PMAC NC FOR MILL APPLICATION

Integrator/Software Manual

3xx-603450-xSxx

June 11, 2004
Copyright Information

© 2003 Delta Tau Data Systems, Inc. All rights reserved.

This document is furnished for the customers of Delta Tau Data Systems, Inc. Other uses are unauthorized without written permission of Delta Tau Data Systems, Inc. Information contained in this manual may be updated from time-to-time due to product improvements, etc., and may not conform in every respect to former issues.

To report errors or inconsistencies, call or email:

**Delta Tau Data Systems, Inc. Technical Support**
Phone: (818) 717-5656  
Fax: (818) 998-7807  
Email: support@deltatau.com  
Website: [http://www.deltatau.com](http://www.deltatau.com)

**Operating Conditions**

All Delta Tau Data Systems, Inc. motion controller products, accessories, and amplifiers contain static sensitive components that can be damaged by incorrect handling. When installing or handling Delta Tau Data Systems, Inc. products, avoid contact with highly insulated materials. Only qualified personnel should be allowed to handle this equipment.

In the case of industrial applications, we expect our products to be protected from hazardous or conductive materials and/or environments that could cause harm to the controller by damaging components or causing electrical shorts. When our products are used in an industrial environment, install them into an industrial electrical cabinet or industrial PC to protect them from excessive or corrosive moisture, abnormal ambient temperatures, and conductive materials. If Delta Tau Data Systems, Inc. products are directly exposed to hazardous or conductive materials and/or environments, we cannot guarantee their operation.
# Table of Contents

## INTRODUCTION

Hardware/Software Components................................................................. 1  
  Hardware Components ........................................................................... 1  
  Software Components ........................................................................... 1  

## NC 32 BIT FOR MILL APPLICATION

NC Mill Basics ......................................................................................... 3  
  Tool Motion ......................................................................................... 3  
  Tool Movement Specification ............................................................... 3  
  Axis Move Specification ....................................................................... 3  
  Feed Specification ............................................................................... 4  
  Cutting Speed Specification .................................................................. 4  
  Tool Movement Considerations ............................................................ 4  
  Coordinate Systems ........................................................................... 5  
  Machine Coordinates ......................................................................... 5  
  Program Coordinates ......................................................................... 5  
  Absolute Coordinate Positions ............................................................. 5  
  Incremental Coordinate Values ............................................................ 5  
  Reference Point .................................................................................. 6  

G Code Library ...................................................................................... 6  
  G Code Library .................................................................................. 6  

G Codes .................................................................................................. 8  
  G00 Rapid Traverse Positioning .............................................................. 8  
  G01 Linear Interpolation ....................................................................... 8  
  G01.1 Spline Interpolation ..................................................................... 8  
  G02 Circular Interpolation CW (Helical CW) .......................................... 9  
  G03 Circular Interpolation CCW (Helical Interpolation CCW) .............. 10  
  G04 Dwell ............................................................................................ 11  
  G09 Exact Stop ................................................................................... 11  
  G10 Programmable Data Input ............................................................... 11  
  G10.1 PMAC Data Input by Program ..................................................... 11  
  G17/G18/G19 (XY/ZX/YZ) Plane Selection ........................................... 12  
  G25 Spindle Detect Off ......................................................................... 12  
  G26 Spindle Detect On ......................................................................... 12  
  G27 Reference Point Return Check ...................................................... 12  
  G28 Return to Reference Point .............................................................. 13  
  G29 Return from Reference Point ......................................................... 13  
  G30 Return to Reference Point 2nd - 3rd ............................................... 13  
  G31 Move Until Trigger ....................................................................... 13  
  G40/G41/G42 Cutter Compensation ...................................................... 13  
  Compensation Requirements ................................................................. 14  
  How PMAC Introduces Compensation ................................................ 15  
  Speed of Compensated Moves .............................................................. 15  
  Treatment of Inside Corners ................................................................. 15  
  Treatment of Outside Corners .............................................................. 15  
  G43/G44/G49 Tool Length Compensation +/– and Cancel .................... 25  
  G45/G46/G47/G48 Single Block Tool Offsets ....................................... 26  
  G50/G51 Coordinate Scaling ................................................................. 26  
  G50.1/G51.1 Coordinate Mirroring ......................................................... 26  
  G52 Local Coordinate System Set ........................................................ 27  
  G53 Machine Coordinate Selection ..................................................... 27  
  G54-59 Work Coordinate System 1-6 Selection ................................... 28  
  G61 Exact Stop Mode .......................................................................... 28  
  G64 Cutting Mode ............................................................................... 28
PMAC NC for Mill Application

G65 MACRO Instruction ........................................................................................................... 29
G68/G69 Coordinate System Rotation .................................................................................... 29
G70 Bolt Hole Circle Pattern ................................................................................................... 29
G70.1 Bolt Hole, Center Hole Ignore Pattern .......................................................................... 30
G71 Arc Pattern ...................................................................................................................... 30
G72 Bolt Line Pattern ............................................................................................................... 31
G80-89 canned Cycles ............................................................................................................. 31
G80 Canned Cycle Cancel ........................................................................................................ 32
G81 Drilling Cycle .................................................................................................................... 32
G82 Boring, Spotfacing, Counter Sinking Cycle (Free Cutting) ............................................. 33
G83 Deep Hole (Peck) Drilling Cycle ...................................................................................... 34
G84 Tapping Cycle .................................................................................................................. 35
G85 Reaming, Boring Cycle .................................................................................................... 36
G87 Boring Cycle (Manual or Programmed Quill Return) ...................................................... 38
G88 Boring Cycle (Free Cutting, Manual or Programmed Quill Return) ............................... 39
G89 Boring Cycle (Finishing Cut, Free Cutting) .................................................................... 39
G90/G91 Absolute/Incremental Mode ..................................................................................... 40
G90.1/G91.1 Arc Radius Abs/Inc Mode .................................................................................. 40
G92 Work Coordinate System Set .......................................................................................... 41
G93 Inverse Time Feed .......................................................................................................... 41
G94/G95 Feed Per Min/Feed Per Rev .................................................................................... 42
G98/G99 Canned Cycle Return Point ..................................................................................... 42
M00 Program Stop .................................................................................................................. 42
M01 Optional Stop .................................................................................................................. 43
M02 Program Rewind .............................................................................................................. 43
M03 Spindle Clockwise .......................................................................................................... 43
M04 Spindle Counterclockwise ............................................................................................. 43
M05 Spindle Stop ................................................................................................................... 43
M06 Tool Change .................................................................................................................... 43
M08 Coolant On ..................................................................................................................... 44
M09 Coolant Off .................................................................................................................... 44
M19 Spindle Orient ................................................................................................................ 44
M30 End of Program (and Rewind) ....................................................................................... 44
M87 Start Data Gathering ...................................................................................................... 44
M88 End Data Gathering ....................................................................................................... 44
M98 Subroutine Call .............................................................................................................. 45
M99 Return from Subroutine ............................................................................................... 46
T-Codes ................................................................................................................................. 48
T-Code Format ....................................................................................................................... 48
Miscellaneous ........................................................................................................................ 48
  Block Delete Character: / .................................................................................................. 48

PARAMETRIC PROGRAMMING ................................................................. 49
Introducing Parametric Programming ...................................................................................... 49
  Example Programs .............................................................................................................. 49
Parametric Subroutines .......................................................................................................... 52
  Variables ............................................................................................................................. 53
  Expressions ......................................................................................................................... 59
Program Control ..................................................................................................................... 62
  Formatted Output .............................................................................................................. 64
  Parameter Display .......................................................................................................... 65
M Code Aliasing ..................................................................................................................... 66
  General Concepts ............................................................................................................ 66
  Aliasing M Codes ........................................................................................................... 66
Integration ............................................................................................................................... 68

TOUCH PROBING FOR PMAC-NC .................................................................................. 69
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>PMAC-NC Probing Appendix</td>
<td>69</td>
</tr>
<tr>
<td>Hardware Integration Manual</td>
<td>69</td>
</tr>
<tr>
<td>Differences Between PMAC and PMAC2</td>
<td>69</td>
</tr>
<tr>
<td>PMAC Wiring for Skip Signal</td>
<td>69</td>
</tr>
<tr>
<td>Installing a Spindle and Table Probe</td>
<td>70</td>
</tr>
<tr>
<td>Optical Interface</td>
<td>71</td>
</tr>
<tr>
<td>Software Integration Manual</td>
<td>72</td>
</tr>
<tr>
<td>Probing Memory Map</td>
<td>72</td>
</tr>
<tr>
<td>Installation of Software</td>
<td>72</td>
</tr>
<tr>
<td>Files Required for Installation</td>
<td>73</td>
</tr>
<tr>
<td>Testing the Probe</td>
<td>73</td>
</tr>
<tr>
<td>Testing G31 Operation</td>
<td>74</td>
</tr>
<tr>
<td>Machining Center Operators Manual</td>
<td>74</td>
</tr>
<tr>
<td>Probing Cycles</td>
<td>74</td>
</tr>
<tr>
<td>Calling Method</td>
<td>75</td>
</tr>
<tr>
<td>Calibration</td>
<td>75</td>
</tr>
<tr>
<td>Safe Axis Movement</td>
<td>78</td>
</tr>
<tr>
<td>Measurement</td>
<td>79</td>
</tr>
<tr>
<td>Miscellaneous Macros</td>
<td>88</td>
</tr>
<tr>
<td>Alarms</td>
<td>93</td>
</tr>
<tr>
<td>Broken Tool</td>
<td>93</td>
</tr>
<tr>
<td>A Input Missing</td>
<td>93</td>
</tr>
<tr>
<td>B Input Missing</td>
<td>93</td>
</tr>
<tr>
<td>C Input Missing</td>
<td>93</td>
</tr>
<tr>
<td>D Input Missing</td>
<td>93</td>
</tr>
<tr>
<td>Data #130-#139 Missing</td>
<td>93</td>
</tr>
<tr>
<td>Format Error</td>
<td>93</td>
</tr>
<tr>
<td>G65 Address Code Missing</td>
<td>93</td>
</tr>
<tr>
<td>G65 Nesting Level Exceeded</td>
<td>93</td>
</tr>
<tr>
<td>H Input Not Allowed</td>
<td>93</td>
</tr>
<tr>
<td>M Input Not Allowed</td>
<td>93</td>
</tr>
<tr>
<td>No Feed Rate</td>
<td>93</td>
</tr>
<tr>
<td>No Tool Length Active</td>
<td>93</td>
</tr>
<tr>
<td>Path Obstructed</td>
<td>93</td>
</tr>
<tr>
<td>Probe Fail</td>
<td>93</td>
</tr>
<tr>
<td>Probe Open</td>
<td>94</td>
</tr>
<tr>
<td>Runtime Error</td>
<td>94</td>
</tr>
<tr>
<td>S Input Not Allowed</td>
<td>94</td>
</tr>
<tr>
<td>SH Input Mixed</td>
<td>94</td>
</tr>
<tr>
<td>ST Input Mixed</td>
<td>94</td>
</tr>
<tr>
<td>T Input Missing</td>
<td>94</td>
</tr>
<tr>
<td>T Input Not Allowed</td>
<td>94</td>
</tr>
<tr>
<td>TM Input Mixed</td>
<td>94</td>
</tr>
<tr>
<td>Tool Out of Range</td>
<td>94</td>
</tr>
<tr>
<td>X Input Missing</td>
<td>94</td>
</tr>
<tr>
<td>XY Input Missing</td>
<td>94</td>
</tr>
<tr>
<td>XY Input Mixed</td>
<td>94</td>
</tr>
<tr>
<td>XYZ Input Missing</td>
<td>94</td>
</tr>
<tr>
<td>XYZ Input Mixed</td>
<td>94</td>
</tr>
<tr>
<td>Y Input Missing</td>
<td>94</td>
</tr>
<tr>
<td>Z Input Missing</td>
<td>94</td>
</tr>
<tr>
<td>ZK Input Mixed</td>
<td>94</td>
</tr>
</tbody>
</table>
INTRODUCTION
Although this manual discusses NC-32 in terms of the Adv 810 Control Panel, it is equally applicable to
the Adv 800 and Adv 600, with regard to software and operation.

The manual assumes that the Control Panel package is wired to the machine already or is to be used with
the UMAC DEMO box.

Hardware/Software Components
Normally, the system is delivered with the following components installed:

Hardware Components
- Adv 810 Control Panel
- UMAC

Software Components:
- NC UI32
- CNC Auto Pilot
- Executive software to talk
- Turbo Setup program
- PMAC Plot program
NC 32 BIT FOR MILL APPLICATION

This section defines the basics of CNC Mills, and instructions for the PMAC NC 32 for Windows application. The goal of this document is to provide descriptions of the software within the required hardware environment and a detailed description of RS-274 style G-Code programming.

The default G-codes delivered with PMAC NC 32 are designed to emulate a Fanuc 10 style of G-codes. Hence a CNC program posted for a Fanuc 10 should work without any changes.

NC Mill Basics

Unlike a lathe tool which moves around the work piece to produce a shape, a rotating mill tool remains stationary while the table moves, moving the work piece around the tool. But NC programmers describe the operation of both machines in the same way, as if the tool moves around the work piece.

That is not a problem when dealing with a lathe, but this section discusses mill operation from a programmer’s perspective. So although we know that physically the table moves and not the tool, the section discusses operation in terms of tool motion.

Tool Motion

The tool moves through lines and arcs within the table boundaries as required to manufacture a part. In a working machine, the table is moved in relation to the rotating tool, so the actual table displacement will be the reverse of commanded tool motion.

Tool Movement Specification

Program commands for NC machines are called the preparatory functions, also known as G codes. The function of moving the table along straight lines and arcs is called interpolation. Preparatory functions specify the type of interpolation used. The three basic interpolation preparatory functions are:

1. Table movement along straight line: \texttt{G01}
2. Table movement along circular arc: \texttt{G02 / G03}
3. Table movement along specified trajectory: \texttt{G01.1}

Reference to the axis position word executes motion. The PMAC controller coordinates the movement of the axis motors to execute the command. In this document, the generalized form of the axis position word, \texttt{X_Y_Z_}, is used.

Axis Move Specification

The last commanded position is the starting position of a move and the final position is the commanded position. The final position may be either an absolute position (a point referenced to program zero) or a relative move (signed incremental distance from the previous point). This is specified with axis move or position words, the axis address letter followed by a numeric literal:

\texttt{N100 X5.2Y0Z-.001} (length units in. or mm.)
Feed Specification
Movement of the table at a specified speed for cutting a work piece is called the feedrate. Feedrates can be specified similarly with the feed word:

N100 F150.0 (length/time units in./min. or mm./min.)

Length units are within program control (see the G-code definitions in the next section). The machine builder sets time units.

![Tool Feedrate Example](image)

Cutting Speed Specification
The relative rotational speed of the tool with respect to the work piece during a cut is called the spindle speed. As for the CNC, the spindle speed can be specified in rpm units, using the S address letter followed by the value:

N100 S250 (rpm units)

Tool Movement Considerations
At multiple move (or block) boundaries, the CNC applies a coordinated ramp of the vector velocity into and out of the point without stopping. The result of this is called move blending. Because of blending, corners are not cut sharply. If sharp corners are required to be cut, Exact Stop or Dwell must be commanded in the block or set modally (see G04, G09, G61). This forces an in-position stop before starting the next move. In-position means that the feed motor is within a specified range about the commanded position.

![Move Blending Example](image)
Coordinate Systems
There are two types of coordinate systems. One is fixed by the machine mechanics and the other is a
relative coordinate system specified by the NC program that coincides with the part drawing. The control
is aware only of the fixed one. Therefore, to correctly cut the work piece as specified on the drawing, the
two coordinate systems must be specified at machine startup. When a work piece is set on the table, these
two coordinate systems are as follows:
- Coordinate system specified by the CNC: Machine Coordinates
- Coordinate system specified by the part: Program Coordinates

Machine Coordinates
The machine zero point is a standard reference point on the machine. The machine coordinate system is
established when the reference point return is first executed after the machine power is turned on or the
homing cycle is executed. Once the machine coordinate system is established, it is not changed. A G-
code program will not execute without the machine coordinate system being established first (i.e., all the
machine axes must be homed before a G-code program can be executed).

Program Coordinates
The Program coordinates are always within one of the Work coordinate systems, \( G54 \) through \( G59 \), and
are either absolute positions or incremental values. A Work coordinate offset, \( W_{off} \), defines the position
within the Machine coordinate space. Within the Work coordinate system, a Local coordinate offset, \( L_{off} \),
may define a Local coordinate system. When there are no Work or Local offsets in effect, or the work
coordinates are zero then the Machine and Program coordinates are the same. It is possible that the
Machine zero position is not accessible by the tool.

Absolute Coordinate Positions
The table moves to a point at the distance from zero point of the coordinate system (i.e. to the position of
the coordinate values). Specify the table movement from point A to point B by using the coordinate
values of point B.

Incremental Coordinate Values
This specifies table moves relative to the current table position. A move from point A to point B will use
the signed difference between the two points. The term Relative is also used.
**Reference Point**

Aside from Machine zero, a machine tool may need to locate other fixed positions corresponding to attached hardware (i.e. a tool changer). This position is called the reference point (which may coincide with Machine zero). The tool can be moved to the reference point in two ways either manually or automatically.

In general, manual reference point return is performed first after the machine power is turned on. Usually this is the same as the homing function, since the reference point is at a fixed offset from the Machine zero position. In order to move the tool to the reference point for tool change thereafter, the function of automatic reference point return is used.

**G Code Library**

**CNC G-Codes**

<table>
<thead>
<tr>
<th>G-Code</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>G00</strong></td>
<td>Rapid Traverse</td>
</tr>
<tr>
<td>G01</td>
<td>Linear Interpolation</td>
</tr>
<tr>
<td>G01.1</td>
<td>Spline Interpolation</td>
</tr>
<tr>
<td>G02</td>
<td>Circular Interpolation, CW</td>
</tr>
<tr>
<td>G03</td>
<td>Circular Interpolation, CCW</td>
</tr>
<tr>
<td>G02 &amp; G03</td>
<td>Helical Interpolation (X, Y, &amp; Z in the G code command line)</td>
</tr>
<tr>
<td>G04</td>
<td>Dwell</td>
</tr>
<tr>
<td>G09</td>
<td>Exact Stop Check</td>
</tr>
<tr>
<td>G10</td>
<td>Program Data Input</td>
</tr>
<tr>
<td>G10.1</td>
<td>PMAC Data Input</td>
</tr>
<tr>
<td><strong>G17</strong></td>
<td>XY Plane Selection</td>
</tr>
<tr>
<td>G18</td>
<td>ZX Plane Selection</td>
</tr>
<tr>
<td>G19</td>
<td>YZ Plane Selection</td>
</tr>
<tr>
<td>G20</td>
<td>Inch Mode</td>
</tr>
<tr>
<td>G21</td>
<td>Metric Mode</td>
</tr>
<tr>
<td>G25</td>
<td>Spindle Speed Detect Off</td>
</tr>
<tr>
<td><strong>G26</strong></td>
<td>Spindle Speed Detect On</td>
</tr>
<tr>
<td>G27</td>
<td>Reference Point Return Check</td>
</tr>
<tr>
<td>G28</td>
<td>Return To Reference Point</td>
</tr>
<tr>
<td>G29</td>
<td>Return From Reference Point</td>
</tr>
<tr>
<td>G30</td>
<td>2nd Reference Point Return</td>
</tr>
<tr>
<td>G31</td>
<td>Move Until Trigger</td>
</tr>
<tr>
<td><strong>G40</strong></td>
<td>Cutter Compensation Cancel</td>
</tr>
<tr>
<td>G41</td>
<td>Cutter Compensation Left</td>
</tr>
<tr>
<td>G42</td>
<td>Cutter Compensation Right</td>
</tr>
<tr>
<td>G43</td>
<td>Tool Length Compensation, + Direction</td>
</tr>
<tr>
<td>G44</td>
<td>Tool Length Compensation, - Direction</td>
</tr>
<tr>
<td>G45</td>
<td>Tool Offset Increase</td>
</tr>
<tr>
<td>G46</td>
<td>Tool Offset Decrease</td>
</tr>
<tr>
<td>G47</td>
<td>Tool Offset Double Increase</td>
</tr>
<tr>
<td>G48</td>
<td>Tool Offset Double Decrease</td>
</tr>
<tr>
<td><strong>49</strong></td>
<td>Tool Length Compensation Cancel</td>
</tr>
<tr>
<td><strong>G50</strong></td>
<td>Scaling Cancel</td>
</tr>
<tr>
<td><strong>G51</strong></td>
<td>Scaling</td>
</tr>
<tr>
<td>G50.1</td>
<td>Mirror Cancel</td>
</tr>
<tr>
<td>G51.1</td>
<td>Mirror Image</td>
</tr>
<tr>
<td>G52</td>
<td>Local Coordinate System Setting</td>
</tr>
<tr>
<td>G53</td>
<td>Machine Coordinate System Setting</td>
</tr>
</tbody>
</table>
### G54 - Work Coordinate System 1
### G55 - Work Coordinate System 2
### G56 - Work Coordinate System 3
### G57 - Work Coordinate System 4
### G58 - Work Coordinate System 5
### G59 - Work Coordinate System 6

<table>
<thead>
<tr>
<th>G61</th>
<th>Exact Stop Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>G64</td>
<td>Cutting Mode (Cancel Exact Stop Mode)</td>
</tr>
<tr>
<td>G68</td>
<td>Coordinate System Rotation</td>
</tr>
<tr>
<td>G69</td>
<td>Coordinate System Rotation Cancel</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>G70</th>
<th>Bolt Hole Circle Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>G70.1</td>
<td>Bolt Hole, Center Hole Ignore Pattern</td>
</tr>
<tr>
<td>G71</td>
<td>Arc Pattern</td>
</tr>
<tr>
<td>G72</td>
<td>Bolt Line Pattern</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>G80</th>
<th>Canned Cycle Cancel</th>
</tr>
</thead>
<tbody>
<tr>
<td>G81</td>
<td>Spot Drilling Canned Cycle</td>
</tr>
<tr>
<td>G82</td>
<td>Counter Boring Drilling Cycle</td>
</tr>
<tr>
<td>G83</td>
<td>Peck Drilling Cycle</td>
</tr>
<tr>
<td>G84</td>
<td>Tapping Cycle</td>
</tr>
<tr>
<td>G85</td>
<td>Fine Boring Canned Cycle</td>
</tr>
<tr>
<td>G86</td>
<td>Boring Canned Cycle</td>
</tr>
<tr>
<td>G87</td>
<td>Back Boring Canned Cycle</td>
</tr>
<tr>
<td>G88</td>
<td>Reverse Tapping Canned Cycle</td>
</tr>
<tr>
<td>G89</td>
<td>Canned Cycle Recall</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>G90</th>
<th>Absolute Command Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>G91</td>
<td>Incremental Command Mode</td>
</tr>
<tr>
<td>G90.1</td>
<td>Arc Radius Abs/Inc Mode</td>
</tr>
<tr>
<td>G91.1</td>
<td>Arc Radius Abs/Inc Mode</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>G92</th>
<th>Absolute Zero Point Programming</th>
</tr>
</thead>
<tbody>
<tr>
<td>G92.1</td>
<td>Absolute Zero Point Programming Cancel</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>G93</th>
<th>Inverse Time Feed</th>
</tr>
</thead>
<tbody>
<tr>
<td>G94</td>
<td>Feed Per Minute</td>
</tr>
<tr>
<td>G95</td>
<td>Feed Per Revolution</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>G98</th>
<th>Return To Initial Point in Canned Cycle</th>
</tr>
</thead>
<tbody>
<tr>
<td>G99</td>
<td>Return to R Point in Canned Cycle</td>
</tr>
</tbody>
</table>
G Codes

G00 Rapid Traverse Positioning
This is used to position the tool from the current programmed point to the next programmed point at maximum traverse rate for all axes. **G00** is group 01 modal. It is canceled by other group 01 functions. The rapid move is not axis coordinated. Each axis has a different endpoint velocity ramp. Each axis may also have a different maximum traverse rate. The axis with the longest move time (move distance/axis velocity) will finish last and provide the final in-position for end of block registration. Rapid moves are never blended with adjacent blocks.

**Syntax:**  
G00X_Y_Z_

**Example Code:**
N005 G49 G54 G20 G90 G40 G80  
N010 S2500 M03  
N015 G55  
N020 G20 G90 G0 X0 Y0  
(inch, abs, rapid to work piece x,y zero psn)

G01 Linear Interpolation
Linearly interpolates the position of the tool from the current point to the programmed point in the **G01** block. Segmentation control for all interpolation is controlled by the PMAC I13 parameter. The speed of the tool is controlled by the modal feedrate word **F** and is the vector velocity of the tool path defined by:

$$F_x = F \times \frac{L_x}{\sqrt{L_y^2 + L_x^2}}; \quad F_y = F \times \frac{L_y}{\sqrt{L_y^2 + L_x^2}}$$

Linear moves may blend with adjacent interpolative blocks. If the **G01** block contains a Dwell (**G04**) or an Exact Stop (**G09**), a controlled deceleration to a stop with in-position going true will inhibit blending with the next block. If the **G61** modal Exact Stop is active, no blending between linear blocks will occur until canceled (**G64** Cutting Mode). **G01** is group 01 modal. It is canceled by other group 01 functions.

**Syntax:**  
G01X_Y_Z_F_

**Example Code:**
N030 X1.125 Y2.25  
N040 G61 G1 Z-.02 F20  
(exact stop mode, linear, plunge cutter, 20 ipm)  
N050 G64 G3 X0.5 Y2.0 R0.375  
G01.1 Spline Interpolation
Interpolates as a three point Cubic Spline, a segmented profile (trajectory of points) with no change in acceleration at segment boundaries (smooth contouring). A fixed move time of **R/F** for all segments is specified indirectly with a segment size and feedrate in the initial **G01.1** block with **R** and **F**, respectively. Actual commanded velocities, a result of the Spline calculations, are smooth. Accelerations are matched at segment boundaries. Subsequent blocks are blended to fit a 3-point cubic Spline with the adjacent blocks until dwell, new segment word (**R**) or modal change (i.e. **G00** or **G01**). Zero length intervals of **R/F** time units are added at the endpoints to facilitate entry and exit of the Spline. The PMAC segmentation parameter I13 does not effect Spline mode.

Intermediate positions are relaxed somewhat to meet the velocity and acceleration constraints imposed and may be calculated from the following equation:

$$\text{err}_n = \frac{[\text{Seg}_{n+1} - \text{Seg}_n]}{6}$$

It applies to vector sum of axis components, for simultaneous multiple axis splines. If a segment size within the block sequence deviates from that specified in the initial block **R** word, then the above equation gives the error amount. **G01.1** is modal in-group 01. It is canceled by other group 01 functions.

**Syntax:**  
G01.1R_X_Y_Z_F_
Example Code:
N6 Z.1 H1 M8
N7 G1.1 R.05 F150. (spline mode seg size of .05 in at 150 ipm)
N8 X10Y10   (point 1)
N9 X10.2236Y10.2236 (point 2)
N10 X10. 0.4729Y10. 0.4729 (point 3)

**G02 Circular Interpolation CW (Helical CW)**
Circular interpolation uses the axis information contained in a block to move the tool in a clockwise arc of a circle, up to 360 degrees. The velocity at which the tool is moved is controlled by the feedrate word and is a vector tangent in the interpolation plane:

\[ F_t = \sqrt{f_x^2 + f_y^2} \]

All circles are defined and machined by programming three pieces of information to the PMAC. They are:
- Start Point of the arc
- End Point of the arc
- Arc Center of the arc or Arc Radius

The Start Point is defined prior to the G02 line, usually by a G01 or G00 positioning move. The End Point is defined by the axis coordinates within the G02 line. The Arc Center is defined by the I, J, and K values (vector incremental from the start point) or the R value within the G02 line. The full format for a G02 line must reflect in which plane the arc is being cut. This is accomplished by use of a G code to define the interpolation plane and the letter addresses I, J, and K.

- G17 (XY - Plane) Letter address I for X Letter address J for Y
- G18 (XZ - Plane) Letter address I for X Letter address K for Z
- G19 (YZ - Plane) Letter address J for Y Letter address K for Z

The I, J and K vector incremental values are signed distances from where the tool starts cutting (Start Point) the arc to the Arc Center. For 90-degree corners or fillets, the I, J and K values can be determined easily. The G17 (XY - Plane) is the default or power on condition. If another axis not specified in the circular interpolation is programmed then helical cutting will be affected. The feedrate of the linear axis will be:

\[ F = \frac{F \times (length \ of \ linear \ axis)}{length \ of \ arc} \]

**Syntax:**

\[
\begin{align*}
\text{G17/G18/G19} & \ G02 \ _X_\ Y_\ Z_\ I_\ J_\ K_\ F_\ \_ \\
\text{G17/G18/G19} & \ G02 \ _X_\ Y_\ Z_\ R_\ F_\ \_
\end{align*}
\]

**Example Code:**
N040 G73 G1 Z-.02 F20
N050 G64 G2 X0.5 Y2.0 R0.375   (cut mode, cw circle)
N060 G1 Y1.5625

---

**Circular Interpolation Example**
G03 Circular Interpolation CCW (Helical Interpolation CCW)
Circular contouring control uses the axis information contained in a block, to move the tool in a counterclockwise arc of a circle, up to 360 degrees. The velocity at which the tool is moved is controlled by the feedrate word and is vector tangential \( F_t = \sqrt{f_x^2 + f_y^2} \). All circles are defined and machined by programming three pieces of information to the control:

- Start Point of the arc
- End Point of the arc
- Arc Center of the arc

The Start Point is defined prior to the G03 line, usually by a G01 linear positioning move. The End Point is defined by the X- and Y-axis coordinates within the G03 line when in the XY - Plane. The Arc Center is defined by the I, J and K values (vector incremental from the start point) when in the X-Y - Plane, or the R value within the G03 line. The full format for a G03 line must reflect in which plane the arc is being cut. This is accomplished by use of a G code to define the plane and the letter addresses I, J, and K.

- G17 (XY - Plane) Letter address I for X Letter address J for Y
- G18 (XZ - Plane) Letter address I for X Letter address K for Z
- G19 (YZ - Plane) Letter address J for Y Letter address K for Z

The I, J and K vector incremental values are signed distances from where the tool starts cutting (Start Point) the arc to the Arc Center. For 90-degree corners or fillets, the I, J and K values can be determined easily. The G17 (XY - Plane) is the default or power on condition.

If another axis not specified in the circular interpolation is programmed, then helical cutting will be affected. The feedrate of the linear axis will be:

\[ F \times \text{(length of linear axis / length of arc)} \]

**Syntax:**

- \[ [G17/G18/G19]G03X_Y_I_J_F_ \]
- \[ [G17/G18/G19]G03X_Y_R_F_ \]

**Example Code:**

```
N4 G0 G90 G17 S500 M3
N5 X0 Y1.0156
N6 Z1 H1 M8
N7 G03 I1 J1 Y0 X2 F150.
```
G04 Dwell
When programmed in a block following some motion such as G00, G01, G02 or G03, all axis motion will be stopped for the amount of time specified in the F, P or X word in seconds. Only axis motion is stopped; the spindle and machine functions under PLC control are unaffected. The numerical range is from .001 to 99999.999 seconds. If no parameter is specified then a default value of 0 seconds dwell is executed.
Syntax: G04X_

Example Code:
N4 G0 G90 S500 M3
N5 X0 Y1.0156
N6 Z.1 H1 M8
N7 G04 X10 (dwell 10 seconds)
N8 G04 P0.055 (dwell 0.055 seconds)

G09 Exact Stop
This forces a controlled deceleration to a stop, with in-position registration, at the end of the block. This is used to prevent move blending with the next block (i.e. sharp corners are cut). G09 is not modal. It is valid for the current block only and is affected by issuing a dwell of zero time (see G73 for modal Exact Stop).
Syntax: G09

Example Code:
N030 X1.125 Y2.25
N040 G73 G1 Z-.02 F20 (exact stop mode, linear, plunge cutter, 20 ipm)
N050 G64 G3 X0.5 Y2.0 R0.375

G10 Programmable Data Input
R_, IP_ represents the X, Y, Z, U, V, W, A, B or C offsets to change. R is used when changing a cutter compensation value. An axis value must be present to get its offset value altered. If the current mode is in incremental, the offsets are incremented by the IP_ point. Otherwise, the IP_ is substituted for the current offset. P represents the offset to change as follows. If P is between 1 and 6 the offset changed is the work offset. P1 will change G54, P2 will change G55 up through P6 for G59.

Example Code:
G10 P1 X-10 Z-30 (Set G54 offsets)
G10 P2001 X-0.001 Z-0.001 R0.025 (Set tool wear offsets & cutter comp)
G10 P1001 X-5 Z-5 (Set tool geometry offsets and cutter comp)

G10.1 PMAC Data Input by Program
Allows configuration of PMAC I-variables from NC program. The I, Q, and P are addresses for the PMAC like variables. The K address holds the value.
Syntax: G10.1 I_ K_  G10.1 Q_ K_  G10.1 P_ K_

Example Code:
G10.1 I125 K1228804 (I125=1228804)
G10.1 Q10 K8 (Q10=8)
G10.1 P11 K7 (P11=7)
G17/G18/G19 (XY/ZX/YZ) Plane Selection
When cutting motion is for X and Y using circular interpolation, the G17 plane must be in effect. The G17 plane is a power on default, so normally is not programmed. When cutting motion is for Z and X using circular interpolation, the G18 plane must be in effect. When cutting motion is for Y and Z using circular interpolation, the G19 plane must be in effect.

Syntax: G17/G18/G19

Example Code:
N4 G0 G90 G17 S500 M3
N5 X0 Y1.0156
N6 Z.1 H1 M8
N7 G03 I1 J1 Y0 X2 F150

G25 Spindle Detect Off
G25 sets the system flag, SPND_SPEED_DETECT, false. This will be interpreted by the CNC as cancellation of Spindle Speed detect. This will make the CNC program disregard whether the spindle is at speed.

Syntax: G25

G26 Spindle Detect On
G26 sets the system flag, SPND_SPEED_DETECT, true. The CNC will prevent the next block from executing until spindle rpm's are within a specified percentage of the commanded value.

Programmer’s note: This is reported via system flags: CS_SPND_AT_SPEED and CS_SPND_AT_ZERO.

Syntax: G26

G27 Reference Point Return Check
G27 positions the tool at rapid traverse to the optional intermediate point (ip) and then the reference point. The ip is saved for subsequent use by G29.

Syntax: G27 (X__Y__Z__)

Example Code:
N4 G0 G90 S500 M3
N5 G27 X0 Y1.0156 Z.1
**G28 Return to Reference Point**
The tool is returned to the reference point via an intermediate point (ip) specified in the block. The ip is saved for subsequent use by G29.

**Syntax:**  
G28(X__Y__Z__)

**Example Code:**
N4 G0 G90 S500 M3  
N5 G74 X0 Y1.0156 Z.1

**G29 Return from Reference Point**
The tool is moved to the point specified in the block via the ip stored by G28/G27.

**Syntax:**  
G29X__Y__Z__

**G30 Return to Reference Point 2nd - 3rd**
Implementation may be machine dependent. The system integrator provides this functionality. In general, the tool is moved to the second reference point (the P address) via the ip specified in the block. The ip is saved for subsequent use by G29.

**Syntax:**  
G30P__(X__Y__Z__)

**G31 Move Until Trigger**
Does a move until trigger (skip). When the skip signal occurs, the move will decelerate to a stop and the location at which the skip signal occurred will be stored for later use.

**Syntax:**  
G65 P9810 X__Y__F__

**G40/G41/G42 Cutter Compensation**
While cutting the programmed contours of lines and curves, being dependent on the direction of cutting and spindle rotation, the operator must keep the tool consistently oriented to the cutting surface, at the offset needed to maintain the depth of cut and surface finish called for in the print. Usually calculations involving moving surface normals and curve tangencies are required. Cutter radius compensation will provide cutter orientation and tool offset automatically.

The control will offset the tool normal to the instantaneous surface tangent of the work piece with respect to the direction of tool motion in the compensation plane. This allows a programmer to compensate for cutters of different radial dimensions without the need for complex trigonometric code changes. Climb milling will use G41 to instate cutter radius compensation. Conventional milling will use G42 to instate cutter radius compensation. Of greatest concern is how to position the tool just prior to the start up of cutter radius compensation. PMAC-NC will not engage compensation unless a move having a vector component in the compensation plane is commanded.
- **G40** - Cancel cutter radius compensation
- **G41** - Cutter compensation, tool on the left of the work piece (in the feed direction)
- **G42** - Cutter compensation, tool on the right of the work piece (in the feed direction)

When activating cutter compensation (**G41/G42**), care must be taken in selecting a clearance move in the compensation plane. On start up, the tool will move a vector distance equal to the offset value + the initial compensation in-plane move. The tool must be positioned so that as the compensation engages, the tool begins cutting normal to the surface. In addition, the center of the cutter must be at least the cutter radius away from the first surface to be machined. Cutter radius compensation is modal. Once cutter radius compensation is correctly engaged, it will remain in effect until it is canceled.

Make any zero component compensation plane axis moves before cutter compensation. Make an axis startup move, having a non-zero component in the compensation plane (**G17/18/19**), on or immediately after the G41 or G42 block. The compensation adjustment will be vectored with this move.

The programmer must consider this effect when moving out of the current plane, as in-depth changes in pocket milling. Execute a move whose vector component in the compensation plane parallel to the last in-plane compensation move, but have opposite direction is interpolated with the intended out-of-plane axis move.

When deactivating cutter compensation (**G40**), care must be taken in selecting a clearance move. If the move is omitted, the control will not cancel cutter radius compensation (and resulting axis motion) until a block with a non-zero move component in the compensation plane is executed. Do not cancel cutter compensation on any line that is still cutting the part. Cancel of cutter compensation may be a one- or two-axis move. When cutter compensation is active, the control applies a virtual cutter of zero diameter. The physical or actual diameter of the cutter is stored in the control by the operator on the page that contains the cutter tool lengths and diameters. The tool length is addressed by an **H** word, and the tool diameter is addressed by a **D** word. A tool offset number (**T** word) addresses both, using values stored in the Tools page.

Normally the tool length and the tool diameter are assigned the same tool offset number. Cutter compensation takes the stored value for the diameter and calculates the cutter path offset from that value. Because of look-ahead, care must be taken that programmed moves do not violate the called-for compensation.

**Compensation Requirements**

**Plane**

Several parameters must be specified for the compensation. First, the plane in which the compensation is to be performed must be set. Any plane in xyz-space may be specified. Executing **G17/G18/G19** does this. For example, **G17**, by describing a vector parallel to the z-axis in the negative direction, specifies the xy-plane with the normal right/left sense of the compensation. (This same command also specifies the plane for circular interpolation.)
Radius
The amount of compensation must be set using the d address and data word, as in: G42x.5d1. The units of the argument are the user units of the x, y, and z-axes. Negative and zero values for radius are possible, although not necessarily useful.

Direction
The direction of compensation is determined by the G41/G42. As mentioned above, the compensation is turned on by the rs-274 G-codes G41 and G42, respectively. The compensation is turned off by G-code G40.

How PMAC Introduces Compensation
Any change in compensation is introduced gradually and linearly over the move immediately following the change. The change could be turning compensation on, turning compensation off, or changing radius. All are treated the same – as a change in compensation radius. When compensation is off, it is effectively zero radius. When the direction of offset is changed (left to right or vice versa), the endpoint of the move is changed (extended or shortened) so that the next move will start on the proper side of the corner. The path of the move to that point is not changed. When the change in compensation is introduced over a linear move, the compensated tool path will be at a diagonal to the programmed move path. When the change in compensation is introduced over a circular arc move, the compensated tool path will be a spiral.

Speed of Compensated Moves
Tool center speed for the compensated path remains the same as that programmed by the F parameter. On an arc move, this means that the tool edge speed (the part of the tool in contact with the part) will be different from that programmed by the fraction Rtool/Rarc.

Treatment of Inside Corners
Inside corners are still subject to blending. The longer the acceleration time, the larger the rounding of the corner. (The corner rounding starts and ends a distance F*TA/2 from the compensated, but unblended corner.) The greater the portion of the blending is S-curve, the squarer the corner will be. When coming to a full stop at an inside corner, PMAC will stop at the compensated, but unblended corner.

Treatment of Outside Corners
For outside corners, PMAC introduces an arc move to cover the additional distance around the corner. The starting and ending points for the arc are points offset from the programmed corner point, perpendicular to the path on each side at the corner point, by an amount equal to the cutter radius compensation. The arc has its center at the programmed corner point. (Any outside corner with a change in angle less than 1 degree does not introduce an arc; it simply blends the offset corner using TA and TS.) When coming to a full stop at an outside corner (e.g. Step, Quit, or Dwell at the corner), PMAC includes the added arc move around the outside of the corner before stopping.

Syntax: 
G41/G42X_Y_F_D_ 
G40X_Y_F_
a) When going around an inside corner

i) Linear → Linear

ii) Linear → Circular

b) When going around the outside corner

i) Linear → Linear

ii) Linear → Circular

S = Intersection
L = Linear
C = Circular

Offset Start-up (Sheet 1 of 2)
c) When going around the outside of an acute angle

i) Linear — Linear

ii) Linear — Circular

Programmed path

Tool center path

S = Intersection
L = Linear
C = Circular

Offset Start-up (Sheet 2 of 2)
a) When going around an inside corner

i) Linear → Linear

ii) Linear → Circular

iii) Circular → Linear

iv) Circular → Circular

v) Straight line to Straight line
b) When going around an outside corner at an obtuse angle

i) Linear → Linear

ii) Linear → Circular

iii) Circular → Linear

iv) Circular → Circular

NOTE: When the change in angle is less than 1° (α > 179°), no circular segment is added. There is simply the blending from the incoming segment to the outgoing segment over one TA time.

Offset Mode (Sheet 2 of 3)
c) When going around an outside corner at an acute angle

i) Linear $\rightarrow$ Linear

iii) Circular $\rightarrow$ Linear

ii) Linear $\rightarrow$ Circular

iv) Circular $\rightarrow$ Circular

Offset Mode (Sheet 3 of 3)
i) Linear → Linear

ii) Linear → Circular

iii) Circular → Circular

iv) Circular → Circular

v) When an intersection is not obtained if offset is normally performed

i) Linear → Linear

ii) Linear → Circular

---

$r = CCR$

$S = \text{Intersection}$

$L = \text{Linear}$

$C = \text{Circular}$

$P = \text{Parabolic}$

---

Change of Offset Direction
a) When going around an inside corner

i) Linear — Linear

ii) Circular — Linear

b) When going around the outside corner

i) Linear — Linear

ii) Circular — Linear

S = Intersection
L = Linear
C = Circular

Offset Cancel (sheet 1 of 2)
c) When going around the outside of an acute angle

i) Linear — Linear

ii) Circular — Linear

d) When the tool goes around the outside linear at an acute angle less than 1 degree, compensation is performed as follows

S = Intersection
L = Linear
C = Circular

Offset Cancel (sheet 2 of 2)
i) Machining an inside corner at a radius smaller than the cutter radius

ii) Machining a groove smaller than the tool diameter

Over cutting by Cutter Compensation (Sheet 1 of 2)
iii) When machining a step smaller than the tool radius

![Diagram of circular movement and tool center path]

The PMAC does not stop, and overcutting occurs. The PMAC backs up to correct when intersection is determined (at 1 block ahead).

iv) Machining small segments

![Diagram of many small segments]

The PMAC calculates trajectory 2 blocks ahead. If the radius of compensation is sufficiently large and programmed segments are short, overcutting can occur. Turbo calculates in segments. If there are too many segments, overcutting can occur.

Overcutting by Cutter Compensation (Sheet 2 of 2)

G43/G44/G49 Tool Length Compensation +/- and Cancel

Program zero is a point of reference for coordinates in a part program, usually from a key location on the work piece. The position of the tool’s center in X and Y does not change as the tool changes. In the Z-axis, this is not the case. If the length of the tool changes, so does the distance from the tip of each tool to the program zero point in Z. Note that each tool has a different distance from the tip of the tool to a surface on the part.

Tool length compensation lets the control call out the Z-axis movements in a program as the tool changes, although, physical interference problems between the work piece and the tool must still be overcome by the programmer.
The programmer initializes tool length compensation in each tool’s first Z-axis approach move to the work piece. This initialization command includes a \texttt{G43/G44} word and an \texttt{H} or \texttt{T} word to invoke the desired tool offset. It must also contain a Z-axis positioning move. Tool length compensation is modal. Once instated it remains in effect until cancelled or changed. \texttt{G49} cancels tool length compensation that is in effect.

**Syntax:**
\[
\texttt{G43Z\_} \\
\texttt{G44Z\_} 
\]

**Example Code:**
\[
\begin{align*}
\text{N020} & \quad \text{G20 G90 G0 X0 Y0} \\
\text{N025} & \quad \text{G43 Z0.25 H1 (move to z0+.25 with tool offset comp)} \\
\text{N030} & \quad \text{X1.125 Y2.25}
\end{align*}
\]

**G45/G46/G47/G48 Single Block Tool Offsets**
Single block increase and decrease of stored tool offset. It uses the last modal H code.

- \texttt{G45}  Increase offset by stored value
- \texttt{G46}  Decrease offset by stored value
- \texttt{G47}  Increase offset by stored value X 2
- \texttt{G48}  Decrease offset by stored value X 2

**Syntax:**
\[
\texttt{G45/G46/G47/G48} 
\]

**G50/G51 Coordinate Scaling**
\texttt{G51} is coordinate scaling. \texttt{X_Y_Z\_} is the center of scaling. These parameters are meaningful only in absolute mode. \texttt{I_J_K\_} is the scaling magnification of the \texttt{X-}, \texttt{Y-}, and \texttt{Z}-axis, respectively. When performing circular interpolation specified by a Radius, the maximum value of the scaling magnification for the appropriate plane is applied to the Radius component. For example, if the selected plane is the \texttt{X-Z} plane then the maximum magnification of \texttt{X Z} is used to scale \texttt{R}. Likewise, if the selected plane is the \texttt{X-Y} plane, the maximum magnification of \texttt{Y} is used to scale \texttt{R}. When performing circular interpolation with \texttt{I J K} components, each component is magnified by its appropriate scale factor.

Coordinate scaling is canceled with \texttt{G50}.

**Syntax:**
\[
\texttt{G51X_Y_Z_I_J_K\_} \\
\texttt{G50} 
\]

**G50.1/G51.1 Coordinate Mirroring**
\texttt{G51.1} is mirroring. \texttt{X_Y_Z\_} is the axis to mirror about. The value of this parameter is meaningful only in absolute mode. It indicates the line about which mirroring occurs. In incremental mode, only the axis letter is meaningful and the actual value may be anything.

Mirroring is canceled with \texttt{G50.1}.

**G51 G-Code Values**

<table>
<thead>
<tr>
<th>Mirrored Program (Absolute Moves)</th>
<th>Positions</th>
<th>Mirrored Program (Incremental Moves)</th>
<th>Positions</th>
</tr>
</thead>
<tbody>
<tr>
<td>G1 G90 X0 Y0</td>
<td>X 0.0000  Y 0.0000</td>
<td>G1 G90 X0 Y0 G91</td>
<td>X 0.0000  Y 0.0000</td>
</tr>
<tr>
<td>G51.1 X3</td>
<td>X 0.0000  Y 0.0000</td>
<td>G51.1 X3</td>
<td>X 0.0000  Y 0.0000</td>
</tr>
<tr>
<td>X1</td>
<td>X 5.0000  Y 0.0000</td>
<td>X1</td>
<td>X -1.0000 Y 0.0000</td>
</tr>
<tr>
<td>Y1</td>
<td>X 5.0000  Y 1.0000</td>
<td>Y1</td>
<td>X -1.0000 Y 1.0000</td>
</tr>
<tr>
<td>G50.1</td>
<td>X 5.0000  Y 1.0000</td>
<td>G50.1</td>
<td>X -1.0000 Y 1.0000</td>
</tr>
</tbody>
</table>

Mirroring is canceled with \texttt{G50.1}.

**Syntax:**
\[
\texttt{G51.1X\_Y\_Z\_} \\
\texttt{G50.1} 
\]
**G52 Local Coordinate System Set**

While programming in a work coordinate system, it is sometimes more convenient to have a common coordinate system within all the work coordinate systems. This coordinate system is called a local coordinate system. The **G52** specifies the local coordinate system. The Local CS (X'Y') is offset from the Work CS (XY) by the vector (A) that makes the current tool point in the Local CS equal to the position word in the **G52** block (**G52X100Y100**). When a local coordinate system is set, the move commands in absolute mode (**G90**), which is subsequently commanded, are the coordinate values in the local coordinate system. The local coordinate system can be changed by specifying the **G52** command with the zero point of a new local coordinate system in the work coordinate system. To cancel the local coordinate system, specify **G52X0Y0**.

**Syntax:**

```
G52X__Y__Z__
```

**Example Code:**

```
N4 G0 G90 S500 M3
N5 G52 X.0157 Y1.0156 Z0
```

**Local Coordinate System Example**

---

**G53 Machine Coordinate Selection**

The machine zero point is a standard point on the machine. A coordinate system having the zero point at the machine zero point is called the machine coordinate system. The tool cannot always move to the machine zero point. The machine coordinate system is established when the reference point return is first executed after the power is on. Once the machine coordinate system is established, it is not changed by reset, change of work coordinate system (**G92**), local coordinate system setting (**G52**) or other operations unless the power is turned off. Occasionally it is necessary to move the axes to a specific position in relation to machine zero and ignore any tool and work offsets that are active. This is accomplished using **G53** for machine coordinate programming. Machine coordinates are always expressed as absolute coordinates. If the **G91** incremental mode is active, the **G53** command is ignored. All **G92** codes and offsets are ignored. The interpolation mode must be either **G00** or **G01**. The tool is moved to the absolute Machine coordinates expressed in the **G53** block.

**Syntax:**

```
G53X__Y__Z__
```

**Example Code:**

```
N4 G53 X0 Y0 Z0
```
G54-59 Work Coordinate System 1-6 Selection

Six coordinate systems proper to the machine tool are set in advance, permitting the selection of any of them by G54 to G59.

- Work coordinate system 1 G54
- Work coordinate system 2 G55
- Work coordinate system 3 G56
- Work coordinate system 4 G57
- Work coordinate system 5 G58
- Work coordinate system 6 G59

The six coordinate systems are determined by setting distances (work zero offset values) in each axis from the machine zero point to their respective zero points. The offsets are saved in the OFS page of the PMAC-NC program.

Example Code:  

```
G55G00X20.0Z100.0;
X40.0Z20.0;
```

In the above example, positioning is made to positions (X=20.0, Z=100.0) and (X=40.0, Z = 20.0) in work coordinate system 2. Where the tool is positioned on the machine depends on work zero point offset values.

Work coordinate systems 1 to 6 are established after reference point return (or homing) after the power is turned on. When the power is turned on, G54 coordinate system is selected by default.

Syntax: G54-59

G61 Exact Stop Mode

G61 causes a stop between block moves so that no corner rounding or blending between the moves is done (i.e. sharp corners are cut). When G61 is commanded, deceleration is applied to the end point of the cutting block, and the in-position check is performed every block thereafter. The G61 is valid until G64 (cutting mode) or G73 (tapping mode) is commanded. Cutting mode (G64) is the startup default.

Syntax: G61

G64 Cutting Mode

When G64 is commanded, deceleration at the end point of each block is not performed thereafter, and cutting is blended to the next block. This command is valid until G61 (exact stop mode), or G63 (tapping mode) is commanded. However, in G64 mode, feed rate is decelerated to zero and in-position check is performed in the following cases:

- Positioning mode (G00)
- Block with exact stop check (G09)
- Next block is a block without movement command

Syntax: G64
G65 MACRO Instruction
G68/G69 Coordinate System Rotation
A programmed shape can be rotated about a point. By using this function, it becomes possible, for example, to modify a program using a rotation command when a work piece has been placed with some angle rotated from the programmed position on the machine. Further, when there is a pattern comprising some identical shapes in the positions rotated from a shape, the time required for programming and the length of the program can be reduced by preparing a subprogram of the shape and calling it after rotation. The angle of rotation (+ is the CCW direction) is commanded with a signed angle value in decimal degrees using the R address in the G68 block. The center of rotation is specified in the block with axis address data; X, Y, and Z. After this command is specified, subsequent commands are rotated by the specified parameters. Command the angle of rotation (R) within the range of –360 to 360 degrees.

A rotation plane must be specified (G17, G18, G19) when G68 is designated, though not required to be designated in the same block. G68 may be designated in the same block with other commands. Tool offsets, such as cutter compensation, tool length compensation, or tool offset is performed after the coordinate system is rotated for the command program. The coordinate system rotation is cancelled by G69.

Syntax: 

G68X_Y_Z_R_
G69

Example Code:
N4 G17 G69 X1 Y1 R90 (90 Degree rotation, CCW in the XY plane, about X1Y1)

G70 Bolt Hole Circle Pattern
When commanded, the tool will first drill a center hole, and then drill holes located at points equally distributed on the circle. This G-code must be preceded by a valid canned drilling cycle (i.e., G81 ~ G88). The canned cycle G-code must precede G70 to establish the method of drilling for the pattern cycle. The X_ and Y_ parameters specified on the line containing the G81 ~ G88 determine where the center of the pattern will reside. The drilling canned cycle cannot reside on the same line as the drilling pattern cycle, G70.

Syntax: 

G70 I_ J_ L_

I: Radius of circle must be greater than 0.
J: Angle formed by X-axis and vector from center of circle to start point.
L: Number of points in the circle.

Programming Example:
G83 X_ Y_ Z_ R_ L_
G70 I3 J45 L8
G80
G84 X_ Y_ Z_ R_ L_ F_ P_ Q_
G70 I3 J45 L8
G80

The code excerpt above would first drill the center hole, then drill a hole at the points in the picture with a peck drill cycle, then would tap holes with the tap drill cycle at the same points.
G70.1 Bolt Hole, Center Hole Ignore Pattern
When commanded, the tool will first locate, but not drill a center hole, then drill holes located at points equally distributed on the circle. This G-code must be preceded by a valid canned drilling cycle (i.e., G81 - G88). The canned cycle G-code must precede G70.1 to establish the method of drilling for the pattern cycle. The X_ and Y_ parameters specified on the line containing the G81 - G88 determine where the center of the pattern will reside. The drilling canned cycle cannot reside on the same line as the drilling pattern cycle, G70.1.

Syntax:  
G70.1 I_ J_ L_
I:  Radius of circle must be greater than 0.
J:  Angle formed by X axis and vector from center of circle to start point.
L:  Number of points in the circle.

Programming Example:
G83 X_ Y_ Z_ R_ L_
G70.1 I3 J45 L8
G80
G84 X_ Y_ Z_ R_ L_ F_ P_ Q_
G70.1 I3 J45 L8
G80

The code excerpt above would first reference, but not drill the center hole, then drill a hole at the points in the picture with a peck drill cycle, then would tap holes with the tap drill cycle at the same points.

G71 Arc Pattern
When commanded, the tool is located at points distributed equally on an arc. This G-Code must be preceded by a valid canned cycle (i.e. G81, G82, G83, G84, G85, G86, G87, G88). The canned cycle G-code must precede G71 to establish the method of drilling for the pattern cycle. The X_ and Y_ parameters specified on the line containing G81- G88 determine where the center of the pattern will reside. The canned cycle G81 - G88 cannot reside on the same line as the pattern cycle G71.

Syntax:  
G71 I_ J_ K_ L_
I:  Radius of arc must be greater than 0.
J:  Angle formed by X-axis and vector from center of arc to start point.
L:  Number of points in the arc
K:  Angle between points on the arc

Programming Example:
G83 X_ Y_ Z_ R_ L_
G71 I3 J0 L8
G80
G70.1
G84 X_ Y_ Z_ R_ L_ F_ P_ Q_
G71 I3 J0 L8
G80
The code excerpt above would first drill a hole at the points in the picture with a peck drill cycle then would tap holes with the tap cycle at the same points.

**G72 Bolt Line Pattern**

When commanded the tool is located at points distributed equally on a line. This G-Code must be preceded by a valid canned cycle (i.e. G81, G82, G83, G84, G85, G86, G87, G88). The canned cycle G-code must precede G72 so as to establish the method of drilling for the pattern cycle. The X_ and Y_ parameters specified on the line containing G81- G88 determine where the start of the pattern will reside. The canned cycle G81 - G88 cannot reside on the same line as the pattern cycle G72.

**Syntax:**

\[ G72 \ I_ \ J_ \ L_ \]

- **I:** Distance between drill points, must be greater than 0.
- **J:** Angle formed by X-axis and vector of line.
- **L:** Number of points on the line.

**Programming Example:**

```
G83 \ X_ \ Y_ \ Z_ \ R_ \ L_
G72 \ I1 \ J45 \ L5
G80
G84 \ X_ \ Y_ \ Z_ \ R_ \ L_ \ F_ \ P_ \ Q_
G72 \ I1 \ J45 \ L5
G80
```

The code excerpt above would first drill a hole at the points in the picture with a peck drill cycle, then would tap holes with the tap cycle at the same points.

**G80-89 Canned Cycles**

A canned cycle simplifies programming through the use of single G codes to specify machine operations normally requiring several blocks of NC code. The canned cycle consists of a sequence of five operations as shown here:

1. Position of axes
2. Rapid to initial point
3. Hole body machining
4. Hole bottom operations
5. Retract to reference point

A canned cycle has a positioning plane and a drilling axis. The positioning plane is the G17 plane. The Z-axis is used as the drilling axis. Whether the tool is to be returned to the reference point or to the initial point is specified according to G98 or G99. Use G99 for the first drilling and G98 for the last drilling. When the canned cycle is to be repeated by L in G98 mode, the tool is returned to the initial level from the first time drilling. In the G99 mode, the initial level does not change even when drilling is performed.
The drilling data can be specified following the G and a single block can be formed. This command permits the data to be stored in the control unit as a modal value. The machining data in a canned cycle is specified as shown below.

**G80 Canned Cycle Cancel**
Cancels any active canned cycles.

**G81 Drilling Cycle**
When this cycle is commanded, the tool is located to the specified X, Y at rapid traverse rate, followed by a rapid traverse to the R-value. Normal drilling is then performed at the specified feedrate to the specified Z position. The tool is then immediately retracted from the bottom of the hole at rapid traverse rate. The return point in Z is either the value of Z when the canned cycle is called if G98 mode is active. Otherwise, the return point in Z is the value of R specified on the G81 line if G99 mode is active. This cycle will occur on every line, which includes an X and Y move, until the mode is canceled with G80 canned cycle cancel.

**Syntax:**
G81 X_ Y_ Z_ R_ F_ L_
X: Center location of hole along X
Y: Center location of hole along Y
Z: Depth to drill to
R: Reference plane in Z
F: Cutting feedrate
L: Number of repeats

**Programming Example:**
G99G81X-3.Y-2.75Z-0.05R0.1F250L2
X-2.75
X-2.5L2
X-2.25
G80
Drilling Cycle with G99 Active

Programming Example:
G98 G81 X-3. Y-2.75 Z-0.05 R0.1 F25.0 L2
X-2.75
X-2.5 L2
X-2.25
G80

Drilling Cycle with G98 Active

**G82 Boring, Spotfacing, Counter Sinking Cycle (Free Cutting)**

When this cycle is commanded, the tool is located to the specified X, Y at rapid traverse rate, followed by a rapid traverse to the R-value. Normal drilling is then performed at the specified feedrate to the specified Z position. A dwell then occurs at the bottom of the hole for P seconds. The tool is then retracted from the bottom of the hole at rapid traverse rate. The return point in Z is either the value of Z when the canned cycle is called if **G98** mode is active. Otherwise, the return point in Z is the value of R specified on the **G82** line if **G99** mode is active. This cycle occurs on every line that includes an X and Y move until the mode is canceled with **G80** canned cycle cancel.

**Syntax:**
G82 X_ Y_ Z_ R_ F_ L_ P_

X: Center location of hole along X
Y: Center location of hole along Y
Z: Depth to drill to
R: Reference plane in Z
F: Cutting feedrate
L: Number of repeats
P: Number of seconds of bottom dwell
Programming Examples:
G99G8.2X-3Y-2.75Z-0.05R0.1F25.0L2P2
X-2.75
X-2.5L2
X-2.25
G80

Boring, Spotfacing, Counter Sinking Cycle with G99 Active

Programming Examples:
G98G8.2X-3Y-2.75Z-0.05R0.1F25.0L2P2
X-2.75
X-2.5L2
X-2.25
G80

Boring, Spotfacing, Counter Sinking Cycle with G98 Active

G83 Deep Hole (Peck) Drilling Cycle
When this cycle is commanded, the tool is located to the specified X, Y at rapid traverse rate, followed by a rapid traverse to the R-value. Normal drilling is then performed at the specified feedrate to a depth of K below the R-value. The tool is then retracted from the bottom of the hole at rapid traverse rate to the R-value.

The tool is then moved at rapid traverse rate to the height of the last drilling plus the R parameter. Normal drilling is then repeated to a depth of K below the last hole. The tool is then once again retracted from the bottom of the hole at rapid traverse rate to the R-value.

This pattern is repeated until the depth of the Z parameter is achieved. This cycle permits intermittent cutting feed to the bottom of the hole, to assist in removing chips from the hole.
The return point in Z is the value of $Z$ when the canned cycle is called if G98 mode is active. Otherwise, the return point in Z is the value of R specified on the G83 line if G99 mode is active. This cycle occurs on every line that includes an X and Y move until the mode is canceled with G80 canned cycle cancel.

**Syntax:**

G83 X_ Y_ Z_ R_ F_ L_ K_
X: Center location of hole along X
Y: Center location of hole along Y
Z: Depth to drill to
R: Reference plane in Z
F: Cutting feedrate
L: Number of repeats
K: Peck depth

**Programming Examples:**

G83X-2Y-1Z-0.600K0.150R0.1F25
G80
G98
G83X-2Y-1Z-0.600K0.150R0.1F25
G80

---

**G84 Tapping Cycle**

When this cycle is commanded, the tool is located to the specified X, Y at rapid traverse rate, followed by a rapid traverse to the R-value. Linear movement is then performed at the programmed feedrate to the specified Z position. At this point a dwell of P seconds occurs. The spindle direction is then reversed and Z is fed linearly to the R-value. The return point in Z is the value of Z when the canned cycle is called if G98 mode is active. Otherwise, the return point in Z is the value of R specified on the G84 line if G99 mode is active. This cycle occurs on every line that includes an X and Y move until the mode is canceled with G80 canned cycle cancel. During this cycle, manual feedrate override is ignored.

**Syntax:**

G84 X_ Y_ Z_ R_ F_ L_ P_
X: Center location of hole along X
Y: Center location of hole along Y
Z: Depth to drill to
R: Reference plane in Z
F: Cutting feedrate, IPM (RPM * (1 / number of threads per inch))
L: Number of repeats
P: Dwell in seconds at bottom of Z travel
Programming Examples:
G99
G84X-2Y-1Z-0.5Q1R0.1F15.625P.5
X-3 Y-1
G80
G98
G84X-2Y-1Z-0.5Q1R0.1F15.625P.5
X-3 Y-1
G80

G85 Reaming, Boring Cycle
When this cycle is commanded, the tool is located to the specified X, Y at rapid traverse rate, followed by a rapid traverse to the R-value. Linear movement is then performed at the programmed feedrate to the specified Z position. Z is then fed linearly to the R-value. The return point in Z is the value of Z when the canned cycle is called if G98 mode is active. Otherwise, the return point in Z is the value of R specified on the G85 line if G99 mode is active. This cycle occurs on every line that includes an X and Y move until the mode is canceled with G80 canned cycle cancel. During this cycle, manual feedrate override is ignored.

Syntax:
G85 X_ Y_ Z_ R_ F_ L_
X: Center location of hole along X
Y: Center location of hole along Y
Z: Depth to drill to
R: Reference plane in Z
F: Cutting feedrate
L: Number of repeats

Programming Examples:
G99
G85X-3.Y-2.75Z-0.005P.5R0.1F25.0
X-2.75
X-2.5
G80
G98
G85X-3.Y-2.75Z-0.005P.5R0.1F25.0
X-2.75
X-2.5
G80
G86 Boring Cycle (Finishing cut)
When this cycle is commanded, the tool is located to the specified X, Y at rapid traverse rate, followed by a rapid traverse to the R-value. Linear movement is then performed at the programmed feedrate to the specified Z position. At this point, the spindle is stopped and a dwell of P seconds will occur. Z is then fed rapidly to the R-value. The return point in Z is either the value of Z when the canned cycle is called if G98 mode is active. Otherwise, the return point in Z is the value of R specified on the G85 line if G99 mode is active. This cycle occurs on every line that includes an X and Y move until the mode is canceled with G80 canned cycle cancel. During this cycle, manual feedrate override is ignored.

**Syntax:**

G86 X_ Y_ Z_ R_ F_ P_ L_

- **X:** Center location of hole along X
- **Y:** Center location of hole along Y
- **Z:** Depth to drill to
- **R:** Reference plane in Z
- **F:** Cutting feedrate
- **P:** Dwell in seconds at the bottom of the cut
- **L:** Number of repeats

**Programming Examples:**

G99
G86X-3.Y-2.75Z-0.005F.5R0.1P25.0
X-2.75
X-2.5
G80
G98
G86X-3.Y-2.75Z-0.005F.5R0.1P25.0
X-2.75
X-2.5
G80
**G87 Boring Cycle (Manual or Programmed Quill Return)**

When this cycle is commanded, the tool is located to the specified X, Y at rapid traverse rate, followed by a rapid traverse to the R-value. Linear movement is then performed at the programmed feedrate to the specified Z position. At this point the spindle is stopped. The Z-axis is returned either manually or with programmed instructions.

**Syntax:**

G87 X_ Y_ Z_ R_ F_ L_

- **X:** Center location of hole along X
- **Y:** Center location of hole along Y
- **Z:** Depth to drill to
- **R:** Reference plane in Z
- **F:** Cutting feedrate
- **L:** Number of repeats

**Programming Examples:**

G99

G87X-3.Y-2.75Z-0.005P.5R0.1F25.0

X-2.75

X-2.5

G80

G98

G87X-3.Y-2.75Z-0.005P.5R0.1F25.0

X-2.75

X-2.5

G80
G88 Boring Cycle (Free Cutting, Manual or Programmed Quill Return)
When this cycle is commanded, the tool is located to the specified X, Y at rapid traverse rate, followed by a rapid traverse to the R-value. Linear movement is then performed at the programmed feedrate to the specified Z position. At this point, a dwell of P seconds is performed and then the spindle is stopped. The Z-axis is returned either manually or with programmed instructions.

Syntax:    
G88 X_ Y_ Z_ R_ F_ P_ L_
X:  Center location of hole along X
Y:  Center location of hole along Y
Z:  Depth to drill to
R:  Reference plane in Z
F:  Cutting feedrate
P:  Dwell in seconds at the bottom of the cut
L:  Number of repeats

Programming Examples:
G99
G88X-3.Y-2.75Z-0.005P.5R0.1F25.0
X-2.75
X-2.5
G80
G98
G88X-3.Y-2.75Z-0.005P.5R0.1F25.0
X-2.75
X-2.5
G80

Free Cutting, Manual or Programmed Quill Return Example

G89 Boring Cycle (Finishing Cut, Free Cutting)
When this cycle is commanded, the tool is located to the specified X, Y at rapid traverse rate, followed by a rapid traverse to the R-value. Linear movement is then performed at the programmed feedrate to the specified Z position. At this point a dwell of P seconds is performed. Z is then fed linearly to the R-value. The return point in Z is the value of Z when the canned cycle is called if G98 mode is active. Otherwise, the return point in Z is the value of R specified on the G85 line if G99 mode is active. This cycle occurs on every line that includes an X and Y move until the mode is canceled with G80 canned cycle cancel. During this cycle, manual feedrate override is ignored.

Syntax:    
G88 X_ Y_ Z_ R_ F_ P_ L_
X:  Center location of hole along X
Y:  Center location of hole along Y
Z:  Depth to drill to
R: Reference plane in Z
F: Cutting feedrate
P: Dwell in seconds at the bottom of the cut
L: Number of repeats

Programming Examples:
G99
G88X-3.Y-2.75Z-0.005P.5R0.1F25.0
X-2.75
X-2.5
G80
G98
G88X-3.Y-2.75Z-0.005P.5R0.1F25.0
X-2.75
X-2.5
G80

G90/G91 Absolute/Incremental Mode
Program commands for movement of the axes may be programmed either in incremental movement commands or in absolute coordinates. The absolute mode is selected automatically when the power is turned on or the control is reset. In the absolute mode (G90), all axis word dimensions are referenced from a single program zero point. The algebraic signs (+ or -) of absolute coordinates denote the position of the axis relative to program zero.

In the incremental mode (G91), the axis word dimensions are referenced from the current position. The input dimensions are the distance to be moved. The algebraic sign (+ or -) specifies the direction of travel.

Syntax:
G90  (Absolute mode)
G91  (Incremental mode)

Example Code:
N020 G20 G90 G0 X0 Y0 (inch, abs, rapid to work piece x,y zero psn)
N025 G43 Z0.25 H1
N030 X1.125 Y2.25

G90.1/G91.1 Arc Radius Abs/Inc Mode
Program commands for movement of the axes may be programmed either in incremental or absolute movement commands.

Syntax:
G90  (Absolute mode)
G91  (Incremental mode)
G92 Work Coordinate System Set
This command establishes the work coordinate system so that a certain point of the tool (e.g. tool tip) becomes IP in the established work coordinate system. Any subsequent absolute commands use the position in this work coordinate system. Meet the programming start point with the tool tip and command G92 at the start of program (G92X25.2Z23.0). When creating a new work coordinate system with the G92 command, a certain point of the tool becomes a certain coordinate value; therefore, the new work coordinate system can be determined irrespective of the old work coordinate system. If the G92 command issued to determine a start point for machining based on work pieces, a new coordinate system can be created even if there is an error in the old work coordinate system. If the relative relationships among the G54 to G59 work coordinate systems are correctly set at the beginning, all work coordinate systems become new coordinate systems as desired.

Syntax: G92X__Y__Z__

Example Code:
N4 G53X0Y0Z0
N5 G92X0 Y1.0156
N6 Z.1 H1 M8

G93 Inverse Time Feed
G93 specifies inverse time mode: move is specified by move time. F word is in time units of seconds and is derived from the Rate x Time = Distance equation applied to the specific block move: (ΔX_{in} / F_{ipm}) x 60 = ΔT_{sec}

Example:
Assume X is at zero.
Specify the following as Inverse Time.
G01X1F100
a. Solve for move time.
   \[ F_{ipm} = 100; \Delta X_{in} = 1 \]
   \[ \Delta T_{sec} = (1 \div 100) \times 60 = .6\text{sec}. \]
b. Recode the block
   G01G93X1F0.6

Syntax: G93F_
**G94/G95 Feed Per Min/Feed Per Rev**
The G94 preparatory function code specifies the feed rate in terms of vector per unit time. The G95 preparatory function code specifies feed rate in terms of vector feed per spindle revolution. The G94 and G95 preparatory functions are modal and remain in effect until replaced by the opposite code. The mode is set to G94 by power on, data reset and the M30 code.

Syntax: G94/G95

**G98/G99 Canned Cycle Return Point**
Used in a canned cycle block to determine the return point. G98: Initial point. G99: clearance plane or reference point. G98 causes the tool to return to the point from which it was first called. G99 causes the tool to return to the point specified by the R address.

Syntax: G98/G99

**Example Code:**
N4 X0Y0
N5 G98
N6 G81X1 Y1R0.1Z-3
.
.
.
N4 Z5
N5 G99
N6 G81X1Y1R0.1Z-3

**M Code Library – CNC M-Codes**

<table>
<thead>
<tr>
<th>G-Code</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>M00</td>
<td>Program Stop</td>
</tr>
<tr>
<td>M01</td>
<td>Optional Stop</td>
</tr>
<tr>
<td>M02</td>
<td>Program End &amp; Rewind</td>
</tr>
<tr>
<td>M03</td>
<td>Spindle CW</td>
</tr>
<tr>
<td>M04</td>
<td>Spindle CCW</td>
</tr>
<tr>
<td>M05</td>
<td>Spindle Stop</td>
</tr>
<tr>
<td>M06</td>
<td>Tool Change</td>
</tr>
<tr>
<td>M08</td>
<td>Coolant On</td>
</tr>
<tr>
<td>M09</td>
<td>Coolant Off</td>
</tr>
<tr>
<td>M19</td>
<td>Spindle Orient</td>
</tr>
<tr>
<td>M30</td>
<td>Program End &amp; Rewind</td>
</tr>
<tr>
<td>M87</td>
<td>Start Data Gathering</td>
</tr>
<tr>
<td>M88</td>
<td>End Data Gathering</td>
</tr>
<tr>
<td>M98</td>
<td>Subprogram Call</td>
</tr>
<tr>
<td>M99</td>
<td>Subprogram Return</td>
</tr>
</tbody>
</table>

**M00 Program Stop**
Unconditional stop of part program at current block. The coolant and spindle are stopped with this command. Machine state does not change until restart or rewind.
**M01 Optional Stop**
Same as M00, but conditional on Optional stop switch setting.

**Example:**
```
  . .
  X-1.25
  X-1.
  G80
  M1   (OPT STOP M1)
```

**M02 Program Rewind**
This resets the program buffer to the beginning of the program, cancels tool compensation and resets coolant and spindle to off.

**Example:**
```
  . .
  G0G49X0Y0Z0
  Z.5M5M9
  G90G0G49M5M9
  X0Y0Z0
  M2
```

**M03 Spindle Clockwise**
Starts the spindle in the clock-wise direction (CW) using the current setting for speed.

**Example:**
```
  . .
  N30 G54 G0 X-3.7185 Y-.1649
  N40 S5000 M3 T1
  N50 G43 H1 Z.1
  . .
```

**M04 Spindle Counterclockwise**
Starts the spindle in the counter-clockwise direction (CCW) using the current setting for speed.

**M05 Spindle Stop**
Turns off the coolant and stops the spindle.

**Example:**
```
  . .
  N1940 G28 X0. Z0.
  N1945 M5
  N1950 M2
  . . tool change lighted pushbutton.
```

**Note:**
The tool change position is above the home position.

**Example:**

**M06 Tool Change**
Moves to the tool change position and blinks the
```
  . .
  G0G49X0Y0
  T3M6
  M3S100
  M8
  G0X1.5Y-1.5
  . .
```
**M08 Coolant On**
Engages the coolant pump.

**Example:**

```gcode
G43Z0.5H10
M8
```

**M09 Coolant Off**
Disengages the coolant pump.

**Example:**

```gcode
X-4.1657Y-5.4552
G2X-4.2073Y-5.4421I-0.0056J0.0547
G0Z0.5M5M9
```

**M19 Spindle Orient**
The spindle rotates to a known angle.

**Example:**

```gcode
X-4.1657Y-5.4552
M19
Z-2.01
```

**M30 End of Program (and Rewind)**
Same as M2.

**Example:**

```gcode
Z.5M5M9
G90G0G49M5M9
X0Y0Z0
M30
```

**M87 Start Data Gathering**
Setup the Data Gathering buffer and begin gathering data.

**Example:**

```gcode
Z.5M5M9
G90G0G49M5M9
X0Y0Z0
M87
```

**M88 End Data Gathering**
End the Data Gathering.

**Example:**

```gcode
Z.5M5M9
G90G0G49M5M9
X0Y0Z0
M88
```
**M98 Subroutine Call**

**Syntax**

M98 \[[L____] [P____ ] [:]
M98 \[L____] (C:\… ) [:]
M98 \[L____] [O___ ] [:]

Where:

- **L____** - specifies the number of times to execute the program
- **O____** - specifies a program name of the form O____.nc
- **P____** - specifies a program name of the form P____.nc
- **(C:\…)** - specifies a program name as a path and filename in a comment.

**Description**

M98 is the NC program’s method of transferring control to another program from an executing program. When an M98 command is encountered in a currently executing program (the calling program), control is transferred to the specified program in the M98 block, the called program. The name of the calling program is saved by the Control. The called program can transfer control back to the calling program with an M99 block.

There are three ways of specifying a called program in an M98 block:
1. P Code specification
2. Comment specification
3. Code specification

**P Code Specification:** The most used and standard method is by referencing a program number with a P address code. This is the method that is suggested if running the programs on other controls. When the control sees a Pnnnn code on the M98 line, it constructs a filename from the number following the P address code. The filename is of the following form: Onnnn.nc. The nc part of the filename is called the file extension. By default the file extension is nc. The control then searches the directory that the calling program was executed from for the program that has that filename. If the program is not found, an alarm is issued and program execution stops. Otherwise, that program is loaded and program execution continues from the first block in the called program, the subroutine.

**Comment Specification:** Another way to specify a program is by specifying a full path and filename in a comment that is on the M98 line. This is a way to transfer control to routines that are not in the same directory as the calling routine. All that is necessary is to place a valid file path name in the comment. If the path or filename is invalid, then an alarm is issued.

**Note:**

When specifying a filename explicitly (C:SUBPROG.NC), the full path is not necessary. If the full path is not specified, NCUI will look for the file in the directory specified by the Start in property of the windows shortcut that launched the program.

**O Code Specification:** When neither of these methods is present in a block, the control constructs a filename using the number associated with the most recently invoked O address code. This means that O can be used just like a P code.

**Note:**

When in MDI mode, the calling program exists in the directory specified by the Start in property of the windows shortcut that launched the NCUI. This means that when in MDI mode the subprograms identified by an O should reside in the Start in directory. MDI mode does not support subprograms calls, such as M98P100.
Note:
If a P code or comment on an M98 line is missing and there is an O code at the top of the program, the program will be called again recursively.

If more than one of these methods exists, the control selects a method based on priority. Only one method is selected. The priorities are first P code, then Comment, and finally O code. So if there is a P and O address code on the same block, the P code is used to construct a filename for program execution.

A called program (subroutine) can return control back to the calling program by executing an **M99**.

The number of subroutines that can be nested is limited only by PC memory. However, nest no more than ten levels deep. This value is often encountered in other controls.

A subroutine can be called more than once within the same block. This is called looping. The number of loops is specified with the L address code. **M98 P10 L4**; would execute program O10.nc four consecutive times.

**Examples**
Program O98.nc calls O100.nc once and PRG.nc 100 times.
Program O98.nc ---

```cnc
% O98 (Subroutine call example)  
G04 X1  
M98 P100  
M98 (C:\CNC\PRG.NC) L100  
G04 x2  
M30  
%
```

Program O100.nc in the same directory as program O10.nc ---

```cnc
% O100  
G91 G81 X.5Z-1.0 F30;  
G90 M99  
%
```

PRG.NC is a NC program with a M99 for a return from subprogram. ---

```cnc
% (PRG.NC)  
G1 X5 Z5  
( . . . )  
G0 X2  
M99  
%
```

**M99 Return from Subroutine**

**Syntax**

```
M99 [L____ ] [P____ ] [:]
```

**Where:**

- **L____** - specifies the number of times to execute the program
- **P____** - specifies a program block to branch to.

**Description**

M99 transfers program control to a calling program or to a different location of the current program being executed. The action of **M99** is different depending on whether **M99** is encountered in a subroutine or in the main program.
Subroutine program action of M99:
- If M99 is encountered in a subprogram and no L or P address code is in the block, processing is transferred to the first block after the M98 block or the calling program that called the current program.
- If an L is on the M99 line, the subroutine resumes execution at the first block of the subroutine and loops L times. This L overrides any L in the calling M98 block. In a subroutine, control is always transferred to the first block, even if there is a P on the M99 line.
- If a P address code is on the M99 line in a subroutine, execution is resumed in the calling program, not at the line after the M98 call, but at the first N address code found after the M98 call matching the P address code. The control searches from the first block after the block with the M98 address code to the end of the program, and then continues from the top of the program to the block containing the M98. If no match is found an alarm is issued and the program execution stops.

Main program action of M99:
- If M99 is encountered in the main program and no L or P code is in the block, processing is transferred to the first block of the program. In this manner a program can be commanded to loop indefinitely.
- If an L is on the M99 line, the program loops L times and then executes an M30.
- If a P address code is on the M99 line, processing is transferred to a block that contains a matching N address code. The control searches from the first block after the block with the M99 address code, to the end of the program and then continues from the top of the program to the block containing the M99. Control is transferred to the first block found with a matching N address code in it. If no match is found, an alarm is issued and program execution stops.

Examples:
Program O99.nc performs initialization and loops indefinitely.
--- Program O99.nc ---
%  O99  (M99 example)  ;
    (Initialization code. Executed one time only)  ;
    (...);
    N50  ;
    (The part program. Executed indefinitely)  ;
    (...);
    M99 P50  ;
%
Program O990.nc calls O991.nc twice. Each time O991.nc loops 5 times and returns.
Program O990.nc
%  O990  (M99 example)
    G90
    M98 O991
    G0 X-5.0  
    M98 P991
    M30
%
Program O991.nc. Called by O990.nc
%  O991  (Subroutine)
    G91
    G81 X.5 Z-.5 F30.0  ;
X.4 ;
X,2 ;
G90 G80
M99 L5 ;
%

**T-Codes**

**T-Code Format**

Tnn  Where nn specifies tool number from the Tools page in the NC display.

**Example:**

(TOOL 4 = .437 DRILL)
(TOOL 3 = 1/2-13 TAP)
G90G80G49G40G20G17G56
T4M6
M3S3000
M8
G0X1.5Y-1.5
...

**Miscellaneous**

**Block Delete Character: /**

Prevents execution of the block when Block Delete is on. Must be the first character in the block.
PARAMETRIC PROGRAMMING

Introducing Parametric Programming

Parametric programming is an extension to NC (Numeric Control) programming. It gives the programmer of NC products the ability to use variables and to perform conditional branching within an NC program. Subroutines are extended to accept arguments. Predefined functions such as sine and cosine can be used. Expressions can be evaluated. With parametric programming it is possible to create libraries of routines that can be used and reused. Custom canned cycles and families of parts can be programmed with less effort. Parametric programming increases the productivity and versatility of machine tools and reduces the cost of machined products. Parametric programming is not meant to be used in lieu of CADCAM systems. It is provided to add flexibility to the NC control and to provide compatibility to controls that have made use of parametric programming in the past.

Delta Tau offers two forms of parametric programming. The programmer has the ability to program in PMAC native code, or he can program in FANUC compatible (macros) parametric programming. This section describes the parametric programming that is compatible with FANUC macro code.

Example Programs

This section contains example programs. Several applications are presented here to give an idea of the power of parametric programming. Each program is described by three paragraphs.

- **Purpose** explains the program and includes supporting documentation to fully explain the program.
- **Routines** contains commented listings of supporting routines used in the main program.
- **Program** is a commented listing of the main program.

The following is a list of example programs displayed in this section:

- Clearing global variables
- Drilling custom bolt-hole patterns
- Simple pocket milling

Clearing Global Variables

**Purpose:**

This is an example of how to initialize variables. The program calls a generalized subroutine that initializes a range of variables. The program is identified on disk as O100.NC and the parametric subroutine as O9400.NC. Start and ending numbers define the range of variables. If no value is passed in argument V, the range is initialized to #0 which indicates that the variables value is undefined. If the range of variables is invalid an alarm is generated. The range checking is not necessary in the sense that if an invalid variable is passed, the control will automatically alarm. In a subroutine like the one below, the range of variables may be limited to protect variables reserved for the application.

**Routines:**

```
%09400 (Init variables);
(V=value to write to range of variables);
(S=variable number to start writing to);
(E=variable to end writing to);
(Write V into variables starting with S thru and including);
(Variable E.  S must be less than e and both arguments must);
(be supplied and valid.  V is always used to write to variables);
IF [#19 EQ #0]   GOTO N9410        (S not passed);
IF [#8   EQ #0]   GOTO N9410        (E not passed);
(Invalid S argument);
IF [[#19 GE 1] AND [#19 LE 33]]  GOTO 9402  (1..33 is OK);
IF [[#19 GE 100] AND [#19 LE 199]]  GOTO 9402  (100..199 is OK);
```
IF [[#19 GE 500] AND [#19 LE 599]] GOTO 9402 (500..599 is OK);
GOTO 9420 (ERROR)

N9402
(Invalid E argument);
IF [[#8 GE 1] AND [#8 LE 33]] GOTO 9404 (1..33 is OK);
IF [[#8 GE 100] AND [#8 LE 199]] GOTO 9404 (100..199 is OK);
IF [[#8 GE 500] AND [#8 LE 599]] GOTO 9404 (500..599 is OK);
GOTO 9420 (ERROR)

N9404
IF [#19 GE #8] GOTO N9420 (ERROR IF S>=E);
WHILE [#19 LE #8] DO1;
#19=#19+1;                    (Increment destination);
END1;
GOTO 9499 (successful return);
(Alarms follow);
N9410 #3000=941(Missing argument);
N9420 #3000=942(Variable range definition error);
N9499 M99;
%

Program:
%
O94 (Main program that demonstrates variable initialization)
G65 P9400 S500 E589 (Global variables are undefined)
G65 P9400 S590 E599 V0.0 (set 590..599 to 0.0)
G65 P9400 S500 E600 (Generates an alarm)
M30
%

Drilling Custom Bolt-Hole Patterns

Purpose:
This example shows how to program a routine to drill and tap a custom bolt hole pattern. Here, the bolt hole pattern is for a standard clamp that may be used in holding parts on a fixture. The subroutine can be saved and used later for drilling the bolt holes on other fixtures.

The bolt-hole pattern is a simple rectangular pattern. The absolute position of the center of the bolt-hole pattern is passed in arguments X and Y. The height and width of the bolt spacing is passed as arguments H and W. The rotation of the pattern is passed in A. The initial position plane is passed in R and the incremental hole depth from R is passed in Z. Assume ¼-20 tap is being used. To use this routine, there must be a drill and tap set up in tool holders 1 and 2 and the appropriate arguments must be passed. This routine could be improved by adding a center drilling operation or by passing a feed rate parameter to accommodate various materials.
Routines:

\%
O9600 (Drill and tap a rectangular bolt-hole pattern);
(X=Absolute X location of center of bolt-hole pattern);
(Y=Absolute Y position of center of bolt-hole pattern);
(H=Y distance between holes with 0 rotation)
(W=X distance between holes with 0 rotation);
(A= Angular rotation of pattern in degrees);
(R = Return plane of reference, Z start plane);
(Z = Incremental Z depth to tap holes);

(Generate four tapped holes given the location of the center of the rectangular pattern, the rotation and);
(the depth of the holes. If X and Y are not passed, assume center is at current location);
(if A is not passed, assume no rotation; if R is not passed, assume R is at current Z position)

If [#11 EQ #0]   GOTO9610        (H not passed);
If [#23 EQ #0]   GOTO9620        (W not passed);
If [#26 EQ #0]   GOTO9630        (Z not passed);

(Record R, X, and Y; if they are not passed)
If [#24 EQ #0] then #24=#5041 (current X position);
If [#25 EQ #0] then #25=#5042 (current Y position);
If [#18 EQ #0] then #18=#5043 (current Z position);

(Calculate X and Y offset from center using H and W);
#31=#11/2;
#32=#23/2;

(Move to tool change position)
G0 G53 Z0
G0 G53 X0 Y0

(Select drill T1, #7 .201 drill)
T1 M06
S1000 M3 (spindle on)
M8 (coolant on)
G0 X#24 Y#25 (move to XY location);
Z#26    (move to R plane);

(Rotate if requested);
IF [#1 EQ #0] GOTO9602
G68 R#1;
N9602
(Drill four holes at incremental offsets from center);
G81 X[#24+#31] Y[#25+#32] Z#26 F5. (upper right);
X[#24 - #31] Y[#25 +#32]     (upper left);
X[#24 - #31] Y[#25 - #32]    (lower left);
X[#24 + #31] Y[#25 - #32]    (lower right);
G69 (cancel rotation);
M5 (spindle off);
M9 (coolant off);

(Move to tool change position)
G0 G53 Z0
G G53 X0 Y0

(Select drill T2, ¼-20 TAP)
T2 M06
PMAC NC for Mill Application

S100 M3 (spindle on)
M8 (coolant on)
G0 X#24 Y#25 (move to XY location);
Z#26    (move to R plane);
(rotate if requested);
IF [#1 EQ #0] GOTO9604
G68 R#1;
N9604
(Tap four holes at incremental offsets from center);
G84 X[#24+#31] Y[#25+#32] Z[#26-.2] F5. (upper right);
X[#24 - #31] Y[#25 +#32]    (upper left);
X[#24 - #31] Y[#25 - #32]    (lower left);
X[#24 + #31] Y[#25 - #32]    (lower right);
G69 (cancel rotation);
M5 (spindle off);
M9 (coolant off);
GOTO 9699 (successful return);
(Alarms follow);
N9610 #3000=961(missing H);
N9620 #3000=962(missing W);
N9630 #3000=963(missing Z);
N9699 M99;
%

Program:
%
O96 (main program that demonstrates bolt-hole patterns);
G65 P9600 X1.0 Y1.0 H1.25 W1.75 Z-.75  (clamp 1);
G65 P9600 X7.0 Y8.0 H1.25 W1.75 Z-.75 A-135 (clamp 2);
M30
%

Parametric Subroutines

A parametric subroutine is an extended version of an M98 subroutine. The following list identifies the features that make up a parametric subroutine.

- Numeric arguments can be passed from the calling program to the subroutine.
- Each subroutine called has access to its own private variables that are not altered by other subroutine calls. These variables are local to the subroutine.
- Subroutines can be aliased.
- The parametric subroutine is invoked with a line containing G65 and a P code.

Example: N100 G65 P50 A1.0 B2.0 C3.0

In the above example, block 100 invokes program 50 as a parametric program. It passes to program 50 arguments A, B and C.

The G65 command loads the subroutine into memory and allocates a set of local variables for its use. The local variables of the calling routine are saved and are restored when the subroutine returns with an M99. This concept of saving and restoring of local variables is known as nesting. The Delta Tau NC nesting is limited to four levels. The main program is considered to be nesting level 0.

Following is an example of how nesting takes place and how variables are allocated in G65 and M98 subroutines.
There are 33 local variables allocated when G65 is invoked. They are initialized to an undefined value, indicating that they have no value assigned to them. This value is called undefined and its symbol is #0. If arguments are passed to the subroutine, the values of the arguments are stored into the local variables of that subroutine. Address codes G, L, N, O and P cannot be used as arguments. Refer to the section on local variables for more information.

**Variables**

Variables are what make parametric programming possible. Variables are used to replace literal values in the programs. A literal value is a constant that cannot be changed.

**Example:**

G1 X1.2 Y3.5 (examples of literal constants)  
#1=1.2  
#2=3.5  
G1 X#1 Y#2 (example of replacing literals with variables)

Variables can be modified in a program by using assignment statements. Variables are floating point numbers. They are referenced by using a #<integer> notation where <integer> is a positive integral number. The value of <integer> is restricted (i.e. only values defined in this manual can be used).

Variables can be accessed in an indirect method by replacing <integer> with [<expr>], where <expr> is any expression. The following all refer to the same variable.

**Example:**

#1  
#[1]  
#[(1+2+3)/6]  
#2=1  
#[#2]  
#[SIN(90.0)]
Variables can be used in conditional expressions, assignment expressions, GOTO expressions and address code expressions.

Example:
   IF [#1 EQ 3.0] GOTO#5;
       #3=#3+2;
       G1 X#1 Y-#2 Z#3;

The categories of variables in a FANUC compatible parametric program are:
- Undefined #0
- Local
- Common
- System

Each category is discussed in detail below.

In order to determine if a program is executing correctly, it is necessary to be able to view variables as they are being modified in the program. The parameter display is ideal for this. Screens for displaying Local and Common variables are predefined on the parameter display. Refer to the Parameter Display section for details on how to access and use this useful tool.

Undefined #0

A variable that has not been assigned a value is called undefined. It is referenced in an NC program with #0. Undefined variables allow the programmer to determine if a subroutine is being called correctly or to determine if a certain logic path has been taken. Allowing variables to be undefined requires special processing.

Algebraic expressions convert undefined variables to 0.0. If the equation is singular, then the expression evaluator returns undefined.

Example:
   #1=#0;  (#1 is undefined);
   #3=#1  (#3 is undefined);
   #4=#1+#3  (#4 is 0.0);
   #5=#1*3  (#5 is 0.0);

Address Codes using variables that evaluate to undefined are ignored. This means that if an address code is parsed with an undefined variable it is just as if the parser did not see that address code.

Example:
   #1=#0;
   #2=3.5;
   G1 X#1 Y#2  (same as G1 Y3.5);
   X[#1+#0]  (same as X0.0);
   Z-[#1]    ( Z is ignored);

Conditional expressions convert undefined values to 0.0 in the same manner as algebraic expressions. If the algebraic expression results in an undefined value, it is treated as 0.0 except for EQ and NE.

Conditional
   [#1 EQ #0] True False
   [#1 NE 0] True False
   [#1 EQ 0] False True
   [#1 GE #0] True True
   [#1 GT 0] False False
   [#1 LT 0] False False

If using a variable as for example, a counter and comparing for 0.0, use 0.0. If comparing to see if a variable has been assigned to, use #0. It is important to distinguish between the two.
Local Variables
Local variables, as discussed above, are allocated and initialized each time a G65 is executed. Local variables are numbered #1..#33. They are initialized to undefined which is equivalent to #0. Thus, if the conditional phrase [#1 EQ #0] evaluates to true, then #1 has never been assigned a value and either argument A was not passed to the current level of nesting or argument A was #0.

M98 differs from G65 in that the local variables are not nested. M98 subroutines can still access local variables. In an M98 subroutine the local variables that are accessed belong to the most recent G65 nesting level.

M99 in a G65 subroutine will un-nest local variables and the values are lost. M99 in a M98 subroutine do not un-nest local variables.

Local variables are used to hold arguments passed to a G65 parametric subroutine call. Local variables can be tested against #0 to determine if an argument with a value was passed. Address code arguments are passed according to the following table.

<table>
<thead>
<tr>
<th>Address Code</th>
<th>Local Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>#1</td>
</tr>
<tr>
<td>B</td>
<td>#2</td>
</tr>
<tr>
<td>C</td>
<td>#3</td>
</tr>
<tr>
<td>D</td>
<td>#7</td>
</tr>
<tr>
<td>E</td>
<td>#8</td>
</tr>
<tr>
<td>F</td>
<td>#9</td>
</tr>
<tr>
<td>H</td>
<td>#11</td>
</tr>
<tr>
<td>I</td>
<td>#4</td>
</tr>
<tr>
<td>J</td>
<td>#5</td>
</tr>
<tr>
<td>K</td>
<td>#6</td>
</tr>
<tr>
<td>M</td>
<td>#13</td>
</tr>
<tr>
<td>Q</td>
<td>#17</td>
</tr>
<tr>
<td>R</td>
<td>#18</td>
</tr>
<tr>
<td>S</td>
<td>#19</td>
</tr>
<tr>
<td>T</td>
<td>#20</td>
</tr>
<tr>
<td>U</td>
<td>#21</td>
</tr>
<tr>
<td>V</td>
<td>#22</td>
</tr>
<tr>
<td>W</td>
<td>#23</td>
</tr>
<tr>
<td>X</td>
<td>#24</td>
</tr>
<tr>
<td>Y</td>
<td>#25</td>
</tr>
<tr>
<td>Z</td>
<td>#26</td>
</tr>
</tbody>
</table>

Address codes G, L, N, O and P cannot be used as arguments. Note that the mapping is irregular. This follows the FANUC convention.

A subroutine can access arguments by referring to the associated variable name.

Example: #31=#1 * 2 (assign to variable #31 the argument A times 2);
          G1 G91 X#24 (Feed X the incremental amount of argument X);

Note that subsequent assignments will destroy the value of the passed argument.
Common Variables

Common variables are always accessible in an NC program. Another term used is global variables. Common variables are numbered #100 through #199 and #500 through #599.

Variables #100..#199 are initialized to undefined when the control is turned on. Variables #500..#599 values are maintained between power up and down conditions.

Common variables can be used to pass information from one parametric subroutine to another. Once a value is set, it remains available, regardless of nesting level, until it is later modified.

System Variables

System variables give the NC programmer access to static parameters built into the control. Occasionally the programmer needs access to these parameters in order to alter or automate machine setup. A summary of system variables is below:

<table>
<thead>
<tr>
<th>Variable #</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1000-#1031</td>
<td>Discrete Inputs</td>
</tr>
<tr>
<td>#1100-#1131</td>
<td>Discrete Outputs</td>
</tr>
<tr>
<td>#2000-#2999</td>
<td>Tool Compensation</td>
</tr>
<tr>
<td>#3000</td>
<td>User Alarm with message</td>
</tr>
<tr>
<td>#3001-#3002</td>
<td>System Timers</td>
</tr>
<tr>
<td>#3003-#3004</td>
<td>Single block and override suppression</td>
</tr>
<tr>
<td>#3006</td>
<td>Programmable Stop with message</td>
</tr>
<tr>
<td>#3007</td>
<td>Mirroring</td>
</tr>
<tr>
<td>#4001-#4120</td>
<td>Look ahead time modal information</td>
</tr>
<tr>
<td>#4201-#4320</td>
<td>Run time modal information</td>
</tr>
<tr>
<td>#5001-#500n</td>
<td>Target work coordinate position of last executed block. Tool offset included.</td>
</tr>
<tr>
<td>#5021-#502n</td>
<td>Commanded machine coordinate position, tool offset not included.</td>
</tr>
<tr>
<td>#5041-#504n</td>
<td>Commanded work coordinate position, tool offset not included.</td>
</tr>
<tr>
<td>#5061-#506n</td>
<td>Current work coordinate skip position, tool offset not included.</td>
</tr>
<tr>
<td>#5081-#508n</td>
<td>Current tool offset applied</td>
</tr>
<tr>
<td>#5101-#511n</td>
<td>Current following error</td>
</tr>
<tr>
<td>#5201-#520n</td>
<td>Common work coordinates</td>
</tr>
<tr>
<td>#5221-#522n</td>
<td>G54</td>
</tr>
<tr>
<td>#5241-#524n</td>
<td>G55</td>
</tr>
<tr>
<td>#5261-#526n</td>
<td>G56</td>
</tr>
<tr>
<td>#5281-#528n</td>
<td>G57</td>
</tr>
<tr>
<td>#5301-#530n</td>
<td>G58</td>
</tr>
<tr>
<td>#5321-#532n</td>
<td>G59</td>
</tr>
<tr>
<td>#7001-#795n</td>
<td>G54.1 P1..P48 extra offsets</td>
</tr>
</tbody>
</table>

System Variables

#1000-#1031  Discrete Inputs

Parametric programming allows use of discrete inputs in the NC program. These inputs are from #1000 to #1031, total 32 inputs. Each number represents a Bit.

Requirements:
- Input card (Acc34 Style) to be map as discrete input.
- Code in the control panel PLC to read inputs.

Example:
```
//-------------
// ACC34 board 2 IN
//-------------
IN_2_CHNG_M = ACC34_2A
```
IF (IN_2_M != IN_2_CHNG_M)  // has one or more input bit(s) changed?
IN_2_M=IN_2_CHNG_M         // update change flag
ENDIF

How to Use in the NC program:

This is an example for using ACC34_2A as discrete input:
In the ADDRESS.H there are input images for the inputs.

IN_2_M M231
#1000 represents the LSB input Bit and #1031 represents MSB input bit.
1. Start the NC program. Make sure to enable New Diagnostic feature (Pages.DAT).
2. Select DIAG (F7) menu and select Common Variable #100 to #131 page since using #100 in the program.
3. In the MDI mode, write the following program:
   G103 P1
   #100 = #1000
   #101 = #1001
   #102 = #1002
   #131 = #1031
   M30 (Use M99 for continuous execution)

4. Execute the MDI program by pressing Cycle Start.
If the inputs are connected, then in the DIAG page status will be displayed.
If the inputs are not connected, use PEWIN32 and write M231 = 7 and the DIAG page will display #100 = 1, #101 = 1 and #102 = 1.

Discrete Outputs #1100 - #1131
Parametric programming allows using discrete output in the NC program. These outputs are from #1100 to #1131, total 32 outputs. Each number represents a Bit.

Requirements:
• Output card (ACC-34 style) to be mapped as discrete outputs.
• Code in the control panel PLC to read output.

Example:

//--------------
// ACC34 board 2 IN
//--------------
IN_2_CHNG_M = ACC34_2A
IF (IN_2_M != IN_2_CHNG_M)  // has one or more output bit(s) changed?
IN_2_M=IN_2_CHNG_M        // update change flag
ENDIF

How to Use in the NC program:

Here is an example for using ACC34_2A as discrete output:
In the ADDRESS.H there are output images for the outputs.

OUT_2_M M251
#1100 represents the LSB output bit and #1131 represents MSB output bit.
1. Start the NC program. Make sure to enable New Diagnostic feature (Pages.DAT).
2. Select DIAG (F7) menu and select Common Variable #100 to #131 page as we are using #104 in the program.
3. In the MDI mode write the following program:

   G103 P1
   #1100 = #104
   #1101 = #104
   #1102 = #104
   M30  (Use M99 for continuous execution)

4. Execute the MDI program by pressing **Cycle Start**.

If the outputs are connected, then in the DIAG page set #104 a value the outputs will be operated.

If the outputs are not connected then use PEWIN32 and set #104 to 7 in DIAG page #104 = 7 and read M251 in PEWIN32.

**#2000-#2999 Tool Compensation**

For a Mill, Tool compensation system variables are organized by H and D codes. The following are reserved for tool geometry and wear:

- #2000 Always returns zero when used in an expression, Associated with H0 and D0.
- #2001-#2200 H code Geometry for tool 1..200 (Tool Length offset).
- #2201-#2400 H code Wear for tool 1..200 (Tool Length Wear).
- #2401-#2600 D code Geometry for tool 1..200 (Tool Diameter offset).
- #2601-#2800 D code Wear for tool 1..200 (Tool Diameter Wear).

These system variables are associated with the values of the tool offsets on the tool offset display. For instance, tool 004 would be referenced with system variables #2004,#2204,#2404, and #2604 for Z GEOM, Z WEAR, CC GEOM and CC WEAR respectively.

The values of the tool offsets can be read from and written to by use of the above system variables.

When Tool offsets are modified with an assignment statement, the PC side block look ahead is halted and look ahead processing is not continued until the look ahead queue is exhausted. Assignment to a tool offset sends a G10 through the rotary buffer to be executed by PMAC.

**#3000 User Alarm with Message**

Fatal alarms can be generated from within a parametric program by assigning a value to #3000. The alarm generated will have this value as a reference in the alarm message. If a comment is on the block assigning #3000, it will be displayed on the error page.

**Example:**

   IF [#24 NE #0] GOTO 5 ;
   #3000=1024 (X argument required) ;
   N5 (conditional was true, X argument passed)

**#3001-#3002 System Timers**

These timers are millisecond timers. Timer #3001 is continuously running and will wrap around after 49.7 days of running. #3001 is initialized to zero at power up. Timer #3002 is an hour timer based on #3001. Hours are accrued only when a program is running. #3002 is saved at power down and is restored on power up. Both of these timers can be initialized with an assignment statement.

**Example:**

   dwell 3.5 seconds
   #31=#3001;
   N1 IF [#3001 LT [#31+ 3500]] GOTO1 ;

**#3003-#3004 Single Block and Override Suppression**

**#3006 Programmable Stop with Message**

A programmable stop (M00) can be generated from within a parametric program by assigning a value to #3006. A comment can be placed on the block to assist the operator in what operation is to be performed.

**Example:**

   #3006=2001 (message for the operator) ;

**#3007 Mirroring**

**#4001-#4026 Look ahead time modal group information**
Use these variables to determine what the look ahead time code is for any group. These variables contain the modal group information for the last parsed G-code block. #4001 through #4026 correspond to groups 1 through 26. For example to know if cutter compensation is off, check to make sure that group 7 is 40.

IF [#4007 EQ 40] GOTO ??? (cutter compensation is off)

### Look Ahead Time Modal Address Code Information

Use these variables to determine what the look ahead time value is for any address code A through Z. These variables contain the value of the most recently parsed address codes. #4101 through #4126 correspond to A through Z and are mapped in the same manner as the address codes are (see the Local Variables section). For example, to know what the last commanded S code was, inspect #4119.

IF [#4119 GT 1500] GOTO ??? (excess spindle RPM)

### Target Work Coordinate Position

These variables return the current position, in machine coordinates, of the specified axis. #5021 returns the machine coordinate of PMAC’s #1 axis, #5022 returns the machine coordinate of PMAC’s #2 axis, etc. Tool offsets are not included.

### Current Machine Coordinate Position

These variables return the current position, in work coordinates, of the specified axis. #5021 returns the work coordinate of PMAC’s #1 axis, #5022 returns the work coordinate of PMAC’s #2 axis, etc. Tool offsets are not included.

### Current Work Coordinate Skip Position

These variables return the most recent position sensed as a skip or trigger during a G31 move. The position is returned in work coordinates. #5061 returns the skip position of PMAC’s #1 axis, #5062 returns the skip position of PMAC’s #2 axis, etc. Tool offsets are not included.

### Current Tool Offset Applied

### Current Following Error

### Common Work Coordinates

These variables return the common work coordinates in effect at look ahead time. Fanuc also refers to these as external work coordinates. The common work coordinates can be modified in a G code program by assigning values to these variables. When these variables appear on the left of an assignment statement, the PC side look ahead queue is allowed to empty and the coordinates will change before further look ahead is allowed. #5201 corresponds to PMAC’s #1 axis, #5202 corresponds to PMAC’s #2 axis.

These variables do not refer to G92.

### G54, G55, G56, G57, G58, G59

### G54.1 P1..P48 extra offsets

Expressions

The evaluation of an expression is how data is created and how decisions are made in a parametric program. This section explains expressions. It defines how they are formed and where they can be used in a program.

An expression is made of three elements. These elements are operands, operators, and precedence brackets.
An operand is a variable or literal number. Variables appear as #<integer>. Literals are constants such as 1.0, 5, or 0.

An operator is a function that uses operands and derives a resultant operand. Operators are single character operators like + and /. They can be functions like SIN[ ] or LOG[ ]. Or operators can be conditional operators like EQ or GT.

Order of evaluation is from left to right. As an expression is evaluated from left to right, operations are either performed immediately or deferred based on the operator’s precedence.

Operators with higher precedence are executed first. So that in the expression #5=#4+ 5 * 9, #5 will be assigned the value of 49. This is because multiplication has a higher priority than addition and its operation is executed first even though the addition comes first in a right to left scan of the expression. Brackets can override the default order of evaluation determined by operator precedence. The precedence brackets are [and], (i.e. square brackets). The following table defines the precedence of operators in the Delta Tau control.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
<th>Precedence</th>
</tr>
</thead>
<tbody>
<tr>
<td>EQ</td>
<td>Equal</td>
<td>(cond.) 1</td>
</tr>
<tr>
<td>NE</td>
<td>Not equal to</td>
<td>(cond.) 1</td>
</tr>
<tr>
<td>GT</td>
<td>Greater than</td>
<td>(cond.) 1</td>
</tr>
<tr>
<td>GE</td>
<td>Greater than or equal to</td>
<td>(cond.) 1</td>
</tr>
<tr>
<td>LT</td>
<td>Less than</td>
<td>(cond.) 1</td>
</tr>
<tr>
<td>LE</td>
<td>Less than or equal to</td>
<td>(cond.) 1</td>
</tr>
<tr>
<td>+</td>
<td>Binary Addition</td>
<td>2</td>
</tr>
<tr>
<td>-</td>
<td>Binary Subtraction</td>
<td>2</td>
</tr>
<tr>
<td>OR</td>
<td>Bitwise Logical or</td>
<td>2</td>
</tr>
<tr>
<td>XOR</td>
<td>Bitwise Exclusive or</td>
<td>2</td>
</tr>
<tr>
<td>*</td>
<td>Multiplication</td>
<td>3</td>
</tr>
<tr>
<td>/</td>
<td>Division</td>
<td>3</td>
</tr>
<tr>
<td>AND</td>
<td>Bitwise Logical product</td>
<td>3</td>
</tr>
<tr>
<td>MOD</td>
<td>Remainder</td>
<td>3</td>
</tr>
<tr>
<td>+</td>
<td>Unary +</td>
<td>6</td>
</tr>
<tr>
<td>-</td>
<td>Unary -</td>
<td>6</td>
</tr>
<tr>
<td>POPEN</td>
<td>Peripheral I/O device open</td>
<td>7</td>
</tr>
<tr>
<td>PCLOS</td>
<td>Peripheral I/O device close</td>
<td>7</td>
</tr>
<tr>
<td>DPRNT</td>
<td>Print to Device</td>
<td>7</td>
</tr>
<tr>
<td>#</td>
<td>Indirect operation</td>
<td>7</td>
</tr>
<tr>
<td>ABS</td>
<td>Absolute value</td>
<td>7</td>
</tr>
<tr>
<td>ACOS</td>
<td>Arccosine</td>
<td>7</td>
</tr>
<tr>
<td>ASIN</td>
<td>Arcsine</td>
<td>7</td>
</tr>
<tr>
<td>ATAN</td>
<td>Arctangent</td>
<td>7</td>
</tr>
<tr>
<td>COS</td>
<td>Cosine</td>
<td>7</td>
</tr>
<tr>
<td>EXP</td>
<td>Exponential</td>
<td>7</td>
</tr>
<tr>
<td>FIX</td>
<td>Truncation (floor)</td>
<td>7</td>
</tr>
<tr>
<td>FUP</td>
<td>Round up (ceiling)</td>
<td>7</td>
</tr>
<tr>
<td>LN</td>
<td>Log (natural, base e)</td>
<td>7</td>
</tr>
<tr>
<td>ROUND</td>
<td>Round off</td>
<td>7</td>
</tr>
<tr>
<td>SIN</td>
<td>Sine</td>
<td>7</td>
</tr>
<tr>
<td>SQRT</td>
<td>Square root</td>
<td>7</td>
</tr>
<tr>
<td>TAN</td>
<td>Tangent</td>
<td>7</td>
</tr>
</tbody>
</table>

FANUC differs from the above table in that FANUC defines the conditional operators to have the same precedence as binary addition. If concerned about portability of the programs. Include precedence brackets around the operands of a conditional expression. For example:

[ 0.0 LT [#1+#2] ]
Examples of Expressions Follow:

#1   Singular expression
3.14159 Literal constant <literal>
#1/2  compound expression
[#1+#3]/2 compound expression with precedence override
[#1 NE #0] Logical expression
SIN[#1] / COS[#2] Expression using functions

When discussing the syntax of parametric programming, identify how expressions can be used. In the following, general expression is identified, as in the examples above, by <expr>.

There are four places where expressions can be used in a parametric program. Expressions are used in the following syntactical forms:

- Assignment statements
- Address codes
- Conditional expressions
- GOTO expressions

They are defined below.

Assignment Statements <assign> allow variables to be modified. Assignment statements have the following form:

1.  #<integer>=<expr> ; Simple assignment
   
   A simple assignment explicitly states what variable to modify.
   
   Example:  
   
   #1=5.0;
   #3000=5(alarm);

   An indirect assignment states the variable to modify with an expression.
   
   Example:  
   
   #1=#0
   #2=1.1
   G0 X#1 Y#2 (same as G0 Y1.1)

   Address Code statements can use expressions in the following form:
   
   X<literal>  address code using literal value.
   X[<expr>]  address code using expression to define value.
   X-[<expr>]  address code negating value of expression.

   If an address code is followed by an expression that results in #0, undefined, the address code in the block is ignored.

   Example:  
   
   #1=#0
   #2=1.1

   Conditional Expressions make use of the following conditional operators.

   EQ  Equal to
   NE  Not equal to
   GT  Greater than
   GE  Greater than or equal to
   LT  Less than
   LE  Less than or equal to

   These conditional operators are binary operators that return a value of 1.0 or 0.0. If the condition represented by the operator is true, the value of 1.0 results. If the condition represented by the operator is false, 0.0 results.

   Thus, a conditional expression has the following general form:  [<expr>]

   In most cases, a conditional expression will be less general:
<expr> <cond> <expr> where:  <cond>:: EQ|NE|GT|GE|LT|LE

To maintain FANUC compatibility, this form of a conditional expression should be adhered to.

GOTO expressions <goto> can be followed by an expression. The form of the GOTO is explained in the next section.

**Program Control**

Parametric programming allows additional control of program processing. The following constructs, when combined, provide the NC programmer with complete flexibility and control of the program.

- **Branching**
- **GOTO**
- **Conditional block execution**
- **IF**
- **Iteration**
- **WHILE**
- **Pause or abort**
  - #3006=n(stop), #3000=n(alarm)

Branching is available with the GOTO statement. The GOTO statement must be followed by an expression that evaluates to the N code of a block. The block is searched for in the currently executing main program or subroutine.

Searching is performed in the following manner:

- If the expression evaluates to a positive number, the search starts from the currently parsed block and proceeds to the end of the program. If the target block is not found, then searching resumes from the beginning of the program and continues through to the current block.
- If the expression evaluates to a negative number, searching is performed in the reverse direction working toward the beginning of the program from the current block.
- If the search fails to find the target block number, an alarm is generated.
- If the block is found, program execution is transferred to that block.

The GOTO statement <goto> can be used in the following forms:
- **GOTO<integer>**  Branch searching forward
- **GOTO-<integer>** Branch searching backward
- **GOTO<expr>**  N code is derived from expression

The GOTO statement, when alone on a line, is called an unconditional branch. That is, the branch always occurs.

**Example:**

```
N10  GOTO20  (forward branch)
...
N20
...
N30  GOTO-20 (backward branch)
#1=10 (mystery program?)
#2=1
N10  #2=#2+#3*.5
N20  #2=#2-#2*.4
N30  #2=#2+#2*.3
N40  #2=#2-#2*.2
N50  #2=#2+#2*.1
#1=#1+10
GOTO#1 (example of <expr>)
N60  M0 (Pause so variables do not clear)
M30
```

Conditional block execution allows the programmer to execute a statement based on a conditional expression. This is accomplished with the **IF** statement. The **IF** statement has the following forms:

```
IF [<expr><cond><expr>] <goto>
IF [<expr><cond><expr>] THEN <assign>
```
In the above \(<expr><cond><expr>\) is a conditional expression containing one of the conditional operators EQ, NE, GT, GE, LT, LE. In general, the bracketed syntax can be any expression, but for FANUC portability, it is best to follow the above syntax. Conditional operators always return 0.0 or 1.0.

If the conditional expression evaluates to 0.0 (false), then the statement following the conditional expression is not performed and the next block is executed.

On true (not 0.0 and not #0), the statement following the conditional is performed. This means that any expression that evaluates to a non-zero value is considered to be true, not just 1.0.

Form 1 will branch conditionally, whereas Form 2 will perform the assignment statement. The keyword then is optional.

**Example:**

```plaintext
IF [#24 EQ #0] GOTO99 (no X argument);
N10 #1=0;
N20 #1=#1+1;
N30 IF [#1 LT 10] GOTO20 (looping);
IF [#500 NE 0.0] then #500=0.0 (assignment);
```

Iteration is available with the **WHILE** statement. The **WHILE** statement has the following form:

```plaintext
WHILE \(<expr><cond><expr>\) DO
...
ENDn
```

or:

```plaintext
DO
...
ENDn
```

Where: \(n=1..3\)

And **WHILE** can be replaced with **WH**

In the above, each **WHILE** statement must have matching **DO** and **END** words. The **DO** and **END** labels for any **WHILE** must match in number. For example:

```plaintext
WHILE [#1 GT 0.0] DO1
...
END1
WHILE [#500 NE #550] DO2
...
END2
```

**DO**, of the **WHILE** loop, will branch to the first block after its matching end block if the conditional statement preceding it is false; otherwise execution continues to the block just following the **DO** statement. If there is no conditional on the block, **DO-END** is looped through forever.

Each program, subroutine, or parametric subroutine can have up to three **WHILE** - **DO-END** loops active at any one time. The **DO** loops of the subroutines are all independent. This means that subroutines can be called from within a **WHILE** loop, and that subroutine will have three more **WHILE** loops available to it.

When the calling program is returned to, any **WHILE** loops that were active prior to the call are restored.

**WHILE** loops can be nested as follows:

```plaintext
N10 WHILE [#1 NE 0.0] DO1
N20 WHILE [#2 NE 0.0] DO2
N30 WHILE [#3 NE 0.0] DO3
N40 END3
N50 END2
N60 END1
```

**WHILE** loops can be repeatedly used in the same program:

```plaintext
N10 WHILE [#1 NE 0.0] DO1
N20 END1
...
N60 WHILE [#1 NE 0.0] DO1
```
Branding outside of a **WHILE** loop is allowed. Branching cannot be made into a **WHILE** loop. The following code is allowed:

```
N20 WHILE [#1 LT 10] DO1
N30 ...
N40 IF [#100 NE 1.0] GOTO 70
N50 ...
N60 END1
N70
```

**WHILE** loops cannot overlap. The following is incorrect code and will alarm.

```
WHILE [#1 NE 0.0] DO1
WHILE [#1 NE 0.0] DO2
END1
END2
```

Pausing or aborting the program is another form of program control. Writing to system variables #3006 and #3000 will accomplish this. Writing to system variable #3006 can generate a program stop. An alarm can be generated, and servos stopped, by writing to #3000. Refer to the System Variables section for more on #3000 and #3006.

```
N10 #3006=1(turn part over);
N20 #3000=1(G43 is not invoked);
```

**Formatted Output**

Formatted Output is the way that a part program can send ASCII text strings to a serial port or a file. The program can generate reports as a part is run, allowing for run time generation of positional data. This data can be used later for quality assurance. The following is a list of Fanuc compatible commands that Delta Tau’s NC control supports.

**DPRNT**

This sends out ASCII text or formatted variables to the current output file or device. The current output file is determined by a registry entry as described in the Integration section. In addition, the output file can be set in the motion applet under the probing tab. The syntax follows:

```
DPRNT [ <ASCII text>|<formatted variable>...]
```

Where:

- **ASCII text**: is A..Z, 0..9,*,,/,#,+,-

And

- **Formatted variable** has the form:  
  
  ```
  #<var integer>[<whole integer><decimal integer>]
  ```

  where:

  - **var integer** is any valid local or global variable number.
  - **Whole integer** is the number of places to reserve for the **Whole part** of a floating point number.
  - **Decimal integer** is the number of places to reserve for the **Fractional part** of a floating point number.

**Examples:**

Given:

```
#1   = 4.13
#24  = 7.9
#100= -5.9
```

Then

```
DPRNT[A=#1[22], X=#24[34] AND VARIABLE #100=#100[34]]
```

Sends out:

```
A=4.13, X=7.9000 and variable #100=-5.9000
```
Given:

\[
\begin{align*}
#9 &= 30.0 \\
#24 &= -.125 \\
#25 &= 1.0
\end{align*}
\]

Then

\[
\text{DPRNT}[G1 \ X#24[44] \ Y-#25[44] \ F#9[30]]
\]

Sends out:

\[
G1 \ X-0.1250 \ Y-1.0000 \ F30
\]

Given:

\[
\begin{align*}
#500 &= 100 \\
#501 &= #0
\end{align*}
\]

Then

\[
\text{DPRNT}[\text{THIS IS PART #501[30] OF #500[30].}]
\]

Sends out:

\[
\text{This is part 0 OF 100.}
\]

POpen

This prepares or opens the output file or device for output. It should be included prior to any DPRNT statement for Fanuc compatibility.

PClos

This closes any output device opened by POPen. For FANUC compatibility, it should be used before terminating a program.

The proper sequence for POPen, PClos and DPRNT is illustrated in the skeleton subroutine below.

O999 (DPRNT UTILITY)

POpen (Open device for output)

. .

DPRNT[... ] ;

DPRNT[... ] ;

. .

PClos

M99

Parameter Display

The parameter display is an important tool for developing and determining if a parametric program is working as expected. The programmer can use the parameter display to view local and global parametric variables. The operator of the machine tool can view the value of these variables at any time, regardless of the level of parametric subroutine nesting that the program is at. The user is allowed to change the values of these parameters, thus giving the program a rudimentary form of operator input. Below is a picture of the first page of local variables on the Parameter display. Access the parameter display by pressing F7, the DIAG soft key. It is arranged as two columns of 17 fields, enough room for the 33 local variables. The first field of the first column is the real time value of the G65 nesting level.

To see the variables for the current nesting level, make sure that the page being viewed matches this number. As PMAC-NC is shipped, there are five pages of local variables corresponding to nesting levels 0 to 4 and 8 pages of global variables. The parameter display is a powerful tool for viewing system parameters.
**Note:**
The display can be modified to show any address in dual-ported RAM or within the PMAC. To see how this can be done, review the commentary at the beginning of the pages.dat file located in the starting directory of the ncui32.exe application. It is strongly recommended that the end-user not change the pages.dat file. The machine tool OEM or integrator should process the setup.

---

**M Code Aliasing**
This section describes the parametric programming feature of aliasing codes to G65 Pnnnn calls. This feature is available on the NCUI32 product. It allows the NC programmer to alias Address codes to a G65 Pnnnn macro subroutine or an M98 subroutine.

**General Concepts**
Address code aliasing is a powerful feature that allows the NC programmer to modify the behavior of the standard codes supplied by the OEM manufacturer. Standard G-code programming and familiarity with parametric programming is required to modify standard codes. As an example, suppose a programmer wanted to increment a part counter every time M30 was executed. The programmer could increment the counter prior to every M30 as in the following code:

```
#510=#510+1 (part counter)
M30
```

If this approach were used, the programmer would have to modify all existing programs. But with code aliasing he can redesign M30 to increment the part counter automatically. In this case no programs have to be modified. The programmer would just alias M30 to a program that executes the above code.

Currently, in the Delta Tau control, only M codes can be aliased.

**Aliasing M Codes**
M Address codes can be aliased by adding an M code definition into the alias.ini file. The definition must be under the section titled M_CODE_G65_ALIAS in the alias.ini file. It must have the following format:

```
Mnnn.nn= <program specifier>
<program specifier> := <Ocode> or <file name>
```

where: nnn - is a positive integer from 0..999 (M98 and M99 cannot be aliased.)
PMAC NC for Mill Application

- **nn**: is an optional extended M code specifier
- **Ocode**: is a positive 5 digit integer corresponding to a program number. The program is found in the current working directory and has the form of `Onnnnn.nc`
- **file name**: - if the Ocode form is not used, a file name can be specified. Any file name with path can be specified. If the file name does not include a path, the current working directory will be searched.

Examples of properly formed definitions follow:

- `M30 =9021`
- `M9 =O9022.nc`
- `M5 =C:\Nc_Programs\Aliases\M5.nc`
- `M3.1=9022`

Only ten M codes can be aliased at a time. Any additional definitions in alias.ini beyond the first ten are ignored. Any ill-formatted definitions are ignored. A control alarm will not be received when a definition is ill formatted. Test any M code definition that has been added to the alias.ini file.

When an M code is aliased, the M code is replaced with `G65 <program specifier>`, where `<program specifier>` is either Pnnnn or a comment indicating what the filename is. An aliased M code must be the first address code on the line after any N, L, or O code.

Any address code following the aliased M code will be passed to the macro subroutine as arguments. This is just as in the case of a standard G65 macro subroutine call.

An aliased M code is not sent for processing (added to the rotary buffer). If any aliased M code program or any subroutine called from that program has the same M code as that being aliased, then that M code is treated as a normal M code definition and it is sent to the control for processing (down the rotary buffer).

The alias.ini file is more flexible than Fanuc’s method of aliasing M codes. Fanuc is limited to using programs O9020 to O9029. Limit the aliased M code programs to these values if concerned with maintaining Fanuc compatible code. In addition, Fanuc is limited to M codes 1 to 97.

**Example:**
The following is an example of how to alias M6 to do tool management. In this example, tools 1 through 10 are for program use and are considered slot 1 tools. Tools 11 through 20 are backup tools and are considered slot 2 tools. When the main tool (1-10) exceeds its wear limit, then the back up tool will be used. This program can be expanded upon and can be made more complete.

**alias.ini definition file**

```ini
[M_CODE_G65_ALIAS]
M06=9026
```

**Program 09026.nc**

```c
O9026 (Modifies M6 Tnn behavior)
(assume that the tool wear was measured prior to this tool change and the tool length wear was updated)
(variable 551..559 contains the current tool being used, 1 or 11, 2 or 12, etc.)
(check for valid tool number)
IF [[#20 LT 1] OR [#20 GT 10]] GOTO97
N5
#3=1 (first slot attempted)
#1=#[550+#20] (get the current tool slot)
IF [[#1 EQ #20] OR [#1 EQ [#1+10]]]] GOTO10
#1=#20 (use slot 1 as default)
N10 (test wear)
#2=#[2200+#1] (get the tool length wear)
IF [#2 GT .0015] GOTO30
```

**Parametric Programming** 67
N20 (use the tool in the current slot)
#10[#550+#20]=#1  (save which slot is being used)
M06  T#1
M99
N30  (try the other slot)
IF  [#3 GE 2] GOTO98
#3=#3+1 (second slot attempted)
IF  [#1 LT 11] GOTO40 (current is slot 1)
#1=#20
GOTO10 (try slot 1)
N40
#1=#20+10
GOTO10 (try slot 2)
N97 #3000=9026(alarm: illegal tool number)
N98 #3006=9026(stop: replace first slot tool specified in argument 20)
(Remember to set the tool length wear to zero)
#10[#550+#20]=#20
GOTO5
%

**Integration**

This section describes system parameters that are used by parametric programming. If these parameters are set incorrectly, parametric programming may not exhibit correct behavior or it may not work at all. In the following read BASE1 as HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\Pmac and read BASE2 as HKEY_CURRENT_USER\Software\Delta Tau.

1. Set bPCdoesMacroCall in the registry to 1. This registry variable is found in the registry path \BASE1\Device0\Nc0\Code.
2. Set UseNewDiagnostics to 1. This registry variable is found in the registry path \BASE2\NCUI 32.
3. Ensure that CallHighSpeedMachProg is set to 1065. This registry variable is found in the registry path \BASE1\Device0\Nc0\Code\Group0
4. Ensure that macroCall is set to 65. This registry variable is found in the registry path \BASE1\Device0\Nc0\Code\Group0.
5. BlockLookAheadDefault is set to 3 by default. This registry variable is in the registry path \BASE1\Device0\Nc0\SYSTEM. This variable controls the PC side look ahead parsing of G code and it is not related to PMAC’s look ahead. When set to 3, it conforms to the traditional Fanuc look ahead amount. When this variable is set to 0, PC side look ahead is wide open. That is, G code blocks will be parsed ahead in the PC as far as memory permits. Look ahead can be limited when required, by placing G103 P3 into the G code. G103 P3 limits block look ahead to three blocks. Alternately, the default can be left set at 3 and G103 P0 can be placed into the G code.
6. The DPRNT output file is specified in \BASE1\Device0\Nc0\Probing\ReportFile.
TOUCH PROBING FOR PMAC-NC

PMAC-NC Probing Appendix

This section outlines in detail the necessary steps to implement Renishaw or Marposs probing on the PMAC-NC. It assumes that there is a working CNC installation to communicate with PMAC via the PEWIN executive.

Hardware Integration Manual

This section explains the differences that must be understood when installing probing on a Delta Tau control. It explains and outlines the hookups necessary for the probing interfaces. After reading this section, the implementers should be able to install a spindle probe or table probe and have it working and ready for use and testing with Delta Tau’s Software.

Differences Between PMAC and PMAC2

PMAC and PMAC2 have different capabilities when it comes to capturing current position data through the internal position capture register. This impacts probing because probing makes use of the triggering mechanism available with position capture. (See the PMAC manual for further information.) The PMAC has limited inputs available for triggering the position capture latch. As a result, in the PMAC, the home flag must serve double duty as both the input for the probe skip signal and input for the home flag. Since these functions are mutually exclusive, there is no problem. One does not probe while homing nor home while probing. The PMAC2 has additional inputs for triggering the position capture register. The PMAC2 uses the USERn flag for probe skip signal input. It is easier to wire and implement probing on the PMAC2. However, there is no difference in the functionality between PMAC and PMAC2 when the probing interfaces are fully integrated.

The following paragraphs outline the wiring for a PMAC and a PMAC2. Identify which PMAC version is being used and review the section that is appropriate for the installation. To find the type of PMAC being used, enter the TYPE command within the PMAC Executive.

PMAC Wiring for Skip Signal

This section explains how to setup a PMAC to accept a skip signal for probing and how to wire it to use the HMFLn flag for position capture.

Although the PMAC-NC package ensures that PMAC is properly configured for probing, PMAC-NC may not be running when hooking up the probe. The following is a list of I-Variables and the values to set them to for wiring and testing probe installation.

Encoder n capture flag control
I903=0  - X-axis uses Home flag to trigger position capture.
I908=0  - Y-axis uses Home flag to trigger position capture.
I913=0  - Z-axis uses Home flag to trigger position capture.

When the probe Skip signal occurs, all enabled axes must be triggered to capture their current position. Use the following wiring to make this occur.

![Probing Wiring Diagram]
PMAC2 Wiring for Skip Signal

This section explains how to setup a PMAC2 for probing and how to wire it to use the USERn flag for position capture.

Although the PMAC-NC package ensures that PMAC is properly configured for probing, PMAC-NC may not be running when hooking up the probe. The following is a list of I-Variables and the values to set them to for wiring and testing probe installation.

Encoder n capture flag control

I913=2 X axis uses USER flag to trigger position capture.
I923=2 Y axis uses USER flag to trigger position capture.
I933=2 Z axis uses USER flag to trigger position capture.

When the probe Skip signal occurs, all active axes must be triggered to capture their current position. Use the following wiring to make this occur.

**Note:**

Currently the G31 is configured to command axes X, Y, and Z. If motion on an axis is not specified, the control will command PMAC to the same location. If any axis is not receiving its skip signal, there may be a dwell after the skip has occurred. This indicates a line break. In addition, if planning to use only one or two axes and not wiring an axes capture flag, then modify the G31 code in mill.g by removing the unused axes in the move-until-trigger command.

Installing a Spindle and Table Probe

This section describes what is required for connecting an optical interface to a Delta Tau control.

After the PMAC side of the SKIP signal input is available, the machine side needs to be addressed. This is different depending on what signaling method the machine is using whether it is optical or hardwired, and who the manufacturer is. Refer to the probe manufacturers wiring specifications when wiring the probes. Here sample diagrams are included for both Renishaw and Marposs probe systems.

It is convenient if one wiring harness is used to collect power and signal wires entering and leaving the PMAC-NC. Below is a suggestion for a probe wiring harness using a DB-9 connector.

<table>
<thead>
<tr>
<th>DB-9 Connector Pin Assignments</th>
</tr>
</thead>
<tbody>
<tr>
<td>1- Black AGND (PMAC analog ground) (out)</td>
</tr>
<tr>
<td>2- Red Table Probe Select Signal (out) \ Signal pins</td>
</tr>
<tr>
<td>3- Green Skip Signal (in) /</td>
</tr>
<tr>
<td>4- White Start Enable Signal (out) -</td>
</tr>
<tr>
<td>5- Not Used</td>
</tr>
<tr>
<td>6- Black Common 24V (out) -</td>
</tr>
<tr>
<td>7- Green Chassis Ground (out) \Power pins</td>
</tr>
<tr>
<td>8- Red + 24V (out) /</td>
</tr>
<tr>
<td>9- White Available + -</td>
</tr>
</tbody>
</table>
**Optical Interface**

Two relays must be available. One relay for the Start Spindle Probe signal sent to the optical interface and another relay for selecting the table probe. Use discrete input 13 for the Start Spindle Probe signal and discrete input 14 for selecting the table probe. Any two relays can be used providing the defined variables used by M codes supplied with PMAC-NC are modified to reflect this. As shipped, PMAC-NC uses these relays by default.

**Renishaw**

This wiring chart is based on the Renishaw MI-12, the OMM, and the M18 interface unit. The chart addresses both the spindle and table probe. The pin designations use cc(pp) where cc is the component and pp is the pin on the component.

**Renishaw Wiring Chart Key Components**

| PP       | Probe Harness Power Pin          | OW = OMM wire |
| SP       | Probe Harness Signal Pin         | TW = Table probe wire |
| OI       | Optical Interface Pin (MI12)    |
| HI       | Hardwire Interface Pin (M18)    |
| MI12 SW3 | set to 1-2-3-4                   |
| MI12 SW2 | set to 1-2-3-4                   |
| M18      | Output set to 1 (N/C normally closed) |

**Probe Signal Pins**

| SP( 1) - Black - OI(23) | (AGND Analog ground) |
| SP( 2) - Green - OI(23) | (Table probe select) |
| SP( 3) - Red  - OI(24)  | (SKIP signal) |
| SP( 4) - White - OI(21)  | (Probe Enable) |

**Probe Power Pins**

| PP( 6) - Black - OI(17)  | (Common 24V) |
| PP( 7) - Green - OI(18)  | (Chassis ground) |
| PP( 8) - Red  - OI(16)  | (+24V) |

**OMM Wires**

| OW - Yellow - OI( 1) |
| OW - Grey - OI( 2) |
| OW - White - OI( 3) |
| OW - Green - OI( 4) |
| OW - Brown - OI( 5) |
| OW - Ylw/Grn - OI(18) | (Shield) |

**Table Probe Interface**

| HI(A1) - Blue - TW |
| HI(A2) - Red - TW |
| HI(A3) - - TW | (Shield) |
| HI(A4) - - OI(18) | (Chassis ground) |
| HI(A5) - Red - OI(16) | (+24V) |
| HI(A6) - Black - OI(17) | (Common 24V) |
| HI(B6) - Green - OI(24) | (SKIP signal, through relay M949) |
| HI(B7) - Yellow - OI(23) | (AGND Analog ground) |

**Miscellaneous**

| OI(17) - Black - OI(22) |

**Marpess**

This wiring chart is based on the Marposs E83 optical transmission system. The chart addresses both the spindle and table probe. The pin designations use cc(pp) where cc is the component and pp is the pin on the component.
Marposs Wiring Chart Key Components

PP = Probe Harness Power Pin
SP = Probe Harness Signal Pin
OI = Optical Interface Pin (E83)

E83 Interface Dip Switch Set TO 0-0-1-1-0-1-0-1-0-1

(Where the right most digit is switch 1)

Probe Signal Pins
SP( 1) - Black   - OI(12)
SP( 2) - Green   - OI(25)
SP( 3) - Red     - OI(13)
SP( 4) - White   - OI(11)

PROBE POWER PINS
PP( 6) - Black   - OI(17)
PP( 7) - Green   - OI(18)
PP( 8) - Red     - OI(16)

OMM WIRES
OW     - White   - OI( 1)
OW     - Purple  - OI( 2)
OW     - Green   - OI( 3)
OW     - Blue    - OI( 4)
OW     - Red     - OI(19)
OW     - Black   - OI(20)

TABLE PROBE INTERFACE
OI( 9) - Red     - TW
OI(10) - Blue    - TW

Miscellaneous
OI(16) - Grey    - OI(26)
OI(12) - Black   - OI(14)

Software Integration Manual

This section describes the installation and testing of the probing software. Before testing the probe, install the latest version of the software required for probing. Before loading the software, review the section describing the variables and the variable map for the probing software. If the application uses any of these areas, address these issues. Modify the application rather than the probing software.

Probing Memory Map

The following list identifies PMAC variables reserved for use by probing software. Note the extensive use of P-Variables for probing.

M0..500 (Standard Delta Tau reserved M codes)
P100..199 (Probing variables)
P250..499 (Standard Delta Tau reserved P codes)
P500..599 (Probing variables)
P600..799 (G65 nesting stack space for probing variables)
P950..989 (G65 Miscellaneous variables)
P990..1023 (32 bit selection mask)
Q0..350 (Standard Delta Tau reserved Q codes)

Installation of Software

Execute the following procedure to ensure that all of the software required to run probing is installed. Install the probing in a working CNC system. The following is a list of software required for probing. If probe ready software is available then all of the files should be configured to install and work as is.
Files Required for Installation
Renish.pmc  FANUC to PMAC translated probing routines.
Probe.plc   Initializes probe, monitors skip signal.
Errors.dat  Contains text of alarms specific to probing.
Dprnt.enu   Contains Text of Dprnt statements.
Mill.g      With G31, G65, G103 installed.
Mill.m      With M51 (Probe On),
M52 (Table probe select)
M62 (Spindle probe select, default)
and  M61 (Probe Off).
Mill.t      Updated T codes that allow M06 to operate within
G65 motion program.
Oem.h       1.68 or later.
Registry    default report file.
Registery   G65 definition.

1. Inspect the *.cfg file for the machine. If adding probing to an existing system, ensure that the *.cfg
   file includes probe.pmc  and probe.plc.
2. Within PEWIN, save a copy of the current Configuration. If something goes wrong, use this to
   restore a working environment.
4. If there are any problems downloading the *.cfg file, it is because an old copy of PEWIN is being
   used. Locate a copy of PEWIN in which conditional compilation is working properly.
5. Enable PLCs by typing ENABLE PLC 1..31 and I5=2.
6. Save the configuration to flash by typing SAVE in PEWIN.

Testing the Probe
At this point, the probing hardware is ready to be tested. Before any probe cycles can be run, ensure that
the Skip signal is being received by PMAC and that PMAC is responding to the skip signal. First verify
that the spindle probe and the table probe generates a proper signal when triggered. This can be done in
PEWIN through the executive. Then ensure that servos stop when a trigger occurs. This is best done
through the NC interface using G31 for testing.

Use the following procedure to test for proper operation of the probe:

Testing for a SKIP Signal
1. Enter PMAC-NC.
2. Home all axes.
3. Select the spindle probe by executing M62 from MDI.
4. Enable the spindle probe by executing M51 from MDI.
5. Exit the NC interface and PEWIN, the executive.
6. Inspect M5 either via a watch window or by typing M5.
7. Deflect the spindle probe with a finger.

Inspect M5 or type M5 again.
The value of M5 will change as the probe is toggled. If not, check the wiring and the battery in the probe.
The probe interface should not be in an error condition.

Set the interface as a normally open circuit:
1. Check the dip switches. Check the voltage levels to the interface units.
2. Use the same procedure to ensure that the table probe is working. If both a spindle and table probe
   are connected to the same interface unit, ensure that both are functioning properly. Additional
   voltage shunted to a table probe may cause the spindle probe to stop functioning or vice-verse.
3. Stop the spindle probe by entering M61 from MDI.
4. Select the table probe by entering \textbf{M62} from MDI.
5. When sure that PMAC is receiving a Skip signal, test the \textbf{G31} command.

\textbf{Testing G31 Operation}
1. Enter the PMAC-NC interface.
2. Home all axes.
3. Ensure that the work cell is clear.
4. Place the probe into the spindle.
5. Select the spindle probe by executing M62 from MDI.
6. Enable the spindle probe by executing M51 from MDI.
7. Execute the following G code: G31 G91 X-5.0 F30.
8. During table feed, deflect the probe with a finger. The table should stop before reaching the 5.0 inch destination. This indicates that the spindle is functioning.
9. Perform a similar test on the table probe.

\textbf{Machining Center Operators Manual}

\textbf{Probing Cycles}
This section describes the probing cycles that are supplied with the Delta Tau control. It is currently a subset of probing routines supplied with Renishaw probes. Probing cycle software supplied by Renishaw and Marposs are usually written in the Fanuc extension to G-code, referred to as MACROS. Delta Tau translates Fanuc macro statements into native PMAC code and stores them as subroutines which can be called with a \textbf{G65} command.

The probing cycles are divided into the following categories:

\textbf{Calibration}
- Spindle probe length calibration
- Spindle probe stylus offset from spindle center
- Spindle probe ball radius
- Spindle probe ball radius (vector measuring)
- Table probe length calibration
- Table probe X-Y calibration

\textbf{Safe Axis Movement}
- Protected positioning

\textbf{Measurement}
- Surface measure of X, Y or Z planes
- Web/Trough measurement
- Bore/Boss measurement
- Internal corner location
- External corner location
- Tool length setting
- Tool diameter setting

\textbf{Vector Measurement}
- Deviation of distance to angled surface in the X-Y plane
- Width deviation of an angled web or pocket
- Three point bore/boss measurement

\textbf{Miscellaneous}
- Bolt pattern bore/boss measurement
- Calculate feature to feature distance
- X-Y plane surface angle measurement
**Calling Method**

Use \texttt{G65 P....} to invoke the probing cycle. The system will stop look ahead based on an internal list of O codes designated as programs that must stop look ahead. In the descriptions that follow, any address code that follows \texttt{G65 P....} is an argument to the cycle called by P. Optional arguments are designated by brackets [ ].

**Calibration**

These probing macros are used to ascertain dimensional features of the probe. The features are stored and later used by various probe cycles to adjust for these attributes.

**Spindle Probe Length Calibration**

This cycle moves the Z-axis toward a reference surface. When the probe is triggered, the machine coordinate is saved and stored as the probe calibration length in the offset designated by the currently active H code.

\begin{align*}
\text{G65 P9801 Z T} \\
\text{T} & \quad \text{The current active tool offset H code (usually the tool that the spindle probe resides in) and must be the same as the active H code.} \\
\text{Z} & \quad \text{Reference plane value in the current work coordinate (typically Z0).}
\end{align*}

**Program Example:**

\begin{verbatim}
N9801(Test Probe Length Calibrate)
G90 G80 G40 G0
M19     (Orient Spindle)
G54 X1.1 Y0   (X-Y Position Above Plane)
G43 H1 Z.75    (Z Position, Select Offset)
G65 P9810 Z.25 F30.  (Protected Move Closer)
G65 P9801 Z.0 T1   (Calibrate Length)
G65 P9810 Z.25   (Protected Away)
\end{verbatim}

**Spindle Probe Stylus Offset from Spindle Center Calibration**

This cycle determines how far the probe stylus ball is from the spindle centerline. A calibration ring should be placed in the work cell where the diameter and exact location of the center is known. Optionally a pre-machined bore can be used. The probe must be positioned exactly on the bore center inside the bore. The cycle moves the X and Y axes from the center of the calibration ring until the bore surface is found. Upon completion of the cycle, the X and Y stylus offsets are retained for use in other probing cycles. The X axis offset is stored in variable P502 and the Y axis offset is stored in variable P503.
G65 P9802 D [Z]
D  Actual bore diameter (average)
Z  Current work coordinate Z-axis measuring position when measuring an external boss (Omit if a bore is being measured).

Program Example:
N9802  (Test Probe X-Y Offset Calibrate)
G90 G80 G40 G0
M19  (Orient Spindle)
G54 X0 Y0  (X-Y Position Above Plane)
G43 H1 Z.75  (Z Position, Select Offset)
G65 P9810 Z-.2 F30.  (Protected Move Closer)
G65 P9802 D2.0  (Calibrate Stylus Offset)
G65 P9810 Z.25  (Protected Away)

Spindle Probe Ball Radius Calibration
This cycle determines the probe stylus radius (from stylus center) for the X and Y directions. A calibration ring should be placed in the work cell where the diameter and approximate center location is known. The probe must be positioned inside the bore. The cycle moves X and Y-axes to determine the two radii. Upon completion of the cycle the X and Y stylus radii are retained for later use. The X-axis +/- radius is stored in variable P500 and the Y-axis +/- radius is stored in variable P501.

G65 P9803 D [Z]
D  Actual bore diameter (average)
Z  Current work coordinate Z-axis measuring position when measuring an external boss (Omit if a bore is being measured).

Program Example:
N9803 (Test probe ball radius calibrate)
G90 G80 G40 G0
M19  (Orient spindle)
G54 X0 Y0  (select tool offset, Z approach)
G65 P9810 Z-.2 F30.  (protected move)
G65 P9803 D2.0  (calibrate stylus radius)
G65 P9810 Z.25  (protected depart)

Spindle Probe Ball Radius (Vector Measuring) Calibration
This cycle determines the probe stylus radius (from stylus center) for the X and Y directions. In addition, eight other directions are determined in order to calculate accurate data for vectored measuring cycles. A calibration ring should be placed in the work cell where the diameter and approximate center location is known. The probe must be positioned inside the bore. The cycle moves X and Y-axes to determine the ten radii. Upon completion of the cycle, the radii are retained for later use. The X-axis +/- radius is stored in variable P500 and the Y-axis +/- radius is stored in variable P501. Additional radii are stored according to the table below.
G65 P9804 D [Z]
D Actual bore diameter (average)
Z Current work coordinate Z-axis measuring position when measuring an external boss (Omit if a bore is being measured).

Program Example:
N9804(Test Probe Vector Stylus Radius Calibrate)
M19 (Orient Spindle)
G90 G80 G40 G0
G54 X0 Y0
G43 H1 Z.75 (Select Tool Offset, Z Approach)
G65 P9810 Z-.2 F30. (Protected Move)
G65 P9804 D2.0 (Calibrate Stylus Radii)
G65 P9810 Z.25 (Protected Depart)

Table Probe Length Calibration
This cycle determines the actual machine location for the Z-axis surface of a table probe. The calibration is accomplished using a calibration arbor of precise known length. To calibrate, place the arbor in the spindle and position it directly over and on the center of the table probe stylus surface. The cycle touches off the stylus three times. When the calibration is complete, the stylus surface location is stored in PMAC variable P520.
G65 P9851 K
K   Calibration length of reference arbor (ZREF)

Program Example:
N9851 (Calibrate Arbor Length To Table Probe)
G90 G80 G40 G0
G49  (Cancel Tool Length Offset)
G54 X0 Y0  (Position Above Table Probe)
G65 P9851 K6.0  (Calibrate Length, Arbor is 6.0 Inches)

Table Probe X-Y Calibration
This cycle determines the size of the probe stylus using a reference arbor of precise known diameter. The
reference probe is placed into the spindle and positioned above the probe stylus approximately on center.
The cycle touches off two sides of the probe stylus and determines the exact width of the stylus at the tool
measure position. The width is stored in variable P522 for use when measuring tool diameters.

G65 P9852 S K [Z]
K   Table probe stylus diameter (average)
S   Calibration arbor diameter
Z   Incremental distance from start plane to touch off point (optional, default distance is -.4 inch)

Program Example:
N9853 (Calibrate Arbor Diameter To Table Probe)
G90 G80 G40
G49               (Cancel Tool Length Offset)
G54 X0 Y0         (Position on center of tool test position)
G65 P9852 S1.1255 K.500 Z-.8

Safe Axis Movement
Protected Positioning
Typically, this cycle utilizes G31 to move to a location. It is programmed just like G31. If the path is
obstructed, then motion is stopped with an alarm, whereas G31 will go on to the next block and will not
alarm. It is used when moving about the work cell with the probe in the spindle. If the probe is deflected
because an object hits it in an unanticipated manner, then all motion will stop.

G65 P9810 X Y Z [F]
F   Modal feed rate for the interpolated move (Once set, it is not necessary to place in each protected
positioning block.)
X Y Z   Axis destination linearly interpolated at feed F.
**Program Example:**
N9810 (Protected positioning moves)
G90 G80 G40
G65 P9810 X0 Y0 F30. (approach position)
. (do some probing)
G65 P9810 X10. Y5.0 (depart position)

**Measurement**
These probing cycles are the most commonly used measurement cycles. They have the ability to optionally set tool or work offsets, record true position, or print output, based on input arguments supplied through the application.

**Surface Measure of X, Y or Z Planes**
This cycle measures the position of a surface parallel to the X Y or Z plane. The surface can be used to set a tool offset. It can be used to adjust a work offset. The surface position can be reported to a file.

**G65 P9811 X | Y | Z [F Q S T W]**
- **F** Modal feed rate for the interpolated move (Once set, it is not necessary to place in each protected positioning block.)
- **Q** Maximum search distance beyond target surface before alarming.
- **S** Work offset to adjust to surface S1=G54 through S6=G59 (The work offset is adjusted by the error from the programmed value.)
- **T** Tool offset to update when probe trigger occurs.
- **W** Print results to the currently selected output file.
- **W1** Increment the feature number.
- **W2** Increment component number, set feature number to 1.
- **X Y Z** Axis destination linearly interpolated at feed F.

**Program Example:**
N9811 (single surface measure)
M19 (orient spindle)
G90 G80 G40 G0
G54 Y.5 (work coord, approach Y)
G43 H1 Z.75 (tool offset, approach Z)
G65 P9810 Z-.5 F50. (protected approach)
(FEATURE 1)
G65 P9811 Y0 W1 (report Y edge, should be 0)
G65 P9810 Z1.0 F50. (protected depart)
G0 Z.75
Web/Trough Measurement
This cycle measures the size of a web or trough feature parallel to the X or Y plane. The feature can be used to set a tool offset. It can be used to adjust a work offset. The feature size can be reported to a file.

G65 P9812 X | Y [Z F Q R S T W]
F  Modal feed rate for the interpolated move. Once set, it is not necessary to place in each protected positioning block.
Q  Maximum search distance beyond feature surface before alarming.
R  Added to X Y Size when determining decent position of probe. When R is +, the feature is a WEB. When - , the feature is Trough 2.
S  Work offset to adjust to surface S1=G54 .. S6=G59. The work offset is adjusted by the error from the programmed value.
T  Tool offset to update when probe trigger occurs.
W  Print results to the currently selected output file.
W1 Increment the feature number.
W2 Increment component number, set feature number to 1.
X Y The features nominal Size, web or trough, measured in X or Y.
Z  Absolute Z location to touch off. If not present Trough 1 is assumed.

Program Example:
N9812 (pocket measure)
M19 (orient spindle)
G90 G80 G40 G0
G54 X-1.25 Y-3.25 (work coord, X Y approach
G43 H1 Z.75 (select offset, Z approach)
G65 P9810 Z-.2 F50. (protected approach)(feature 2)
G65 P9812 X2.0 W1 (report on width of 2.0 pocket)
G65 P9810 Y-3.25 (protected depart)
**Bore/Boss Measurement**

This cycle measures the size of a bore or boss feature in the X-Y plane. Measurements are made at 0, 90, 180 and 270 degrees to determine feature size. The feature can be used to set a tool offset. It can be used to adjust a work offset. The feature size can be reported to a file.

![Diagram of Bore/Boss Measurement](image)

**G65 P9814 D [Z F Q R S T W]**

- **D** The features nominal diameter (DIA) bore or boss.
- **F** Modal feed rate for the interpolated move. Once set, it is not necessary to place in each protected positioning block.
- **Q** Maximum search distance beyond feature surface before alarming.
- **R** Added to X Y Size when determining decent position of probe. When R is +, the feature is a BOSS. When -, the feature is Bore 2.
- **S** Work offset to adjust to surface S1=G54 .. S6=G59. The work offset is adjusted by the error from the programmed value.
- **T** Tool offset to update when probe trigger occurs.
- **W** Print results to the currently selected output file.
- **W1** Increment the feature number.
- **W2** Increment component number, set feature number to 1.
- **Z** Absolute Z location to touch off. If not present Bore 1 is assumed.

**Program Example:**

```
N9814              (4-point bore measure)
M19                (orient spindle)
G90 G80 G40 G0     (work coord, X-Y approach)
G54 X-6.75 Y-1.25  (select tool offset, Z approach)
G43 H1 Z.75        (protected approach)
G65 P9810 Z-.2 F50. (report bore size)
G65 P9823 D2.0 W1  (protected depart)
G0 Z.75
```
**Internal Corner Location**

This cycle measures the location of an internal corner at the intersection of two planes. The planes do not have to be orthogonal (90 degrees) to each other. The feature can be used to set a tool offset. It can be used to adjust a work offset. The feature location can be reported to a file.

![Diagram of an internal corner measurement](image)

**G65 P9815 X Y [B F I J M Q S W]**

- **B**: Maximum allowable angular plane deviation of each surface (B5.0 is +/- 5 degrees).
- **F**: Modal feed rate for the interpolated move. Once set, it is not necessary to place in each protected positioning block.
- **I**: Incremental distance along X-axis to move for second touch off point.
- **J**: Incremental distance along Y-axis to move for second touch off point.
- **Q**: Maximum search distance beyond feature surface before alarming.
- **R**: Added to X Y SIZE when determining decent position of probe. When R is +, the feature is a Boss. When -, the feature is Bore 2.
- **S**: Work offset to adjust to surface S1=G54 .. S6=G59. The work offset is adjusted by the error from the programmed value.
- **W**: Print results to the currently selected output file.
- **W1**: Increment the feature number.
- **W2**: Increment component number, set feature number to 1.
- **X Y**: Expected work coordinate location of internal corner.

**Program Example:**

```
N9815  (internal corner)
M19   (orient spindle)
G90   G80  G40  G0
G54  X-7.3  Y-3.3  (coord system, X Y approach)
G43  H1  Z.75   (select offset, Z approach)
G65  P9810  Z-.2  F50.  (protected approach)
G65  P9815  X-7.75  Y-3.75  I.4  J.4  W1  (report SW internal corner)
G65  P9810  Z.75  F50.  (protected depart)
G0   Z.75
```
**External Corner Location**

This cycle measures the location of an external corner at the intersection of two planes. The planes do not have to be orthogonal (90 degrees) to each other. The distance to the first touch off point is the same distance from the probe start point to the X Y corner position. The feature can be used to set a tool offset. It can be used to adjust a work offset. The feature location can be reported to a file.

**G65 P9816 X Y [B I J Q S W]**

- **B** Maximum allowable angular plane deviation of each surface (B5.0 is +/- 5 degrees).
- **F** Modal feed rate for the interpolated move. Once set, it is not necessary to place in each protected positioning block.
- **I** Incremental distance along X axis to move for second touch off point.
- **J** Incremental distance along Y axis to move for second touch off point.
- **Q** Maximum search distance beyond feature surface before alarming.
- **S** Work offset to adjust to surface S1=G54 — S6=G59. The work offset is adjusted by the error from the programmed value.
- **W** Print results to the currently selected output file.
- **W1** Increment the feature number.
- **W2** Increment component number, set feature number to 1.
- **X Y** Expected work coordinate location of external corner.

**Program Example:**

```
N9816     (zero on ne external corner)
M19      (orient spindle)
G90 G80 G40 G0
G54 X.2 Y.2     (X Y approach)
G43 H1 Z.75     (Z approach)
G65 P9810 Z-.5 F50.   (protected approach)
G65 P9816 X0 Y0 I-.5 J-.5 S1 W1  (report NE corner)
G65 P9810 Z.75 F50.   (protected depart)
```
Table Probe Tool Length Setting
This probing cycle determines the length of a tool as referenced from machine zero. The cycle can be used to automatically set tool length offsets or to detect a broken tool. The tool must be positioned on center and just above the probe stylus prior to invoking the cycle. The table probe must be calibrated first.

```
G65 P9851 T [Q S Z]
Q  Maximum search distance beyond feature surface before alarming.
S  Tool diameter, for tools required to rotate (shell mills).
   Positive (+) for right handed tools.
   Negative (-) for left handed tools.
T  Tool offset number to update with length.
Z  Incremental depth for measurement from the start position.
   Default is -.4 inch.
```

Program Example:
```
N9851  (automatically set tool 8 length)
M52   (select table probe)
M19   (orient spindle)
G49   (cancel tool offset)
G54 X0 Y0 (select coord, approach stylus center)
G65 P9851 T8 Q4.0 (set tool length H8, allow four inches overtravel)
```
Tool Diameter Setting
This probing cycle determines the diameter of a tool. The cycle can be used to automatically set tool diameter offsets or to detect a broken tool. The tool must be positioned on center and just above the probe stylus prior to invoking the cycle. The table probe must first be calibrated.

G65 P9852 S D [Z R]
D  Tool radius offset number to update.
R  Radial clearance when traveling down the side of the stylus. Default .16 inches.
S  Tool diameter, for tools required to rotate (shell mills).
   Positive (+) for right handed tools.
   Negative (-) for left handed tools.
Z  Incremental depth for measurement from the start position. Default is -.6 inch.

Program Example:
G53 G1 X-10.0 Y0  (move to tool change position)
G53 G1 Z-4.37
T7 M6              (get tool 7)
G53 G1 Z-4.6       (move to clearance plane)
N9852              (automatically set tool 7 diameter)
M52                (select table probe)
M19                (orient spindle)
G90 G80 G40
G49                (cancel tool offset)
G54 X0 Y.2        (select coord, approach stylus center)
G65 P9852 S.5 D7  (set tool radius D7, .5 diameter tool)
Vector Measurement
These probing macros are used to measure features that are not normal (90 degrees) to the X, Y or Z axis. They involve the recording of some angle. In order to use these cycles, twelve additional points of the probe ball are used. These points are used to compensate for the probe ball not being on the center of the spindle. Before these cycles can be used, the spindle must be oriented and the probe calibrated for use with vector probing cycles.

Deviation of Distance to Angled Surface in the X-Y Plane
This cycle measures the distance to an angled surface. The surface is limited to angles in the X-Y plane. The angle is determined by the programmer. Deviations of the expected distance to the surface are reported.

By using two measurements along a known angle, the true angle and deviation can be determined. See Spindle Probe Ball Radius (Vector Measuring) Calibration section.

G65 P9821 A D [Q S T W]
A Assumed Angle of surface being measured (+/- 180.0 degrees).
D Expected distance to surface along angle A from start position.
Q Maximum search distance beyond target surface before alarming.
S Work offset to adjust to surface S1=G54 — S6=G59. The work offset is adjusted by the error from the programmed value.
T Tool offset to update when probe trigger occurs.
W Print results to the currently selected output file.
W1 Increment the feature number.
W2 Increment component number, set feature number to 1.

Program Example:
N9821 (test angled single surface measure)
G90 G80 G40 G0 (orient spindle)
M19
G54 X-.75 Y-.75 (X-Y approach)
G43 H1 Z.75 (Z approach)
G65 P9810 Z-.2 F50. (protected Z approach)
G65 P9821 A225. D3536 W1 (report distance to surface)
G65 P9810 Z.75 F50. (protected depart)
Width Deviation of an Angled Web or Pocket

This cycle measures the width of an angled web or pocket. The web/pocket is limited to angles in the X-Y plane. The angle is determined by the programmer. Deviations of the expected width are reported. By using two measurements along a known angle, the true angle and deviation can be determined. See the Spindle Probe Ball Radius (Vector Measuring) Calibration section.

![Diagram of Width Deviation of an Angled Web or Pocket]

G65 P9822 A D [Z F Q R S T W]

- **A**: Assumed Angle of web/pocket being measured (+/- 180.0 degrees).
- **D**: Expected width of feature.
- **F**: Modal feed rate for the interpolated move. Once set, it is not necessary to place in each protected positioning block.
- **Q**: Maximum search distance beyond feature surface before alarming.
- **R**: Added to D when determining decent position of probe. When R is +, the feature is a Web. When R is -, the feature is as Trough 2.
- **S**: Work offset to adjust to surface S1=G54 — S6=G59. The work offset is adjusted by the error from the programmed value.
- **T**: Tool offset to update when probe trigger occurs.
- **W**: Print results to the currently selected output file.
- **W1**: Increment the feature number.
- **W2**: Increment component number, set feature number to 1.
- **Z**: Absolute Z location to touch off. If not present Trough 1 is assumed.

Program Example:

```
N9822   (angled pocket measure)
G90  G80  G40  G0   (orient spindle)
M19   (X-Y approach)
G54  X-1.45  Y-1.65
G43  H1  Z.75   (Z approach)
G65  P9810  Z-.2  F50.   (protected approach)
G65  P9822  A-45.  D1.0  W1  (measure angled pocket)
G65  P9810  Z.75  F50.   (protected depart)
```
Three Point Bore/Boss Measurement
This cycle measures the diameter of bore or boss using three points specified by the programmer. The three points are specified using angles reference to the X+ direction. Deviations of the bore/boss can be reported. See the Spindle Probe Ball Radius (Vector Measuring) Calibration section.

G65 P9823 A B C D [Z F Q R S T W]
A  Angle of first touch off point taken from the X+ axis.
B  Angle of second touch off point taken from the X+ axis.
C  Angle of third touch off point taken from the X+ axis.
D  Diameter of bore or boss.
F  Modal feed rate for the interpolated move. Once set, it is not necessary to place in each protected positioning block.
Q  Maximum search distance beyond feature surface before alarming.
R  Added to D when determining decent position of probe. When R is +, the feature is a Boss. When R is -, the feature is as Boss 2.
S  Work offset to adjust to surface S1=G54 — S6=G59. The work offset is adjusted by the error from the programmed value.
T  Tool offset to update when probe trigger occurs.
W  Print results to the currently selected output file.
W1 Increment the feature number.
W2 Increment component number, set feature number to 1.
Z  Absolute Z location to touch off. If not present Bore 1 is assumed.

Program Example:
N9823 (3-point bore measure)
M19 (orient spindle)
G90 G80 G40 G0
G54 X-6.75 Y-1.25 (X-Y approach)
G43 H1 Z.75 (Z approach)
G65 P9810 Z-.2 F50. (protected Z approach)
G65 P9823 A0 B120.0 C240.0 D2.0 W1 (report bore diameter)
G65 P9810 Z.25 (protected depart)

Miscellaneous Macros
These cycles are various probing cycles that are useful yet do not fit into the above categories because they are unusual in some respect. Either they are more complicated or they consist of a combination of the above cycles to achieve some measurement. More probing functionality will be added to the Delta Tau control as the product matures.
**Bolt Pattern Bore/Boss Measurement**

**G65 P9819 C D Z [A B Q R W]**  Boss
**G65 P9819 C D K [A B Q R W]**  Bore

A  Angle at which first hole lies from X+ direction.
B  Total number of holes, default = 1.
C  Pitch diameter, diameter that holes lie on.
D  Bolt hole/boss diameter.
K  Absolute K location to touch off for a bore.
Q  Maximum search distance beyond feature surface before alarming.
R  Added to D when determining decent position of probe. When R is +, the feature is a Boss. When -, the feature is a Bore.
W  Print results to the currently selected output file.
W1 Increment the feature number.
W2 Increment component number, set feature number to 1.
Z  Absolute Z location to touch off for a boss.

**Program Example:**

N9814      (bolt-hole measure)  
M19       (orient spindle)  
G90 G80 G40 G0  
G54 X-4.0 Y-2.0  (X-Y approach)  
G43 H1 Z.75  (Z approach)  
G65 P9810 Z.75 F30.  (protected Z approach)  
G65 P9819 C2.0 D1.0 K-.4 A18.0 B5. W1 (measure 5 bores at 18 deg)  
G65 P9810 Z.75 F30.  (Z depart)  

**Calculate Feature to Feature Distance**

This cycle measures the distance between two independently measured features. No motion takes place when this cycle is invoked. P1 is designated as the first feature and P2 is the second. Distance is measured as P2 relative to P1. After P1 is measured with an appropriate cycle, the programmer invokes this cycle without any arguments. This stores the dimensional data recorded for P1. After the second feature is measured, using an appropriate cycle, this cycle is invoked again. The second invocation must have arguments which indicate what is to be calculated by the cycle. Angular tolerances as well as absolute distances can be measured and reported. Web/Trough measurement (9812) cannot be used with this cycle.
**PMAC NC for Mill Application**

**XY Plane**

![Diagram of XY Plane]

**G65 P9834**
**G65 P9834 X [S T W]**
**G65 P9834 Y [S T W]**
**G65 P9834 X Y [B S T W]**
**G65 P9834 A D [B S T W]**

A  Angle between features as taken from the X + axis (+/- 180.0 degrees).
B  The angular tolerance of A.
D  The minimum distance from P1 to P2.
S  Work offset to adjust to surface S1=G54 .. S6=G59. The work offset is adjusted by the error from the programmed value.
T  Tool offset to update when probe trigger occurs. T can be used only when P2 is measured using 9811 or 9821.
W  Print results to the currently selected output file.
W1 Increment the feature number.
W2 Increment component number, set feature number to 1.
X  The nominal incremental distance along X.
Y  The nominal incremental distance along Y.

**Program Example:**

```
N9834 (feature to feature measure)
G65 P9810 X-.5 Y2.0 F30. (X-Y approach wall)
G65 P9810 Z1.0 F30. (protected Z approach)
G65 P9811 X0 (measure X wall P1)
G65 P9834 (record P1)
G65 P9810 X-4.0 (X-Y approach boss)
G65 P9814 D.5 Z.5 (measure boss P2)
G65 P9834 X-4.0 H.02 W2 (measure and report tolerance)
```
Z Plane

G65 P9834 Z [S T W]
G65 P9834 A Z [B W]
G65 P9834 D Z [B W]
A  Angle between features as taken from the X + axis (+/- 180.0 degrees).
B  The angular tolerance of A.
D  The Minimum distance from P1 to P2 (+/-). Note that D is independent of X and Y.
S  Work offset to adjust to surface S1=G54 .. S6=G59. The work offset is adjusted by the error from the programmed value.
T  Tool offset to update when probe trigger occurs. T can be used only when P2 is measured using 9811 or 9821.
W  Print results to the currently selected output file.
W1 Increment the feature number.
W2 Increment component number, set feature number to 1.
Z  The nominal incremental distance expected in the Z axis.

Program Example:
N9834 (feature to feature measure)
G65 P9810 X1.5 Y0 F30 (X-Y approach P1)
G65 P9810 Z1.0 F30 (protected Z approach)
G65 P9811 Z.5 (measure Z AT P1)
G65 P9834 (record P1)
G65 P9810 X3.1 (X-Y approach P2)
G65 P9811 Z.25 (measure Z AT P2)
G65 P9834 D1.6 Z.25 B.5 W2 (measure and report tolerance)
X-Y Plane Surface Angle Measurement
This canned cycle measures two points along the X axis or the Y axis to determine the angular direction of a plane. The points are measured in the same Z plane equidistant on opposite sides of the start point.

G65 P9843 X | Y D [A B Q W]
A Angle of surface relative to the X + direction. Select A +/-90 from the direction that measuring takes place.
X direction, default for A is 90 degrees.
Y direction, default for A is 0 degrees.
B The angular tolerance of A.
D The distance between the two measuring points along the X direction if using Y, the Y direction if using X.
Q Maximum search distance beyond feature surface before alarming.
W Print results to the currently selected output file.
W1 Increment the feature number.
W2 Increment component number, set feature number to 1.
X X dimension that plane surface is expected to be. When X is selected, measurements are along the X axis.
Y Y dimension that plane surface is expected to be. When Y is selected, measurements are along the Y axis.

Program Example:
N9843 (angle measurement of Z plane)
G65 P9810 X0 Y2.0 F30. (X-Y approach midpoint)
G65 P9810 Z1.0 F30. (protected Z approach)
G65 P9843 X2.0 D.8 A60. W2 (measure angle)
G65 P9810 Z5.0 (Z depart)
Alarms
The following alarms have been added for probing cycles. The alarms are listed in alphabetical order according to the text seen on the alarms display. Following the text is the probable cause for the alarm along with the corrective action.

**Broken Tool**
In tool detection probing cycles, this indicates that the probe signal was not detected within the tolerance specified. Replace the tool or open the tolerance.

**A Input Missing**
An A address code is required on a G65 block.

**B Input Missing**
A B address code is required on a G65 block.

**C Input Missing**
A C address code is required on a G65 block.

**D Input Missing**
A D address code is required on a G65 block.

**Data #130-#139 Missing**
Indicates that the first feature in a feature-to-feature distance calculation was not recorded. P1 must be recorded prior to P2 with G65 P9834.

**Format Error**
Indicates that the combination of arguments used is not allowed. Refer to the operator’s manual for the proper format.

**G65 Address Code Missing**
Indicates that a G65 line was program without a P code in the block.

**G65 Nesting Level Exceeded**
Indicates that too many nested G65 calls were made. Nesting cannot exceed five levels.

**H Input Not Allowed**
H is not allowed in the G65 probing cycle. Refer to the operator’s manual.

**M Input Not Allowed**
M is not allowed in the G65 probing cycle. Refer to the operator’s manual.

**No Feed Rate**
Indicates that probing variable 117 has no value. No feed rate has been specified on any previous probing cycle.

**No Tool Length Active**
Group 8 is in G49 state meaning that no tool length is in effect. Specify a G43 or G44 prior to G65 call.

**Path Obstructed**
The probe was deflected during a protected positioning move. Change the programmed path, or check the hardware a spurious signal was detected.

**Probe Fail**
The SKIP signal indicates that the probe never deflected and it should have. The surface being probed was not encountered or the probe is not working. See the integration manual for testing of the probe. Make sure that the probing PLC is running.
**Probe Open**
The SKIP signal indicates that the probe is deflected and it should not be. Make sure that the probe is activated with the proper M code. Make sure that the hardware is functioning correctly. Make sure that the probe PLC is running. See the software integration manual for testing of the probe.

**Runtime Error**
May indicate that the G65 line contains a P address code in which the subroutine indicated by the P code does not exist as a motion program.

**S Input Not Allowed**
S is not allowed in the G65 probing cycle. Refer to the operator’s manual.

**SH Input Mixed**
S and H cannot be on the same G65 block. Refer to the operator’s manual.

**ST Input Mixed**
S and T can not be on the same G65 block. Refer to the operator’s manual.

**T Input Missing**
A T address code is required on a G65 block.

**T Input Not Allowed**
T is not allowed in the G65 probing cycle. Refer to the operator’s manual.

**TM Input Mixed**
T and M cannot be on the same G65 block. Refer to the operator’s manual.

**Tool Out of Range**
The tool diameter is larger than the diameter specified in variable 121. Use a smaller cutter or increase the value of variable 121.

**X Input Missing**
An X address code is required on a G65 block.

**XY Input Missing**
An X or Y address code is required on a G65 block.

**XY Input Mixed**
X and Y cannot be on the same G65 block. Refer to the operator’s manual.

**XYZ Input Missing**
A X, Y or Z address code is required on a G65 block.

**XYZ Input Mixed**
X, Y, and Z cannot be on the same G65 block. Refer to the operator’s manual.

**Y Input Missing**
A Y address code is required on a G65 block.

**Z Input Missing**
A Z address code is required on a G65 block.

**ZK Input Mixed**
Z and K cannot be on the same G65 block. Refer to the operators manual.