34  35  36  37
 Cutter compensation output  =    (  n    n    n    n  )
                                     !    !    !    !
                                     !    !    !  0  =  No cutter compensation output
```

```
        !    !    !  1  =  Output G41/G42 and G40 codes
        !    !    !
        !    !    0  =  Obsolete, no longer needed
        !    !
        !    0  =  No radius offset ( Usually D )
        !    1  =  Output radius offset ( D ToolNumber )
        !
        0  =  Obsolete, no longer needed
```

Note :- A minimum linear block must be defined as follows :-

```
    define block move linear
       N ; linear ; G2 ; tool radius ; x coord ; y coord ; z coord ; feedrate ; M1 ; M2
    end define
```

Movement following Tool Change

```
●              integer          16
Tool reset coordinates  =    (  n  )
                               !
                             0  =  No special output
                             1  =  Force out X, Y and Z after tool change
                             2  =  Force out X and Y after tool change
                             3  =  Force out X and Y, then Z after tool change
                             4  =  Force out Z then X and Y after tool change
```

Split Moves

```
●    integer       24
split move  =    (  n  )
                    !
                  0  =  N0 special output for X Y Z moves ( Best for 5 axes )
                  1  =  Output X Y move before Z for downward moves
                        Output Z move before X Y moves for upward moves
```

A note on splitting moves:- Be extremely careful of using split move for Multi-Axes options as it does not work in the same way as for Three Axes. It is a little too complex to explain in this brief note, so if you have problems refer back to Delcam Support for assistance.

Also ensure that a *linear block* has moves defined, some inbuilt controls haven't.

Canned Cycles (With the latest drilling in PowerMill ignore these, leave as the default except for integer 69)

```
●      integer       19
Cycle output  =    (  n  )
                      !
                    0  =  Output canned cycles ( G81 etc. )
                    1  =  NO Output of canned cycles
```

```
( Peck depth output )
●      integer   4  =    ( n )
                           !
                        0  =  Output as required in cycle
                        2  =  Output even if only ONE peck in all cycles
```

```
( Tapping Feedrate reduction )
●    integer   69  =    ( n )
                          !
                       0  =  Reduce normal feedrate by approx 8%
                       2  = Apply normal plunge feedrate  ( with PM3 and later Drilling always use this)
```

(**Clearance Plane**)

- **integer 54 = (n)**

  !

  0 = Absolute position value output

  1 = Incremental move value output

(**Drill Depth**)

- **integer 55 = (n)**

  !

  0 = Absolute value output

  1 = Incremental output

See also Ductpost Integer list

Spindle and Coolant

(Up-dated 29/12/2000)

Spindle and Coolant output is usually controlled by defining the appropate keys, codes and Array / Integer settings.

Example :- (Normal default settings in source files for Spindle and Coolant)

```
define keys
  aux function  = M1    ## defines M1 as key word for Auxiliary functions
  spindle       =  S    ## defines S as key word for Spindle output
end define

define codes            ## defines M1 as key word for Auxiliary Code functions
  spindle on cw          = M1  3
  spindle on cc          = M1  4
  spindle off            = M1  5
  spin coolant on cw     = M1  13
  spin coolant on ccw    = M1  14
  coolant on mist        = M1  7
  coolant on             = M1  8
  coolant on flood       = M1  8
  coolant off            = M1  9
end define

   spindle output   =  ( 2  1  1  0 )   ** From DP1205 replaced by Integer values
   coolant output  =  ( 2  1  )         ** From DP1205 replaced by Integer values
```

## defines how the Spindle and Coolant functions wil be handled by the post processor

Spindle output = Integer array, four elements

```
    integer 10 = 0  -- Output spindle data when called by CLdata
               = 1  -- Output spindle data with next block
               = 2  -- Output spindle data with next move
       integer 11 = 0 -- NO output for spindle OFF
                  = 1 -- Output S0 for spindle OFF
          integer 12 = 0 -- NO M codes for spindle control
                     = 1 -- Output spindle control codes ( M3, M4 etc )
             integer 12 = 0 -- Output spindle and coolant coded separately
                        = 1 -- Output spindle and coolant in one code ( M13 ).
```

Coolant output = Integer array. two elements

```
    integer 6 = 0 -- Output Coolant ON when encountered in CLdata
              = 1 -- Output Coolant ON in next block
              = 2 -- Output Coolant ON with next move
       integer 7 = 0 -- Output Coolant OFF when encountered in CLdata
                 = 1 -- Output Coolant OFF in next block
                 = 2 -- Output Coolant OFF with next move
```

The ***define block move rapid*** would probably be defined as follows :-

```
define block move rapid
  N ; G1 ; G2 ; G3 ; G5 ; G6 ; X ; Y ; Z ; T ; H ; S ; M1 ; M2
end define
```

A typical example of a source file settings is given above and the output is presented below :-

```
%
:0001
G91 G28 X0 Y0 Z0
```

```
G40 G17 G80 G49
G0 G90 Z10.
T10 M6
G54 G90 T12
X12.446 Y10.931 S1500 M3
G43 Z44.5 H1 M8
```

(Return to TOP)

DuctPost Integer List

(Up-dated 10/10/2001)

Always use the name format indicated in *blue* = n unless none available.

*Symbols :- **prefered use format**: (**absolute**) indicates usual default **:** (**false**) indicates usual alternative*
Those marked + see Array Data Also Integers, Reals / Characters

- **integer 1** not used

----

message output* = true or *false

- **integer 2** *= 0* **NO** messages output
  = 1 Messages output

- **integer 3** = 0 Multi-axes dissabled
  = 1 Multi-axes enabled

- **integer 4** = 0 Output peck depth as required
  = 2 Output peck depth even if only for one peck

- **integer 5** not used

----

coolant output = (n n)+

coolant ON

- **integer 6** = 0 Output when encountered in cldata
  = 1 Output in next block
  = 2 Output with next move

coolant OFF

- **integer 7** = 0 Output when encountered in cldata
  = 1 Output in next block
  = 2 Output with next move

- **integer 8** not used

----

block order* = false *or true (DP1203 onwards)

- **integer 9** = 0 Define block output format as **word order** sequence
  = 5 Define block output format as ***block order*** sequence

----

spindle output = (n n n n)+

- **integer 10** *= 0* Output when encountered in cldata
  = 1 Output in next block
  = 2 Output with next move

- **integer 11** *= 0* **NO** S output for spindle off
  = 1 Output S0 for spindle off

- **integer 12** *= 0* **NO** M code output
  = 1 M code output for spindle

- **integer 13** *= 0* **NO** link with spindle M code to coolant
  = 1 Output coolant with spindle M code

- **integer 14** not used

----

tlo output* = true or *false

- **integer 15** *= 0* **NO** special code output
  = 1 Output special code with offset

----

tool reset coordinate = (n)+

- **integer 16** *= 0* **NO** special output after tool change
  = 1 Force out X, Y and Z after tool change
  = 2 Force out X and Y after tool change (Z may be present)
  = 3 Force out X, Y then Z on next line after tool change
  = 4 Force out Z then X, Y on next line after tool change

- **integer 17** not used

- **integer 18** not used

----

cycle output = (n)

- **integer 19** *= 0* **NO** canned cycle output move codes
  = 1 Output canned cycles (G81 etc)

----

circle output = (n n n n)

- **integer 26** *= 0* **NO** circular output
  = 1 Circular output

- **integer 20** *= 0* **NO** circle X, Y coordinates output
  = 1 Output circle X, Y coordinates

full circle = **true or** ***false***

- **integer 27** *= 0* Single quadrant interpolation
  = 1 Full circular interpolation

incremental centre = **true or** ***false***

- **integer 28** = 0 Absolute position of Circle Centre
  = 1 *Incremental* position of Circle Centre from Start point

----

- **integer 21** not used

----

block start = (n)

- **integer 22** = n Block start value (i.e. 10)

----

block increment = (n)

- **integer 23** = n Block increment value (i.e. 2)

----

split move = (n) (Linear XYZ moves)

- **integer 24** = 0 **NO** special X, Y, Z splitting moves (Best for 5 axes)
  = 1 Output X, Y move before Z for downward moves and
  Output Z move before X, Y move for upward move (RAPID only)
  = 2 Output as above for Rapid and Linear moves

----

tape split on tool change = **false or** ***true*** **(** New for DP1310 **)**

- **integer 25** = 0 Tape split on tool change not enabled
  = 1 Tape split on tool change enabled (Useful for Sub-routine work)

● **integer 29** not used

----

go home output* = true or *false

● **integer 30** *= 0* Suppress go home move
= 1 Output go home move

----

use partid* = true or *false

● **integer 31** *= 0* Request PART NAME at run time (i.e. Prompt)
= 1 Use PARTNO from Duct, PART NAME from PowerMILL

----

use progid* = true or *false

● **integer 32** *= 0* Request PROGRAM NUMBER at run time
= 1 Use **default** program number (**1**) or as defined by integer 33

----

program id start* = *n (default program number = 1)

● **integer 33** = n Reset program number to "**n**" value

----

cutter compensation = (n n n n)+

● **integer 34** = 0 **NO LONGER REQUIRED**

● **integer 35** = 0 **NO** radius compensation offset (Usually D)
= 1 Output radius compensation offset (Usually D ToolNum)

● **integer 36** = 0 **NO LONGER REQUIRED**

● **integer 37** = 0 **NO** compensation codes output
= 1 Output compensation codes (G41 G42 & G40)

----

rapid feed code* = *n

● **integer 38** = 0 **NO** feedrate output on rapid moves
= 1 Output feedrate on rapid moves.
(Needs F defined in rapid block)

----

units* = *input* or metric or *imperial

● **integer 39** *= 30* Imperial output
= 40 Metric output
= 50 Input = Output

----

coordinates* = absolute or *incremental

● **integer 40** = 10 All output will be in absolute values (default)
= 20 All output will be in incremental values
(**Note :-** These can be redefined in the option move blocks)

----

rotary axis parameters

----

azimuth axis units = none or *degrees*

● **integer 41** = 0 Azimuth axis units linear
= 1 Azimuth axis units degrees

----

azimuth axis direction* = positive or *negative

● **integer 42** *= 0* Azimuth axis direction *negative*
= 1 Azimuth axis direction positive

----

elevation axis units* = none or *degrees

● **integer 43** *= 0* Elevation axis units *linear*
= 1 Elevation axis units degrees

----

elevation axis direction* = positive or *negative

● **integer 44** *= 0* Elevation axis direction *negative*
= 1 Elevation axis direction positive

----

spindle x motion* = false or *true

● **integer 45** *= 0* Spindle movement **NOT** in x axis
= 1 Spindle movement in x axis

----

spindle y motion* = false or *true

● **integer 46** = 0 Spindle movement NOT in y axis
= 1 Spindle movement in y axis

----

spindle z motion* =trueor*false

● **integer 47** *= 0* Spindle movement **NOT** in z axis
= 1 Spindle movement in z axis

----

spindle w motion* = false or *true

● **integer 48** = 0 Spindle movement **NOT** in w axis
= 1 Spindle movement in w axis

----

spindle azimuth rotation* = false or *true

● **integer 49** = 0 Rotation on **table** azimuth axis
= 1 Rotation on **spindle** azimuth axis

----

spindle elevation rotation* = false or *true

● **integer 50** = 0 Rotation on **table** elevation axis
= 1 Rotation on **spindle** elevation axis

● ***integer 51*** *= (n)* ***decimal output formats***

= 1 numbers less than 1 written as **. xxxx** (eg. **. 871**)
integers written as **xxxx .** (eg. **34 .**)

= 2 numbers less than 1 written as **0 . xxxx** (eg. **0 . 871**)
integers written as **xxxx .** (eg. **34 .**)

= 3 numbers less than 1 written as **. xxxx** (eg. **. 871**)
integers written as **xxxx . 0** (eg. **34 . 0**)

= 4 numbers less than 1 written as **0 . xxxx** (eg. **0 . 871**)
integers written as **xxxx . 0** (eg. **34 . 0**)

= 5 same as 3

= 6 numbers less than 1 written as **. xxxx** (eg. **. 871**)
integers written as **xxxx** (eg. **34**)

● **integer 53** not used

● **integer 54** = 0 Cycle Clearplane **absolute position**
= 1 Cycle Clearplane ***incremental move***

● **integer 55** = 0 Cycle Drilling depth **absolute end position**
= 1 Cycle Drilling depth ***incremental move to end***

----

maximum block number = (n)

● **integer 56** = 0 57385 default number before reset to 0

----

maximum segment = (n) Maximum tape length, or tool path distance, before a tape insert
(Link with Int 60)

● **integer 57** = **0 .** **NO** segmentation insert (**Note: Point needed**)
= **100 .** Insert at 100 feet of tape, or 100 feet of tool travel.

----

minimum tape blocks = (n) Minimum number of blocks, etc. in split tap and segment.

● **integer 58** = 0 NO action

● **integer 59** not used

----

segment type = (n) *(Link with Int 57)*

● **integer 60** = 0 Segment tape using feet of tape
= 1 Segment tape using distance tool travelled
= 2 Segment tape using number of blocks

----

counter start = (n)

● **integer 61** = 0 Counter start value (for use with variable counter)

----

counter increment = (n)

● **integer 62** = 0 Counter increment value

----

tape split retract distance = (n)

● **integer 63** = n Distance for the tool to retract from surface when the tape splits.

= 0 NO retract
= 100 Retract 100 units from Lift Point
= -999 Any **NEGATIVE** number, retract to Datum Z above surface

----

● **integer 64** not used

● **integer 65** = 0 Not used
= n Used only for robot control ??

● **integer 66** = 0 Print file width (0 = 132 character width)

●

integer 67 = ? Final tool number for preselection (-1 last tool)
(Extremely rare use - Use ToolNum[x] , NextTool , etc.,)

- **integer 68** not used

- **integer 69** = 0 **Reduce** plunge feedrate for tapping (0.8 Normal Feedrate)
  = 2 **Normal** plunge feedrate for tapping

- **integer 70** = 0 **Normal** feedrate output
  = 2 Calculate **separate** feedrates for X, Y, Z axis moves

----

- **integer 71** = 0 **NO** special output
  = 1 Special multiaxis feedrate (*set with Int 72*)

- **integer 72** = 0 **NO** special output
  = 1 Special multiaxis feedrate (*Inverse time*)

- **integer 73** = (n) Special character at tape end (default 0 : 19 =)

- **integer 74** = 0 Normal multiaxis output
  = 2 Multiaxes used for tool vector output

- **integer 75** = 0 Normal feedrate output
  = 1 Special feedrate output for Heidenhain **h33** control

- **integer 76** not used

(Constant Contour speed code output)

- **integer 77** = 0 Ramp down code calculated on next angle (Heid default)
  = 2 Ramp down code calculated on previous angle (All other controls)

- **integer 78** = 1 Split Tape increment (e.g. t01 , t02 etc.)

- **integer 79** = 0 Normal standard output
  = 1 Output start and end in agie wirespark format

- This list refers to the most common of Ductpost used integers and named aliase, the information is as accurate as we can make it but changes in the development of Ductpost may effect an output, so use with care and check thoroughly if used in an option.

(Return to TOP)

CLdata Code List

(Up dated 31/01/2001)

This list provides an indication of the British Standard (BS5110-Pt.2 : 1978) CLdata used by DUCT and PowerMill.

NOTE :- PowerMill has extended the use of some of these functions that will not be recorded here.

MAJOR WORD	INTEGER CODE NUMBER	MEANING
END	1	END
STOP	2	STOP
OPSTOP	3	OPTIONAL STOP
RAPID	*5*	*RAPID*
SWITCH	6	SWITCH
RETRCT	7	RETRACT
DRESS	8	DRESS
UNLOAD	10	UNLOAD
PENUP	11	PEN UP
PENDWN	12	PEN DOWN
BREAK	16	BREAK
GOHOME	17	GO HOME
HEAD	1002	HEAD
MODE	1003	MODE
CLEARP	1004	CLEARANCE PLANE
TMARK	1005	TAPE MARK
REWIND	1006	REWIND
CUTCOM	1007	CUTTER COMPENSATION
FEDRAT	1009	FEEDRATE
DELAY	1010	DELAY
AIR	1011	AIR
OPSKIP	1012	OPTIONAL SKIP
LEADER	1013	LEADER
PPLOT	1014	POST-PROCESSOR PLOT
MACHIN	1015	MACHINE
MCHTOL	1016	MACHINING TOLERANCE
SEQNO	1019	SEQUENCE NUMBER
DISPLY	1021	DISPLAY
AUXFUN	1022	AUXILIARY FUNCTION
TOOLNO	1025	TOOL NUMBER
ROTABL	1026	ROTATE TABLE
ORIGIN	1027	ORIGIN
COOLNT	1030	COOLANT
SPINDL	*1031*	*SPINDLE*
TURRET	1033	TURRET
ROTHED	1035	ROTATE HEAD

THREAD	1036	THREAD
TRANS	1037	TRANSLATE
OVPLOT	1042	OVERPLOT
LETTER	1043	LETTER
PPRINT	1044	POST-PROCESSOR PRINT
PATNO	*1045*	*PART NUMBER*
INSERT	1046	INSERT
PREFUN	1048	PREPARATORY FUNCTION
COUPLE	1049	COUPLE
PITCH	1050	PITCH
CYCLE	*1054*	*CYCLE*
LOADTL	1055	LOAD TOOL
SELCT	1056	SELECT TOOL
CLRSRF	1057	CLEARANCE SURFACE
DRAFT	1059	DRAFT
LINTOL	1067	LINEARISATION TOLERANCE.
CLDIST	1071	CLEARANCE DISTANCE
CHUCK	1073	CHUCK
CLAMP	1074	CLAMP
PPFUN	*1079*	*POST-PROCESSOR FUNCTION*
STAN	1080	SETTING ANGLE
OFSTNO	1083	OFFSET SWITCH NUMBER
PIERCE	1090	PIERCE
SAFPOS	1094	SAFE POSITION

The selction below indicates use, taken from a PowerMill Cldata output example :-

```
    1    2000   1045      PARTNO
               cachsendrehung
----------------------------------------------------------------------------
    7    2000   1031      SPINDL ON
               SPINDLE SPEED =   10000.0 RPM CLW
               RANGE Undefined (Select range)
               TOOL TIP RADIUS = 0.0
----------------------------------------------------------------------------
    5    2000   1079      PPFUN
               FEDRAT 10000 500 500
----------------------------------------------------------------------------
    6    2000   1079      PPFUN
               CUTLEN 79.60
----------------------------------------------------------------------------
    7    2000   1079      PPFUN
               CUTTIM  9.00
----------------------------------------------------------------------------
   17    2000      5      RAPID
----------------------------------------------------------------------------
   18    5000      5      RAPID TOOL MOTION RECORD
   X-COORD    Y-COORD    Z-COORD        TOOL X      TOOL Y      TOOL Z
 19.136200   0.000000   16.996048     -0.139173   0.000000    0.990268
----------------------------------------------------------------------------
```

```
51    2000   1054       CYCLE (288)
             CLEAR PLANE =  5.000000
             FEEDRATE = 500.000000
             NO. OF PECKS = 15
             CYCLE Z HEIGHTS (PECK HEIGHTS) =
               7.000000
```

Word[x] Numbers

(Up-dated 29/12/2000)

In the inbuilt machine source files of Ductpost there is an initial list of defined words, the lists given below are the values of those words, and the associated letter for the various controls :-

[No.]	Standard Word / *Address Letter List*	Other Controls	Variations
1	define word **/** address letter = "/"	.	.
2	define word **N** address letter = "N"	**h155** **heid** **heid400**	address letter = " " address letter = " " address letter = " "
3	define word **G1** address letter = "G"	**dyna** **h155** **heid** **heid400** **siem850**	address letter = "(" address letter = "L" address letter = "L" address letter = "L" address letter = "GD"
4	define word **G2** address letter = "G"	**dyna** **h155** **heid** **heid400**	address letter = ")" address letter = "C" address letter = "C" address letter = "C"
5	define word **G3** address letter = "G"	.	.
6	define word **G4** address letter = "G"	**h155** **heid** **heid400**	address letter = "CYCL DEF" address letter = "CYCL DEF" address letter = "CYCL DEF"
7	define word **G5** address letter = "G"	**h155** **heid** **heid400**	address letter = " " address letter = " " address letter = " "
8	define word **G6** address letter = "G"	**h155** **heid** **heid400**	address letter = "CYCL DEF" address letter = "CYCL DEF" address letter = "CYCL DEF"
9	define word **G7** address letter = "G7"	**ab84** **fanuc** **fanucom** **fanuc6m** **fanuc11m** **fanuc15m** **h33**	address letter = "G" address letter = "G" address letter = "G" address letter = "G" address letter = "G" address letter = "G" address letter = "G"
10	define word **X** address letter = "X"	.	.
11	define word **Y** address letter = "Y"	.	.
12	define word **Z** address letter = "Z"	.	.
13	define word **I** address letter = "I"	**h155** **heid** **heid400**	address letter = "X" address letter = "X" address letter = "X"
14	define word **J** address letter = "J"	**h155** **heid** **heid400**	address letter = "Y" address letter = "Y" address letter = "Y"
15	define word **K** address letter = "K"	**h155** **heid** **heid400**	address letter = "Z" address letter = "Z" address letter = "Z"
16	define word **R** address letter = "R"	**bosch** **deckel4** **deckel11** **maho** **ph432** **siem850**	address letter = "," address letter = "SA" address letter = "SA" address letter = "B" address letter = "B" address letter = "R2="
17	define word **R2** address letter = "R"	**h155** **heid** **heid400** **heidiso** **num** **siem850**	address letter = ".1 SET UP" address letter = ".1 SET UP" address letter = ".1 SET UP" address letter = "PO1" address letter = "ER" address lettter ="R2"
		bosch **deckel4**	address letter = "," address letter = "MI"

18	define word **Q** address letter = "Q"	**deckel11** **fidia** **heidiso**	address letter = "MI" address letter = "D" address letter = "PO3"
19	define word **A** address letter = "A"	.	.
20	define word **B** address letter = "B"	.	.
21	define word **C** address letter = "C"	.	.
22	define word **U** address letter = "U"	.	.
23	define word **V** address letter = "V"	.	.
24	define word **W** address letter = "W"	.	.
25	define word **Z2** address letter = "Z"	**bosch** **deckel4** **deckel11** **h155** **heid** **heid400** **heidiso** **fidia** **siem850**	address letter = "[" address letter = "TA" address letter = "TA" address letter = ".2 DEPTH" address letter = ",2 DEPTH" address letter = ".2 DEPTH" address letter = "PO2" address letter = "E" address letter = "R3="
26	define word **F** address letter = "F"	.	.
27	define word **S** address letter = "S"	.	.
28	define word **T** address letter = "T"	.	.
29	define word **M1** address letter = "M"	**h155** **heid**	address letter = " " address letter = " "
30	define word **M2** address letter = "M"	**h155** **heid**	address letter = " " address letter = " "
31	define word **L** address letter = "L"	.	.
32	define word **P** address letter = "P"	.	.
33	define word **D** address letter = "D"	**fagor**	address letter = "**.**"
34	define word **E** address letter = "E"	**dyna** **heidiso**	address letter = "$" address letter = "*"
35	define word **H** address letter = "H"	**h155** **heid** **heid400**	address letter = "D" address letter = "D" address letter = "D"
36	define word **O** address letter = "O"	.	.
37	define word **MS** address letter = "("	**acra8** **bosch** **bostom** **tiger**	address letter = "(MSG," address letter = "(MSG," address letter = "%" address letter = "/"
38	define word **EM** address letter = ")"	.	.
39	define word **EOB** address letter = "*"	**ab84** **acra8** **bosch** **bostom** **deckel4** **deckel11** **dyna** **eberle** **elexa** **fadal** **fagor** **fanuc** **fanucom** **fanuc6m** **fanuc11m** **fanuc15m** **fidia** **h33** **h155** **heid** **heid400** **heidiso** **hurco** **maho**	 define word OP address letter = "/" define word N2 address letter = "**:**" define word ID address letter = "(DFS, P" define word OP address letter = "/" define word N2 address letter = "N*" define word OP address letter = "/" define word OP address letter = "/" define word N2 address letter = "N*" define word OP address letter = "/" define word OP address letter = "/" define word OP address letter = "/" define word OP address letter = "/" define word OP address letter = "/" define word OP address letter = "/" define word OP address letter = "/" define word OP address letter = "/" define word null address letter = " " define word OP address letter = "/" define word FF address letter = "F" define word FF address letter = "F" define word FF address letter = "F" define word CF address letter = "PO5" define word OP address letter = "/" define word K2 address letter = "J0.2K" define word OP address letter = "/"

		matsura	define word OP address letter = "/"
		mazak	define word OP address letter = "/"
		mitsu	define word OP address letter = "/"
		num	define word OP address letter = "/"
		okuma	define word K2 address letter = "J0.2K"
		ph432	define word OP address letter = "/"
		selca	define word OP address letter = "/"
		siem850	define word OP address letter = "/"
		tiger	
40	.	ab84	
		acra8	define word null address letter = " "
		bosch	define word OP address letter = "/"
		bostom	define word PA address letter = ","
		deckel4	define word null address letter = " "
		deckel11	define word OP address letter = "/"
		dyna	define word ID address letter = "%"
		eberle	define word null address letter = " "
		elexa	define word OP address letter = "/"
		fadal	define word ID address letter = ":"
		fagor	define word ID address letter = ":"
		fanuc	define word ID address letter = "%"
		fanucom	define word ID address letter = ":"
		fanuc6m	define word ID address letter = ":"
		fanuc11m	define word ID address letter = ":"
			define word ID address letter = ":"
		fanuc15m	define word ID address letter = ":"
			define word null address letter = " "
		h33	define word XXF address letter = " "
		h155	define word XXF address letter = " "
		heid	define word T1 address letter = "TOOL DEF"
		heid400	define word ID address letter = "%"
		heidiso	define word Z3 address letter = "Z"
		hurco	define word OP address letter = "/"
		maho	define word ID address letter = "O"
		matsura	define word ID address letter = "O"
		mazak	define word ID address letter = "O"
		mitsu	define word ID address letter = " "
		num	define word ID address letter = "O"
		okuma	define word OP address letter = "/"
		ph432	define word ID address letter = ":"
		selca	define word ID address letter = "&MPF"
		siem850	define word ID address letter = ":"
		tiger	
41	.	acra8	
		bosch	define word ID address letter = "(MSG,"
		deckel4	define word EI address letter = ", R W E D)"
		deckel11	define word ID address letter = "(&%"
		eberle	define word P1 address letter = " (M-D11-000000--"
		elexa	define word ID address letter = "(&%"
		fadal	define word QZ address letter = "QZ"
		fagor	define word Q1 address letter = "Q"
		fanuc	define word null address letter = " "
		fanucom	define word Q1 address letter = "Q"
		fanuc6m	define word Q1 address letter = "Q"
		fanuc11m	define word Q1 address letter = "Q"
			define word Q1 address letter = "Q"
		fanuc15m	define word Q1 address letter = "Q"
			define word MP address letter = "M"
		h155	define word MP address letter = "M"
		heid	define word T2 address letter = "TOOL CALL"
		heid400	define word null address letter = " "
		heidiso	define word null address letter = " "
		hurco	define word ID address letter = "N9"
		maho	define word null address letter = " "
		matsura	define word null address letter = " "
		mazak	define word Q1 address letter = "Q"
		mitsu	define word P1 address letter = "P1="
		num	define word Q1 address letter = "Q"
		okuma	define word ID address letter = "N9"
		ph432	define word null address letter = " "
		selca	define word R11 address letter = "R11="
		siem850	define word null address letter = " "

		tiger	
42	.	acra8 bosch deckel4 deckel11 eberle elexa fadal fanuc fanucom fanuc6m fanuc11m fanuc15m h155 heid heid400 maho ph432 num siem850	define word P1 address letter = "P1=" define word OP address letter = "?" define word ED address letter = "/000000)" define word P2 address letter = "." define word ED address letter = "/000000)" define word PA address letter = "PA" define word null address letter = " " define word null address letter = " " define word null address letter = " " define word null address letter = " " define word null address letter = " " define word null address letter = " " define word T1 address letter = "TOOL DEF" define word T1 address letter = ";TOOL DEF" define word TL address letter = "L" define word P1 address letter = "P1=" define word P1 address letter = "P1=" define word null address letter = " " define word null address letter = " "
43	.	acra8 bosch deckel11 eberle elexa h155 heid heid400 maho ph432	define word null address letter = " " define word EC address letter = "]" define word P3 address letter = "." define word null address letter = " " define word null address letter = " " define word T2 address letter = "TOOL CALL" define word T2 address letter = "TOOL CALL" define word RR address letter = "R" define word null address letter = " " define word null address letter = " "
44	.	bosch deckel11 h155 heid heid400	define word null address letter = " " define word P4 address letter = "-" define word TL address letter = "L" define word TL address letter = "L" define word null address letter = " "
45	.	deckel11 h155 heid heid400	define word null address letter = " " define word RR address letter = "R" define word RR address letter = "R" define word Z3 address letter = ".3 PECKG"
46	.	h155 heid heid400	define word null address letter = " " define word null address letter = " " define word ID address letter = " "
47	.	h155 heid heid400	define word Z3 address letter = ".3 PECKG" define word Z3 address letter = ".3 PECKG" define word Z4 address letter = "Z"
48	.	h155 heid heid400	define word ID address letter = " " define word ID address letter = " " define word OP address letter = "/"
49	.	h155 heid heid400	define word Z4 address letter = "Z" define word Z4 address letter = "Z" define word DR address letter = "DR"
50	.	h155 heid heid400	define word OP address letter = "/" define word OP address letter = "/" define word circ address letter = "CC"
51	.	h155 heid heid400	define word DR address letter = "DR" define word DR address letter = "DR" define word DW address letter = "DWELL"
52	.	h155 heid heid400	define word circ address letter = "CC" define word circ address letter = "CC" define word FMAX address letter = "F9999"
53	.	h155 heid	define word DW address letter = "DWELL" define word DW address letter = "DWELL"
54	.	h155 heid	define word FMAX address letter = "F9999" define word FMAX address letter = "FMAX"

For use of word values see :- Word Values

(Return to TOP)

Option File Construction Example.

Lesson 1

(Up-dated 10/10/2001)

Introduction

There is no hard and fast rule to the construction of an option file, and the following is but a guide that may make the task a little easier for those who are first timers, or who may even be old timers requiring enlightenment.

An option file is a means of modifying the inbuilt machine standard DUCTpost Processor output to meet a customer's preferred output format, or to add additional features to the standard format. More than one option can be constructed for the same Licenced machine control to meet different requirements, such as output for Metric or Imperial, Absolute or Incremental, as but examples. Each of course, will have a different option file name, but will internally be very similar in content.

The requirements to create an option file are as follows:-

1/ A plane text editor.
2/ A PAF Licence for the particular machine control. ******
3/ A PAF Licence for **LONG** if the option exceeds **50 working lines** of code. ******
(Excludes spaces, blank lines, or # commented lines)
(**** This will no longer be required from DP1331**)

Two ways to start to create an option are available, one is to copy the dummy option for the machine control, residing in the ***/dcam/config/ductpost/***directory, or to literally start from scratch with a blank sheet. If one starts from scratch the following four lines are essential.

Line 1. machine xxxxx ## xxxxx is the machine control name. e.g. ***fanuc***
Line 2 *blank*
Line 3. end ## ensure to hit return after typing
Line 4 *blank*

" machine xxxxx" , must always be the first line, and " end " the last but one in the option file, and have no leading spaces

Save with an appropriate **[filename]** with the extension **.opt**, usually to your **working** directory whilst constructing and testing it .

This will now be the base for the rest of the option file construction.

Example Tape File

The following is a typical example of a customer's requested output format for his tape files and we have been informed that the control is Fanuc.

```
%
O5222
(THIS PROGRAM MACHINES THE)
(FOR PJ#  )
(CNC MILL LOCATION= X, Y, Z)
(FILE NAME  )
(DATE LAST RAN:)
(DATE LAST EDITED:)
(OPERATOR   )
(PROGRAMMER:)
G17G20G90G54
T99
T1
M6
```

```
S2400M3(TOOL 1 IS A 1/2" HOG BALL)
G0X0Y0
G43Z3.H1
M8
G0X0Y.0475
G0Z.5
G1Z0F15.
........................
.......................
G0Z.2
T2
M6
S2400M3(TOOL 2 IS A 1/4" CARB BALL)
G0X0Y0
G43Z3.H2
M8
G0Z.5
G1Z0F30.
........................
........................
T99
M6
M2
%
```

(Return to :- Tool Change Option)

The suggested approach to providing an option file to produce this output format is as follows:-

STAGE 1, Essentials

First rule is to ascertain what machine control an option is to be provided for, Fanuc et al, can cover a wide variety, as can some of the others, e.g.. Heid, Fadal etc.

Second rule is to try and obtain a good representative tape file covering the various operations the machine does, such as, initialisation start, tool change functions, circular output requirements, is it multi-axes, and if so, details of axes etc., in fact as much information as possible. This makes guess work minimal. (In the above example the information is sparse, but covers tape start, tool changes, tape end, and additional feature requirements).

Third rule base your option on the nearest Ductpost inbuilt control to the actual one you want to produce for. This is not strictly necessary but a good start base, and will only fall down if there isn't a representative control available. In this case you have a choice, use the " **standard** ", or an in-built control that works similarly.

To obtain a listing of the in-built controls if you don't have the DuctPost User Guide manual by you, or access to the Web Page, type in a window :- **ductpost -l** (small L).

It is a big help if you have two, or three, short working cut files that will enable you to test the option to ensure it is producing the required output. These cut files should cover various functions so that the majority of operations can be verified, as it is not unknown that something one has done in one area may affect something else entirely unrelated. Ductpost unfortunately is full of surprises, so beware, and test thoroughly to ensure you aren't caught out.

(Review this section Example)

(Back to Tape End)

Basic Start :-

Right, lets start from the begining, and work up the option in stages to produce the Tape file indicated.
We will, in this case, use a **Fanuc** control.

With a cut file at hand, first run it through the in-built to see what the base output looks like. In the working window type : - **ductpost fanuc [filename].cut** ## ***.cut*** can usually be omitted, unless the extension is other than " **.cut** "

It is recommended that all Post Processing is done in a working window as it is easier to spot any error messages in the posting operation, though with the latest release of Ductpost, it will stop posting if there are any major errors in the option. (The error message will usually give a good indication as to what is wrong.)

This will also occur if there are any changes in the later releases that affect older versions of Ductpost, so be warned. If you get a customer saying he can't post with the new version, ask for the error message which should indicate what has to be changed in his option file. (See What's NEW in Ductpost)

With **PowerMill 3.0** this will not be necessary as Ductpost has its own window, but you may find it awkward to use if there are a number of changes to test out.

{As an example " ***retract distance*** *" if included in an option, will need to be changed to "* ***tape split retract distance*** *". }*

If you compare what we want, with what the base output produces, it probably at first glance doesn't look very similar.

To start with, open a text dump of the relevant in-built machine control to use as source. To do this, in the operating window type :

ductpost -w [control name] > [control name].dmp (Keep this for future reference in a separate directory)

First thing to note is the start produced by the in-built, and the start required.

IN-BUILT OUTPUT	REQUIRED
%	%
:0001	O5222
N10G91G28X0Y0Z0	(Messages - 8 lines)
N20G40G17G80G49	G17 G20 G90 G54
N30G0G90Z10.	

The following convention will be used to try and avoid confusion, ' **X** ' will indicate original and, ' **X** ' will indicate the change, in both cases ignore the quotes unless appropriate.

The programme ID is different ' **:** ' instead of ' **O** ', and block numbers are output. So tackle those first.

In the source dump, find ' **ID** ' which will be ' **define word ID** ', copy this block of three lines to the new option file, inserting it below the first blank line after ***machine fanuc***, and change the following :- ' **define word ID** ' to ' **define format (ID)** ' and ' **address letter = ":"** ' to ' **address letter = "O"** '

Do the same for ' **N** ' , placing it beneath the ID block, but this time delete the address letter line and substitute ' **not permanent** '

Save and Run this with your cut file to see what the output is like.

You will see that the ' **:** ' is now ' **O** ' , and the block numbers have disappeared, but we now have a whole string of messages (------------) that will probably not be required. So to suppress these, insert in the option after the last block, and before ' **end** ' , ' **message output = false** '

Next, tackle the tape header. Copy the ' **define block tape start** ' into the option after the ' **message output = false** ' line.

Ignore the (required messages) for the moment, some of these we can't supply anyway, get the set up codes right first. The first two lines of the block are correct, the last three can be reduced to one line, and the **T99** added as shown :-

Old method	**Prefered method**

N ; G3 17 ; G4 20 ; G5 90 ; G6 54	N ; xy plane ; imperial data ; absoloute data ; G6 54
N ; T 99	N ; tool number 99

The G codes are taken from the ***dump*** file ' **define codes** ' list, but as the ' **xy plane** ' is **G3 17**and, the ' **from** ' code is **G3 54**, we can't have two **G3** groups on the same line, so set **G3 54** as **G6 54** , which is acceptable. Try and keep them in sequence, it is easier to debug, and usually the machine is not particular about the order.

Next from the example we see that there is a **T99** at the start and end, so it can be assumed that these are fixed. Therefore add this in the start block as shown above. You will note that **tool number** (**T)** has been used for this and set to **99**.

It is perhaps pertinent to say something about the format at this point, and certain conventions in constructing an option file.

a) **NEVER** use the **Tab key** to indent, always the SPACE BAR.
b) **SPACES** must be used before and after the **;** (semi-colon) separator.
c) A **SPACE** is required between a Word and its Value. i.e. **T** space **99** space **;** space **G6** space **54**
d) A **#** is used as **REM** statement, for a comment.
c) It is recommended that a SPACE is entered between each block definition to aid clarity.

e.g. end define
SPACE or **#**
define block **.............**

Save, and test to see the results so far.

Tool change comes next, and if we look at the ***dump*** file you will note there are three, ' **define block tool change** ' , ' **define block tool change first** ' , ' **define block tool change clear** ' . The ' **first** ' and ' **change** ' have data defined in them, whereas ' **clear** ' is blank.

' **define block tool change first** ' is defined if there is any special requirements needed to be output at this stage, and not at any other tool change operation thereafter, which are taken care of in ' **define block tool change** '.
In this case we note from the example that all indicated tool changes are the same output format so we can use ' **define block tool change** ' for all of them and delete the data in ' **define block tool change first** '. Better still, use the following format :-

define block tool change first
N ;
end define

Ignore ' **define block tool change clear** ' as it is very rarely used, but **always check** to see that it isn't. (We will not spend time on this one, save to say it clears all tool change functions specified within it)

Copy both blocks from the source file and drop them after the ' ***define block start*** '. Edit them as follows :-
' **first** ' - delete all, leaving just ' **define block tool change first** ' and ' **end define** ' lines (or preferably as indicated above)
' **change** ' - delete the first three lines, as there are no pre-sets before a change required, and also the last line, leaving just

N ; T ToolNumber ; change tool (**N ; tool number ToolNum ; change tool** can also be used)

To obtain the basic output we require for the tool changes, edit as follows, ignore for the moment the (Tool data).

N ; T ToolNumber
N ; M1 6 (or, change tool)
N ; S ToolSpeed[ToolNum] ; M1 3 (N ; spindle ToolSpeed[ToolNum] ; spindle on cw)

Save and test. You should find that the tool change output is nearly right to the **M8** in the example, except that there is an extra Spindle and M3 output on the first move line.

These come from the **rapid block** so that is the next to tackle.

Copy from the ***dump*** file the rapid block and then edit it as follows :-

```
   N ; G1 ; G2 ; G3 ; G5 ; G6 ; X ; Y ; Z ; H ; M1 ; M2
 or preferably
    N ; rapid ; G2 ; G3 ; G5 ; G6 ; x coord ; y coord ; z coord ; tool length ; M1 ; M2
```

This will leave the **M3** and to get rid of this, insert just below the ' **message output = false** ' line, ' **integer 12 = 0** '.

This will suppress the **M3** output (See Array Data Information)

Save and test. The output should now resemble the basic format of the tape start, and the tool changes, without the (MESSAGES) which will be tackled later in **Lesson 2**.

(Review this section Example)

(Back to Option File)

The only changes required at this stage is the tape ending.

However, before we do that we need to re-dress an omission in the in-built source code (***dump list***), in that there is nothing defined in the ' **define block move linear** '. Normally in the earlier versions of Ductpost this would not be of consequence, and even now may not cause a problem, but there have been a couple of instances with the up dated Ductposts where this omission has caused problems.
So as a matter of course it is now recomended adding a definition whenever this is missing from the in-built. The minimum needed is as follows, adding this to the option below the ' **define block move rapid** '.

```
define block move linear
   N ; linear ; G2 ; x coord ; y coord ; z coord ; tool radius ; feedrate ;  M1 ; M2
end define
```

Back to tape end. Copy the ' **define block tape end** ' over to your new option and drop it in after the new ' **define block move linear** ' we have just created.
This you will note will require a complete revision as there is a lot of surplus information provided by the in-built that is not required by the new option output. It is therefore sensible to **<u>delete</u>** the block contents and start afresh, and add the following new data :-

```
N  ;  tool number 99
N  ;  change tool
N  ;  end of tape
"%"
```

One additional factor now needs to be included, in that the option is required for the USA that works to the Imperial measure (**Inches**), so we need to insure Ductpost defaults to this system.
This can be achieved in two ways by inserting one, or the other, of the following lines below the ' **integer 12 = 0** ' line.

' **units = input** ' or ' **units = imperial** '

Ductpost usually defaults to the former, whereby whichever units are input, the output will reflect this, whereas the second will convert **Metric** input to **Imperial** output. (Conversely, ' **units = metric** ' will convert **Imperial** input to **Metric** output.) Use the second.
(**WARNING :- Will NOT CONVERT,** set values such as :-
withdrawal amount = 100.0 ; **tape split retract distance = 50.0** ; **xcoord 53000** ; **And others** so these will need to be defined in the appropriate unit values)

Save, and try.

With this change we should find that the output matches the bulk of the example tape, and we can now go on to add the (messages) in the tape header. (See Message Data added)

Before we close on this stage it is worth adding to the option file a history record so that any changes can be recorded if modifications are made. An example of this can be seen below :-

(Review this section Tape end)

Option File contruction so far :-

```
machine fanuc
#2
#   History
#   Option for North American Support - WizzyWig M/c. with Fanuc control
#   am 15/12/98  Ref.No. 0280  Issue 1.1
#
#
#8                                  ## Line number to aid debugging.
   define format ( N )
     not permanent
   end define
#12
   define format ( ID )
    address letter = "O"
   end define
#16
   message output = false
   integer 12         = 0
   units              = imperial
#20
  define block tape start
     "%"
     ID ProgID
     N ; xy plane ; imperial data ; absolute data ; G6 54
     N ; tool number 99
  end define
#27
  define block tool change first
     N ;
  end define
#31
  define block tool change
     N ; tool number ToolNumber
     N ; change tool
     N ; spindle ToolSpeed[ToolNum] ; spindle on cw
  end define
#37
  define block move rapid
     N ; rapid ; G2 ; G3 ; G5 ; G6 ; x coord ; y coord ; z coord ; tool length ; M1 ; M2
  end define
#41
  define block move linear
     N ; linear ; G2 ; x coord ; y coord ; z coord ; tool radius ; feedrate ; M1 ; M2
  end define
#45
  define block tape end
     N   ; tool number 99
     N   ; change tool
     N   ; end of tape
    "%"
  end define
#52
end
```

(Back to :- Start , Example , Tool Change Top)

Option File Construction Example.
Lesson 2

(Up-dated 10/10/2001)

Introduction

Following on from **Lesson 1** (Basic Start) we will now look at introducing (where possible) the tape start message content of the example , the relevant parts of the tape file are shown below :-

```
%
O5222
(THIS PROGRAM MACHINES THE)
(FOR PJ#  )
(CNC MILL LOCATION= X, Y, Z)
(FILE NAME  )
(DATE LAST RAN:)
(DATE LAST EDITED:)
(OPERATOR   )
(PROGRAMMER:)              ##____^^ Tape Start Info ^^____
G17G20G90G54
T99
T1
M6
S2400M3(TOOL 1 IS A 1/2" HOG BALL)    ## Tool Info
G0X0Y0
G43Z3.H1
M8
```

The capabilities of Ductpost prior to DP1205 were extremely limited in providing bespoke text output, and as such we could not meet anywhere near that indicated in the example tape requirements.   However, we are now able to get at individual parts of the ' **Part Name** ' data output by Powemill, which allows us a little more flexibility.   From DP1331 there is more information becoming available from PowerMill that can be utelised by versions later than this.

(Back to Prog.Number)

Tool Information Data

If we first tackle the Tool information required, e.g. **(TOOL x IS A 1/2" HOG BALL)**.

As it stands we have no chance, the best we can provide would be the following :- **(TOOL x Dia. 0.5 Tip Rad. 0.25)** as indicated, or all in capitals, take it or leave it.   Tool length could be added but as this is not required we'll forget it, but the same method of approach would apply if it was. (It is not possible to provide Imperial fraction format)

First we need to create some new words to latch data to, these will be for Tool Dia. and Tip Radius. If we add the following to our option to start with, placing them after the ' **machine xxxx** ' line and before ' **define format (N)** '.

[The reason for this is that new words are added to the end of the in-built list, in the order they are placed in the option file, and if we need to obtain the word[number] value for any reason, it is easier to find this out with a ***dump*** of the option if they are placed at the start, sooner than scattered all through the option].

```
    define word TN
        address letter  = "(TOOL "
        address width  = 5
        field width        = 2
    end define
#
   define word TD
        address letter  = " Dia. "
        address width  = 6
        field width        = 5
         scale factor      = 2
    end define
#
   define word TR
        address letter  = " Tip Rad. "     ## include spaces if required
        address width  = 10                ## count spaces as width
    field width       = 5                    ## set to number of places + point
  end define
```

We then can format the additional general characteristics of the above, adding this after the new words :-

```
  define format ( TD  TR )               ##  TN is defined sufficiently above.
    metric formats
    decimal point   = true
    decimal places = 3
    leading zeros   = false
    trailing zeros   = false
    imperial formats = metric formats
  end define
```

and finally we need to add these words to the word order list, otherwise nothing will appear in the output.

The simplest way to start off with is to tack them on the end of the existing in-built list by adding the following line to the option :- " **word order = (+ TN TD TR)** " placed in the order they are to appear in the tape file. Put this before the ' **message output = false** ' line.

Next is to add the data to the ' **define block tool change** ' as below :-

```
   N ; spindle ToolSpeed[ToolNum] ; spindle on cw ; TN ToolNumber ;
      TD ToolRadius[ToolNum]  ; TR ToolRadius[ToolNum] ; EM =C
```

This is a one line output, the semi-colon at the end of the first line, hangs the second line on to it.
REMEMBER at all times use the space bar to create indents, **NEVER** the tab key. Ductpost doesn't like tabs.

An explanation of the above information is perhaps helpful at this point, though in part it may be self evident.

N Outputs the blocknumber (Line number). In this case not required but best to include, the customer may change his mind!!
spindle (S) Spindle speed output, defined by the variable ' ToolSpeed[ToolNum] '.
spindle on cw (M1 3) Auxilary Function used in this case to output **M3** Spindle on clockwise. An integer value entered with a word will invariably mean that this value is carried through and output until changed. (More about this later)
TN Is address holder to output the **ToolNumber**
TD Is the address holder for the Tool Diameter provided by the **ToolRadius** statement, this is ***multiplied by 2*** in the **TD scale factor** setting.
It is recomended that ' **ToolRadius[ToolNum]** ' is always used and **NOT** just ' ***ToolRadius*** ' on its own, as there is the possibility in certain circumstances where an incorrect value can be output.
TR Similar to TD and used to output the **Tip Radius** of the tool.
EM This is the ' **)** ' word of the messages output, and here we use the **=C** to force it out. One point while we are on this, it is possible that the EM word will output as **) 0** , which means the inbuilt definition field width has not been set to zero. If this is the case then it will mean that EM will need to be re-formatted. It would have been in order to use **")"** on its own instead of EM.

Note:-
spindle on cw (M1 3) = This indicates the prefered format (This represents the old format)
spindle on cw etc. must be **lowercase** and be defined in the key, or code blocks.

If we save and try these additions we note that the output produces the following :-

```
     S1850M3 ) (TOOL 1 Dia. 10. Tip Rad. 5.
```

Nearly right, but the ' **)** ' is not at the end as required. Why?

Well if we look at the inbuilt ***dump*** file ' **word order** ' list, we note that **EM** is not the last word in the list, and to make matters worse we have added further words to the end of this list in the option.
To overcome this we have two options,
1/ Copy the word order list from the inbuilt, add the additions we have made in the option, and then re-order the list to ensure the words output in the order we require them.
2/ Insert ' **block order = true** ' into the option below the ' **integer 12 = 0** ' line to ensure that the defined order in each block will output as required, overiding the word order listing.

(use **integer 9 = 5** for versions of **Ductpost1100, or earlier**, **BUT BEWARE IT CAN PLAY TRICKS**, especially with Heidenhains. Later versions from DP1203 been stablised, and either format is acceptable, preferably block order = true)

If we save and try it now we find that the **")"** is where it should be.

```
S1850M3 (TOOL 1 Dia. 10. Tip Rad. 5.)
```

A Note on Square [ ] brackets.

The statement ' **ToolRadius[ToolNum]** ' means output the ToolRadius for the tool specified for the particular tool change.
The statement ' **ToolRadius[2]** ' means output the ToolRadius for the tool specified for the 2nd Tool change, NOT Tool No. 2
Similarly **' S ToolSpeed[2] '** will output the 2nd Tool change speed setting.

Example :- **Temporarily** add this line to the tool change to see what the output produces.

```
N ; spindle ToolSpeed[2] ; spindle on cw ; TD =C ; TN ToolNumber[2] ;
   TD ToolRadius[2]  ; TR ToolRadius[2] ; EM =C
```

Output should look something like this, providing your cut file has a second tool change in the programme.

```
S1850M3 (TOOL 1 Dia. 10. Tip Rad. 5.)
 S2250M3 (TOOL 2 Dia. 6. Tip Rad. 3.)
```

Having tried this example **remove** the additional line, and we have now completed the first stage of the message requirement as far as tool data is concerned.

(Review this section - tool information)

(Back to Prog Data input)

1/ Tape Start Data. (Programme Number Input)

```
1   %
2   O5222
3   (THIS PROGRAM MACHINES THE)
4   (FOR PJ#  )
5   (CNC MILL LOCATION= X, Y, Z)
6   (FILE NAME  )
7   (DATE LAST RAN:)
8   (DATE LAST EDITED:)
9   (OPERATOR   )
10  (PROGRAMMER:)
11  G17G20G90G54
12  T99
```

The only details we could have provide Prior to DP1205 would be Line 2, the programme number, Line 3 the programme details, and the date the programme was compiled. From this up-date we can manage a little more with a little bit of effort.
It is possible to provide the CNC MILL LOCATION, if it is the Datum point of the Cut File, but as this is not specified in the request, further information on this point would be needed.
However, for this excersise we will assume that it is, and provide the necessary data.

Programme Number, **O5222**

The default **Ductpost** output is ' **1** ' via ' **ProgID** ', which can be re-defined by inserting ' **program id start = 5222** ', for example, in the option file.
This will then be the **fixed** program number hereafter, every time the option is used.
The default ' **1** ' can be invoked to produce ' **0001** **or** **1000** ' by re-defining the format of **word ID** to have ' **leading zeros = true** '**or**' **trailing zeros = true** ' inserted in the **define format (ID)** block.

However this may not be a desireable choice as the Number 5222 is obviously a variable number, and will normally be input at the time the cut file is post processed.

Various Methods of outputing a variable programme number are open to us:-

1/

Ductpost has a means of outputing a Variable Programme Number, but it has to be ' **Prompted** ' for at the time of running the post processor. This can be set up by inserting ' **use progid = false** ' in the option.
However, this will not work if the Post Processing is carried out through PowerMILL to produce an NC **tape** output instead of a

cut file, as a **prompt** is ignored, therefore we have to tackle this in rather round about ways to overcome this.

2/

Perhaps the simplest way is to use the **cut file**, **or tape file**, name to produce this via the variable ' **JobName** ', and defining the cut file, or tape file, as **5222.cut**, **or** **5222.tap**.

In the option ' ***define block tape start*** ' we need to change **ID ProgID** to **ID Jobname** . We will use this method in our example.

(NOTE:- The tape method will not function with Ductpost versions **prior** to DP1206)

3/

Another method open to us is via the PowerMILL **Part Name** input, which will allow a string of information to be input from which we can extract data in a limited way. To get a **Part Number** from the **Part Name** , it is essential that a consistant format of input is maintained at all times if this method is to be used.

If we take the following input as an example :-

(This comprises of a *Programme No*. **max 5 digits -** *Part Description* **max 25 characters -** *Company Name* **max 12 characters -** *Part No*. **max 6 characters -** *Programmer's Initials* **max 3 characters** with each block delimited by **a --,** and padded with **0 or ***)

Part Name 05222-PORT WING STRUT OF B52***-BOEING INC**-325A59-AM*

the part we are initially interested in lies in the first **5 characters** which is required to cater for a five figure number, but as 5222 is only four figures, a **leading zero** has been added to act as padding. **Leading zeros are essential** to maintain a consistant field width of 5 characters. (the rest of the string we will examine later)

The option as it stands now will need to be modified in the ' ***define block tape start*** ' by changing ' **ID ProgID** ' to , ' **ID PartID** '.

Because **word ID** is restricted at the moment to a field width of **4** places, this would therefore only extract the **first 4 characters** in the string, ' *0*5222 ', outputting *O*522 only, if **leading zeros** are not defined. As a five figure number is possible, the ***field width*** of **word ID** will therefore have to be re-defined to equal **5**.

PartID , this selects the first character as its starting point in the string, and the ***field width*** of **word ID** limits how many characters of the string are output from this point.

Our choices therefore are as follows:-

```
define format ( ID )
  leading zeros = true
end define

define block tape start
  N ; "%"
  N ; ID ProgID
   ......etc.
end define

%
O0001
.....etc.
```

or >

```
program id start = 5222

define block tape start
  N ; "%"
  N ; ID ProgID
   ......etc.
end define

%
O5222
.....etc.
```

```
cut file name ( and/or tap file from DP1206 )
5222.cut  ( 5222.tap )

define block tape start
  N ; "%"
  N ; ID JobName
   ......etc.
end define

%
O5222
.....etc.
```

----

or >

```
define format ( ID )
  field width = 5
end define

define block tape start
  N ; "%"
  N ; ID PartID
   ......etc.
end define

%
O5222
.....etc.
```

Note :- The trend would appear to be to use the ***tape / cut*** file name as the Programme Identifier so **JobName** is probably the best choice.

ID will usually be set to an 8 character field width as most machine tool controls will only accept this number.

(Review this section Programme Number)

(Back to Prog Date)

2/ Tape Start Data (Programme Data input)

Next we attack the rest of the Tape Start input to give some output for Lines 3, to 10 , and for this we will require some additional words defined.

We will start with line 3 first.   3   **(THIS PROGRAM MACHINES THE** [Part Descrition] **)**

Add the following to the option file, following the same method as we have done previously, placing the additions after the last defined word block..

```
define word ID2
   address letter  = "(THIS PROGRAM MACHINES THE "
   address width  = 27
   field width       = 33
end define
```

Add **ID2** to the ***word order*** line, before the **TD** in the brackets. Always ensure there is a space between each word.

Now modify define block tape start as follows :-

```
define block tape start
   "%"
   ID Jobname ( or ID PartID )
   ID2 PartID ; EM =C
   N ; xy plane ; incremental data ; absolute data ; G6 54
   N ; tool number 99
end define
```

Save and try

The output will probably look something like this :-

```
%
O5222
(THIS PROGRAM MACHIN       05222-PORT WING STRUT OF B52***-B )
G17G20G90G54
T99
```

Not quite what we want.   The **ID2** address has been truncated due to the number of characters allowed per word being limited to **19**, therefore we will have to re-compose the address to fit within this limit.
Also we note the limitation on the data output, which is retricted to **33 characters** by the ***field width*** of **ID2**

Re-define **ID2** word address to be :-   **" (PROGRAM MACHINES- "** , and change address width to **19**, and reset ***field width*** to **25**

```
define word ID2
   address letter  = "(PROGRAM MACHINES -"
   address width  = 19
   field width       = 25
end define
```

We will also need to redefine the ***tape start block*** as follows :-

```
define block tape start
   "%"
   ID Jobname ( or ID PartID)
   ID2 PartID[ 7 ] ; EM =C        ##  NotePartID[7]
   N ; xy plane ; incremental data ; absolute data ; G6 54
   N ; tool number 99
end define
```

PartID[7] *will select the* ***7th*** *character in the string as its start point, and the field width will restrict the string output to the part* *description only.*

The resultant output should be as follows :-

```
%
O5222
(PROGRAM MACHINES- PORT WING STRUT OF B52***)
```

(Review this section Programme data 2)

(Back to Tool Datum)

3/ Tape Start Data (Programme Data input cont'd)

The other sections can now be tackled in a similar fashion :-

```
define word ID3
   address letter = "(FOR- "
   address width = 6
   field width      = 12
end define
```

```
define word ID4
   address letter = "(PART NAME- "
   address width = 12
   field width      = 6
end define
```

```
define word ID5
   address letter = "(PROGRAMMER- "
   address width = 13
   field width      = 3
end define
```

Add the new words defined above in to the option, following on from **ID2** and the other additional words.
Also add them to the ***word order*** after **ID2**.
Modify the ***define block tape start*** as follows:-

```
define block tape start
    "%"
    ID Jobname ( or ID PartID)   ## PartID will start at 1st Character
    ID2 PartID[ 7 ] ; EM =C      ## PartID[ 7 ] starts at the 7th Character
    ID3 PartID[33] ; EM =C       ## PartID[33] starts at the 33rd Character
    ID4 PartID[46] ; EM =C       ## PartID[46] starts at the 46th Character
    ID5 PartID[53] ; EM =C       ## PartID[53] starts at the 53rd Character
    N ; xy plane ; incremental data ; absolute data ; G6 54
    N ; tool number 99
 end define
```

Which should produce something like the following :-

```
%
O5222
(PROGRAM MACHINES- PORT WING STRUT OF B52***)
(FOR- BOEING INC**)
(PART NAME- 325A59)
(PROGRAMMER- AM*)
```

We are getting there!!

(Review this section Programme data 3)

(Back to Programme Date)

4/ Tape Start Data (Datum From Point)

The next thing to tackle is **(CNC MILL LOCATION= X, Y, Z)** which we are treating as the **Datum Point** for this exercise.

This is relatively straight forward as all that is needed in this case is to add the following line to the ***define block tape start***
*Note:- that we are inserting a string of text and therefore the quotation marks **" are essential.** It is also worth noting that space(s) can be inserted at the begining of a text string "* **(CNC..** , ***but is not possible at the end*** **..ON ="** ;

```
define block tape start
    "%"
    D Jobname ( or ID PartID)
    ID2 PartID[7] ; EM =C
    ID3 PartID[33] ; EM =C
    ID4 PartID[46] ; EM =C
    ID5 PartID[53] ; EM =C
   "(CNC MILL LOCATION =" ; X FromX ; Y FromY ; Z FromZ ; ")"
    N ; xy plane ; incremental data ; absolute data ; G6 54
    N ; tool number 99
 end define
```

The " **X FromX** etc "., will output the **Tool From**, or **Datum**, co-ordinates.

With this addition the output should look something like this :-

```
%
O5222
(PROGRAM MACHINES- PORT WING STRUT OF B52***)
(FOR- BOEING INC--)
(PART NAME- 325A59)
(PROGRAMMER- AM*)
(CNC MILL LOCATION =X0Y0Z.5)
```

(Review this section Tool Datum)

(Back to Quotes)

5/ Tape Start (Prog.Date Input)

Our final requirement is some form of date record.

Here we have to compromise as we cannot provide both of the inputs required :-
(DATE LAST RAN:)
(DATE LAST EDITED:)

The best we can do is one line, such as :- (DATE COMPILED :- 10 / 01 / 99) which will be the date the programme is Post Processed. NOTE :- ***European*** date format is DD / MM / YY , ***American*** date format is MM / DD / YY.

We will now require some more words defined, and formated (if required) to accommodate this.
Add the new words on after the previous additions, and the formating before, or after, the other formating blocks.

```
define word %M
   address letter   = "(DATE
 COMPILED :-"
    address width  = 18
    field width       = 2
 end define
```

```
define word %D
    address letter  = " / "
    address width  = 3
    field width        = 2
 end define
```

```
define word %Y
    address letter  = " / "
    address width  = 3
    field width        = 2
 end define
```

As the formationg is included in the definitions there is no need for a ***define format block***
Add the new words to the word order list:-

```
word order = ( + ID2 ID3 ID4 ID5 %D %M %Y TD TN TD1 TR )
```

Next we modify the define block tape start as follows :- (***American*** Format)

```
define block tape start
    "%"
    .... etc.
    "(CNC MILL LOCATION =" ; X FromX ; Y FromY ; Z FromZ ; ")"
    %M Month ; %D Day ; %Y Year ; ")"
    N ; xy plane ; incremental data ; absolute data ; G6 54
    N ; tool number 99
end define
```

Resulting in the following output :-

```
%
O5222
... etc.
(CNC MILL LOCATION =X0Y0Z5)
(DATE COMPILED :- 06 / 01 / 99)
```

To output the ***European*** Format transpose the following :-.

Change the **define word %M** to **%D** and **%D** to **%M** and do the same for the ***define block tape start***.

%D Day ; %M Month ; %Y Year ; ")"

Resulting in the following output :-

```
%
O5222
(PROGRAM MACHINES- PORT WING STRUT OF B52***)
(FOR- BOEING INC--)
(PART NAME- 325A59)
(PROGRAMMER- AM*)
(CNC MILL LOCATION =X0Y0Z.5)
(DATE COMPILED :- 01 / 06 / 99)
G17G20G90G54
T99
T1
M6
S2800M3 (TOOL 1 Dia. .5 Tip Rad. .02)
```

This now completes all that we can achieve with the message, and tool data content of this example.

(Review this section Prog Date)

Other Types of Text Handling

Another example text requirement in a tape output is given below, where " (Double Quotes) are needed :-

```
N24 G99 [GEOTRA TRANS 1 (R0,R0,[VDH])]
N25 G99 [TEXT 1 ( "LET OP --->" )]
N26 G99 [TEXT 2 ( "***CONTROLE VDH***=",VDH)]
N27 M00
```

This means of output could only be achieved on **UNIX** up until version DP1330 but the following method can now be used on the NT platform :-

```
N ; " G99 [GEOTRA TRANS 1 (R0,R0,[VDH])]"
N ; " G99 [TEXT 1 (\"LET OP --->\")]"
N ; " G99 [TEXT 2 (\"***CONTROLE VDH***=\",VDH)]"
N ; opt stop ( M1 0 )
```

If **Double Quotes** are required in ad address letter then a \\ (double backslash) is needed :-

```
define word ID4
   address letter =  " ( \\" DOUBLE QUOTES "
```

(Back to :- Top)

Option File Construction so far :-

```
machine fanuc
#2
#   History
#   Option for North American Support
#   am 15/12/98  Ref.No. 0280  Issue 1.1
#   Prog.Number,Text handling,Tool data,and Date added.
#   am 06/01/99  Issue 1.2  (Up dated 01/02/99)
#
#9
  define word TN
    address letter  = " (TOOL "
    address width  = 7
    field width       = 2
  end define
#
  define word TD
    address letter  = " Dia. "
    address width  = 6
    field width      = 6
    scale factor    = 2
  end define
#
  define word TR
    address letter  = " Tip Rad. "
    address width  = 10
    field width       = 5
  end define
#
  define word ID2
    address letter  = "(PROGRAM MACHINES- "
    address width   = 19
    field width     = 25
  end define
#
  define word ID3
    address letter  = "(FOR- "
    address width   = 6
    field width     = 12
  end define
#
  define word ID4
    address letter  = "(PART NAME- "
    address width   = 12
    field width     = 6
  end define
#
  define word ID5
    address letter  = "(PROGRAMMER- "
    address width   = 13
    field width     = 3
  end define
#
  define word %M
    address letter  = "(DATE COMPILED :-"
    address width   = 18
    field width     = 2
  end define
#
  define word %D
    address letter  = " / "
    address width   = 3
    field width     = 2
  end define
#
  define word %Y
    address letter  = " / "
    address width   = 3
    field width     = 2
  end define
#
 define format ( TD  TR )
   metric formats
   decimal point    = true
   decimal places   = 3
   leading zeros    = false
   trailing zeros   = false
   imperial formats = metric formats
 end define
#
  define format ( N )
    not permanent
  end define
#
  define format ( ID )
   address letter = "O"
  end define
#
  define format ( M1 M2 )
    field width  = 4
  end define
#
  define format ( F )
    imperial formats
    decimal point  = true
    decimal places = 1
  end define
#
  word order = ( + ID2 ID3 ID4 ID5 %M %D %Y TN TD TR )
#
  define codes
    spin coolant on cw    =  M1 13
    spin coolant on ccw  =  M1 14
    spin coolant off        =  M1 05
    coolant on mist         =  M1 07
    coolant on                =  M1 08
    coolant on flood        =  M1 57
    coolant off                =  M1 09
  end define
#
  message output = false
  integer 12         = 0
  units                  = input
  use progid         = true
  block order        = true
#
 define block tape start
   "%"
   ID JobName
   ID2 PartID[7] ; EM =C
   ID3 PartID[33] ; EM =C
   ID4 PartID[46] ; EM =C
   ID5 PartID[53] ; EM =C
   "(CNC MILL LOCATION =" ; X FromX ; Y FromY ; Z FromZ ; ")"
  %M Month ; %D Day ; %Y Year ; ")"
   N ; xy plane ; compensation off ; absolute data ; G6 54
   N ; tool number 99
 end define
#
 define block tool change first
 end define
#
 define block tool change
   N ; tool number ToolNumber
   N ; change tool
   N ; spindle ToolSpeed[ToolNum] ; spindle on cw ; TN  ToolNumber ;
       TD ToolRadius[ToolNum]  ; TR TipRadius[ToolNum] ; EM =C
 end define
#
 define block move rapid
   N ; rapid ; G2 ; G3 ; G5 ; G6 ; x coord ; y coord ; z coord ; tool length ; M1 ; M2
 end define
```

```
#
  define block move linear
    N ; linear ; G2 ; x coord ; y coord ; z coord ; tool radius ; feedrate ; M1 ; M2
  end define
#
  define block tape end
    N   ; tool number 99
    N   ; change tool
    N   ; end of tape
    "%"
  end define
#
end
```

The final thing left to do in this option is to provide coolant handling which will be **Lesson 3** (Coolant Handling).

(Back to:- Lesson 1(Basic Start) , Tool Info , Tape Start (Prog.Num) , Tape Start (Prog.Data 2) , Tape Start (Prog Data 3) , Tape Start (Datum Location) , Tape Start(Prog.Date) , Text Data (Quotes) , Option , TOP ,)

Option File Construction Example.

Lesson 3

(Up-dated 10/10/2001)

Introduction

Following on from **Lesson 1** (Basic Start) , and **Lesson 2** (Message data), we will now look at coolant control, the relevant parts of the tape file are shown below :-

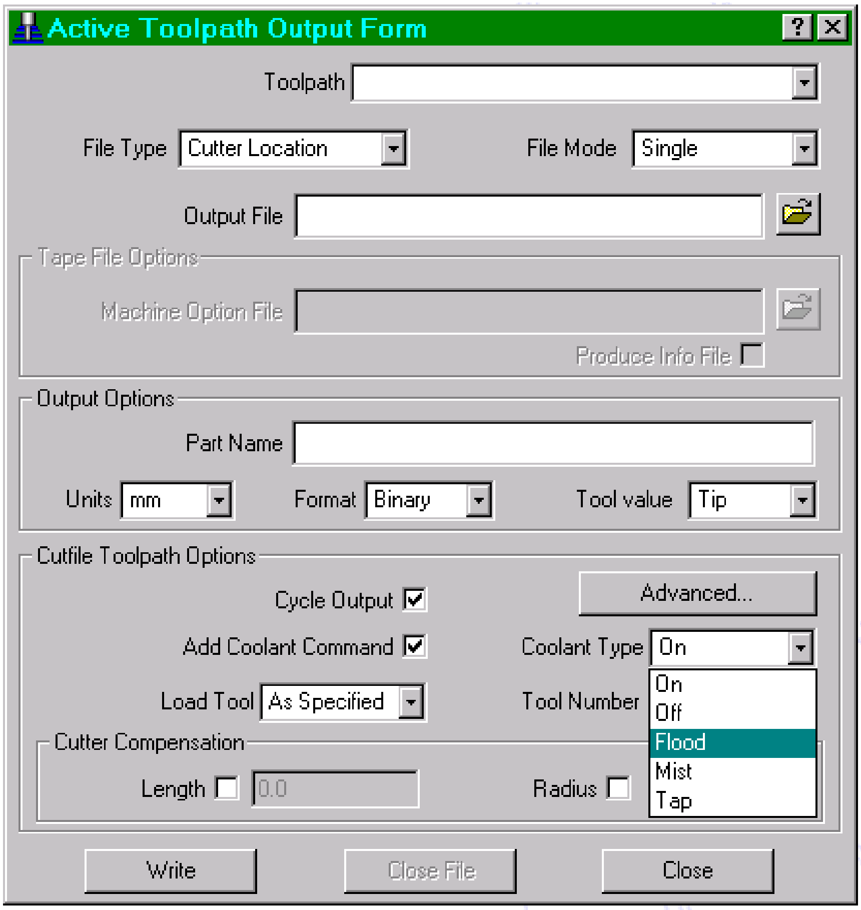
```
T1
M6
S2400M3(TOOL 1 IS A 1/2" HOG BALL)
G0X0Y0
G43Z3.H1
M8              ## Coolant call
..............
..............
T2
M6
S2400M3(TOOL 2 IS A 1/4" CARB BALL)
G0X0Y0
G43Z3.H1
M8            ## Coolant call
..............
..............
G0Z.5
G1Z0F30.
T99
M6
M2
%
```

Coolant handling is dependant on the input provided, and the preferred format required by the customer, so the example below will cover the basics of ensuring the ***ON*** and ***OFF*** are dealt with sensibly. (It will be noted in the above example there is no **M9**, Coolant **OFF** call, prior to the tool change, or tape ending)

The four types of coolant control provided by DUCT / PowerMILL are as follows :-

DUCT	PowerMILL
APT COOLANT ON	ON
APT COOLANT FLOOD	FLOOD
APT COOLANT MIST	MIST
APT COOLANT TAP	TAP
APT COOLANT OFF	OFF

(PowerMILL is via the " **Active Tool Path Output Form - Add Coolant Command** ", the default is ***OFF***, that is greyed out.



Active Toolpath Output Form

Toolpath

File Type: Cutter Location — File Mode: Single

Output File

Tape File Options

Machine Option File

Produce Info File □

Output Options

Part Name

Units: mm — Format: Binary — Tool value: Tip

Cutfile Toolpath Options

Cycle Output ☑

Add Coolant Command ☑ — Coolant Type: On

Load Tool: As Specified — Tool Number

On
Off
Flood
Mist
Tap

Cutter Compensation

Length □ 0.0 — Radius □

One thing to bear in mind with PowerMILL is there are no means of switching coolant **OFF** in multiple tool change files, only at the end of producing a cut file will a coolant ***OFF*** be initiated.) [DUCT can provide ***ON / OFF*** for each tool change segment.]

Machine coolant codes

The standard ISO codes used by the majority of machine tools are :-

ON - M08
OFF - M09
MIST - M07
with FLOOD / TAP usually - M08.

However, there are variations which can usually be handled in Ductpost by re-defining the coolant codes in an option.
(Note :- The leading " **0** " is optional)

(Review this section Introduction)

(Back to Default Opt , SpinCool Opt)

Ductpost default Coolant Structure

If we examine our *[fanuc].dmp* file the following sections refer to the coolant parts.

```
define word M1
   address letter = "M"
end define

define word M2
   address letter = "M"
end define
```

```
 define format ( / G6 S T M1 M2 L P D E H O )
    address width       =   1
    field width          =   2
    exponent width     =   0
    scale factor          =   1
    tape position        =   0
    print position        =   1
    sign                     = none
    not permanent
    not modal
    metric formats
    leading zeros         = false
    trailing zeros          = true
    decimal point         = false
    decimal places       =   0
    imperial formats
    leading zeros         = false
    trailing zeros          = true
    decimal point         = false
    decimal places       =   0
end define
```

```
define codes
   ..............
   spin coolant on cw        =  M1 13
   spin coolant on ccw      =  M1 14
   spin coolant off            =  M1 5
   coolant on mist             =  M1 7
   coolant on                    =  M1 8
   coolant on flood            =  M1 8
   coolant off                    =  M1 9
   ..............
end define
```

```
coolant output    =   ( 2  1 ) - OR  -
integer   6         =  2
integer   7         =  1
```

```
spindle output  =  ( 1  1  1  1 )- OR  -
integer 10        = 1
integer 11        = 1
integer 12        = 1
integer 13        = 1
```

(See Array Data)

The word letter is **M1 or M2**, with the defined handling code in this instance being **M1**.
The way the coolant call is treated is controlled by the " **coolant output** " and " **spindle output** " array, **or** " ***integers 6, 7***, and ***12, 13*** ".
(The **array** format was shown in Ductposts up to version **DP1100**, and **integer** values thereafter. Both methods are acceptable as definitions in an option file, use one or the other.)

To test the coolant function it is advised you have a cut file that incorporates all the relevant modes.

(Review this section Ductpost Structure)

(Back to Default Opt)

Ductpost Default Test

To identify the output we can make the following code format changes to be used instead of the ISO codes, and the Option file will temporarily be modified by the following insertions :-

(This method can be a used as a means to debug an option in other applications)

```
   define format ( M2 )
      field width =  4
   end define
#
   define codes
      .............
      spin coolant on cw        =  M1 13
      spin coolant on ccw       =  M1 14
      spin coolant off          =  M2 "SPOF"
      coolant on mist           =  M2 "MIST"
      coolant on                =  M2 "ON"
      coolant on flood          =  M2 "FLD"
      coolant off               =  M2 "OFF"
      .............
   end define
```

It is worth noting that in the above we have used **M2** " **string definitions** " instead of integer values, hence the quotes, and increased the field width of **M2** to accommodate them.   Also, **M2** has been used as the operational carrier to avoid overwriting other values of **M1** which " ***string definitions*** " have a tendency to do.

If we run the option as it stands at the last up-date i.e. Issue 1.2 (Lesson 2) with our ***coolant test cut fil***e, the output will probably look something like the following :- (The ***MXX*** is substituting for the [**M0x**] codes)

Default Output :-

```
T1
M6
S1500M3 (TOOL 1 Dia. .394 Tip Rad. .197)
G0X.49Y.4303
G43Z1.752H1
G1F98Z1.748
..................
G0Z1.9488
T2
M6
S1500M3 (TOOL 2 Dia. .394 Tip Rad. .197)
X.49Y.4303
G43Z1.752H2MON  [M08]
................
G0Z1.9488
T4
M6
S1500M3 (TOOL 4 Dia. .394 Tip Rad. .197)
X.49Y.4303
G43Z1.752H4MFLD [M08]
................
G0Z1.9488
T5
M6
S1500M3 (TOOL 5 Dia. .394 Tip Rad. .197)
X.49Y.4303
G43Z1.752H5MMIST [M07]
...............
G0Z1.9488
```

```
MOFF [M09]
T99
M6
M2
%
```

From the above we can see there is **NO** coolant call for Tool 1, coolant ***ON*** **[M08]** for Tool 2, coolant ***FLOOD*** **[M08]** for Tool 4, coolant ***MIST*** **[M07]** for Tool 5, and we have coolant ***OFF*** **[M09]** only at the tape end.

This output is the default for the Ductpost settings as indicated in the DP Structure section, and as such could be acceptable except for the lack of coolant **OFF** calls at the tool change positions. [These probably would be output from a DUCT session]

The ***TAP*** coolant call will be ignored as there is no PM Output and no key definition in Ductpost at this stage..

The coolant ***OFF*** at tool changes could be ignored, as **M06** will act as an ***OFF*** switch, but this is not usually acceptable to customers. So to get an ***OFF*** command at the end of the cutting cycle, and prior to the tool change, the following method can be incorporated in the option file.

(Review this section Default)

(Back to :- SpinCool Opt)

Default Option method

As it is unlikely we would be using " ***text strings*** " for the codes, and because it is not possible to test for a text string the example will revert back to the normal defined codes, except for ***FLOOD*** coolant which will be re-defined as **M57**.

```
        define codes
           ..............
         coolant on flood    =  M1 57
           ..............
        end define
```

We can also remove the " **define format (M2)** " block that was temporarily added.

In the " ***define block tool change*** "only, ***add*** the following :-

```
    define block tool change
       if ( word[29] = 8 or  word[29] = 7 or  word[29] = 57 )
          N ; M1 09
       end if
       N ; tool number ToolNumber
       N ; change tool
       N ; spindle ToolSpeed[ToolNum] ; spindle on cw ; TN ToolNumber ;
          TD  ToolRadius[ToolNum]  ; TR ToolRadius[ToolNum] ; EM =C
    end define
#
```

The ***if (statement)*** allows us to check whether or not a coolant call has been made at the start of the cut path, and if so outputs an ***OFF*** **[M09]** if it has. By this method we do not output an unnecessary *[M09]* if coolant is not called.

word[29] = M1, the **29th** defined word in the ***[control].dmp*** file and it is being used to check the setting of the coolant code.

If (M1 07 ***or*** M1 08 ***or*** M1 57) is ***true*** then output **M09**, if ***not***, then don't.

The result should look something like this :-

```
T1
M6
S1500M3 (TOOL 1 Dia. .394 Tip Rad. .197)
G0X.49Y.4303
G43Z1.752H1            ## No Coolant call
G1F98Z1.748
..................
G0Z1.9488              ## No coolant OFF
T4
M6
S1500M3 (TOOL 4 Dia. .394 Tip Rad. .197)
X.49Y.4303
G43Z1.752H4M57   ## Coolant Flood call
................
G0Z1.9488
M9                     ## Coolant OFF
T5
M6
S1500M3 (TOOL 5 Dia. .394 Tip Rad. .197)
X.49Y.4303
G43Z1.752H5M7    ## Coolant Mist call
...............
G0Z1.9488
M9                     ## Tape end Coolant OFF
T99
M6
M2
%
```

This would probably be sufficient to cover the majority of cases except where a customer wishes to have the coolant incorporated with the **Spindle ON** call. i.e. **M13**. (**M14** if the spindle is reversed)

The following example deals with this case.

(Review this section Default Option)

Spindle Coolant Option method

To accomplish this requirement it will be necessary to make some changes to the structure of the option, for as it stands the spindle output is forced to output **M3** at all times in the tool change block.

The first addition is to insert the following two Arrays, or the corresponding integer values, for Coolant and Spindle output. Both are acceptable, though only use one, or the other, format in the option.
The default settings in the in-built ***dump*** file are :-

```
integer  6  = 2
integer  7  = 1
integer 10 = 2
integer 11 = 1
integer 12 = 1
integer 13 = 0
```

OR alternatively

```
coolant output = ( 2 1 )
spindle output = ( 2 1 1 0 )
```

Insert beneath the ***block order = true*** entry.

Next, modify the ***define block tool change*** by deleting the " **spindle on cw** " and subsitute **M1** , leaving the following :-

```
   N ; spindle ToolSpeed[ToolNum] ; M1 ; TN ToolNumber ;
       TD ToolRadius[ToolNum] ; TR TipRadius[ToolNum] ; EM =C
```

If we save and run this now we will find that the **M3** is missing, and nothing has changed as regards the coolant output.

To get this to work at all we will need to make some distinct changes to the construction of the option file and these are set out below:-

Change :- ***define codes***

```
spin coolant on cw   =  M1 13
spin coolant on ccw =  M1 14
spin coolant off      =  M2 05
coolant on mist       =  M2 07
coolant on              =  M2 08
coolant on flood    =  M2 57
coolant off             =  M2 09
```

Change :-   Arrays (or alternative integer values)

```
coolant output =  ( 2  0 )
spindle output =  ( 1  1  1  1  )
```

Change :-   ***define block tool change***

by changing the ***if (statement)*** to

```
if ( word[30] = 8 or word[30] = 7 or word[30] = 57 or word[29] = 13 )
```

by removing **S ToolSpeed[ToolNum] ; M1 ;** from

```
N ; TN ToolNumber ; TD ToolRadius[ToolNum] ; TR TipRadius[ToolNum] ; EM =C
```

Change :- ***define block move rapid***
by adding **spindle (S) ;**

```
N ; rapid ; G2 ; G3 ; G5 ; G6 ; x coord ; y coord ; z coord ; spindle ; M1 ; M2
```

Which should produce an output each time a coolant call is made as follows :-

```
T2 M6
G54 G90 X12.446 Y10.931 S1500 M13
G43 Z44.5 H2
G1 Z44.4
```

It is not **100 %** foolproof and occasionally something set elsewhere in the option can throw it out.

(Review this section SpinCool Opt)

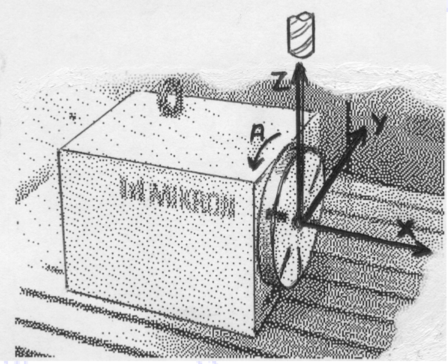
For further tutorials refer to **Option Construction** in the header bar.

(Back to:- DuctPost Structure , Default Test , Default Option , Top ,)
(Back to:- **Lesson 1** (Basic Start), **Lesson 2** (Message data))

4th Rotary axis Examples

(Up-dated 10/10/2001)

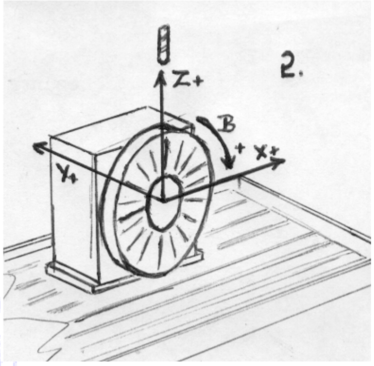
1/ Fourth rotary table axis :- (A. rotating about X.)



```
define format ( A )
   decimal point  = true
   decimal places = 3
end define

word order = ( + A )

define keys
    azimuth axis     not used
    elevation axis    = A
end define

spindle elevation rotation  = false   ## Default
elevation units                    = degrees
elevation axis direction      = positive

azimuth axis parameters  = ( 0. 0. 0.  0. 0. 0. )
elevation axis parameters = ( 0. 0. 0.  1. 0. 0. )

define block move rapid
   N ; rapid ; G2 ; G3 ; G6 ; x coord ; y coord ; z coord ; elevation axis ; tool length ; M1 ; M2
end define

define block move linear
   N ; linear ; G2 ; x coord ; y coord ; z coord ; elevation axis ; tool radius ; feedrate ; M1 ; M2
end define
```

(Return to Top)

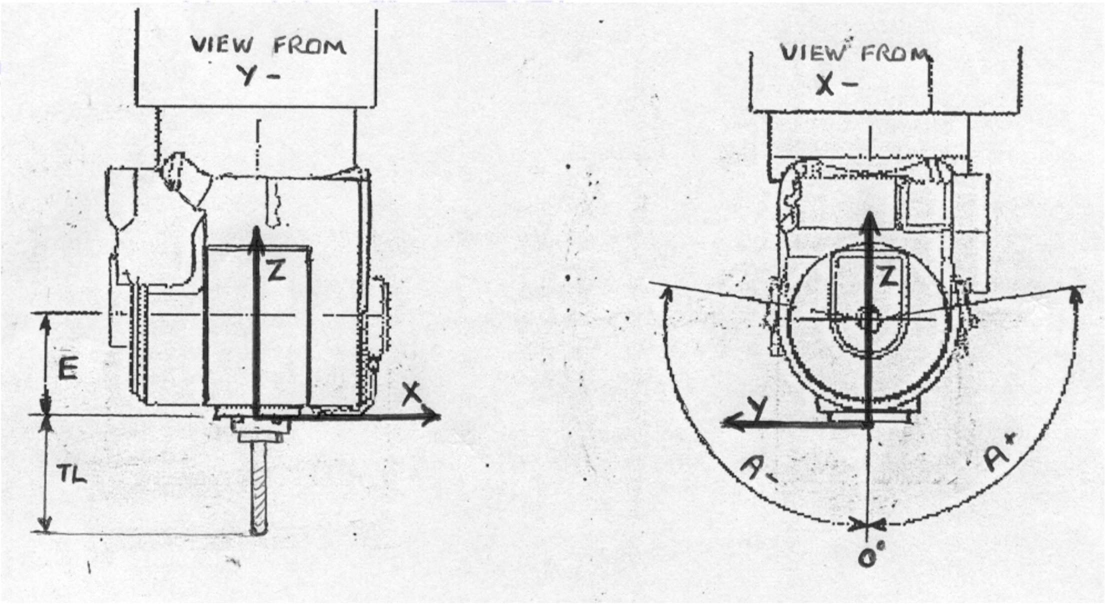
2/ Fourth rotary table axis :- (B. rotating about Y.)



```
define format ( B )
  decimal point  = true
  decimal places = 3
end define

word order = ( + B )

define keys
   azimuth axis    not used
   elevation axis   = B
end define

spindle elevation rotation = false   ## Default
elevation units                  = degrees
elevation axis direction      = positive

azimuth axis parameters  = ( 0. 0. 0.  0. 0. 0. )
elevation axis parameters = ( 0. 0. 0.  0. 1. 0. )

define block move rapid
   N ; rapid ; G2 ; G3 ; G6 ; x coord ; y coord ; z coord ; elevation axis ; tool length ; M1 ; M2
end define

define block move linear
   N ; linear ; G2 ; x coord ; y coord ; z coord ; elevation axis ; tool radius ; feedrate ; M1 ; M2
end define
```

(Return to Top)

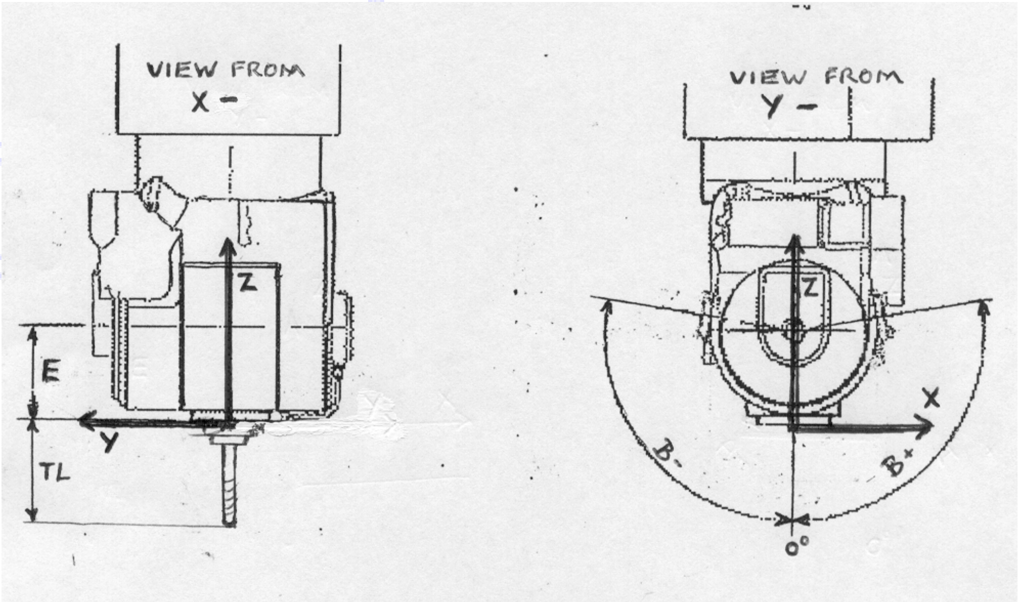
3/   Fourth rotary Spindle axis :- (A. rotating about X.)



```
define format ( A )
   decimal point  = true
   decimal places = 3
end define

word order = ( + A )

define keys
    azimuth axis    not used
    elevation axis   = A
end define

spindle elevation rotation = true   ## Default false
elevation units                    = degrees
elevation axis direction      = positive

elevation centre               =  (  0.   0.   180.5  )     ##  E distance
azimuth axis parameters   =  ( 0  0  0  0  0  0  )
elevation axis parameters =  ( 0  0  0  1  0  0  )

define block move rapid
    N ; rapid ; G2 ; G3 ; G6 ; x coord ; y coord ; z coord ; elevation axis ; tool length ; M1 ; M2
end define

define block move linear
    N ; linear ; G2 ; x coord ; y coord ; z coord ; elevation axis ; tool radius ; feedrate ; M1 ; M2
end define
```

(Return to Top)

4/ Fourth rotary Spindle axis :- (B. rotating about Y.)

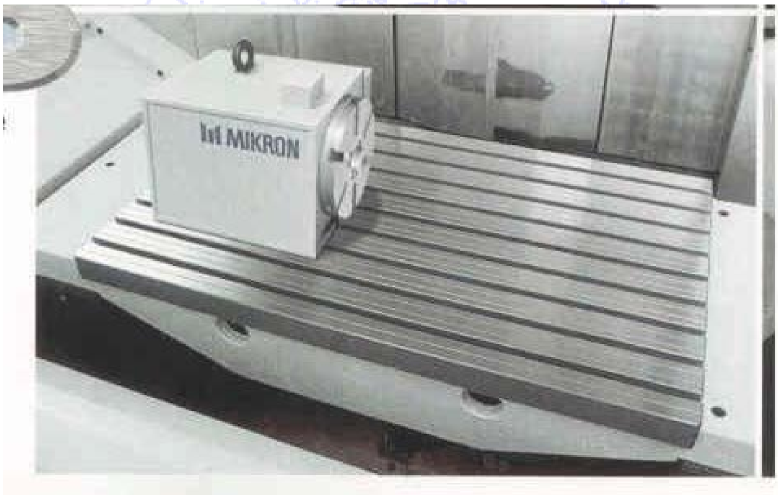


```
define format ( B )
    decimal point  = true
    decimal places = 3
 end define

 word order = ( + B )

 define keys
     azimuth axis    not used
     elevation axis    = B
end define

 spindle elevation rotation = true   ## Default false
 elevation units                    = degrees
 elevation axis direction      = positive

 elevation centre               =  (  0.   0.   180.5  )
 azimuth axis parameters   = ( 0  0  0   0  0  0  )
 elevation axis parameters = ( 0  0  0   0  1  0  )

 define block move rapid
     N ; rapid ; G2 ; G3 ; G6 ; x coord ; y coord ; z coord ; elevation axis ; tool length ; M1 ; M2
 end define

 define block move linear
     N ; linear ; G2 ; x coord ; y coord ; z coord ; elevation axis ; tool radius ; feedrate ; M1 ; M2
 end define
```

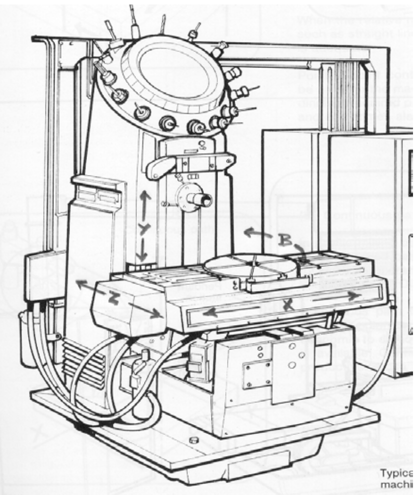
(Return to Top)

Multi axes (4th Rotary axis)

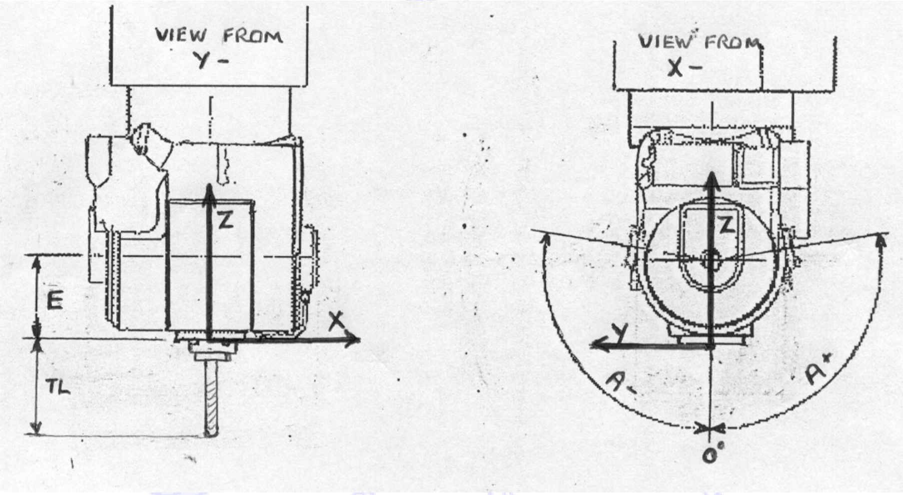
1/ 4th axis Table rotating about X axis.



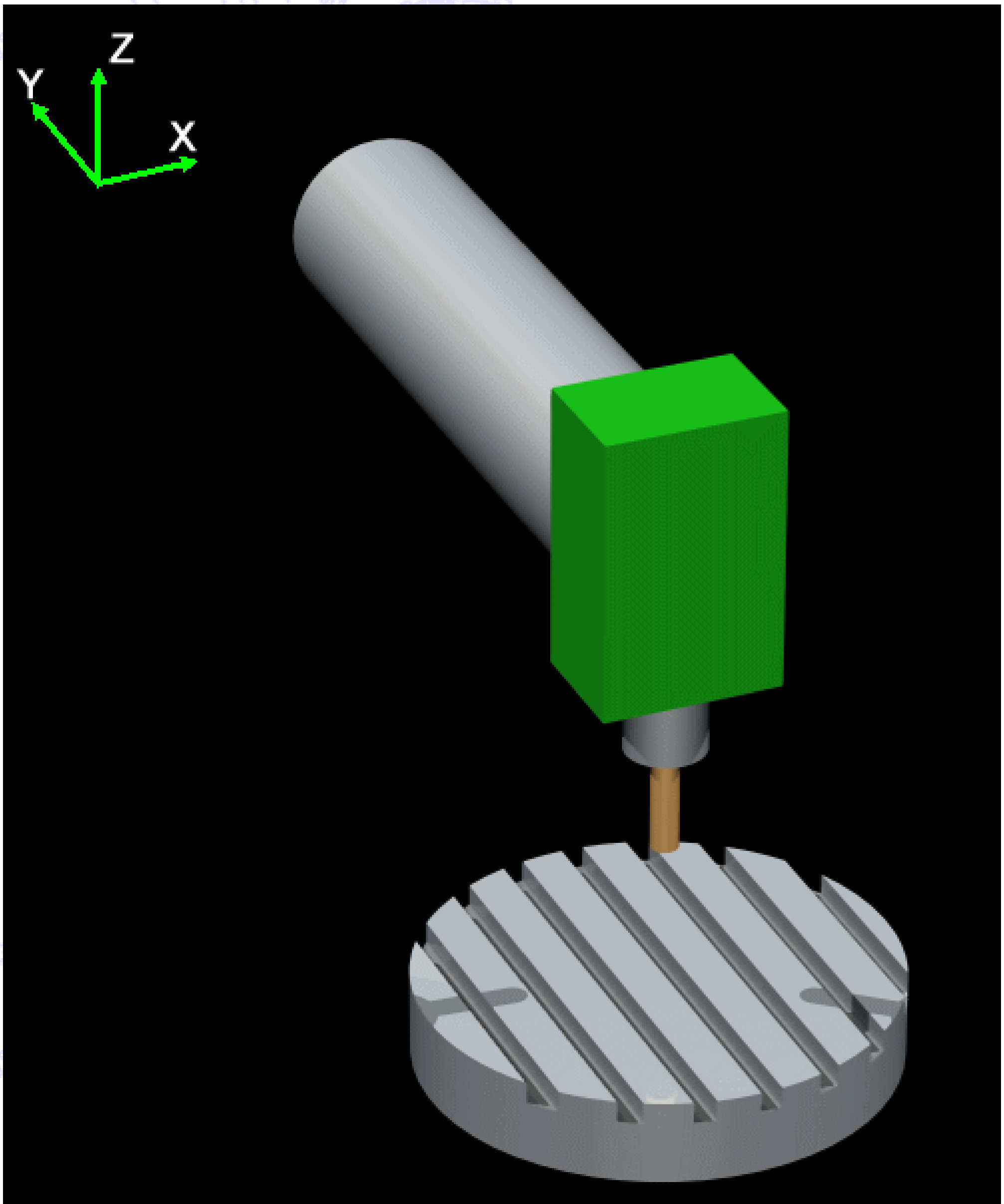
2/ 4th axis Table rotating about Y axis.



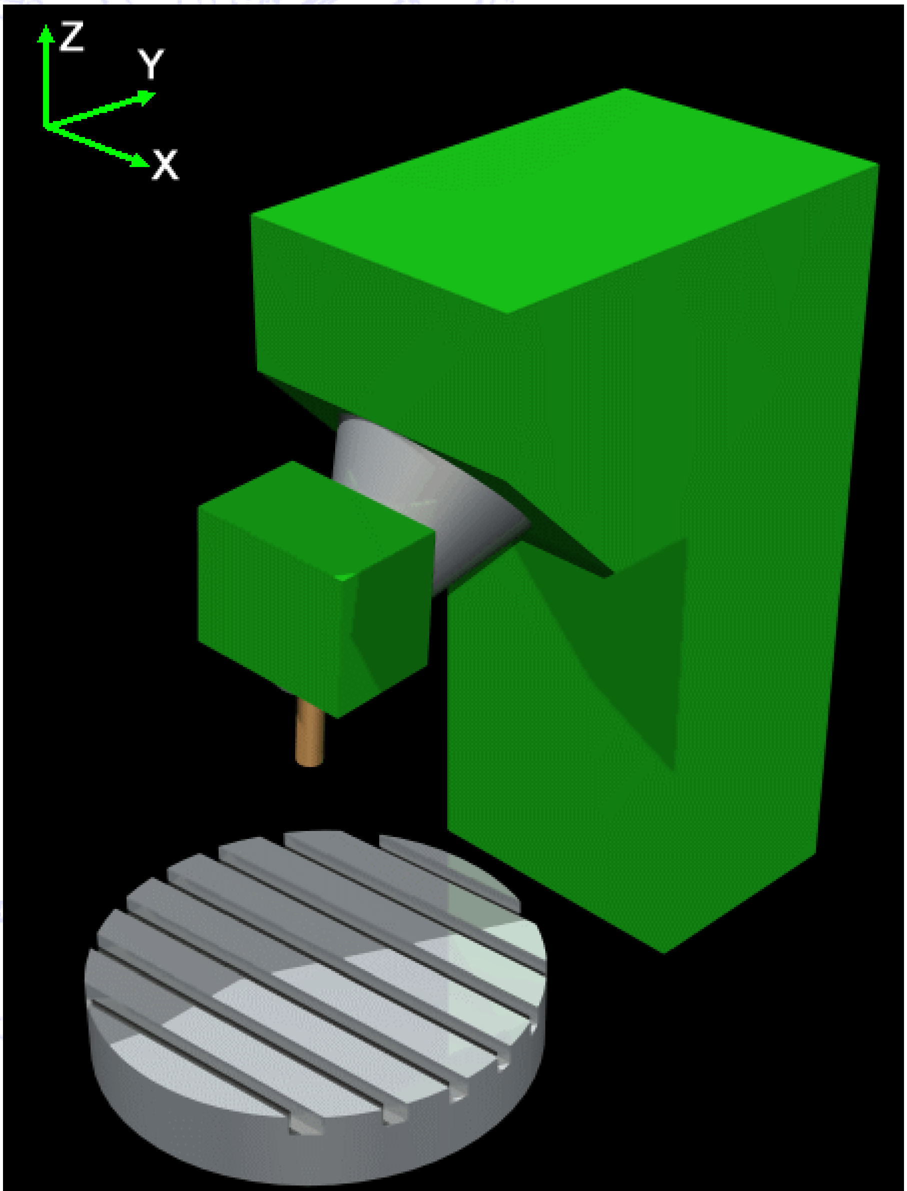
3/ 4th axis Head rotating about X, or Y, axis



back



Compound 45º rotary table



Compound (45º) Swivel Head + Rotary Table

Rotary 5 axes Examples

(Up Dated 24/02/2006 – Still under Constuction)

The examples below are typical **Minimum** definitions required for multi-axes machines, some of the formating may not be needed if it is already defined in the source code, so it pays to check to avoid uneccessary addition to the option file.

5 axes machines may well require more informational input than is actually given below depending on the type of control and capabilities of the control functions, some points on this are given in the notes at the end of this section.

1/ Five axes table unit

Azimuth Tilt table **(A)** rotating about **X** axis, **Elevation** table **(C)** rotating about **Z axis**, when **(A)** ***zero*** is aligned with **Z** axis

```
    define word M5
       address letter    = “M”
       address width   = 1
       field width        = 2
       modal
    end define

    define word M6
       address letter    = “M”
       address width   = 1
       field width        = 2
       modal
    end define

   define format ( A B C )     ## A is not specifically defined correctly in the source code.
       field width        = 8
       modal
       metric formats
       decimal point    = true
        decimal places  = 3
       trailing zeros     = false
       leading zeros      = false
       imperial formats
       decimal point    = true
        decimal places  = 4
       trailing zeros     = false
       leading zeros      = false
    end define

   word order  =  ( +  A  C  )

   define keys
        azimuth axis     = A
        elevation axis   = C
    end define
   define codes
#        elevation clamping codes           ## These are dealt with elsewhere ( See Clamping )
       clamp on              =  M5  11
```

```
        clamp off             =  M5  12
#       azimuth clamping codes
        azimuth clamp on  =  M6 21
        azimuth clamp off  =  M6 22
    end define

    spindle azimuth rotation     =  false
     azimuth axis units              =  degrees
     azimuth axis direction        =  positive
     azimuth axis parameters    =  ( 0.0  0.0  0.0  1  0  0  )      ## RTC zero. See notes-1 below  ( Type A )
  or
     azimuth axis parameters    =  ( 0.0  0.0  55.0  1  0  0 )     ## RTC   55mm above zero. See notes below  ( Type B )
   or
     azimuth axis parameters    =  ( 0.0  0.0 -55.0  1  0  0 )     ## RTC -55mm below zero. See notes below  ( Type C )

    spindle elevation rotation  =  false
    elevation axis units          = degrees
    elevation axis direction     =  positive
     elevation axis parameters  =  ( 0.0  0.0  0.0  0  0  1 )

    linear axis limits          =  ( -1750.0  1200.0  -875.0  875.0  -250.0  250.0  )
     rotary axis limits          =  (-95.0  35.0  -3600.0  3600.0    0.01     1  )

    initial tool vector                      =  ( 0   0  1 )        ## Spindle Alignment with Z axis.   default
    workplane angles                        =  none     ##  See notes-6 below
    multiaxis coordinate transform  =  true
    linearise multiaxis moves            =  true
     retract at angular limit               =  true    ## Default  =  false
    withdrawal amount                     = 100.0
     integer 3                                  = 1

    define block move rapid
        N ; rapid ; G6 ; x coord ; y coord ; z coord ; azimuth axis ; elevation axis ; S ; H ; M1 ; M2
     end define

    define block move linear
      N ; linear ; G2 ; x coord ; y coord ; z coord ; azimuth axis ; elevation axis ; tool radius ; feedrate ; M1 ; M2
    end define
```

## This is the minimum requirement, it could well require additional embellishment to meet certain functions.

## It is not essential to have the rotary axes in the Rapid block as it isn't used for multi axis working.

NOTE : For Heidenhain use G1 in place of " rapid " and " linear ", leave out G2, and us RR instead of " tool radius "

(Return to TOP)

2/ Five axes combination head and table

Swivel tool spindle **(B)** rotating about **Y**, table **C** rotating about **Z** when **B** is **zero** (***vertical***)

```
    define format ( A B C  )
        field width         = 8
        modal
        metric formats
        decimal point    = true
         decimal places  = 3
        trailing zeros    = false
        leading zeros     = false
        imperial formats
        decimal point    = true
         decimal places  = 4
        trailing zeros    = false
        leading zeros     = false
     end define

    word order = ( + B C  )

    define keys
        azimuth axis   = B
        elevation axis = C
     end define

    spindle azimuth rotation    =  true
     azimuth axis units          =  degrees
     azimuth axis direction      =  positive
     azimuth centre              =  ( 0.0   0.0  185.56  )       ## See notes-2 below
     azimuth axis parameters    =  ( 0.0  0.0  0.0   0  1  0 )

    spindle elevation rotation =  false
    elevation axis units          = degrees
    elevation axis direction     =  positive
    elevation centre              =  ( 0.0  0.0  0.0  )
     elevation axis parameters =  ( 0.0  0.0  0.0   0  0  1 )

    linear axis limits         =  ( -1750 0  1200.0  -875.0  875.0  -250.0  250.0  )
     rotary axis limits         =  ( -110.0   110.0  -3600.0  3600.0    0.01      1  )

    initial tool vector                   =  ( 0   0  1 )        ## Spindle Alignment with Z axis.   default
    workplane angles                      =  none     ##  See notes-6 below
    multiaxis coordinate transform  =  true
     linearise multiaxis moves          =  true
    minimise multiaxis retraction      =  true
    retract at angular limit              =  true    ## Default  = false
     withdrawal amount                  = 100.0
     integer 3                          = 1

    define block move rapid
        N ; G1 ; G6 ; x coord ; y coord ; z coord ; azimuth axis ; elevation axis ; S ; H ; M1 ; M2
     end define


    define block move linear
       N ; G1 ; G2 ; x coord ; y coord ; z coord ; azimuth axis ; elevation axis ; tool radius ; feedrate ; M1 ; M2
    end define
```

## This is the minimum requirement, it could well require additional embellishment to meet certain functions.

## It is not essential to have the rotary axes in the Rapid block as it isn't used for multi axis working.

NOTE : For Heidenhain use G1 in place of " rapid " and " linear ", leave out G2, and us RR instead of " tool radius "

(Return to TOP)

3/ Five axes Spindle Head

Rotary head (C) rotating about Z, Swivle tool (B) rotating about Y, when (C) is **zero** aligned with Y axis

```
define format ( A B C  )
    field width        = 8
    modal
    metric formats
    decimal point    = true
     decimal places  = 3
    trailing zeros     = false
    leading zeros      = false
    imperial formats
    decimal point    = true
    decimal places  = 4
    trailing zeros     = false
    leading zeros      = false
 end define

word order  =  ( +  B C  )

define keys
    azimuth axis   = C
    elevation axis = B
 end define

spindle azimuth rotation    =  true
 azimuth axis units            =  degrees
 azimuth axis direction       =  positive
 azimuth centre                  =  ( 0.0  0.0  0.0  )
 azimuth axis parameters   =  ( 0.0  0.0  0.0   0  0  1 )

spindle elevation rotation  =  true
elevation axis units           =  degrees
elevation axis direction     =  positive
elevation centre                =  ( 0.0  0.0  193.25 )       ## See notes-3 below
 elevation axis parameters =  ( 0.0  0.0  0.0   0  1  0 )

linear axis limits          = ( -1750 0  1200.0  -875.0  875.0  -250.0  250.0  )
 rotary axis limits         = ( -200.0  320.0  -100.0  100.0    0.1      1  )

initial tool vector                    =  ( 0   0  1 )        ## Spindle Alignment with Z axis.   default
workplane angles                       =  none      ##  See notes-6  below
multiaxis coordinate transform  =  true
 linearise multiaxis moves           =  true  or false
 retract at angular limit              =  true    ## Default  = false
minimise multiaxis retraction      =  true
withdrawal amount                      = 100.0
 integer 3                                  = 1

define block move rapid
    N ; rapid ; G6 ; x coord ; y coord ; z coord ; azimuth axis ; elevation axis ; S ; H ; M1 ; M2
 end define
```

```
define block move linear
   N ; linear ; G2 ; x coord ; y coord ; z coord ; azimuth axis ; elevation axis ; tool radius ; feedrate ; M1 ; M2
end define
```

## This is the minimum requirement, it could well require additional embellishment to meet certain functions.

## It is not essential to have the rotary axes in the Rapid block as it isn't used for multi axis working.

NOTE : For Heidenhain use G1 in place of " rapid " and " linear ", leave out G2, and us RR instead of " tool radius "

(Return to TOP)

4/ Five axes 45deg, Swivel Head / Rotary Table (DMG - DMU*P Series) [see DMG Animation]

Swivel head **(B)** rotating 45 degrees about **ZY** axes, Elevation Table **(C)** rotating about **Z** when **(B)** is **zero (*Head Vertical*).**

(These machines usually have Heidenhain controls - with Siemens or MillPLUS as an alternative)

```
define format (  B C  )
    field width       = 8
    modal
    metric formats
    decimal point    = true
     decimal places  = 3
    trailing zeros    = false
    leading zeros     = false
    imperial formats
    decimal point    = true
    decimal places  = 4
    trailing zeros    = false
    leading zeros     = false
 end define

word order  =  ( +  B C  )

define keys
    azimuth axis   = B
    elevation axis = C
 end define

spindle azimuth rotation   =  true
 azimuth axis units           =  degrees
 azimuth axis direction      =  positive
 azimuth axis parameters  =  ( 0.0  0.0  0.0   0  1  1 )    ##  NOTE two axes and signs.
 azimuth centre                  =  ( 0.0  0.0  0.0  )
#       azimuth centre                =  ( 0.0  0.0  329.22  )        ## See notes-4 below

spindle elevation rotation  =  false
elevation axis units          =  degrees
elevation axis direction     =  positive
elevation axis parameters =  ( 0.0  0.0  0.0   0  0  1 )

linear axis limits         =  ( -710 0   710.0  -600.0  600.0  -325.0  325.0  )
 rotary axis limits         =  (   0.0  180.0  -3600.0  3600.0    0.1      1  )

initial tool vector                    =  ( 0   0  1 )        ## Spindle Alignment with Z axis.   default
workplane angles                  =  none      ##  See notes-6  below
```

```
 multiaxis coordinate transform  =  true
  linearise multiaxis moves        =  true  or  false
  retract at angular limit          =  true
  withdrawal amount                = 100.0
  integer 3                      = 1

 define block move rapid
    N ; G1 ; G6 ; x coord ; y coord ; z coord ; azimuth axis ; elevation axis ; S ; H ; M1 ; M2
  end define
                    ##  It is not essential to have the rotary axes in the Rapid block.
 define block move linear
   N ; G1 ;  x coord ; y coord ; z coord ; azimuth axis ; elevation axis ; RR ; feedrate ; M1 ; M2
 end define
```

## This is the minimum requirement, it could well require additional embellishment to meet certain functions.

## It is not essential to have the rotary axes in the Rapid block as it isn't used for multi axis working.

(Return to TOP)

5/ Five axes 45deg, Swivel & Rotary Tables / Vertical Spindle (DMG - DMU*V Series) [see DMG Animation]

Swivel **Azimuth Table (B)** rotating 45 degrees about **ZY** axes, Rotary **Elevation Table (C)** rotating about **Z** axis, when **B** is **zero** (***Table Horizontal***).

(These machines usually have Heidenhain controls - with Siemens or MillPLUS as an alternative)

```
 word order  =  ( +  B C  )

 define keys
      azimuth axis     =  B
      elevation axis   =  C
  end define

 spindle azimuth rotation     =  false
  azimuth axis units           =  degrees
  azimuth axis direction        =  positive
  azimuth axis param           =  ( 0.0   0.0   0.0    0   1   -1  )   ## NOTE two axes and signs.
  azimuth axis param           =  ( 0.0   0.0   159.325    0   1   -1  )    ## MillPLUS

 spindle elevation rotation   =  false
 elevation axis units          =  degrees
 elevation axis direction      =  positive
 elevation axis param          =  ( 0.0   0.0   0.0    0   0   1  )  ## See notes-5 below

 linear axis limits        =  ( -500.0   500.0  -500.0  500.0   -325.0   325.0 )
  rotary axis limits        =  (  -7.0  180.0   -7200.0   7200.0    0.1   1  )

 initial tool vector                  =  ( 0   0  1 )       ## Spindle Alignment with Z axis.   default
workplane angles                    =  none     ##  See notes-6  below
 multiaxis coordinate transform  =  false
  linearise multiaxis moves        =  false
  retract at angular limit          =  true       ## Default  = false
  withdrawal amount                = 100.0
  integer 3                      = 1

 define block move rapid
```

```
    N ; G1 ; G6 ; x coord ; y coord ; z coord ; azimuth axis ; elevation axis ; S ; H ; M1 ; M2
  end define

  define block move linear
    ………………………..
    ………………………..
    if ( Toolpath5axis )
      if ( swb )
        N ; G1 ;  M3 128  ;  " ;  Heidenhain RTCP ON CODE"
        N ; G1 ;  M4 126  ;  " ;  Heidenhain TAKE Shortest Rotation Path"
        unset swb
      end if
      N ; G1 ;  x coord ; y coord ; z coord ; azimuth axis ; elevation axis ; RR ; feedrate ; M1 ; M2
    end if
  end define
```

## This is the minimum requirement, it could well require additional embellishment to meet certain functions.

## It is not essential to have the rotary axes in the Rapid block as it isn't used for multi axis working.

(Return to TOP)

NOTES

These notes cover various points that are indicated in the above examples, but are by no means exhaustive as each type of 5 axes needs to be considered on its own merits, and careful consideration give to the particular control requirements and customer operating preferences in the way it is constructed.

1a/ In **Type A**, **NO** ***centre of rotation offset*** is defined, therefore the Machine datum X0 Y0 Z0 is defined at the centre of the **Elevation** table surface, **zero** aligns with the **Azimuth** tilt table **X** (or **Y**) axis ***centre line of rotation***.

1b/ In **Type B**, the Machine datum X0 Y0 Z0 is defined at the centre of the **Elevation** table surface, and the table surface lies 55.0 mm. **below** the centre line of the **Azimuth** tilt table rotation then the offset must be accounted for as shown.

1c/ In **Type C**, the Machine datum X0 Y0 Z0 is defined at the centre of the **Elevation** table surface, and the table surface lies 55.0 mm. **above** the centre line of the **Azimuth** tilt table rotation then the offset must be accounted for as shown.

The following parameters must also be set as :-
multiaxis coordinate transform = **true**
linearise multiaxis moves = **true** or **false**

IF the machine tool control has the capability of looking after its own **RTCP calculations**, called by an appropriate **G** or **M** or (Siemens **TRAORI**) code definition, that must be inserted prior to, or with, the first azimuth angular move.

Then the parameters will be defined as, and **NO** offsets will be needed :-
multiaxis coordinate transform = **false**
linearise multiaxis moves = **true** or **false**

(See examples Type A ; Type B ; Type C - Back to 1)

2/ In this example a ***centre of rotation offset*** **is defined**, and as it is a Spindle **Azimuth** Swiveling Tool the offset is defined in the " ***azimuth centre*** parameter " **Z axis** alignment as a POSITIVE value. The offset is the distance between the ***centre of rotation*** of the Swivel Head and the face of the the tool holder (**E** - see example - Back to 2)

Note:- TL is the tool length and is usually not taken into account in the option configuration.

3/ In this example a ***centre of rotation offset* is defined**, and as it is a Spindle **Elevation** Swiveling Tool the offset is defined in the " ***elevation centre*** parameter " **Z axis** alignment as a POSITIVE value. The offset is the distance between the ***centre of rotation*** of the Swivel Head and the face of the the tool holder (**E** - see example - Back to 3)

Note:- TL is the tool length and is usually not taken into account in the option configuration.

4/ In this example we have a **Swivel Spindle Head** that rotates between **0** and **180** degrees about the **ZY** axes (**0** vertical, to **180** horizontal, with continuous movement), and a **Rotary Elevation Table** that rotates about the **Z** axis when the Swivel Head is Vertically ***zero.***

NOTE :- DMG have a similar **P series** that has a Swivel Head that **Indexes** from Vertical to Horizontal, and has a trunion table with ***Tilting and Rotary Tables,*** it is **not** configured as shown here, and requires ***TWO*** options.

If the machine is fitted with either Heidenhain, or Siemens, control it is unusual to have to define any offset, as ***RTCP*** is normally standard. The example is defined for **RTCP** operation!

However, the **MillPlus** control is the exception and an offset **has** to be defined.

The ***offset*** is the mean of the two distances between the vertical and horizontal head positions, which are obtained from the machine installation setup sheet, and the result placed in the ***azimuth centre*** parameter **Z** axis, also " ***multi axis linearisation*** " will be set to **true**!

For other machines with a similar configuration the above will probably apply.

(Back to 4)

5/ In this example we have a **Swivel Table** that rotates between **0** and **180** degrees about the **ZY** axes (**0** horizontal, to **180** vertical, with continuous movement), and a **Rotary Elevation Table** that rotates about the **Z** axis when the Swivel Table is Horizontally ***zero.***

If the machine is fitted with either Heidenhain, or Siemens, control it is unusual to have to define any offset, as ***RTCP*** is normally standard. The example is defined for **RTCP** operation!

However, the **MillPlus** control is the exception and an offset **has** to be defined.

The ***offset*** is the mean of the two distances between the vertical and horizontal table positions, which are obtained from the machine installation setup sheet, and the result placed in the ***azimuth axis parameters*** **Z** axis. (i.e. azimuth axis parameters = (0.0 0.0 **154.997** 0 1 -1), **not** the elevation centre, and " ***multi axis linearisation*** " will be set to **true**!

For other machines with a similar configuration the above will probably apply.

NOTE :- The DMG DMU50eV-linear the Table rotation is about **51 degrees** and the settings for this are as follows :-

```
        azimuth axis param = ( 0.0 0.0 0.0  0.584825  0.573576  -0.573576 )
```

(Back to 5)

6/ " **workplane angles =** " has three settings that control the function of the post processor in outputting Datum Shift and Workplane Orientation requirements and their special Formats depending upon the Type of Control.

" **none** " This setting will not output any Datum Shift or Workplane Orientation even if defined.

" **machine tool** " This is used where specific controls only require two Angular Definitions to define a Workplane Rotation.

(Post Configuration is defined as WorkplaneAz and WorkplaneEl NOT Azpos, Elpos)

" **apparent** " This is used where specific controls use Spatial Angular Definitions to define a Workplane Rotation.

(Post Configuration is defined as WorkplaneA ; WorkplaneB ; WorkplaneC)

(Back to 1 ; 2 ; 3 ; 4 ; 5)

Configuration Parameters

```
AZIMUTH AXIS    Head/Head     always the rotation        TRUE
AZIMUTH AXIS    Head/Table    always the Head rotation   TRUE
AZIMUTH AXIS    Table/Table   always the TABLE ROCK      FALSE

    spindle azimuth rotation   = false          false == Table   true == Head
    azimuth axis units         = degrees        Always
    azimuth axis direction     = positive       ALWAYS
```

```
         azimuth centre              = ( 0.0 0.0 0.0 )  For Head Offsets and none RTCP
         azimuth axis param          = ( 0.0 0.0 0.0  0  0  1 ) Axes Alignment in Z
                       ( For a 45 degree ZY Table =  0  1 -1 )
                       ( For a 45 degree ZY Head  =  0  1  1 )
#
     ELEVATION AXIS    Head/Head     always the SWIVEL          TRUE
     ELEVATION AXIS    Head/Table    always the TABLE ROTATION  FALSE
     ELEVATION AXIS    Table/Table   always the TABLE ROTATION  FALSE

         spindle elevation rotation = false          false == Table  true == Head
         elevation axis units        = degrees        Always
         elevation axis direction    = positive       ALWAYS
         elevation centre            = ( 0.0 0.0 0.0 )  For Head Offsets and none RTCP
         elevation axis param        = ( 0.0 0.0 0.0  0  0  1 )   Rotation of Table/Swivel of
 Head
                              [To reverse the rotations, use -1 ]
#
         pcs origin                  = ( 0.0  0.0  0.0  0  0  0 ) DO NOT USE

                              Min/Max X travel  Min/Max Y travel  Min/Max Z travel
         linear axis limits = ( -500.0 500.0     -420.0  420.0     -380.0  380.0 )
                                    ( NOT CRITICAL but helpful )
#
                         ( Min/Max Azimuth - Min/Max Elevation - Tolerance - obsolete always 1
  )
         rotary axis limits = ( -0.0  180.0   -360000.0  360000.0    0.01   1  )

                                 X  Y  Z
         initial tool vector = ( 0  0  1 )  Spindle Alignment
         workplane angles    = apparent     Relates to RTCP control
                                         ## Eular Angles, Three angles of rotation
                                               WorkplaneA, WorkplaneB, WorkplaneC

                             = machine tool Machine controls that only work with two angular
 rotations
                                                WorkplaneAZ, WorkplaneEl

                             = none         NO workplane orientation available

         multiaxis coordinate transform = false  RTCP ENABLED    = true  RTCP DISABLED
         linearise multiaxis moves      = false  No Smoothing for angular moves  = true
 Smoothing to Tol. above
         retract at angular limit       = true   If a rotary axis hits a set limit in " rotary
 axis limits " there will
                                                  be a retracton of the spindle and the rotary
                                                 axes will reconfigure
                                                  by 180 degrees if possible.

         retract and reconfigure style  = none       ) These define how the retraction act –
 none Split moves
#         retract and reconfigure style  = combine   ) combine   All axes together
#         retract and reconfigure style  = linearise ) linearise add additional points to
 Tolerance

         withdrawal amount              = 100.0  This is the amount the retraction moves away
 from the job
         integer 3                      = 1      Multiaxis enabling code – ALWAYS
```

Examples

1/   HEAD/HEAD with offsets in Z and X

```
                                 X        Y      Z
           azimuth centre      = ( 0.0     0.0   293.25 )
           azimuth axis param  = ( 0.0     0.0     0.0  0  0  1 )
           elevation centre    = ( 274.112 0.0   293.25 )
           elevation axis param = ( 0.0     0.0     0.0  1  0  0 )
```

2/   HEAD/TABLE with offsets in Z Head and Y Table

```
           azimuth centre      = ( 0.0 0.0 125.270 )
           azimuth axis param  = ( 0.0 0.0 125.270  -1  0  0 )
           elevation centre    = ( 0.0 0.0   0.0 )
           elevation axis param = ( 0.0 -64.993 0.0  0  0  1 )
```

3/   TABLE/TABLE with offsets in Z and X Table

```
           azimuth centre      = ( 0.0 0.0 0.0 )
           azimuth axis param  = ( 0.0 0.0 -23.275  -1  0  0 )
           elevation centre    = ( 0.0 0.0   0.0 )
           elevation axis param = ( 24.993 0.0 0.0  0  0  1 )
```

Heidenhain "*Apparent*" Format :-

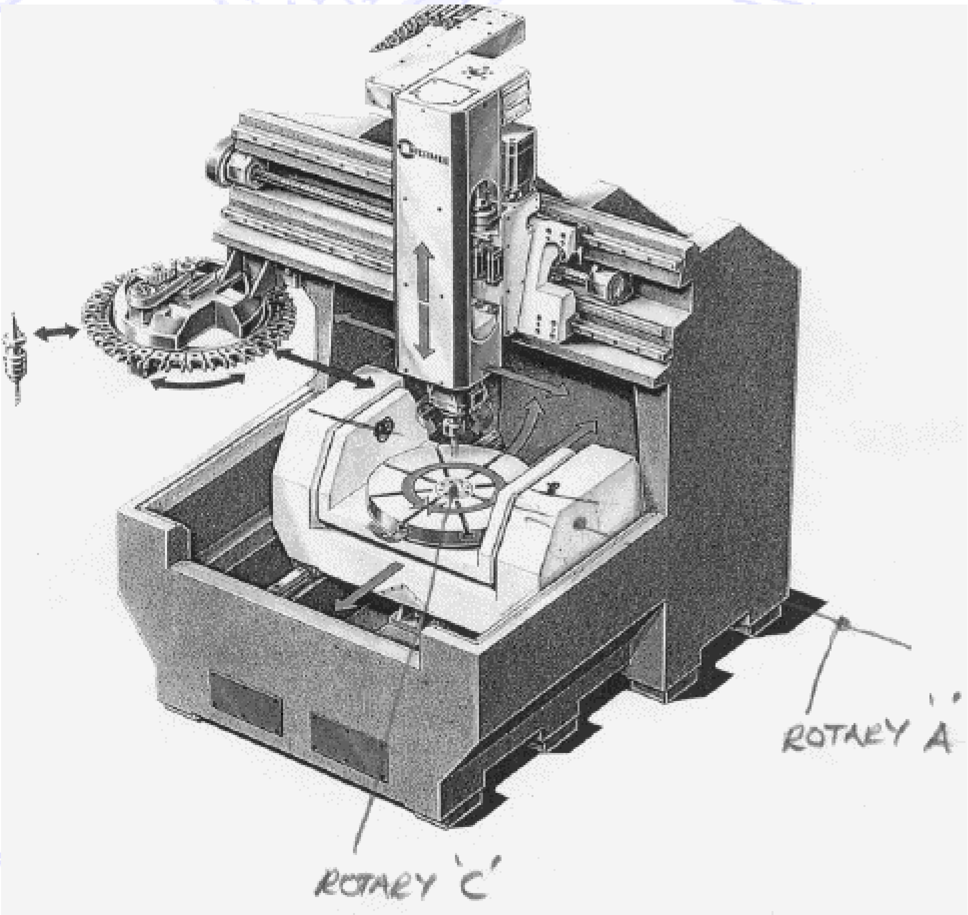
```
    if ( WorkplaneA != 0.0 or WorkplaneB != 0.0 or WorkplaneC != 0.0 )
      N ; G6 190 ; " WORKING PLANE   ; NEW WORKING PLANE ORIENTATION"
      N ; G6 191 ; WPA WorkplaneA =C ; WPB WorkplaneB =C ; WPC WorkplaneC =C
      N ; " L B+Q121 C+Q122" ; FMAX ; M4 126
```

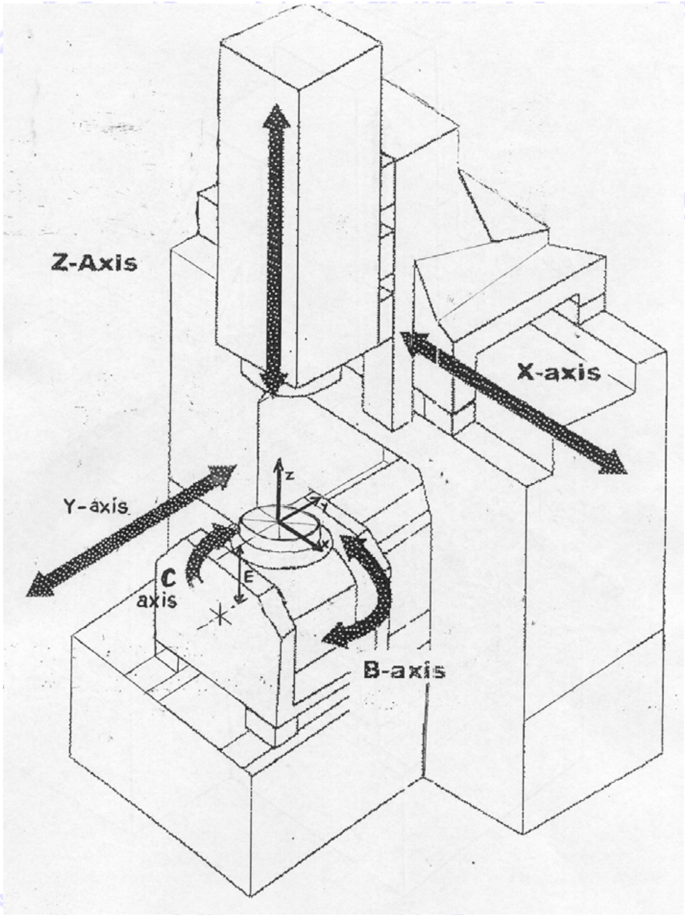
Heidenhain "***Machine Tool***" Format :-

```
    if ( WorkplaneAz != 0.0 or WorkplaneEl != 0.0 )
      N ; G6 190 ; " WORKING PLANE   ; NEW WORKING PLANE"
      N ; G6 191 ; AWK WorkplaneAz =C ; CWK WorkplaneEl =C
      N ; G1 ; AWK =C ; CWK =C ; FMAX ; M4 126
```

Return to (TOP)

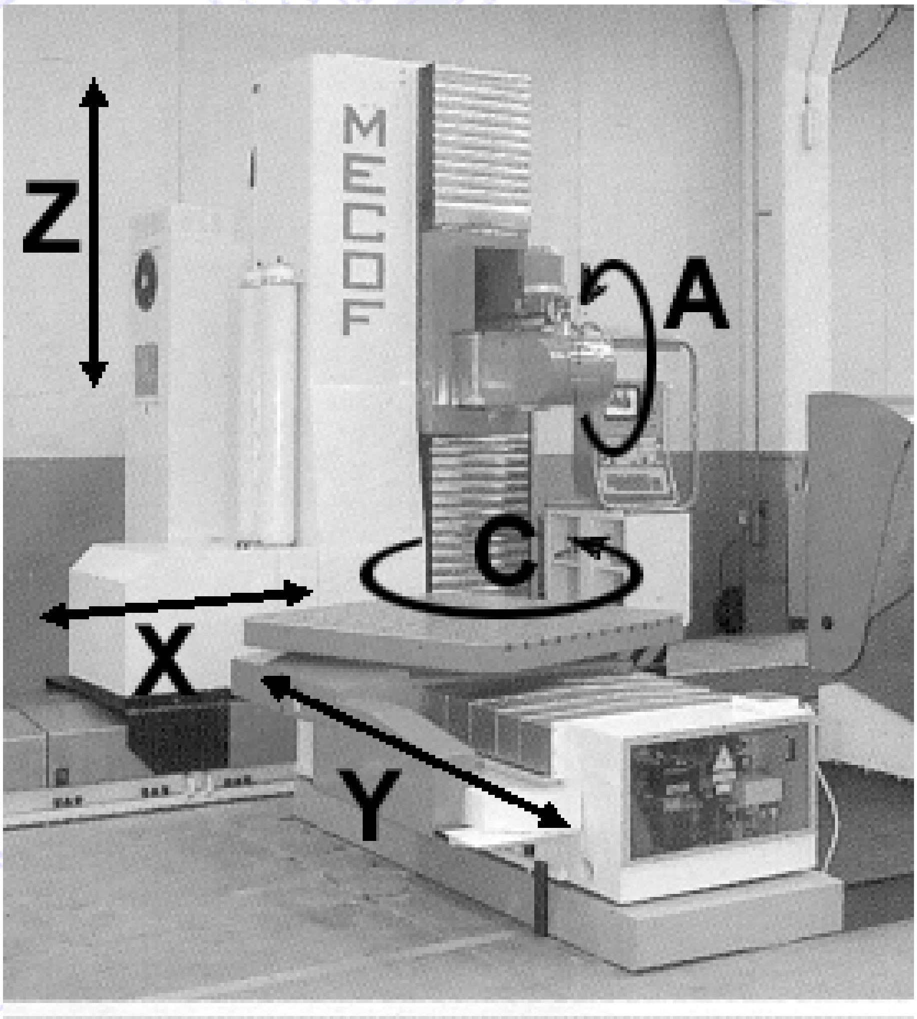
Multi axes (5 axes Table)





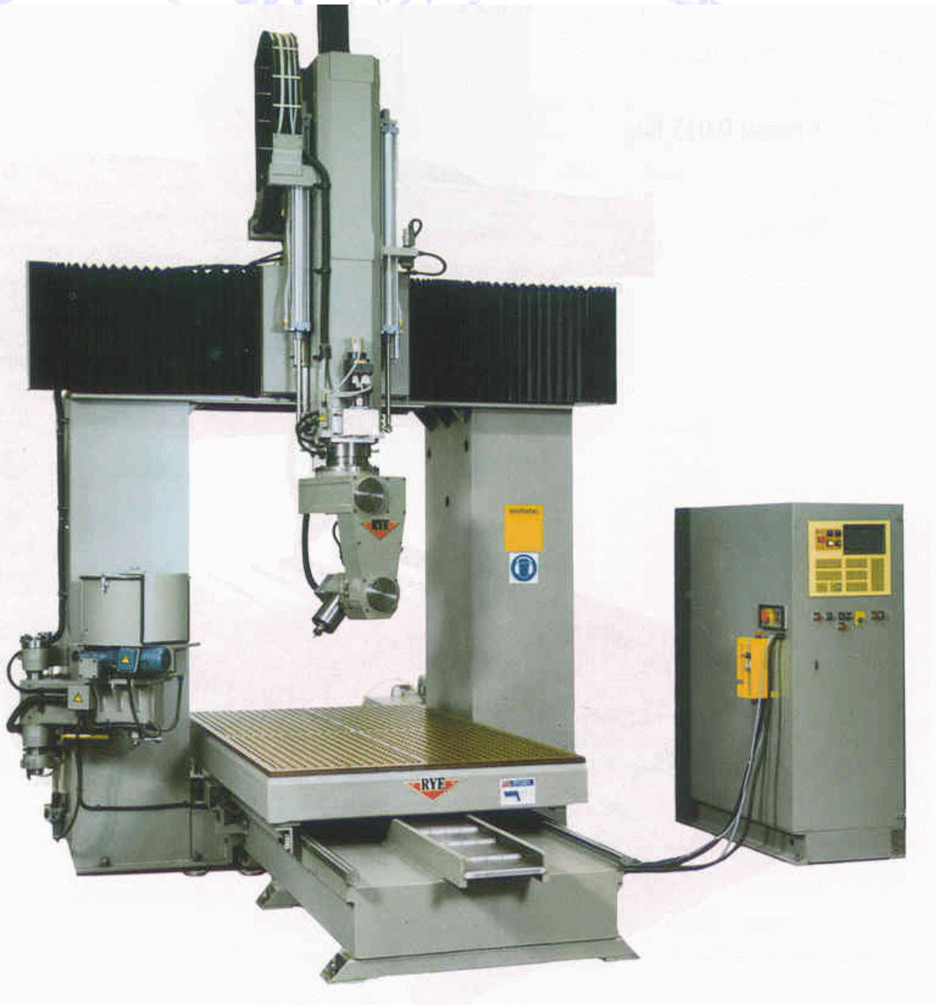
Return

Multi axes (5 axes Head/Table)



Return

Multi axes (5 axes Head)



Return

Frequently asked questions about DUCTpost

(Up-dated 13/08/2002)

Questions raised :-

Q1 After changing the option file the tape file does not change ?

Q2 How do you make DUCTpost prompt for a program name or number ?

Q3 How do you make the nc program split into separate files ?
How do you prevent the nc program splitting into separate files ?

Q4 How do you remove or add block numbers ?

Q5 How do you change the block start number or increment ?

Q6 How do I make S and M come out in a different order ?

Q7 How do you remove or add leading zeros to G and M codes ?

Q8 If I change "X" in a block the position does not seem to update ?

Q9 How do you make the spindle speed and feedrate appear on the tool change line ?

Q10 How do you get the **Skim** feed rate to output as a **Rapid** move ?

Q11 How do you get the Heidenhain tape start units to match the unit input ?

Q12 Please find the attached .opt file which I'm having trouble with. I can't get G17 to show up in the second line.

Q13 Is there a date variable that I can use to output in the start section of a tap file the day/ month/ year, and if there is a time variable how do I access this?

Q14 Why absolute vs incremental NC code for High Speed Machining?

- Is it a necessity for HSM to use incremental? We see incremental post examples on MAZAK and MAKINO.
- Is it effecting surface quality?
- Is it effecting machining time?
- Considering backlash in the drive system , is it diminishing or exaggerating effects of backlash?

Q15 How can I limit the **arc radius** to suit the machine tool **maximum** and **mimimum** limits?

Q16 Is there a way of having a desk top short cut for Ductpost User Guide?

Q17 If I post process a multiple toolpath output using Load Tool ***Automatic*** to suppress same tool tool change, is there a way to include a comment, such as the tool path name, to indicate the new toolpath?

Q18 The Arc rotations are incorrect for **G19** on a **Horizontal** machine configuration with the **Spindle** aligned in the **Y axis** how can I change this?

Q19 What is " **initial tool vector = (0 0 1)** " used for?

Q20 What variables are available to provide data from PowerMill without resorting to the messages output?

Q21 Is it possible to suppress arc output in G18 and G19 planes?

Q22 Can I output my own comments to the tape file?

Q23 I have an option file called heid430.opt and want all my tape files to have the file extension " **.h** " automatically so I don't have to edit each file?

Q24 Can I force the tool length to be zero in the TOOL DEF statement when post processing for the Heidenhain machine.
It worked in PowerMill 3 but doesn't work in PowerMill 4.

Q25 Can I prevent any coolant codes being output to the tape file, even if I have accidentally left the coolant on in PowerMill?

Q26 I have a Roeders machine and want to output a specific function for this control, the format is SM = 0.015 where the value is calculated as 33% of the thickness.

Q27 I am using a Heid option file and parametric feedrates (Q DEFs) but I get lots of lines output that just contain **FQ2** and no XYZ move.

Q28 My machine tool control will only accept angles between -360 and +360
That is to say the C axis requires the output **C25** rather than **C385**. Is it possible to achieve this output from Ductpost.

Q30 How do I get **"** (Quotes) in the NC output?

(Return to TOP)

ANSWERS

Q1. After changing the option file the tape file does not change ?

A. Make sure that the option file you are using is the one that you have changed, it is probable that you made a copy of the original (***At least you should have done***) and made the changes in this, but forgot to exchange the **old for** the ***revised*** option. It is recommended that you **change the extension of the original** to " ***.old*** " or " ***.sav*** " at this point, which will prevent it being used until replaced by the revised option.

DUCTpost will print out the option file name and the location directory when it runs, either in the shell window, or in the command window of PowerMill.

```
                         *************************
                         * DUCTPOST VERSION 1.3 *
                              *     Revision 59      *
                         *************************


          Copyright (c) Delcam plc, Birmingham, England
          1991, 1992, 1993, 1994, 1995, 1996, 1997, 1998 , 1999, 2000

          Licensed to Joe Bloggs Ltd.,

          Supplied by Delcam Plc.

          Using option file . /localhomes/am/DIRECTORIES/opts_cur/mazak/mazak640m.opt   ## Option file used and directory location
          Using built-in    mazak
          Reading cut file records...........
          199 records read
```

DUCTpost will look for a file called **mazak640m.opt** in the **current** directory, then in ***/dcam/config/ductpost*** or as indicated above.
It is advised that the option file should reside in ***/dcam/config/ductpost***

NOTE :- The option file is intended to be independent of the DUCTpost version so an option file in ***/dcam/product/ductpostNNNN/lib*** will not be used.

(Return to TOP)

Q2. How do you make DUCTpost prompt for a program name or number ?

A. Add one of the following lines to the option file :-

```
use progid = false
use partid  = false
```

(The ***first*** will prompt for a number, the ***second*** for a name, but only if run in a shell window. **Not** from within PowerMill which requires a different method. For a better explanation see Option Construction Basic Start Tutorial).

You must also check that the tape start has the variable ***ProgID*** for **program number**, or ***PartID*** for a **program name**

e.g.

```
define block tape start
   "%"
    ID ProgID    or   ID PartID
    rapid ; absolute data ; end if drill
end define
```

See Tape Start or Basic Start Tutorial

(Return to TOP)

Q3. How do you make the nc program split into separate files ?
How do you prevent the nc program splitting into separate files ?

A. By setting either, the real flag " **maximum tape length** ", which will break the overall programme in to small sections of a size set by the value specified. (see below)

The real " **maximum tape length** " is in **feet** and must have a **decimal point** with the length required.

A length of **100.** (feet) = 12000 bytes approximately.
A length of **0.** (zero) = do not split tape. (Usually the default setting for most controls)

or,

the integer flag " **maximum tape blocks = 60000** ", will do the same function, but after the predefined number of 60,000 line blocks, or what ever number is defined.

It is **recommended** that only ***one***, is used, **not both** together.
See also tape splitting

(Return to TOP)

Q4 How do you remove or add block numbers ?

A. Re-format the function of N in the option as follows :-

```
define format ( N )
  not permanent
end define
```

will **remove** existing block numbers

```
define format ( N )
  permanent
end define
```

will ***add block*** numbers to all the blocks where **N** has been defined in the block definition.
(Check the source file and/or the option)

See also block numbers .

(Return to TOP)

Q5. How do you change the block start number or increment ?

A Add the following lines to the option file :-

```
  block start       =  10
  block increment  =  5
```

will give block numbers starting at 10 and increasing in 5's.
e.g. N10..., N15..., N20...., N25....

See also block numbers .

(Return to TOP)

Q6. How do I make S and M come out in a different order ?

A. The simplest way is to copy the word order lines from the source file and change the position of the groups (in this case S and M1)

Copy source

```
word order = ( N G1 G2 G3 X Y Z )
word order = ( + S F T H D M1 )
```

to option file and change

```
word order = ( N G1 G2 G3 X Y Z )
word order = ( +  F T H D M1 S)
```

See also word order.

(Return to TOP)

Q7. How do you remove, or add leading zeros to G and M codes ?

A. Re-format the function of G1 and M1 in the option as follows :-

```
define format ( G1 M1 )
  leading zeros = true
end define
```

will produce G01, G02, G03 and M03, M08 etc.

```
define format ( G1 M1 )
  leading zeros = false
end define
```

will produce G1, G2, G3, M3 M8 etc. (***Normally the default***)

See also word formats.

(Return to TOP)

Q8. If I change "**X**" in a block the position does not seem to update ?

A. If you have a block like this :- (Not exactly a good example)

```
define block linear
  if ( swa )
    linear ; X 100 ; Y 1000 ; Z 10000
  else
    linear ; x coord ; y coord ; z coord
  end if
end define
```

then on the tape you will get **X0.100 Y1.000 Z10.000** (depending on other settings).
However, the **x coordinate** will not be changed internally because the syntax used above uses output via the **word X**.
The ***word "X"*** knows nothing about the **x coordinate**
For the **x coordinate** to update, one must use a **key**, not a word. In such a case the above would better defined as :-

```
define block linear
  if ( swa )
    linear ; x coordinate 100 ; y coordinate 1000 ; z coordinate 10000
  else
    linear ;  x coordinate ; y coordinate ; z coordinate
  end if
end define
```

and the ***internally*** held **x coordinate*****would*** be updated in this case.

(Return to TOP)

Q9. How do you make the spindle speed and feedrate appear on the tool change line

A. Use the variables ToolSpeed and ToolFeed in the tool change block.

```
define block tool change
   N ; tool number ; change tool ; spindle ToolSpeed[ToolNum] ; feedrate ToolFeed[ToolNum]
end define
```

It is likely that other changes will be needed as well.

See also tool change and Basic Start

(Return to TOP)

Q10. How do you get the Skim feed rate to output as a Rapid move ?

A. There are two methods available depending on the version of Ductpost used.

1/ Prior to and including DP1205 :-

```
define block move linear
   if ( feedrate => 3000 )      ## 3000 is the Powermill default setting
      N ; rapid  ; x coord ; y coord ; z coord ; M1 ; M2
   else
      N ; linear ; x coord ; y coord ; z coord ; feedrate ; M1 ; M2
   end if
end define
```

2/ From DP1300 :- (The above will also work)

```
define block move linear
   if ( srat => 3000 )      ## 3000 is the Powermill default setting
      N ; rapid  ; x coord ; y coord ; z coord ; M1 ; M2
```

```
     else
        N ; linear ; x coord ; y coord ; z coord ; feedrate ; M1 ; M2
    end if
  end define
```

If a feed rate is required for the Rapid move then the following line will need to be substituted

```
        N ; rapid ; x coord ; y coord ; z coord ; feedrate ; M1 ; M2
```

See also Skim Feeds

(Return to TOP)

Q11 How do you get the Heidenhain tape start units to match the unit input ?

A The following example shows how to output MM or INCH depending on the tape input.

```
    block order    =  true
      units            =  input
#
   define codes
    imperial data          =  G4 70
    metric data            =  G4 71
   end define
#
  define block tape start
     if ( TapeUnits = 71 )          ##    Checks for units defined in input toolpath.
       N ; " BEGIN PGM" ; ID JobName ; " MM"
    else
       N ; " BEGIN PGM" ; ID JobName ; " INCH"
    end if
  end define
#
  define block tape end
     if ( TapeUnits = 71 )
       N ; " END PGM" ; ID =C ; " MM"
    else
       N ; " END PGM" ; ID =C ; " INCH"
    end if
  end define
```

The items in **blue** are essential changes from the normal source format!

(Return to TOP)

Q12 Please find the attached .opt file which I'm having trouble with. I can't get **G17** to show up in the second line.

A The information provided is shown below :-

Data supplied as to requirement	**Option file as provided**
START G0 G40 G54 G80 G90 X.876	machine fanuc15m units = input message output = false

```
Y1.2391
S5000 M3
```

It should be :

```
START

G0 G17 G40
G54 G80
G90 X.876
Y1.2391
S5000 M3
```

```
define format (F)
  modal
  imperial formats
  decimal places = 1
  decimal point   = true
  leading zeros   = false
  trailing zeros    = false
end define

define block tape start
  "START"
end define

define block tool change first
end define

define block tool change
end define

define block tool change clear
    N  ; G1 0 ; xy plane ; G3 54 ; comp off ;
         G5 80 ; G6 90  ; G1 =C ;
end define

define block tape end
   N  ; G1 0  ; G1 =C ; G3 17 ; G4 28 ; G5 91 ;
          Z 0 ; Z =C
   N  ; G2 40 ; G5 90 ; M1 30
  "%"
 end define

end
```

If we look at the option we see that both " ***tool change first*** " and " ***tool change*** " blocks are not used, indicating there is to be **NO tool *change function.*** **However, "** ***tool change clear*** " has been defined with a set of parameters that are to be carried forward to the next line - indicated by the **;** (semi-colon) at the end of the last parameter.

We note that G1 0 ; G3 54 ; G5 80 and G6 90 have set values defined, whereas **xy plane** and **comp off** don't, relying on the pre-set values in the source code.

The problem lies with the **xy plane** and **G3 54**. The xy plane in the source code is **G3 17** which in this case will be **overwritten** by **G3 54** as you can't have two **G3 s in the same line, as the latter will always take preference in output.**

Two points should be borne in mind when constructing option files :-

1/ Stick as closely as possible to the original group coding in the source files. (see the revised option below)
If you do require the output to be in a specific sequence then define the sequence you want in the block and use " **block order = true** " to ensure the output is as you have defined, and not as the source code uses.
(This is probably why the designations in the option have been set as they are, though it is unusual for a machine tool control to be so fussy)

2/ It is recommended that " ***tool change clear*** " is **not used** as this has the function of clearing all the registers previously set and may not be what you had intended.

Revised Option :-

machine fanuc15m

```
      message output    = false
      units                   = input
    block order       = true

  define format ( F )
      modal
      imperial formats
      decimal places      = 1
      decimal point        = true
      leading zeros        = false
      trailing zeros         = false
  end define

  define block tape start
      "START"
  end define

  define block tool change first
  end define

  define block tool change
      N  ; G1 0  =C ; G3 17 =C ; G2 40 ; G6 54 ; G4 80 ; G5 90 ;    ##    ( Old format )
      N ; rapid =C ; xy plane ; end of compensation ; G6 54 ; end of drill ; absolute data ;    (Preferred format)
  end define

  define block tool change clear
  end define

  define block tape end
      N ; G1 0 =C ; G3 17 =C ; G5 91 ; G6 28 ; Z 0 =C    ##    ( Old format )
      N ; rapid =C ; xy plane ;  incremental data ; G6 28 ; z coord 0 =C    (Preferred format)
      N ; end of compensation  =C ; absolute data  ; end of prog
      "%"
  end define

end
```

(Return to TOP)

Q13 Is there a date variable that I can use to output in the start section of a tap file the day/ month/ year, and if there is a time variable how do I access this?

A Not a specific variable but it is possible to build up this format using the variables that are available. The following example of the construction necessary to provide this is given below, along with time :-

```
  define word %D
    address letter   = "/"
    address width  = 1
    field width       = 2
    leading zeros   = true
  end define
#
  define word %M
    address letter   = "(Date "
    address width  = 5
    field width       = 2
    leading zeros   = true
  end define
#
  define word %Y
    address letter   = "/20 "
    address width  = 3
    field width       = 2
    leading zeros  = true
  end define
#
  define word %h
    address letter   = " "
    address width  = 1
    field width       = 2
  end define
#
  define word %m
    address letter   = ":"
    address width  = 1
    field width       = 2
  end define
#
  define word %s
    address letter   = ":"
    address width  = 1
    field width       = 2
  end define
#
  word order = ( + %D %M %Y  %h %m %s  )
#
   block order            = true
#
  define block tape start
     ID JobName ; " ,MX"
    %M Month ; %D Day ; %Y Year ;
            %h hour ; %m Minute ; %s second ; " )"
   end define
#
```

Produces the following output :-

```
%
Oxyzx ,MX
( Date 19-10-2000 15:15:25 )
G40G80G90
M98P4
N1M6T10
```

This is the ***American*** format, MM / DD / YY, if the ***European*** format DD / MM / YY is required then just switch the sequence :-

```
%D Day ; %M Month ; %Y Year
```

Note words %M and %D will need their address letters switched as well.

(Return to TOP)

Q14 Why absolute vs incremental NC code for High Speed Machining?
- Is it a necessity for HSM to use incremental? We see incremental post examples on MAZAK and MAKINO.

- Is it effecting surface quality?
- Is it effecting machining time?
- Considering backlash in the drive system , is it diminishing or exaggerating effects of backlash?

A 1/ Is it a necessity for HSM to use incremental?

A/ Some CNC's do insist on incremental coordinates for high speed work. The Matsuura for example. I think they do this to simplify the calculations that must be done by the CNC for each program block.

Some incremental formats are more compact than their absolute equivalents, which means the CNC may have less data to process.

2/ - Is it effecting surface quality?

A/ In theory, an incremental program is just a different way of encoding the same data as an absolute program. Therefore, the machine should follow exactly the same trajectory, and the finished result should be the same.

However, there may be second order effects that influence the result, specifically rounding errors converting between absolute and incremental positions, and block processing times may affect the actual feed rate achieved by the machine.

Both of these could influence surface quality, but it's not certain whether incremental or absolute would be better.

3/ - Is it effecting machining time?

A/ Only if the block processing time for the CNC is a factor.

4/ - Considering backlash in the drive system , is it diminishing or exaggerating effects of backlash?

A/ This is doubtful. The machine should execute the same moves whether the program is absolute or incremental, and it is the path geometry that will determine the effect of backlash.

Again, if the dynamic performance of the machine is different, then backlash effects will be different too, but this is a third order effect that I would not expect to be significant on a machine in good condition.

Summary :-

Possible Advantages.

1. Less data, which can help with HSM, though this is not always the case.
2. The same program can be run without alteration as a canned cycle, so it's useful for repeated work, e.g. a multi cavity tool path where dimensions are identical.

Possible Disadvantages.

1. Accumulation of rounding errors (DUCTpost includes code to correct this)
2. Difficulty restarting if the program is interrupted in the middle.
3. Difficulty relating actual machine position to coordinates in the program.
4. Some CNC functions are unavailable in incremental modes. (Especially in special High Speed incremental modes. This means it's quite hard to produce a 100% incremental program.)

Views.

1/ Absolute programming is easier to follow, especially in relation to determining position on the job
2/ There is no choice if the HSM programme requires incremental input.
3/ Incremental is probably an advantage if repeat cycles are required.

(Return to TOP)

Q15 How can I limit the arc radius to suit the machine tool **maximum** and **mimimum** limits.

A Some machine tools can only accept a circular arc of a pre-set maximum radius and raise an alarm if this is exceeded in the NC tape input. The same can apply with very small arcs

CLArcFit can often produce very large radii which are difficult to spot when the tape is produced, so in order to prevent this there is a controlling REAL in Ductpost that will break an arc down to straight line moves over a predetermined radius.

By inserting **real 16 = nnnnn.0** will limit the maximum arc output to the value of " n " (***Don't forget the point***)

From **DP1335** ***arc radius limit*** = nnnnn.0 will be the new name for ***real 16***
the ***default value*** set at ***10000.0 mm*** this will need to be re-defined as ***390.0inch*** for **Imperial** input

PowerMill4.0 can generate very small arcs under certain tool path strategies that some machines will object to.

From DP1358 these can be suppressed by the use of ***arc minimum radius*** = 0.nnn
the ***default value*** setting is **0.0** - re-define this to the machine limit (e.g. **0.016** mm or **0.001** inch)

(Return to TOP)

Q16 Is there a way of having a desk top short cut for Ductpost?

A To place a short cut to Ductpost on your DeskTop carry out the following procedure :-
Go to ***/dcam/product/ductpost1335/file/help/master.html*** click and drag to desktop screen area.
Select the **html** desktop icon and right click to bring up the dialog box, then select ***properties***.
In the Properties box select ***Change Icon***, and in the new ***Icon Box*** browse for the ***Ductpost icon*** in ***/dcam/product/ductpost1335/file/help/gifs/ductpost.ico*** (see Ductpost Short Cut Screen dialog)

(Return to TOP)

Q17 If I post process a multiple toolpath output using Load Tool ***Automatic*** to suppress same tool tool change, is there a way to include a comment, such as the tool path name, to indicate the new toolpath?

A To do this you will need to include in the option file something along these lines :-

```
define word TPN
   address letter  = "( Tool Path :- "
   address width = 15
   field width      = 20
end define

word order    = ( + TPN )

define block cldat 1094
   N ; TPN  ToolPathName  ; " )"
end define
```

This is probably the simplest way, there are other methods.

(Return to TOP)

Q18 The Arc rotations are incorrect for **G19** on a **Horizontal** machine configuration with the **Spindle** aligned in the **Y axis** how can I change this?

A The following method can be employed for any control fitted to a machine tool configuration as indicated. (NOTE :- This may not apply to all machines)
The only thing to be careful of is that the word[?] numbers correspond to the actual word list placement in the source dump, especially if it is a NEW word added in the option. (See Word List) All New words are added to the list in the order they appear in the option.

The example is for a Heidenhain control.

```
 define keys
    x coordinate = Z    ## Note all coordinates changed
    y coordinate = X
    z coordinate = Y
    key i           = K
    key j           = I
    key k           = J
 end define

 define codes
   circle cw    = DR 2   ## Originally  DR "-" ( If statements can not check "strings" )
   circle ccw  = DR 3   ## Originally   DR "+"
   xy plane     = G3 18  ## Originally  not used
   zy plane     = G3 19  ## Originally  not used
   xz plane     = G3 17  ## Originally  not used
 end define

 define block move circle
   if ( WORD[5] = 19 )       ## Word[5]  = G3
     if ( WORD[51] = 2 )      ## Word[51] = DR
       N ; circ 3 ; key i ; key j ; key k
       N ; G2  3 ; x coord ; y coord ; z coord ; DR "+" ; RR =C ; FF =C ; feedrate ; MP =C
     else
       N ; circ 3 ; key i ; key j ; key k
       N ; G2  3 ; x coord ; y coord ; z coord ; DR "-" ; RR =C ; FF =C ; feedrate ; MP =C
     end if
   else
     if ( WORD[51] = 2 )
      N ; circ 3 ; key i ; key j ; key k
      N ; G2  3 ;  x coord ; y coord ; z coord ; DR "-" ; RR =C ; FF =C ; feedrate ; MP =C
    else
      N ; circ 3 ; key i ; key j ; key k
      N ; G2 3 ; x coord ; y coord ; z coord ; DR "+" ; RR =C ; FF =C ; feedrate ; MP =C
    end if
  end if
end define
```

The following is for ISO code other than HeidISO machines

```
 define keys
   x coordinate = Z    ## Note all coordinates changed
   y coordinate = X
   z coordinate = Y
   key i           = K
   key j           = I
   key k           = J
 end define

define codes
  circle cw    =  G1  2
  circle ccw  =  G1  3
  xy plane      = G3 18    ## Originally G3 17
  zy plane      = G3 19
  xz plane      = G3 17    ## Originally G3 18
end define
```

```
define block move circle
    N ; G3
    if ( word[5] = 19 )
      N ; G1 ( 5 - word[3] ) ; x coord ; y coord ; z coord ;        ## Word[3] = G1
            key i ; key j ; key k ; feedrate ; M1 ; M2
    else
      N ; G1 ; x coord ; y coord ; z coord ;
            key i ; key j ; key k ; feedrate ; M1 ; M2
   end if
end define
```

Be aware, there should be no requirement to change the direction of **X** or **Y** by **Scale factor = -1** as this will give a " wrong "; G2/G3 in G19 or G18.;(The axes switch in define keys takes care of this)

(Return to TOP)

Q19 What is " initial tool vector = (0 0 1) used for?

A This function is available from DP1357 and effectively replaces the above requirement by automatically making the necessary axis alignments to the NC output.

It's default setting is as shown above with the Spindle aligned in the **Z** axis. (M35)

If the machine Spindle is aligned in any other axis then the setting is changed accordingly (e.g. M32 **Y** axis (0 **1** 0) or **X** axis (**1** 0 0))

It is theoretically possible to have the Spindle aligned along any angle (e.g. M33 30 deg ZY axes (0 **0.5** **0.866**)

(See example angled heads)

(Return to TOP)

Q20 What variables are available to provide data from PowerMill without resorting to the messages output?

A The following variables can be used to provide basic information relating to general, tool, and tool path data

Calendar Day ; Month ; Year Time Hour ; Minute ; Second
Tool Data : Tool Diameter (ToolRadius[ToolNum] * 2) Tool Radius ToolRadius[ToolNum]

Tool Tip Radius TipRadius[ToolNum] Tool Length ToolLength[ToolNum]
Tool Defined ToolType Tool Identity ToolName

Tool Path data Cutting Tolerance Tolerance Material left Thickness
Name of Tool Path ToolPathName

The recomended holding words for the above, to try and maintain a standard, are as follows : -
Calendar %D ; %M ; %Y Time %h ; %m ; %s
Tool Data Tool Dia. TD Tip Radius TR Tool Length TLH Tool Id. TN Tool Type TT
Tool Path Path Name TPN Thicknes THK Tolerance TOL

```
e.g.   define word TPN                              N ; TPN  ToolPathName ; ")"
          address letter    =  "(ToolPath "
          address width     =  10
          field width       =  20
       end define
```

(Return to TOP)

Q21 Is it possible to suppress arc output in **G18** and **G19** planes?

A From **DP1358** version it is now possible to suppress arcs in individual major planes by inserting in the option file the following :-

```
        suppress xy arc      =   true   default  =  false
        suppress xz arc      =   true   default  =  false
        suppress yz arc      =   true   default  =  false
```

(Return to TOP)

Q22 Can I output my own comments to the tape file?

A Use the ***commands*** tab in the PowerMill ***toolpath parameters form*** and type in the text string you wish to output, preceeding it with a " / ".

The comment line can be configured to be output either **before** of **after** the toolchange.

The option file **MUST** have the ***message output*** flag set to ***true***, however, this will then output **ALL** messages.

To prevent the " **standard** " messages being output, set the ***standard messages option*** to "none" in the toolpath parameters form in PowerMill.

(Return to TOP)

Q23 I have an option file called heid430.opt and want all my tape files to have the file extension " **.h** " automatically so I don't have to edit each file?

A This is a PowerMill function now and is no longer controlled by Ductpost.
Enter the following line into the users ***pmuser.mac***

```
EDIT NCPROGRAM PREFERENCES EXTENSION "heid430" "h"
```

You can enter a line for each machine option file with their appropriate prefered extension.

(Return to TOP)

Q24 Can I force the ***tool length*** to be **zero** in the TOOL DEF statement when post processing for the Heidenhain machine.

It worked in PowerMill 3 but doesn't work in PowerMill 4.

A This is because PowerMill for now outputs a new variable for the ***tool compensation length*** and the block definition should now read......

```
define block tool change
   N ; T1 ToolNumber ; TL CompensationToolLength ; R 0
   N ; T2 ToolNumber ; " Z" ; S ToolSpeed[ToolNum]
 end define
```

(Return to TOP)

Q25 Can I prevent any coolant codes being output to the tape file, even if I have accidentally left the coolant on in PowerMill?

A This can be achieved by re-defining the coolant codes in the option file to the following :-

```
define codes
   coolant on mist    not used
   coolant on         not used
   coolant on flood   not used
   coolant on tap     not used
   coolant off        not used
end define
```

However, this will mean that it is permanently not available so consider this as a last resort.

(Return to TOP)

Q26 I have a Roeders machine and want to output a specific function for this control, the format is SM = 0.015 where the value is calculated as 33% of the thickness.

A The easiest way to solve this sort problem is to use the scaling function when defining the word. In this case.....

```
  define word SM
    address letter    = “SM=”
    address width     = 3
    field width       = 5
    sign              = none
    scale factor      = 33
    scale divisor     = 100
    metric formats
    decimal places    = 3
    decimal point     = true
    leading zeros     = false
    trailing zeros    = false
    imperial formats  = metric formats
  end define
```

```
  define block tool change
    N ; tool number  ToolNum
    N ; spindle  ToolSpeed[ToolNum]  ;  spindle on cw
    N ; “ RMAX= 7  RADMAX= 3000” ;  TOL  Tolerance ; SM Thickness
end define
```

Note that only **integers** are allowed when scaling hence the need to use both the ***scale factor*** and ***scale divisor***.

(Return to TOP)

Q27 I am using a Heid option file and parametric feed rates (Q DEFs) but I get lots of lines output that just contain **FQ2** and no XYZ move.

A This really is a frequently asked question and the answer is simple.

The user has set the **plunge** and **cutting** feed rates in PowerMill to the same value.

This causes a problem when using parametric feed rates as Ductpost has to determine the feed rate to output the correct parameter and if the two are the same the line is output twice with second line only having the FQ2 as XYZ are the same and being modal are not output.

Get the customer to set **different** values – even if only different by **1mm/min** and all will be well.

We are now starting to include in the option a check to prevent this and the following is an example :-

```
N ; T2 ToolNumber =C ; " Z" ; S ToolSpeed[ToolNumber] ; " DL+0.0 DR+0.0"
    QF 1 ; QV Prat ; E ; "PLUNGE FEEDRATE"
if ( Frat  =  Prat )
   error " FEEDRATE AND PLUNGE RATE THE SAME - SEE NC TAP FILE FOR TOOLPATH!!"
else
   QF 2 ; QV Frat ; E ; "CUTTING FEEDRATE"
end if
    QF 3 ; QV Srat ; E ; "RAPID SKIM FEEDRATE"
```

This will stop the posting and inform that the two feedrates need to be set different. (Suggestion that the feed rates

(Return to TOP)

Q28 My machine tool control will only accept angles between -360 and +360

That is to say the C axis requires the output **C25** rather than **C385**. Is it possible to achieve this output from Ductpost.

A The answer, of course, is yes.....BUT, you will need to be pretty conversant with how Ductpost works in order to solve this problem.

For those that want an explanation – read on.....

An additional word (say B360) will need to be created and formated the same as the " **B** " word

```
 define word B360
    address letter  =  "B"
    address width =  1
 end define
#
 define format ( B B360)
    field width       =  8
    sign              = if negative
    modal
    metric formats
    decimal places  = 3
    decimal point   =  true
    leading zeros    =  false
    trailing zeros    =  true
  end define
```

```
.... in the linear block the following logic

  if ( Elpos < -360.0 or Elpos > 360.0 )
     N ; linear ; x coord ; y coord ; z coord ;
         B360 ( Elpos  %  360.0 ) ; feedrate ; M1
  else
     N ; linear ; x coord ; y coord ; z coord ;
         B Elpos ; feedrate ; M1
  end if
```

The % sign means " the remainder after division ", so by dividing the angle outside the +/-360 range we always get an angle within +/-360, otherwise we can just use the " B " value.

(Return to TOP)

Q29 The rotary axis is rotating in the wrong direction – how can I **reverse** it.

A A simple solution, but you must change the correct line in the option file as at first glance, what looks like the obvious thing to change may not give the correct results.

First, determine which axis is to be reversed (***azimuth*** or ***elevation***) by looking in the define keys block in the option file.

Then modify the appropriate line in the option file. (say it's the ***azimuth***)

Search for :-

```
    azimuth axis parameters    =   ( 0.0 0.0 0.0  1  0  0 )
change to......
    azimuth axis parameters    =   ( 0.0 0.0 0.0  -1  0  0 )
```

changing the direction of the axis vector.

DO NOT CHANGE ***azimuth axis direction = positive*** to ***negative*** . THIS CAN occasionally give errors.

(Return to TOP)

Q30 How do I get **"** (Quotes) in the NC output?

A This is now possible and is achieved in the following manor :-

```
define word %D
   address letter  =  "(\\" Date : "
    address width  =  9
   field width       = 2
end define
```

```
define block tape start
   %D  Day  ;  %M  Month  ;  %Y  Year  ; " \")"
   ........
   ........
 end define
```

NOTE the difference if the define word and define block!

(Return to TOP)

Post Processor Questionnaire

The following questions are the basic information in respect of the machine control functions, axes parameters and type of format for control output to produce an initial post processor option. It is essential that this is completed as fully as possible.

Machine and Control Details :-

Machine	NC Control
1/ Machine Manufacturer:-	13/ Maker :-
2/ Model Number :-	14/ Model Type :-
3/ Spindle Vertical or Horizontal :- Which Axis :-	15/ Option Post Processor to be based on nearest control type (YES / NO *)
4/ Number of axes :- (3, 4, 5 or more)	16/ Option Post Processor to be based on alternative control type (YES / NO *)
5/ X axis Limits of travel :-	17/ State Control to be used if NO :-
6/ Y axis Limits of travel :-	**Machine Parameters**
7/ Z axis Limits of travel :-	18/ Tool changer fitted :- (YES/NO*) If YES see question ***27 B & C***
8/ (A) Rotary axis Rotates about :- Limits of travel :-	19/ Max. Spindle Speed :-
9/ (B) Rotary axis Rotates about :- Limits of travel :-	20/ Max. RAPID Feedrate (MM/Inch*/min) :-
10/ (C) Rotary axis Rotates about :- Limits of travel :-	21/ Max LINEAR Feedrate (MM/Inch*/min) :-
11/ State type of axis rotary unit (Spindle, Table or Combination) :-	22/ Units used (Metric / Inch *) :-
12/ Rotary axis Offset(s) from Centre of Rotation :-	23/ Co-ordinate system (Absolute / Incremental *) :-

Additional Requirements :-

24/ **5 axes** machine control - Has it automatic co-ordinate transformation & linearisation capabilities. (YES/NO*) If YES see question ***27***	25/ Spline Machining Capability (YES/NO *) If so what Spline type :- ***Example of tape format to be provided.***
.	.
.	.

* Delete as required

26/ Drawing, or sketch, of Multi-axes Machine configuration (**essential**):-

Post Processor Questionnaire (Cont'd)

27/ Sample of Tape Input indicating the following salient points, with an indication of any special G or M codes and an explanation of their function :-

A/ Tape start heading

B/ First Tool loading requirement

C/ Subsequent tool changes

D/ Tool Offset requirements (e.g. G43 Z100.0 H2 D2)

E/ Circular functions (I,J,K or R)

F/ Drill/Tap Cycle format and codes.

G/ Programme ending.

H/ Any other requirements. (e.g. rotary axes clamping, special coolant etc.)

The clarity of the information provided will assist in producing a quicker result and help to reduce unnecessary retrials.
However, special requirements, and unusual machine tools, may need extra work and cannot always be guaranteed to be feasible. (PLEASE check with support before commitment.)

Agent Contact Name :- eMail address :-

Tel. No. :- Fax. No. :-

Company Post Processor to be supplied to, Name :-
Tel. No. :-
eMail Address :-
Contact :-
(This information will only be used if absolutely required otherwise all contact will be via the agent)

Return Questionaire to am@delcam.com Delcam UK. Tel.No.:- +44 (0) 121 766 5544 Ex.54
Fax.No. :- +44 (0) 121 766 5511

(Form revision:- Issue 2. Date 13/02/2001)

What's New in DUCTpost 1490

(Up-dated 08/09/2006)

Ductpost 1490

DUCTpost User Guide pages up-dated:

Typical Integer, Reals and Characters

Linearisation problem for Head/Head machines

There was previously a problem with the linearisation causing DUCTpost to hang when coordinates were updated using the format illustrated below in the option file :-

```
define block move linear
   N ; linear ; x coord ; y coord ; z coord ( zcoord + ToolLength[1] ) ;
                  azimuth axis ; elevation axis ; feedrate ; M1 ; M2
end if
```

The use of this format is mainly for setting up machines without an RTCP control function and enables easier setting up of the machine. There could well be other instances where this may be useful.

This is now resolved. [ductpost#1098]

Additional Coolant Codes

New coolant codes have been added to complement the new coolant settings available in PowerMILL. We now have
" coolant on air ", " coolant on double " and " coolant on through ".

[ductpost#1103]

Multi-axis Drilling Workplane errors

A problem was discovered when using multiaxis drilling. If we had a vertical hole after a series of non-vertical holes, we could get incorrect values for WorkPlaneA, WorkPlaneB and WorkPlaneC.

This has now been fixed.   [ductpost#1112]

Ductpost 1480

DUCTpost User Guide pages up-dated:

Variables used in Blocks ; Rotary 5 axes Examples ;

Arcs in none principal planes

Some problems have been fixed with arcs, particularly where the tool axis was not in a principal plane.
Occasionally these arcs were incorrectly reported as being non-planar. [ductpost#1079, ductpost#1087]

Tool Change and Intermediate Moves.

The New functions introduced in PowerMill 6.0 as a Preview, to select the Tool Change position at the end of a tool path either BEFORE, or AFTER, an intermediate move had a problem with setting the **Tool Change Position** to "*Before Connection*" in the PowerMILL NC Program form.

DUCTpost did not output the necessary 3+2 angles correctly in all cases. This issue is now solved. [ductpost#1083]

New Drilling Function Variable

Direct access has been provided to the drill cycle "code number" that is written in the CLDATA.
This "code number" uniquely identifies the cycle type. The block variable " ***CycleCodeNumber*** " provides this. [ductpost#1086]

Retract and Re-Configuration

A problem has been fixed with the block variables " ***ModelX, ModelY, ModelZ*** " usage.
The issue was that these were not working correctly when DUCTpost was performing a 5-axis retract-and-reconfigure function. [ductpost#1090]

Posting Error Message “Warning: axis direction reversal. Please refer to the "What's new document" ”

This is an old fault that has been corrected 5 years ago but when posting the above error message will occur if an multiaxis option file contains something like this :-

```
spindle azimuth rotation =  true
azimuth axis units        =  degrees
azimuth axis direction    =  negative
azimuth centre           =  ( 0.0 0.0 0.0 )
azimuth axis param       =  ( 0.0 0.0 0.0  0  0  1 )
```

To reverse the direction of a rotary axis ON NO ACCOUNT use “ negative ”, **ALWAYS** use :-

```
azimuth axis param      = ( 0.0 0.0 0.0  0  0  -1 )
```

Go to (DP1460 ; DP1440 ; DP1430 ; DP1425 ; DP1414 ; DP1405)
(DP1364 ; DP1361 ; DP1360 ; DP1357 ; DP1353 ; DP1350)

Ductpost 1470

PAF Change

Requires PAF 2005.06 codes.

Change in Drilling function

DUCTpost 1460 introduced a change in behaviour for SafeZ in canned cycles. Although output is correct, this change could produce results that may be unexpected compared to previous. Therefore, this behaviour is no longer the default.
Users requiring the newer behaviour for SafeZ in canned cycles can set :-

use true safez in cycles = true   [ductpost#936]

Ductpost 1460

DUCTpost User Guide pages up-dated:

Variables used in Blocks ; Circles ; Block Definitions

Helical milling cycles that are expanded by DUCTpost

These now go through the “ ***cycle start*** " block immediately before the Helical moves begin using “ if (Cycle = = 11) ”. [ductpost#975]

A new “ *define block special record* ” has been added.

This block is triggered whenever a PowerMILL “ **special record** ” is encountered.

PowerMILL ***special records*** are general-purpose records and are mostly used to carry information that is useful, rather than essential. They are characterised in CLDATA by a Major Word of **29000** and a Minor Word of **0**, and an Integer Code. (see DP1440)

A block variable “ ***SpecialRecordCode*** ” has been added, which returns the Integer Code used to identify the particular special record in question. One example where this functionality might be useful is for the case of “ **expanded drilling cycles** ”.

A special record is written out by PowerMill to indicate that this expansion has been taken place. With the previously, DUCTpost needed changing each time such a new record was added. With this new scheme, it is now possible to make use of these new records without having to await an update to Ductpost.

An example of this is :-

```
define block special record
         if ( SpecialRecordCode = = 21 )
           N ; “( Moves for an expanded drill cycle follow ) ”
        end if
  if ( SpecialRecordCode = = 40 )
   if ( SpecialRecordString[40] = = "Drill" )
    set swg
   end if
 end if
 if ( SpecialRecordCode = = 46 )
   N ; " ; Variable Feedrate Output"
   set swj
 end if
end define
```

If desired, the “ **real** ” and “ **string** ” parts of the special record can be extracted using the constructs “ ***SpecialRecord[SpecialRecordCode]*** ” for the real part, and “ ***SpecialRecordString[SpecialRecordCode]*** ” for the string part (see above). [ductpost#976]

Helical milling radial compensation

A problem existed with cutter compensation not being turned **off**, when cutter compensation was used with expanded helical multiaxis drilling.

This has been fixed. [ductpost#973]

The option file dump command, “ ductpost –w ”, would incorrectly report that both “ heid ” and “ heid400 ” have “ *tape split on toolchange = true* ”.

This has now been fixed. [ductpost#393]

Workplane Definitions

In order to prevent confusion, it is now only allowable to use “ WorkplaneA ; WorkPlaneB ; WorkplaneC ” for option files where “ ***workplane angles = apparent*** ”.

Similarly, it is now only possible to use “ WorkplaneAz ; WorkplaneEl ” if “ ***workplane angles = machine tool*** ”. [ductpost#988]

Arc output not in Principal Planes

Previously, it was possible for arcs to be output that were not exactly in principal planes. This is no longer possible. [ductpost#989]

Operational adjustments to Ductpost

Some slight tweaks were made to DUCTpost's handling of CLDATA to ensure consistency with future versions of PowerMILL.
[ductpost#1012 , ductpost#1021 , ductpost#1027]

Linearisation problem at " anti-parallel " orientation

Previously, there was a problem with linearisation when the tool axis reached an orientation that was anti-parallel to the singular axis.
This has been corrected. [ductpost#1010]

Defined blocks exceeding allowed number of instruction with inclusion of User Block Calls

If a " ***define block xxxx*** " is " **expanded** " by including any " ***defined user block*** " the total number of instructions could be greater than the maximum allowed, and DUCTpost would have crashed.
This has been fixed, and the number of instructions allowed in a block has been increased to make this ***less*** likely to occur. [ductpost#1016]

Workplane Variable additions

Some additional block variables have been provided to give access to the Toolpath Workplane Data.
" ***ToolpathWorkplaneA*** " and so on, grant access to the Euler angles for a Toolpath Workplane.
" ***ToolpathWorkplaneX*** " and so on, grant access to the Position of the Toolpath Workplane.

Note that in most cases, users should carry on using " ***WorkplaneA*** " and so on, since that always gives the ***current*** workplane, and properly takes care of the workplane shifts used in 5-axis drilling. [ductpost#958]

Machine Workplane Conformity

TABLE-TABLE machines that have the settings " ***workplane angles = apparent*** " or " ***workplane angles = machine tool*** " now use machine tool workplanes when drilling cycles are output, similar to HEAD-HEAD and HEAD-TABLE machines. [ductpost#957]

Inconsistancy between " Retract and Recofiguration " & " Expanded Helical Cycles "

It is now possible to use " ***retract and reconfigure style*** " and " ***expand helical drilling cycles*** " without incorrect interaction between them. [ductpost#1036]

Addition of a new linearisation flag

A flag " ***linearise first move = true*** " has been added. The flag **default** is ***true***, but if it is set to ***false***, the move to the first position in the CLDATA file is **not** linearised. [ductpost#900]

Problem with Linearisation of First Moves

A problem with the linearisation of the very first move has been corrected. Previously, if the " ***from block*** " was not empty, and didn't contain angles, then the coordinate values were out of step with the angles as output. This caused the linearisation to go wrong, since the initial position was wrong. [ductpost#900]

Rapid Move Feedrates

In a couple of cases, RAPID moves in CLDATA were not being converted to fast feedrate moves correctly, this has been corrected. [ductpost#1035]

A new Variable added for Tapping Cycles

A block variable " ***pitch*** " has been added, which gives the designated PowerMill **pitch** setting as required by some machines for tapping. [ductpost#814]

Drill Cycle " SafeZ " operation corrected

A change has been made to the " ***SafeZ*** " block variable when used in cycles. Previously, " ***SafeZ*** " behaved differently in cycles to elsewhere. When used in cycles, " ***SafeZ*** " returned the Z value of the **last coordinate** before the cycle. This inconsistency has been removed.

The previous behaviour is obtainable by using the new variable " ***CycleSafeZ*** ", if so desired. [ductpost#936]

Addition of a new flag for Toolpath Safe & Start Z values

A flag " ***use toolpath safe heights = false*** " [default] has been added.
If set to " ***true*** " then the current ***SafeZ*** and ***StartZ*** values are taken from those set for the current toolpath.
Previously, the first ***SafeZ*** and ***StartZ*** values were used for **all** toolpaths. [ductpost#1041]

Addition of new code words for Clamping

Two new code words have been added " ***azimuth clamp on*** " and " ***azimuth clamp off*** ".
Use these for **Azimuth** axis clamp codes and the "***clamp on*** " and " ***clamp off*** " for the **Elevation** axis clamp codes. [ductpost#632]

Addition of new flag to prevent clearing drill cycle modal words

A setting has been added that can prevent the " **clearing** " of modal words by default that takes place at the start of a drill cycle.
To **stop** the clearing of modal words, use the setting " ***clear modal words for cycles = false*** " [ductpost#1042]

Number of address characters for words increased

The number of characters allowed in the address letter of a word has increased to " **128** " from the original **19** maximum. [ductpost#1044]

Problem with multiaxis Arcs

A problem with arcs in CLDATA 5-axis has been fixed.

Previously, if an arc was output in 5-axis mode for a toolpath with an extremely fine tolerance the arc may not have been output correctly by DUCTpost. [ductpost#1047]

Compensation variable problem

" ***ToolComp*** " is now working correctly.
Previously, this may have only worked fully if a tool change was present between toolpaths. [ductpost#1045]

-

Go to (TOP ; DP1470 ; DP1460 ; DP1430 ; DP1425 ; DP1414 ; DP1405)
(DP1364 ; DP1361 ; DP1360 ; DP1357 ; DP1353 ; DP1350)

Ductpost 1440

Access to CLDATA 29000 records provided. (Please note, further work is is in progress on this feature)

Two block variables have been added to provide access to a class of CLDATA record that are written by PowerMILL.
The records are all of the form :-

29000 0 <integer_code> <string_value> <real_value>

For example, there exists a record that directly gives the number of pecks in a drilling cycle :-

29000 0 11 Number_pecks <real_value>

Also, there is a record that gives the output workplane name :-

29000 0 20 <string_value> 0.0

With DUCTpost it is possible that it lags behind PowerMILL in providing access to these new records, hence the two new block variables. :-

" ***SpecialRecord*** " returns the **real value** present, and " ***SpecialRecordString*** " returns the string value in the record.

For example the " ***Special record*** " with code ***[22]*** gives the ***peck depth*** :-

```
define block cycle start
   N ; G4 ; cycle retract ; x coord xcoord  =C ; y coord ycoord =C ;
                    drill hole depth ; clearplane ; Q1 SpecialRecord[22] ;
                    cycle dwell ; feedrate Prat
     end define
```

Another example, consider :-

```
define block rapid
   if ( swa )
     N ; " (-----------------)"
     N ; " (TOOLPATH" ; WPN SpecialRecordString[3] ; " )"      ## = ToolPathWorkplaneName
     N ; " (-----------------)"
    unset swa
  end if
end define
```

NOTE :- these are read directly from the CLDATA output, and if the record has not been " **seen** ", an **error** message will result!

For example, if the " ***SpecialRecordString[3]*** " is used in the " **tape start** " block which is called before the CLDATA for this record is read, an error message will occur :-

```
define block tape start
   N ; " (-----------------)"
   N ; " ( *** WON'T WORK BECAUSE WE HAVEN'T SEEN THE RECORD YET *** ) "
   N ; " (OUTPUT" ; WPN SpecialRecordString[3] ; " )"
   N ; " (-----------------)"
end define
```

Reference [ductpost#938]

A problem existed when using " ToolVectorX " etc. with continuous 5-axis toolpaths.

In an option file that used " *workplane angles = apparent* " or " *workplane angles = machine tool* ", the tool vector returned would be for the Z-axis of the workplane.

This has been fixed, and the correct toolvector is now returned. [ductpost#942]

The number of string literals allowed on an output line has been increased.

Increased from five to one hundred.
When this limit is exceeded, an **error** message results.
Previously, DUCTpost could crash, or act unpredictably in these circumstances. [ductpost#946]

Multiaxis drilling has been enabled for HEAD-HEAD and HEAD-TABLE machines.

For HEAD-HEAD and HEAD-TABLE machines that are capable of supporting an inclined workplane function, multiaxis drilling is now possible.

This functionality ***requires*** the use of the " define block **datum shift** " to ensure that the workplane definitions are correctly output.

To ensure that the workplane is ***turned off*** where necessary, use a " define block **multiaxis transition** ". (see DP1405 for info on these blocks) [ductpost#926]

An option has been added to disregard the translational part of toolpath workplanes

When using machine tool workplanes, setting " ***workplane origin shift = false*** [default = ***true***]", the translational part of a workplane is not used.

(This has been added to cater for a Hyperthetical case where a machine tool requires the datum shifted ToolPath Workplane brought back to the Output Workplane origin.

It is unlikely to be needed but has been include whilst other remedial work was being carried out.) [ductpost#926]

Rotary 4th axis problems

Some extra error checking has been added for 4-axes option files. [ductpost#949]

A problem with moves at rapid feedrate while in inverse time

This has been fixed. [ductpost#949]

Helical Drilling Cycles

An option has been added that allows " **helical drill** " cycles to be expanded into a series of helical moves.

To use this, set " ***expand helical drilling cycles = true*** ". [ductpost#923]

Please Note:- Setting Radial compensation with this feature has a bug, in that we forgot to implement the compensation OFF call, this will be fixed in the next release.

Drill cycles Peck Depth

A problem where the peck depth could be reported incorrectly has been fixed. [ductpost#812]

Reconfiguration Moves

The option: " ***retract and reconfigure style = combine*** [default = ***none***]" has been added. [ductpost#952]

When this option is chosen, then DUCTpost's behaviour on **retract-and-configure** changes slightly. The following should illustrate this more clearly :-

```
====================================
retract and reconfigure style = none
====================================
G94 Z280.173 F500
G93 A-30.001 C360.0 F5445.9
X55.708 Y-82.098 F5445.9
G94 Z196.173 F500
Z180.173 F1000

=======================================
retract and reconfigure style = combine
=======================================
G94 Z280.173 F500
G93 X55.708 Y-82.098 A-30.001 C360.0 F5445.9
G94 Z196.173 F500
Z180.173 F1000
```

Additionally an option: " ***retract and reconfigure style = linearise*** " has been added.
If this option is selected, then DUCTpost inserts extra moves in an attempt to ensure that the tip stays in the same position relative to the part while the new angles are adopted.
This should result in less extreme movements.

```
======================================
retract and reconfigure style = none
======================================
X-55.692 Y75.806 Z182.728 A27.506 F6018.5
X-55.708 Y82.098 Z180.173 A30.001 F5445.9
G94 Z280.173 F500
G93 A-30.001 C360.0 F5445.9
X55.708 Y-82.098 F5445.9
G94 Z196.173 F500
Z180.173 F1000
G93 X55.727 Y-88.989 Z177.118 A-32.807 F4592.5

======================================
retract and reconfigure style = linearise
======================================
X-55.692 Y75.806 Z182.728 A27.506 F6018.5
X-55.708 Y82.098 Z180.173 A30.001 F5445.9
G94 Z280.173 F500
G93 X-80.608 Y68.053 Z277.869 A21.183 C206.452 F556.8
X-88.522 Y63.356 Z276.569 A11.85 C234.453 F556.8
X-73.944 Y57.861 Z282.001 A1.806 C264.582 F556.8
X-35.091 Y34.742 Z293.09 A-9.257 C297.77 F556.8
X18.504 Y-20.92 Z295.906 A-21.042 C333.125 F556.8
X55.708 Y-82.098 Z280.173 A-30.001 C360.0 F556.8
G94 Z196.173 F500
Z180.173 F1000
G93 X55.727 Y-88.989 Z177.118 A-32.807 F4592.5
```

Detection of reconfiguration move

A block variable has been added that indicates when DUCTpost is performing a **retract-and-reconfigure** series of moves.
The variable " ***Reconfiguring*** " is **true** while DUCTpost is in *retract-and-reconfigure* mode, and **false** otherwise. [ductpost#965]

Go to (TOP ; DP1470 ; DP1460 ; DP1440 ; DP1425 ; DP1414 ; DP1405)
(DP1364 ; DP1361 ; DP1360 ; DP1357 ; DP1353 ; DP1350)

Ductpost 1430

DUCTpost User Guide pages up-dated:

Variables used in Blocks; Block Definitions

Linearisation could occasionally fail soon after a tool change had occurred.

This has been corrected. [ductpost898]

Simple "Multiaxis drilling" with cycles has been implemented.

Even when " ***workplane angles = none*** ".
However, this is only for TABLE-TABLE machines at this stage. [ductpost895]

Control of rewind at angular rotation limit.

The function flag " ***full rewind at limit = false*** " has been added to control the behaviour on reaching a rotary angular limit.

When set to " ***true*** ", then DUCTpost rewinds the axis by several revolutions to try and minimise the number of retractions.

As set to " ***false*** ", then the old behaviour is used, which is to rewind by no more than **one** revolution from the limit until rotation is completed. [ductpost520]

Pure-geometric workplane angles (often known as Euler angles)

DUCTpost now has the ability to produce pure-geometric workplane angles (often known as Euler angles) in all the possible conventions.

The syntax is this: " ***workplane angle convention = 1*** ". [**default**]

Allowable values for the integer range from 1 to 24, and each integer corresponds to a particular convention, this is indicated in the table below.

"**s**" denotes that the rotations are about world-space axes.
i.e. XYZs means that the convention is rotate about world X, then world Y, then world Z.
"**r**" denotes that the workplane rotates about its own axes.
i.e. XYZr means rotate about X, then about ***new*** Y, then about ***new*** Z

	s		**r**
1	XYZs	**2**	ZYXr
3	XYXs	**4**	XYXr
5	XZYs	**6**	YZXr
7	XZXs	**8**	XZXr
9	YZXs	**10**	XZYr
11	YZYs	**12**	YZYr
13	YXZs	**14**	ZXYr
15	YXYs	**16**	YXYr
17	ZXYs	**18**	YXZr
19	ZXZs	**20**	ZXZr
21	ZYXs	**22**	XYZr
23	ZYZs	**24**	ZYZr

Occasionally, the angles could be missed from the start of a toolpath.

This has now been fixed. [ductpost914]

Multi-axis working.

A function flag " ***use fiveaxis always = false*** " has been added.

If this is set to " ***true*** ", then even if CLDATA files do not contain a MULTAX ON record the post processor will go through the multiaxis code, with any corresponding offsets applied. [ductpost875]

A Windows 2000 problem where "OptionFileName" would occasionally return the name with extra characters appended.

This has been fixed. [ductpost899]

Problem with “ ToolVectorZ ”

There was a problem with “ ***ToolVectorZ*** ”, which would return 0.0 for pure 3-axis CLDATA when the option file used either “ ***workplane angles = apparent*** ” or “ ***workplane angles = machine tool*** ”. This has now been corrected. [ductpost909]

Problem with incremental output in multi-axis work.

An old bug has been fixed when working with incremental in multi-axis work. It used to be the case that identical angular increments would **NOT** be output, incorrectly. Identical increments in (say) X would produce output, correctly. [ductpost679]

Additional Variables added to give PowerMill Version.

Two variables have been provided to give access to the codebase of the version of PowerMILL that was used to write the CLDATA file.

“ ***PMCodeBase*** ” provides the 5-digit part of the codebase number (The codebase number itself for a 5-digit number, or the first 5 digits of a 7-digit codebase number),
“ ***PMCodeBaseRevison*** ” provides the extended part of a codebase number (i.e. “ *PMCodeBaseRevison* ” is 0 for a 5-digit number, or is the last two digits of a 7-digit number).
As an example, for a cutfile produced from **PowerMILL CB 1050632**, “ *PMCodeBas*e ” would return **10506**, and “ *PMCodeBaseRevison* ” would return **32**.

Go to (TOP ; DP1470 ; DP1460 ; DP1440 ; DP1430 ; DP1414 ; DP1405)
(DP1364 ; DP1361 ; DP1360 ; DP1357 ; DP1353 ; DP1350)

DUCTpost 1425

DUCTpost User Guide pages up-dated:

Circular Moves ; Drilling Cycles ; Variables used in Blocks

New block variables:

" **ModelX** ", " **ModelY** ", and " **ModelZ** " have been added to give access to the coordinates in the model system in all circumstances. These were introduced for a specific operational debug and as such will not normally be of general use. [ductpost854]

" **MoveType** " enables various types of tool path movement to be found, usually used in " **if (** *MoveType = = n* **)** statements.

Feed Move = **0** ; Rapid Move = **1** ; Rapid Feed Move = **2** ; Plunge Move = **3**
Lead IN Move = **4** ; Lead OUT Move = **5** ; Cycle Link Move = **6**
Cycle Plunge Move = **7** ; Cycle Rapid Move = **8** ; Cycle Output Move = **9**
Intermediate Join Move = **10** ; Intermediate Multi-axes Move = **11**

" **NumberTools** " this will provide the number of tools used in the NC programme.

New operational flag:

The flag " **multiaxis toollength used** " has been introduced to control whether or not to take tool length into account for the machine tool. This is most useful for HEAD-TABLE machines.
The default is " **true** ", meaning that the *tool length is taken into account*. Set to " **false** " if tool length is *not* to be accounted for. [ductpost790]

Incremental Move problems:

A problem where retract-and-reconfigure could fail in incremental mode has been solved. [ductpost858]
A small problem with angles when working in incremental mode has been fixed. [ductpost867]

Multi-axes retractions:

" ***minimise multiaxis retractions = true*** " could occasionally lead to the singular axis wandering slightly before reaching the correct value. This has been corrected. [ductpost869]

Linearisation convergence problem:

In some circumstances, DUCTpost's linearisation algorithm could fail to converge in a region where a retract-and-reconfigure was required. This is now fixed. [ductpost885]

Drilling Cycles with 3+2 orientation:

For TABLE-TABLE machines, the restriction that prevented 3+2 drilling has been lifted. [ductpost877]

Tool paths with Contact Normals:

CLDATA with contact normals that also contained drilling tool paths, or arc moves could have caused DUCTpost to misbehave. This is no longer the case. [ductpost852]

Arc problems:

For some arcs, where the tolerance was quite fine, DUCTpost could occasionally need to split the polygonisation of the arc and then get confused. This has now been corrected. [ductpost894]

Arc CLDATA records presented in continuous 5 axis tool paths (as arc leads, for example) could confuse DUCTpost, if the next tool path was 3+2. This situation has been rectified. [ductpost896]

Go to (TOP ; DP1470 ; DP1460 ; DP1440 ; DP1430 ; DP1425 ; DP1405)
(DP1364 ; DP1361 ; DP1360 ; DP1357 ; DP1353 ; DP1350)

DUCTpost 1414

An option has been added that influences DUCTpost's choice of rotary angles:

In order to try and avoid reaching for an angular move that is likely to exceed the rotation limit, a new function has been provided :- " ***minimise multiaxis retractions = true*** [**default = false**] ".
This should improve things by removing some initial gratuitous reconfigurations although we can not eliminate the need to retract-and-reconfigure for a tool path.
However, if this behavior is chosen, then it is likely that solutions may be chosen that are not " **close** " to the current solution. [ductpost778]

Convergence of linearisation of multiaxis moves:

A rare problem with the convergence of linearisation of multiaxis moves has been fixed. [ductpost834]

Previously, the vector linearisation capability of DUCTpost wasn't working correctly:

This has now been fixed. [ductpost777]

Tool vector variables have been added to provide easy access to the linearised tool vector:

These are " **ToolVectorX** ", " **ToolVectorY** " and " **ToolVectorZ** ". [ductpost665]

The block variable " *OutputWorkplaneName* " has been added:

This makes the name of the PowerMILL Output Workplane available. [ductpost768]

A 3+2 tool path that was preceded by a tool change point could be misidentified as being continuous 5 axes:

For such a tool path, " **ToolPath5Axis** " was true. This has now been rectified. [ductpost780]

There existed a theoretical possibility of a crash when using tool path work planes:

If the tool path work planes as written in CLDATA are not correct, then previously this could cause DUCTpost to fail with a runtime error. Now, the CLDATA is subject to some sanity checking to ensure that this can no longer happen. [ductpost789]

A possible problem with " *workplane angles = machine tool* " has been fixed:

Previously, the angles chosen for the work plane were correct, but not necessarily optimal. [ductpost793]

If " ***machine tool angles*** " has been selected, the variables " **WorkplaneAz** " and " **WorkplaneEl** " now store the angles that will be used to transform a given work plane. Using these ***variables***, rather than **Azpos** and **Elpos** means that there is no possibility of the work plane transformation getting out of synchronization with the angles used to define it. [ductpost801, dicc44927]

The ability to base feed rates on the contact point, rather than the tip position, has been added:

To make use of this, set " ***contact point based feedrate = true*** " [**default = false**].

Within PowerMILL, it is necessary to switch on **3D machine compensation** in the NC program edit form.
It's also necessary to have calculated tool paths with **contact normals preserved** , to do this choose ***Tools*** --> ***Options***, select the " ***Toolpath*** " tab and check the " ***Contact Normals*** " option. [ductpost776, DICC44777]

Improvement in the calculation of Inverse Time Feed rates:

Some slight changes have been made that should improve the calculation of inverse time feed rates to better reflect the true requirement. [ductpost804]

The first point after a tool change was sometimes missed out if both " *tool reset coordinates = 1* " and either " *workplane angles = apparent* or *machine tool* " defined:

This has been corrected. [ductpost805]

DUCTpost could hang while trying to post-process certain 5 axis CLDATA:

A rare problem has been fixed, which was that DUCTpost could hang while trying to post process certain 5 axis CLDATA. This was only likely to occur for option files with restricted angle movement. [ductpost826]

DUCTpost now allows concatenation of strings within block expressions:

Both literal strings (i.e. enclosed in quotes **""**) and strings returned by block variables are allowed.
The concatenation operator is " + ".
For example:-

```
define block tape start
  STRING ( "(OPT. NAME: " + OptionFileName + "  JOBNAME: " + JobName + ")" )
 end define
```

Note:- Type checking for expressions has also been made more stringent and expressions such as " **(OptionFileName + 5.0)** " are no longer allowable. [ductpost830, ductpost833]

Support has been added for the " deep drill 2 " cycle from PowerMILL:

The value of the " **cycle** " variable for this cycle is **13**. [ductpost829]

The number of words has been slightly increased:

The previous limit was **80** this is now increased to **95** [ductpost376]

Access to " contact normals " has been provided:

The new block variables " **ContactNormalX ; ContactNormalY ; ContactNormalZ** ", provide access to the contact normals of the tool path.

Note:- If an option file attempts to access ***contact normal information*** for an tool path that **does not *contain contact normal information***, then an **error** occurs.
In order to help with writing option files that can deal with CLDATA that ***may not reliably*** contain contact normals, the block variable " **GotContactNormals** " can be used to check if the CLDATA contains contact normal information. [ductpost665]

The following example block will not cause an error, even if ***contact normals*** are not present:-

```
define block move linear
     if ( GotContactNormals )
       N ; x coordinate =C ; y coordinate =C ; z coordinate =C ;
             azimuth axis =C ; elevation axis =C ;
           TX ToolVectorX =C ; TY ToolVectorY =C ; TZ ToolVectorZ =C ;
           NX ContactNormalX =C ; NY ContactNormalY =C ; NZ ContactNormalZ =C ; IVF feedrate
     else
       N ; x coordinate =C ; y coordinate =C ; z coordinate =C ;
             azimuth axis =C ; elevation axis =C ;
           TX ToolVectorX =C ; TY ToolVectorY =C ; TZ ToolVectorZ =C ; IVF feedrate
     end if
 end define
```

A problem with DUCTpost wire spark functionality has been fixed:

Previously, some wire spark option files would fail even when the CLDATA was legitimate. [ductpost844]

When comparing string variables using comparison operators such as = =, !=, <, >, <=, >= (and the others):

These were not guaranteed to work as expected. For example, the following may not have worked predictably:[ductpost845]

```
if ( ToolName != ToolPathName )
     N ; "; Toolname isn't the same as the toolpath name"
 end if
```

NOTE:- This only occurred when comparing ***variables***... When comparing with ***littorals***, all worked as desired. So, the following always worked correctly, since it uses the string littoral " MY_FAVOURITE_TOOL "

```
if ( ToolName != "MY_FAVOURITE_TOOL" )
     N ; "; why aren't we using my favourite tool?"
 end if
```

Coolant change with no tool change:

A problem still exists with this and also with coolant " **none** " : It is hoped to rectify this in the next release.

Go to (TOP ; DP1470 ; DP1460 ; DP1440 ; DP1430 ; DP1425 ; DP1414)
(DP1364 ; DP1361 ; DP1360 ; DP1357 ; DP1353 ; DP1350)

DUCTpost 1405

User blocks:

The concept of " user blocks " has been added to DUCTpost. A user block is defined as below:-

```
 define block user ToolList
      N ; " ;"
      N ; " ; WERKZEUGLISTE : "
      N ; " ; NR. Bezeichnung/Durchmesser/Eckenradius/Länge"
      repeat ( NumberTools )              ( See DP1364 )
         N ; STORE ToolPathFromTool[LoopCounter] ;
              T ToolNum[Word{STORE}] ; T1 ToolName[Word{STORE}] ;
              TD ( ToolRadius[Word{STORE}] * 2 ) ; TR TipRadius[Word{STORE}] ;
              TL ToolLength[Word{STORE}]
      end repeat
      N ; " ;"
  end define
```

The block is called in the following manner:-

```
  define block tape start
      N ; " BEGIN PGM" ; ID JobName ; " MM"
      if ( DPVersion < 1405 )
         error "AKTUALISIEREN SIE ZU VERSION 1405 ODER SPÄTERES"
      else
         N ; VER DPVersion ; OFN OptionFileName
      end if
     call block ToolList
      N ; " BLK FORM 0.1 Z" ; X BlockMinX ; Y BlockMinY ; Z BlockMinZ
      N ; " BLK FORM 0.2"   ; X BlockMaxX ; Y BlockMaxY ; Z BlockMaxZ
      ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
  end define
```

User blocks are intended to make sure that bits of ***repeated code*** are factored out, and remain in step when changes are made. Also, when used carefully, they should make option files more readable.

There are some restrictions:-

The number of user blocks is limited to 50

User blocks ***can not call*** other user blocks at this stage.

The total number of instructions available when a block is " ***expanded*** " by ***including all user blocks*** is limited to be the same as under " **normal** " circumstances. **Using user blocks is not a means of getting more instructions into a block.**

Currently, the maximum number of instructions is **512**.   [ductpost711]

Toolpath workplane names:

The variable " **ToolpathWorkplaneName** " has been added to allow these to be written to the NC file. [ductpost445]

Variables for types of move:

New variables " **MoveType** ", " **MoveComponentType** ", and " **MoveMinorType** " have been added.

These can be used in certain circumstances to take particular actions for a particular class of move.

For example, a " ***connection move*** " has a " **MoveType** " value of **11**.

This work is in the the first stage of completion, and will be made easier to use in a subsequent release. [ductpost752]

Remove anomalous use of rapid block in 5-axis:

When performing a retract-and-reconfigure, previously DUCTpost could insert point-to-point (**G0**) moves.

Now, all those moves are done as linear (**G1 *rapid skim feedrate***), in just the same manner as all other moves. [ductpost603]

Prevent filename truncation:

Certain variables, such as " **JobName** " and " **OptionFileName** " could be ***truncated*** into **8.3** style file names when DUCTpost was run from PowerMILL. This should no longer occur. [ductpost762]

" datum shift " and " multiaxis transition " blocks added:

Two new blocks have been added to DUCTpost that should make writing option files more straightforward.
These blocks are purely **optional**: if they are not defined, then DUCTpost will work exactly as before.
The first of the new blocks is a " **datum shift** " block. This block, if present, is triggered immediately before a positional 5-axis toolpath (also known as a 3+2 toolpath).
A typical use of this block would be to simplify the datum and orientation shifts that are used by " **machine tool workplanes** ".
The other new block is the " **multiaxis transition block** ". This block, if present, is triggered immediately before the first full toolaxis move.
A typical use of this block would be to make sure that any " **RTCP** " activation code is output. [ductpost766]

DUCTpost 1400

clarification of multiaxis toolpath types:

Consider those cases when DUCTpost is defining output for machine tool workplanes, with either " **workplane angles = apparent** " or " **workplane angles = machine tool** ".
DUCTpost has certain variables that classify the toolpaths to make it easier to write more complex option files. These are " **ToolPath3axis** ", " **ToolPath5axis** " and " **ToolPath3Plus2** ".
When machine tool **workplane angles** is defined (i.e. other than **= none** [**default**]), then **all** toolpaths are classed as either " ToolPath5axis " (Continuous 5-axes toolpaths) **OR** " ToolPath3Plus2 " (all 3-axes and 3+2). This ensures that the workplanes in PowerMILL are used as defined, even if this does not require any rotation of the machine tool axes. [ductpost723, ductpost716]

machine tool workplanes:

When DUCTpost was using " machine tool " workplanes, previously not all moves went through the linear block. This was inconsistent, and has been harmonised. [ductpost738]

Go to (TOP ; DP1470 ; DP1460 ; DP1440 ; DP1430 ; DP1425 ; DP1414 ; DP1405)
(DP1361 ; DP1360 ; DP1357 ; DP1353 ; DP1350)

DUCTpost 1364

Coolant change with no tool change:

Previously, this was not working correctly, and has now been corrected. [ductpost709]

Simple " loop " construct for DUCTpost:

A simple option file **loop** function has been intruduced.

The syntax and a suggested use is as follows :-

In the example below the following new words would need to be defined and formated to accommodate the various values attributed to them, and of course added to the word order list :-
STORE ; INT1 ; STR1 ; STR2 ; NUM1 ; NUM2 ; NUM3 ; TPN

New variables have been added :-

" ***LoopCounter*** " , which gives the number of times that the loop has been executed to date.

" ***NumberToolPaths*** ", allows a list of toolpaths to be produced.

" ***NumberTools*** ", to enable an iteration over the number of tools used, which returns the number of unique tools within the CLDATA file.

" ***ToolPathFromTool*** ", which maps toolnumbers to a toolpath number. This is necessary because DUCTpost indexes all quantities on the toolpath.

```
define block tape start
   "%"
    ID JobName
   "("
   "( TOOL LIST : " ; STORE NumberTools  ; " unique tools "
   "( No. Type       ID                      Diameter   Tip Rad   Length"
repeat ( NumberTools )
      STORE ToolPathFromTool[LoopCounter]
   #   we store the toolpath number for this tool
   #   then print a list of tool properties, using the stored toolpath number as the index with the construct
XXX  YyyyYyyy[Word{STORE}]
      INT1 ToolNum[Word{STORE}]  ; STR1 ToolType[Word{STORE}] ;
         STR2 ToolName[Word{STORE}]  ; NUM1 ToolRadius[Word{STORE}] ;
         NUM2 TipRadius[Word{STORE}]  ; NUM3 ToolLength[Word{STORE}]
end repeat
repeat ( NumberToolPaths )
     N ;  TPN ToolPathName[LoopCounter]
end repeat
     ..........................................
 end define
```

This produces something like the following NC output :-

```
%
 ( File : TC-Test
 (
( TOOL LIST :  4 unique tools
( No. Type       ID                        Diameter  Tip Rad  Length
( 3   ENDMILL        20mm End Mill            20.0       0.0     100.0
( 6   ENDMILL        6mm End Mill              6.0       0.0      30.0
( 8   BALLNOSED   10mm BNSD                 10.0       5.0      50.0
( 19  TIPRADIUSED 10 Dia 3 Tip End MIll   10.0       3.0      50.0
(
( TOOLPATH - Rough
( TOOLPATH - Semi-Finish
( TOOLPATH - Finish
( TOOLPATH - Finish2
(
 N1 G91 G28 X0 Y0 Z0
```

The " **repeat** " block is fairly limited at this stage and can not be nested at the moment. [ductpost710]

An " impossible " 5-axis case was not identified:

Strange moves could occur for toolpaths that were in fact " **impossible** " to machine with the defined axes configuration. This could ***only*** occur when the move was geometrically impossible, rather than was made impossible by some angular limit.

This is now fixed. [ductpost715]

Crashes in option file reading:

Previously, certain errors in option file construction could cause DUCTpost to crash while reading the option file.

For example, a missing " **right hand side** " to a flag, such as :- **integer 26 =**

These crashes have been fixed, and the errors are now reported. [ductpost713]

Problem with machine tool workplanes in 3-axis:

When using ***workplane angles = apparent***, or ***machine tool***. If the **output workplane** and the **toolpath workplane** have the same Z-axis alignment, DUCTpost would report the toolpath as **3-axis** (" ***ToolPath3Axis*** " would be **true**).

However, DUCTpost would incorrectly apply workplane transformations.

This has now been fixed and transformations will not apply. [ductpost716]

Division by zero in option files:

When using the division or the modulus operators (" **/** " and " **%** "), DUCTpost did not check that the divisor was non-zero.

For example, the following code if included in a definintion block would have caused previous versions of DUCTpost to crash.

```
edit inta 0
    OUTPUT ( xcoord / inta )
```

DUCTpost now detects **division by zero** and fails with an appropriate error message. [ductpost707, ductpost708]

" else if " block statement:

Previously, this was not working correctly. The problem is now fixed. [ductpost407]

Go to (TOP ; DP1470 ; DP1460 ; DP1440 ; DP1430 ; DP1425 ; DP1414 ; DP1405)
(DP1364 ; DP1360 ; DP1357 ; DP1353 ; DP1350)

DUCTpost 1361

Absolute data and incremental data:

These are used to output the appropriate NC codes **G90 / G91**. In addition, the block variables switch DUCTpost into the appropriate output mode (**absolute** or **incremental**).

The problem was that this wasn't working at all when " **=C** " was used to force output, this problem has now been corrected. [ductpost688]

Arc problem for okuma:

A problem that resulted in incorrect output for arcs in the XZ and YZ planes has been corrected. [ductpost583]

Option file name available as a variable:

In order to help determine the precise option file used to create an NC program, the name of the option file is now available as a variable ***OptionFileName***. [ductpost687]

New drilling cycles:

PowerMILL is now able to differentiate between floating tapping and rigid tapping.

Floating tapping retains the cycle code " **4** ", and Rigid tapping uses the cycle code " **10** ".

Support has also been added for PowerMILL's " **Helical** " drilling cycles.

These uses the cycle code :- " **11** " for a ***helical cycle with a linear retract***.

and cycle code :- " **12** " for a ***helical cycle with a helical retract***.

New ***codes*** have been introduced :- ***rigid tap***, ***helical drill*** and ***helical drill retract***. [ductpost636]

Coolant code variable:

This wasn't working correctly, and has been fixed.
e.g. **M CoolantCode** will now return the appropriate output value M7 ; M8 etc. instead of M578256 [ductpost469]

Increase maximum number of toolpaths:

This has been increased from **100** to **300**. [ductpost599]

Units for option file constants:

For those few users who use **one** option file and make use of different output units it is now possible to specify the **units** for the option file **constants**.
The users who take advantage of this are those with ***units = input*** in an option file, and who use both **Metric** or **Imperial** CLDATA with that option file.
The relevant keyword is " ***option file units*** " and it can take the value " **= metric** " or " **= imperial** ".
The constants which need to be defined in the appropriate unit measure are :-
rapid feedrate ; minimum feedrate ; maximum feedrate ; tape split retract distance ; arc radius limit ; arc minimum radius ; linear axis limits ; withdrawal amount [ductpost343]

New variable *checkcounter*:

The option file variable ***counter*** is used as a counter.
However, wherever this variable is used in the option, it increments. This makes it difficult to examine the contents of the counter without changing it.
For this reason, the variable ***CheckCounter*** has been added so that the contents of the counter can be examined without incrementation. [ductpost390]

Add ability to transform workplane origin:

By **default**, the co-ordinates of the origin of a workplane (WorkplaneX etc.) are not transformed.
For many machines, this is acceptable. However, for some machines and/or controls, it is necessary to transform these co-ordinates.
Therefor where transformation of the workplane origin is necessary, set ***transform workplane origin = true*** [ductpost447]

Change of ToolSpeed when no tool change occurs:

A bug meant that ToolSpeed did not work correctly if the spindle speed changed and yet there was no tool change.
This problem has been corrected. [ductpost538]

More work on removing duplicate points:

Some more work has been done to prevent near-duplicate points causing " **noise** " in NC programs. [ductpost645]

Retrieving Word values:

The block variable " **Word[n]** " has been extended to allow a symbolic lookup of words.
Previously, if a word " ASTORE " had been defined, in order to retrieve its contents, it was necessary to find its' order number in the list of defined words (If we assume it is the **43rd** word, for the sake of argument) we would use " **Word[43]** " to obtain the contents.
This was awkward, and not very portable. The integer index could vary from option file to option file.
A new syntax has been introduced that allows the retrieval of a word without knowing the index. The contents of the word " ASTORE " would now be retrieved using the syntax " **Word{ASTORE}** ". NOTE the use of **{ curly }** instead of **[square]**[ductpost699]

Retrieving word *string* values:

A **new** block variable has been added to allow the recovery of any **strings** that have been associated with a DUCTpost word.

This variable can be indexed with an integer, as was the case for the " **Word[n]** " block variable. For example, assuming that " ASTORE " is the **46th** word to be defined, " **WordString[46]** " will extract any string associated with ASTORE.

The "symbolic" style of indexing is also allowed, so an alternative to the above would be to use " **WordString{ASTORE}** ". [ductpost700]

Go to (TOP ; DP1470 ; DP1460 ; DP1440 ; DP1430 ; DP1425 ; DP1414 ; DP1405)
(DP1364 ; DP1361 ; DP1357 ; DP1353 ; DP1350)

DUCTpost 1360

Improved precision of internal calculations:

DUCTpost now performs all of its internal calculations with a greater degree of precision. This is particularly important for obtaining good results with 5-axis linearisation [ductpost606]

DUCTpost 1359

Drilling problem with machine tool workplanes and "Horizontal machines":

Some of the drilling parameters were being incorrectly calculated when the "initial tool vector" was set to (0 1 0), this has been corrected. [ductpost669]

Make tests for equality in "if (statements)" more tolerant:

When DUCTpost checked for equality of real numbers, they were considered equal only if they were, ***exactly*** equal.

This comparison has now been made more tolerant. (So, "if (Elpos = 0.0)" will now more likely be honoured even if there is small 0.000001 difference, and it should be no longer necessary to bracket the 0.0 point with " if (Elpos > -0.0001 and Elpos < 0.0001)" [ductpost671]

Go to (TOP ; DP1470 ; DP1460 ; DP1440 ; DP1430 ; DP1425 ; DP1414 ; DP1405)
(DP1364 ; DP1361 ; DP1360 ; DP1353 ; DP1350)

DUCTpost 1357

3+2 arc reversal problem :

A problem which caused some arcs to have their sense reversed, when in 3+2 mode has been fixed. [ductpost655]

Problem with " workplane angles = apparent " for "horizontal machines":

When the " **initial tool vector** " was set to (**0 1 0**), then the workplane angles were not calculated correctly. Also, the coordinate transformations were not correct. This has now been corrected. [ductpost647]

Valid machine configurations being rejected :

DUCTpost could reject valid multi-axes machine configurations, claiming they were not valid. This has now been fixed. [ductpost438]

Allow circular arcs in particular planes to be suppressed:

Circular arcs can be supressed in each individual principal plane. For example, " Vertical " arcs can be suppressed by adding to the option file :-   [ductpost600]

suppress xz arc = **true**   default   = **false**

suppress yz arc = **true**   default   = **false**

Small circular arcs can be suppressed :

" **arc minimum radius = 0.0** " has been added, and circular arcs with a radius smaller than a value entered will be polygonised.   (e.g. arc minimum radius = 0.016) [ductpost634]

Make "tool vector output" work properly for 3+2 :

String output no longer padded with spaces :

Previously, when a string was output, the output was padded with spaces so that the output string was always the field width. This padding no longer takes place. [ductpost513]

Make sure first linearised move is correct :

The first move is linearised for safety, and a " previous " position was invented to allow this linearisation. However, sometimes this initial position was not correct.   [ductpost607]

Go to (TOP ; DP1470 ; DP1460 ; DP1440 ; DP1430 ; DP1425 ; DP1414 ; DP1405)
(DP1364 ; DP1361 ; DP1360 ; DP1357 ; DP1350)

DUCTpost 1353

Missing Tap coolant code "coolant on tap" added:

We can now make proper use of all coolant settings provided by PowerMILL. [ductpost302]

Tool Path ouput to Tool Tip or Center:

Variables for " ***tip***" or " ***center*** " coordinates has been added.
If we are using " ***tip*** " coordinates, then ***TipCoordinates*** will be **true**.
If we are using " ***center*** " coordinates then ***TipCoordinates*** will be **false**. [ductpost558]

Linearisation fix:

If the coordinates were updated in the option file, then this could lead to linearisation problems since DUCTpost didn't properly take this coordinate update into account. [ductpost589]

Misleading "anti-parallel vectors" error message:

This error could be triggered incorrectly, with **DUCT CLDATA**.
This has been fixed. [ductpost586]

Misleading error messages with DUCT CLDATA:

With **DUCT CLDATA**, the error message " ***Machining impossible, Check tool orientation*** " was often produced, even if the real cause of the problem was something completely different.
This has been fixed.
The error message has also been changed to read " ***Error: the tool vector is impossible with this machine tool configuration*** ", which should be more informative. [ductpost585]

ToolLength, CompensationToolLength:(See Ductpost 1350)

The variable " ***ToolLength*** " once again refers to the **length of the tool** when it is created.

The variable " ***CompensationToolLength*** " has been added to give the length as specified in the **length compensation** box from the NC program menu in PowerMill. [ductpost569]

A Hole Diameter variable added:

The variable " ***HoleDiameter*** " has been added to give the diameter of holes for those drilling cycles that can use this information. [ductpost593]

Cutting time calculation problem:

A change made to tidy PowerMILL 4 CLDATA uncovered a bug in DUCTpost.

This meant that the *"cutting times"* reported by DUCTpost were very misleading when DUCTpost was used with PowerMILL 4.

This has been corrected. [ductpost578]

DUCTpost had a tendency of choosing extreme angles to start:

DUCTpost 1350/1351 had a tendency to choose a starting angle that was at the extremes of the allowable range, which has now been corrected. [ductpost567]

The resetting of rotary angles after tool change:

This was a fairly long standing problem in that DUCTpost used to reset the rotary angles after a tool change, even when this was not necessary.

This no longer happens by default, and the flag **"** ***unwind at tool change*** **"** should be used to control this behaviour. [ductpost573]

Failure to linearise has now been made an error:

If DUCTpost has been asked to linearise the moves and can not do so, DUCTpost now terminates with an error message.

The previous behaviour was to issue a warning. One, rare, problem with the linearisation was also fixed, which was uncovered by this change. [ductpost525, ductpost568]

Problems with retract and reconfigure:

There was a long standing problem where DUCTpost could very occasionally get into a cycle of unending retraction and reconfiguring, which has been fixed.   This particular instance of that problem would only occur when not using linearisation for a machine with non-orthogonal rotary axes. [ductpost550]

Problems with finding angles to retract and reconfigure:

Older versions of Ductpost would sometimes claim that it was impossible to retract and reconfigure to get across angular limits, when in fact it actually was possible.   This version of DUCTpost now searches harder to find a solution. [ductpost551]

Problems resulting in no longer having " FROM " points:

A fix has been done to overcome this problem, especially where a non optimal choice of rotary angle can occur. [ductpost555]

Error messages for Multi-axes work:

These have been improved [ductpost553]

Go to (TOP ; DP1470 ; DP1460 ; DP1440 ; DP1430 ; DP1425 ; DP1414 ; DP1405)
(DP1364 ; DP1361 ; DP1360 ; DP1357 ; DP1353)

DUCTpost 1350

Filter out 5-axis duplicate points:

For 3-axis work, duplicate points have always been filtered out completely.

There has been a long-standing inconsistency in that for 5-axis work duplicate points were not filtered out, and as a consequence this could lead to lines being output where only the feedrate was output.

This inconsistency has now been removed. [ductpost60]

Correct arc direction for all 3+2 cases:

In some circumstances, for 3+2 machining the direction of the arc was not always correct.

For example, G2 may have been used where G3 was required. This has now been corrected. [ductpost494]

Multiaxis tool length:

In earlier versions of DUCTpost, the **tool length** was not properly distinguished. The **tool length** used for ***tool length compensation*** (for machines that need this), and the **tool length** used in 5-axis where the post-processor must take account of the tool length were originally considered as the same.

The "***multiaxis length***" now comes from the length specified on creation, and the "***compensation length***" comes from that specified in the NC program.

The "***compensation length***" can be accessed using "***ToolLength***", as always.

The "***multiaxis length***" can be accessed using the block variable "***MultaxToolLength***", if this is necessary. [ductpost484]

Multiaxis tool changes:

Better control has been provided of the behaviour of DUCTpost at tool changes.

Previously, there was an implicit assumption within DUCTpost that the machine did not " ***unwind*** " at a tool change.

For some machines this is not true, particularly Spindle Head units which need to come back to zero for tool changing. It is not so critical for table units to zero.

By setting the flag :-

unwind at tool change = false (**default**) to **true**

the machine will unwind, and will linearise the moves needed to unwind as much as is possible. [ductpost521]

GOHOME records:

PowerMILL 4 no longer outputs **GOHOME** records. Therefore, if DUCTpost detects that complex processing is taking place in a ***go home block*** an error will occur and some alternative means should be found to achieve the desired output. [ductpost485]

It is **recomended** that ***define block gohome xy move*** and ***define block gohome z move*** are changed to include **ONLY** the following in any option with anything different. **N ; rapid ; x coord ; y coord ; z coord ; feedrate**

Exactly opposite tool vectors:

If DUCTpost is presented with two directly opposite tool vectors, then it is impossible for the post processor to make the correct decision as to how to move between them.

Therefore, if DUCTpost is given CLDATA that has successive tool vectors that are exactly opposite, this will now cause an **error** and the post processing to terminate with an **error message**. [ductpost517]

Interpolating through a singularity:

A singularity for a 5-axis machine tool is when the tool is orientated so that whatever elevation angle is chosen, the orientation of the tool does not change relative to the workpiece.

Previously we used to deal with these by retracting the tool and rotating the singular axis to reach the desired value.

This has now been changed and the ***tool no longer retracts***, **by default**. [ductpost519]

Retract at angular limit:

When an angular limit is reached, DUCTpost has no choice but to withdraw the tool and reconfigure the rotary axes. This can be undesirable, since the resulting moves are not seen in PowerMILL.

By default, if a retract is needed because an angular limit is reached, then an ***error message*** will be given and **post processing terminated**.

The error message can be removed by setting :-

retract at angular limit = false to **retract at angular limit = true**

If you choose to allow these retractions, DUCTpost will warn you that these extra moves have been inserted. [ductpost518]

Tool type is now available:

A new variable " ***ToolType*** " has been provided that gives the type of the tool.

Detecting 5-axis moves:

A means has been provided of determining if a move is 5-axis (has a tool vector or not).

This may sometimes be helpful with the **NEW** "**joinup moves**" which are 5-axis move sequences that join together sections of 3+2 programming.

ToolPathMinY:

This was not working in the previous release, and has now been fixed. [ductpost498]

Linearisation of RAPID moves:

To avoid potential difficulties, RAPID moves in 3+2 or 5-axis are now linearised. [ductpost530]

Go to (TOP ; DP1470 ; DP1460 ; DP1440 ; DP1430 ; DP1425 ; DP1414 ; DP1405)
(DP1364 ; DP1361 ; DP1360 ; DP1357 ; DP1353 ; DP1350)