



DELTA TAU

Power PMAC-NC 2016

Motion Commander Foundation
© 2016 Greene & Morehead Engineering, Inc.



Software User Manual

Power PMAC-NC16



Delta Tau Data Systems, Inc.

December 8, 2016

0019-E-01

COPYRIGHT INFORMATION

Software: © 2014 Delta Tau Data Systems, Inc. All rights reserved.

Software User Manual: © 2014 Delta Tau Data Systems, Inc. All rights reserved.

Motion Commander Foundation: © 2012-2014 Greene & Morehead Engineering, Inc. All rights reserved.

This document contains proprietary information of Delta Tau Data Systems, Inc. The information contained herein is not to be used by or disclosed to third parties without the express written permission of an officer of Delta Tau Data Systems, Inc.

TRADEMARK ACKNOWLEDGMENT

Windows, Visual Studio and .NET Framework are registered trademarks of Microsoft Corporation. MTConnect is a registered trademark of the MTConnect Institute. Other brands, product names, company names, trademarks and service marks are the properties of their respective holders.

REVISION HISTORY

Version	Date	Description
1.0	4/1/2015	Initial release
1.11	2/25/2016	1.11 Code Release. Additional Canned Cycles Supported.
1.12	10/5/2016	Added New Message Capabilities.
	//2017	

Power PMAC-NC16™ - Software User Manual

Contents

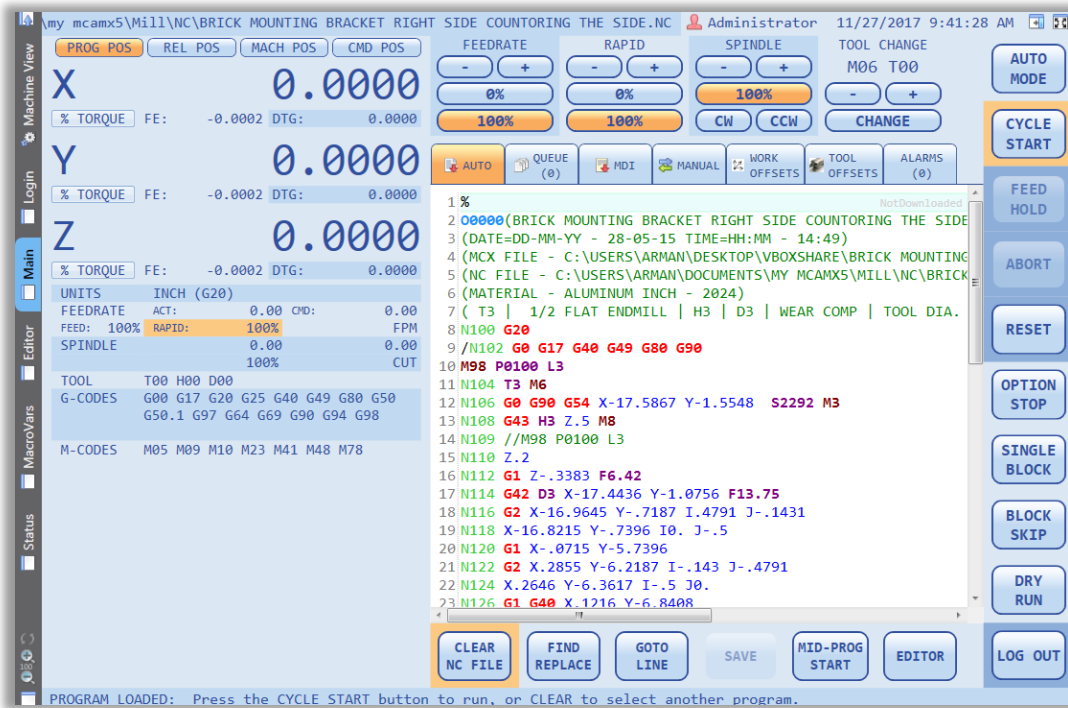
- Introduction 5
 - Versions..... 6
 - Motion Commander Foundation 6
 - Requirements..... 6
 - Installation 6
 - Deployment..... 7
 - The Power PMAC Project..... 7
 - Hardware Key..... 8
 - Configuration File..... 8
- Runtime Operation 9
 - Main Screen 9
 - Currently Loaded Program Display 11
 - Soft Panel 11
 - Login Display 12
 - Time/Program Elapsed Time display..... 13
 - Message Log Slider View..... 13
 - Full Screen Mode 14
 - Vertical Button Bar..... 14
 - Tabbed View Screen Selections 14
 - Program Editor 15
 - Run Screen 21
 - MDI Screen..... 23
 - Manual Mode Screen..... 24
 - Work Offset Screen 25
 - Tool Offset Screen..... 29
 - Alarms Screen 35
- Machine View 39
- Users **Error! Bookmark not defined.**
- Foreign Language Support 66
- Skins **Error! Bookmark not defined.**
- Ctrl and Shift Keys 68

NC Files.....	69
The NC File Parser	69
NC File Custom Pre-Parser	70
NC File Configuration	70
Subprograms	70
Native PMAC Commands and Expressions	71
G and M-Code Groups.....	72
Mid-Program Start	73
Fixed Cycles	75
Using M99 to Repeat the Main Program	76
The NC Program Queue	76
NC File Comments.....	77
NC File Size Limitations	77
Aliasing -#define and #include	78
Customizing the Application	79
Private Labeling.....	79
Messages.xml.....	80
External Assemblies	84
The Visual Studio Project Template	Error! Bookmark not defined.
Customized Parsing Using the NC File Pre-Parser.....	107
Telnet Server Support	113
Appendix A. The Source Files	114
Appendix B. The Configuration File.....	117
Appendix C. Turbo PMAC Support	121
The Turbo PMAC Project.....	121
Appendix D. Source Code Exclusions	122
Appendix E. Send1 Command List.....	123
Appendix F. Included G & M Codes.....	126

Power PMAC-NC 16™ - Software User Manual

Introduction

The *Power PMAC-NC 16* HMI (PPNC16) is a host PC application for Delta Tau Power PMAC controlled CNC machines. This document is the Software User Manual for the *Power PMAC-NC 16* application. It contains information about how to use the software, what features the software includes, and also describes what can be customized.




- Supports standard RS-274 style G-code programs as well as native Power PMAC Programs.
- Split screen Subprogram visualization with embedded and external subprograms supported.
- Configurable for 1-10 axes, type of application, and machine/velocity units.
- Software and Hardware Control Panel support built in.
- Secure SSH/SFTP communications with Power PMAC.
- Built-in Power PMAC command terminal and Linux terminal.
- Colorized NC file editor optimized for large files.
- NC file Execution Queue for remote or unattended machine automation.
- Real-time Execution Monitoring including Subprograms.
- Mid-Program Start Capability.
- Parametric Programming Capability.
- User Login system with Definable Feature Access.
- Built in Foreign Language Translation.
- MTConnect 2.0 Agent, HTTP Server, and Telnet Server for supervisory data collection.
- Fully portable application deployment (no installation required < 5MB total file size!).
- Fully customizable with the *Software Development Kit* (SDK) version.
- External assembly (plugin) system for custom Screens, Code Groups and other data.
- Custom Parser.

- Customizable color schemes and login screens for OEM branding.
- Private labeling including Login image, Company name, and Splash image.

Versions

Power PMAC-NC 16 is offered in two different versions - *Power PMAC-NC 16 SDK* and *Power PMAC-NC 16 Runtime*. *Power PMAC-NC 16 Runtime* is designed for users who do not have any intention of modifying the actual HMI screens or layout. *Power PMAC-NC 16 SDK* is designed for users who intend on customizing the HMI and includes extensive source code for this purpose.

 References to the folder paths of the two versions in this manual can be used interchangeably. For instance if we refer to `C:\..\GitHub\PowerPmacNc16-Runtime` you can assume the path will be `C:\..\GitHub\PowerPmacNc16-SDK` for SDK users.

Motion Commander Foundation

The Power PMAC-NC 16 program is based on the *Motion Commander Foundation* (MCF) .NET framework for machine control applications. The *Motion Commander Foundation Developer's Guide* is also included in the SDK for reference. <http://www.MotionCommander.com>



Requirements

The PPNC16 program is compatible with Windows 7 or newer (64-bit or 32-bit).

The application requires .NET Framework 4.6.1 and the Visual C++ 2010 runtime libraries. If .NET Framework 4.6.1 is missing, users will be asked to install it manually but Visual C++ 2010 library will be installed automatically. The following links can be used for manual installation of the same libraries.

Microsoft .NET Framework 4.6.1 (Web Installer)


<https://www.microsoft.com/en-us/download/details.aspx?id=49981>


Microsoft Visual C++ 2010 Redistributable Package (x64)

<http://www.microsoft.com/en-us/download/details.aspx?id=14632>

Microsoft Visual C++ 2010 Redistributable Package (x86)


<http://www.microsoft.com/en-us/download/details.aspx?id=5555>


 If you are using the SDK version Visual Studio 2013 or newer (Express or Professional) is required to build the application from the SDK source code.

 Some versions of the PPNC16 program also support the Turbo PMAC and are 32-bit applications, requiring the x86 version of the Visual C++ Redistributable - even on 64-bit Windows systems.

Installation




















The *PPNC16 Software* is distributed via a private GitHub repository and on media directly from Delta Tau Data Systems. In order to access the online repository sign up for a [free GitHub account](#) and provide the account name to **Delta Tau Data Systems** Power PMAC NC16 is purchased. Then, read-only access to the appropriate repository will be granted. The same repository can be used for future updates. Install [GitHub for Windows](#) on a development PC, log in, and "Clone" the repository. "Sync" occasionally to insure using the latest release version.

 It is highly recommended to make a working copy of *PPNC16* in order to avoid losing changes when “Sync” is requested. If a Sync fails for any reason, simply delete the entire “GitHub\PowerPmacNc16-Runtime” folder and Clone again.

 “PowerPmacNC.ini” file might be updated for any new features; compare to the old one and apply necessary changes. “Messages.xml” and “PowerPmacNC_Settings.xml” are required to be copied and pasted from the old folder to a new folder in order to apply same settings and messages to the updated version. If a modified version is used, make sure to copy and paste all required files.

Deployment

The *PPNC16* HMI can be deployed by simply copying the “PowerPmacNc16-Runtime” folder to any location on any desired machine. The folder may be renamed if desired. The distribution must include the files shown below. (Files not shown in this list may be deleted without affecting the application.)

Name	Type	Size
 Languages	File folder	
 MessageLogViewer.exe	Application	52 KB
 PowerPmacNC.exe	Application	974 KB
 PowerPmacNC.ini	Configuration settings	3 KB
 DynamicDataDisplay.dll	DLL File	350 KB
 ICSharpCode.AvalonEdit.dll	DLL File	612 KB
 MCF.CustomControls.dll	DLL File	12 KB
 MCF.DeltaTau.dll	DLL File	51 KB
 MCFoundation.dll	DLL File	1,084 KB
 Microsoft.WindowsAPICodePack.dll	DLL File	104 KB
 Microsoft.WindowsAPICodePack.Shell.dll	DLL File	530 KB
 Renci.SshNet.dll	DLL File	450 KB
 Routrek.Granados.dll	DLL File	136 KB
 SecureDongle_Control32.dll	DLL File	111 KB
 SecureDongle_Control64.dll	DLL File	146 KB
 PowerPmacNC.pdb	Program Debug Database	194 KB
 DeviceMembers.xml	XML Document	107 KB
 Messages.xml	XML Document	14 KB
 PowerPmacNC_Settings.xml	XML Document	21 KB

The Power PMAC Project

The PPNC16 application requires the Power PMAC controller be configured with its source code counterpart to enable proper functionality and handshaking. This Power PMAC project comes with the product.

The Power PMAC project will be located in the following folder:

“PowerPmacNc16-Runtime\PMAC Source Code\PowerPMAC\PPCNC_ProjectSource”

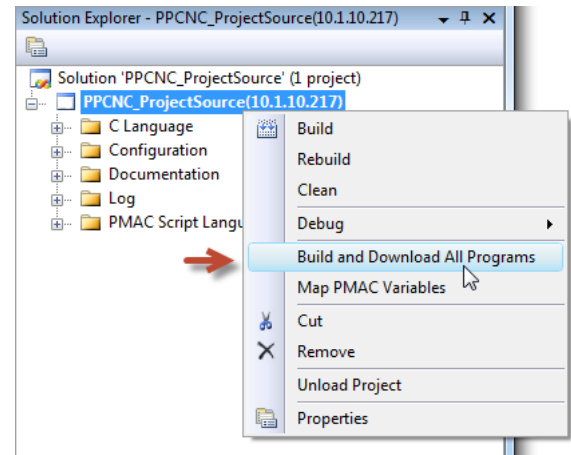
Make a working copy of this directory before you download the project to the controller.

Open the "PPCNC_ProjectSource.PowerPmacSuite_sln" solution file in the *Power PMAC IDE*, right-click and select "Build and Download" as shown. Look for the "Download Successful" message in the Output window.

```
Download Successful.
Total Project Download time = 13.057 sec
Total Project Build and Download time = 24.212 sec
```



After downloading the project, use the Terminal window to issue a "**save**" command to copy the project to nonvolatile flash memory, then issue a "\$\$\$" command to reset the controller.



At this point the Power PMAC controller is now ready to work with the *PPNC16* program in a virtual mode. Actual machine functionality will require the appropriate integration of the motors, I/O, safety systems, etc. The default Power PMAC code is used as a starting point for all machine integrations which will utilize the PPNC16 software.

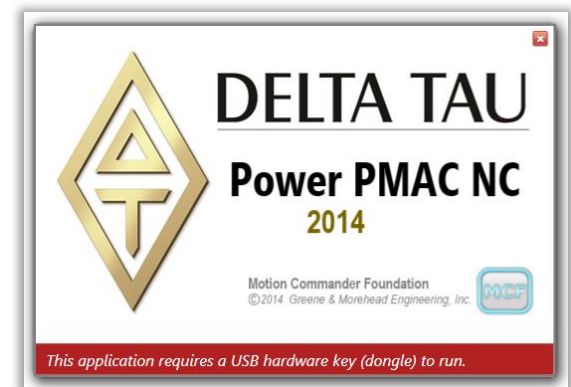


"ppnc_virtualmotors.pmh" file which is located in "Global Includes" folder contains configurations of eight virtual motors. Exclude this file from a project after making sure Power PMAC NC16 works properly and start implementing real motors configurations in a new header file.

Hardware Key

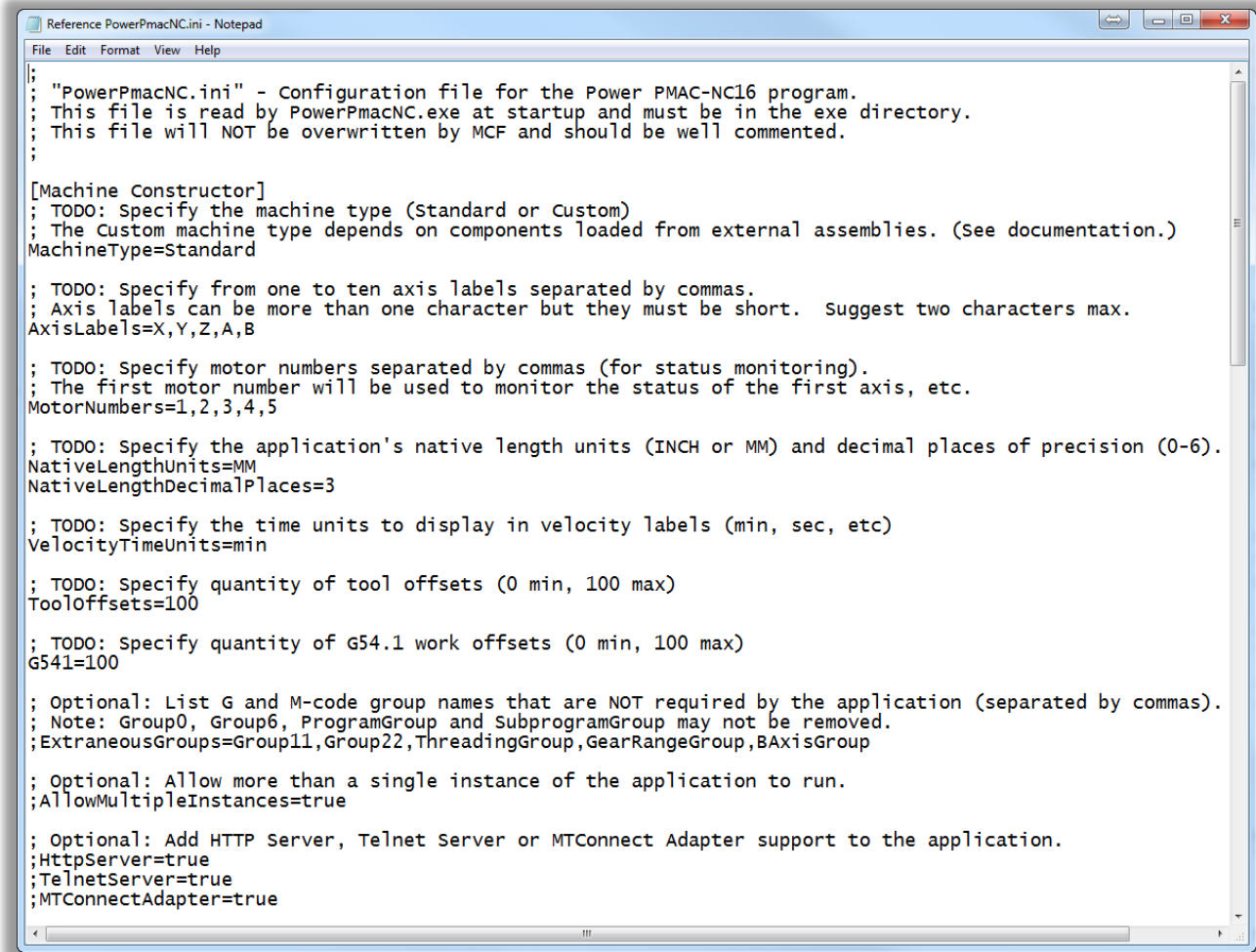
This application requires a USB hardware key (dongle) to run. Hardware keys will be included when copies of PPNC16 program are purchased from Delta Tau Data Systems.

The hardware key is compatible with all versions of Windows and does not require a driver to be pre-installed.



Configuration File

The application reads the "PowerPmacNC.ini" configuration file in its exe directory at start-up to obtain its configuration data. A *Reference* copy of this file is included in the project for convenience. For PPNC16 runtime, a Copy of "Reference PowerPmacNC.ini" is included in the exe directory. Rename it to "PowerPmacNC.ini", and edit it to specify machine type, axis definitions, units, velocity units, and other important parameters. For PPNC16 SDK, a Copy of "Reference PowerPmacNC.ini" is included in the solution directory. Copy it to a "Debug" folder that is located in a "bin" folder and rename it to "PowerPmacNC.ini", and edit it to specify machine type, axis definitions, units, velocity units, and other important parameters. The configuration file is well commented for convenience.



```
File Edit Format View Help
;
; "PowerPmacNC.ini" - Configuration file for the Power PMAC-NC16 program.
; This file is read by PowerPmacNC.exe at startup and must be in the exe directory.
; This file will NOT be overwritten by MCF and should be well commented.
;
;
[Machine Constructor]
; TODO: Specify the machine type (Standard or Custom)
; The Custom machine type depends on components loaded from external assemblies. (See documentation.)
MachineType=Standard

; TODO: Specify from one to ten axis labels separated by commas.
; Axis labels can be more than one character but they must be short. Suggest two characters max.
AxisLabels=X,Y,Z,A,B

; TODO: Specify motor numbers separated by commas (for status monitoring).
; The first motor number will be used to monitor the status of the first axis, etc.
MotorNumbers=1,2,3,4,5

; TODO: Specify the application's native length units (INCH or MM) and decimal places of precision (0-6).
NativeLengthUnits=MM
NativeLengthDecimalPlaces=3

; TODO: Specify the time units to display in velocity labels (min, sec, etc)
VelocityTimeUnits=min


; TODO: Specify quantity of tool offsets (0 min, 100 max)
ToolOffsets=100

; TODO: Specify quantity of G54.1 work offsets (0 min, 100 max)
G541=100

; Optional: List G and M-code group names that are NOT required by the application (separated by commas).
; Note: Group0, Group6, ProgramGroup and SubprogramGroup may not be removed.
; ExtraneousGroups=Group11,Group22,ThreadingGroup,GearRangeGroup,BAxisGroup

; Optional: Allow more than a single instance of the application to run.
; AllowMultipleInstances=true

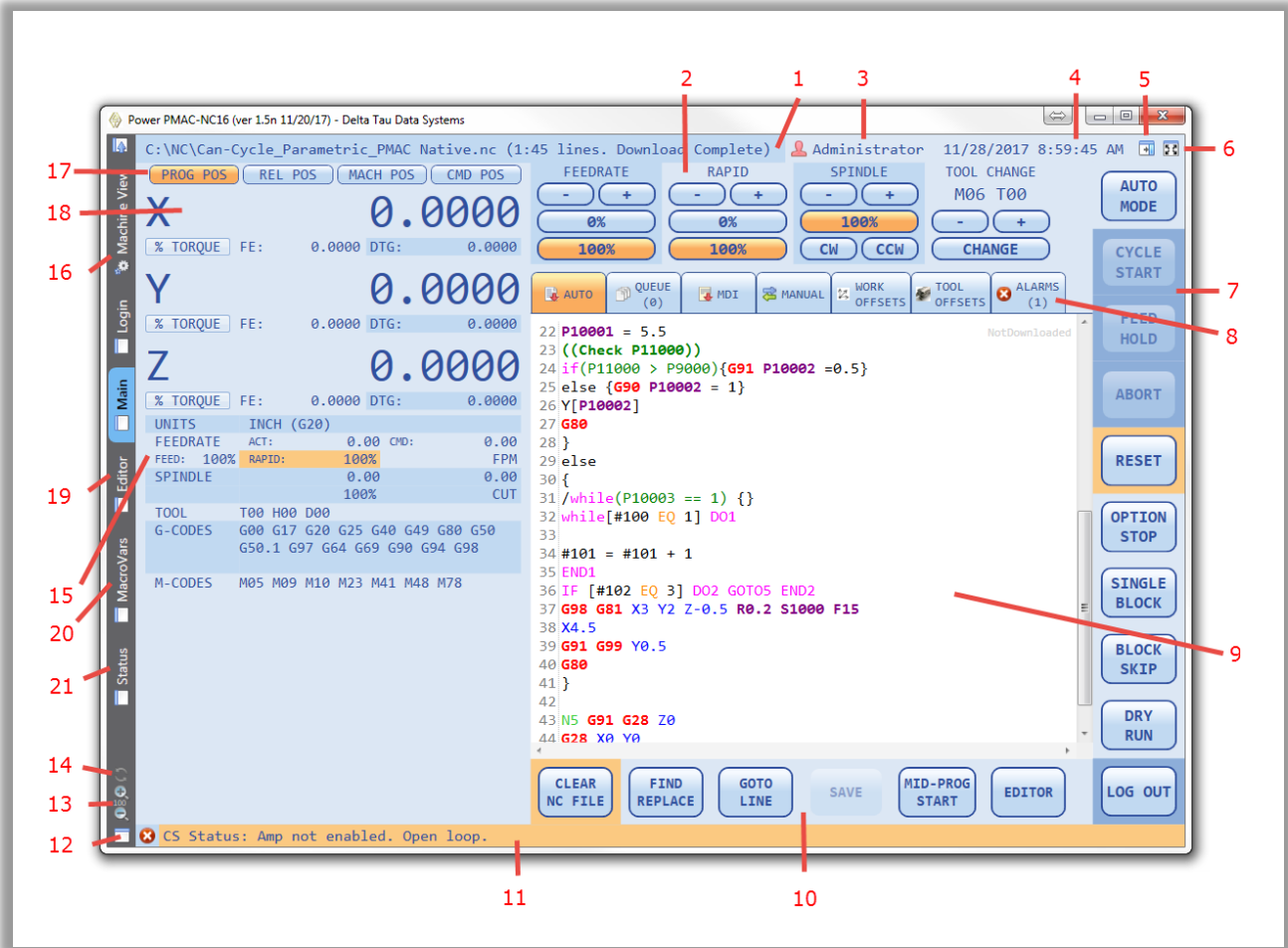
; Optional: Add HTTP Server, Telnet Server or MTConnect Adapter support to the application.
; HttpServer=true
; TelnetServer=true
; MTConnectAdapter=true
```

 A copy of configuration file is included in Appendix B of this manual for users' convenience as a reference.

Runtime Operation

Main Screen

The Main Screen serves as the base of operation during runtime. It is a modern feature rich implementation of a traditional CNC interface console with many added features specifically optimized for the Windows environment. Operators will find this screen intuitive and easy to use.

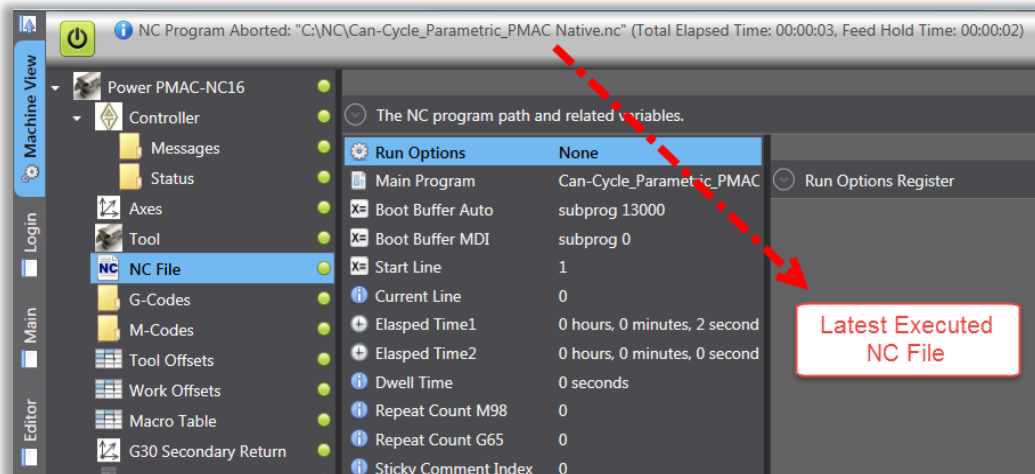
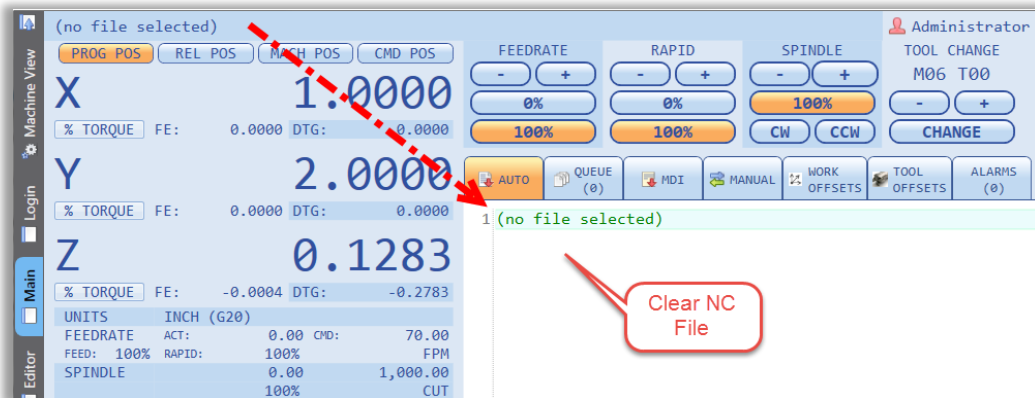


1. Currently loaded program and path display.
2. Soft Control Panel.
3. Login display, shows current user.
4. Time/Program Elapsed Time display.
5. Message log slider view.
6. Full Screen mode.
7. Vertical Button Bar - Main software operator controls.
8. Tabbed view screen selections.
9. Program editor and run screen.
10. NC File Control Button Bar
11. Message status bar.
12. Dual-Screen pop out control.
13. HMI scaling controls.
14. HMI watchdog indicator.
15. NC parameter display window.
16. Axis parameter display window.
17. Position mode selector buttons.
18. Machine view selection.
19. Full Screen Program Editor
20. Macro Variables Table
21. Coordinate System/Motor Status Window

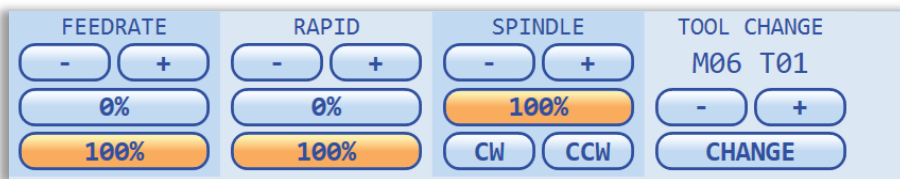
Currently Loaded Program Display

C:\NC\Can-Cycle_Parametric_PMAC Native.nc (1:45 lines. Download Complete)

The currently loaded part program will display in the upper left hand corner of the HMI. The part program name, path, and number of lines are displayed. Additionally the first line to execute after a cycle start will be displayed. This can be useful for mid-program starts. If the file being downloaded is large, the display will show the HMI is in process downloading. The PPNC16 will load the last file loaded on application startup. If the last file has been subsequently deleted, editor window shows “no file selected”. However if the file has been executed partially or completely, its path and status still will be displayed on “Machine View” top status bar as it is shown below:



Soft Panel




The soft panel is used to display and change Feedrate Override, Spindle Override, as well as control tool changes. If hardware control panel is presented, by default soft panel only shows the status of its members only such as zero or hundred percent override. If hardware control panel is not presented, soft panel can be used to show status as well as

controlling override and tool change functionalities. In certain situations the system integrator may find it useful to include both for specialty applications.

The soft panel can be added or removed from the Main Screen by changing the following element in “PowerPmacNC.ini” file:


```
[User Interface]
; Option to hide the Feedrate/Spindle/Tool Change controls on the main screen.
HideUpperControlPanel=false
```

 By default soft panel is shown at all times regardless of hardware control panel being presented or not.


The feedrate override can be adjusted in increments of 10%, or set to either 0% or 100% immediately using the buttons provided. If the machine is currently at 0% or 100%, the button will illuminate accordingly.

The spindle override can be adjusted similarly. Additionally there are modal buttons for CW/CCW spindle direction.




The Tool Change mechanism allows the operator to set the desired tool, and then initiate a tool change directly from the Soft Panel.

 By default, if hardware control panel is presented (PowerPendPresent = 1), feedrate/rapid/spindle speed override functionalities only can be done using hardware control panel.

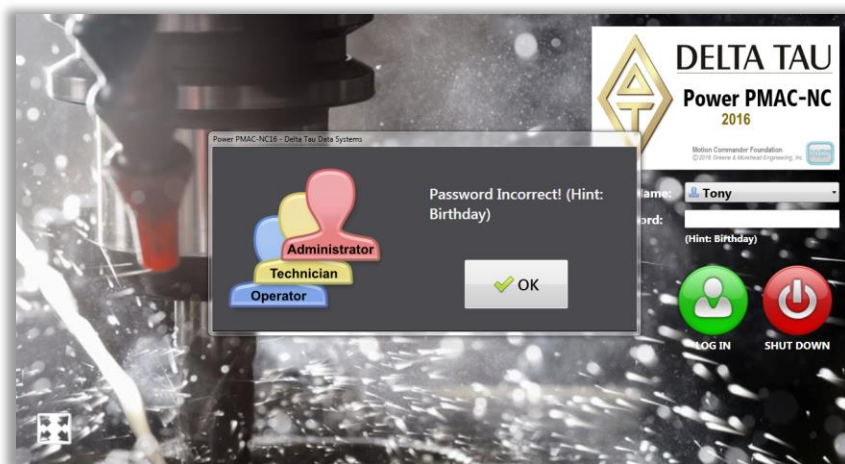
Login Display


 Cornelius Clemens

The Login Display will show the currently logged in machine operator and their user access level shown by color. There are three access levels as shown below.

 **The Administrator**
 **The Operator**
 **The Supervisor**

If the password for any user contains a “Hint”, it will be shown each time a “Password Incorrect” box is shown.



 Changing users' information (such as language, name, and etc) is only allowed when a user is logged in as an administrator.

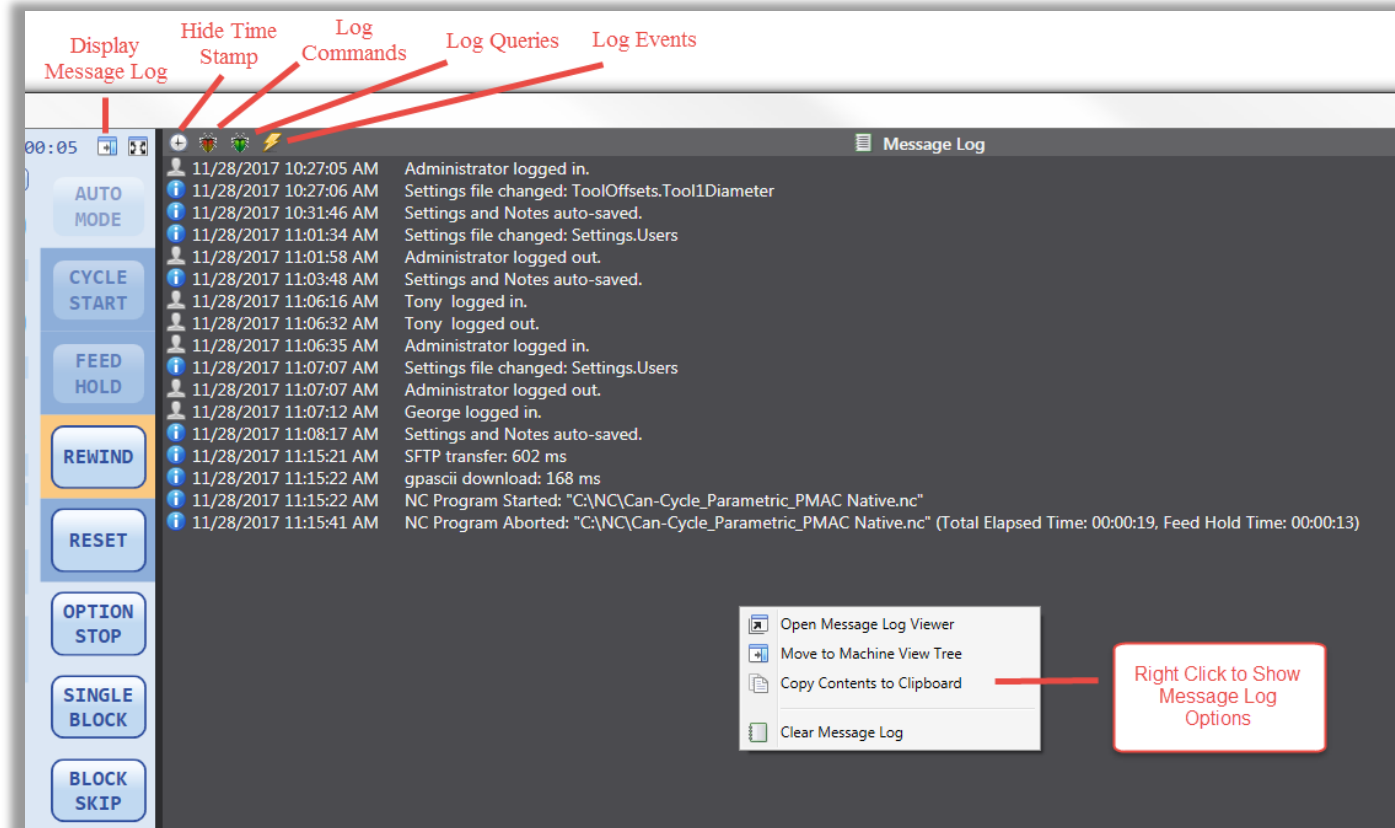
Time/Program Elapsed Time display

Elapsed Time: 00:01:59


4/30/2015 4:49:13 PM

The Time/Program Elapsed time display will toggle between real computer time and program elapsed time during program execution. In addition to this the automatic message log will keep track of program start time, end time, overall elapsed time, and actual elapsed time (non-feedhold time).

Message Log Slider View



The Message Log Slider view allows access to the Message Log directly from the PPNC16 Main Screen. The message log can be expanded and contracted as necessary for the operator to view relevant information. The time stamp can be hidden if the operator prefers a more compact view of the messages. The “Log Commands” tool will show every command being sent back and forth between the control and the HMI. The “Log Queries” tool will show statuses of different elements regardless of them being changed or not. The “Events” tool will show when a value of a member is changed based on a new event. This should only be used for troubleshooting purposes. The Message Log window can be cleared by selecting this option by right clicking in the message area and selecting this option.

 Clearing the messages from the Message Log window does not delete the information from the ongoing message log utility.

Full Screen Mode



Full Screen mode will extend the PPNC16 to the video resolution boundaries. The Windows task bar will be covered by the bottom portion of the PPNC16 application. The operator can either use the Window button or ALT-TAB to move to other applications if desired. If Full Screen mode is selected it will be retained through application shut down.

Vertical Button Bar

The main operator control buttons and mode display will be found in the Vertical Button Bar. These buttons will illuminate or fade depending on the operating mode and functionality. If a button is faded it cannot be used. The top mode button can be used to switch modes and view the current mode of operation.

Tabbed View Screen Selections

These tabs display the main screens which the operator utilizes during normal operation of the machine. The first three screens, AUTO, MDI, and MANUAL will switch the operational mode when selected.



Offset	X	Y	Z
G54	1.5040	2.0100	5.6230
G55	3.0230	5.0000	4.2516
G56	5.6323	7.2560	0
G57	0	0	0
G58	0	0	0
G59	0	0	0
G54.1 P1	0	0	0
G54.1 P2	0	0	0
G54.1 P3	0	0	0
G54.1 P4	0	0	0
G54.1 P5	0	0	0
G54.1 P6	0	0	0
G54.1 P7	0	0	0

SET WORK OFFSETS: (Manual Mode Only)

ALL X Y Z

Tool Index	Tool Length	Tool Wear	Tool Diameter	Diameter Wear
Tool 1	-8.2171	-0.0005	0.5000	-0.0010
Tool 2	-9.7016	-0.0002	0.5000	0
Tool 3	-10.9171	0	0.3750	-0.0001
Tool 4	-11.4810	0	0	0
Tool 5	-12.4895	-0.0002	0.1250	0
Tool 6	-13.5632	0	0.1250	-0.0006
Tool 7	-15.2360	0	0.1250	0
Tool 8	-12.2405	0	0	0
Tool 9	0	0	0	0
Tool 10	0	0	0	0
Tool 11	0	0	0	0
Tool 12	0	0	0	0
Tool 13	0	0	0	0

SET TOOL OFFSETS: (Manual Mode Only)

SET TOOL LENGTH

CS Error Status: Mismatch between # of motors used and # in lookahead buffer
5/1/2015 4:56:01 PM

Main Screen Program Editor

The PPNC16 includes a main screen editor which serves as the Run Screen as well. In edit mode NC codes are colorized by their code type (G, M, T, D, Comment, S-code, etc.). All programs lines are automatically pre-pended with line numbers which are used by the program for line display and mid-program starts. These auto-assigned line numbers do not conflict with CAM generated "N" line numbers in any way.

Cursor Position

Program Status

Auto Assign Line Numbers

ReadyToRun

Colorized Editor

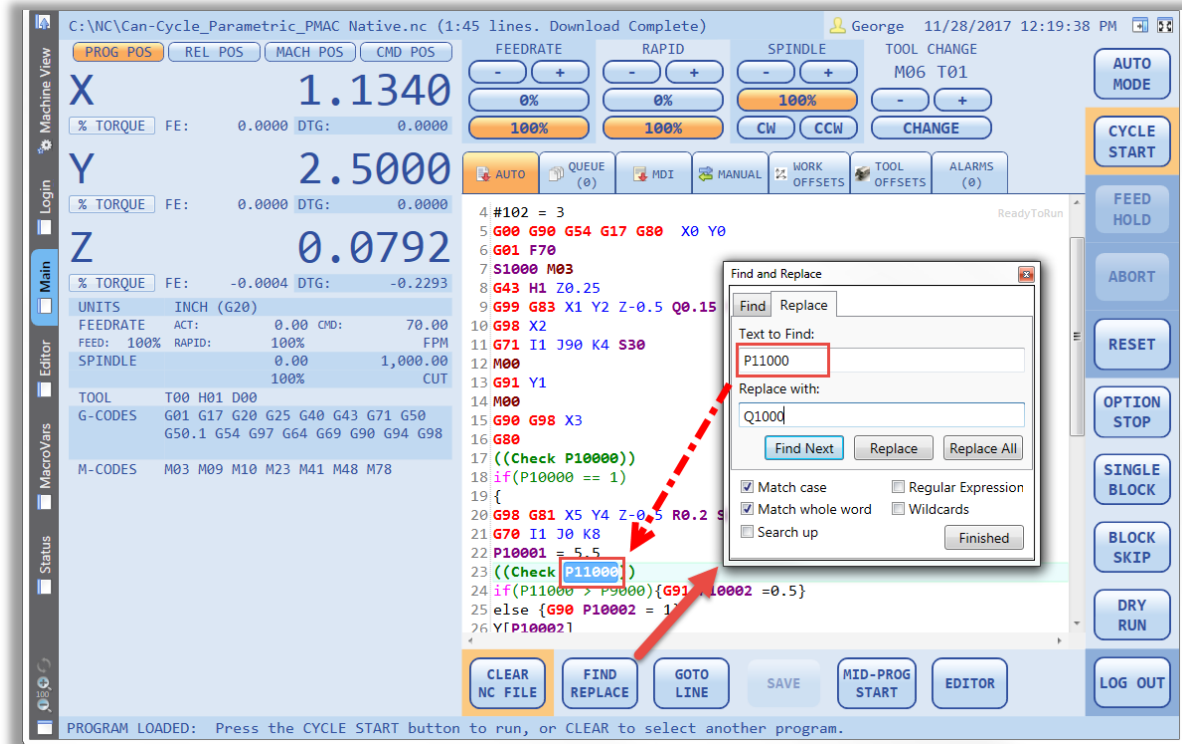
```

1 %
2 00000(MOLD_PUMPKIN1)
3 (DATE=DD-MM-YY - 24-10-13 TIME=HH:MM - 11:20)
4 (MATERIAL - ALUMINUM MM - 2024)
5 ( T1 | H1 | XY STOCK TO LEAVE - 1. | Z STOCK TO LEAVE - 0. )
6 G21
7 G0 G17 G40 G49 G80 G90
8 T1 M6
9 G0 G90 G54 X103.738126 Y30.758289 A0. S20000 M5
10 G43 H1 Z15.
11 Z5.
12 G1 Z.673184 F0.
13 X81.692036
14 G0 Z5.
15 Z15.
16 X75.699911 Y32.748606
17 Z5.
18 G1 Z.673184
19 X109.703385
20 X114.401006 Y34.738924
21 X75.784024

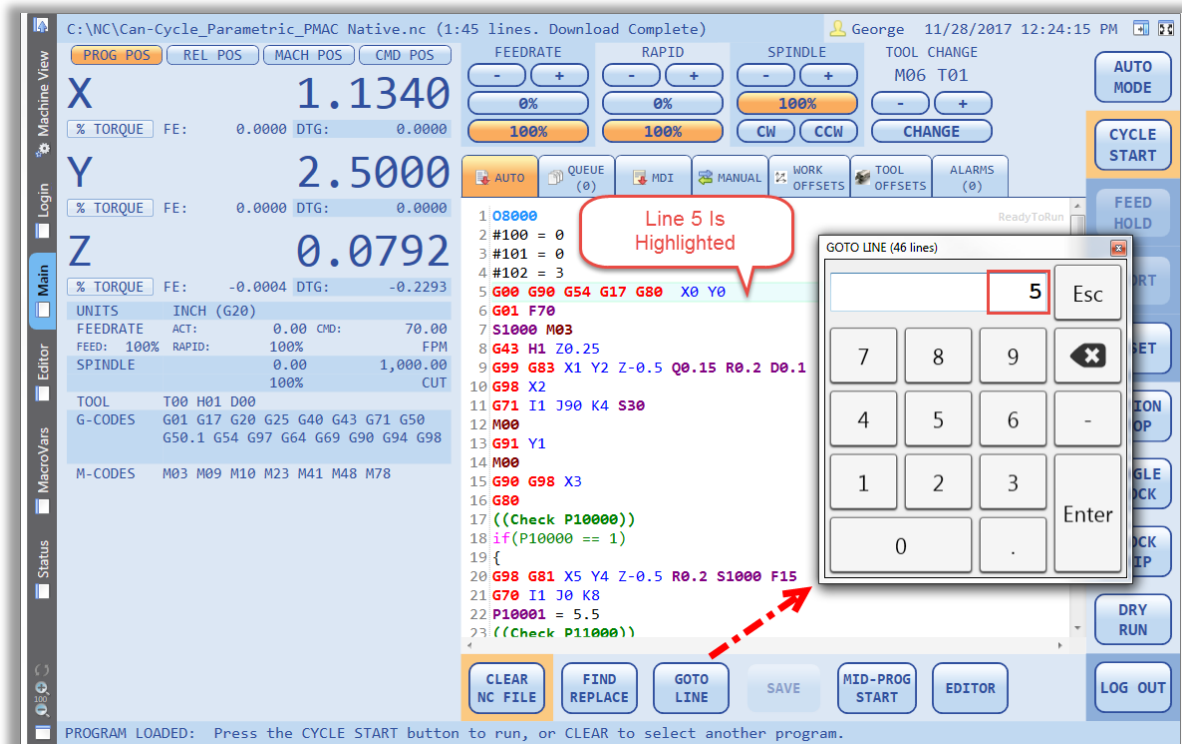
```

The download status of the part program is displayed in the upper right corner of the program editor. The following standard features are supported by the Program Editor:

- Load/Clear NC File
- Find Replace

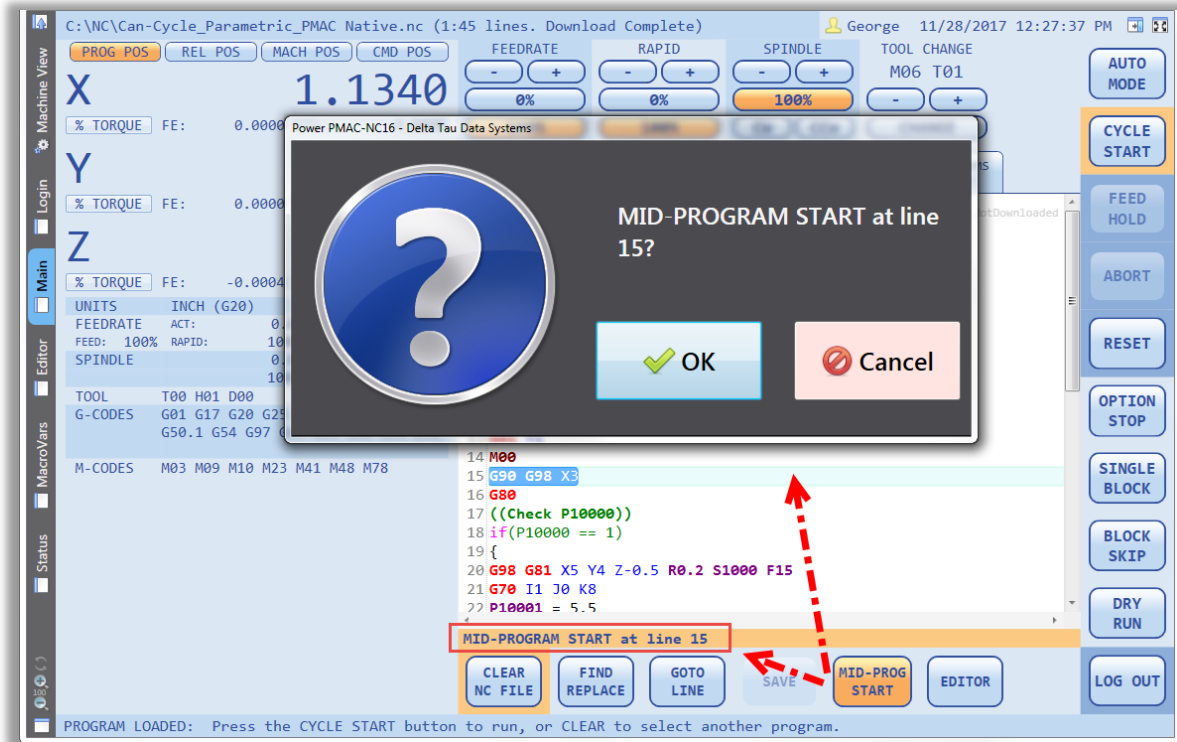


- Goto Line Number

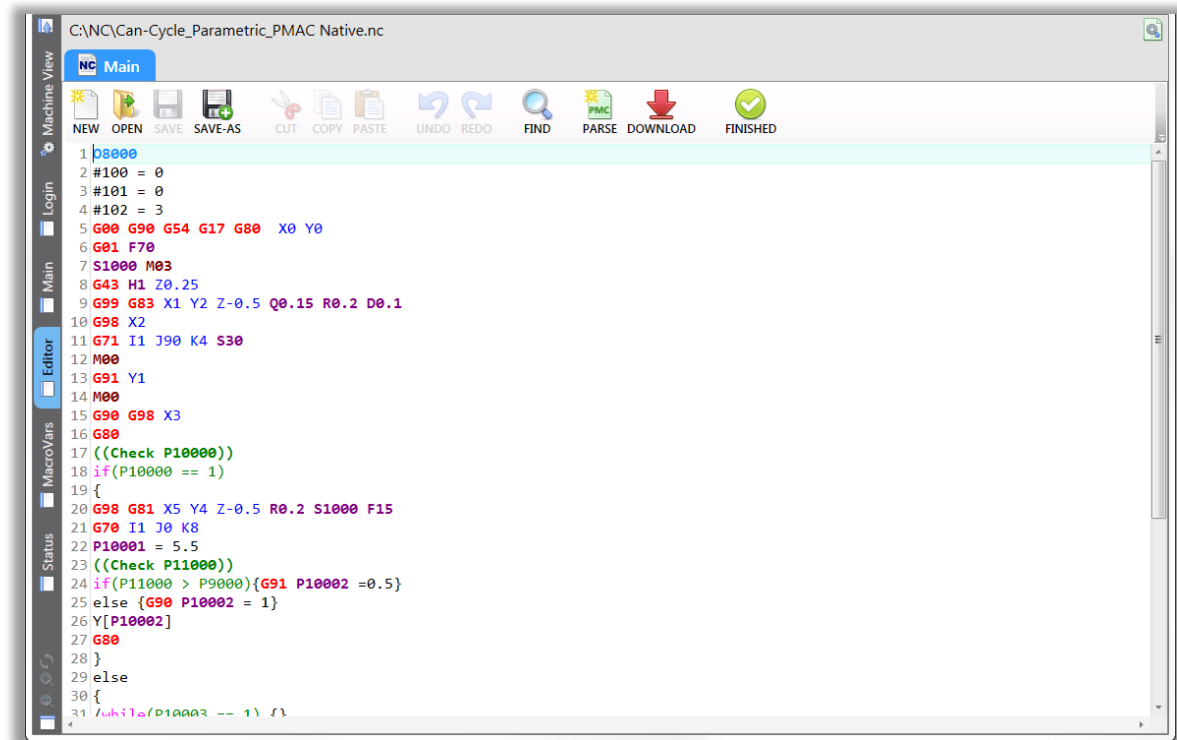


- Save

- Mid-Program Start

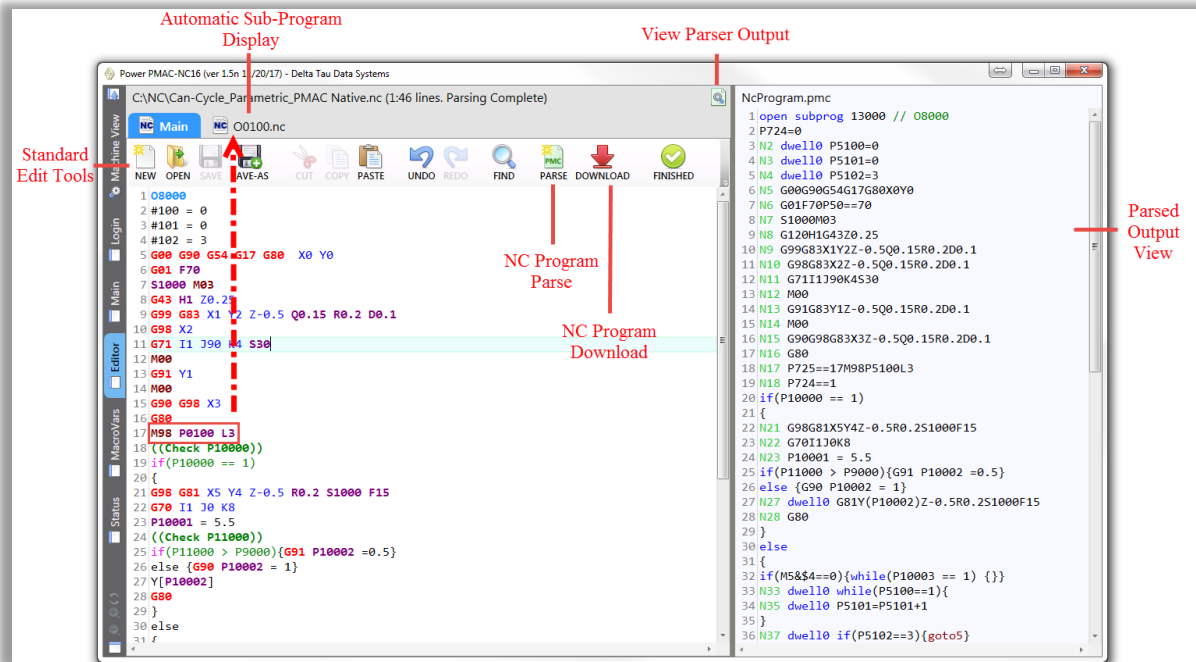


- Editor



Full Screen Program Editor

The PPNC16 includes a powerful full screen editor in addition to the main screen editor. The full screen editor includes many powerful features including automatic sub-program and parser output displays.



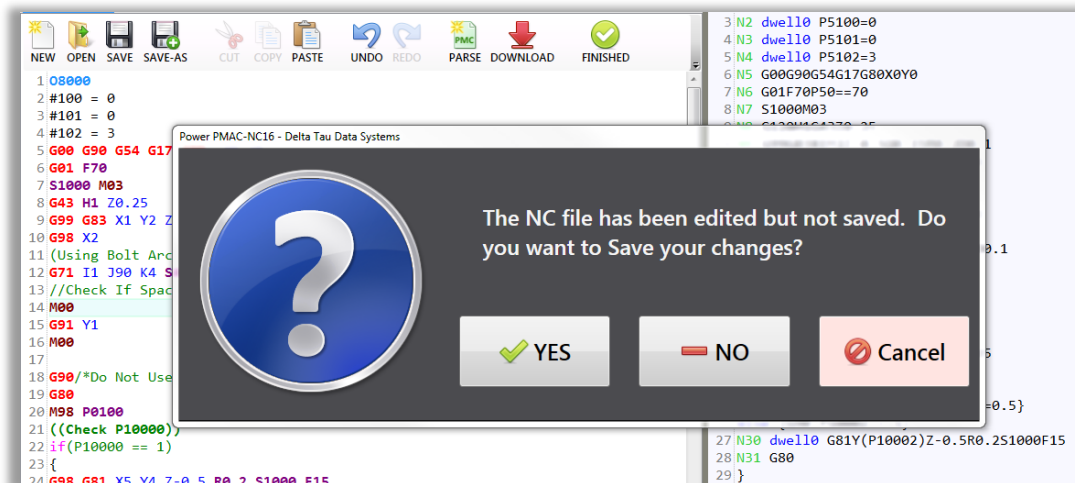
“Parse” button is designed to show the parsed code without downloading being done to the controller (Power PMAC.) Such a powerful tool is designed for complex automation programs debugging. “Download” button, simply load the NC program to the controller, ready to be executed.



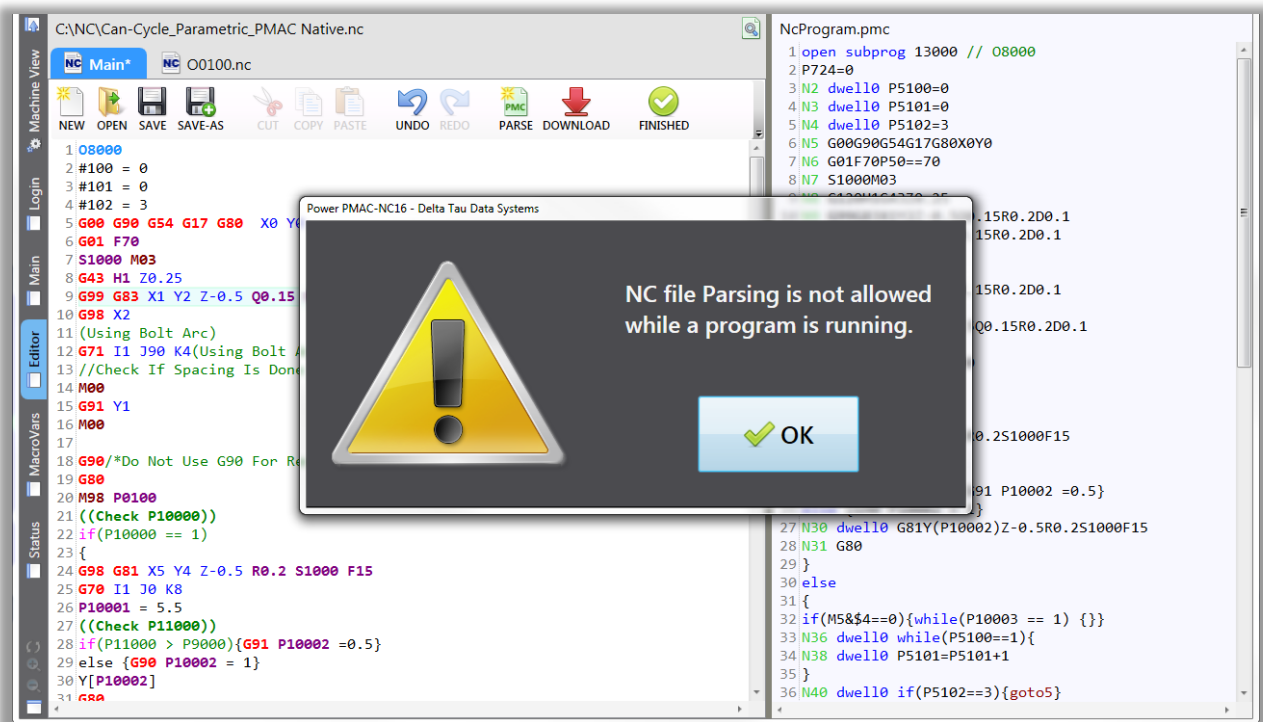
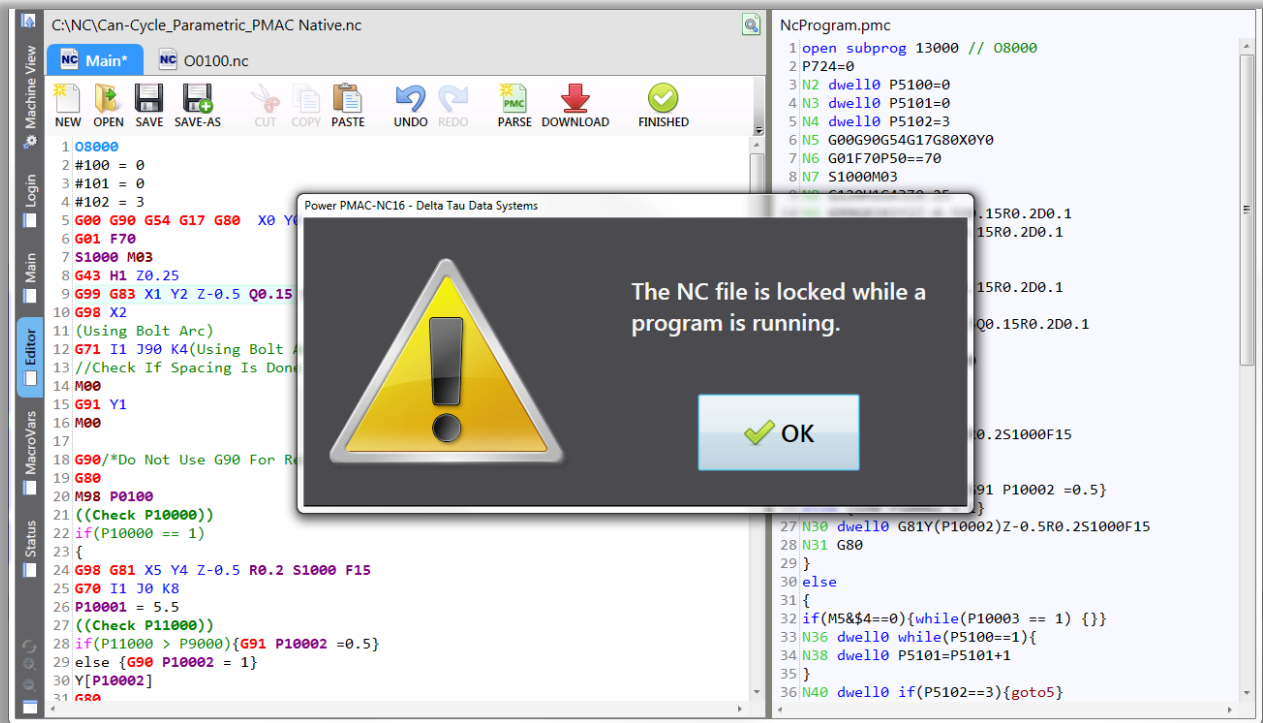
If the NC file is revised, clicking on “Parse” button, automatically issue a “Save”. Make sure to keep track of changes by adding comments or using multiple versions.



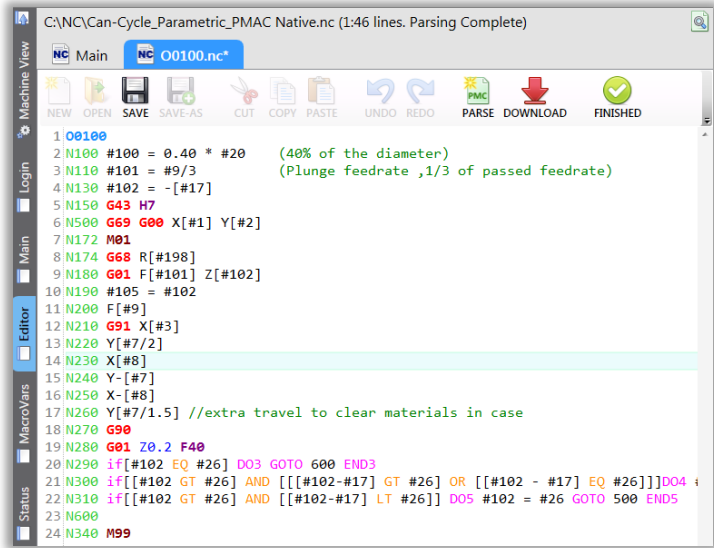
If the NC program is modified but not saved (using “Save” or “Parse” button), as user try to switch tabs, NC16 will notify the user if it is desired to save changes.



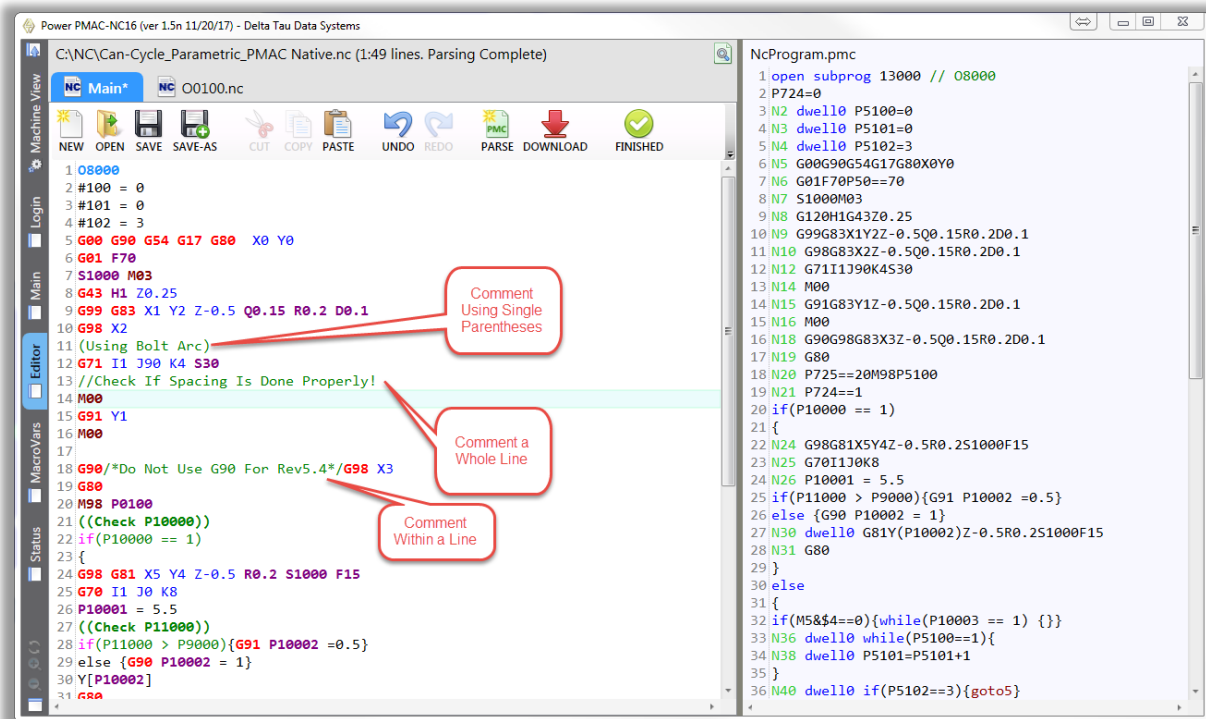
💡 Modification or Parsing a NC file is not allowed when the program is running. PPNC16 shows two different message based on each taken action as follow:



If a subprogram is called within a NC program, upon its existence in a single file or a designated folder (C:\NC by default) will be sorted and shown as an individual tab. When each tab becomes active, content of the called subprogram will be shown in the “Editor” window.

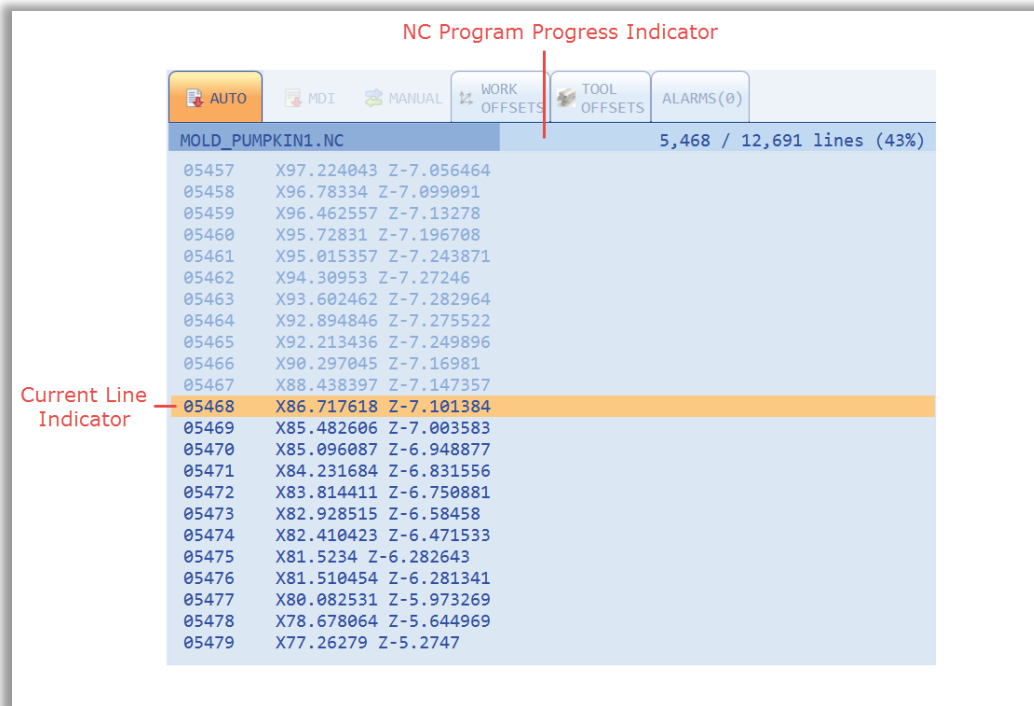


Power PMAC NC16 editor supports three different types of comments in any NC program. These three types are shown below:

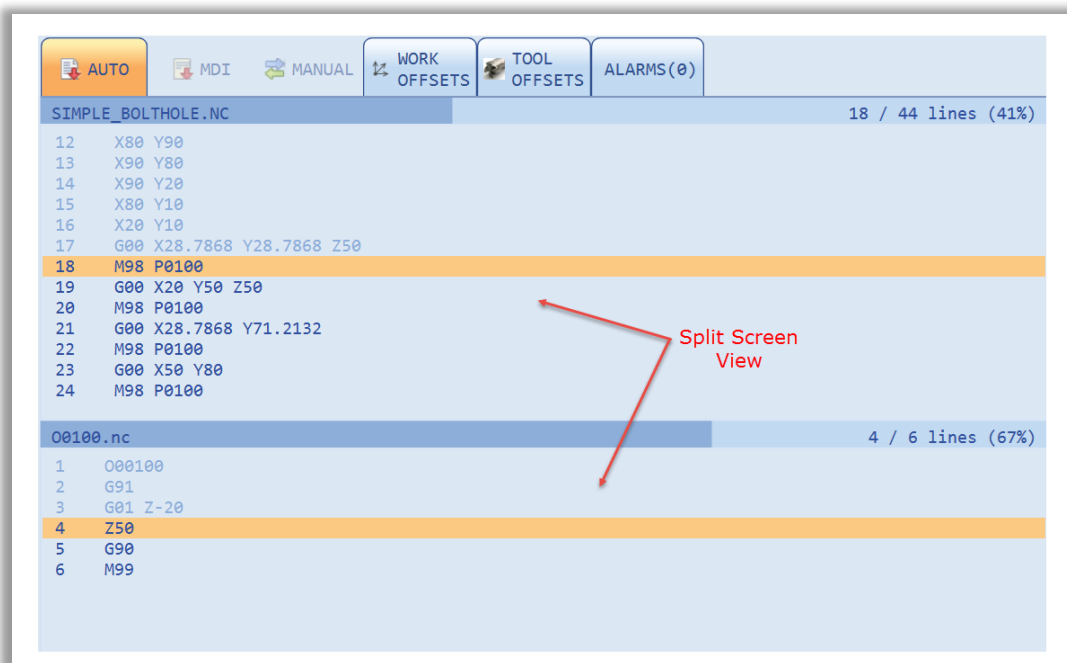


Run Screen

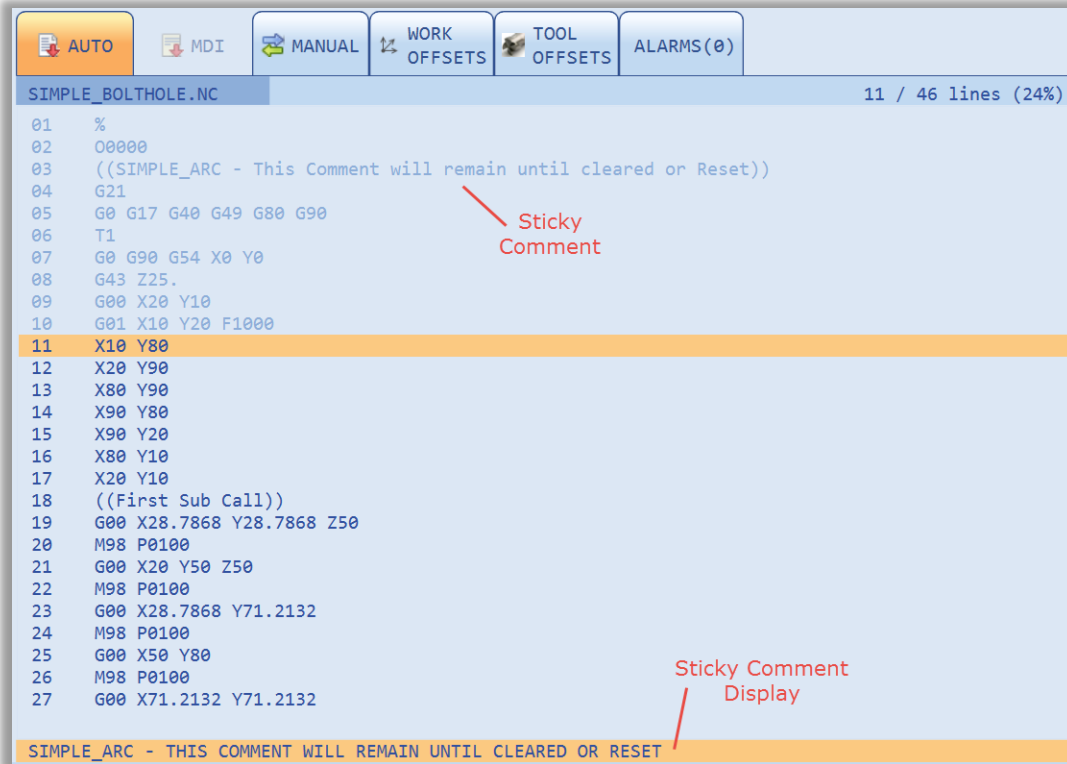
During Run Mode the editor screen will change background colors and the text will show as a lighter color once executed. The current executing line will be highlighted by the horizontal indicator as shown below. The NC program progress indicator will display the part program name and progress both a horizontal bar graph, current line number over total lines, as well as percent of lines executed.



During Sub-Program calls the Run Mode screen will morph into a split screen view simultaneously showing line tracking for both the main program as well as the sub-program.



During Run Mode the NC program monitor supports “Sticky Comments”. Sticky comments are designated by using double parenthesis. The sticky comment will display at the bottom of the Run Mode screen until a subsequent sticky comment is encountered, or the program finishes. This can be a powerful feature for annotating NC files with operator instructions.



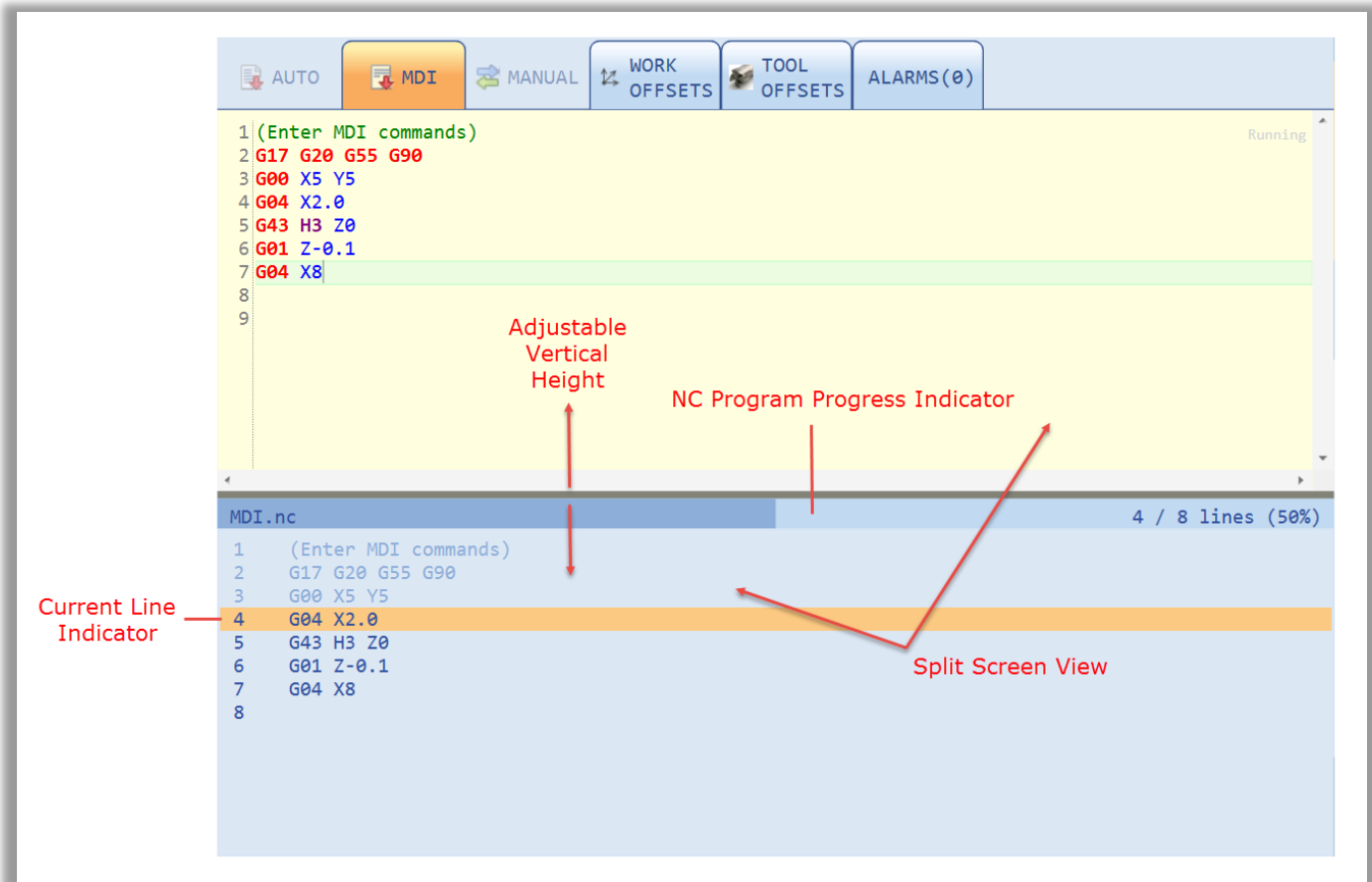
The screenshot shows the NC program monitor interface. At the top, there are tabs for AUTO, MDI, MANUAL, WORK OFFSETS, TOOL OFFSETS, and ALARMS(0). The current program is SIMPLE_BOLTHOLE.NC, and it is at line 11 of 46 lines (24%). The program code is displayed in a list format. Line 03 contains a sticky comment: ((SIMPLE_ARC - This Comment will remain until cleared or Reset)). A red arrow points to this comment with the label "Sticky Comment". Line 11 is highlighted in orange. At the bottom of the screen, a yellow bar displays the sticky comment: SIMPLE_ARC - THIS COMMENT WILL REMAIN UNTIL CLEARED OR RESET. A red arrow points to this bar with the label "Sticky Comment Display".

```
01  %
02  00000
03  ((SIMPLE_ARC - This Comment will remain until cleared or Reset))
04  G21
05  G0 G17 G40 G49 G80 G90
06  T1
07  G0 G90 G54 X0 Y0
08  G43 Z25.
09  G00 X20 Y10
10  G01 X10 Y20 F1000
11  X10 Y80
12  X20 Y90
13  X80 Y90
14  X90 Y80
15  X90 Y20
16  X80 Y10
17  X20 Y10
18  ((First Sub Call))
19  G00 X28.7868 Y28.7868 Z50
20  M98 P0100
21  G00 X20 Y50 Z50
22  M98 P0100
23  G00 X28.7868 Y71.2132
24  M98 P0100
25  G00 X50 Y80
26  M98 P0100
27  G00 X71.2132 Y71.2132
```

SIMPLE_ARC - THIS COMMENT WILL REMAIN UNTIL CLEARED OR RESET

MDI Screen

The MDI Mode screen is a split screen view which includes an MDI editor on top, with an execution monitor on the bottom. The vertical height of these screens is adjustable by dragging the split bar up or down. MDI programs are downloaded to the PMAC when a Cycle Start is executed. The application will automatically sense if the program is modified. If a subsequent Cycle Start is executed the PPNC16 will re-download the MDI program to the control buffer.



Manual Mode Screen


When a hardware control panel is not present, a software Manual screen may be enabled. By default the Manual screen tab will be enabled. The operator will find jog speed select buttons, axis select buttons, continuous jogging buttons, home button, and incremental jog buttons. The incremental jog distance is an operator parameter which can be input. There are two configurations available for the jog speed select buttons, five buttons and three buttons, depending on the integrators preference. Such a task can be achieved by applying following change to the PowerPmacNC.ini file:

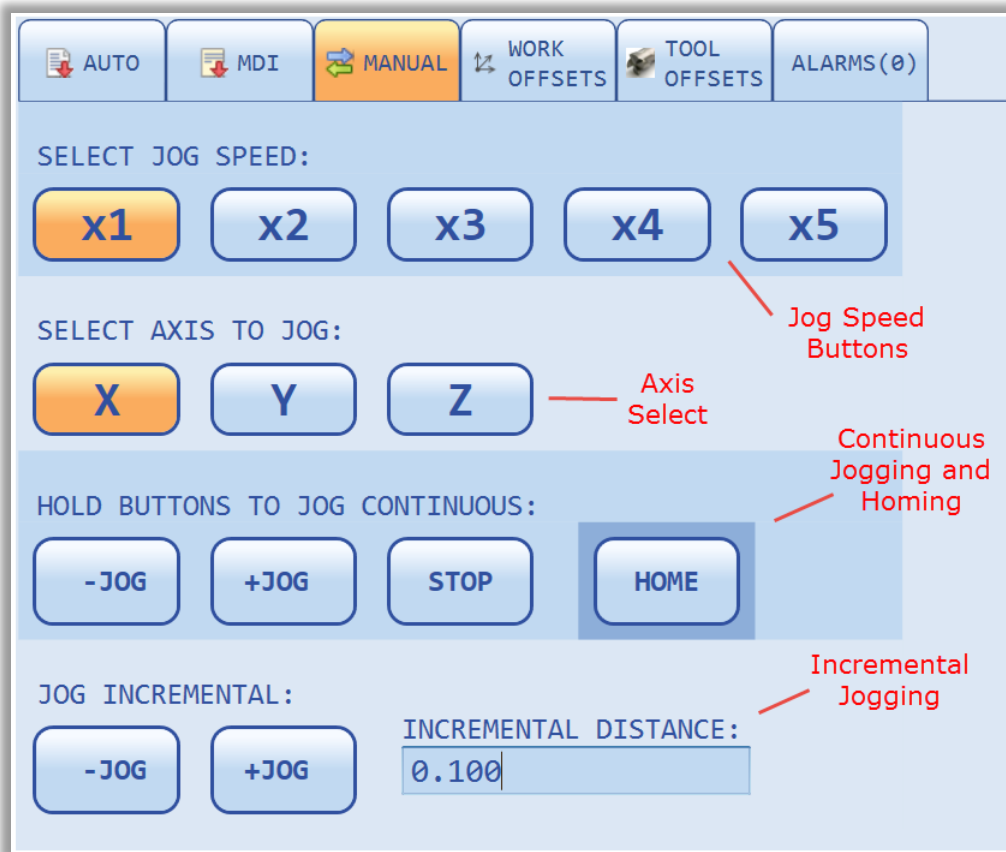
```
; Specify either three or five jog speed buttons to match the pendant.  
ThreeJogSpeeds=True
```

 By default "ThreeJogSpeeds" is set to false.

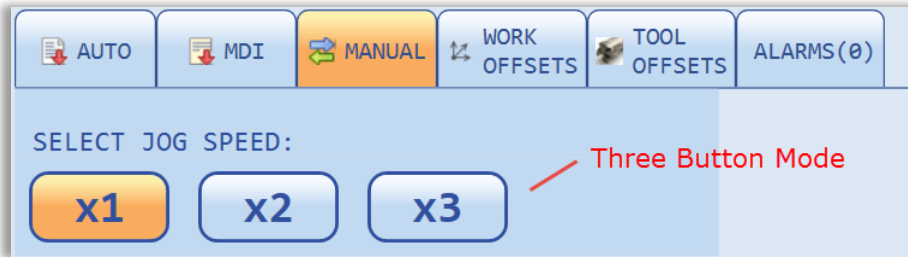
The Manual mode screen may be disabled completely if the integrator wishes to rely on a hardware control panel. This is done by the following code in the Power PMAC project:

```
sendl "HideManual"          // Hides the Manual Screen and Tab  
  
sendl "ShowManual"         // Shows the Manual Screen and Tab
```

 The default PMAC project includes code to automatically show/hide this panel depending on whether a hardware pendant is present (see ppnc_hmimonitor.plc in the Power PMAC project).



Three button mode:



Work Offset Screen

The Work Offset screen displays and allows modification of the coordinate system offset values. The values can be modified manually directly in the input boxes or can be set automatically by using the *Set Work Offsets* buttons at the bottom of the screen. When the buttons are used, the current machine position will be required and used as the offset. The offsets can be modified only while in manual mode. Based on which mode is currently active (Auto, MDI, or Manual) cells will be highlighted in White or None-White color based on a chosen skin. White color cells means, cells are “read only” and a machine is either in the Auto or MDI mode. None-White color cells means, cells can be modified and a machine is in the Manual mode. Below figure, is an example of a Work Offset table in a machine manual mode:

The image shows the Work Offset screen interface. At the top is a row of buttons: AUTO, QUEUE (0), MDI, WORK OFFSETS, TOOL OFFSETS, and ALARMS (0). Below this is a table with columns for Offset, X, Y, and Z. The table contains 19 rows of data. A red callout bubble points to the cell for G54.1 P1 in the X column, containing the text "Modifiable Cell Machine AUTO/MDI Mode". At the bottom of the screen, there is a label "SET WORK OFFSETS: (Manual Mode Only)" and four buttons: ALL, X, Y, and Z.

Offset	X	Y	Z
G54	1.5040	2.0100	5.6230
G55	3.0230	5.0000	4.2516
G56	5.6323	7.2560	0
G57	0	0	0
G58	0	0	0
G59	0	0	0
G54.1 P1	0	0	0
G54.1 P2	0	0	0
G54.1 P3	0	0	0
G54.1 P4	0	0	0
G54.1 P5	0	0	0
G54.1 P6	0	0	0
G54.1 P7	0	0	0
G54.1 P8	0	0	0
G54.1 P9	0	0	0
G54.1 P10	0	0	0
G54.1 P11	0	0	0

SET WORK OFFSETS: (Manual Mode Only)

ALL X Y Z

Offset	X	Y	Z
G54	1.5040	2.0100	5.6230
G55	3.0230	5.0000	4.2516
G56	5.6323	7.2560	0
G57	0	0	0
G58	0	0	0
G59	0	0	0
G54.1 P1	0	0	0
G54.1 P2	0	0	0
G54.1 P3	0	0	0
G54.1 P4	0	0	0
G54.1 P5	0	0	0
G54.1 P6	0	0	0
G54.1 P7	0	0	0
G54.1 P8	0	0	0
G54.1 P9	0	0	0
G54.1 P10	0	0	0
G54.1 P11	0	0	0

SET WORK OFFSETS: (Manual Mode Only)

ALL X Y Z

Read-Only Cell
Machine AUTO/MDI Mode

Each cell is capable of performing four basic mathematical operations (one at a time). If any of these operations is applied in a chosen cell, a new window under the active cell shows a calculated value based on a chosen operation as follow:

Offset	X	Y	Z
G54	1.5040	2.0100	5.6230
G55	3.0230	5.0000	4.2516
G56	5.6323	7.2560	0
G57	0	- 0.25	0
G58	0	= 7.006	0
G59	0	0	0
G54.1 P1	0	0	0
G54.1 P2	0	0	0
G54.1 P3	0	0	0
G54.1 P4	0	0	0
G54.1 P5	0	0	0
G54.1 P6	0	0	0
G54.1 P7	0	0	0
G54.1 P8	0	0	0
G54.1 P9	0	0	0
G54.1 P10	0	0	0
G54.1 P11	0	0	0

SET WORK OFFSETS: (Manual Mode Only)

ALL X Y Z

If it is desired to use a calculated value, press “Enter” and then a confirmation box appears to make sure that it is desired to change a cell’s value permanently; a “Yes” button simply replaces the old value with a new one as it is shown in the below figure:



The PPNC16 software always highlight the current active work offset in a table. If no work offset is active or it is desired to change another work offset beside the one is active, highlight it by clicking on it and then use automatic work offset buttons to set any or all positions.

PROG POS REL POS MACH POS CMD POS

X -1.5040
Y -2.0100
Z -5.6230

% TORQUE FE: -0.0010
% TORQUE FE: 0.0000
JOG SPEED: x5

UNITS INCH (G20)
FEEDRATE ACT: 0.00 CMD: 0.00
FEED: 100% RAPID: 100% FPM
SPINDLE 0.00 0.00 CUT
TOOL T00 H00 D00
G-CODES G00 G17 G20 G25 G40 G49 G80 G50
G50.1 G54 G97 G64 G69 G90 G94 G98
M-CODES M30 M05 M08 M10 M23 M41 M48 M78

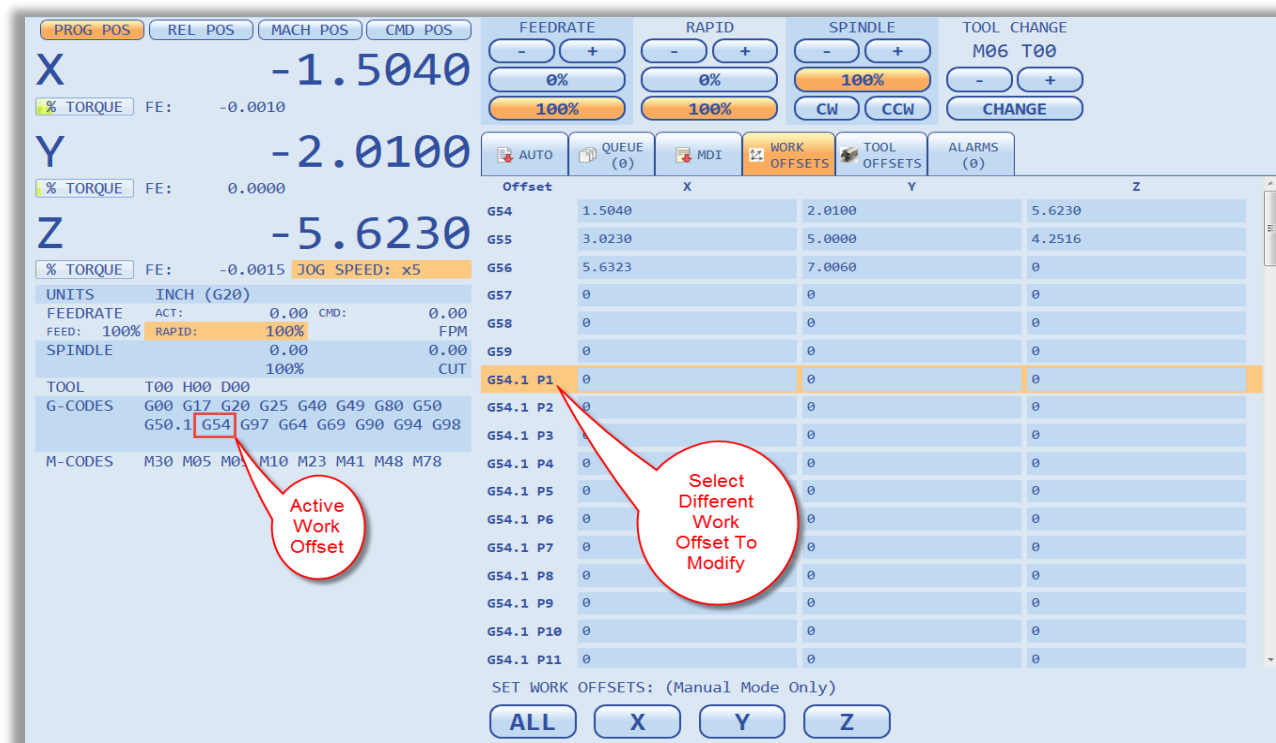
Offset	X	Y	Z
G54	1.5040	2.0100	5.6230
G55	0.0000	5.0000	4.2516
G56	0.0000	7.0060	0
G57	0	0	0
G58	0	0	0
G59	0	0	0
G54.1 P1	0	0	0
G54.1 P2	0	0	0
G54.1 P3	0	0	0
G54.1 P4	0	0	0
G54.1 P5	0	0	0
G54.1 P6	0	0	0
G54.1 P7	0	0	0
G54.1 P8	0	0	0
G54.1 P9	0	0	0
G54.1 P10	0	0	0
G54.1 P11	0	0	0

Highlighted Work Offset

Active Work Offset

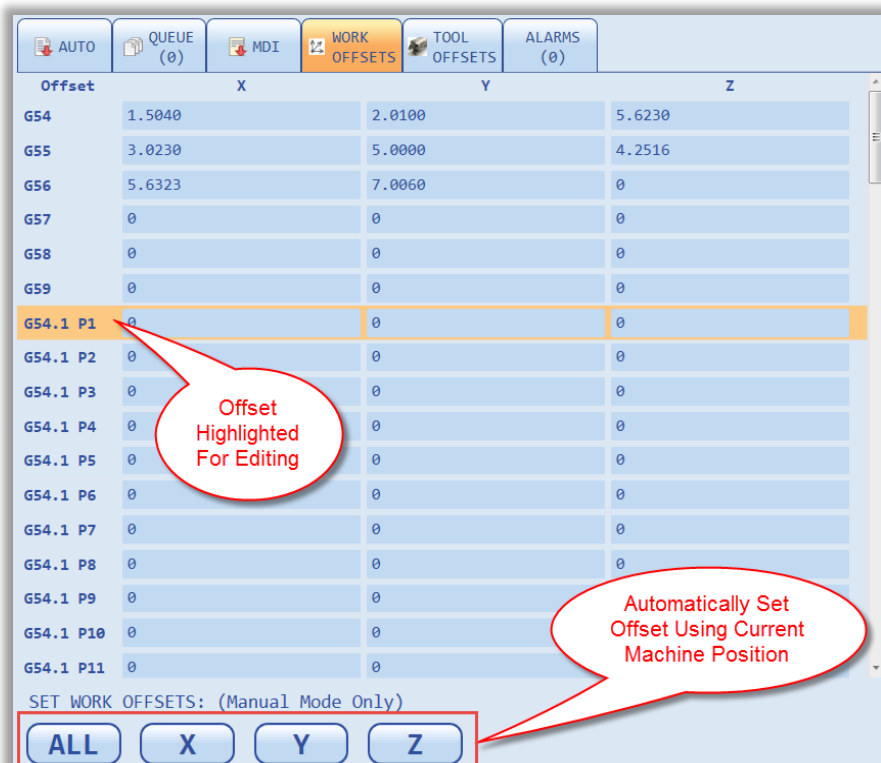
SET WORK OFFSETS: (Manual Mode Only)


ALL X Y Z





The PPNC16 software supports 100 auxiliary work offsets (G54.1 P1- G54.1 P100). The number of auxiliary G54.1 Px offsets is configurable in the PowerPmacNC.ini file.

Automatic work offset buttons in a manual mode, allow users to choose a current machine position to set a chosen offset with respect to each axis. Or if it is desired, “All” button will set all axes positions at once for a chosen work offset.



 The PPNC16 software includes a mechanism so the programmatic setting of work offsets can be achieved directly from the controller regardless of machine mode. This allows for the simple integration of touch probes or other automated systems for this purpose (see ppnc_worktooloffset.plc in the Power PMAC project).

 Confirmation box shows up each time a work offset values are modified regardless of being done manually or using automatic set work offsets buttons.

 The actual data from the Work Offset table is saved in the PowerPMACSettings.xml file. In some special cases this file can be manually configured for special tooling or setups. This xml file cannot be edited while the application is running. Care should be taken whenever modifying settings files and a backup should be made prior to doing any modification.

Tool Offset Screen

The Tool Offset screen displays and allows modification of the Tool offset parameters (Length, Length Wear, Diameter, and Diameter Wear). The values can be modified manually directly in the input boxes or can be set automatically by using the *Set Tool Length* button at the bottom of the screen (length only). When the buttons are used, the current machine position will be required and used as the offset. The offsets can be modified only while in manual mode. Based on which mode is currently active (Auto, MDI, or Manual) cells will be highlighted in White or None-White color based on a chosen skin. White color cells means, cells are “read only” and a machine is either in the Auto or MDI mode. None-White color cells means, cells can be modified and a machine is in the Manual mode. Below figure, is an example of a Tool Offset table in a machine manual mode:

AUTO		QUEUE (0)		MDI		WORK OFFSETS		TOOL OFFSETS		ALARMS (0)	
Tool Index	Tool Length	Tool Wear	Tool Diameter	Diameter Wear							
Tool 1	-8.2171	-0.0005	0.5000	-0.0010	1/2" END MILL 2 FLUTE						
Tool 2	-9.7016	-0.0002	0.5000	0	1/2" END MILL 3 FLUTE						
Tool 3	-10.9171	0	0.2500	-0.0001	1/4" END MILL 3 FLUTE						
Tool 4	-11.4810	0	0	0	DRILL CHUCK						
Tool 5	-12.2395	-0.0002	0.1250	0	1/8" END MILL 3 FLUTE						
Tool 6	-13.5632	0	0.1250	-0.0006	1/8" CARBIDE END MILL 3 FLUTE						
Tool 7	-15.2360	0	0.1250								
Tool 8	0	0	0	0							
Tool 9	0	0									
Tool 10	0	0									
Tool 11	0	0									
Tool 12	0	0	0	0							
Tool 13	0	0	0	0							
Tool 14	0	0	0	0							
Tool 15	0	0	0	0							
Tool 16	0	0	0	0							
Tool 17	0	0	0	0							

SET TOOL OFFSETS: (Manual Mode Only)

SET TOOL LENGTH

Read-Only Cell
Machine AUTO/MDI
Mode

AUTO		QUEUE (0)		MDI		WORK OFFSETS		TOOL OFFSETS		ALARMS (0)	
Tool Index	Tool Length	Tool Wear	Tool Diameter	Diameter Wear							
Tool 1	-8.2171	-0.0005	0.5000	-0.0010	1/2" END MILL 2 FLUTE						
Tool 2	-9.7016	-0.0002	0.5000	0	1/2" END MILL 3 FLUTE						
Tool 3	-10.9171	0	0.2500	-0.0001	1/4" END MILL 3 FLUTE						
Tool 4	-11.4810	0	0	0	DRILL CHUCK						
Tool 5	-12.2395	-0.0002	0.1250	0	1/8" END MILL 3 FLUTE						
Tool 6	-13.5632	0	0.1250	-0.0006	1/8" CARBIDE END MILL 3 FLUTE						
Tool 7	-15.2360	0	0.1250	0							
Tool 8	0	0	0	0							
Tool 9	0	0	0	0							
Tool 10	0	0	0	0							
Tool 11	0	0	0	0							
Tool 12	0	0	0	0							
Tool 13	0	0	0	0							
Tool 14	0	0	0	0							
Tool 15	0	0	0	0							
Tool 16	0	0	0	0							
Tool 17	0	0	0	0							

SET TOOL OFFSETS: (Manual Mode Only)

SET TOOL LENGTH

Modifiable Cell
Machine Manual Mode

Each cell is capable of performing four basic mathematical operations (one at a time). If any of these operations is applied in a chosen cell, a new window under the active cell shows a calculated value based on a chosen operation as follow:

AUTO

QUEUE (0)

MDI

WORK OFFSETS

TOOL OFFSETS

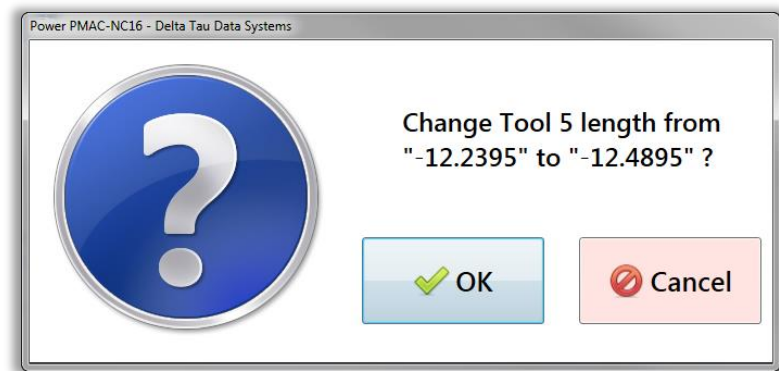
ALARMS (0)


Tool Index	Tool Length	Tool Wear	Tool Diameter	Diameter Wear	
Tool 1	-8.2171	-0.0005	0.5000	-0.0010	1/2" END MILL 2 FLUTE
Tool 2	-9.7016	-0.0002	0.5000	0	1/2" END MILL 3 FLUTE
Tool 3	-10.9171	0	0.2500	-0.0001	1/4" END MILL 3 FLUTE
Tool 4	-11.4810	0	0	0	DRILL CHUCK
Tool 5	-12.2395	-0.0002	0.1250	0	1/8" END MILL 3 FLUTE
Tool 6	- 0.25		0.1250	-0.0006	1/8" CARBIDE END MILL 3 FLUTE
Tool 7	= -12.4895		0.1250	0	
Tool 8	0	0	0	0	
Tool 9	0	0	0	0	
Tool 10	0	0	0	0	
Tool 11	0	0	0	0	
Tool 12	0	0	0	0	
Tool 13	0	0	0	0	
Tool 14	0	0	0	0	
Tool 15	0	0	0	0	
Tool 16	0	0	0	0	
Tool 17	0	0	0	0	

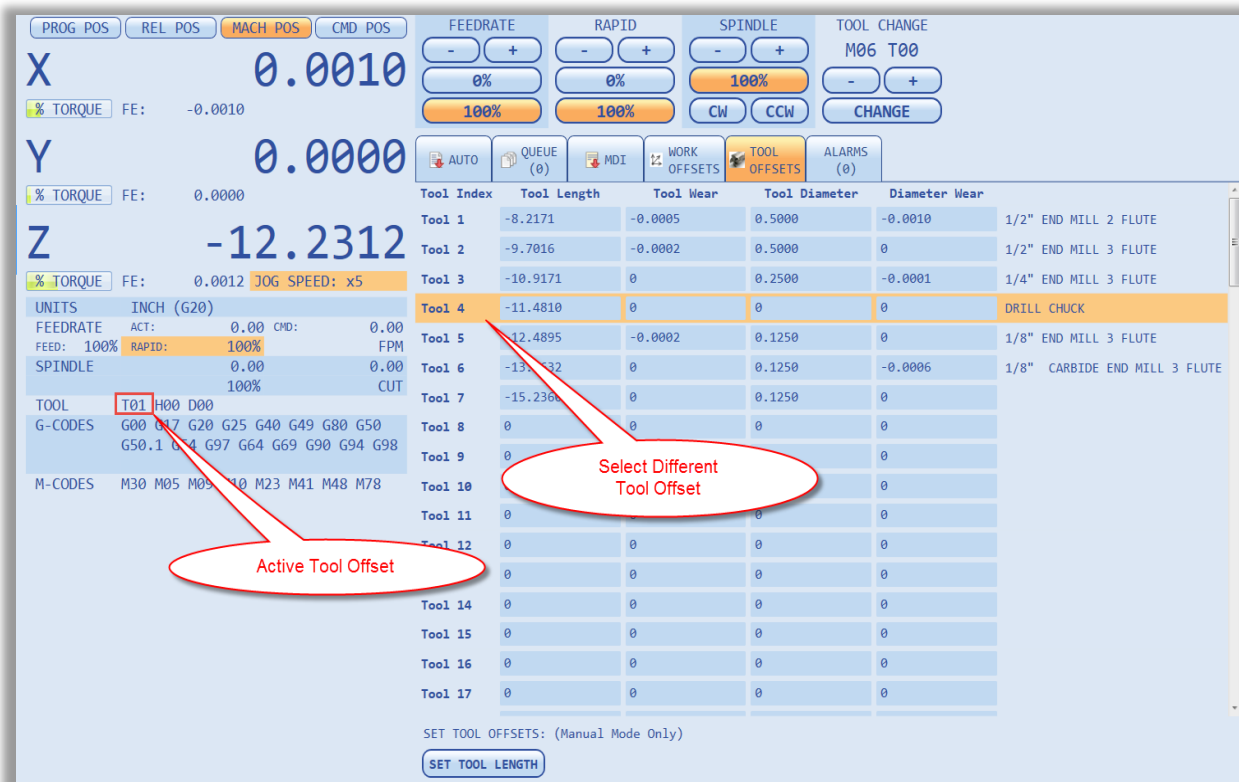
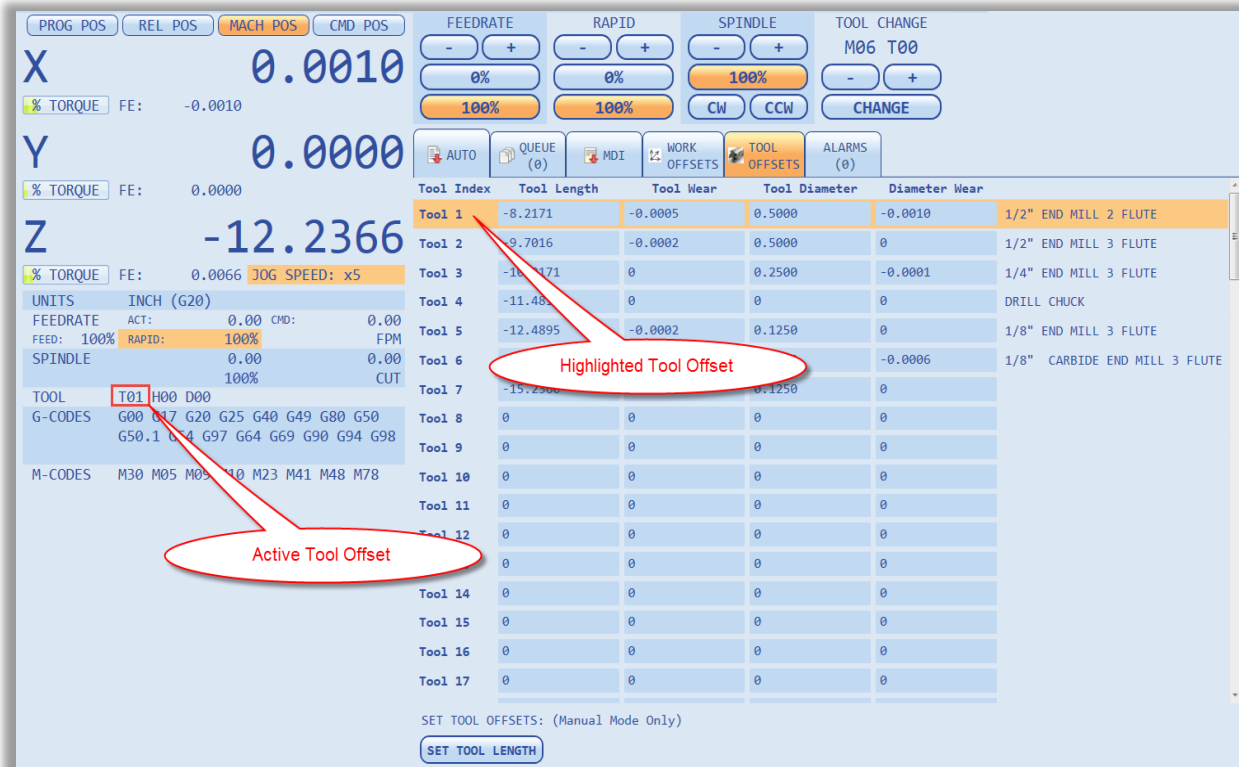
SET TOOL OFFSETS: (Manual Mode Only)

SET TOOL LENGTH

If it is desired to use a calculated value, press “Enter” and then a confirmation box appears to make sure that it is desired to change a cell’s value permanently; a “Yes” button simply replaces the old value with a new one as it is shown in the figure below:



 The PPNC16 software always highlight the current active tool offset in a table. If no tool offset is active or it is desired to change another tool offset beside the one is active, highlight it by clicking on it and then use automatic work offset button to set it.



The PPNC16 software supports 100 tool offsets .The number of tool offsets is configurable in the PowerPmacNC.ini file.

Automatic tool offset button in a manual mode, allow users to choose a current machine position to set a chosen tool offset. When the button is pressed the current machine position will be queried and used as the tool offset. The offsets can be modified only while in manual mode.

AUTO

QUEUE (0)

MDI

WORK OFFSETS

TOOL OFFSETS


ALARMS (0)


Tool Index	Tool Length	Tool Wear	Tool Diameter	Diameter Wear	
Tool 1	-8.2171	-0.0005	0.5000	-0.0010	1/2" END MILL 2 FLUTE
Tool 2	-9.7016	-0.0002	0.5000	0	1/2" END MILL 3 FLUTE
Tool 3	-10.9171	0	0.2500	-0.0001	1/4" END MILL 3 FLUTE
Tool 4	-11.4810	0	0	0	DRILL CHUCK
Tool 5	-12.4895	-0.0002	0.1250	0	1/8" END MILL 3 FLUTE
Tool 6	-13.5632	0	0.1250	-0.0006	1/8" CARBIDE END MILL 3 FLUTE
Tool 7	-15.2360	0	0.1250	0	
Tool 8	-12.2405	0	0	0	
Tool 9	0	0	0	0	
Tool 10	0	0	0	0	
Tool 11	0	0	0	0	
Tool 12	0	0	0	0	
Tool 13	0	0	0	0	
Tool 14	0	0	0	0	
Tool 15	0	0	0	0	
Tool 16	0	0	0	0	
Tool 17	0	0	0	0	


SET TOOL OFFSETS: (Manual Mode Only)

SET TOOL LENGTH

Automatic Tool Length Set


 The PPNC16 software includes a mechanism so the programmatic setting of tool offsets can be achieved directly from the controller. This allows for the simple integration of touch probes or other automated systems for this purpose (see ppnc_worktooloffset.plc in the Power PMAC project).

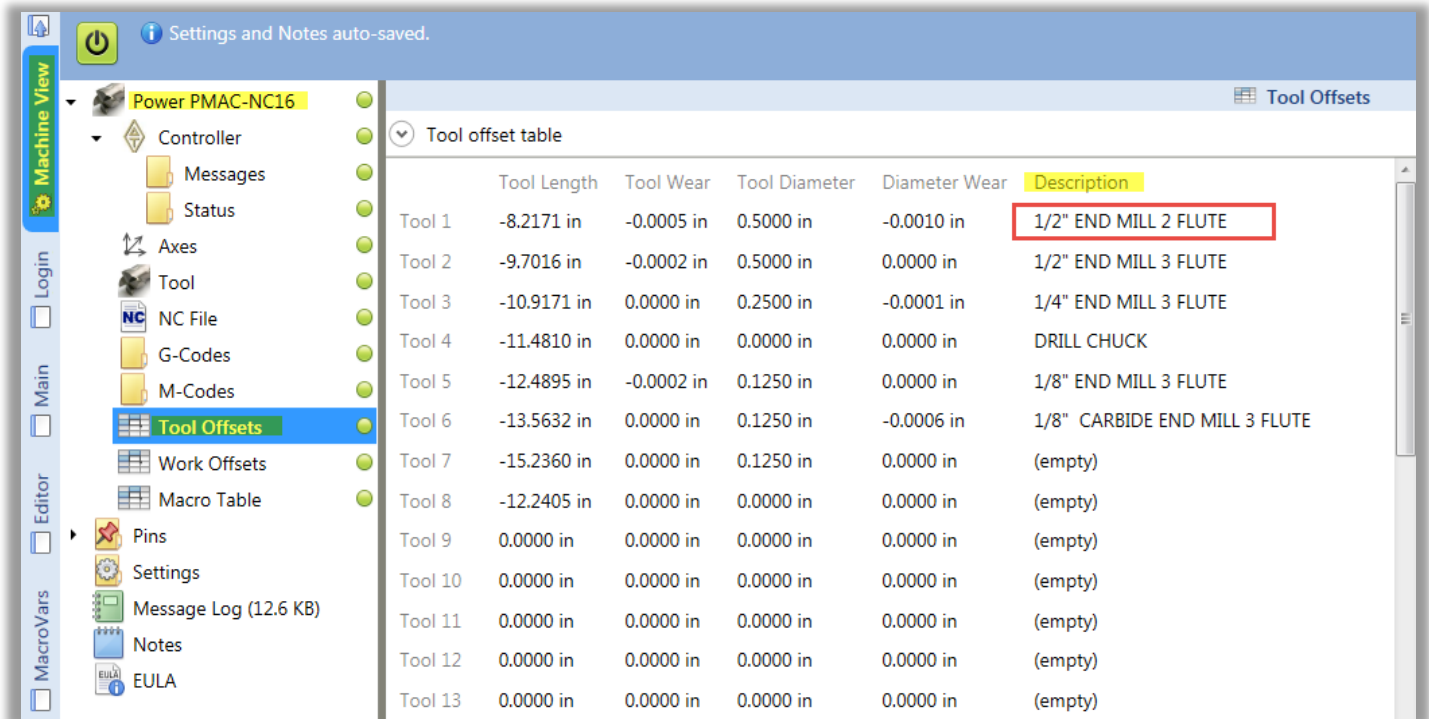

 Confirmation box shows up an each time a tool offset value is modified regardless of being done manually or using “SET TOOL LENGTH” button.


 The actual data from the Tool Offset table is saved in the PowerPMACSettings.xml file. In some special cases this file can be manually configured for special tooling or setups. This xml file cannot be edited while the application is running. Care should be taken whenever modifying settings files and a backup should be made prior to doing any modification.

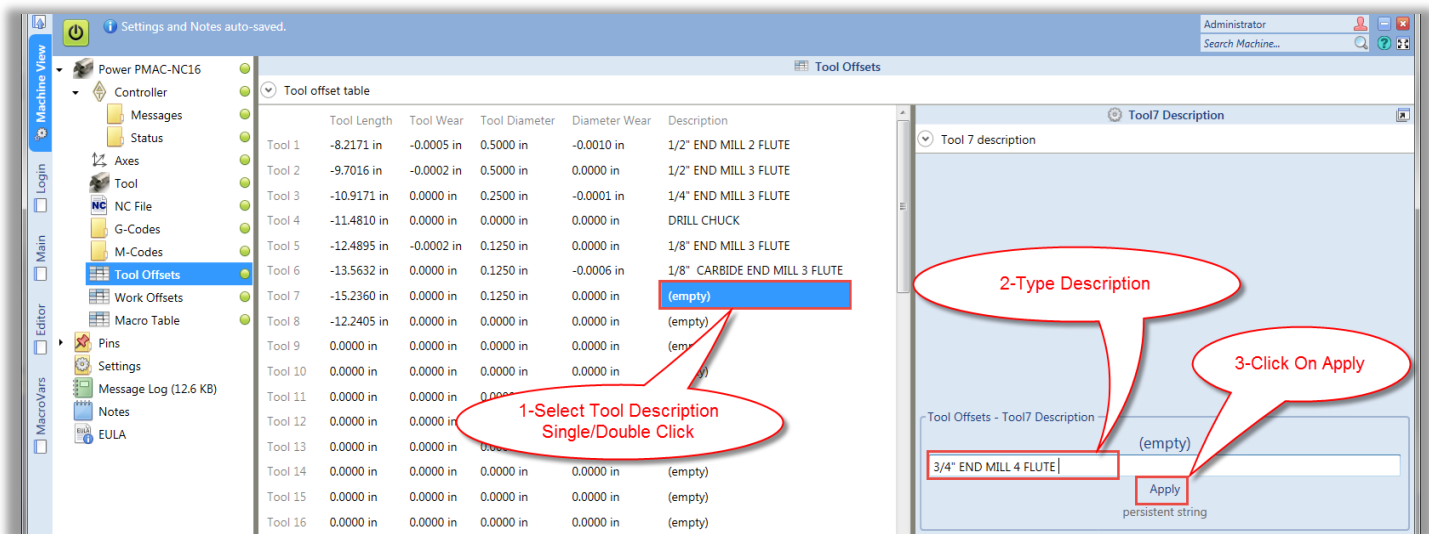
Tool offset screen also provides unique ability of assigning description to each tool in order to assist users with tool identification. This feature can be configured in the PowerPmacNC.ini file.

In order to assign description to each tool, choose the “Tool Offsets” by using a following address:

Machine View -> Power PMAC-NC16 -> Tool Offsets



Click on any description that is required to be assigned or modified and apply a description as follow:



<div> <div>AUTO</div> <div>QUEUE (0)</div> <div>MDI</div> <div>WORK OFFSETS</div> <div>TOOL OFFSETS</div> <div>ALARMS (0)</div> </div>					
Tool Index	Tool Length	Tool Wear	Tool Diameter	Diameter Wear	
Tool 1	-8.2171	-0.0005	0.5000	-0.0010	1/2" END MILL 2 FLUTE
Tool 2	-9.7016	-0.0002	0.5000	0	1/2" END MILL 3 FLUTE
Tool 3	-10.9171	0	0.2500	-0.0001	1/4" END MILL 3 FLUTE
Tool 4	-11.4810	0	0	0	DRILL CHUCK
Tool 5	-12.4895	-0.0002	0.1250	0	1/8" END MILL 3 FLUTE
Tool 6	-13.5632	0	0.1250	-0.0006	1/8" CARBIDE END MILL 3 FLUTE
Tool 7	-15.2360	0	0.1250	0	3/4" END MILL 4 FLUTE
Tool 8	-12.2405	0	0	0	
Tool 9	0	0	0	0	
Tool 10	0	0	0	0	
Tool 11	0	0	0	0	
Tool 12	0	0	0	0	
Tool 13	0	0	0	0	
Tool 14	0	0	0	0	
Tool 15	0	0	0	0	
Tool 16	0	0	0	0	
Tool 17	0	0	0	0	

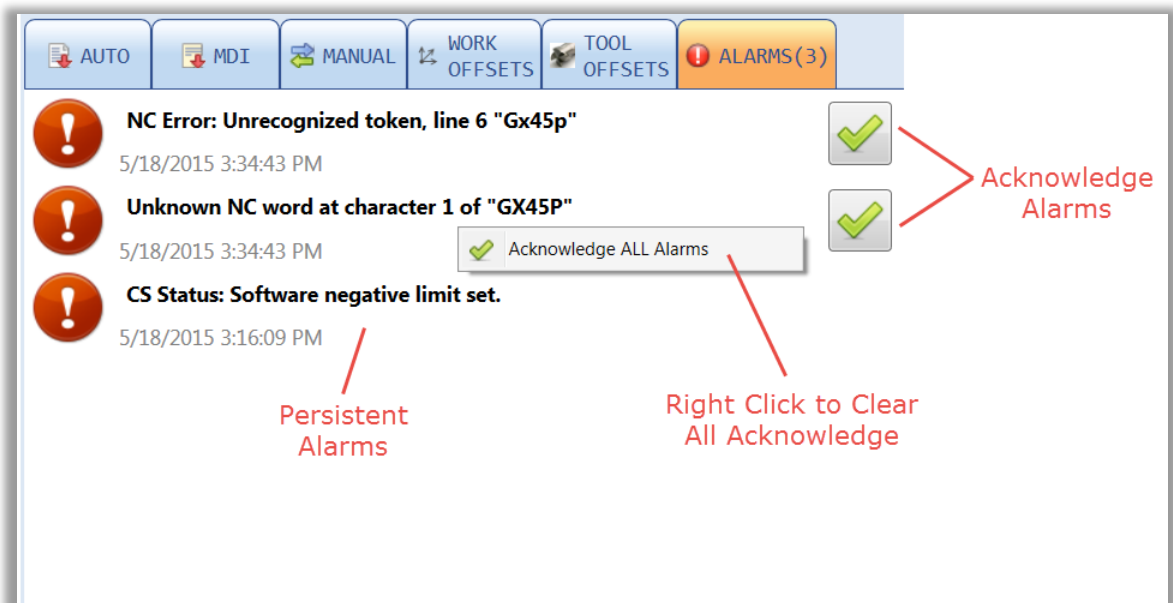
SET TOOL OFFSETS: (Manual Mode Only)

SET TOOL LENGTH

Assigned Tool Offset Description

Alarms Screen and Dialog Message Boxes

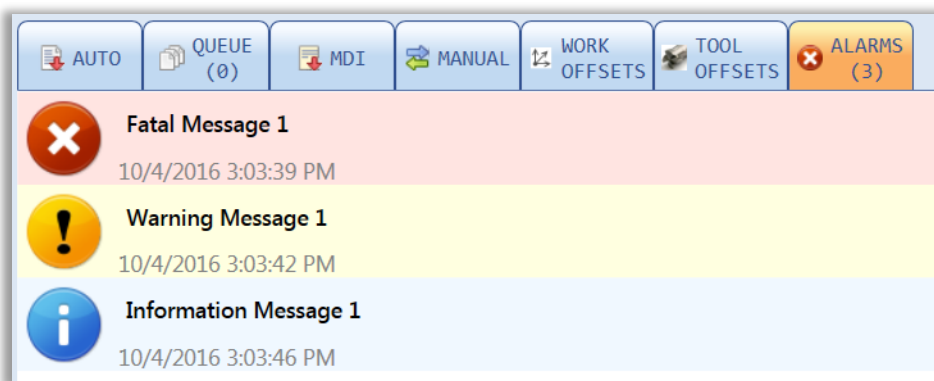
The Alarm screen will display any active alarms, warning, messages, etc. Any alarm or message which is displayed will also be sent to the message log with a time and date stamp. In general, there are two types of messages, persistent, and acknowledge. The persistent messages will remain active until the underlying fault is cleared. Acknowledge messages will appear with a check box to the right of the message. These messages can be cleared by clicking on the check to box to acknowledge the message. If there are multiple acknowledge messages present they can all be cleared by right clicking in the Alarms screen and selecting the "Acknowledge ALL Alarms" option.



There are four types of custom messages which the machine builder may choose to utilize. Each type includes sub-types with different priority visualizations. In all there are 14 different ways to display messages in Power PMAC NC 16.

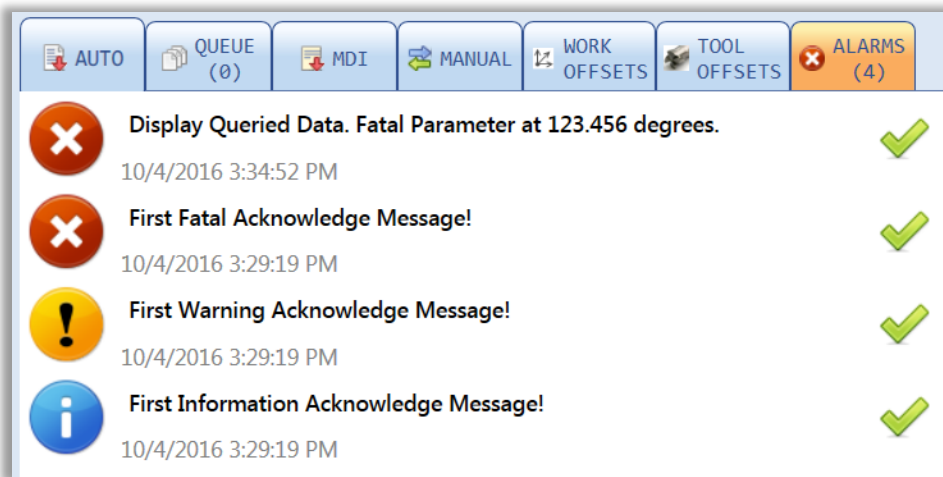
The four main message types include the following:

- Type 1 – Persistent, Bitwise Message Display
 - Fatal Message
 - Warning Message
 - Information Message

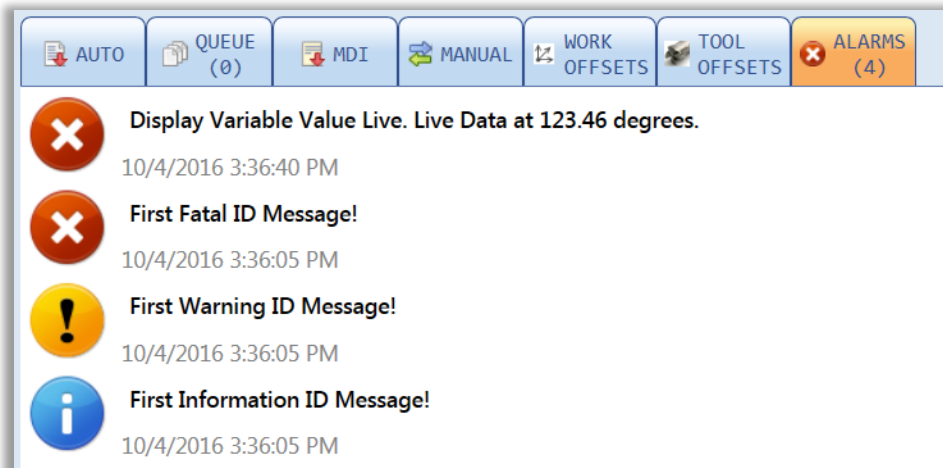


31 configurable messages are available for each type in the Messages.xml file. Each message can be modified to a desired message and be saved in this file.

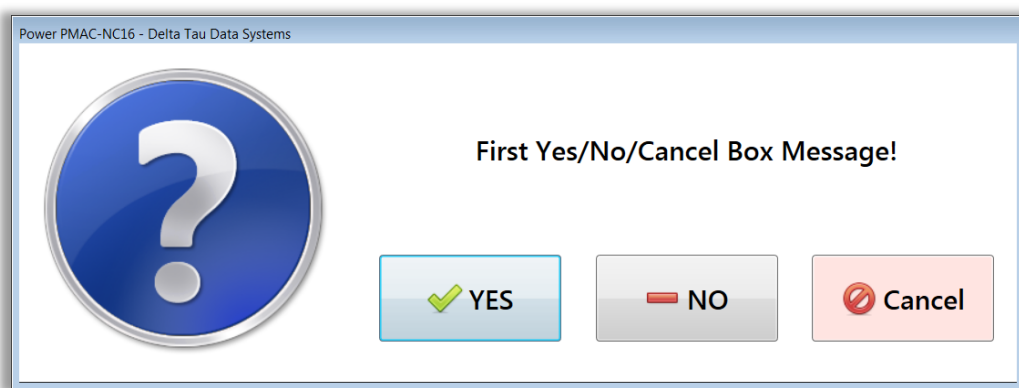
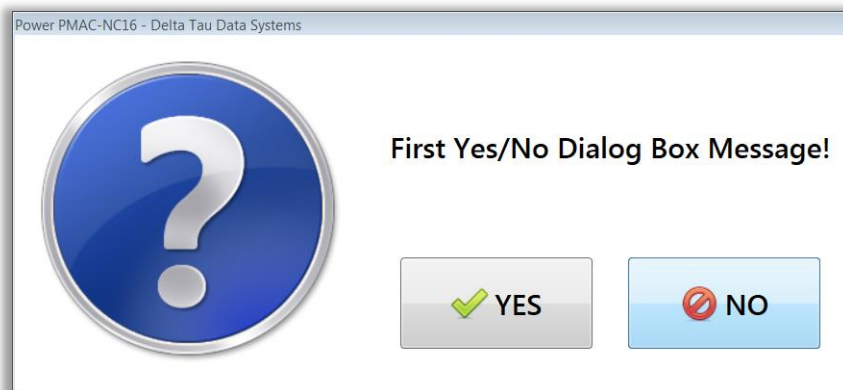
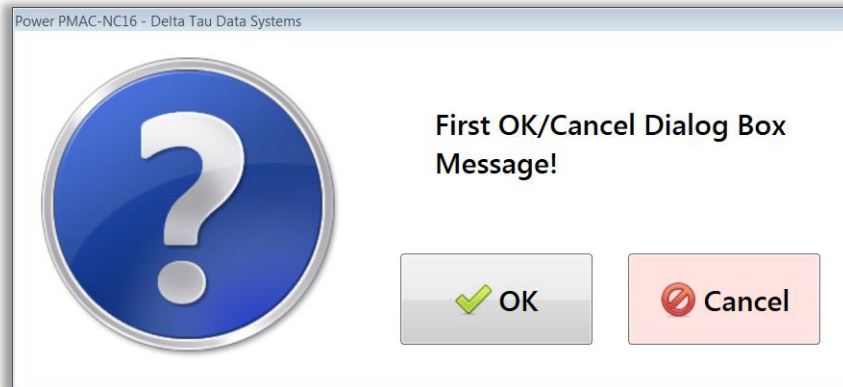
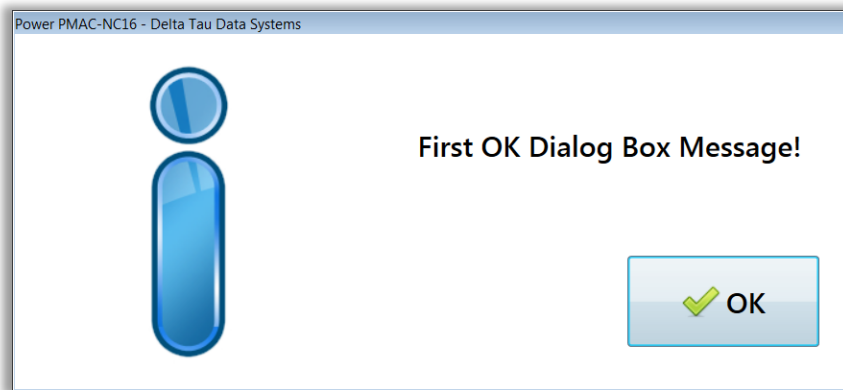
- Type 2 – Unsolicited “Send1” Acknowledge Message (Can display queried data within message)
 - Fatal Message
 - Warning Message
 - Information Message
 - Log Only Message (Will not display, sent to message log only)

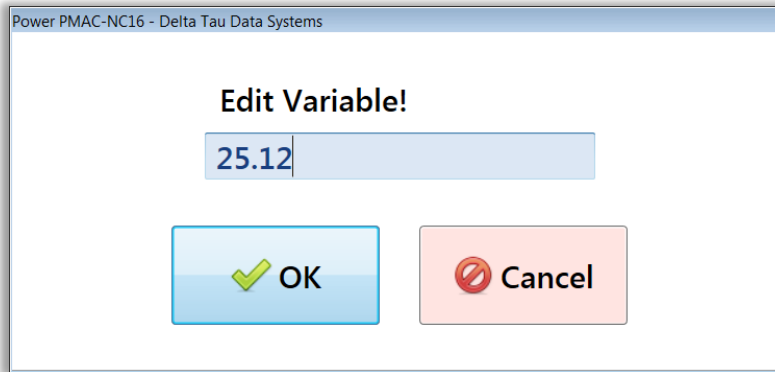


- Type 3 – Unsolicited “Send1” Persistent Message with ID (Can display queried plus live data within message)
 - Fatal Message
 - Warning Message
 - Information Message



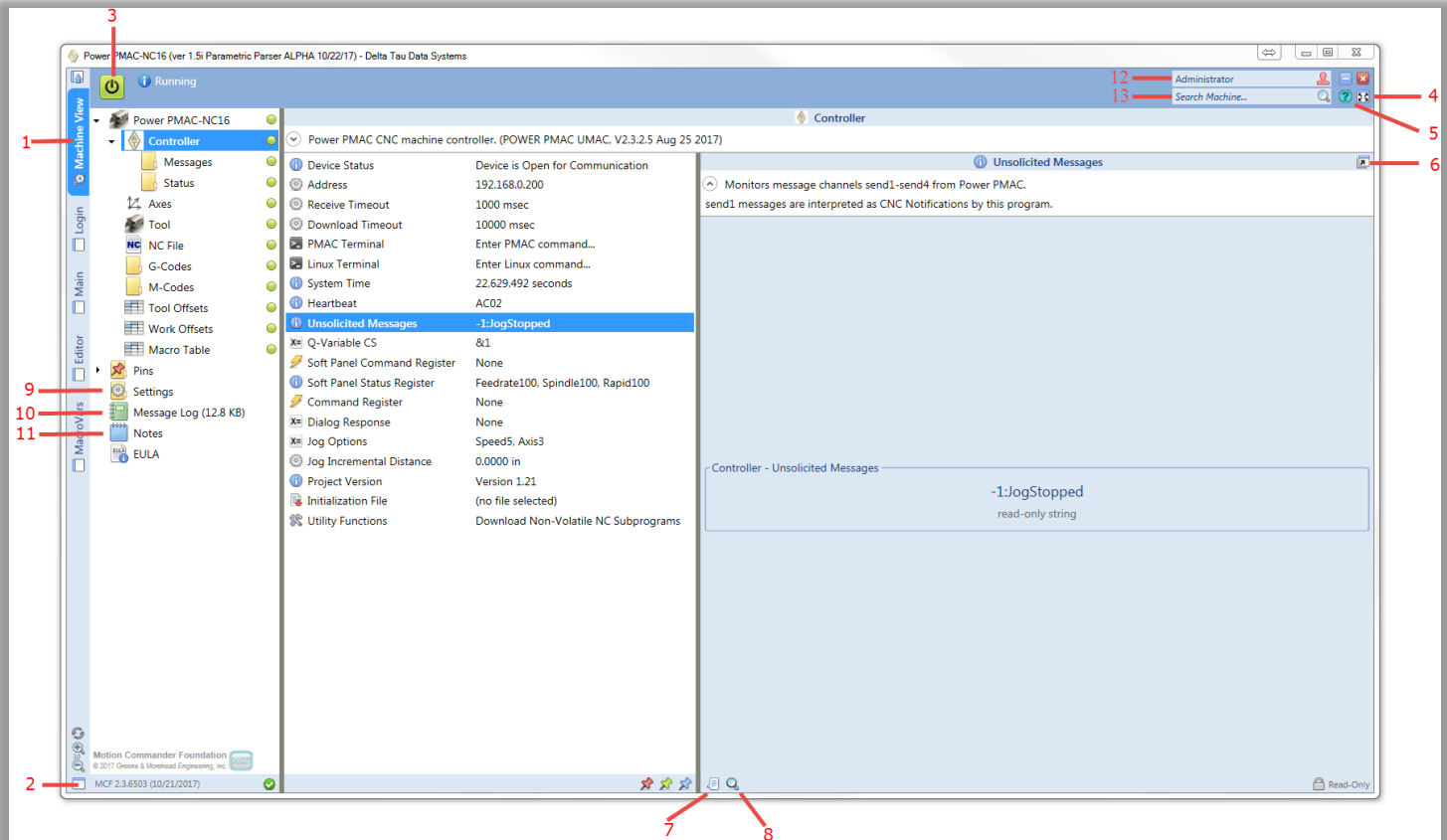
- Type 4 – Pop-Up Dialog Message Boxes
 - Dialog OK Message Box
 - Dialog OK/Cancel Message Box
 - Dialog Yes/No Message Box
 - Dialog Yes/No/Cancel Message Box
 - Dialog Input Message Box (Allows the user to input a value)





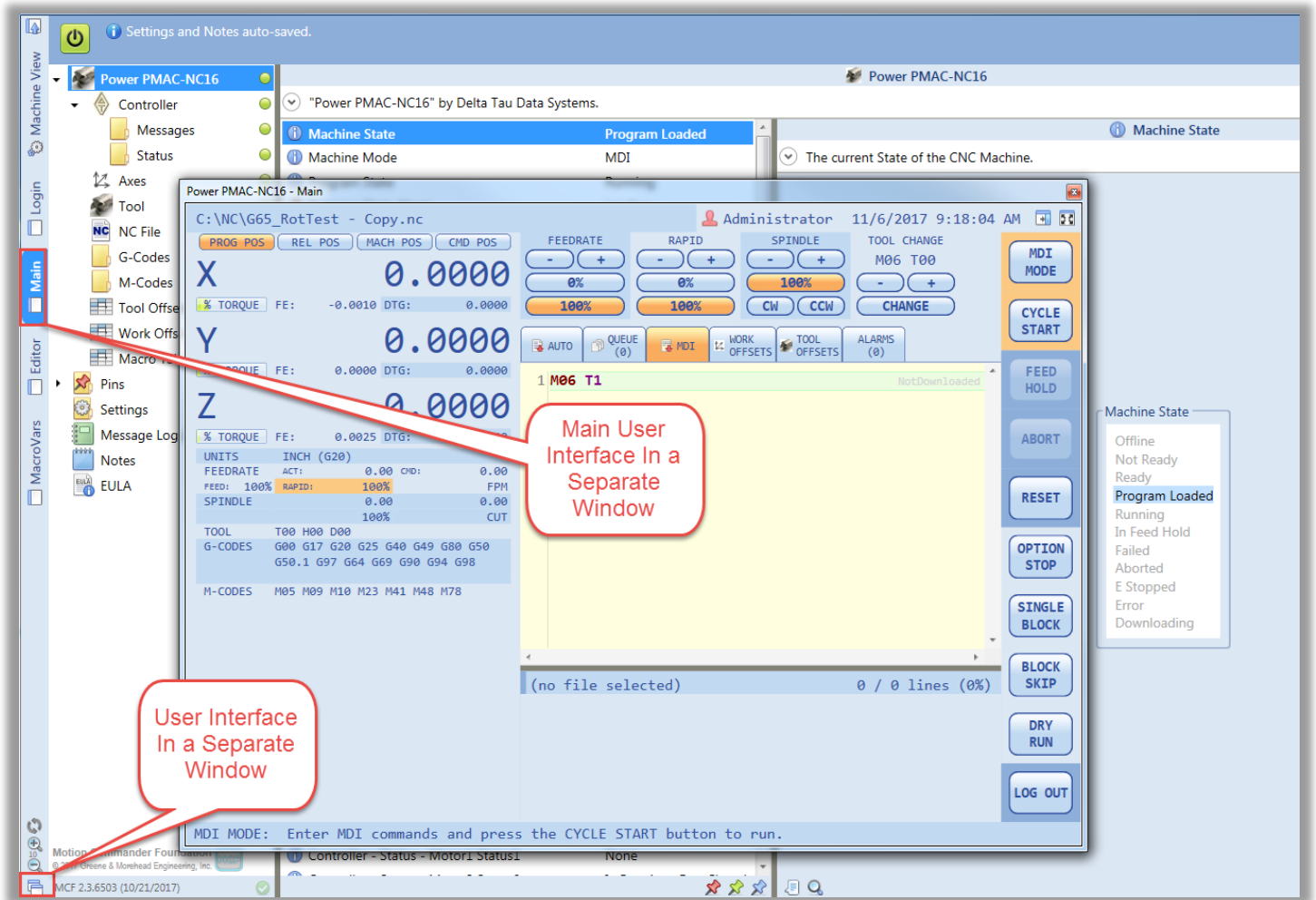
Machine View Tab

The *Machine View* screen is an Explorer-style view of the data shared between the host PC program and the motion controller. Machine View also includes powerful tools for logging and plotting this data, controlling the runtime state of the application, editing settings, viewing the message log, etc.



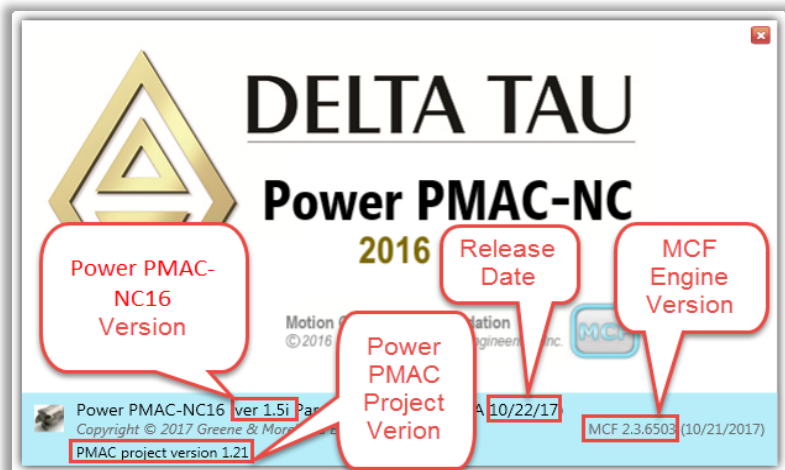
1. Navigate between Machine View and the User Interface.
2. Move the User Interface to a Separate Window.
3. Open/Close Communications, control the update cycle.
4. Toggle Full Screen mode.
5. Open the About Box for version information.
6. Open the member page in a Separate Window.
7. Log changes in value to the Message Log.
8. Open the time-stamped list of changes in value.
9. Edit the program Settings.
10. View the Message Log.
11. Use Notes for punch lists, milestones, production records, etc.
12. Log Out
13. Search Machine

- 1) Machine View Tab and its sections are explained in details in the “Machine View” section of this document.
- 2) Depends on which tab is active, this button (or Shift + Left Mouse Click) Create the user interface in a separate window. Such feature allows users to work simultaneously on both windows as it is shown below:

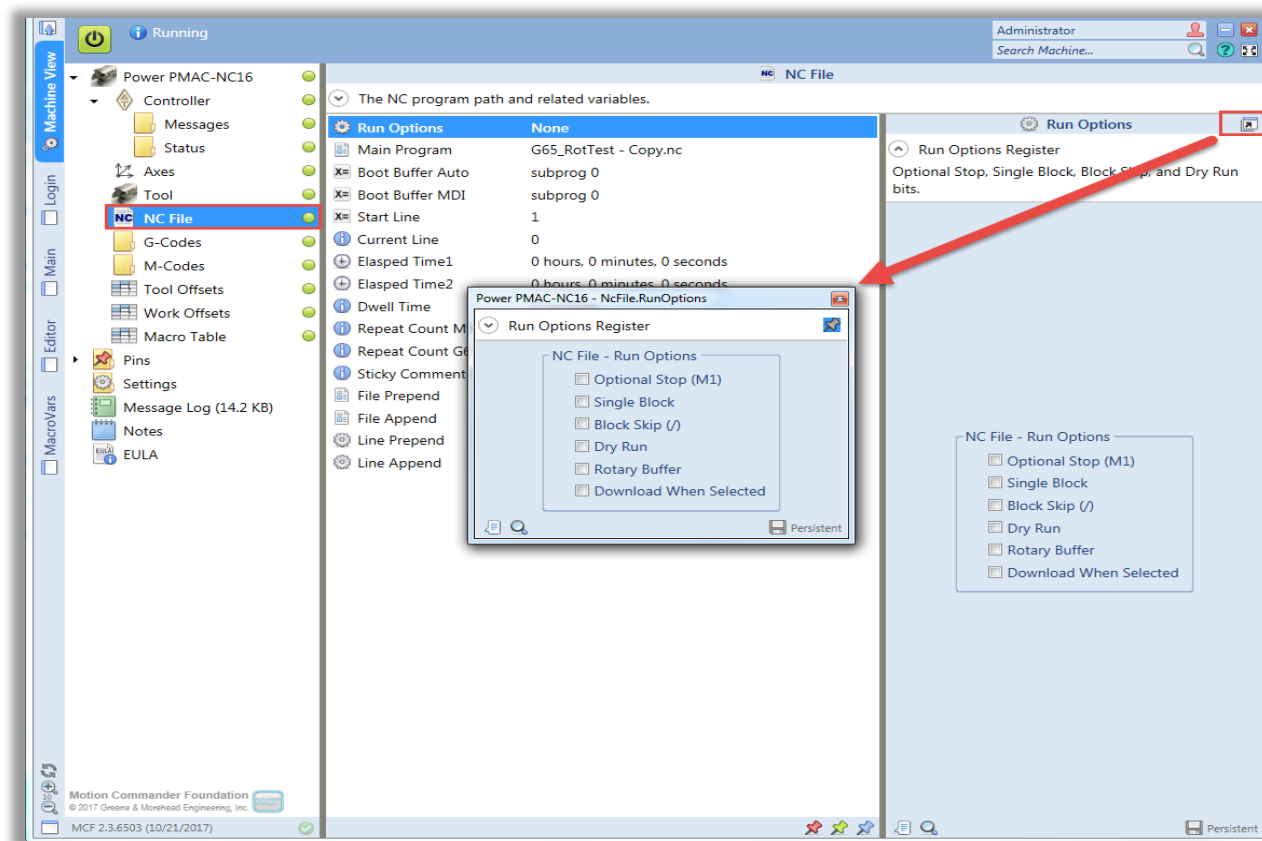


To exit this mode, simply close the separated user interface window. This feature can be used also while Main, Editor, MacroVars, or Login tab is active.

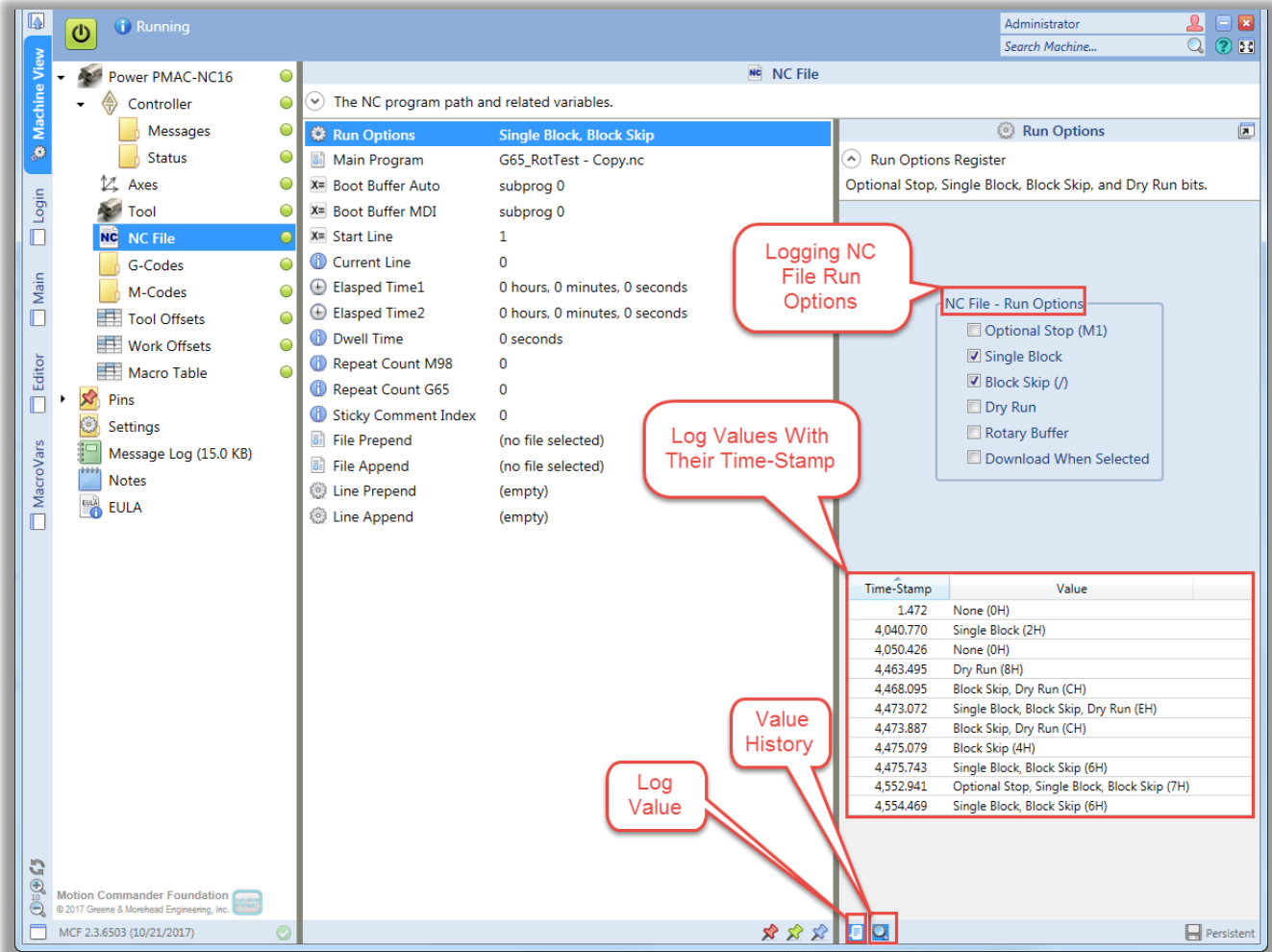
- 3) Connect/disconnect button allows users to connect or disconnect Power PMAC-NC16 to connect or disconnect from a controller (Power PMAC.) This button being “Green” means, Power PMAC-NC16 is connected. Otherwise, Power PMAC-NC16 is disconnected.
- 4) “Full Screen” button, shows the Power PMAC-NC16 in a full screen mode.
- 5) “About” button opens a window which contains Power PMAC-NC16 version information as follow:



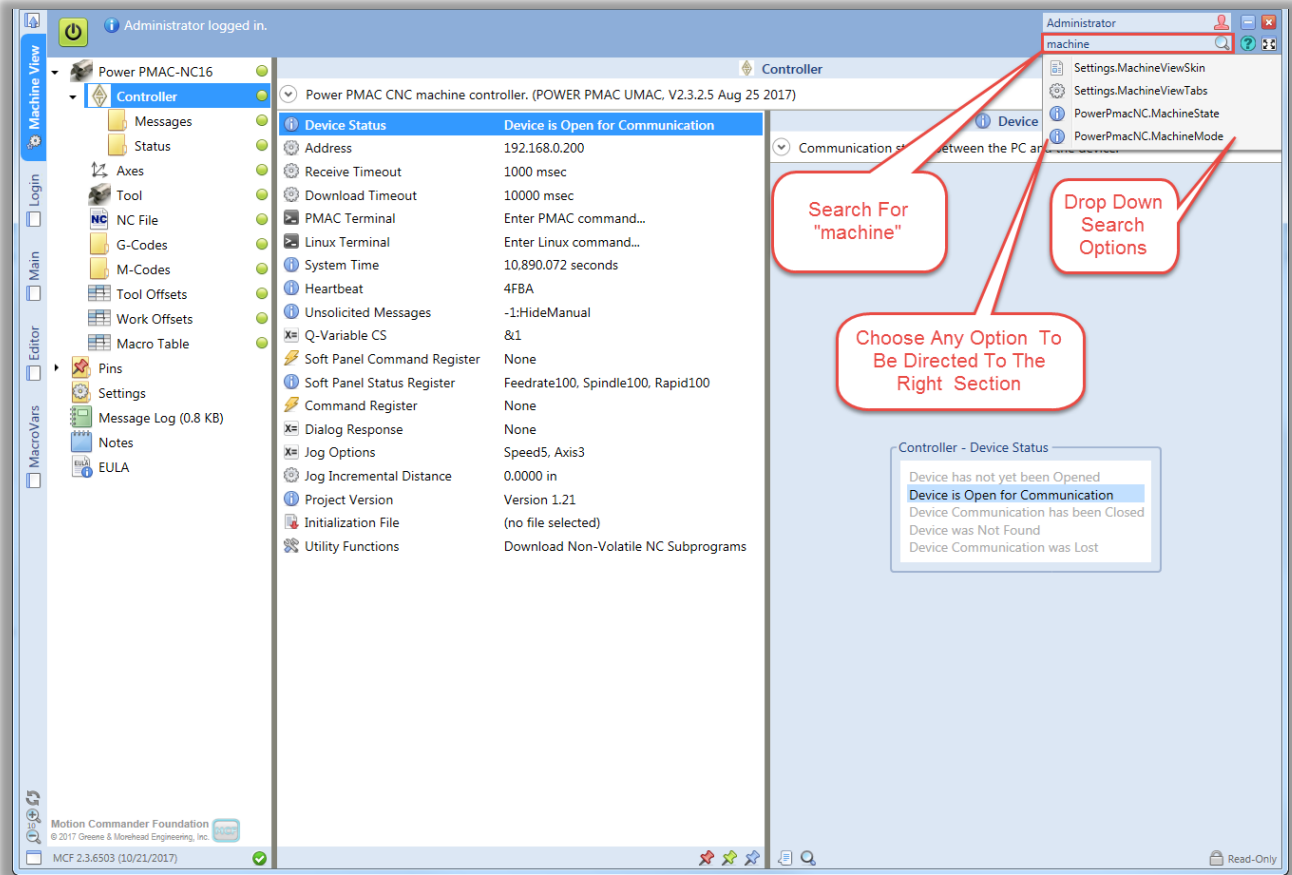
- 6) This button, opens the active member in a separate new window in order to ease accessibility of that specific member. For example, if it is desired to have the “Nc File” run options window open and accessible at all time, click on the “Nc File” member and then click on the “In New Window” button in order to have “Nc File” run options in a new window as follow:



- 7) “Log Values” button is very helpful for trouble shooting or logging any specific member. For example, if it is desired to log the “Nc File” run options, click on “Log Values” followed by “Value History” button which actively shows what and when options have been used, running a NC program file as follow:



- 8) "Value History" button, shows tabulated log values with their time stamps as it is shown above.
- 9) Settings section, contains majority of Power PMAC-NC16 settings such as language support, Jog Speeds, Machine Skins, and etc. This section is explained in more details in "Machine View" section under "Settings".
- 10) "Message Log" is also a very powerful tool that can be used for troubleshooting. It provides three different options to log. Commands, Queries, and Events. This section is also explained in more details "Machine View" section under "Message Log".
- 11) "Notes" is designed to be used for production records, milestones, or etc. This nice editor is also capable of saving figures and photos along with words. Its figure along with some of its features are shown more in details in "Machine View" section under "Not".
- 12) "Log Out" button can be used to log out from the Power PMAC-NC16.
- 13) "Search Machine" can be used to search for any member in the Power PMAC-NC16. This feature looks for the searched key words and if they are found, they will be shown as different search options for users. For example, if "machine" is typed in the search window , Power PMAC-NC16 provides following options:



Machine View

Machine View tab includes six main sections. Each section includes statuses and settings relative to either the controller or Power PMAC-NC16. Following provides a list of sections and subsections which can be seen under the Machine View:

1) Power PMAC-NC16

This section, includes controller (Power PMAC), axes, tool, NC file, G-Codes, M-Codes, tool offsets, work offsets, and macro table information. Each subsection is explained in details along with its main members below.

Machine State: It shows what state the machine is currently at. These states and their descriptions are as follow:

Offline: The controller is not online and/or the GUI is not running

NotReady: The controller is not ready to run a program (not homed, manual mode, or etc)

Ready: The controller is ready to have a program loaded and run

ProgramLoaded: The NC program has been loaded and the controller is ready to start

Downloading: The NC program is being downloaded to the controller

Running: The controller is running the NC program

InFeedHold: The controller is in Feed Hold (paused)

Completed: The NC program has completed successfully

Failed: Failed The NC program has failed

Aborted: The NC program has been aborted by the operator

Estopped: The hardware E-Stop button is pressed

Error: Error condition (lost communication, etc)

Machine Mode:

Auto: Auto Mode for running NC programs

Manual: Manual Mode for Jogging and Homing

MDI: MDI Mode for executing code from the MDI tab

Controller – Device Status:

This member shows the current status of controller in regard of communication.

Controller – Address:

This member holds the controller IP address. This address is used when online button is pressed to establish a communication between Power PMAC NC16 and a controller (Power PMAC.)

Controller – Receive Timeout:

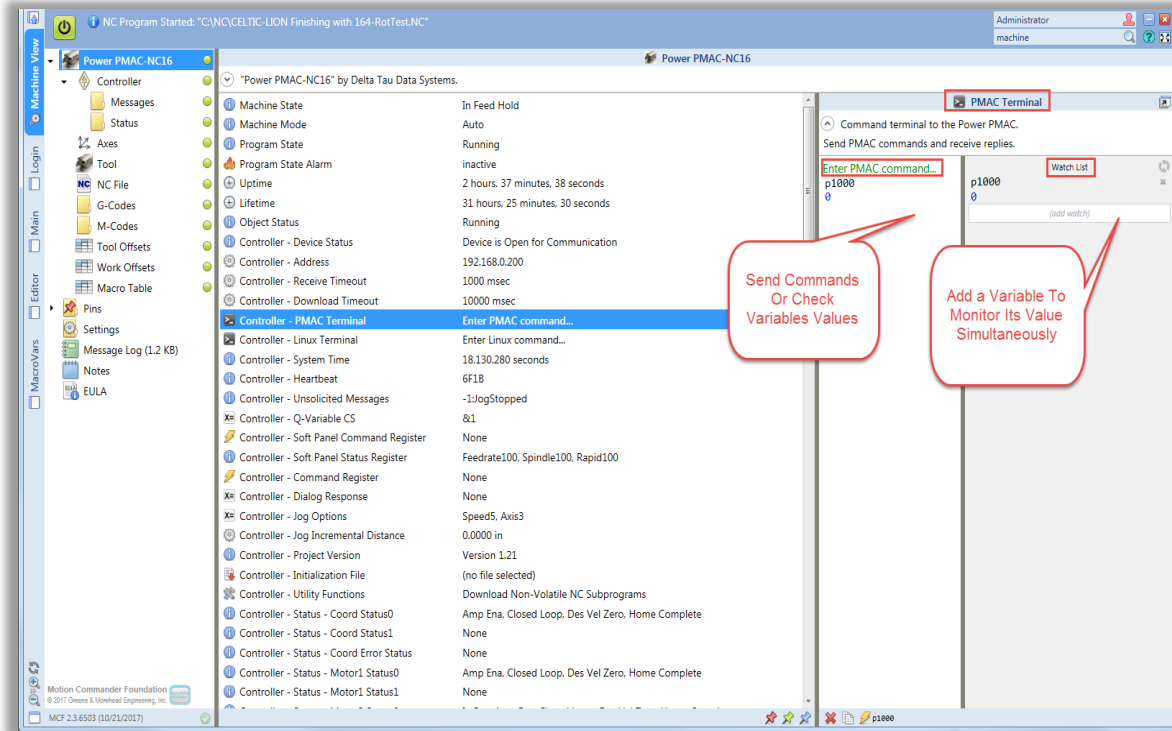
This is the amount of time that a Telnet or SSH read operation will wait for data. If it takes more than a defined time, Power PMAC NC16 will send an alarm.

Controller – Download Timeout:

This is the amount of time that a download will wait for completion. If downloading is in progress and it takes more than a defined time, Power PMAC NC16 will send an alarm.

Controller – PMAC Terminal:

This terminal is used to send commands directly to a controller. This subsection also contains “watch List” which allows to watch variables for troubleshooting or other purposes. For example, value of P1000 can be either checked in a terminal or monitored in a “watch list” as follow:



Controller – Linux Terminal:

This terminal can be used to send bash shell commands and receive replies. The active project is in “/var/ftp/usrflash” and the Power PMAC Linux applications are in “/opt/ppmac”. This terminal is a powerful tool for trouble shooting, using Linux commands as shown below:

```

Enter Linux command...
ls
bkgcommand          libppmac            rtpmacposix
brick               libSecureDongle    sendgetsends
build-ppmacx-linux-g++-32-posix-eip.sh  macropp            setup
build-ppmacx-linux-g++-32-posix.sh      modbus             Sysmac-CNC
ctrlpanel           modbusserver        test
doc                 muxio               testApp
ecmaster            odtAcontis.patch   tune
EcMasterDemo        ot2                 usralgo
etc                 ppconfig            usrflash
gather_csv          ppmachw             usrflash.1
geterrors           ppmacserver         usrflash.2
getsends            ppstruct            usrflash.3
gpascii             projpg              usrflash.4
gppmac              projpp              usrflash.5
libedsparser        Release Notes       version.h
libmath             ringorder
libopener           rtpmac

root@192.168.0.200:/opt/ppmac#
cd usrflash
root@192.168.0.200:/opt/ppmac/usrflash#
ls
Database Project Temp
root@192.168.0.200:/opt/ppmac/usrflash#
cd project
root@192.168.0.200:/opt/ppmac/usrflash/project#
ls
Bin          Configuration Log          PPCNC_ProjectSource.ppproj
C Language   Documentation  PMAC Script Language
root@192.168.0.200:/opt/ppmac/usrflash/project#

```

Controller – Heart Beat:

The heartbeat register is incremented while the Power PMAC NC16 is running. This member can be monitored and to be used for different safety purposes.

Controller – Unsolicited Messages:

This is a read only member that shows the most recent executed command by the controller. Power PMAC NC16 monitors message channels (buffers) send1-send4 from the controller. For example, if axis jogging has been used as the most recent activity, controllers issue a jog stop command, using a plc. In order for Power PMAC NC16 to get acknowledged, Send1 “JogStopped” is sent by a controller as follow:

The screenshot displays the Power PMAC-NC16 software interface. The left sidebar shows the 'Machine View' with various components like Controller, Messages, Status, Axes, Tool, NC File, G-Codes, M-Codes, Tool Offsets, Work Offsets, Macro Table, Pins, Settings, Message Log (1.2 KB), Notes, and EULA. The main window shows the 'Controller - Unsolicited Messages' section, which is highlighted. A red arrow points to the '-1:JogStopped' message, labeled 'Most Recent Received Message'. Below this, a table titled 'Logged Recent Received Messages For Troubleshooting' lists various messages with their time-stamps and values.

Time-Stamp	Value
1.503	(empty)
5.716	-1:Initialized
5.723	-1:requestmanualmode
5.739	-1:HideManual
7.411.477	-1:Initialized
7.411.479	-1:requestmanualmode
7.411.489	-1:HideManual
7.607.642	-1:RequestAutoMode
7.612.227	-1:RequestMdiMode
7.617.925	-1:RequestOptionStop
7.619.036	-1:RequestOptionStop
7.624.138	-1:RequestManualMode
7.624.149	-1:ManualSubmodeHandk
7.676.385	-1:JogStopped
7.676.386	-1:Initialized
7.676.388	-1:requestmanualmode
7.676.398	-1:HideManual
7.676.457	-1:JogStopped
7.736.261	-1:CycleStarted
7.736.412	-1:InFeedHold
7.736.846	-1:requestmanualmode
7.736.855	-1:HideManual
7.736.870	-1:JogStopped

Controller – Q-Variable CS:

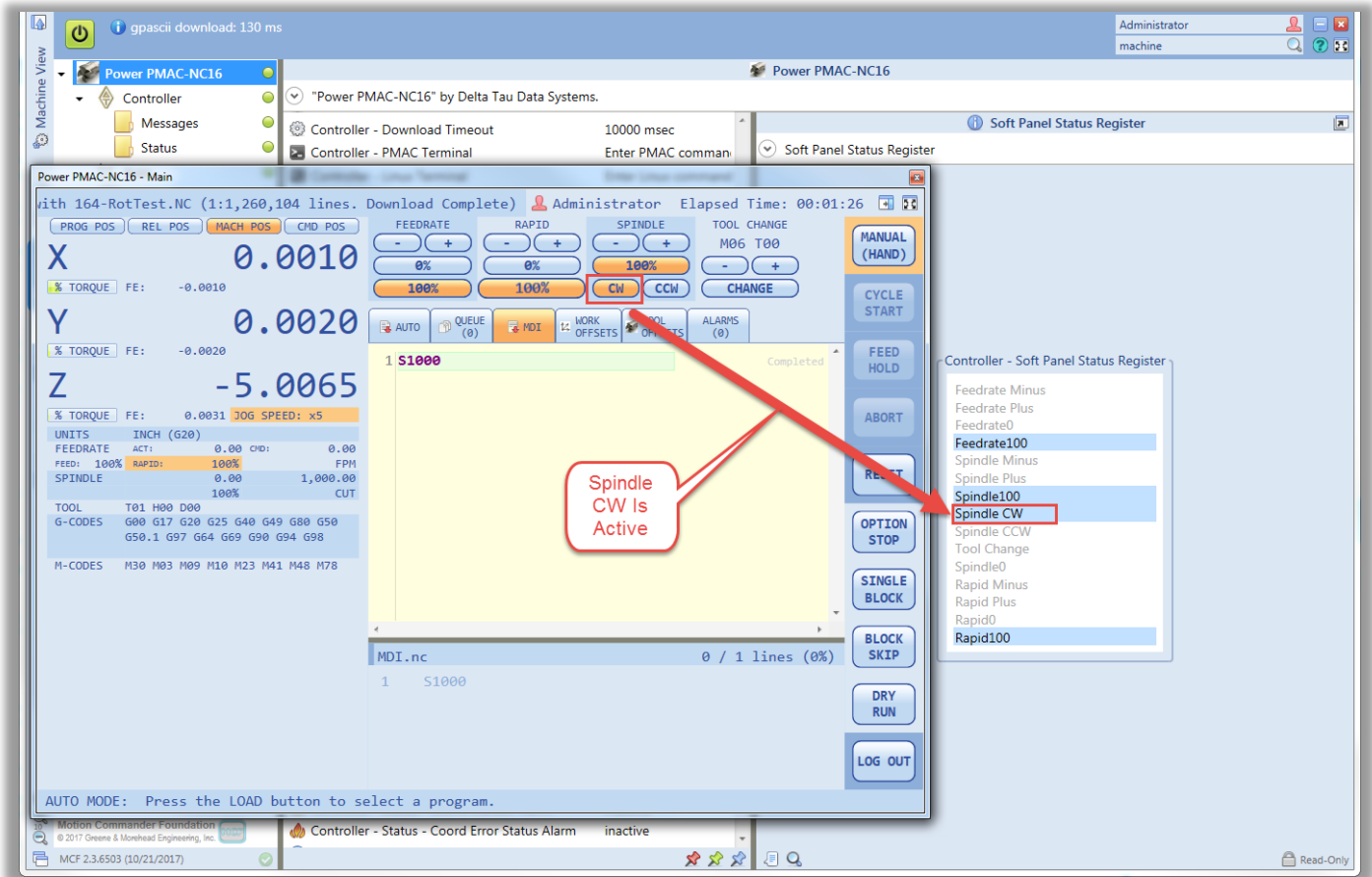
This member defines what coordinate system is required to be used when any Q-Variable is getting executed in a program. Since Q-Variables are coordinate system specific, they can have different values with respect to different coordinate system.

Controller – Soft Panel Command Register:

Shows the latest activity done using the soft panel in Power PMAC NC16. Each command is set by a PC which waits for controller to clear the command before setting another.

Controller – Soft Panel Status Register:

This member shows the status of soft panel members. For example, if “CW” is pressed on the soft panel, soft panel register shows Spindle CW member is active as follow:

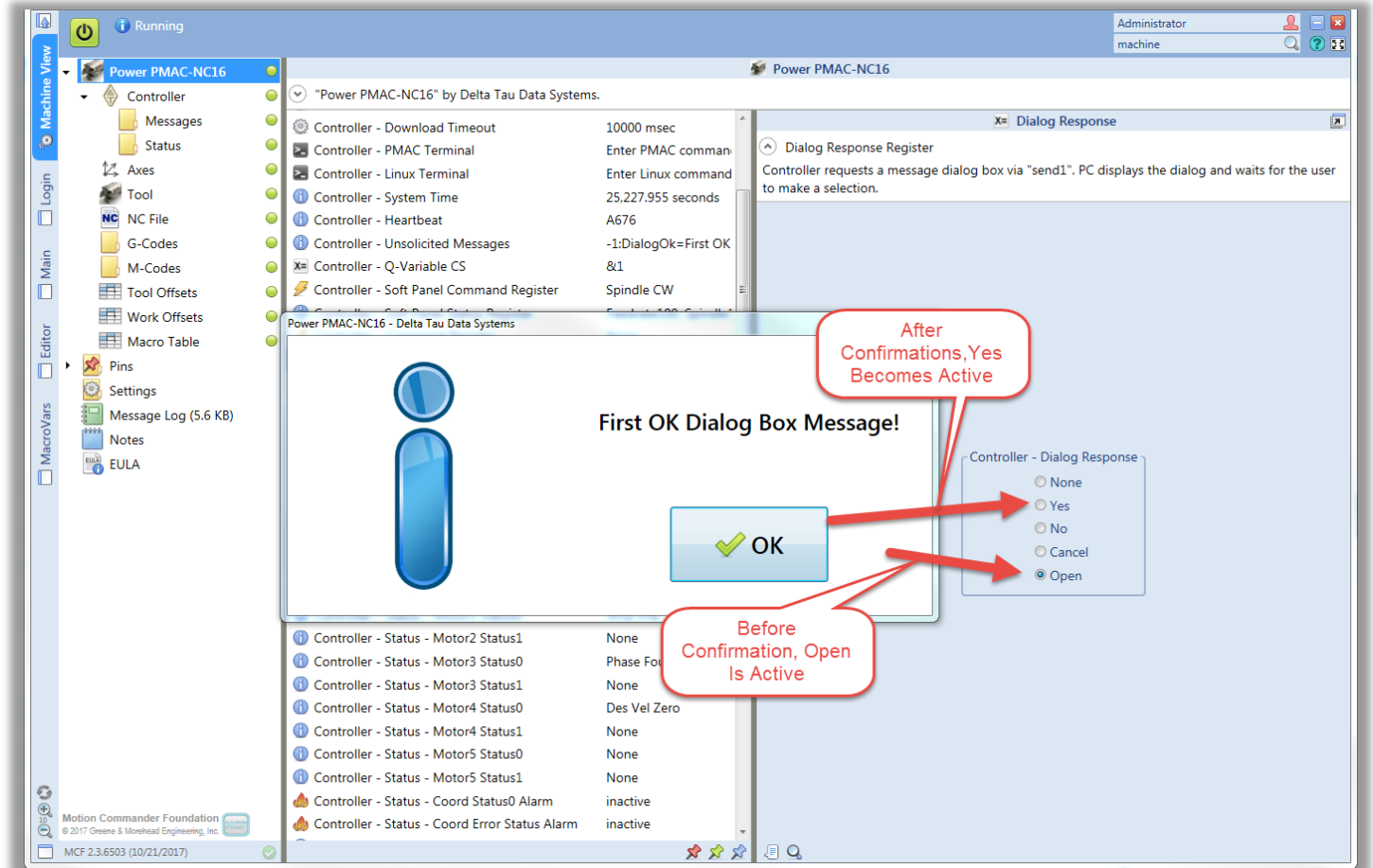


Controller – Command Register:

This member shows the latest set command by the Power PMAC NC16. After Power PMAC NC16 sets a command, controller clears the command on receipt and performs the action. Power PMAC NC16 waits for controller to clear the command code before setting another.

Controller – Dialog Response:

This member shows the latest user selection or response to a message dialog box. If the dialog box is active and Power PMAC NC16 has not received any response, status will be “Open”. Then based on chosen response, this member will get set to either “Yes”, “No”, or “Cancel”.



Controller – Jog Options:

This member shows what axis and what jog speed are chosen.

Controller – Initialization File:

This file (initialization file) is automatically downloaded when communication is opened with the controller. It is very easy and powerful tool to initialize controller settings at machine startup.

Controller – Utility Functions:

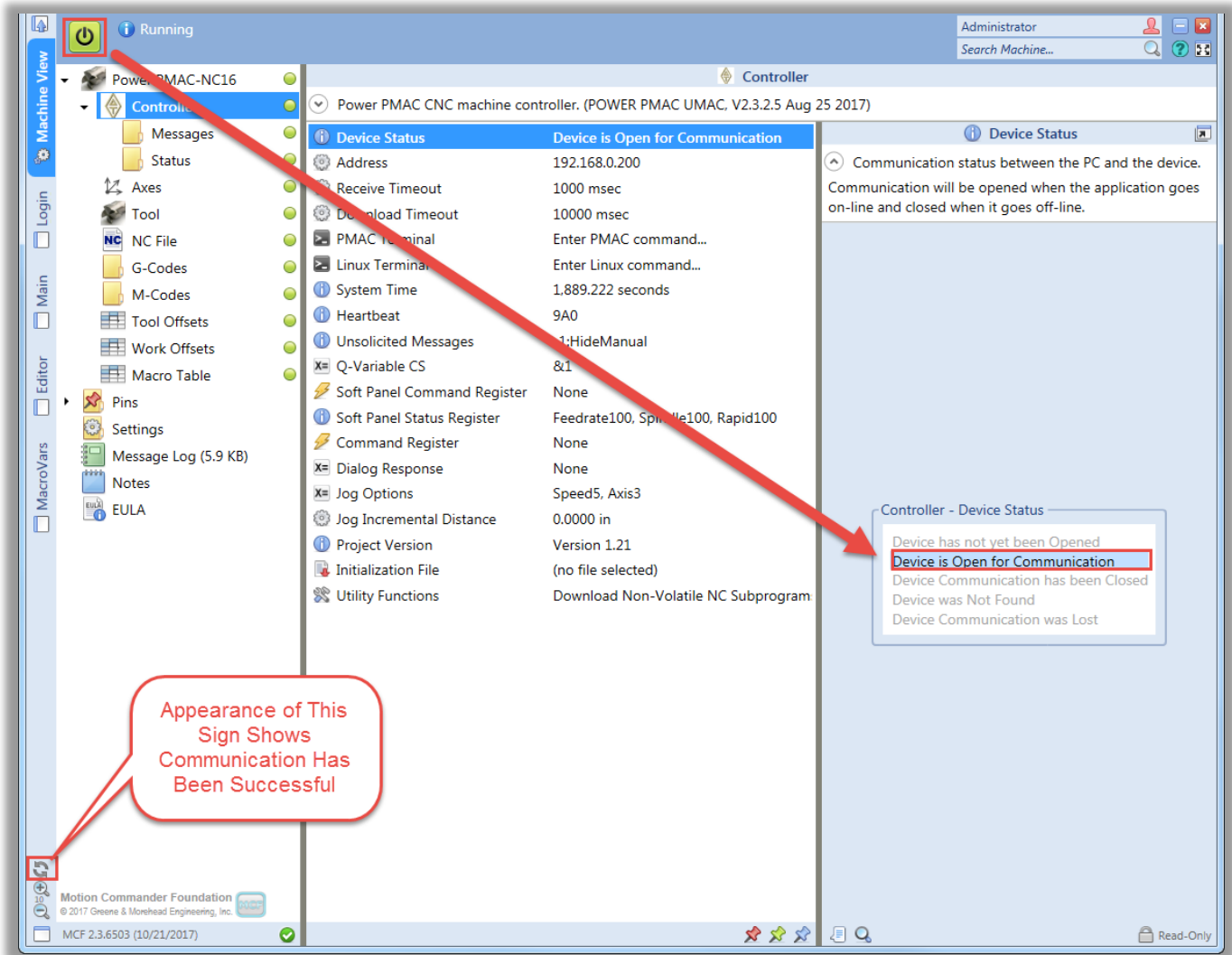
This utility allows downloading Non-Volatile NC subprograms to the controller. Controller will store these programs for different future use.

a) Controller

Controller is a subsection of Power PMAC-NC16 which its members are settings and statuses to set or report some of the controller's features.

Device Status:

This member is designed to report the status of communication between Power PMAC NC16 and the controller (Power PMAC.)



Controller Address:

This member is used to set an Internet Protocol (IP) address for the controller (Power PMAC) in order to be used to establish a communication. The factory default address is 192.168.0.200.

System Time:

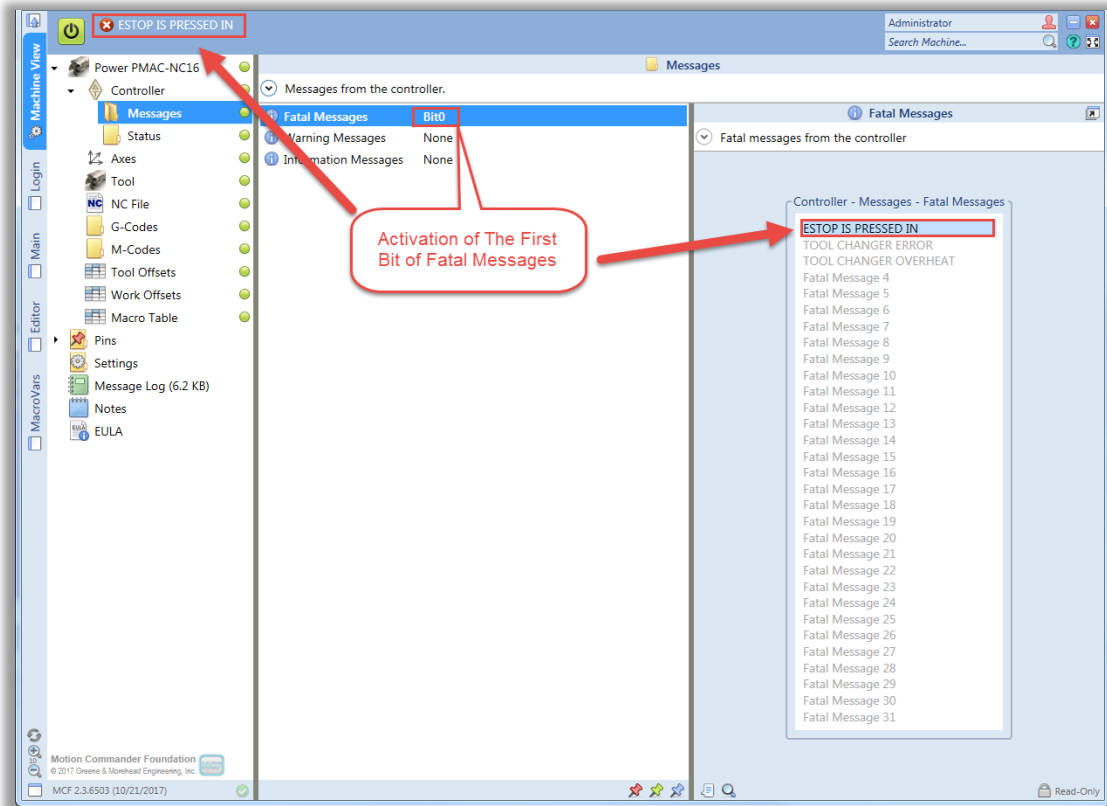
This member reports the controller up time. It reports the value of "Sys.Time" in seconds.

Project Version:

This member reports the version of a project which is used by the controller (Power PMAC) to perform the handshaking with Power PMAC NC16.

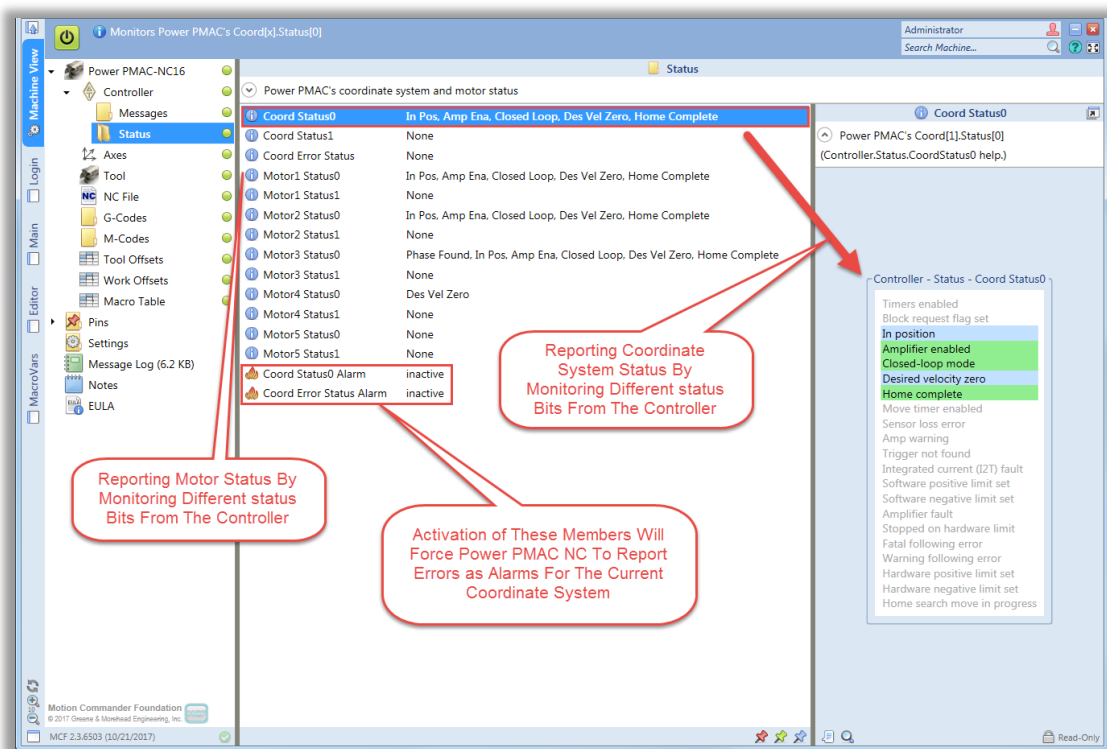
i) Messages

This subsection of *Controller* shows active fatal, warning, or information messages. As it was mentioned before, each type provides 31 messages which can be used for different purposes as follow:



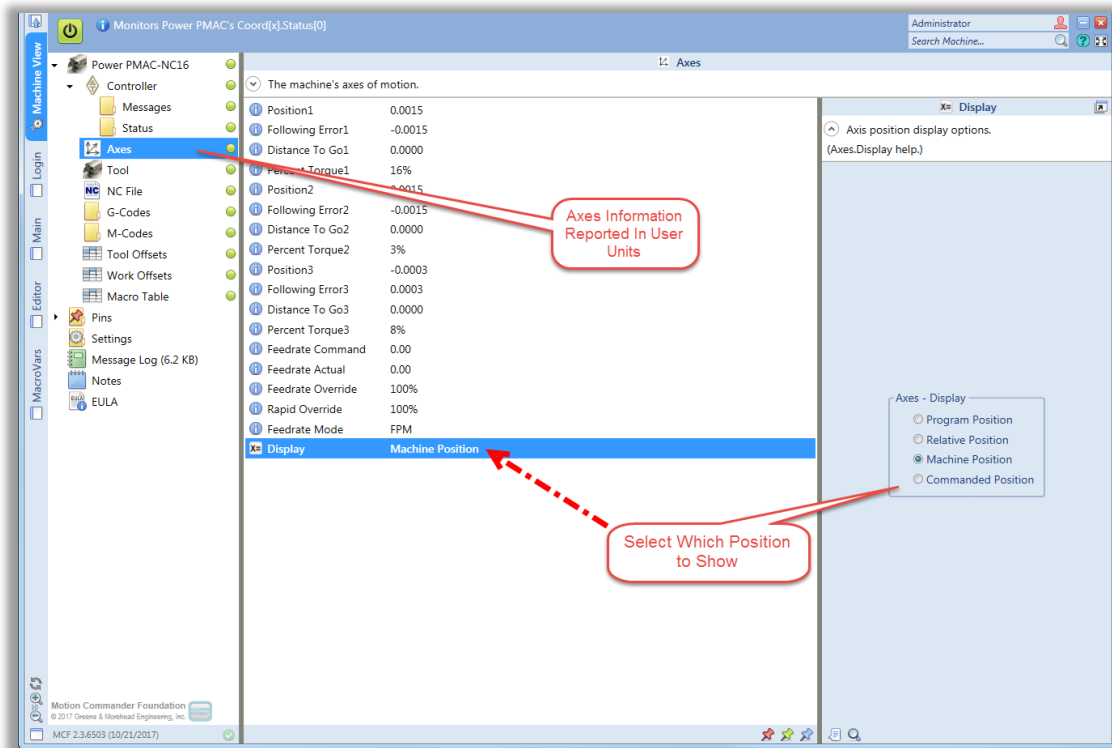
ii) Status

This member shows the status of coordinate system and its motors by reporting directly from controller's coordinate system and motors status bits.



b) Axes

This *Controller* subsection works in conjunction with the “Main” window to reports axes information such as positions, following errors (servo deviations), torques, and etc.

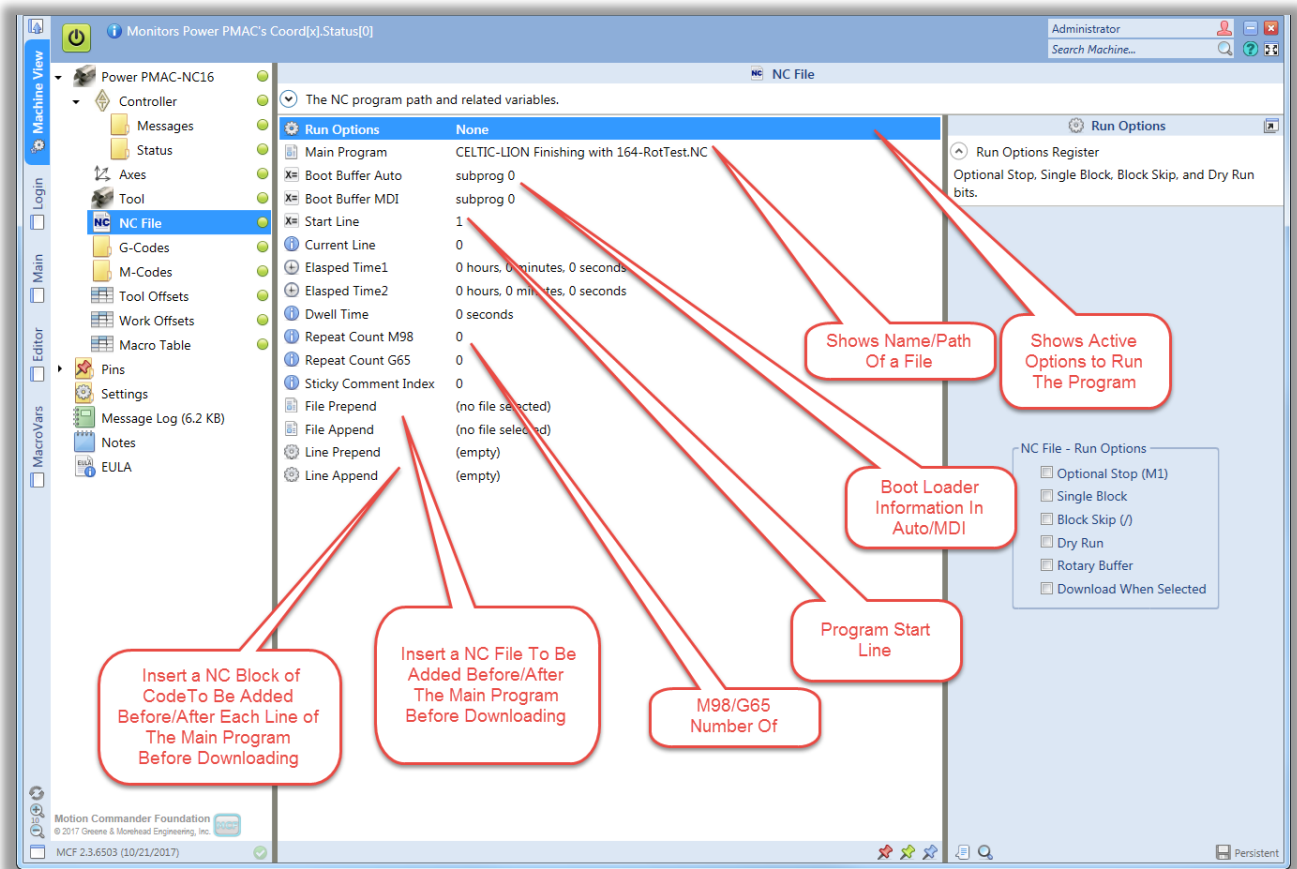


c) Tool

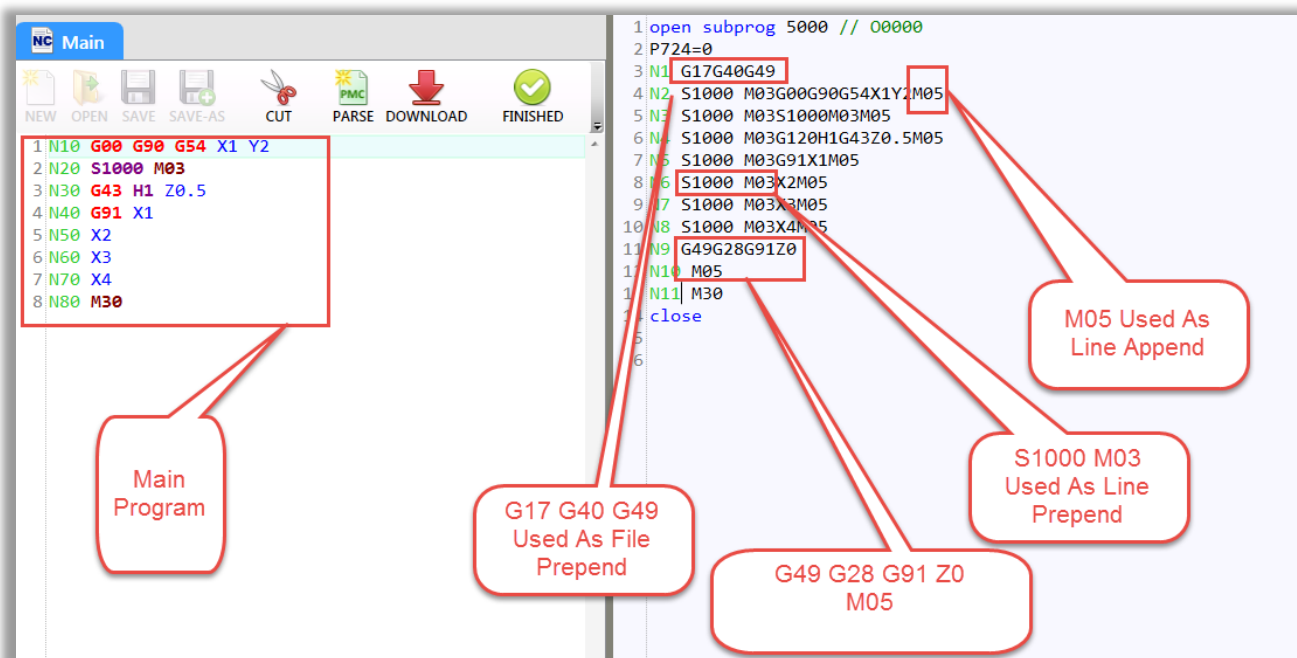
This *Controller* subsection is designed to report tool and spindle status. It works in conjunction with the “Main” window to show Spindle Command (latest executed S-Command), actual spindle RPM, spindle over ride percentage, and spindle mode (CUT or Constant Surface Speed.) It also shows the latest executed T,H, and D-Code.

d) NC File

This subsection of *Controller* is designed to provide information in regard of a loaded program in “Auto” mode as it is shown below:



File prepend/append are two powerful tools provided to users to select a file (*.NC) to be added in the beginning or at the end of the main program respectively. Line prepend/append are also powerful tools provided to users to insert a block of code to the beginning or end of each line in the main program. Following figure demonstrates all these four capabilities of Power PMAC NC 16:



e) G-Codes

This *Controller* subsection demonstrates active G-Codes with their definitions. Members belong to each group can be seen by selecting of each group on the right side.

f) M-Codes

This *Controller* subsection demonstrates active M-Codes with their definitions. Members belong to each group can be seen by selecting of each group on the right side.

g) Tool Offsets

This table is the extended version of “Tool Offsets” tab in “Main” with the difference of extra ability to modify tools’ descriptions. Selecting each cell shows its value, native user unit, and P-Variable assign to it.

h) Work Offsets

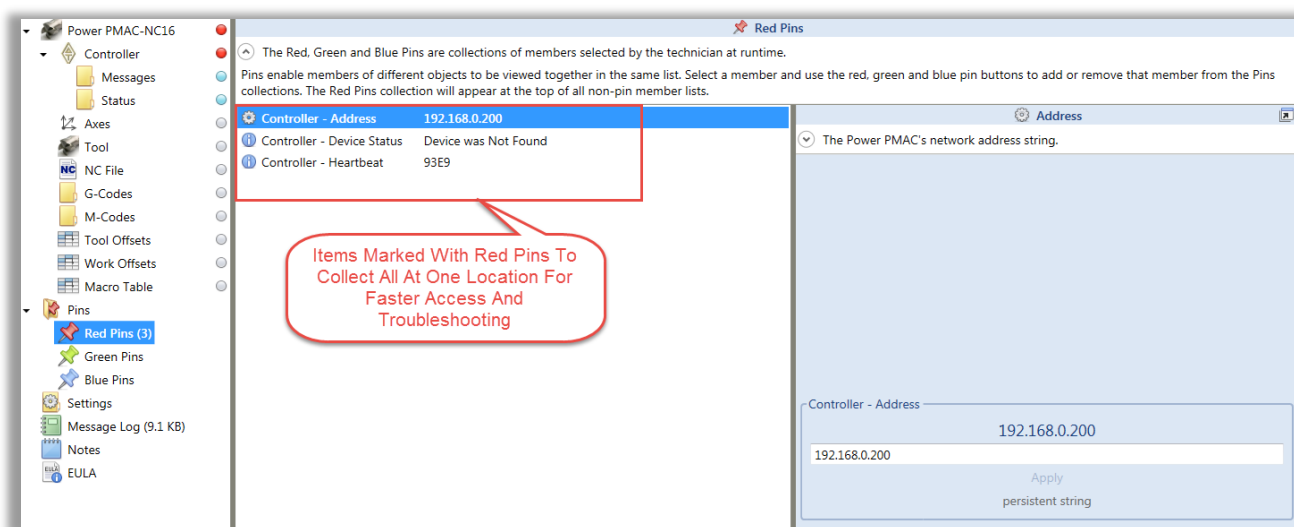
This table is as same as “Work Offsets” tab in “Main” with the difference of extra ability to modify tools’ descriptions.

i) Macro Table

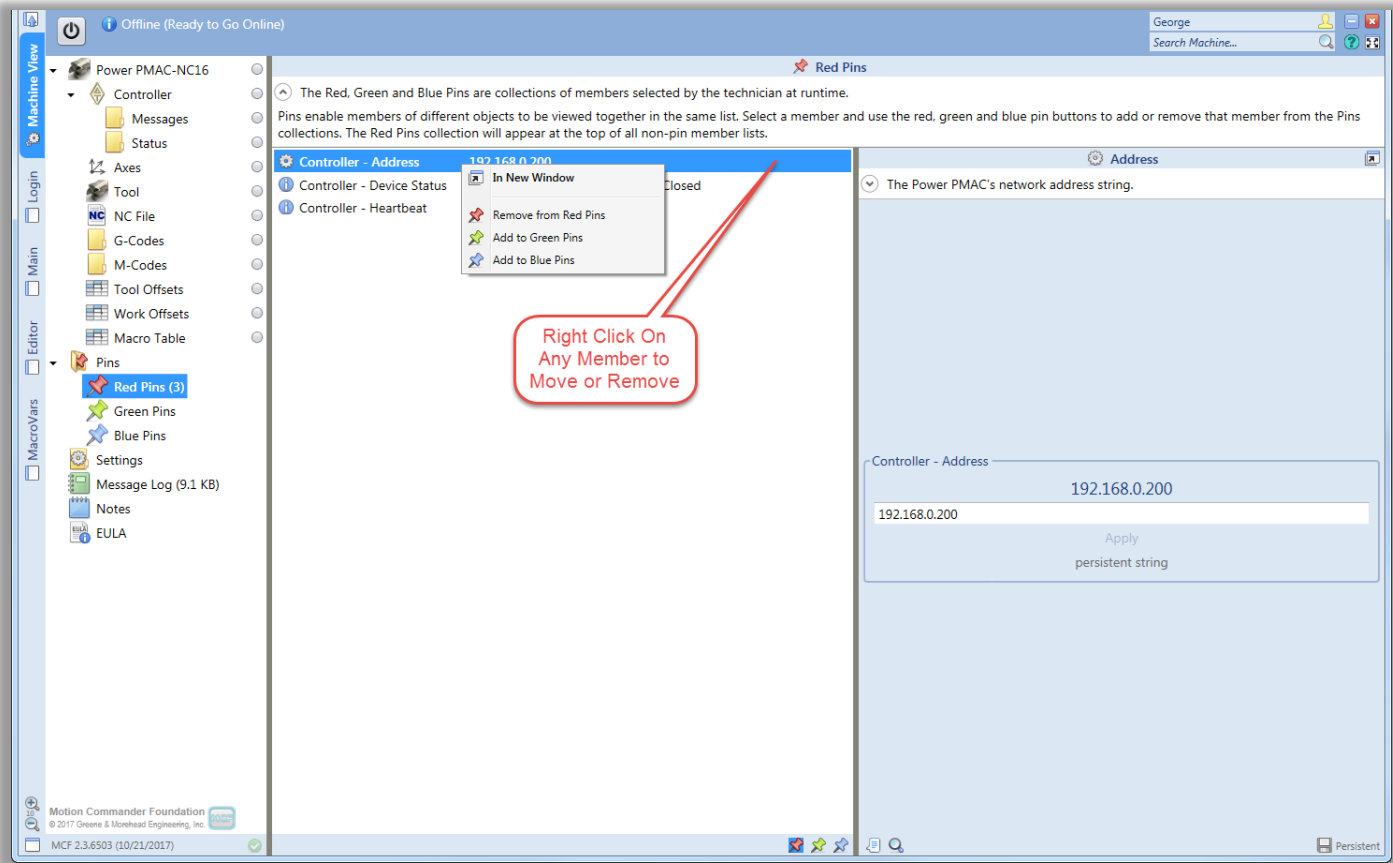
This table works as same as “MacroVars” tab which shows values of local and global macro variables.

2) Pins

Pins are designed to collect and list different members at one place for technicians in order to perform faster troubleshooting. Following is a simple example of communication troubleshooting:



Red, Green, and Blue pins can be used to organize members in three different categories. If it is desired to remove the member or move it from one group to another, right click on any listed member and select the desired action as follow:



3) Settings


This section includes general Power PMAC NC16 settings such as start up or shut down procedures, update interval, users, and etc.

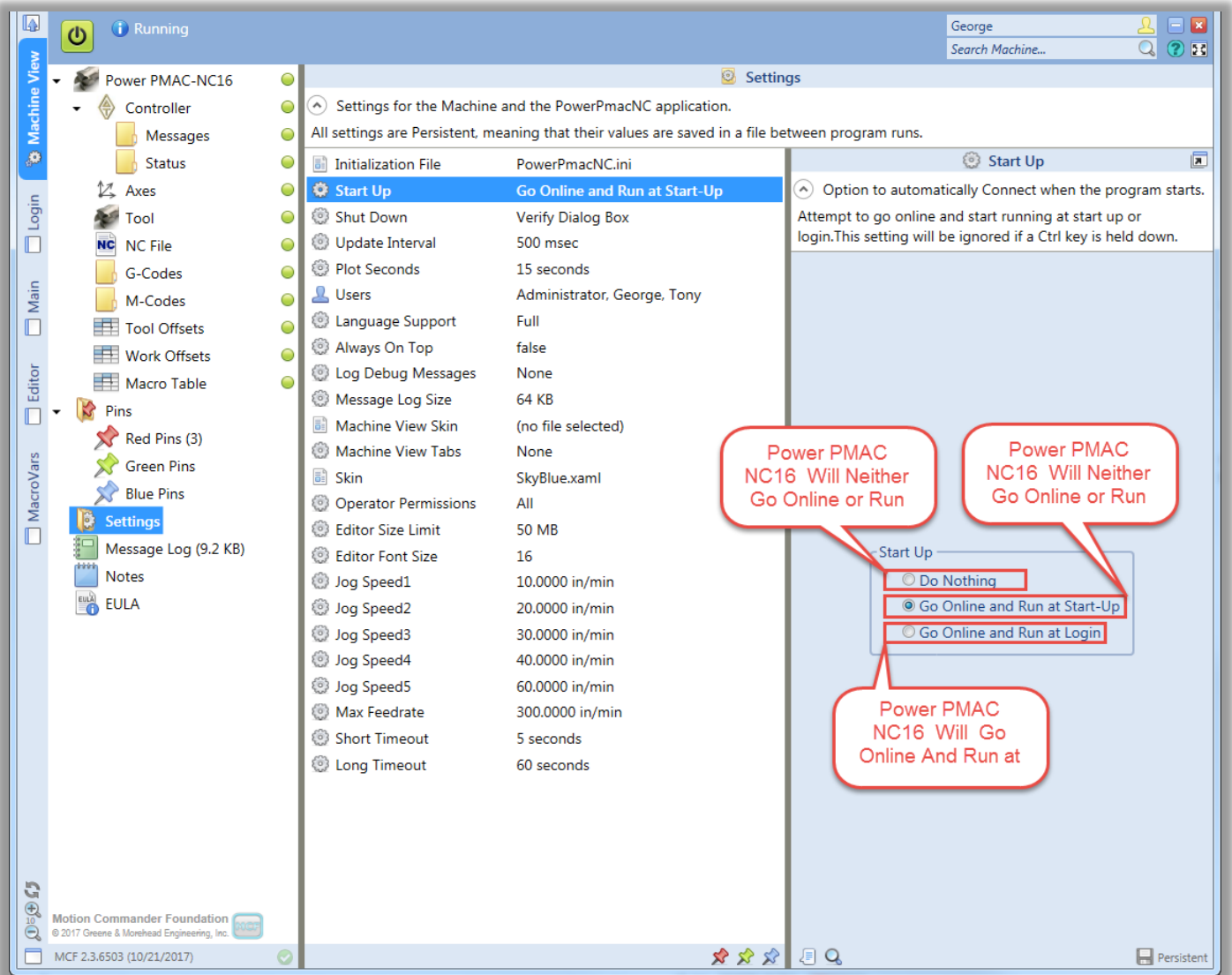
Initialization File:

This section allows users to modify or load **PowerPmacNC.ini** file. If it is desired to change this file, select the member and apply any modifications on the right side. Changes will become effective after restarting the application. If it is desired to change the file completely, either use a drag or drop feature or "Select File" icon to locate the desired file. For more details, refer to **Power PMAC NC ini Configuration Manual**.

Start Up:

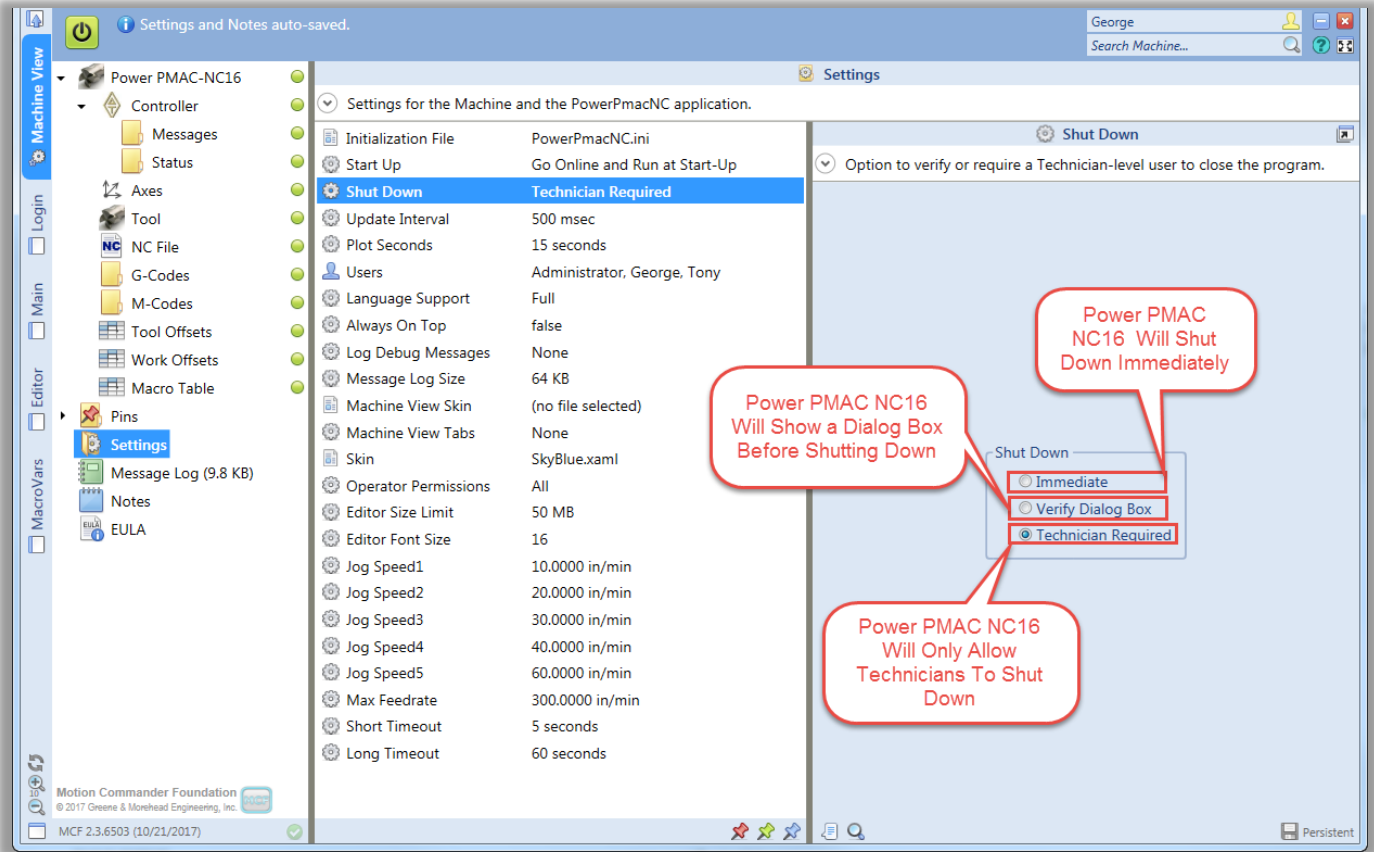
This member address start up procedures taken by Power PMAC NC16. Three options provided by this member are as shown in the figure below:

 Navigate to Machine View to change application **Settings** and manage the list of Users. In particular, you will most likely want to change the "Start Up" setting to "Go Online and Run at Login".



Shut Down:

This member address shut down procedures taken by Power PMAC NC16. Three options provided by this member are as shown in the figure below:

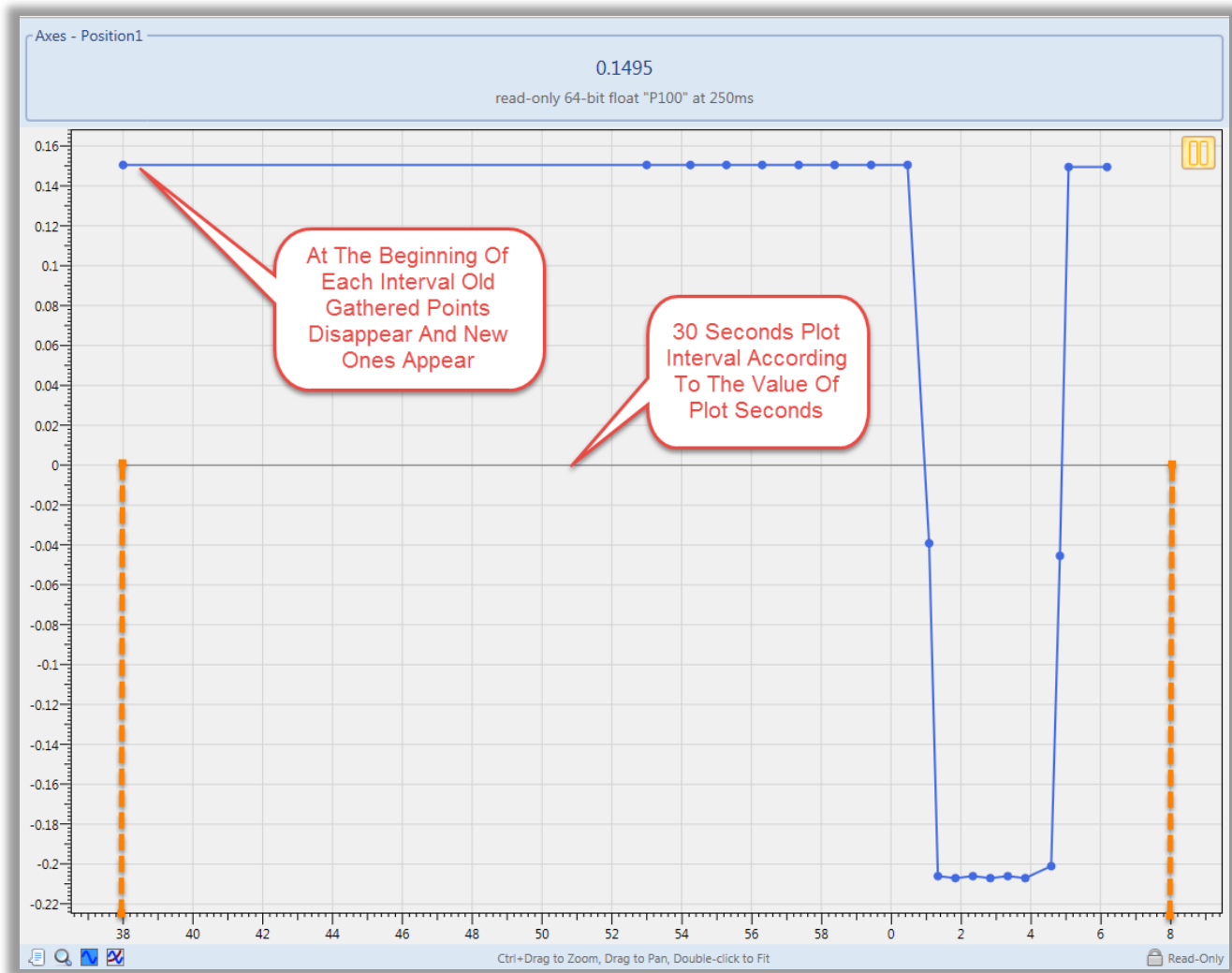


Update Interval:

This member defines the update interval in unit of milliseconds. It is used by Power PMAC NC16 to update controller's heart beat and machine events at specified intervals. Controller monitors the Power PMAC NC16 heart beat for different customize shutdown safety procedures. Machine events are designed to handle the controller and Power PMAC NC16 handshaking in a way that Power PMAC NC16 sets the event and the controller respond to it.

Plot Seconds:

This member defines a display intervals in seconds for plotting. This means, Power PMAC NC16 refreshes and shows gathered samples for a set value in seconds. For better understanding of what this member does, following figure is provided:



Users:

The user login system supports four access levels.

- Operator** Access to the Operator screens but no access to Machine View
- Supervisor** Access to the Operator screens and read-only access to Machine View
- Technician** Unrestricted access to all UI pages and Machine View (except the list of Users)
- Administrator** Unrestricted access including the list of Users

💡 Log in as the Administrator and navigate to Machine View Settings to manage the list of Users. Following figures demonstrate how users list can be managed and used:

Users

List of users for the login system. (Administrator access only)

Settings.Users

Name	Language	Level	Password	Hint
E. Chalumeau	French	Administrator		
Friedrich Nietzsche	German	Technician		
Sun Tzu	Chinese	Technician		
The Administrator	English	Administrator	1234	One through four.
The Operator	English	Operator		
The Supervisor	English	Supervisor		
The Technician	English	Technician		

User Name: **The Administrator**

Password: **E. Chalumeau**

Friedrich Nietzsche

Sun Tzu

The Administrator

The Operator

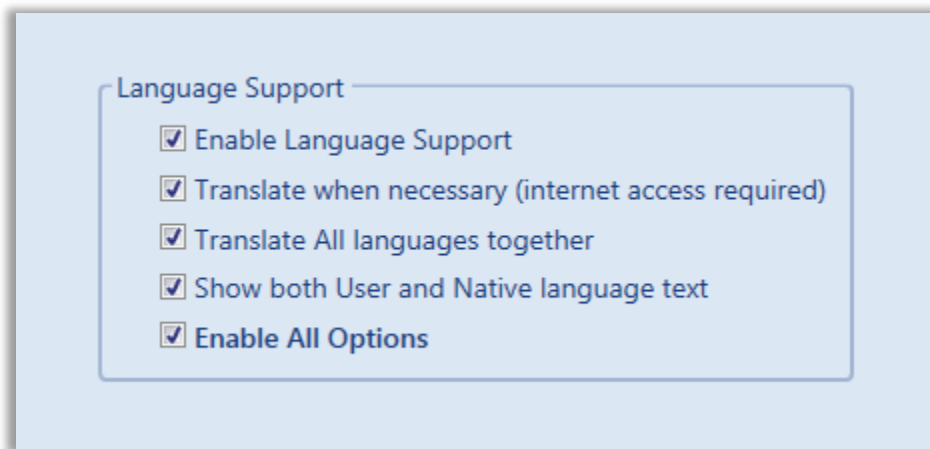
The Supervisor

The Technician

LOG IN SHUT DOWN

Language Support:

This member introduces extra flexibility by providing options in regard of language support and translation. Following figure is provided as a reference to show these available options:

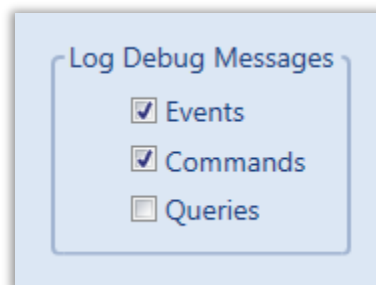


Always On Top:

When this member is set to "true", Power PMAC NC16 will be shown always on top of other open windows in a non-full screen mode in the windows operator system display.

Log Debug Messages:

Three options are provided by this member to control the logging functionality of messages by Power PMAC NC16 for debugging purposes. These options are also available on the "Message Log" window. These options are shown below:



Message Log Size:

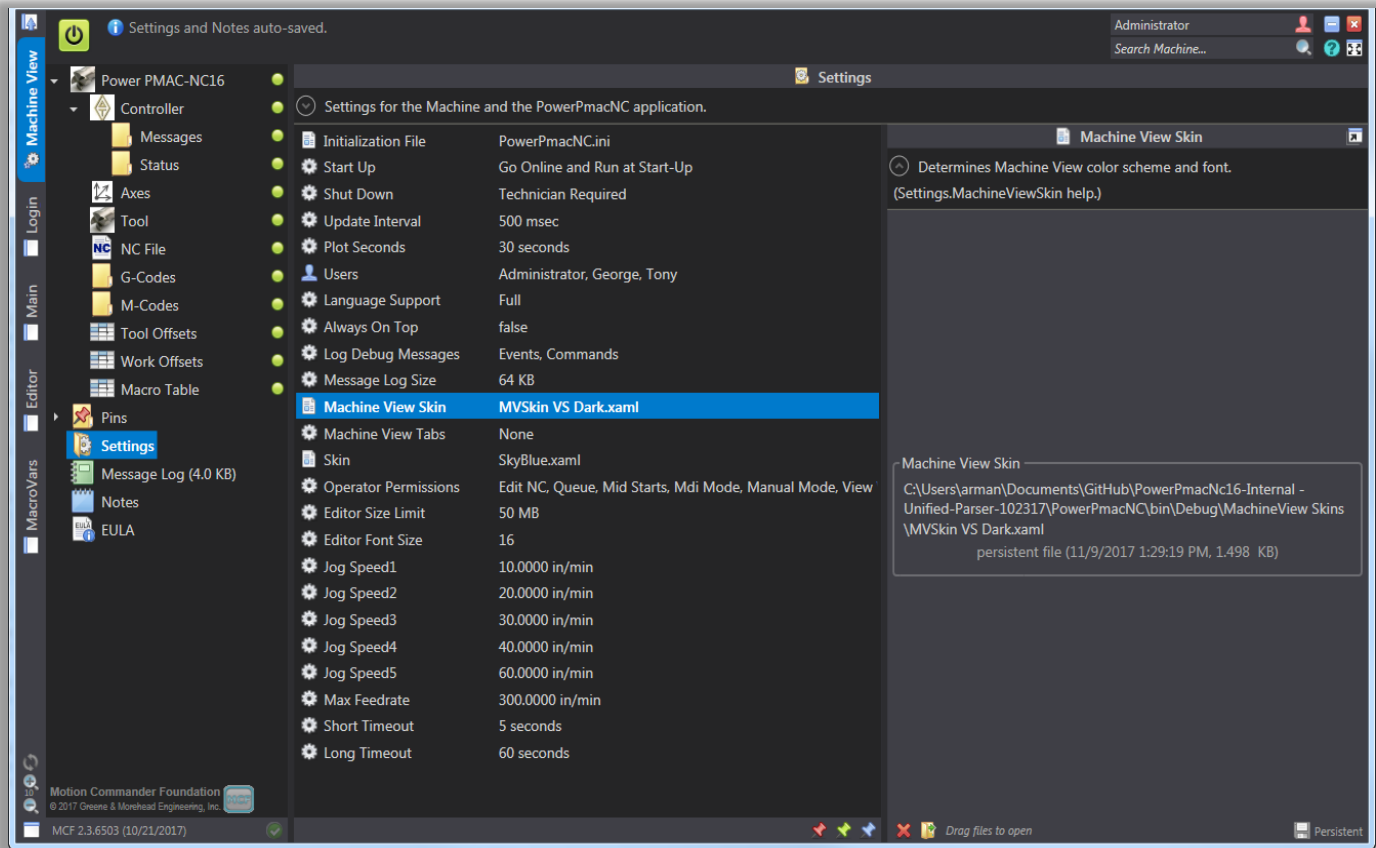
This unique feature allows users to define a log file size (64KB by default.) As primary log file reaches this size, it will be closed as the secondary log file and deleted to conserve disk space.



Log files will not be created if this value is set to zero.

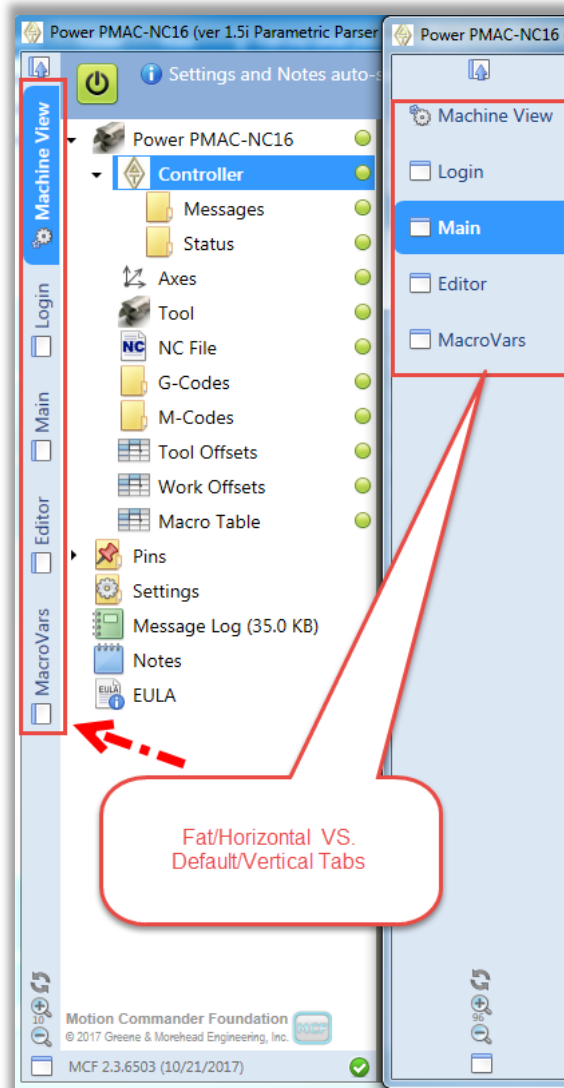
Machine View Skin:

This member determines "Machine View" color scheme and font. Following figure, demonstrates one of the options:



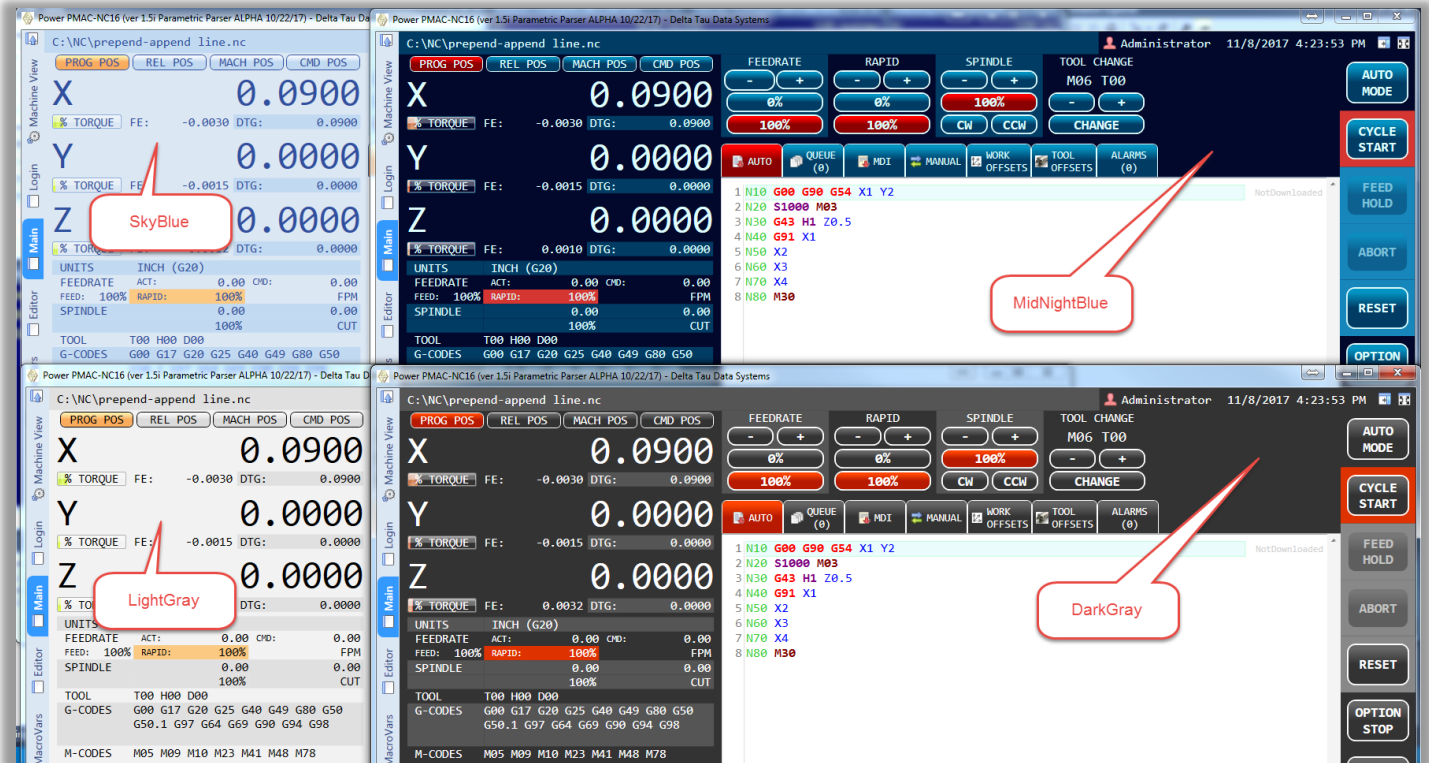
Machine View Tabs:

This member offers different options to introduce more flexibility when Power PMAC NC16 is used with touch screen. “Fat Tabs” provides more touching surface area on screen in order to ease accessibility of Power PMAC NC16 tabs regardless of being horizontal or vertical. “Horizontal Tabs” switches the orientation of Power PMAC NC16 tabs from being vertical to horizontal. “Always Visible” makes Power PMAC NC16 tabs visible at log out. Following figure is provided as a reference to demonstrate the functionality of these options:



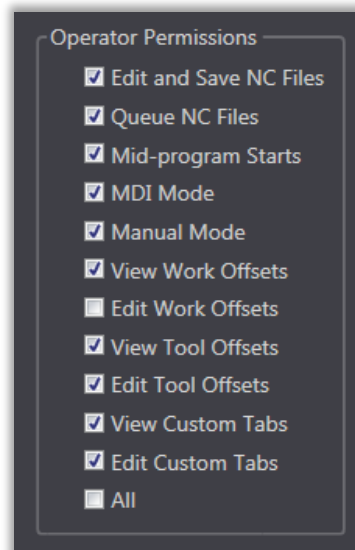
Skins:

This member determines “Main Screen” color scheme and style. Four premade skins are available under the “Skins” folder as part of Power PMAC NC16 package and if it is desired custom skins can be made, added, and used by users. Double click on a member (or drag and drop), select the desired skin, and apply. Following figure shows these four premade skins:

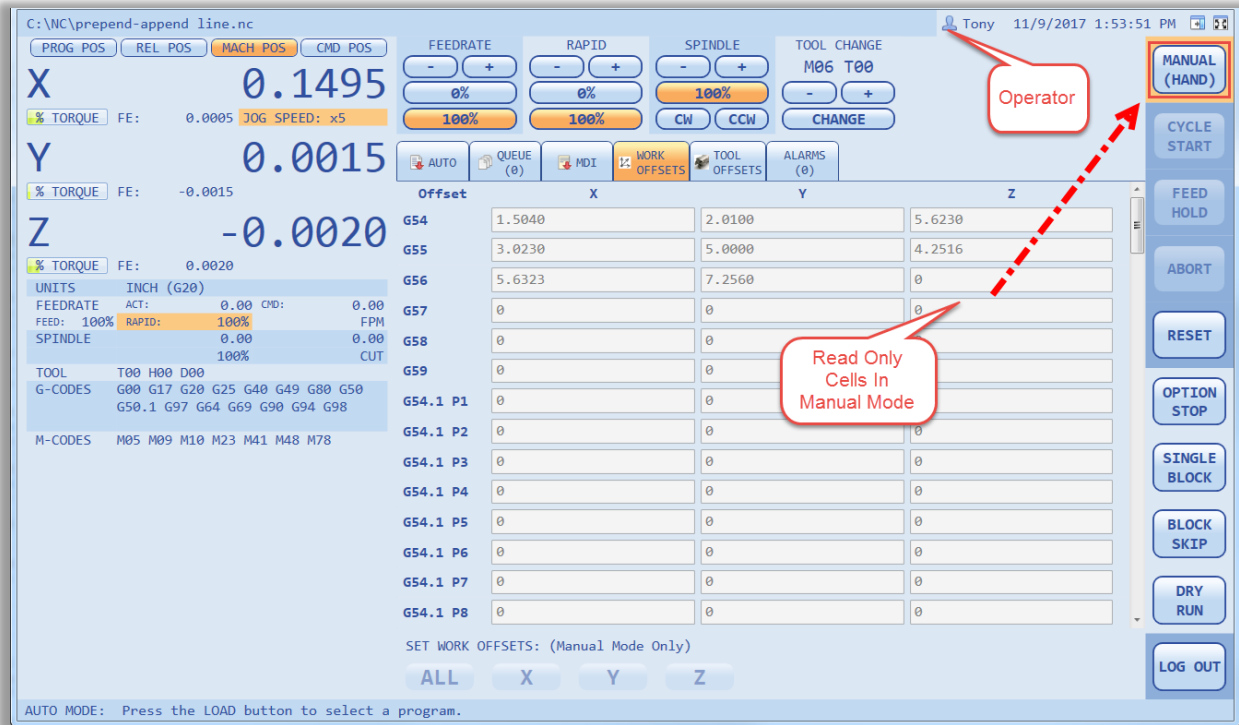


Operator Permissions:

This member provides handful of options to define an operator access level to different Power PMAC NC16 features or sections such as “Edit Work Offsets” and “Edit Tool Offsets”. Following figure, demonstrates all available options for this member:

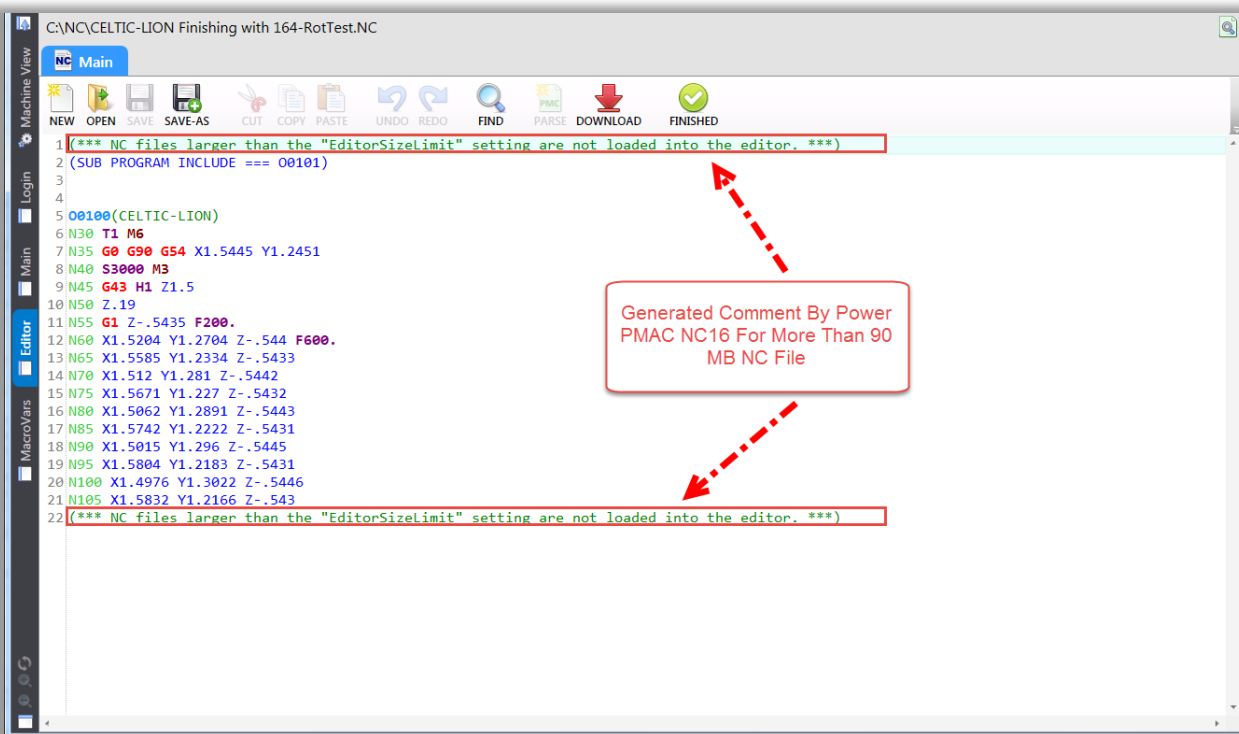


For example, if it is desired to restrict operators from setting any work offsets in a manual mode of Power PMAC NC16, simply deactivate the “Edit Work Offsets”. Figure below, shows that setting a work offset in manual mode as an operator is not permitted by Power PMAC NC16:



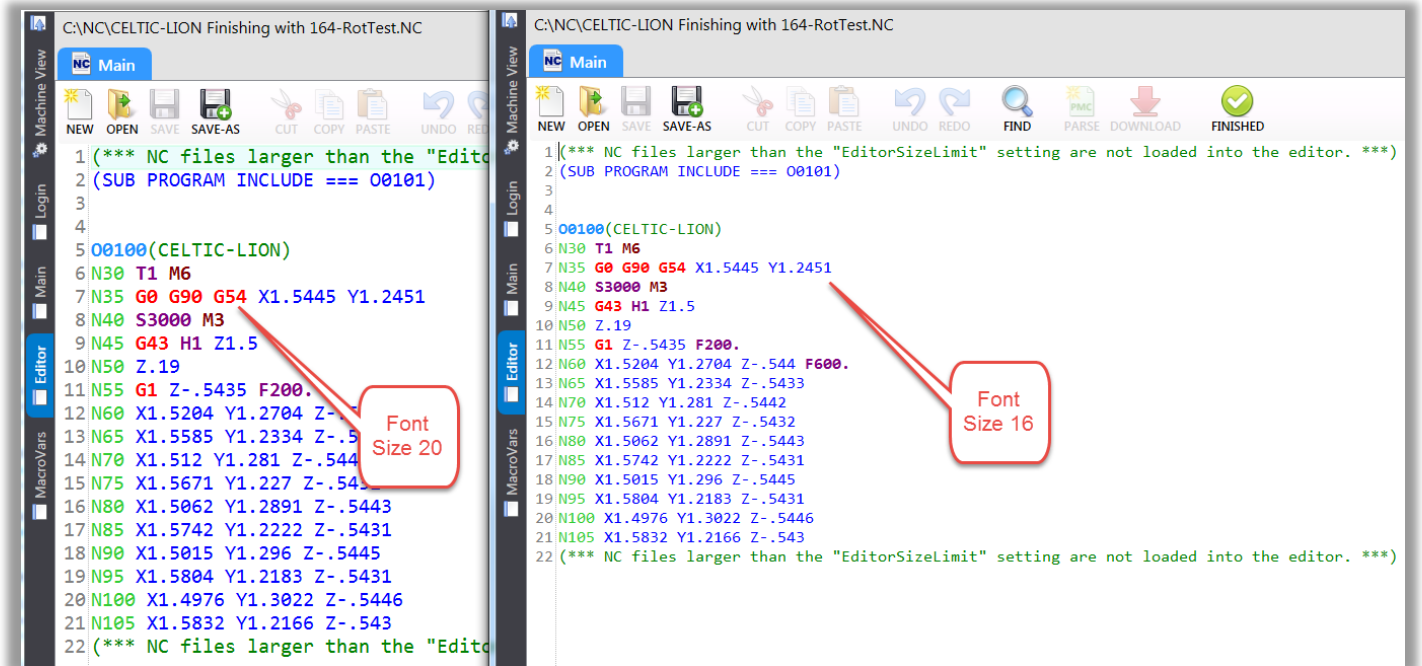
Editor Size Limit:

This member defines the size limit of the editor in Power PMAC NC16; it covers the range of 1 to 90 MB. Larger NC files will be still supported but editor will only display the first few lines. Following figure shows a comment inserted by Power PMAC NC16 when a large file (more than 90 MB) is used:



Editor Font Size:

This member defines editor font size according to its value. If it is desired to use a larger value to ease code readability, simply replace its default value (16) with a larger one. Following figure shows applied changes:



Jog Speeds:

Power PMAC NC16, provides five configurable jog speeds which they can be defined in user unit. Changing the value of any of these five options will immediately take effect.

Max Feedrate:

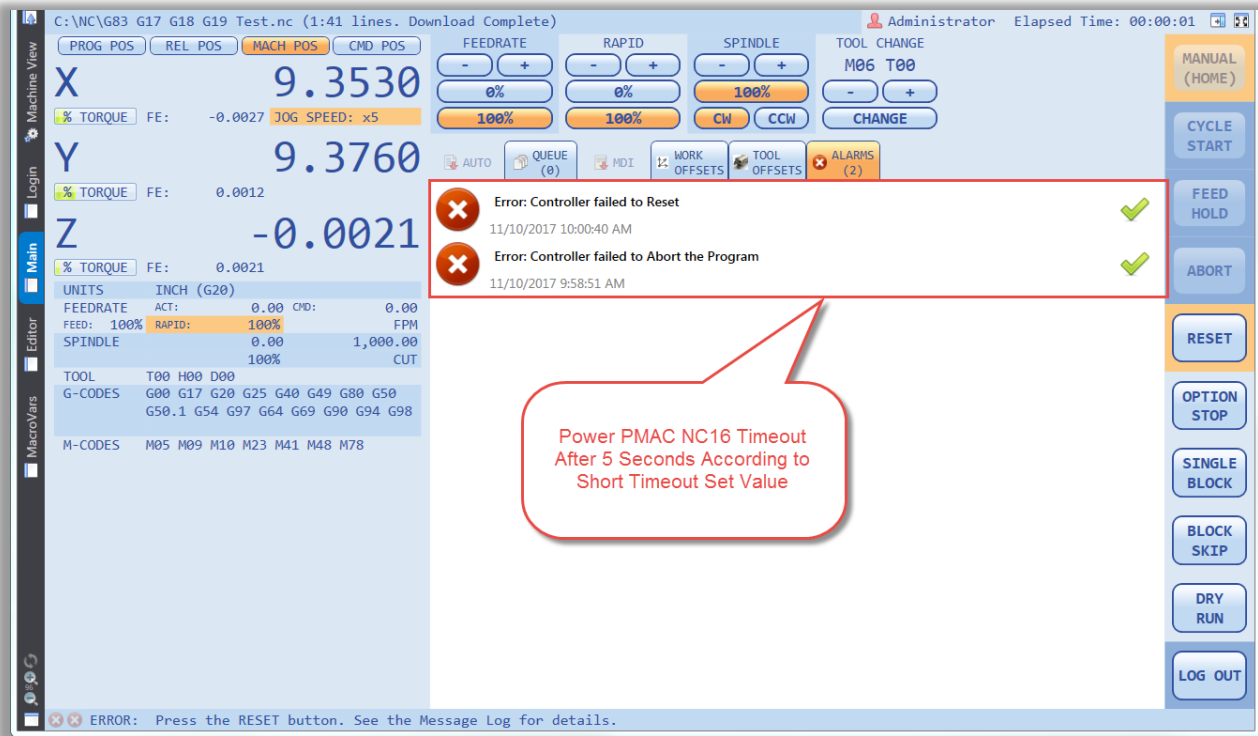
This member defines the maximum allowable feedrate value for a machine. Changing the value of this member will immediately take effect. If the commanded feedrate in a NC program is more than the value of this member, controller adjusts the feedrate according to this member's value.

Short Timeout:

The value of this member (in seconds) defines a timeout value for "Reset" and "Abort". This means, Power PMAC NC16 after setting the command register for reset or abort, waits according to this member's value to receive acknowledgment from the controller. If this acknowledgment is not received within this period of time, Power PMAC NC16 will send an alarm that reset or abort has not been successful.



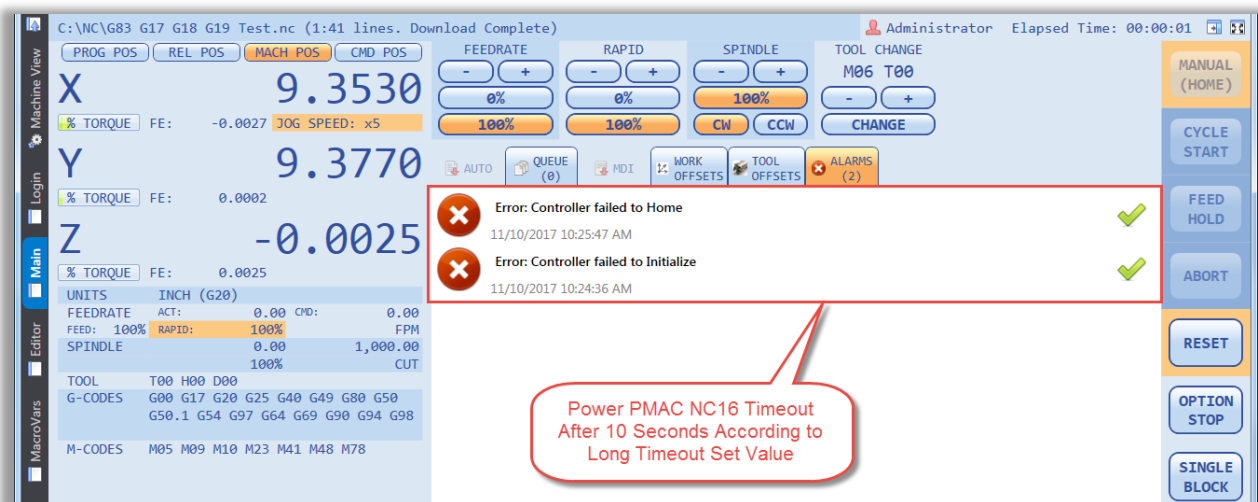
Adjust the value of this member according to machine reset or abort time in order to not receive a false message caused by process taking more time than set value. Such time outs are shown below as references:



Long Timeout:

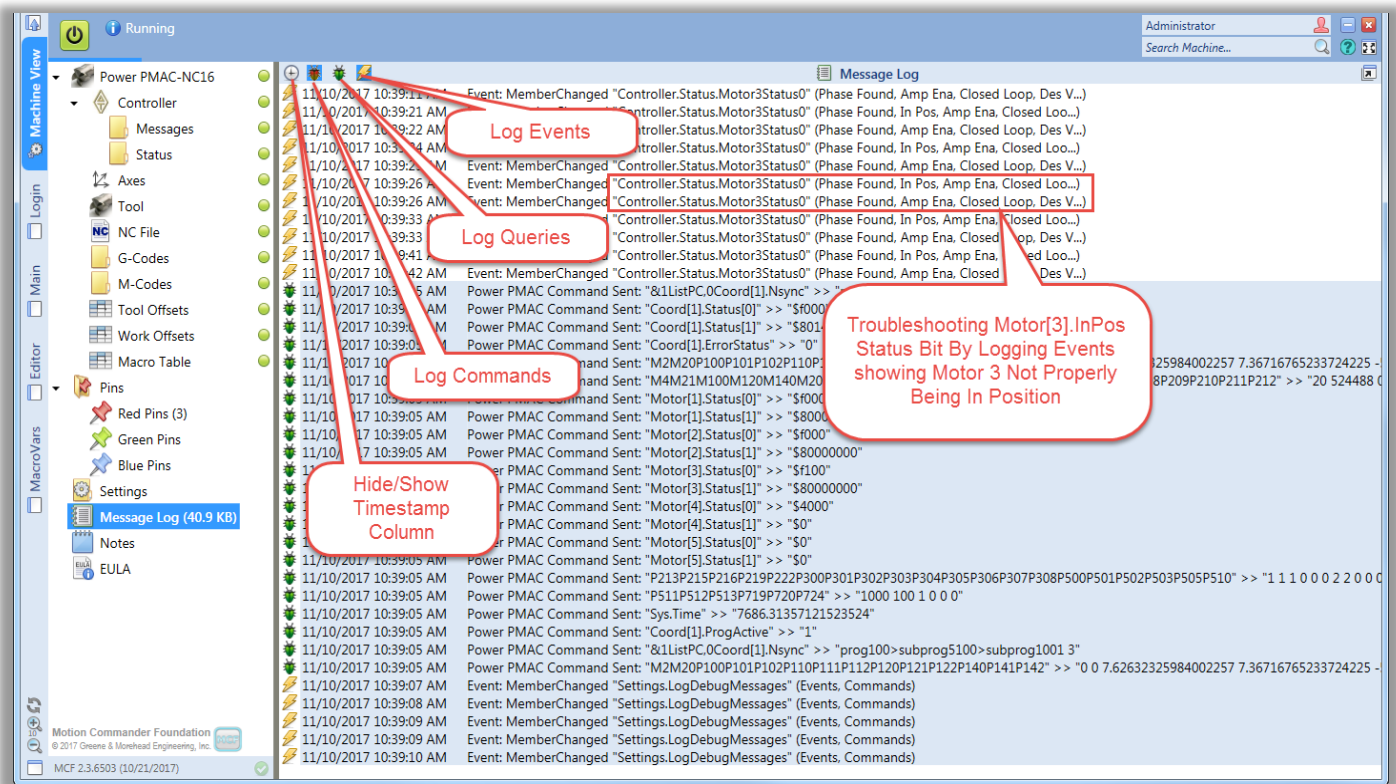
The value of this member (in seconds) defines a timeout value for “Homing” and “Initialization”. This means, Power PMAC NC16 after setting the command register for homing or initialization, waits according to this member’s value to receive acknowledgment from the controller. If this acknowledgment is not received within this period of time, Power PMAC NC16 will send an alarm that homing or initialization has not been successful.

💡 Adjust the value of this member according to machine homing or initialization routine time in order to not receive a false message caused by process taking more time than set value. Such time outs are shown below as references:



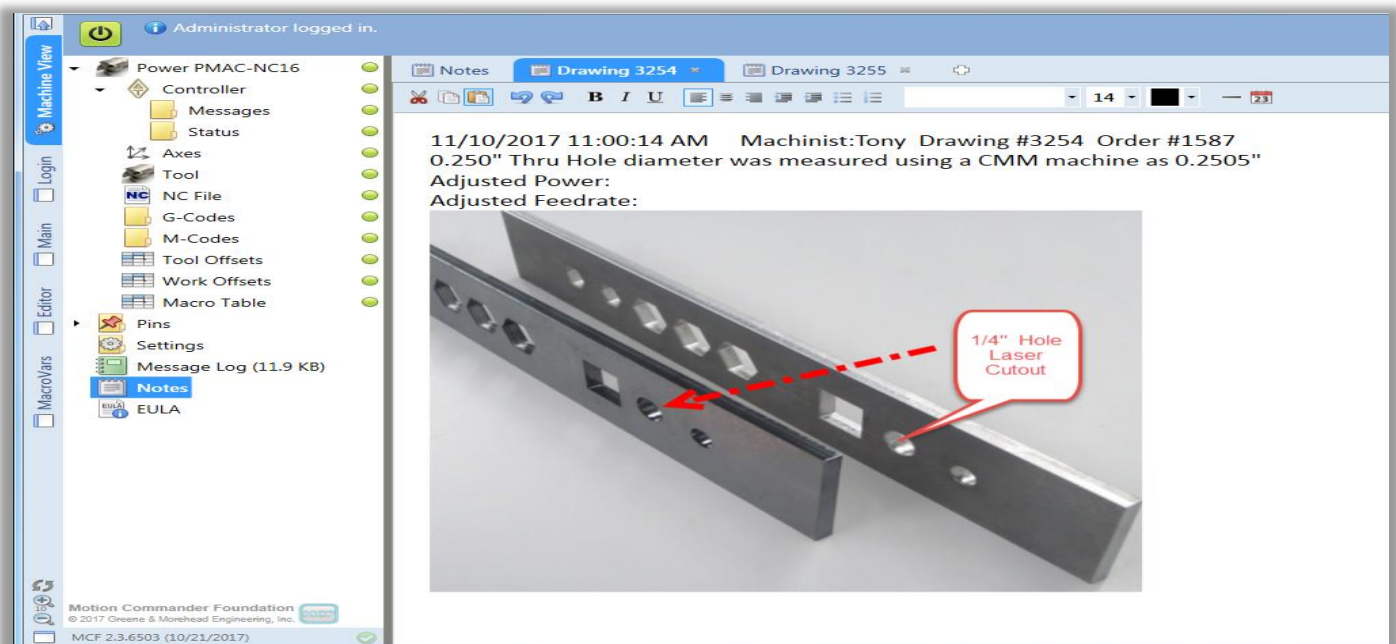
4) Message Log

This power full tool, logs Events, Commands, and Queries based on their activation. Therefore, technicians will be able to easily identify problems and resolve them.



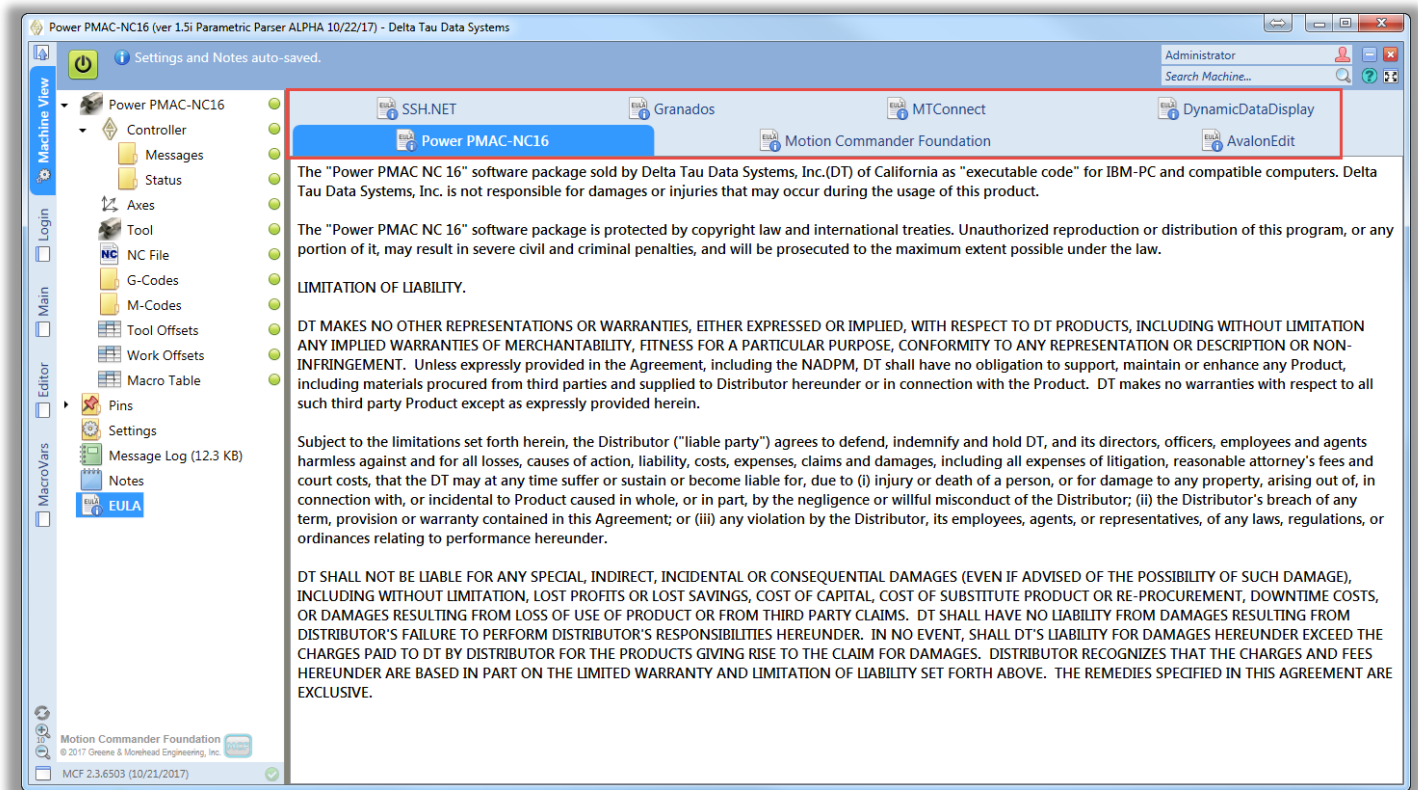
5) Notes

"Notes" can be used for punch lists, milestones, production records, and etc. Each document will be automatically saved.



6) EULA

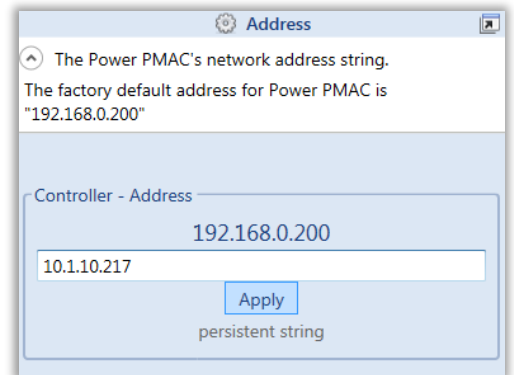
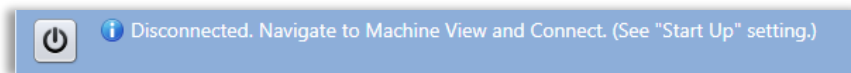
This section contains all legal agreements in regard of using Power PMAC NC16. Each agreement can be selected from the upper section as shown below:



Go Online

Start the Power PMAC-NC 16 program, log in as the Administrator, and navigate to Machine View via the tab along the left edge of the window.

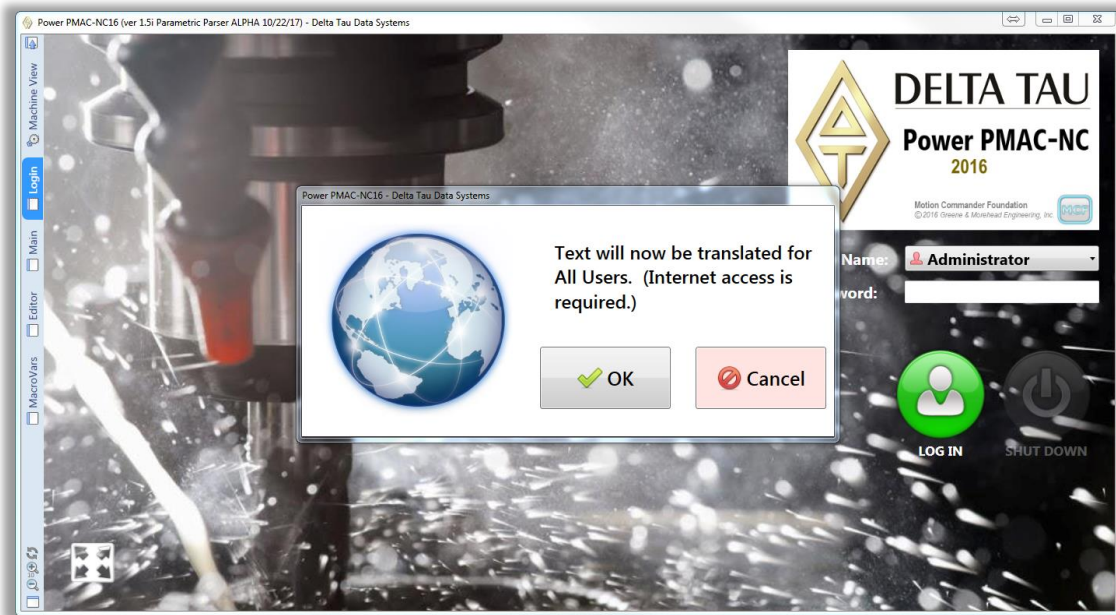
Select the **Controller** object to specify the IP address of Power PMAC, then click the "Go Online" button to open communication.



Foreign Language Support

The Power PMAC-NC 16 program includes a sophisticated foreign language translation system. Each user's language is specified in his or her login profile. When a foreign language user logs in, Machine View text and selected UI Page text is displayed in the user's language. The "Language Support" setting provides runtime control of the language translation system.

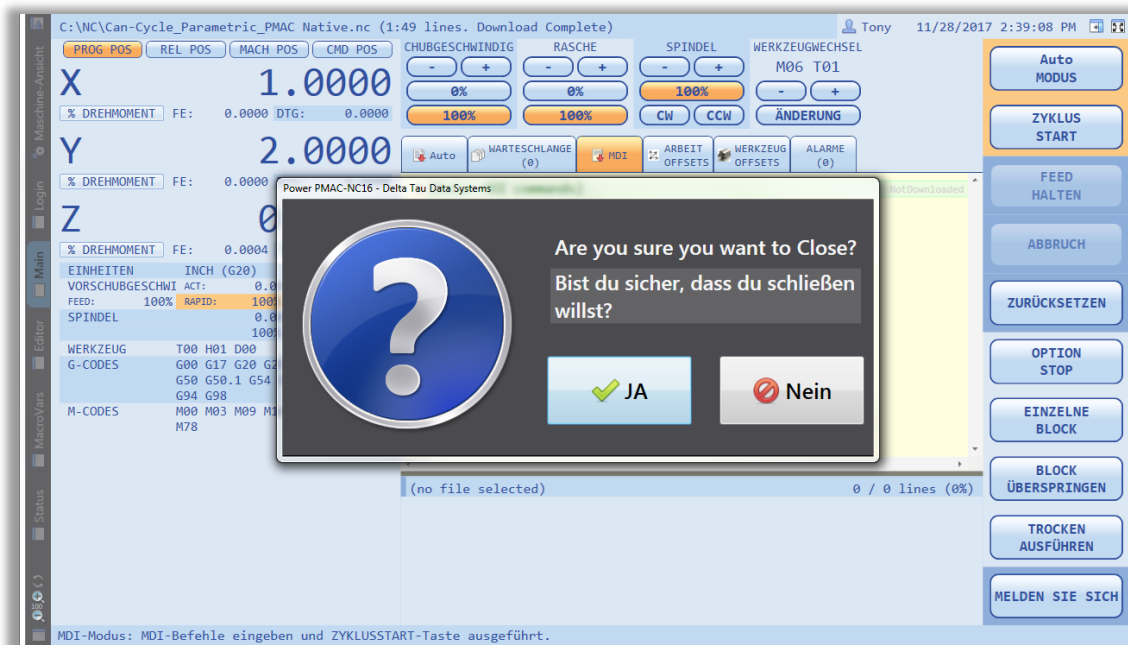
Initial translations are obtained from the *Microsoft Translator* web service. Therefore, an internet connection will be required the first time that the foreign language user logs in. If internet connection is found, Power PMAC NC16 will show a following message to confirm for translation at its startup:



 If Power PMAC NC16 is unable to find internet connection, it will continue by loading its default language (English.)

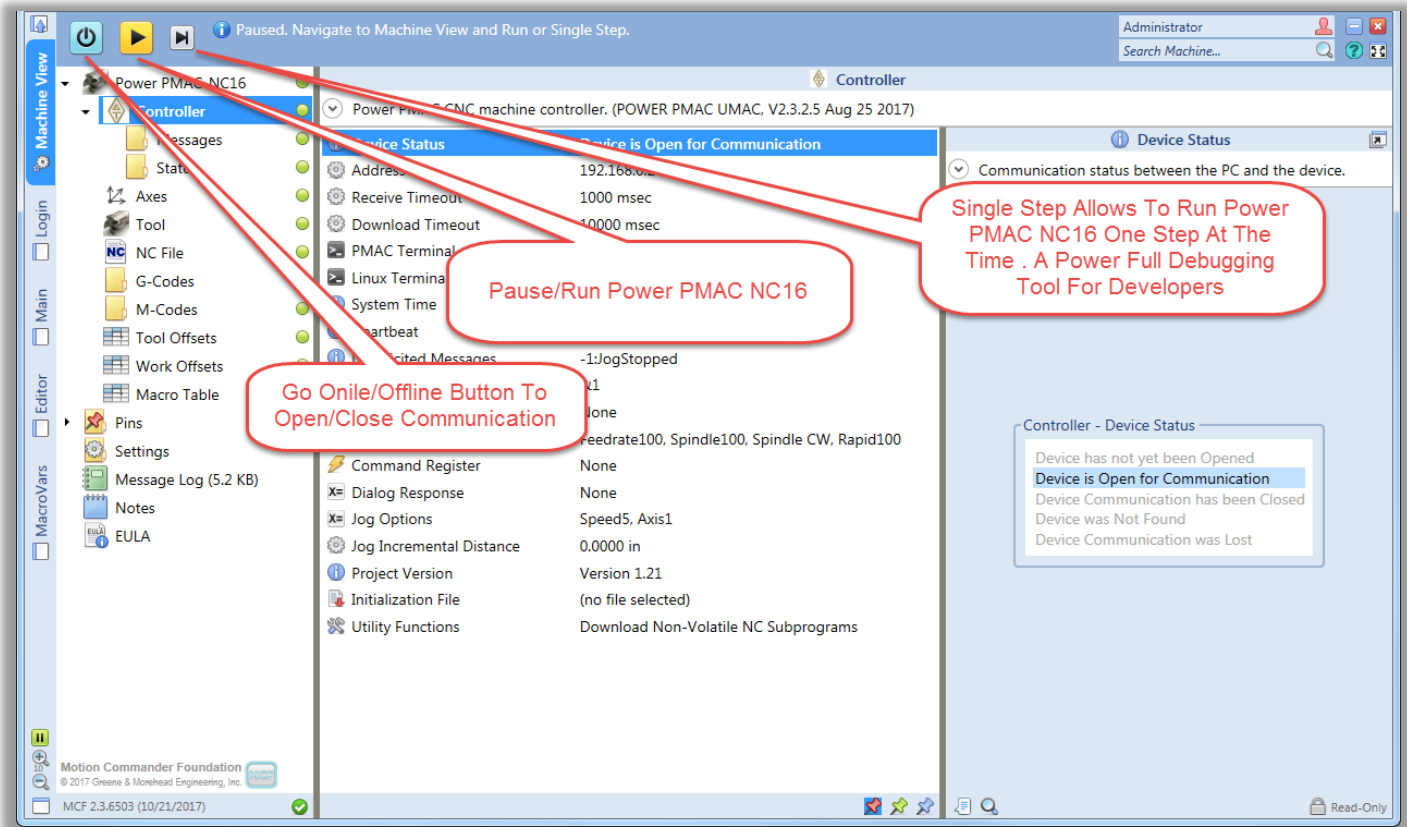
The translated text will be saved to a file named "*Languages\PowerPmacNC_Language_xx.txt*" where xx is one of 38 language codes. These files store native language strings and their foreign language translations, enabling the foreign language text to be edited by a human translator.

English: Are you sure you want to Close?
 German: Bist du sicher, dass du schließen willst?



Ctrl And Shift Keys

💡 If the **Ctrl** key is pressed while the **PowerPMACNC16.exe** is clicked then the user will be logged in but communications will not be opened. This feature is useful for working off line without a controller connected. Such unique feature activates two buttons next to the “Go Online/Offline” button as follow:



If the **Shift** key is pressed while the program is closed then the program will close immediately without the "Are you sure" dialog box or the splash screen. This feature saves time during development.

NC Files

The Power PMAC-NC 16 program parses the selected NC file and generates a temporary file named "NcProgram.pmc" for downloading to the controller. All NC programs including main and subroutines are enclosed in "open subprog n/close" statements for loading into the PMAC program buffer. A bootloader is used regardless of program executed in MDI or AUTO mode to "call" subroutines (subprograms). For automation purposes, starting program number can be set and transferred to boot loader. Multi main (including M30s) and subroutine(including M99s) programs can be used in a single file .Any existing block numbers are stripped and a block number is prepended to each line for execution monitoring as shown in this example.

The NC file

```
1 G21
2 G00 G90 G17 G40 G49 G80
3 M06 T1
4 X0 Y0
5 G43 H1 Z0.5
6 G01 F60
7 X1
8 G91
9 Y1
10 X1
11 G91 G28 Z0
12 G28 X1 Y2
13 M30
```

The Parsed NC file

```
NcProgram.pmc
1 open subprog 5000 // 00000
2 P724=0
3 N1 G21
4 N2 G00G90G17G40G49G80
5 N3 T1M06
6 N4 X0Y0
7 N5 G120H1G43Z0.5
8 N6 G01F60P50==60
9 N7 X1
10 N8 G91
11 N9 Y1
12 N10 X1
13 N11 G91G28Z0
14 N12 G28X1Y2
15 N13 M30
16 close
17
18
```

The NC File Parser

- Wraps the NC in "open prog n/close" for downloading
- Prepends block numbers for execution tracking
- Strips comments (NC-style and C-style) and normalizes line endings
- Detects volatile subprogram calls and includes them in the download
- Tracks volatile subprogram file time-stamps and re-downloads if modified
- Detects nonvolatile subprogram calls and uploads them from the controller
- Throws an error if any called subprogram is not found
- Throws an error if duplicated subprogram is found
- Ability to call subprograms that are included in a NC file and those located in a designated folder
- Detects native PMAC commands and expressions and passes them unmodified
- Detects and differentiate PMAC native variables , #define variables , and Macro local/global variables
- Tracks modal G and M-codes and D/F/H/S/T values for Mid-Program Starts
- Support multiple main files (multiple M30s and M99s) in a single file
- Support Macros and parametric programming
- Supports the Block Skip option
- Supports Fixed Cycles
- Supports Line and File Prepend and Append features
- Supports defined range of allowable G/M Codes
- Supports No-Lookahead suppression in conjunction with Macro programming
- Supports #define and #include style parameter aliasing and substitution
- Flexibility of assigning different subprograms' base numbers for MDI and AUTO modes

- Flexibility of assigning different ranges for volatile and non-volatile subprograms
- Flexibility of changing local variable stack offset and maximum number of jump labels in subprograms
- Ability to parse G09 and F-Code at the beginning or end of line based on users' applications
- Capable of finding subprograms based on their assigned numbers or provided file path
- Ability to read Macro assignments from a text file and assign users' define Power PMAC variables
- Capability of using M/T codes alone within any Can Cycle
- Capability of using PMAC native variables for axis positioning within any Can Cycle
- Ability of differentiating planes(G17-G18-G19) for Can Cycles to provide more machine flexibility
- Capability of switching between G90/G91 and G98/G99 within any Can Cycle
- Ability to detect "GoTo" commands with missing jump labels and creating an alarm

NC File Custom Pre-Parser

Power PMAC NC16 includes a user customizable pre-parser. This feature allows the machine builder to add custom logic to the parser. If the custom pre-parser is enabled it will call the custom pre-parser method for every NC line and evaluate it before it is downloaded to the PMAC control. The Power PMAC NC 16 SDK (Software Development Kit) is required to make changes to the pre-parser.

NC File Configuration

The application's "PowerPmacNC.ini" configuration file includes a section for NC files. These are the default values:

```
[NC Files]
CoordinateSystem=1
SubprogramFolder="C:\NC"
; NC programs are parsed and downloaded to PMAC subprog buffer ('O' program number + SubprogramBaseAddress)
SubprogramBaseAddress=5000
SubprogramBaseAddressMDI=6000
; "Volatile" NC programs are downloaded to PMAC along with the main NC program.
VolatileSubprogramMin=0
VolatileSubprogramMax=899
; "Non-volatile" NC programs are downloaded to PMAC by a utility and saved.
; To disallow non-volatile subprograms, set range to (0,-1)
NonvolatileSubprogramMin=900
NonvolatileSubprogramMax=999
; Optional "open subprog" parameters (buffer number or *, local variable stack offset, max jump labels).
;SubprogOpenParams=5000,256,1024
;SubprogOpenParams=*,256,1024
```

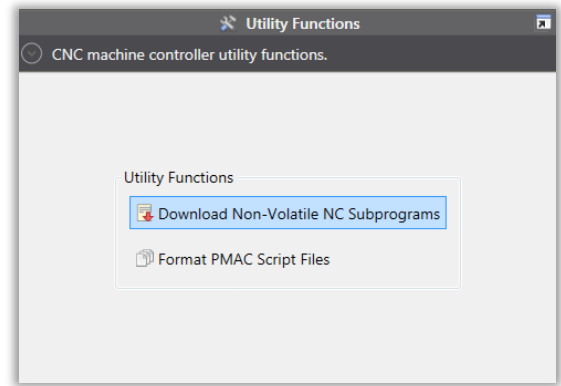
Subprograms

Subprograms can be called by 'O' program number or by path as shown in this example. Subprograms may call other subprograms with the one restriction that *nonvolatile* subprograms may only call other *nonvolatile* subprograms.

"Volatile" subprograms exist in NC files and are downloaded along with the main program. "Nonvolatile" subprograms exist on the controller and must be downloaded via a utility included in the Power PMAC-NC 16 program and saved.

(Main Program) ... M98 P0101 M98 P0002 L5 // repeat count M98 (C:\myfolder\myname.nc) L3 ... M30 O0101 (subprogram #101) ... M99 O0102 (subprogram #102) ... M99	(C:\NC\O0103.nc) O0103 (subprogram #103) ... M99	(Nonvolatile Subprogram File) O0001 (subprogram #001) ... M98 P0002 ... M99
	(C:\myfolder\myname.nc) O0104 (subprogram #104) ... M99	O0002 (subprogram #002) ... M99

"Volatile" subprograms are always downloaded together with the main program via a temporary file named "NcProgram.pmc". "Non-Volatile" subprograms are downloaded via a utility included in the Power PMAC-NC 16 program. An NC file may contain more than one nonvolatile subprogram. The *Power PMAC IDE* must be used to SAVE after nonvolatile subprograms are downloaded. Volatile and nonvolatile subprogram 'O' numbers must be within the ranges specified in the configuration file. These ranges may be set to (0, -1) to disallow subprograms of either type.



Volatile subprograms called by path may reside in any folder under any filename, and their 'O' program numbers must be specified as the first line of the NC program. Volatile subprograms called by 'O' program number may reside in the main program's NC file or in the Subprogram Folder specified in the configuration file. Volatile subprogram NC files in the Subprogram Folder must be named "Onnnn.nc" where "nnnn" is the 'O' program number.



Program numbers must be in the NonvolatileSubprogram Min/Max range.

All subprograms will be loaded into PMAC buffers at the specified Subprogram Base Address plus 'O' program number. For example, if the subprogram base address is 5000 and called subprogram is O0200, buffer 5200 (subprog 5200) will be assigned to this subprogram.



After downloading a program, a "NonVolatileSubs.pmc" will be created in "TempNC" folder. If it is desired, this file can be renamed and required to be imported under the "Motion Programs" folder as part of a project. "Download" and "Save" the project in order to store subprograms in controller memory.



Following features are not allowed in non-volatile subprograms:

- "M98(path)" is not allowed in non-volatile subprograms
- "({})" Sticky Comments are not supported in nonvolatile subprograms

Native PMAC Commands and Expressions

The NC file parser detects native PMAC *command lines* and passes them unmodified. Tokens that identify a line as a native PMAC command are "if, else, while, do, goto" or the presence of an equals sign '=' or curly bracket '{}'. In addition, lines with a pipe '|' as the first character will be treated as native PMAC commands. The pipe will be stripped.

```

1 03000
2 (Performing a CW Circular Move Using PMAC Native Language)
3 #define R P10000
4 #define Angle P10001
5 #define Increment P10002
6 Angle = 360
7 R = 1
8 Increment = 0.1
9 G00 G90 G54 X0 Y0
10 G43 H1 Z0.5
11 G01 F15
12 Z-0.2
13 F30
14 D0
15 {
16 X[R*cos[Angle]] Y[R*sin[Angle]]
17 Angle = Angle - Increment
18 #100 = #100 + 1
19 }While(Angle !< 0)
20 G01 F30 Z0.5
21 G91 G28 Z0
22 M30

```

Native PMAC *expressions* can be enclosed in square brackets within a line of NC as shown in this example:

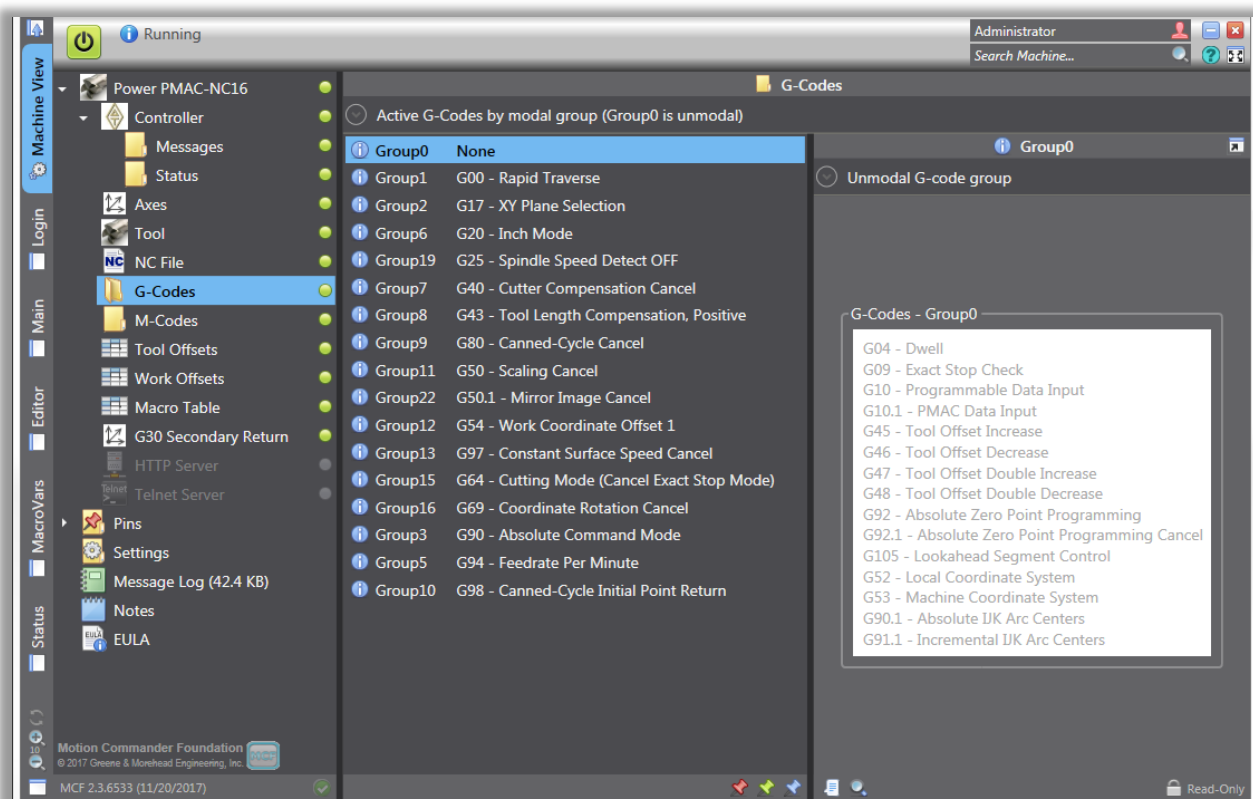
```

G54.1 P[-1+Q511*2] (1ST PART OFFSET)
X[cos(P30)] Y[P32]

```

G and M-Code Groups

The Power PMAC-NC 16 program includes support for the common G and M-code groups as shown in Machine View. Group 0 is "unmodal" and the rest are "modal" (the codes in the group are mutually exclusive). Group 6 switches the application between English (INCH) and Metric (MM) modes.



Each G and M-code group is linked to a PMAC variable specified in "DeviceMembers.xml".

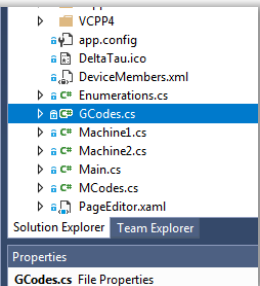
```
<!-- <Member Name="GCodes.Group0" Getter="P200" TimeBetweenUpdates="500" /> -->
<Member Name="GCodes.Group0" Getter="M200" TimeBetweenUpdates="500" />
<Member Name="GCodes.Group1" Getter="P201" TimeBetweenUpdates="500" />
<Member Name="GCodes.Group2" Getter="P202" TimeBetweenUpdates="500" />
<Member Name="GCodes.Group6" Getter="P206" TimeBetweenUpdates="500" />
```



Applied changes will become effective after the application is restarted.

In PPNC16 SDK version, refer to "GCodes.cs" and "MCodes.cs" in the *PowerPmacNC* project for the numerical values of each code. For example, if P206=1 then G21 of Group 6 is active.

```
/// <summary>G-code Group 6 enumeration.</summary>
public enum EGroup6
{
    /// <summary>Inch Mode</summary>
    [Description("G20 - Inch Mode")]
    G20,
    /// <summary>Metric Mode</summary>
    [Description("G21 - Metric Mode")]
    G21,
};
```



In PPNC16 SDK version, refer to "CustomCodeGroups.cs" in the *CustomExamples* project to learn how to add custom G and M-code groups to the Power PMAC-NC 16 program. Conversely, G and M-code groups that are not required by the application may be removed by specifying them in the application's "PowerPmacNC.ini" configuration file. (Group0, Group6, ProgramGroup and SubprogramGroup are used by the program and may not be removed.)

```
; Optional: List G and M-code group names that are NOT required by the application (separated by commas).
; Note: Group0, Group6, ProgramGroup and SubprogramGroup may not be removed.
; ExtraneousGroups=Group11,Group22,ThreadingGroup,GearRangeGroup,BAxisGroup
```

Mid-Program Start

The line of NC where the cursor is located will become the *Mid-Program Start line* when the "MID-PROG START" button is pressed. Pressing the button a second time clears the Mid-Program Start function.

When the CYCLE START button is pressed, the NC file will be scanned down to the Mid-Program Start line and a block of NC will be generated which sets all of the modal G and M-codes and D/F/H/S/T values encountered in the scan.

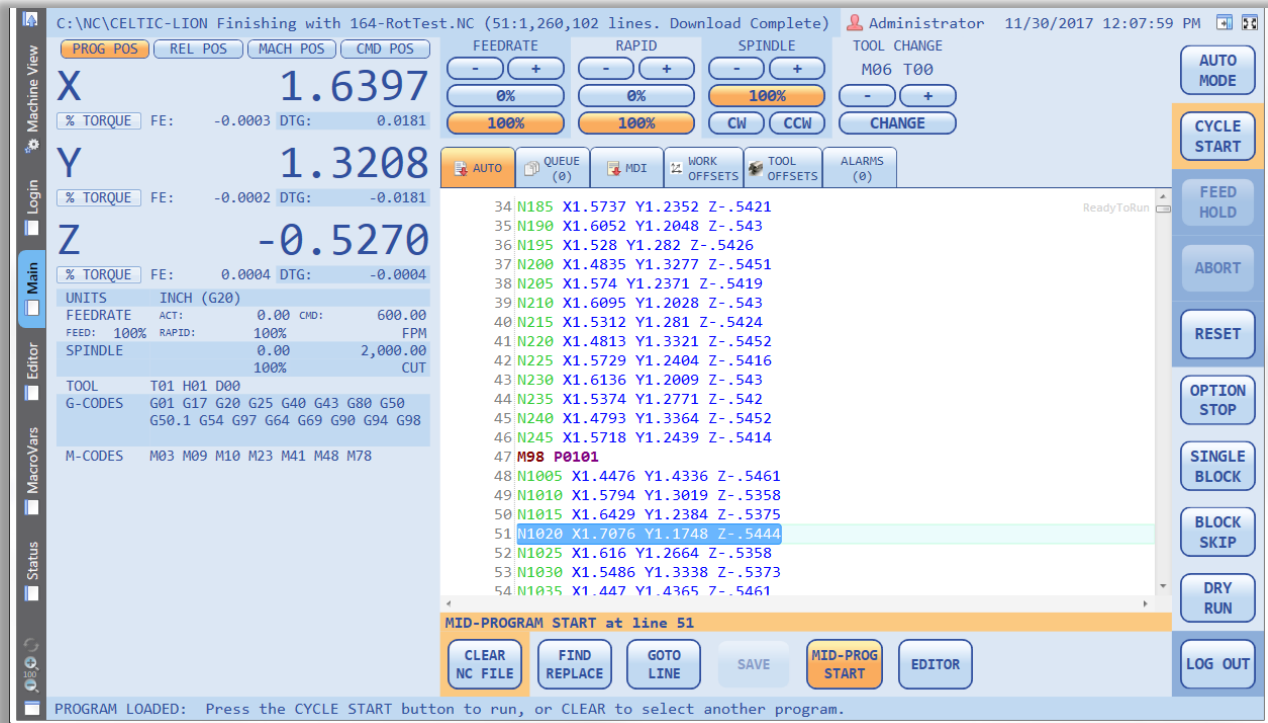
Mid-Program Starts from within subprograms are not supported at this time.

Mid-Program Starts from within Fixed Cycles ("Canned Cycles") are supported. Keep in mind that in **incremental** mode a machine is required to be positioned according to the latest program positions before the Mid-Prog start point.



Mid-Prog start does not support G90, G91, G99, and G98 used as a single line within Fixed Cycles. In order to assure proper NC program execution, using Mid-Prog start, make sure to use such G-Codes combine with programed axes positions.

Mid-Prog Start Example:

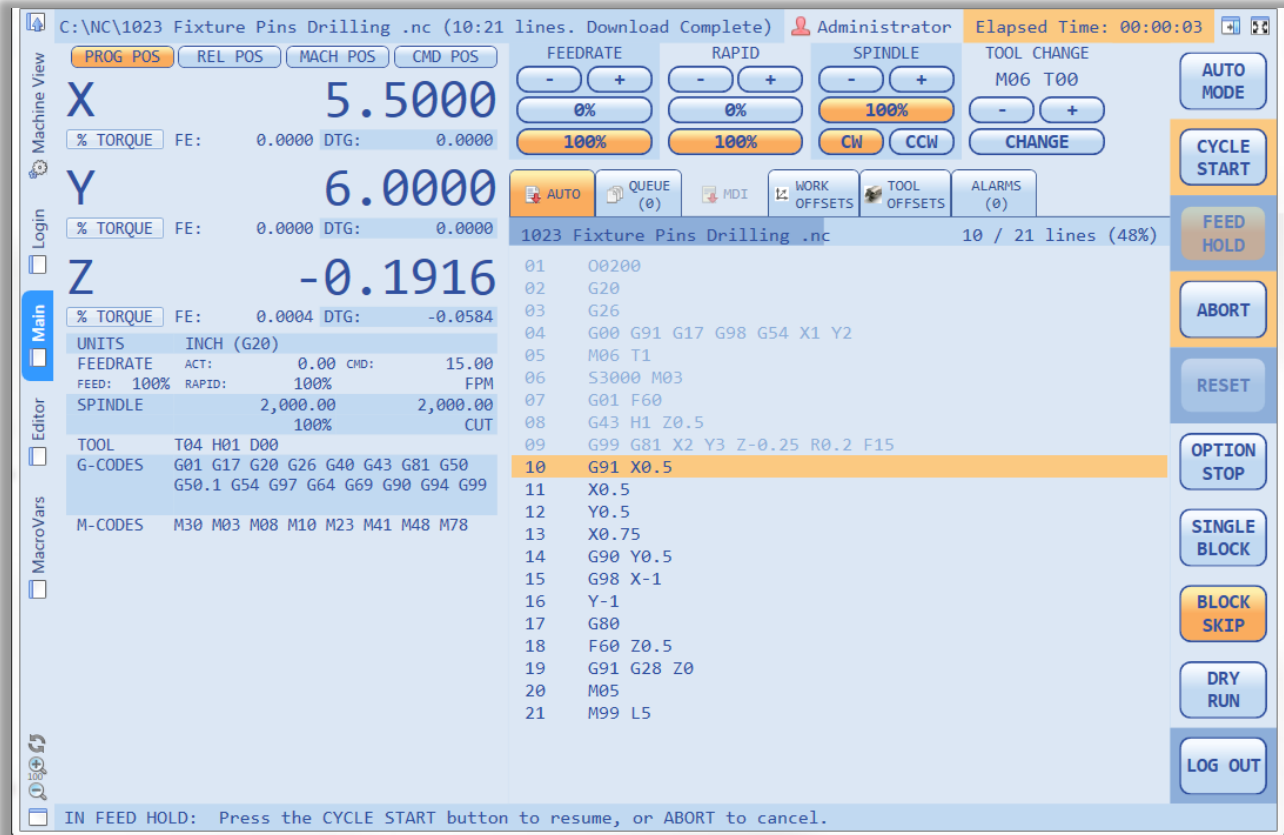


```
open subprog 5100 // 00100 (CELTIC-LION)
P724=0
G01G90G54G120H1G43S3000T1M06M03F600P50==600 // MID-PROGRAM START at line 51
N51 X1.7076Y1.1748Z-.5444
N52 X1.616Y1.2664Z-.5358
N53 X1.5486Y1.3338Z-.5373
N54 X1.447Y1.4365Z-.5461
N55 X1.5828Y1.3007Z-.5355
N56 X1.6459Y1.2377Z-.5375
N57 X1.7102Y1.1744Z-.5445
N58 X1.6175Y1.2671Z-.5356
N59 X1.5492Y1.3355Z-.5372
N60 X1.4464Y1.4394Z-.5461
N61 X1.5814Y1.3044Z-.5354
N62 X1.6441Y1.2417Z-.5371
N63 X1.7128Y1.1741Z-.5445
N64 X1.6207Y1.2662Z-.5355
N65 X1.5577Y1.3293Z-.5365
N66 X1.4459Y1.4422Z-.5461
N67 X1.5873Y1.3007Z-.535
...
M30
Close
```

Diagram illustrating the Mid-Program Start point:

- A red dashed arrow points from the line `G01G90G54G120H1G43S3000T1M06M03F600P50==600 // MID-PROGRAM START at line 51` to a box labeled "Chosen Start Point".
- A red dashed arrow points from the line `N51 X1.7076Y1.1748Z-.5444` to a box labeled "Codes Scanned And Inserted By Parser".

Canned-Cycle Mid-Prog Start Example:



```
open subprog 5200 // o0200
P724=0
G20G26G01G91G17G99G54G120H1G43S3000M03F60P50==60 // MID-PROGRAM START at line 10
N10 G91G81X0.5Z-0.25R0.2F15
N11 G81X0.5Z-0.25R0.2F15
N12 G81Y0.5Z-0.25R0.2F15
N13 G81X0.75Z-0.25R0.2F15
N14 G90G81Y0.5Z-0.25R0.2F15
N15 G98G81X-1Z-0.25R0.2F15
N16 G81Y-1Z-0.25R0.2F15
N17 G80
N18 F60P50==60Z0.5
N19 G91G28Z0
N20 M05
N21 M99L5;return
close
```

Modal Codes

Canned Cycle Mid-Prog Start Point

Fixed Cycles

Fixed Cycles ("Canned Cycles") supported include G73, G74, G76, and G81 through G89. G80 cancels the fixed cycle function, as shown in the following example:

The NC File:

```
G99 G73 X2 Y3 Z-0.25 R0.2 F15 S1500
G91 X0.5
/X0.5
//With Option 2
IF(P10000 == 1) {M09 M07}
IF(P10001 == 3){
X0.875
}
IF[#3 EQ #100] D01 X[#101] END1
IF[#5 EQ #102] D01
X[#103]
END1
M05
M06 T7
IF(P10003 == 1) M03
ELSE M04
Y0.5
X0.75
G90 Y0.5
G98 X-1
Y-1
G80
```


The Resulting Parser Output:

```
N9 G99G73X2Y3Z-0.25R0.2F15S1500
N10 G91G73X0.5Z-0.25R0.2F15S1500
N11 if(M5&$4==0){G73X0.5Z-0.25R0.2F15S1500}
IF(P10000 == 1) {M09 M07}
IF(P10001 == 3){
N15 G73X0.875Z-0.25R0.2F15S1500
}
N17 dwell0 if(L3==P5100){X(P5101)}
N18 dwell0 if(L5==P5102){
N19 dwell0 G73X(P5103)Z-0.25R0.2F15S1500
}
N21 M05
N22 T7M06
IF(P10003 == 1) M03
ELSE M04
N25 G73Y0.5Z-0.25R0.2F15S1500
N26 G73X0.75Z-0.25R0.2F15S1500
N27 G90G73Y0.5Z-0.25R0.2F15S1500
N28 G98G73X-1Z-0.25R0.2F15S1500
N29 G73Y-1Z-0.25R0.2F15S1500
N30 G80
```

As can it be seen in figures above, PPNC16 has a unique capability of differentiating Power PMAC native language, parametric programming language, and conventional NC program language within Fixed-Cycles. M, F, T, and S codes also can be used within Fixed-Cycles as long as they are not used with any programmed axis position on the same line.

Using M99 to Repeat the Main Program

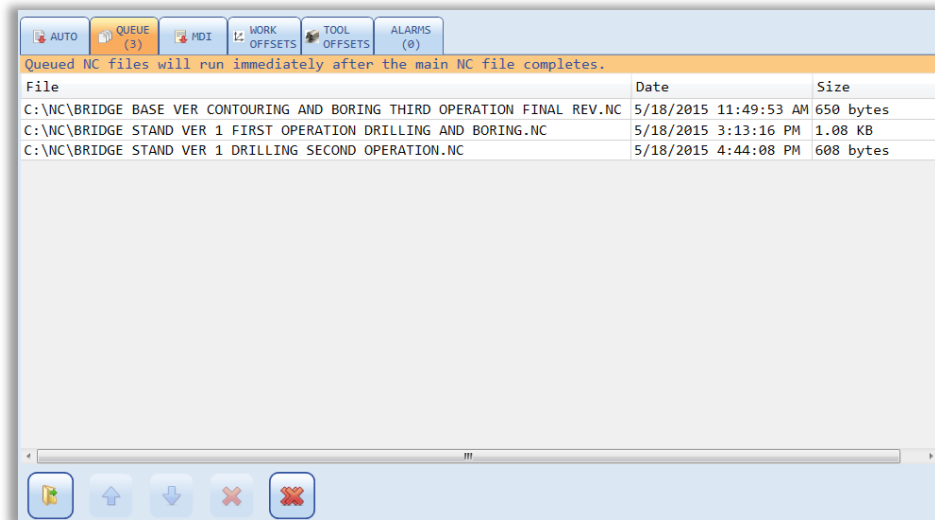
Power PMAC NC16 is capable of repeating a program for desired number of times. A repeat count may be specified using the 'L' parameter. "M99 L5" will cause the main NC program to be repeated five times.


 If M99 in a main program is used without 'L' parameter, the main NC program will be executed in an infinite loop until program is "Aborted".

The NC Program Queue

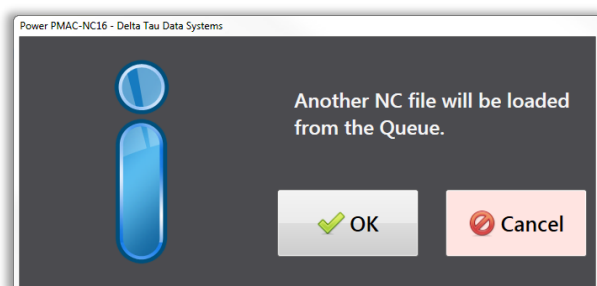
The first NC file loaded will always appear in the editor. Additional NC files will be loaded into the *Queue* as shown. The Queue includes tools for re-ordering and deleting files.

 Drag-and-drop from *Windows Explorer* may be used to load multiple NC files.



 Queued NC files will be downloaded and executed immediately after the initial program completes without the need for further CYCLE START commands.

If the loaded program in the editor is not needed anymore, simply select “Clear NC File”. Therefore, PPNC 16 will load the next program from the “Queue” by showing the following message:



NC File Comments

Comments (displayed in green in the editor) are stripped before the NC program is downloaded. NC-style () comments, C-style /**/ comments and CPP-style // comments are supported, as shown in the example.

C-style /**/ comments may not span multiple lines.

Subprogram paths are also surrounded by parentheses in M98 calls as shown in the example.

```

12 X34.924836 Z-7.966783
13 M98 P0109 // CPP style comment
14 X33.072664 /*C style comment*/ Z-7.281055
15 M98 P0109 (NC style comment)
16 X31.393792 Z-6.587248
17 M98 (C:\NC\00105.nc)
18 X29.726016 Z-5.826994

```

The NC Editor File Size Limitations

The NC editor has a 90 MB file size limitation. This can be adjusted in the Settings section of Machine View.

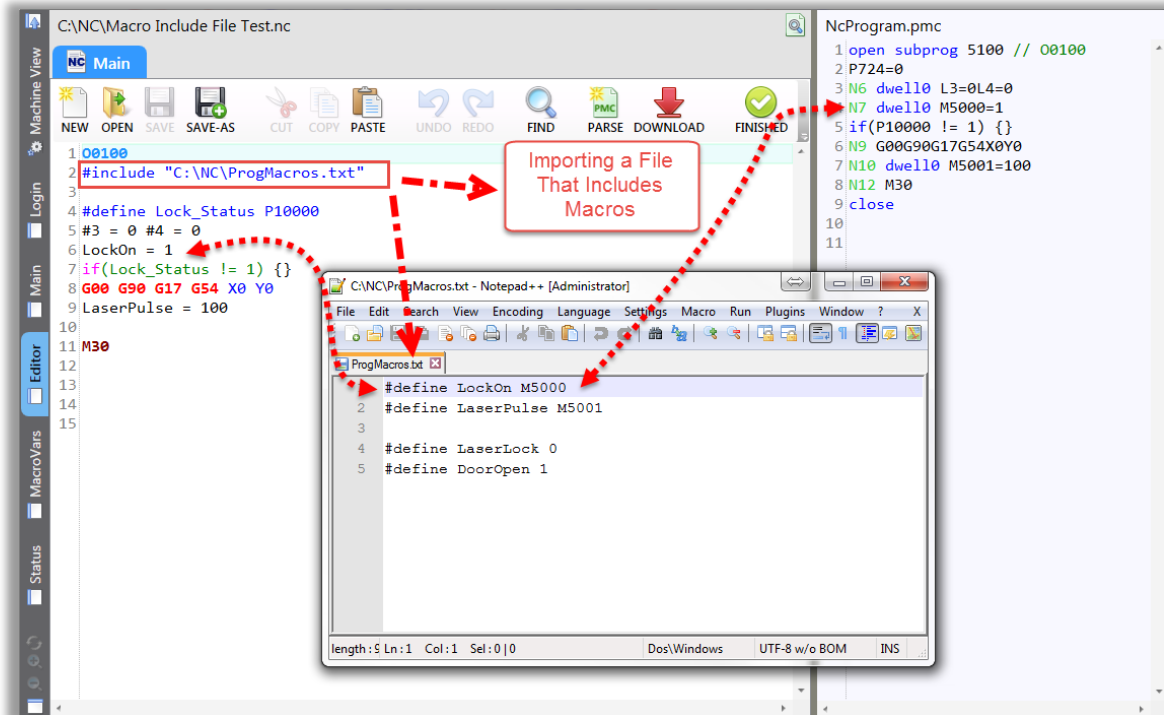
The actual NC part program file can be much larger than the editor maximum. The editor will load only the amount specified in the Editor Size limit. Also keep in mind the PMAC Program Buffer size must be set accordingly to

accommodate large part program files. This is done through the Power PMAC IDE. Also the timeout settings will need to be increased in the Settings section of Machine View to accommodate the long parse and download periods.

For very large part programs we suggest using the rotary buffer option available under the NC File section of the Machine View. This allows real-time streaming of the part program into the Power PMAC. The exact size limitation depends on various settings but part program files in the gigabytes have successfully been processed this way.

Macro Substitutions -#define and #include

PPNC16 supports macro substitutions when enabled in the .ini configuration file. Macro substitutions can be very powerful when coding NC part programs. In addition to embedded #define statements, the program can utilize the #include command directly in the NC part program to reference a file which includes the actual #define statements.



Customizing the Application

There are three ways that the Power PMAC-NC 16 program can be customized to suit a particular machine.

1. *Configure* the application by editing "PowerPmacNC.ini" and "Messages.xml".
2. *Extend* the application by creating external assemblies (plugin DLL's, PPNC16 SDK version only.)
3. *Modify* the application by editing its source code (PPNC16 SDK version only.)

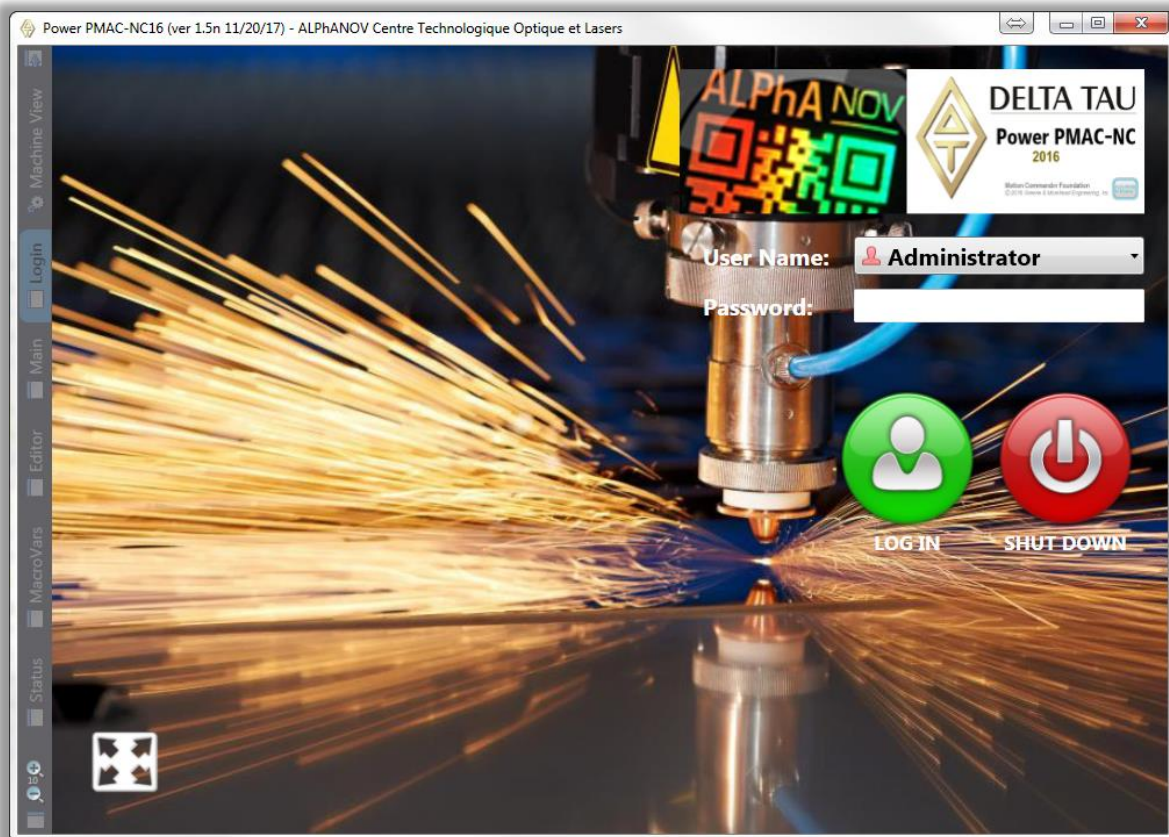


The configuration file and external assemblies are the preferred methods for customization. In this way, the main application can be upgraded without the need to merge source code changes back in. If it is required to modify the main application, it is recommended to clearly mark edited sections with block-start and block-end comments to make merges easier in the future.

Private Labeling

If it is desired to have a custom "log in" page, uncomment following lines in a "PowerPmacNC.ini" file to private-label the program and specify your own splash image and login screen background. Images should be PNG or JPEG format and must be located in the exe directory. The splash image should be approximately 500x300 pixels and the login image should be around 1000x700.

```
[Private Label]
CompanyName="ALPhANOV Centre Technologique Optique et Lasers"
SplashImage="LaserSplashImage.png"
LoginImage="LaserLoginImage.jpg"
```



Custom Messages and Dialog Boxes

Type 1 – Persistent, Bitwise Message Display

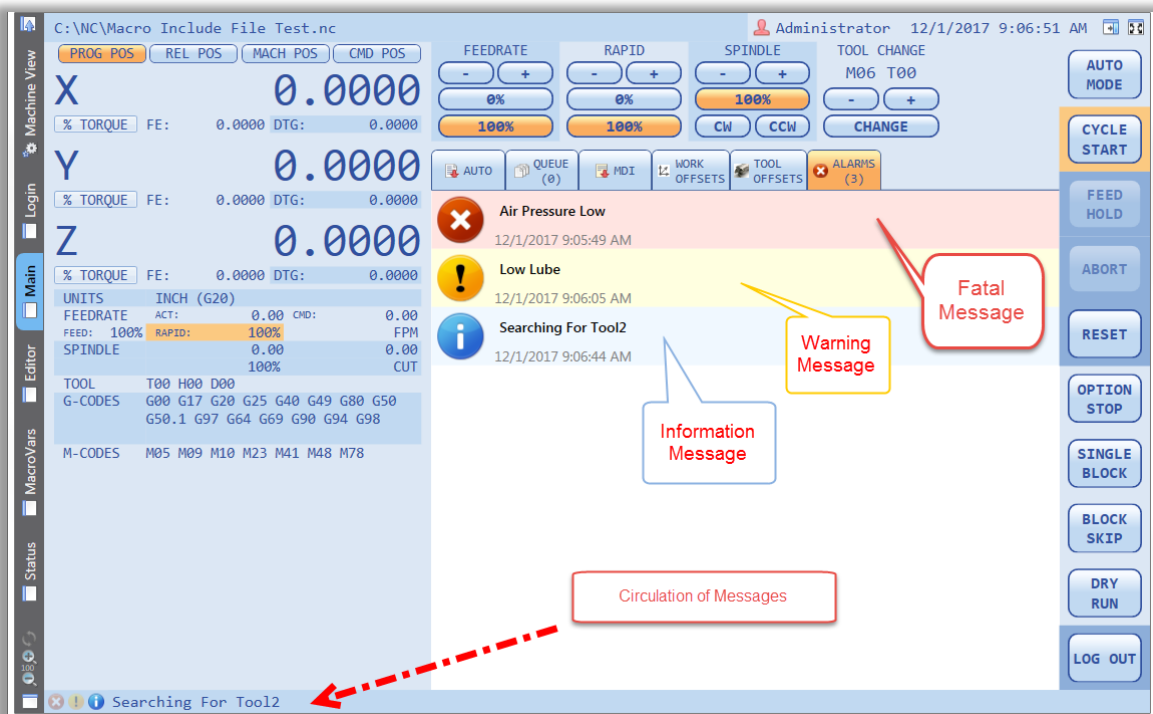
The Power PMAC-NC 16 program includes three bitwise message registers (Fatal, Warning and Information) that the controller can use to display messages to users. The message strings are specified in the "Messages.xml" file in the executable directory. A *Reference* copy of this file is included in the project for convenience.

```
<FatalMessages>
  <!-- These alarms activate on the rising edge of their bits and deactivate on the falling edge. -->
  <Message Bit="0" Description="Fatal Message 1">Fatal - Message 1.</Message>
  <Message Bit="1" Description="Fatal Message 2">Fatal - Message 2.</Message>
  <Message Bit="2" Description="Fatal Message 3">Fatal - Message 3.</Message>
```

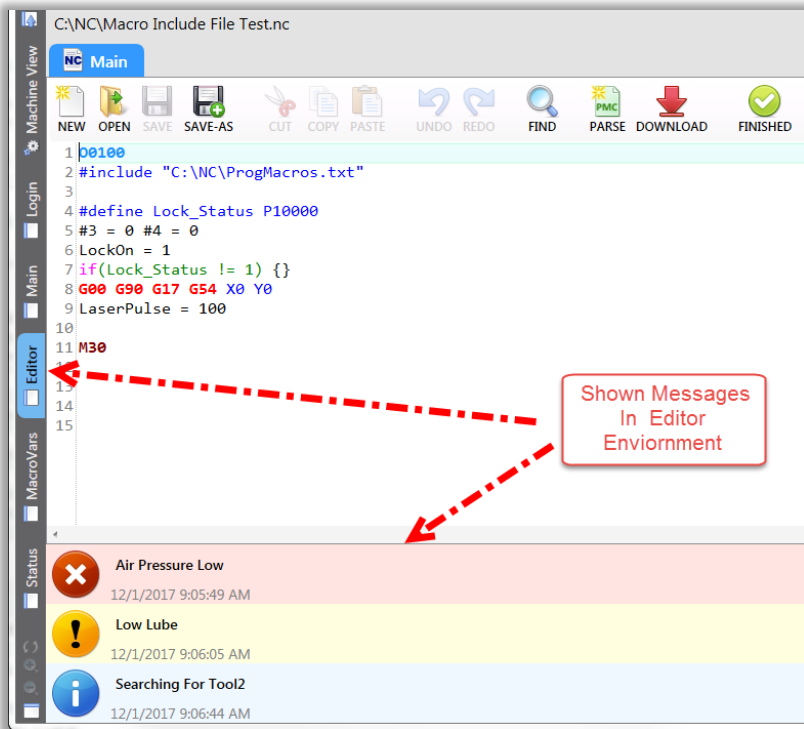
```
<WarningMessages>
  <!-- These alarms activate on the rising edge of their bits and deactivate on the falling edge. -->
  <Message Bit="0" Description="Warning Message 1">Warning - Message 1.</Message>
  <Message Bit="1" Description="Warning Message 2">Warning - Message 2.</Message>
  <Message Bit="2" Description="Warning Message 3">Warning - Message 3.</Message>
```

```
<InformationMessages>
  <!-- These alarms activate on the rising edge of their bits and deactivate on the falling edge. -->
  <Message Bit="0" Description="Information Message 1">Information - Message 1.</Message>
  <Message Bit="1" Description="Information Message 2">Information - Message 2.</Message>
  <Message Bit="2" Description="Information Message 3">Information - Message 3.</Message>
```

"Fatal" messages are marked with a red icon, while "Warning" and "Information" messages are marked with yellow or blue icons and are highlighted in the status bar. All messages upon their activation circulate in the status bar as it is shown below:



Active messages are also shown in the editor environment as it is shown below:



Each message register is linked to a PMAC variable specified in "DeviceMembers.xml".

```
<Member Name="Controller.Messages.FatalMessages" Getter="M100" TimeBetweenUpdates="500" />
<Member Name="Controller.Messages.WarningMessages" Getter="M120" TimeBetweenUpdates="500" />
<Member Name="Controller.Messages.InformationMessages" Getter="M140" TimeBetweenUpdates="500" />
```

Each variable has 31 bits in "ppnc_messages.pmh" as shown below:

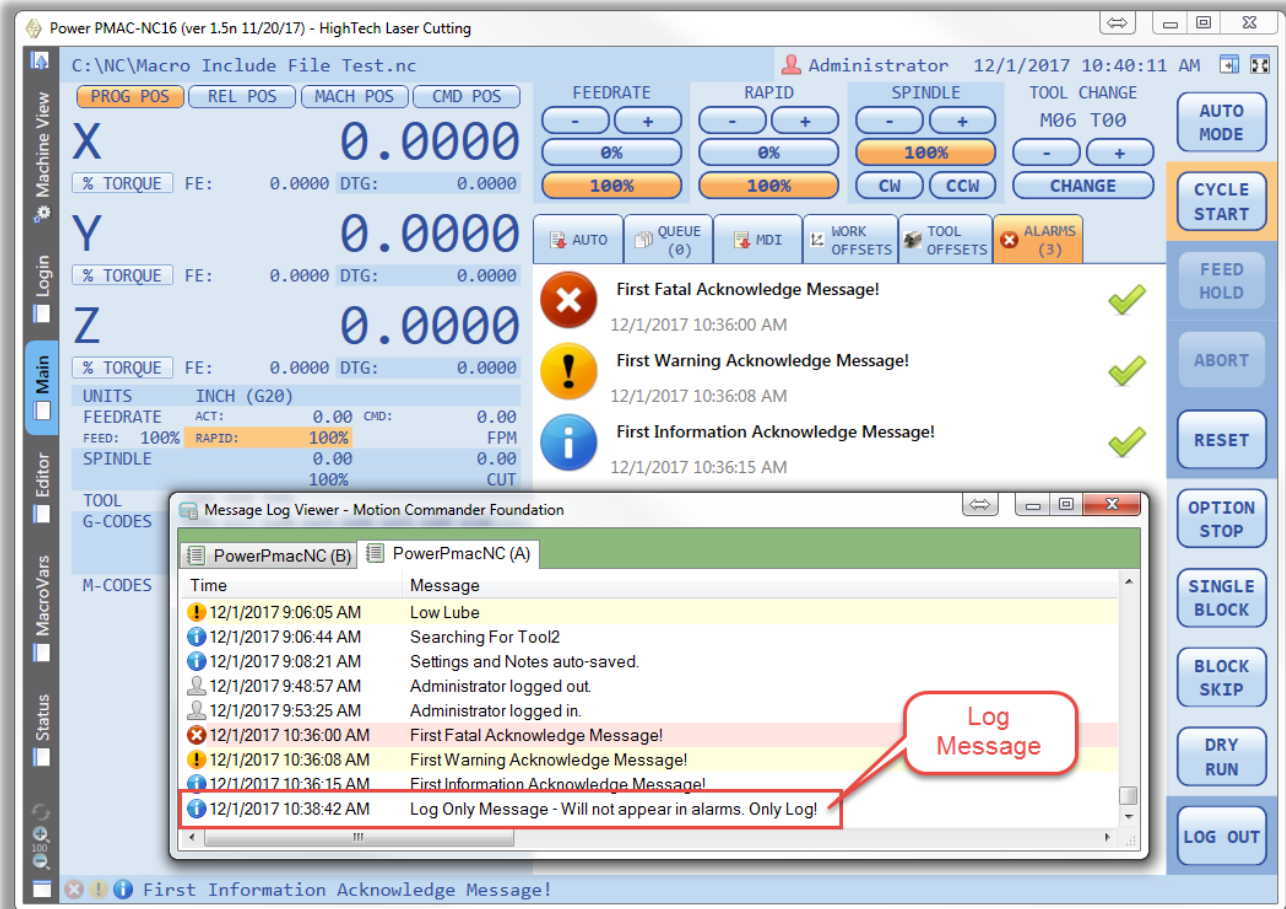
```
ppnc_messages.pmh
// *****
// PowerPAC NC Fatal Error Bits
#define MSG_Fatal_01 $00000001
#define MSG_Fatal_02 $00000002
#define MSG_Fatal_03 $00000004
#define MSG_Fatal_04 $00000008
#define MSG_Fatal_05 $00000010
#define MSG_Fatal_06 $00000020
#define MSG_Fatal_07 $00000040

<Messages>
  <FatalMessages>
    <!-- These alarms activate on the rising edge of their bits and deactivate on edge. -->
    <!--Message Bit="0" Description="Fatal Message 1">Fatal - Message 1.</Message>
    <Message Bit="0" Description="Air Pressure Low">Fatal - Message 1.</Message>
    <Message Bit="1" Description="Fatal Message 2">Fatal - Message 2.</Message>
    <Message Bit="2" Description="Fatal Message 3">Fatal - Message 3.</Message>
    <Message Bit="3" Description="Fatal Message 4">Fatal - Message 4.</Message>
    <Message Bit="4" Description="Fatal Message 5">Fatal - Message 5.</Message>
    <Message Bit="5" Description="Fatal Message 6">Fatal - Message 6.</Message>
    <Message Bit="6" Description="Fatal Message 7">Fatal - Message 7.</Message>
    <Message Bit="7" Description="Fatal Message 8">Fatal - Message 8.</Message>
```

Type 2 – Unsolicited “Send1” Acknowledge Message (Can display queried data within message)

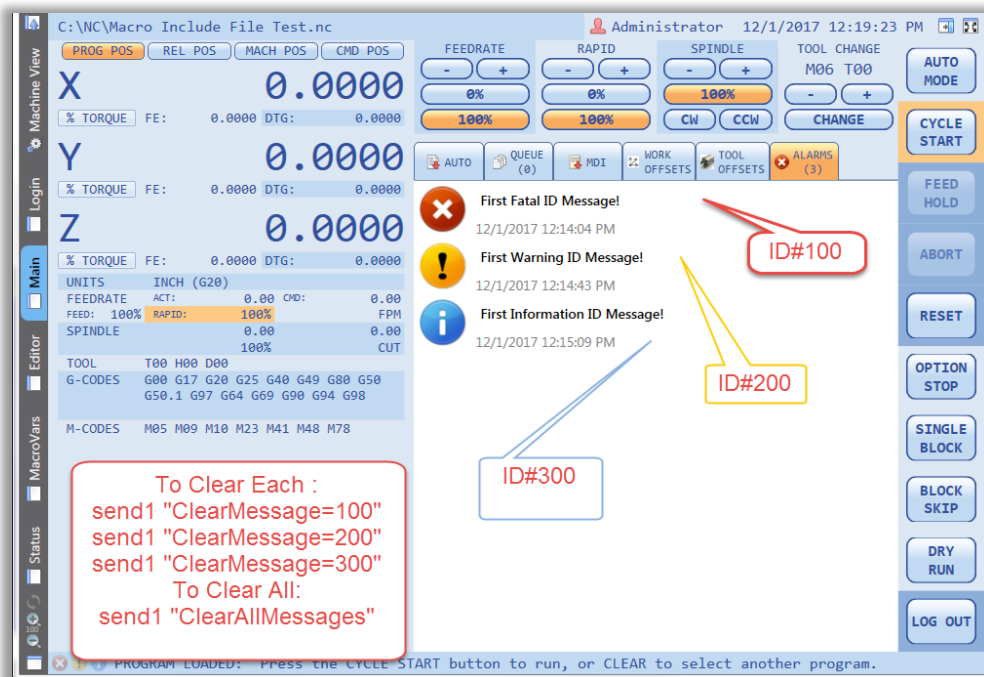
Type 2 messages are very simple to implement. All that is necessary is the “send1” unsolicited command structure. The four command structures are shown below:

- send1 "fatalmessage=First Fatal Acknowledge Message!"
- send1 "warningmessage=First Warning Acknowledge Message!"
- send1 "informationmessage=First Information Acknowledge Message!"
- send1 "logmessage=Log Only Message - Will not appear in alarms. Only Log!"



Type 3 – Unsolicited “Send1” Persistent Message with ID (Can display queried plus live data within message)

Type 3 message are similar to Type 2 messages but must be cleared programmatically via the ID tag sent with the message string. The command structures are shown below:



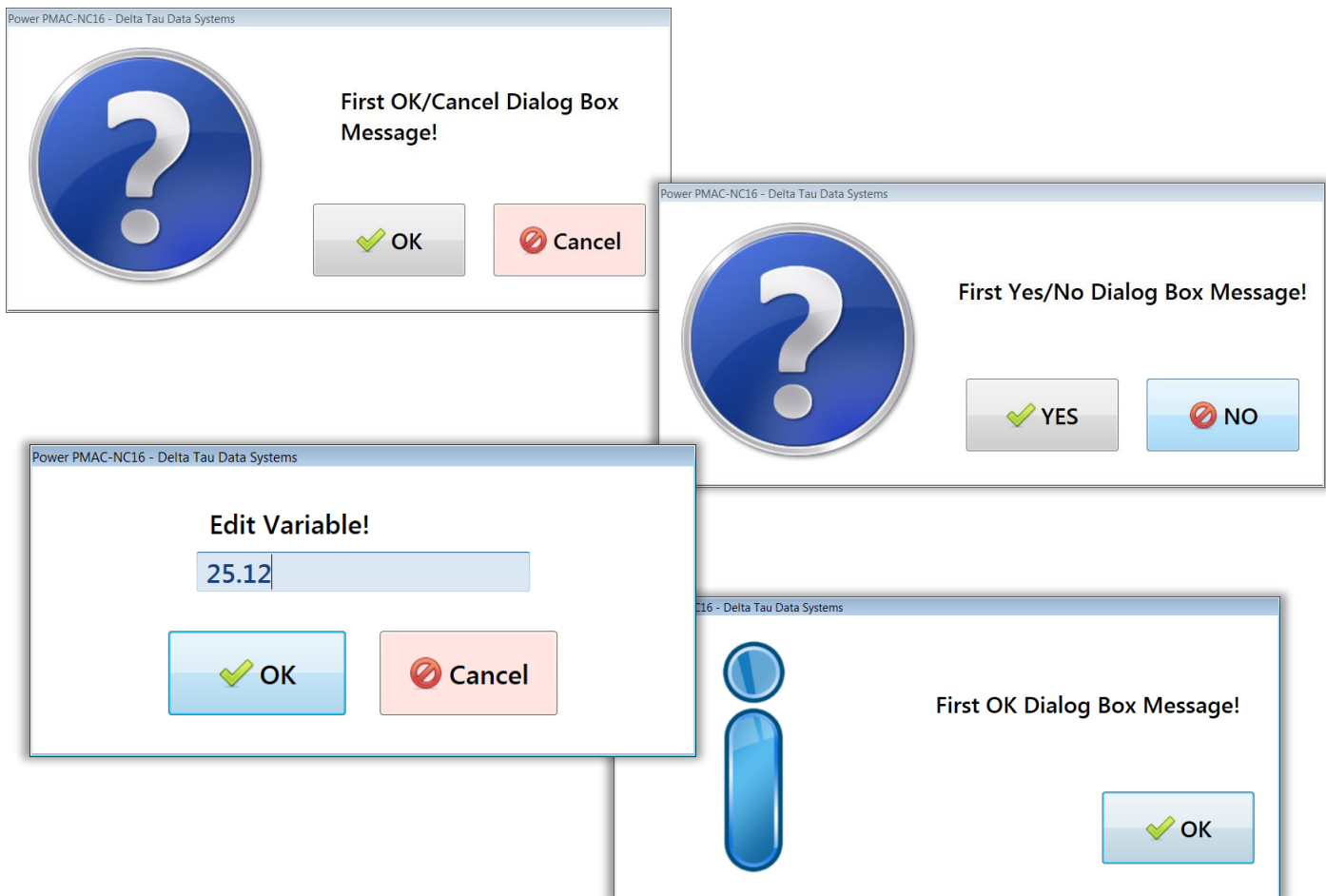
- send1 "SetFatalMessage=id,text"
- send1 "SetWarningMessage=id,text"
- send1 "SetInformationMessage=id,text"
- send1 "ClearMessage=id"
- send1 "ClearAllMessages"

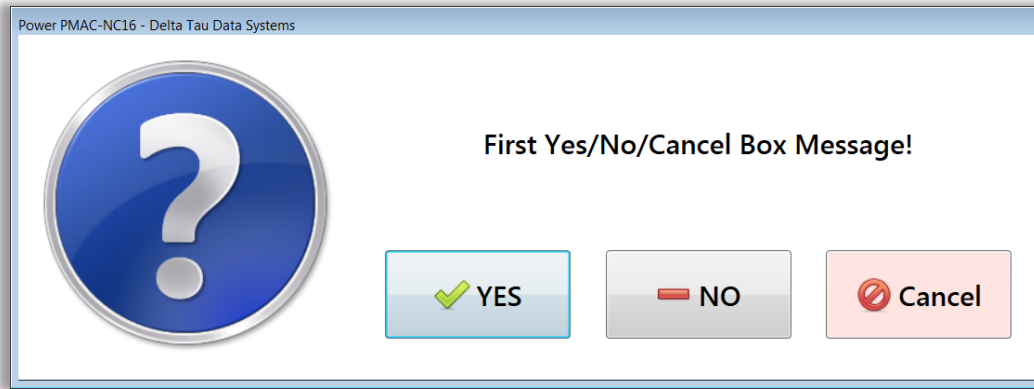
Type 4 – Pop-Up Dialog Message Boxes

Type 4 messages trigger pop-up dialog boxes with custom messages to be displayed. There are four different kinds of dialog messages. There are two acknowledge resultants which can be evaluated by the PMAC script code to create logic which reacts to the user input. The command structures are shown below:

- send1 "DialogOk=prompt"
- send1 "DialogOkCancel=prompt"
- send1 "DialogYesNo=prompt"
- send1 "DialogYesNoCancel=prompt"
- send1 "EditVariable=variable,prompt"

Member definitions directly from the HMI can be modified as well (ex. send1 "EditMember=WorkOffsets.G54.A1, Edit X offset!.")





The Visual Studio Project

This section briefly explains and shows the PPNC16 SDK solution file and its content. In order to be able to open and modify such a file followings are necessary:

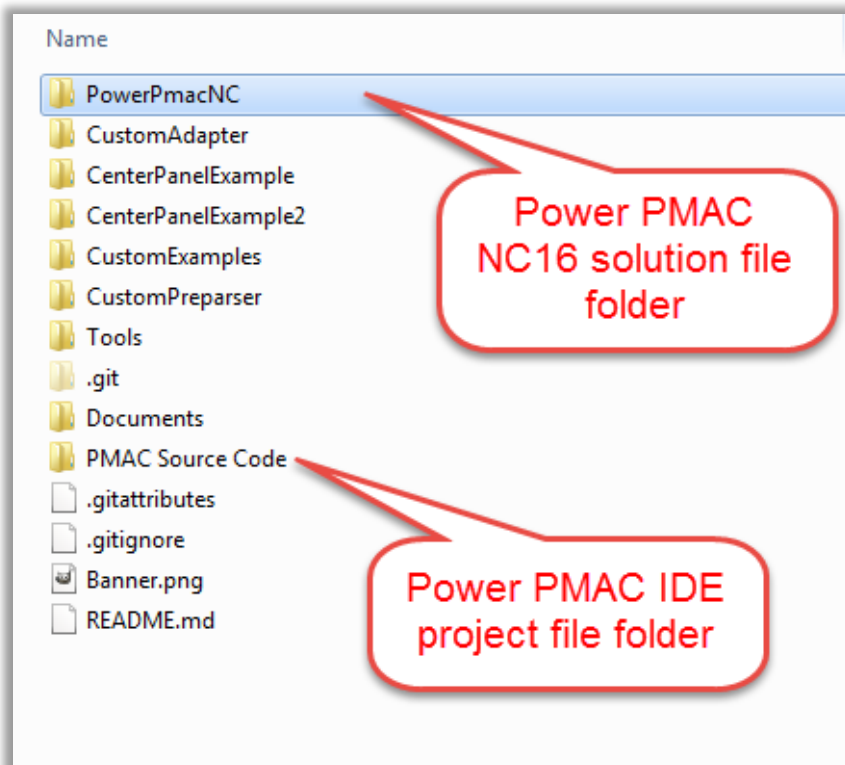
Visual Studio Community 2017:

<https://www.visualstudio.com/thank-you-downloading-visual-studio/?sku=Community&rel=15#>

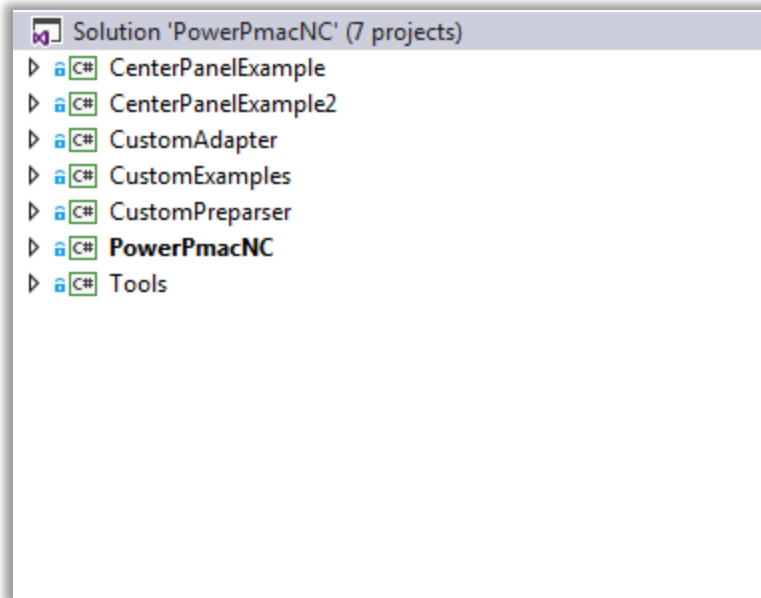
💡 PPNC16 SDK solution file is written in a way to work completely with either Visual Studio Community version or Visual Studio Professional for users' convenience.

💡 Make sure Microsoft .NET Framework 4.6.1 and Microsoft Visual C++ 2010 Redistributable Package (x64 or X86) are both installed to run and debug the application.

PPNC16 SDK folder includes following files and folders:



The “Power PMAC NC” solution file can be found in “PowerPmacNC” folder. The solution file includes seven projects. The main project is “PowerPMACNC” and rest are examples which are explained in “External Assemblies” section.





External Assemblies

PPNC16 SDK version solution file includes different examples written with C#/WPF in Visual Studio environment. These examples can be used either as “plug in” with slight modifications from users side or can be used as a start point for developers. Majority of these examples are well implemented with intention of saving development time. Such feature can be activated/deactivated by configuring the “PowerpmacNC.ini”. The source files are well commented for convenience. A custom object adds its own device members to the Machine View hierarchy and also attaches "Changed" handlers to members created by the main program. A custom WPF Panel responds to UI Skin changes and is registered for foreign language translation. Simply add the following lines to a “PowerPmacNC.ini” file to test the custom plugin DLL.

Users’ custom WPF Pages can be displayed by the main program in five different locations as shown in the illustration. Panels designed for the left, center and right columns should be tall and narrow, panels designed for the main screen tab area should be square, and panels designed to be full-screen *User Pages* may be much larger and more complex. Custom panels are hosted inside WPF *Viewboxes* so that they will be sized to fit the available screen area.

The intention of this section is, explaining these examples in more details to assist users for further implementations and software developments of PPNC16 SDK version.

 Note: When distributing the binaries to custom made machines, Do NOT move "CustomExamples.dll" to the main directory with "PowerPmacNC.exe" because the custom library has its own "DeviceMembers.xml" file. Instead, create a subdirectory for the library and its dependencies, and change the lines in "PowerPmacNC.ini" to “[External Assemblies]”.

 MCF libraries in this subdirectory already exist in the main directory.

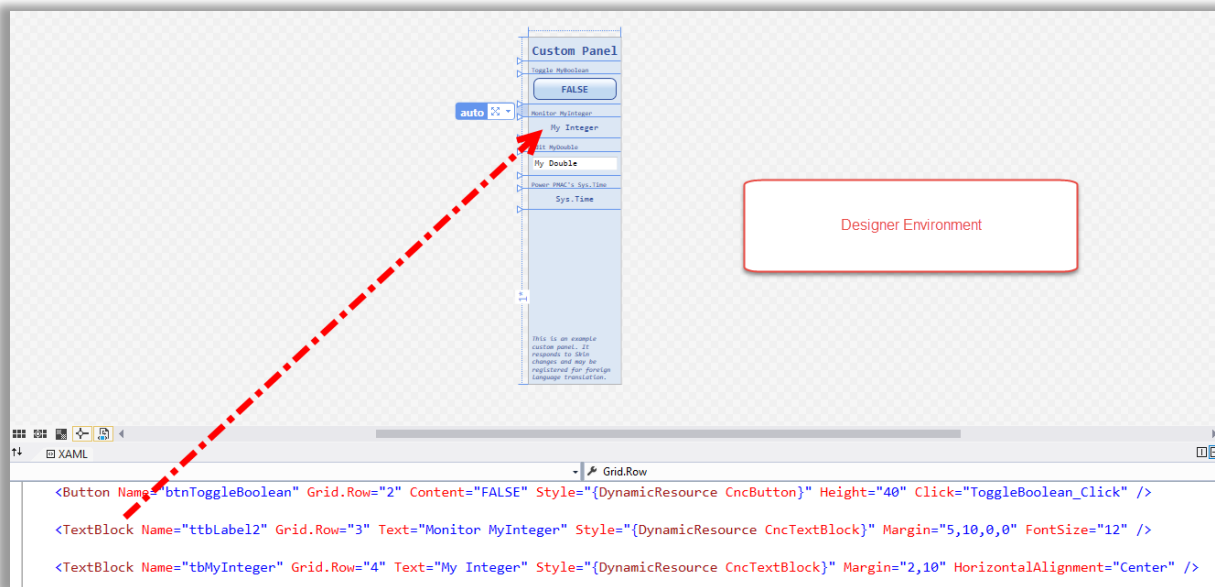
💡 Currently PPNC16 SDK version includes “CenterPanelExample”, “CenterPanelExample2”, “Custom Adapter”, “Custom Examples”, “Custom Preparser”, and “Tools”.

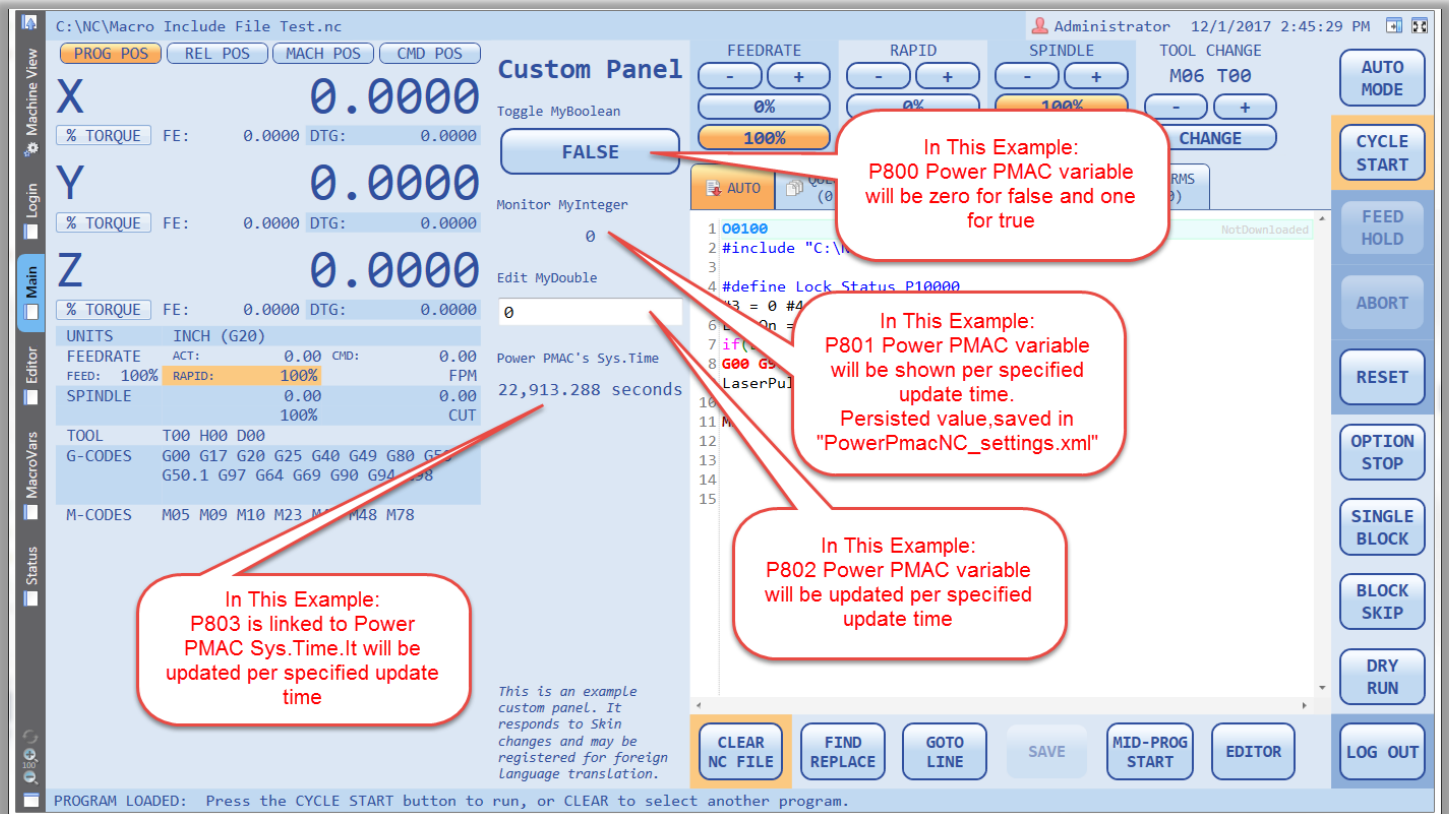
CenterPanelExample:

This is the first example that can be seen in PPNC16 SDK version solution in Visual Studio solution explorer. Add the following lines to a “PowerPmacNC.ini” file to activate such a feature:

```
Object="..\..\..\CenterPanelExample\bin\Debug\CenterPanelExample.dll;CenterPanelExample.CustomObject"  
CenterCustomFrame="..\..\..\CenterPanelExample\bin\Debug\CenterPanelExample.dll;CenterPanelExample.PageCenterPanel"
```

💡 If it is desired to add extra buttons, text blocks (displays), and text boxes, simply copy/past codes following the original feature. Rename new features, assign different variables, and add proper change of events for each.





💡 Change "CenterCustomFrame" to "RightCustomFrame" or "LeftCustomFrame" in a "PowerPmacNC.ini" file to move the panel to the right or left of the screen respectively. Syntaxes are provided below as references:

LeftCustomFrame="..\..\..\CenterPanelExample\bin\Debug\CenterPanelExample.dll;CenterPanelExample.PageCenterPanel"

Or

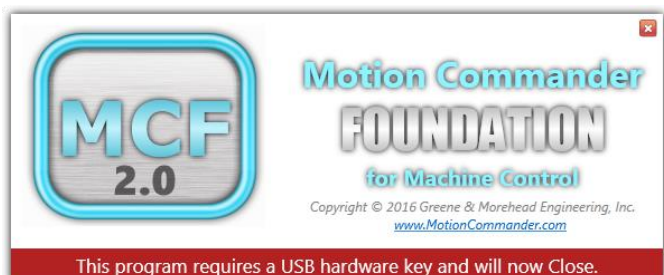
RightCustomFrame="..\..\..\CenterPanelExample\bin\Debug\CenterPanelExample.dll;CenterPanelExample.PageCenterPanel"

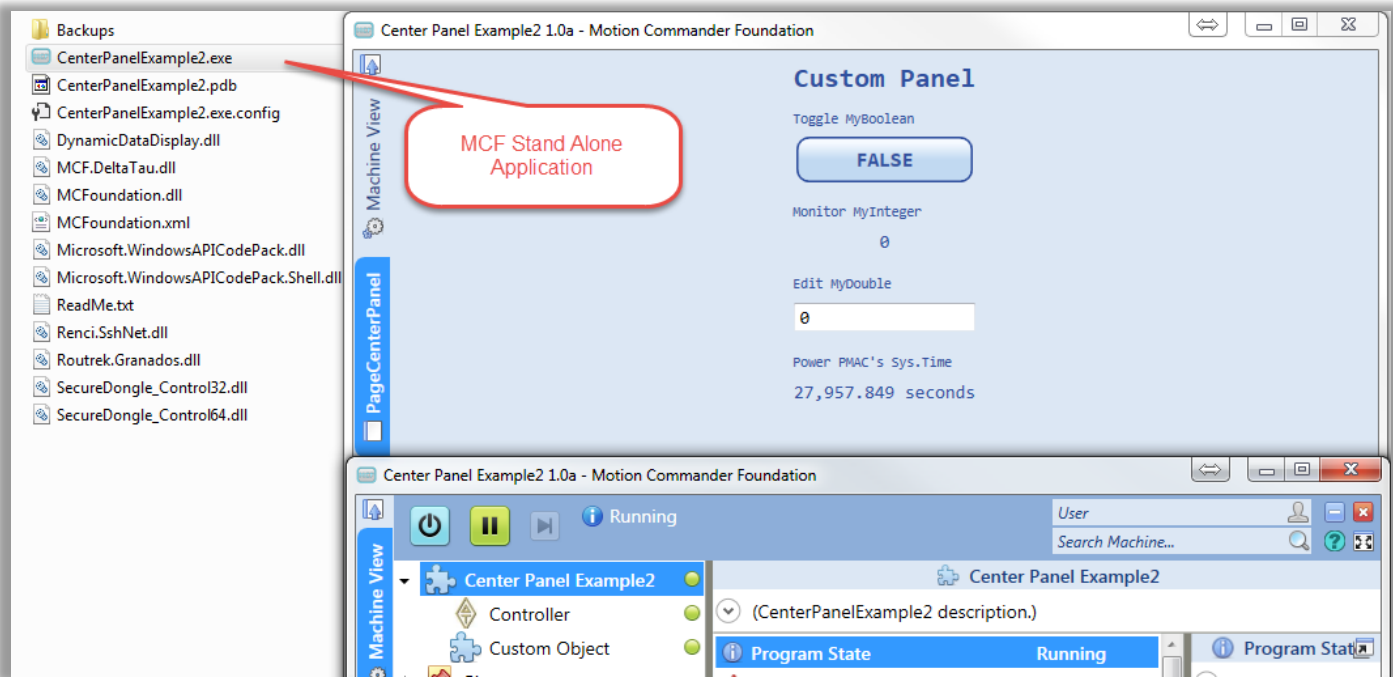
CenterPanelExample2:

This assembly is a companion project to the "CenterPanelExample". The difference is, this assembly is a minimal Motion Commander Foundation (MCF) stand-alone application which is capable of hosting "PageCenterPanel.xaml". After applying changes and performing a "build" in Visual Studio, use the following path to run this stand-alone application:

"..\PowerPmacNc16-SDK\CenterPanelExample2\bin\Debug\ CenterPanelExample2.exe"

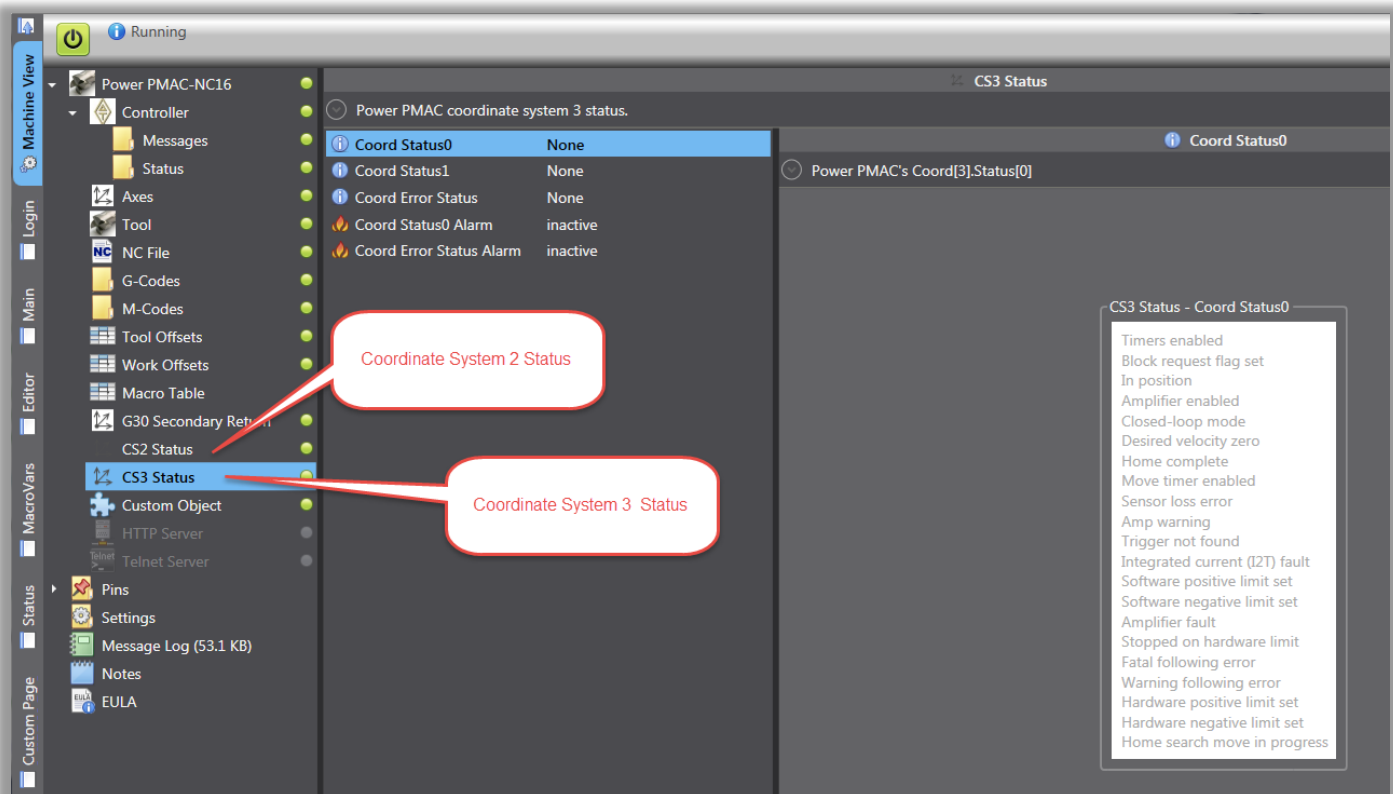
💡 This application requires a USB hardware key (dongle) to run.





Custom Examples:

This assembly includes several different features. As it was shown in “External Assemblies” section, based on “PowerPmacNC.ini” file, custom page, custom tab, and custom panels can be added all to the PPNC 16. Beside these features, users are able to monitor status of multiple coordinate systems. By default CS2 and CS3 are included in “CoordStatus.cs”.



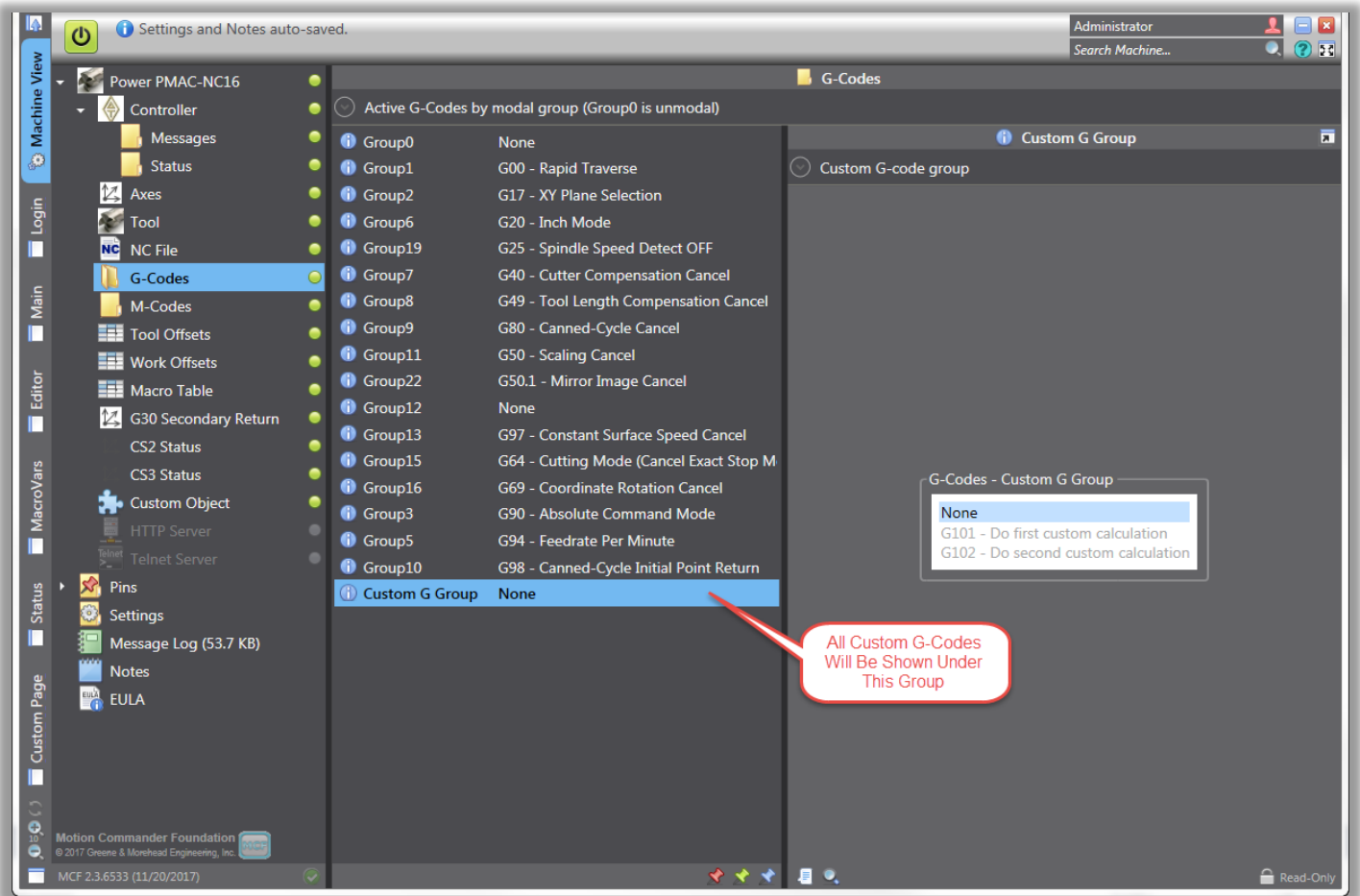
To enable this feature, add the following line to “PowerPmacNC.ini” file:

Object="..\..\..\CustomExamples\bin\Debug\CustomExamples.dll;CustomExamples.CS2Status"

AND/OR

Object="..\..\..\CustomExamples\bin\Debug\CustomExamples.dll;CustomExamples.CS3Status"

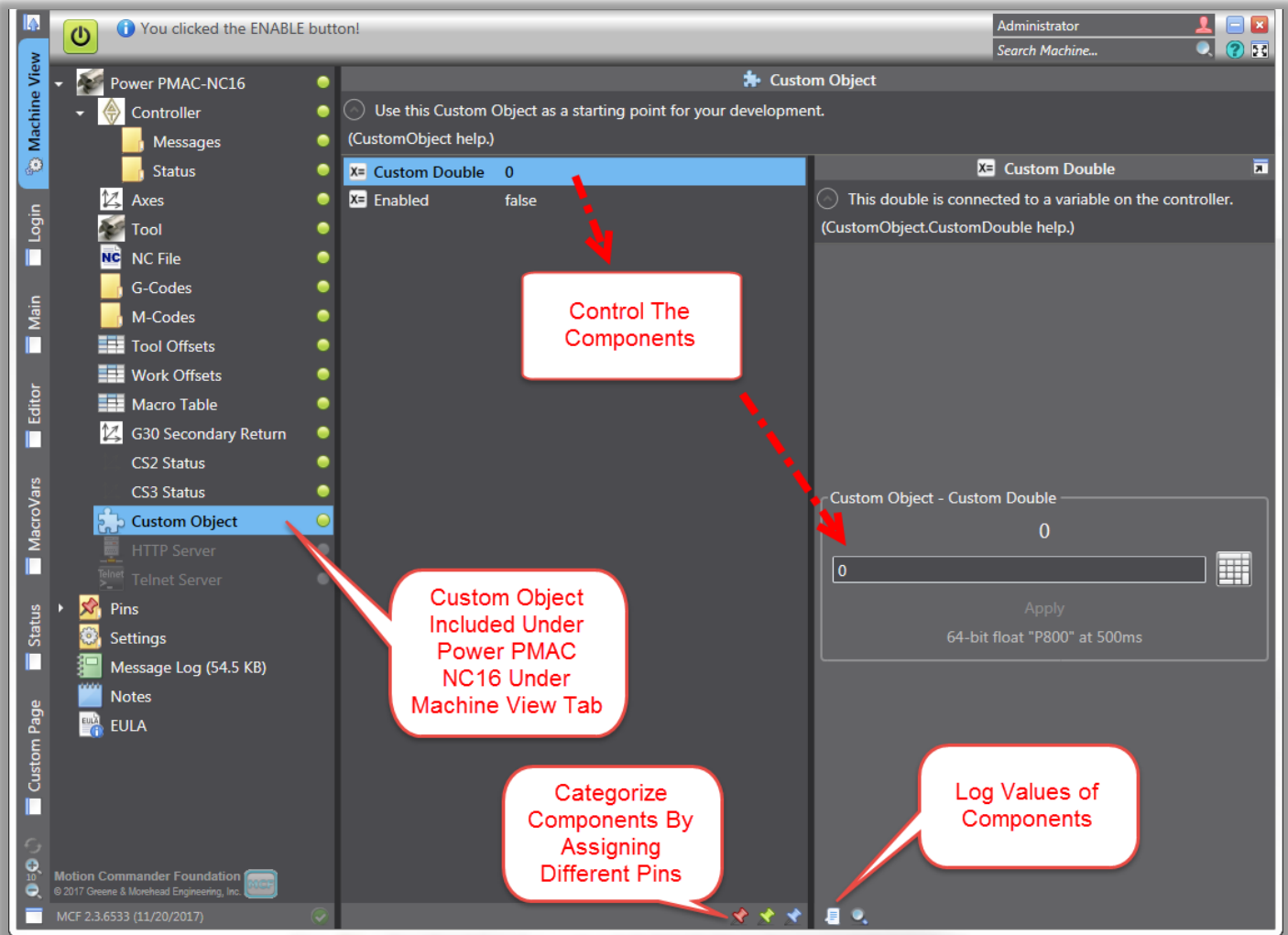
Another feature that this external assembly provides as an example is “Custom G Group” which is included in “CustomCodeGroups.cs”. Such a feature allows users to create their own custom G/M-Codes as it is shown below:



To enable this feature, add the following line to “PowerPmacNC.ini” file:

CodeGroups="..\..\..\CustomExamples\bin\Debug\CustomExamples.dll;CustomExamples.CustomCodeGroups"


Another feature that this external assembly provides as an example is “Custom Object”. Such a feature creates an extension for “Custom Page” members under “Power PMAC-NC16” section in “Machine View” tab. This feature allows users to apply access restrictions and all other features that PPNC16 provides for logging and troubleshooting.




FkeyHandler:

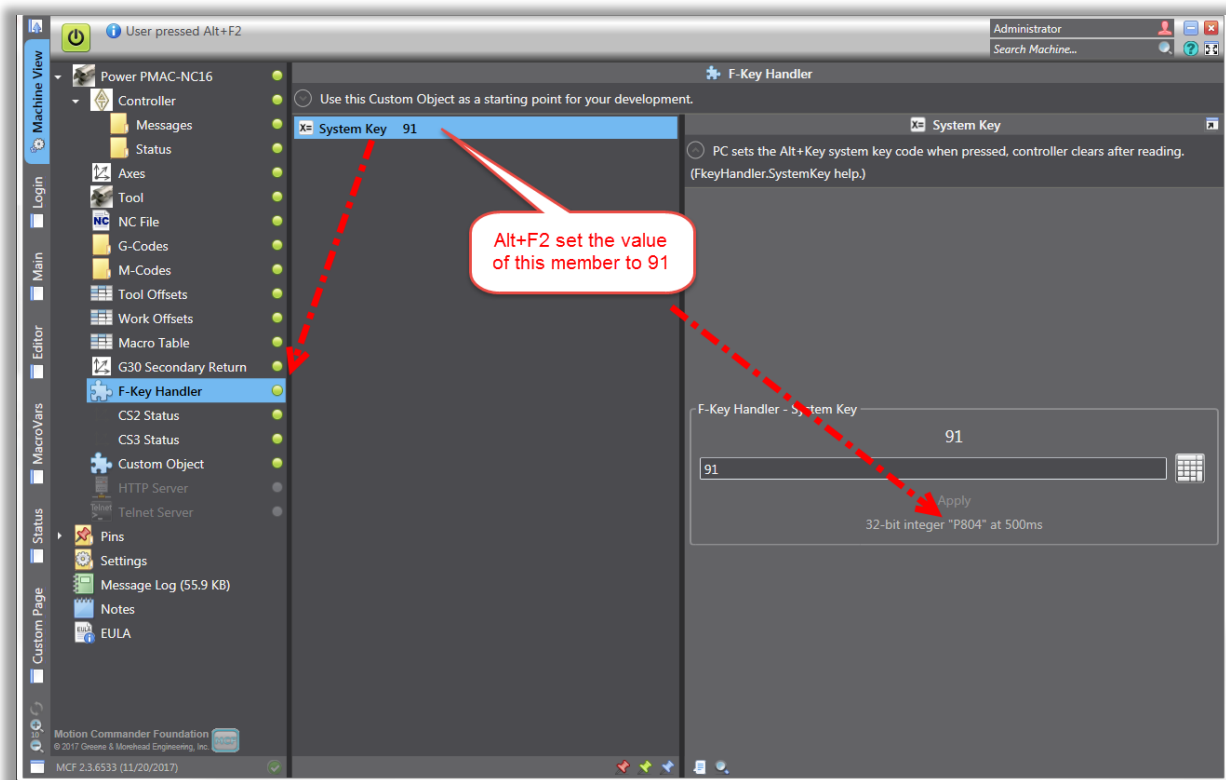
This feature, as part of “CustomExamples” external assembly provides examples of having PPNC16 performs different tasks based on pressing any of functional keys (F1-F12), Ctrl + functional keys, shift + functional keys, or Alt + any key. To enable this feature, add the following line to “PowerPmacNC.ini” file:

Object=“..\..\CustomExamples\bin\Debug\CustomExamples.dll;CustomExamples.FkeyHandler”

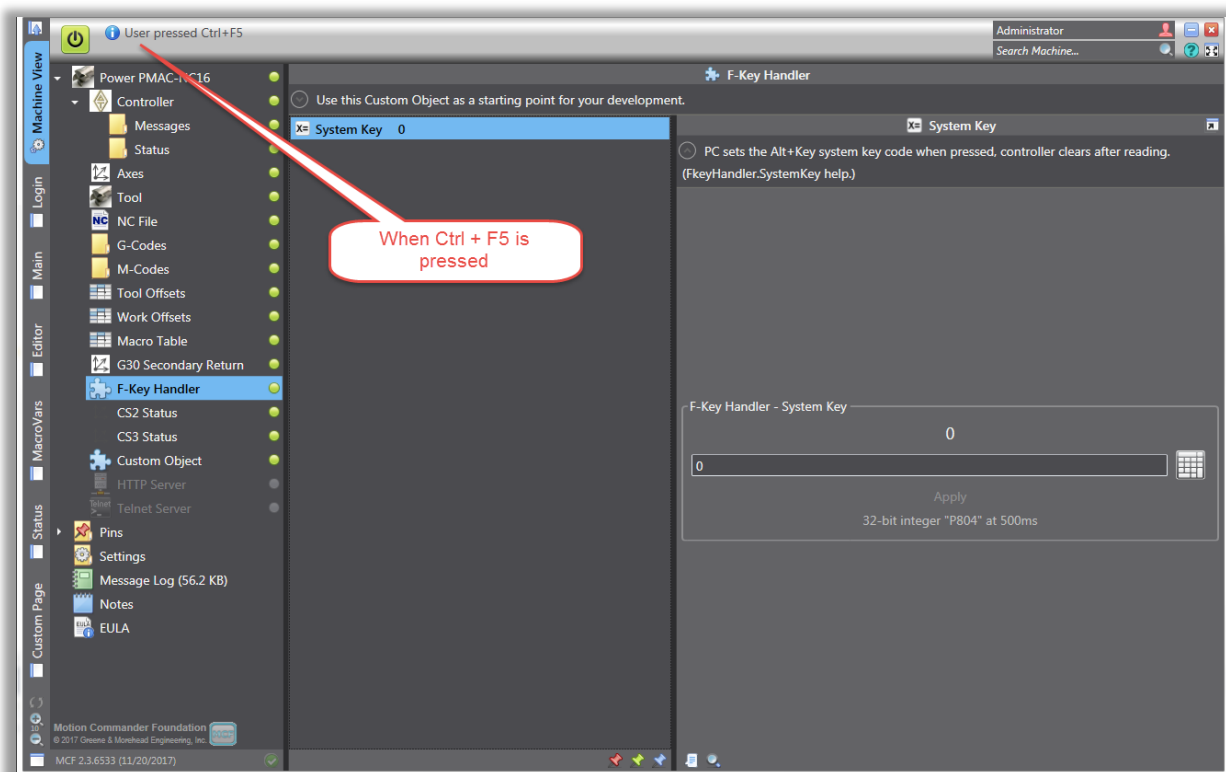
 Example one, demonstrates how PPNC16 assigns different values to Power PMAC p-variable (P804) based on “Alt + key” combinations. Each combination has its own unique identification number. Therefore, controller can be programmed to perform variant tasks based on each combination.

 Example two and three, demonstrates how PPNC16 detects “Ctrl + Functional key” or “Shift + Functional Key” combinations. Therefore, through SDK version, PPNC16 can be programmed to perform variant tasks based on each combination.

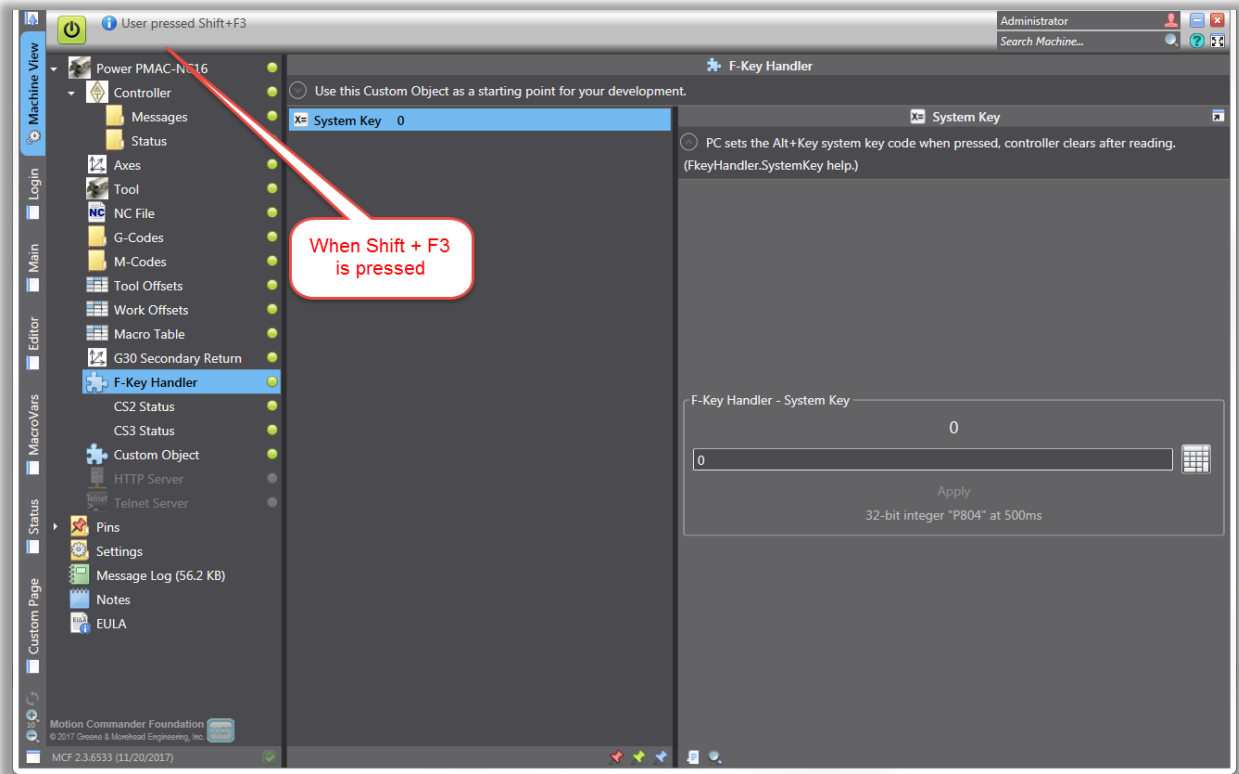
Example.1 Alt + Key Combinations:



Example.2 Ctrl + Functional Key Combinations:



Example.3 Shift+ Functional Key Combinations:



GridLengthAnimation:

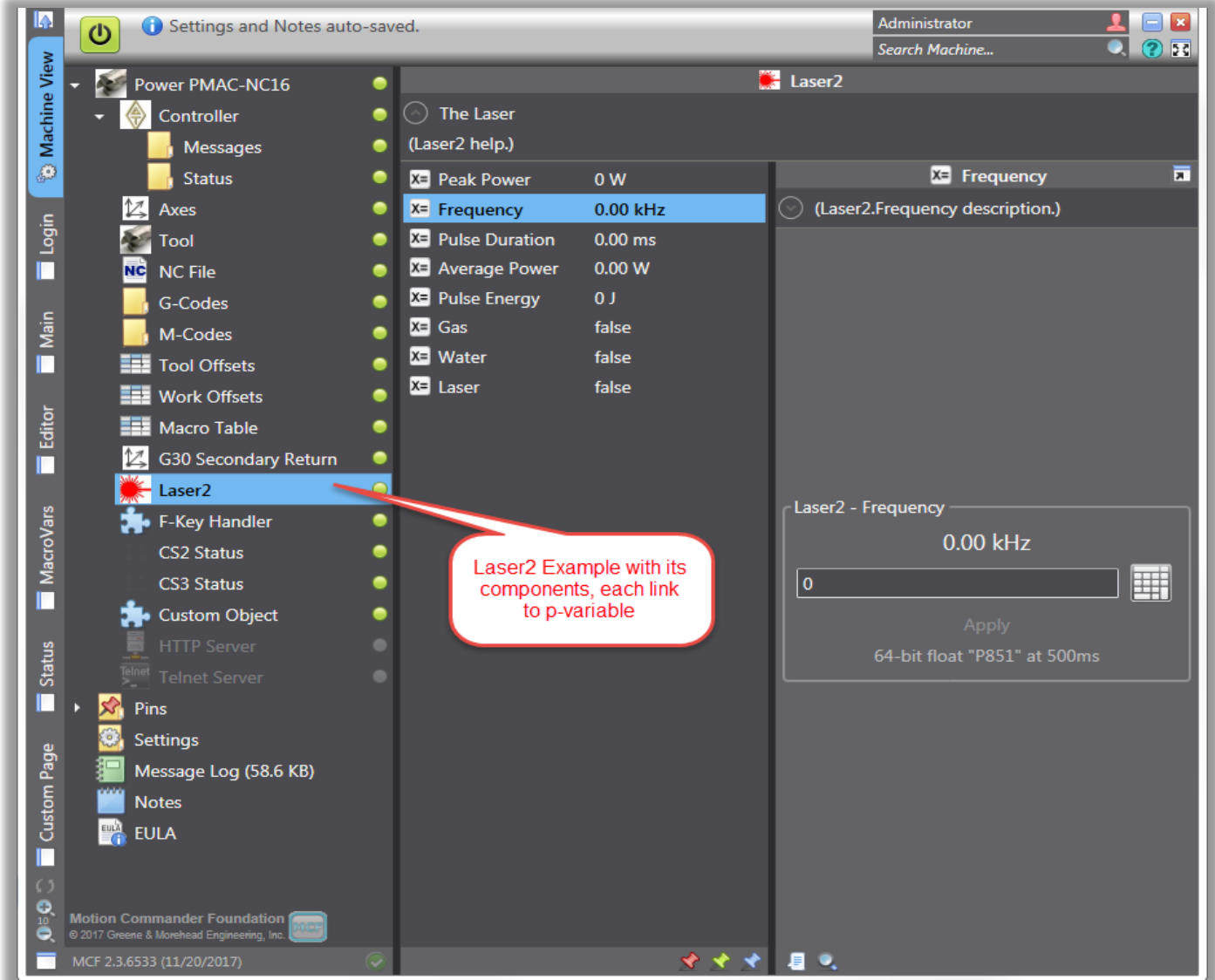
This project handles the split-screen feature in PPNC16. As it was explained in “Run Screen” section when any subprogram is called from another program, second screen slides up and splits the screen to two in order to show both programs at the same time.



This project is provided as a reference **ONLY** . It is strongly **NOT** recommended to modify this project for any reason.

Laser2:

Laser2 is a small example, provided as part of “Custom Examples” which can be used as a start point for laser applications development.



NcLinePreparer:

This feature which is part of “Custom Examples” allows users to create custom parser in addition to PPNC16 parser. To enable this feature, add the following line to “PowerPmacNC.ini” file:

NcLinePreparer="..\..\..\CustomExamples\bin\Debug\CustomExamples.dll;CustomExamples.NcLinePreparer"

Visual Studio Code:

```
public bool ParseNcLine(ref string line, int lineNumber)
{
    if (string.IsNullOrEmpty(line) || string.IsNullOrEmpty(lineNumber))
    {
        return false;
    }

    if (!macros.ParseNcLine(ref line, lineNumber))
    {
        return false;
    }

    if (line.StartsWith("Cat=")) // perform your custom processing
    {
        cat = line.Substring(4);
        line = "";
        g.Report(this, EReportType.InformationAlarm, "NcLinePreparer: Cat={0}", cat);
    }
}
```

As soon as "Cat=" is detected, PPNC16 generates this message

PPNC16 preparser behavior:

As soon as "Cat=" is detected, PPNC16 generates this message

1 05000
2 G00 G90 G54
3 X0 Y0
4 Cat=5
5 Dog=3
6 //Fish=5
7 Square
8

NcProgram.pmc
1 open subprog 10000 // 05000
2 P724=0
3 N2 G00G90G54
4 N3 X0Y0
5 N7 P724==1
6 dwell10
7 if(P265000<1)
8 {
9 N7 P265000=1
10 }
11 if(P265000<1)P265000=1
12 N7 G01X25Y25
13 N7 X50
14 N7 Y50
15 N7 X25
16 N7 Y25
17 N7 P724==2
18 close
19
20

NcLinePreparer: Dog=3
12/4/2017 2:32:46 PM
NcLinePreparer: Cat=5
12/4/2017 2:32:46 PM

Multi-line pre-parser output example:

```

if (line.StartsWith("Cat=")) // perform your custom processing
{
    cat = line.Substring(4);
    line = "";
    g.Report(this, EReportType.InformationAlarm, "NcLinePreparser: Cat={0}", cat);
}
else if (line.StartsWith("Dog="))
{
    dog = line.Substring(4);
    line = "";
    g.Report(this, EReportType.InformationAlarm, "NcLinePreparser: Dog={0}", dog);
}
else if (line.StartsWith("Fish="))
{
    g.Report(this, EReportType.ErrorAlarm, "NC Error: Pre-parser, line {0} \"{1}\"", lineNumber, line);
    return false;
}
else if (line.Equals("Square", System.StringComparison.InvariantCultureIgnoreCase))
{
    // Multi-line pre-parser output. Separate lines with "\n" or "\r\n".
    line = "((Square Begin))\ndwell0\nif(P265000<1)\n{\n\nP265000=1\n}\nif(P265000<1)P265000=1\nG01 X25 Y25\nX50\nY50\nX25\nY25\n((Square End))";
}

```

When "square" is detected in the NC program, it will be replaced according to this line

C:\NC\CustomParser.nc (1:7 lines. Parsing Complete)

NC Main

1 05000
2 G00 G90 G54
3 X0 Y0
4 Cat=5
5 Dog=3
6 //Fish=5
7 Square
8

When "square" is detected in the NC program, it will be replaced by these lines

NcProgram.pmc

```

1 open subprog 10000 // 05000
2 P724=0
3 N2 G00G90G54
4 N3 X0Y0
5 N7 P724==1
6 dwell0
7 if(P265000<1)
8 {
9 N7 P265000=1
10 }
11 if(P265000<1)P265000=1
12 N7 G01X25Y25
13 N7 X50
14 N7 Y50
15 N7 X25
16 N7 Y25
17 N7 P724==2
18 close
19
20

```

NcLinePreparser: Dog=3
12/4/2017 2:32:46 PM

NcLinePreparser: Cat=5
12/4/2017 2:32:46 PM

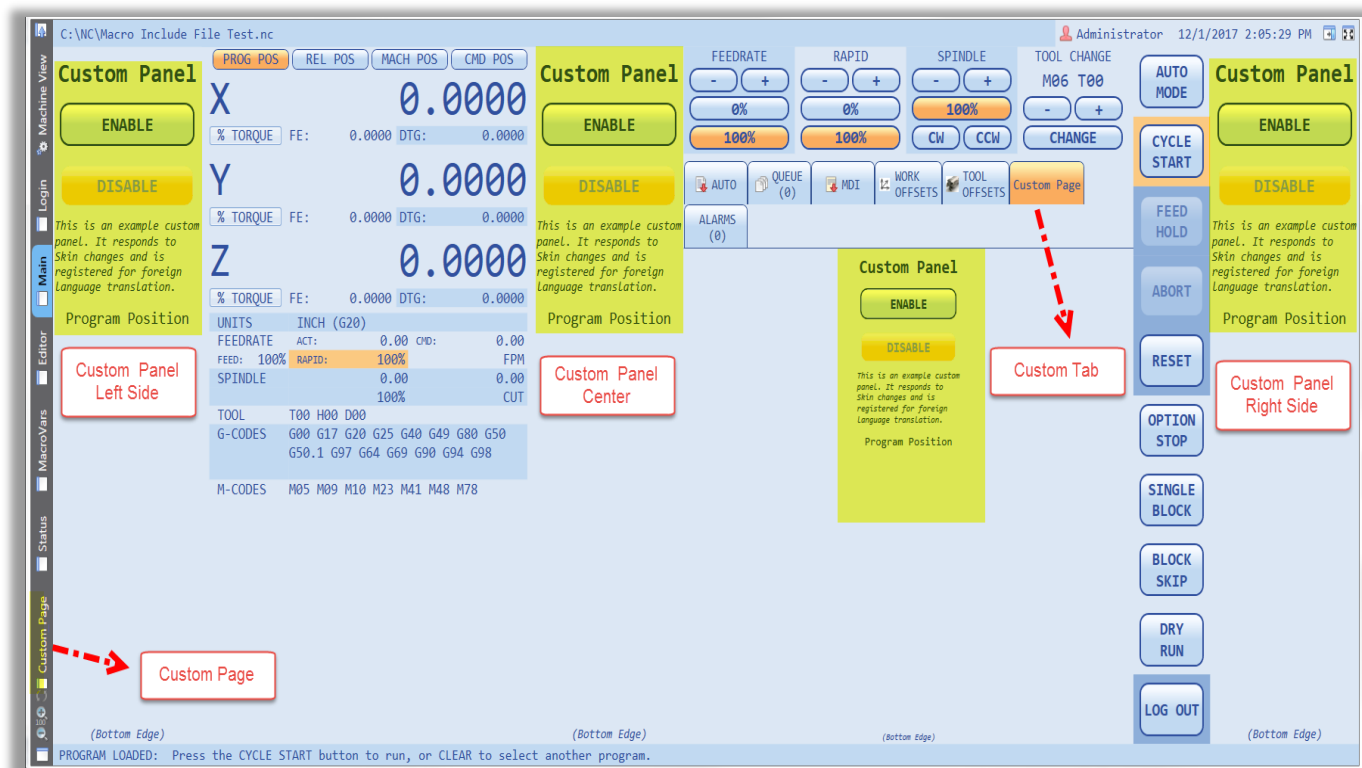


MacroProcessor() is used to support #define macro substitutions. Delete this line if it is not desired to support such a feature.

PageCustom:

This project allows users to add a custom panel as a “custom tab” or “custom page”. Also it allows users to shift the custom made panel in a main screen from left to right or center. Such settings are all configurable in “PowerpmacNC.ini” file as it is shown below:

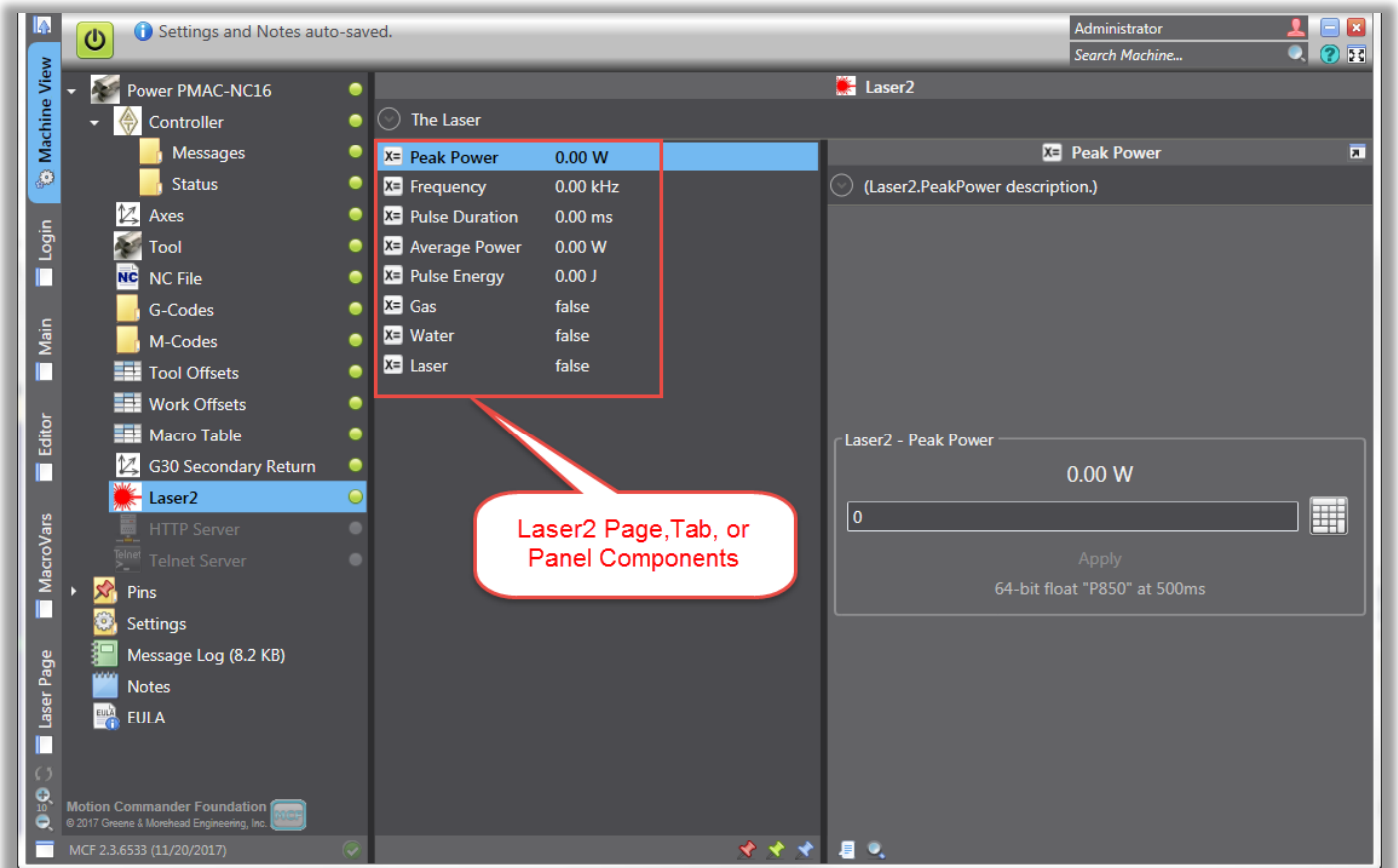
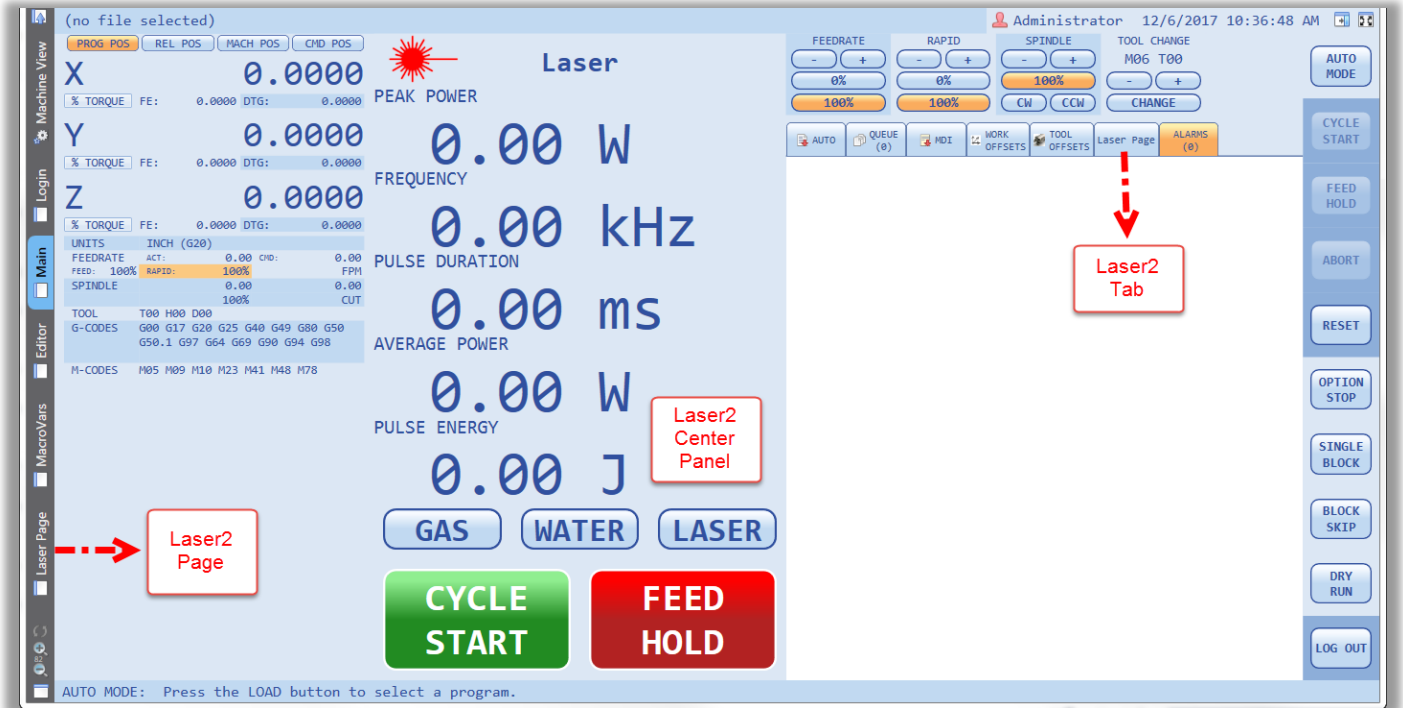
```
Object="..\..\..\CustomExamples\bin\Debug\CustomExamples.dll;CustomExamples.CustomObject"  
UserPage="..\..\..\CustomExamples\bin\Debug\CustomExamples.dll;CustomExamples.PageCustom"  
CustomTab="..\..\..\CustomExamples\bin\Debug\CustomExamples.dll;CustomExamples.PageCustom"  
LeftCustomFrame="..\..\..\CustomExamples\bin\Debug\CustomExamples.dll;CustomExamples.PageCustom"  
CenterCustomFrame="..\..\..\CustomExamples\bin\Debug\CustomExamples.dll;CustomExamples.PageCustom"  
RightCustomFrame="..\..\..\CustomExamples\bin\Debug\CustomExamples.dll;CustomExamples.PageCustom"
```



In the above example same “Custom Panel” is used in different places. Therefore, a change of its component using either feature, will be applied to all.

PageLaser2:

This example as part of “Custom Examples” external assembly is implemented to be used in conjunction with “Laser2.cs” project. Activation of such feature will add the following based on a chosen option in “PowerpmacNC.ini” file as either a page, tab, or panel:



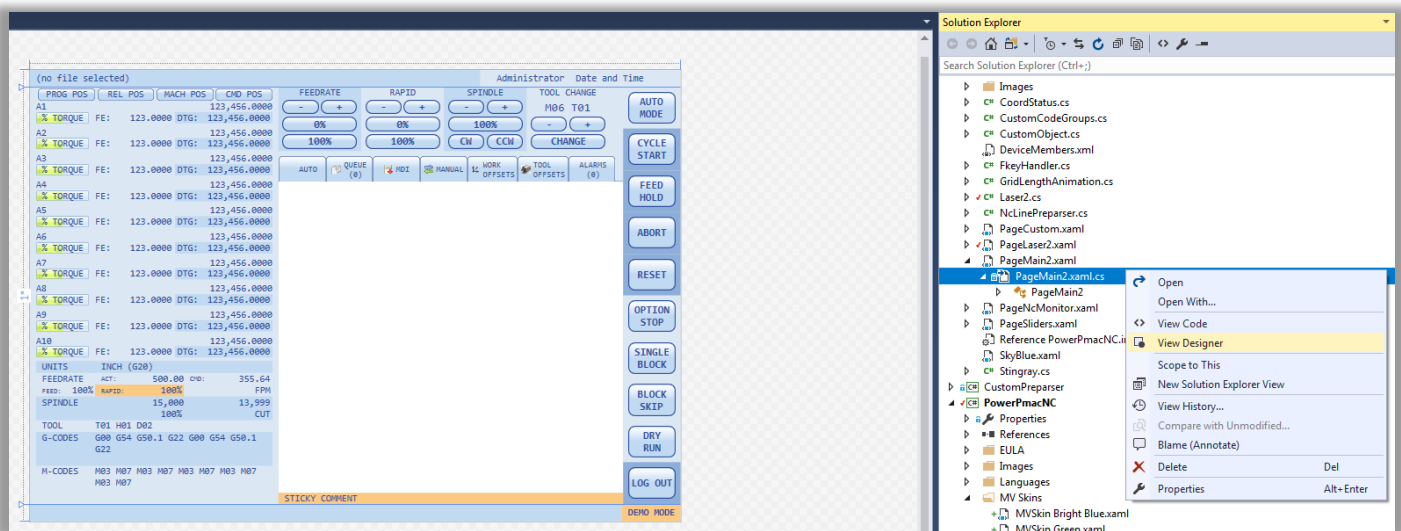


“PowerpmacNC.ini” file can be configured in different ways to be shown as a tab, panel (left,right, or center), or page as shown below:

```
Object=".....\CustomExamples\bin\Debug\CustomExamples.dll;CustomExamples.Laser2"
UserPage=".....\CustomExamples\bin\Debug\CustomExamples.dll;CustomExamples.PageLaser2"
CustomTab=".....\CustomExamples\bin\Debug\CustomExamples.dll;CustomExamples.PageLaser2"
;LeftCustomFrame=".....\CustomExamples\bin\Debug\CustomExamples.dll;CustomExamples.PageLaser2"
CenterCustomFrame=".....\CustomExamples\bin\Debug\CustomExamples.dll;CustomExamples.PageLaser2"
;RightCustomFrame=".....\CustomExamples\bin\Debug\CustomExamples.dll;CustomExamples.PageLaser2"
```

PageMain2:

This project as part of “Custom Examples” allows user and developers to modify the “Main” PPNC16 page according to their custom designs. Following figure shows default components that come and load with this feature when application starts based on “PowerpmacNC.ini” configurations:



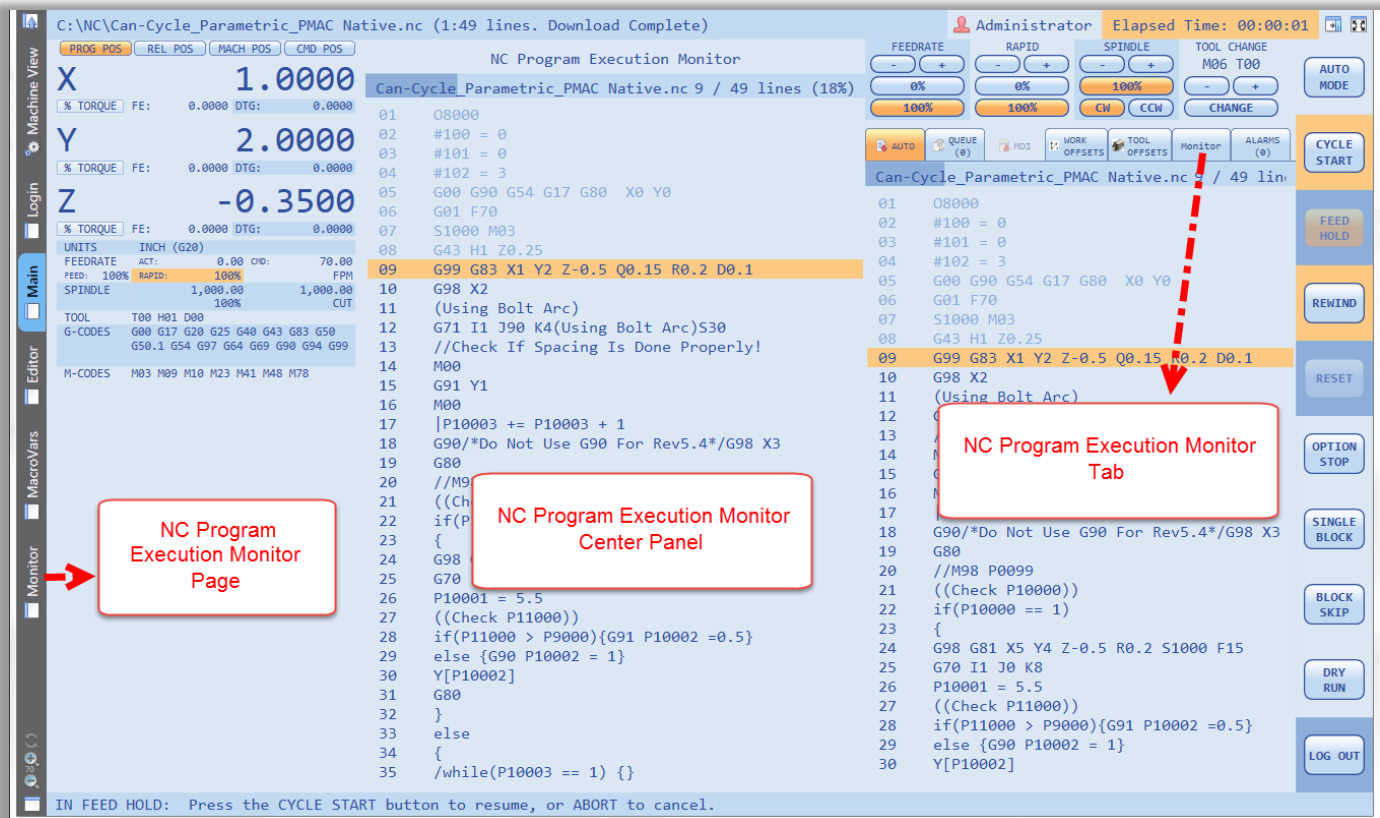
PageNCMonitor:

This is another feature provided under “Custom Examples” that allows users to add “NC Program Execution Monitor” as a custom page, tab, or panel upon its activation in “PowerpmacNC.ini” file.

```
UserPage=".....\CustomExamples\bin\Debug\CustomExamples.dll;CustomExamples.PageNCMonitor"
CustomTab=".....\CustomExamples\bin\Debug\CustomExamples.dll;CustomExamples.PageNCMonitor"
;LeftCustomFrame=".....\CustomExamples\bin\Debug\CustomExamples.dll;CustomExamples.PageNCMonitor"
CenterCustomFrame=".....\CustomExamples\bin\Debug\CustomExamples.dll;CustomExamples.PageNCMonitor"
;RightCustomFrame=".....\CustomExamples\bin\Debug\CustomExamples.dll;CustomExamples.PageNCMonitor"
```



If it is desired, a frame can be shifted to left, center, or right side of the main page by modifying the “PowerpmacNC.ini” file.



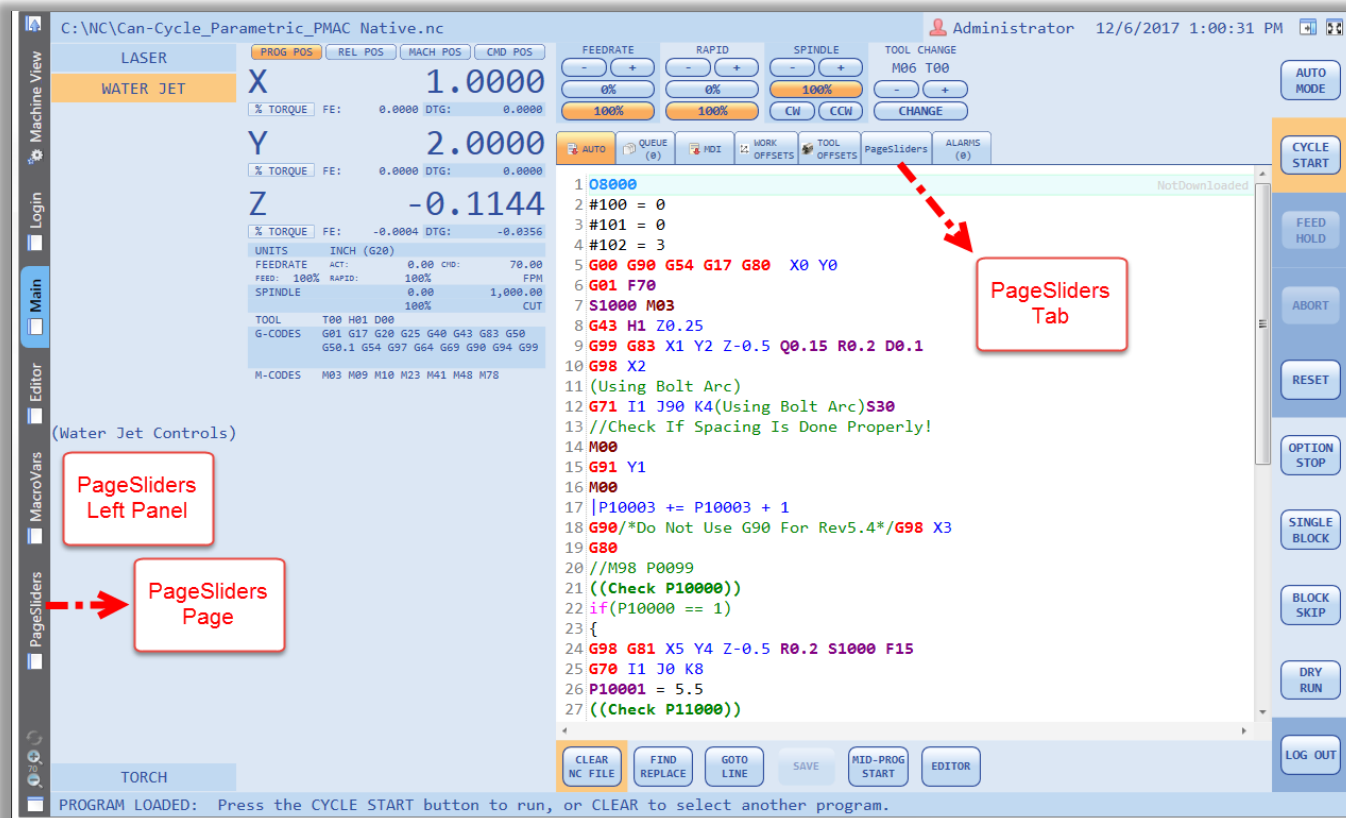
PageSliders:

As part of “Custom Examples” this feature provides different pages that activation of each will make the active page slide over the previous one in an animated way using “GridLengthAnimation” based on a define “TimeSpan”.

💡 By default the value of a timer (TimeSpan.FromMilliseconds(value/variable)) is 400 milliseconds. Users can use a variable or fix value to set this timer.

💡 If it is desired, each page can be designed to have its own theme and skin. By default, skin for each page get set according to the chosen skin under “Settings”.

💡 “PageSliders” feature is a very useful tool if users or developers are planning to have multi coordinate systems. “Main” page with its feature can be implemented in different pages with respect to the chosen page and coordinate system.



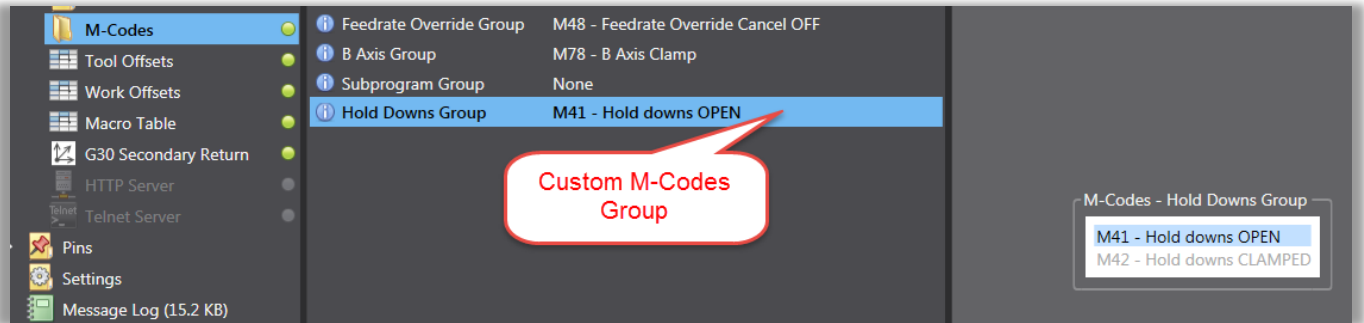
Stingray:

This unique project as part of “Custom Examples” generates a CSV file that contains arbitrary chosen P-variables (P850-P852). This project includes two custom M-Codes such as M41 (Hold downs OPEN) and M42 (Hold downs CLAMPED). Upon M42 execution, PPNC16 creates a “Test.CSV” file that includes latest values of P850-P852. In this example, pressing F3 on a keyboard will create the same outcome (for testing and simulation purposes.) Activate such a feature by adding the following to the “PowerpmacNC.ini” file as follow:

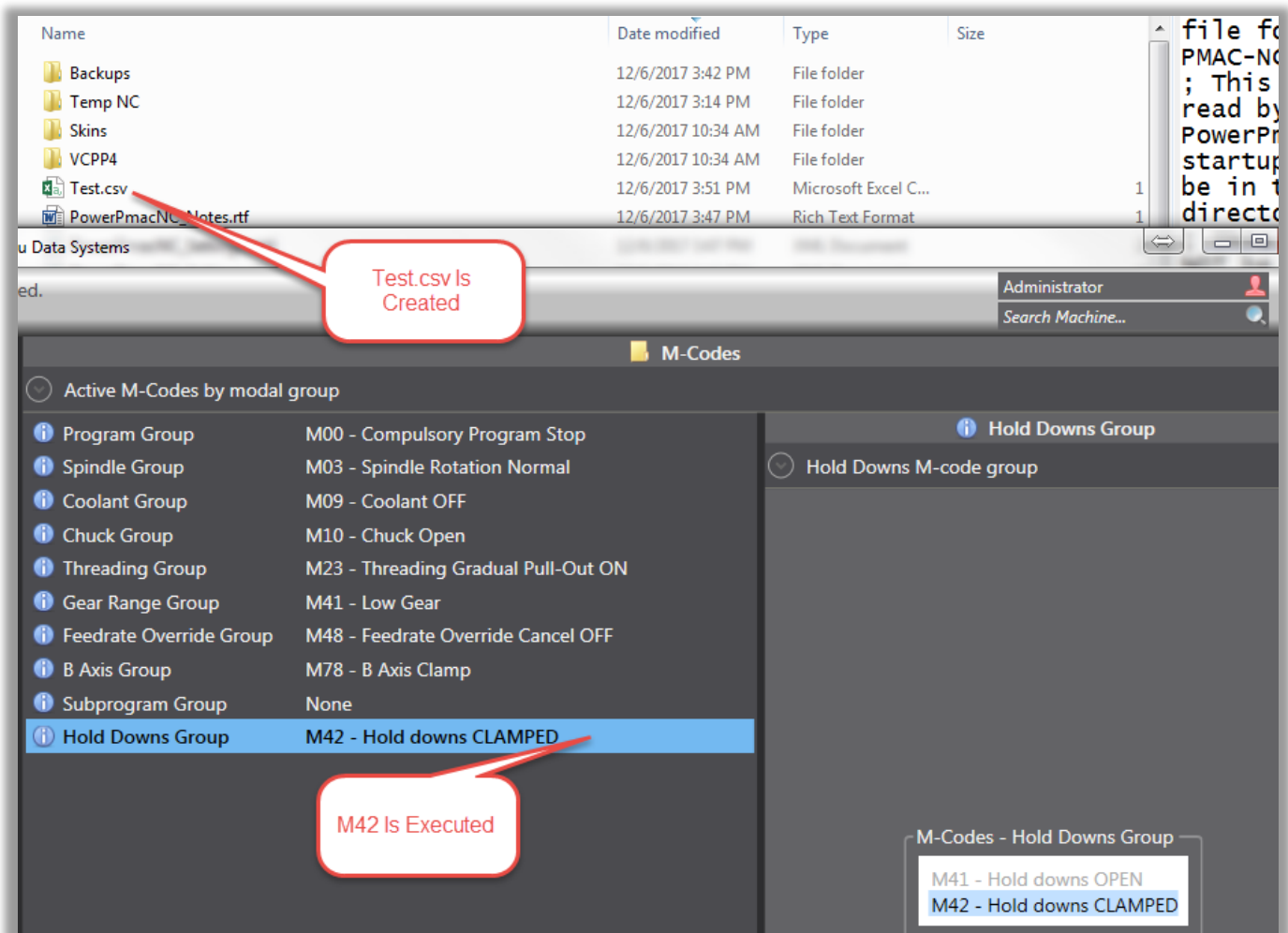
```
CodeGroups="..\..\..\CustomExamples\bin\Debug\CustomExamples.dll;CustomExamples.Stingray"
```

```
public enum EHoldDownsGroup
{
    [Description("M41 - Hold downs OPEN")]
    M41,
    [Description("M42 - Hold downs CLAMPED")]
    M42,
};
```

M41 by default is active as follow:



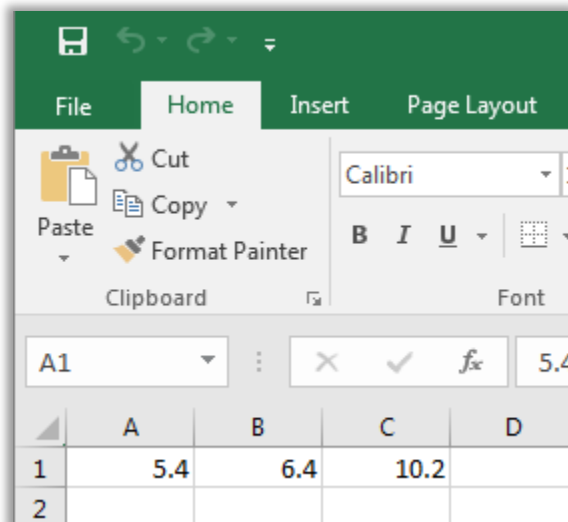
Execution of M42 or pressing F3 will create a Test.CSV file as follow:



Values Of P850..P852 in Power PMAC IDE terminal window:

Test.CSV file contents latest values of P850..P852:

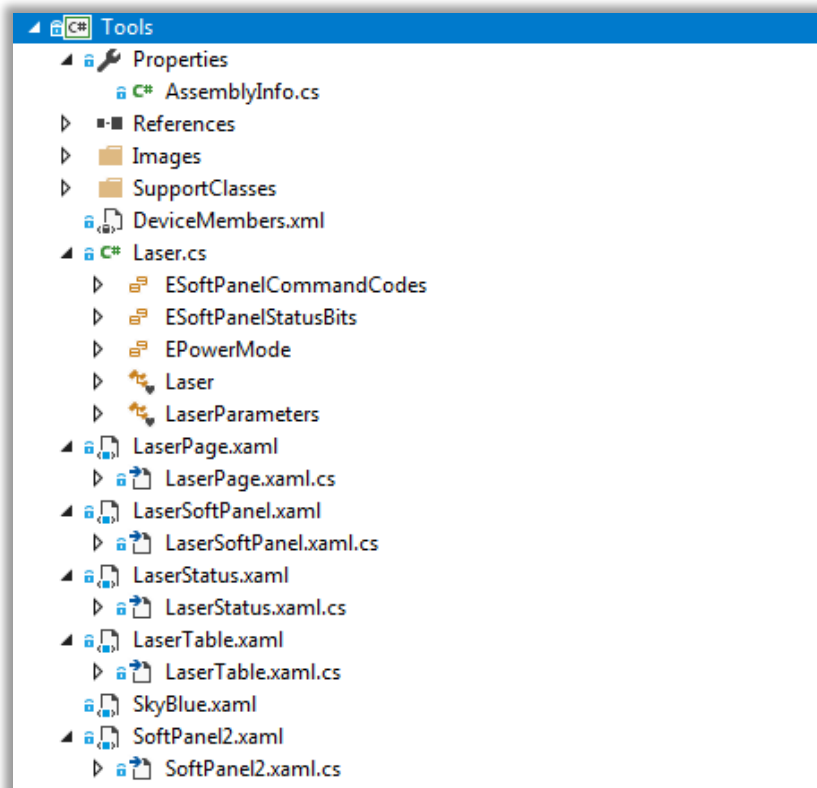
```
Terminal: Online [192.168.0.200 : SSH]
P850..852
P850=5.40000000000000036
P851=6.40000000000000036
P852=10.1999999999999993
```



	A	B	C	D
1	5.4	6.4	10.2	
2				

Tools:

This project is designed specifically for laser applications. This section of the manual is dedicated to explain this project in more details. This project can also be used by users and developers as a start development point to merge their customized software needs. As can be seen in the following figure, this project includes two soft panels (one made specific for laser applications and the other it's a custom one) , laser table (which includes T-Code, Speed, Power, Frequency, Height Offset, and Path Offset), and laser status.



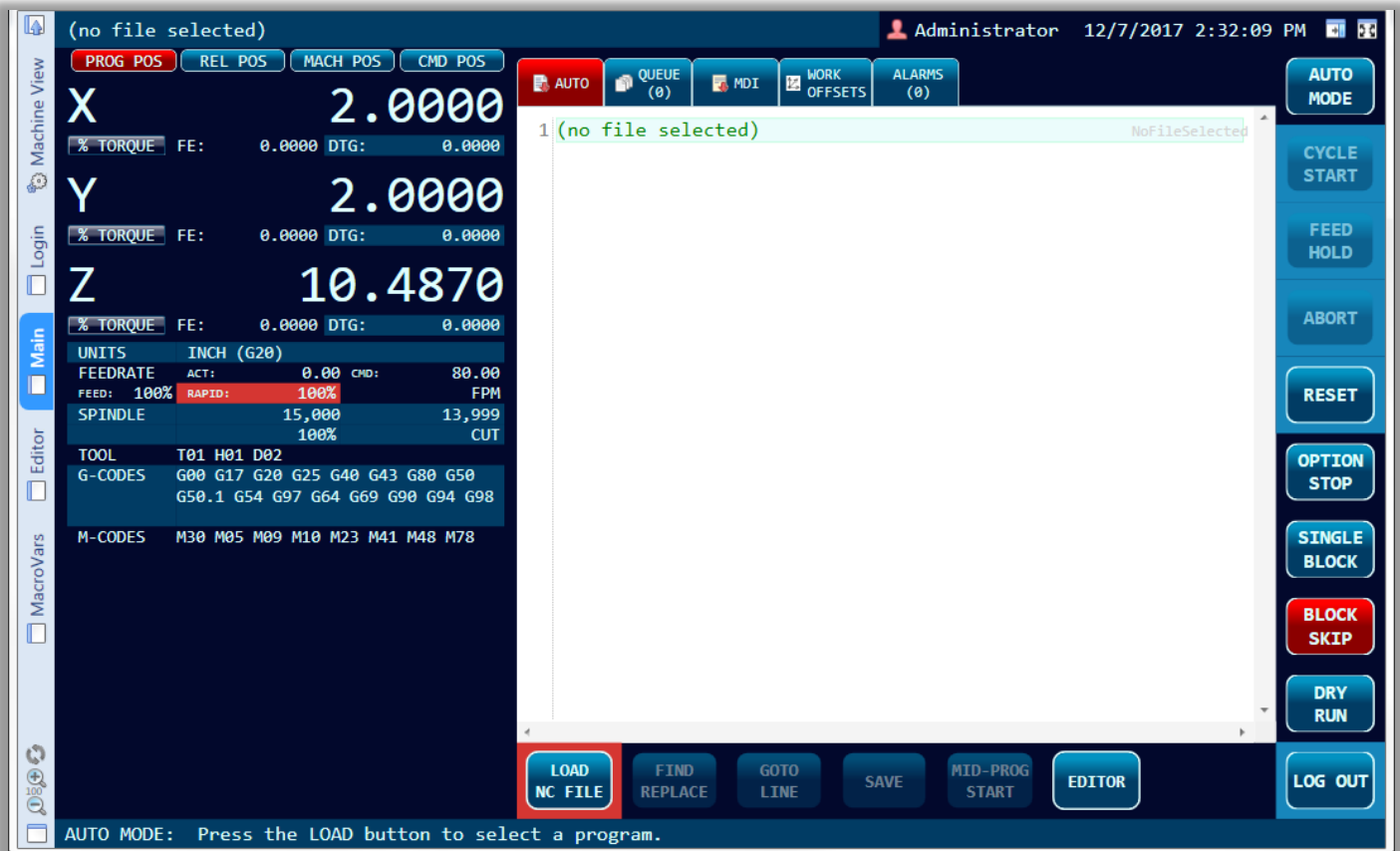
In order to enable this project including its features by adding the following to the "PowerpmacNC.ini" file as follow:

```
[Machine Constructor]
; TODO: Specify the machine type (Standard or Custom)
; The Custom machine type depends on components loaded from external assemblies. (See documentation.)
MachineType=Custom
```



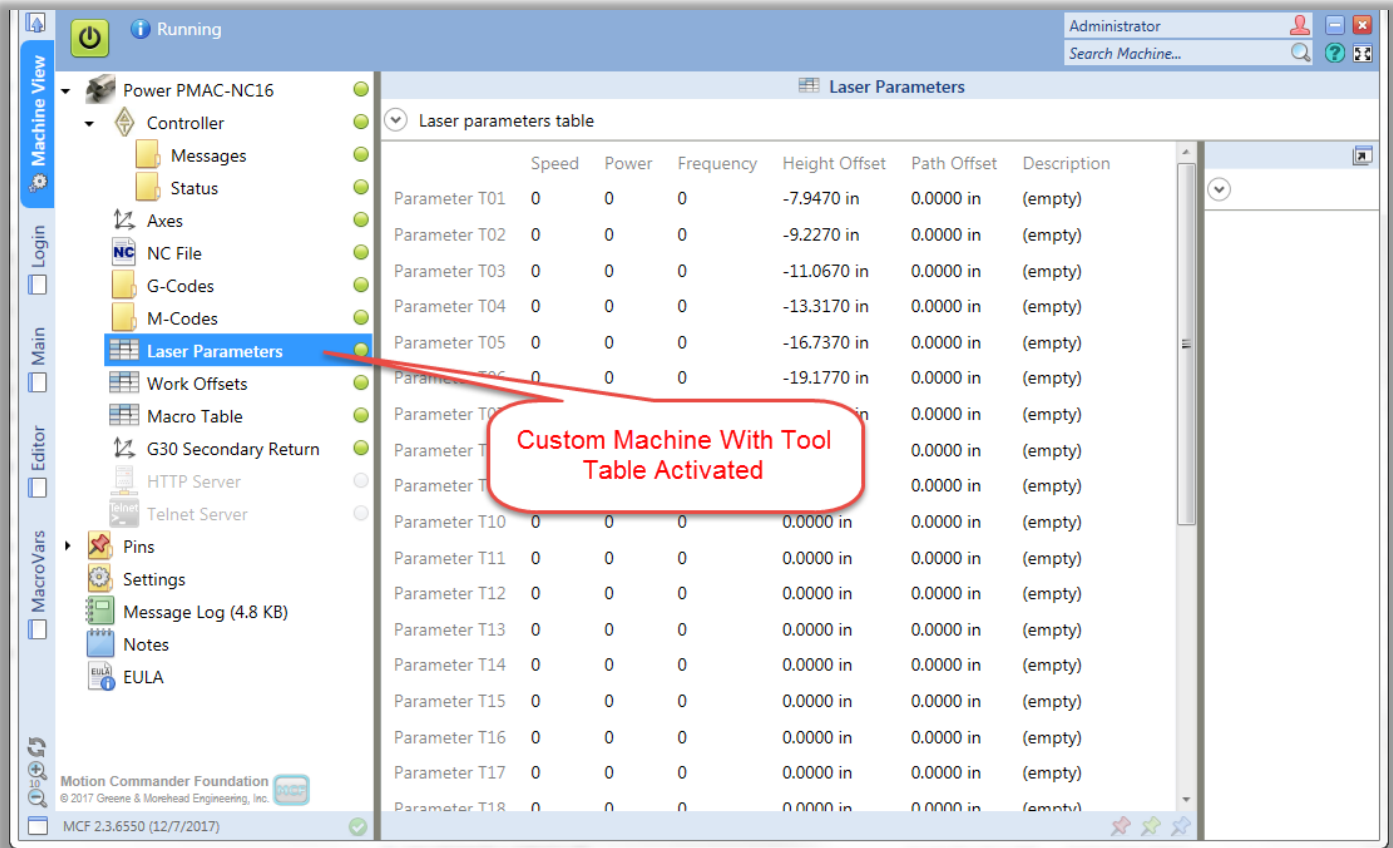
"Custom" machine will load PPNC16 basic features. Features which can be activated are soft panel, Tool offset table, and Tool Tab. Following figure is provided as a reference to show how "Custom" machine looks like:

Custom Machine:



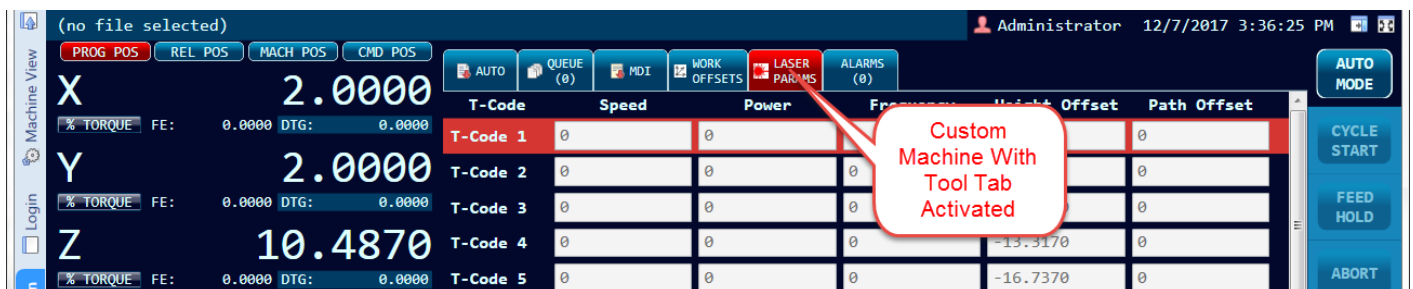
Tool table can be added as a subassembly to a custom machine by adding the following line to the "PowerpmacNC.ini" file:

```
ToolTable="..\..\..\Tools\bin\Debug\Tools.dll;Tools.LaserParameters"
```



Tool tab can be added to a custom machine by adding the following line to the "PowerpmacNC.ini" file:

```
Tool="..\..\..\Tools\bin\Debug\Tools.dll;Tools.Laser"
ToolTable="..\..\..\Tools\bin\Debug\Tools.dll;Tools.LaserParameters"
ToolTab="..\..\..\Tools\bin\Debug\Tools.dll;Tools.LaserTable"
```

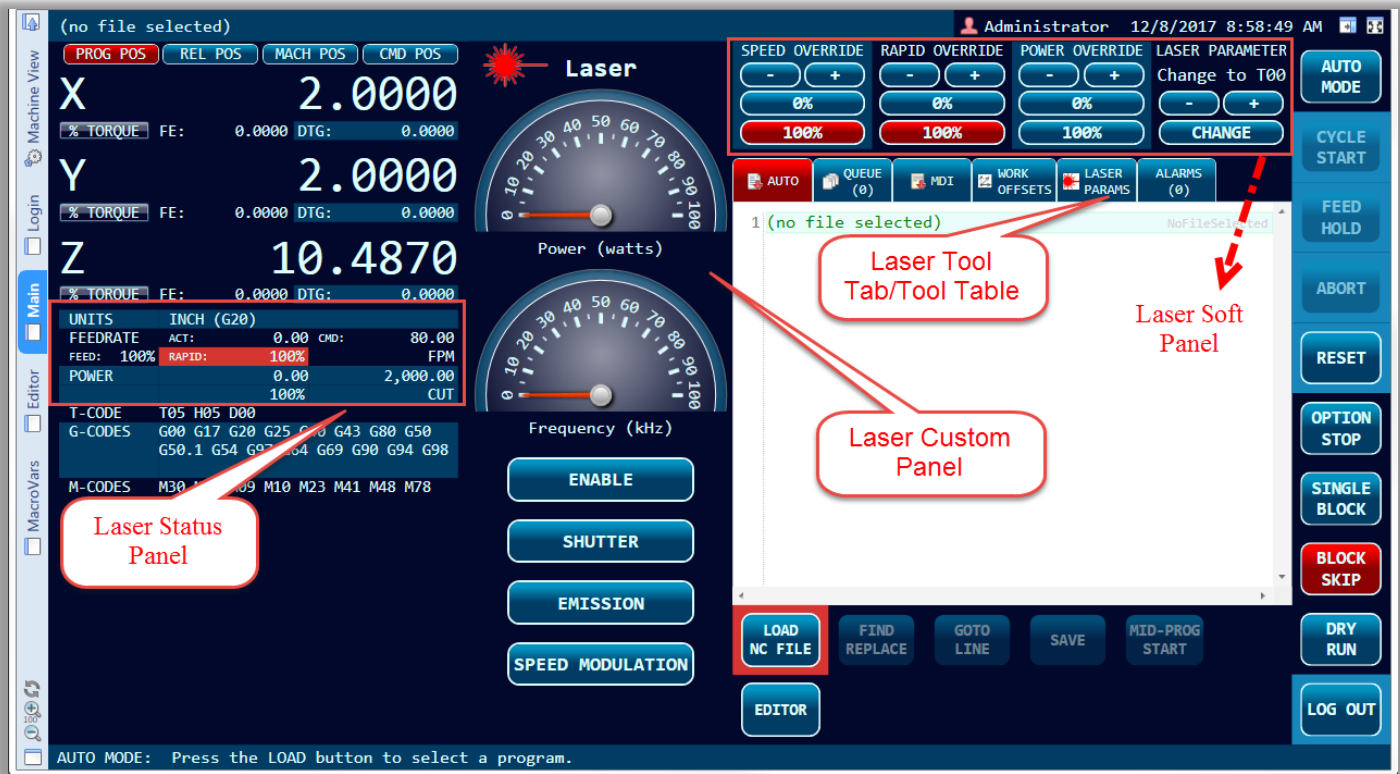


💡 In this project by default components are referenced in subassemblies. Therefore, "ToolTable" subassembly is the only one that can be activated as standalone subassembly in this project.

💡 Number of tool offsets is set to 100 by default. This project by default only supports 31 tools. Therefore, to activate this project, users are required to set "ToolOffsets" to 31. Any value higher than 31, will cause PPNC16 showing an error message during the start up.

In order to add “Tools” and “Soft Panel” features for laser applications, in addition to “MachineType = Custom” modification, add the following to the “PowerpmacNC.ini” file:

```
Tool="..\..\..\Tools\bin\Debug\Tools.dll;Tools.Laser"
ToolTable="..\..\..\Tools\bin\Debug\Tools.dll;Tools.LaserParameters"
SoftPanel="..\..\..\Tools\bin\Debug\Tools.dll;Tools.LaserSoftPanel"
StatusPanel="..\..\..\Tools\bin\Debug\Tools.dll;Tools.LaserStatus"
ToolTab="..\..\..\Tools\bin\Debug\Tools.dll;Tools.LaserTable"
CenterCustomFrame="..\..\..\Tools\bin\Debug\Tools.dll;Tools.LaserPage"
```



💡 Change the “CenterCustomFrame” to “LeftCustomFrame” or “RightCustomFrame” if it is desired to shift the panel to the left or right side of the “Main” page.

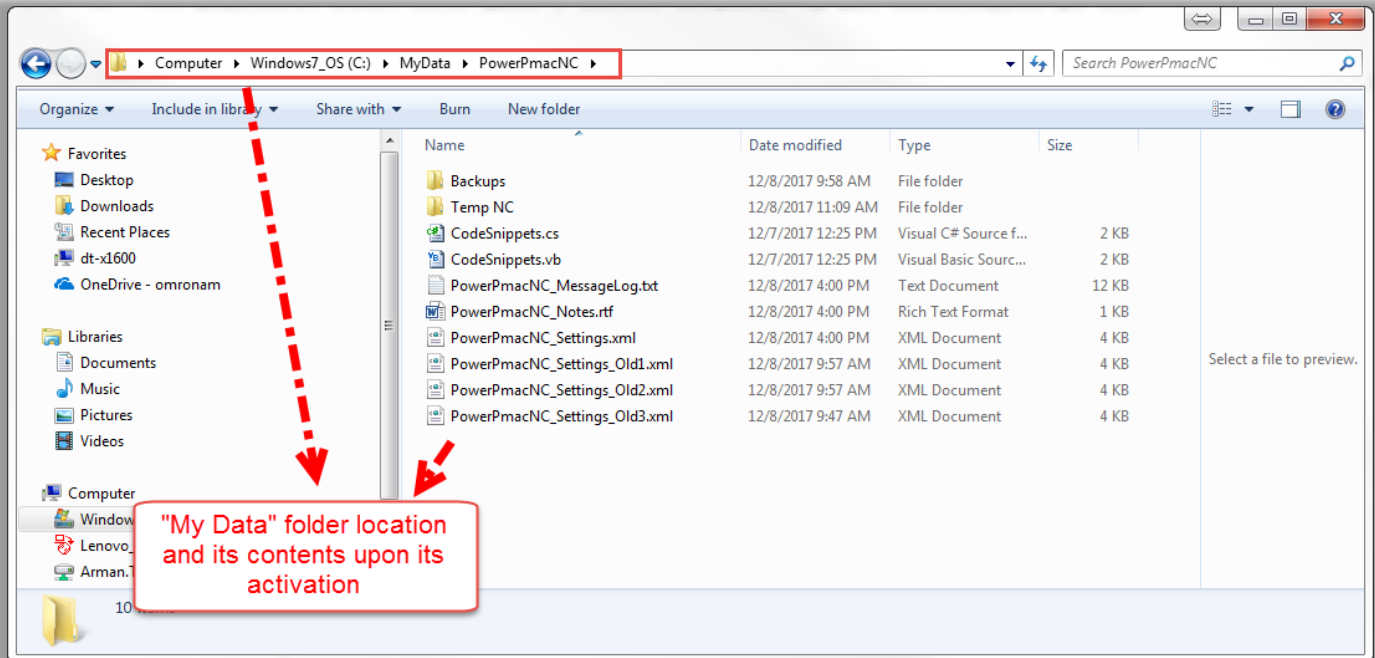
💡 If it is desired to add “Laser Page” as a custom tab in “Main” or as a user page, add the following “PowerpmacNC.ini” file:

```
UserPage="..\..\..\Tools\bin\Debug\Tools.dll;Tools.LaserPage"
CustomTab="..\..\..\Tools\bin\Debug\Tools.dll;Tools.LaserPage"
```

Data Folder:

“Data Folder” includes template folder for CNC files, message log files, notes, PPNC16 setting files, and some other miscellaneous files. PPNC16 by default, uses the folder which contains the executable file to save all mentioned files and folders. However, if it is desired, “PowerPmacNC.ini” file can be configured as follow to create a “Data Folder” according to the provided path:

```
; Optional: Specify a folder other than the exe directory for Settings, Messages, Notes, etc.  
;DataFolder="C:\MyData\PowerPmacNC"
```



HTTP Server, Telnet Server, and MT Connect:

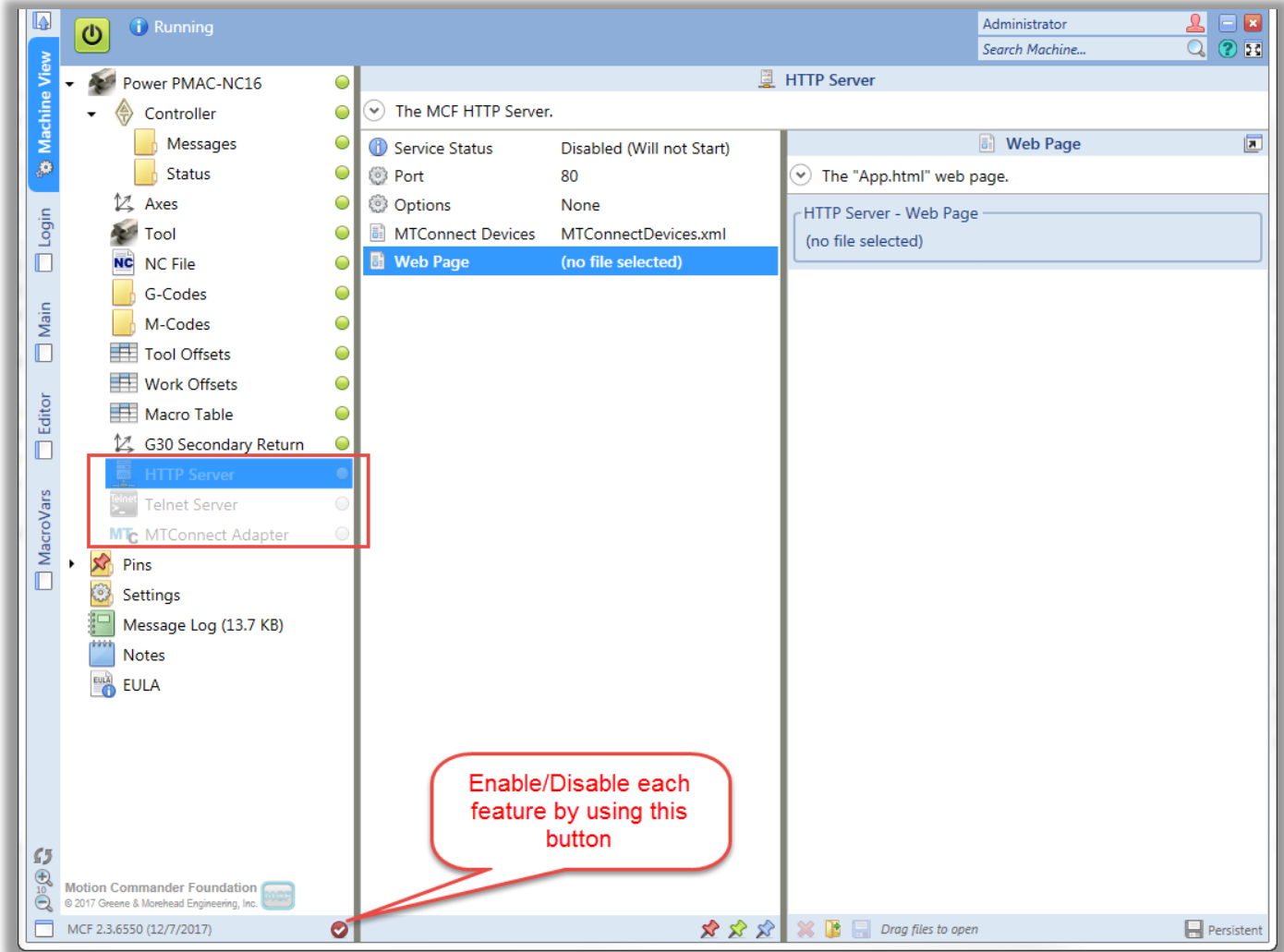
PPNC16 provides three different tools which make users capable of monitoring machine variables and states. Such supervisory interfaces can be synced with users' custom supervisory applications in order to fully monitor and control machines on a manufacture floor. Each tool can be activated by configuring the “PowerPmacNC.ini” file as follow:

```
; Optional: Add HTTP Server, Telnet Server or MTConnect Adapter support to the application.  
HttpServer=true  
TelnetServer=true  
MTConnectAdapter=true
```

💡 In order to activate the “MTConnect” properly “MTConnectAdapter” is required to be enabled by configuring the “PowerpmacNC.ini” file as follow:

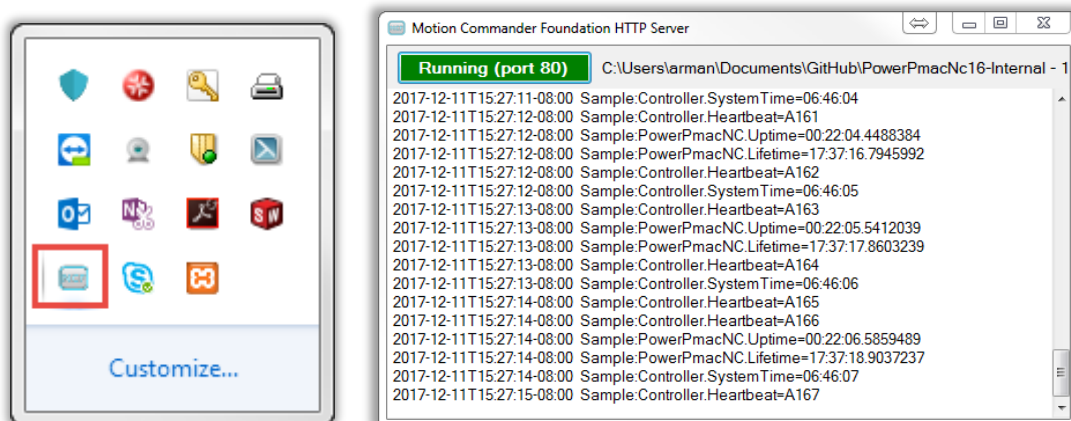
```
MTConnectAdapter="..\..\..\CustomAdapter\bin\Debug\CustomAdapter.dll;CustomAdapter.MyAdapter"
```


Following figure, demonstrates these three features:



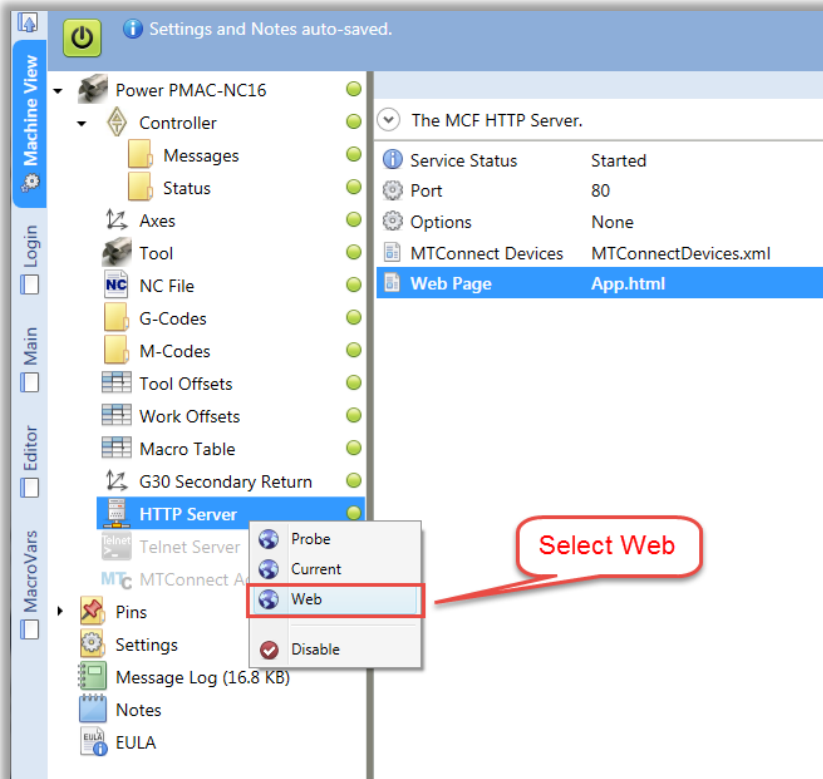
HTTP Server:

Enable the "HTTP Server" by either right clicking on it and click on "Enable" or highlighting the "HTTP Server" and click on a small button as it is shown in figure above. "Service Status" reports the status of "MCF Http Server" using a port number 80 by default. After "PowerPmacNC.ini" is configured and PPNC16 application is restarted, as the application loads up, a small icon which is for MCF HTTP Server, appears.

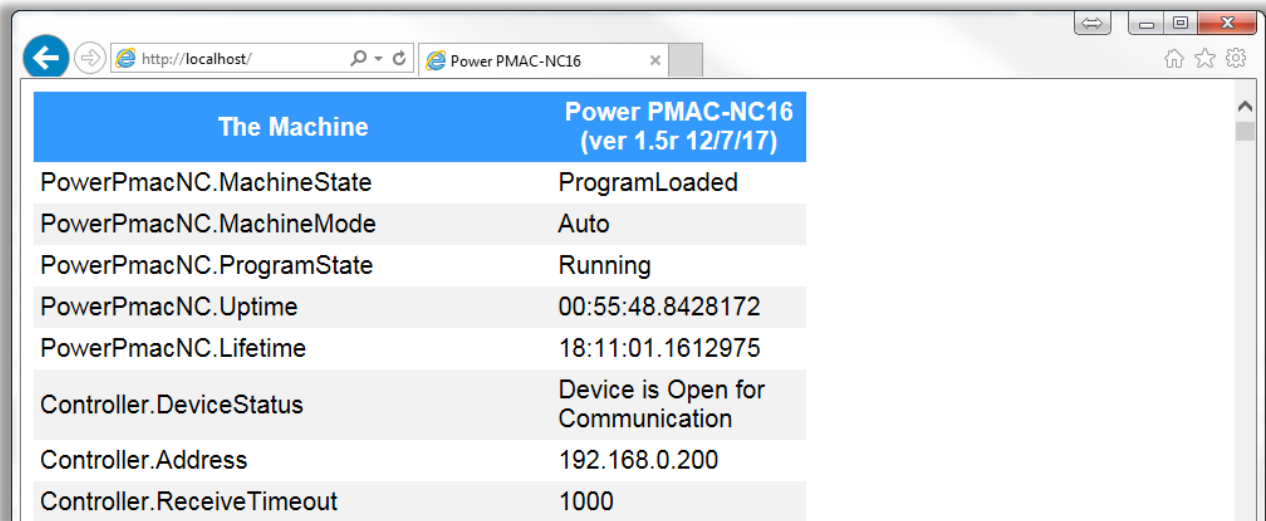


💡 If the MCF HTTP Server fails to start or stops working, the highlighted green button, shown above, will become red. Therefore, the MCF HTTP Server is required to be restarted.

After PPNC16 is initialized and no error is detected by MCF HTTP Server, do a right click on the HTTP Server and select “Web” as shown below:



By doing such an action, PPNC16 will open a new web page called <http://localhost/> using Internet Explorer as follow:

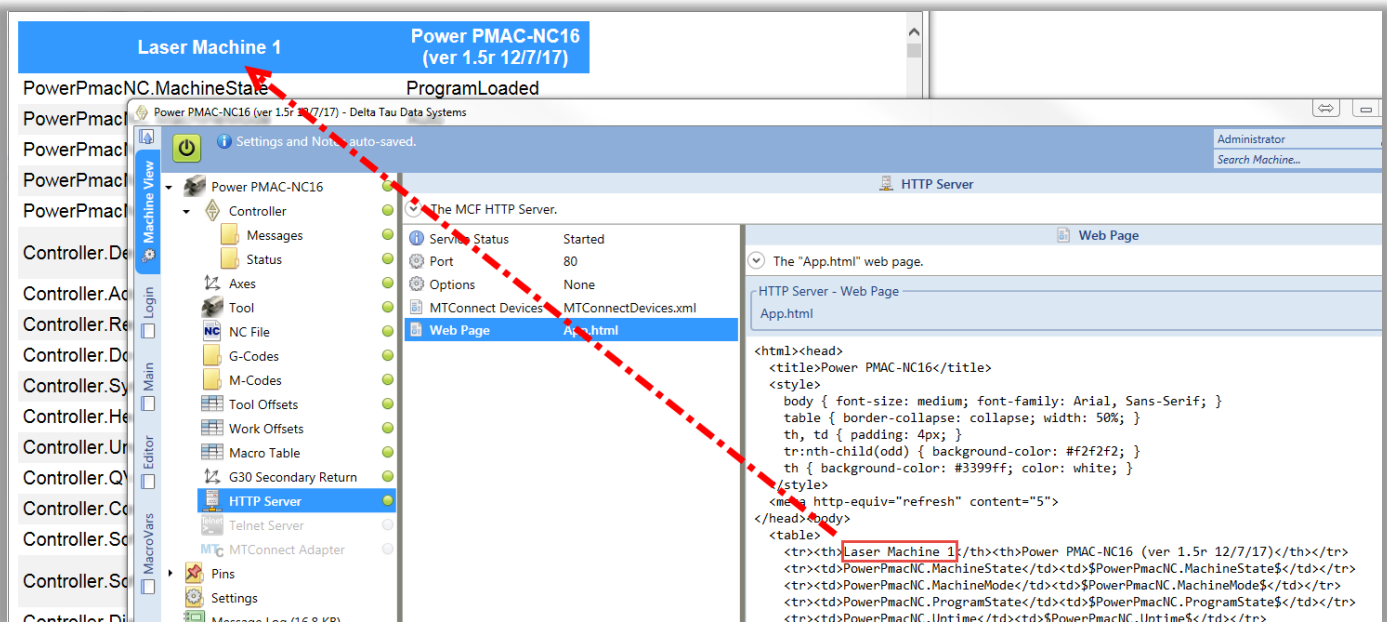


💡 It is intended that users develop their own custom page and implement such a feature in that custom page. The example shown above can be used as is or as a start point for further development. Desired members that are included in the “App.html” or a custom “index.html” file, also have to be included in the “MTConnectDevices.xml” file.

💡 By default a “page refresh” happens every 5 seconds. If it is desired to change the default value, simply change the following line in the “App.html” or a custom “index.html” file as follow:

```
<meta http-equiv="refresh" content="5">
```

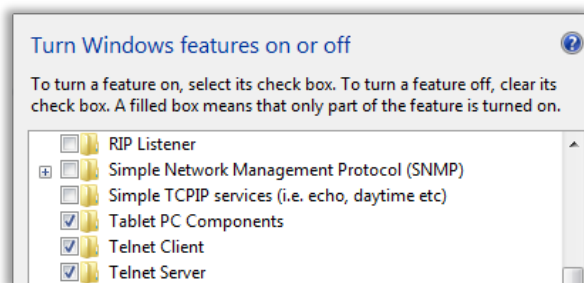
Such a unique feature allows users to change and test contents of a page as it is running in an interactive way. For example, if it is desired to change the machine name (The Machine by default), simply modify it to the desired name and **save** it using PPNC16 application as follow:



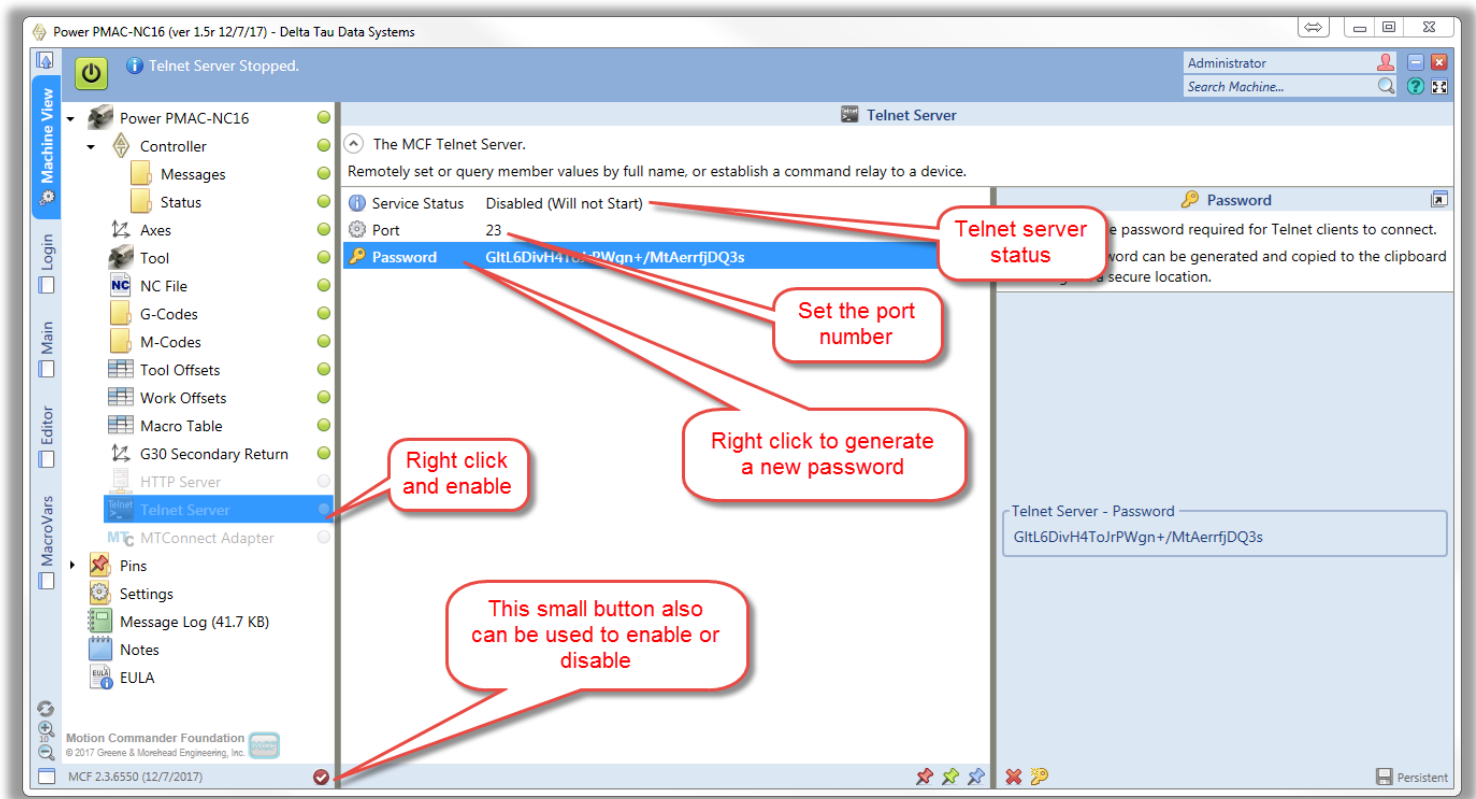
TelNet Server:

The MCF Telnet Server, implemented in PPNC16, enables users to remotely set or query member values by full name or establish a command relay to a device. To use this feature, standard windows “TelNet Client” is required to be installed on a host computer.

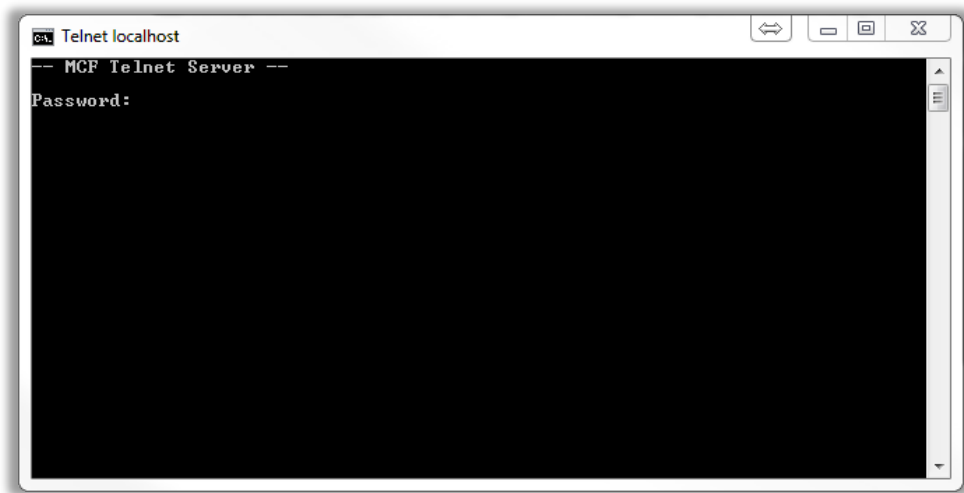
💡 To activate (install) “TelNet Client” use “Turn Windows features on or off” tool under the “Control Panel” in “Uninstall or change a program” section.



The intention of using such a feature is to use a supervisory application on a computer, tablet, or smartphone in order to remotely set or query member values in PPNC16. As an example, this section intends to show how this feature works by manually typing some commands and providing some figures. In order to enable Telnet Server, do a right click on its title and click “Enable” as follow:



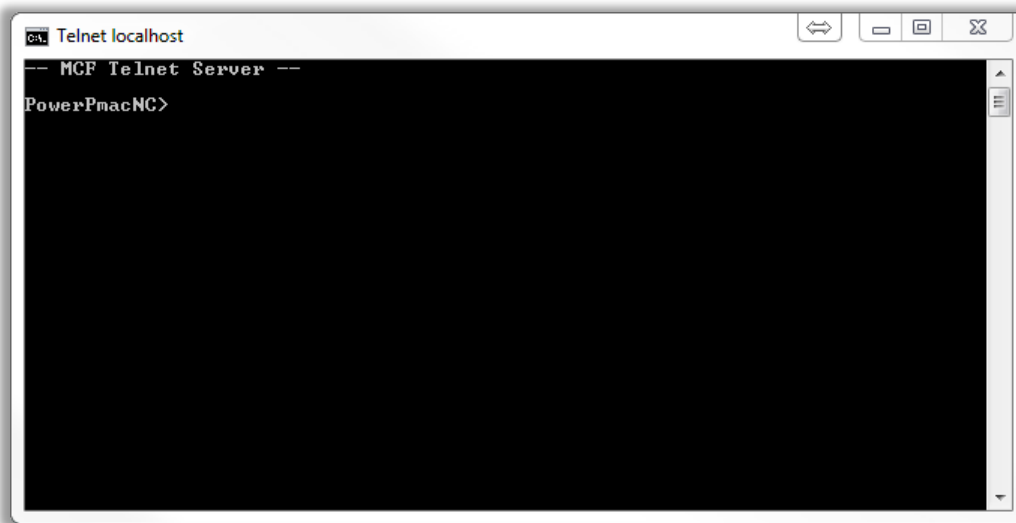
After enabling the MCF Telnet Server in PPNC16, run the “Command Prompt” and type “Telnet localhost”.If the port is open and connection is successful, MCF Telnet Server will required a password to connect to PPNC16 as follow:



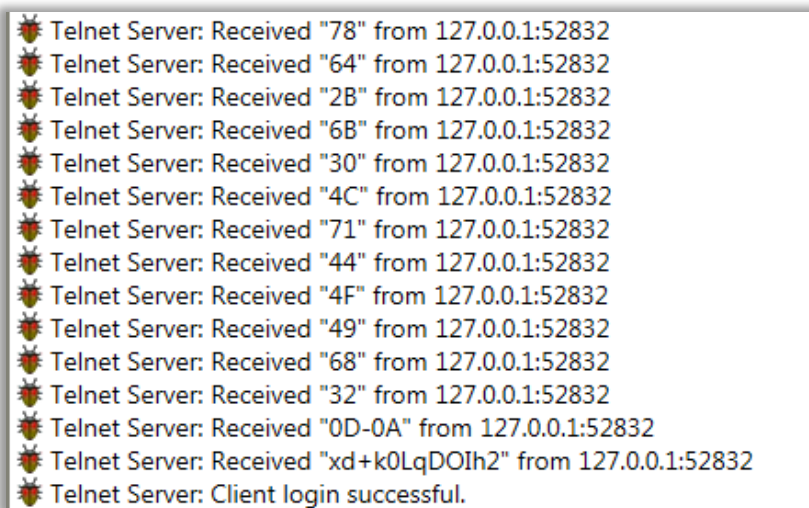
In order to copy the password, do a right click on the “Password” in PPNC16 Telnet Server section and click on generate a new password. Then confirmation window will show up as follow:



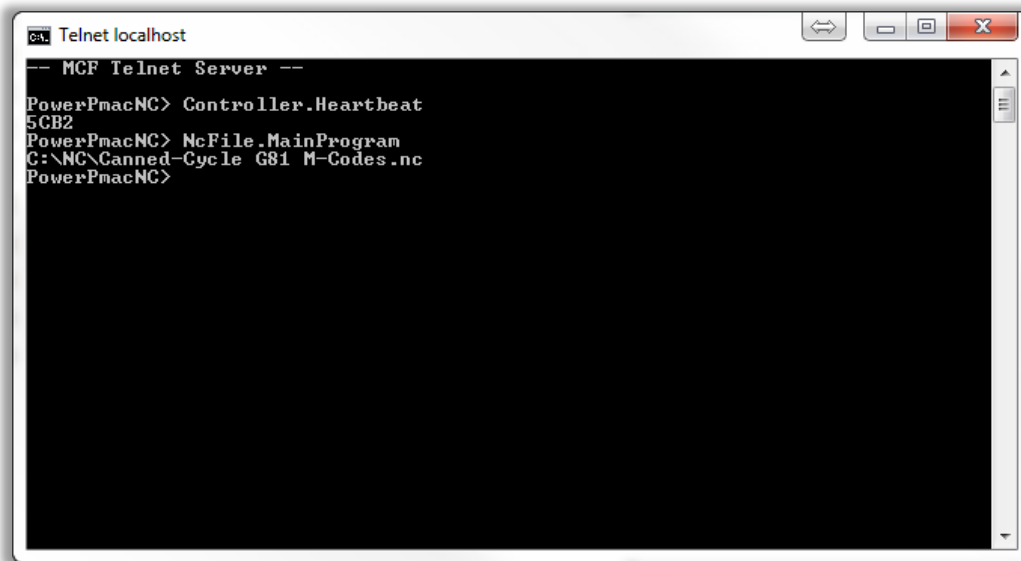
After confirming and creating a new password, do a right click on a command prompt and have it past there. If the password is correct, pressing “Enter” will show the following in Command Prompt window:



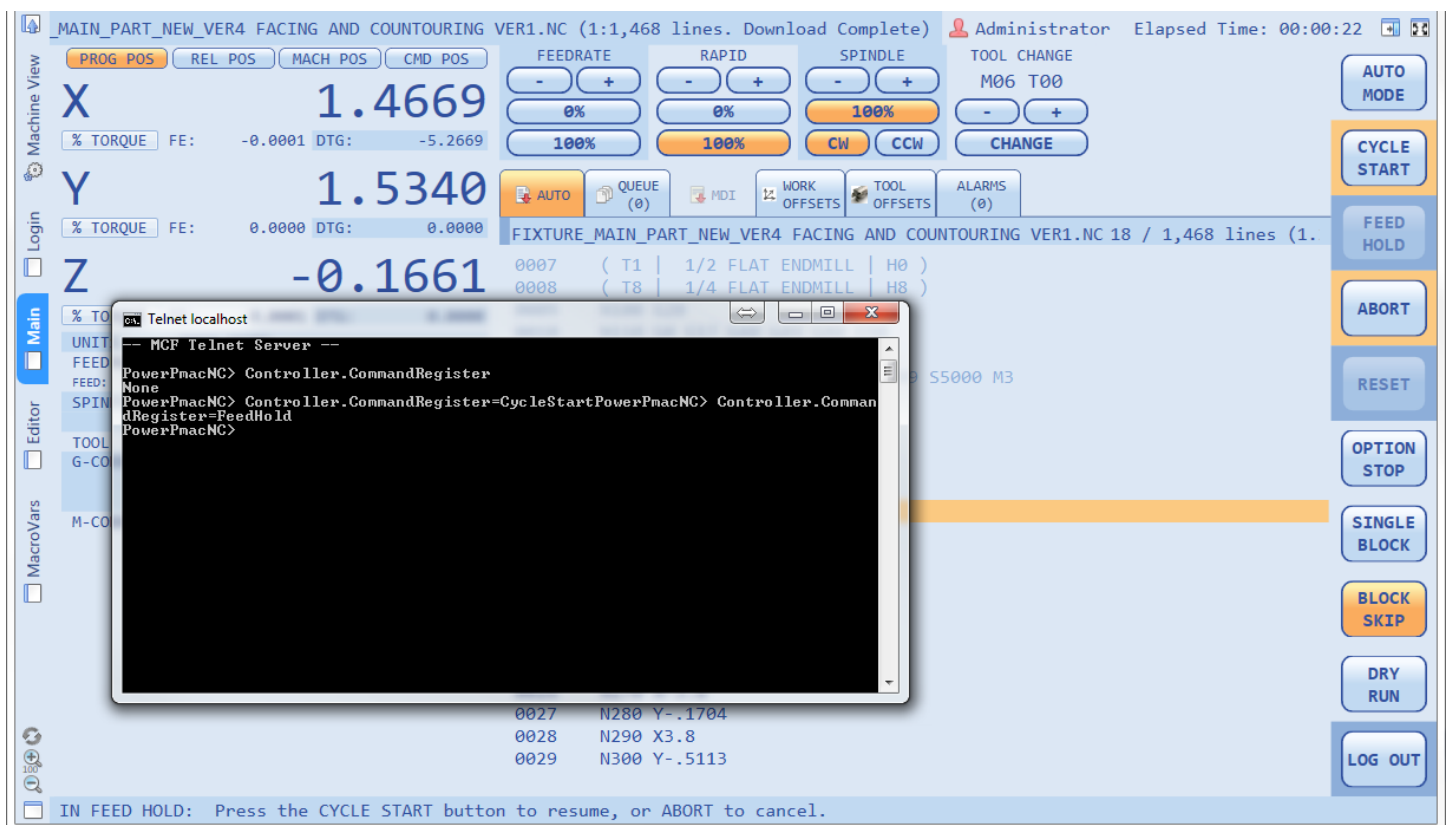
All activities done using Telnet Server, will be shown in a PPNC16 “Message Log” section.



If it is desired to query member values, simply execute a command with member’s full name. For example, type “Controller.Heartbeat” or “NcFile.MainProgram” in a command prompt window (as shown above) as follow to observe what each is set to:



If it is desired to set member values, execute a command that includes member's name equal to a desired value. For example, type "Controller.CommandRegister = CycleStart" and "Controller.CommandRegister = FeedHold" in a command prompt window while the program is loaded as follow to have PPNC16 start the program and have it on "feedhold" as follow:

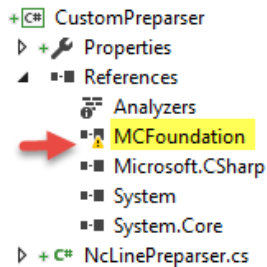


Unrecognized members, commands, and characters sent by Telnet show as alarms in PPNC16.

MTConnect Agent:

Customized Parsing Using the NC File Pre-Parser

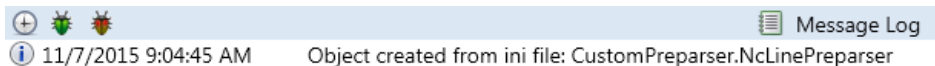
Make sure the project reference to *Motion Commander Foundation* is correct. Right-click on References and make sure the path to the MCFoundation.DLL is correct. The project should build successfully if the path is correct.



Open your NC16's "PowerPmacNC.ini" configuration file and add the following line to the [External Assemblies] section, adjusting the path to match the relative location of the DLL.

```
NcLinePreparser="..\..\..\CustomPreparser\bin\Debug\CustomPreparser.dll;CustomPre  
parser.NcLinePreparser"
```

Run NC16 and verify that the following message appears in the Message Log at startup:



Open "NcLinePreparser.cs" in Visual Studio and customize to your needs. Experience with C# string handling will be very helpful.

Telnet Server Support

You can open a Telnet session and send commands to the app as shown below. Use "device=(device name)" to connect directly to Power PMAC for a terminal session. The app may optionally specify a password that would be required from Telnet clients at login.

Consider automating a room full of machines via a Telnet session:


```
> MachineState
<   Ready
> NcFile.MainProgram=(your path)
> MachineMode=Auto
> Controller.CommandRegister=CycleStart
> MachineState
<   Running
```

Appendix A. The Source Files

GitHub\PowerPmacNcRelease

This folder will be copied to your PC when you “Clone” the repository from either the *GitHub* website or the *GitHub for Windows* application. You will want to use the *GitHub for Windows* application to periodically “Sync” to update your local copy to the most recently released version.



It is highly recommended that you make working copies of both the *PowerPmacNC* Visual Studio solution and the Power PMAC project in order to avoid losing your edits when you Sync. If a Sync fails for any reason, simply delete the entire “GitHub\PowerPmacNcRelease” folder and Clone again.

README.md, Banner.png, .gitattributes, .gitignore
The Git repository configuration files. Do not edit.

GitHub\PowerPmacNcRelease\PowerPmacNC
Source code for the Power PMAC-NC 16 C# application. (See detail below.)

GitHub\PowerPmacNcRelease\PMAC Source Code\PowerPMAC
The Power PMAC project that works together with the host PC application to run NC files. The *Delta Tau Power PMAC IDE* will be used to download this project to the controller.

GitHub\PowerPmacNcRelease\PMAC Source Code\TurboPMAC
The Turbo PMAC project that works together with the host PC application to run NC files. The *Delta Tau PEWin32-PRO2 PMAC Executive Program* will be used to download this project to the controller.

GitHub\PowerPmacNcRelease\PowerPmacNC Demo Build
A demonstration version of the Power PMAC-NC 16 program intended for marketing and training purposes.

GitHub\PowerPmacNcRelease\CustomExamples
This example C# project produces an external (plugin) DLL that can be loaded by the Power PMAC-NC 16 program to extend its functionality with custom operator panels, variables, and G/M-code groups.

GitHub\PowerPmacNcRelease\PowerPmacNC

Source code for the *Power PMAC-NC 16* application. This is a .NET 4.0/C#/WPF application based on the *Motion Commander Foundation* .NET framework. Visual Studio 2010 or newer (Express or Professional) can be used to build this project.

PowerPmacNC.sln, PowerPmacNC.csproj, DeltaTau.ico

The Visual Studio solution and project files, and the application icon.

`Main.cs`

The entry point for MCF-based applications, including the *Machine* definition.

`Enumerations.cs`

The enumerations used by the application.

`PowerPmacController.cs, TurboPmacController.cs`

Power PMAC and *Turbo PMAC* CNC controller support classes.

`DeviceMembers.xml`

The PMAC variable assignments and update rates for each device member in the *Machine* definition.

`GCodes.cs, MCodes.cs`

The standard G and M-code group definitions. Custom G and M-code groups should be added via external (plugin) DLL's instead of directly editing these files.

`PageLogin.xaml, PageLogin.xaml.cs`

The user login WPF page. Alternative background images may be specified in the application's configuration file.

`PageMain.xaml, PageMain.xaml.cs`

The main operator panel WPF page. Most of the process logic that operates the CNC machine is in the code-behind of this WPF page.

`Reference PowerPmacNC.ini`

The application reads the "PowerPmacNC.ini" configuration file in its exe directory at start-up to obtain its configuration data (machine type, axis definitions, units, and other important parameters). A *Reference* copy of this file is included in the project for convenience.

`Reference PowerPmacNC_Settings.xml`

The application saves the values of all persistent variables in the "PowerPmacNC_Settings.xml" file in its exe directory. A *Reference* copy of this file is included in the project for convenience.

`Reference Messages.xml`

The app includes three bitwise message registers (Fatal, Warning and Information) that the controller can use to display messages to the operator. The message strings are specified in the "Messages.xml" file in the exe directory. A *Reference* copy of this file is included in the project for convenience.

`SecureDongle_Control32.dll, SecureDongle_Control64.dll`

The hardware key (dongle) driver libraries for both 64-bit Windows and 32-bit Windows.

`GitHub\PowerPmacNcRelease\PowerPmacNC\Properties`

Standard C# project directory that contains assembly information.

`GitHub\PowerPmacNcRelease\PowerPmacNC\References`

This folder contains the MCF libraries and other DLL's required by the application.

`GitHub\PowerPmacNcRelease\PowerPmacNC\SupportClasses`

This folder contains C# and WPF support classes for the application.

`GitHub\PowerPmacNcRelease\PowerPmacNC\Skins`

This folder contains the user interface "Skins" WPF resource dictionaries.

GitHub\PowerPmacNcRelease\PowerPmacNC\Images

This folder contains image files for the application. Do not edit these images. Alternative background images may be specified in the application's configuration file.

GitHub\PowerPmacNcRelease\PowerPmacNC\Languages

This folder contains the foreign language files. MCF generates a language file the first time that a foreign language user logs in. This file may then be edited by a skilled human translator to refine the machine translations. For example, the German translation of FEEDRATE must be abbreviated to display correctly.

English: FEEDRATE

German: *VORSCHUBGESCHWINDIGKEIT*

Appendix B. The Configuration File

```
;
; "PowerPmacNC.ini" - Configuration file for the Power PMAC-NC16 program.
; This file is read by PowerPmacNC.exe at startup and must be in the exe directory.
; This file will NOT be overwritten by MCF and should be well commented.
;

[Machine Constructor]
; TODO: Specify the machine type (Standard or Custom)
; The Custom machine type depends on components loaded from external assemblies. (See documentation.)
MachineType=Standard

; TODO: Specify from one to ten axis labels separated by commas.
; Axis labels can be more than one character but they must be short. Suggest two characters max.
AxisLabels=X,Y,Z,A,B

; TODO: Specify motor numbers separated by commas (for status monitoring).
; The first motor number will be used to monitor the status of the first axis, etc.
MotorNumbers=1,2,3,4,5

; TODO: Specify the application's native length units (INCH or MM) and decimal places of precision (0-6).
NativeLengthUnits=MM
NativeLengthDecimalPlaces=3

; TODO: Specify the time units to display in velocity labels (min, sec, etc)
VelocityTimeUnits=min

; TODO: Specify quantity of tool offsets (0 min, 100 max)
ToolOffsets=100

; TODO: Specify quantity of G54.1 work offsets (0 min, 100 max)
G541=100

; Optional: List G and M-code group names that are NOT required by the application (separated by commas).
; Note: Group0, Group6, ProgramGroup and SubprogramGroup may not be removed.
;ExtraneousGroups=Group11,Group22,ThreadingGroup,GearRangeGroup,BAxisGroup
```

```

; Optional: Allow more than a single instance of the application to run.
;AllowMultipleInstances=true

; Optional: Add HTTP Server, Telnet Server or MTConnect Adapter support to the application.
;HttpServer=true
;TelnetServer=true
;MTConnectAdapter=true

[User Interface]
; Option to hide the Feedrate/Spindle/Tool Change controls on the main screen.
HideUpperControlPanel=false
; Option to hide the NC file queue tab on the main screen.
HideQueueTab=false
; Specify either three or five jog speed buttons to match the pendant.
ThreeJogSpeeds=false
; Option to display tool offset descriptions in the main screen tab.
ToolOffsetDescriptions=false
; Option to hide the "More..." button on the large format file open dialog.
HideMoreButton=false
; Add diagnostic screens of up to six members each.
; MembersPage=Title,MemberFullName1,MemberFullName2,...
;MembersPage=Status,Controller.Status.CoordStatus0,Controller.Status.CoordStatus1,Controller.Status.Motor1Status
0

[NC Files]
CoordinateSystem=1
SubprogramFolder="C:\NC"
; NC programs are parsed and downloaded to PMAC subprog buffer ('O' program number + SubprogramBaseAddress)
SubprogramBaseAddress=5000
SubprogramBaseAddressMDI=6000
; "Volatile" NC programs are downloaded to PMAC along with the main NC program.
VolatileSubprogramMin=0
VolatileSubprogramMax=899
; "Non-volatile" NC programs are downloaded to PMAC by a utility and saved.
; To disallow non-volatile subprograms, set range to (0,-1)
NonvolatileSubprogramMin=900
NonvolatileSubprogramMax=999
; Optional "open subprog" parameters (buffer number or *, local variable stack offset, max jump labels).

```

```

;SubprogOpenParams=5000,256,1024

;SubprogOpenParams=*,256,1024


[Parser Options]

; TODO: Specify the level of macro support (None, Simple, Parametric)

; No macros, simple #define macro substitutions (the default), or parametric programming.
MacroSupport=Simple

; Base address for macro variables #1-#999 (5000 will map #1 to PMAC variable P5001)
MacroVariableBaseAddress=5000

; NC Parser Options default to false. Uncomment the desired options to change to true.

;IgnoreFixedCycles=true

; Fixed Cycle allowed range is G70-G89. G80 always cancels.

; Fixed Cycle G-codes may be specified in a range "G73-G89" or as a comma-separated list.

;PrimaryFixedCycles=G73-G89

;SecondaryFixedCycles=G70,G71,G72,G90,G91,G98,G99

;FCodesAtEnd=true

;G09AtEnd=true

; Do not add dwell0 to lines that include expressions.

;NoLookaheadSuppression=true

; Surround blocks with BSTART and BSTOP.

;BlockStartStop=true

;OnlyAllowedGMCodes=true

; Allowed G/M-codes appear in Machine View plus these optional comma-separated lists.

;AllowedGCodes=G5426,G5427

;AllowedMCodes=M17,M18


[External Assemblies]

; TODO: Specify the assembly path and type name in quotes separated by a semicolon

; Object="path to DLL;FObject class name"

; MainPage="path to DLL;Page class name"

; UserPage="path to DLL;Page class name"

; CustomTab="path to DLL;Page class name"

; LeftCustomFrame="path to DLL;Page class name"

; CenterCustomFrame="path to DLL;Page class name"

; RightCustomFrame="path to DLL;Page class name"

; CodeGroups="path to DLL;class name"

; NcLinePreparser="path to DLL;class name"

```

```
; MTConnectAdapter="path to DLL;class name"

; Example:  MainPage="..\..\..\..\MySolution\MyProject\bin\Debug\MyProject.dll;MyNamespace.MyMainPage"

; Example:  Object="..\..\..\..\MySolution\MyProject\bin\Debug\MyProject.dll;MyNamespace.MyFObjectClass"

; Example:  CenterCustomFrame="..\..\..\..\MySolution\MyProject\bin\Debug\MyProject.dll;MyNamespace.MyWpfPage"

; Example:  CodeGroups="..\..\..\..\MySolution\MyProject\bin\Debug\MyProject.dll;MyNamespace.MyClass"

; Example:  MTConnectAdapter="..\..\..\..\CustomAdapter\bin\Debug\CustomAdapter.dll;CustomAdapter.MyAdapter"
```

[Private Label]

; Optional private labeling. Images should be PNG or JPEG format and must be in the exe directory.

; Splash image should be around 500x300 pixels and login image should be around 1000x700.

```
;CompanyName="My Company Name"
```


```
;SplashImage="MySplashImage.png"
```

```
;LoginImage="MyLoginImage.jpg"
```


Appendix C. Turbo PMAC Support

If your version of the Power PMAC-NC 16 program includes Turbo PMAC support then you may specify that controller in the application's "PowerPmacNC.ini" file.

```
[Machine Constructor]
; TODO: Select the controller (PowerPmacController, TurboPmacController or MockController)
Controller=TurboPmacController
```

 The Delta Tau Pro2 (*PcommServer*) communications library must be installed on the PC and properly configured. If your Turbo PMAC is not device 0 then the device number may be changed in the Power PMAC-NC 16 program by selecting the **Controller** object in *Machine View*.

The Turbo PMAC Project

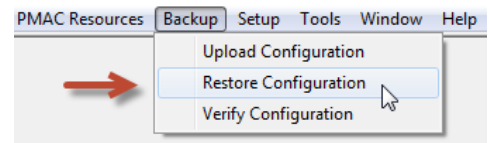
The "TurboPMAC\Configuration" folder contains a configuration file that can be downloaded to prepare your Turbo PMAC for use with the Power PMAC-NC 16 program.

"GitHub\PowerPmacNcRelease\PMAC Source Code\TurboPMAC\Configuration\TurboNcPlus_Build.cfg"

Open this file in a text editor and uncomment the "tpnc_umacdemobox.cfg" include line for a Demo Box test system or "tpnc_virtualmotors.cfg" for a controller-only test setup running software-simulated motors.


```
//Comment in for UMAC Demo Box
// #Include "tpnc_umacdemobox.cfg"
//Comment in for virtual motors 1-8
// #Include "tpnc_virtualmotors.cfg"
```

Run the *PEWin32-PRO2 PMAC Executive Program* and use the Terminal window to issue a "\$\$\$***" command to initialize the controller, then select the "Configure|M-variables" menu option and click the "Download Suggested M-variables" button. Now select the "Backup|Restore Configuration" menu option to download the "TurboNcPlus_Build.cfg" configuration file.



Verify no warnings or errors in the Results window.

```
Device 0-> Total Warnings: 0
Device 0-> Total Errors: 0
Device 0-> END.
```

 After downloading the configuration, use the Terminal window to issue a "**SAVE**" command to copy to nonvolatile flash memory, then issue a "\$\$\$" command to reset the controller.

The Turbo PMAC controller is now ready to work with the Power PMAC-NC 16 program!

Appendix D. Source Code Exclusions

Included in the Source Code version:

- The WPF operator screens (Login, Main, and future screens) and associated logic (code-behind).
- The logic that defines the operation of the NC application.
- The device member definitions and G and M-code definitions.
- CS and Motor status monitoring.
- The NC file editor (AvalonEdit).
- Application skins support.
- Initialization file configuration support.
- The Mock Controller.
- Example External Assembly projects that demonstrates how to add custom panels and device members.

NOT Included in the Source Code version:

- The MCF source code (Machine View, runtime engine, logging, alarming, foreign language support, MTConnect, etc)
- The low-level Power PMAC communications (sending commands, downloading files, etc)
- The NC file parsing (support for execution monitoring, subprograms, mid-tape start, etc)

Appendix E. Send1 Command List

The following is a list of the “send1” commands used by the Power PMAC to communicate status and requests to the PPNC16:

```
send1 "initialized"  
// HMI Handshake - Send after initialization code has completed.  
  
send1 "resetcompleted"  
// HMI Handshake - Send after reset sequence code has completed.  
  
send1 "homecompleted"  
// HMI Handshake - Send after initialization code has completed.  
  
send1 "cyclestarted"  
// HMI Handshake - Send after cycle start.  
  
send1 "infeedhold"  
// HMI Handshake - Send after feedhold.  
  
send1 "programcompleted"  
// HMI Handshake - Send after program completes (See M2 and M30).  
  
send1 "programfailed"  
// HMI Handshake - Send after a program fails for any reason.  
  
send1 "programaborted"  
// HMI Handshake - Send after an operator issued abort.  
  
send1 "estoppressed"  
// HMI Status Request – Puts HMI into Estopped condition.  
  
send1 "estopreleased"  
// HMI Status Request – Puts HMI into Estop Clear condition.  
  
send1 "workoffsetsset"  
// HMI Read/Write Request – Initiates read of CS offset variables and writes to xml settings file.  
  
send1 "tooloffsetsset"  
// HMI Read/Write Request – Initiates read of tool offset variables and writes to xml settings file.  
  
send1 "pendantconnected"  
// HMI Status Request – Hardware Pendant connected, hide soft panel.  
  
send1 "pendantdisconnected"  
// HMI Status Request – Hardware Pendant not available, show soft panel.  
  
send1 "hidemanual"  
// HMI Status Request – Do not show Manual Mode Tab (soft jog screen).  
  
send1 "showmanual"  
// HMI Status Request – Show Manual Mode Tab (soft jog screen).  
  
send1 "manualsubmodenone"  
// HMI Status Request – Show “Manual Mode” only in mode button.  
  
send1 "manualsubmodecontinuous"  
// HMI Status Request – Show “Manual Mode - Cont” in mode button.
```

```

send1 "manualsubmodehandle"
// HMI Status Request – Show “Manual Mode - Hand” in mode button.

send1 "manualsubmodehome"
// HMI Status Request – Show “Manual Mode - Home” in mode button.

send1 "requestautomode"
// HMI Request – Change to Auto Mode.

send1 "requestmanualmode"
// HMI Request – Change to Manual Mode.

send1 "requestmdimode"
// HMI Request – Change to MDI Mode.

send1 "requestcyclestart"
// HMI Request – Request Cycle Start.

send1 "requestfeedhold"
// HMI Request – Request Feedhold.

send1 "requestabort"
// HMI Request – Request Abort.

send1 "requestreset"
// HMI Request – Request Reset.

send1 "requestoptionstop"
// HMI Request – Toggle Option Stop.

send1 "requestsingleblock"
// HMI Request – Toggle Single Block.

send1 "requestblockskip"
// HMI Request – Toggle Block Skip.

send1 "requestspindlecw"
// HMI Request – Toggle Spindle CW.

send1 "requestspindleccw"
// HMI Request – Toggle Spindle CCW.

send1 "requesttoolchangeplus"
// HMI Request – Increment Manual Tool Change positive.

send1 "requesttoolchangeminus"
// HMI Request – Increment Manual Tool Change negative.

send1 "requestjogspeed1"
// HMI Request – Select Jog Speed1.

send1 "requestjogspeed2"
// HMI Request – Select Jog Speed2.

send1 "requestjogspeed3"
// HMI Request – Select Jog Speed3.

send1 "requestjogspeed4"
// HMI Request – Select Jog Speed4.

```

```

send1 "requestjogspeed5"
// HMI Request – Select Jog Speed5.

send1 "requestjog1"
// HMI Request – Request Axis 1 jog.

send1 "requestjog2"
// HMI Request – Request Axis 2 jog.

send1 "requestjog3"
// HMI Request – Request Axis 3 jog.

send1 "requestjog4"
// HMI Request – Request Axis 4 jog.

send1 "requestjog5"
// HMI Request – Request Axis 5 jog.

send1 "requestjog6"
// HMI Request – Request Axis 6 jog.

send1 "requestjog7"
// HMI Request – Request Axis 7 jog.

send1 "requestjog8"
// HMI Request – Request Axis 8 jog.

send1 "requestjog9"
// HMI Request – Request Axis 9 jog.

send1 "requestjog10"
// HMI Request – Request Axis 10 jog.

send1 "requesthome"
// HMI Request – Request Home Sequence.

send1 "jogging"
// HMI Status Request – Disable manual controls while incremental jogging.

send1 "jogstopped"
// HMI Status Request – Send jog stopped status.

send1 "canceled"
// HMI Handshake – Status Bar operation cancel sequence complete.

send1 "fatalmessage=xxxx"
// HMI Fatal Ack Message – Sends xxxx text to HMI as fatal message.

send1 "warningmessage=xxxx"
// HMI Warning Ack Message – Sends xxxx text to HMI as warning message.

send1 "informationmessage=xxxx"
// HMI Information Ack Message – Sends xxxx text to HMI as information message.

send1 "logmessage=xxxx"
// HMI Log Only Message – Sends xxxx text to HMI as Log Only message.

```

Appendix F. Included G & M Codes

The following is a list of G & M codes included with the default project. It is important to note the user can add custom G & M codes as necessary to complete their requirements.

```
// G00 Rapid move mode declaration
// G01 Linear move mode declaration
// G02 Clockwise circle move mode declaration
// G03 Counter Clockwise circle move mode declaration
// G04 Dwell for time of F, P, or X value in seconds
// G09 Exact stop (non-modal)
// G17 XY plane declaration for circles and radius comp
// G18 ZX plane declaration for circles and radius comp
// G19 YZ plane declaration for circles and radius comp
// G20 Set English (inch) mode
// G21 Set metric (mm) mode
// G28 Return to Reference Point
// G40 Cutter radius compensation cancel
// G41 Cutter radius compensation on left
// G42 Cutter radius compensation on right
// G43 Tool Length Offset
// G44 Tool Length Offset (Minus Direction)
// G49 Tool Length Compensation Cancel
// G50 Cancel scaling
// G50.1 Disable mirroring
// G51 Set scaling factors
// G51.1 Enable mirroring
// G52 Set Local Coordinate System
// G53 Set Machine Coordinate System
// G54 - G59 Set Work Coordinate System
// G54.1 Px set auxiliary work coordinate system offsets
// G61 set exact stop mode
```

// G64 set cutting mode

// G68 coordinate system rotation

// G69 Cancel rotation

// G70 Bolt-Hole Circle

// G71 Bolt-Hole Arc

// G72 Bolt-Hole Line

// G73 High Speed Peck Drilling Cycle - Short Retract

// G74 Left Hand Float Tapping

// G76 Fine Boring Cycle

// G80 Canned Cycle Cancel

// G83 Peck Drilling Cycle - Long Retract

// G84.2 Rigid Tapping – Right Hand Thread

// G84.3 Rigid Tapping – Left Hand Thread

// G85 Fine Boring Cycle – No Dwell

// G86 Boring Cycle (Spindle On Feed-In / Spindle Off Feed-Out)

// G87 Back Boring Cycle

// G89 Boring Cycle – With Dwell

// G90 absolute move mode

// G90.1 absolute move mode (IJK Abs Arc Center)

// G91 incremental move mode

// G91.1 incremental move mode (IJK Inc Arc Center)

// G92 position set mode

// G93 Inverse Time

// G94 Feed Per Minute

// G98 Return Initial Level

// G99 Return R Level

// M0 - Feed Hold
// M1 - Option Stop
// M2 - Stop with Rewind
// M3 - Spindle CW
// M4 - Spindle CCW
// M5 - Spindle Stop
// M6 - Tool Change
// M7 – Coolant Mist ON
// M8 – Coolant Flood ON
// M9 – Coolant OFF
// M30 - Stop with Rewind
// M98 - Sub-Program Call
// M99 - Return From Sub-Routine