

ArtiosCAD 16.1

Using the General Numeric Controller Driver

02 - 2018

Contents

- 1. Location and Copyright.....5**
- 2. Using the General Numeric Controller driver..... 7**
 - Before You Start.....9
 - About Your Equipment..... 9
 - About Your NC Controller.....10
 - Example - A Sample Maker with a Tangental Knife..... 11
 - Notes on Examples..... 12
 - Terminology..... 12
 - Overview of the NC Export Default Setup..... 12
 - Concept of @directives and Templates..... 12
 - Templates..... 13
 - Literal Strings..... 14
 - @Directives.....14
 - Modal Data..... 14
 - Main Initialization..... 15
 - Line..... 15
 - Using the GNC Driver in an Output.....17
 - Additional Position Options.....18
 - Concept of Run Time Data.....19
 - General Tab..... 21
 - Output Units..... 21
 - Output Mode for Main Program.....22
 - Main Program Number.....22
 - Use File Segmenting.....22
 - Block End Delimiter.....23
 - Expression Parentheses.....23
 - Comment Indicator..... 23
 - Readability.....24
 - Block Numbering.....25
 - Define Reference Tool.....26
 - Subs Tab..... 26
 - Output Mode for Subs.....27
 - Output Subs.....27
 - Subprogram Numbering Scheme..... 28
 - Controllers That Use Block Numbers to Call Subprograms..... 29
 - Subprogram Mirroring..... 30
 - Subprogram Rotation.....31
 - Start and End Points for Subprograms.....31

Codes Tab.....	32
Units @UnitsCode.....	33
Abs/Inc Mode @AbsIncCode.....	34
Interpolation Code @InterpCode.....	34
Flute/Grain Code @FluteCode & Design Side Code @SideCode.....	36
Fly Mode @FlyModeCode.....	36
Axis Data Tab.....	38
Axis 1 / Axis 2.....	39
[This Axis] Is [A] Rotary Axis.....	40
@RotaryRadius.....	40
Pseudo Linear.....	41
Addresses.....	41
Arc Data Tab.....	43
Arcs Defined by Center Point.....	44
Arcs Defined by Radius.....	45
Arcs Defined by Subtended Angle.....	46
Arcs Defined by Start and End Angles.....	47
Special Cased Arc Processing.....	48
Faceting for Large Radii.....	48
Controllers Unable to Process Arcs.....	49
Controllers Requiring Different Formatting for Full Circles.....	49
Speed Tab.....	50
Controllers Basing Speed on Tool Selection.....	51
Controllers Basing Speed on Stored Variables and a Display Screen.....	51
Controllers Providing Basic Speed Control.....	52
Format.....	52
Number of Speed Values.....	53
Use of Speed Values.....	54
Speed Parameters.....	54
What Data Gets Entered?.....	56
Speeds Based on Stored Variables.....	56
Speeds Based on Values.....	57
Small Radii Factor Formula.....	58
Use of Run Time Data.....	59
Tools Tab.....	60
Create New Tool / Redefine Tool.....	61
Tool Data.....	62
Tool Control Functions.....	62
Tool Down/On After Selection.....	63
Up/Off Required Before De-Selection.....	63
@Directives Suitable for Tool Templates.....	63
Support Default Tool.....	64
Tangential Tool.....	64

Arc Block With and Without Tangential Data..... 65

Setting the Angle During a Move.....65

Machines with Tangential Control Controllers.....66

Facet Small Radii (for Tangential Tools only)..... 66

Tool Offset Done by Software.....66

Short Move Control - Minimum Move..... 67

Use Fly Mode and Speed Values..... 68

Tool Rot[ation] Tab..... 68

 Always Absolute / Sign Indicates Direction of Travel..... 70

 Per Tool Data.....71

Text Tab.....71

Templates Tab..... 73

 List of Templates..... 73

 List of @Directives.....74

 Additional Notes on @Directives..... 80

 Optional Keywords..... 80

 General Comments on Templates.....82

 Insert @Directive..... 82

 Insert Run Time Data.....83

 Terminate with End Block Number..... 84

Run Time Data..... 84

 Dataset Defaults Setup..... 84

 Edit RTD (Run Time Data) Definition..... 85

 During the Output Process..... 87

 Caching of Run Time Data Entries.....88

1. Location and Copyright

Esko

Kortrijksesteenweg 1095

BE-9051 Gent

Belgium

Tel.: (32) (9) 216-92-11

Fax: (32) (9) 216-94-64

Other offices worldwide.

Written and revised by Adam Hartfield, January 2018.

For use with ArtiosCAD 16.1.1 or greater.

© Copyright 2016 Esko Software BVBA, Gent, Belgium.

All rights reserved. This material, information and instructions for use contained herein are the property of Esko Software BVBA. The material, information and instructions are provided on an AS IS basis without warranty of any kind. There are no warranties granted or extended by this document. Furthermore Esko Software BVBA does not warrant, guarantee or make any representations regarding the use, or the results of the use of the software or the information contained herein. Esko Software BVBA shall not be liable for any direct, indirect, consequential or incidental damages arising out of the use or inability to use the software or the information contained herein.

The information contained herein is subject to change without notice. Revisions may be issued from time to time to advise of such changes and/or additions.

No part of this document may be reproduced, stored in a data base or retrieval system, or published, in any form or in any way, electronically, mechanically, by print, photoprint, microfilm or any other means without prior written permission from Esko Software BVBA.

This document supersedes all previous dated versions.

This software is based in part on the work of the Independent JPEG Group.

Adobe, Acrobat, Illustrator, and PostScript are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States and/or other countries.

Apple and QuickTime are registered trademarks of Apple, Inc.

Microsoft and the Microsoft logo are registered trademarks of Microsoft Corporation in the United States and other countries.

The Esko software may contain an implementation of the LZW algorithm licensed under U. S. Patent 4,558,302 and foreign counterparts.

The Esko software may contain the "RSA Data Security, Inc. MD5 Message-Digest Algorithm."

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems in the U.S. and other countries.

Strip Clip, Strip Fork and Strip Clip[®] System are products, registered trademarks and patents of Vossen Profitec[®] GmbH Germany.

OpenGL is a registered trademark of Silicon Graphics, Inc.

Contains PowerNest library Copyrighted and Licensed by Alma, 2005 – 2007.

The geometry macros contained with this release of ArtiosCAD to facilitate the use of Vossen Profitec components are approved by Vossen Profitec GmbH and are used with their permission. The Vossen Prax[®] documentation should be consulted and used to ensure correct use and placement of these geometry macros. The shapes and offsets used are in accordance with Vossen Profitec GmbH specifications. Usage and placement of these geometry macros to ensure effective stripping performance however, remains the responsibility of the user. Vossen Profitec GmbH may be contacted for details of worldwide representation at (49) (7771) 920-136 or by e-mail at info@vossen-profitec.de.

This software may use libxml2 - Copyright © 1998-2003 Daniel Veillard - All rights reserved.

All other product names are trademarks or registered trademarks of their respective owners.

Correspondence regarding this publication should be forwarded to:

Esko

Kortrijksesteenweg 1095

BE-9051 Gent

Belgium

info.eur@esko.com

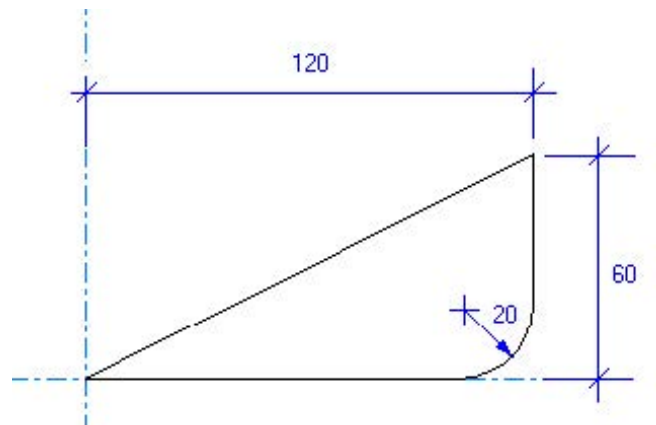
2. Using the General Numeric Controller driver

ArtiosCAD is used for design in the packaging industry, the sign making industry, and the gasket industry. Various types of equipment make samples, individually cut production items, tooling such as die boards, upper and lower stripping jigs, or counters. Most output machines used in these industries use some form of computerized numeric controller (CNC) to interpret the data sent to them. A typical machine can move in two axes, set different depths of cutting, change tools, turn them on and off (or raise them up and down) and so forth. Each of these operations is controlled by simple sets of instructions. Each machine may be slightly different, but there is a high degree of commonality.

There are some standard formats such as those used by most computerized numerically controlled machines and some standards such as the Hewlett-Packard Graphics Language (HPGL, which is often used by plotters). Other common formats include DDES, DDES3, and CFF2. This driver can generate data in all such forms.

The GNC driver behaves like any other ArtiosCAD driver, but is highly configurable. The driver sits at the end of an ArtiosCAD Output process. The data presented to the driver is in the form of lines, arcs, and text *after* the appropriate ArtiosCAD Output processing has been performed. Thus, special line type information has already been resolved to a series of simple lines (straight or arcs) or enveloped outlines. (When the GNC driver is used to produce a CFF2 or a DDES3 compatible file, the data contains line and arc information, but it does not contain special line type information, since this already has been resolved by the Output processing.)

Shown below is an example structure to be made on a CNC machine.



The example above could be constructed by any of the three data examples below:

NC type data	HPGL type data	CFF2 type data
T1	SP1;	
G1X100Y0	PD4000,0;	L,2,1,0,0,0,100,0,0,0
G2X120Y20J20	AA4800,800,90.00,1;	A,2,1,0,100,0,120,

NC type data	HPGL type data	CHF2 type data
G1Y60	PD4800,2400;	20,100,20,1,0,0
X0Y0	PD0,0;	L,2,1,0,120,20, 120,60,0,0
T0	PU;	L,2,1,0,120,60,0,0,0,0

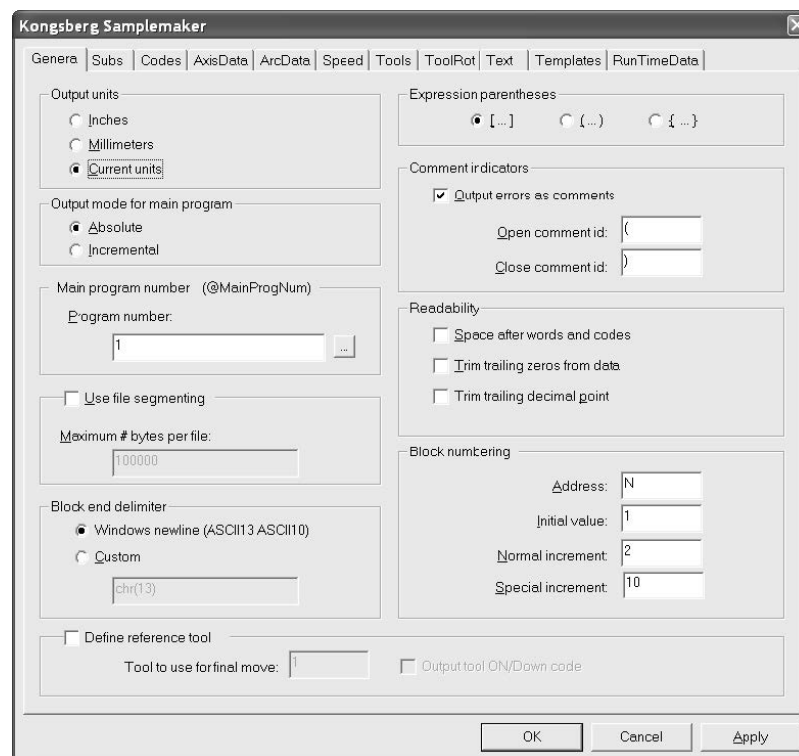
Each ArtiosCAD Output type performs some processing to form the lines needed to make the required finished article. Consider a dieboard with a die split. The dieboard output processing segments the die into each die split segment, offsets the edge line types, sequences the lines according to the CAM Tooling Setup, and places them in a temporary area ready to be processed by the driver.

The General Numeric Control (GNC) driver takes these lines, arcs, and text and formats the NC data correctly for cutting on the particular die cutter selected. The GNC driver can format data for nearly any laser die cutter, routing machine, or water jet cutter used in the target ArtiosCAD industries.

Note:

This driver **does not** perform customized processing.

Examples of NC driver configurations are contained in the NC Export tuning table catalog in Defaults. Shown below is the one for a Kongsberg Samplemaker.



To actually use the driver once you have configured it, set up an Output of type **CAM**, and on the Device tab, choose **CAM Driver** for the Driver Type. Set the Tuning Entry to **Custom Setup**,

set the CAM driver to **GNC**, and then click the **Browse** button at the end of the Tuning Filename field. Choose the appropriate tuning from the Export Tuning List. Finish setting up the Output normally, making sure to select a CAM Tooling Setup entry on the Processing tab.

Before You Start

To set up this driver satisfactorily, you need a good working knowledge of:

- The features and functionality available on the target equipment's controller
- The functionality available in ArtiosCAD Outputs
- CAM Tooling Setup configuration and/or plotting styles. (CAM Tooling Setup provides a lot more flexibility)
- The GNC driver

The first item is very important. This is information that is provided by the machine manufacturer (integrator).

The way the equipment is used may also influence how ArtiosCAD is configured to deal with these needs. For example, is the laser cutter used to cut widely varying materials, or just die boards?

If you are uncertain about any of these items, Esko solution architects can help. We actually may have no specific knowledge about the first (and most important) item. However, we can assist you in finding the necessary information, or may be able to help you establish it by observing the current operation during a site visit.

About Your Equipment

Let us consider a laser die cutter for this example. These machines come in various degrees of sophistication.

A typical die cutter comes with some kind of NC controller, designed to move a tool head around and to control a laser which generates the cutting beam. The laser itself may have a control system allowing different powers and pulse modulations to be set and controlled. The beam delivery nozzle may also be movable to allow the distance between the focal point and the work piece to be controlled. There may also be controlled auxiliary equipment, such as fume extractors, water coolers, safety fences, and so forth necessary for the safe operation of the system. The machine builder will have integrated these components to a greater or lesser extent. For example, it may be that the power settings on the laser are directly controlled by codes in the NC data. There may be an NC code at the start of a program which turns the extractor fan on automatically. Alternately the extractor fan may be completely independent of the controller and controlled by a switch. However, there may be an interlock that ensures

that the fan is running before the motion control can activate. You should know all such dependencies before creating a GNC driver entry.

About Your NC Controller

The property tabs in the General NC driver dialog box are a good guide to the information to gather. If for example, there is no subprogram functionality provided by the controller in question, there is no need to complete the information on the Subs tab.

Our example machine may use a general NC controller (such as a Bosch, GE Fanuc, etc.) which has been customized and integrated with the other components, or the whole system may use proprietary equipment and control systems. Either way, it is essential to have a manual describing all the functions available to successfully and adequately drive the equipment.

Where a General NC controller is used, there are two components to this information:

- The manual for the NC controller.
- The list of implemented functions and codes and what they do.

There may well have been a manual supplied with the controller. However, not every feature that is available for this controller may have been purchased or implemented by the machine integrator. It is thus necessary to know which functions from the manual are available for this particular implementation. Furthermore, there is particular functionality programmed into the controller to deal with the functions of the specific equipment.

For example, the way in which the laser beam is turned on and off was decided by the machine integrator. How this is implemented is almost certainly not something that is described in the general manual for the controller. It is necessary to have a list of the functions that have been implemented, and the special codes selected by the machine integrator to perform the various tasks controllable by the NC data, such as tool selection, shutter control, extractor fan control, and so forth.

Since most machines work in a similar manner, the more important of the two is the information provided by the machine integrator. If this is available, **and** there are existing programs available that currently drive the equipment, the general manual for the controller, though useful, is less important to have.

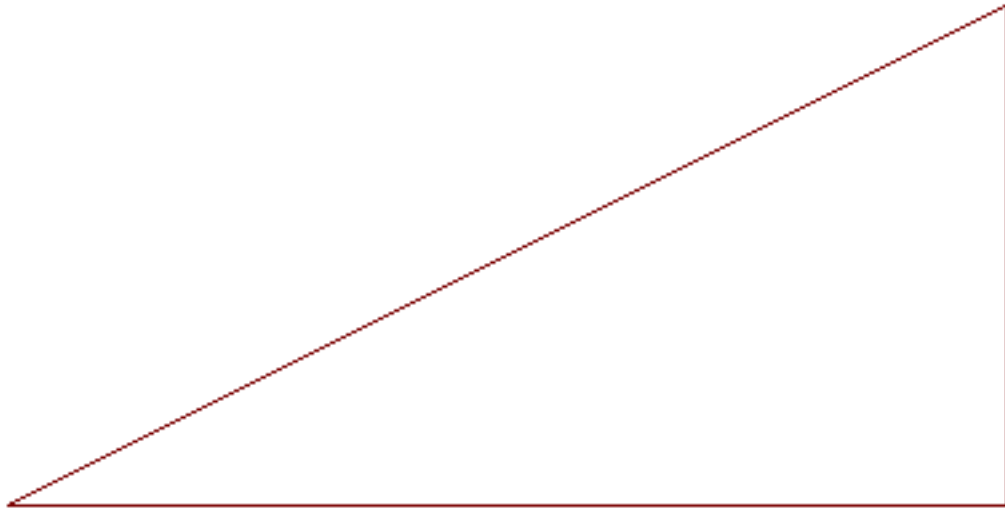
Moving the tool head over the work piece is the most straightforward part. On the other hand, how to configure the 'tools' needs some planning. Some things to consider are:

- What tools are needed and available? (2pt, 3pt etch pulsed/cw)
- What is needed to select and deselect these tools?
- Tool sequencing. (For example, can the etch tool be selected during regular cutting, or must all etching be performed first? Sometimes low power cutting is not achievable without a delay.)
- How the speed and power is controlled for each tool.
- What differences are needed for processing different materials, and where and how this is handled, and so forth.

It is useful to have example code from the machine integrator which shows the use of each of the implemented functions. It is also a smart idea to try it and see that it really does do what the machine integrator says it does (sometimes this data does not get updated to reflect the way that the particular machine in question **actually** works).

Example - A Sample Maker with a Tangential Knife

A cutting knife on a sample maker must be rotated into the direction of the line to cut. The controller may deal with this completely by itself or it may be completely dumb. Let us consider various degrees of intelligence that may be exhibited by a controller for this sample to cut:



Unsophisticated Controller	Sophisticated Controller
G90C0.00 (0=in line with Y+)	T0
T0 (tool up)	G0X0.00Y0.00
G0X0.00Y0.00C-90.00	T1
T1 (tool down)	G1X100.00
G1X100.00	Y50.00
T0	X0.00Y0.00
C0.00 (turn to Y+ axis)	T0
T1	
G1Y50.00	
T0	
C116.57	
T1	
X0.00Y0.00	
T0	

Unsophisticated Controller

All tool angles and tool up/down commands must be encoded in the NC data by the driver.

Sophisticated Controller

The controller looks at the next line to be cut. It decides whether the tool is pointing in the correct direction. If it is not, it automatically lifts the tool, sets the tool to the correct angle and then lowers the tool to cutting depth before cutting the next line. There may even be a

parameter set somewhere that says how big an angle can be turned without the need to lift the tool.

Further Considerations

It may be that the sophisticated controller sets the angle if it is not set in the data, but better throughput or quality may be achieved by providing the up/turn/down information.

For example, after a move, the controller may put the tool down only to discover that the angle is that of the previous cut. Only when it is about to cut does it determine that the angle is not correct. Not only does it now have to pick up the tool, set the angle and put it down again, but it has physically marked the sample.

This sort of information is harder to gather. It may be implied rather than stated. Example NC data of realistic samples helps to find these implied requirements.

Notes on Examples

One of the most complex devices driven by ArtiosCAD is a laser die cutter. For this reason, these machines generally serve as examples when describing the supported features, particularly when discussing speed control. This driver provides very comprehensive support for a very wide selection of equipment and controllers. Not all functionality is used by all equipment. If the section does not sound relevant to the equipment you are driving, we suggest you read it quickly, without bothering to digest all the details, at least for the first pass.

When discussing tangentially controlled tools, sample makers serve as examples.

Terminology

An **address** is a command character that precedes data. It is not a physical or logical location.

A **block** is one line of data. It usually ends with a delimiting character.

Absolute measurements are measured from the origin.

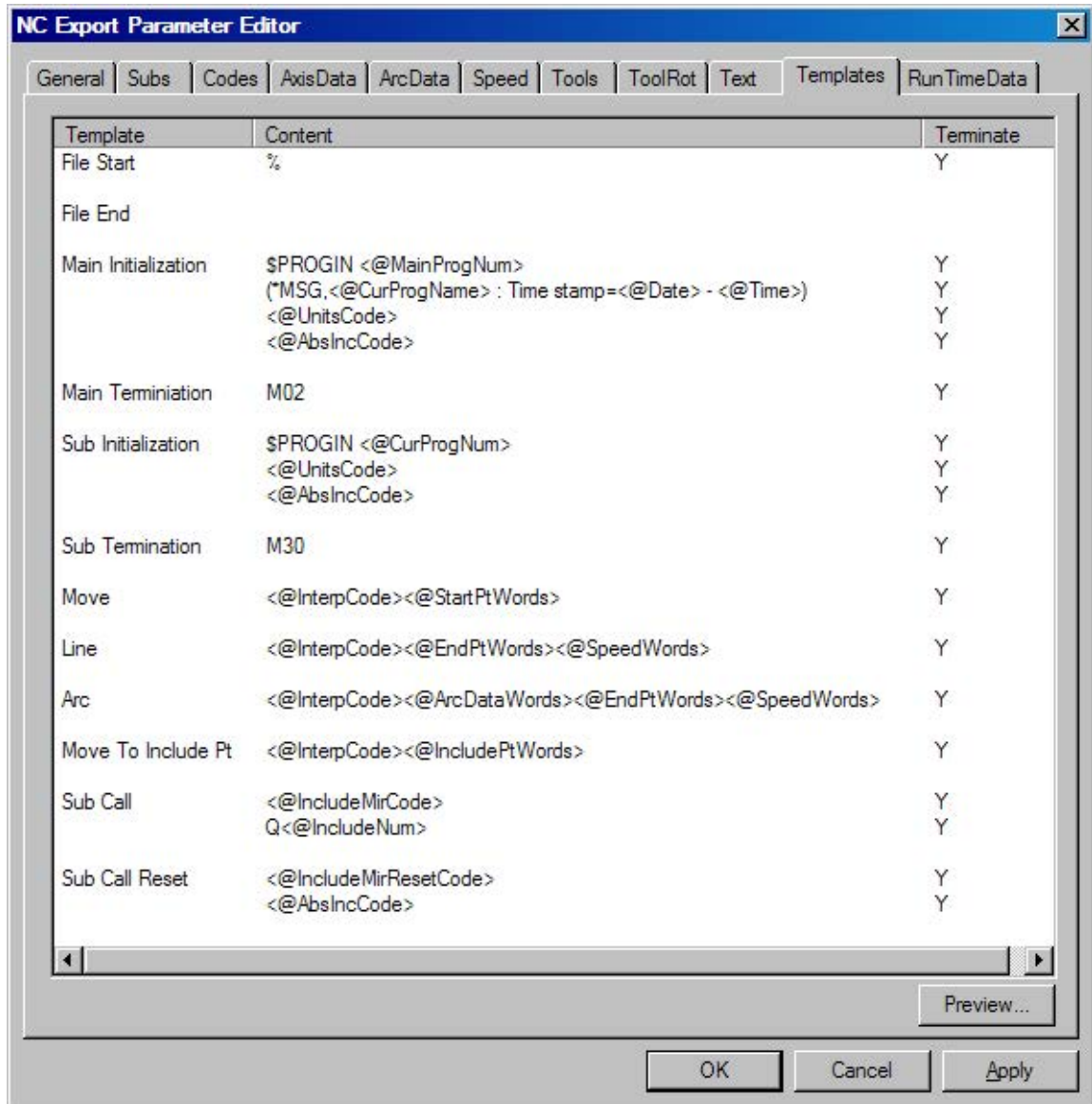
Incremental measurements are measured from the last current position.

Overview of the NC Export Default Setup

Concept of @directives and Templates

The GNC driver translates and reformats basic data (lines, arcs, and text) in ArtiosCAD to drive many different machines.

The driver performs this using *@directives*, which are variables expanded at time of Output to form the actual data sent to the machine, and *templates*, which are collections of @directives. More examples and lists of all templates and @directives are found in the description of the Template tab. Shown below is an entry in the NC Export Tuning Table catalog in Defaults.



Templates

Shown above is an example of a set of templates. Each template describes the code necessary for the various common segments of an NC program. There are templates for the start of data, the end of data, the heading of program segments, for describing lines and arcs, and so forth. There is also a set of templates for each tool (not shown above). The tool templates describe how each tool is selected, deselected, and turned on and off. The example shown is a real example that can drive a laser die cutter (though not all features are used).

Each template may contain:

- **Literal text strings.** This is text that is directly placed into the NC data as is.

- **@Directives.** An @directive represents a frequently used set of NC data. For example, the data representing the coordinates of the start point of a line is represented by the @directive **@StartPtWords**.
- **Run time data.** This is data that can be changed each time an Output using the GNC driver is run. It may be used directly in a template, or it may form part of an @directive. Run Time Data might be used to set the speed and depth of cutting for a particular tool.

In the example shown on the previous page, the Main Termination template contains only the literal text string (M02), the **Move** template, two @directives (**@InterpCode** and **@StartPtWords**), while the **Sub Initialization** template contains a combination of a literal text string ('\$PROGIN') and three @directives (**@CurProgNumber**, **@UnitsCode** and **@AbsIncCode**). No run time data is used in this example. This advanced feature is described at length in a section of its own.

Not all templates need to contain data. In this example, there was no requirement to have any additional data to indicate **File End**. It was therefore left blank.

Not all possible templates may be used. Some may appear depending on the options selected on some of the other property pages. A full list of templates and their use may be found in the Templates section.

Literal Strings

Literal strings are placed into the NC data exactly as entered in the template. Use the **chr(dec.ascii.code)** function to enter non-printable characters in a template. Consult the Internet for lists of ASCII codes (<http://www.asciitable.com> is one such place to look).

A single template may generate multiple lines of code. The Main Initialization template as shown on the previous page generates four lines of code.

@Directives

@Directives are used in templates. In general, the format and content of an **@directive** is described on one of the other property tabs. For example, the **@StartPtWords** directive is described on the Axis Data property tab.

Modal Data

Some @directives are defined as being modal, or may contain *modal* parts. Modal data is output only when necessary.

The interpolation method **@AbsIncCode** is often modal data. This controls whether the axis data describes a rapid move from one point to another, a straight line between two points, or a clockwise or counterclockwise arc. The interpolation method remains as set until it is changed. It is therefore necessary to output one linear interpolation code only before a series of connected Line templates.

Examining the example template for the Main Initialization and the template for a straight line helps to better explain the basic idea.

Main Initialization

```
$PROGIN <@MainProgNum>
(*MSG,<@CurProgName> : Timestamp=<@Date> - <@Time> )
<@UnitsCode>
<@AbsIncCode>
```

The **bold** data is output literally, while the @directives are replaced by the actual data at the time they are invoked.

This template might expand to:

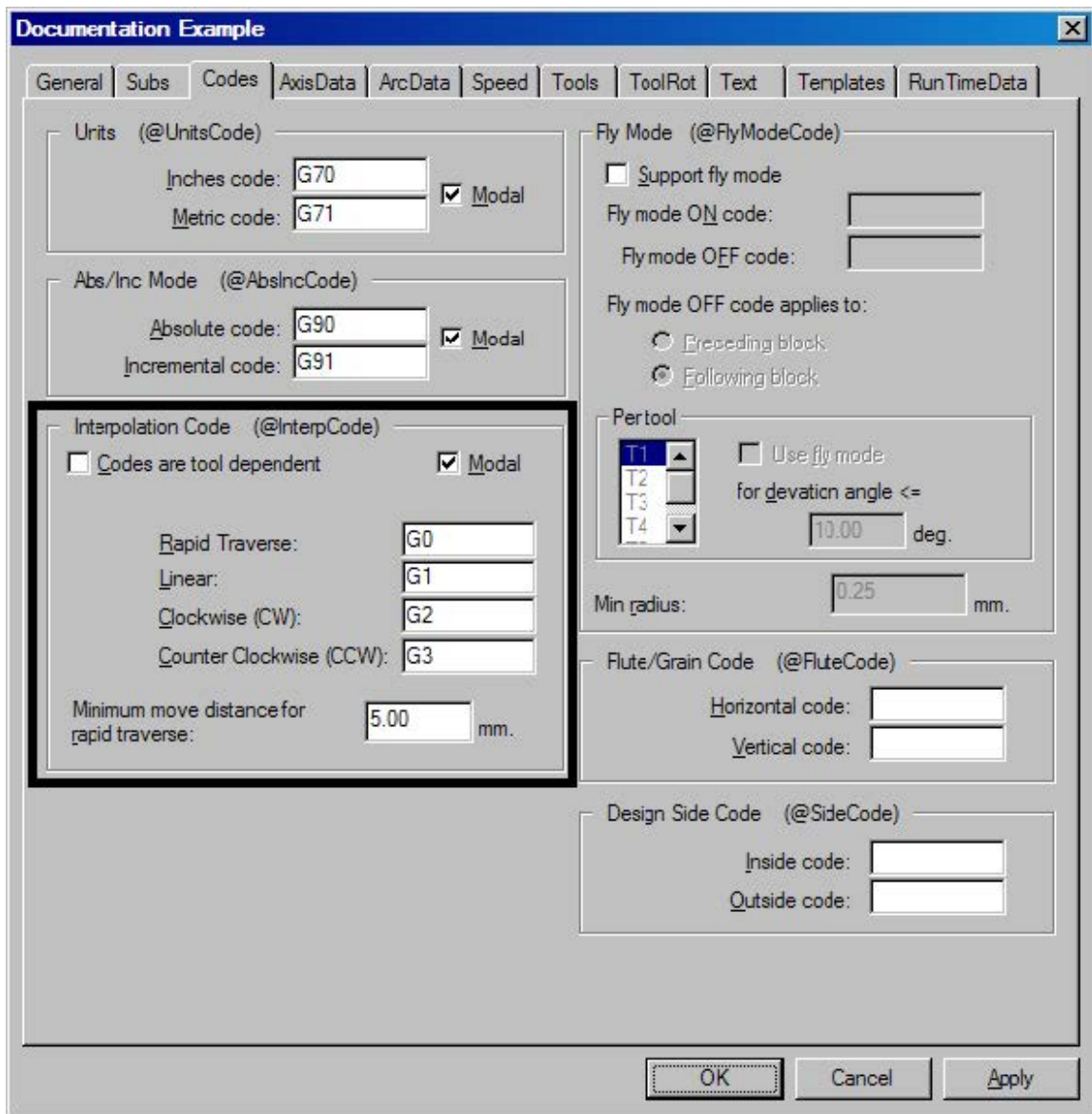
```
$PROGIN 100 (*MSG,NiceThingsInSmallBoxes.MFG : Timestamp=07/12/2004-18:55:4)
G71
G90
```

Line

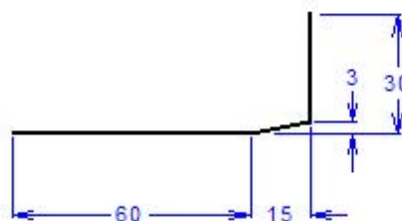
```
<@InterpCode><@EndPtWords><@SpeedWords>
```

Not all @directives create data all the time. The template to produce a line includes @directives to set the interpolation method to use, the end point, and the speed at which the machine is to move while cutting the line.

The setup for the **@InterpCode** directive, for example, is found on the Codes tab.



This same Line template invoked three times might generate the following NC data.



Modal Data	Non-Modal Data
G1 X60.00 Y0.00 F20	G1 X60.00 Y0.00 F20
X75.00 Y3.00	G1 X75.00 Y3.00 F20

Modal Data	Non-Modal Data
Y30.00 F21	G1 X75.00 Y30.00 F21

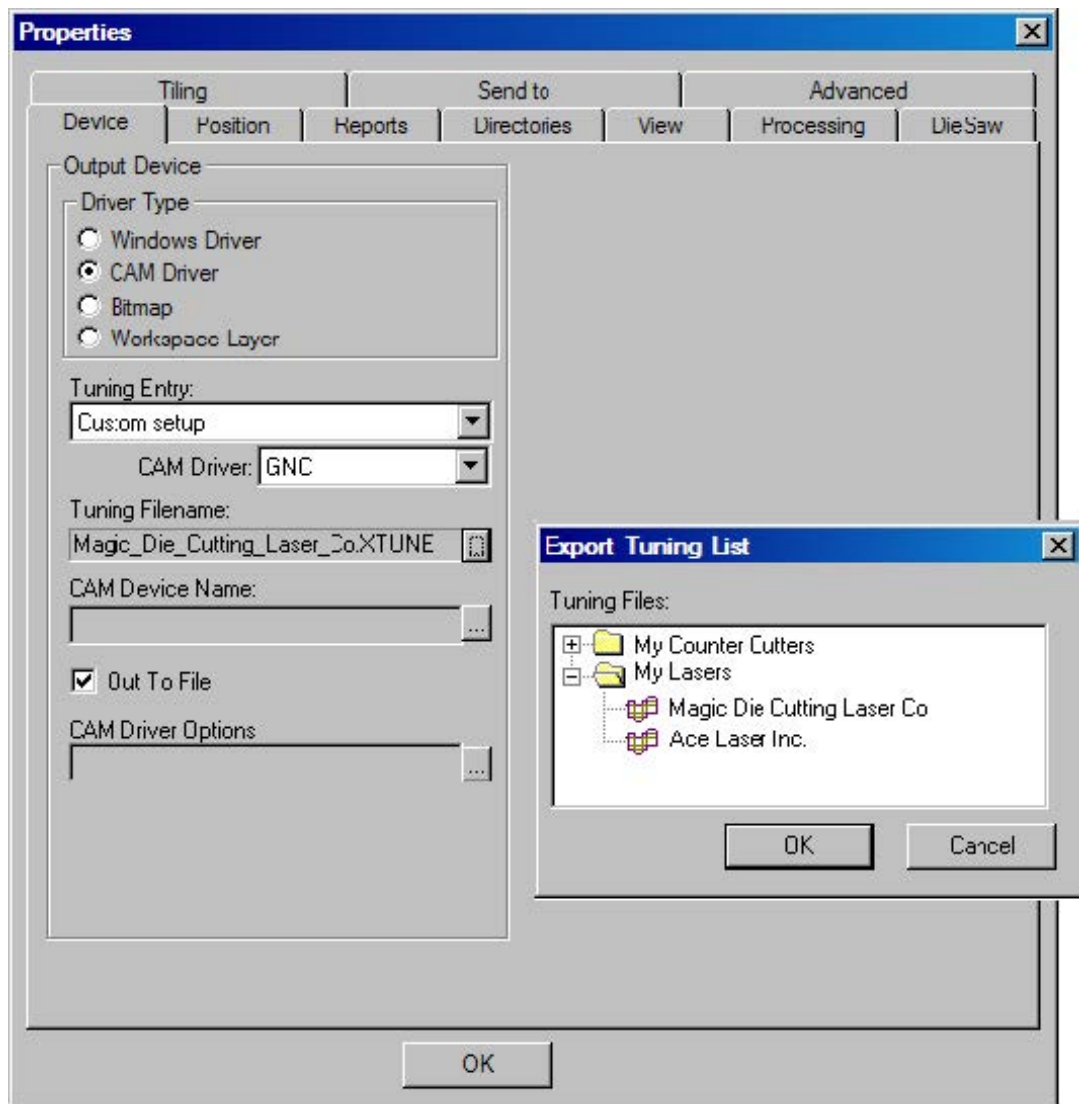
All three @directives used in the Line template were configured to be modal, and each contained modal information.

@InterpCode produced the G1 only once, since all lines were linear. **@SpeedWords** created nothing on the second line, presumably because the speed to travel in a direction close to the X axis was the same the speed to travel in only the X direction. On the third line, the speed changed, but the X portion of **@EndPtWords** did not. Therefore some data was generated by **@SpeedWords**, but nothing was generated for the X component of **@EndPtWords**.

The use of templates and @directives makes for a very concise description of the required NC data. The example set of templates shown at the start of this section is sufficient to drive a 2-axis NC machine that supports subprograms and mirroring. Of course there are other templates that describe how to select/deselect and activate/deactivate tools, but the principles are the same.

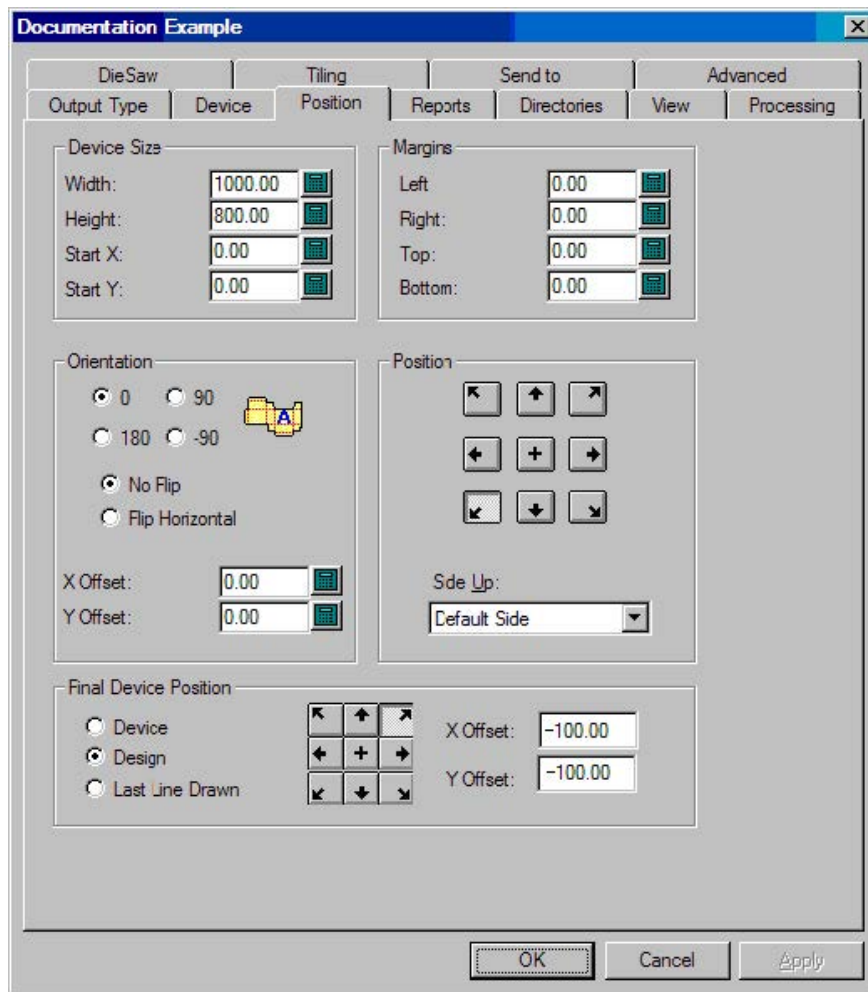
Using the GNC Driver in an Output

The General NC driver, together with appropriate tuning, are selected on the Device tab of an Output. The driver name is GNC and is selected in the same way as any of the device-specific CAM drivers. The tuning is selected from a tree control, rather than directly selecting a tuning file. The tree control contains a list of all available GNC tunings.



Additional Position Options

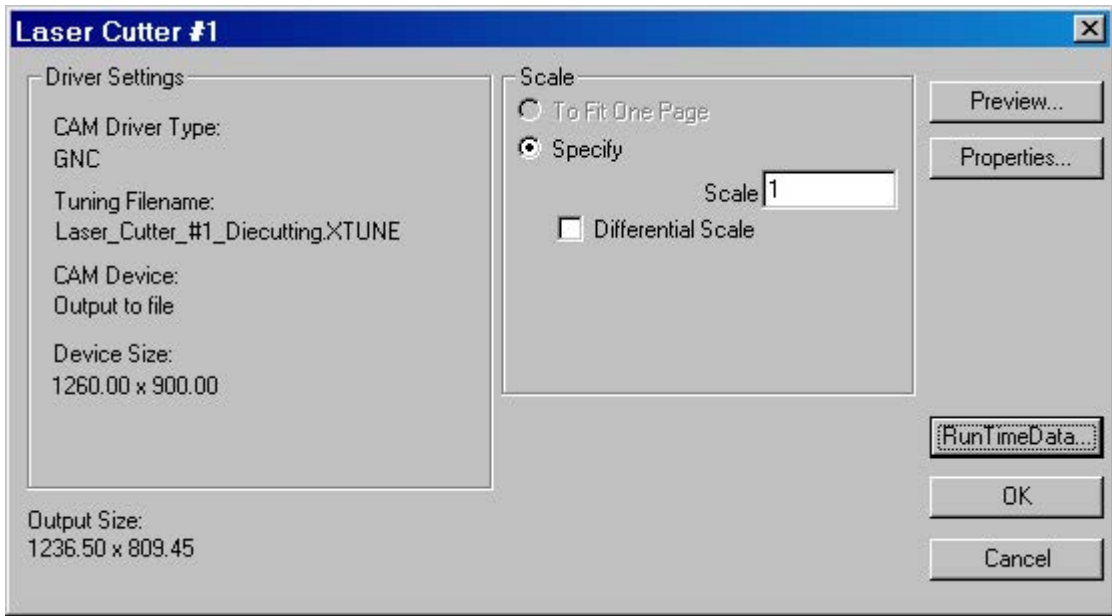
Selecting the GNC driver on the Device tab causes additional options to appear on the Position tab in the Final Device Position group. These control of the final position of the tool head when the output is finished. The tool head may be driven to a position offset from any of the nine anchor points of the design or device.



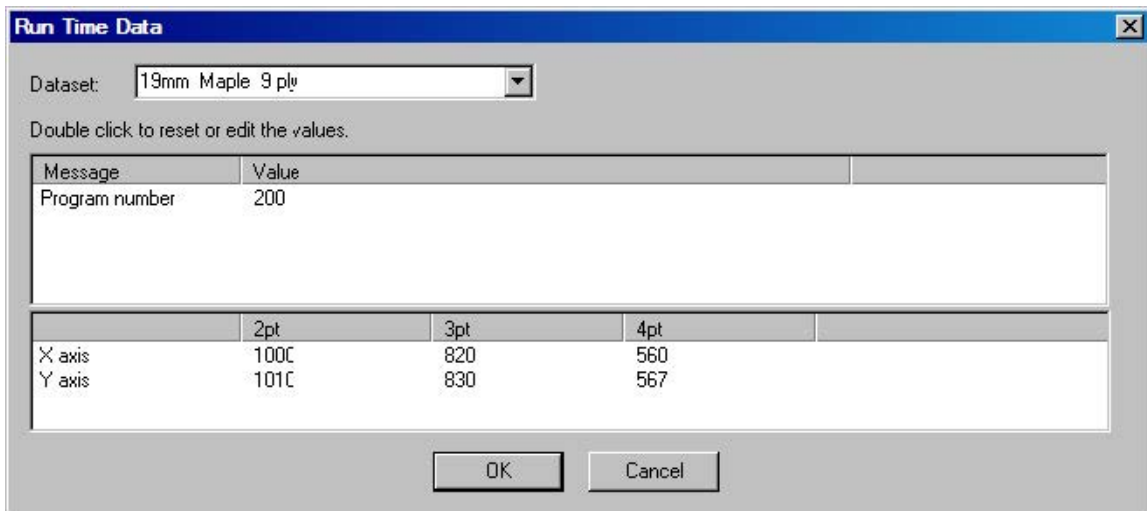
Concept of Run Time Data

Some of the parameters which control the behavior of the General NC driver may be set as *run time data* elements. These may be set and adjusted each time an Output is executed. For example, the speeds for cutting a die on a particular laser die cutter may vary from time to time depending upon the dryness of the wood and other factors. Run time data elements may be used in any of the templates, and many of the data entry fields that help build the templates. This is described in more detail later.

When the Output is executed, click **RunTimeData** to access the run time data.



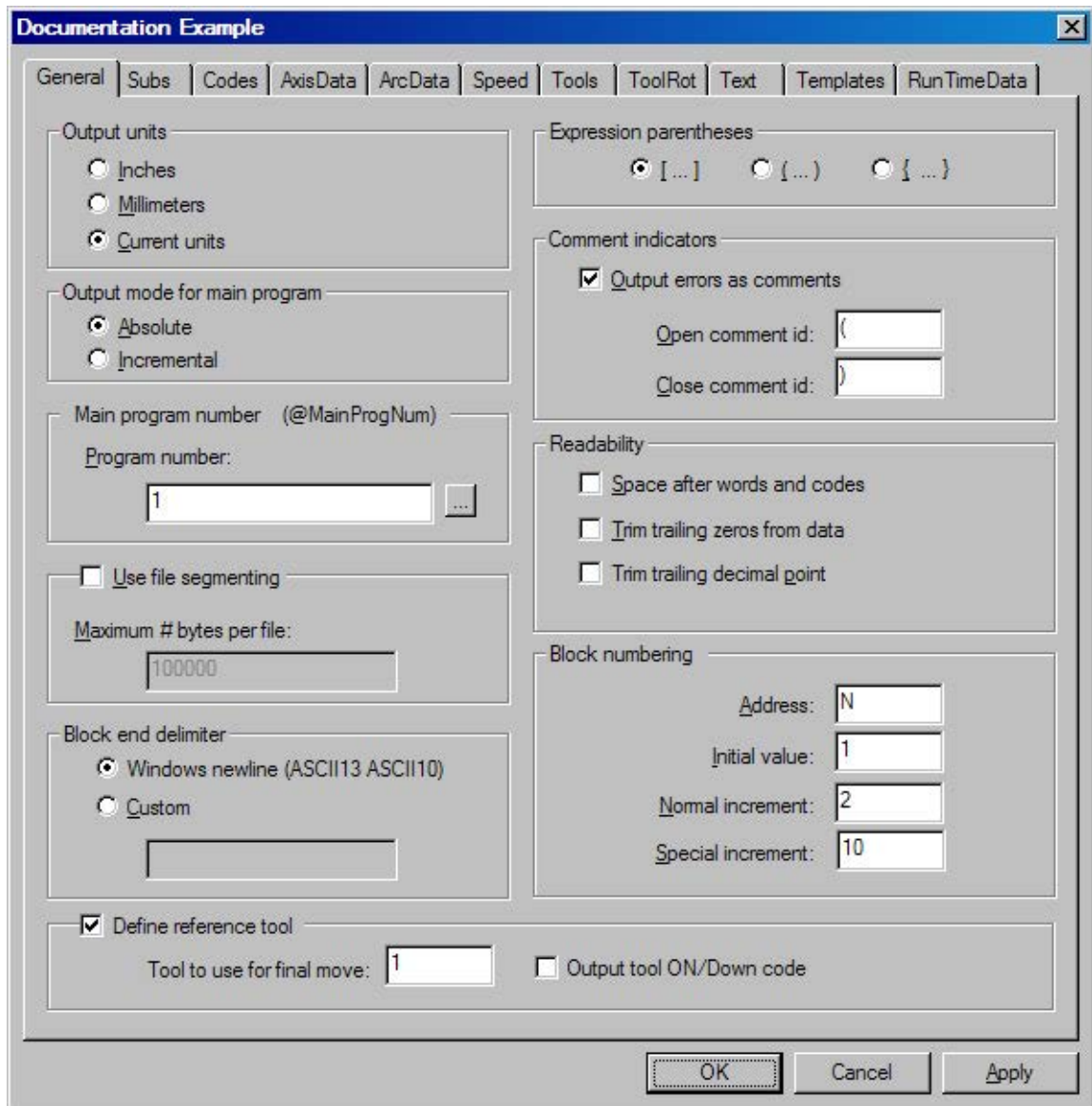
Below is an example where the user has chosen to make the program number and the speed values for cutting 2, 3, and 4 point slot available for changing on Output. Note the **Dataset** drop-down list box. A number of such datasets may be defined. For example, you could define datasets for 19 mm Maple 9 ply wood, 12 mm Maple, and 12mm Birch, each with different speed settings suitable for each material.



Run time data is cached each time the Output is run. Once test cuts are made, and appropriate speed settings established, subsequent uses of this Output are seeded with these "speeds for the day" until they are changed.

General Tab

The General tab contains general setup information and initialization data.



Output Units

This setting determines whether the NC data is generated in metric or inch units. It also determines the code generated by the **@UnitsCode** directive.

Normally the operator of the equipment expects data in either metric or inch units. However, it may be that the NC data should be the same as the current units at the time it was output.

Output Mode for Main Program

The main program movement data may be encoded in absolute (measured from the origin) or incremental (measured from the last current position) form. This setting determines the code generated by the **@AbsIncCode** directive when used in the main program.

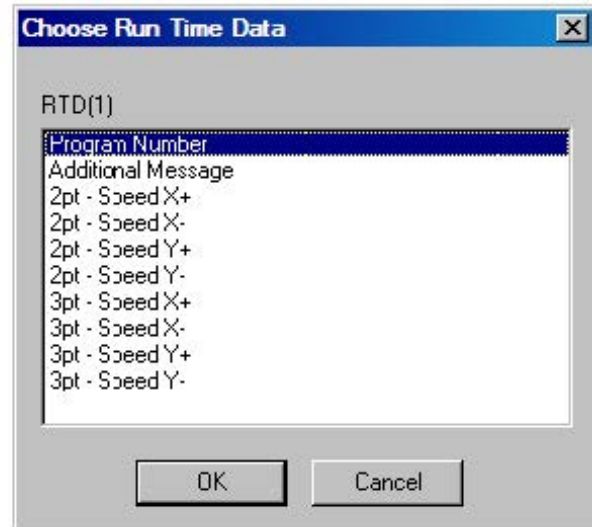
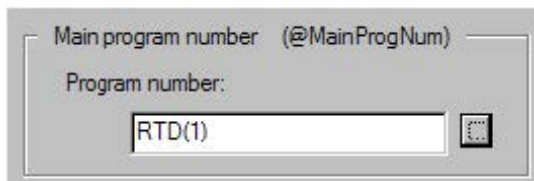
Choose absolute mode for the main program unless there is a compelling reason to use incremental mode.

Main Program Number

This controls the number assigned to the main program. To change this value at runtime, click the ... button and select the RTD element to associate with this field. In most cases this is one of the *messages*, or single value elements.

Note:

You must define run time data elements before you can use them.



Use File Segmenting

Some controllers have limited memory for storing programs. If a program will not fit into memory with all its subprograms, the program may be output in *flat* (no subprograms) segments.

Set the maximum number of bytes per segment to the available memory size. If the amount of NC data for a large design exceeds this value, it is segmented into many smaller program

files, each one being less than the maximum segment size. Each of these files contains one Main program; in effect, the data is flattened. Running each of the segmented program files completes the large design.

The size of the memory may not exactly match the size of the file. For example, some controllers use slightly more memory to store the data than the size indicated by the equivalent data written to a disk file. Others may eliminate spaces or convert CR/LF delimiters to a single block end character. Find the right value by experimentation.

Warning

Suppose a 10-up straight layout of a very complicated design just fails to fit into the memory of a (really small) 32 KB machine. The data in the main program for the 10 placements would probably amount to no more than perhaps 1 KB. Thus the one-up design must be almost 32 KB by itself. The flat design is therefore going to be 10 x 32 KB. This is going to run in at least 10 segments, maybe even more, but the product is a complete die.

Block End Delimiter

The normal delimiter placed at the end of each data block is Line Feed /Carriage Return (ASCII 10. 13.). When the Custom selection is chosen, any string of characters may be defined. Use the `chr(dec.ascii.code)` function to generate non-printable ASCII characters.

Entry	Resulting Delimiter
<code>chr (10) chr (13)</code> <code>; chr (10)</code>	Line Feed / Carriage Return (ASCII 10. 13.) Semicolon followed by Line Feed. Usually for HPGL.
<code>chr (9)</code> <code>chr (10)</code>	Tab delimited blocks of data. Line Feed (ASCII 10.)

Some controllers require special delimiters. For example, earlier serial IBH controllers may misinterpret some data unless the delimiter is a LF only.

Expression Parentheses

Some controllers support the use of system variables and expressions. In this driver, speed information may be encoded either as a value (such as `F1000`) or as a function of system variables. (such as `F#161` setting the speed to the value assigned to variable #161). Speeds may also be set to different values when moving in X and Y. It is also possible to set the speed of a diagonal line to be proportioned between these two values. If variables are used, an expression such as `F[#161*0.33 + #171*0.67]` is required. In this example, the `[. .]` are the parentheses used for the expression. Choose the ones defined for the target controller.

Comment Indicator

It is possible to report errors and warnings as comments in the NC code as it is generated. If your controller supports comments, enter the comment indicators here. If there is only one (such as `;` indicates the rest of a line is a comment) enter it as the first one and leave the second blank.

Errors as Comments

Check the **Output errors as comments** checkbox in the Comment indicators group to insert error or warning messages in the output file.

The following errors may be reported in the output:

Could not evaluate:

The GNC driver did not recognize an @Directive used in a template.

WARNING: Text is not supported.

The ArtiosCAD file contains a text entity but the GNC tuning does not support text.

WARNING: Short arc being treated as short line.

An arc with chord length less than two pixels is output.

WARNING: zero radius arc being treated as zero length line.

A circle with zero radius is output as a zero length line.

ERROR: BLOCK NUM NOT SET.

The **@MarkAsSub** directive has been used in a template but **Use block numbers** on the Subs tab has not been selected.

WARNING: type %d maps to tool 0. Line ignored.

A line is ignored because its line type maps to tool 0.

WARNING: type %d maps to tool 0. Arc ignored.

An arc is ignored because its line type maps to tool 0.

Text ignored in incremental mode.

A text item is supposed to be output in incremental mode.

Maxratio: %f ratio: %f.

The Fly Mode minimum radius is used so that you can evaluate how best to tune this value.

Readability

Space after words and codes does not functionally change the data. It does increase the size of the program, however, which may be significant for controllers with limited memory.

Strip trailing zero from data saves memory if any of the **Round to** formats **.1** through **.4** are used. It would convert `X123.100` to `X123.1` thereby saving two bytes.

Both Checkboxes Off	Both Checkboxes On
M60	M60
G0X0.00Y0.0	G0 X0. Y0.
M61	M61
G3I0.00J25.40X25.40Y25.40	G3 I0. J25.4 X25.4 Y25.4

Both Checkboxes Off	Both Checkboxes On
G2I25.40J0.00X50.80Y50.80	G2 I25.4 J0. X50.8 Y50.8

Block Numbering

Block numbers are used to identify blocks of NC data. They may be used to select a block from which to start execution or a block that needs to be edited. It is now uncommon to edit a block on an NC controller. Normally the program is created, for example, by ArtiosCAD and simply loaded into the controller for execution.

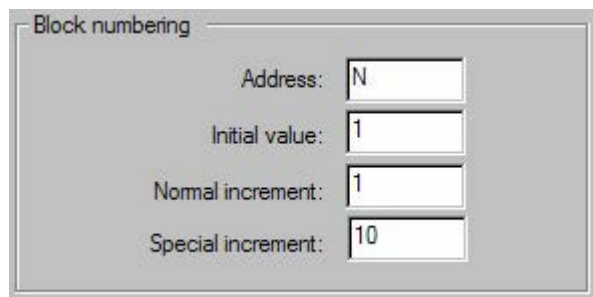
Using block numbers makes it possible (or easier) to find specific points within a program, but they also take up memory in the controller.

Some controllers use block numbers to control subprogram branching, in which case their use is mandatory.

Here are some considerations about how or whether to use block numbers:

- Are block numbers mandatory?
- Do you want block numbers?
- Is memory at a premium?
- Do all blocks need to be numbered?
- Should/may only sensible 'restart' points have block numbers?

The data in this section controls what is generated when the various **@NBlockOn** and **@NBlockOff** directives are used in templates.



The image shows a dialog box titled "Block numbering" with four input fields:

- Address: N
- Initial value: 1
- Normal increment: 1
- Special increment: 10

This dialog box sets the address, initial value, and increment values to be used.

@NBlockOn. Turns block numbering on. It continues with the next number with the specified increment. There are modifiers as follows:

- NORMAL
- SPECIAL
- SETTO (nnn)
- INCBY (nnn)

Using **@NBlockOn** with the `NORMAL` modifier starts the numbering with the normal increment, while `SPECIAL` uses the special increment. The initial number is first rounded up to the nearest increment value.

For example, if the last generated block number was 16 and the `SPECIAL` increment was 10, **@NBlockOn[SPECIAL]** starts at block number 20.

The `SETTO` and `INCBY` modifiers allow the start number and increment to be set on the fly. Listed below are examples of modifier use:

```
@NBlockOn[SETTO(100)]  
@NBlockOn[SETTO(100), INCBY(2)]  
@NBlockOn[NORMAL]  
@NBlockOn[SPECIAL]
```

@NBlockOff. Turns block numbering off. If is it used by itself in a template, it does not generate a blank line.

@MarkAsSub. When calls to subprograms are made using block numbers, it is necessary to identify the block which represents the start of a subroutine. This directive should be used in the first line of the Sub Call template. It is important that some NC data is generated in the block on which this directive is used. It should contain either some literal text or the @directive used should be non-modal.

Define Reference Tool

After mounting a new work piece, often a particular tool is moved to the required start position, and the program origin set. If tool head offsets are applied, it is necessary to select the reference tool at the end of the program. When this checkbox is checked, the select code for the specified tool is encoded at the end of the program.

If the tool is selected by way of the tool on/off codes (meaning that a tool select template is not used), check the **Output tool On/Down code** checkbox. The upside is that the correct tool is selected, the downside is that this tool is flashed on/off at the position of the final line.

Subs Tab

An ArtiosCAD layout having many placements of the same design actually holds only one copy of the design. This design is referenced as needed to form the complete layout. If the subprogram feature is enabled, ArtiosCAD makes a subprogram for each design in a layout. It also generates a main program which will calls the subprogram(s) to form the layout. On a simple layout, normally one subprogram is made for each design. However, the actual number made depends upon whether the controller supports mirroring or rotation features. It also depends upon how the ArtiosCAD Output groups the various lines that form the complete output. All of this is handled automatically as needed.

The screenshot shows the 'NC Export Parameter Editor' dialog box with the 'Subs' tab selected. The 'Support subprograms' checkbox is checked. Under 'Output mode for subs', 'Incremental' is selected. Under 'Output subs', 'Main followed by Subs' is selected. The 'Subprogram numbering scheme' is set to 'By formula' with the formula: $(@MainProgNum + 0) \times 1 + (SubSequence\# + 0) \times 1 + 0$. Under 'Subprogram mirroring', 'Mirroring supported' is checked, with 'Mirror in X' set to G39X1, 'Mirror in Y' to G39Y1, 'Mirror in X and Y' to G39X1Y1, and 'Mirror reset' to G38. Under 'Subprogram rotation', 'Rotations supported' is unchecked, with 'Address' set to R, 'Factor by' to 1.000000 deg, 'Round to' to 3 places, 'Rotation code' to G40, and 'Reset code' to G41. At the bottom, 'Start subprograms at' is set to 'Center of design' and 'End subprograms at' is also set to 'Center of design'. Buttons for 'OK', 'Cancel', and 'Apply' are at the bottom right.

Output Mode for Subs

If you have a choice, use **Absolute** for the main program and **Incremental** for the subprograms. This is the easiest to set up. On controllers where both mirroring and rotation are supported, it may be necessary to use **Absolute** mode for the main program and **Absolute** mode for the subprograms.

Output Subs

There are two methods for the program order, **Main followed by Subs** or **Subs followed by Main**. Normally the controller configuration determines this setting. For example, some controllers require that all subprograms be loaded into memory before the main program is

loaded. Clearly this would dictate that the subprograms came before the main program. The program that is current in the controller is often the last one loaded. Since this is the one that is run, it is thus convenient to load it last.

Subprogram Numbering Scheme

This controls the program numbers assigned to the main program and the subprograms. Many controllers accept a simple program numbering arrangement. However, some require that all subprograms associated with a main program must start with the main program number with the subprogram number forming a suffix. This is best explained by examples as shown below.

	Example 1	Example 2	Example 3	Example 4	Example 5
Main Program	1	100	123	123	40
Sub #1	2	110	12300100	12301001	10
Sub #2	3	120	12300200	12301002	20
Sub #3 and so on	4	130	12300300	12301003	30

Examples

Example 1

The main program number (set on the General tab) should be **1**.

This is equivalent to setting By formula: to **Use sequence starting at 1**.

Subprogram numbering scheme (@CurProgNum)

By formula
 Use sequence starting at 1
 Use block numbers

$(@MainProgNum + 0) \times 1 + (SubSequence\# + 0) \times 1 + 0$

Example 2

The main program number (set on the General tab) should be **100**.

Subprogram numbering scheme (@CurProgNum)

By formula
 Use sequence starting at 1
 Use block numbers

$(@MainProgNum + 0) \times 1 + (SubSequence\# + 0) \times 10 + 0$

Example 3

The main program (set on the General tab) should be **123**.

Subprogram numbering scheme (@CurProgNum)

By fomula
 Use sequence starting at 1
 Use block numbers

$(@MainProgNum + 0) \times 100000 + (SubSequence\# + 0) \times 100 + 0$

Example 4

The main program (set on the General tab) should be **123**.

Subprogram numbering scheme (@CurProgNum)

By fomula
 Use sequence starting at 1
 Use block numbers

$(@MainProgNum + 0) \times 100000 + (SubSequence\# + 0) \times 1 + 1000$

Example 5

This should use **@CurProgNum** for both the main program and the subroutine headers. (Normally the main program would use **@MainProgNum**.) Furthermore, the Output Subs setting should be set to **Subs followed by Main**.

Subprogram numbering scheme (@CurProgNum)

By fomula
 Use sequence starting at 1
 Use block numbers

$(@MainProgNum + 0) \times 1 + (SubSequence\# + 0) \times 10 + 0$

Controllers That Use Block Numbers to Call Subprograms

Support subprograms

Output mode for subs

Absolute
 Incremental

Output subs

Main followed by Subs
 Subs followed by Main

Subprogram numbering scheme (@CurProgNum)

By fomula
 Use sequence starting at 1
 Use block numbers

$(@MainProgNum + 0) \times 100000 + (SubSequence\# + 0) \times 1 + 1000$

Subprogram Calls by Block Number

ArtiosCAD needs to know the block numbers assigned to the start of each subroutine before it can generate the main program. For this reason the main program must be last. There are special @directives that are inserted in the templates to record the block numbers of the starts of subprograms as they are encoded. These are subsequently referenced by the main program.

@MarkAsSub. The **@MarkAsSub** directive indicates that this location is the start of a subroutine. It must be used as part of the Subroutine Initialization template when the option **Subprogram calls by Block Number** is in use.

An example of such a subroutine initialization template might be:

Template	Example NC Code
<@MarkAsSub> (Start of subprogram)	N345 (Start of subprogram)
<@UnitsCode>	N346 G71
<@AbsIncCode>	N347 G91

The File Start template used in the example above contains **@NBlockOn** to start the block numbering on all subsequent blocks (including these). **@MarkAsSub** records the block number 345 as the start of the subroutine. The main program will later branch to this block number as the subroutine call.

Note:

Make sure that the block on which **@MarkAsSub** is placed ALWAYS generates data. Otherwise the block will not be encoded and the main program will try to branch to the wrong block or a non-existent block. Use some literal text, or make sure the directive is non-modal.

Subprogram Mirroring

The codes necessary to set and reset the mirroring of the designs are defined here. Some controllers require a code and data as shown above. In this example G39 sets a mirror based on the presence of some axis data. G39X1 will turn a mirror on in the X axis. (Usually, any value for X will cause a mirror in X). G38 turns the mirror off. Other controllers may simply use a code such as M11 to turn on a mirror in say the X-axis, M12 for the Y-axis, M13 for both X and Y axes, and M10 to remove the mirror. In either case, these are just strings of data that need to be encoded.

Subprogram mirroring (@IncludeMirCode, @IncludeMirResetCode)

Mirroring supported

Mirror in X: Mirror in X and Y:

Mirror in Y: Mirror reset:

(The GggXxxx method is usually a controller-supplied function, where as an M-code method is usually implemented by the machine integrator and will not normally be documented in the controller manual.)

@IncludeMirCode and **@IncludeMirResetCode** should be used in the Sub Call and Sub Call Reset templates as necessary.

When the mirroring function is not enabled, mirrored designs generate an additional subprogram for each additional transformation encountered. This uses more controller memory

Note:

If **Tool offsets done by Software** is checked on any of the Tool Property tabs, the mirroring function should not be enabled.

Subprogram Rotation

This is similar to setting up the Address and scaling for the Axis data. The Set and reset codes are similar to the setting of the mirror codes.

The rotation address and value are encoded only for the **@IncludeRotWord** directive.

If the rotation angle needs to be explicitly reset, the reset code can be defined as, for example G41R0.0.

The directives **@IncludeRotWord** and **@IncludeRotResetWord** should be used in the Sub Call and Sub Call Reset templates as necessary.

If the rotation function is not enabled, mirrored designs will generate an additional subprogram for each additional transformation encountered. This will use more controller memory.

Designs which are rotated 180 or mirrored in both X and Y are equivalent. If only mirroring or only rotation functions are available, only one subprogram is generated for designs that are so transformed.

Note:

If **Tool offsets done by Software** is selected on any of the Tool Property tabs, the rotation function should not be enabled. (See **Tool Offsets** under **Tools**.)

Start and End Points for Subprograms

Start subprograms at	End subprograms at
<input checked="" type="radio"/> Center of design	<input checked="" type="radio"/> Center of design
<input type="radio"/> First line	<input type="radio"/> Last line

The subprogram can be encoded to start at either the center of the design or the start of the first line. Similarly the end position can be chosen as the center of the design or the end of the last line. The latter in each case will save wasted moves. For CNC-type data this is the best choice. If the data is to drive, for example, a step and repeat plate making machine, the center point is most likely the correct choice.

When the main program is encoded in incremental mode, the End subprograms at group is unavailable and the subprogram starts and ends at the selected start point.

Codes Tab

Units @UnitsCode

These codes are used to identify the operating units. In turn they indicate how to interpret the linear axis data.

When **@UnitsCode** is used in a template, one of these data strings is encoded in the output data. The appropriate code is encoded depending upon the settings for units on the General tab. It is most likely used once at the beginning of the main program.

If it is likely that subprograms be run independently, it is prudent to use **@UnitsCode** in them as well.

Abs/Inc Mode @AbsIncCode

Absolute measurements are measured from the origin, while incremental measurements are measured from the last current position.

Abs/Inc Mode (@AbsIncCode)

Absolute code: Modal

Incremental code:

When **@AbsIncCode** is used in a template, one of these data strings is encoded in the output data. The appropriate code is selected dependent upon the absolute/incremental settings defined for the main program and subprograms on the General and Subs tabs. It is usually used once at the beginning of the main program and each of the subprograms. It is also frequently used after returning from a subroutine call.

Interpolation Code @InterpCode

Interpolation Code (@InterpCode)

Codes are tool dependent Modal

Rapid Traverse:

Linear:

Clockwise (CW):

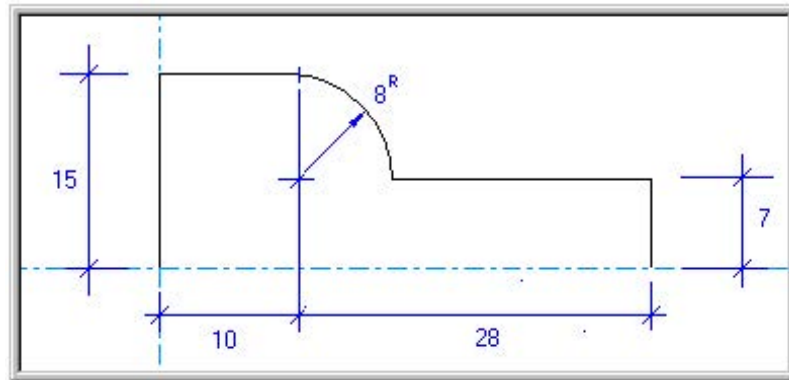
Counter Clockwise (CCW):

Minimum move distance for rapid traverse: in.

When **@InterpCode** is used in a template, one of these data strings is encoded in the output data. The appropriate code is selected dependent upon whether a rapid move, a line, a clockwise arc, or a counterclockwise arc is described.

The Modal Checkbox

Modality is the term used to indicate that once a state has been set by a code, it remains in effect until set by another code in the same modal group. With the Modal checkbox checked, when the @directive is encountered in a template, the appropriate data string is encoded in the output data only if the mode for that modal group needs changing.



The following code shows the difference between a modal and non-modal @InterpCode:

Modal OFF	Modal ON
G71 (mm units)	G71 (mm units)
G90 (Abs)	G90 (Abs)
G0 X0 Y0	G0 X0 Y0
M61 (tool on)	M61 (tool on)
G1 X0 Y15	G1 X0 Y15
G1 X10 Y15	X10 Y15
G2 X18 Y7 I0 J-8	G2 X18 Y7 I0 J-8
G1 X38 Y7	G1 X38 Y7
G1 X38 Y0	X38 Y0
M60 (tool off)	M60 (tool off)
M02	M02

The interpolation codes shown above are generally members of the same modal group. The group usually comprises the codes G0, G1, G2, G3 and G4. G4 generally indicates a dwell. For some reason it is usually included in the interpolation group.

Modality can apply to ordinate data as well as codes such as the interpolation method.

Modal OFF	Modal ON for ordinates too
G71 (mm units)	G71 (mm units)
G90 (Abs)	G90 (Abs)
G0 X0 Y0	G0 X0 Y0
M61 (tool on)	M61 (tool on)
G1 X0 Y15	G1 Y15
G1 X10 Y15	X10
G2 X18 Y7 I0 J-8	G2 X18 Y7 I0 J-8
G1 X38 Y7	G1 X38

Modal OFF	Modal ON for ordinates too
G1 X38 Y0 M60 (tool off) M02	Y0 M60 (tool off) M02

When ordinate data is modal, it is omitted if there is no change in the value. When the data is absolute, a repeated value may be omitted. When the data is incremental, a value of 0 may be omitted. The distance from the start of an arc to its center (indicated in this example by the addresses I and J) are normally incremental values, even when the program itself is absolute; hence the I0.

Minimum Move Distance for Rapid Traverse

Normally moves are made using rapid traverse interpolation (and consequently, high speed). However, some older machines perform better if small moves are performed at the speed used for linear interpolation. A minimum move distance of zero causes all moves to be carried out using rapid traverse interpolation.

If an intermediary speed is needed for bridges (faster than regular cutting but not as aggressive as rapid traverse), use CAM Tooling Setup to output bridges to a different tool. Assuming all other tools are set to require a tool up/off before deselection, set the select/deselect and on/off commands for the tool doing the move over bridges to do nothing.

Flute/Grain Code @FluteCode & Design Side Code @SideCode

Flute/Grain Code (@FluteCode)

Horizontal code:

Vertical code:

Design Side Code (@SideCode)

Inside code:

Outside code:

These are provided primarily to support Kongsberg sample makers. They are also useful for any other system that can use this information. For example, the codes could be used to set mirrors and rotations to control the orientation of the output onto the device if such a function were available.

Fly Mode @FlyModeCode

Fly Mode (@FlyModeCode)

Support fly mode

Fly mode ON code:

Fly mode OFF code:

Fly mode OFF code applies to:

Preceding block

Following block

Per tool

T	▲
T1	■
T2	■
T3	▼
.	

Use fly mode
for deviation angle <= deg.

Min radius: in.

Two common functions provided by CNC's are accommodated by this feature.

- In Position Logic, or IPL
- Acceleration ramps suppression.

The ArtiosCAD 'Fly Mode' can be used to control both IPL, where this is performed by switch codes, as well as acceleration ramps suppression.

In Position Logic (IPL)

IPL describes how a controller reaches a programmed end point. One might expect that a controller would always reach the true end point before processing the next block of data. However, this is not always the case. When machining a metal part, it is quite an advantage to very slightly round the corner of sharp transitions, as there is less de-burring to do on the machined part. Indeed, this is often the norm, and IPL is used to indicate where the rounding of corners is NOT to be performed. IPL may be controlled in one of two ways.

- Use of alternate interpolation codes
- On/Off switch codes to indicate where it is to be applied or not.

If the controller uses alternate interpolation codes to achieve this functionality, simply use these for the interpolation codes. This does imply, however, that IPL will either always be used or never be used. If, however, the controller uses switch codes, the ArtiosCAD Fly Mode feature may be employed, and IPL can selectively be turned on and off.

Acceleration Ramps Suppression

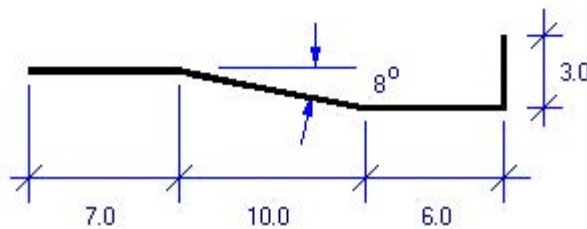
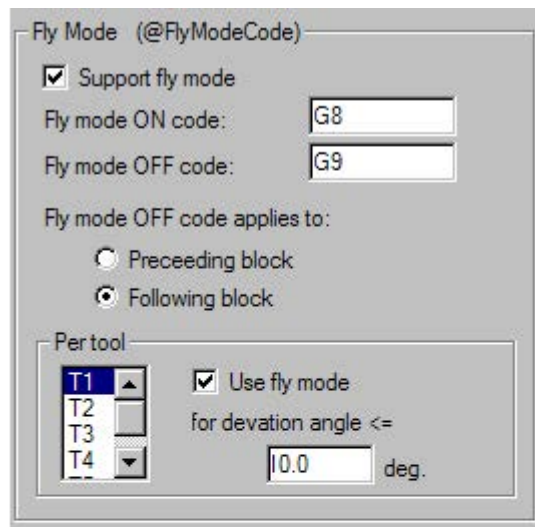
Many controllers decelerate at the end of each movement block so to more accurately come into position. When there are a series of collinear lines to cut, or close to collinear, it is not necessary to come to a halt at the end of each block. Some controllers support codes whereby the deceleration/acceleration between such movement blocks can be suppressed.

The Support of These Features in ArtiosCAD

ArtiosCAD supports both IPL and acceleration ramp suppression. However, if controller codes are used to suppress acceleration ramps there is a kind of double negative implied. The Fly mode ON code should use the code that suppresses acceleration ramps, while the Fly mode OFF code should use the code to indicate that acceleration ramps are enabled.

Some controllers require the Fly mode OFF code to be issued before the movement block on which it is to apply, while others require the code after the movement block on which it is to apply. This is because different manufacturers provide the same functionality in different ways.

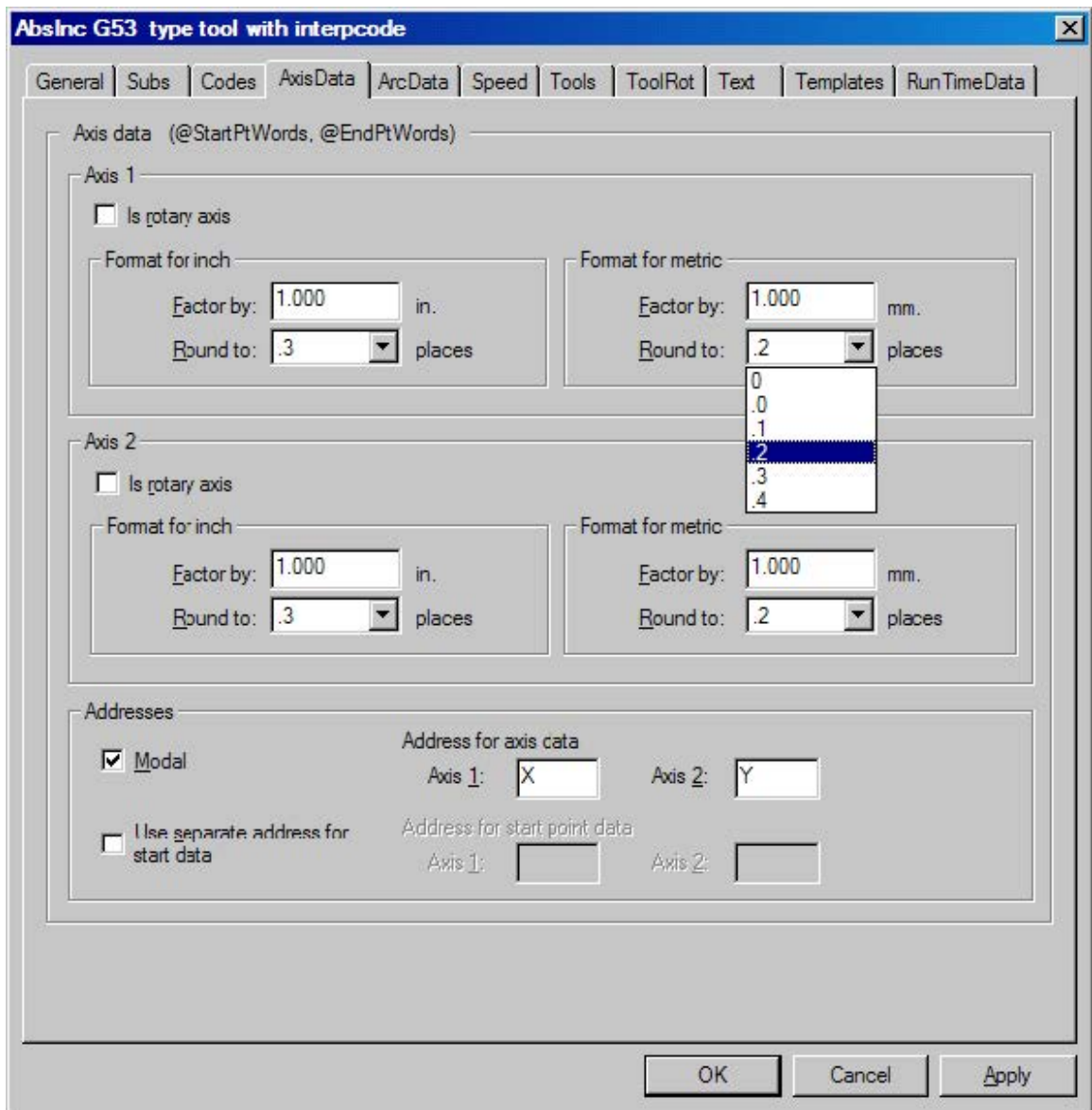
The maximum deviation angle at which the feature activates is configurable for each tool.



The above example might produce code that looked like:

```
M61
G8
G1X7.0
X10Y-1.41
G9
X6.0
Y3.0
M60
```

Axis Data Tab



Axis 1 / Axis 2

This tab defines the basic format and scaling for the axis movement data. Separate scaling and factor settings are available for inch and metric output. The same NC Export tuning may therefore make data for either unit system. For example, when ordinate data is output with no decimal places, metric data is usually factored by 100, while imperial data is usually factored by 1000.

Distance to move	Required encoded data	Factor	Round to: <i>nn</i> places	Resulting pixel size
1.23	123	100.0	0	0.01
1.23	1230	1000.0	0	0.001
1.23	1230.	1000.0	.0	0.001

Distance to move	Required encoded data	Factor	Round to: <i>nn</i> places	Resulting pixel size
1.23	1.23	1.00	.2	0.01
1.23	1.230	1.00	.3	0.001
1.23	31.2	25.4	.1	0.003937

The scale factor does not need to be a power of 10.

The Round to: *nn* places field indicates whether a decimal place is required or not, and the number of decimal places that should be encoded. It is not a rounding value.

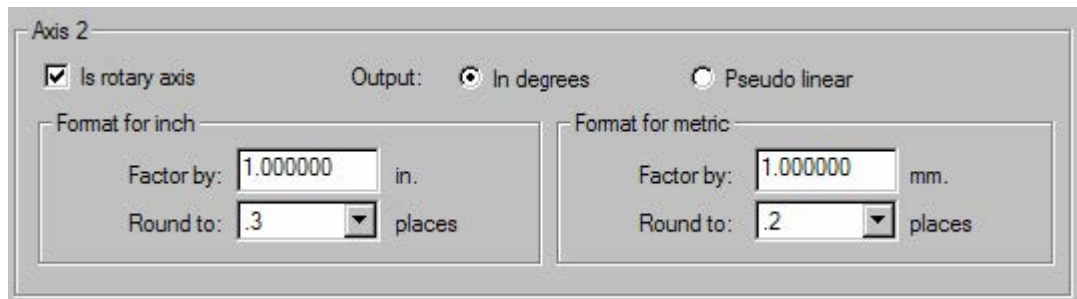
The pixel size is:

$$1 / (\text{Axis factor} * 10^{(\# \text{ of decimal places displayed})})$$

When encoding arcs, it is sometimes necessary to very slightly adjust the center points or the radius value to account for the fact that the start and end coordinates have been pixelized. These values are set to multiples of this pixel value. The first non-rotary axis pixel value is used for this purpose.

Trailing zeros can usually be removed from data encoded with a decimal point. This is set by the option **Trim trailing zeros from data** under **Readability** on the General tab. Sometimes it is also possible to trim the trailing decimal point. It depends how the controller interprets numbers without decimal points.

[This Axis] Is [A] Rotary Axis



A die cutter for cutting rotary die boards has one rotary axis. The machine builder may have defined this as a true rotary axis using angular displacement values, or may drive it as a pseudo-linear axis with an implied working radius.

For a true angularly-controlled rotary axis, the values passed to this driver are in degrees. If the rotary axis values need to be in radians, set the **Factor by** value to:

$$\pi/180 = 0.01745329$$

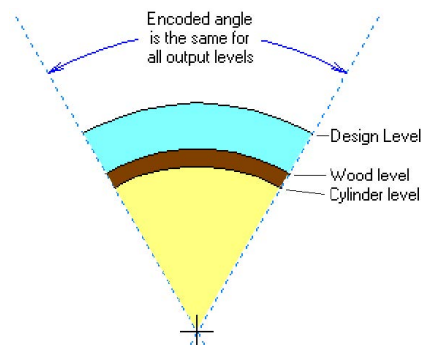
@RotaryRadius

Sometimes it is necessary to provide a working radius for the controller to compute a tool speed at the working radius. When a rotary die board is output, the radius is available as the value **@RotaryRadius**. Use this to encode the necessary block.

An example of a main program initialization template might be:

Template	Example NC Code
(DFS, <@MainProgNum>)	(DFS, 100)
<@UnitsCode>	G71
<@AbsIncCode>	G90
G20 R<@RotaryRadius>	G20 R250.00

The @RotaryRadius value is computed from the cylinder radius defined for the die press together with the rotary shrink and stretch factors. These are used to compute the appropriate radius for the output level set on the View tab of the Output. The output level may be Design, Wood level or Cylinder level. The level also affects how the geometry is scaled.



Note:

The resulting angular data should be the same regardless of the level is used. (The lines to be output are differentially scaled by the Output processing to reflect the selected level. However, since they are then divided by the appropriate radius, the resulting angle is the same.) If the Rotary defaults have been modified for the individual workspace being output, and these changes were not done consistently with the thickness of wood or cylinder diameter in mind, the resulting radius may not be set to the correct working level. **It is important that this data is set up correctly.**

Pseudo Linear

When **Pseudo linear** is selected, the controller uses its own internally-set value to establish how to scale the data to produce a correctly cut die. If the NC data produced does not use an **@RotaryRadius** value for example, it is necessary to match the output level used by ArtiosCAD to produce the NC data with matching parameters in the controller. Contact the equipment manufacturer for information on how to do this.

There is actually little difference between selecting **Pseudo linear** and not checking the **Is rotary axis** checkbox. The data is differentially scaled by the Output to the level set on the View tab in the Output definition.

Addresses

The addresses (command indicators) for the two linear axes are defined in the above example. This might be used for a regular controller where the two primary axis addresses are X and Y. The X and Y data is marked as being modal. These addresses are used when encoding **@StartPtWords**, **@EndPtWords** and **@IncludePtWords**.

Example Template	Example Generated Code
<@StartPtWords>	X20.0Y-2.5 X50.0 Y2.5 X20.0Y-2.5

Data formats such as CFF2 have **Line** and **Arc** descriptions that have both the start and end points in the same data line.

Use Separate Address for Start Data

Checking **Use separate address for start data** allows different addresses for first and second axis data for the start and end point. It also removes the Move template from the Template tab.

CFF2 also uses commas between data values. The comma is used as the address.

Example Template	Example Generated Code
L,2,<@LineType>0 <@StartPtWords><@EndPtWords>, <@BridgeNum>,<@BridgeWidth>	L,2,140,0,10.0,20.0, 80.7107,90.7107,0,0.0

The bold commas are those generated by the addresses. The small ones are text literals in the template.

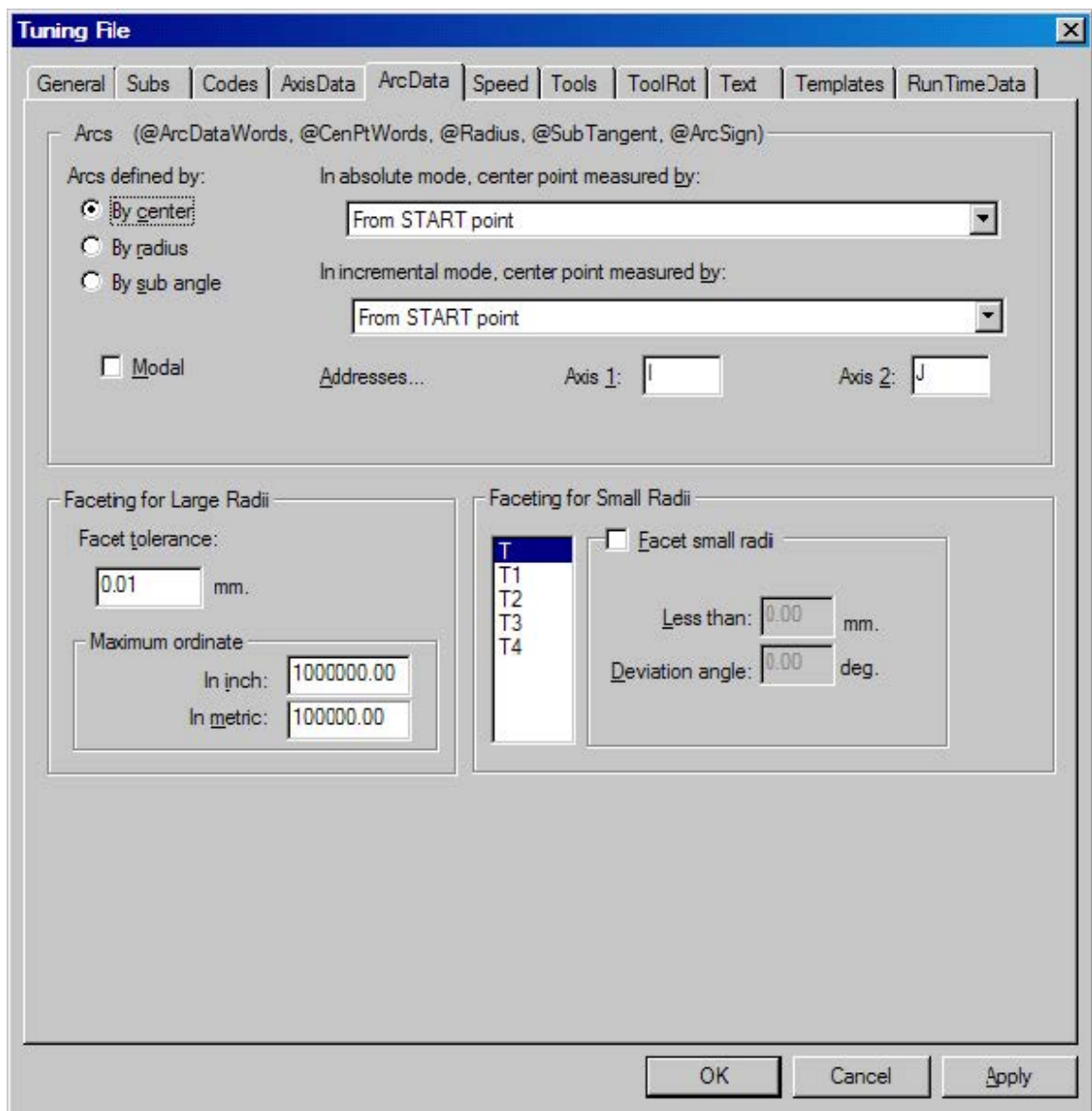
The addresses could be left blank, and all commas defined in the template. However, the first and second axis data would have to be defined individually to allow the comma to be placed between them. This may be achieved with @directive modifiers as shown. In this case, A1

indicates that only axis 1 data is encoded. Modifiers are described after the complete list of @directives.

Example Template	Example Generated Code
<pre>... <@StartPtWords[A1]>, <@StartPtWords[A2]>, ...</pre>	<pre>... 10.0,20.0, ...</pre>

Arc Data Tab

Settings on this tab controls the way the GNC driver encodes the @CenPtWords, @Radius, @Subtended, and @ArcSign directives.



Arc defined by controls the data encoded in **@ArcDataWords** as well as the encoding of full circles. The first two methods are most commonly used for normal NC data formats. The Subtended Angle method is required by formats such as HPGL.

Arc Defined by Setting	Actual @directive used for <@ArcDataWords>	Typical Data Generated by <@ArcDataWords>
By center	@CenPtWords	I19.69 J5.27
By radius	@Radius	R10.0
By sub angle	@Subtended	A30.0

All the above @directives are encoded regardless of how **Arc defined by** is set.

The formatting definitions on the AxisData tab are used for formatting **@CenPtWords** and **@Radius**. The formatting for **@Subtended** is on this tab.

HPGL arc formats require the center point and the subtended angle to be defined, but do not require the end point to be defined. The hand (clockwise/counter-clockwise) of the arc is defined by the sign of the subtended angle. It is therefore necessary to set up addresses and formatting for both methods and use both **@CenPtWords** and **@Subtended** rather than just **@ArcDataWords** (you can always use the specific @directives rather than **@ArcDataWords**).

Arcs Defined by Center Point

Normally, arcs are defined by an interpolation mode, an end point and by either a center point or a radius. These are reflected by the first two choices on the Arc menu.

When the arc is defined by the center point, there are three further options.

- From START Point (shown)
- From END Point
- In absolute

The center point is usually defined by incremental distances measured from the start point, though in rare situations it is measured from the end point. Either way, the data is normally incremental whether the primary axis data is incremental or absolute. On rare occasions, when the primary axis data is absolute, the center point is also given in absolute coordinates.

Select the appropriate settings from the drop-down list boxes. The example dialog box on the previous page is by far the most common arrangement.

Example Template	Example Generated Code
<pre><@InterpCode><@CenPtWords> <@EndPtWords> <@InterpCode><@ArcDataWords> <@EndPtWords></pre>	<pre>G2 I-4.44J-69.86 X65.0Y-61.0</pre>

@CenPtWords formatting is controlled by this setting. HPGL uses a subtended angle and the center point to define an arc. It is therefore necessary to make settings for the **@CenPtWords** directive while the option button is set to **By center**.

Arcs Defined by Radius

Arcs (@ArcDataWords, @CenPtWords, @Radius, @SubTangent, @ArcSign)

Arcs defined by:

By center
 By radius
 By sub angle

Modal

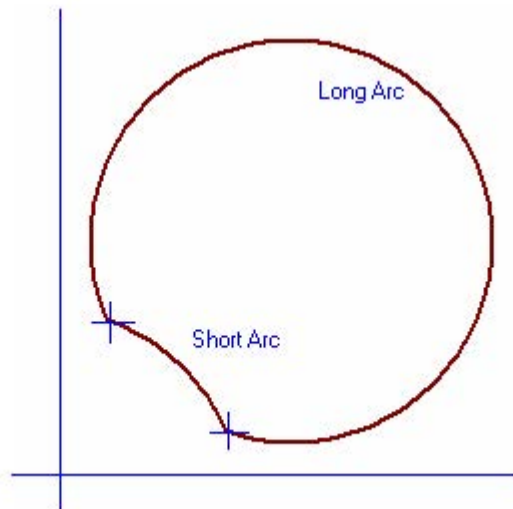
For CW arcs, positive radius indicates:

Short arc Long arc

For CCW arcs, positive radius indicates:

Short arc Long arc

Address for radius:



When using **By radius**, it is necessary to say whether a positive radius represents the short radius or the long radius. It is possible for the sign of the short radius to be positive for a clockwise arc and negative for a counterclockwise arc. Consult the manual for the controller to determine which choices to select, as there is no rule about how these are defined.

The resulting code for the arc template data:

```
<@InterpCode><@ArcDataWords><@EndPtWords>
```

might be:

```
G2 R70.0 X65.0Y-61.0
```

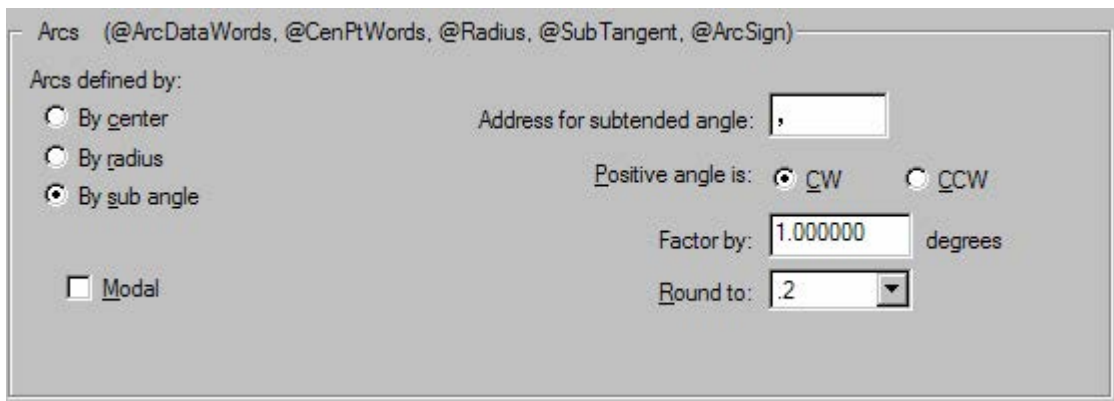
Example Template	Example Generated Code
<pre><@InterpCode><@Radius> <@EndPtWords></pre>	G2 R70.0 X65.0Y-61.0 (short - if set as shown)
<pre><@InterpCode><@ArcDataWords></pre>	G2 R-70.0 X65.0Y-61.0 (long - if set as shown)

Example Template	Example Generated Code
<@EndPtWords>	

Splitting of Full Circles

Full circles may not generally be defined using this method. For that reason, when this method of encoding is chosen, full circles are divided into two arcs, one of 90 degrees and one of 270 degrees. 180 is not used because very small rounding errors in the encoding of R will result in (larger) changes in the position of the center of the circle.

Arcs Defined by Subtended Angle

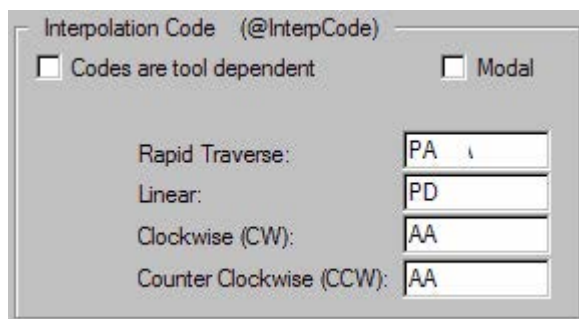


This option is provided primarily to support HPGL. However, HPGL defines an arc by both the center point and a subtended angle. Therefore, use both **@CenPtWords** and **@ArcDataWords** (with the radius button set to **By sub angle**) to encode the HPGL arc command.

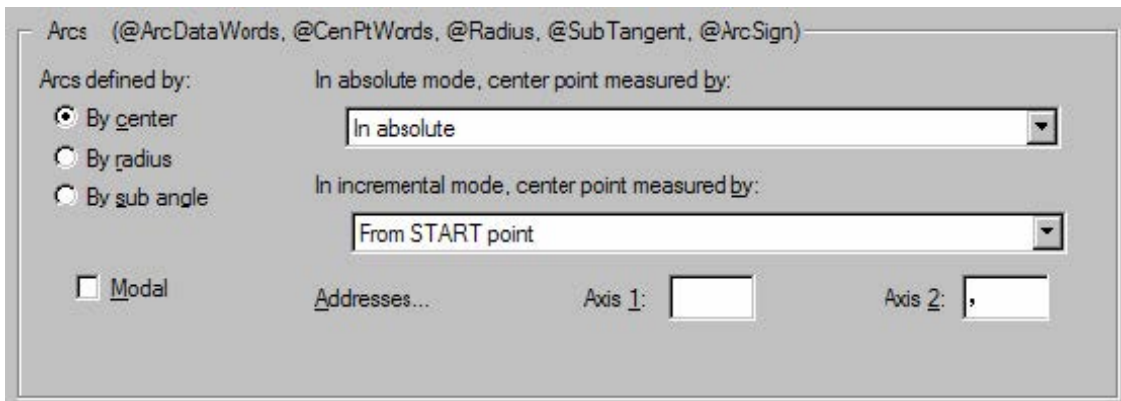
The syntax of the HPGL arc command is:

AAxxxx,yyyy,aaaa,c; where xxxx,yyyy is the absolute coordinates of the center of the arc, aaaa is the subtended angle, and c is chord control (not always used).

The AA should be set up as the circular interpolation code for both CC and CCW arcs on the Code tab. Make sure they are not modal.

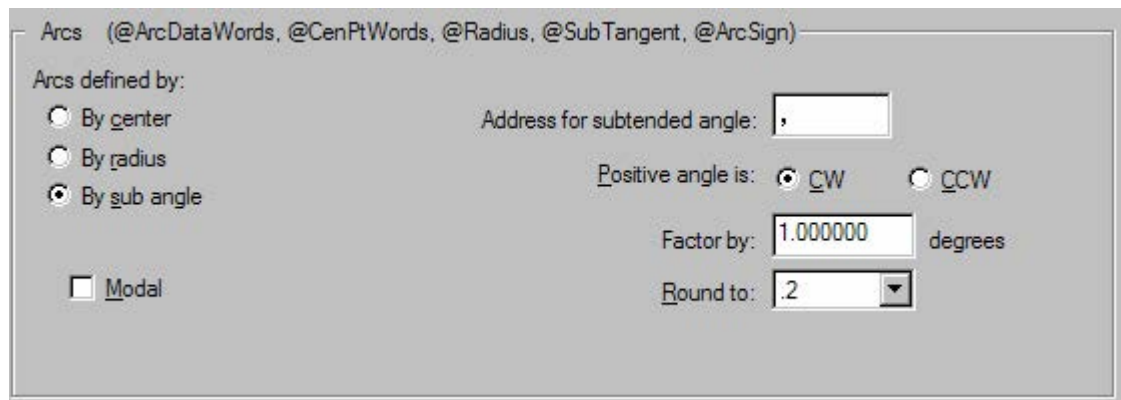


On the Arc Data tab, first set Arcs defined by: to **By center**.



@CenPtWords must be defined as having absolute coordinates, while the address for axis 1 should be a null entry and the address for axis 2 should be a comma.

Next, set Arcs defined by to **By sub angle** where it should remain.



The subtended angle (controlling **@ArcDataWords**) may be set up as shown above. It is important that Arcs defined by is set to **By sub angle**, to force **@ArcDataWords** to generate the subtended angle code.

Example Template	Example Generated Code
<pre><@InterpCode><@CenPtWords> <@Subtended>,1 <@InterpCode><@CenPtWords> <@ArcDataWords>,1</pre>	<pre>AA-177,-2794,-79.11,1</pre>

It does not matter whether the option button is set to **By center** or **By sub angle**, so long as the specific **@directives** are used (as shown).

Arcs Defined by Start and End Angles

@AtAngle and **@FinAngle** control tangential tools and generate data only when the tool is defined as tangential. When the definition of arcs use these **@directives**, it is necessary to

set **ALL** tools to be tangential. Machines that use this method of defining arcs and that have tangentially controlled tools will use some code by which to define whether or not the tool is tangential. This can be placed in the tool selection/deselection template. The direction of the tool is controlled directly by the controller. An example of such a machine is the Wild sample maker.

Special Cased Arc Processing

- **Very Short Arcs:** Arcs having start and end points closer than 1 pixel are converted to a straight line. For example, if the format for axis data is .3, an arc with length of only 0.001 units long is cut as a straight line. This ensures a full circle is never generated in place of a short arc.
- **Full circles with very small radii:** If the arc is a circle with radius smaller than 1 pixel, then the arc is output as a line.
- **Full circles output using the radius method:** (rather than center points method) are output as two arcs (1/4 full circle and 3/4 full circle) to avoid an ill-defined arc.
- **Modality of arc data:** In absolute mode, **@EndPtWords**, **@CenPtWords**, and **@ArcDataWords** are not modal unless the MODAL keyword is specified.

Faceting for Large Radii

Faceting for Large Radii

Facet tolerance:
 mm.

Maximum ordinate

In inch:

In metric:

Arcs with a radius bigger than the maximum value that a controller can decode can not be directly encoded. Consider a segment of a 10000 mm radius that needs to be encoded. A sample encoding could be:

```
G3 X834850 Y400000 I1000000 J0
```

If the controller can only decode 6 significant digits (999999), the radius may not be so described even though any ordinate along the arc is within the addressable range.

When such an arc is encountered, it is faceted. The maximum deviation from the true arc will not exceed the **Facet tolerance**.

The maximum addressable ordinates for inch and metric units should be provided.

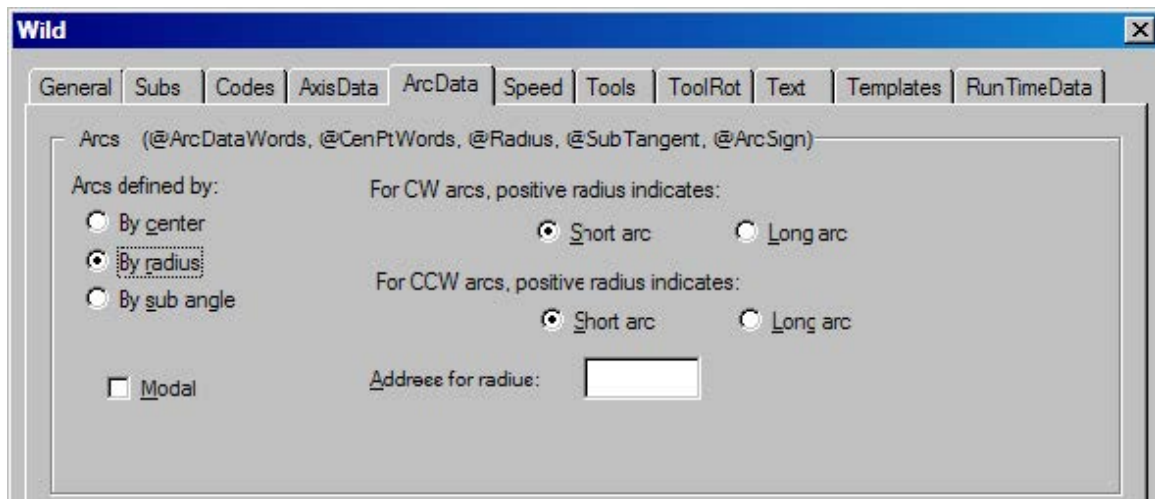
Controllers Unable to Process Arcs

Some rotary machines use a pseudo linear axis for the rotary axis and some use an angular axis. When using an angular axis, it is not always possible to use clockwise and counterclockwise interpolation methods. (The controller is not able to compute the interpolation unless an effective radius is provided. Not all implementations appear to use this necessary value.) In this case, all arcs need to be faceted so that all movements use linear interpolation.

Set the maximum ordinate to **0** to achieve this.

Controllers Requiring Different Formatting for Full Circles

Some controllers require full circles to be defined in a different way than that used for general arcs. Since the GNC driver does not provide this functionality, it is necessary to set the Arcs defined by option button to **By radius** even if actual data to be encoded does not use the By radius method. In this situation **@ArcDataWords** must *not* be used.



When outputting to a Wild, there are further considerations. The angles required are measured from the center point to the start of the arc, and the center point to the end of the arc rather than being the actual tangent angles. To achieve this, add the **RADIAL** keyword to **@AtAngle** and **@FinAngle**.

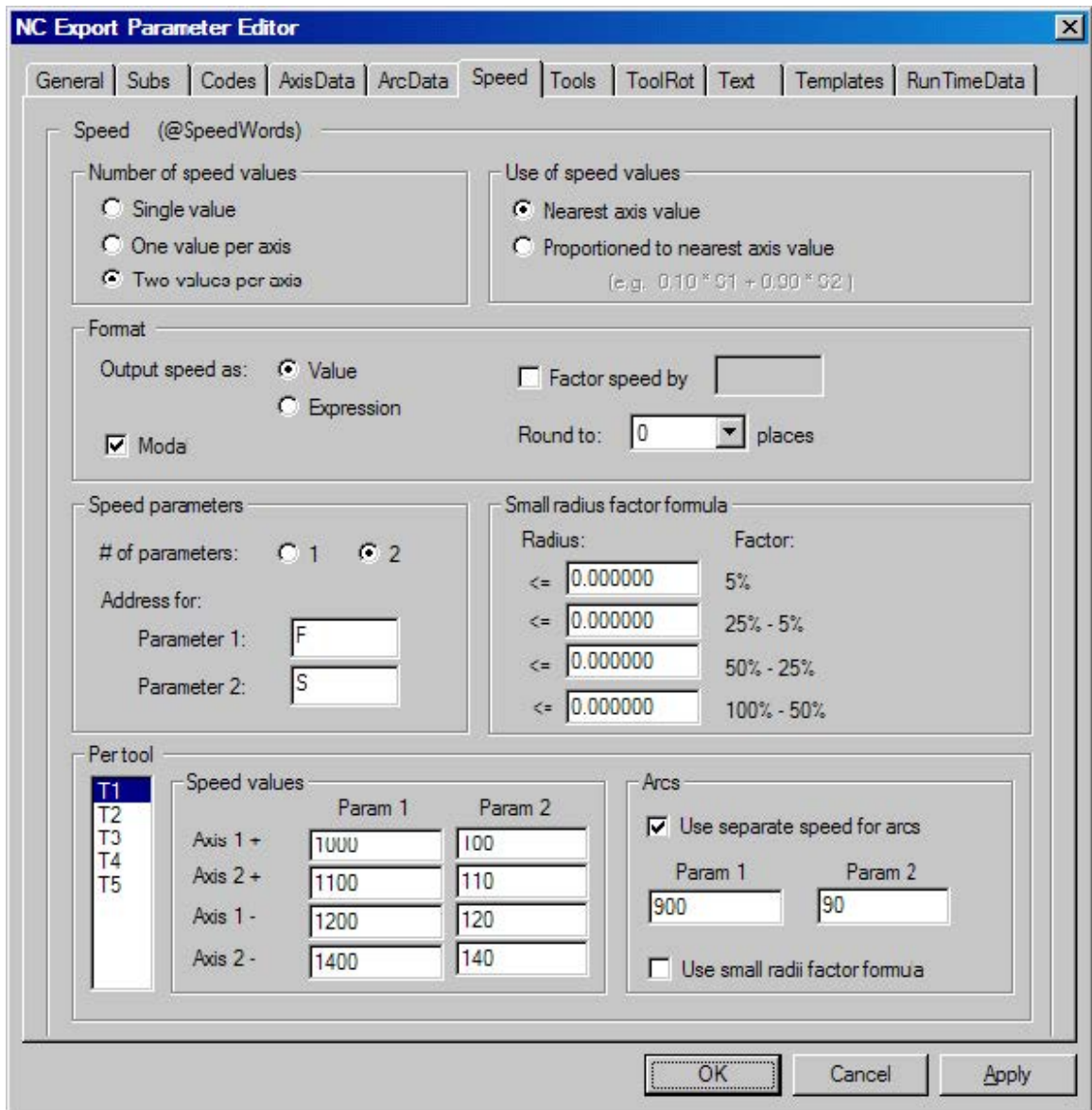
The syntax of the Wild arc command is:

`E`*x*, *y**c*, *rad*, *hand*, *acs*, *ace* where **E** indicates an arc, *x*_{*c*}, *y*_{*c*} is the center of the arc, *hand* C=clockwise or A=anti- or counterclockwise, *acs* is the angle from the center point to the start, and *ace* is the angle from the center point to the end.

Example Template	Example Generated Code
<code>E<@CenPtWords>,<@Radius>,<@InterpCode>,<@AtAngle[Radial]>,<@FinAngle[Radial]>,<@SubTangent>,<@ArcSign></code>	<code>E5005,5005,5000,C,27000,24000E5,-3655,4999,A,24000,27000</code>

Example Template	Example Generated Code
<@FinAngle [Radial]>	

Speed Tab



The slot cut by laser machines in steel rule cutting dies must be carefully controlled. The slot width is a function of many things:

- The dryness and density of the plywood.
- Grain direction and the number of ply in each direction.
- The beam diameter, mode, and power.
- Nozzle to wood distance.

Controllers have varying degrees of sophistication. Some provide independent speeds and height settings for all four directions of cut, with the ability to factor these when a cut does not lie directly in one of these axis directions. The least complex scheme is where a single speed is used for all directions, with degrees of complexity between these extremes depending upon the functions available in the controller.

The term **Speed** is used here to cover the controls that are available for setting speed and whatever other characteristic is chosen.

The Speed controls available with this driver allow for the very simple single fixed speed all the way through the two parameter proportioned control. These parameters are normally speed, and either power or height of the nozzle above the work piece. The driver provides up to two parameters; you get to decide how best to use them depending on what is controllable and what is not.

Controllers Basing Speed on Tool Selection

Some controllers deal with all speed control issues directly. For each particular 'tool' (e.g. 2pt cutting) there may be settings for speed, laser beam power, pulse or CW, nozzle height and so forth. There may be speed and nozzle height settings for each of the four axis directions or not. It depends how sophisticated a solution the machine integrator has provided. There may be sets of tables of these tool parameter settings. For example, there may be a set for cutting 19 mm 17 ply birch, another for 12 mm 5 ply maple and another for ½ inch clear plastic. These are often referred to as material tables.

Once a material table is selected on the controller, the selection of a tool in the NC data (such as T1) sets all the speed, height, and power characteristics necessary to control the tool speed for that particular material.

In this case, this @Speed section may be completely ignored, and no @SpeedWords directives are needed on any of the movement templates.

(The mapping of type of cut in the ArtiosCAD design to the tool on the machine is set in the CAM Tooling Setup catalog.)

Controllers Basing Speed on Stored Variables and a Display Screen

A common implementation is when the machine builder provides a data entry screen on the controller which allows the user to set values for a set of fixed variables. These variables are in turn used to set the speed for particular tools. Usually, the data entry screen can be brought up and the value for a variable changed while the machine is running a program. This becomes almost identical to the situation in the previous section. The only difference is that it is necessary to know the variable name used for each of the speed functions for each of the tools.

Set Output speed as to **Expression**.

The machine integrator may have allocated different blocks of variables to be used for different materials. The NC data must then be encoded with the right block of variables for the material being cut.

Suppose the following scheme is used for the different materials (#*nnn* is a variable name):

Material	Variables for Cutting Closest to the X Axis	Variables for Cutting Closest to the Y Axis
19 mm birch	#101	#102
12 mm maple	#111	#112

The code generated for an L shape might be:

Material	Generated Code
19 mm birch	G1 X-100.00 F#101 Y200.00 F#102
12 mm maple	G1 X-100.00 F#111 Y200.00 F#112

The run time data feature of the General NC driver has a Data Set function which provides a simple way to select the appropriate variables. At the time an Output is run, the user simply selects the appropriate Data Set corresponding to the material to cut.

An alternative is to set up a different set of General NC driver tuning for each material, and set up a different Output for each material using this tuning, however, this is more work to set up and more complicated to use.

Controllers Providing Basic Speed Control

Some controllers provide only basic speed control in which a speed value is simply encoded in the NC data (e.g. F1000). However, the GNC driver provides full differential speed control for controllers that support only simple speed values. The speed values are evaluated and coded in the NC data when it is generated. This means that the differential speed may not be changed during the execution of the program. Only the overall speed can be varied by setting the speed control on the controller itself. For such controllers, set Output speed as to **Value** as shown in the example below.

The values set in the Run Time Data dialog box are cached for each output. They are reestablished the next time the output is used. Run some test cuts on the laser machine to establish suitable "speeds for the day." These may then be used until reset. Thus, even the simplest controller can be driven in a manner close to that of the more sophisticated ones. However, it is not possible to change the relative differential speeds between axes

Format

Format

Output speed as: Value Factor speed by

Expression

Modal

Round to: places

Output Speed As

Read the next few sections about the various controller types supported before setting this value. As a basic rule, if your controller uses speed information stored in variables, use **Expression**; otherwise, use **Value**. If your controller does all its own speed control, skip this whole section.

Factor Speed By

Occasionally when variables are used, it is necessary to factor the values stored in them. This is usually when switches control the speed rather than a screen-based user interface. Normally it may be ignored.

Round To

When **Value** is the method, this formats the computed data.

Number of Decimal Places to Round to	Example Code
0	F1250
.2	F12.50

When **Expression** is the method, it controls the number of decimal places used for the factors.

Round to	Factor	Example Code
.1	100	F[#101*87.4+#102*12.6
.2	1 (or off)	F[#101*0.87+#102*0.13]

Modal

Speed @directives can be modal. If there is no change in the value, they are omitted.

Number of Speed Values

The Number of speed values: group has the following options:

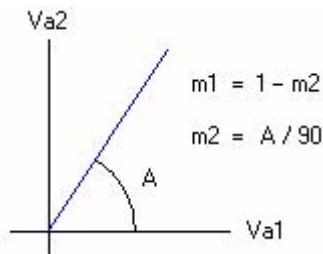
- **Single value:** The same speed is used in all directions of travel.
- **One value per axis:** A different speed is provided for traveling in each axis.
- **Two values per axis:** A different speed is provided for traveling in the positive and negative direction of each axis.

If the controller sets speeds based on stored variables, set the number of speed values to match the controller configuration. If actual speed values are being supplied here, choose the one that best suits your needs.

Use of Speed Values

<p>Number of speed values</p> <p><input type="radio"/> Single value</p> <p><input type="radio"/> One value per axis</p> <p><input checked="" type="radio"/> Two values per axis</p>	<p>Use of speed values</p> <p><input type="radio"/> Nearest axis value</p> <p><input checked="" type="radio"/> Proportioned to nearest axis value</p> <p>(e.g. $0.10 * S1 + 0.90 * S2$)</p>
---	--

If more than one speed is provided, there are two further options. The speed of a diagonal line can be set to that of the closest axis, or to a speed proportioned between the two values.



Nearest Axis Value	Proportioned Between Axes
[Va * Factor]	[[Va1*m1*Factor] + [Va2*m2*Factor]]

The angle of the line is 11.3 degrees.

```

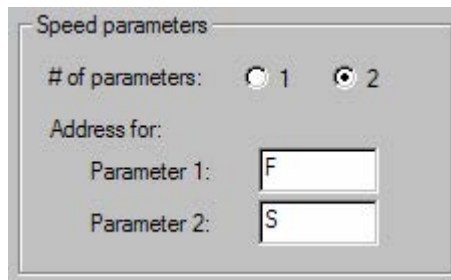
A = 11.3 degrees
Va1 = 1000
Va2 = 1100
m2 = 11.3/90 = 0.126
m1 = 1-0.1256 = 0.874
V = 1000*0.874+1100*0.126 = 1012.6
    
```

If the proportioned method is used, a controller using variables must also be capable of using expressions. This is probably the case, but should be checked. An sample expression is:

```
F[#101*0.874]+[#102*0.126]
```

It is understood that a simple proportion between the axes is strictly not correct, however, given that the speed values in each axis are likely to be similar, the method is reasonable.

Speed Parameters

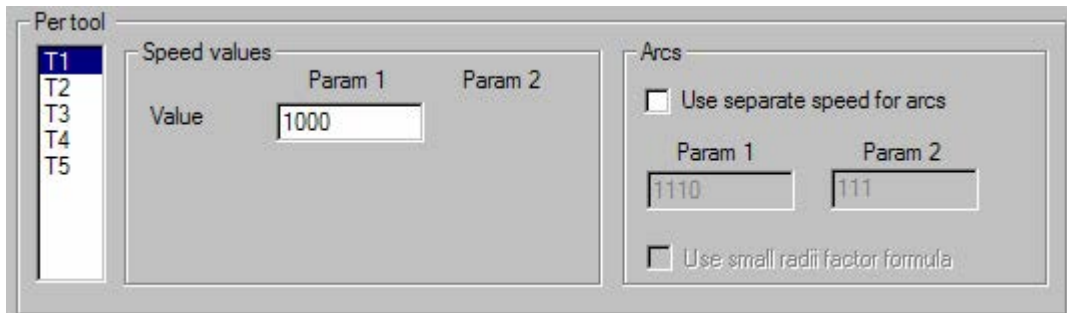


You can set one or two speed parameters. As discussed previously, the second parameter is not really a speed setting. For a laser die cutter, it is likely a nozzle height setting, a power setting, or a pulse mark-space setting. This also needs to be set in a similar manner to that chosen for speed (1, 2, or 4 values, proportioned or not). The addresses for each parameter should be provided. For a CNC, the true speed address is almost certainly F, while for HPGL it is most likely VS.

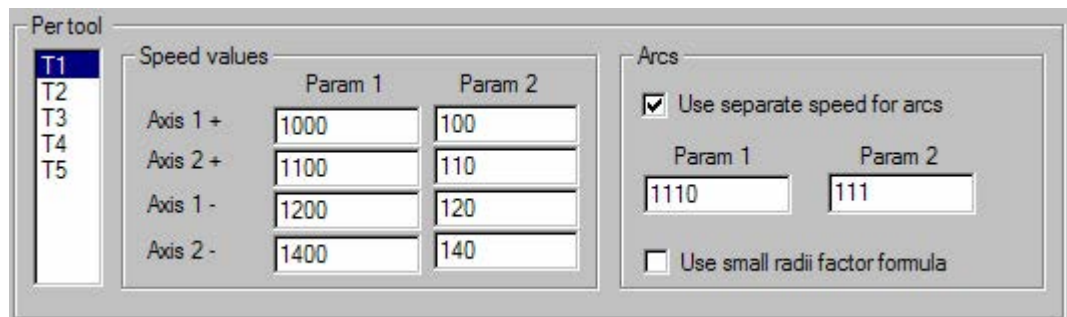
If more than one parameter is controllable (such as the nozzle height) find out the address for that parameter.

Per Tool

The number of speed settings per tool depends upon the **Number of speed values** and the number of **Speed parameters** selected.



Shown above is the result of selecting one single value and one speed parameter. Shown below is the result of selecting two values per axis and two parameters.



Also shown in the example above is a separate set of speed values for cutting all arcs. It is not uncommon to cut arcs with a looser cut. It is easier to place bent rule into a very slightly wider

slot, but there is very little chance of it working loose because it is bent. Straight horizontal and vertical crease rule kerf, however, needs careful control.

What Data Gets Entered?

As discussed previously, there are three basic classes of controllers:

- Controller performs all speed control functions by itself.
- Speeds based on stored variables.
- Speed values need to be encoded in the NC data.

If your controller is the first type, you could have skipped this section. You still can.

Speeds Based on Stored Variables

This class of controller has memory locations stored in the controller. These can be referenced by the variable name and the values stored in them used as data in the NC code. Often these are used for setting speeds and other parameters to control the way the laser cuts the different kerf widths.

The machine integrator usually provides some form of user interface to easily update the values. The format of the variable names is a function of the controller. Examples of such names are: #100, V100, \$100, X1, XS1, and so on. Valid names are described in the controller manual. However, unless the machine integrator uses them and provides a user interface, you may not be able to use them. The machine integrator must also supply information indicating which variables control what functionality.

NC Export Parameter Editor

General | Subs | Codes | AxisData | ArcData | Speed | Tools | ToolRot | Text | Templates | RunTimeData

Speed (@SpeedWords)

Number of speed values

Single value

One value per axis

Two values per axis

Use of speed values

Nearest axis value

Proportioned to nearest axis value
(e.g. 0.10 * G1 + 0.90 * G2)

Format

Output speed as: Value Expression

Modu

Factor speed by:

Round to: places

Speed parameters

of parameters: 1 2

Address for:

Parameter 1:

Parameter 2:

Small radius factor formula

Radius:	Factor:
<= 0.000000	5%
<= 0.000000	25% - 5%
<= 0.000000	50% - 25%
<= 0.000000	100% - 50%

Per tool

	Param 1	Param 2
T2	<input type="text"/>	<input type="text"/>
T3	<input type="text"/>	<input type="text"/>
T4	<input type="text"/>	<input type="text"/>
T5	<input type="text"/>	<input type="text"/>

Speed values

Axis 1:

Axis 2:

Arcs

Use separate speed for arcs

Param 1:

Param 2:

Use small radii factor formula

OK Cancel Apply

Shown above in the Speed values group is an example of a machine using one speed variable for each axis. The variable used for axis 1 is #101, and for axis 2, #102. The variable names for each of the axis speeds for each of the remaining tools should be filled in accordingly.

Note that Output speed as is set to **Expression**.

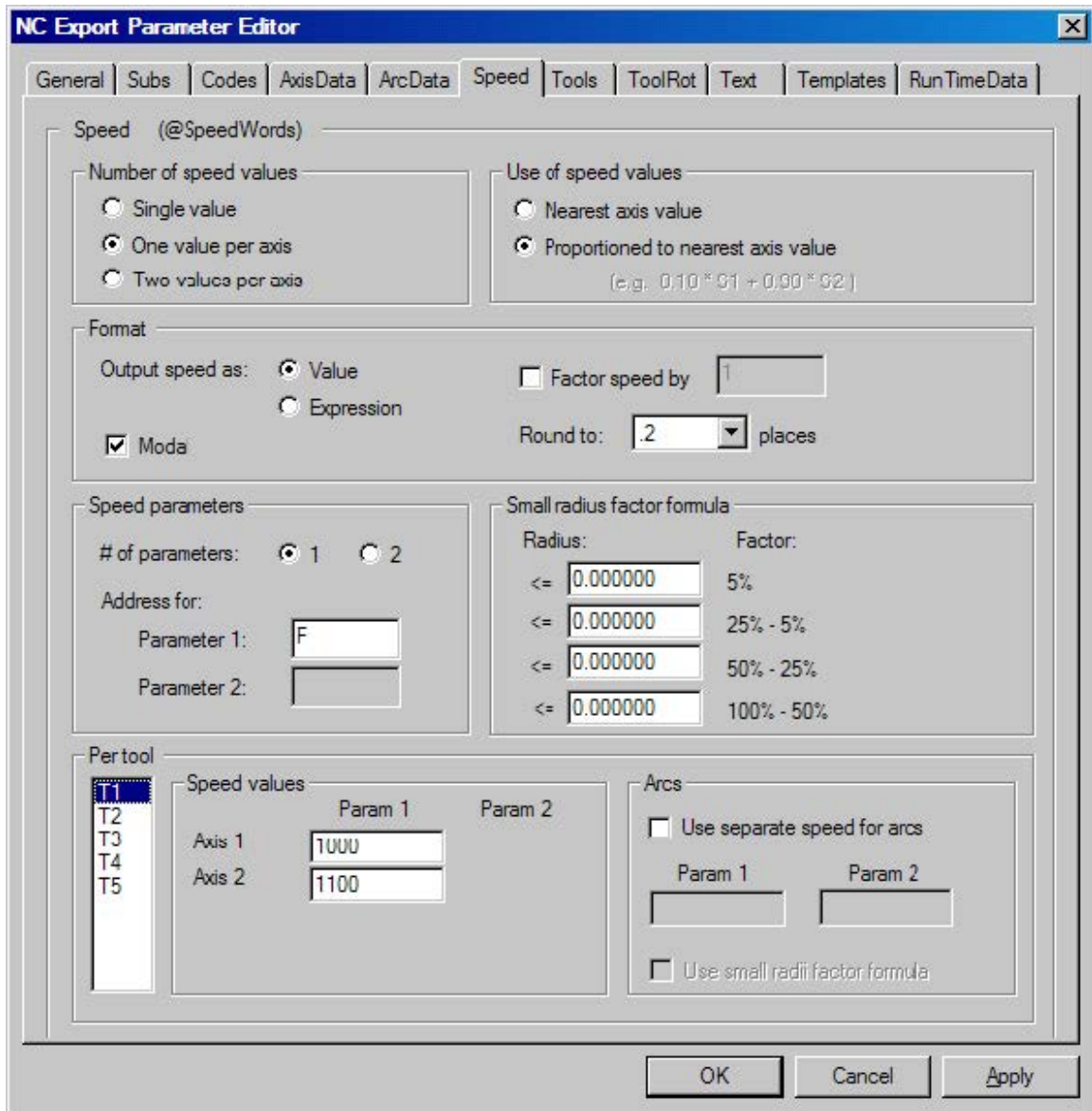
Such a setup causes the **@SpeedWords** directive to encode data like:

```
F[#101*0.87+#102*0.13]
```

Speeds Based on Values

This method is used when the controller uses speeds directly encoded into the NC data. A controller that uses this method is probably rather unsophisticated. Most likely it has one speed control which sets a percentage of the programmed speed to use. This is then used for all speeds set.

The GNC driver Run Time Data (RTD) feature allows such data to be supplied conveniently each time the Output is used. Controllers with limited speed control are prime candidates for using RTD. There are some brief notes on RTD at the end of this section.



Shown above is an example which has speed values encoded into the NC data.

Note that Output speed as is set to **Value**.

Such a setup causes the **@SpeedWords** directive to encode data like:

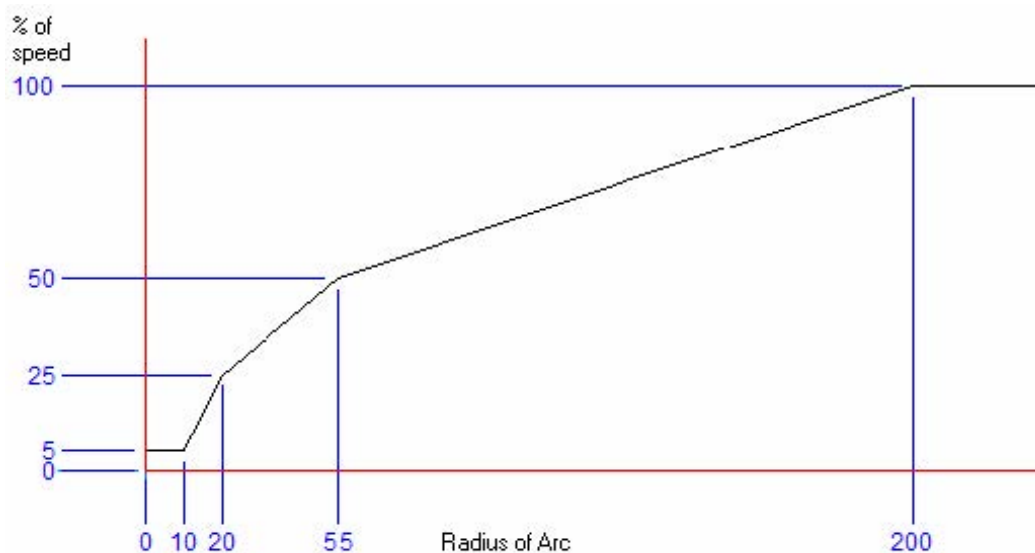
F1013

Small Radii Factor Formula

Traveling around small radii can cause high accelerations. For a given speed, the smaller the radii, the higher the accelerations. There is a limit to the acceleration that the motors driving

a particular machine can provide. Modern controllers usually have parameters included and are set up by the machine integrator to limit the accelerations when going around small radii. Machines with less sophisticated motion control systems need to have lower speeds set for smaller radii. The **Small radius factor formula** provides this functionality.

Arcs		Small radius factor formula	
<input checked="" type="checkbox"/>	Use separate speed for arcs	Radius:	Factor:
Param 1	Param 2	<=	10.000000 5%
100	100	<=	20.000000 25% - 5%
<input checked="" type="checkbox"/>	Use small radii factor formula	<=	55.000000 50% - 25%
		<=	200.000000 100% - 50%



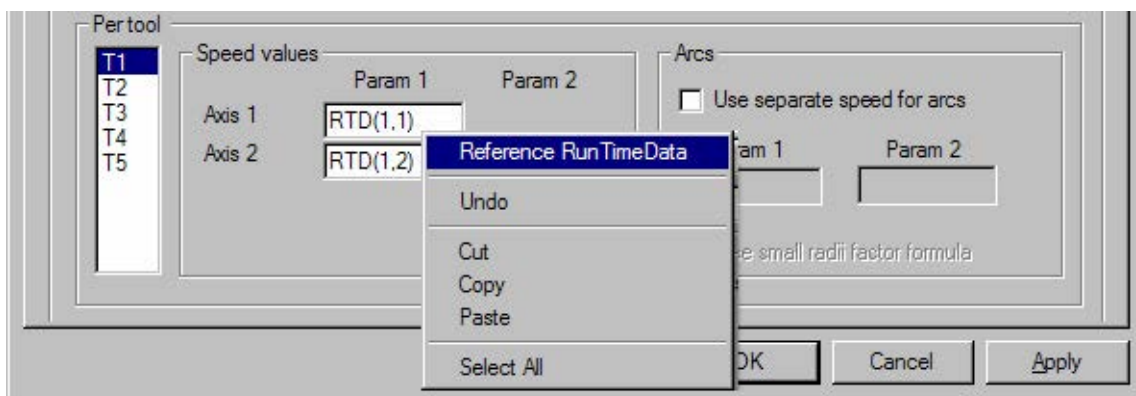
The example above shows a typical setup. Radii of 200 and above are cut at 100% of the normal speed, while smaller radii are cut at a proportionally lesser speed. Radii below the lowest defined radii are all cut at 5% of the maximum speed. A speed of 0 causes the machine to stop, which is why the graph does not begin at 0.

Use of Run Time Data

The things that change most often with an output to a CNC laser die cutter are the speed parameters. Even when variables are used, different materials usually require that the same basic NC data be encoded with a different set of values or variables for the different speed settings.

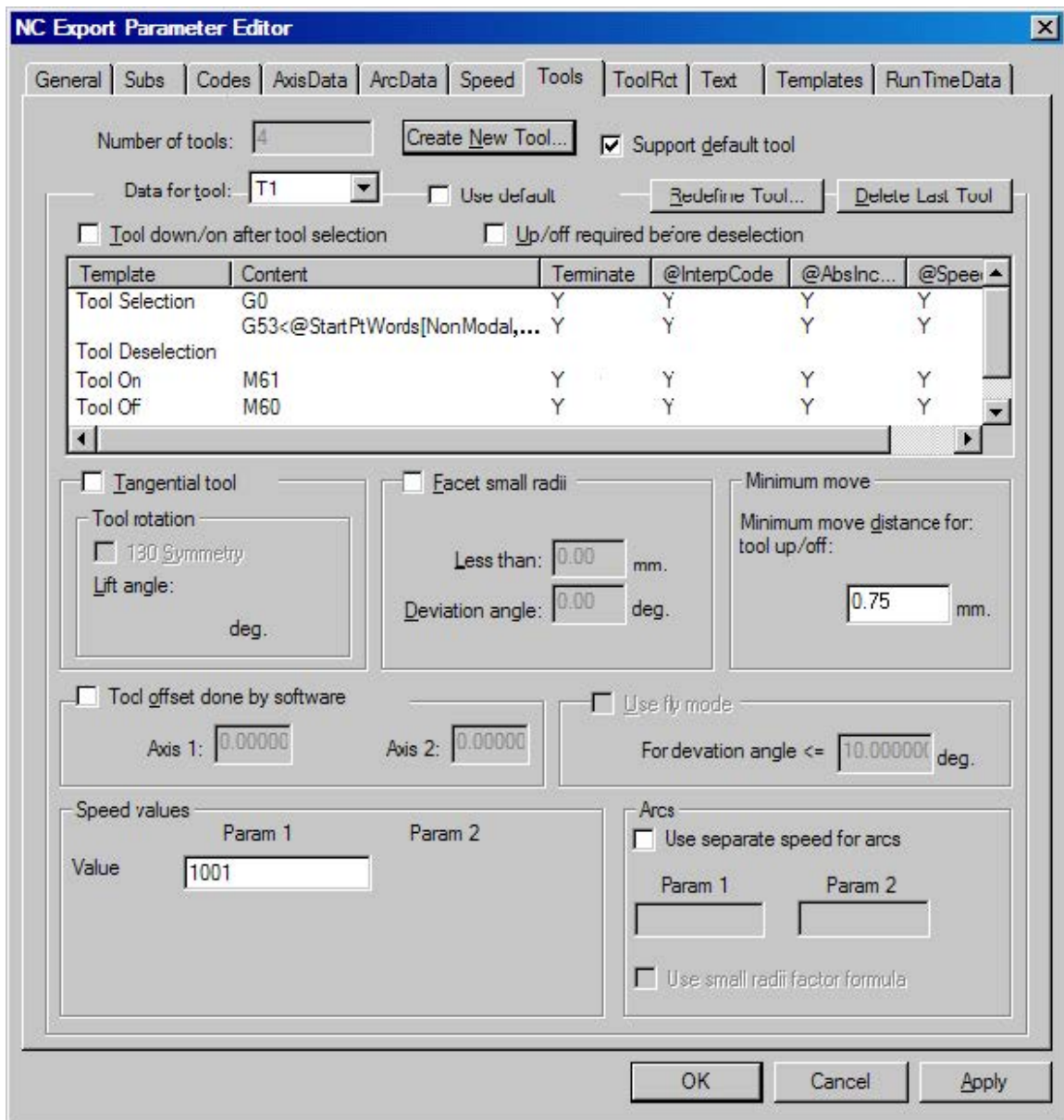
Almost any field in the tuning configuration can use run time data, but the speed parameters are the most likely to use RTD.

To use run time data, right-click the mouse button in a field and click **Reference RunTimeData** on the context menu (or type the run time data reference directly).



Tools Tab

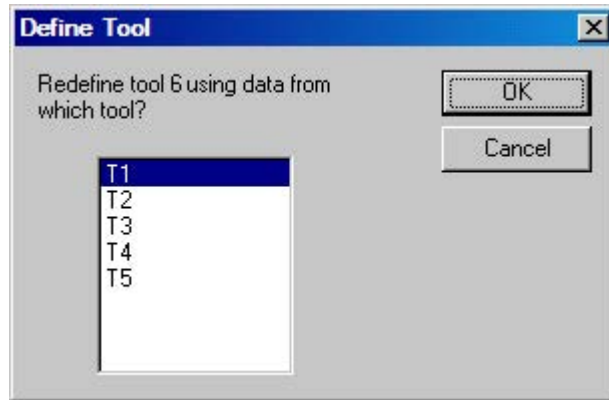
Each tool can be configured independently from the others. Select the tool to configure in the Data for tool drop-down list box.



Create New Tool / Redefine Tool

Initially this sets up data for tool 1. Most tool definitions are similar. Subsequently clicking **Create New Tool** opens the Define Tool dialog box, which allows the new tool to be cloned from a previously defined tool.

Clicking **Redefine Tool** redefines the current tool using data from another tool.



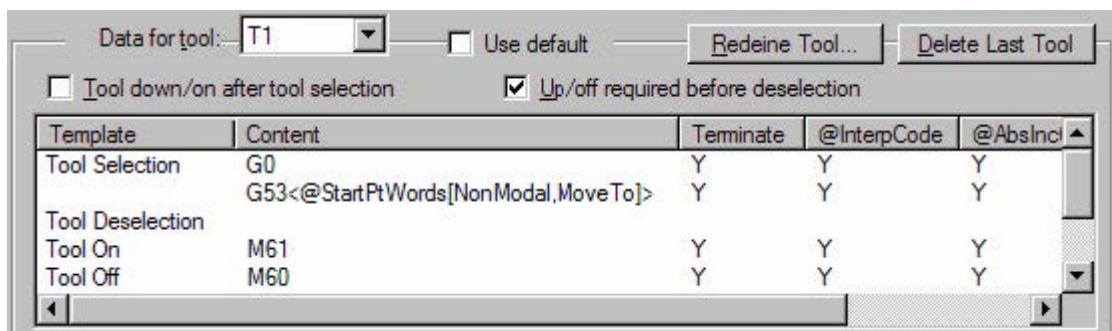
Tool Data

This tab contains all the tool-dependent data (with one exception; tool dependent interpolation codes are specified on the Codes tab only). Speed details are found on the Speed tab, Fly Mode on the General tab. The rest are discussed in this section.

Tool Control Functions

There are four tool templates provided for controlling a tool:

- Tool Selection
- Tool Deselection
- Tool On
- Tool Off



In the example shown above, the tool selection code shows a common way of applying a tool head offset correction. G53 selects a set of offsets applicable (in this case) to tool 1, and encodes a movement to the start of the next line. The move and the offsets are applied at the same time. Both axes are encoded to ensure the tool offsets for both axes are applied, even though there may be no change in position (in one or both axes) between the end of the last line and the start of the next. The **MoveTo** modifier ensures the current position is correctly updated.

It is often unnecessary to deselect a tool. Tools that are stored in a carousel, or need a router motor turned off and so forth may need to be explicitly deselected.

In the example shown above, M61 turns on tool 1 and M60 turns it off. (M60 probably turns any tool off. It would be used in all Tool Off templates).

Tool Down/On After Selection

Check this checkbox if the tool selection code also places the tool in the On state. This will avoid a duplicate ON code sequence. (The Tool On code is used the next time the selected tool needs to be turned on.)

Up/Off Required Before De-Selection

When this checkbox is checked, the Tool Off template is encoded followed by the Tool Deselect template. In general, this checkbox should be selected as the tool is always turned off before changing tools. However, it is sometimes possible to change the tool without turning the previous tool off.

Consider a laser cutter which changes the width of cut (the “tool”) by altering the power, speed of cut, and nozzle height (defined in the Tool Select template). All this may be performed quickly enough so that the shutter can be left open while it happens. (The open/ close operation of the shutter or switch is defined in the Tool On/Off templates.)

@Directives Suitable for Tool Templates

@ToolID can be used with a default tool setup in the Tool Select or Tool On template.

@LineType is more likely used in a Line or Arc template for describing data like DDES3 or CFF2. However, it might be used in a tool template.

@StartPtWords. If tool offset values are defined in the controller, it is almost certain that the Tool Select template uses **@StartPtWords** to invoke the tool offset as part of the move before the tool is actually turned on.

@EndPtWords. There may be circumstances where a line of a particular type marks the need to perform a particular operation. This operation may take place at the start or end of this special line. For example, one might need a hole to be drilled at the end of a line of a particular type.

@SpeedWords may be used in the Tool Select template if the unsophisticated approach to tool speed is taken. However, it is not much more effort to set the speed correctly.

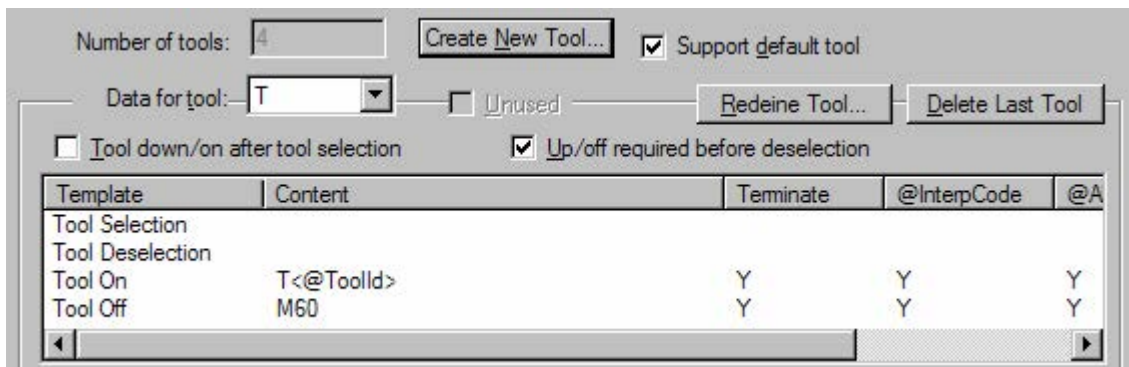
@AbsIncCode, **@InterpCode**, and **@SpeedWords** may/should be used as discussed above, if the corresponding modal flags are cleared.

@AtAngle. Using this in a Tool Select template where controller-stored tool offsets are used causes the tool to rotate as it is moved into position.

Any @directive may be used in a template; these are just the most likely.

Support Default Tool

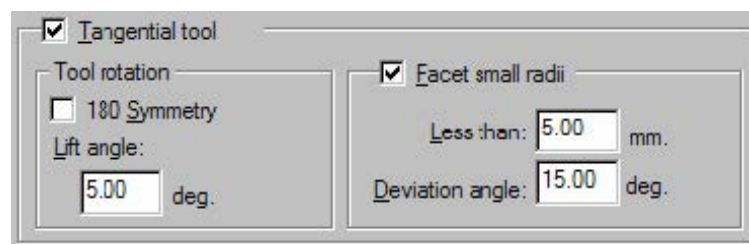
If most tool data is exactly the same other than the tool number, configuring a default tool makes setting up tools easier.



The directive **@ToolID**, when used in a template, generates the number of the appropriate tool. If a tool is marked as unused, it uses the default tool definition.

Any tool may be explicitly defined, in which case its definition is used in preference to the default tool definition.

Tangential Tool



Tool Rotation

The tool is lifted only when the discontinuity between two connected lines or arcs exceeds the **Lift angle**. If the lift angle is 0, the tool lifts between every line unless they are absolutely collinear at the junction. If **Lift angle** is set to a value greater than 180°, the tool is never lifted.

This indicates that the tool has **180° symmetry** and may be used equally well in either direction. A crease wheel is such a tool. Some knives may also be symmetric. The tool will never turn more than 90° between lines, thereby improving throughput.

Setting Up a Tangential Tool

Tangential tools use the @directives **@AtAngle** and **@FinAngle**. The formatting for these @directives is controlled by the settings on the ToolRot tab.

The **Tangential tool** checkbox indicates that this is a tangential tool. Only while a tangential tool is selected do **@AtAngle** and **@FinAngle** generate data. This is because a controller might error if tangential data is encoded for tools which do not have tangential control.

Template Name	Template Data
Move	<@InterpCode><@StartPtWords><@AtAngle>
Line	<@AtAngle><@InterpCode> <@EndPtWords>
Arc No Finish Angle	<@AtAngle><@InterpCode><@ArcDataWords><@EndPtWords>
Arc With Finish Angle	<@AtAngle><@InterpCode><@ArcDataWords> <@EndPtWords><@FinAngle>
Tool Change With Tool Offset	G0 G53<@StartPtWords [NonModal, MoveTo]><@AtAngle>

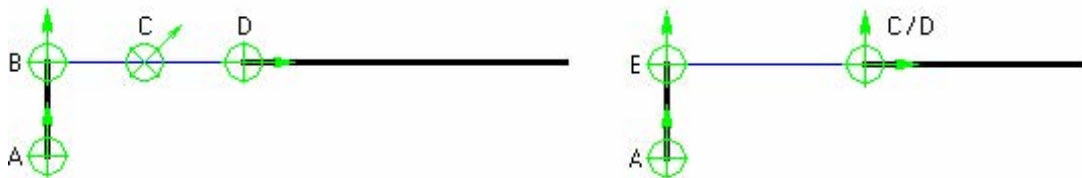
Arc Block With and Without Tangential Data

There are two common methods by which tangential control is performed on arcs:

- The tool is rotated by the controller to remain tangential with the direction of travel without any angle values set on the block.
- An angle is encoded on the same block as the arc data. This defines the finish angle of the arc (or in incremental mode, the angle through which to turn while forming the arc).

Setting the Angle During a Move

It is not necessary that **@AtAngle** be included in the Move or Tool Change templates. However, if it is, the throughput is better.



No @AtAngle in Move Template	@AtAngle used in Move Template
N10 M61 (A)	N10 M61 (A)
N20 G1 Y10	N20 G1 Y10
N30 M60 (B)	N30 M60 (B)
N40 G0 X30 C-90 (C)	N40 G0 X30
N50 M61 (D)	N50 C-90 (C)

No @AtAngle in Move Template	@AtAngle used in Move Template
N60 G1 X50	N55 M61 (D) N60 G1 X50

The above diagram shows the tool direction while cutting two lines at different angles separated by a move. If **@AtAngle** is present in the move template, the necessary tool rotation takes place while moving between the cut lines (C) rather than being executed once the tool is in position at the start of the second line.

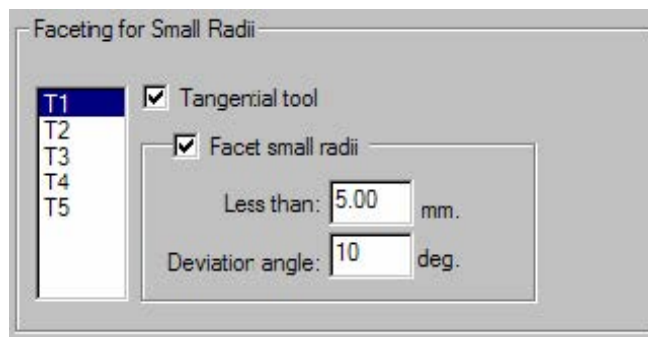
Similarly, when **@AtAngle** is used in Tool Select templates, the appropriate tool rotation takes place while the tool is being selected and moved into position.

Machines with Tangential Control Controllers

There are many controllers that provide all the necessary tangential control for the tools based purely on the NC movement data. Since no tangential control NC data is required, there is no need to mark any tool as tangential, nor use **@AtAngle** or **@FinAngle** in any template.

However, to facet small radii, it is necessary to select the **Tangential tool** checkbox. (Set the lift angle to any value above 180 to prevent any tool up commands from being encoded. Appropriate tool up/down operations are still performed by the controller.)

Facet Small Radii (for Tangential Tools only)



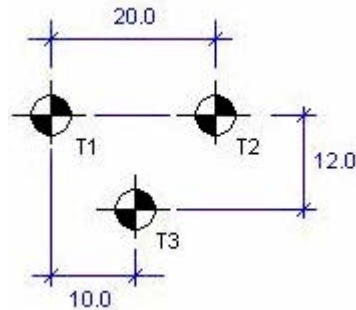
Small radii are not easily cut with a tangential knife, particularly when the material is thick. It is often better to cut such arcs using a series of small straight lines. The geometry of the knife has a considerable influence on the deviation angle that should be used. The radius at which faceting occurs and the facet deviation angle may be defined for each tool. Radii **Less than** the set value are cut with a series of straight lines. A sufficient number of lines are used to ensure the **Deviation angle** is less than or equal to the value set in the field.

The tool automatically lifts between each facet, regardless of the lift angle defined for the tool.

Tool Offset Done by Software

There are two common configurations for tool offsets. The first is where different “tools” are all positioned in the same place. An example of this is a laser die cutter, in which all the tools are

located at the nozzle. The different tools are selected by changing the power and/ or the height of the nozzle from the board. Another might be a counter cutting machine with a single spindle where the different diameter tools are selected from a carousel.



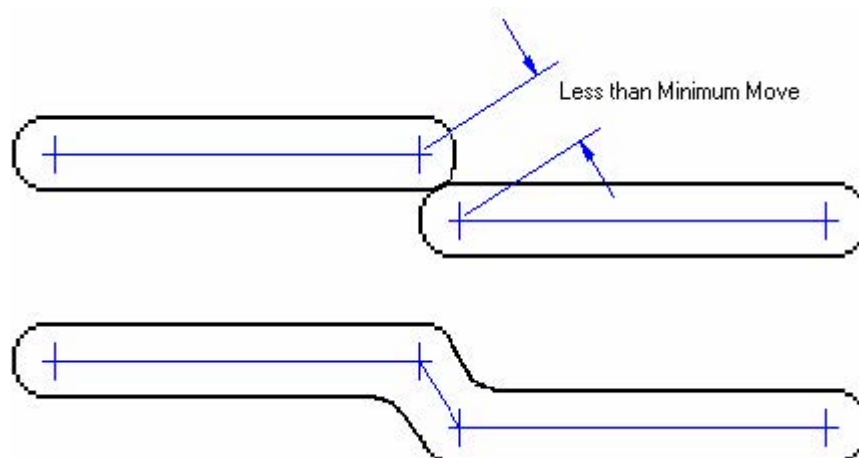
The second configuration is where all the tools are in a tool head which moves all the tools together. Sample makers, plotters, and some counter cutters are most often configured this way. The tools are separated by some fixed distance. On a tool change, the new tool moves to the active position. There are two ways in which this is done:

- As a tool is selected, the controller is instructed to make (or implicitly makes) adjustments to place the tool in the correct position based on offset values stored in the controller, *or*
- The GNC driver tuning includes tool offset parameters. These are applied to the movement data to ensure the tool is correctly positioned.

When the software tool offset feature is used, the offsets between tools are fixed, absolute values. Mirroring and/or rotation of subprograms may **not** be used in combination with the software tool offset feature. (The subroutine code could not be shared between mirrored and non-mirrored calls because the tool offset values may not be mirrored or rotated.)

There are machines that use both sorts of tool setup. A laser die cutter may do all the laser cutting through one nozzle, but also have a pen with which to plot and/or a drill for drilling bolt holes. These might be mounted on a tool head separate from the laser beam nozzle.

Short Move Control - Minimum Move



When the start of a line is only a small distance from the end of the previous line (for instance, between 0.5 and 1.0 times the tool diameter) there is little point in lifting the tool. Set the value to the minimum acceptable distance that should generate a cut instead of a move. A value of 0 implies a Tool Off/On sequence is generated for moves as small as 1 pixel.

A laser may dwell for a few milliseconds at the start of a cut. This may result in a bigger hole than simply cutting through the small move, particularly if the lines are collinear.

If the tool is tangential, small moves are ignored only if they are collinear. (It is slower to rotate the tool for the small make up line than it is to do the move without the rotation.)

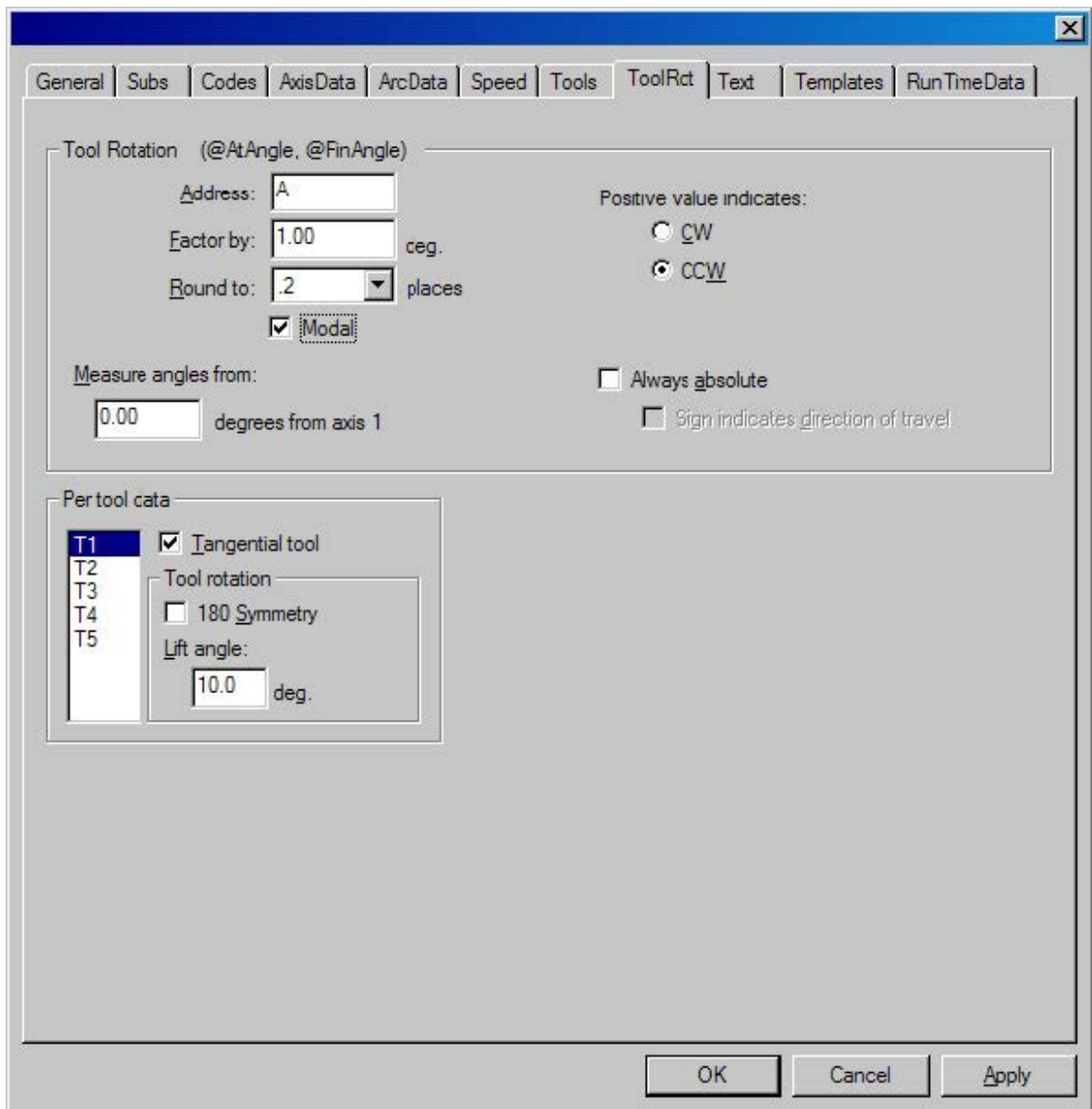
Use Fly Mode and Speed Values

See the General tab section and the Speed tab section for a full description of these features.

These settings are duplicated on the Tools tab for your convenience. Note that the proper tool must be selected in the Data for tool drop-down list box for the correct information to appear.

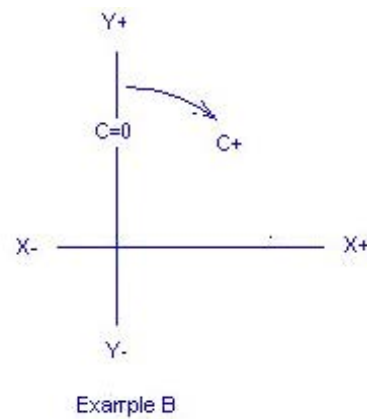
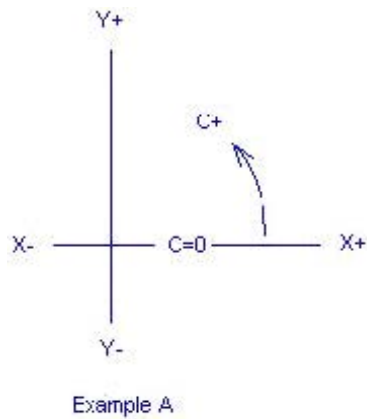
Tool Rot[ation] Tab

This tab describes the formatting for the @directives **@AtAngle** and **@FinAngle**. The general formatting concepts are the same as for the axis data.



When **Modal** is checked, rotational data is output only when the angle changes. It is assumed that the rotation is applied to all the tangential tools.

It is necessary to define from where angles are measured, and what is considered a positive angle. By default the 0 angle is considered to be the positive axis 1 direction.

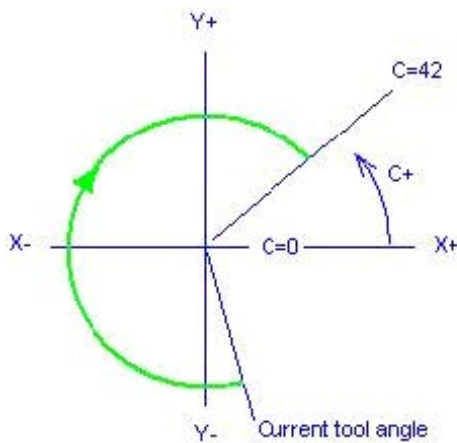


For example A, Measure angles from should be set to **0.0** and Positive value indicates should be set to **CCW**.

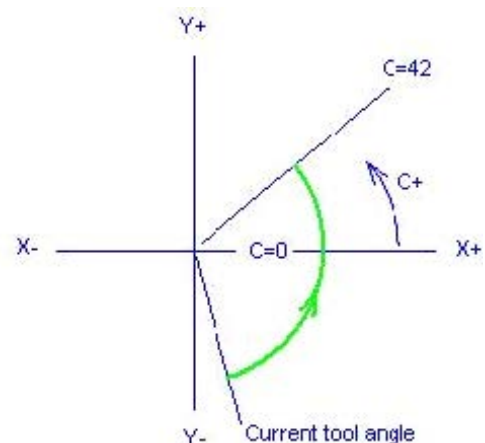
For example B, Measure angles from should be set to **90.0** and Positive value indicates should be set to **CW**.

Always Absolute / Sign Indicates Direction of Travel

On some machines, the sign of the angle indicates the direction of rotation. If so, the angle value is always encoded in absolute mode.



Movement when the block C-42 is executed



Movement when the block C+42 is executed

Block Executed	Action on Machine
C+42	Move clockwise to +42 degrees in the positive direction.

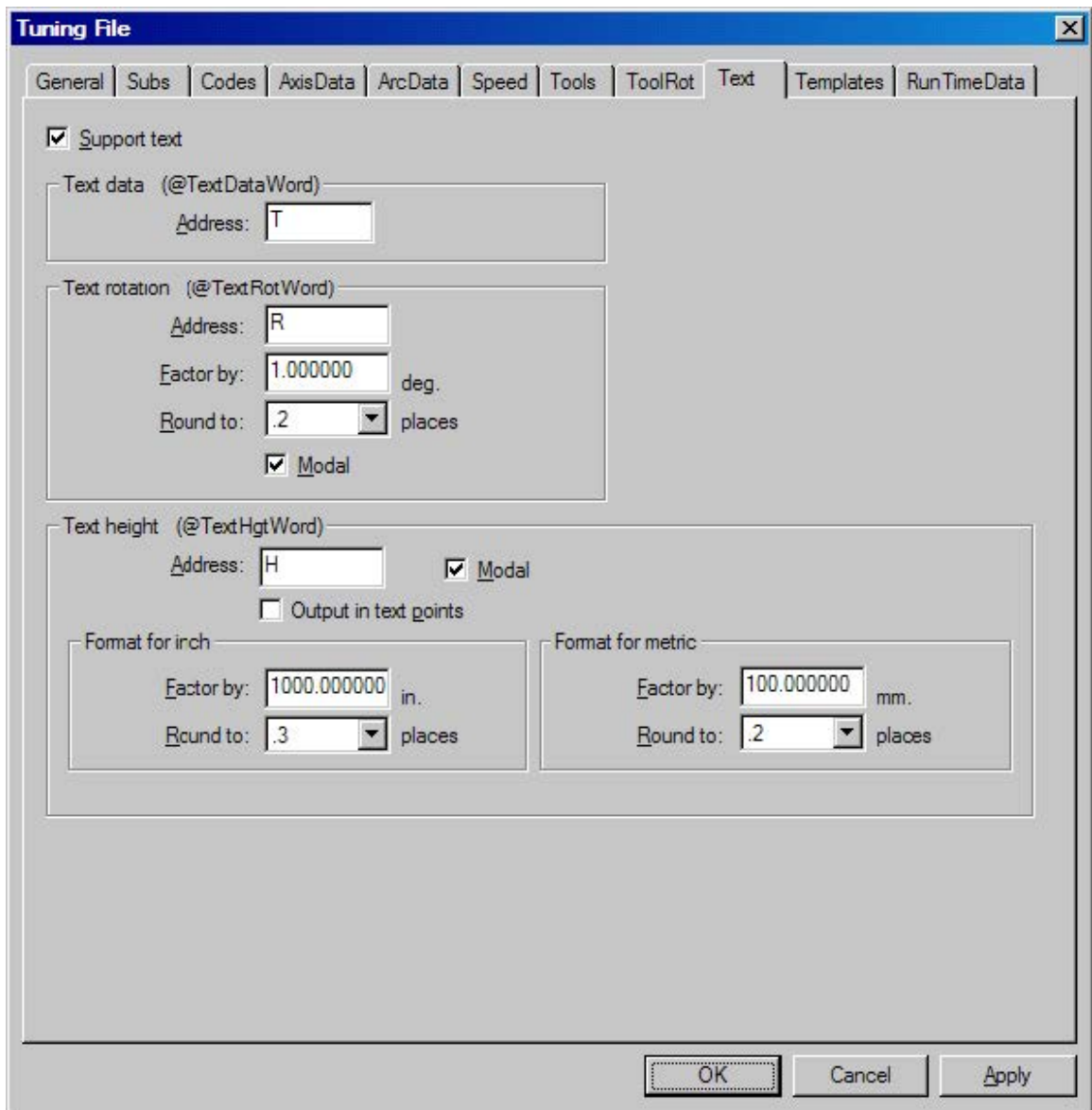
Block Executed	Action on Machine
C-42	Move counterclockwise to +42 degrees in the negative direction.

Per Tool Data

The Per tool data group is replicated from the Tools tab for your convenience. Rotation data is applied to tangential tools only.

Text Tab

This tab describes the formatting for **@TextDataWord**, **@TextRotWord**, and **@TextHgtWord**. The general concepts are the same as for the axis data.



A Text template appears on the Templates tab only if **Support text** is checked. The example shown above might produce text code that looked like this:

Example Template	Example Generated Code
<pre><@TextHgtWord><@TextRotWord> <@TextDataWord></pre>	<pre>H0.250 R11.25 TQuarter inch high Text at 11.25</pre>

HPGL text is normally terminated by CTRL-C. This should be entered as a text literal as part of the text template.

Example Template	Example Generated Code
<pre><@TextDataWord>chr(3)</pre>	<pre>LBQuarter inch high Text at 11.25°ctrl-C</pre>

HPGL has its own special way of describing text height and angle. See the Worked Examples document for a full description.

Check **Output in text points** if the height is in points regardless of the units used. The Format groups are consequently disabled.

Templates Tab

List of Templates

The following table shows a list of all the possible templates which together control the encoding of the NC data. Some templates are present only if the feature is enabled on one of the other tabs. For example, the Fly Mode template appears only if the feature is enabled on the Codes tab.

General Format Templates	Sample Functions
File Start	Characters to indicate the start of the NC data such as % or CTRL-A A subprogram to branch to the main program when using subprogram calls to block numbers.
File End	Characters to indicate the end of the file such as CTRL-D
Main Initialization	Program number Header messages indicating the size of the design etc. State initialization
Main Termination	Any code required to tidy the state of the machine The main program end code (such as M02)
Sub Initialization	Subprogram number State initialization
Sub Termination	The subprogram end code (such as M30)
Move	Movement code
Line	Line code
Arc	Arc code
Text	Text code
Move To Include Pt	The move to get to the place where the subprogram is called. This is necessary so that this special location can be recorded properly.
Sub Call	Setting Mirrors Rotations

General Format Templates	Sample Functions
Sub Call Reset	Transformations Call to the subprogram Re-setting Mirrors Rotations Transformations Initialization code needed to continue the main program (such as @AbsIncCode and so forth.)
Fly Mode	Enable and disable fly mode

The following table lists tool control templates and possible applications.

Tool Control Templates	Sample functions
Tool Selection	Tool select code to make the tool change Open security shutter covers Starting router spindles and spinning up to speed Turning on coolant/extractor fans etc.
Tool Deselection	Close security shutter covers Stop router spindle Turning off coolant/extractor fans etc.
Tool On	Turning tool on Any necessary dwells
Tool Off	Turning tool off Any necessary dwells

List of @Directives

@Directives control what data is output, how it is output, and where it is output.

The following table shows a list of all @directives. For each one, the templates in which it is most likely used are indicated. However, there is no restriction on using any @directive in any template. It is up to the user to decide whether it is appropriate. There are also some brief notes on each directive and some typical code that it might generate in a configuration of a regular CNC type controller. Any modifiers or keywords must be typed manually into the template when defining it.

Line-Generating Data

Name	Typically Used In Template	Notes	Example Output
@InterpCode	Move, Line, Arc Tools	Sets current interpolation state Optional keywords: [MODAL/NONMODAL]	G0 / G1 / G2/ G3
@StartPtWords	Move, Line, Arc Tool Selection	Start point data Optional keywords: [ABS/INC, WORD/ DATA, MODAL / NONMODAL, A1/A2, MOVETO]	X123 Y543
@EndPtWords	Line, Arc (Move)	End point data Optional keywords: As @StartPtWords	X123.4 Y543.2
@CenPtWords	Arc	Center point data Optional keywords: [ABS/INC,WORD/ DATA,MODAL/ NONMODAL,A1/A2]	I123.4 J543.2
@ArcDataWords	Arc	Arc generating data (sometime same as @CenPtWords) Optional keywords: As @CenPtWords	I123.4 J543.2 R25.4
@Radius	Arc	Radius or arc. Optional keywords: [WORD/ DATA,MODAL/ NONMODAL]	R25.4
@SubTangent	Arc	Optional keywords: [WORD/ DATA,MODAL/ NONMODAL]	A12.34
@ArcSign	Arc	Arc templates for outputs such as CFF2. Outputs 1 or -1 depending on whether arc is CW or CCW and whether CW indicates a positive or negative subtended angle	1 / -1

Tangential Tool Control

Name	Typically Used in Template	Notes	Example Output
@SetToolDir	Tool Selection	Allows you to specify the actual tool direction so that tool direction (such as C90) is correct in incremental mode. Optional keywords: [SETTO(degrees)]	Generates no output
@AtAngle	Line, Arc, Tools	The start angle of the current Line or Arc Optional keywords: [ABS/INC, WORD/DATA, MODAL/NONMODAL]	C135.00
@FinAngle	Line, Arc, Tools	The finish angle of the current Arc (line) Optional keywords: Same as @AtAngle	C-135.00

Line Attributes / Tool IDs / Speeds

Name	Typically Used in Template	Notes	Example Output
@ToolId	Line, Arc, Tools	The ID number of the current tool	10
@LineType	Line, Arc (Tools)	Used for outputs such as CFF2 ArtiosCAD Line type (May be used in Tool Select templates)	100
@CutWidth	Line, Arc, Tools	The width of the current tool/line in current units. (Not pointage) Optional keywords: [NDEC(vv) FORMAT(vv)]	0.711
@BridgeWidth	Line, Arc	Used for outputs such as CFF2. Width of bridges on the line.	5.00
@BridgeNum	Line, Arc	Used for outputs such as CFF2	3

Name	Typically Used in Template	Notes	Example Output
@SpeedWords	Line, Arc (Tools)	Number of bridges on a line. Control of speed and other tool/directional related parameters such as power or nozzle height.	F100.00 F[#10*0.82+#11*0.18]
@FlyModeCode	Fly Mode	Control of acceleration ramps or In Position Logic	G8 / G9

Text

Name	Typically Used in Template	Notes	Example Output
@TextWord	Text	Text data	TSome Text
@TextRotWord	Text	Text Rotation Optional keywords: [COS/SIN, FACTOR(#), NDEC(#), WORD/DATA, MODAL/NONMODAL]	R30.0
@TextHgtWord	Text	Text Size Optional keywords: [FACTOR(#), NDEC(#), WORD/DATA, MODAL/NONMODAL]	H10.0

Program Name and Number Definition

Name	Typically Used in Template	Notes	Example Output
@MainProgNum	Main Initialization	Main program number.	100
@CurProgName	Main Initialization	Part of a heading message	A01234
@IncludeNum	Sub Call	Numeric part of subprogram reference	103
@IncludeName	Sub Call	The name of the file without the extension.	A01234
@CurProgNum	Sub Initialization	Subprogram number	103
@NumOfSubs	Main / Sub Initializatio	Number of Subprograms	5
@MarkAsSub	Sub Initialization	Marks a particular block as being a subprogram entry point.	Generates no output

Name	Typically Used in Template	Notes	Example Output
		Must be used in the Sub Initialization template when Use Block Numbers is enabled on Subs tab.	

Subroutine Transformation Control

Name	Typically Used in Template	Notes	Example Output
@IncludePtWords	Sub Include Pt	Included design reference point data Optional keywords: As @StartPtWords	X123.4 Y543.2
@IncludeMirCode	Sub Call	Optional keywords: [NONMODAL] Usually used in Origin Transformation for Sub Call If NONMODAL keyword is used then outputs mirror code even if no mirrors are used	G39 X1 Y1
@IncludeMirResetCode	Sub Reset	Mirror Reset code Optional keywords: [NONMODAL]	G38
@IncludeRotWord	Sub Call	Optional keywords: [NONMODAL] Origin Transformation for Sub Call If NONMODAL keyword is used then outputs rot code even if no rotation	G41 A90
@IncludeRotResetWord	Sub Reset	Rotation Reset code Optional keywords: [NONMODAL]	G40

Handy Heading Data

Name	Typically Used in Template	Notes	Example Output
@Date	Main Initialization	Part of a heading message	10/31/2017

Name	Typically Used in Template	Notes	Example Output
@Time	Main Initialization	Part of a heading message	10:30:15 PM
@CADFile	Main Initialization		
@OutputFile	Main Initialization	Part of a heading message	A01234.CNC
@MaterialFile	Main Initialization		B-flute-34
@DataSetName	Main Initialization	Part of a heading message	Maple 19mm
@TuneFile	Main Initialization	Name of CAM tuning file name (Used for testing)	
@TuneMenuPick	Main Initialization		

Program Initialization

Name	Typically Used in Template	Notes	Example Output
@AbsIncCode	Main Initialization Sub Initialization Tools	Sets Absolute or Incremental mode Optional Keywords: [MODAL/NONMODAL]	G90 / G91
@UnitsCode	Main Initialization Sub Initialization (Tools)	Sets NC data units Optional Keywords: [MODAL/NONMODAL]	G70 / G71
@GrainCode	Main Initialization	Identifies the Grain/ Flute direction	%FLUTE
@SideCode	Main Initialization	Identifies the sidedness of the output	\$INSIDE\$

Design Size and Rotary Effective Radius Data

Name	Typically Used in Template	Notes	Example Output
@BlankX	Main Initialization	Part of a heading message	123.45
@BlankY	Main Initialization	Part of a heading message	1234.56
@LowerLeft	Main Initialization Main Termination	The axis data for the lower left point of the data.	X-100 Y-100
@UpperRight	Main Initialization Main Termination	The axis data for the upper right point of the data.	X1234 Y543
@RotaryRadius	Main Initialization	Used to define the radius for a rotary axis	R123 such as G20 X0 Y0 R123

Name	Typically Used in Template	Notes	Example Output
		so that speeds can be correctly controlled.	

Block Number and IO Dwell Control

Name	Typically Used in Template	Notes	Example Output
@NBlockOff	Must not be on same line as @NBlockOn	Suspends block numbering.	Generates no output
@NBlockOn	Anywhere	Optional keywords: [SETTO(#),NORMAL/SPECIAL/INCBY(#)] Starts block numbering with optional settings.	Generates no output
@IODwell	Main and Sub Initialization and Termination.	Suspends I/O for 1000 ms or specified number when outputting to a port or spooler. Optional argument: [#Milliseconds]	Pauses output

Additional Notes on @Directives

@ArcSign

Outputs 1 or -1 depending on whether arc is CW or CCW and whether CW indicates a positive or negative subtended angle.

@SetToolDir

When issued, the current angle internally is set to the default 0 position. This should be used when the selection of a new tangential tool also initializes the tool angle. It should not be used if tool selection does not reset the angle (such as for ELCEDE IBH controlled sample makers).

Optional Keywords

The following are used with axis ordinate @directives **@StartPtWords**, **@EndPtWords**, **@CenPtWords**, **@IncludePtWords**, and **@ArcDataWords**, as indicated in the table above.

ABS or INC

This forces the mode of the data for this particular use of the @directive regardless of the current state of **@AbsInc**.

MODAL or NONMODAL

This overrides the modal setting for the @directive.

NONMODAL may also be used for the @directives **@IncludeMirResetCode** and **@IncludeRotCode**.

WORD or DATA

WORD causes the address and the data to be encoded while **DATA** causes just the ordinate value to be encoded.

A1 or A2

Outputs data for only axis 1 or axis 2.

MOVETO

This causes the current position to be updated with the data that has just been encoded. This is typically used when a controller supplied tool offset is required as part of a tool selection. The tool offset movement is incorporated with the move to the start of the next line. Such a template might be:

```
G0
G53<@StartPtWords [MoveTo, NonModal] >
```

Examples	Sample Result
@EndPtWords [ABS]	X1123Y456
@EndPtWords [INC, NONMODAL]	X123Y0
@EndPtWords [INC, MODAL]	X123
@EndPtWords [A1, WORD]	X123
@EndPtWords [A1, DATA]	123

The following are used with **@TextHgtWord** and **@TextRotWord**:

WORD/DATA, MODAL/NONMODAL

as described above.

FACTOR(#)

Multiplies the **@TextHgtWord** by the value in parentheses. This is used in HPGL where it is necessary to use a text height and width. The width is set to some factor of the height (such as .75).

Formatting is applied after the factor is applied.

NDEC(#)

Sets the number of decimal places.

COS/SIN (@TextRotWord only)

Outputs the cosine/sine of the rotation angle.

Examples	Sample Output
@TextHgtWord	H10.5
@TextHgtWord [FACTOR (2)]	H21.0
@TextHgtWord [NDEC (2)]	H10.50
@TextRotWord	R30.0
@TextRotWord [COS]	R0.866

The following are used only with **@NBlockOn**:

SETTO(nn)

This sets the block count number to the value set by `nn` (such as `SETTO (100)`).

NORMAL/SPECIAL/INCBY(nn)]

Sets the current block number to the next highest multiple of the `NORMAL`, `SPECIAL` value or the value set by `INCBY`, and sets the increment to this value.

The following is used only with **@IODwell**:

[#seconds]

The default dwell is 1 second. The optional argument provides alternative values.

The dwell is invoked only when writing data to a port.

General Comments on Templates

Inserting Control and Other Special Characters

Any ASCII character including control characters can be inserted into the string using the function `chr (d)`, where `d` is the decimal value of the ASCII character required. For example, `chr (9)` inserts a tab character in the string. The text `chr` must be in lower case.

Examples:

Some plotters using HPGL require data starting with the characters Escape, Period, Open parenthesis. The Start of File template therefore might look like:

```
chr (27) . (
```

The string `<@>` errors if entered like this as it looks like an invalid `@directive`.

If you need to encode the string, use `chr (60) chr (64) chr (62)` or `<chr (64)>`.

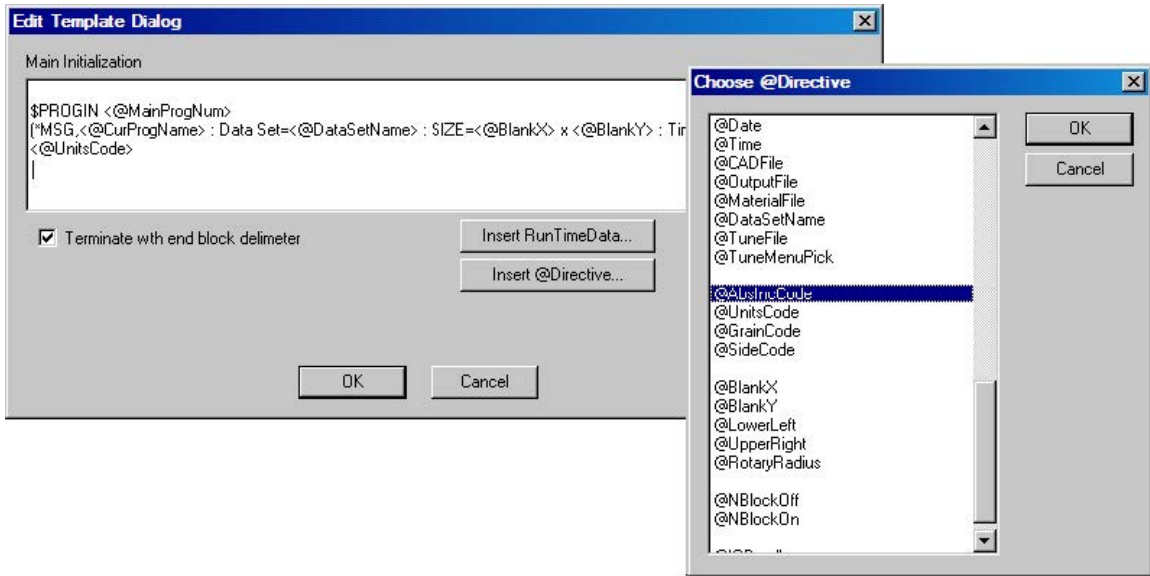
Comments in Templates

The character `!` (exclamation point) is used to denote that the remainder of a line is a comment. If this character is to be used in a template, it must be entered as `chr (33)`.

Empty Templates

It is quite permissible for a template to have no data. For example, if there is no need for anything to be placed at the head of the file, the File Start template may be left empty.

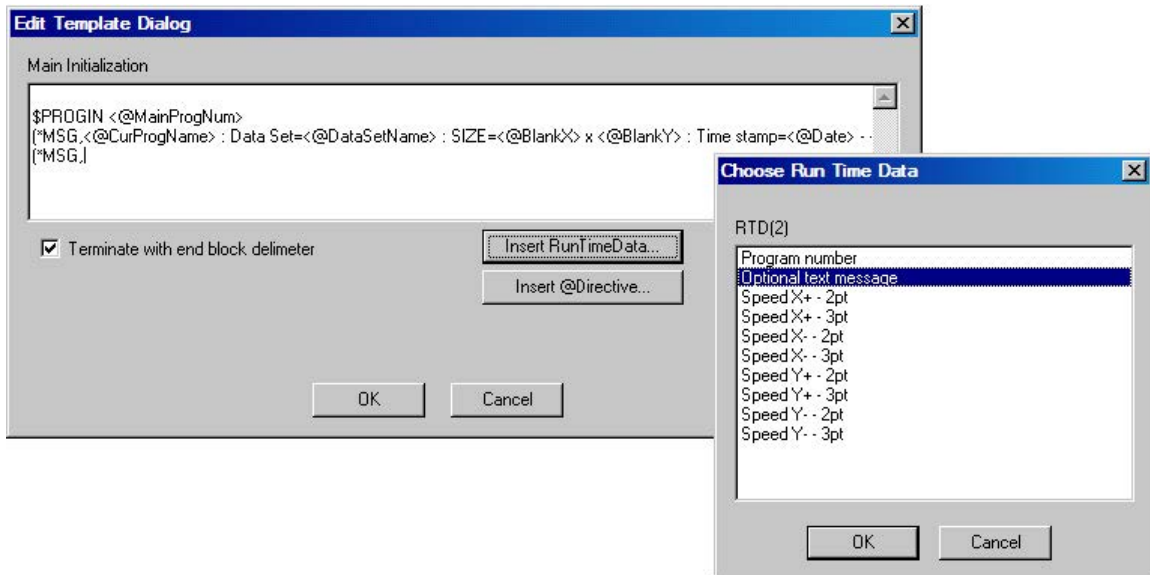
Insert @Directive



This opens a dialog box of all the permissible @directives. Selecting one inserts it in the template at the current cursor position.

Insert Run Time Data

This opens the Choose Run Time Data dialog box from which any RTD element may be selected. Most likely, a message type of RTD is used here as shown.



Terminate with End Block Number

Normally this checkbox should be checked. This adds the block end characters (defined on the General tab) to the end of the template. If this checkbox is not checked, no block end characters are appended.

Templates can contain many lines of data. When building a template, press `ENTER` to start a new line. As the template is encoded to form NC data, each line is terminated with the appropriate block end characters.

Run Time Data

Run time data is data that is entered or altered when the Output is performed. It is therefore useful for frequently-changing values like cutting speed, when cutting speeds must be set in the NC data. (If the controller allows direct control of speeds while the die is being cut, it is not necessary to use RTD for these values.)

A number of datasets may be defined. If the same machine is used for cutting 19 mm and 12 mm die board as well 10 mm and 6 mm clear plastic) four such datasets, each with its own set of speeds, make operation easier.

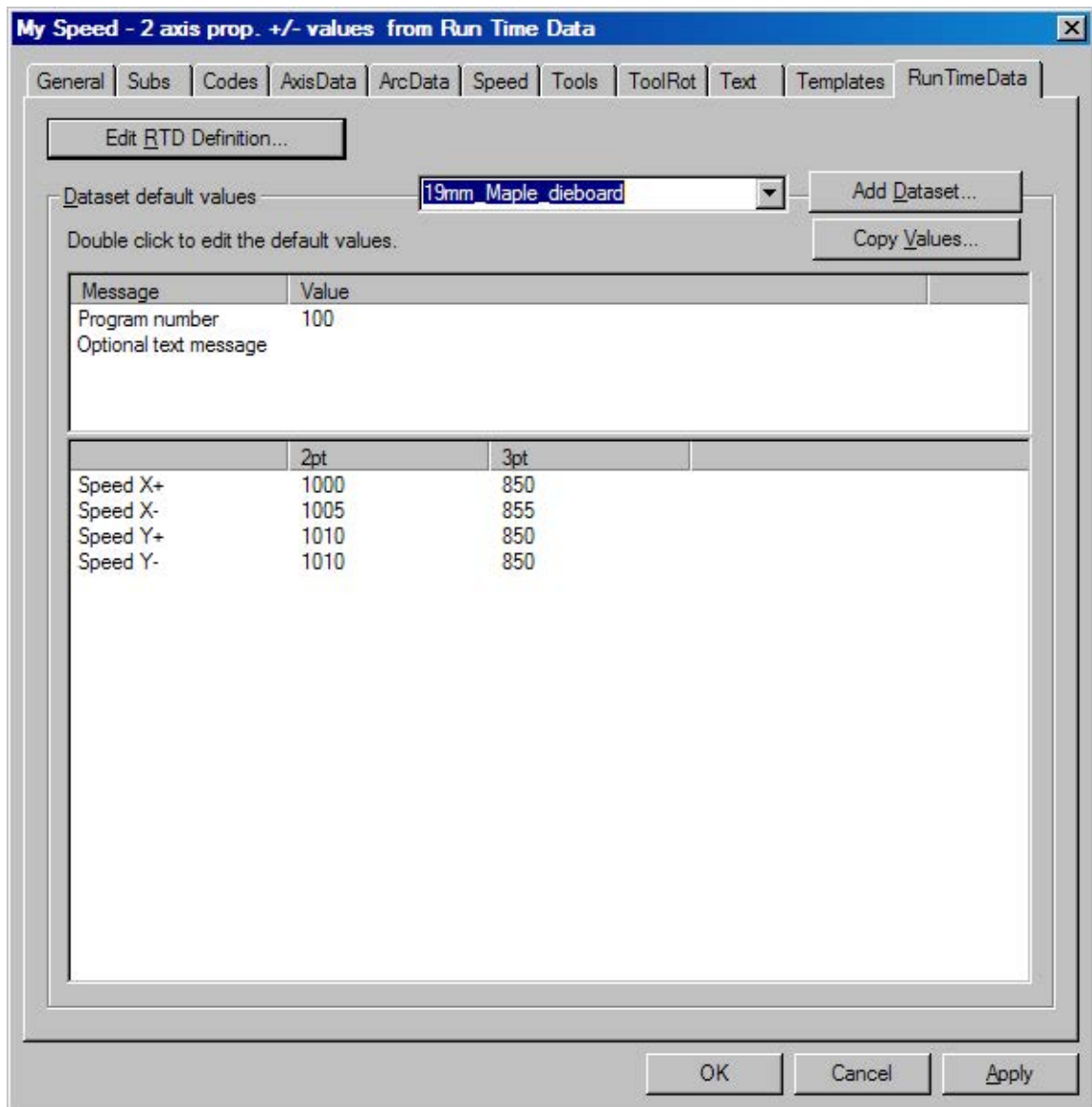
It is possible to configure these default values using a Library function. This allows values to be initialized based on some function of the design criteria. For example, the depth of a partial cut could automatically be set to 60% of the caliper of the board being used, or the speed decreased for a thicker board.

Dataset Defaults Setup

There are two types of datasets:

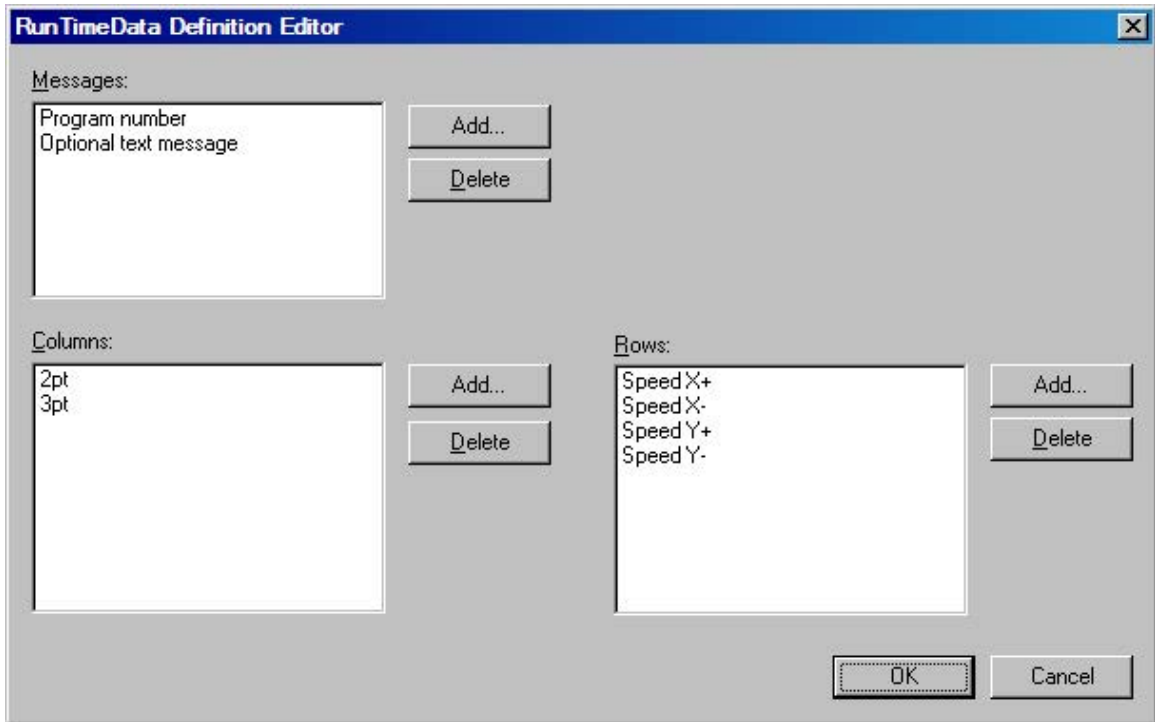
- Any number of single value items (called messages).
- One grid of data elements comprising a number of rows and columns.

Typically, messages provide a program number or some message information to display on the controller, while the grid of data items might be used to enter speed information.



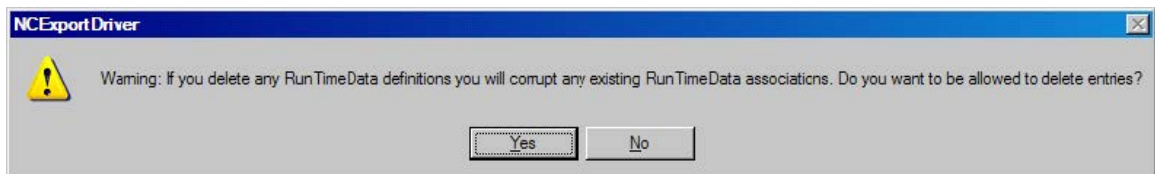
Edit RTD (Run Time Data) Definition

Clicking **Edit RTD Definition** allows the definition of headings for the messages, and the definition of row/column names of the grid elements. The definition applies to all datasets.



Note:

Deleting data from these tables at a later time may lead to problems. The RTD items may have been used in templates that have already been built!



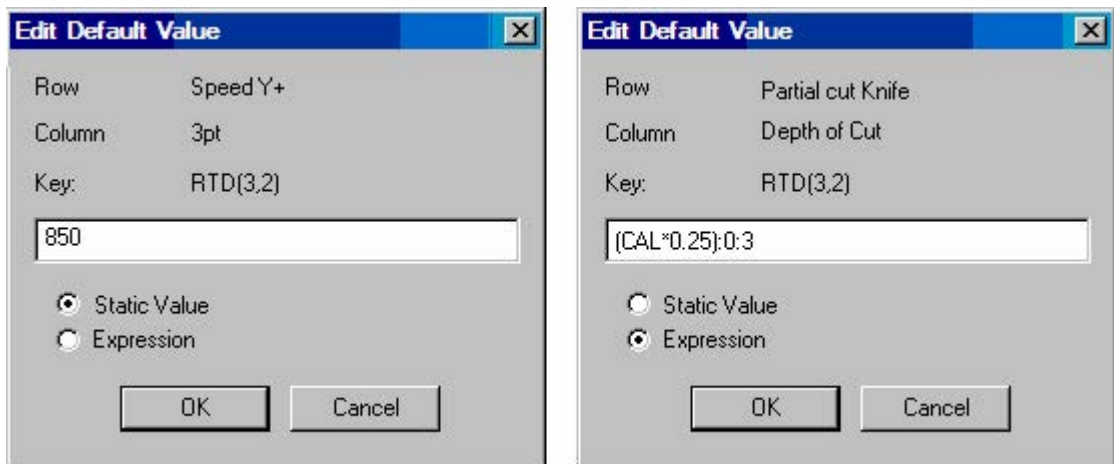
It is the user’s responsibility to ensure that the RTD setup and the RTD references used in the templates are correctly used. Changing the general arrangement of the RTD will cause the default values used at run time to be invalid. They will resort to the initial default values again.

The RTD headings, once set, may not be edited.

RTD Definitions must be defined before adding datasets.

Add Data Set creates additional datasets, while **Copy Values** copies the values of a previously defined dataset to the current dataset.

Double-clicking an individual element lets you enter a default value.



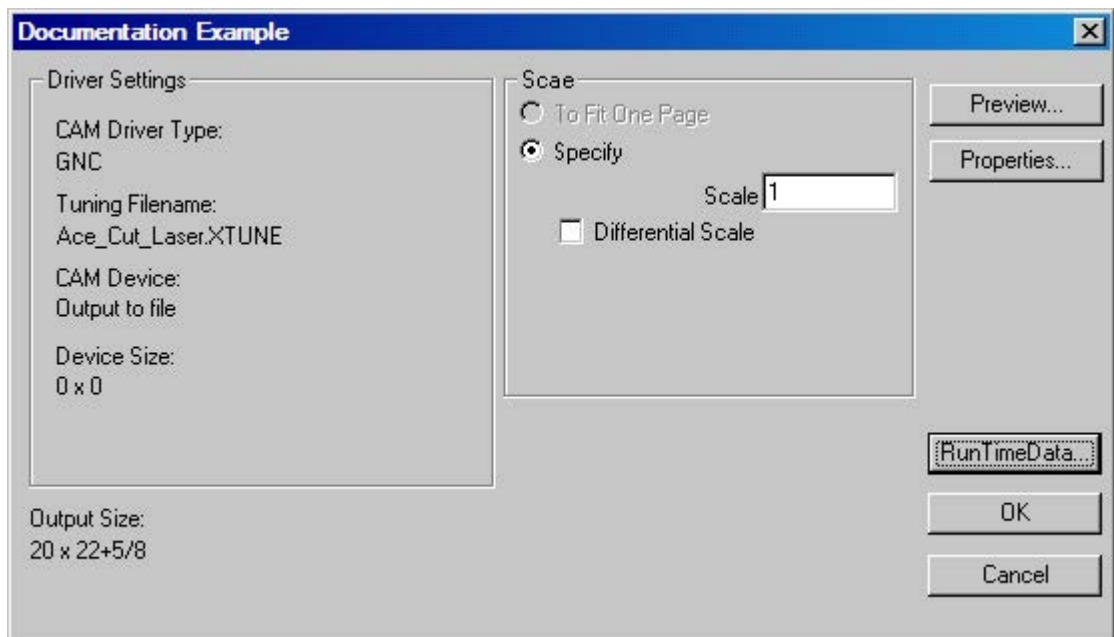
Default values may be simple data or a text expression. Expressions can use Library and Tlibrary functions and any of the normal text expressions. Numeric expressions are formatted using the normal formatting unless explicitly specified. Suppose the value of CAL is 0.40 in the above example. The result is 0.100.

Example Template	Resulting Code
G1 Z<RTD(3,2)>	G1 Z0.100

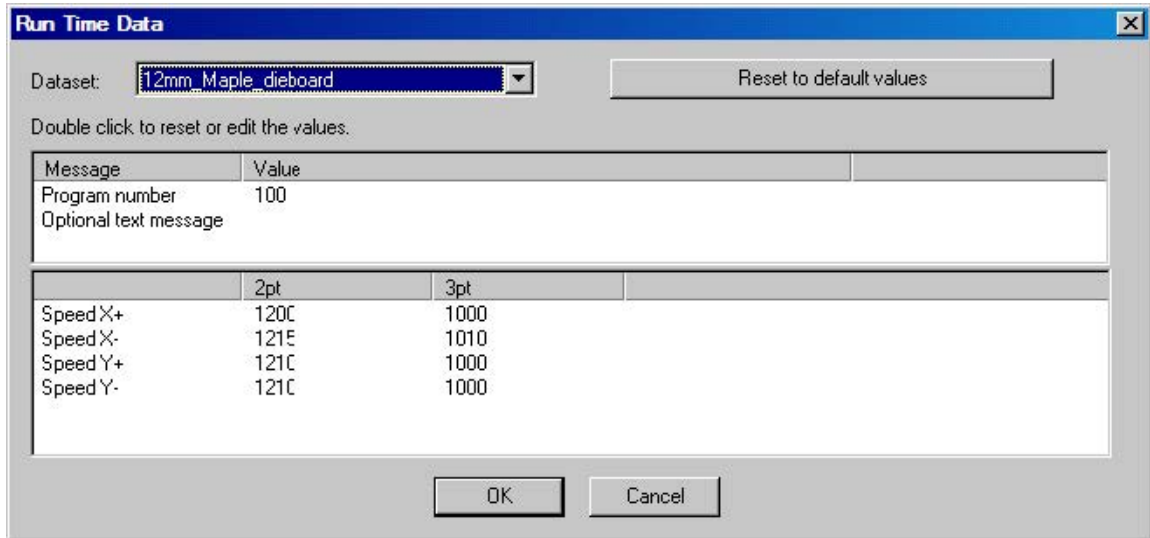
Default values set to **Expression** are evaluated each time the Output is used. The value is not cached.

During the Output Process

RunTimeData is available in the normal Output dialog box when using the GNC driver.

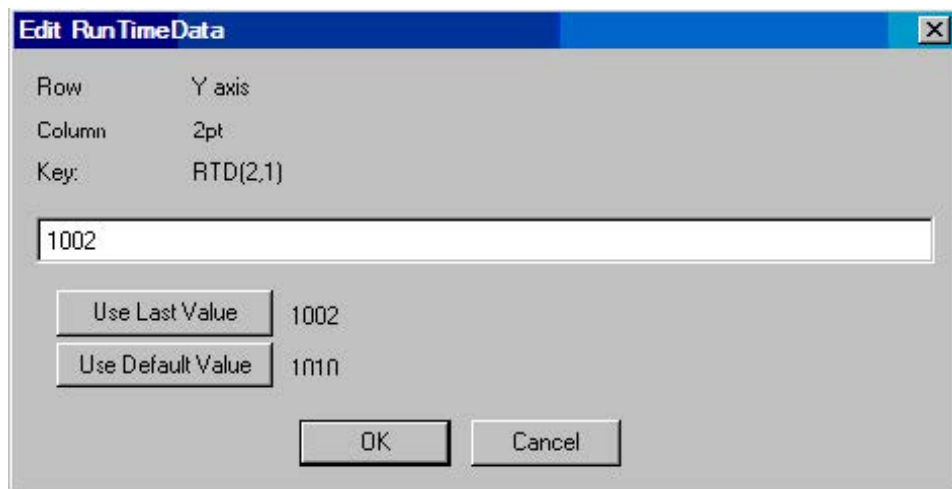


In the dialog box shown below, double-click a message or a value to change it.



Caching of Run Time Data Entries

Double-clicking a value opens the following dialog box.



The first time the Output is used, the RTD is seeded with the values set up in Defaults. The values can be changed in the above dialog box. Once the Output is complete, the current RTD values (other than RTD elements set by an expression) are cached. When the Output is next used, the cached values are seeded rather than the default values. The values defined by an expression are recomputed each time the Output is used.

For example, normally the speeds for cutting a particular type of wood vary a little from day to day. This feature provides a convenient way to accommodate "speeds of the day." These are used until changed or reset.

If changes to the configuration of the RTD or the Output are made, the default values are used.

The cached RTD values are stored in the Windows Registry (`HKEY_USERS\user system ID\Software\Artios\NCDriver\name of Output`) for each Output and dataset used.