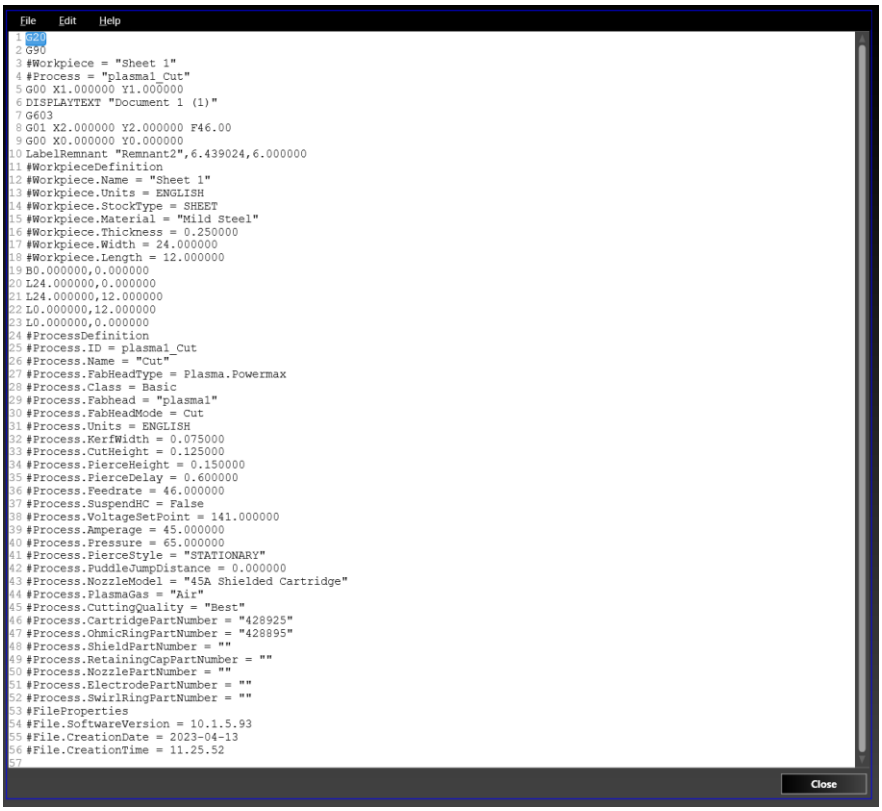
FLASHCUT CNC

CONTROL MADE SIMPLE ®

FlashCut CNC 10 Programming Reference



Notices

Revised April 13, 2023
Version 10.1.5.093

About this document

Because new features may be added to the software in the future, this manual applies only to the version for which it was released.

There may be differences between the information presented here and the features found in either older or newer versions of the software.

Screen captures used in this document are edited for clarity, and may differ in minor ways from the actual FlashCut application.

Disclaimer

FlashCut CNC and its affiliates are not responsible for the safe installation and use of this product. You and only you are responsible for the safety of yourself and others during the operation of your CNC machine tool. FlashCut CNC supplies this product, but has no control over how it is installed or used. Always be careful!

FlashCut CNC is not responsible for damage to any equipment or workpiece resulting from use of this product.

If you do not understand and agree with all of the above, please do not use this product.

Safety and Usage Guidelines

Automated machining is potentially dangerous. Please take the time to completely read through this manual to understand operation of the software before running the system.

A working knowledge of the PC and the Windows operating system is required in order to install, use, and troubleshoot the software.

Since FlashCut is a real time control program, it must have full control of the operating system while running. It is very important that you do the following before running FlashCut:

- Disable all screen savers and power management programs.
- Make sure there are no background programs running, such as back-up software and calendar reminders.
- Make sure no other programs are open.

Safety is of the utmost importance. To use FlashCut CNC to control your automated machine tool in a safe and proper fashion, the following safety guidelines must be followed:

- Never let the machine tool run unattended.
- Require any person in the same room as a running machine tool to wear safety goggles, and to stay a safe distance from the machine.
- Allow only trained operators to run the machine tool. Any operator must have:
    - Knowledge of machine tool operation
    - Knowledge of personal computer operation
    - Knowledge of Microsoft Windows
    - Good common sense
- Place safety guards around the machine to prevent injury from flying objects. It is highly recommended that you build a safety shield around the entire tool envelope.
- Never place any part of your body within the tool envelope while the machine has power, since unexpected machine movement can occur at any time.
- Always keep the tool envelope tidy and free of any loose objects.
- Be alert for computer crashes at all times.

Technical Support

Expert technical support is provided for all of our products. Many resources are available to help you resolve your problems quickly. We recommend that you use these resources in the following order:

Website

http://www.flashcutcnc.com/

Our website has product specifications, documentation, support videos, and other information.

Dealer Support

If you purchased FlashCut CNC from a dealer or other machine tool manufacturer (OEM), please contact them first. They will have the best knowledge of your complete system.

Email

support@flashcutcnc.com

Email is the most organized way to communicate an issue to our support staff. In your e-mail, please state your problem completely. The email should include this information:

- FlashCut version
- Computer processor and speed
- Windows version
- Signal generator serial number.

In addition, please attach the following files:

- Setup and Tooling files (usually found in a folder named **c:\FlashCut Data**)
- G-code file with which you are having problems (when appropriate)
- CADCAM file with which you are having problems (when appropriate)

Alternatively, you can attach a single FlashCut support file. The support file is in ZIP format and contains all relevant files needed by technical support to resolve your issue. To generate the file:

- Select the Help button in the upper right corner of the main window
- Click the **Build Support File** button
- Name and save the file using the Windows dialog box.

Note The Support file will only include a G-Code file that has been saved and it **will not** include a CADCAM file. If you are having a CADCAM issue, please e-mail the CADCAM file separately.

Please see the **User's Guide** for more details.

Phone/fax Support

If email is unavailable to you, please call our telephone support number. We will normally respond to your call within 24 hours.

Phone: (847) 940-9305 (9:00 AM-5:00 PM, US Central Time, M-F)
Fax: (847) 940-9315

Contents

Introduction

FlashCut supports ANSI standard G-Code to control machine tool movement and peripheral devices. This reference describes how to create and open G-Code files, and lists the G-Codes supported.

Programming G-Code in FlashCut CNC

Drawings are created, edited, and saved as CADCAM files. G-Code files are generated from the CADCAM drawing before cutting the parts in FlashCut CNC.

A new G-Code file is automatically generated by FlashCut CAM whenever the drawing is sent to FlashCut CNC. You can review, edit, and/or save the G-Code generated by the program by using the editor.

FlashCut CNC can open one or more already existing G-Code file. FlashCut can also be configured to automatically insert specific G-Code commands when specific conditions are met. See the **User's Guide** for more details, especially the sections on configuring FlashCut and using the FlashCut CNC interface.

How to Create or Modify a G-Code File

There are three ways you can create or modify G-Code files:

- Create a G-Code file from a drawing
- Open an existing G-Code file
- Create a new file in the G-Code editor

Create a G-Code File from a Drawing

With a drawing open in **FlashCut CAM:**

1. Select **Create G-Code File** from the ribbon. The Save File dialog will open.
2. Select a location.
3. Name the file. It will be saved with the extension ***.nc**.

The file may also be reviewed and edited:

4. Select **Open Editor** from the ribbon. The file just saved will be loaded into the editor.

The features of the editor are covered in detail below: see **G-Code Editor**

Open an Existing G-Code File

In **FlashCut CNC**, you can load either a single G-Code file, or a collection of files from the same folder.

To open one file:

1. Click the **Load G-Code File** icon. The Open G-Code File window appears.
2. Navigate to the G-Code file you wish to open.
3. Double-click the file name, or click on the file name and choose Open.

To load multiple files:

1. Click the **Load all G-Code files in a folder** icon. The Browse window appears.
2. Navigate to the desired folder, select it, and choose OK.

The files will be loaded into the drawing workspace in order. Each file may also be reviewed and edited:

3. Select **Edit G-Code**. The G-Code file will be loaded into the editor.

Edit the current drawing in FlashCut CNC

New in version 7.04: Double-clicking the G-Code window in FlashCut CNC will open the G-Code editor. If there is a drawing in the workspace, the G-Code for that drawing will be loaded into the editor.

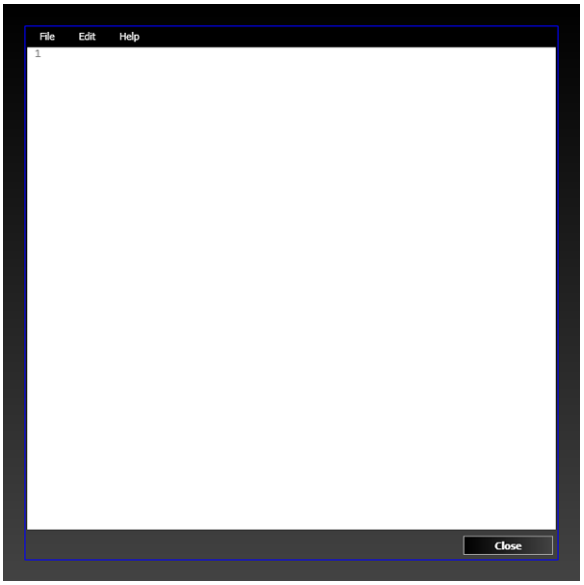
Create a new file in the G-Code Editor

In either FlashCut CAM or FlashCut CNC, selecting the editor icon with no G-Code file loaded will open a new blank editor window.

If you prefer, you can also use a text editor of your choice, such as Notepad or WordPad. If you use a different editor, make sure you save the file as **text only** and add the extension ***nc** to the file name.

The features of the G-Code editor are covered in detail below.

G-Code Editor



FlashCut provides a full-featured editor for creating or modifying G-code files. You can launch the editor from either **FlashCut CAM** or **FlashCut CNC**.

The same icon appears in both the **FlashCut CAM** ribbon and the **FlashCut CNC** G-Code window.

Clicking the icon opens the editor.

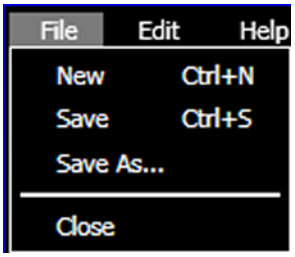
Note G-code files that use the older FlashCut-specific file extension ***.FGC** can also be opened.

Note New in version 7.04: double-clicking the G-Code window in **FlashCut CNC** will open the G-Code editor.

The menus are described below:

- **File Menu**
- **Edit** Menu
- **Help Menu**

File Menu



The **File** menu presents the following commands:

- **New**
- **Save**
- **Save As...**
- **Close**

New

Opens a new, empty file.

Open...

Opens an existing G-code file for editing. You may save it once changes are applied.

Save

Saves the G-code file.

Save As...

Saves the G-code file under a new name.

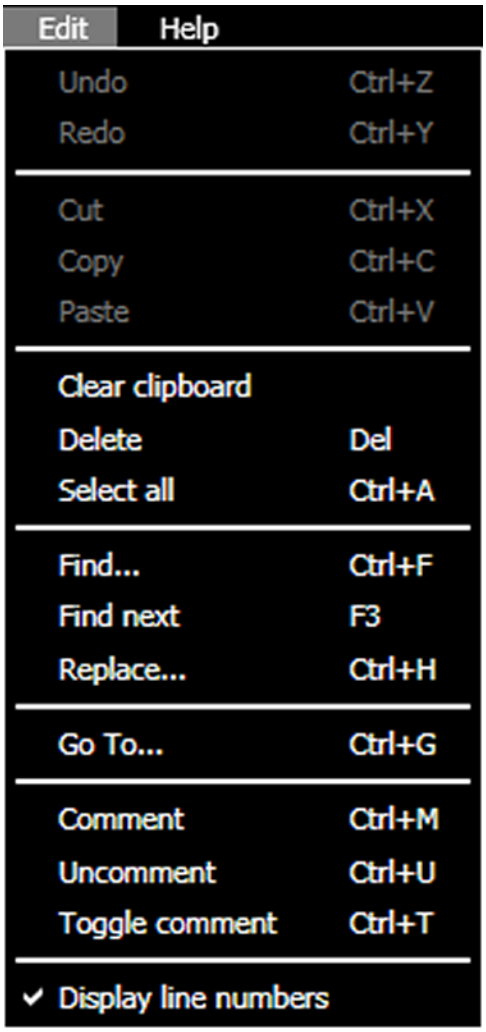
Close

Closes the editor. If changes have been made, FlashCut prompts you to save or discard these changes. After the editor has closed, any changes made will be displayed in the programming window.

Edit Menu

The **Edit** menu contains the following commands:

- Undo
- Redo
- Cut
- Copy
- Paste
- Clear Clipboard
- Delete
- Select All
- Find
- Find Next
- Replace
- Go To
- Comment
- Uncomment
- Toggle Comment
- Display Line Numbers



Undo

Retracts the last edit you made.

Redo

Repeats the last edit that you made.

Cut

Moves the currently selected text to the Windows clipboard.

Copy

Copies the currently selected text to the Windows clipboard.

Paste

Inserts the text on the Windows clipboard into the editor, at the current cursor location.

Clear Clipboard

Removes all copied data from the clipboard

Delete

Deletes the currently selected text.

Select All

Selects all the text in the editor.

Find

Opens the Find dialog box, which lets you search for text.

Find Next

Finds the next occurrence of the search text.

Replace

Opens the Replace dialog box, which lets you search for text and replace it with new text.

Go To

Opens the Go To dialog box, which lets you jump to any line in the file.

Comment

Adds an open parenthesis to the beginning of all selected lines, so FlashCut will ignore the lines.

Uncomment

Removes the open parentheses from the beginning of all selected lines, so FlashCut will execute the lines.

Toggle Comment

Either adds or removes an open parenthesis to the beginning of all selected lines depending on their current state (e.g. the selected line(s) is commented out this command will remove this open parenthesis).

Display Line Numbers

Turns editor line numbering on and off. Note that these numbers are different than the "N" numbers found in some G-code Files.

Help Menu

The **Help** menu provides the following command:

- Programming Reference

Programming Reference

Displays this FlashCut CNC Programming Reference. Requires a PDF viewer, such as Adobe® Acrobat®

Codes and Commands

This section lists the commands that FlashCut supports:

G-Codes Supported

G00 Rapid Tool Positioning

G01 Linear Interpolated Feedrate Move

G02 Clockwise Circular Feedrate Move

G03 Counter Clockwise Circular Feedrate Move

G04 Dwell

G17 XY Plane Selection

G18 XZ Plane Selection

G19 YZ Plane Selection

G20 Inch Units (same as G70)

G21 Metric Units (same as G71)

G27 Home to Switches

G28 Move to Reference Point 1

G29 Rapid Return from Reference Point

G29.1 Feedrate Return from Reference Point

G30 Move to Reference Points 2-20

G31 Seek Sensor

G40 Cancel Cutter Compensation

G41 Cutter Compensation (Left)

G42 Cutter Compensation (Right)

G50 Cancel Scaling

G51 Scaling

G52 Use Local Coordinate System

G53 Linear Move to Machine Coordinates (Rapid)

G53.1 Linear Move to Machine Coordinates (Feedrate)

G54-59 Set Program Zero (Fixture Offsets)

G54.1 Set Program Zero (Extended Fixture Offsets)

G68 Initiate coordinate system rotation

G69 Cancel coordinate system rotation

G70 Inch Units (same as G20)

G71 Metric Units (same as G21)

G73 High Speed Peck Drilling Cycle

G74 Left-Handed Tapping Cycle

G76 Thread Cutting Cycle

G80 Cancel Canned Cycle

G81 Drilling Cycle

G82 Counterboring Cycle

G83 Peck Drilling Cycle

G84 Standard Tapping Cycle

G85 Reaming Cycle

G90 Absolute Positioning Mode

G91 Incremental Positioning Mode

G92 Set Program Zero

G93 Inverse Time Feedrate

G94 Standard Feedrate

G98 Return to Initial Level (Canned Cycles)

G99 Return to Rapid Level (Canned Cycles)

G120 Sense Tool Length

G140 Set Engraving Font

G141 Engrave Text

G180 Turn Off Safe Envelope

G181 Turn On Safe Envelope

G600 End Plasma Cutting

G601 Begin Plasma Cutting

G602 Suspend Torch Height Control

G603 Resume Torch Height Control

G605 Move to Initial Height

G607 Move to Cut Height

G610 Create Dimple

G611 Pierce

G612 Laser Pierce

G630 Resume Auto Lift Axis Moves

G631 Suspend Auto Lift Axis Moves

M-Codes Supported

M00 Program Pause

M01 Optional Program Pause

M02 End of Program

MXX Custom Programmable (See "M-code Definitions" in the Initial Setup section of this manual). Typical functions are:

M03 Spindle On

M05 Spindle Off

M06 Tool Change

M07 Mist Coolant On

M08 Flood Coolant On

M09 Coolant Off

M50 Plasma On

M51 Plasma Off

M30 End of Program (Reset)

M30.1 End of Program (Reset and Restart)

M98 Subroutine Call

M99 Return From Subroutine

M100 Wait for Input Line (Normal State)

M101 Wait for Input Line (Tripped State)

M102 Enable Feed Hold Input Line

M103 Disable Feed Hold Input Line

M106 Select Fabrication Head

Other Commands Supported

F	Set Feedrate
T	Select Tool
S	Set Variable Output Port (eg. spindle speed)
( )	Comment
/	Optional Line
BEGINGRID	Begin a Grid of Parts
ENDGRID	End a Grid of Parts

Advanced Keyword Commands and Functions

EnableFO	Enable Feedrate Override
DisableFO	Disable Feedrate Override
SetFO	Set Feedrate Override Percent
SaveState	Save Current Modal States
RestoreState	Restore Saved Modal States
GetRackIndex	Get Tool Rack Index

Key Programming Concepts

There are two basic programming concepts you should understand before learning the G- and M codes:

- Mode
- Absolute vs. Incremental

Mode

Most G-code commands supported by FlashCut are modal, meaning they put the system into a particular mode of operation and don't need to be repeated on every program line. A modal command stays in effect until another command changes the mode. Related modal commands that affect one aspect of program execution are called a mode group.

These are the mode groups supported by FlashCut:

- Movement
- Circular Interpolation
- Units
- Cutter Compensation
- Error! Reference source not found.
- Scaling
- Set Program Zero
- Canned Cycles
- Canned Cycle Return
- Positioning
- Safe Envelope
- Feedrate
- Independent Modal Commands

Movement

G00 Rapid Tool Positioning

G01 Linear Interpolated Feedrate Move

G02 Clockwise Circular Feedrate Move

G03 Counter Clockwise Circular Feedrate Move

Circular Interpolation

G17 XY Plane Selection

G18 XZ Plane Selection

G19 YZ Plane Selection

Units

G20 Inch Units (also G70)

G21 Metric Units (also G71)

Cutter Compensation

G40 Cancel Cutter Compensation

G41 Cutter Compensation (Left)

G42 Cutter Compensation (Right)

Scaling

G50 Cancel Scaling

G51 Scaling

Set Program Zero

G54-59 Set Program Zero (Fixture Offsets)

G54.1 Set Program Zero (Extended Fixture Offsets)

G92 Set Program Zero (XYZA Parameters)

Canned Cycles

G73 High Speed Peck Drilling Cycle

G74 Left-Handed Tapping Cycle

G76 Thread Cutting Cycle

G80 Cancel Canned Cycle

G81 Drilling Cycle

G82 Counterboring Cycle

G83 Peck Drilling Cycle

G84 Standard Tapping Cycle

G85 Reaming Cycle

Canned Cycle Return

G98 Return to Initial Level

G99 Return to Rapid Level

Positioning

G90 Absolute Positioning Mode

G91 Incremental Positioning Mode

Safe Envelope

G180 Turn Off Safe Envelope

G181 Turn On Safe Envelope

Feedrate

G93 Inverse Time Feedrate

G94 Standard Feedrate

Independent Modal Commands

G52 Use Local Coordinate System

F Set Feedrate

S Set Spindle Speed

G140 Set Engraving Font

Absolute vs. Incremental

All moves are either absolute or incremental. In an absolute move, the ending point is defined relative to Program Zero. In an incremental move, the ending point is defined relative to the current tool location. The G90/G91 commands tell the system which of these two modes to use (described below).

While there will be cases where incremental programming is useful, generally you should define your moves as absolute since it is a less error prone method of programming. All of the examples in the following section use absolute positioning unless otherwise noted.

G and M-code Reference

G00 Rapid Tool Positioning

The G00 command moves the tool to the designated coordinate at the rapid rate using linear interpolation. The rapid rate is calculated from the Maximum Feedrates defined on the Feedrate/Ramping panel of the Configuration dialog box.

Example:

```
G00 X1.0 Y2.0 Z1.5          Moves the tool to program coordinates X=1.0,
                            Y=2.0, Z=1.5 at the rapid rate
```

When using G00, there are several things to keep in mind:

- You do not need to specify all coordinates, only the ones for which you want movement.

  Example:

  ```
  G00 X4.0 Y3.0             Moves the tool to program coordinates X=4.0,
                            Y=3.0, leaving the Z position unchanged
  ```

- This is a modal command, meaning that all successive moves will be treated as rapid moves until another modal move command (G01, G02 or G03) occurs.

  Example:

  ```
  G00 X1.0 Y2.0 Z1.5        Rapid Move
  X4.0 Y6.5 Z1.0            Rapid Move
  G01 X3.0 Y3.0 Z1.4        Feedrate Move
  X2.8 Y1.4 Z0              Feedrate Move
  ```

- The interpretation of the coordinates depends on the G90/G91 command in effect.
- For rotary axes, FlashCut will make the shortest possible move in the direction indicated by the command (always less than 360 degrees.)

  Example (starting at program coordinate A = 0):

  ```
  G00 A365                  Rapid move 5 degrees, after which the DRO reads
                            A=365
  ```

- Express Mode is a setting that affects how the signal generator processes motion commands. It's located on the Controller panel of the Configuration screen. When the signal generator is running in Express Mode, it can move up to five axes simultaneously.

  When **not** in Express Mode, FlashCut can only move up to three axes simultaneously. If you specify a move on four axes, FlashCut will perform two separate moves, one for the Z axis and one for the X-Y-A axes. The Z move will occur first if the direction is towards the home end of the axis (specified by the Home End setting on the Basic Homing panel of the Configuration dialog box). Otherwise the X-Y-A move will occur first. If this behavior doesn't suit your needs, use two G00 commands.

  Example (starting at program zero, home end for Z axis is positive):

  ```
  G00 X1.0 Y2.0 Z1.5 A0     Moves the tool to program coordinates X=1.0,
                            Y=2.0, Z=1.5, leaving the A position unchanged
  ```

```
                              (the A parameter is included but does not specify
                              any motion)
G00 X2.0 Y3.0 Z1.0 A90        First moves the tool to X=2.0, Y=3.0, A=90,
                              leaving the Z position unchanged, then moves the
                              tool to Z=1.0, leaving the X, Y and A axes
                              unchanged
```

G01 Linear Interpolated Feedrate Move

The G01 command moves the tool to the designated program coordinate at the specified feedrate using linear interpolation.

Example:

```
G01 X2.0 Y1.0 Z-1.5 F2.0      Moves the tool to program coordinates X=2.0,
                              Y=1.0, Z=-1.5 at a feedrate of 2.0 in/min
```

When using G01, there are several things to keep in mind:

- You do not need to specify all coordinates, only the ones for which you want movement.

  Example:

  ```
  G01 X4.0 Y3.0 F2.0          Moves the tool to program coordinates X=4.0,
                              Y=3.0, leaving the Z position unchanged
  ```

- This is a modal command, meaning that all successive moves will be treated as linear feedrate moves until another modal move command (G00, G02 or G03) occurs.
- The interpretation of the coordinates depends on the G90/G91 command in effect.
- The F command is used to designate a feedrate. The feedrate set with the F command is modal (stays in effect until another F command occurs).

  Example:

  ```
  G01 X4.0 Y3.0 Z1.0 F7.0     Moves the tool to program coordinates X=4.0,
                              Y=3.0, Z=1.0 at a feedrate of 7.0 in/min
  X2.0 Y2.5                   Moves the tool to program coordinates X=2.0 Y=2.5,
                              leaving the Z axis unchanged at Z=1.0 (the
                              feedrate remains 7.0 in/min)
  ```

- For any move that includes a rotary axis (A, B or C), there are two ways FlashCut can interpret the current F setting. FlashCut uses your setting for 'F Command Interpretation for Rotary Moves' on the Rotary Axes panel of the Configuration dialog box.

  Linear Feedrate (Distance/Minute) – This option directs FlashCut to interpret the feedrate command (F) as a linear feedrate. The speed of the tool tip relative to its point of contact on the rotating workpiece will be at the specified F value. In this mode, you can use a single F command to cover linear, linear plus rotary, and rotary moves.

  Rotary Feedrate (Degrees/Minute) – This option directs FlashCut to interpret the feedrate command (F) as a rotary feedrate. The rotary axis controls the timing. The F parameter is interpreted as degrees/minute.

  Example:

  ```
  G01 X1 A90 F360
  ```

The A axis turns 90 degrees while the X axis moves 1 inch. The A axis turns at exactly 360 degrees/minute. The timing for the X axis is based on the A axis motion and is not explicitly listed in the G-code file.

Note that if you choose the rotary feedrate interpretation, the F feedrate in use is always interpreted as degrees/minute for any move that contains motion on a rotary axis. For all other moves, it's interpreted as inches/minute (or mm/min). Therefore, you must be very careful to reset F every time you switch between pure linear and rotary-included moves.

- Express Mode is a setting that affects how the signal generator processes motion commands. It's located on the **Controller** panel of the Configuration screen. When the signal generator is running in Express Mode, it can move up to five axes simultaneously.

  When **not** in Express Mode, FlashCut can only move up to three axes simultaneously. If you specify a move on four axes, FlashCut will perform two separate moves, one for the Z axis and one for the X-Y-A axes. The Z move will occur first if the direction is towards the home end of the axis (specified by the Home End setting on the Basic Homing panel of the Configuration dialog box). Otherwise the X-Y-A move will occur first. If this behavior doesn't suit your needs, use two G01 commands.

  Example (starting at program zero, home end for Z axis is positive):

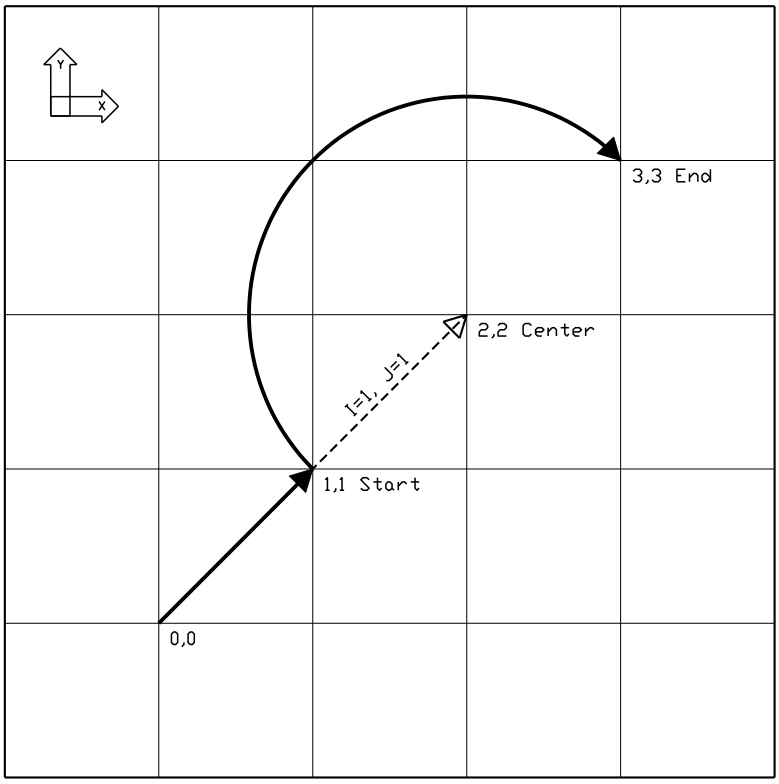
`G01 X1.0 Y2.0 Z1.5 A0`	`Moves the tool to program coordinates X=1.0, Y=2.0, Z=1.5, leaving the A position unchanged (the A parameter is included but does not specify any motion)`
`G01 X2.0 Y3.0 Z1.0 A90`	`First moves the tool to X=2.0, Y=3.0, A=90, leaving the Z position unchanged, then moves the tool to Z=1.0, leaving the X, Y and A axes unchanged`

G02 Clockwise Circular Feedrate Move

The G02 command moves the tool in a clockwise path from the starting point (the current tool position) to the designated ending point in the currently selected plane (see G17-G19). The I , J, and K parameters represent the incremental X, Y, and Z distances (respectively) from the starting point of the arc to the center point of the arc.

Example:

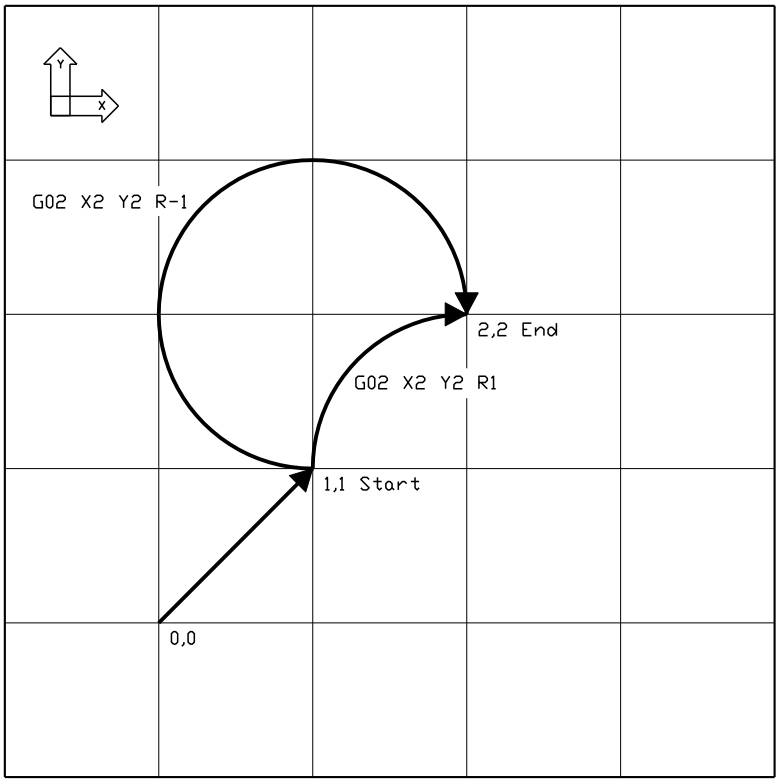
`G01 X1 Y1 F3`	Moves the tool to program coordinates X=1, Y=1 at a feedrate of 3 in/min
`G02 X3 Y3 I1 J1`	Moves the tool using clockwise circular interpolation to program coordinates X=3, Y=3 with a center point of X=2, Y=2 at a feedrate of 3 in/min



An alternative to specifying the distance to the center point is to specify the radius, using the R parameter. Usually this is easier than determining the correct I, J and K values. For any given radius, generally there are two possible arcs: one that sweeps an angle less than 180 degrees, and one that sweeps an angle greater than 180 degrees (see diagram below). To specify an angle less than 180 degrees, make R a positive number; to specify an angle greater than 180 degrees, make R a negative number.

Example:

`G01 X1 Y1 F3`	Moves the tool to program coordinates X=1, Y=1 at a feedrate of 3 in/min
`G02 X2 Y2 R1 (or R-1)`	Moves the tool using clockwise circular interpolation to program coordinates X=2, Y=2



When using the R word, please note:

- If the arc sweeps a 180 degree angle, it doesn't matter whether R is negative or positive.
- If the ending point is the same as the starting point, FlashCut will ignore the command, since the center point cannot be determined.

You can use the G02 command to specify a helical move (helical interpolation). During a helical move, the circular motion described above is combined with linear motion that's perpendicular to the plane containing the arc. For example, circular motion in the XY plane is combined with linear motion along the Z axis to form a helix.

To specify helical motion, add an X, Y or Z parameter to the command, to indicate the ending point of the linear motion. In the following example, a Z parameter has been added to the G02 command for an arc in the XY plane.

Example:

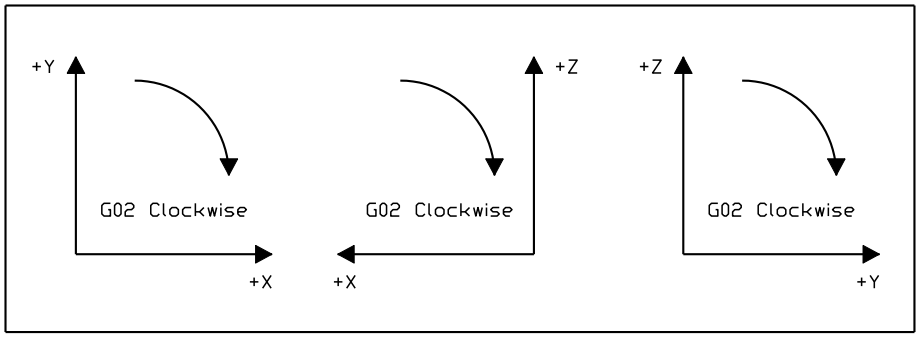
```
G01 X1 Y1 Z1 F3          Moves the tool to program coordinates X=1, Y=1,
                         Z=1 at a feedrate of 3 in/min

G02 X3 Y3 Z2 I1 J1       Moves the tool using clockwise circular
                         interpolation to program coordinates X=3, Y=3 with
                         a center point of X=2, Y=2, while simultaneously
                         moving the tool in a straight line along the Z
                         axis to Z=2
```

The accuracy of the helical move, relative to a theoretically perfect helix, is controlled by the Helical Interpolation Accuracy setting on the G-code panel of the Configuration dialog box (see "G-code Settings" in the Initial Setup section).

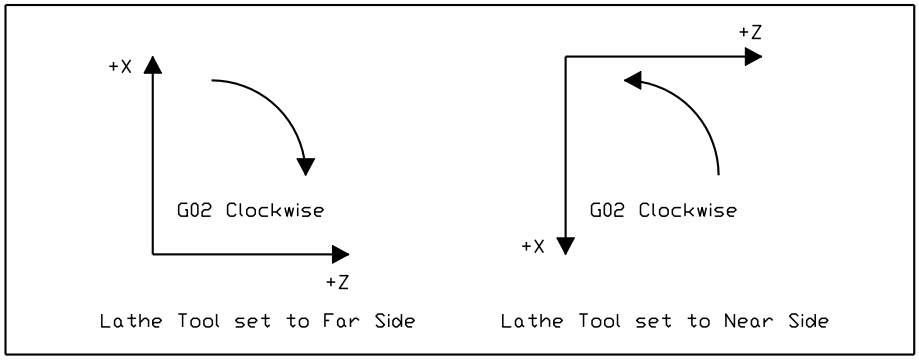
In knife cutting applications, you can include rotary motion as part of the G02 command, by adding the A, B or C parameter to the G-code line. The rotary axis will turn proportionately throughout the arc to keep the knife blade tangent to the cutting direction. To use this option you must first configure your machine for knife cutting. Please see "Rotary Axes Settings" in the Initial Setup section for more information. Also note that the Helical Interpolation Accuracy setting on the G-code panel of the Configuration dialog box applies to these arc moves.

When using G02, there are several things to keep in mind:

- This is a modal command, meaning that all successive moves will be treated as clockwise circular feedrate moves until another modal move command (G00, G01 or G03) occurs.
- The interpretation of the X, Y and Z coordinates depends on the G90/G91 command in effect. The I, J and K values are unaffected by G90/G91.
- The tool will move at the current feedrate set by the last F command.
- Only XY arcs can be cut when G17 is active, only XZ arcs can be cut when G18 is active, and only YZ arcs can be cut when G19 is active. Arcs cannot be specified for the A or W axes.
- The clockwise direction of rotation is as viewed from the positive end of the unused axis (the axis not in the plane of motion). For example, a G02 arc move in the XY plane is clockwise as viewed from the positive end of the Z axis (i.e. from above). The following diagram illustrates this behavior for all three arc planes:



In some cases, a clockwise arc as defined above will be displayed counter clockwise in a workspace. For example, in lathe applications, the Lathe Tool setting (on the Basic Definition panel of the Configuration dialog box) determines whether the X positive direction is up or down in the ZX workspace. When Lathe Tool is set to Near Side, the X positive direction is down and G02 arcs are displayed counter clockwise on the screen.

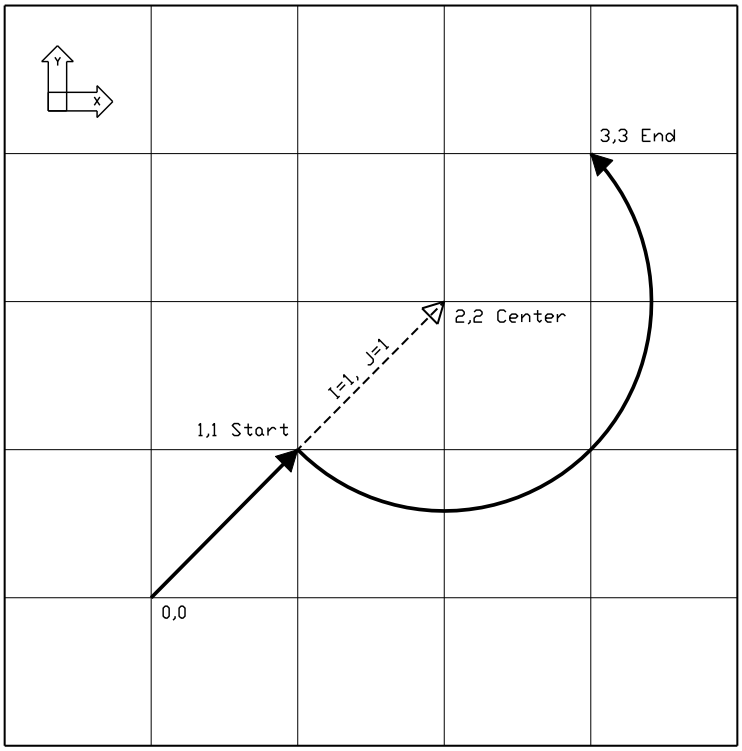


G03 Counter Clockwise Circular Feedrate Move

The G03 command is identical to the G02 command, but it moves the tool in a counter clockwise arc instead of a clockwise arc.

Example:

`G01 X1 Y1 F3`	Moves the tool to program coordinates X=1, Y=1 at a feedrate of 3 in/min
`G03 X3 Y3 I1 J1`	Moves the tool using counter clockwise circular interpolation to program coordinates X=3, Y=3 with a center point of X=2, Y=2 at a feedrate of 3 in/min



G04 Dwell

The G04 command causes the program to dwell for a specified amount of time. The command format is:

```
G04 Xn
```

where n is the number of seconds to wait. For safety reasons there is a maximum time allowed for each dwell command.

Example:

```
G04 X1.5                    Program pauses for 1.5 seconds before moving on to
                            the next line of G-code
```

For compatibility reasons, FlashCut also allows the following format:

```
G04 Pn
```

where n is the number of seconds or milliseconds to wait, depending on your "G04 P Parameter Units" setting on the G-code panel of the Configuration dialog box.

Example (P parameter in milliseconds):

```
G04 P3000                   Program pauses for 3 seconds before moving on to
                            the next line of G-code
```

G17, G18, G19 Arc Plane Selection

These commands specify the plane used for circular interpolation as follows:

G17 XY plane

G18 XZ plane

G19 YZ plane

When using G17-G19, there are several things to keep in mind:

- Unless you explicitly use the G18 or G19 command, FlashCut assumes G17 as the default.
- The three commands are modal, i.e. one command remains in effect until another in the set is used.

G20, G21 Inch Units and Metric Units

The G20 command indicates that all G-code commands are in inch units. FlashCut then assumes all distances are in inches and all feedrates are in inches/minute. For compatibility reasons, FlashCut accepts G70 as equivalent to G20.

The G21 command indicates that all G-code commands are in metric units. FlashCut then assumes all distances are in millimeters and all feedrates are in millimeters/minute. For compatibility reasons, FlashCut accepts G71 as equivalent to G21.

You will only need these commands when the units used in the G-code file don't match the System Units setting on the System:General panel of the Configuration dialog box. If you don't use either command, FlashCut assumes all program values are consistent with the configuration setting.

G27 Home to Switches

The G27 command homes the machine tool to its home switches, in a similar manner as the Home All command on the Home dropdown menu of the Machine Coordinates Panel in the CNC window. The purpose of this command is to allow checking for positioning errors, and to allow re-zeroing of any axis. This is especially useful when running large G-code files. The command lets you home the machine at any point in the file, and optionally have FlashCut display the homing discrepancy dialog if the positioning error is not within the desired tolerance (which pauses execution of the file).

If you want the G27 command to home only some of the axes on your machine tool, you can limit homing to those axes by adding each axis letter followed by a zero (eg. "X0") after the command. Then, FlashCut homes only the indicated axes.

Example:

```
G27                          Homes all axes
G27 X0 Y0                    Homes the X and Y axes
```

Note that Machine Zero must be set before you run a G-code file that uses this command.

G28, G30 Move to Reference Point

The G28 and G30 commands move the tool at the rapid rate to the associated reference point defined on the Reference Points panel of the Configuration dialog box. These positions are defined either in machine or program coordinates. When a position is defined in machine coordinates, Machine Zero must be set for this command to be used. Note that G28 typically is used for moving to the tool change position and/or Machine Zero (reference point 1).

The sequence of axis motions follows a general-purpose scheme that's described under "Point Control Panel" in the Main Screen Features section of this manual.

If you want the G28/G30 command to move only some of the axes on your machine tool, you can limit the movement to those axes by adding each axis letter followed by a zero (eg. "X0") after the command. Then, FlashCut moves only the indicated axes to the reference point coordinates. A typical use is to raise only the Z axis for a manual tool change ("G28 Z0").

Example (milling application):

Reference point 1 is defined as machine coordinate X=1, Y=1, Z=-1. Reference point 2 is defined as machine coordinate X=2, Y=2, Z=-2.

```
G01 X1.5 Y2.5 Z-3 F10        Linear move to the program coordinates X=1.5,
                             Y=2.5, Z=-3
G28 Z0                       Rapid move in the Z axis only to machine
                             coordinate Z=-1
G01 X1.5 Y2.5 Z-3 F10        Linear move to the program coordinates X=1.5,
                             Y=2.5, Z=-3
G28                          Rapid move in the Z axis to machine coordinate Z=-
                             1 followed by a rapid move in the XY plane to
                             machine coordinates X=1, Y=1
```

```
G01 X1.5 Y2.5 Z-3 F10          Linear move to the program coordinates X=1.5,
                               Y=2.5, Z=-3
G28 X0 Y0 Z0                   Rapid move in the Z axis to machine coordinate Z=-
                               1 followed by a rapid move in the XY plane to
                               machine coordinates X=1, Y=1 (identical to the G28
                               command with no parameters specified)
G30 P2                         Rapid move in the XY plane to machine coordinates
                               X=2, Y=2 followed by a rapid move in the Z axis to
                               machine coordinate Z=-2
```

G29, G29.1 Return from Reference Point

The G29 command moves the tool to the designated program coordinate at the rapid rate. The G29.1 command moves the tool to the designated program coordinate at the specified feedrate. The feedrate for G29.1 is specified with a L parameter for linear feedrates and a R parameter for rotatory feedrates.

The sequence of axis motions follows a general-purpose scheme that's described under "Point Control Panel" in the Main Screen Features section of this manual.

Example:

Reference point 1 is defined as machine coordinate X=1, Y=1, Z=-1.

```
G01 X1.5 Y2.5 Z-3 F10          Linear move to program coordinates X=1.5, Y=2.5,
                               Z=-3
G28                            Rapid move in the Z axis to machine coordinate Z=-
                               1 followed by a rapid move in the XY plane to
                               machine coordinates X=1, Y=1
G29 X2 Y3 Z-2                  Rapid move in the XY plane to program coordinates
                               X=2, Y=3 followed by a rapid move in the Z axis to
                               program coordinate Z=-2
G29.1 X4 Y5 Z-3 L10            Feedrate move in the XY plane to program
                               coordinates X=4, Y=5 followed by a feedrate move
                               in the Z axis to program coordinate Z=-3
```

When using G29, there are several things to keep in mind:

- You do not need to specify coordinates for all machine tool axes, only the ones for which you want movement.

  Example:

  ```
  G29 X4.0 Y3.0                Moves the tool to program coordinates X=4.0,
                               Y=3.0, leaving the Z position unchanged
  ```

- The interpretation of the coordinates depends on the G90/G91 command in effect.

G31 Seek Sensor

G31 tells FlashCut to move a single axis until a sensor is activated, similar to tool length sensing and homing. The syntax is:

G31 Xx [Yy] [Zz] [Aa] [Bb] [Cc] Ii Ss Ee Ff

where the parameters are

x, y, z, a, b, or c:

Endpoint for the move (only one axis may be used)

i: Input line to monitor

s: Switch state that ends the move (0 = untripped, 1 = tripped)

e: Coordinate system for the endpoint (0 = program coordinates, 1 = machine coordinates)

f: Feedrate

Example:

```
G31 Z-2.0 I4 S1 E0 F5          Moves the Z axis to program coordinate -2.0 at 5
                               inches/minute, stopping when input line 4 becomes
                               tripped
```

A common use for the G31command is to adjust a tool offset, not because the length of the tool has changed, but to allow the system to adjust its motion to match an irregularly shaped workpiece. For example, a milling machine can compensate all Z motion to match a workpiece with an unknown height, by sensing the top surface and adjusting the Z offset for the current tool.

To set a tool offset using G-code, set the system variable #Tool as shown in these two examples:

```
#Tool{1}.z = #Machine.Z        Sets the Z offset for tool 1 to the current Z
                               machine coordinate

#Tool{3}.x = #Machine.X + 2    Sets the X offset for tool 3 to the current X
                               machine coordinate plus 2
```

For more details on using variables, see **Advanced Programming Reference** later in this section.

Note: In previous versions of FlashCut a ‘C’ parameter was used for the coordinate system instead of the ‘E’ parameter. If your machine tool does not have a C axis, FlashCut still allows the ‘C’ parameter for the coordinate system in the G31 command.

G40, G41, G42 Cutter Compensation

Cutter compensation allows FlashCut to automatically adjust the toolpath to account for the radius of the cutting device (tool, torch, or other device). This provides the following benefits:

- You can write your G-code program to reflect the dimensions of the part, rather than calculating where the centerline of the tool will move.
- You can easily compensate for tool wear, when the exact diameter of the tool is not known when the G-code file is created.
- You can precisely control the final dimensions of a part, by artificially changing the tool diameter to reposition the toolpath.

The commands are defined as:

G41 Compensate to the left of the programmed path

G42 Compensate to the right of the programmed path

G40 Cancel cutter compensation

The format for the G41/42 command is

G41 Dn (or G42 Dn)

where n is the tool number for the current tool. FlashCut offsets the toolpath by half the diameter.

Note FlashCut now allows G41/G42 cutter compensation with no tool parameter.

Prior to the first G41/42 command, you should indicate the current tool using the M06 command (e.g. M06 T1 to select tool 1). If you leave this out, FlashCut will warn you that the tool number in the G41/42 command does not match the current tool (but the file will still run).

The G40 command (cancel cutter compensation) requires no parameters. Note that G54-59 and G92 also cancel cutter compensation.

You'll activate cutter compensation just before cutting a feature (profile, pocket, etc.) then cancel cutter compensation immediately upon completing the feature. The typical sequence of commands is –

1. Select tool with M06 command (e.g. M06 T1)
2. Move Z axis up to safe height above workpiece
3. Activate cutter compensation using G41 or G42
4. Move X and Y axes together (G00 or G01) to a point on the feature (called the 'approach' move)
5. Move Z axis down into the workpiece
6. Do any number of XY moves (lines and arcs) to cut the feature, all at the same Z axis depth
7. Move Z axis up to safe height above workpiece
8. Cancel cutter compensation using G40
9. Repeat steps 3-8 for all features

Be sure to follow these steps for each distinct feature of the part. Cutter compensation should be turned off whenever the machine is not actually cutting a feature (e.g. when returning to Machine Zero or the tool change position).

The following example demonstrates use of cutter compensation.

Example:

This G-code file cuts on the inside of two 1-inch squares.

```
F10.00                      Set feedrate
M06 T1                      Select tool 1
G00 Z 0.25                  Raise Z to safe height
G41 D1                      Activate cutter comp left
G00 X 1.00 Y 1.00           Approach move
G01 Z-0.10                  Plunge into workpiece
G01 X 2.00 Y 1.00           Cut first side of square
G01 X 2.00 Y 2.00
G01 X 1.00 Y 2.00
G01 X 1.00 Y 1.00           Cut last side of square
G00 Z 0.25                  Raise Z to safe height
G40                         Cancel cutter compensation
```

```
G41 D1                      Activate cutter comp left
G00 X 3.00 Y 1.00           Approach move
G01 Z-0.10                  Plunge into workpiece
G01 X 4.00 Y 1.00           Cut first side of square
G01 X 4.00 Y 2.00
G01 X 3.00 Y 2.00
G01 X 3.00 Y 1.00           Cut last side of square
G00 Z 0.25                  Raise Z to safe height
G40                         Turn off cutter compensation
G00 X 0.00 Y 0.00           Return to program zero
```

G50, G51 Scaling

The G51 command lets you scale all motion about program zero or any other point you choose.

The simplest syntax is:

G51 Pp

where ‘p’ is the scale for all axes.

Example:

```
G51 P1.5                    Scales motion on all axes by a factor of 1.5,
                            relative to program zero
```

You can also set the scaling per axis using this syntax:

G51 Ii Jj Kk

where the parameters are

i: X axis scale

j: Y axis scale

k: Z axis scale

Example:

```
G51 I2 J2 K1                Scales motion on the X and Y axes by a factor of
                            2, relative to program zero. Z axis motion is not
                            scaled.
```

Finally, instead of scaling motion about program zero, you can specify the scaling origin using the following syntax:

G51 Xx Yy Zz Ii Jj Kk

where the additional parameters are

x: X axis scaling origin

y: Y axis scaling origin

z: Z axis scaling origin

Example:

```
G51 X3 Y4 Z-1 I2 J2 K0.5        Scales motion on the X, Y and Z axes about the
                                point X=3, Y=4, Z=-1
```

The G50 command turns scaling off.

Example:

```
G51 P4
G01 X1 Y1 F5                    This move is scaled
G50
G01 X2 Y2                       This move is not scaled
```

G52 Local Coordinate System

The G52 command activates a local coordinate system that FlashCut uses in place of your original program coordinates for all absolute positioning moves. The X, Y, Z, A, B and C parameters indicate the offset from your Program Zero location to the origin for the local coordinate system.

For example, "G52 X1 Y2 Z-4" would activate a local coordinate system whose origin is at a distance of 1, 2, -4 from the original Program Zero.

All absolute moves are made relative to the new local coordinate system. To cancel use of the local coordinate system in the middle of a G-code file, use the command “G52 X0 Y0 Z0 A0 B0” (using only the letters for axes used on your machine).

When FlashCut reads a G52 command, it displays a magenta dot in the workspace showing the origin of the local coordinate system.

Note that the local coordinate system only applies to the G-code file being executed. The G52 command has no effect the Program Zero location. FlashCut automatically cancels the local coordinate system when it completes execution of a G-code file.

If you reposition Program Zero using the G54-59 or G92 commands, the G52 command remains in effect, with local coordinate system now offset from the new Program Zero location.

Example:

```
G01 X1.0 Y3.0 Z-1.5 F12         Moves the tool to program coordinates X=1.0,
                                Y=3.0, Z=-1.5
G52 X3 Y-7 Z0                   Activates a local coordinate system with origin at
                                X=3, Y=-7, Z=0 relative to Program Zero (the
                                machine tool does not move)
G01 X1.0 Y10.0 Z2.0             Moves the tool to the point X=1.0, Y=10.0, Z=2.0
                                relative to the local coordinate system as defined
                                by the G52 command above
G52 X0 Y0 Z0                    Cancels use of the local coordinate system. All
                                absolute moves are again relative to Program Zero
                                as you set it before running the program
```

G53, G53.1 Linear Move to Machine Coordinates

The G53 and G53.1 commands move the tool to the designated Machine coordinate using 3-axis linear interpolation. The G53 command causes a rapid move, while the G53.1 command causes a feedrate move. These commands are identical to the G00 and G01 commands except for the following:

- The destination is specified in Machine coordinates.
- The commands are not modal. You must include the G53 or G53.1 command on any G-code line where you want motion to a Machine coordinate.
- You may include an optional 'D' parameter to limit each axis to one direction of motion, as follows:

  D1: Motion must be positive. Any axis that would move in a negative direction doesn't move.

  D-1: Motion must be negative. Any axis that would move in a positive direction doesn't move.

  A typical use for the 'D' parameter is to suppress motion in an unwanted direction, when the move is part of a subroutine or M-code macro (e.g. for automatic tool changing).

Example:

```
G53.1 X2 Y3 Z-4 F12          Moves the tool to machine coordinates X=2.0,
                             Y=3.0, Z=-4.0 at a feedrate of 12

G53 X3 Y4 Z-5 D1             Moves the tool to machine coordinates X=3.0,
                             Y=4.0, Z=-4.0 at the rapid rate (Z axis motion is
                             suppressed since it would be in the negative
                             direction)
```

For more information, please see **G00 Rapid Tool Positioning** and **G01 Linear Interpolated Feedrate Move** above.

G54-59, G54.1, G92 Set Program Zero Commands

The G54-59, G54.1 and G92 commands reset the Program Zero location.

The G54-59 and G54.1 commands set Program Zero to a predefined offset away from Machine Zero. They are particularly useful when using a workpiece fixture; the offset from Machine Zero to the fixture can be set once, and then used over and over. These commands use the offsets defined on the Fixture Offsets panel of the Configuration dialog box.

Example:

If the offset for G54 is set to

X = 1

Y = 2

Z = -3

A = 90

on the Fixture Offsets panel, the G54 command would place Program Zero at machine coordinate X=1, Y=2, Z=-3, A=90. The result is the same as if you moved the machine tool to machine coordinate X=1, Y=2, Z=-3, A=90, then clicked the Zero button for program coordinates. The G54-59 and G54.1 commands require Machine Zero to be set.

The G54.1 command allows access to 100 fixture offsets using the following syntax:

G54.1 Pn

where 'n' is any number 1 through 100. This command raises the total number of available fixture offsets to 106 (6 for G54-59, plus 100).

The G92 command is similar to the G54-59 and G54.1 commands, except **the new program coordinates for the current location** are specified in the command itself. The command syntax is:

G92 [Xx] [Yy] [Zz] [Aa] [Bb] [Cc]

where 'x', 'y', 'z', 'a', 'b' and 'c' specify the new current program coordinates. These parameters are optional but you must include at least one.

Example:

```
G92 X1 Y2 Z-3 A90
```

sets the current program coordinates to X=1, Y=2, Z=-3, A=90. *The new offset from Machine Zero to Program Zero is not specified, so its value depends on where the machine happens to be positioned.* For this reason, the 54-59 and G54.1 commands are generally preferred to the G92 command, since they use a more reliable method for setting the Program Zero location.

For the G54-59, G54.1 and G92 commands, the Program Zero that's established remains in effect after the G-code file has finished execution. The function of these commands is identical to that of the Program Coordinates Zero and Set buttons on the main screen.

M06 Tool Change and T Select Tool Commands

To indicate tool changes in the G-code file, use the M06 and T commands as follows:

M06 Tn

where n is the tool number on the Tool Library panel of the Configuration dialog box.

Example:

```
M06 T3
```

The T command can be used on any line prior to the M06 command; it does not need to be on the same line as M06.

FlashCut handles the M06 command according to the settings on the Tool Change panel of the Configuration dialog box. See "Tool Change" in the Initial Setup section for details.

The M06 command does not move the machine tool to the tool change position. This is done using the G28 command described above. If you're changing tools manually, it's good practice to place the G28 command (and the M05 spindle off command) on the lines directly preceding the M06 command.

For compatibility with Fanuc G-code for lathes, FlashCut also supports the Fanuc-style lathe tool change command. The command is:

Txxyy

where the parameters are

xx: tool number

yy: wear offset number

When FlashCut encounters a T command formatted this way, it does the following:

1. Performs a tool change to tool number xx

5. Applies the geometry offset for tool number xx

Example:

```
T0404                          Changes to tool #4 and applies the geometry offset
                               for tool #4 (the ending "04" is ignored)
```

G68, G69 Coordinate Rotation Commands

The coordinate rotation commands allow you to rotate the coordinates used by the program. Coordinate rotation has several applications. For example, it can be used to correct for a skewed work piece.

FlashCut supports the following coordinate rotation commands:

G68 Initiate coordinate rotation

G69 Cancel rotation

The syntax of the G68 command is:

G68 Xx Yx Rr

where the parameters are

x: X coordinate of the center of rotation

y: Y coordinate of the center of rotation

r: Angle of rotation, which can be positive or negative. The angle is measured from the positive X axis in a counterclockwise direction

All parameters (X, Y, and R) are required.

A G68 command cannot be issued when coordinate system rotation is currently in effect. The current rotation must be canceled first.

The syntax of the G69 command is:

G69

This command cancels coordinate system rotation.

Coordinate system rotation occurs only in the X-Y plane.

Program coordinates are the only ones affected by the rotation.

The G-Code commands that are affected by the rotation angle are:

- G00, G01, G02, and G03 moves
- G28, G29, G30: reference points defined in program coordinates

- G73, G80, G81, G82, G83, and G85 cycles: the X-Y positioning move only
- G141
- G170: coordinates in the .fid file

Several commands will have their behavior modified by the rotation, although they are not directly affected by the coordinate system rotation:

- G40, G41, and G42 – Cutter compensation is applied to the toolpath, and the compensated toolpath is rotated accordingly.
- G50 and G51 – Scaling is applied to the toolpath, and the scaled path will be rotated.
- G90 and G91 – These modal commands are processed appropriately for the rotated system.

The following commands are not affected by coordinate rotation, even though they involve movement

- G28, G29, G30: reference points defined in machine coordinates
- G31
- G53
- G76

The fixture offset G-Codes (G54 – G59) cause coordinate rotation to be cancelled. G92 (setting program zero) is allowed while a rotation is in effect, and will be processed in the rotated environment.

Since program coordinates are affected by coordinate system rotation, point moves to program coordinates are performed in the rotated environment. Point moves to machine or relative coordinates are not affected by the rotation. Incremental point moves are treated as moves in the machine coordinate system, and therefore are not affected by coordinate system rotation.

Jogging (continuous and discrete) is not affected by coordinate system rotation.

Examples:

```
G68 X1 Y2 R1                A rotation angle of 1 degree, centered around
                            (1,2), will be applied to the program

G69                         Cancel rotation

G68 G54 X0 Y1 R0.13         The G54 fixture offset will be applied, then a
                            rotation angle of 0.13 degrees will be applied to
                            program coordinates, centered about (0, 1).
```

G73, G80, G81, G82, G83, G85, G98, G99 Drilling Canned Cycle Commands

Drilling canned cycles simplify programming by letting you specify standard drilling operations with a single command. FlashCut supports the following drilling cycle commands:

G73 High Speed Peck Drilling Cycle

G80 Cancel Canned Cycle

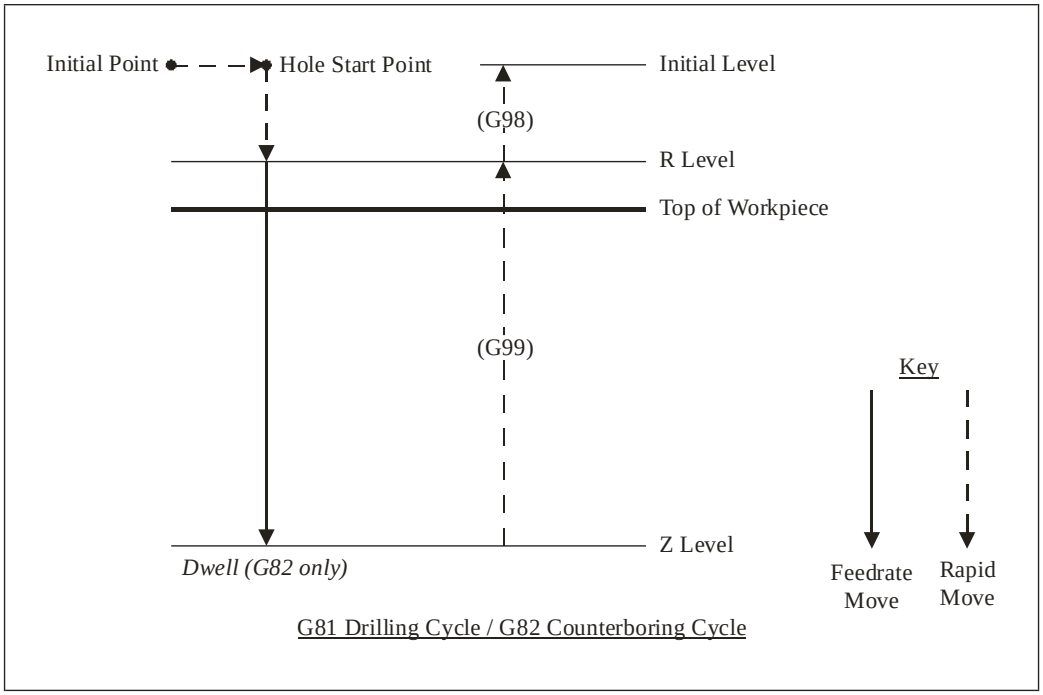
G81 Drilling Cycle

G82 Counterboring Cycle

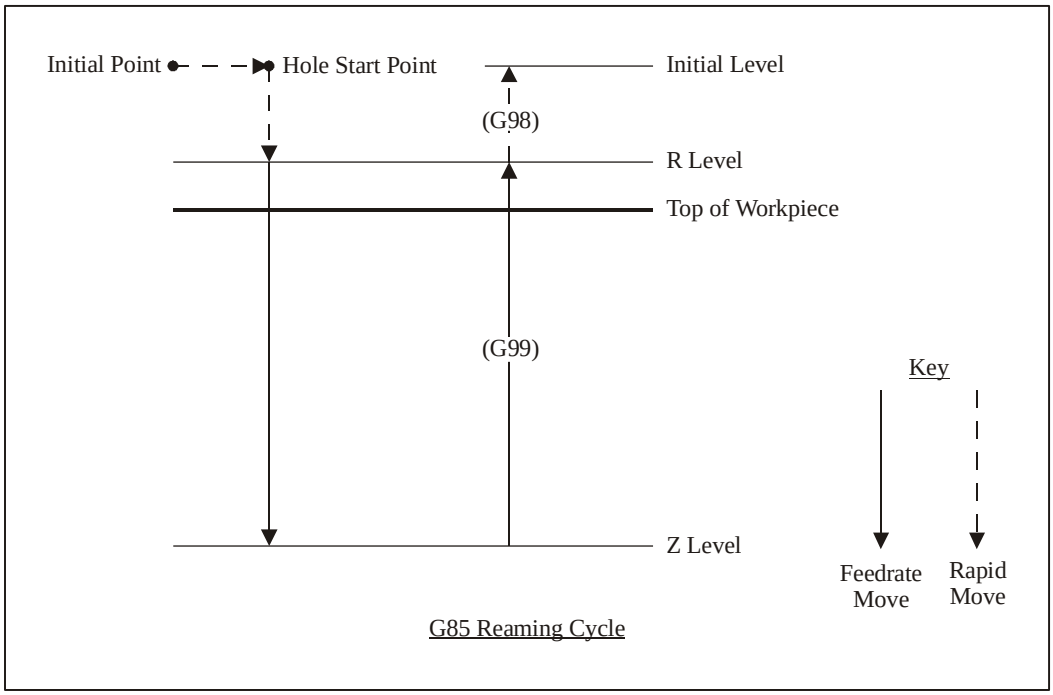
G83 Peck Drilling Cycle

G85 Reaming Cycle

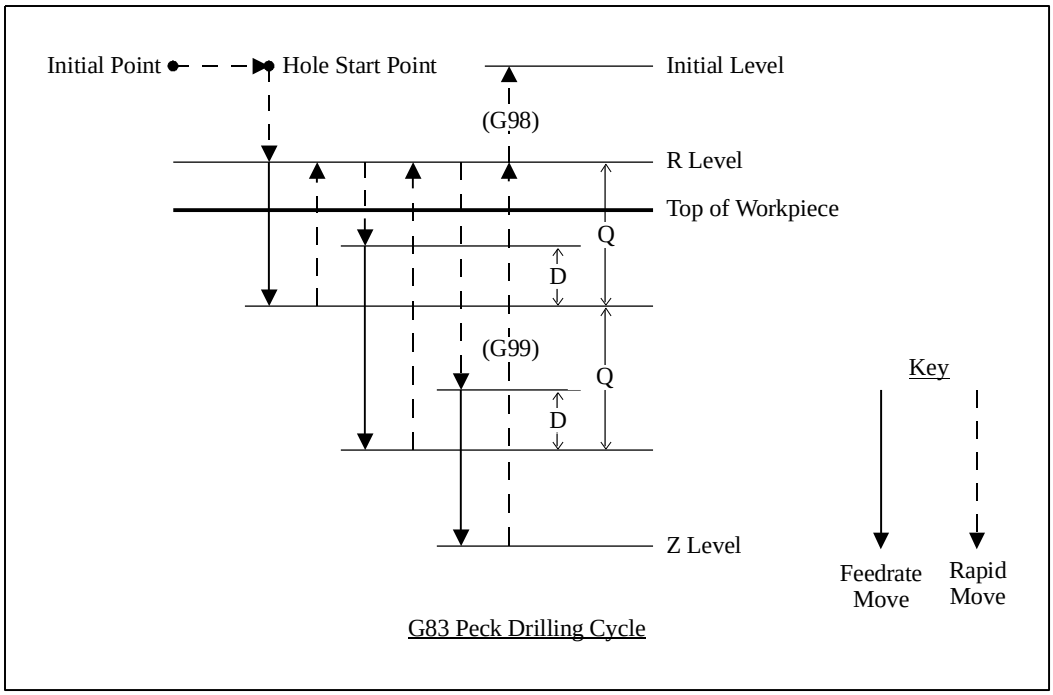
The following diagrams illustrate the sequence of moves for all drilling cycles.



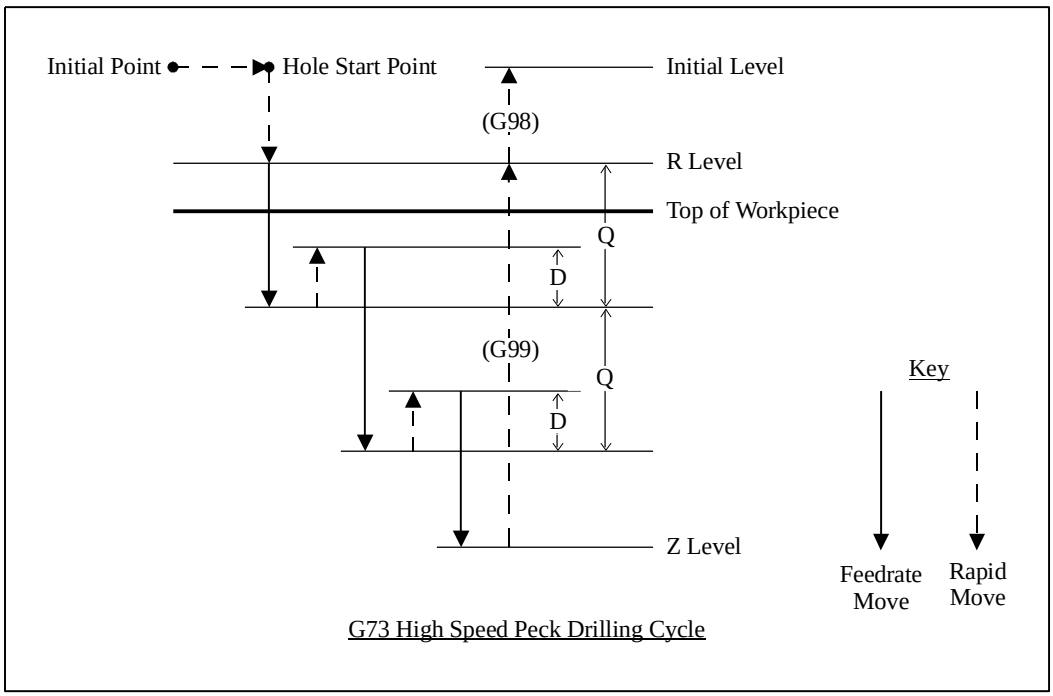
G81 Drilling Cycle / G82 Counterboring Cycle



G85 Reaming Cycle



G83 Peck Drilling Cycle



G73 High Speed Peck Drilling Cycle

The general command syntax for all drilling cycles is as follows:

Gn Xx Yy Rr Zz Qq Pp Ff Ll

where the parameters are

n: Drilling cycle number (73, 81, 82, 83 or 85)

x: X coordinate of Hole Start Point (G90 mode) or distance for rapid move from Initial Point to Hole Start Point (G91 mode)

y: Y coordinate of Hole Start Point (G90 mode) or distance for rapid move from Initial Point to Hole Start Point (G91 mode)

r: Z coordinate of R Level (G90 mode) or distance for rapid move from Hole Start Point to R Level (G91 mode)

z: Z coordinate of Z Level (G90 mode) or distance from R Level to Z Level (G91 mode)

q: Depth of cut for each downward cutting move (G73/G83 only)

p: Dwell time at Z level (milliseconds, G82 only)

f: Feedrate for each downward cutting move (and retract move for G85 only)

l: Number of times to repeat the canned cycle (G91 mode only)

The required parameters are R, Z, F (plus Q for G73/G83, and P for G82).

The D distance is specified in the G73/G83 Retract Distance field on the G-code panel of the Configuration dialog box.

Example (assumes D is 0.050 in.):

```
G98                              Optional

G00 X0.0 Y0.0 Z1.0               Moves the tool to the Initial Point, program
                                 coordinate X=0.0, Y=0.0, Z=1.0 (not required, for
                                 illustration only)

G83 X1.0 Y2.0 R0.1 Z-1.0 Q0.5 F8.0

                                 First moves the tool to the Hole Start Point,
                                 program coordinates X=1.0, Y=2.0, Z=1.0. Then peck
                                 drills a 1.0" deep hole (rapid down to 0.1,
                                 feedrate down to -0.4, rapid up to 0.1, rapid down
                                 to -0.350, feedrate down to -0.9, rapid up to 0.1,
                                 rapid down to -0.850, feedrate down to -1.0, rapid
                                 up to 1.0)
```

When using the drilling cycle commands, there are several things to keep in mind:

- For all drilling canned cycles, the G98 and G99 commands affect the final Z axis rapid move, upward out of the completed hole (rightmost move in each diagram), as follows:

  G98: Z axis moves up to the Initial Level

  G99: Z axis moves up to the R Level

  G98 and G99 compose a mode group, with G98 being the default if neither is specified in the G-code file. You can include either command on the same line as the canned cycle command (before or after the canned cycle command and all parameters).

- G73, G74, G76, G80, G81, G82, G83, G84 and G85 compose a mode group.

- G80 cancels the active canned cycle command. The G00, G01, G02 and G03 commands also cancel the active canned cycle command.

- While a drilling cycle command is active, the command and most of its parameters do not need to be repeated on every G-code line. However, the X and/or Y positioning parameters must be included, and must start the G-code line for the line to be interpreted as a drilling cycle command.

  Example (drills 3 holes in a row along the X axis using the G81 cycle):

  ```
  G81 X1.0 Y2.0 R0.25 Z-0.5 F5
  ```

```
X2.0
X3.0
```

- When you switch to a new command in the mode group, you must include all parameters required for the command.
- The feedrate set by the F parameter for any canned cycle command remains as the current feedrate for subsequent G01, G02 or G03 moves, unless you explicitly set F to another value for the next move.

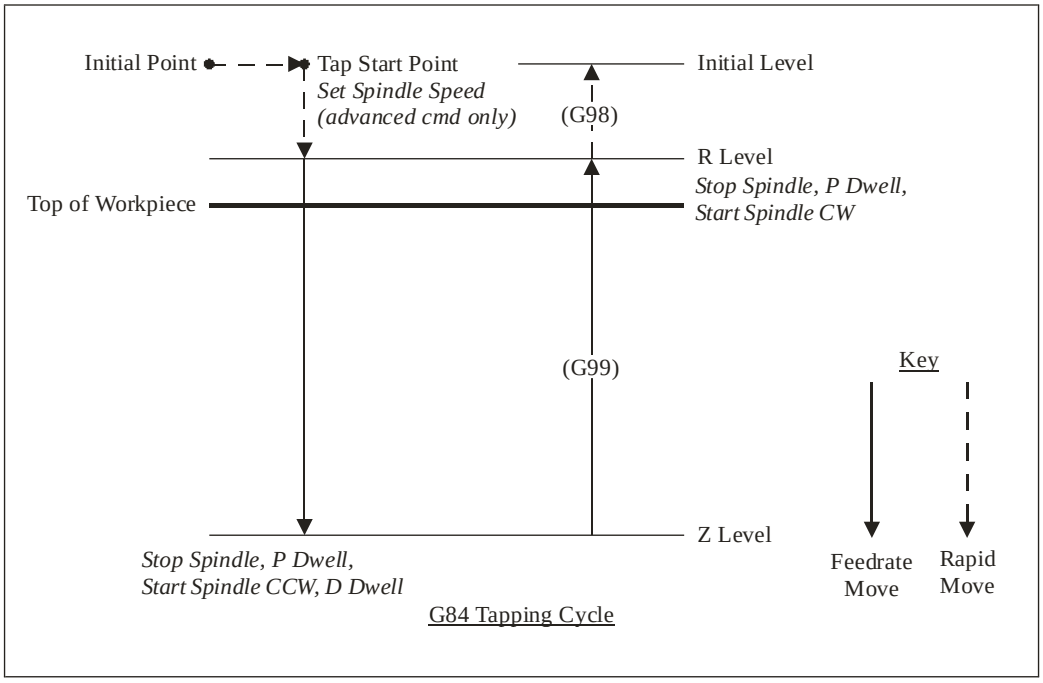
G74, G84 Tapping Canned Cycle Commands

Tapping canned cycles simplify programming by letting you specify standard tapping operations with a single command. FlashCut supports the following tapping cycle commands:

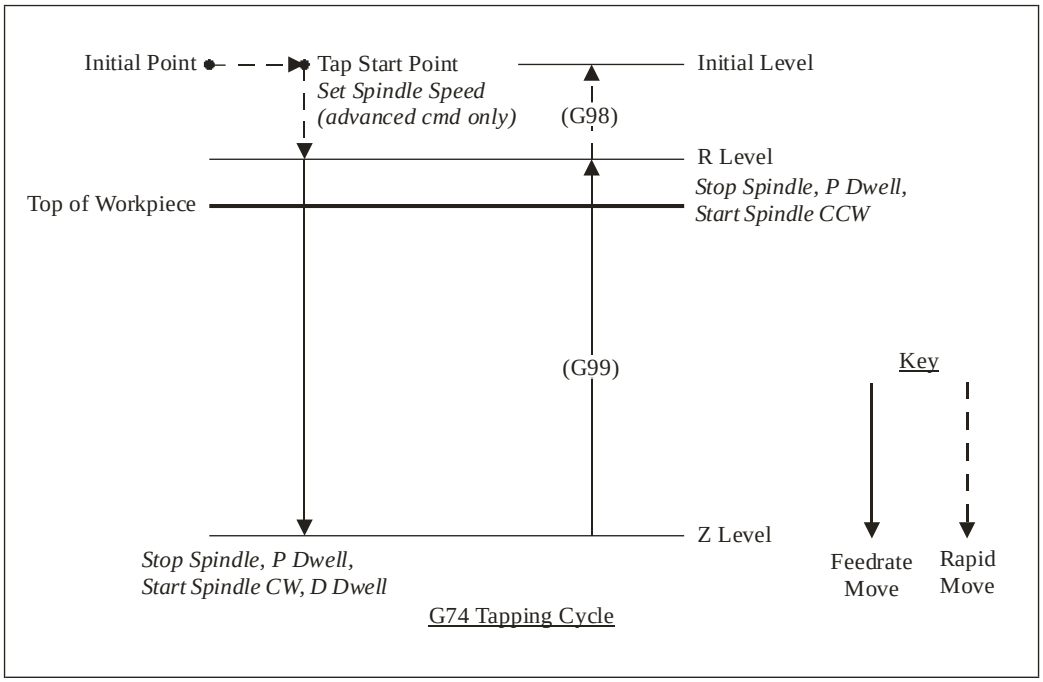
G84 Standard Tapping Cycle

G74 Left-Handed Tapping Cycle

The following diagrams illustrate the sequence of moves for the tapping cycles.



G84 Tapping Cycle



G74 Tapping Cycle

There are two versions of the tapping cycle commands: Advanced and Fanuc-compatible. We recommend using the advanced command as the Fanuc command is provided for compatibility purposes only.

Advanced Tapping Cycle

Gn Xx Yy Rr Zz Qq Ss Ee Pp Dd Ll

where the parameters are

- n: Tapping cycle number (84 for standard tapping or 74 for left-handed tapping)
- x: X coordinate of Tap Start Point (G90 mode) or distance for rapid move from Initial Point to Tap Start Point (G91 mode)
- y: Y coordinate of Tap Start Point (G90 mode) or distance for rapid move from Initial Point to Tap Start Point (G91 mode)
- r: Z coordinate of R Level (G90 mode) or distance for rapid move from Tap Start Point to R Level (G91 mode)
- z: Z coordinate of Z Level (G90 mode) or distance from R Level to Z Level (G91 mode)
- q: Thread pitch
- s: Spindle speed (rpm)
- e: % increase in feedrate for the retract move (eg. E2 means the Z axis will retract 2% faster than it plunged into the hole)
- p: Dwell time after spindle is stopped (milliseconds)
- d: Dwell time after spindle is reversed (milliseconds)
- l: Number of times to repeat the canned cycle (G91 mode only)

The required parameters are R, Z, Q and S.

Fanuc-Compatible Tapping Cycle

Gn Xx Yy Rr Zz Ff Pp Ll

where the parameters are

n: Tapping cycle number (84 for standard tapping or 74 for left-handed tapping)

x: X coordinate of Tap Start Point (G90 mode) or distance for rapid move from Initial Point to Tap Start Point (G91 mode)

y: Y coordinate of Tap Start Point (G90 mode) or distance for rapid move from Initial Point to Tap Start Point (G91 mode)

r: Z coordinate of R Level (G90 mode) or distance for rapid move from Tap Start Point to R Level (G91 mode)

z: Z coordinate of Z Level (G90 mode) or distance from R Level to Z Level (G91 mode)

f: Feedrate (must be calculated based on spindle speed previously set by S command, and desired thread pitch)

p: Dwell time after spindle is stopped (milliseconds)

d: Dwell time after spindle is reversed (milliseconds)

l: Number of times to repeat the canned cycle (G91 mode only)

The required parameters are R, Z, and F.

Before using the tapping cycles you must configure the system for tapping as follows:

1. Enter M-codes for controlling the spindle on the G-code panel of the Configuration dialog box as follows:

- Turn the spindle on clockwise (typically M3)
- Turn the spindle on counterclockwise (typically M4)
- Turn off the spindle (typically M5)

2. If you want to use the Fanuc-compatible tapping cycle, check Use Fanuc-Compatible G74/G84 Tapping Cycles on the G-code panel of the Configuration dialog box.

Warning: The Feed Hold button (and feed hold input lines) are disabled during a tapping cycle, since it is not safe to stop moving the tool while the spindle is still turning. If you need to stop motion during a tapping cycle, turn off power to the spindle and signal generator.

When using the tapping cycle commands, there are several things to keep in mind:

- The spindle must be turned on before a tapping cycle command is executed (clockwise for G84, counterclockwise for G74).
- The G98 and G99 commands affect the final Z axis rapid move as follows:

  G98: Z axis moves up to the Initial Level

  G99: Z axis remains at the R Level (no rapid move)

  G98 and G99 compose a mode group, with G98 being the default if neither is specified in the G-code file. You can include either command on the same line as the canned cycle command (before or after the canned cycle command and all parameters).
- G73, G74, G76, G80, G81, G82, G83, G84 and G85 compose a mode group.

- G80 cancels the active canned cycle command. The G00, G01, G02 and G03 commands also cancel the active canned cycle command.
- While a tapping cycle command is active, the command and most of its parameters do not need to be repeated on every G-code line. However, the X and/or Y positioning parameters must be included, and must start the G-code line for the line to be interpreted as a tapping cycle command.

  Example (taps 3 holes in a row along the X axis using the G84 cycle):

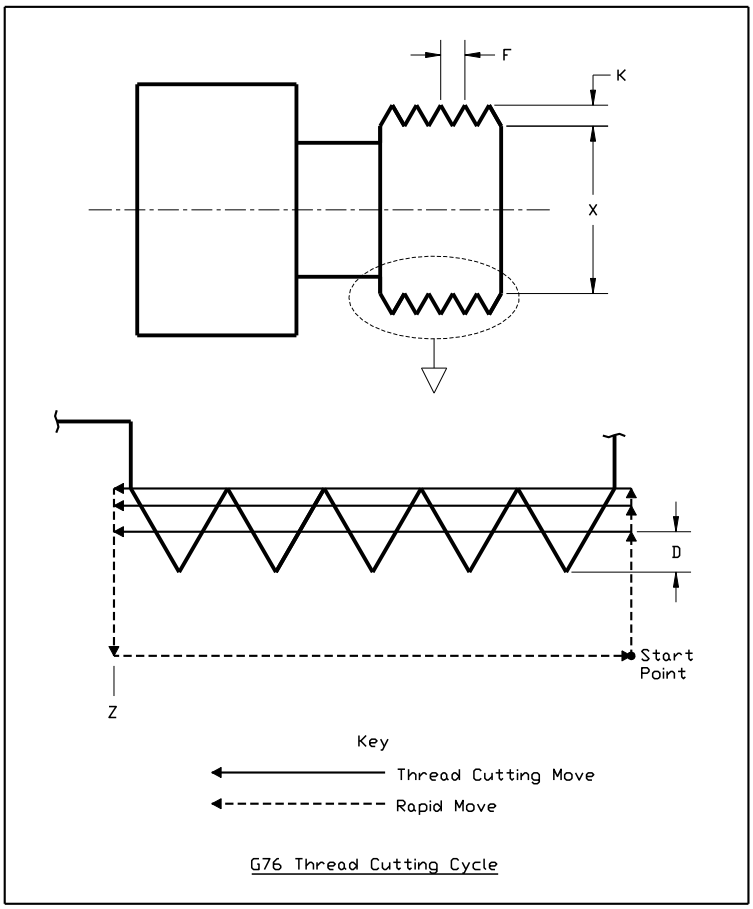
  ```
  G84 X1.0 Y2.0 R0.25 Z-0.5 Q0.05 S200 P500
  X2.0
  X3.0
  ```

- When you switch to a new command in the mode group, you must include all parameters required for the command.
- The feedrate set by the F parameter for any canned cycle command remains as the current feedrate for subsequent G01, G02 or G03 moves, unless you explicitly set F to another value for the next move.

G76 Thread Cutting Canned Cycle Command

The thread cutting canned cycle lets you cut threads on a lathe. Your lathe must be equipped with a spindle encoder that is plugged into the I/O Expansion Board.

The following diagram illustrates the sequence of moves for the thread cutting cycle. The rest of this section refers to this diagram.



The general command syntax is as follows:

G76 Xx Zz Kk Dd Ff [Aa] [Ii] [Pp]

where the parameters are

x: Final diameter (minor diameter for external threads, major diameter for internal threads)

z: Z Coordinate at end of thread

k: Height of thread

d: Depth of first pass specified as an integral number of ten-thousandths of an inch or mm, e.g. ‘2000’ means 0.2000 inches or mm (depending on the units you’re using)

f: Thread pitch

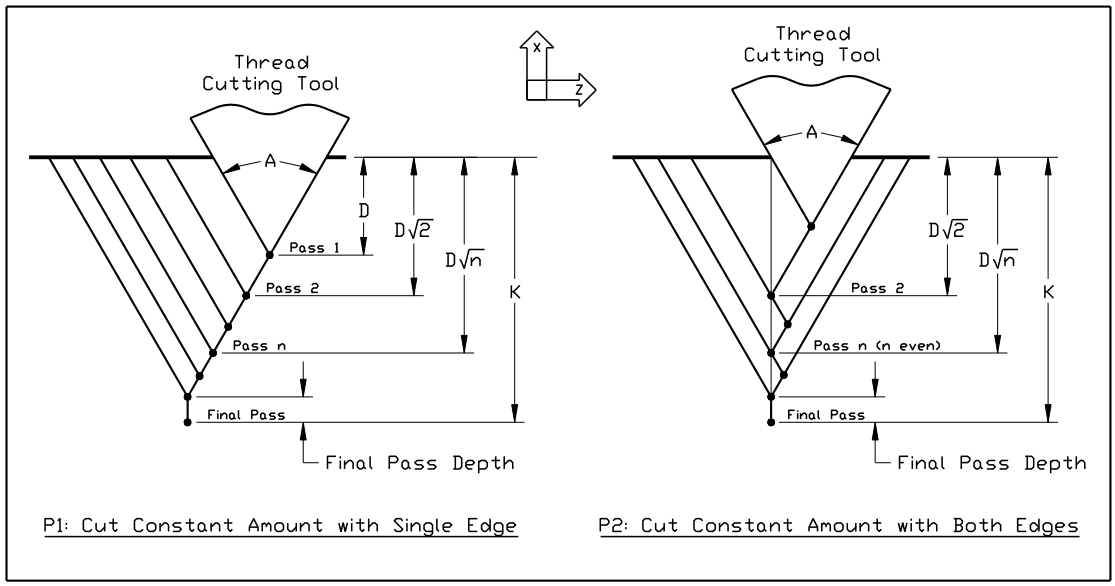
a: (Optional) Tool angle (determines lead-in angle, default is zero)

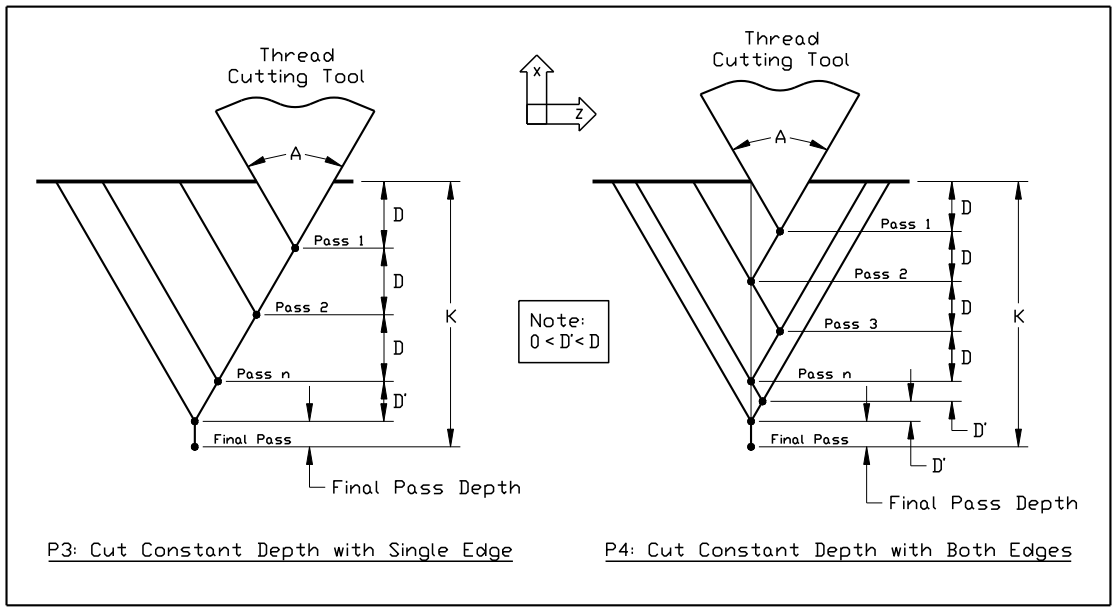
i: (Optional) Amount of taper (difference between initial radius and ending radius, default is zero)

p: (Optional) Cutting method (values 1-4, sets strategy for successive plunges, default is '1')

Note that in the diagram above, the 'A' and 'I' parameters are zero, and the cutting method is '1' (all default values).

Cutting a thread generally requires several passes. The 'P' parameter controls the depth, starting position and plunge angle of each pass, as shown in the following diagrams. For all four cutting methods shown in the diagrams below, the 'A' parameter is shown as nonzero to illustrate its affect. When 'A' is zero, the plunges are straight into the workpiece, along the X axis, as shown in the diagram above.





The settings on the Threading panel of the Configuration dialog box also affect the cutting passes as follows.

Minimum Depth of Cut – The incremental depth of any pass will be greater than or equal to this value (except for the final pass and any spring passes).

Final Pass Depth of Cut – Incremental depth of the final pass (finishing pass).

Spring Passes – Number of additional passes performed at the same depth as the final pass.

Warning: The Feed Hold button (and feed hold input lines) are disabled during a threading pass, since it is not safe to stop moving the tool while the part is still turning. If you need to stop motion during a cutting pass, turn off power to the spindle and signal generator.

Example (assumes Minimum Depth of Cut is 0.010, Final Pass Depth of Cut is 0.005, Spring Passes is set to 1):

```
G00 X0.75                      Move to the Start Point
G76 X1.0 Z-2 K0.125 D500 F0.2
                               Performs a total of 8 threading passes (including
                               1 spring pass) to make a two inch long thread with
                               minor diameter 1.0, major diameter 1.25, at 5
                               turns per inch. By default the cutting method is
                               P1, the tool angle is 0, and there is no taper.
                               The first pass is cut at a depth of 0.05.
```

Note that the F parameter does not set the feedrate for subsequent non-threading moves, since its meaning is different from the standard F command.

G90 Absolute Positioning Mode

The G90 command puts the system into absolute positioning mode. All XYZABC coordinates are treated as points relative to Program Zero (or a local coordinate system set by the G52 command). This command stays in effect until a G91 command occurs.

Note that absolute positioning is the default positioning mode for FlashCut. It is not necessary to include this command in your G-code file if all your moves are absolute.

G91 Incremental Positioning Mode

The G91 command puts the system into incremental positioning mode. All XYZABC coordinates are treated as incremental move distances. This command stays in effect until a G90 command occurs.

Example:

```
G01 X1.0 Y3.0 Z-1.5 F12        Moves the tool to program coordinates X=1.0,
                               Y=3.0, Z=-1.5 (G90 is assumed)
G91                            All XYZA coordinates after this command will be
                               interpreted as incremental distances
G01 X1.0 Y2.0 Z-0.5            Moves the tool a distance of X=1.0, Y=2.0, Z=-0.5
                               from the current tool location (this corresponds
                               to the program coordinates X=2.0, Y=5.0 Z=-2.0)
G02 X1.0 Y-1.0 I0.5 J-0.5      Moves the tool using counter-clockwise circular
                               interpolation to the program coordinates X=3.0,
                               Y=4.0, Z=-2.0 with a center point at program
                               coordinates X=2.5, Y=4.5, Z=-2.0
G90                            All XYZA coordinates after this command will be
                               interpreted as program coordinates
```

```
G01 X1.0 Y2.0 Z-0.5          Moves the tool to program coordinates X=1.0,
                             Y=2.0, Z=-0.5
```

G120 Sense Tool Length

G120 runs the tool length sensing routine, the same as if you clicked the Length button in the Tool Box. There are no parameters.

Example:

```
G120                         Senses the length of the current tool and stores
                             the result in the Tool Library
```

See “Tool Box” in the Main Screen Features section for more information.

G140, G141 Engraving

FlashCut CNC supports engraving with the G140 and G141 commands.

G140 is used to select the font. The syntax is:

G140 "font" Hh [Kk] [Ss]

where the parameters are

- font: The name of the font. Currently one font “SLF1” is built into FlashCut. FlashCut lets you create your own plug-in fonts. Please contact technical support for more information.
- h: The height of the font
- k: (Optional) The kerning, or spacing, between characters in the text to engrave (default is 1/10th of the height)
- s: (Optional) The character width style to use (0 = fixed, 1 = variable, default is variable)

Example:

```
G140 "slf1" H1.5 K0.1 S1
```

The font specified in the G140 command must have been loaded by FlashCut CNC upon startup or an error will result.

The height parameter specifies the height of the tallest character in the font. All other characters are scaled as appropriate, and the width of each character is scaled so that the proper aspect ratio of the character is maintained.

Fixed character width is determined by the widest character in the font. For fixed character widths, each character is centered in a cell that is the width of the widest character. The kerning value, then, is the space between each cell. For variable character width, the cell for each character is as wide as the character itself.

The G140 command selects the font for all engraving (G141) commands that follow. The first G141 command in a G-code file must be preceded by a G140 command or an error will result.

The G141 command specifies the desired text to engrave. The syntax is:

G141 "text" ["text2"] [Xx] [Yy] [Zz] [Ww] Ff Pp Rr [Dd]

where the parameters are

- text: The text to engrave

text2: (Optional) The text to engrave with a second spindle (see below)

x, y: (Optional) The XY position of the lower left corner of the text (program coordinates). If you leave out these parameters FlashCut will engrave the text at the current XY location. In G91 mode these parameters represent the offset from the current XY location to the lower left corner of the text.

z: (Z or W is required) The depth of the engraving by the Z spindle (Z program coordinate)

w: (Z or W is required) The depth of the engraving by the W spindle (W program coordinate)

f: The feedrate at which to engrave

p: The feedrate at which to plunge the tool during engraving

r: The height to which the tool is retracted between cuts (Z/W program coordinate)

d: (Optional) The direction (or rotation) of the engraving, in degrees (rotated about the point specified as the lower left corner, default is zero)

Example:

```
G141 "Happy Birthday" X1.1 Y0.5 Z-0.2 F10 P2 R0.1 D45
```

The text will be engraved in the font selected by the most recent G140 command. If no G140 command was specified, an error will result.

FlashCut supports engraving with up to 2 spindles simultaneously. The supported axes for the two spindles (for engraving purposes) are Z and W. For multi-spindle engraving, the following rules apply:

- Either Z or W can be specified, or both
- Both depths, if specified, must be the same. In other words, Z and W must be equal.
- The plunge feedrate applies to both axes, if both Z and W are specified.
- As long as the characters in text and text2 are the same, engraving will occur simultaneously. Once the first difference is encountered, each spindle will engrave the remaining characters separately (see example below).
- If only one spindle (Z or W) is specified in the G141 command and two strings are specified, an error will result.
- If both spindles are specified and only one string is specified, the string will be engraved simultaneously by both spindles.

Example:

```
G141 "Serial 12356" "Serial 12456" Z-0.2 W-0.2 F10 P2 R0.1
```

In this case, “Serial 12” will be engraved simultaneously, “356” will be engraved by the Z spindle, and “456” will be engraved by the W spindle.

G180, G181 Safe Envelope Commands

Use of the Safe Envelope can be controlled from G-code by using the G180 and G181 commands. G180 turns off safe envelope restrictions, and G181 turns safe envelope restrictions on. These commands allow the G-code to temporarily deactivate safe envelope restrictions when necessary.

Neither command has parameters. When FlashCut encounters G180 in the G-code, safe envelope restrictions are ignored and normal machine envelope restrictions apply. When FlashCut encounters G181, the motion of the machine is bounded by the safe envelope (defined on the Advanced Definition panel of the Configuration dialog box.)

For more information, see "Coordinates Menu" in the Main Screen Features section. Note that the menu commands take priority over G180 and G181.

G600, G601 Automatic Lift Axis Cutting

These commands only apply to fabrication heads with automatic lift axis control, such as plasma and laser.

The G601 command is required to initiate plasma and laser cutting, including the move from Pierce Height to Cut Height.

If Height Control is enabled, this command also initiates the automatic height control processing.

The G600 command is required to end plasma and laser cutting before a subsequent rapid move.

These commands are typically configured in the Start of Cut and End of Cut automatic macros for the fabrication head.

G602, G603 Suspend & Resume Automatic Height Control

These commands only apply to plasma and laser fabrication heads.

The G602 command temporarily disables automatic height control processing.

The G603 command enables automatic height control processing.

G605 Move to Initial Height

The G605 command moves the Lift Axis to the initial height before cutting begins. This command is typically configured in the Start of Cut automatic macro for the fabrication head.

This command only applies to plasma, laser, and waterjet fabrication heads.

Plasma and Laser situations:

If touch-off is enabled, the machine touches off on the sheet and retracts to the pierce height.

If touch-off is not enabled, the machine moves directly to the pierce height.

Waterjet situations:

If touch-off is enabled, the machine touches off on the sheet and retracts to the cut height.

If touch-off is not enabled, the machine moves directly to the cut height.

G607 Move to Cut Height

The G607 command moves the Lift Axis to the cut height.

G610 Create Dimple

This command only applies to fabrication heads that support dimpling.

The G610 command executes the dimple macro defined in the fabrication head configuration, then retracts the Lift Axis to the dimple safe height.

G611 Pierce

This command only applies to fabrication heads that support piercing.

The G611 command executes the pierce routine as defined for the selected fabrication head.

G612 Laser Pierce

This command only applies to the laser fabrication heads.

The G612 command executes the pierce routine as defined for the selected fabrication head.

G630, G631 Suspend & Resume Automatic Lift Axis Moves

The G631 command temporarily disables automatic moves of the Lift Axis. This allows for the processing of G-Code commands that include motion using the Lift Axis.

The G630 command enables automatic moves of the Lift Axis, such as moves to the Safe Height.

M00 Program Pause

The M00 command pauses processing of the G-code program. The syntax is:

M00 Pp “message”

where the parameters are

p: (Optional) Display message to operator (0 = no message, 1 = display message)

message (Optional) Specific message to the operator (replaces standard system message).

If you don’t include the P parameter, FlashCut uses the Message on M00 Program Pause checkbox, on the G-code panel of the Configuration dialog box, to determine whether or not to display a message. The P parameter overrides the configuration setting.

Example:

```
M00 "Place a new blank in the fixture"
```

M01 Optional Program Pause

The M01 command is identical to the M00 command, except that it may be deactivated when desired. To control activation of the M01 command, use the Execute M01 checkbox on the G-code panel of the Configuration dialog box. When the checkbox is unchecked, FlashCut ignores the command.

M30, M30.1 End of Program

The M30 command ends processing of the G-code program and automatically resets the program to the first executable line. The M30.1 command is similar to M30 except it also restarts the G-code program. This allows you to run a program over and over indefinitely.

M98, M99, M02 Subroutine Commands

Subroutines allow you to eliminate repetitive programming. FlashCut supports the use of subroutines with the M98, M99, and M02 (or M30) commands. Use of these commands is best explained through a simple example. The following G-code program uses one subroutine called "mysub":

Example:

```
G01 X1 Y1 F10                First line of main program
M98 Pmysub                   Jump to subroutine "mysub"
G01 X0 Y0                    Continued execution after "mysub" ends
M02                          End of main program
Omysub                       First line of the subroutine called "mysub"
G01 X2 Y2                    Continued execution within the subroutine
M99                          End of subroutine "mysub"
```

In the main program, the M98 command causes program execution to jump to the first line of the subroutine named "mysub". Notice that the letter "P" must immediately precede the name of the subroutine with no spaces.

The subroutine definition begins with the letter "O" followed immediately by the subroutine name with no spaces. The subroutine must end with the M99 command as shown. M99 causes program execution to jump back to the main program, continuing with the line immediately following the M98 line (G01 X0 Y0 above).

When subroutines are used, the main program must end with M02 or M30, the "End of Program" commands. Each subroutine must have a unique name, consisting of any number of alpha-numeric characters. You can "nest" subroutines as much as you like, meaning one subroutine may call another subroutine, which in turn may call another subroutine, and so on.

To call the same subroutine multiple times, include the L parameter as shown below.

Example:

```
M98 Pmysub L10               Execute subroutine "mysub" 10 times
```

This feature provides a simple means to repetitively loop through any block of G-code. Just place the G-code in a subroutine, then call the subroutine with the L parameter set to the number of loops required.

It's sometimes convenient to place subroutines in files separate from the main G-code file. One use of this feature is to create a library of reusable subroutines that you can apply to various projects and jobs. You'll call the subroutines using the same M98 syntax used for subroutines listed in the main program file.

FlashCut recognizes two types of subroutine files: 'single subroutine' files and 'included' files.

- Single Subroutine Files

  You may put a single subroutine in a file and name the file as follows:

  O[subroutine name].[extension]

  The extension must match the main program file's extension, and the file must be located in the same folder as the main program file. Since the file solely consists of the subroutine, you do not need to include the subroutine definition line (starting with 'O') or the M99 line marking the end of the subroutine.

  Using the example above, the subroutine 'mysub' could be stored in a separate file as follows:

  File Name: Omysub.fgc

  File Contents:

  ```
  G01 X2 Y2
  ```

- Included Files

  You may put any number of subroutines in an included file. Each included file must be listed at the top of the main G-code file using the following syntax:

  #[file path and name]

  Once a file has been included in the main G-code file, FlashCut treats the included file's subroutines the same as if they were listed at the bottom of the main G-code file. If you leave out the path, FlashCut will look for the file in the same folder as the main G-code file.

  Example:

  Main G-code File (Engraves "A B")

  ```
  #C:\GCODE\ALPHABET.FGC
  G00 X0 Y0
  M98 PmakeA
  G00 X1 Y0
  M98 PmakeB
  M02
  ```

  ALPHABET.FGC File

  ```
  OmakeA
  G91
  [Incremental G-code to cut the letter "A"]
  G90
  M99
  OmakeB
  ```

```
G91
[Incremental G-code to cut the letter "B"]
G90
M99
```

M100, M101 Wait for Input Line

The M100 and M101 commands wait for an input line to reach a desired state as follows:

M100 Wait for normal state

M101 Wait for tripped state

The Wiring setting on the Input Lines panel of the Configuration dialog box affects the behavior of these commands as follows:

Wiring Setting	M100 Waits For Input Line to be...	M101 Waits For Input Line to be...
Normally Closed	Closed	Open
Normally Open	Open	Closed

The command syntax is as follows:

M100 Ix "[optional custom message to operator]"

M101 Ix "[optional custom message to operator]"

where x is the number of the input line to monitor.

Examples:

```
M100 I2                      System waits for input line 2 to reach the normal
                             state, displays standard message if timeout is
                             reached
M101 I7 "Low air pressure"   System waits for input line 7 to become tripped,
                             displays customized message if timeout is reached
```

You must define the specified input line as "Control" in the Switch Function pull-down menu, on the Input Lines panel of the Configuration dialog box (see "Input Line Settings" in the Initial Setup section for more details).

FlashCut uses debounce for these commands. For a switch to successfully reach a desired state, it must hold that state continuously for the debounce duration. M100 and M101 use the Debounce when Idle time on the Input Lines panel of the Configuration dialog box.

The M100 and M101 commands time out once the maximum allowable time has passed and the input line has not reached the desired state. When the commands time out, FlashCut halts processing of the G-code program and displays a message to the operator. You can set the M100/M101 Timeout duration on the G-code panel of the Configuration dialog box.

A typical use for these commands is detecting completed motion of an auxiliary device, such as an indexer or air cylinder.

M102 Enable Feed Hold Input Line

This command causes the system to treat an input line as a feed hold input line. The syntax is:

M102 I[input line] S[trigger state]

where the trigger state is 0 or 1:

0 to trigger feed hold on untripped state

1 to trigger feed hold on tripped state

M103 Disable Feed Hold Input Line

This command cancels the effect of M102 for the specified input line.

The syntax is:

M103 I[input line]

M106 Select Fabrication Head

The M106 command selects the specified fabrication head. The syntax is:

M106 H[fabrication head id]

M03, M05, M07, M08, M09, M50, M51, MXX Auxiliary Device Control

Using the M-code Definitions panel of the Configuration dialog box you can define M-codes to turn on or off various devices via the output lines. You can also define M-codes to digitally control external devices (using a group of output lines as digital input into the control lines of the device). See “M-code Definitions” in the Initial Setup section for details on how to set up the M codes.

When FlashCut processes an M-code that sets output lines, if the current output line state already reflects the M-code’s intended action, FlashCut skips the M-code, including any associated delay. For example, if the spindle is on, and FlashCut processes an M-code to turn on the spindle, the delay associated with turning on the spindle (e.g. 2 seconds) is skipped.

Typical M codes include:

M03 Spindle On

M05 Spindle Off

M07 Mist Coolant On

M08 Flood Coolant On

M09 Coolant Off

M50 Plasma On

M51 Plasma Off

F, G93, G94 Feedrate Commands

The F command specifies the feedrate. The feedrate is modal, which means it stays in effect until another F command occurs.

For all linear moves, specify the feedrate in inches/minute for English units and millimeters/minute for Metric units.

Example:

```
G01 X4.0 Y3.0 Z1.0 F7.0        Moves the tool to program coordinates X=4.0,
                               Y=3.0, Z=1.0 at a feedrate of 7.0 in/min
```

Please note the following cases:

- For any move that includes a rotary axis, specify the feedrate either as a linear feedrate (length/minute) or rotary feedrate (degrees/minute), depending on your setting for 'F Command Interpretation for Rotary Moves' on the Rotary Axes panel of the Configuration dialog box. For more information see the detailed explanation of rotary feedrates under "G01 Linear Interpolated Feedrate Move" above.
- FlashCut supports two feedrate modes: standard feedrate (G94) and inverse time feedrate (G93). The explanation above is for G94 mode, which is the predominant mode in CNC machining and the default in FlashCut. In G93 mode, FlashCut interprets all feedrate (F) commands as inverse time feedrates. The definition of the F command is

  F = 1 / Time to complete the move in minutes

  When using inverse time feedrates, there must be an F command at the end of every line containing a G01, G02 or G03 command. G93 is used primarily for simultaneous 4 and 5 axis machining.

  Example:

```
G93
G01 X1 Y1 Z-1 A90 F10      Moves the tool to program coordinates X=1, Y=1,
                           Z=-1, A=90 in 6 seconds (0.1 minutes)
```

S Set Variable Output Port

The S command sets an analog, PWM or PFM output port signal to correspond to the supplied value. Common uses include setting the speed for a spindle or the power for a laser.

The S command affects the current variable output port (1 or 2). The current port is selected by:

- Configuration settings on the Variable Output panel of the Configuration dialog box.
- The G210 command (see "G210 Select Variable Output Port" above).

Example:

```
S1000                      On a mill or lathe, sets the spindle speed to 1000
                           rpm
```

See "Variable Output Settings" in the Initial Setup section for more details, including how FlashCut converts the desired S value to an output signal.

Program Comments

You can add comments to your program by enclosing them in parentheses. FlashCut ignores anything enclosed in parentheses as shown below.

Example:

```
(Move to beginning of the next feature)
G00 X1.0 Y3.0 (Ready to move Z axis down)
G00 Z-1.5
(Begin next feature)
G01 Z-1.6 F8
G01 X3.0 Y7.5
```

Optional Line

The optional line command is a forward slash ('/') at the start of a G-code line. FlashCut only executes the line if the Execute Optional G-code Lines checkbox is checked, on the G-code panel of the Configuration dialog box. If the checkbox is unchecked, FlashCut ignores the line.

Example:

```
G00 X2 Y4
/G00 Z1                        This line is not always executed
G00 X3 Y5
```

BEGINGRID, ENDGRID

You can generate a grid of parts using G-code for a single part and the BEGINGRID and ENDGRID commands.

All G-code that appears between the BEGINGRID and ENDGRID commands will be repeated as appropriate to produce a grid of parts. The syntax for the BEGINGRID command is:

BEGINGRID Cols Rows X Y IncrX IncrY ByRow ZigZag

where the parameters are

Cols: The number of columns in the grid

Rows: The number of rows in the grid

X: The X program coordinate of the lower left corner of the first part

Y: The Y program coordinate of the lower left corner of the first part

IncrX: The grid spacing in the X direction for each column of the grid

IncrY: The grid spacing in the Y direction for each row in the grid

ByRow: Indicates whether the grid will be formed column-by-column (0) or row-by-row (1)

ZigZag: Indicates whether the grid will start each new row/column at the first row/column (0) or zigzag through the grid (1)

Example:

```
BEGINGRID 3 5 1.2 1.4 0.3 0.7 0 1      Produces a grid that is 3 columns by 5
                                       rows, the first element in the grid is at program
                                       coordinate X=1.2, Y=1.4, columns in the grid are
                                       offset 0.3 and rows are offset 0.7, the grid is
                                       formed column-by-column and in a zigzag pattern
```

The grid must end with an ENDGRID command. Then ENDGRID command takes no parameters.

Example:

```
G00 X0 Y0 Z0.1
F10
BEGINGRID 3 4 1 1.5 2.0 2.5 0 1
G00 X0.1 Y0.1                  Move to the start of the square
G01 Z-0.1                      Plunge the tool
G01 X0.9 Y0.1                  Cut the square
G01 X0.9 Y0.9
G01 X0.1 Y0.9
G01 X0.1 Y0.1
G01 Z0.1                       Raise the tool
ENDGRID
G00 X0 Y0 Z0.1
```

This is a 3 x 4 grid of squares. The lower left of the first square in the grid is at (1.1, 1.6). This is because the starting position of the square (0.1, 0.1) is added to the starting position of the grid (1.0, 1.5). The next square in the grid will start at (1.1, 4.1) because the grid is being created column by column. FlashCut builds each column completely before moving onto the next column. Once all four rows are built in the first column, with squares staring at (1.1, 1.6), (1.1, 4.1), (1.1, 6.6), and (1.1, 9.1), the next square will start at (3.1, 9.1) on the 4th row of the 2nd column. This is because the grid is being built in a zigzag pattern. If it were not in a zigzag pattern, the next column would have been started at (3.1, 1.6), which is the 1st row of the 2nd column.

Advanced Programming Reference

FlashCut provides a number of advanced programming features, allowing you to create more flexible and powerful G-code programs. This section presents these topics:

Values

In every G-code example up to this point, all values were **literal** values. A literal value is one that's typed into the G-code program and cannot change. Examples of literal values are:

```
1
2.33
"Please close the door"
```

An example of these values used in G-code is:

```
T1
G00 X2.33
M00 "Please close the door"
```

Variables

In addition to literal values, the FlashCut G-code interpreter understands variables.

A variable is a placeholder for a value. There are two categories of variables in FlashCut: System, Program and User. System variables are defined and set by the FlashCut software itself. These variables represent status values in the software, and therefore allow you to access information about the current state of the program. Most system variables cannot be modified in a G-code program. However, all system variables can be read in a G-code program. The system variables currently supported are:

- #FabHead.CurrTool – This variable represents the number of the tool currently loaded. Its value is 0 if no tool is loaded. This value can be modified in a G-code program. It is typically modified in a tool change macro.
- #FabHead.NextTool – This variable represents the number of the tool that will be loaded on the next tool change command (M06). Its value is 0 if no tool has been specified with the T command. This value can be modified in a G-code program.
- #CurrFeedrate – This variable represents the current feedrate. This value can be modified in a G-code program.
- #TlChgFeedrate – This variable represents the feedrate at which the controlled rate portions of an automatic tool change will occur.
- #FabHead.TlChgCheckPower – This variable represents the state of the Monitor Tool Chuck Power checkbox on the Tool Change panel of the Configuration dialog box (checked = true).
- #FabHead.TlChgPowerInputLine – This variable represents the setting of the Input Line text box on the Tool Change panel of the Configuration dialog box.
- #FabHead.TlChgCheckPowerInputNormal – This variable represents the setting of the Power On State pull-down menu on the Tool Change panel of the Configuration dialog box (Normal = true).
- #Tool – This is an array variable that is used to access properties of any configured tool.
- #RefPoint – This is an array variable that is used to access properties of any defined reference point.
- #FixOff – This is an array variable that is used to access properties of any defined fixture offset.
- #TlRackPos – This is an array variable that is used to access the properties of any tool rack position.
- #Machine – This variable represents the current position of the machine tool in machine coordinates.
- #Program – This variable represents the current position of the machine tool in program coordinates.
- #ToolOff – This variable represents the tool offset currently in effect.
- #Input – This variable represents the current state of the input lines.
- #Output – This variable represents the current state of the output lines.

#Tool, #RefPoint, #FixOff, #TlRackPos, #Input and #Output are array variables. Array variables are discussed later in this section.

#Machine, #Program, #ToolOff, #Input, and #Output are runtime variables. Runtime variables are discussed later in this section.

In addition to the above variables, there are several general-purpose system variables available that do not represent any system status values. These variables have been provided for use in macros. You are free to set and read these variables as desired.

- #SysInteger1 - #SysInteger8 – Eight system-declared INTEGER variables.
- #SysReal1 - #SysReal8 – Eight system-declared REAL variables.
- #SysBoolean1 - #SysBoolean8 – Eight system-declared BOOLEAN variables.
- #SysString1 - #SysString8 – Eight system-declared STRING variables.

Program variables are defined and maintained by the G-code programmer. In order to use a Program variable, you must first declare it with a simple statement that includes the type of the variable and its name, as in this example:

```
GLOBAL INTEGER #Count
```

In the above statement, the word "GLOBAL" indicates the scope of the variable (more on that later), "INTEGER" is the type, and "#Count" is the variable name. The types supported by FlashCut are:

- INTEGER – These are whole numbers (i.e. numbers without decimal points). They can be positive or negative (or zero).
- REAL – These are rational (or fractional) numbers (i.e. numbers with decimal points). They can be positive or negative (or zero).
- BOOLEAN – Variables of this type can have one of two values: TRUE or FALSE.
- STRING – Variables of this type represent text.

Example declarations:

```
GLOBAL INTEGER #Index
GLOBAL REAL #X
GLOBAL BOOLEAN #IsDone
GLOBAL STRING #Message
```

As with all other words in a FlashCut G-code program, the case of the letters does not matter. In other words, these two declarations are the same:

```
GLOBAL REAL #X
Global REAL #x
```

If these two lines appeared in the same G-code file, an error would be reported because the same variable is being declared twice.

Variable names can be any sequence of letters or numbers or the underscore character (_), but must always start with a number sign (#). The following are all legal variable names:

```
#X
#Index
#3rdAxis
#Is_Done
```

The word "GLOBAL" in a variable declaration indicates the scope, or context, of the variable. A variable with GLOBAL scope can have its value changed or retrieved anywhere in the G-code program, even in a file that is included into the program. GLOBAL variables must be declared before any line of G-code other than lines that include files using #.

If the word GLOBAL is omitted from the variable declaration, that variable is considered to have local scope. This means that the variable's value can only be changed or retrieved in the same subroutine in which it was declared.

Currently, FlashCut only supports GLOBAL variables, so the word "GLOBAL" is optional. Future releases of FlashCut will support local variables. In these releases, the word GLOBAL will be required to indicate variables with global scope.

One or more variables can be declared on the same line. For example,

```
REAL #X #Y #Z
```

declares three variables, #X, #Y, and #Z, which are all of type REAL. When declaring more than one variable on the same line, the variable names must be separated by spaces.

The value of a variable can be set by using the assignment operator. The assignment operator is an equals sign (=). Variables can be assigned literal values, as shown below:

```
INTEGER #Count
#Count = 0
```

The above code declares an integer variable called #Count, and then assigns #Count the literal value of 0. The process is similar for all variable types.

```
REAL #X #Y
BOOLEAN #IsDone
STRING #Message
#X = 1.0
#Y = 2.1
#IsDone = FALSE
#Message = "Please make sure the enclosure is closed."
```

A variable can be used in place of a literal value in a G-code file. For instance,

```
REAL #X #Y
#X = 1.0
#Y = 2.1
G00 X#X Y#Y
```

will cause a rapid move to X = 1.0, Y = 2.1. Here are more examples of variable use:

```
INTEGER #ToolNumber
STRING #Message
#ToolNumber = 1
M06 T#ToolNumber
#Message = "Please make sure the enclosure is closed."
M00 #Message
```

Some variables types will be automatically converted to the correct type as necessary. For instance, motion commands take REAL type parameters, but INTEGER type parameters will be automatically converted. Example:

```
INTEGER #X #Y
#X = 1
#Y = 2
G00 X#X Y#Y
```

In this case, #X and #Y will be converted automatically to REAL values. Similarly, REAL values will be automatically converted to INTEGER values. In this case, the fractional part of the number is lost.

```
REAL #ToolNumber
#ToolNumber = 2.7
M06 T#ToolNumber
```

The above code will perform a tool change to tool number 2. It is important to note that when a REAL value is converted to an INTEGER value it is **not** rounded. In other words, 2.001 converted to an integer is 2, and 2.999 converted to an integer is 2.

Any type of a variable can be converted to a STRING. This means that the following lines of code are all legal:

```
STRING #Message
REAL #X
INTEGER #Count
BOOLEAN #IsDone
#Message = "Test message"
#X = 1.0
#Count = 0
#IsDone = FALSE
M00 #Message
M00 #X
M00 #Count
M00 #IsDone
```

Variables can be assigned to each other as well.

```
REAL #X1 #X2
INTEGER #ToolNumber
#X1 = 3.14                    Value of #X1 becomes 3.14
#X2 = #X1                     Value of #X2 becomes 3.14
#ToolNumber = #FabHead.CurrTool  Value of #ToolNumber becomes the currently
                              loaded tool (note that #FabHead.CurrTool is a
                              system variable)
```

Some system variables represent objects, rather than simple types. The system variables in this category are #Machine, #Program, and #ToolOff. Here is a description of the properties of each:

- Machine
  - X – the current position on the X axis in machine coordinates
  - Y – the current position on the Y axis in machine coordinates
  - Z – the current position on the Z axis in machine coordinates
  - A (W) – the current position on the A (or W) axis in machine coordinates
- Program
  - X – the current position on the X axis in program coordinates

  - Y – the current position on the Y axis in program coordinates
  - Z – the current position on the Z axis in program coordinates
  - A (W) – the current position on the A (or W) axis in program coordinates
- Program Zero Offset
  - X – the current offset from machine to program coordinates on the X axis
  - Y – the current offset from machine to program coordinates on the Y axis
  - Z – the current offset from machine to program coordinates on the Z axis
  - A (W) – the current offset from machine to program coordinates on the A (or W) axis
- Tool Offset
  - X – the currently applied tool offset on the X axis
  - Y – the currently applied tool offset on the Y axis
  - Z – the currently applied tool offset on the Z axis

Since these variables represent objects rather than simple types, there is a special way to use them. The values represented by these variables can be accessed by specifying the desired property using a period (dot) followed by the property name. For instance,

```
#Machine.X
```

is how you would access the current position on the X axis in machine coordinates.

Array variables are groups of variable values that are indexed. They can be accessed by supplying the desired index value (position in the array). An array variable is declared in much the same way as any other variable, except that the size of the group, also known as the dimension of the array, must be specified. Example:

```
REAL #XPositions{5}
```

The above code declares a REAL array variable called #XPosition that contains 5 values. Arrays can be created for any type (INTEGER, REAL, BOOLEAN, or STRING). The name of the array variable must be followed by a pair of braces, {}, that surround a number. The lowest number allowed for an array dimension is 1. An array of 1 is exactly the same as a non-array variable. The highest number allowed for an array dimension is 10,000. In other words,

```
INTEGER #Val{0}
BOOLEAN #MyArray{10001}
```

will both result in errors when the program is loaded.

An array value is accessed in the same manner as any other variable, except that the index, must be specified. For example,

```
REAL #XPositions{3} #YPositions{3}
#XPositions{1} = 0.0
#XPositions{2} = 1.0
#XPositions{3} = 2.0
#YPositions{1} = 4.0
#YPositions{2} = 3.0
```

```
#YPositions{3} = 2.0
```

The lowest index number is 1, and the highest index number is the dimension of the array variable. In other words, if an array variable is declared with a dimension of 3, the highest index number is 3.

The index itself can be a literal value or a variable value.

```
REAL #XPositions{3} #YPositions{3}
#XPositions{#FabHead.CurrTool} = 0.0
#YPositions{#FabHead.CurrTool} = 2.0
```

In the above example, if no tool is loaded (i.e. #FabHead.CurrTool = 0), an error will result because the lowest index number is 1. Similarly, if #FabHead.CurrTool = 4 or higher, an error will result because both array variables were declared with a dimension of 3.

The system variables #Tool, #RefPoint, #FixOff, and #TlRackPos are array variables. However, they have special meaning because their types are not any of the built-in types (INTEGER, REAL, BOOLEAN, or STRING). Instead, these variables refer to objects (defined in the Configuration dialog box) that in turn have properties. Specifically, #Tool refers to tools, #RefPoint refers to reference points, #FixOff refers to fixture offsets, and #TlRackPos refers to tool rack positions. Here are the properties of each object:

Tool

- X – the X offset of the tool
- Y – the Y offset of the tool
- Z – the Z offset of the tool
- DIAM – the diameter of the tool
- DESC – the description of the tool

Reference Point

- X – the X coordinate of the reference point
- Y – the Y coordinate of the reference point
- Z – the Z coordinate of the reference point
- A(W) – the A (or W) coordinate of the reference point
- DESC – the description of the reference point

Fixture Offset

- X – the X coordinate of the fixture offset
- Y – the Y coordinate of the fixture offset
- Z – the Z coordinate of the fixture offset
- A(W) – the A (or W) coordinate of the fixture offset
- DESC – the description of the fixture offset

Tool Rack

- X – the X coordinate of the tool rack position
- Y – the Y coordinate of the tool rack position
- RAISED – the position to which the tool is to be raised
- NEARGRIP – the position to which the machine tool will move rapidly prior to moving into position to grip the tool
- NEARRELEASE – the position to which the machine tool will move rapidly prior to moving into position to release the tool
- GRIP – the position at which the machine tool will grip the tool in the chuck
- RELEASE – the position at which the machine tool will release the tool from the chuck

Because #Tool, #RefPoint, #FixOff, and #TlRackPos refer to objects, they cannot be accessed directly. However, each individual property of these objects can be retrieved. This is done by adding a period (dot) after the variable name, and then the name of the property. For example,

```
G00 X#RefPoint{1}.X
```

This will cause a rapid move on the X axis to the X coordinate of reference point #1. Similarly,

```
M00 #Tool{1}.DESC
```

will display a message containing the description of tool number 1.

Since #FabHead.CurrTool and #FabHead.NextTool are INTEGERs, you could display the description of the currently loaded tool by typing:

```
M00 #Tool{#FabHead.CurrTool}.DESC
```

User variables are defined on the User Variables panel of the Configuration dialog box. Other than that, FlashCut treats these variables the same as Program variables. For more information see “User Variable Settings” in the Initial Setup section.

Runtime Variables

In the previous discussion of variables, the value of each variable is known when the program is loaded. However, sometimes it is desirable to know the current state of the system while a program is running. For this reason, FlashCut CNC has the ability to examine the current values of certain system variables. This gives the user the ability to have the system execute different parts of a G-code file based on the current value of a desired variable. This feature is called runtime branching, but the capabilities actually extend further than just branching.

The following system variables represent values of the current state of the FlashCut CNC system:

Variable name	Type	Description
#Input	Boolean Array	Value of the input lines
#Output	Boolean Array	Value of the output lines
#Machine	Position	Current position of the machine in machine coordinates

Variable name	Type	Description
#Program	Position	Current position of the machine in program coordinates
#ToolOff	Offset	Current tool offset

The examples below illustrate possible uses of these variables.

Example #1 – Branching based on the state of an input line (runtime branching)

If you wish to execute different G-code depending on the state of the first input line, you could do the following:

```
IF #Input{1} THEN
   (Things to do if input line 1 is tripped)
ELSE
   (Things to do if input line 1 is not tripped)
ENDIF
```

Example #2 – Measuring

If you wish to check the distance between two points, you could do the following:

```
GLOBAL REAL #MyVarBegin
GLOBAL REAL #MyVarEnd
GLOBAL REAL #Diff
(Some G-code
#MyVarBegin = #Machine.X (Current X location, this is our first point)
(Some more G-code
#MyVarEnd = #Machine.X (Current X location. This is our second point)
#Diff = #MyVarEnd - #MyVarBegin
IF #Diff > 3.0 THEN
   M00 "The spacing is too large!"
ENDIF
```

Since runtime variables require information that is only known while the G-code is actually running, two FlashCut features behave differently when the G-code contains runtime variables.

- Toolpath Preview – FlashCut previews the G-code file only up to the first use of a runtime variable.
- Running a G-code file when not connected to the signal generator – When FlashCut encounters a G-code line that requires the value of #Input or #Output, it displays a dialog that asks you to supply the required value. This allows you to simulate the G-code file under various runtime conditions.

Runtime variables may also be used in the Tool Length Formula (on the Tool Length Sensing panel of the Configuration dialog box).

Fabrication Head Commands

For the currently selected fabrication head, #FabHead can be used to issue commands as well as interact with properties of the fabrication head.

Commands appear on a G-code line by themselves, for example:

```
#FabHead.TurnOn
```

For properties, #FabHead is used like a variable:

```
#FabHead.PierceDelay = 2.5 (Sets the pierce delay to 2.5 seconds)
G04 X#FabHead.PierceDelay (Dwells for the pierce delay amount)
```

Except where noted below, numeric property values must be greater than or equal to 0.

General Fab Head Commands

#FabHead.TurnOn	Turns the fabrication head on.
#FabHead.TurnOff	Turns the fabrication head off.
#FabHead.Up	Raises the fabrication head of two-position lift axis.
#FabHead.Down	Lowers the fabrication head of two-position lift axis.
#FabHead.Mode	Sets the fabrication mode (CUT, MARK).

Example: `#FabHead.Mode = CUT`

2D Fab Head Commands

These are commands for 2D fabrication heads such as plasma, laser, waterjet, oxyfuel, etc.

#FabHead.PierceDelay	Gets/sets the pierce delay property of the fabrication head.
#FabHead.Pierce	Executes the fabrication head's configured pierce macro
#FabHead.RunPierceStyle	Executes the pierce style (Stationary, Wiggle, Circular, etc.)
#FabHead.WaitForReadyToFab	Waits for the configured input line state. Similar to M100 or M101 but based on configuration settings. For plasma this will wait for the arc transfer signal
#FabHead.WaitForFabStopped	Waits for the configured input line state. Similar to M100 or M101 but based on configuration settings. For plasma this will wait for a loss of the arc transfer signal

Laser Fab Head Commands

#FabHead.FocalLength	Gets/sets the focal length of the laser.
#FabHead.PowerLevel	Gets/sets the power level of the laser.
#FabHead.TurnPointingOn	Turns the pointing laser on.
#FabHead.TurnPointingOff	Turns the pointing laser off.

#FabHead.EndOfCutDelay	Gets the end of cut delay property of the laser. Read only.

Mill Fab Head Commands

#FabHead.CurrTool	Gets/sets the current tool for the mill.
#FabHead.NextTool	Gets/sets the next tool to be used by the mill.
#FabHead.SpindleSpeed	Gets/sets the spindle speed for the mill.
#FabHead.OpenChuck	Opens the mill chuck.
#FabHead.CloseChuck	Closes the mill chuck.
#FabHead.TurnOnCW	Turns on the mill to spin in a clockwise direction, if capable.
#FabHead.TurnOnCCW	Turns on the mill to spin in a counterclockwise direction, if capable.
#FabHead.ChuckIsOpen	True if the chuck is open. Read only.
#FabHead.TlChgCheckPower	True if the mill is configured to monitor chuck power. Read only.
#FabHead.TlChgPowerInputLine	Gets the chuck power input line. Read only.
#FabHead.TlChgCheckPowerInputNormal	True if input tripped when chuck power is on. Read only.
#FabHead.ChuckPosition	Gets the chuck lift axis position. Read only.

Oxyfuel Fab Head Commands

#FabHead.TurnIgnitionOn	Turns the oxyfuel ignition on.
#FabHead.TurnIgnitionOff	Turns the oxyfuel ignition off.
#FabHead.TurnPreheatOn	Turns oxyfuel preheat on.
#FabHead.TurnPreheatOff	Turns oxyfuel preheat off.
#FabHead.TurnLowPreheatOn	Turns oxyfuel low preheat on.
#FabHead.TurnLowPreheatOff	Turns oxyfuel low preheat off.
#FabHead.TurnHighPreheatOn	Turns oxyfuel high preheat on.
#FabHead.TurnHighPreheatOff	Turns oxyfuel high preheat off.
#FabHead.TurnWaterOn	Turns oxyfuel water on.
#FabHead.TurnWaterOff	Turns oxyfuel water off.
#FabHead.UseWater	True if configured to use water.
#FabHead.UseAutomaticIgnition	True if configured to use automatic ignition.
#FabHead.IgnitionDelay	Gets/sets the oxyfuel ignition delay property.
#FabHead.PurgeDelay	Gets/sets the oxyfuel purge delay property.
#FabHead.WaitForPreheat	Pauses G-code program execution while preheat is occurring.

Waterjet Fab Head Commands

#FabHead.TurnLowPressureOn	Turns waterjet low pressure on
#FabHead.TurnLowPressureOff	Turns waterjet low pressure off

#FabHead.TurnAbrasiveOn	Turns waterjet abrasive on
#FabHead.TurnAbrasiveOff	Turns waterjet abrasive off
#FabHead.UsingAbrasive	True if the use of abrasive configured
#FabHead.AbrasiveOnDelay	Gets/sets the abrasive on delay property
#FabHead.AbrasiveOffDelay	Gets/sets the abrasive off delay property
#FabHead.EndOfCutDelay	Gets/sets the end of cut delay property
#FabHead.UseAutoPierce	True if automatic piercing is currently in use
#FabHead.PierceLength	Gets/sets the pierce length property for wiggle pierce
#FabHead.PierceRadius	Gets/sets the pierce radius property for circular pierce
#FabHead.PierceFeedrate	Gets/sets the feed rate for circular and wiggle pierce movements

Workpiece Definitions

The CAM postprocessor will put workpiece information into the generated g-code file, including remnant information.

These definitions are automatically generated by CAM and are generally not intended to be edited by the user.

Depending on the stock type, there may be more or less workpiece definition parameters in the g-code file. Below is an example of some typical workpiece definitions, but this is not a complete list.

#Workpiece	Specifies the name of workpiece that will be in use.
	Specified at the beginning of the file, before fabrication commands.
#WorkpieceDefinition	Specifies where the workpiece information is within the g-code program.
	No fabrication commands are allowed after this.
#Workpiece.Name	Defines the workpiece name, for example “Sheet 1”
#Workpiece.Units	Defines the units (ENGLISH, METRIC)
#Workpiece.StockType	Defines the stock type (SHEET, BLOCK, PIPE, RECT_TUBE, etc.)
#Workpiece.Material	Defines the material type, for example “Mild Steel”
#Workpiece.Thickness	Defines the workpiece thickness, for example 0. 250000
#Workpiece.Width	Defines the workpiece width
#Workpiece.Length	Defines the workpiece length
#Workpiece.OD	Defines the outside diameter of a round pipe
#Workpice.CornerRadius	Defines the corner radius of a rectangular tube
LabelRemnant	This command will display the remnant name in the CNC viewport.

The positions and sizes of the workpiece or remnant are also defined with a series of B and L values.

Process Settings

The CAM postprocessor will put fabrication head process information into the generated g-code file.

These settings are automatically generated by CAM and are generally not intended to be edited by the user.

Depending on the fabrication head and type of process, there may be more or less process parameter settings in the g-code file. Below is an example of some typical process settings, but this is not a complete list.

#Process	Specifies the ID of the process to use.
	Specified at the beginning of the file, before fabrication commands.
#ProcessDefinition	Specifies where the process information is within the g-code program.
	No fabrication commands are allowed after this.
#Process.ID	The process ID, which is the fabrication head ID and process name.
	For example, “plasma1_Cut”
#Process.Name	The process name, for example “Cut”
#Process.FabHeadType	The fabrication head type and model.
	For example, Plasma.Powermax
#Process.Class	The process class
#Process.Fabhead	The fabrication head ID, for example “plasma1”
#Process.FabHeadMode	The fabrication head mode (Cut, Mark, etc.)
#Process.Units	The process units (ENGLISH, METRIC)
#Process.KerfWidth	The kerf
#Process.CutHeight	The cut height
#Process.PierceHeight	The pierce height
#Process.PierceDelay	The pierce delay
#Process.Feedrate	The feedrate
#Process.SuspendHC	Specifies if height control is suspended or not (True, False)
#Process.VoltageSetPoint	The set point voltage
#Process.Amperage	The amperage
#Process.Pressure	The gas pressure
#Process.PierceStyle	Specifies the pierce style (STATIONARY, WIGGLE, CIRCULAR, etc.)

File Properties

The CAM postprocessor will put g-code file properties into the generated g-code file.

These properties are automatically generated by CAM and are not intended to be edited by the user.

#FileProperties	Specifies where the file properties are within the g-code program.

	No fabrication commands are allowed after this.
#File.SoftwareVersion	The software version the g-code file was generated
#File.CreationDate	The date the file was created, yyyy-mm-dd. For example, 2023-04-01
#File.CreationTime	The time the file was created, hh.mm.ss. For example, 14.30.59

Operators

In addition to declaring variables, setting their values, and using their values, FlashCut supports the use of several operators to make G-code programs more flexible and powerful. There are mathematical, grouping, string, and comparison operators.

Mathematical Operators

The mathematical operators work for INTEGER and REAL values. They are:

- `+` adds two values
- `-` subtracts two values
- `*` multiplies two values
- `/` divides two values

Examples:

```
INTEGER #Count #Index
REAL #X #Y #Z
#Count = 0
#Index = #Count + 1
#Count = #Count - 2
#X = 1.0
#Y = #X * 2.0
#Z = #Y / #X
```

More than one mathematical operator can be used on one line. For example,

```
INTEGER #Val
#Val = 1 + 2 - 3
```

When using mathematical operators, the arithmetic precedence applies. This means that multiplication and division operators are interpreted first, then addition and subtraction. If more than one multiplication or division operator is on the same line, the operators are processed from left to right. The same is true for addition and subtraction operators. Here are some examples:

```
INTEGER #Val
#Val = 2 + 2 - 3              #Val = 1; 2 + 2 is evaluated first, then 3 is
                              subtracted from the result
#Val = 4 / 2 * 3              #Val = 6; 4 / 2 first, then result multiplied by 3
```

```
#Val = 1 + 2 * 3              #Val = 7; 2 * 3 first, then added to 1
#Val = 1 – 2 * 3 + 4          #Val = -1; 2 * 3 first, then subtracted from 1,
                              then 4 added to result
#Val = 4 / 2 – 3 * 1          #Val = -1; 4 / 2 first, then 3 * 1, then second
                              result subtracted from first
```

Grouping Operators

If a different order of evaluation is desired, FlashCut recognizes grouping operators. These are like parentheses in mathematics, except brackets [] are used. Brackets are used because in G-code the left parenthesis starts a comment. Any expression inside a pair of brackets is evaluated before operators outside the brackets. For example,

```
INTEGER #Val
#Val = 1 + 2 * 3              #Val = 7 as explained above
#Val = [1 + 2] * 3            #Val = 9; 1 + 2 first, then result multiplied by 3
```

Pairs of brackets can be nested to clarify complicated expressions:

```
INTEGER #Val
#Val = [[[1 + 2] * 3 – [4 + 5]] / 6 + 7] * 2
                              #Val = 14
                              first 1 + 2 = 3
                              then 3 is multiplied by 3 = 9
                              then 4 + 5 = 9
                              then 9 is subtracted from 9 = 0
                              then 0 is divided by 6 = 0
                              then 7 is added to 0 = 7
                              then 7 is multiplied by 2 = 14
```

String Operator

The only string operator is the concatenation operator.

& combines two strings to form a single string

Example:

```
STRING #Message1 #Message2
#Message1 = "Please change tool"
#Message2 = " and close enclosure"
M00 #Message1 & #Message2
```

The above code will cause the following message to appear:

Please change tool and close enclosure

Since any other type of value can be converted to a STRING, numbers can be displayed very easily. For example,

```
M00 "The currently loaded tool is " & #FabHead.CurrTool
```

Here, the system variable #FabHead.CurrTool is automatically converted to a STRING. If #FabHead.CurrTool is 1, the message is

The currently loaded tool is 1

Note: Do not surround the variable name in quotes, otherwise the variable name itself will be displayed. Surrounding the name in quotes will turn it into a literal value. For example,

```
M00 "The currently loaded tool is " & "#FabHead.CurrTool"
```

will cause the following message to be displayed:

The currently loaded tool is #FabHead.CurrTool

Several concatenation operators can be used on the same line. This allows complicated text messages to be built. Example:

```
M00 "Tool #" & #FabHead.CurrTool &
   " will be replaced by tool #" & #FabHead.NextTool
```

If the current tool is 1 and the next tool is 2, the message displayed will be:

Tool #1 will be replaced by tool #2

Mathematical operators can appear on the same line as the concatenation operator. If this is the case, all mathematical operations are evaluated first. The results of the mathematical operations are converted to strings and the concatenation takes place. Example:

```
M00 "The result " & 1 + 2 &
   " was obtained by adding 2 to 1"
```

The expression 1 + 2 is evaluated first, then the result is converted to a STRING and concatenated with the string literals. The message that is displayed is:

The result 3 was obtained by adding 2 to 1

Comparison Operators

The comparison operators work for any type of value, but both values must be the same type (or must be convertible to the same type). The comparison operators are:

=	TRUE if both values are equal, FALSE if not
!=	TRUE if the values are different, FALSE if they are the same
>	TRUE if the value to the left of the operator is greater than the value to the right, FALSE if not
>=	TRUE if the value to the left of the operator is greater than or equal to the value to the right, FALSE if not
<	TRUE if the value to the left of the operator is less than the value to the right, FALSE if not
<=	TRUE if the value to the left of the operator is less than or equal to the value to the right, FALSE if not

Examples of comparison operator use appear in the following sections.

Boolean Operators

FlashCut also supports Boolean logic operators. These operators can be used wherever a Boolean value is expected.

AND

The operator performs a Boolean AND operation (ie. if both operands are TRUE, the expression is TRUE, otherwise it is FALSE).

Syntax:

```
BOOLEAN operand1 AND operand2
```

Example:

```
BOOLEAN #Test1
BOOLEAN #Test2
BOOLEAN #Result
#Test1 = TRUE
#Test2 = FALSE
#Result = #Test1 AND #Test2   #Result is FALSE
IF #Test1 AND #Test2 THEN     The 'IF' condition is FALSE
…
```

OR

The OR operator performs a Boolean OR operation (ie. if either operand is TRUE or both are TRUE, the expression is TRUE, otherwise it is FALSE).

Syntax:

```
BOOLEAN operand1 OR operand2
```

Example:

```
BOOLEAN #Test1
BOOLEAN #Test2
BOOLEAN #Result
#Test1 = TRUE
#Test2 = FALSE
#Result = #Test1 OR #Test2    #Result is TRUE
WHILE #Test1 OR #Test2 DO     The 'WHILE' condition is TRUE
```

Flow of Control

Flow of control commands let you build simple or complex logic into your G-code programs. FlashCut provides two types of flow of control: Branches and Loops. Branches make it possible for a G-code program to execute different sections of the code based on a condition. The syntax for a Branch is:

```
IF condition THEN
```

```
…
ENDIF
```

where **condition** is a Boolean value or expression. If **condition** is TRUE, the code between the IF line and the ENDIF line is executed. If **condition** is FALSE, the code is skipped and execution continues with the first line after ENDIF.

At times there is a need to execute some code if a condition is true and execute different code if that condition is FALSE. This can be done by using an ELSE statement:

```
IF condition THEN
   …
ELSE
   …
ENDIF
```

In this case, the code between the IF and the ELSE is executed if **condition** is TRUE, and the code between ELSE and ENDIF is executed if **condition** is FALSE.

Example:

```
IF #FabHead.NextTool = 1 THEN
   M00 "Please load tool #1"
ELSE
   M00 "Please load tool #2"
ENDIF
```

In this example, if #FabHead.NextTool (a system variable) is 1, the user will see a message to load tool number 1; otherwise a message to load tool number 2 will appear. The above example includes an example of the use of the '=' comparison operator as well. Remember that the comparison operator result is BOOLEAN (that is, TRUE or FALSE).

For even greater flexibility, additional conditions can be specified:

```
IF #FabHead.NextTool = 1 THEN
   M00 "Please load tool #1"
ELSEIF #FabHead.NextTool = 2 THEN
   M00 "Please load tool #2"
ELSEIF #FabHead.NextTool = 3 THEN
   M00 "Please load tool #3"
ELSE
   M00 "An invalid tool was selected"
ENDIF
```

The above example lets you display different messages based on the next tool to be loaded. Note that only one ENDIF was necessary. Also, note that THEN is required for each IF or ELSEIF.

IF statements can be nested. This means that inside a block of code between IF and ENDIF, one or more IFs can appear. For example,

```
IF #FabHead.CurrTool > 0 THEN
   IF #FabHead.CurrTool = 1 THEN
      M00 "The first tool is loaded"
   ELSEIF #FabHead.CurrTool = 2 THEN
      M00 "The second tool is loaded"
   ENDIF
ELSE
   IF #FabHead.CurrTool = 0 THEN
      M00 "No tool is loaded"
   ELSE
      M00 "An invalid tool is loaded"
   ENDIF
ENDIF
```

The lines of code between the first IF and the ELSE will be executed if the currently loaded tool number is greater than zero. The actual message that will appear depends on the number of the tool that is loaded. Notice that each block of code that starts with an IF eventually ends with an ENDIF. These pairings must be present, or an error will result when the G-code file is loaded.

The other type of flow of control statement is the Loop. The syntax of the Loop is:

```
WHILE condition DO
   …
ENDWHILE
```

The lines of code between the WHILE and the ENDWHILE will be executed as long as **condition** is TRUE. When **condition** is FALSE, the loop is exited and execution continues with the line of code following the ENDWHILE. Example,

```
INTEGER #Count
#Count = 0
WHILE #Count < 5 DO
   M00 "Loop count = " & #Count
   #Count = #Count + 1
ENDWHILE
M00 "Not in loop"
```

This code will cause the following messages to appear:

Loop count = 0

Loop count = 1

Loop count = 2

Loop count = 3

Loop count = 4

Not in loop

If the condition specified in the loop is initially FALSE, the code in the loop will be skipped entirely.

Note: When creating loops, you must be careful to ensure that the loop condition will eventually be FALSE. Otherwise, an infinite loop will result and the program will never load. In the example above, omitting #Count = #Count + 1 creates an infinite loop.

Loops can appear inside other loops or branches. Branches can appear inside loops as well as other branches. Here's a program that illustrates this as well as most of the other features discussed in this section.

```
(Use GLOBAL for these variables because their values
(are used in the main routine and in subroutines
GLOBAL INTEGER #Row
GLOBAL INTEGER #Col
GLOBAL REAL #XBase #YBase
GLOBAL REAL #RowSpacing #ColSpacing
GLOBAL REAL #XLoc #YLoc
GLOBAL BOOLEAN #MakeMeSmile
GLOBAL INTEGER #Result
(Don't need GLOBAL for these variables because they
(are only used in the main routine
INTEGER #TotalRows
INTEGER #TotalCols
#TotalRows = 3
#TotalCols = 4
#XBase = 0.3
#YBase = 0.3
#RowSpacing = 0.8
#ColSpacing = 0.7
#Row=0
#Col=0
F10
WHILE #Row < #TotalRows DO
   WHILE #Col < #TotalCols DO
      M98 PUpdateLocation
      #Result = #Col / 2
      IF [#Col - #Result * 2] = 1 THEN
         #MakeMeSmile = TRUE
      ELSE
         #MakeMeSmile = FALSE
      ENDIF
```

```
      M98 PSmiley
      #Col=#Col+1
   ENDWHILE
   #Row=#Row+1
   #Col=0
ENDWHILE
G00 X0 Y0
OUpdateLocation
#XLoc = #XBase + #Col * #ColSpacing
#YLoc = #YBase + #Row * #RowSpacing
G00 X#XLoc Y#YLoc
M99
OSmiley
G00 Z0
G91
G00 X-0.15 Y0.1
G01 Z-0.5
G02 I0.05
G01 Z0.5
G00 X0.3
G01 Z-0.5
G02 I-0.05
G01 Z0.5
G00 Y-0.2
G01 Z-0.5
IF #MakeMeSmile THEN
   G02 X-0.3 I-0.15 J0.1
ELSE
   G03 X-0.3 I-0.15 J-0.1
ENDIF
G01 Z0.5
G00 X-0.15 Y0.1
G01 Z-0.5
G02 I0.3
G90
G00 Z0
M99
```

File Access Commands

Three file access commands are supported in FlashCut. These commands allow the G-code programmer to save data to a file while the program is running.

CreateFile Command

The CreateFile ommand creates and opens a new file. If the file already exists, it is overwritten. If the file cannot be created, an error message is displayed. If another file is open, it is closed; only one file can be open at a time.

Syntax:

```
CreateFile[filename]
```

where `filename` is the name of the file you wish to create. The file name can include an absolute path, or a path that's relative to the current G-code file.

Examples:

```
CreateFile["MyFile.txt"]      Creates MyFile.txt in the same folder as the G-
                              code file

CreateFile["C:\Users\Guest\Documents\AnotherFile.txt"]

STRING #FileName

#FileName = "TheFile.abc"

CreateFile[#FileName]
```

OpenFile Command

The OpenFile command opens an existing file. If the file does not exist a new one is created. Anything that is saved to the file after it is opened will be saved to the end of the file. If the file cannot be opened, an error message is displayed. If another file is open, it is closed; only one file can be open at a time.

Syntax:

```
OpenFile[filename]
```

where `filename` is the name of the file you wish to open. The file name can include an absolute path, or a path that's relative to the current G-code file.

Examples:

```
OpenFile["MyFile.txt"]        Opens MyFile.txt located in the same folder as the
                              G-code file

OpenFile["C:\Users\Guest\Documents\AnotherFile.txt"]

STRING #FileName

#FileName = "TheFile.abc"

OpenFile[#FileName]
```

WriteLine Command

The WriteLine command writes a line of data to an open file. If a file has not been opened, an error message is displayed.

Syntax:

```
WriteLine[line]
```

where `line` is the text to write into the file.

Examples:

```
WriteLine["This is a test"]
STRING #TestLine
#TestLine = "Another test"
WriteLine[#TestLine]
WriteLine["Program X: " & #Program.X]
```

G-Code Data Files

FlashCut supports G-code data files. These files can be read while a G-code program is running to provide data that can be used in the G-code program. A typical use for G-code data files is to provide point locations. But due to the flexibility of the design, G-code data files can be used for any purpose to supply data to a program.

A G-code data file is an XML file that contains the data format and the data. An XML file is a collection of tags. These tags come in pairs, an opening tag and a closing tag. The tag name can be any string of letters, but cannot contain any spaces. A tag is surrounded by angle brackets (<tag>). The opening tag contains only the name inside the brackets (<tag>) but the closing tag contains a slash in front of the name (</tag>). XML tags must come in pairs (opening and closing). Tags can be nested within each other, but must be closed in the reverse order they were opened. For example:

```
<OuterTag>
   <InnerTag>
   </InnerTag>
</OuterTag>
```

An XML tag defines an element. The element's name is the tag name. So <tag> defines an element named "tag". Some elements can have attributes. Attributes are specified inside the angle brackets with the tag, and the value of the attribute must be enclosed in quotes. For instance:

```
<Test Field="XYZ">
```

defines an element named "Test" and sets its "Field" attribute to "XYZ".

The names of elements and the attributes that are valid for each element are defined by the program that reads the XML file. In the case of G-code data files, that is FlashCut CNC.

While opening tags must have a closing tag, there is a shorthand way to close a tag. The slash can appear at the end of the tag line (<tag attribute="X"/>). Often, the use of this shorthand gives the file a cleaner look. However, it is just as valid to write <tag attribute="X"></tag> and the writer of the XML file is free to choose the form that he prefers.

Here is an example of a G-code data file:

```
<GCodeData>
   <Format>
      <Attributes>
```

```
            <Attribute Name="X" Type="REAL"/>
            <Attribute Name="Y" Type="REAL"/>
            <Attribute Name="FLAG" Type="INTEGER"/>
         </Attributes>
      </Format>
      <Data X="1.0" Y="2.0" FLAG="1"/>
      <Data X="2.0" Y="1.0" FLAG="2"/>
      <Data X="0.0" Y="0.0" FLAG="3"/>
   </GCodeData>
```

The first tag, <GCodeData>, is required. The format and data lines must appear in the file between this tag and its closing </GCodeData>.

The next tag, <Format>, is required as well. This tag tells FlashCut how to interpret the data lines in the file.

The <Attributes> tag appears between <Format> and </Format>. This is also required. This is where the fields (or attributes) of each data line are defined.

Each attribute of the data is defined by the <Attribute> tag. The data contained in the G-code data file can have any attribute, as long as it is specified in this section.

There are two types of <Attribute>: Name and Type. The Name attribute defines the name of the data attribute. This name can be any sequence of letters, but must not contain any spaces. The Type attribute defines the type of the data attribute. The Type must be one of the following:

- REAL – a floating point number
- INTEGER – a whole number, positive or negative (or zero)
- STRING – text
- BOOLEAN – must be TRUE or FALSE

In the example above, the G-code data have three attributes: X, which is a real number; Y, which is a real number; and FLAG, which is an integer.

The <Data> elements appear after the format section. Any (or all) of the attributes for the <Data> may be specified, but it is not an error if an attribute is not specified. However, any attribute that is not specified will have an undefined value.

Any number of <Data> elements can be specified.

Loading Data Files

The G-code data file is loaded into the running program by calling the LoadData function.

Syntax:

```
LoadData[filename]
```

where `filename` is the name of the XML file containing the G-code data.

Examples:

```
LoadData[“MyData.xml”]
LoadData[“C:\Users\Guest\Documents\MoreData.xml”]
```

```
STRING #FileName
#FileName = "StillMoreData.xml"
LoadData[#FileName]
```

If the file name does not include a path, FlashCut looks for the file in the same folder as the G-code file.

Multiple data files can be loaded, but the format for the data elements must be the same or errors will occur. The data in each file that is loaded will be appended to the data that have already been loaded.

Using Data Files

The data contained in the G-code data files are read and stored in an array. This array can be accessed via the predefined variable #DataList. #DataList is an array variable. The order of the data items in #DataList is the same as the order they were specified in the G-code data file.

The attributes of each item in #DataList are accessed as properties. For example:

```
#DataList{2}.FLAG
```

accesses the FLAG attribute of the second item in the data list.

Another predefined variable, #DataListCount, contains the number of items in #DataList. Using this variable, it's easy to iterate through all items in the data list:

```
INTEGER #Item
#Item = 1
WHILE #Item <= #DataListCount DO
   G01 X#DataList{#Item}.X Y#DataList{#Item}.Y (Move to point)
   #Item = #Item + 1 (Don't forget to increment #Item!)
ENDWHILE
```

Clearing Data

As mentioned above, multiple data files can be loaded, and the data from each file will be appended to the data list. Because of this, a means for clearing the data list has been provided. The ClearData command clears the data list and sets the list count to 0. If ClearData is followed by a LoadData, the data in the file specified by the LoadData function will be the only data in the list.

It is not necessary to clear the data list before loading the first G-code data file. The data list is automatically cleared each time a new G-code file is loaded.

Feedrate Override Commands

There are two commands that affect feedrate override.

EnableFO/DisableFO

The EnableFO/DisableFO pair of keywords controls how feedrate override affects the G-code program. These keywords are provided because there are cases when an override to the programmed feedrate cannot be tolerated. For example, in an automatic tool change macro, feedrate moves are often used. Overriding these feedrates might cause problems, so DisableFO can be used to ensure that the programmed feedrate will be enforced.

Here's an example of EnableFO and DisableFO:

```
G00 X1 Y1
DisableFO                    Make sure the programmed feedrate is not
                             overridden
G01 Z-0.5 F3
EnableFO                     Override is okay from this point
…                            Continue
```

In the above example, the programmer ensures that the tool will plunge at a rate of 3 inches/minute by disabling feedrate override before performing the plunge.

SetFO

The SetFO keyword sets the feedrate override percent to a specified value. A typical use for this command is to reset feedrate override to 100% at certain points in a G-code file, to override any changes made by the operator.

Example:

```
F10
G01 X20 Y0                   During this move the operator adjusts feedrate
                             override to 150%, so the machine tool is moving at
                             15 inches/minute
SetFO[100]                   Feedrate override is set to 100%
G01 Y5                       The feedrate for this move is 10 inches/minute
…                            Continue
```

Saving and Restoring the System State

The SaveState/RestoreState pair of keywords causes certain state variables to be saved or restored. The specific status items affected are:

- Offset from Machine to Program coordinates (G92/G54/G55/G56/G57/G58/G59/G54.1 /Program Set Button)
- Absolute vs. relative positioning (G90/G91)
- English vs. Metric units (G20/G21 and G70/G71)
- Canned cycle return position (G98/G99)
- Feedrate (F)
- Spindle Speed (S)

- Local coordinate system (G52)
- Arc plane (G17/G18/G19)
- Motion mode (G00/G01/G02/G03/G73/G76/G81/G82/G83/G85)
- Cutter compensation (G41/G42)
- Scaling (G51)
- Feedrate override enable/disable (EnableFO/DisableFO)
- Safe envelope (G180/G181)
- Feedrate mode (G93/G94)

The SaveState/RestoreState keywords are most useful in macros and subroutines where you wish to restore the state of the program before exiting from the macro or subroutine. Here's an example of keyword use:

```
G91                          Relative positioning mode
G20                          English units
…                            Do some things
M98 PSub1
…                            Do some more things
M02
OSub1
SaveState
G90
…                            Do some things
RestoreState
M99
```

By using SaveState and RestoreState, the programmer does not have to change back to G91 mode before exiting Sub1. Certainly, a change to G91 would be sufficient for the above example; however, consider the following example:

```
G91                          Relative positioning mode
…                            Do some things
M98 PSub1
…                            Do some more things
G90                          Absolute positioning mode
M98 PSub1
…                            Do some more things
M02
OSub1
SaveState
G90
…                            Do some things
```

```
RestoreState
M99
```

In this example, Sub1 is called when the main program is in G91 mode, and then called again when the main program is in G90 mode. Without the use of SaveState and RestoreState, the programmer would not be able to ensure that the correct mode was in effect when Sub1 was exited.

A powerful feature of SaveState and RestoreState is that they can be nested. This is important when a subroutine calls another subroutine and both are using SaveState and RestoreState. Here is the effect of nesting SaveState and RestoreState:

```
G91                         Relative positioning mode
SaveState
G90                         Absolute positioning mode
SaveState
G91                         Relative positioning mode
RestoreState                This puts the program back into absolute
                            positioning mode
RestoreState                This puts the program back into relative
                            positioning mode
```

While the above examples only demonstrated setting the G90/G91 mode, FlashCut would handle all state variables in similar manner.

Mathematical Functions

FlashCut provides a complete set of mathematical functions for use in your G-code file.

SIN

This function returns the sine of the supplied angle. The angle is in degrees.

Syntax:

```
REAL SIN[angle]
```

Example:

```
REAL #SineValue
#SineValue = SIN[30]          #SineValue is set to 0.5
```

COS

This function returns the cosine of the supplied angle. The angle is in degrees.

Syntax:

```
REAL COS[angle]
```

Example:

```
REAL #CosineValue
#CosineValue = COS[60]        #CosineValue is set to 0.5
```

TAN

This function returns the tangent of the supplied angle. The angle is in degrees. Calling this function will result in an error if the angle is 90 or 270 (or equivalent increments of 360 degrees).

Syntax:

```
REAL TAN[angle]
```

Examples:

```
REAL #TangentValue
#TangentValue = TAN[45]       #TangentValue is set to 1.0
#TangentValue = TAN[90]       Error! The tangent is infinity
```

ASIN

This function returns the arcsine of the supplied number. The angle returned is in degrees and will be from -90 to 90. The number supplied is the normalized size of the side adjacent to the desired angle and must be between -1 and 1 or an error will result.

Syntax:

```
REAL ASIN[number]
```

Example:

```
REAL #ArcsineValue
#ArcsineValue = ASIN[0.707]   # ArcsineValue is set to 45.0
```

ACOS

This function returns the arccosine of the supplied number. The angle returned is in degrees and will be from 0 to 180. The number supplied is the normalized size of the side opposite of the desired angle and must be between -1 and 1 or an error will result.

Syntax:

```
REAL ACOS[number]
```

Example:

```
REAL #ArccosineValue
#ArccosineValue = ACOS[-0.707]   # ArccosineValue is set to 135.0
```

ATAN

This function returns the arctangent of the supplied x and y values. The angle returned is in degrees and will be from -180 to 180. The x and y values supplied are the sizes of the sides adjacent and opposite (respectively) to the desired angle.

Syntax:

```
REAL ATAN[yvalue,xvalue]
```

Example:

```
REAL #ArctangentValue
#ArctangentValue = ATAN[1,1]  # ArctangentValue is set to 45.0
```

ROUND

This function returns an integer that is the nearest whole number to the value supplied.

Syntax:

```
INTEGER ROUND[origValue]
```

Examples:

```
INTEGER #IntegerValue
#IntegerValue = ROUND[2.5]    #IntegerValue is set to 3
#IntegerValue = ROUND[2.49]   #IntegerValue is set to 2
#IntegerValue = ROUND[-2.5]   #IntegerValue is set to -3
#IntegerValue = ROUND[-2.49]  #IntegerValue is set to -2
```

ROUNDDOWN

This function returns an integer that is the nearest whole number less than or equal to the value supplied.

Syntax:

```
INTEGER ROUNDDOWN[origValue]
```

Alternate syntax for Fanuc compatibility:

```
INTEGER FIX[origValue]
```

Examples:

```
INTEGER #IntegerValue
#IntegerValue = ROUNDDOWN[2.5]   #IntegerValue is set to 2
#IntegerValue = ROUNDDOWN[2.99]  #IntegerValue is set to 2
#IntegerValue = ROUNDDOWN[-2.5]  #IntegerValue is set to -3
#IntegerValue = ROUNDDOWN[-2.01] #IntegerValue is set to -3
```

ROUNDUP

This function returns an integer that is the nearest whole number greater than or equal to the value supplied.

Syntax:

```
INTEGER ROUNDUP[origValue]
```

Alternate syntax for Fanuc compatibility:

```
INTEGER FUP[origValue]
```

Examples:

```
INTEGER #IntegerValue
#IntegerValue = ROUNDUP[2.5]     #IntegerValue is set to 3
```

```
#IntegerValue = ROUNDUP[2.01]    #IntegerValue is set to 3
#IntegerValue = ROUNDUP[-2.5]    #IntegerValue is set to -2
#IntegerValue = ROUNDUP[-2.99]   #IntegerValue is set to -2
```

SQRT

This function returns the square root of the value supplied. The supplied value must be 0 or greater or an error will result.

Syntax:

```
REAL SQRT[baseValue]
```

Example:

```
REAL #SquareRoot
#SquareRoot = SQRT[81]        #SquareRoot is set to 9.0
```

ABS

This function returns the absolute value of the number supplied.

Syntax:

```
REAL ABS[origValue]
```

Examples:

```
REAL #RealValue
#RealValue = ABS[2.0]         #RealValue is set to 2.0
#RealValue = ABS[-2.0]        #RealValue is set to 2.0
```

MOD

This function returns the remainder of the dividend divided by the divisor.

Syntax:

```
INTEGER MOD[dividend,divisor]
```

Example:

```
INTEGER #Remainder
#Remainder = MOD[27,4]        #Remainder is set to 3
```

POW

This function returns a value that is the base raised to the power of the exponent.

Syntax:

```
REAL POW[base,exponent]
```

Example:

```
REAL #RealValue
```

```
#RealValue = POW[4,3]          #RealValue is set to 64.0
```

LN

This function returns the natural logarithm for the value supplied.

Syntax:

```
REAL LN[baseValue]
```

Example:

```
REAL #RealValue
#RealValue = LN[4.0]           #RealValue is set to 1.38629
```

EXP

This function returns a value that is ‘e’ raised to the power of the exponent.

Syntax:

```
REAL EXP[baseValue]
```

Example:

```
REAL #RealValue
#RealValue = EXP[1.38629]      #RealValue is set to 4.0
```

LOG

This function returns the base 10 logarithm of the value supplied.

Syntax:

```
REAL LOG[baseValue]
```

Example:

```
REAL #LogValue
# LogValue = LOG[100]          #LogValue is set to 2.0
```

REAL

This function converts the integer value supplied to a real value.

Syntax:

```
REAL REAL[integerValue]
```

Alternate syntax for Fanuc compatibility:

```
REAL ADP[integerValue]
```

Example:

```
REAL #RealValue
#RealValue = REAL[2]           #RealValue is set to 2.0
```

Other Functions

The function GetChangerIndex, lets you determine the tool changer position for a given tool.

Syntax:

```
INTEGER GetChangerIndex[toolNumber]
```

Example:

```
INTEGER #Index
#Index = GetChangerIndex[2]   Get the tool changer position of tool #2
```

Remote Control

FlashCut CNC has the ability to be controlled by an external program. This can be accomplished by writing an application that uses the FlashCut Remote Control Interface (FCRCI).

The FCRCI is a set of functions that have been defined and packaged in such a way as to be accessible via any .NET language program. This interface is in the form of a managed DLL called Remote.DLL. The classes defined in Remote.DLL contain useful functions that allow the remote application to control various aspects of the FlashCut CNC program.

The steps for using an application to control FlashCut CNC are simple:

1. Obtain Remote.DLL and RemoteLoader.DLL from FlashCut. See http://www.flashcutcnc.com/ or call technical support for more information. Place both of these files in the same folder as the FlashCut CNC application (usually something like **C:\Program Files\FlashCut CNC 5**).
6. Write a .NET application that exercises the part of the FCRCI that you desire. You can use any .NET language you wish.
7. Place your application in the same folder as the FlashCut CNC application.
8. Run FlashCut CNC.
9. Run your application.

When your application runs, it should attempt to connect to the FlashCut CNC application. If the attempt is successful, your application will be able to control FlashCut CNC.

Two example programs are included in the FCRCI package: one written in Visual Basic and the other in C#. These samples should give you the information you need to build your own remote control program.

The namespace that contains all classes in the FCRCI is called CNCRemote. In the CNCRemote namespace, the main class is API. The API class contains most of the functions and therefore is the object through with most activity will take place. When writing your application, you must first set up .NET remoting (examples of how to do this are in both sample programs). Once .NET remoting is set up, you can obtain an instance of the API class. Finally, after getting the API object, you can call the Connect() function on it. This establishes the connection to the FlashCut CNC program.

An additional step that was added is to call **Ping()** after the connection has been established. Testing has shown that the first function called after connecting via .NET remoting usually takes much longer to execute than all subsequent functions. For this reason, a function call was added to "prime the pump" and get that time delay out of the way.

Functions

Once you have an API object, you can call any of the functions listed below on your API object:

void Connect(Responder^ responder) – This function attempts to connect the remote application to the FlashCut CNC program.

Parameters: responder – A handle to the object that will receive response messages from FlashCut CNC.

void Disconnect(Responder^ responder) – This function attempts to disconnect the remote application to the FlashCut CNC program.

Parameters: responder – A handle to the object that is receiving response messages from FlashCut CNC. Kkk

void LoadGCodeFile(System::String^ fileName) – This function loads the G-code file specified by the supplied fileName. Any errors that occur while loading the file are reported in the FlashCut CNC program

Parameters: fileName – The full path to the file to be loaded.

void StartGCode()– This function causes the G-code file loaded in FlashCut CNC to run.

void Move(CommonTypes::Position^ position,

CommonTypes::PointMoveType iMoveType,bool bIsRapid, double dFeedrate) – This function causes FlashCut CNC to move to the specified point in the manner described by the parameters supplied.

Parameters:

- position – The position to which the system is to move.
- iMoveType – The type of move (incremental, program, machine, or relative).
- bIsRapid – An indication of whether a rapid or feedrate move is desired.
- dFeedrate – The feedrate at which a feedrate move is to take place. Must be greater than 0, even if this is a rapid move.

CommonTypes::Position^ GetCurrentMachineCoordinates()

CommonTypes::Position^ GetCurrentProgramCoordinates()

CommonTypes::Position^ GetCurrentRelativeCoordinates() – These functions return the current coordinates in the appropriate coordinate system.

void Ping(Responder^ responder) – This function causes FlashCut CNC to send a message back to the remote application.

Parameters: responder – A handle to the object that will receive the ping response message from FlashCut CNC.

Messages

Messages are used to communicate from the FlashCut CNC system to the remote application. Each message contains a specific ID that indicates the purpose of the message. Useful message IDs that are sent to the remote application are:

- 501 – A feed hold occurred (motion was stopped).
- 706 – The G-code program is loaded.
- 708 – Program execution is complete.
- 731 – The move to a specific point completed successfully.
- 746 – Ping response code.
- 1001 – An error occurred while loading a G-code file.
- 2000 – A limit switch was hit while motion was occurring. This is an error code. This code will not occur during homing.
- 2014 – A safety interlock was triggered. This will only occur if your system has interlocks configured.

- 99999 – An error occurred while loading a file. The system may be busy.

Additional message that are sent, but are of limited use, are:

- 702 – Motion mode update (rapid vs. feedrate, line vs. arc)
- 716 – The feedrate has been updated.
- 717 – The feedrate override has been updated.
- 720 – Motion has stopped.
- 725 – Progress update (percentage completion).
- 738 – Controller status update (eventually will contain position data).
- 742 – Run time estimate update.
- 900 – Preview message (for displaying the toolpath).

By responding to the messages that are sent from FlashCut CNC, you can ensure that your remote application will operate in harmony with FlashCut CNC.