Dolphin CADCAM

Copyright © 1997-2017 Dolphin CADCAM Ltd.

This manual is copyrighted and all rights are reserved. This document may not, in whole or in part, be copied or reproduced in any form, or by any means without the prior written consent of Dolphin CadCam Ltd..

Dolphin CadCam Ltd.. Makes no representations or warranties with respect to the contents hereof and specifically disclaims any warranties of fitness for any particular purpose. The information in this documentation and the software programmes to which it refers are subject to change without notice.

Dolphin CadCam Ltd.. assumes no responsibility for any errors that may appear in this document.

Dolphin CadCam Ltd.
13, Silverwood Road,
Kilmarnock. KA3 6LS
Scotland.

Tel 01563 543989
Email info@dolphincadcam.com

Dolphin CAM Post Processor

Introduction

DCAM and DWES do not produce files which can be sent direct to a machine tool controller, instead they produce a TPD (Tool Path Data) file which must be translated by a post processor into the format required to suit a particular machine tool controller.
DCAMPPR is a general purpose post processor which translates the intermediate TPD files produced by DCAM and DWES both of which it is an inegral part. In order to achieve this the post processor first reads information specific to the machine tool controller from a PPR (PostProcessoR) file, and then as each instruction in the TPD file is read, it is converted to the correct format and written to the output file. DCAM and DWES automatically invokes the post processor when the user issues a PostProcess command. Post processor output is written to a file with the same jobname and a .PUN extension[1].

The post-processor has NO embedded machine tool knowledge and relies entirely on the instructions in the .PPR file to generate output. No output of any kind is generated except in response to an instruction within a .PPR file. The post processor does not know nor care what type of machine tool the target is. It may be used to drive any 2 or 3 axis machine tool including mills, lathes and plotters. The post-processor has no embedded rules governing what may be done at any time, and no error checking (other than for syntax within the .PPR file).

Creating Post Processors

PPR files contain all the information required to convert a Tool Path Data file into the format required by a specific machine tool controller. A PPR file contains a number of sections, each of which provides information or instructions relating to a single topic. Each section must conform to the following general structure :-

SECTION_NAME:
    data
END:

Where **SECTION_NAME:** is one of the valid section names listed below, **data** represents the contents of that section and **END:** terminates the section.

Section Name	Description
TITLE:	Used to store the machine tool and controller name(s).
WORDS:	Used to define word formats.
CYCLES:	Conveys information on how to deal with cycles.
TOOLFILE:	Defines whether or not the toolfile will be combined with the main output file
SUBROUTINE:	Defines if and where subroutines will appear in the output file
AXES:	Provides information on axis scaling, output units and origin handling.
GROUPS:	Used to define groups of mutually exclusive preparatory and miscellaneous modal codes - G and M codes.
MACROS:	Used to define macros for use in the RULES: section.
RULES:	Used to define an output rule for each cldata record type. Output is generated only in response to entries within this section !

[1] The default file extension can be changed by modifying the file DCAM.TXT or DWES.TXT as appropriate.

The TITLE Section

The TITLE section exists only to provide a description of the PPR file. It should contain the machine tool type, controller type and so on. There may be up to 5 entries may be within the TITLE: section.

```
TITLE:
    T1 = { Matchmaker VMC with Fanuc 11 controller}
    T2 = { V1.0 Author  Dolphin CADCAM September 1995 }
END:
```

The WORDS section

The WORDS: section exists to define word formats. Each word format is assigned a unique name by the user, and may be referenced by name within the RULES: section and the MACROS: section. A word format defines how a particular word of output should look in order to be acceptable to the target machine tool controller. It comprises optional literal text together with zero or more formatting characters and an optional decimal point or comma.

Suppose the block number is to comprise the character “N” followed by 1 to 4 digits. This would be written as :-

:BLOCK = {”N”DDDD}

or :-

:BLOCK = {”N”ZDDDD}

which will output N followed by exactly 4 digits.
If the controller manual describes the X axis word as X-4.3 in metric or X-3.4 in imperial this could be written as :-

:XAXIS = {”X”DDDD.ddd} {”X”DDD.dddd}

In this example each D serves as a place holder for a leading digit i.e. before the decimal point, d acts similarly for trailing digits after the decimal point and “X” causes X to be output as the first character. Two formats are supplied the first is used for metric output and the second for imperial units.

Formatting characters

Each tape word comprises optional literal text enclosed in quotation marks, combined with characters from the list below. These characters completely define the appearance of the tape word.

Character	Definition
D	place holder for leading digits
d	place holder for trailing digit
Z	insert leading zeroes to field width defined by number of D’s
S	insert leading spaces to field width defined by number of D’s
z	insert trailing zeroes after last significant digit
S	insert trailing spaces after last significant digit
+	sign positive values
-	sign positive values
0	sign zero using sign character above e.g. +0.0
I	suppress decimal point if no significant digits follow
“	delimit literal strings e.g. “CC”, “G01”
.	decimal point character
,	alternate decimal point character.
2	force last digit output to be 0, or a multiple of 2.
5	force last digit output to be 0 or 5.
A	If 2 decimal places are defined. Output angle as degrees and minutes. If 4 decimal places are defined output angles as degrees, minutes and seconds. If not present angles are output as degrees and decimal degrees

Any characters not enclosed in quotation marks are assumed to be formatting characters and will give rise to an error if they are not in the list above except that spaces are ignored unless quoted !

Formatting errors are displayed as either ?c? , where c is an unexpected format character, or **** which indicates that not enough leading D spaces were reserved to format the number.

The CYCLES section.

The CYCLES section tells the post processor how to deal with “canned cycles”. Each entry consists of a CYCLENAME followed by a CYCLETYPE. CYCLETYPE is one of EXPAND, CANNED or CALL, and informs the post processor just how the cycle referred to as CYCLENAME should be treated. i.e. it determines which cycles shall be dealt with as “canned cycles” and which shall be expanded.

Cycle Name	Description
DRILL	normal drilling cycle with or without delay
NDEEP	deep drilling cycle in which the drill is fully retracted from the hole between pecks
NPECK	peck drilling cycle in which the drill is partially retracted between pecks
TAP	tapping cycle in which the spindle is reversed to extract the tap.
TAPNOREV	Tapping cycle in which the spindle is not reversed
LHTAP	Tapping cycle using left handed tap.
LHTAPNOREV	left-handed tapping cycle without a spindle reverse.
BORE	boring cycle in which the tool is extracted at feedrate with the spindle stopped.
REAM	cycle in which the tool is extracted at feedrate
GOSPRLP	an area clearance cycle in a rectangular pocket
GOSPRLC	an area clearance cycle in a circular pocket
BOXTURN	A turning cycle defined within a rectangular area
BOXFACE	A facing cycle defined within a rectangular area
TURN	A generalised turning cycle which uses a profile to define the finished part shape
FACE	A generalised facing cycle which uses a profile to define the finished part shape
RECESSR/A	A grooving cycle executed on a diameter
TREPANR/A	A grooving cycle executed on a face
SCRCUT	An external screw cutting cycle
THREAD	An internal screw cutting cycle

Output Option	Description
EXPAND	should be used if the machine tool has no knowledge of the cycle, if this is the desired output format the cycle need not be mentioned in this section.
CANNED	should be used for all controllers where a cycle definition also causes the cycle to be executed e.g. FANUC. If CANNED option is selected then the RULES: section should contain an entry for “CANCELCYCLE” unless the cycle preparatory function is non modal.
CALL	should be used for controllers which use a define cycle preparatory function together with an execute cycle preparatory function e.g. the HEIDENHAIN conversational format . If this option is selected then the RULES: section must contain an entry for “CALLCYCLE”.

DRILL, DEEPDRILL, PECKDRILL BORE, REAM, TAP etc. are all cycles derived from the DCAM DRILL command. The post-processor checks the options programmed to decide which cycle best fits the DRILL command.

Separate cycle names are not used for cycles which include a DELAY option, instead the user must examine the post processor variable $CDELAY which will be non-zero if a DELAY was programmed in the cycle.

FACE and TURN are both derived from the TURN statement, the RADIAL / AXIAL options govern which cycle is output.

RECESSR and RESSESA, TREPANR and TREPANA all derive from the Groove command, the cycle chosen depends on the RADIAL/AXIAL option in the statement.

DCAM cycles will not always exactly match what the controller is capable of, the user should adopt a “best fit” approach, or simply not mention the cycle in this section - in which case the post processor will output an expanded version of the cycle .

Circular Interpolation

If the controller is not capable of circular interpolation then the entry ARC VECTOR must appear in the cycles section. If circular interpolation is limited to quadrant boundaries then ARC QUADRANT option should be specified. If neither option is present then 360 degree circular interpolation is assumed.

If the controller cannot move the Z axis whilst interpolating arcs, or is limited to quadrant boundaries then the entries HELIX VECTOR and HELIX QUADRANT can be used to inform the post-processor. HELIX and ARC commands can be used together, e.g.

ARC QUADRANT
HELIX VECTOR

will cause the postprocessor to output arcs (G2/G3) only when there is no change in Z value, and linear three axis moves to interpolate helices.

If linear interpolation is not supported by the controller, this must be dealt with by an output rule (see later), the post processor will not attempt to vector linear moves.

The AXES Section

The AXES: section is used to provide the post processor with information on ORIGIN handling, axis scaling and output units. All entries within this section are optional. Each keyword , if present, will turn on a particular feature within the post processor.

Keyword	Description
XYORIGIN	Translate all XY co-ordinates by the values in the ORIGIN statement. Default : do not translate XY co-ordinates.
ZORIGIN	Translate all Z co-ordinates by the Z axis value of the ORIGIN statement. Default : do not translate Z co-ordinates.
XSCALE x	Scale all X axis values by the factor x. Default x = 1.0
YSCALE y	Scale all Y axis values by the factor y. Default y = 1.0
ZSCALE z	Scale all Z axis values by the factor z. Default z = 1.0
UNITS MM	Force output to be in millimetres, irrespective of input units.
UNITS INCH	Force output to be in inches, irrespective of input units.
TURRET	Output turret datum co-ordinates - not applicable to milling machines
CENTER	Output tool centre co-ordinates

The order of transformations is :-

1. Translate tool co-ordinates to either TURRET or CENTER co-ordinates
2. Add origin value.
3. Multiply by axis scale.
4. Apply Units Conversion.

If no UNITS entry appears in this section, the post processor will assume whatever units the DCAM / DWES job was programmed in.

Lathes in which the X axis is programmed by diameter rather than radius values should use YSCALE 2.0 - in DCAM lathes are programmed in the XY plane and converted to ZX by the post-processor.

The GROUPS Section

This section allows the user to group together mutually exclusive preparatory and miscellaneous function codes (G and M codes). It is neither required nor possible to define what each function code does. One shot or non-modal function codes should not be defined at all. If function codes are grouped within this section the post-processor can do modality checking within each defined group to prevent un-necessary duplication of function codes. Function code groups are usually described in the controller's programming manual and can usually be entered exactly as they are described. e.g. :-

```
GROUPS:
        G1 = { G00 G01 G02 G03 }
        G2 = { G70 G71 }
        G3 = { G80 G81 G82 G83 G85 }
        G4 = { G98 G99 }
        M1 = { M00 M03 M04 M05 M06 }
END:
```

Codes defined within the GROUPS section may include the underscore character, e.g. _G01 _G02. On output the underscrore character is replaced by a space character. This can make the output file more legible, and some users seem to prefer this. If this technique is used then _G01 must be used in place of G01 etc. everywhere that it appears in the post-processor file. _G01 and G01 are different codes as far as the post-processor is concerned.

The SUBROUTINE section

The subroutine section defines where subroutines will be positioned in the output file, the options are

Keyword	Description
START	all subroutines will be concatenated together and placed at the start of the main output file, usually referenced by program number
EMBED	subroutines are embedded in the main output file as received, usually these are then referenced by their initial block number.
END	subroutines are concatenated to the end of the main output file

e.g. :-

```
        SUBROUTINES:
                END
        END:
```

If this section is not present then any subroutines seen by the post-processor will be expanded.

The TOOLFILE section

The TOOLFILE section defines how the toolfile - if any - is incorporated into the main output file, the options are :-

Keyword	Description
START	the toolfile is prepended to the main output file
END	the toolfile is appended to the main output file

e.g. :-

```
        TOOLFILE:
                START
        END:
```

- If this section is not present then a separate toolfile is created.
- See the FILE command and the TOOLFILE command.

The MACROS section

The MACROS section exists to enable the user to adopt a shorthand notation within the output RULES: section. The most common use of macros is to associate an internal variable name with a format word.

The syntax of a MACRO is identical to that of an a output RULE q.v. with only minor exceptions, a MACRO may NOT call itself - or any other macro - , the MACRO name must start with a # character, the MACRO should not contain any comments, otherwise MACROS may contain all else that is legal in an output RULE. (described later)

```
MACROS:
    #N = { $BLOCKNO:blockno }
    #Y = { $X:x }
    #X = { $Y:y }
    #CRC = { [ CUTCOM ? G40 / G41 / G42 ] }
END:
```

Assuming the output word :blockno has been defined as :blockno = { “N”DDD } within the WORDS section, then to output the current block number it would be necessary to enter the statement - $BLOCKNO:blockno everywhere that a block number is needed. To make the output rule more legible #N could be used in its place.

MACROs should be used to make the RULEs more legible by avoiding duplication of code. If used properly the output RULEs can end up looking almost exactly like the programming examples given in the machine tool manual.

The Output RULES section

This is the most important section of a PPR file. It is the one that produces all the output, even so it is not compulsory. A PostProcessoR file with no output rules can be used to produce an ASCII text file from the tool path data, useful when de-bugging new PPR files.

The structure of an output rule is :-

name: = { body }

Where NAME is a DCAM major word, (more correctly it is one of the words listed under the heading CLDATA Record types, which comprises DCAM major words and a number of “pseudo” major words known only to the post-processor) and body is a sequence of one or more words which comprise the actions to be done by the post processor when it encounters a cldata record corresponding to that major word. Cldata records are normally received from the cam system, however in certain circumstances the post processor will generate cldata records internally.

Each word or group of words in the body of the output rule is an instruction to the post processor. Using these instructions the post processor may do any or all of :-

- output any variable, formatted using one of the format words defined in the WORDS: section,
- output literal text, i.e. a sequence of characters enclosed in quotation marks,
- set or alter the current value of a variable,
- direct output to be written to one of a number of files, e.g. a toolfile or an operator’s instructions file,
- make decisions (using an “if” test) based on the current value of a variable.

Each block of NC output is built up a word at a time, an output rule may generate NC blocks of up to 255 characters in length and any number of NC blocks.

Modality checking is not automatic but is done only on request, any word to be output may be checked for modality and it is not necessary to declare which words in the NC output are modal, if fact any word may be treated as being both modal and non-modal at any time.

It is important to remember that output is only generated on request, nothing is output for free. The order in which NC blocks are generated is governed solely by the order in which cldata records are received, however it is possible to defer output in certain circumstances and to rearrange blocks.

The post processor uses a small vocabulary of command words to control the actions performed within an output rule. These are :-

;, set, unset, file, toolfile start, toolfile close, errmsg, if...then else endif, null, eob, map, umap

and are described below.

Comment ;

A semicolon is used to introduce a comment. Comments continue to the end of the line in which they appear, but do not include trailing **}** characters, and are totally ignored by the post processor. e.g. :-

```
:GOTO = { #N #X #Y ; Output XY co-ordinates }
```

The closing } need not be on a new line.

SET

The SET command is used to assign a new value to a variable and takes the form :-

```
SET $variable_name = expression
SET [flag_name] = expression
```

The list of valid variable names is included under the headings Floating Point variables and Integer variables. Floating point variables (those with a decimal part) always have their name preceded by a dollar sign, whilst integer variables always have their name enclosed in square brackets.

UNSET

The UNSET command is used to cancel modality on a format word or function group. A function group is one of the groups of preparatory (G) or miscellaneous (M) functions defined in the GROUPS: section.

```
UNSET:X     ; cancel modality on tape format word X
UNSET(g1)   ; cancel group modality on the modal group g1
```

UNSETALL

The UNSETALL command cancels modality on all format words and function groups. It should be used when producing a safe-start sequence e.g. at a tool change or when a new output file is opened (typically in the NEWSEC: output rule).

FILE

The FILE command is used to open a file, which will then receive all output until another FILE command is encountered. If no prefix is added then the jobname prefix is assumed . e.g.

```
FILE = PUNFILE      output to file JOBNAME.PUN (default)
FILE = TOOLFILE     output to file JOBNAME.TOL
FILE = ".XYZ"       output to file JOBNAME.XYZ
FILE = "myfile"     output to file MYFILE
FILE = "myfile.txt" output to file MYFILE.TXT
```

TOOLFILE START

This command diverts post-processor output to the toolfile JOBNAME.TOL. It is the same as entering FILE TOOLFILE

TOOLFILE CLOSE

This command diverts post-processor output to the main output file JOBNAME.PUN. It is the same as entering FILE PUNFILE

ERRMSG

The ERRMSG command causes an error message to be displayed on screen, the error message is also written to the output file.

```
if ( $X GT 350 ) then
    ERRMSG “X axis limit exceeded”
endif
```

Apart from reporting syntax errors the post processor does not do any other error checking. It is necessary to build any required checks into the output rules.

IF...THEN ... ENDIF

The IF command allows selective execution of commands based on the result of a logical expression or condition, i.e. one that yields the result TRUE or FALSE.

IF (expression) THEN commands ENDIF

Expression	Description
X EQ y	is TRUE if x is equal to y
X NE y	is TRUE if x is not equal to y
X GT y	is TRUE if x is greater than y
X GE y	is TRUE if x is greater than or equal to y
X LT y	is TRUE if x is less than y
X LE y	is TRUE if x is less than or equal to y
x AND y	is TRUE if both x and y are non zero.
X OR y	is TRUE if either x or y is non zero.

where x and y are any valid arithmetic expression or constant.

Logical expressions cannot be combined with arithmetic expressions.

- The operators AND and OR have lower precedence than EQ, NE, GT, GE, LT and LE which all have the same precedence.
- The operators AND and OR are used to combine two or more logical expressions e.g.
  IF($DELTAZ gt 0.0 AND $DELTAX ne 0.0) then ; this is a ZX move

ELSE

The ELSE command may be included in a sequence of commands between an IF and an ENDIF command.

IF (expression) THEN commands1 ELSE commands2 ENDIF

If condition evaluates to TRUE (non-zero) then the commands represented by commands1 are executed, otherwise the commands represented by commands2 are executed.

- IF commands may be nested to a maximum of 10 levels deep.
- ELSE commands always belong to the most recent unmatched IF.
- ENDIF always belongs to the most recent unmatched IF.

```
IF condition1 THEN
    commands1
ELSE
    IF condition2 THEN
        commands2
    ELSE
        commands3
    ENDIF
ENDIF
```

in the above example commands1 is executed if condition1 is true, otherwise commands2 is executed if condition2 is true. If both conditions are false (i.e. they evaluate to zero) then commands3 is executed.

NULL

NULL is the No output command and does nothing ! It exists to preserve the syntax of a selective output statement :-

```
[ SPIN ? NULL / “M03” / “M04” ]
```

in this example no output is generated in response to a SPINDLE STOP command,

It can also be used to create an empty output rule as in :-

```
CUTCOM = { NULL }
```

and is the only permitted entry in the LOOKFOR output rule.

EOB

The EOB command is used to terminate each block of NC output. Whenever the EOB command is executed, the current NC block is written to the output file, the block number is incremented, and the system prepared to construct the next block of output. Consecutive EOB commands are effectively ignored since the post processor will not output any NC blocks unless it considers that they contain useful NC words. If no EOB command is seen before the next cldata record arrives, any output that may have been generated is thrown away.

MAP

The MAP command is used to temporarily map one format to another and is best explained by example :-

```
map :XAXIS = :ZAXIS
map :ZAXIS = :XAXIS
```

The effect of these two mappings is to swap all X and Z co-ordinates, more precisely any word output using the XAXIS format word is output as if the ZAXIS format word had been specified instead, similarly any word output using the ZAXIS format word is actually output as if the XAXIS word had been specified.

UMAP

UMAP cancels all word mappings.

Output Statements

Output statements are used to generate ALL post-processor output, each statement is considered to generate a single word of output.

Formatted output

Formatted output is generated by combining a variable name together with a word format name as in **$X:XFMT** where **$X** - the X axis target position is the variable name, and **:XFMT** is a word format name defined in the WORDS: section of the PPR file. The colon serves to introduce the word format name and is mandatory. The statement $X:XFMT says output the variable X formatted using the word defined as XFMT.

Modality checking is invoked by enclosing the output statement in parentheses as in **($X:XFMT)** which means output the variable X formatted using XFMT, only if the word which would be output is different to the last word output using the XFMT tape word - OR - if modality has been cancelled using the command UNSET:XFMT.

The variable name may be replaced by an arithmetic expression, as in **$X+$XORIGIN:XFMT** which says add X to XORIGIN and output the result formatted using XFMT. Modality checking is enabled as before by enclosing the output statement in parentheses.

Modality checking does NOT occur on the variable being output, but on the format word used to output it.

Preparatory and Miscellaneous Functions

Preparatory and miscellaneous functions are output in one of two ways.

1. If they have been defined within a modal group in the GROUPS: section then the code is entered enclosed in parentheses as in **(G01)** modality checking is enabled and the code is output only if it is not the most recently output code within the group , or if the group modality has been cancelled using the UNSET command.
2. If the code has not been defined within a modal group, or if it is desired to force output of the code irrespective of modality, then the code is entered enclosed in quotation marks **"G01"**

Selective output statement

The selective output statement is used to select whichever is the appropriate word to output, depending on the current value of one of the integer variables described in appendix 1. These variables are used as status flags. The format of a selective output statement is

[variable_name ? word1 / word2 / word3 / word4]

and is interpreted as ; if the variable referred to as variable_name has a value equal to 1 output word1 , if it is 2 output word2, if it is 3 output word3 , if it is 4 output word4 . If the variable has any other value output nothing. NOTE there must not be any space between the opening square bracket and the variable name.

Any command or statement may appear in place of word1, word2,word3 and word4, including another selective output statement but NOT an IF command or a comment. The statement may be terminated after word1 or word2 if it is only required to test the first one two values of the variable. If no output is required corresponding to any particular variable value, the corresponding output word should be replaced by NULL - the no output command.

Selective output statements may be nested indefinitely, but however they are constructed only one word is ever selected for output. Exactly the same selection process may be constructed using (nested) IF ELSE ENDIF commands.

```
[CUTCOM ? (G40) / (G41) / (G42)]
```

and

```
if ( [CUTCOM] eq 1 ) then
    (G40) ; cutter radius compensation cancel
else
    if ([CUTCOM] eq 2) then
        (G41) ; cutter compensation left
    else
        (G43) ; cutter compensation right
    endif
endif
```

will both achieve the same results. Obviously the selective output command is preferable and should always be used except when more than one word of output is to be generated. To improve legibility in the output rules then either of the above formats can be replaced by a macro.

Literal text output statement

Any text enclosed within quotation marks is output exactly as it appears.

"Print this text"

Character output statement

To output a non-printing character use CHR(n) where n is the ASCII value of the character. To output a ControlC character for instance enter :-

```
CHR(3)
```

This is most often used in either the START rule or the FINISH rule to place special characters at the start or end of the NC program. e.g.

```
:FINISH = { #N "M30" eob CHR(3) }
```

These special characters can often be eliminated if the NC program is to be sent to the controller using Dolphin Systems DCOMMS communications programme.

Arithmetic expressions

An arithmetic expression may be placed anywhere that a variable name is expected, except in the variable_name position of a selective output statement.

Expressions may be formed using any variable, numeric constant , the intrinsic functions listed below and the numeric operators.

Operator precedence is used to govern the order of evaluation, which may be further controlled by enclosing parts of the expression within parentheses. If arguments are separated by operators with equal precedence the order of evaluation is undefined and no assumption may be made about intermediate results. The numeric operators in order of diminishing precedence are :-

Operator	Definition
-	unary minus e.g. -1 highest precedence
*****	Multiplication
/	Division
+	Addition
-	Subtraction - lowest precedence.

Intrinsic functions

The post processor's expression evaluator supports a number of intrinsic functions. These are written as FUNCTION_NAME(x), where x is any numeric expression. Functions currently supported are :-

Function	Description
SIN(x)	Evaluates the sine of x (x is an angle measured in radians)
COS(x)	Evaluates the cosine of x
TAN(x)	Evaluates the tangent of x. Values of x greater than 10^8 may cause an error
ASIN(x)	Evaluates the angle whose sine is x. Values of x outside the range -1.0 to+1.0 cause an error message to be printed, and x is assumed to be the nearest legal value.
ACOS(x)	Evaluates the angle whose cosine is x. Values of x outside the range -1.0 to +1.0 cause an error message to be printed, and x assumed to be the nearest legal value.
ATAN(x)	Evaluates the angle whose tangent is x. all values of x are legal.
ATANYX(y,x)	Evaluate the angle in degrees whose tangent is given by y/x
SIGN(x)	Returns 0 if x is zero, -1 if x is less than zero and 1 if x is greater than zero.
ABS(x)	Returns the value x * SIGN(x).
INT(x)	Returns the truncated integer part of X
SQRT(x)	returns the square root of ABS(x)

Numeric constants in the range -1.0 to 1.0 must have a leading zero, e.g. 0.5, trailing decimal points are not required. Floating point variable names all start with a $ character, e.g. $X.
Integer variable name all start with a [character, e.g. [CUTCOM].

All angles are measured in radians, to convert to degrees multiply by $RTODEG and to convert degrees to radians divide by $RTODEG.

The is no conversion between integer and real numbers, to output an integer value use a word format that has no decimal part, the number output is automatically rounded to the nearest integer. - the INT function q.v. returns the truncated integer part of a variable e.g. INT(1.1) = 1.0

Floating point variables

Generally speaking these are variables set by information held within received cldata records. They may be accessed, tested and set by the user at any time. Although permitted it is meaningless to access most variables if they have not been set by the current cldata record.

For example it would be pointless to output the arc centre co-ordinate variables unless the current output rule is dealing with an arc move (either a GOCLW or a GOACLW).

Variable name	Definition
$ARCTOL	The current value used when vectoring arcs. Default 0.1mm
$BARFEED	The distance entered in a barfeed command. (Turning only)
$BLOCK, $INCR.	These are the current block number and block number increment. Default values at start-up are 1 and 1 respectively. They may be set to any initial value within the “START” output rule.
$CCLDIST	The retract value used in a Peck drill cycle
$CD1, $CD2, $CD3, $CD4,CD5	These are intermediate depths specified (or implied) within the cycle. Only the first 5 intermediate depths are stored as yet. If $CNDEPTH is 1 (i.e. not a pecking type cycle) then $CD1 contains the final depth, and $CD2-$CD5 contain zero.
$CDELAY	This contains a time delay if the current cycle includes a ,DELAY,n minor word pair otherwise it is set to zero.
$CDEPTH	Contains the final depth, absolute from Z0 for the current cycle.
$CGXSTART, $CGXEND, $CGXCEN, $CGYSTART, $CGYEND, $CGYCEN, $CGANGLE, $CGRAD	These variables are updated only by the POINT CIRCLE and LINE cldata records, output by the turning cycles FACE and TURN. Using these you can output the finish part geometry for turning cycles. The X values are actually Z co-ordinates, The Y values are X co-ordinates. If YSCALE=2 is programmed, then Y values are diameters
$CLDIST	The feed change plane, default 3mm or 0.125”
$CLEARP	The clearance plane defined in the set-up dialogue.
$CNDEPTH	Contains the number of intermediate depths specified by a DEEP or, PECK drill cycle.
$CRETRACT	Contains the retract plane to be used by the cycle if the Retract option is ticked
$CUTDEPTH	The maximum cutting depth of the current tool.
$CWSURF	Contains the work surface plane as used within the current cycle.
$CXSTART, $CYSTART, $CXEND, $CYEND	Start and end position for turning cycles i.e.. the limits over which the cycle is operating. Set by TURN, FACE, RECESS, TREPAN, SCRCUT cycles.
$DELTAX, $DELTAY, $DELTAZ	This is the last commanded position expressed as incremental co-ordinates from the current tool position. There is no equivalent incremental coordinate for rotary axes.
$DEPTH.	The current depth plane.
$DISTANCE	This is the absolute distance from OLDX,OLDY,OLDZ to X,Y,Z and is updated every time the tool is commanded to move.
$FEEDANGLE, $ROUGHCUTS, $FINISHCUTS, $STARTS, $PULOFF	Values generated by the SCRCUT statement.
$FINCUT, $ZFINCUT	Finishing allowances specified by Goround, AreaClear, Turn etc.
$FPM, $FPR	FPM and FPR are set by a SelectTool command and may be modified by changing the spindle speed also. Both feed per minute and feed per rev are always available irrespective of the feedrate type specified. Whenever the spindle speed is set to 0, one of the feedrates may also be set to 0 by the post processor.
$FZMOD	The percentage vale of the XY feedrate to be used for Z negative moves.

$GOSPRLX, $GOSPRLY, $LENGTH, $WIDTH, $ANGLE, $NOCUTS $DIAMETER	The pocket cycles update these variables, $GOSPRLX,$GOSPRLY is the centre of the pocket, $LENGTH, $WIDTH and $ANGLE define a rectangular pocket, $DIAMETER a circular pocket. $NOCUTS is the number of passes used to achieve the final depth
$JOBTEXT	The only string variable, initially it contains the part number, it is also used to store Remarks entered into the program. To output this variable just enter $JOBTEXT without a format word.
$LASTOOL	Previous tool number
$LIPRAD, $ROOTRAD	Lip and root fillet radius values used in RECESS and TREPAN cycles. Negative values indicate a chamfer rather than a fillet.
$MAXBLOCKS	The maximum number of blocks to be written to the output file. If the NC program requires more than $MAXBLOCKS blocks then the output file is split into 2 or more files. The files are numbered sequentially using the first 5 characters of the job name followed by a 3digit index e.g. myprog.pun, mypro001.pun mypro002.pun and so on. The output rules ENDSEC and NEWSEC must be present.
$MAXSPIN	Is the maximum permitted spindle speed when constant surface speed has been programmed.
$NEXTOOL	Next tool number - tool pre-selection.
$OLDX, $OLDY, $OLDZ	This is the tool position prior to executing the last commanded move. X - OLDX = DELTAX $OLD(XYZ)1 - lathe rear turret tool position $OLD(XYZ)2 - lathe front turret tool position.
$ORIGINX, $ORIGINY, $ORIGINZ	The origin position defined in the set-up dialogue
$PI	Initialised by the post-processor to 3.14159
$PITCH, $LEAD, $TPI	The pitch , lead and thread per inch value calculated from the Scewcut command .
$PRECISION	If $PRECISION is set to a value other than 0 in either the INIT: or START: rule, the post processor modifies it's arc splitting routine (used only if the CYLES: section contains the entry ARC QUADRANT) and adjust arc end points which are close to quadrant boundaries to be exactly on the quadrant boundary. Close is defined to be +/- (1 *10-$PRECISION) radians.
$PROGNO	This is the programme number
$RANGE	The spindle speed gear range (default is 1)
$RTODEG	Converts from radians to degrees
$SPINDLE	Is the current spindle speed in revolutions per minute. Spindle direction is stored elsewhere. (see SPIN)
$SUBID, $SUBBLOCK, $ENDBLOCK, $REPEATS	These are maintained by the SUBROUTINE, END and WORK cldata records. $SUBID identifies the subroutine, $REPEATS is the number of times it is to be called. $SUBBLOCK and $ENDBLOCK contain the start and end block numbers of the subroutine. The subroutine can be called by either its program number ($SUBID) or its starting block number ($SUBBLOCK).
$SURF	Is the programmed constant surface speed in either meter/min or feet/min.
$TDIM1, ..., $TDIM10	These are the tool dimensions of the current tool. There interpretation depends on the tool type, (refer to the chapter Using DCAM for more details).
$TOOLNO, $TLCNO, $CRCNO	These are set by a SelectTool command. TOOLNO - Tool number CRCNO - cutter radius compensation register TLCNO - tool length compensation register
$USR1, ..., $USR15	Are 15 user defined variables. They may be used for any purpose.$USR10 to $USR15 are set by the 6 parameter values used in a subroutine call
$WSURF	The current air/metal interface plane
$X, $Y, $Z, $A, $B	This is the last commanded position, i.e. the target co-ordinate of the current GOTO/ GOACLW/ etc. X,Y and Z are updated by each and every DCAM command that implies tool motion. X,Y and Z are always absolute. A and B rotary axis coordinates are output only if the RotaryAxis command was used in DCAM, and replace either Y or X linear coordinates Lathe co-ordinates are stored in X and Y corresponding to Z and X lathe co-ordinates.

$XCEN, $YCEN, $ZCEN, $ARCRAD, $STRANG, $INCANG, $ENDANG	These variables contain the (absolute) co-ordinates of the centre of the last arc move. ARCRAD contains the radius of the last arc move. They are only valid if the current cldata record is GOACLW or GOCLW. $STRANG and $ENDANG are tangents (in degrees) of the start and end of the arc, $INCANG is the angle subtended by the arc.
$XSAFPOS, $YSAFPOS, $ZSAFPOS.	The Safe position defined in the set-up dialogue, they are unaffected by axis scaling and ORIGIN values. $(XYZ)SAFPOS - lathe- safe position for the current tool, $(XYZ)SAFPOS1 -lathe- safe position for the rear turret $(XYZ)SAFPOS2 -lathe- safe position for the front turret.
$XSET, $YSET	The tool tip offsets corresponding to the currently active tool.
$XTLOAD, $YTLOAD, $ZTLOAD	The tool change position defined in the set-up dialogue - unaffected by axis scaling. $(XYZ)TLOAD1 - lathe rear turret tool change position $(XYZ)TLOAD2 - lathe front turret tool change position.
$ZERO	The variable $ZERO causes all other $ variables which have a smaller absolute value to be set to zero. The default value for $ZERO is 0.00001, any variable with a smaller value is considered to be 0.0.

Integer variables (status flags)

Generally speaking these are variables set by either by information held within received cldata records, or they are set (and unset) by the post-processor in response to certain combinations of conditions and may be considered to be status flags. They may be accessed, tested and set by the user at any time. Care must be exercised when setting these variables since each value they hold has a pre-defined and precise meaning. Normally they will have a value between 0 and 3. A value of 0 means that the post-processor has seen nothing to set the value of the variable, that is, it is UNSET or meaningless.

The format for accessing a variable is :-

[variable_name]

if the variable is being accessed from within an expression, or :-

[variable_name ? action1 / action2 / action3]

if the variable is being accessed with a selective output statement.

Flag name	Definition
[BARFEED]	Indicates status of last barfeed command :- 0 undefined 1 Bar feed device retracted 2 Bar feed device fed forward (distance is in $BARFEED)
[CATCHER]	Indicates status of last part catcher device command 0 undefined 1 Closed – retracted 2 Open - ready to collect component
[CDELAY]	Indicates whether or not the current (DRILL) cycle contains a delay option. 0. no cycles seen yet 1. current cycle command does NOT contain a delay option 2. current cycle command contains a delay option the duration of the delay is stored in variable $CDELAY
[CHUCK]	Indicates status of last chuck command (Turning only) 0 Undefined 1 Chuck OPEN 2 Chuck CLOSE
[CLIMB]	Climb is set by Pocket cycle commands 0 No pocket cycles yet 1 Climb milling 2 Upcut milling
[CNOREV]	Indicates whether or not a tap must be reversed 0 no tapping cycles seen 1 tap must be reversed 2 do not reverse tap.

[COOLANT]	Indicates status of last coolant command :- 0. no coolant commands received 1. Coolant OFF 2. Coolant ON
[CRETRACT]	Indicates for the current cycle whether the tool should be traversed at the current feed change plane or the current clearance plane 0. no cycle seen 1. traverse at feed change plane (return to R plane) 2. traverse at clearance plane (return to initial plane) the traverse plane for use by a cycle is stored in the variable $CRETRACT
[CUTCOM]	Used to tell the post processor whether or not to apply cutter radius compensation when contour milling or profile turning. 0. no GORND or PROTRN records received yet. 1. cutcom OFF e.g. G40 2. cutcom LEFT e.g. G41 3. cutcom RIGHT e.g. G42
[EXTERNAL]	Indicates whether the last Threadcut / Screwcut command is Internal or External 0 Undefined 1 External 2 Internal
[FEEDAX]	Indicates which axis feedrate move is in. 0. no feedrate records have been received 1. Feedrate move is in XY plane, or Z positive direction 2. Feedrate move is in Z negative direction.
[FEEDTYPE]	Indicates how the current feedrate was specified within the DCAM job. 0. no feedrate records have been received. 1. feedrate was specified in units per minute 2. feedrate was specified in units per rev.
[GEOMETRY]	Normally this flag has the value 0, and has no influence on the post processor. Setting it to the value 1 in either the INIT: or START: rules, will cause the post processor to re arrange the order in which the geometry associated with the TURN and FACE canned cycles is seen by the output rules. 0. output geometry before canned cycle 1. output the geometry after the canned cycle definition
[OPSKIP]	Indicates whether or not block skip codes should be output 0 undefined 1 Block skip is OFF 2 Block skip is ON
[PITCHTYPE]	The type of pitch programmed on a Screwcut command 0 No screw cutting commands seen yet 1 PITCH 2 LEAD 3 TPI
[PSPROG]	PSPROG is set by several commands and indicates whether or not part surface programming has been requested. 0 Undefined 1 Part surface programming requested 2 Offset tool path requested.
[RAPID]	Is used to indicate whether the next move is to be executed at feedrate or rapid feed. 0. no cldata records received yet. 1. move is at feedrate e.g. G01 2. move is at rapid feed. e.g. G00 The [RAPID] flag is set to a value of 1 by the cldata record which follows the RAPID command, that is RAPID is only ever set to a value of 2 for the duration of one cldata record.
[ROTAXIS]	Indicates whether or not a rotary axis is currently enabled. It always has a non zero value. 1. Rotary axis conversion disabled. 2. A axis enabled, Y coordinates converted to A. 3. B axis enabled, X coordinates converted to B

[SETDIR]	Tool setting direction from the DefineTool command :- 0 NONE 1 RADIAL 2 LEFT 3 RIGHT
[SETPOS]	Tool setting position from the define tool command 0 NONE 1 REAR 2 FRONT
[SPIN]	Indicates spindle direction. 0. No spindle speed set 1. Spindle stop e.g. M05 2. spindle CLW e.g. M03 3. SPINDLE ACLW e.g. M04
[SPINTYPE]	Spindle speed type 0 Undefined 1 Constant Spindle speed 2 Constant Surface speed.
[TOOLTYPE]	Contains a number representing the current tool type. 0 No tool definitions have been received. 07 Current tool is a boring bar 16 Current tool is a drill 38 Current tool is an endmill 55 Current tool is a reamer 66 Current tool is a tap 68 Current tool is a threading tool. 90 Current tool is a turning or facing tool 99 Current tool is a groove or partoff tool.
[TURRET]	Contains the current turret number :- 1 Default or first rear turret 2 First front turret
[UNITS]	Contains the units specified in the set-up dialogue 0 Units data not yet received by the post 1 metric units – mm 2 imperial units – inch

CLDATA record types.

If DCAM is started with the /d (for debug) option, an ASCII readable version of the cldata is produced. All cldata records for both milling and turning are correctly translated, but you should note that the translation is affected by the settings in the CYCLES section. The listing is written to a file "jobname.ASC".

All incoming cldata records will attempt to invoke an output rule whether or not it is one of the cldata record types listed below, but only the following cldata records actually update the post processors internal data (and can therefor produce useful NC output) The names of variables and flags affected by each cldata record type is also listed.

CLData Record types	Post-processor variables and flags updated by the cldata record.
ARCTOL:	$ARCTOL
BARFEED:	[BARFEED], $BARFEED
BOXTURN: BOXFACE: TREPANA: RECESSR:	$XSTART , $DSTART , $XEND , $DEND
CATCHER:	[CATCHER]
CHUCK:	[CHUCK]
CIRCLE:	$GXSTART, $GYSTART, $GXCEN, $GYCEN, $GXEND, $GYEND, $GRADIUS
CLDIST:	$CLDIST
CLEARP:	$CLEARP
CUTCOM:	[CUTCOM]
CUTDEPTH:	$CUTDEPTH
DELAY:	$DELAY , $CYCLETIME ,[DELAY]

DEPTH:	$DEPTH
DRILL: DRILLDWELL: NDEEP: NPECK: REAM: BORE: TAP: LHTAP: TAPNOREV: LHTAPNOREV	$CWSURF , $CDEPTH , $CCLDIST , $CRETRACT , $CDELAY , $CNDEPTH $CD1,.. $CD5 [CDELAY] , [CRETRACT]
FEDRAT:	$FPR, $FPM, $FZMOD
GOCLW: GOACLW:	$X , $Y , $Z , $OLDX , $OLDY , $OLDZ $XCEN , $YCEN , $ZCEN , $ARCRAD $DELTAX , $DELTAY , $DELTAZ $DISTANCE , $CYCLETIME, $STRANG, $ENDANG, $INCANG
GOHOME:	[TURRET] , $OLDX , $OLDY , $OLDZ , $X , $Y , $Z , $DELTAX $DELTAY , $DELATZ , $DISTANCE
GOSPRLC:	$GOSPRLX, $GOSPRLY, $DIAMETER, $NOCUTS, [CLIMB]
GOSPRLP:	$GOSPRLX, $GOSPRLY, $LENGTH, $WIDTH, $ANGLE, $NOCUTS [CLIMB]
GOTO: SCRCUT:SCREWING:	$X , $Y , $Z , $OLDX , $OLDY , $OLDZ $DELTAX , $DELTAY , $DELTAZ, $DISTANCE, $CXSTART, $CYSTART, $CXEND, $CYEND
LINE:	$GXSTART, $GYSTART, $GXEND, $GYEND, $GANGLE
ORIGIN:	$ORIGINX , $ORIGINY, $ORIGINZ
PARAMETER:	$USR10 to $USR15 - subroutine parameter values
POINT:	$GXSTART, $GYSTART
PRESEL:	$NEXTOOL - tool pre-selection
PROGNO:	$PROGNO
RAPID:	[RAPID]
RETRACT:	$Z , $OLDZ , $DELTAZ , $DISTANCE
SAFPOS:	$XSAFPOS , $YSAFPOS , $ZSAFPOS , $XOLDtur , $YOLDtur , $ZOLDtur , $XSAFPOStur , $YSAFPOStur , $ZSAFPOStur
SCRCUT: & THREAD:	$XSTART , $DSTART , $XEND , $DEND , $PITCH , $TPI , $LEAD , $FEEDANGLE , $ROUGHCUTS , $FINISHCUTS , $STARTS , $PULOFF , [PITCHTYPE]
SELCTL:	$TOOLNO , $CRCNO , $TLCNO , $X , $Y , $Z , $OLDX , $OLDY , $OLDZ , $DISTANCE , $DELTAX , $DELTAY , $DELTAZ , $CYCLETIME, $LASTOOL [TURRET]
SETOOL:	$XSET , $YSET , [SETPOS] , [SETDIR]
SPINDLE:	$SPIN , [SPIN] , FPR , FPM , [SPINTYPE] , $RANGE
SUBROUTINE: END: WORK:	$SUBID, $REPEATS, $SUBBLOCK, $ENDBLOCK
TLOAD:	$XTLOAD , $YTLOAD , $ZTLOAD , $XTLOADtur , $YTLOADtur , $ZTLOADtur
TOOL:	$TDIM1,... $TDIM10 , [TOOLTYPE]
TURN: FACE:	$XSTART, $DSTART, $XEND, $DEND,
UNITS:	[UNITS]
WSURF:	$WSURF , $DEPTH , $CLDIS

Pseudo cl-data records

The following are not DCAM cldata record types, nevertheless output rules may be written for them. They may be considered to correspond to cldata generated by the post processor itself.

INIT:

The INIT rule is processed exactly once when the post processor first reads the PPR file, and before any cldata records are received. It is important to use the INIT rule to set post processor variables that may be accessed by DWES, and not the START rule.

START:

If a START rule is present the post processor will execute it exactly once, after the INIT: rule and before processing any cldata records.

NEWSEC:

The NEWSEC rule is called when the post-processor starts a new output file (i.e. more than $MAXBLOCKS NC blocks have been written to the current output file) It should produce NC blocks to

take the machine tool to the position that existed at the end of the previous output file, i.e. it should output the current tool, coolant, feedrate etc. to ensure a safe start condition. N.B. the command UNSETALL can be used to cancel all word and group modalities.

ENDSEC:

The ENDSEC rule is called when the post-processor is about to close the current output file and open a new one (i.e more than $MAXBLOCKS NC blocks have been written to the current output file). The rule should retract the tool to a safe position and stop the machine tool so that the next program can be loaded and run to continue machining the part.

CALLCYCLE:

Variables are updated as for a GOTO record. CALLCYCLE is generated at each XYZ co-ordinate that a drill type cycle is to be executed, and is required only if any cycle defined in the CYCLES: section is defined with the keyword CALL.

CANCELCYCLE:

No variables are updated. CANCELCYCLE is required only if any cycles mentioned in the CYCLES: section are defined with the keyword CANNED

RESYNCH:

Once the last cldata record has been processed, the post processor will attempt to execute the RESYNCH rule (if present) $TOOLNO times. This is to enable tool carousel re-orientation commands to be output. If only one re-orientation command is required irrespective of the number of tools used then the RESYNCH rule should include the command :- SET $TOOLNO = 0

RENUMBER:

Renumber rule causes all blocks written to the output file to be renumbered. To use this effectively the original format word for the block number should output a literal string containing the format word to be used when blocks are re-numbered. e.g.

:BLOCK = { “N\DDDD\” }.

LOOKFOR:

The LOOKFOR rule is unique in that it cannot be used to generate any output, its sole purpose is to trigger the post processors look ahead function. It use has been largely superseded by other functions within the post processor. If the rule is present the post processor scans the output file looking for the sequence <<n>>, where n is an integer number, if it finds such a sequence, it then searches for a sequence >>n<<, and if it finds it, it replaces <<n>> with the remainder of the block following >>n<<.

- If any <<n>> has no matching >>n<< then it is silently removed from the output file.
- All blocks containing >>n<< are removed from the output file whether or not they have been matched to a corresponding <<n>> sequence.
- Multiple substitutions per block are permitted, but each replacement (i.e. >>n<<) must be in a separate block.
- The RENUMBER rule should be used to tidy up the output file.

FINISH:

If a FINISH rule is present the post processor will execute it exactly once, after all other processing is complete.

Post Processing for Dolphin Wire Erosion System

DWES generates a number of cldata records, and uses post processor variables, not normally used by DCAM, unlike DCAM the post-processor is an integral part of DWES, and the PPR file is read before any cldata records are generated. This arrangement allows DWES to respond to machine tool specific information and configures the program to use only features supported by the target machine tool.

Additional post processor variables

Variable name	Usage
$ANGLE	this should be set in the INIT rule to the maximum wire angle permitted by the machine tool
$ARCTOL	Max deviation from true arc when vectoring.
$DELTAP	Change in wire angles
$DELTAQ	Change in wire angles
$DELTAU	Distance moved by top wire guide
$DELTAV	Distance moved by top wire guide
$GANGLE	this should be set in the INIT rule to the maximum arc angle allowed for conic moves.
$OLDP, $OLDQ	Current wire angles
$OLDU	Top wire guide current position
$OLDV	Top wire guide current position
$P	Angle of wire normal to cutting direction in degrees from vertical, at target UV co-ordinate
$Q	Angle of wire parallel to cutting direction in degrees from vertical, at target UV co-ordinate
$U	Top wire guide target position parallel to X axis
$UCEN	Top wire guide arc centre, parallel to X axis
$UINT, $VINT	The intersection point (in the UV plane) of lines tangent to the start/end of an arc.
$UVRAD	Top wire guide arc radius.The bottom wire guide co-ordinates are in X, Y, XCEN, YCEN
$V	Top wire guide target position parallel to Y axis
$VCEN	Top wire guide arc centre, parallel to Y axis
$XINT, $YINT	The intersection point (in the XY plane) of lines tangent to the start/end of an arc.
$XYRAD	Bottom wire guide arc radius
$ZERO	This variable has the effect of zeroing any $variable which has a smaller value then the one assigned to $ZERO. The default value assigned to $ZERO is 0.0001. It can be assigned any positive value greater than 0.00001.

Additional Post processor Flags

The flags described below with the exception of XYARC and UVARC, must be set to a non-zero value in the INIT: rule in order to enable the feature in DWES and thus to generate the correct cldata.

For any feature which is not enabled, DWES will attempt to imitate the feature by interpolation,

Flag name	Definition
[COMPOF]	When set to 1 DWES will output CUTCOM OFF after the retract move Default - output CUTCOM OFF before the retract move.
[COMPON]	When set to 1 DWES will output CUTCOM records before the approach move. default - turn CUTCOM on after the approach move.
[FOURAX]	Set to 1 to indicate that the controller has 4 axis capability
[ISORAD]	controller can interpolate iso radius
[LINRAD]	controller can interpolate arc and line simultaneously
[LSTSPN]	When set to 1, The current geometry record is the last on the shape to be cut, and will be followed only by the return move to the pilot hole.
[NOWIRE]	Used in conjunction with SUBRUT, when set to 1 will cause wire threading and wire cutting commands to be EXCLUDED from any subroutines (i.e. they will become part of the main programme). default - wire threading is included within the subroutine

[NXTSPN]	Indicates the type of geometry record that will follow the current record. Possible values are :- 0 The next record type is unknown 1 The next geometry record will be LINLIN 2 The next geometry record will be STDCON 3 The next geometry record will be ISORAD 4 The next geometry record will be PRGRAD 5 The next geometry record will be ARCARC 6 The next geometry record will be LINARC
[PRGRAD]	Controller can interpolate programmable radius
[PRPTST]	Approach move must be perpendicular
[RADRAD]	Controller can interpolate top and bottom arcs simultaneously
[STDCON]	Controller can interpolate standard conic
[SUBRUT]	When set to 1, will cause DWES to output each contour to be machined as a separate sub-routine. default – don't output subroutines. In order to use subroutines the PPR file must contain a SUBROUTINES section to define where the subroutines should be output (i.e. before or after the main programme), if it is missing then the post processor will ignore all subroutine cldata records. Additionally there must be output rules provided for SUBROUTINE:, END: and WORK: cldata records.
[UVARC]	Motion indicator for top wire guide. The following values are possible for both XYARC and UVARC :- 0 rapid motion 1 linear motion at feedrate 2 clockwise interpolation 3 counter clockwise interpolation. When motion is at rapid the RAPID flag is also set, and both XYARC and UVARC will bet set to 0. At all other times XYARC and UVARC may contain different values. e.g., to set the XY axis motion word use :- [RAPID ? [XYARC ? G01 / G02 / G03] / G00]
[XYARC]	Motion indicator for bottom wire guide

Additional output rules

These output rules, together with INIT, START, FINISH and possibly LOOKFOR are all that need to be implemented to create a post-processor for a wire erosion machine.

Cldata record type	Description
LINLIN	a linear move in XY and UV axes
RAPLIN	a rapid linear move
APRLIN	an approach move to contour
RETLIN	the retract move from the contour
STDCON	an arc move in XY and UV axes, in which XCEN,YCEN is vertically below UCEN,VCEN, and which is an arc of a right cone.
ISORAD	an arc move in which the XYRAD and UVRAD are the same forming an arc on a cylinder
PRGRAD	an arc move in which the top and bottom radii differ and have differing centre points, forming an arc of an inclined (elliptical) cone
LINARC	a move in which XY motion is linear and UV motion is an arc, or vice versa, the case is determined by examining the flags XYARC and UVARC
ARCARC	both top and bottom movements are arcs, the general case of the conic moves
THRDWR	Instruction to thread wire (at pilot hole position)
ANGWIR	Instruction to change wire angle without movement in the XY plane.
CUTWIR	Instruction to cut the wire (prior to rapid move)
PPFUN/100	Provides information to the post-processor concerning the state of machining

Post processors supplied

It should be noted that the post-processor supplied with the Dolphin CAD/CAM system have mostly been developed in conjunction with end-users, and therefor have been written to produce output according to a particular client's wishes.

Although great effort has been made to ensure that these post-processors produce output suited to the target machine tool / controller, it can never be guaranteed that the post-processor will be correct in all cases. Most control systems are fitted to a wide range of machine tools and even when fitted to machines from the same manufacturer, will have different options implemented - tool change sequences being a case in point.

Neither can Dolphin CADCAM Systems accommodate all the known variations in style and local practice, the user should feel free to experiment with the post-processors in order to provide himself with a tailored solution.

NAME	Milling machines
M_145	Heidenhain 141 / 145
M_BOSCH	Wadkin drill / router with Bosch CC100
M_BOSS60	Bridgeport Series 1CNC with BOSS6.0
M_BOSS71	Bridgeport Series1 CNC with BOSS 7.1
M_BRI6	Bridgeport Series1 CNC with BOSS 3, 4, 5
M_CRUS2M	Anilam Crusader 2
M_DCAD	Converts toolpath to DCAD input file
M_F0MF	Leadwell MCV/OP with Fanuc 0MF
M_F0TRIA	Denford Triac with Fanuc 0MF
M_F6MB	Fanuc 6MB - not machine specific
M_FP2	Deckel Dialogue 2
M_FP4AR	Deckel Dialogue 4
M_H131	Heidenhain 131 / 135 - vectored arcs
M_H155B	Bridgeport Interact 1 with Heidenhain 155
M_H155M	Matchmaker with Heidenhain 155
M_H155T	Matchmaker with Heidenhain 155 - auto tool change
M_H350	Bridgeport BPC720 with Heidenhain 355 - calls PGM1 at tool change
M_HURCO	Hurco KMB 1X
M_MAH432	Maho 432 with Philips controller
M_MC1M	Moog with Mark Century 1M+
M_MICON	21/2 axis machine with MICON controller
M_MOOG	Moog Hydrapath.
M_MX2	XYZ Prototrak with MX2 / MX3 controller
M_MX3	XYZ Prototrak with MX3 controller
M_NUSTEL	Actimill with PC based MINT controller.
M_P2800	Acton PowerMill - Posidata 2800 controller
M_PANCUT	Newing Hall Engraving machine with PNC2 controller
M_PNC2	Denford EasyMill - PNC2 controller
M_PNC3	Denford EasyMill - PNC3 controller
M_SIEM6B	Siemens Sinumeric 6MB - Uses S1 through S56
M_SIN3MB	Siemens Sinumeric 3MB
M_TRIAC	Denford Triac milling machine

NAME	Turning machines
T_3NE300	Hitachi Seiki 3N3/300 with Fanuc controller
T_3NE300	Hitachi Seiki 3NE300 with Fanuc controller
T_BOXFORD	Boxford ANC M568/1 - Diameter programming
T_CRUS2L	Anilam Crusader 2L
T_DSG	Dean, Smith & Grace - Philips 6682 controller
T_EMCO	Emco training lathe
T_F0T	Fanuc OT Diameter programming
T_F3TC	Colchester with F3T - no cycles

T_F6T	Mori-Seiki with Fanuc 6T
T_GE1050	AL10 with GE1050 controller
T_GE2000	Moog with GE2000 - expanded cycles
T_LX2	XYZ ProtoTRAK Lathe with LX2 controller
T_OLIVE	Antares lathe with Olivetti control - Dia. Programming
T_PNC3	Denford EasyTurn - PNC3 controller
T_S810T	Weiler 160CNC with Sinumeric 810, Diameter programming
T_TRAUB	Demo.

NAME	Wire erosion machines
W_AGIE	Agie DEM315 2 & 4 axis
W_CH530	Charmilles 530 2 axis
W_CHARM	Charmilles Robofil 4020 2 & 4 axis
W_DCAD	Converts wire path to DCAD input file
W_FW0	Fanuc 0W 2 axis only
W_FWECL	Fanuc 2 & 4 axis
W_JAPAX2	Japax 2 axis
W_M2AXIS	Mitsubishi Wire EDM 2 axis
W_M4AXIS	Mitsubishi Wire EDM 4 axis
W_SEIBU	Seibu Wire EDM
W_SODIC2	Sodic wire EDM 2 axis
W_SODIC4	Sodic wire EDM 4 axis

NAME	Other machines
G_ABRADL	Moore jig grinder with Allen Bradley controller
G_CHMILL	Charmille jig grinder with Andrews controller
G_GE1050	Moore jig grinder with GE1050, C axis control
M_PRECIX	High speed engraver - very strange - uses extended HPGL
P_TORNAD	Turner & Fowle Plasma cutter with Tornado controller

A Simple post processor

The postprocessor shown below illustrates most of the programming instructions described in this section of the manual.

```
TITLE:
      :T1 = { MOOG Hydrapath, Copyright : © 1997 Dolphin CadCam Ltd.. }
END:

WORDS:
      :N = {"N"DDDDZ}
      :X = {"X"DDDD.ddd} {"X"DDD.dddd}
      :Y = {"Y"DDDD.ddd} {"Y"DDD.dddd}
      :Z = {"Z"DDDD.ddd} {"Z"DDD.dddd}
      :R = {"R"DDDD.ddd} {"R"DDD.dddd}
      :W = {"W"DD.dd}    {"W"D.ddd}
      :I = {"I"DDDD.ddd} {"I"DDD.dddd}
      :J = {"J"DDDD.ddd} {"J"DDD.dddd}
      :F = {"F"DDDD}     {"F"DDD.d}
      :S = {"S"DDDD}
      :T = {"T"DD}
      :P = {"P"DD}
      :E = {"E"DD}
      :D = {"D"DD.d}
END:

MACROS:
      #N = { $BLOCK:N }
      #X = { ($X:X) }
      #Y = { ($Y:Y) }
      #Z = { ($Z:Z) }
      #R = { ($CCLDIST:R) }
      #W = { ($CD1:W) }
      #I = { $XCEN:I }
      #J = { $YCEN:J }
      #F = { if(($FPM lt $USR1)OR($FPM gt $USR2))then
           ERRMSG "Feedrate out of range"
       Endif
       [RAPID ? ($FPM:F) / NULL] }
      #S = { if( [SPIN] gt 1 ) then
              if( ( abs($SPINDLE) lt 50 ) OR ( abs($SPINDLE) gt 4500 ) ) then
                  ERRMSG "Spindle speed out of range"
              endif
              $SPINDLE:S
       endif }
      #P = { $PROGNO:P }
      #D = { $DELAY:D }
      #RAP={ [RAPID ? (G01) / (G00)] }
      #CRC={ [CUTCOM ? (G40) / (G41) / (G42)] }
      #T = { $TOOLNO:T }
      #RP= { [CRETRACT ? (M81) / (M82)] }
END:

AXES:
      XSCALE 1
      YSCALE 1
      ZSCALE 1
END:

GROUPS:
      :G1 = { G90 G91 }
      :G2 = { G00 G01 }
      :G3 = { G40 G41 G42 }
      :G4 = { G70 G71 }
      :G5 = { G80 G81 G82 G83 G84 G85 }
      :M1 = { M03 M04 M05 }
      :M2 = { M07 M08 M09 }
END:
```

```
CYCLES:
     DRILL       CANNED
     DEEPDRILL   CANNED
     PECKDRILL   CANNED
     TAP         CANNED
     BORE        CANNED
     HELIX      VECTOR
END:

RULES:
    :INIT   = { ; machine startup  condition
                set $MAXBLOCKS = 1000   ; split output files after 1000 blocks
                (G00) (G17) (G40) (G80) (G90)(M09) (M05) }
    :START  = { "%" eob set $BLOCK = 5 set $INCR = 5 }
    :PROGNO = { #N #P eob }
    :UNITS  = { if([UNITS] eq 1) then
                    set $USR1 = 8    ; min metric feedrate
                    set $USR2 = 8750 ; max metric feedrate
                else
                    set $USR1 = 0.3  ; min inch feedrate
                    set $USR2 = 350  ; max inch feedrate
                endif
                #N [UNITS ? (G71) / (G70)] eob
                #N (G17) eob #N (G90) eob  }
    :TLOAD  = { #N "G99" $XTLOAD:X $YTLOAD:Y $ZTLOAD:Z eob }
    :ORIGIN = { #N "G92" $TABLOFF:E $ORIGINX:X $ORIGINY:Y $ORIGINZ:Z eob }
    :SELCTL = { #N #T "M06" eob }
    :INSERT = { #N "(" $JOBTEXT ")" eob }
    :SPINDLE= { #N #S [SPIN ? (M05) / (M03) / (M04) ] eob }
    :COOLANT= { #N [COOLANT ? (M09) / (M08) ] eob }
    :GOTO =   { #N #RAP #CRC #X #Y #Z #F eob }
    :GOCLW =  { #N "G02" #X #Y #I #J #F eob unset (G2) }
    :GOACLW=  { #N "G03" #X #Y #I #J #F eob unset (G2) }
    :GOHOME = { #N (G00) $ZSAFPOS:Z eob #N $XSAFPOS:X $YSAFPOS:Y eob }
    :RETRACT= { #N (G00) #Z eob }
    :STOP   = { #N "M00" eob }
    :OPSTOP = { #N "M01" eob }
    :DRILL  = { #N [CDELAY ? (G81) / (G82)] #X #Y #R #Z #F #RP eob }
    :NDEEP  = { #N (G83) #X #Y #R #W #Z #F #RP eob }
    :BORE   = { #N (G85) #X #Y #R    #Z #F #RP eob }
    :TAP    = { #N (G84) #X #Y #R    #Z #F #RP eob }
    :NPECK  = :NDEEP
    :CYCLEOFF={ #N (G80) eob unset (G2) unset :R unset :W unset (M3)}
    :FINISH = { #N "M30" eob "%" eob }
    :ENDSEC = { #N (G00) $ZSAFPOS:Z eob #N $XSAFPOS:X $YSAFPOS:Y eob
                #N (M09) eob
                #N "M30" eob "%" eob  }
    :NEWSEC = { "%" eob unsetall
          #N #P eob
          #N [UNITS ? (G71) / (G70)] eob
          #N (G17) eob
          #N (G90) eob
          #N "G99" $XTLOAD:X $YTLOAD:Y $ZTLOAD:Z eob
          #N "G92" $TABLOFF:E $ORIGINX:X $ORIGINY:Y $ORIGINZ:Z eob
          #N #T "M06" eob
          #N #S [SPIN ? (M05) / (M03) / (M04) ] eob
          #N (G00) #X #Y eob
          #N (G00) [UNITS ? ($Z+3):ZAXIS : ($Z+0.1):ZAXIS) eob
          #N (G01) #Z #F eob
          #N [COOLANT ? (M09) / (M08) ] eob  }
END:
```