

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	Return from Reference Point
G30	Move to Reference Points 2-20
G40	Cancel Cutter Compensation
G41	Cutter Compensation (Left)
G42	Cutter Compensation (Right)
G43	Tool Length/Geometry Compensation (Plus)
G44	Tool Length/Geometry Compensation (Minus)
G49	Cancel Tool Length/Geometry Compensation
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)
G70	Inch Units (same as G20)
G71	Metric Units (same as G21)
G73	High Speed Peck Drilling Cycle
G76	Thread Cutting Cycle
G80	Cancel Canned Cycle
G81	Drilling Cycle
G82	Counterboring Cycle

G83	Peck Drilling Cycle
G85	Reaming Cycle
G90	Absolute Positioning Mode
G91	Incremental Positioning Mode
G92	Set Program Zero (XYZA Parameters)
G98	Return to Initial Level (Canned Cycles)
G99	Return to Rapid Level (Canned Cycles)

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)

Other Commands Supported

F	Set Feedrate
T	Select Tool
S	Set Spindle Speed
()	Comment
/	Optional Line

Advanced Keyword Commands and Functions

EnableFO	Enable Feedrate Override
DisableFO	Disable Feedrate Override
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 and 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.

The following list shows the mode groups supported by FlashCut.

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)

Tool Length/Geometry Compensation

G43 Tool Length/Geometry Compensation (Plus)

G44 Tool Length/Geometry Compensation (Minus)

G49 Cancel Tool Length/Geometry Compensation

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

G76 Thread Cutting Cycle

G80 Cancel Canned Cycle

G81 Drilling Cycle

G82 Counterboring Cycle

G83 Peck Drilling 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

Independent Modal Commands

G52	Use Local Coordinate System
F	Set Feedrate
S	Set Spindle Speed

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 3-axis linear interpolation. The rapid rate is calculated from the Maximum Feedrates defined in the Feedrate/Ramping panel of the Configuration dialog box.

Example:

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

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

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

Example:

G00 X4.0 Y3.0	<i>Moves the tool to program coordinates X=4.0, Y=3.0, leaving the Z position unchanged</i>
---------------	---

- 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	<i>Rapid Move</i>
X4.0 Y6.5 Z1.0	<i>Rapid Move</i>
G01 X3.0 Y3.0 Z1.4	<i>Feedrate Move</i>
X2.8 Y1.4 Z0	<i>Feedrate Move</i>

- The interpretation of the coordinates depends on the G90/G91 command in effect.
- FlashCut can move up to three axes simultaneously. If you specify a move on all 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 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 3-axis 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 three 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 the A (rotary) axis, there are two ways FlashCut can interpret the current F setting, depending on your setting in the G-Code 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 pure linear and rotary-included moves.

Rotary Feedrate (Degrees/Minute) – This option directs FlashCut to interpret the feedrate command (F) as a rotary feedrate. The A 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 the A 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.

- FlashCut can move up to three axes simultaneously. If you specify a move on all 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 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.

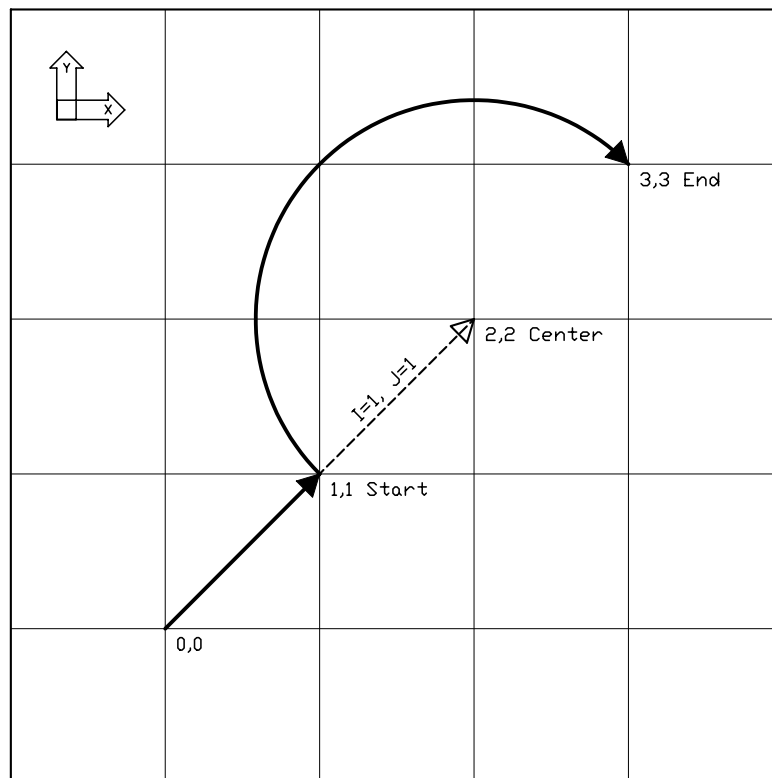
Example:

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.

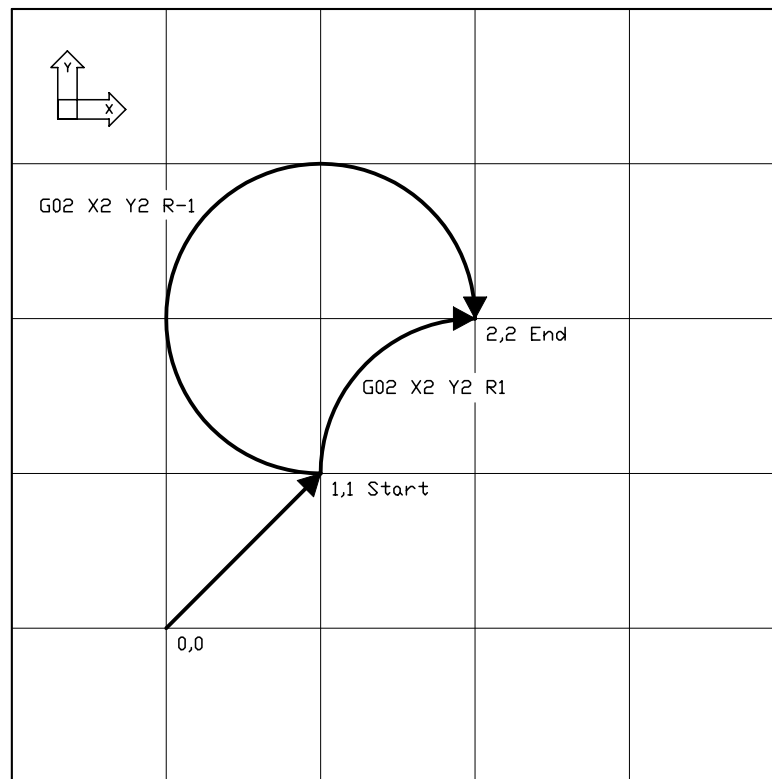
Example:

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

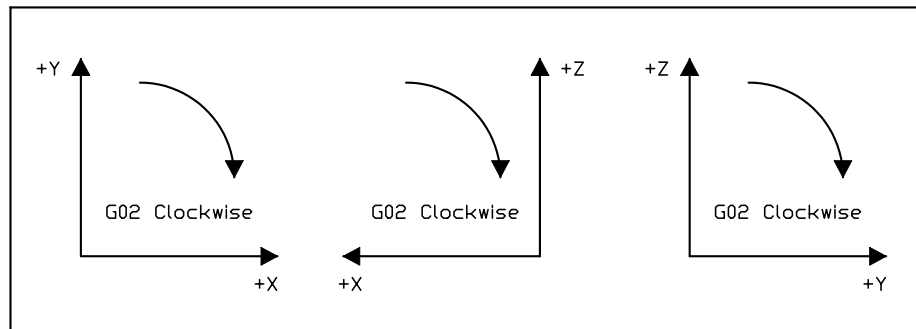
```
G02 X3 Y3 Z2 I1 J1
```

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

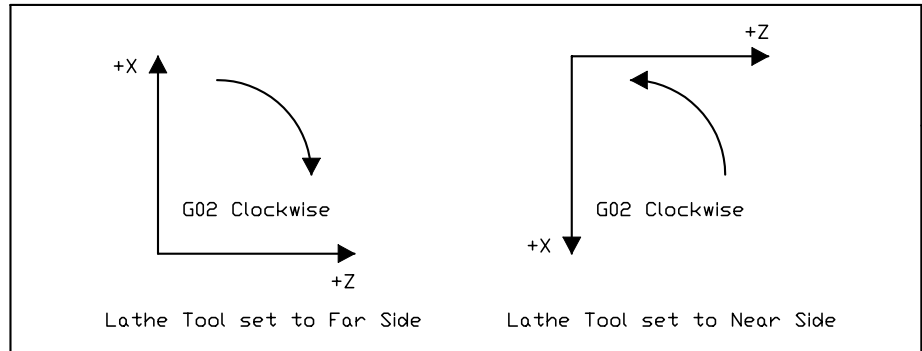
The accuracy of the helical move, relative to a theoretically perfect helix, is controlled by the Helical Interpolation Accuracy setting on the G-Code panel of the Configuration dialog box (see “G-Code Settings” in the Initial Setup section).

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 viewport. For example, in lathe applications, the Lathe Tool setting (on the Machine Tool panel of the Configuration dialog box) determines whether the X positive direction is up or down in the ZX viewport. 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.

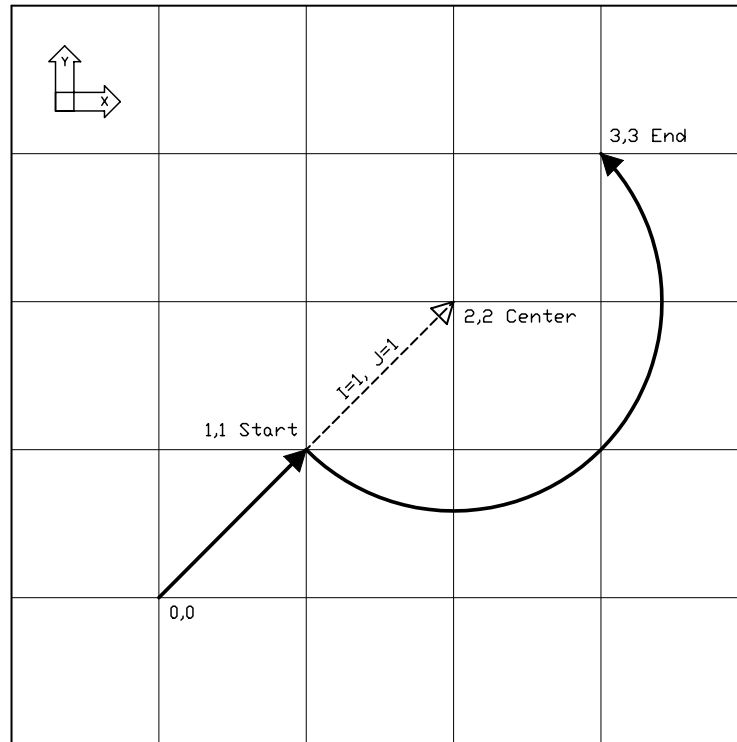
Example:

```
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

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 Options 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 Seek Home button on the Home Control Panel. The purpose of this command is to allow checking for positioning errors, and to allow re-zeroing to the switches. 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).

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

For more details, see "Home Control Panel" in the Main Screen Features section, and "Homing Settings" in the Initial Setup section.

G28, G30 Move to Reference Point

The G28 and G30 commands move the tool at the rapid rate to the associated reference point defined in 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 the parameters "X0", "Y0", "Z0" or "A0" 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	<i>Linear move to the program coordinates X=1.5, Y=2.5, Z=-3</i>
G28 Z0	<i>Rapid move in the Z axis only to machine coordinate Z=-1</i>
G01 X1.5 Y2.5 Z-3 F10	<i>Linear move to the program coordinates X=1.5, Y=2.5, Z=-3</i>
G28	<i>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</i>
G01 X1.5 Y2.5 Z-3 F10	<i>Linear move to the program coordinates X=1.5, Y=2.5, Z=-3</i>
G28 X0 Y0 Z0	<i>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)</i>
G30 P2	<i>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</i>

G29 Return from Reference Point

The G29 command moves the tool to the designated XYZA program coordinate at the rapid rate.

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	<i>Linear move to the program coordinates X=1.5, Y=2.5, Z=-3</i>
G28	<i>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</i>
G29 X2 Y3 Z-2	<i>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</i>

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.

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 you've entered for the tool (i.e. the radius) in the Tooling panel of the Configuration dialog box.

Prior to the first G41/42 command, you should indicate the current tool using the M06 command (eg. 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 (eg. 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).

Note that cutter compensation is available only in milling mode (Machine Type drop-down menu on the Machine Tool panel of the Configuration dialog box).

For information on customizing cutter compensation, please see "Cutter Compensation Setup" in the Initial Setup section of this manual.

The following example demonstrates use of cutter compensation.

Example:

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

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

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

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 and A 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" (using only the letters for axes used on your machine).

When FlashCut reads a G52 command, it displays a magenta dot in the Toolpath Viewport Box 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	<i>Moves the tool to program coordinates X=1.0, Y=3.0, Z=-1.5</i>
G52 X3 Y-7 Z0	<i>Activates a local coordinate system with origin at X=3, Y=-7, Z=0 relative to Program Zero (the machine tool does not move)</i>
G01 X1.0 Y10.0 Z2.0	<i>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</i>

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 (eg. 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 in 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

in 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

where 'x', 'y', 'z' and 'a' are the new current program coordinates.

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 in the Tooling 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.

G43, G44, G49 Tool Length/Geometry Compensation Commands

Tool length/geometry compensation lets FlashCut account for differences in tool lengths and geometry when the G-Code file is executed. This lets you create a G-Code program without knowing the exact geometry of the tools that will be used to cut the part.

In milling mode, compensation is generally used to account for differences in tool lengths in the Z direction. Generally, X and Y offsets are not used. However, to accommodate a milling-style machine with multiple spindles, the offset between spindles can be handled using the X and Y components of the tool offset.

In lathe mode, compensation is generally used to account for differences in tool tip positions, in both X and Z directions.

THESE COMMANDS ARE NOT FOR THE NOVICE CNC USER. WHEN NOT PROPERLY USED, TOOL LENGTH AND GEOMETRY COMPENSATION CAN CRASH THE MACHINE TOOL, CAUSING SERIOUS DAMAGE TO YOUR WORKPIECE OR MACHINE TOOL.

When applying tool length and geometry compensation, FlashCut uses the offsets defined in the Tooling panel of the Configuration dialog box. See “Tooling Settings” in the Initial Setup section for more information on defining your tool offsets.

For tool length and geometry compensation to work properly, FlashCut must know what tool is in use at all times. Therefore, you must indicate every tool change with an M06 command in the G-Code file. Similarly, whenever you change a tool between running G-Code files, you should select it in the Current Tool pull-down menu. The Compensation Mode pull-down menu is set to Positive (G43) by default.

Technically, the M06 command indicating your starting tool is optional, but we recommend you include it so the file is self-contained and less prone to operator error. However, if you choose not to include an M06 command for your starting tool, you must make certain you choose your starting tool from the Current Tool pull-down menu on the main screen, before running the file.

Once the M06 command has set the current tool, the G43 command applies the proper offset to account for the current tool’s length or geometry as follows:

G43 Hn

where n is the tool number for the current tool.

Milling Mode (Tool Length Compensation) – The G43 command tells FlashCut to shift all subsequent Z axis moves away from the workpiece (in the positive Z direction) by an amount equal to the Z offset for tool ‘n’.

Lathe Mode (Tool Geometry Compensation) – The G43 command tells FlashCut to shift all subsequent moves away from the workpiece (in the positive X and Z directions) by an amount equal to the X and Z offset for tool ‘n’.

If tool length/geometry compensation is already in effect for a previous tool, the resulting shift away from the workpiece equals the difference between offsets for tool ‘n’ and the previous tool.

Milling Mode Example:

G43 H3

Shifts all subsequent Z axis moves away from the workpiece (in the positive Z direction) by the Z offset for Tool #3

Lathe Mode Example:

G43 H3

Shifts all subsequent X and Z axis moves away from the workpiece (in the positive X and Z directions) by the X and Z offset for Tool #3

The G44 command is identical to the G43 command, except that it shifts all moves in the direction opposite from G43. Unless you are an experienced CNC programmer and know how to use G44 correctly, G43 is the preferred command.

The G49 command cancels tool length/geometry compensation.

When using tool length compensation there are several important things to keep in mind:

- You must predefine all tools and associated offsets in the Tooling panel of the Configuration dialog box.
- It's good practice to include an M06 tool change and a G43 compensation command for the first tool used, near the beginning of the G-Code program.
- FlashCut does not cancel tool offset when it finishes processing a G-Code file. **TO AVOID CRASHES, IT IS VERY IMPORTANT THAT YOU UPDATE THE CURRENT TOOL DROP-DOWN MENU, WHENEVER YOU MANUALLY CHANGE A TOOL.**
- The G43, G44 and G49 commands are modal, so the current tool offset remains active until FlashCut executes another one of these three commands, or you set the Compensation Mode pull-down menu.
- It's good practice to use the G43 command in the line directly following the M06 command.

The following example illustrates proper use of the tool change and tool length compensation commands.

Milling Mode Example:

There are two basic methods for setting Program Zero, and both will be covered in this example.

- Use G54-G59 to apply a fixture offset ("Fixture Offset Method"). This method requires Machine Zero and the correct fixture offset to be set.
- Touch the tip of the tool to the top of the workpiece and click the Program Coordinates Zero button on the main screen ("Tool Touch-Off Method"). This method doesn't require Machine Zero or a fixture offset to be set.

In this example, the tool change position (reference point 1 on the Reference Points panel of the Configuration dialog box) is defined as machine coordinates X=0, Y=0, Z=0 (Machine Zero). Tool #1 is loaded in the machine tool, and selected in the Current Tool pull-down menu on the main screen. The Compensation Mode pull-down menu is set to Positive (G43). Machine Zero has been set. The machine tool is at Machine Zero. The tooling configuration is shown below.

Tooling					
Tool #	Description	Diameter (inches)	X Offset (inches)	Y Offset (inches)	Z Offset (inches)
1	1/4" End Mill	0.2500	0.0000	0.0000	1.5000
2	1/8" Ball End Mill	0.1250	0.0000	0.0000	2.2500
3		0.0000	0.0000	0.0000	0.0000
4		0.0000	0.0000	0.0000	0.0000
5		0.0000	0.0000	0.0000	0.0000
6		0.0000	0.0000	0.0000	0.0000
7		0.0000	0.0000	0.0000	0.0000
8		0.0000	0.0000	0.0000	0.0000
9		0.0000	0.0000	0.0000	0.0000

Program Zero is set as follows:

Tool Touch-Off Method:

Program zero is set at machine coordinate X=1, Y=1, Z=-4, with the tip of Tool #1 touching the top surface of the workpiece.

Fixture Offset Method:

Program zero is temporarily set at Machine Zero. The fixture offset for G54 is set to X=1, Y=1, Z=-5.5, A=0 in the Fixture Offsets panel of the Configuration dialog box. The fixture offset is the machine coordinate when the spindle nose touches the top surface of the workpiece. When the G54 command in the G-Code file is executed, Program Zero will be set at its associated offset value.

The G-Code file is as follows:

G54	<i>Sets Program Zero (Fixture Offset Method Only). Program coordinates change to X=-1, Y=-1, Z=4</i>
G28	<i>The machine is already at the tool change position so nothing happens</i>
M05	<i>Turns off the spindle</i>
M06 T1	<i>Changes the tool to Tool #1, which is already loaded</i>
G43 H1	<i>Applies tool length compensation for Tool #1. Since Tool #1 was already loaded, and Compensation Mode was set to Positive (G43) when the program started, this command doesn't do anything. Program coordinates and machine coordinates remain unchanged</i>
M03	<i>Turns on the spindle</i>
G29 X0 Y0 Z.25	<i>Moves the X and Y axes across, then the Z axis down to program coordinates X=0, Y=0, Z=0.25, machine coordinates X=1, Y=1, Z=- 3.75</i>

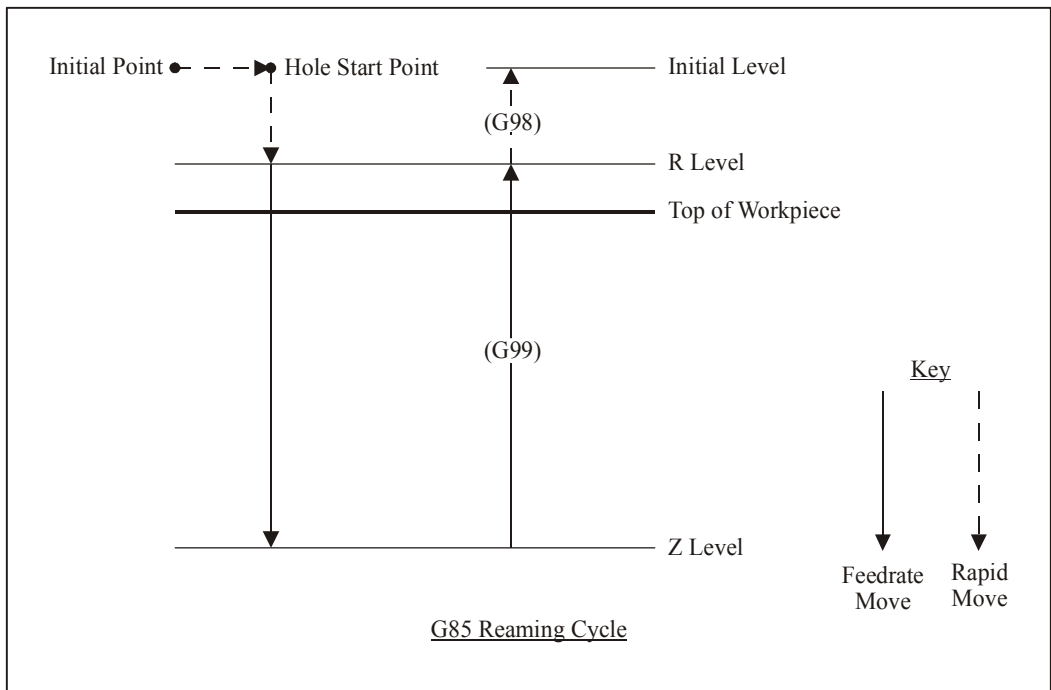
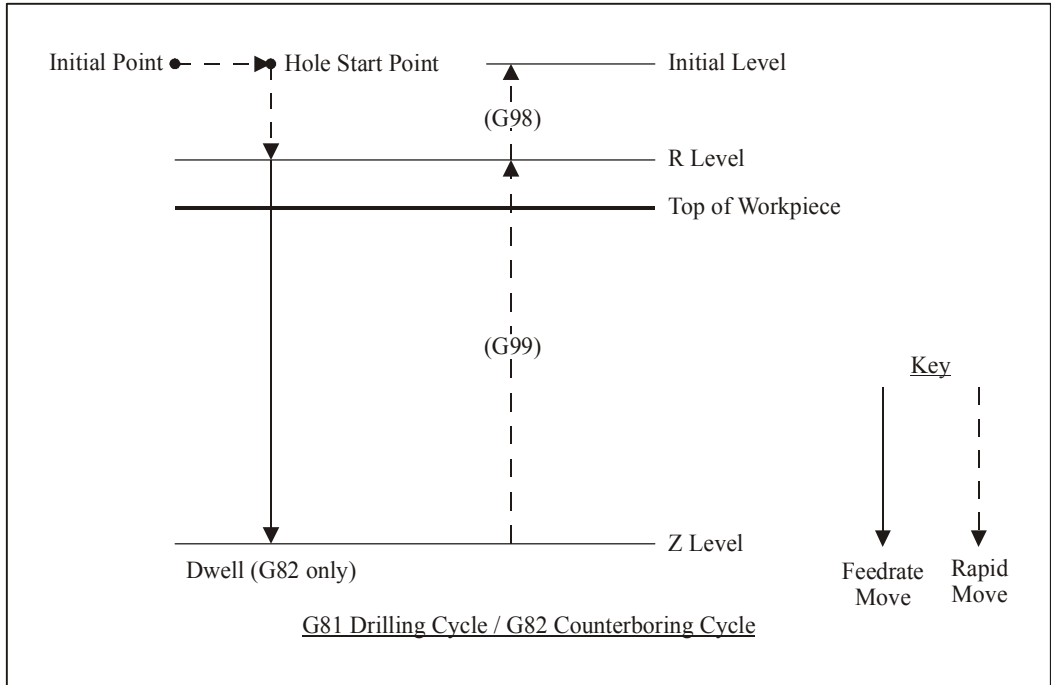
G01 Z-1.0 F10	<i>Moves the Z axis down to program coordinates X=0, Y=0, Z=-1, machine coordinates X=1, Y=1, Z=-5</i>
G01 X3 Y3 Z-1 F20	<i>Linear interpolation to program coordinates X=3, Y=3, Z=-1, machine coordinates X=4, Y=4, Z=-5</i>
G28	<i>Moves the Z axis up, then the X and Y axes across to the tool change position, program coordinates X=-1, Y=-1, Z=4, machine coordinates X=0, Y=0, Z=0</i>
M05	<i>Turns off the spindle</i>
M06 T2	<i>Changes the tool to Tool #2</i>
G43 H2	<i>Applies tool length compensation for Tool #2. The Z program coordinate shifts by -0.750 (1.500 - 2.250). The program coordinates are now X=-1, Y=-1, Z=3.25. The machine coordinates remain unchanged at X=0, Y=0, Z=0</i>
M03	<i>Turns on the spindle</i>
G29 X0 Y0 Z0.25	<i>Moves the X and Y axes across, then the Z axis down to program coordinates X=0, Y=0, Z=0.25, machine coordinates X=1, Y=1, Z=-3.0</i>
G01 Z-1.0 F10	<i>Moves the Z axis down to program coordinates X=0, Y=0, Z=-1, machine coordinates X=1, Y=1, Z=-4.25</i>
G01 X3 Y3 Z-1 F20	<i>Linear interpolation to program coordinates X=3, Y=3, Z=-1, machine coordinates X=4, Y=4, Z=-4.25</i>
G28	<i>Moves the Z axis up, then the X and Y axes across to the tool change position, program coordinates X=-1, Y=-1, Z=3.25, machine coordinates X=0, Y=0, Z=0</i>
M05	<i>Turns off the spindle</i>

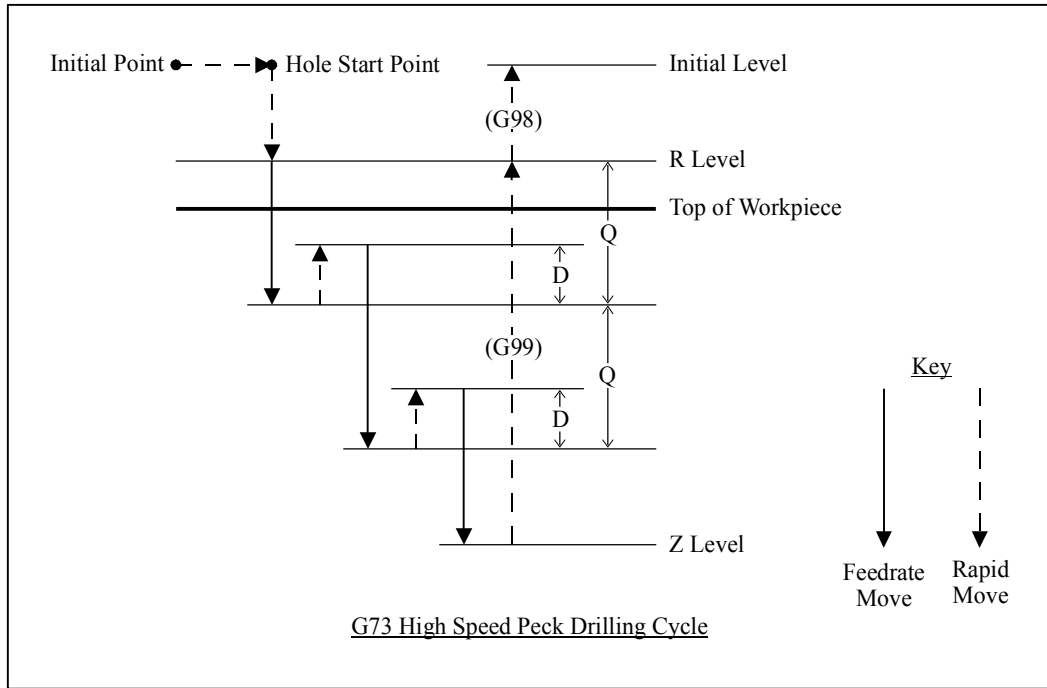
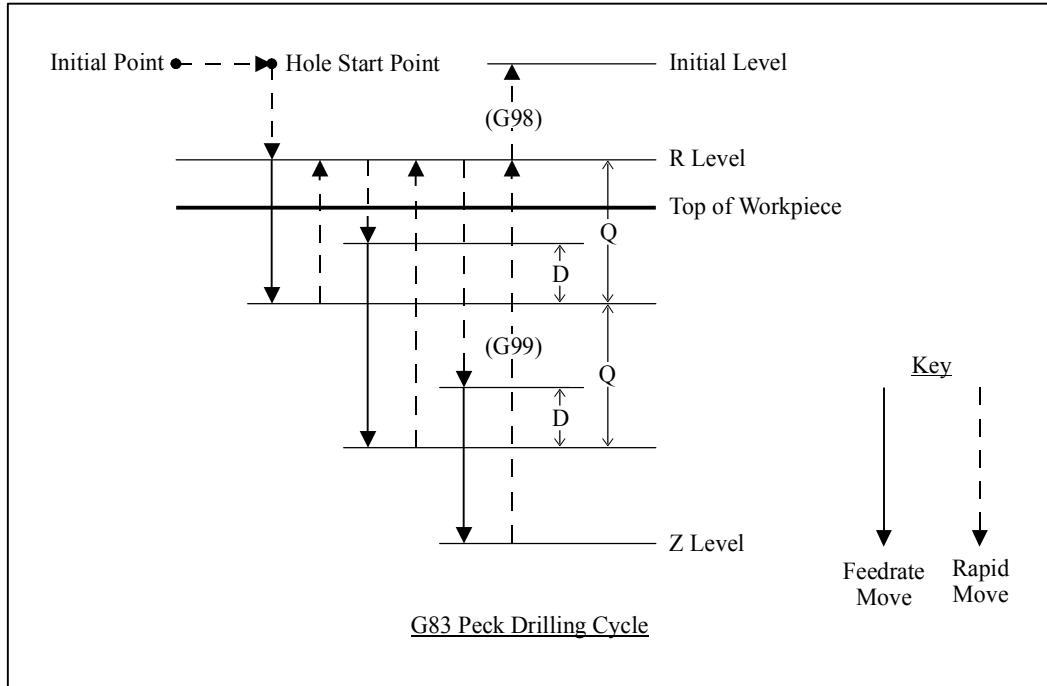
G73, G80, G81, G82, G83, G85, G98, G99 Drilling Canned Cycle Commands

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

- 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. The rest of this section refers to these diagrams.





The general command syntax for 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, in 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 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 drilling cycle command (before or after the drilling cycle command and all parameters).

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

Example (assumes D is 0.050 in.):

<pre>G98 G00 X0.0 Y0.0 Z1.0 G83 X1.0 Y2.0 R0.1 Z-1.0 Q0.5 F8.0</pre>	<p><i>Optional</i></p> <p><i>Moves the tool to the Initial Point, program coordinate X=0.0, Y=0.0, Z=1.0 (not required, for illustration only)</i></p> <p><i>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)</i></p>
---	---

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

- G73, G76 (lathe thread cutting), G80, G81, G82, G83 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 its R, Z, Q, P, F, and L 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):

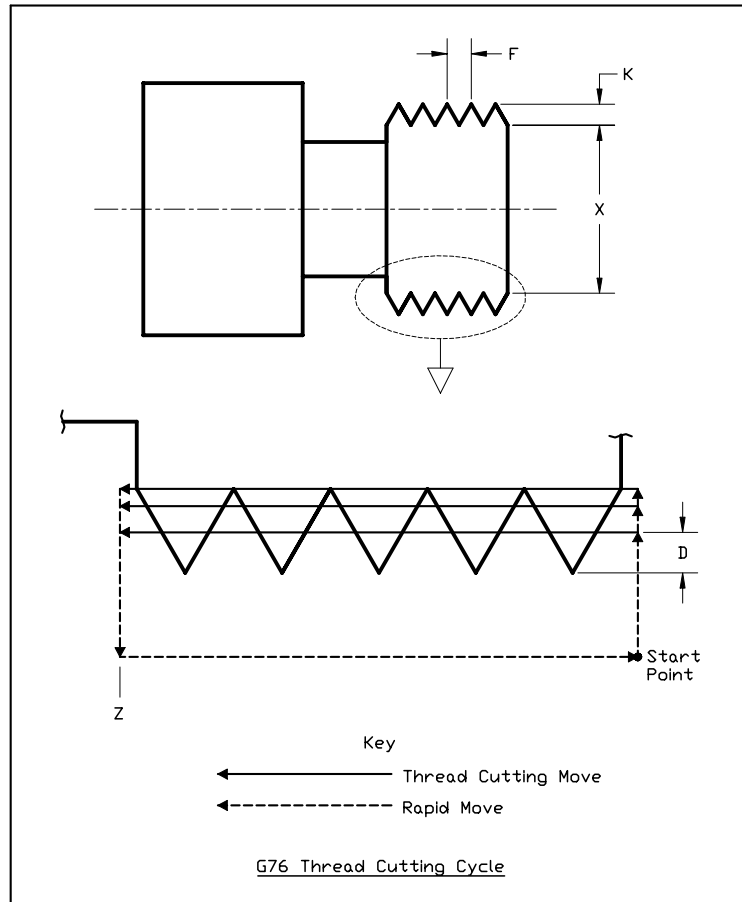
```
G81 X1.0 Y2.0 R0.25 Z-0.5 F5
X2.0
X3.0
```

- 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.

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:

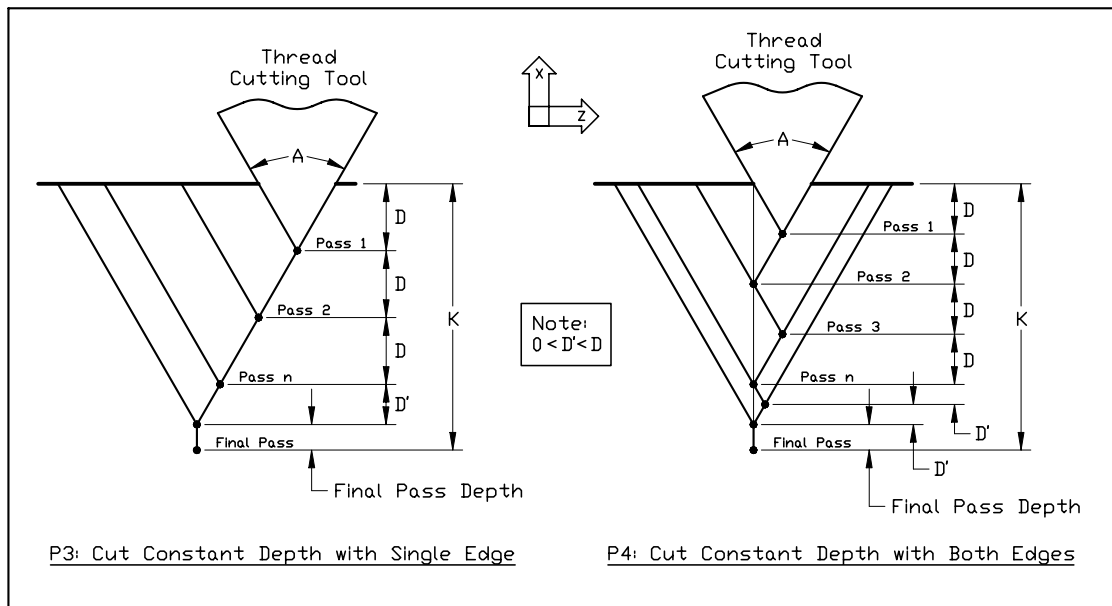
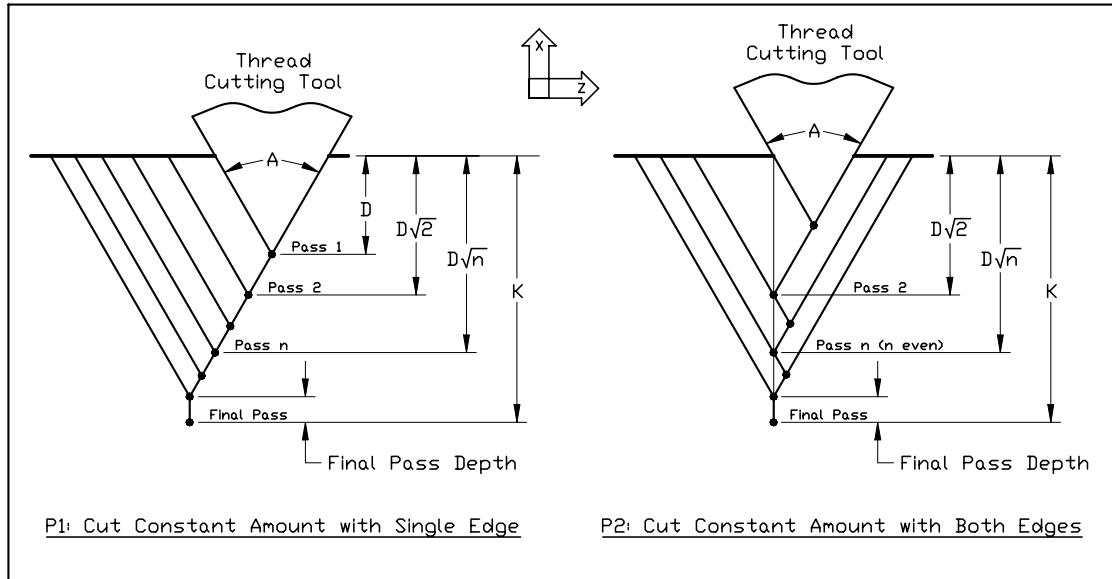
G76 Xx Zz Kk Dd Ff Aa Ii Pp

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, eg. '2000' means 0.2000 inches)
- f: Thread pitch
- a: Tool angle (determines lead-in angle, default is zero)
- i: Amount of taper (difference between initial radius and ending radius, default is zero)
- p: Cutting method (values 1-4, sets strategy for successive plunges, default is '1')

The required parameters are Z, X, K, D, F. 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 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 XYZA 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 XYZA 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	<i>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.</i>
G90	<i>All XYZA coordinates after this command will be interpreted as program coordinates</i>
G01 X1.0 Y2.0 Z-0.5	<i>Moves the tool to program coordinates X=1.0, Y=2.0, Z=-0.5</i>

M00 Program Pause

The M00 command pauses processing of the G-Code program. By default, FlashCut displays a dialog box to inform you that it has paused processing. You can control whether or not this dialog box appears using the Message on M00 Program Pause checkbox in the G-Code panel of the Configuration dialog box.

If you want to display a specific message to the operator, you can add the message in quotes after the M00:

```
M00 "[message]"
```

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	<i>First line of main program</i>
M98 Pmysub	<i>Jump to subroutine "mysub"</i>
G01 X0 Y0	<i>Continued execution after "mysub" ends</i>
M02	<i>End of main program</i>
Omysub	<i>First line of the subroutine called "mysub"</i>
G01 X2 Y2	<i>Continued execution within the subroutine</i>
M99	<i>End of subroutine "mysub"</i>

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. Subroutine names may use any number of alphanumeric 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	<i>Execute subroutine "mysub" 10 times</i>
----------------	--

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. These files are called **included files**, and you may use as many as you like. You must list each included file 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
```

Notice that the syntax for calling the included subroutines is the same as for calling subroutines listed in the main G-Code file.

You may create as many subroutines as you like in the main G-Code file and all included files, but each must have a unique name.

FlashCut's support for included files lets you create a library of reusable subroutines that you can apply to various projects and jobs.

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 in 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, in 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. You can set the M100/M101 Debounce time in the G-Code 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 in 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.

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 up to 50 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 (eg. 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 Set Feedrate Command

The F command specifies the feedrate. The feedrate set with the F command is modal (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
```

For any move that includes the A (rotary) axis, specify the feedrate either as a linear feedrate (length/minute) or rotary feedrate (degrees/minute), depending on your setting in the G-Codes panel of the Configuration dialog box. Please see the detailed explanation of rotary feedrates under “G01 Linear Interpolated Feedrate Move” above.

S Set Spindle Speed Command

The S command sets the spindle speed to the indicated value (specified in RPM).

Example:

```
S1000                             Sets the spindle speed to 1000 rpm
```

See “Analog Output Line Settings” in Initial Setup for details on how FlashCut converts the desired RPM to an analog 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
G00 X3 Y5
```

This line not always executed

Advanced Programming Reference

FlashCut provides a number of advanced programming features, allowing you to create more flexible and powerful G-Code programs.

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 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:

- `#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.
- `#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.
- `#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).
- `#TlChgPowerInputLine` – This variable represents the setting of the Input Line text box on the Tool Change panel of the Configuration dialog box.
- `#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.
- #ProgZeroOff – This variable represents the offset from the machine coordinate system to the program coordinate system.
- #ToolOff – This variable represents the tool offset currently in effect.

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

#Machine, #Program, #ProgZeroOff, and #ToolOff are runtime variables. This means that their values are not accessible in a G-Code program. Rather, these variables are intended to be used in the Tool Length Formula (on the Tool Length Sensing panel of the Configuration dialog box).

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.

User variables are defined and maintained by the G-Code programmer. In order to use a User 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 (ie. numbers without decimal points). They can be positive or negative (or zero).
- REAL – These are rational (or fractional) numbers (ie. 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`, that 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
#X2 = #X1
#ToolNumber = #CurrTool
```

Value of #X1 becomes 3.14

Value of #X2 becomes 3.14

Value of #ToolNumber becomes the currently loaded tool (note that #CurrTool is a system variable)

Some system variables represent objects, rather than simple types. The system variables in this category are #Machine, #Program, #ProgZeroOff, 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{#CurrTool} = 0.0
#YPositions{#CurrTool} = 2.0
```

In the above example, if no tool is loaded (ie. #CurrTool = 0), an error will result because the lowest index number is 1. Similarly, if #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 #TIRackPos 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 #TIRackPos 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
 - Z – the Z 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 #TIRackPos 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 #CurrTool and #NextTool are INTEGERS, you could display the description of the currently loaded tool by typing:

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

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

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

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

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 " & #CurrTool
```

Here, the system variable #CurrTool is automatically converted to a STRING. If #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 " & "#CurrTool"
```

will cause the following message to be displayed:

The currently loaded tool is #CurrTool

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

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

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.

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

In this example, if #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 #NextTool = 1 THEN
    M00 "Please load tool #1"
ELSEIF #NextTool = 2 THEN
    M00 "Please load tool #2"
ELSEIF #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 #CurrTool > 0 THEN
  IF #CurrTool = 1 THEN
    M00 "The first tool is loaded"
  ELSEIF #CurrTool = 2 THEN
    M00 "The second tool is loaded"
  ENDIF
ELSE
  IF #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

```

Additional Keywords

In addition to the G-Code commands supported by Flashcut, there are some additional commands that you'll specify using keywords. These keywords extend the capabilities of a G-Code program.

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.

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/Zero Button/Set Button)
- Absolute vs. relative positioning (G90/G91)
- English vs. Metric units (G20/G21 and G70/G71)
- Canned cycle return position (G98/G99)


```

OSub1
  SaveState
  G90
  ...
  RestoreState
M99

```

Do some things

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
SaveState
G90
SaveState
G91
RestoreState
RestoreState

```

Relative positioning mode

Absolute positioning mode

Relative positioning mode

This puts the program back into absolute positioning mode

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.

Built In Functions

FlashCut provides one built-in function, GetRackIndex, which lets you determine the tool rack position for a given tool. Example:

```

INTEGER #Index
#Index = GetRackIndex[2]

```

Get the tool rack position of tool #2