



DELTA TAU

Power PMAC-NC 2016

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



Quick Start Manual

Power PMAC-NC16



Omron Delta Tau Data Systems, Inc.

April, 2016

O017-E-01

COPYRIGHT INFORMATION

Software: © 2016 Omron Delta Tau Data Systems, Inc. All rights reserved.

Software User Manual: © 2016 Omron 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	6/16/2015	Initial release
2.0	4/18/2016	Second Release – Version 2.
2.0.01	2/27/2017	Added SDK First Build Section.

Power PMAC-NC16™ - Quick Start Manual

Contents

Introduction	5
Requirements.....	6
Installation	6
Advantages and Functionality of GitHub	7
The Power PMAC-NC 16 Application	8
Power PMAC-NC 16 Project Folder Contents	8
Working With Power PMAC NC Software.....	8
Using GitHub to Clone the Project	8
Using GitHub to Update your Project	9
Using GitHub to Revert Back to a Previous Version.....	9
Power Pmac Source Code	10
Downloading the Power PMAC Project	12
Power PMAC-NC 16 HMI Code	13
Hardware Key.....	15
Power NC Software Levels	15
Quick Start – Get your NC HMI Running	16
Overview	16
Target PC System Setup	16
Power PMAC Setup	16
Creating a Virtual Motor Configuration	17
Downloading a Configuration File to PMAC.....	17
Power PMAC NC Software Setup	19
Running the Machine	22
What Next ?	24
Pmac Logic	24
Power NC HMI Settings	24
Target Machine Deployment	25
The PMAC Code	25
The HMI Code	25
Deployment Summary	26


The Power PMAC NC SDK Package – Building Your Project for the First Time 27

Appendix A - Power PMAC Virtual Motors Configuration File Sample..... 30

Power PMAC-NC16™ - Quick Start Manual

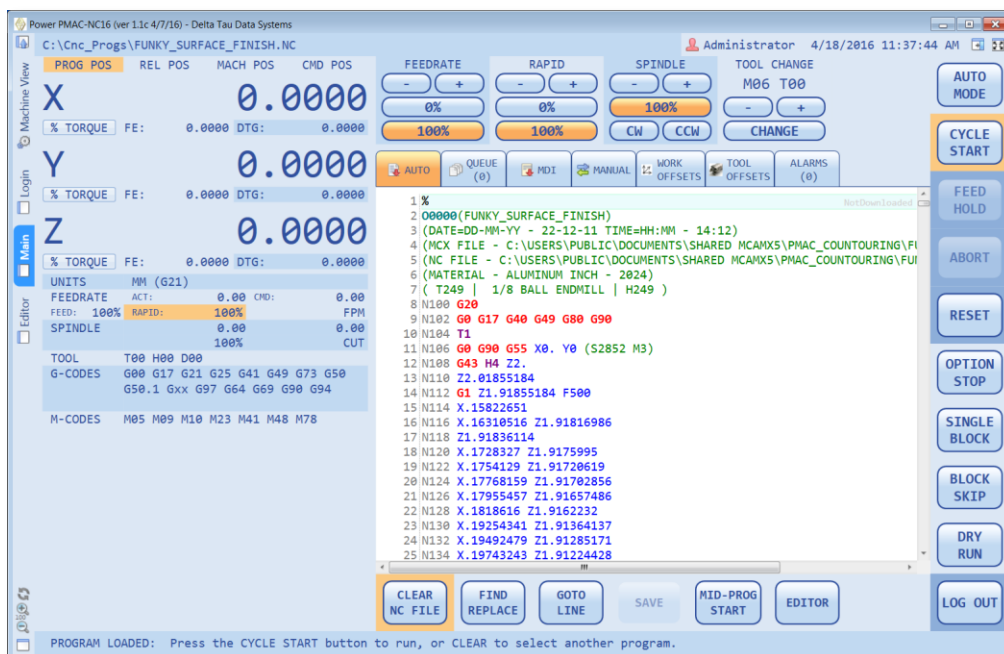
Introduction

The *Power PMAC-NC 16* HMI (PPNC16) is a host PC application for Delta Tau Power PMAC controlled CNC machines. This document is a Quick Start Manual for the *Power PMAC-NC 16* application. It contains information on how to quickly configure the Power PMAC-NC 16 software to run on a desktop PC. This document is not intended to replace a user's manual, but is intended to provide information to assist the integrator in gaining rapid familiarity with the basics of the software product architecture. The Quick Start manual makes assumptions with respect to the level of knowledge of the integrator. Please consult the full manual set to fill in the knowledge gaps. It should be mentioned 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.

 Note that much of the example code and detailed manuals are only supplied with the **SDK** version of the NC16 software and are not included with the basic **Runtime** version.

Other Manuals to refer to:

File Name	Description
<i>MCF Developer's Guide.pdf</i>	Reference for how to use MCF coding engine
<i>PowerPmacNC Software Users Manual.pdf</i>	Software Reference manual for Power PMAC-NC 16 application.
<i>Using GitHub.pdf</i>	Short form reference for using GitHub
<i>PowerPmacNC Mill Manual.pdf</i>	G-Code Reference manual – milling.
<i>PowerPmacNC Complete Integration Manual.pdf</i>	Not Yet Available - Detailed code reference for integrators.



Requirements

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

The application requires .NET 4.0 and the Visual C++ 2010 runtime libraries. The application will install these components automatically if not present. The following links can be used for manual installation of the same libraries.

Microsoft .NET Framework 4 (Web Installer)

<http://www.microsoft.com/en-us/download/details.aspx?id=17851>

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>



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 Runtime Software* is distributed in 2 ways:

1. On Delta Tau Media available by Download, CD, or USB drive as a set of install style files.
 - Insert the media and select the "Setup.exe" application
 - Follow the Wizard style prompts to install the Power NC Software
 - If your system does not meet the requirements you will be prompted to have these installed
2. Via a private GitHub repository. In order to access the online repository sign up for a [free GitHub account](#) and send your username (password not required) to your **Delta Tau Data Systems** support representative. You will be given read-only access to the repository. Install [GitHub for Windows](#) on your development PC, log in, and "Clone" the repository. You will want to "Sync" occasionally to insure you have the latest release version.



Once Github is successfully installed and you have logged into your account, Github will allow you to clone the repository on your local machine (Click on the plus sign in the upper left corner). If you have been invited to the repository you will see a Delta Tau icon. Click on the icon to begin the cloning process. GitHub will ask for the desired clone location and will copy the project to the selected folder. If you do not see the Delta Tau icon in the available clone list, you have not been given access to the repository - contact Delta Tau for assistance.



It is highly recommended you make a working copy of the *PPNC16* project to avoid losing your changes if and when you Sync. Simply copy the folder from the Github repository, place it somewhere you feel comfortable working with, and rename it. If a Sync fails for any reason, simply delete the entire “Github\PowerPmacNc16-Runtime” folder and Clone again.

Advantages and Functionality of GitHub

Although Power PmacNC16 can be delivered in various ways, GitHub is the preferred method. The user will receive automatic notifications when new versions are available and can sync the project from anywhere with internet access. When updating, the user should take care in merging the code to be sure any modifications will not be lost. A code comparison tool is recommended for this purpose.



Keeping separate copies of the source and working code is very important during machine development or following an update. Failure to do so can result in inadvertent loss of your work.

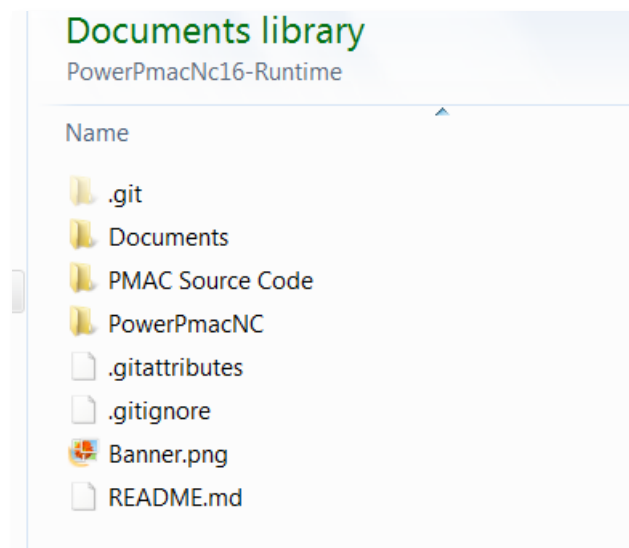
The following steps are meant to help users keep source and working code separate:

1. Use GitHub for Windows to “Clone” or copy the software to your PC.
2. Copy this set of files to a working directory on the system.
3. Leave the “Clone” version alone and only update as needed to keep this in sync with the online version.
4. If the official “Clone” version becomes corrupt, simply delete the clone folder and clone again using GitHub.

The Power PMAC-NC 16 Application

The Power PMAC-NC 16 machine control utilizes hybrid control architecture. Both a PC based HMI and separate control logic exists to achieve the total application. The various components of the entire system are shown below:

Power PMAC-NC 16 Project Folder Contents



.git - Various files and folders which are used by the GitHub application. These files are required in the clone folder, but can be deleted in the working folder.

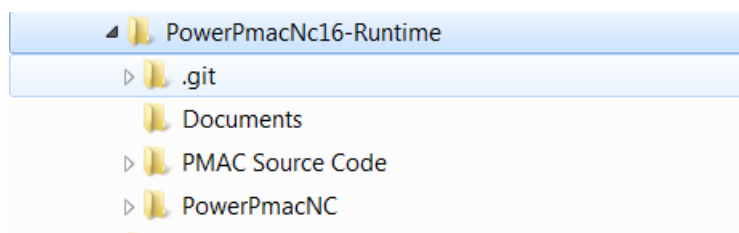
Documents - This folder will contain the latest Power PMAC-NC 16 documentation.

PMAC Source Code - This folder will contain the Power PMAC software project which is required by Power PMAC-NC 16 to function.

PowerPmacNC - The Power PMAC-NC 16 HMI executable and associated files will be located in this folder.

Working With Power PMAC NC Software

You should now have the Power Pmac NC Software suite installed and ready to use whether you did this using media from Delta Tau, USB drive, or via the GitHub repository. You should now have a single installed directory on your development system that looks similar to below:



If you used GitHub to download the repository you should first make a copy of this GitHub version and put the copy in a different directory on your development system. This will keep your ongoing working version of code and files separate from the version that is being synchronized with the GitHub repository online. Failure to do this will result in lost code.

Using GitHub to Clone the Project

1. Use GitHub for Windows to “Clone” or copy the software to your PC.
2. Copy this set of files to a working directory on the system.
3. Leave the “Clone” version alone and only update as needed to keep this in sync with the online version.

4. If the official “Clone” version becomes corrupt, simply delete the clone folder and clone again using GitHub.

Using GitHub to Update your Project

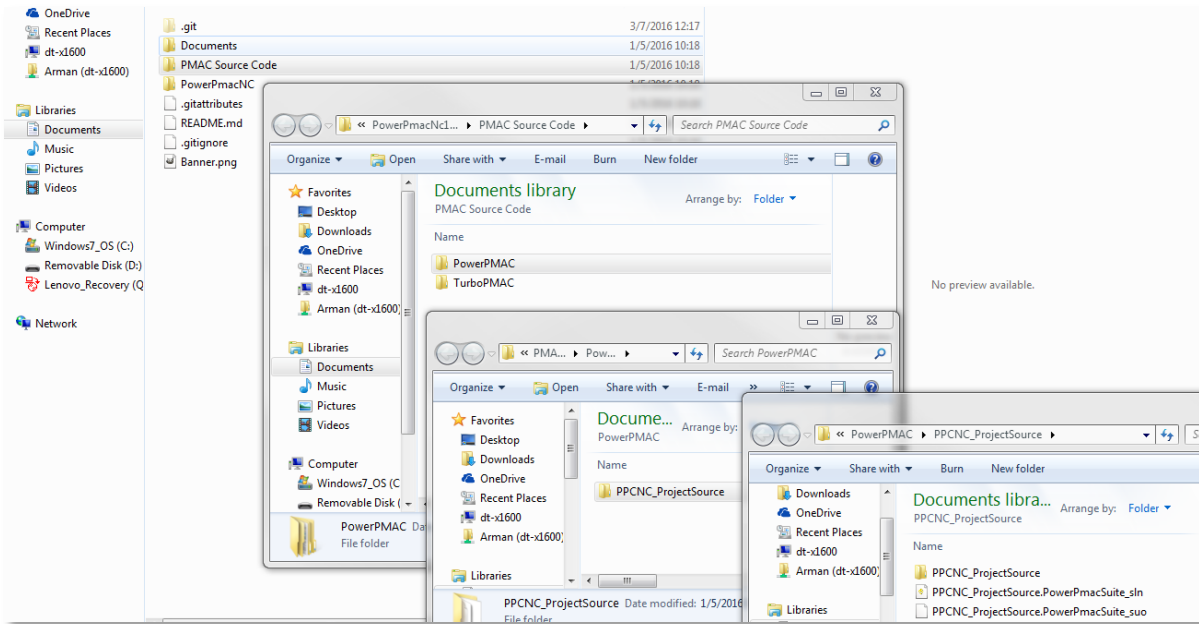
1. Using the steps above allow GitHub to update your Clone version.
2. Use a file comparison tool to merge the updated code with your working source files.
3. The files in the **PowerPmacNC** folder can simply be copied over your current working set (backup working first). Your key setup files in the Deployment folder will not be overwritten
4. The files in the **PMAC Source Code** folder must be carefully compared and edited one by one since your PMAC code has likely now been customized for your machine.
 - a. Typically changes to these files are usually limited to the NC files (M/G Codes) and some functions related to the HMI interaction
 - b. Newly created custom PLC's and Programs would not be affected by any changes made to the core group of PMAC files used by the NC HMI

Using GitHub to Revert Back to a Previous Version

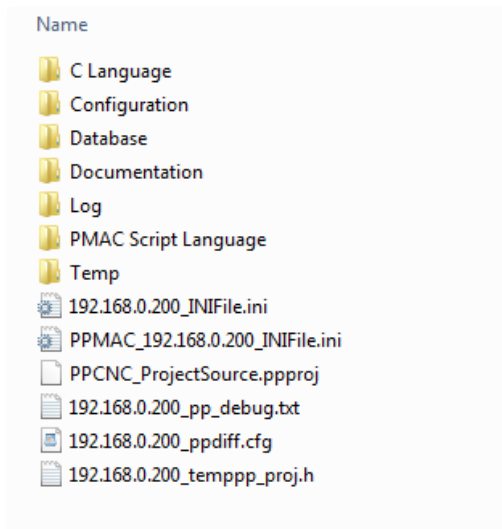
1. From the GitHub Desktop History Window select the version desired.
2. Select the GitHub online link in upper left – this takes you to online GitHub in Browser.
3. Select the Browse Files Button in upper Right.
4. Select the “Download ZIP” button in lower Right.
5. This will download a zipped set of files (Note name includes the version Hash value).

Power Pmac Source Code

The **Pmac Source Code** folder contains both **PowerPMAC** and **TurboPMAC** default software projects. Use the corresponding project for whichever controller you are using. The **PowerPMAC** folder contains a folder called **PPCNC_ProjectSource** which includes the main project for Power PMAC based application as shown below. This document will concentrate solely on Power PMAC based applications.



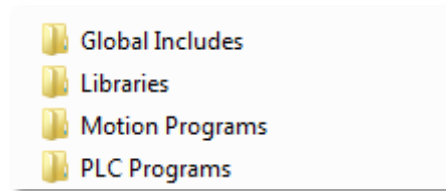
The **PPCNC_ProjectSource.PowerPmacSuite_slm** project is the starting point for all CNC applications. This project contains the default source code which enables the Power PMAC-NC 16 software application to function. The application will not function until this project is successfully downloaded to Power PMAC.



As shown above, the **PPCNC_ProjectSource** project contains various folders. A short description for each is provided below:


- C Language - Contains script written in C for foreground/background CPLCs.
- Configuration - Contains setup, startup, and other scripts which configure the Power PMAC.
- Database - Contains files used by the IDE for project management.
- Documentation – Place configuration notes here which are not compiled by the IDE.
- Log – Used by the IDE for tracking usage and error history.
- PMAC Script Language – Contains all of the PMAC script language files and folders.
- Temp - Contains temporary files used by the IDE.

The main files which are required by the Power PMAC-NC 16 application reside in the **PMAC Script Language** folder. The **PMAC Script Language** folder contains four main folders as shown below:




The **Global Includes** folder contains header files in the **.pmh** format as follow:

- ppnc_spindleparameters - Parameters relative to spindle functionality.
- ppnc_messages - Custom messages defined by the integrator.
- ppnc_csparameters - Variables and parameters required by the coordinate system.
- ppnc_dtpendant - Variables and parameters required by the pendant when used.
- ppnc_virtualmotors - Virtual parameters which can be used for initial setup if no actual motors exist.
- ppnc_ncinterfacedefinitions - Parameters definitions required by the HMI for setup and handshaking.

 The ppnc_virtualmotors file is meant for demonstration and testing purposes. This file will be replaced by your actual motor setup. An explanation of how to exclude this file from the main project in the Power Pmac IDE will follow.

The **Libraries** folder contains subprograms in the **.pmc** format as follow:


- ppnc_dcodes - D-Code script which reads cutter radius and wear for each tool.
- ppnc_gcodes - G-Code script which contains the actual G-code instructions followed by PMAC.
- ppnc_mcodes - M-Code script which contains the actual M-code instructions followed by PMAC.
- ppnc_timer – PMAC subprogram timer used by other script files.
- ppnc_xforms – PMAC transformation matrix which handles offsets, rotation, mirroring, scaling, etc.


 Keep in mind most machines will include custom G-Codes and M-Codes. Use extreme caution when editing and reviewing these files after an update. Any unwanted modification can be hazardous and cause machine damage and or injury if not handled properly.

The **Motion Programs** folder contains a sample motion program. No other files are used by the default configuration of Power PMAC NC16.

The **PLC Programs** folder contains the PLCs which actively monitor and respond to commands and set status for the HMI application and machine in general. The following is the default list of PLCs required by Power Pmac NC16:

- ppnc_function - Monitors the Command/Status states and executes logic accordingly.
- ppnc_handwheel – Executes handwheel jogging motion when a hardware MPG is present.
- ppnc_hmimonitor - Monitors and performs specific tasks when the HMI is present or otherwise.
- ppnc_home - Performs the homing functionality.
- ppnc_initialize - Initializes parameters required by the PMAC on boot.
- ppnc_mip_pendant - Monitors and performs different tasks related to the MIP pendant.
- ppnc_override - Monitors and performs override functionality.
- ppnc_positionreport - Monitors and reports position data for each axis.
- ppnc_reset - Monitors and performs the reset state and executes reset logic when require.
- ppnc_softpanel - Monitors and performs different software panel tasks.
- ppnc_spindle - Monitors and performs different tasks related to a spindle when present.
- ppnc_worktooloffset – Performs the Work and Tool offset functionality.

 Homing is different for every machine. The integrator will modify this code to suit his specific machine configuration. The included ppnc_home.plc provides the required handshaking for the HMI and can be used as a starting point for the actual logic which will be required by the machine.

 The ppnc_spindle.plc included with the source code is for general purpose use. This code will also need to be modified by the integrator for appropriate functionality.


Downloading the Power PMAC Project

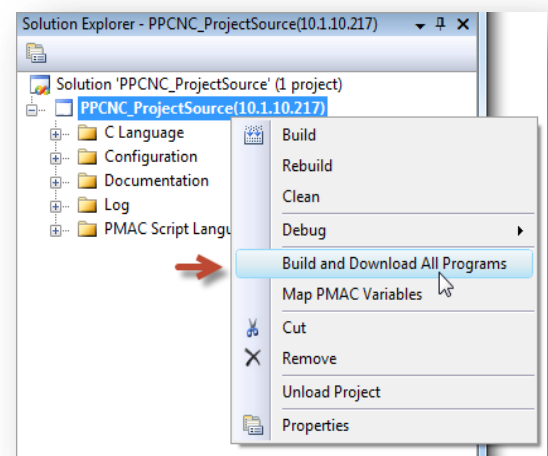
The PPNC16 application requires the Power PMAC controller be configured with its source code counterpart to enable proper functionality and handshaking. This Power PMAC project is included with the Power PMAC-NC 16 product.

The Power PMAC project will be located in the "PowerPmacNc16-Runtime\PMAC Source Code\PowerPMAC" folder. 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 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.

Power PMAC-NC 16 HMI Code

The Power PMAC-NC 16 HMI can be deployed by simply copying the "PowerPmacNc16-Runtime" folder to any location on your machine. The folder may be renamed if desired. Your distribution must include the files shown below. (Files not shown in this list may be deleted without affecting the application.)

Name	Type
Languages	File folder
Skins	File folder
Temp NC	File folder
VCPP4	File folder
DeviceMembers.xml	XML Document
DynamicDataDisplay.dll	Application extens...
ICSharpCode.AvalonEdit.dll	Application extens...
Interop.PCOMMSEVERLib.dll	Application extens...
MCF.DeltaTau.dll	Application extens...
MCF.DeltaTau.PcommServer.dll	Application extens...
MCF.MTConnect.dll	Application extens...
MCF.Foundation.dll	Application extens...
MessageLogViewer.exe	Application
Messages.xml	XML Document
Microsoft.WindowsAPICodePack.dll	Application extens...
Microsoft.WindowsAPICodePack.Shell.dll	Application extens...
PowerPmacNC.exe	Application
PowerPmacNC.ini	Configuration sett...
PowerPmacNC_MessageLog.txt	Text Document
PowerPmacNC_Notes.rtf	Rich Text Format
PowerPmacNC_Settings.xml	XML Document
Renci.SshNet.dll	Application extens...
Routrek.Granados.dll	Application extens...
SecureDongle_Control32.dll	Application extens...
SecureDongle_Control64.dll	Application extens...
WpfGauge.dll	Application extens...



What makes your machine or project unique?

- Several key files located in the PPNC16 directory are used to customize the application for your machine.
- These files will need to be added to the above set of files to make the installation complete.

PowerPmacNC_Settings.xml

- This file is automatically created and updated in real-time during run-time. Any settings within the Power PMAC-NC 16 application will be saved in this file.
- This file can be used to deploy OEM machinery by copying from a reference machine to a target machine.

- Various parameters are saved in this file including Tool and Work offsets.

PowerPmacNC.ini

- Parameters within this file determine the functionality and look of the HMI.
- Rename the file “Reference PowerPmacNC.ini” to “PowerPmacNC.ini” and be sure it resides in the same folder as the executable.
- Edit this file to customize the Power NC application – refer to the user’s manual for detailed information.

The application reads the “PowerPmacNC.ini” configuration file during initialization to obtain its configuration data. A *Reference* copy of this file is included in the project for convenience. Copy “Reference PowerPmacNC.ini” to the executable directory and rename it to “PowerPmacNC.ini”. Edit this file to specify your machine type, axis definitions, units, velocity units, etc. The configuration file is well commented for convenience.

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

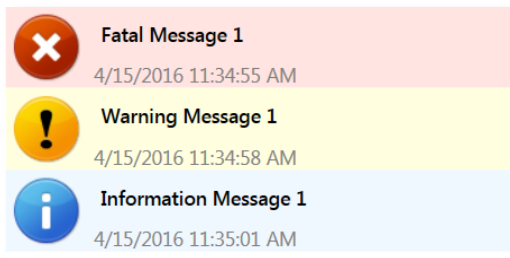
; TODO: Specify motor numbers separated by commas (for status monitoring).
; The first motor number will be used to monitor the status of the first axis, etc.
MotorNumbers=1,2,3,4,5

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

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

Messages.xml

- Rename the file “Reference Messages.xml” to “Messages.xml” and be sure it resided in the same folder as the executable.
- Edit this file to add custom messages for a specific machine integration.



This file includes three different types of messages, fatal, warning, and information messages. When the specific bit for that message goes high, it activates the message.

Hardware Key

This application requires a USB hardware key (dongle) to run. A hardware key will be provided to a user by manufacture when a user purchases copy of the PPNC16 program 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.



Power NC Software Levels

There are now 3 levels of the Power NC software distributed:

DEMO: This is a free to download version which will run in a simulated mode. It will not connect to a PMAC controller, but runs with a Mock Controller. It does not require a Hardware Key to be used. It demonstrates a limited feature set of the NC product and is able to open, load, and run (in simulated mode) NC files.

RUNTIME: This is a single license which allows a fully configured Power NC software package to run a machine. This requires a Hardware Key to be used. It can work with a Power PMAC, Turbo PMAC or no hardware controller in a simulated Mock Controller. This version can be fully customized to show custom panels, screens and extra controls as designed by the OEM/Integrator. However purchasing this version alone does not provide the examples, manuals or needed support to do this, see SDK version for this.

SDK: This version includes the RUNTIME license and requires a Hardware Key to be used. This version also includes extra developer manuals and example(s) of custom panels and "Add-Ins" used with the Basic Power NC. It is intended for an OEM or Integrator to purchase this license to get the extra examples, documentation and support to fully customize the basic Power NC application for his machine or project. Once this customization is done it can be freely distributed with each RUNTIME license purchased for each deployment on an actual machine. This also includes access to the GitHub SDK Repository online which contains continuous updates and example code for download and continued enhancements of projects.

Quick Start – Get your NC HMI Running

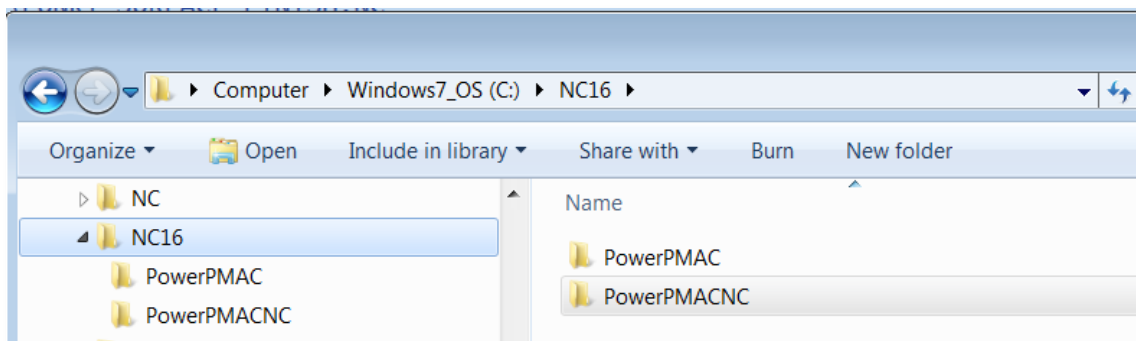
Overview

Here are basic steps to setup a Power PMAC-NC 16 application. The following setup will use a virtual configuration file to setup 4 motors. Additionally we will go thru the basic setup of a 3 axis Mill. Once configured you will be able to Jog motors and run NC programs from the HMI. Using the Power PMAC IDE we can verify the PMAC and HMI are working together to control the motors of the NC machine.

Target PC System Setup

Create a working set of files from the GitHub Clone directory as follows:

1. On your Target HMI PC system create a new Working directory for our Machine development: C:\NC16.
2. From the GitHub clone directory copy both the “PowerPmacNC” and “PMAC Source Code” folders into our new working folder.



Power PMAC Setup

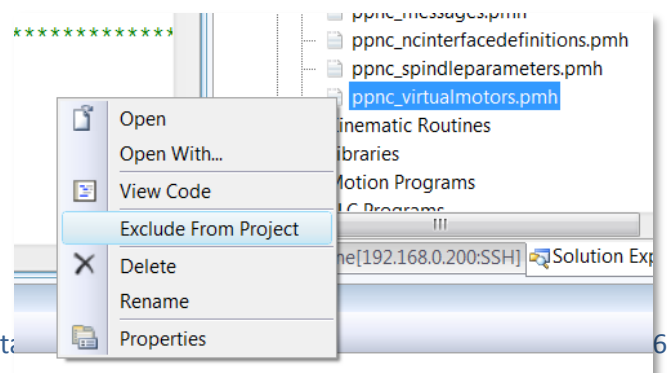
We will use the Power PMAC IDE to configure, download, and setup the Power PMAC to run with the NC HMI.

Open the Power PMAC project in the IDE using the file:

“C:\NC16\PowerPmac\PPCNC_ProjectSource\ PPCNC_ProjectSource.PowerPmacSuite_sln”

At this point you should have the default Power PMAC NC project loaded in the IDE and be communicating with a Power PMAC CPU. For this example the type of Power PMAC does not matter. This will work even with a bare UMAC CPU with no hardware accessories attached.

For this example we want to use a Virtual Motor Configuration file, NOT the header file from the Global Includes folder. If you have the file “ppnc_virtualmotors.pmh” in the “\PMAC Script Language\Global Includes” use the “Exclude From Project” tool



(right click on the file) to remove but not delete it from the project.

Creating a Virtual Motor Configuration

1. In “Configuration” Right-Click – Add – New Item – Custom Config
2. Use the file name “virtualmotor.cfg” – Add
3. You should have a blank configuration file open called “virtualmotors.cfg”
4. You can copy the code from the ppnc_virtualmotors.pmh file you just excluded or use the code included in this manual to create the virtual motors configuration file.
5. If you are using a Power PMAC CPU with NO Gate hardware uncomment the line as shown below:

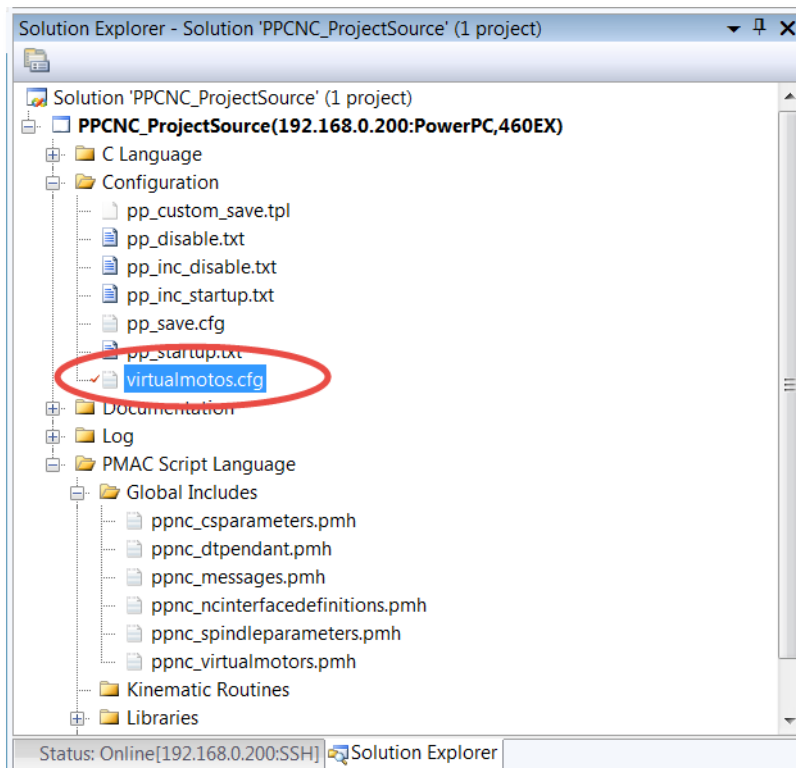
```
//Uncomment if Using Power CPU only  
Sys.CPUTimerIntr=1
```

6. Save the project with this new file

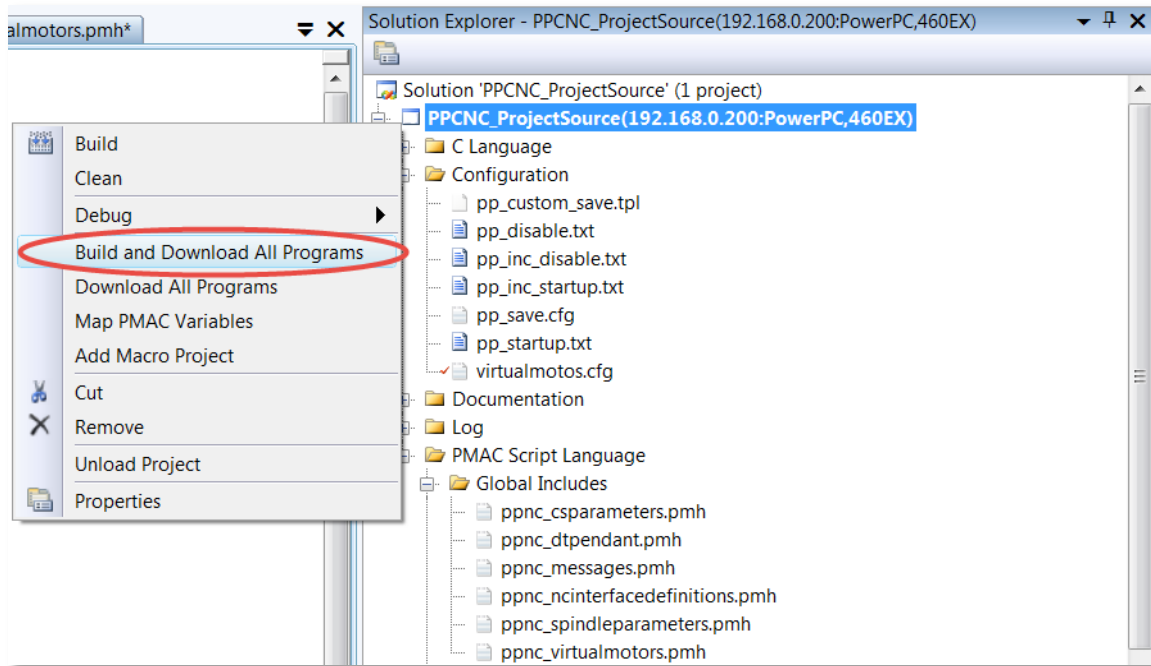
Downloading a Configuration File to PMAC

We will now select and download the configuration file to PMAC.

1. Right-Click the file “**virtualmotors.cfg**” and select “**Check to Download File**”.
2. A small RED Check should be next to the file in the file tree.



3. Re-Initialize the Power PMAC to factory default by issuing a “\$\$\$***” in the terminal window.
4. Right click on the project header and select “Build and Download”.

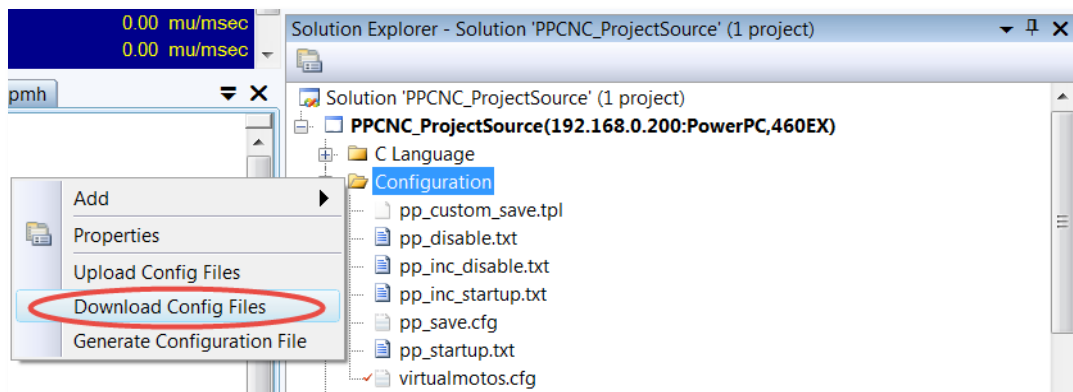


a. ”

5. Watch the Output window at bottom of IDE and verify the download was successful.

```
Download Successful.
Total Project Download time = 7.629 sec
Total Project Build and Download time = 19.797 sec
```

6. Next we will download the configuration file by right clicking the Configuration and selecting “Download Config Files”:



7. Be sure the download was successful by viewing verifying the following message in the output window at the bottom of the IDE:

```
Downloading virtualmotos.cfg to Power Pmac.
Please wait, while loading virtualmotos.cfg
// *** EOF
Downloading Configuration Files Is Successful!
```



Be sure the “virtualmotors.cfg” file includes a Red check, if it does not it was NOT loaded!

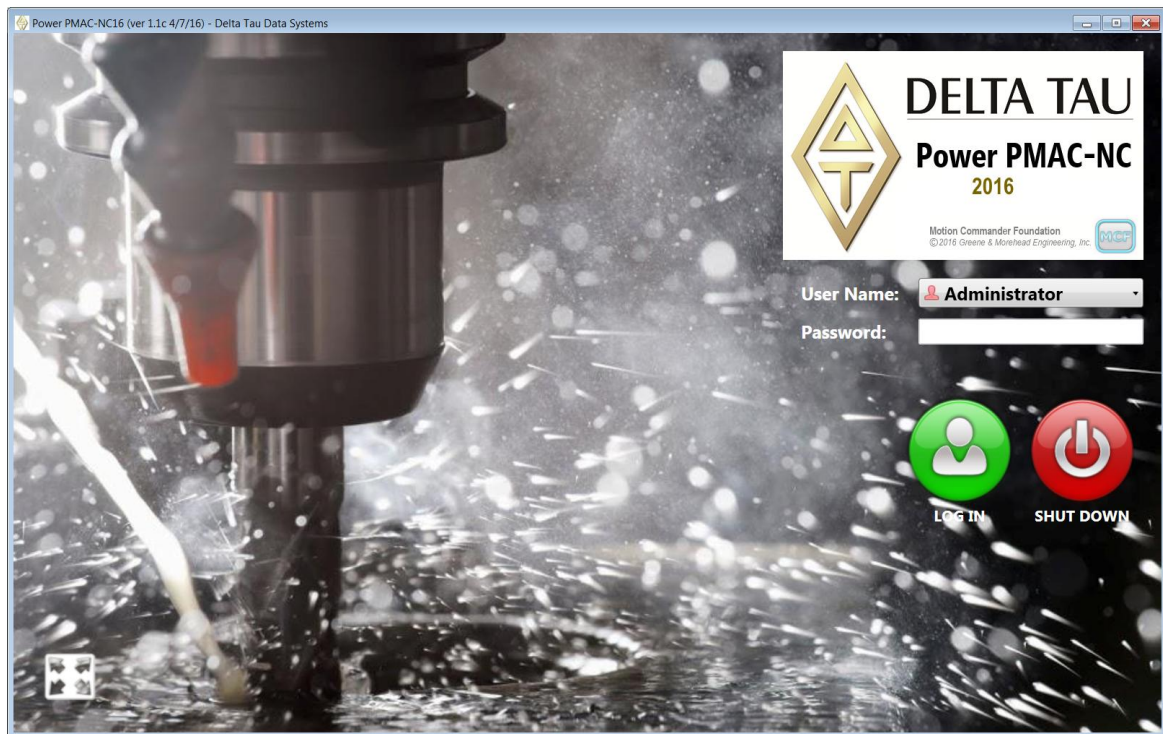
8. Save the PMAC configuration by typing “save” in the terminal window.
9. Reset the PMAC “\$\$\$”.

You should now be able to Jog motors 1-4 and see position update as if a real motor was running

Power PMAC NC Software Setup

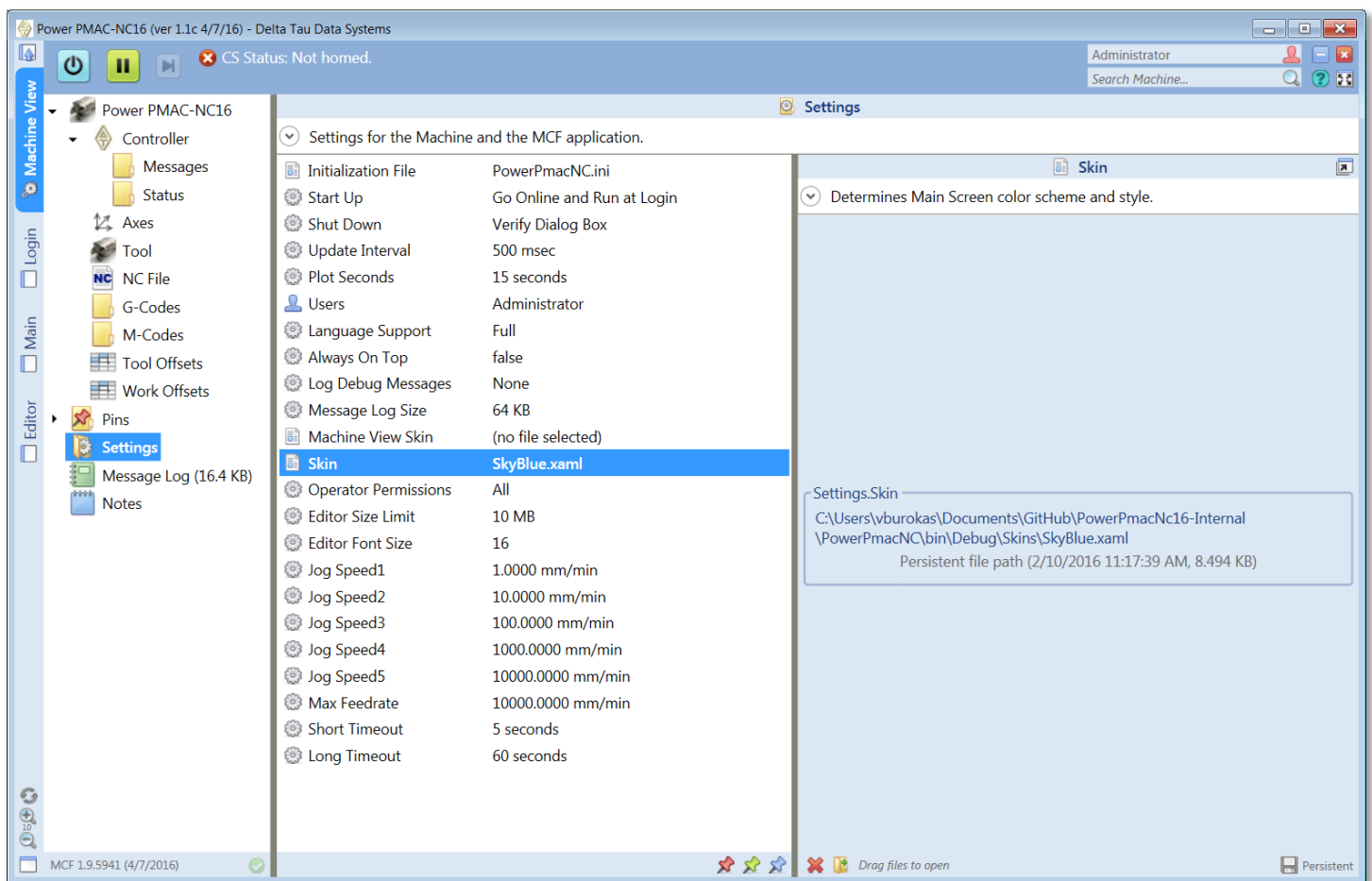
In the following example we will do the initial configuration and setup of the Power PMAC NC application.

1. Make sure the USB hardware key (dongle) is installed in a USB port.
2. While working In NC16 HMI working directory “\NC16\PowerPmacNC\”, **Rename** the file “Reference Messages.xml” to “Messages.xml”. This file contains the user messages which are displayed while running the HMI. The OEM/Integrator will likely edit this with custom messages for his project.
3. **Rename** the file “Reference PowerPmacNC.ini” to “PowerPmacNC.ini”. This file has basic machine configuration information used by Power PMAC NC during initialization. Edit this with Notepad for our simple Test NC16 Machine.
 - a. **Open the “PowerPmacNC.ini” file in Notepad.**
 - b. Make the following modifications:
 - i. AxisLabels=X,Y,Z
 - ii. MotorNumbers=1,2,3
 - c. Save and Close the PowerPmacNC.ini file.
4. With Power PMAC powered and Ethernet connected start the NC16 application.
 - a. **Start** with “\NC16\PowerPmacNC\PowerPmacNC.exe”
 - b. Note the Version number in Top Left corner.
 - c. **Select** “LOG IN” button.



5. Select the Machine View Tab in Upper Left corner. This gets us to Machine View for further system configuration. The following are some typical things to get setup and going:
 - i. Controller – Address – set to your Power PMAC IP Address – select Apply
 - ii. Settings – Start Up – Go Online and Run at Start-Up
 - iii. Settings – Jog Speed1 – 50 – Apply, Repeat for 2-5 with increasing values for faster jog speeds
 - iv. Settings – Max Feedrate – 1000 – Apply (likely adjust this later for proper value)
6. Let's change the Skin:
 - i. Double Click on Settings – Skin – Click on small File Folder bottom Left of Dialog
 - ii. Select MidnightBlue.xaml – Open – close small dialog [x] upper left, now done with this.
7. Now get NC16 to connect with Power PMAC and start the code running:
 - i. Select the Power Button at top Left
 - ii. Select the Run Button next to Power
 - iii. If all goes well the Run Button is now a Pause button and Green

Note the MCF (Motion Commander Foundation) version in lower left corner for reference.



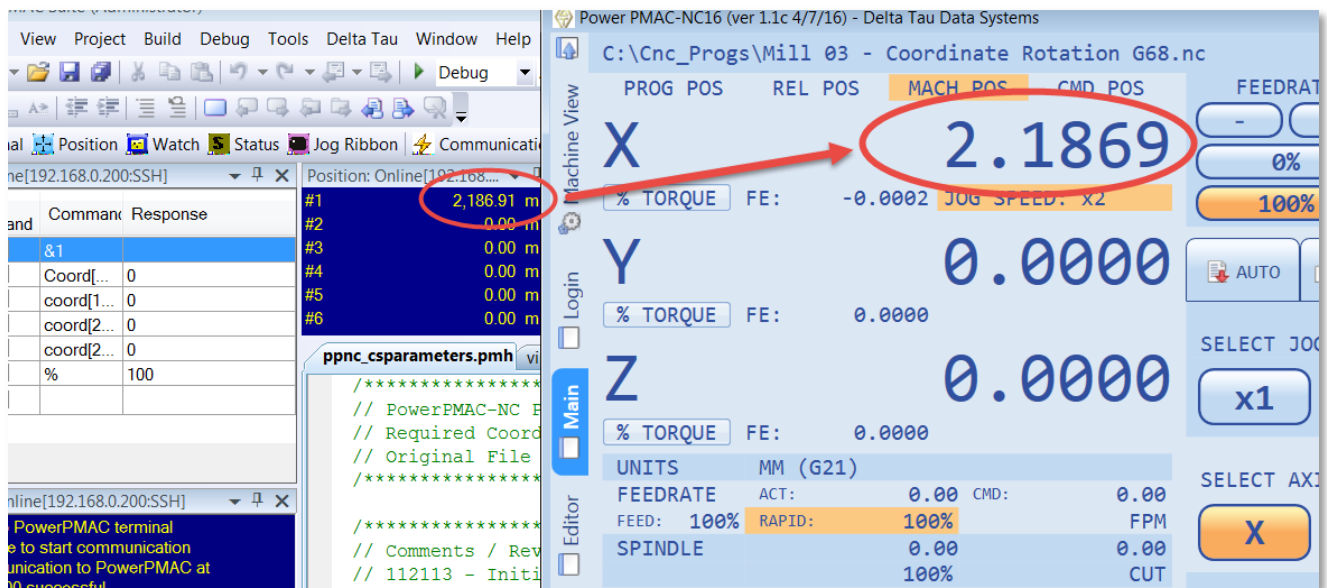
Running the Machine

1. **Select** the “Main” tab on Left side of Panel. This places you at the Main Operator HMI Panel for the NC16 application. You should also be able to see “live” motor data.
2. Select the “Reset” button on HMI.
3. Select “Manual” Tab then Home Button.
4. Check to see if you have any Alarm messages, if you do remedy the issue.

Jog a motor:

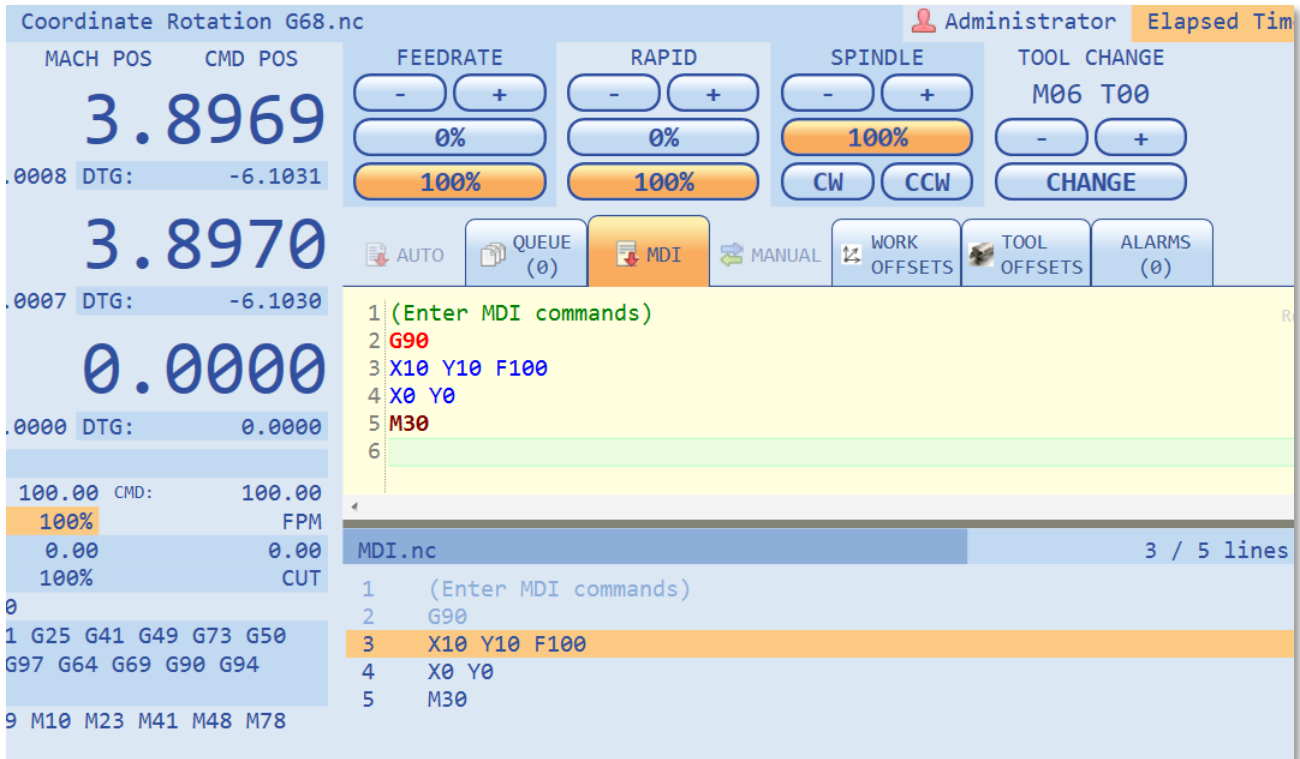
Go to Manual – Select “x2” speed – select “X” – “+Jog”

You should see the position updating in both the NC16 HMI and in the Power PMAC IDE.



Perform a G-code Move in MDI Mode:

1. Select MDI Tab
2. Enter desired moves such as shown.
3. **Press Cycle Start button**
4. **Watch the motor positions move.**
5. **Note lower window that shows lines as executing.**



What Next ?

At this point you should have a working NC HMI with Power PMAC. It should be able to Jog motors and run NC code as expected. Next you will want to modify this simple example to work with a real machine. Likely with real motors, I/O, and special logic needed. At this point you will want to consult the full manual set to understand the system in greater detail. Below is a short hand guide to the next steps in the machine integration process.

PMAC Logic

- If possible do not modify the existing ppnc PMAC files.
 - This will minimize the amount of changes if updating to a newer version later on.
 - If possible create new PLC's, Motion Programs, Sub-Routines, and Header files for custom logic.
- Create a new configuration file similar to the "virtualmotors.cfg" created earlier.
 - This will include the custom configuration needed for your actual machine.
 - You should use the IDE to create this configuration file.
 - This file will have all the necessary setup needed by the system including clock, motor, and PID settings.
- Create custom PLC / Motion Programs as needed.
 - Likely you will edit the Initialization PLC to handle machine power up settings
 - Create Reset/Power UP logic.
 - Create a custom Homing PLC or modify the existing basic logic.
 - Create error handling PLC code.
 - Modify the Spindle plc if needed.
 - Add other PLC's for custom machine functions.
- Edit the NC Sub-Routine files as needed for customer codes
 - These are located in the Libraries section of the script code (e.g. ppnc_mcodes.pmh, ppnc_gcodes.pmh).
 - When done your logic should be able to react and work with the built-in HMI logic for functions like Reset and Homing.

Power NC HMI Settings

- Go thru the Machine View – Settings Page to be sure all settings are as should be for this machine.



The machine settings are saved in file "PowerPmacNC_Settings.xml". This file is saved when data is modified and on application exit.

- Edit the PowerPmacNC.ini file as needed for your machine startup settings.

- Edit the Messages.xml file for your custom messages to the user.
- Refer to the Power NC Developer Manuals for details on how to customize features such as messages.

Target Machine Deployment

This section will give a brief explanation on how to deploy the Power PMAC-NC 16 application to your actual target machine. There are two components to think about, the PMAC code and the HMI code.

The PMAC Code

The PMAC code must be downloaded to the PMAC in order for the application to function. This can be done in a variety of ways. Here we will assume you plan on having a copy of the PMAC project on the target machine.

Simply use the Power PMAC IDE to “Build and Download” your project in the same way you have been doing for the virtual machine configuration we did previously.

Make sure you don’t forget to save the project after you download.

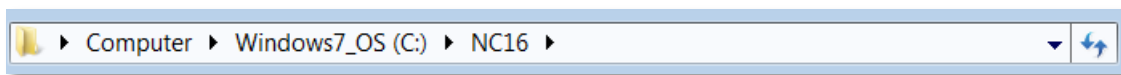


Many OEM’s will choose not to include the Power PMAC Project on the target machine and will configure the PMAC control via configuration files or choose to pre-configure the CPU at the manufacturing facility. Consult the Power PMAC IDE manual for more information.

The HMI Code

Deploying the Power PMAC-NC 16 application to your target machine requires two steps:

1. Copy the **PowerPmacNC** folder from your development machine to your target machine. The location is not important. A good example is as follows:



It is beneficial to place the application code in a folder close to the root (C:\) in order to keep the folder path names short. If the folder is placed too many levels deep it becomes cumbersome to manage the folder path names.

2. Transfer the three setup files which contain your application specific parameters from your development machine to the target machine. You may have done this automatically in the above step.

- a. **PowerPmacNC_Settings.xml** - This file is automatically updated whenever you change a parameter in the Power PMAC-NC 16 Machine View. This file includes your Work and Tool Offset data. Use the file you modified previously on your development machine.
- b. **PowerPmacNC.ini** – Use the file you modified previously from your development machine.
- c. **Messages.xml** – Use the file you modified previously from your development machine.

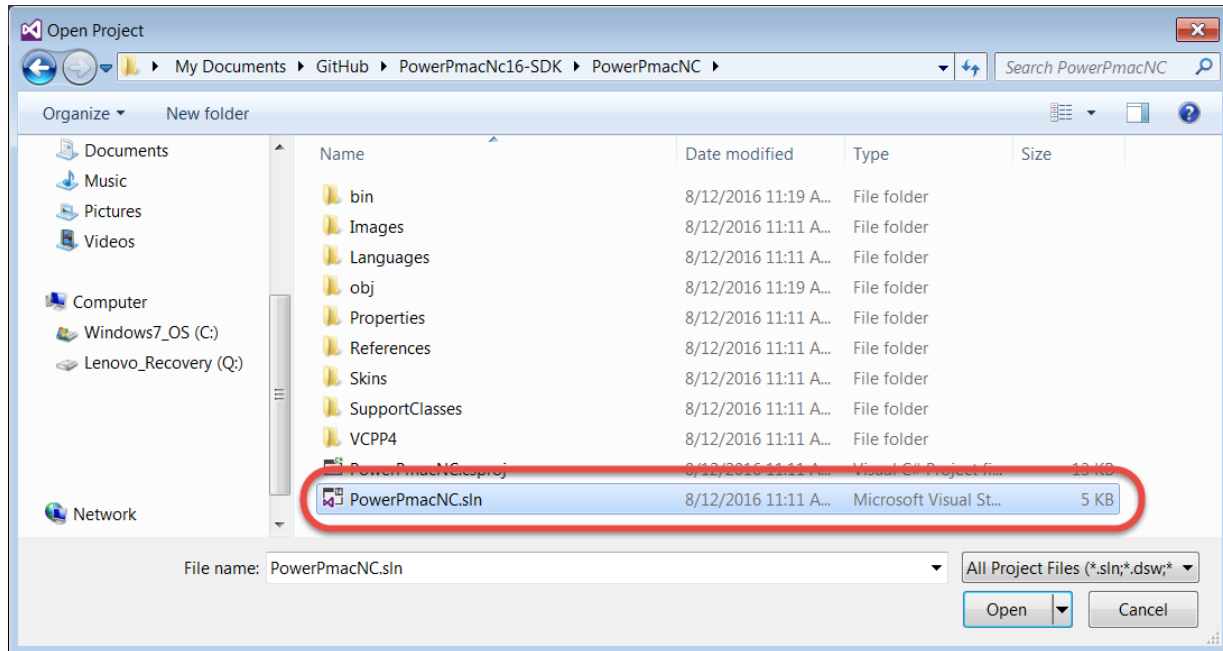
Deployment Summary

1. Download the PMAC configuration files to your PMAC. These source or configuration files do not need to be saved on the target HMI PC, just loaded and saved to the PMAC.
2. Create the application directory for the NC16 HMI on your target system.
3. Copy the **PowerPmacNC** application folder and files to the above folder.
4. Copy your three custom parameter files to your machine:
 - a. PowerPmacNC_Settings.xml
 - b. PowerPmacNC.ini
 - c. Messages.xml
5. Make sure the NC16 Dongle is installed in the USB port of the target PC.
6. Cycle power on the machine and verify **Power PMAC-NC 16** functionality.

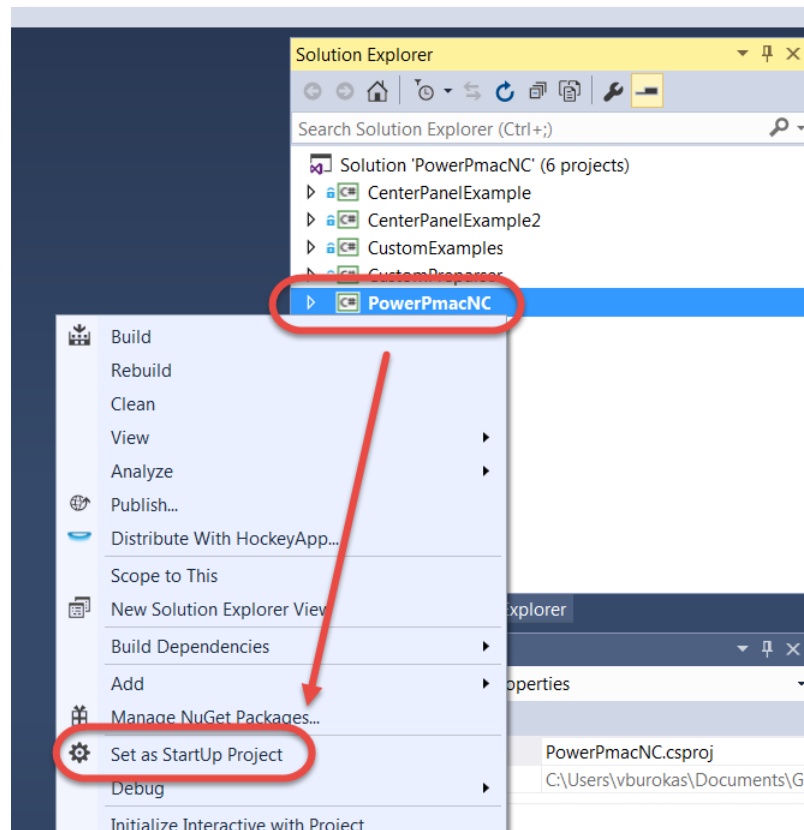
The Power PMAC NC SDK Package – Building Your Project for the First Time

The Power PMAC NC SDK package is the HMI developer's version of the software. It allows programmers to customize the package to suit a specific applications needs. If you are using the SDK version you will need to build the Power PMAC NC SDK project at least once to create the executable files. The SDK project is a Microsoft Visual Studio C# project so we recommend one of the Visual Studio products. You can use the latest *Microsoft Visual Studio Community 2015* product.

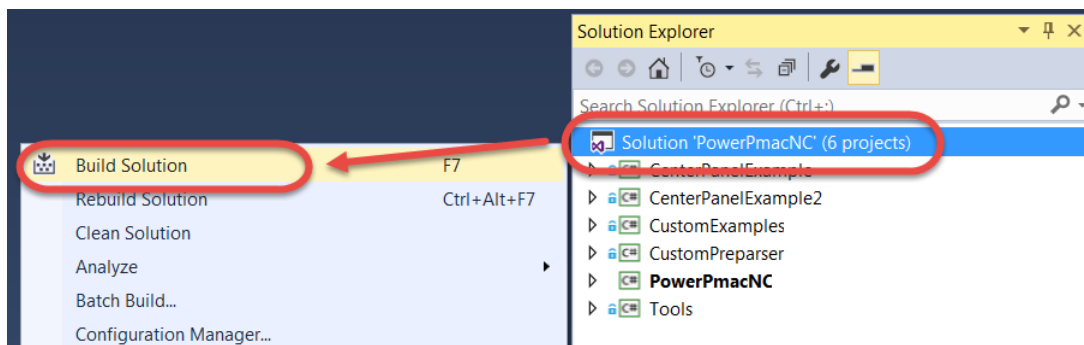
Once you are ready to use Visual Studio you can open the Power Pmac NC SDK solution as shown below:



The project should load without error. Before you build the project you should set the PowerPmacNC as the “Startup Project”. To do this right click the PowerPmacNc project and set it as shown below:



You can now build your project by right clicking the main solution and selecting “Build Solution” as shown (or “Rebuild Solution”).



If all is correct after a few moments you will see the following message in the output box showing successful build of six projects.

```
===== Build: 6 succeeded, 0 failed, 0 up-to-date, 0 skipped =====
```

You will now see the additional \bin\Debug file structure in your project folders. You will find the newly built PowerPmacNC.exe in the debug folder.

Don't forget to copy the "Reference PowerPmacNC.ini" and the "Reference Messages.xml" from the main PowerPmacNC folder to the debug folder and rename to "PowerPmacNC.ini" and "Messages.xml" as explained earlier in the *Power PMAC-NC 16 HMI Code* portion of this manual.

Appendix A - Power PMAC Virtual Motors Configuration File Sample

```
//*****/  
// PowerPMAC-NC Project Source File.  
// 4 Axis Setup File to allow the use of a PowerPMAC CPU Only in Virtual Motor Mode.  
// Original File Date: November 22, 2013  
//*****/  
  
//*****/  
// Comments / Revisions:  
// 112113 - Initial Submit to GitHub  
// 020414 - Added Axes 5-8  
//  
//  
//*****/  
  
//Speed and Accel  
//Setting InvAmax for Motor 1..8  
Motor[1].InvAmax=1,1,1,1,1,1,1,1  
//Setting MaxSpeed for Motor 1..8  
Motor[1].MaxSpeed=1000,1000,1000,1000,1000,1000,1000,1000  
//Setting MaxSpeed for Motor 1..8  
  
//Uncomment if Using Power CPU only  
//Sys.CPUTimerIntr=1  
//Virtual Motor 1 Setting  
Motor[1].pDac=sys.udata[11].a  
EncTable[1].pEnc=sys.udata[11].a  
Motor[1].Ctrl=sys.PosCtrl  
Motor[1].ServoCtrl=1  
Motor[1].FatalFeLimit=0  
Motor[1].pAmpFault=0  
Motor[1].pAmpEnable=0  
EncTable[1].type=1  
EncTable[1].index1=0  
EncTable[1].index2=0  
EncTable[1].index3=0  
EncTable[1].index4=0  
EncTable[1].index5=0  
EncTable[1].pEnc1=Sys.udata[11].a  
EncTable[1].pEnc=Sys.udata[11].a  
EncTable[1].MaxDelta=0  
EncTable[1].ScaleFactor=1  
EncTable[1].TanHalfPhi=0  
EncTable[1].CoverSerror=0  
//Virtual Motor 2 Setting  
Motor[2].pDac=sys.udata[12].a  
EncTable[2].pEnc=sys.udata[12].a  
Motor[2].Ctrl=sys.PosCtrl  
Motor[2].ServoCtrl=1  
Motor[2].FatalFeLimit=0  
Motor[2].pAmpFault=0  
Motor[2].pAmpEnable=0  
EncTable[2].type=1  
EncTable[2].index1=0  
EncTable[2].index2=0  
EncTable[2].index3=0  
EncTable[2].index4=0
```

```

EncTable[2].index5=0
EncTable[2].pEnc1=Sys.udata[12].a
EncTable[2].pEnc=Sys.udata[12].a
EncTable[2].MaxDelta=0
EncTable[2].ScaleFactor=1
EncTable[2].TanHalfPhi=0
EncTable[2].CoverSerror=0
//Virtual Motor 3 Setting
Motor[3].pDac=sys.udata[13].a
EncTable[3].pEnc=sys.udata[13].a
Motor[3].Ctrl=sys.PosCtrl
Motor[3].ServoCtrl=1
Motor[3].FatalFeLimit=0
Motor[3].pAmpFault=0
Motor[3].pAmpEnable=0
EncTable[3].type=1
EncTable[3].index1=0
EncTable[3].index2=0
EncTable[3].index3=0
EncTable[3].index4=0
EncTable[3].index5=0
EncTable[3].pEnc1=Sys.udata[13].a
EncTable[3].pEnc=Sys.udata[13].a
EncTable[3].MaxDelta=0
EncTable[3].ScaleFactor=1
EncTable[3].TanHalfPhi=0
EncTable[3].CoverSerror=0
//Virtual Motor 4 Setting
Motor[4].pDac=sys.udata[14].a
EncTable[4].pEnc=sys.udata[14].a
Motor[4].Ctrl=sys.PosCtrl
Motor[4].ServoCtrl=1
Motor[4].FatalFeLimit=0
Motor[4].pAmpFault=0
Motor[4].pAmpEnable=0
EncTable[4].type=1
EncTable[4].index1=0
EncTable[4].index2=0
EncTable[4].index3=0
EncTable[4].index4=0
EncTable[4].index5=0
EncTable[4].pEnc1=Sys.udata[14].a
EncTable[4].pEnc=Sys.udata[14].a
EncTable[4].MaxDelta=0
EncTable[4].ScaleFactor=1
EncTable[4].TanHalfPhi=0
EncTable[4].CoverSerror=0
//Virtual Motor 5 Setting
Motor[5].pDac=sys.udata[15].a
EncTable[5].pEnc=sys.udata[15].a
Motor[5].Ctrl=sys.PosCtrl
Motor[5].ServoCtrl=1
Motor[5].FatalFeLimit=0
Motor[5].pAmpFault=0
Motor[5].pAmpEnable=0
EncTable[5].type=1
EncTable[5].index1=0
EncTable[5].index2=0
EncTable[5].index3=0
EncTable[5].index4=0

```

```

EncTable[5].index5=0
EncTable[5].pEnc1=Sys.udata[15].a
EncTable[5].pEnc=Sys.udata[15].a
EncTable[5].MaxDelta=0
EncTable[5].ScaleFactor=1
EncTable[5].TanHalfPhi=0
EncTable[5].CoverSerror=0
//Virtual Motor 6 Setting
Motor[6].pDac=sys.udata[16].a
EncTable[6].pEnc=sys.udata[16].a
Motor[6].Ctrl=sys.PosCtrl
Motor[6].ServoCtrl=1
Motor[6].FatalFeLimit=0
Motor[6].pAmpFault=0
Motor[6].pAmpEnable=0
EncTable[6].type=1
EncTable[6].index1=0
EncTable[6].index2=0
EncTable[6].index3=0
EncTable[6].index4=0
EncTable[6].index5=0
EncTable[6].pEnc1=Sys.udata[16].a
EncTable[6].pEnc=Sys.udata[16].a
EncTable[6].MaxDelta=0
EncTable[6].ScaleFactor=1
EncTable[6].TanHalfPhi=0
EncTable[6].CoverSerror=0
//Virtual Motor 7 Setting
Motor[7].pDac=sys.udata[17].a
EncTable[7].pEnc=sys.udata[17].a
Motor[7].Ctrl=sys.PosCtrl
Motor[7].ServoCtrl=1
Motor[7].FatalFeLimit=0
Motor[7].pAmpFault=0
Motor[7].pAmpEnable=0
EncTable[7].type=1
EncTable[7].index1=0
EncTable[7].index2=0
EncTable[7].index3=0
EncTable[7].index4=0
EncTable[7].index5=0
EncTable[7].pEnc1=Sys.udata[17].a
EncTable[7].pEnc=Sys.udata[17].a
EncTable[7].MaxDelta=0
EncTable[7].ScaleFactor=1
EncTable[7].TanHalfPhi=0
EncTable[7].CoverSerror=0
//Virtual Motor 8 Setting
Motor[8].pDac=sys.udata[18].a
EncTable[8].pEnc=sys.udata[18].a
Motor[8].Ctrl=sys.PosCtrl
Motor[8].ServoCtrl=1
Motor[8].FatalFeLimit=0
Motor[8].pAmpFault=0
Motor[8].pAmpEnable=0
EncTable[8].type=1
EncTable[8].index1=0
EncTable[8].index2=0
EncTable[8].index3=0
EncTable[8].index4=0

```



```

EncTable[8].index5=0
EncTable[8].pEnc1=Sys.udata[18].a
EncTable[8].pEnc=Sys.udata[18].a
EncTable[8].MaxDelta=0
EncTable[8].ScaleFactor=1
EncTable[8].TanHalfPhi=0
EncTable[8].CoverSerror=0

// Disable overtravel limit inputs
// May be needed if there are no physical switches present
// Motor CS Def Scale Factors
Motor[1].PosSf=1,1,1,1,1,1,1,1
Motor[1].Pos2Sf=1,1,1,1,1,1,1,1

// Set derivative gain term in servo loop to zero
// This is a Type 1 servo (single integration); does not need Kd
Motor[1].Servo.Kvfb=0,0,0,0,0,0,0,0

// Lower proportional gain term from default
Motor[1].Servo.Kp=40,40,40,40,40,40,40,40

```