CENTROID™

Acorn CNC12
Macro Programming

Introduction to Centroid Acorn CNC12 Macros

CNC Software version: CNC12 V.4.20+
Models: Acorn CNC

Introduction to Centroid CNC Macros

What is a Macro?
A Macro used to describe a set of CNC control commands that are called upon to perform a machine tool function, think of it as a bundle of commands that you can use at the touch of a button or by adding a single M code into a G&M code program. The word macro is a term is being loosely used here to cover a wide variety of CNC "programs", everything from a machine tool homing program to an automatic tool change program. Macros are simply G and M code programs, they can be one line or thousands of lines. Most all CNC12 G&M codes can be used in a macro. There are many stock 'macros' included with CNC12. CNC12's included "M-Functions" are used to perform specialized actions in CNC programs. Most of the CNC12 M-Functions have default actions as described in the Mill and Lathe operators manuals, but can also be customized with the use of macro files.

How do I create or edit a Macro?
A macro program is just a text file like any other G and M code program. We recommend using Notepad++ (not Notepad). Notepad++ is a free powerful text editor and can be downloaded for free here.
https://notepad-plus-plus.org/downloads/

Where to Start?
There are included "stock" macros with the Acorn CNC12 software installation. Many of these macros have instructions and comments built into them so they can be user modified and also used as macro programming learning examples. Most macros end with the .mac file extension with some exceptions for special cases such as the home program files which are "cncm.hom" and "cnct.hom".

The most common Acorn CNC macros for beginners to edit and modify are:

- Machine Tool Homing Macro: cncm.hom (mill or router) or cnct.hom (lathe)
- Machine Tool Parking Macro: park.mac
- Macros that are preassigned to the VCP Aux keys 8,9,10,11: M55(Aux 8),M56 (Aux 9),M57(Aux 10),M58 (Aux 11)
- Macros that are preassigned to the Wireless MPG Macro keys 1-4: Plcmacro1(2,3,4).mac

The Home macro is created by the Acorn Wizard based on the selections made in the Wizard for the type of homing, direction of homing, order of homing. Most Acorn users will not have to edit the Home macro as the Wizard does a good job creating home macros (aka home programs). However there are a cases where it is nice to add some customization to the home program to make the work flow on a machine tool better.

The pre-assigned Macros for the VCP aux keys and the MPG Macro buttons are themselves commented with examples. Once you have Acorn up and running and homed, Press the VCP Aux 8 key (that will run the M55 macro) and the instructions on how to edit M55 appear on the screen (and same with the Wireless MPG Macro 1,2,3,4 buttons). The M-code M55 calls and runs the commands contained in the text file "mfunc55.mac. Opening, studying and modifying these existing macro is a great place to start.

The included macros are located in these directories "cncm", "cnct", "cncm/system" and "cnct/system".

Edit your first macro!

Below is a typical Wizard generated Acorn Home Program for a three axis milling machine.

```
M91/Z L1        ; move Z axis in negative direction seeking home switch, trigger switch and then reverse to clear switch
M26/Z           ; set Z home at current position
M92/Y L1        ; move Y axis in positive direction seeking home switch, trigger switch and then reverse to clear switch
M26/Y           ; set Y home at current position
M91/X L1        ; move X axis in negative direction seeking home switch, trigger switch and then reverse to clear switch
M26/X           ; set X home at current position
```

Lets say we want to set the Z axis Home position .050" away from the Z home switch rather than right after the switch clears and do this automatically.

Open “cncm.hom” with Notepad ++ and insert the line shown highlighted below

```
M91/Z L1          ; move Z axis in negative direction seeking home switch, trigger switch and then reverse to clear switch
G91 Z .050 F25    ; move Z axis incrementally in the positive direction .050”
M26/Z             ; set Z home at current position
M92/Y L1          ; move Y axis in positive direction seeking home switch, trigger switch and then reverse to clear switch
M26/Y             ; set Y home at current position
M91/X L1          ; move X axis in negative direction seeking home switch, trigger switch and then reverse to clear switch
M26/X             ; set X home at current position
```

Using Notepad++ save the file (File, Save). Now the next time you home the machine (or press reset home on the VCP) the machine, after clearing the Z home switch, will back away from the Z home switch an additional .050” before setting the Z home position at the current location. See Operators manual G and M code chapters for description of M26, M91/2 and G91

Edit your second macro! Lets edit M55 so that it will Rapid move to the XY position X=0, Y=0.

M55 is a stock Acorn macro that is preassigned to the VCP (Virtual Control Panel) Auxiliary key #8 so when you press Aux 8 CNC12 will execute the commands contained within M55. Lets edit M55 so that it will rapid move the machine to WCS X=0, Y=0. Open mfunc55.mac with Notepad++. Note: A semi colon ";" tells CNC12 that the line is a comment, remove semi colon for command to be run, add semi colon to beginning of line to be ignored.

```
;-------------------------------------------------------------------------------
; Filename: mfunc55.mac - To run from VCP AUX 8 key, set p195 = 5511
; M55 macro
; Description: User Customizable Macro
; Notes:
; Requires: Machine home must be set prior to use.
; Please see TB300 for tips on writing custom macros.
;-------------------------------------------------------------------------------

IF #50001                   ;Prevent G code lookahead from parsing past here
IF #4201 || #4202 THEN GOTO 1000 ;Skip macro if graphing or searching

M225 #100 "This is an example macro named mfunc55.mac and can be found in ..\cncm or ..\cnct\nEdit it to include the desired functionality.
Press Cycle Cancel/ESC to Exit"

N100                        ;Insert your code between N100 and N1000
; Insert commands here.
N1000                       ;End of Macro
```

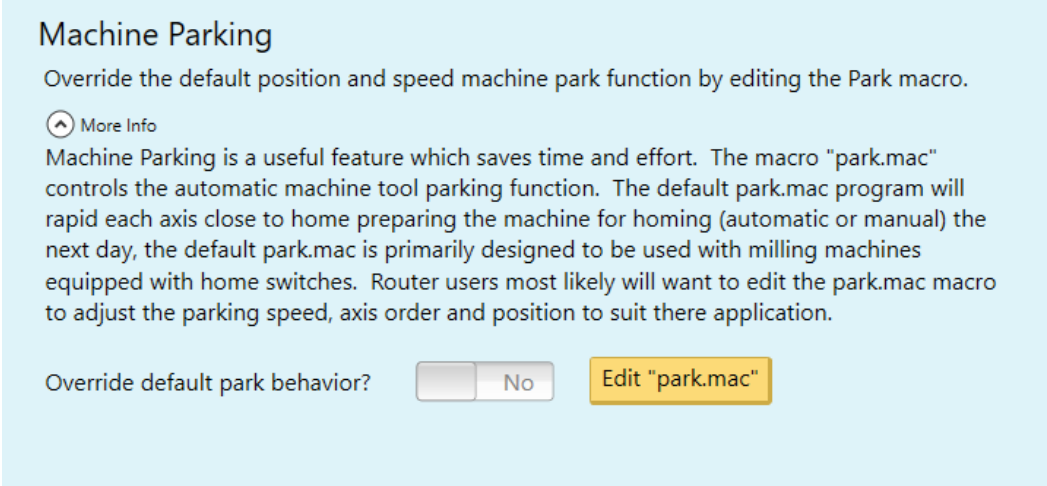
Place a semi colon in front of the M225 line to comment it out, and then insert your custom G and M codes in between lines N100 and N1000.
So, to modify M55 to Rapid X and Y axis to the position X0, Y0 to look like this:

```
; M225 #100 "This is an example macro named mfunc55.mac and can be found in ..\cncm or ..\cnct\……..etc.
;
N100              ;Insert your code between N100 and N1000
G0X0Y0            ; G0 = Rapid Move to position X=0, Y=0 in the current Work Coordinate System
N1000             ;End of Macro
```

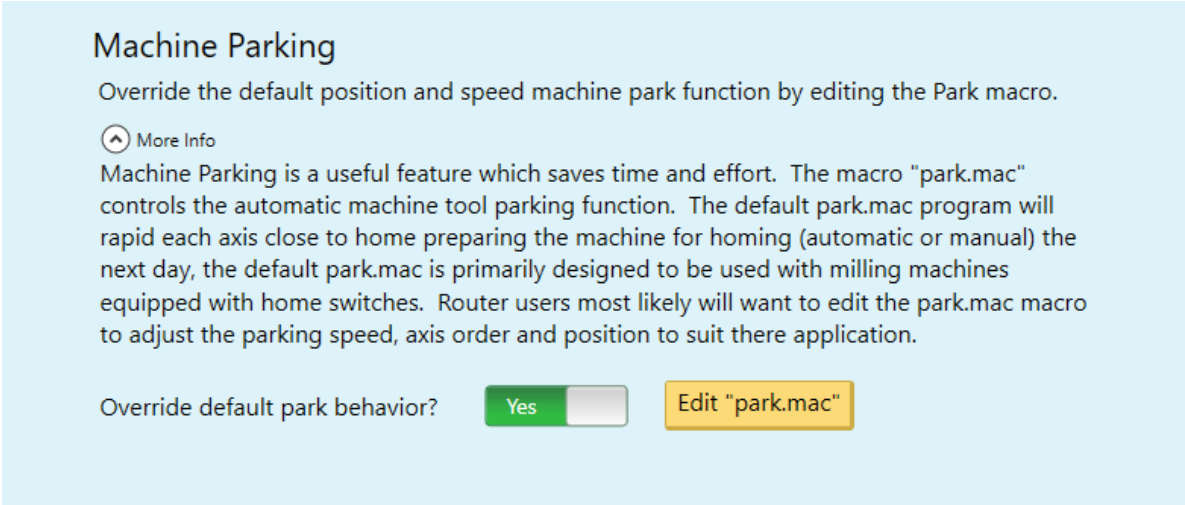
Using Notepad++ save the changes to mfunc55.mac using File, then Save. Now in CNC12 the machine tool will rapid to the XY position 0,0 when Aux 8 is pressed or when a M55 is contained in a G and M code program.

Edit your third Macro!

Park.mac is a Machine Tool Parking macro that is primarily used to park a machine at the end of the day. Typical use of a Park macro will move the machine close to the home switches so that the machine is ready to run the home macro in the morning with minimum of movement. CNC12 has default Parking functionality built into it so, if the Acorn Wizard is set to No as seen below then then the default CNC12 park action is used.



The default action is fine for most average milling machines but, I find that most users will benefit from editing and creating a custom Park macro to suit their particular machine tool and application. The default action will Rapid Move the axes close the to home switches. A common custom Park.mac will move the axis close to the home switches but at a user specified feedrate.



Choose “Yes” to override the default park behavior and edit “park.mac”

The stock Park.mac macro looks like this below, there are three examples given that are commented out with a semi colon.

```
;-------------------------------------------------------------------------------
; Filename: park.mac - Custom Park Button Macro
; Description: This Machine Park macro controls the "F1 Park" command action found in the CNC12 "F10 Shut Down" menu.
; park.mac is a user editable custom macro used for parking a machine tool at a specified position at a specified speed, typically used prior to shut down.
; Notes:
; - This macro overrides the default CNC12 park behavior logic when selected to do so in the Wizard (the Wizard sets CNC12 parameter 59 = 1)
; - This macro allows the user to customize the action and behavior of the "Park" feature in the CNC12 "Shut Down" menu by writing creating a custom G&M code program
; - A semi colon ";" tells CNC12 that the line is a comment, remove semi colon for command to be run, add semi colon to beginning of line to be ignored
;-------------------------------------------------------------------------------
M225 #100 "Please edit c:\cncm\system\park.mac to create a custom parking macro, press ESC to exit."
;-------------------------------------------------------------------------------
; Example Custom Mill Park Macro assuming XYZ home position is set in the negative direction for each axis.
;G53 Z.25 L20  (Moves Z axis first in machine coordinates to .25 inches away from Z home position at 20 inches per minute)
;G53 X1 L200   (Moves X axis second in machine coordinates to 1 inch away from X home position)
;G53 Y1 L200   (Moves Y axis third in machine coordinates to 1 inch away from Y home position)
;-------------------------------------------------------------------------------
; Example Custom Park Macro assuming Y home position is set in positive direction and XZ home position is set in the negative direction for each axis.
;G53 Z.25 L20  (Moves Z axis first in machine coordinates to .25 inches away from Z home position at 20 inches per minute)
;G53 X1 L200   (Moves X axis second in machine coordinates to 1 inch away from X home position at 200 inches per minute)
;G53 Y-1 L800   (Moves Y axis third in machine coordinates to 1 inch away from Y home position at 800 inches per minute)
;-------------------------------------------------------------------------------
; Example Custom Park Macro assuming Y home position is set in positive direction and XZ home position is set in the negative direction for each axis.
;G53 Z.25 L20  (Moves Z axis first in machine coordinates to .25 inches away from Z home position at 20 inches per minute)
;G53 X1 Y-1 L200   (Moves X and Y axis at the same time in machine coordinates to 1 inch away from X home position at 200 inches per minute)
;G53 w0 L360  (Moves W rotary axis in machine coordinates to 0.000 at 360 degrees per minute)
;-------------------------------------------------------------------------------
```

To customize Park.mac, Add a semi colon in front of the M225 line to comment it out.
And then uncomment one of the examples provided and edit the feedrate and direction to your requirements. See changes highlighted in yellow below.

```
;-------------------------------------------------------------------------------
; Filename: park.mac - Custom Park Button Macro
; Description: This Machine Park macro controls the "F1 Park" command action found in the CNC12 "F10 Shut Down" menu.
; park.mac is a user editable custom macro used for parking a machine tool at a specified position at a specified speed, typically used prior to shut down.
; Notes:
; - This macro overrides the default CNC12 park behavior logic when selected to do so in the Wizard (the Wizard sets CNC12 parameter 59 = 1)
; - This macro allows the user to customize the action and behavior of the "Park" feature in the CNC12 "Shut Down" menu by writing creating a custom G&M code program
; - A semi colon ";" tells CNC12 that the line is a comment, remove semi colon for command to be run, add semi colon to beginning of line to be ignored
;-------------------------------------------------------------------------------
;M225 #100 "Please edit c:\cncm\system\park.mac to create a custom parking macro, press ESC to exit."
;-------------------------------------------------------------------------------
; Example Custom Mill Park Macro assuming XYZ home position is set in the negative direction for each axis.
;G53 Z.25 L20  (Moves Z axis first in machine coordinates to .25 inches away from Z home position at 20 inches per minute)
;G53 X1 L200   (Moves X axis second in machine coordinates to 1 inch away from X home position)
;G53 Y1 L200   (Moves Y axis third in machine coordinates to 1 inch away from Y home position)
;-------------------------------------------------------------------------------
; Example Custom Park Macro assuming Y home position is set in positive direction and XZ home position is set in the negative direction for each axis.
;G53 Z.25 L20  (Moves Z axis first in machine coordinates to .25 inches away from Z home position at 20 inches per minute)
;G53 X1 L200   (Moves X axis second in machine coordinates to 1 inch away from X home position at 200 inches per minute)
;G53 Y-1 L800   (Moves Y axis third in machine coordinates to 1 inch away from Y home position at 800 inches per minute)
;-------------------------------------------------------------------------------
; Example Custom Park Macro assuming Y home position is set in positive direction and XZ home position is set in the negative direction for each axis.
G53 Z.25 L20  (Moves Z axis first in machine coordinates to .25 inches away from Z home position at 20 inches per minute)
G53 X1 Y-1 L200   (Moves X and Y axis at the same time in machine coordinates to 1 inch away from X home position at 200 inches per minute)
G53 w0 L360  (Moves W rotary axis in machine coordinates to 0.000 at 360 degrees per minute)
;-------------------------------------------------------------------------------
```

Using Notepad++ save the changes to park.mac using File, then Save. Now in CNC12 the Park feature found in the Shut Down menu will run this custom macro and in the example above will move Z slowly .25” from Z home and then move X and Y simultaneously to 1” away from machine home position and then moves the rotary axis to the W0 machine position.

See Operators manual G and M code chapters for description of G53 and M225 and FAQ section below for description of “L” command.

Tip: If you built your own CNCPC edit the Windows file extension default application settings so Notepad ++ will be the default editor for .mac, .hom, .cnc, .nc, .txt

The mill (and lathe) operator manual has a number of sections covering macro programming.
Once you are familiar with a basic macros such as mfunc55.mac, cncm/t.hom and park.mac you can start to increase your macro knowledge by reading the related macro material in the CNC12 Operators Manuals:

CNC12 Mill Operator Manual (manuals can be found here. https://www.centroidcnc.com/centroid_diy/centroid_manuals.html)

Chapter 11: CNC program codes		page 193
Chapter 13: CNC program M codes		page 243
G65	Call Macro	page 222
M98/99	Call Subprogram	page 251
M100/M101	Wait for PLC bit	page 253
M200/223,224,225,290	Formatted String	page 260
Advanced Macro Statements		page 202
Chapter 14 covers stock ATC macros		page 265

Acorn Wizard canned PLC functions and M Codes.
https://www.centroidcnc.com/centroid_diy/downloads/acorn_documentation/acorn_wizard_input_output_plc_functions.pdf
And be sure to visit these two threads on the Acorn CNC Tech Support Forum for free downloads of custom macros for AutoTool Setting and ATC's. "Tool Setting Options For Routers and Mills" and "Acorn ATC Overview" in the "Acorn CNC Tech Tips Knowledge Base" forum.

Other Related useful Reference Material:

- Centroid CNC12 PLC Programming Manual (https://www.centroidcnc.com/centroid_diy/centroid_manuals.html)
- Centroid PLC Programming Videos on the Centroid CNC Tech support YouTube Channel https://www.youtube.com/playlist?list=PLXhs2C5No0_gFS_RmKNo7hii2WKIedQlQ
- PLC detective https://www.centroidcnc.com/downloads/centroid_PLC_detective_quickstart.pdf
- CNC Services Northwest's web page: Advanced CNC Macro Programming Tips and Techniques for Centroid Controls. http://www.cncsnw.com/AdvancedMacroProgramming.htm

Common Uses of Macros
Machine tool Homing, Machine tool parking, custom Probing, Rack Mount tool changers the possibilities are endless.

Limitations of Macros:
Macros inherit the same limitations as when running G&M programs in MDI or running a job file. One such limitation is that macros require supporting PLC code to directly interact with outputs. Macros can read the states of inputs, outputs, and memory with the m100 and m101 command. Supporting PLC code is required for a macro to toggle an output using the m94 and m95 commands as seen in the examples below.

What is a User Variable?:

Think of a variable as a memory location that you get to use for any purpose in your macro. Just like you would write down a measurement or other number to record it in a notebook to remember it and use it at a later time a User Variable is commonly used this way in a custom macro. There are two types of user variables, volatile and nonvolatile, which are defined as variable numbers 100-149 and150-159 respectively. The value of a volatile variable as the name implies will be "erased" after the macro is finished or a power cycle. The value of a non-volatile variable will be retained after the macro is finished and through a power cycle this makes nonvolatile user variables useful in certain applications such as keeping track of what tool is in the spindle in a custom ATC macro. Throughout this document we will be assigning these user variables 100-149 and 150-159 values in the examples given. User variables are different from system variables (described below) in that they do not have interaction with any stock CNC12 function hence the name "User Variable" therefore by definition variable #'s 100-149 and 150-159 will have no conflict with the standard feature set of CNC12.

Commonly Used Expressions in Macros: (see the entire list Mill Chapter 11.2.14)

- Equals == (or "eq")
- Logical Not ! (or "not")
- Logical and &&
- Logical or ||
- Greater than > (or "gt")
- Less than < (or "lt")

How to Read and Write to a User Variable:
Macros can be used to both read values and write to some system variables in the control software. Common system variables used are user variables numbers: #100-149, non-volatile user variables numbers: #150-159 and the Tool Number indicator (#4120)
Referencing the Manuals, a variable can be read by a macro if it has an "R" in the R\W column of the chart.
If a variable has a "W" then the macro can also write to this value.

```
#100 = 1                        ; writes the value of 1 to the user variable #100
#101 = 1 + 1                    ; writes the value of 2 to the user variable #101
#103 =   [1 + #100 + #101]      ; writes the value of 4 to the user variable #103
#150 = #4120                    ; sets user variable 150 to equal the value of the Requested Tool Number *see note below
```

#4120 is the CNC12 System Variable number for the Tool Number requested by the T command. For example if you entered T9 into MDI, the software sets #4120 to the value of 9. A common use of setting a user variable equal to #4120 is this allows the author of a macro to use the current tool number in a custom m6 ATC macro for tool changes. This line is used to keep track of which tool is currently in the spindle. Examples of this are presented later in this document.

* Note: The line: "#150 = #4120" needs an important "IF #50001" to accompany it and is explained later in this document.

What is a System Variable?
A system variable is a memory location contained in CNC12 defined by a number. System variables reference values, settings, and words found in the CNC12 software. Some of the most commonly used system variables in macros are 9000 to 9999 which represent CNC12 system parameters 0 to 999. These are the same parameters that you can see in the parameter menu (F1 Setup, F3 Config, F3 Params) in CNC12, however when we want to reference these parameters in the macro, we use the system variable that represent the parameter.

Refer to mill operators manual 11.2.16 or lathe operators manual 9.2.15 for full list of CNC12 system variables.

How to interact with an Input

Inputs can be interacted with using M100/M101, M105/M106 or IF statements. M100/M101 commands simply look for the input while M105/M106 moves an axis in the minus or plus direction looking for the input and the IF statements will look for if the expression is true or not.

Example 1: Set current WCS X = 0.0000 when input 1 turns off (a Home Switch for example)

```
N100
M105 /X P1 F30          ;M105 = Move In Minus Direction, /X = Move X axis, P1 = Look at Input 1 to Open,
                        ;F30 = Move at 30" /min (Based on CNC settings, might be 30mm /min if set to metric)
G92 X0                  ;G92 = Set Absolute Position for WCS, X0 = Sets X-axis to zero
N1000
```

Example 2: Set current Machine X = 0.0000 when input 1 turns off (a Home Switch for example)

```
N100
M105 /X P1 F30          ;M105 = Move In Minus Direction, /X = Move X axis, P1 = Look at Input 1 to Open,
                        ;F30 = Move at 30" /min (Based on CNC settings, might be 30mm /min if set to metric)
M26 /X                  ;G26 = Set Axis Machine Home, /X = Sets home for the X-axis
                        ;Sets Machine Coordinates X-axis to zero
N1000
```

Example 3: Look for an input to be closed (on) before continuing the macro:

```
;Note: Input 3 has been defined as "Tool is Unclamped" using the Acorn Wizard.
;
N100                    ;Block 100 (start of macro)
M15                     ;Unclamp tool, M15 is a macro that activates the tool unclamp output
M101 /50003             ;Wait for input 3 to close (turn on) to verify that the ATC spindle has actually unclampled the tool
;                       …..the macro continues on when it sees input 3 has been made!
```

Example 4: Stop and Wait for operator with message.

In this example below a message will be displayed for the operator until the cycle start button is pressed.

```
;Note: Input 3 has been set to "Tool is Unclamped" using the Acorn Wizard, typically a ATC spindle sensor is used
;
#100 = 0            ;Sets user variable #100 (#100 is being used to set a timer for how long the Message will be displayed
;                    a value of 0 means the message will display until user presses cycle start.
; Use M225 to stop, issue message and look for input
;
IF !#50003 THEN M225 #100 "Tool in Spindle, Remove and press cycle start"
;
;If Input 3 is Open (off), Then display the message and wait for operator to press start to continue
;(See Mill Chapter 13.55 for more info on M225)
```

M225 is the display formatted string for a period of time command. In the example above, #100 is a user variable that defines the time the message will display. If a m225 command is given a zero time, it will display the message indefinitely

waiting for the cycle start key to be pressed, otherwise it will wait the amount of time specified then will continue with the macro.

Alternatively, an M00 or M01 can be used if you desire no message and for the macro to simply wait for the operator to hit cycle start. Useful for situations where the need to check over the machine during a particular part of a macro or job is required.

Example 5: Unlike Outputs, Inputs do not have to be defined in the PLC program for a macro to use them. Also, an input that is defined in the PLC program can be used by the macro not only for the intended purpose of the input but also for other functions.

For instance: Assume Input 3 is left unused in the Acorn Wizard but a Surface Plate is wired to input 3, a macro can still "see" input 3 and react to its state.

```
M115 /Z P3 F20          ;Move at fast probing rate until Surface Plate is detected
M116 /Z P3 F10          ;Retract at 10 ipm feedrate until Surface Plate clears
M115 /Z P3 F5           ;Move at 5 ipm feedrate until Surface Plate detected
G92 Z.5                 ;Set Z position to Surface Height Thickness (sets Z 0 at top of work or spoil board)
G4 P.5                  ;Wait half a second
G53 Z.5                 ;Retract Z to .5 in machine coordinates, (.5 away from Z home position)
```

Now, assume Input 3 is defined in the Acorn Wizard as "ToolTouchOffTriggered" AND a Surface Plate is also wired to the same input 3. The very same macro above can be used. It is interesting to note that a macro can "look" for a Surface Plate Detector which is also wired into the same input at the TT device saving on inputs necessary to have both Surface Plate and Tool Touch off functionality. Just keep in mind that any stock functionality of the Input assignment made in the Wizard (like the tool touch off messages and probe protection) will be enabled for that input, even though the macro is not intended to used input 3 as a Tool Touch Off.

What is a parameter?
CNC12 control configuration parameters are used to configure CNC12 to meet the CNC requirements of a particular machine tool, these parameters allow access to hard coded CNC functionality within CNC12 and allow the integrator to choose and fine tune certain aspects of CNC12 software. See the Mill Operator Manual Chapter 15.3 and the Lathe Operator Manual chapter 12.3 for details on the standard CNC12 parameters. You will notice that there are a large number of built-in functions and configurations CNC12 that are controlled by the CNC12 parameters. These parameters allow the user/integrator to pick and choose how they would like the CNC control to act or appear by editing the parameter values to select the choice or functionality they desire. For instance if we would like the CNC12 User Interface language to display in German. Set Parameter #9 value = 5, press F10 to save and bingo, CNC12 is in German.

Typically most Acorn users will never have to edit or change a CNC12 control configuration Parameter as the Wizard sets most all the Acorn the related parameters for you. But there are cases where Acorn users would have to edit a CNC12 parameter manually (such as the Language example above), when installing an ATC and a few special cases when it is desirable to edit a parameter with a Macro. A common example of this is M74, the Auto Squaring macro. In this macro Parameter 967 is edited by the M74 macro to un-pair and then re-pair the two motors on a moving gantry so independent movement can be achieved to align the two sides of the gantry to square.

Not only are Parameters defined values that represent many different features and settings for use in CNC12 but they can also be used in conjunction with the PLC program and Macros. With Macros we can read these parameters and also write to them as described below.

When would I use a Parameter instead of a Variable?

Typically one would use parameter values instead of user variables in cases where a parameter is already defined with the value we want to use and/or we want the operator to have the choice to change the value in software.

Parameters are similar to Non-volatile user variables where you can store values between jobs or through power cycles, however they can be also edited by the operator easily in the parameters menu in software. F1 Setup, F3 Config, F3 Param

The Acorn version of CNC12 makes use of several new parameters that are not in the CNC12 operators manual. A description of these special Acorn parameters are listed below. (A description of most of them are also contained in the beginning of the Acorn PLC program source file (.SRC) in the comments section as well).

Parameters 700 to 799 are allocated as special use parameters reserved specifically for the CNC12 user/integrator to use in a custom application. This reservation of P700-799 allows the user/integrator to create their own custom parameters in both Macro and PLC programs and then the user/integrator can be sure that P700-799 will not be used by CNC12 or a Centroid Standard PLC program in the future, this ensure that any custom PLC or Macro made today using P700-799 will not have a conflict with future CNC12 updates. So, if you need to make a custom parameter use any one in the P700 to 799 range.

How to read a parameter value.
In reference to the system variable chart in the mill or lathe manual, we can find that 9000-9999 are the system variables for parameters 0-999. These are the same parameters that you can find in the parameters menu within CNC12 (F1 setup, F3 config, F3 Params). In Macros, it is very simple to read from the system variable parameter values. We will typically use parameter values instead of user variables in cases where a parameter is already defined with the value we want to use and/or we want the operator to have the choice to change the value in software. We simply need to assign a variable to a parameter, for example:

```
#101 = #9710            ;Sets user variable 101 to the value of parameter 710
```

System Variables do not always need to be assigned to a user variable, most often they can be used directly in the logic.

```
G92 Z[#9710]            ;Set WCS Z position to value of Parameter 710
```

How to write a parameter value.
Referencing the system variables chart in the manual you can read the system variables 9000-9999 (parameters 0-999) as indicated by the R in the R\W column, but we cannot directly write to them as indicated by no “W” in the R\W column. However using the G10 function, a macro can write to these system variables. An example:

```
#103 = 5                ;Sets User variable to value of 5
G10 P710 R[#103]        ;Sets Parameter 710 to the value of #103
```

Typically in macros we will reference values via system variables, however functions such as G10 only requires the parameter number, and does not use the system variable that corresponds to said parameter. Trying to write to a read only variable will result in an error being displayed in the software when trying to run the macro, examples of some cases where this error will appear.

NOT ALLOWED:

```
#9710 = #101            ;This cannot be done, #9710 cannot be directly written to, only read.
#4120 = 5               ;This cannot be done, #4120 is a read only variable, 4120 value is set/changed by CNC12
```

To reiterate, the above is strictly not allowed in this format, but parameter 710 can be written to using the G10 command.

```
G10 P710 R3             ;Sets parameter 710 to the value of 3
```

Similarly, #4120 can be written to by using the T command,

```
T10                     ;Sets current tool to tool 10, Really this sets #4120 to the value of 10.
```

There are Parameters that are considered “machine setup” related and are Read only (a macro cannot write to them).

Most are documented in the CNC12 operators manual. Below is an example of a parameter that can not be written to.

```
G10 P71 R5              ;Will display “704 G10 error:invalid P” in software
                        ;Parameter 71 is a “machine setup parameter and is unable to be changed by G10.
```

“Machine Setup” parameters are read only, they can only be written to by either the Wizard or the parameter menu in CNC12: F1(Setup), F3(Config), F3(Parms).

These parameters can still be read by the macro like other parameters.

```
#100 = #9071            ;Set user variable #100 to value of parameter 71
G92 Z[0+ABS[#9071]]     ;Set Z position to 0 + detector height stored in parameter 71
G92 Z[.75+ABS[#100]]    ;Set Z position to .75 + value stored in user variable 100
```

Lathe Vs Mill Software:
Always reference the correct operators manual (Mill or Lathe) for each version of CNC12 you are working with. The examples in this guide are written with the Mill CNC12 in mind and the same general rules apply to both versions of software. However, G and M codes can be different in either versions of software so, it is important to reference the operators manual for the correct use of G-code. For example, the G10 command is different in functionality in the Lathe CNC12.

Lathe CNC12

```
G10 P5 Z-2              ;Sets tool #5 Z-axis offset to -2 in Offset Library
G10 P1710 R[#103]       ;Sets Parameter 710 to the value of #103
```

G10 in the lathe software sets the tool offsets but also writes to parameters based on the P number. If the P number is a number between 1 and 99 (Pnn, where “nn” is the number) then the corresponding tool offset is set. So “G10 P3 X2” would set the tool X-axis offset of tool 3 to value of 2. If the P number is a number between 1000 and 1999 (P1nnn, where “nnn” is the parameter number) then the parameter is written to. So “G10 P1710 R2” would set the value of 2 to parameter 710.

Difference between = and ==

== is used to compare two variables while = is used to write to a variable or in other words assign to the variable.
For example:

```
#100 = 1          ; writes the value of 1 to the variable #100
#101 = 2          ; writes the value of 2 to the variable #101
IF #100 == #101 ….      ;Compare Variable #100 and #101, if they are equal than the expression is true.
```

Alternatively

```
IF #100 eq #101 ….
```

For more information on expressions, symbols, and letters used, reference the mill operators manual Chapter 11 or lathe operator manual Chapter 9

Using “IF THEN ELSE”:

IF <Expression> THEN <Execute if True> ELSE <Execute if False>

Macro Programming using IF THEN or IF THEN ELSE, are commonly used in macros that require different actions to be taken based on expression. The expression statement can be a variety of logic, for example it can look at inputs, look at expressions, and more.
Below are a few IF THEN, and IF THEN ELSE examples:

```
IF #50005 THEN GOTO 200
                    ;If Input 5 is closed (Green) then goto block 200

IF [#101 == 1 || #101 + #102 == 3] THEN GOTO 200
                    ;If #101 = 1 or #101+#102 = 3 then goto 200
```

- The Execution statement can be a variety of different commands, most commonly GOTO is used as the execution.

```
IF #50005 THEN M5
                    ;If Input 5 is closed then execute M5 (Stops spindle)
#100 = 0            ;Sets user variable #100 to value of 0.
IF #50005 THEN M225 #100 “Input 5 is on”
                    ;If Input 5 is closed then display message “Input 5 is on”
```

- An ELSE is used if a separate execution is desired on a false expression.

```
IF #50005 THEN M5 ELSE GOTO 200
                    ;If Input 5 is closed then execute M5, If Input 5 is open then goto block 200
#100 = 0            ;Sets user variable #100 to value of 0.
IF #50005 THEN M225 #100 “Input 5 is on” ELSE M225 #100 “Input 5 is Off”
                    ;If input 5 Is closed then display “Input 5 is on”, Otherwise display “Input 5 is off”
```

IF statements when looking at inputs are not inverted if the operator inverts the input in the software (Alt-I screen then Ctrl-Alt-I on an input). The IF statement is still looking if the said input is On (Green) or Off (Red) regardless of inversion. Refer to Mill Operators Manual 11.3.2 page 203 or Lathe Operators Manual 9.3.2 page 154 for more information.

Block Numbers and GOTO Command:
Block Numbers are used to identify program lines and are designated by an N# (# being a number). These blocks also act as destinations for the GOTO Command. Example:

```
N100
;Input 5 is a tool sensor in the spindle
IF #50005 THEN GOTO 500     ;IF Input 5 is on, then GOTO Block 500 (N500)
M15                         ;Unclamp Tool
GOTO 1000                   ;GOTO End of macro
N500                        ;Block 500
M16                         ;Clamp Tool
N1000                       ;End of Macro
```

In this example we set up a macro to detect input 5 and perform two different actions based on the state of the input. If the input was off, then the macro will perform an M15 then end the macro. If the input was on, then the macro will jump to block 500 and perform an M16 and end the macro. This logic that “skips part of the macro” to perform a different action for example, is used in ATC macros to skip the “put tool away” section of the macro if there is no tool in spindle when the macro is called.

How to use a macro to interact with an Output:

Macros can be used to activate and deactivate outputs for a wide variety of uses.
CNC12 has two built in M-codes just for the purpose of turning ON and OFF an output: M94 and M95
In order to use M94 and M95 with a particular output number that output number must be defined in the Acorn PLC program. Defining an Acorn output definition can be done one of two ways: 1.) Use the Acorn Wizard and assign an output definition to the output number that you plan on using with the M94/M95. 2.) Edit the Acorn PLC program source file and add in a customized output and compile the PLC program (see example of this in appendix A at the end of this document).

For the following examples, using the Acorn Wizard, we assign the output DustFootActivate to Output 3, and assign the input ToolisUnclamped to Input 2.

Example 1: Assign M61 to turn on Output 1. The Acorn Wizard has canned PLC Output definitions: OUTPUT1, OUTPUT2, OUTPUT3, etc.. that have preassigned macros: M61,M62,M63 etc..respectively. Using the Wizard assign OUTPUT1 to Output 1, “Write Settings to CNC control Configuration” follow the instructions and then open M61 in Notepad++. Note: A semi colon ";" tells CNC12 that the line is a comment, remove semi colon for command to be run, add semi colon to beginning of line to be ignored.

```
;-------------------------------------------------------------------------------
; Filename: mfunc61.mac
; Wizard OUTPUT1 M-code Macro: M61
; Description: User Customizable Macro
; Notes: Use Acorn Wizard i/o map to set Acorn Output 1 = to "OUTPUT1" then this macro (M61) will turn on that output
; Requires:
; Please see TB300 for tips on writing custom macros.
;-------------------------------------------------------------------------------

IF #50010                    ;Prevent lookahead from parsing past here
IF #4201 || #4202 THEN GOTO 1000 ;Skip macro if graphing or searching

N100                         ;Insert your code between N100 and N1000

M94 /61                      ;Request OUTPUT1

N1000                        ;End of Macro
```

Now in MDI type “M61” then press Cycle Start and Output1 will turn on, press cycle cancel or enter M81 and cycle start and Output1 will turn off.

Example 2: Adding a delay after Output1 is turned on for 4 seconds to M61.

```
N100

M94 /61          ;Activate Output1
G4 P4            ;Wait 4 seconds. Then continue on

N1000
```

Example 3: Activate output when an input is activated.

```
N100

M94 /61                          ;Activate Output1
G4 P4                            ;Wait 4 seconds
IF #50002 THEN M94 /28           ;If Input 2 is closed then activate “DustFootActivate”
```

```
N1000
```

Example 4: Adding an input for the operator to choose to turn on or keep off RouterVacuumHoldDown Output during macro operation using M224.

```
N100
G94 /61                                 ;Request Output1
G4 P4                                   ;Wait 4 seconds
IF #50002 THEN M94 /28                  ;If Input 2 is closed then activate “DustFootActivate”

N200
;Display message “Turn on RouterVacuumHolddown?”
;                       To Turn on Enter “1”
;                       To Turn off Enter “2”
M224 #100 “Turn on RouterVacuumHoldDown? \n To Turn on Enter 1 \n To Turn off Enter 2” #105
IF [#100 < 1 || #100 > 2] THEN GOTO 200     ;If operator inputs invalid number, repeat question.
IF #100 = 1 THEN M94 /5                     ;If operator inputs 1, then activate RouterVacuumHoldDown
                                            ;If operator input 2, then macro continues without activating output
N1000
```

Note: These M94/M95 values are found in v4.20+ stock Acorn PLC programs that can be utilized in any macro as is. Also see Acorn Wizard Input and Output “canned” PLC functions and M codes list.
https://www.centroidcnc.com/centroid_diy/downloads/acorn_documentation/acorn_wizard_input_output_plc_functions.pdf

```
SV_M94_M95_1    ;M94/M95 Turn on/off Spindle On Clockwise
SV_M94_M95_2    ;M94/M95 Turn on/off Spindle On Counter Clockwise
SV_M94_M95_3    ;M94/M95 Turn on/off Flood or RouterDustCollection
SV_M94_M95_4    ;M94/M95 Turn on/off OpenChuck, Turn off CloseChuck
SV_M94_M95_5    ;M94/M95 Turn on/off Mist or RouterVacuumHoldDown
SV_M94_M95_6    ;M94/M95 Turn on/off CloseChuck, Turns off OpenChuck
SV_M94_M95_8    ;M94/M95 Turn on/off LubePump Manually
SV_M94_M95_10   ;M94/M95 Turn on/off TurnClampOn
SV_M94_M95_13   ;M94/M95 Turn on/off Cutoff
SV_M94_M95_15   ;M94/M95 Turn on/off ToolUnclamp
SV_M94_M95_19   ;M94/M95 Turn on/off OrientSpindle
SV_M94_M95_22   ;M94/M95 Turn on/off PartChute
SV_M94_M95_27   ;M94/M95 Turn on/off VacuumOn
SV_M94_M95_28   ;M94/M95 Turn on/off DustFootActivate
SV_M94_M95_29   ;M94/M95 Turn on/off LaserAlignActivate
SV_M94_M95_30   ;M94/M95 Turn on/off PopUpPins
SV_M94_M95_31   ;M94/M95 Turn on/off SpindleCooling
SV_M94_M95_32   ;M94/M95 Turn on/off TailStockInOut
SV_M94_M95_35   ;M94/M95 Turn on/off DustCollectionOn
SV_M94_M95_41   ;M94/M95 Turn on/off Spindle Low Range
SV_M94_M95_42   ;M94/M95 Turn on/off Spindle Medium Range
SV_M94_M95_43   ;M94/M95 Turn on/off Spindle High Range
SV_M94_M95_61   ;M94/M95 Turn on/off OUTPUT1
SV_M94_M95_62   ;M94/M95 Turn on/off OUTPUT2
SV_M94_M95_63   ;M94/M95 Turn on/off OUTPUT3
SV_M94_M95_64   ;M94/M95 Turn on/off OUTPUT4
SV_M94_M95_65   ;M94/M95 Turn on/off OUTPUT5
SV_M94_M95_66   ;M94/M95 Turn on/off OUTPUT6
SV_M94_M95_67   ;M94/M95 Turn on/off OUTPUT7
SV_M94_M95_68   ;M94/M95 Turn on/off OUTPUT8
```

These M94/M95 values do not always need to be used in a macro, Running an M94 /15 command in MDI or on a line in a job file for example will activate the logic to unclamp tool. Putting these M94 and M95 commands in macros is usually preferred as additional desired logic is accompanied with these commands such as M101/M100 in most cases.

Subprograms M98/M99:
Macros can be used to call additional subprograms and can have subprograms embedded into the macro itself. Embedded subprograms is a common technique used for Rack Mounted Automatic Tool Change (ATC) macros. Embedded subprograms within the macro are noted by beginning with an O (“O” as in Oscar) followed by a 9100 to 9999 number. These particular subprograms are special in that they self extract themselves out of the macro and return to the macro when finished. An example of an embedded subprogram:

```
;This macro is for a tool rack tool changer with 2 tool positions with no forks (drops tool in pot).
;Subprograms are used to call the position of the tool requested in the X and Y coordinates, “O9101” and “O9102”.

;#4120 = Value of Tool Number requested, this is read only, CNC12 sets this value depending tool number requested by pro
gram or user.
;#100 = -5                       ;Z height of the tool rack

O9101                           ; Embedded Self Extracting SubProgram 9101 that defines the tool pocket location
                                ; Enter the X coordinate of Tool #1
G53 X100 Y10                    ; X and Y Coordinate of Tool # 1
M99                             ; End of subprogram return to main program

O9102                           ; Embedded Self Extracting SubProgram 9102 that defines the tool pocket location
                                ; Enter the X coordinate of Tool #2
G53 X200 Y10                    ; X and Y Coordinate of Tool # 2
M99                             ; End of subprogram return to main program

G90 M98 P[#4120 + 9100] ; Tool Number Requested + 9100, then calls subprogram to move to X and Y coordinates
G53 Z[#100]
```

Refer to mill operators manual 11.2.6 and 13.27 pages 194 and 251 or lathe operators manual 9.2.3 and 11.26 pages 145 and 194.

Creating Custom Operator Prompt Messages that appear when running the Macro to prompt the Operator with information or action needed.

CNC12 can display a custom message by using a file called cncxmsg.txt. For example: This file can be used with macros to display to an operator prompt messages when the macro is waiting on an input with the m101 and m100 commands. To customize a message create a “cncxmsg.txt” file and save it in the cncm directory for mill or the cnct directory for lathe. Once created, if we wanted a custom message to appear with a M100 or M101 command in the macro for input 6, we will input the following into the cncxmsg.txt file.

```
INP6
“My Custom Message for Input 6”
OUT1
“My Custom Message for Output 1”
MEM2
“My Custom Message for Memory 2”
```

Format for cncxmsg.txt is first line state the Input, Output, or Memory and then on the next line the custom message in quotations. Outputs and memory bits can be given their own messages using OUT# for outputs and MEM# for Memory bits (# is the number of output or memory bit).

To use this custom messages created in the cncxmsg.txt file all you have to do is use the M100 or M101 in the macro with the corresponding Input, Output, or Memory number. For example.

```
M101 /50006     ; Will now display the custom message from cncxmsg.txt instead of default “waiting for input #6 (M101)”
M101 /60001     ; Will now display the custom message from cncxmsg.txt instead of default “waiting for output #1 (M101)”
M101 /70002     ; Will now display the custom message from cncxmsg.txt instead of default “waiting for memory #2 (M101)”
```

When Skip Graphing, Skip Searching and Stop Look ahead is needed.
You will commonly see the following lines in Centroid macros.

```
IF #4202 || #4201 THEN GOTO 500          ;Skip if CNC12 is graphing Gcode or searching for Gcode
N500                                     ;End of Macro
```

and

```
IF #50001                                ;Force look ahead to stop processing
```

The first line is used to skip the macro if the software is in graphing or search mode. Search and graphing modes process through both the job and macro files. If there is code in a macro that could cause a loop, it is possible for the search or graph to get “hanged up” at this loop and would no longer continue through the rest of the graph or search. Such loops exist for example in an M19 Spindle orient macro, which may have a loop that tells the operator to turn on auto spindle. This loop can cause the graph or search to get hanged up on the M19 line of the job file, or even the m19 line in an M6 macro.

The CNC12 system variable #4201 is a value of 0 when running a job, or a value of 1 when performing a graph preview. The CNC12 system variable #4202 is a value of 0 when the CNC12 is running G-codes normally or a value of 1 or greater when the CNC12 is not running G-codes normally for example when doing a search. For the case of the G-codes not running normally, the commands are not being sent to the machine but rather CNC12 is running through the g-code to determine the correct state of the machine up to the point that is being searched. If either one of these values are a non-zero then the expression is true and we GOTO the last line of the macro ending it.

The second piece of logic IF #50001 is used to stop the CNC12 G-code look-ahead from processing past that point. This is used mainly when you need to ensure a value or an action is not processed by the G-code look-ahead prior to when the program itself reaches that point in the macro. The CNC12 actively “looks ahead” in the job file and macro to anticipate

changes, and it will change values before the actual program reaches this point. Let us look at the following example: using the “put away tool” section of the Rack Mount ATC Macro example, which is found later in this document.

```
;--Put Tool Away Block 100
N100
IF #150 == 0 THEN GOTO 200
G90 M98 P[#150 + 9100]                    ;Go to X Put away Location
G53 Z[#100]                               ;Move to Z Tool Rack Height
G53 G1 Y[#102] L10                        ;Put Tool in Tool Rack
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
M63                                       ;Turn on Output 3 (Open Valve)
G4 P0.5                                   ;Dwell 0.5 Seconds
M100 /50006                               ;Check if Tool Released
G53 Z[#101]                               ;Move to Z Tool Height
M83                                       ;Turn off Output 3 (Close Valve)
G28 G91 Z0                                ;Move to Z Clearance Height
IF #50001                                 ;Force look ahead to stop processing
#150 = 0                                  ;Set No Tool in Spindle
IF #4120 == 0 THEN GOTO 500
```

For example, if an issue happened and the operator had to press E-Stop between the lines G53 and M63 represented by the ~~~~ line above. Since we had IF #50001 before the line #150 = 0 there were no changes to the Variable #150 as CNC12’s program execution had not reached that line of the macro yet. However, if “IF #50001” was not present before the line #150 = 0, it is possible that even though the operator had pressed E-Stop before the program execution reached the #150 = 0 line. CNC12’s G-code look-ahead may have already assigned the value 0 to variable #150. So the next time the operator runs this macro, the macro will now think #150 is equal to zero even though there is still a tool in the spindle.

It is important to use IF #50001 to stop CNC12’s G-code program look-ahead from processing past a point when you want to make sure the program execution actually made it to that point before changing the variable. In this case we want to make sure that the tool has actually been put away before telling the control that there is no longer a tool in the spindle #150 = 0. The IF #50001 command does not need to be used all the time, only when it is desirable to stop CNC12’s G-code look-ahead from going past a point until that point is reached in the G-code program.

You may have noticed that #50001 is the system variable for input 1. The line IF #50001 means IF Input 1 is closed, then do nothing. Any CNC code that is conditional on a PLC bit is intended to be tested at the time the code actually runs. Therefore, CNC12’s G-code program look ahead waits at the line IF #50001 until the execution has actually caught up to this line. Any PLC bit will work to stop CNC12’s g-code program look-ahead, however #50001 is the common convention.

Skip Graphing Notes:

- There is no need to skip around M0, M1, M100 or M101. CNC12 knows, during search and graph, that those commands do not do anything interesting (that is, anything that affects tool movement or CNC modal values).

- When Skipping is not desirable.

Often times it is desired to save control information, this can be done via a parameter or a user variable as discussed previously. However, we may not always want to skip updating a variable during a search or graph. Lets look at two examples, the first is rather familiar, we will update the “tool in spindle” for a Rack Mount Tool ATC. The second example we will be storing the desired RPM of a sub spindle to a parameter. We choose a parameter instead of a non-volatile user variable for example 2 so the PLC can also retrieve this information. Assume for example 2 we will be writing different RPM values throughout a program, 3500 is just for example purposes.

Example 1 :

```
#150 = #4120              ;Record requested tool as the new tool in spindle
N500                      ;End of Macro
```

Example 2 :

```
G10 P702 R3500          ;Store desired RPM of 3500 to parameter 702
M33                     ;Subspindle Start request to PLC
N500                    ;End of Macro
```

For both examples, it is desirable that we stop CNC12's G-code Look-ahead, as we do not want to update the value of the RPM or the tool before CNC12's G code execution catches up. Additionally, for example 1, we do not want to update the tool in spindle value unless a tool change has been completed. So we will add skip graph and search to Example 1. However, for example 2, we will want to skip when graphing, however we may want it to update the values if searching. Lets look at the examples again with the added code,

Example 1 with Stop Look ahead and Skip Graphing and Searching :

```
IF #50001                                       ;Stop software look ahead
IF #4201 == 0 && #4202 == 0 THEN #150 = #4120   ;Record requested tool as new tool in spindle if not graphing or searching
N500                                            ;End of Macro
```

This is different from previous examples, in this case we are simply not updating the value if graphing or searching instead of skipping the full macro.

For example 2, it is desirable to allow CNC12 to update the parameter as we search. If we do not, then we run the risk of running with an incorrect RPM if we wanted to start a job at a mid-point. (use CNC12's Search feature to start at almost any point in a G code program. F4 Run, F2 Search and type in the line, block or tool number to start at.) Since we are no longer skipping the macro while searching, CNC12 will run through the program and update the parameter when ever it is called in the program. This will give us the correct state of desired RPM (Parameter 702) up to the point in which we searched for.

Example 2 with Stop Look ahead and Skip Graphing but allow Searching :

```
IF #50001                               ;Stop software look ahead
IF #4201 == 0 THEN G10 P702 R3500       ;Store desired RPM of 3500 to Parameter 702 if not graphing
M33                                     ;Subspindle Start request to PLC
N500                                    ;End of Macro
```

Bringing it all together, a Rack Mount ATC Macro Example: Four tool position rack mount ATC with integrated auto tool measurement touch off cycle.

```
;---------Tool Pocket Location Definition Section---------
O9101
;Edit the line below with the X coordinate of Tool #1 pocket location
G53 X100
M99

O9102
;Edit the line below with the X coordinate of Tool #2 pocket location
G53 X200
M99

O9103
;Edit the line below with the X coordinate of Tool #3 pocket location
G53 X300
M99

O9104
;Edit line below with the X coordinate of Tool #4 pocket location
G53 X400
M99
;---------End Tool Location Section---------

; Variable Definitions:
          ;#150 = Tool currently in the spindle
#100 = -90                                  ;Z height of tool rack
#101 = -60                                  ;Z height of Tool (Position where spindle is right above the tool)
#102 = -8                                   ;Y Tool Location
#103 = -70                                  ;Y Clearance Location
#104 = #50002                               ;Tool Touch Tripped Input
;#9014                                      ;Fast Probe Rate Parameter 14
;#9015                                      ;Slow Probe Rate Parameter 15
;#9071                                      ;Tool Touch Off Height, Parameter 71

IF [[#4120] == [#150]] || #4202 || #4201 THEN GOTO 500 ;skip if graphing or searching or if at the same tool

;--Prepare for Tool Change
M5                                                      ;Stop Spindle
G28 G91 Z0                                              ;Move to Z Clearance Height
G53 G90 Y[#103]                                         ;Move to Y Clearance Location
G4 P1                                                   ;Dwell for 1 Seconds

;--Check if Tool in Spindle
N50
IF !#50006 THEN GOTO 200 ELSE GOTO 100                  ;If Tool not in spindle skip Put Away

;--Put Tool Away Block 100
N100
IF #150 == 0 THEN GOTO 200
G90 M98 P[#150 + 9100]                                  ;Go to X Put away Location
G53 Z[#100]                                             ;Move to Z Tool Rack Height
G53 G1 Y[#102] L10                                      ;Put Tool in Tool Rack
M63                                                     ;Turn on Output 3 (Open Valve)
G4 P0.5                                                 ;Dwell 0.5 Seconds
M100 /50006                                             ;Check if Tool Released
G53 Z[#101]                                             ;Move to Z Tool Height
M83                                                     ;Turn off Output 3 (Close Valve)
G28 G91 Z0                                              ;Move to Z Clearance Height
IF #50001                                               ;Force look ahead to stop processing
#150 = 0                                                ;Set No Tool in Spindle
IF #4120 == 0 THEN GOTO 500
```

```
;--Get Requested Tool Block 200
N200
G90 M98 P[#4120 + 9100]                ;Go to X Putaway Location
G53 Y[#102]                            ;Move to Y Tool Location
G53 Z[#101]                            ;Move to Z Tool Height
M63                                    ;Turn on Output 3 (Open Valve)
M101 /50007                            ;Check for Input 7 (Valve Open)
G53 Z[#100]                            ;Move to Z Tool Rack Height
M83                                    ;Turn off Output 3 (Close Valve)
G4 P0.5                                ;Dwell for 0.5 Seconds
M101 /50006                            ;Check for Tool Clamped
IF #50001                              ;Force lookahead to stop processing
#150 = #4120                           ;Set Tool to Requested Tool
G53 Y[#103] L10                        ;Move to Y Clearance Location
G28 G91 Z0                             ;Move to Z Clearance Height
GOTO 300

;--Tool Touch Off Cycle Block 300
N300
G30 G91 X0                             ;Move to X Tool Touch Off Location
G30 G91 X0Y0Z0                         ;Move to Tool Touch Off Reference Location
M115 /Z P[#104] F[#9014]               ;Move at fast probing rate until TT1 detected
M116 /Z P-[#104] F[#9015]              ;Retract at slow probing rate until TT1 clears
M115 /Z P[#104] F[#9015]               ;Move at slow probing rate until TT1 detected
G92 Z[0+ABS[#9071]]                    ;Set Z position to 0 + detector height stored in parameter 71
G4 P1                                  ;Wait 1 second
G28 G91 Z0                             ;Move to Z Clearance Height
G90
GOTO 500

N500 ;End of program
```

Rack Mount Macro Notes:

- The Tool Touch Off device should be setup with the Acorn Wizard and functionality tested before running a macro or tool touch off cycle.
- TT input defined and correct input defined in the macro, user variable #104
- TT subtracts height of TT based on parameter 71. If not desired, edit the macro and remove ABS[#9071] from the “G92 Z[0+ABS[#9071]]” line near the end of the macro.

- Machine tool specific Tool Bin locations of each tool must be manually entered into the macro.
- User variables #100-#103 will most likely need adjusted for each rack mount ATC System.
- It is possible to add more tool slots by adding more Subprograms O910# at the beginning of the macro.

Frequently Asked Questions:

1.) IF an input is Inverted in software, are M100/M101 or IF #5000# expressions Inverted?
These are not inverted, they are still looking at the state of the input (On or off, or in other words Green or Red in the I/O Diagnostic screen Alt-I). For example,

In the case Input 1 is off and not inverted, (shows up red)

```
IF #50001 THEN...     This will turn a false, since the input is off.
M101 /50001           This will be false with message “Waiting for Input 1”
M100 /50001           This will be true, Macro will continue
```

in the case Input 1 is off and inverted (shows up green)

```
IF #50001 THEN...     This will turn a true in this case, since the input is on.
M101 /50001           This will be true, Macro will continue
M100 /50001           This will be false with message “Waiting for Input 1”
```

in the case input 1 is on and not inverted, (shows up green)

```
IF #50001 THEN...     This will turn a true in this case as well, since the input is on.
M101 /50001           This will be true, Macro will continue
M100 /50001           This will be false with message “Waiting for Input 1”
```

in the case input 1 is on and inverted, (shows up red)

```
IF #50001 THEN...     This will turn a false in this case, the input is off
M101 /50001           This will be false with message “Waiting for Input 1”
M100 /50001           This will be true, Macro will continue
```

2.) When to use F or L for Feed Rate?
Use F to set the feed rate for non-rapid moves such as G1, while L is used in some G-codes that would otherwise use the Rapid rate. For example, the L command when used with a G53 (rapid move in machine coordinates) will make the G53 move at the user specified feedrate rather than the Rapid rate. Note: The L value also does not override the feed rate for other moves, Look at the two cases below.

Case1:

```
G1 X1 F100            ;Moves X at 100 units/minute
G1 Y2                 ;Moves Y at 100 units/minute
G53 Z-5 L20           ;Moves Z at 20 units/minute
G1 Y0                 ;Moves Y at 100 units/minute
```

Case2:

```
G1 X1 F100      ;Moves X at 100 units/minute
G1 Y2           ;Moves Y at 100 units/minute
G53 Z-5 F20     ;Moves Z at Max Feed units/minute. The F20 does not effect G53 and is ignored as it defaults to rapid rate
                ;Note: G53 will move at rapid rate unless given L value to command the velocity desired
G1 Y0           ;Moves Y at 100 units/minute
```

G-codes that uses L for feedrate is G28, G30, G30 P3 and G30 P4 and G53. The L allows you to control the rate in which these g codes will move the machine tool. If no L command is specified these commands move at the machines Rapid Rate.
Note: L is not always used for feedrate, for example G65 uses L for the repeat value and not for feed rate.

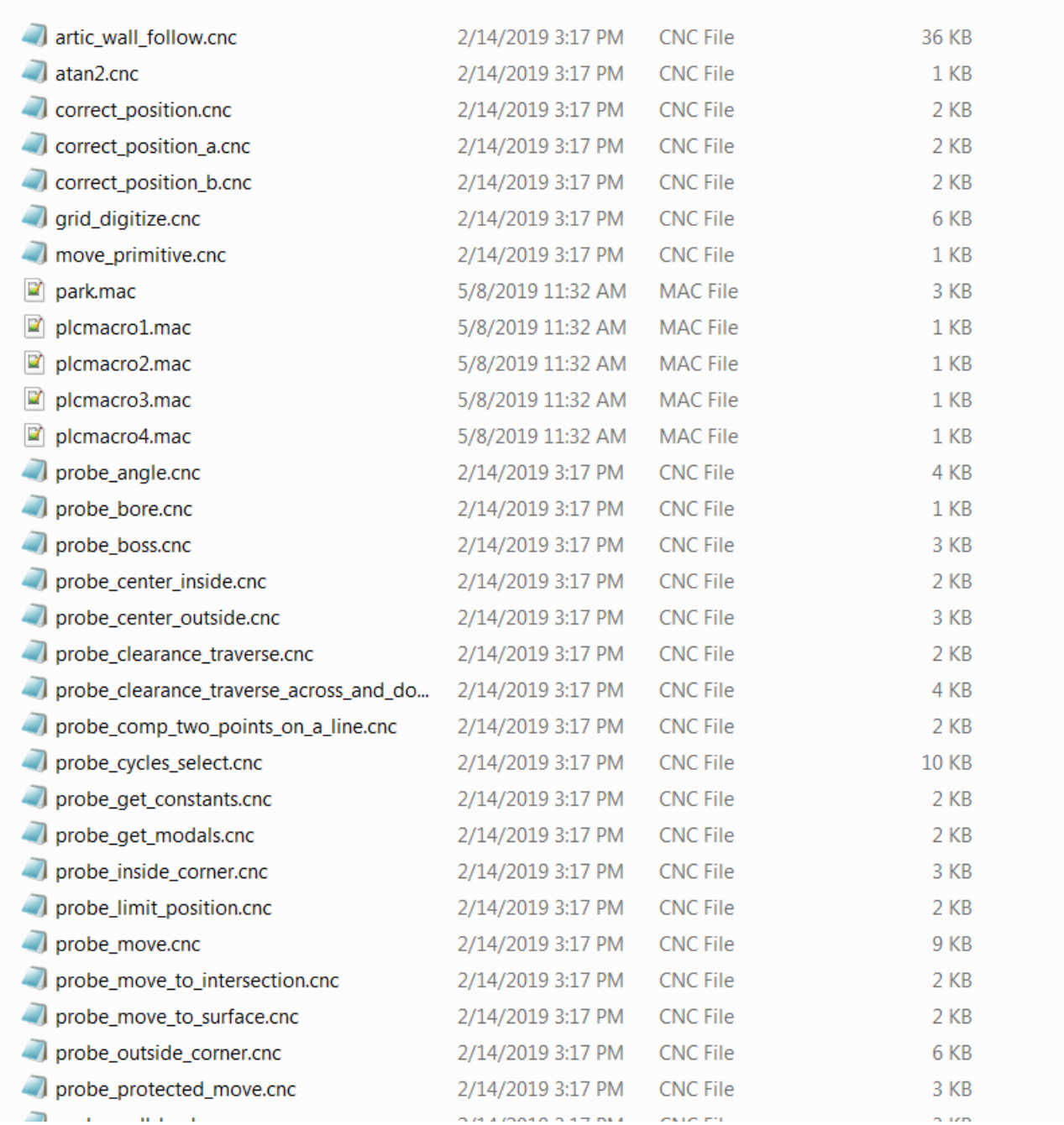
3.) ***Where does variable #150 come from in ATC Macro?***

Variable #150 is a non-volatile variable, meaning it retains its value after the macro ends and through power cycles. Variable #150 is needed in tool rack mount ATCs to keep track of which tool is currently in the spindle. When calling a tool change, the macro uses the value of #150 to determine where to put the current tool back before trying to pick up the next tool.
We use #150 simply due to it being the first non-volatile number, you can use any non-volatile variable for this or even a parameter (however this can cause issues if the operator changes the parameter manually). At the end of the macro we set #150 to equal the requested tool #4120 after the tool change has been completed. This updates the value #150 with the tool currently in spindle, for use when the tool change is called again.

The macro its self is what sets and reads variable #150.\

4.) How do I create a Probing macro?

There are a fair number of stock CNC12 probing macros contained in the cncm\system folder.



artic_wall_follow.cnc	2/14/2019 3:17 PM	CNC File	36 KB
atan2.cnc	2/14/2019 3:17 PM	CNC File	1 KB
correct_position.cnc	2/14/2019 3:17 PM	CNC File	2 KB
correct_position_a.cnc	2/14/2019 3:17 PM	CNC File	2 KB
correct_position_b.cnc	2/14/2019 3:17 PM	CNC File	2 KB
grid_digitize.cnc	2/14/2019 3:17 PM	CNC File	6 KB
move_primitive.cnc	2/14/2019 3:17 PM	CNC File	1 KB
park.mac	5/8/2019 11:32 AM	MAC File	3 KB
plcmacro1.mac	5/8/2019 11:32 AM	MAC File	1 KB
plcmacro2.mac	5/8/2019 11:32 AM	MAC File	1 KB
plcmacro3.mac	5/8/2019 11:32 AM	MAC File	1 KB
plcmacro4.mac	5/8/2019 11:32 AM	MAC File	1 KB
probe_angle.cnc	2/14/2019 3:17 PM	CNC File	4 KB
probe_bore.cnc	2/14/2019 3:17 PM	CNC File	1 KB
probe_boss.cnc	2/14/2019 3:17 PM	CNC File	3 KB
probe_center_inside.cnc	2/14/2019 3:17 PM	CNC File	2 KB
probe_center_outside.cnc	2/14/2019 3:17 PM	CNC File	3 KB
probe_clearance_traverse.cnc	2/14/2019 3:17 PM	CNC File	2 KB
probe_clearance_traverse_across_and_do...	2/14/2019 3:17 PM	CNC File	4 KB
probe_comp_two_points_on_a_line.cnc	2/14/2019 3:17 PM	CNC File	2 KB
probe_cycles_select.cnc	2/14/2019 3:17 PM	CNC File	10 KB
probe_get_constants.cnc	2/14/2019 3:17 PM	CNC File	2 KB
probe_get_modals.cnc	2/14/2019 3:17 PM	CNC File	2 KB
probe_inside_corner.cnc	2/14/2019 3:17 PM	CNC File	3 KB
probe_limit_position.cnc	2/14/2019 3:17 PM	CNC File	2 KB
probe_move.cnc	2/14/2019 3:17 PM	CNC File	9 KB
probe_move_to_intersection.cnc	2/14/2019 3:17 PM	CNC File	2 KB
probe_move_to_surface.cnc	2/14/2019 3:17 PM	CNC File	2 KB
probe_outside_corner.cnc	2/14/2019 3:17 PM	CNC File	6 KB
probe_protected_move.cnc	2/14/2019 3:17 PM	CNC File	3 KB

We have begun to “breakout” as many CNC12 built in probing cycles into editable macros. Copying and then editing these or just opening and learning from them is a great place to start if you are interested in learning more about creating your own custom probing macros.

That being said here is an introduction to controlling a touch probe with a macro.
Unless otherwise noted, the methods and examples below are for current CNC12 Mill software.

5.) How do I control a touch probe?

Use the M115, M116, M125 and M126 codes to move a touch probe in custom probing cycles.

Whenever possible, specify a bounding position, even with M115 and M116 codes.

Note that the positions you specify with these codes are either absolute positions or incremental distance, depending on the current G90/G91 state.

For example:

```
G90
M105/X2.0/Y3.0 P15 F20
```

will move towards an absolute position of X2 Y3, at 20 in/min, until the switch on INP15 closes.

In contrast:

```
G91
M105/X2.0/Y3.0 P15 F20
```

will move in a direction of +2" along X, +3" along Y, at 20 in/min, until the switch on INP15 closes.

Also note that M115 and M116 are interchangeable (as are M125 and M126) when bounding positions are provided, because the direction to the bounding position determines the direction of movement.

Use M115 and M116 when you intend to locate a surface with the probe. The bounding position, if given, should be beyond the farthest likely position of the surface. With M115 and M116, if the movement proceeds all the way to the bounding position without triggering the input (tripping the probe), then the cycle will be canceled with an Error.

Use M125 or M126 for positioning moves during which no probe contact is expected.

Like M115 and M116, M125 and M126 will stop if the input is triggered (the probe is tripped); but M125 and M126 will then cancel the cycle with an error. This protects against probe breakage in case of an unexpected obstacle.

M115, M116, M125 and M126 are most commonly used with a touch probe, but they can be used with any switch input, or other PLC bit.

The P parameter used with these codes is either positive or negative, depending on whether you are watching for an open switch to close, or watching for a closed switch to open.

6.) What are the differences between M105 and M115? (Switch Sense for M1x5 and M1x6 Codes)

Note: The M105 and M106 codes, which are similar to M115 and M116, may also interpret the sign of the P parameter but differently. The behavior for positive and negative P values with the M105/106 and M115/116 is as follows:

Code	CNC11/CNC12	
	+P	-P
M105/M106	Move until open (1 -> 0)	Move until close (0 -> 1)
M115/M116/ M125/M126	Move until close (0 -> 1)	Move until open (1 -> 0)

7.) How do I create a custom home program?

The M105 and M106 codes are useful for axis homing, in situations where the normal M91 and M92 codes are not well suited. For example, suppose we have a mill X axis which we want to quickly home near the center of travel, regardless of where the axis may be be sitting when the machine is powered up.

We install a dedicated home switch, which will be a normally-open proximity sensor that detects a steel rib that runs from near the mid-point, all the way to the plus limit of travel. We wire it to INP7 on the Acorn board. Therefore, we expect INP7 to be open when we are in the minus half of travel, and closed when we are in the plus half.

Our homing macro can contain the following lines for the X axis:

```
M105/X P7 F100        ; move X- until INP7 is open
M106/X P-7 F100       ; move X+ until INP7 is closed
M91/X                 ; run the normal homing sequence
```

The first line will quickly move us to the minus side of the edge, if we were on the plus side to begin with. If we are already on the minus side (if the sensor is already open) then there will be no movement. The second line will quickly move us (back) to the plus side of the edge, where the sensor is closed.

The third line will start the usual minus homing sequence: slow-jog speed minus until the switch opens; slow jog plus until the switch closes again; and slow movement further plus until the encoder index pulse comes around.
The complete sequence will end at the first encoder index pulse to the plus (sensor-closed) side of the edge.

This same general approach could be used with a dedicated home switch that is at one end of travel as well, as long as it trips prior to the limit switch.

This approach cannot be used with a home switch that is also a limit, because M105 and M106 will stop with an Error if a designated limit switch is tripped during the movement. Only M91 and M92 are allowed a one-time pass on tripping a limit switch (and then only for the limit switch they are looking for).

If you do home to a dedicated home switch, separate from the limit switch, it is highly desirable that the switch be set up with an extended ramp, so that the home switch remains tripped from its initial trip point, all the way out to the over-travel limit. Otherwise -- if it is physically possible to be inside the limit switches, on either side of the home switch, with the home switch not tripped -- then the homing macro cannot know whether to move plus or minus when seeking the home switch.

You cannot reliably use M115 and M116 in homing sequences, because those codes always attempt to apply software travel limits. They do not recognize that the software limits are not yet valid when machine home has not yet been set, and so will apply invalid restrictions to axis movement.

M105 and M106 do not apply software travel limits, even if the machine has been homed.

8.) How do I use encoder marker pulse relay output from an axis drive for accurate day to day homing? (aka ZRi homing.)

ZRi homing is a nice feature to implement on an Acorn CNC control system. (Zero Reference Input) ZRi facilitates accurate day to day homing above and beyond the accuracy achieved by home switches alone. While not that critical on a Mill or Router where resetting the WCS is quite easy and often done anyways, ZRi homing is very nice to have on a lathe as a time saving feature so that the Center line of the spindle (X0) is retained after a power cycle of the CNC control. ZRi requires a Servo drive that is equipped with this feature (this feature is often called differently with each drive mfg) such as the Estuns that Centroid sells and DMM. These drives have an output that closes when the marker pulse is detected (there is one marker pulse per revolution on the encoder). That ZRi output is connected to an Acorn input and then the CNC12 home macro uses the home switches to “get in the ballpark” and then uses the ZRi input to home the machine tool very accurately. Here are some common ZRi home program macros. ZRi inputs can be wired to individual inputs or all wired into one Acorn inputs. Please reference the Acorn CNC hookup schematics for Estun and DMM for more info on wiring. https://www.centroidcnc.com/centroid_diy/downloads/acorn_schematics/centroid_acorn_hookup_schematics.zip

Typical Lathe ZRi Example with ZRi inputs wired to same input:

```
M92 /Z L1          ;Move to Z+ home switch, back off until it clears.
M105 /Z P-4 F3     ;Move Z minus at 3 ipm until input 4 closes
M26 /Z             ;Set Z home here
G91 Z-.010 F25     ;Incremental move to clear the Z axis ZRi input
M91 /X L1          ;Move to X- home switch, back off until it clears.
M105 /X P-4 F3     ;Move X plus at 3ipm until input 3 closes
M26 /X             ;Set X home here
```

Typical Router ZRi Example (Gantry (Y) moves in minus direction) with ZRi inputs wired to individual inputs:

```
M92 /Z L1          ;Move to Z+ limit switch, back off until it clears.
M105 /Z P-4 F1     ;Move Z minus at 1ipm until input 4 closes
M26 /Z             ;Set Z home here
;
M92 /Y L1          ;Move to Y+ limit switch, back off until it clears.
M105 /Y P-3 F1     ;Move Y minus at 1ipm until input 3 closes
M26 /Y             ;Set Y home here
;
M91 /X L1          ;Move to X- limit switch, back off until it clears.
M106 /X P-2 F1     ;Move X plus at 1ipm until input 2 closes
M26 /X             ;Set X home here
```

Typical Mill ZRi Example (Y moves in plus direction table toward you) with ZRi inputs wired to one input:

```
M92 /Z L1          ;Move to Z+ limit switch, back off until it clears.
M105 /Z P-4 F1     ;Move Z minus at 1ipm until input 4 closes
M26 /Z             ;Set Z home here
G91 Z.010 F25      ;Incremental move to clear the Z axis ZRi input
M92 /Y L1          ;Move to Y+ limit switch, back off until it clears.
M105 /Y P-4 F1     ;Move Y Plus at 1ipm until input 4 closes
M26 /Y             ;Set Y home here
G91 Y.010 F25      ;Incremental move to clear the Y axis ZRi input
M91 /X L1          ;Move to X- limit switch, back off until it clears.
M106 /X P-4 F1     ;Move X plus at 1ipm until input 4 closes
M26 /X             ;Set X home here
```

9.) How can I run a macro and step through it line by line?

You can "step through" a macro by using "Single Block" mode (the Button on VCP beside cycle cancel, Mill Operators Manual Chapter 2.6, or F55 in the Run menu Chapter 3.4) and adding 2 to parameter 10. (this is a bit-wise parameter...So, if the value is currently 1, set to 3 for example, see FAQ #12 below for more on bit-wise parameters). This will make CNC12 run in single block mode with a Macro. This will let you see line by line what the macro will do and ask you to press cycle start to continue for each line in the Macro (and G code program)

Alternatively you could insert a M0 (Mill Chapter 13) that would cause CNC12 to stop and wait for Cycle Start before continuing to the next line, this is useful when debugging a macro and you only want CNC12 to stop at a single point or a few points in the macro for debug purposes.

10.) When I open cncm (or cnct) there are missing macros for some of the M codes where are they?

Certain basic M codes have their default behavior built right into CNC12. In these cases the corresponding mfuncXX.mac is not present in the cncm or cnct direction, the mfuncXX.mac file is only needed for these M codes if you wish to have that M code function differently than the CNC12 default action for that M code! For instance, the file mfunc6.mac that corresponds to M6 Tool Change is missing from the cncm directory. CNC12 has a built in Tool Change (M6) macro that is the default action for a tool change. If there is no mfunc6.mac present in the cncm directory then CNC12 uses the default tool change action, the default behavior of M6 is described in the Operators manuals. Mill Chapter 13.9.

Lathe tool change macro is name differently than the mill, A Lathe toolchange macro is "cnctch.mac"

11.) How do I write a macro for a Lathe Automatic Tool Changer Turret with Acorn?

Working examples of common Lathe Tool Changer Turrets are found on the Acorn Tech support forum.
See Zip file download at the top of this page.
https://centroidcncforum.com/viewtopic.php?f=63&t=1340#p7671

12.) What is a "bitwise" Parameter?

Most Acorn users do not have to worry about bit wise parameters as the Acorn Wizard takes care of setting the bit wise parameters needed for Acorn CNC functionality based on the CNC setup selections made by the user in the Wizard. That being said power users that are interested in customization should be familiar with the concept.

Many CNC12 system configuration parameters are know as bit wise parameters. Bit wise parameters allow the selection of several options that can be grouped together at once.

For example: Parameter 11 "Macro M function handling/Probe Stop Handling" (Mill Operator manual Chapter 15.3.12) has 4 choices related to how you would like various aspects of a macro to work. Two useful options when writing and debugging a custom macro are:

"Display M & G-codes in M function macros?" Yes =1, No = 0

and "Step through M function macros in Block Mode?" Yes = 2, No = 0

Now to the uninitiated this seems confusing, what does this mean? Do I have to pick between the two? I want both the Macro G code to be displayed and I want it to step thru the Macro in Block mode. So, how does this work?

For a bit wise parameter you add together the values of the Functions you want.

So, for instance if I just want

1.) "Display M & G-codes in M function macros?" set P11=1
2.) if I just want "Step through M function macros in Block Mode?" set P11=2

3.) if I want both “Display M & G-codes in M function macros?” AND “Step through M function macros in Block Mode?” set P11 =3! (add the values of the functions I want 1+2 = 3)

13.) How do I Round Decimal Values to Whole Numbers

It is often necessary to round fractional values to whole numbers (integers). For example, if you need to bring a rotary table, which may be at any angle, to the nearest whole turn (whole multiple of 360°), then you would want to round its position (converted to turns) to the nearest integer, then convert back to degrees and send the axis to that position.

Beginning with CNC12 version 4.14, you can use the functions “FIX” and “FUP” to round down or up to the next integer value. Note that for negative values, FIX still rounds down (thus away from zero) and FUP still rounds up (towards zero).

With CNC12 v4.14 or newer, then, you can round to the nearest integer by first adding 0.5, then rounding down:

```
#100 = FIX[#5044/360.0 + 0.5] * 360.0
G90 G0 A#100
```

14.) How do I Control an Output when the Probe Tool number is selected.

In some cases it may be necessary to turn on an output when the probe tool is selected. Below are two examples of how this can be done, Assuming tool 99 is the probe tool number and OUT7 is defined as Output 7 in the Acorn Wizard.

To accomplish this, at the end of our m6 macro we could have a line that states:

Method 1:

```
IF [#4120 == 99] THEN M67 ELSE M87          ;If tool selected is tool 99 then activate output 7, otherwise deactivate
                                            ;This will activate Output 7 if the tool requested was 99.
```

Alternatively, Parameter 12 could be used as it is the touch probe tool number so, we could instead write.

Method 2:

```
IF [#4120 == #9012] THEN M67 ELSE M87  ;IF tool selected Is equal to touch probe tool number, then activate output
                                       ;7, otherwise deactivate output 7
```

Note: The probe tool number should be set up correctly in the Probe menu of the Acorn Wizard. The value in the Wizard will be written to parameter 12 when pressing “Write Settings to CNC Control Configuration”. If the operator decided that tool 10 should be the probe instead of 99, he could then simply change the value in the Wizard. The first method would require the operator to modify the macro and may lead to damaging the probe if not done correctly. It is generally better to use existing parameters or variables when available for macros instead of “hard-coding” values into the macro.

15.) Can I use “math” in a macro?

Yes, Please see Mill Operator manual Chapter 11.2.14 for complete list of allowed mathematical expressions

Examples:

```
G91 X [13/64] Z [1+3/8]     ; move the X axis 13/64 (0.2031) units and the Z axis 1 3/8 (1.375) units incrementally

X[ SQRT [ ABS [ SIN [#101] - COS [#102]]]] ; Move X as a function of #101 and #102
```

16.) What is System Variable #4203

Currently, Acorn software does not use system variable #4203 for “Tool in spindle” and will result in an “Undefined variable” error in software. For CNC12 for use with Allin1DC and Oak CNC controllers, #4203 is used in five axis machines, primarily for special tool changes involving “Large tools”. System Variable #4203 is set differently depending on how parameters 6 and 160 are set.
If Parameter 6 and 160 are set to a zero, then #4203 is updated when a T command is called, Essentially,

#4203 == #4120
If Parameter 6 is a non-zero value and 160 is set to a zero, then #4203 is updated from a value from the PLC. For Example a line in the PLC may say the following,

SV_PLC_CAROUSEL_POSITION = CarouselPosition_W

SV_PLC_CAROUSEL_POSITION is the tool number displayed at the top right in the software and is the value of #4203, it will not change until the value in the PLC changes. CarouselPosition_W is a variable used in the PLC, it is assigned values which then gives its value to the SV_PLC_CAROUSEL_POSITION variable.

If Parameter 6 and 160 is set to a non-zero, then #4203 is updated after the M6 tool change macro is called and finished. This allows for special operations in the macro for 5-axis machines primarily, where #4120 can be used for the requested tool and #4203 is the previous tool. So,

#4120 = T Number requested
#4203 = Tool in Spindle (Last T number called)
M6 is executed and Finished
#4203 = Tool number requested making it new tool in spindle

17.) What are the differences between M98, G65 with Local Variable Stack

M98 and G65 can both be used to call a CNC subprogram.

The most significant difference between the two is that G65 allows arguments (parameters or variables) to be passed in as part of the subprogram call. The arguments provided with a G65 call are accessible within the subprogram as CNC variables #1 through #33.

For example, any value given with the letter 'A' in a G65 call will appear in variable #1 within the subprogram.

Any CNC program or subprogram can use variables #1 through #33 for their own purposes, whether or not they were given values through a G65 call.

However, therein lies a key difference between M98 and G65: a G65 call advances the "stack", so that the #1 - #33 variables in the called subprogram are a fresh set, independent of the #1 - #33 variables in the calling program.
An M98 call, in contrast, does not advance the stack. Variables #1 - #33 in the M98-called subprogram are the same ones used by the calling program.

Consider this example:

```
; define a macro subprogram, for later use
O9001
  #3 = #1 + #2
  G0 Y#3
M99

; main program starts here
#1 = 10.0
#2 = 5.0
#3 = 1.234
G0 X#3            ; moves X to X1.234
G65 P9001 A3.2 B0.6  ; macro moves Y to Y3.8
G0 Z#3            ; moves Z to Z1.234
```

Because the variables #1, #2 and #3 in the macro subprogram are independent of the ones in the main program, the main program's values are unchanged after the G65 call.
Compare that with the following, identical except that G65 has been replaced with M98:

```
; define a macro subprogram, for later use
O9001
  #3 = #1 + #2
  G0 Y#3
M99

; main program starts here
#1 = 10.0
#2 = 5.0
#3 = 1.234
G0 X#3            ; moves X to X1.234
M98 P9001 A3.2 B0.6  ; A and B do nothing, macro moves Y to Y15.0
G0 Z#3            ; moves Z to Z15.0
```

Using M98 instead, the A and B arguments are non-sensical and are ignored. The macro reads and writes the same local variables as the main program. It therefore calculates #3 = 10.0 + 5.0 = 15.0, and moves Y there. When we get back to the main program, local variable #3 has been changed, so the Z move is affected as well.
As a rule, if you want to use #1 - #33 as local or temporary variables, you should call your macro subprograms with G65.

If you have previously worked with older controls, you might have avoided using G65 when you did not need to pass arguments, because the number of nested G65 calls was limited. For example, on early Fanuc controls, you could nest subprogram (M98 and G65) calls up to 20 levels deep, but only four of those levels could use G65: there was only enough memory for four sets of local variables.

CNC12 allows G65 calls up to 20 levels deep, each with its own set of local variables. Computer memory has come a long way.

18.) What is the PLC detective?

While not directly involved with a macro the PLC Detective is a tool that a macro programmer should at least be aware of and know how to use at least in the basic sense as it allows you to see in real time what is going on inside the PLC program while the machine is running which can be very useful.

From the manual, *“The PLC detective is a free built-in CNC utility software PLC Debug tool that is built into CNC12. The PLC detective makes writing and troubleshooting PLC programs faster and easier. The PLC detective is based on Centroid's already powerful PLC programming source language and enable system integrators to tackle larger and more complex retrofits.*

Every system Integrator who has ever tried to create or modify a PLC program for a new or modified machine knows the frustration of having run some new PLC program, wondering "Whats it doing now?". Now Centroid's "PLC Detective" allows you to clearly see the true/false state of each instruction while the machine is running, highlighted in color!

Everyone who has ever tried to debug a long term intermittent problem knows the frustration of trying to figure out what led up to a fault condition. Now Centroid's " PLC Detective" solves this problem once and for all. Set your trap , and wait , when the error occurs, go look back in time at all the I/O's leading up to the problem. This is "game changing" PLC debug tool for Integrators and Power Users alike.”

To start PLC detective from the main screen of CNC12, press <Ctrl E>, this will bring up PLC detective, allowing you to see what is active and not active in the PLC program and some other useful information regarding the PLC.

See the PLC detective manual for more information.
https://www.centroidcnc.com/downloads/centroid_PLC_detective_quickstart.pdf

19.) How do I edit the PLC program to add Custom PLC Output :
Adding a custom output definition to the Acorn PLC program.
To enable the use of M94 and M95 for a custom output, (not necessary if you are using any one of the stock Wizard outputs) edit the PLC with both the desired Custom Output, and desired logic. For this example we will create an output called “CustomOut” and add it to the Acorn PLC program. The Acorn PLC source file can be found in the cncm directory “acorn_mill_plc.src” or in your cnct directory for lathe “acorn_lathe_plc.src”. Here is the PLC program Example:

```
;-------------------------------------------------------------------------
;                    Output DEFINITIONS
;             Closed = 1 (green)  Open = 0 (red)
;-------------------------------------------------------------------------

CustomOut                    IS OUT2            ;Defines our Custom Output to Output 2

;-------------------------------------------------------------------------
; M functions - The System Variables in this section inform the
; PLC that an M function has been requested.
;-------------------------------------------------------------------------

CustomM94M95  IS SV_M94_M95_18        ;Defines CustomM94M95 to the value of 18 for use in the macro.

;=========================================================================
                    MainStage
;=========================================================================

IF CustomM94M95 THEN (CustomOut)                  ;When CustomM94M95 is ON, Turns on CustomOut.
                                                  ;When CustomM94M95 is Off, Turns Off CustomOut
;Parentheses tells logic that it should be on when expression is true and turn off when it is not.

;Alternatively the logic can be

IF CustomM94M95 THEN SET CustomOut   ;When CustomM94M95 is On, Turns on CustomOut
IF !CustomM94M95 THEN RST CustomOut  ;When CustomM94M95 is Off, Turns Off CustomOut
```

;The ! sign in front of CustomM94M95 is a “not”, so when CustomM94M95 is Off then the expression is true.
;SET is to set or other words turn on, and RST is to reset or in other words turn off.

Once the Output, the M function system variable and logic is defined, you will need to compile the edited source code. Refer to the CNC12 PLC Programming Manual Page 5. Once the plc is compiled we can edit the macro we wish to use this output. To turn on the “CustomM94M95” variable in the plc we will need to use an M94 command, Example Below.

```
M94 /18                    ;Turn On CustomOut Request, Notice the value 18 the same as the value in PLC.
```

This will turn on in the plc “CustomM94M95”, which in turn the plc will look at our written logic and in this case turn on the CustomOut Output. Since we do not have any logic in the PLC that resets the system variable, we will need to use an M95 command to turn it off, example,

```
M95 /18                    ;Turn Off CustomOut Request, Notice the value 18 the same as the value in PLC.
```

This will turn off in the plc “CustomM94M95”, which will turn off the output “CustomOut”
This is a very basic example, with the introduction of the PLC, additional logic can be added to give additional functionality, such as timeout timers, Fault messages, and more when using this output.

More Information for PLC Programming can be found in the Centroid CNC12 PLC Programming Manual and Centroid PLC Programming Videos here is a link to the Playlist Found on Centroid CNC Technical Support Youtube Channel.
https://www.youtube.com/user/CentroidSupport/playlists

https://www.youtube.com/playlist?list=PLXhs2C5No0_gFS_RmKNo7hii2WKIedQlQ

Acorn CNC12 Machine Configuration parameters:

Acorn specific CNC12 parameters have been created to mostly facilitate Acorn Wizard functionality and introduce new features and options for Acorn above and beyond the “stock” CNC12 parameters. Below is a list of Acorn CNC12 specific parameters, most are not covered in the v4.14 CNC12 Mill and Lathe operator manuals. Keep in mind this is not an exclusive list, there are many “stock” Mill and Lathe CNC12 parameters that are documented in the v4.14 operators manual(s) and also apply to Acorn! So, be sure to check them out see Mill Operator Manual Chapter 15.3 and Lathe Operator Manual Chapter 12.3 for the complete list of stock CNC12 parameters. Keep in mind most Acorn users will not be changing the values of the parameters below manually unless they are configuring an ATC as the v4.20 Wizard will automatically set most of these parameters based on the selections made in the Acorn Wizard.

CNC12 Parameter Number	**Parameter Notes**
P150	Run Time Graphics (RTG) ON, with CNC12 start up. Set by Acorn Wizard, also has other funtionality choices not set by Acorn Wizard.
P153	Disable/Enable Probe Protection when using VCP jogging or keyboard jogging. Default is ON =1
P160	Enhanced ATC Default 0: 0 = Off, 1 = Nonrandom, 2 = Random
P161	Max # of Tool Bins. Default 0. See ATC documentation for more information. https://centroidcncforum.com/viewtopic.php?f=63&t=1340
P218	USB MPG. if non-zero value, start UsbMpg Pipe client with CNC12
P219	VCP options. if non-zero value, start VCP with CNC12
P302	Used by Acorn to check the last state of RTG. So, for example if P150 is set to RTG ON, but user turns off RTG in the RUN menu of CNC12 P320 keeps track of that so next time CNC12 starts it returns to the user selected state.
P400	Allow cycle start in run menu. 0= disabled (disallow), 1= enabled (allow)
P401	Forget last loaded job on cnc12 startup. 0= disabled (remember last g code program ran), 1= enabled (forget last g code program ran), set by Acorn Wizard
P403	Disable keyboard jogging legend. 0= Display legend, 1= hide legend
P404	TT-1 probe automeasure memory bit used by CNC12 to allow both the TT and probe to be used at the same time.
P405	Tool Touch Off Type. 0=Mechanical, 1= Conductive, 2= TT-1, 3= TT-2. Set by Acorn Wizard
P406	Probe State When Tripped. 0 = Closed, 1 = Open, Set by Acorn Wizard
P407	Tool Touch Off State When Tripped. 0 = Closed, 1 = Open, Set by Acorn Wizard
P409	Probe Type. 0=mechanical, 1= conductive, 2= DP4, Set by Acorn Wizard
P410	Display Probing reminder message to plug in the probe and make sure it is working before each probing cycle. 0= disabled, 1= enabled.
P411	Wireless MPG model type selection. 0= CWP-4, 1= WMPG-4
P412	Spindle Back Gear Mode, 0= Disabled, 1= Enabled
P413	Park Button Override (Uses park.mac) 0= disabled, 1= Enabled. Setup by Wizard.
P414	Home File Type. 0= Default, 1= Custom, 2= Paired Axes Auto Squaring, 3= Custom Paired Axes Auto Squaring.
P415	Ether1616 Configuration Bits, Set by Acorn Wizard. Default 0: +1 A0, +2 A1, +4 A2, +8 A3, +16 A4, Set by Acorn Wizard
P500	Acorn Axis Pairing Mode, Set by Acorn Wizard. Set by Acorn Wizard
P501	Acorn Axes Squaring Auto or Manual. Set by Acorn Wizard
P502	Acorn Axes Squaring Distance, Set by Acorn Wizard
P503	Acorn Axes Squaring, Set by Acorn Wizard
P504	Acorn Axes Squaring Deadband, Set by Acorn Wizard
P505	Acorn Axes Squaring Salve Input, Set by Acorn Wizard
P506	Acorn Axes Squaring Master Input, Set by Acorn Wizard
P507	Acorn Axes Squaring Slave Axis, Set by Acorn Wizard
P508	Acorn Axes Squaring Master Axis, Set by Acorn Wizard
P800	Acorn DB25 Map Enable, Set by Acorn Wizard

P801-804	Acorn Step 1-4. DB25 Pin Mapping, Set by Acorn Wizard
P805-808	Acorn Dir 1-4. DB25 Pin Mapping, Set by Acorn Wizard
P809-812	Acorn PLC 1-4DB25 Pin Mapping, Set by Acorn Wizard
P813	Acorn DB25 Header Selection, Set by Acorn Wizard
P820	Machine Type ***Currently Not Used***Mill = 0, Lathe = 1, Router = 2, Plasma = 3, Waterjet = 4, (future) etc…
P821	Slaved Home Pair, Set by Acorn Wizard
P822	Home All Input, Set by Acorn Wizard. Limit used for HomeAll Function.
P823	ZRI Input ***Currently Not Used*** Defines the limit used for all ZRI signals.
P824	Limit All Input set by Acorn Wizard. Limit used for all limit switches wired to same input.
P826-829:	Tool Turret Position Bit 1-4, refer to Tool Turret setup documentation. refer to Tool Turret setup documentation here. https://centroidcncforum.com/viewtopic.php?f=63&t=1340
P830	: ATC Type ***Currently not used, will be needed when ATC types are combined into 1 unified PLC***. 0 = none. 1 = incremental, 2 = incremental w/sync, 4 = absolute 3bit, 8 = absolute 4bit, 16 = switched reverse, 32 = self reversing
P831-846	Tool Turret Position 1-16, refer to Tool Turret setup documentation. https://centroidcncforum.com/viewtopic.php?f=63&t=1340
P848	Turret Reverse Timer, Default = 1500, refer to Tool Turret setup documentation.
P849	Tool Change Time Out Timer Default =10, refer to Tool Turret setup documentation.
P850	Time Delay to Start Counting, refer to Tool Turret setup documentation.
P851	ATC Time Before Reversing, refer to Tool Turret setup documentation.
P855	MPG Performance Mode. Set by Acorn Wizard in MPG menu
P856	Jogging Option at start up. Set by Acorn Wizard, in CNC control preference menu. Bit wise. Choose Jogging preference on start up in Acorn Wizard.
P857	: Z Following Options ***Currently Not Used***
P941-946	Limit Inputs 1-6. used by PLC program to determine which inputs to invert when using the VCP Limit Defeat Button.
P960	Divider for Charge Pump Freq. (hz) ex. 1,2000000/ param960= charge pump freq. Example: If param 960 = 96 charge pump freq is 12,500 hz
P962	USB Jogging Control. 0= PC Keyboard Jogging with Legend, =1 Disable PC keyboard Legend and use X keys or other USB jogging device, 2= use Contour Shuttle
P961	Set by Acorn Wizard, Step direction and enable inversion. Bit wise parameter to set by Wizard in Advanced tab to invert Step, Direction and Enable.
P964-967	Used by Acorn Paired and Axes Autosquaring. Set by Acorn Wizard
P968	: Acorn Step and Direction Frequency in (Hz). Allowed Values: 0 = 200000 Hz, 1 = 1200000 Hz, 2 = 600000 Hz, 3 = 400000 Hz, 4 = 300000 Hz 5 = 240000 Hz, 6 = 200000 Hz, 12 = 100000 Hz, ; 30 = 40000 Hz, 60 = 20000 Hz, 100 = 12000 Hz
P975	Time Between Tool Positions. This is for the Time-based-Lathe turrets. The time it takes to go from tool to tool. only used for basic Time Based Tool Changer Turrets.
P985	Door Safety Interlock Mode selection. 0=Disabled, 1= Slowjog while door open, =2 system stop while door open
P990	Spindle Brake Timer (ms). Default is 250ms, Spindle Brake delay in milliseconds us used to delay the spindle brake to avoid vfd faults. Acceptable parameter value range is 1-3000, A value of 0 defaults the delay time to 250 ms, typical values are 150 to 500 ms.
P991	Axis motor Drive Fault Timer (ms)Set by Acorn Wizard
P992	Chuck Timer (ms). Used to Turn off Outputs if no inputs selected or used as a fault out timer
P994	Partchute Timer (ms) / Spindle Lock Timer (ms) ***Not Implemented Fully***. Used to turn off outputs if no inputs selected or used as a fault out timer
P995	Cut Off Timer (ms) . Used to Turn off Outputs if no inputs selected or used as a fault out timer