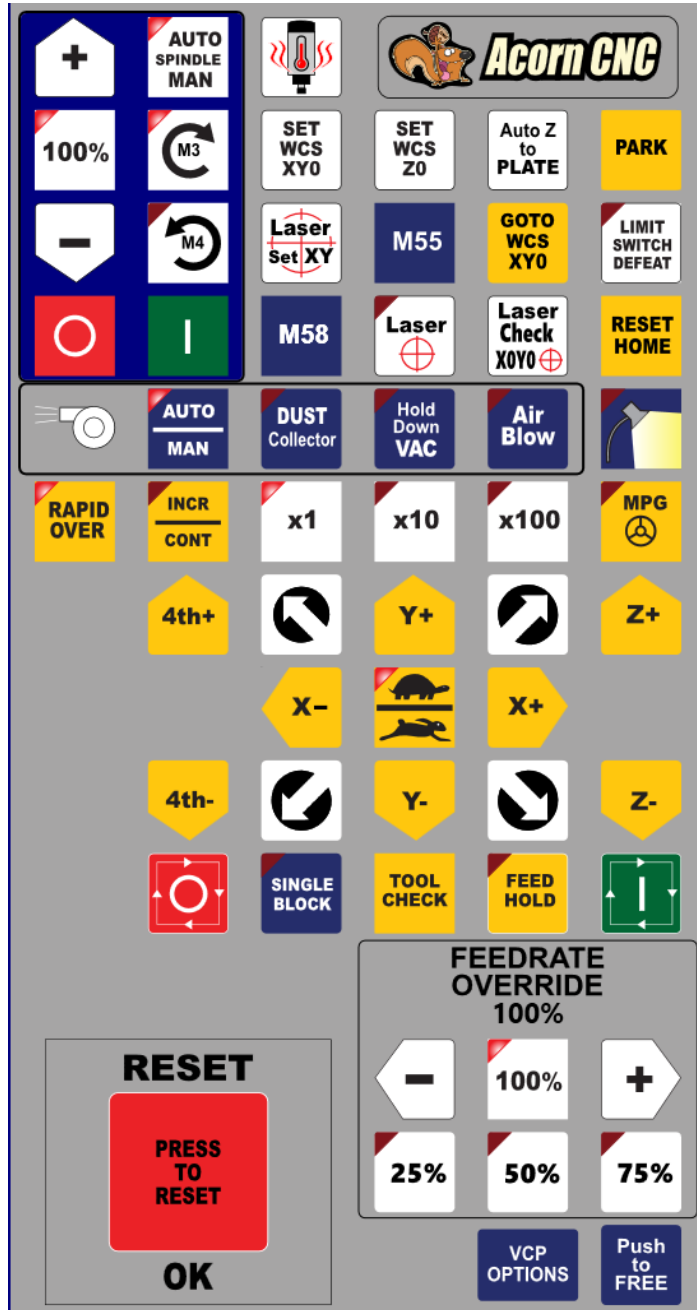




Virtual Control Panel (VCP) 2.0

CNC Software version: CNC12 V.5.08 +

Models: Acorn CNC, Acorn Plasma, AcornSix, Hickory, Oak, Allin1DC, MPU11



Introduction **2**

Button Grid Layout **6**

VCP user editable files **7**

How to move or delete a button **8**

Button Graphics Location and Format **11**

Change Button Graphics and VCP background color **13**

Change the LED indicator light color **16**

Swap Image when button is clicked/pressed **17**

Swap Image when button function is activated **18**

Create a New Button **20**

Logos and Icons **27**

Border and Backgrounds **31**

Mouse Hover/Click and Touch effects **32**

VCP Auxiliary Keys and Macro Assignments **33**

Making BIG buttons **41**

Display Data with PLC Words **43**

Trouble Shooting **53**

Special Cases **59**

Resources **62**

VCP Introduction

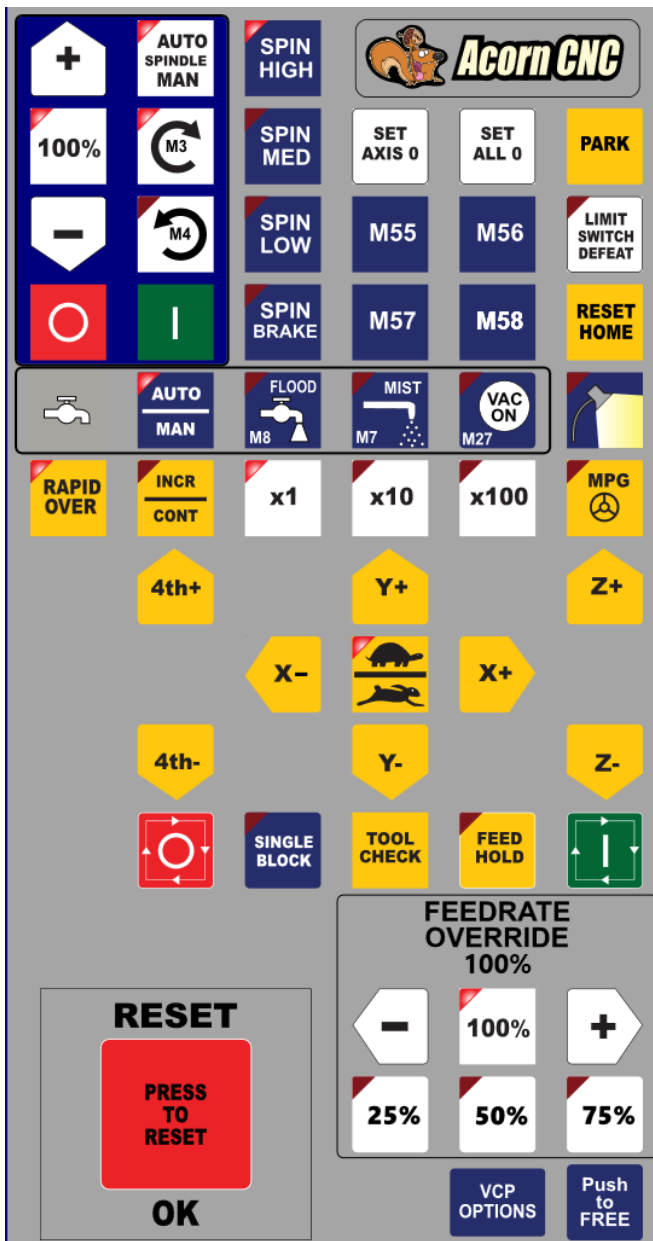
The Centroid Virtual Control Panel (VCP) allows the CNC user to employ a mouse and/or a finger (touch screen monitor) as a Human Interface to the CNC controller. The VCP has a wide variety of features that allow the CNC controller operator interface to be customized to a wide variety of applications in a straight forward manner. Users, Rebuilders and OEM alike can change the look, feel and function of the Centroid CNC controller Virtual Control Panel.

The VCP is defined by a 'skin'. A skin is just a text file that defines what buttons, graphics, borders and colors that make up the VCP. Centroid provides 'stock' VCP skins which can be used as-is or modified.

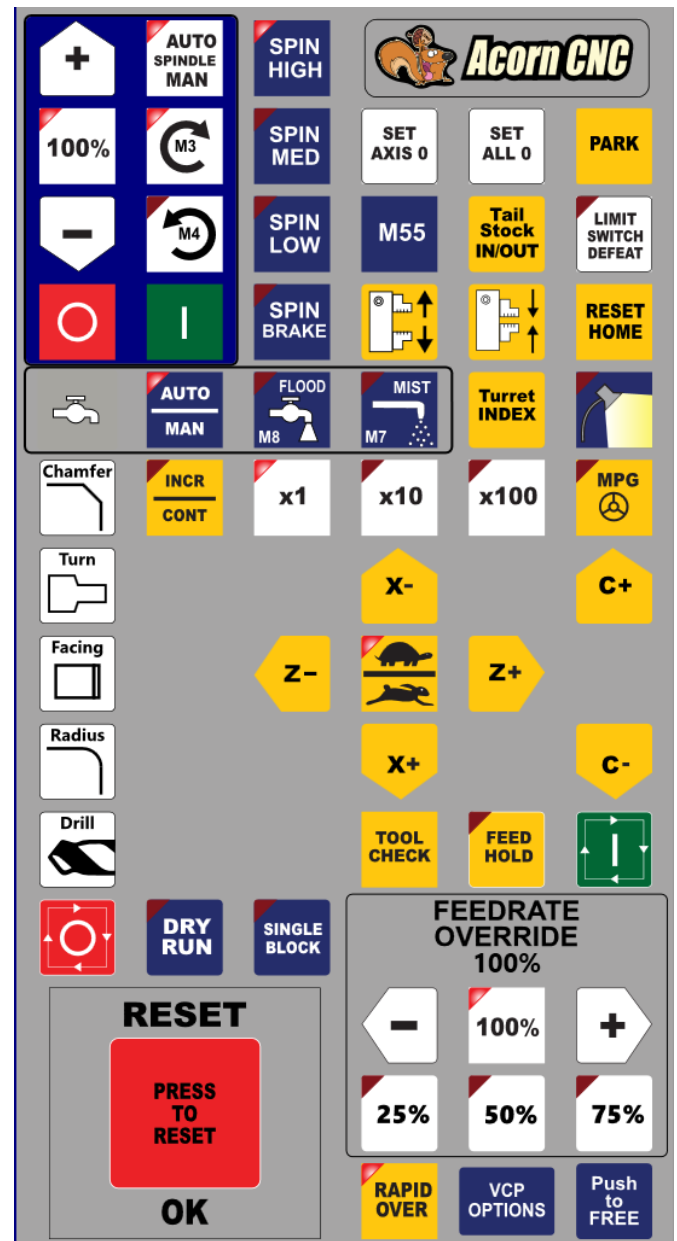
CNC12 will automatically install a stock skin as a default VCP skin so all CNC12 installations will have a running working VCP immediately.

CNC12 Mill, Lathe, Router, Plasma each have its own unique stock VCP skins that are pre tailored to common uses and functions of those respective machine tools. Users can simply use the default VCP "as is" and if so, then most everything in this document will not apply to them. If you wish to modify the default VCP buttons then read on, this VCP Users Manual will show you how to edit the VCP skin and buttons to customize the VCP to meet the machine requirements or to simply suit the users taste.

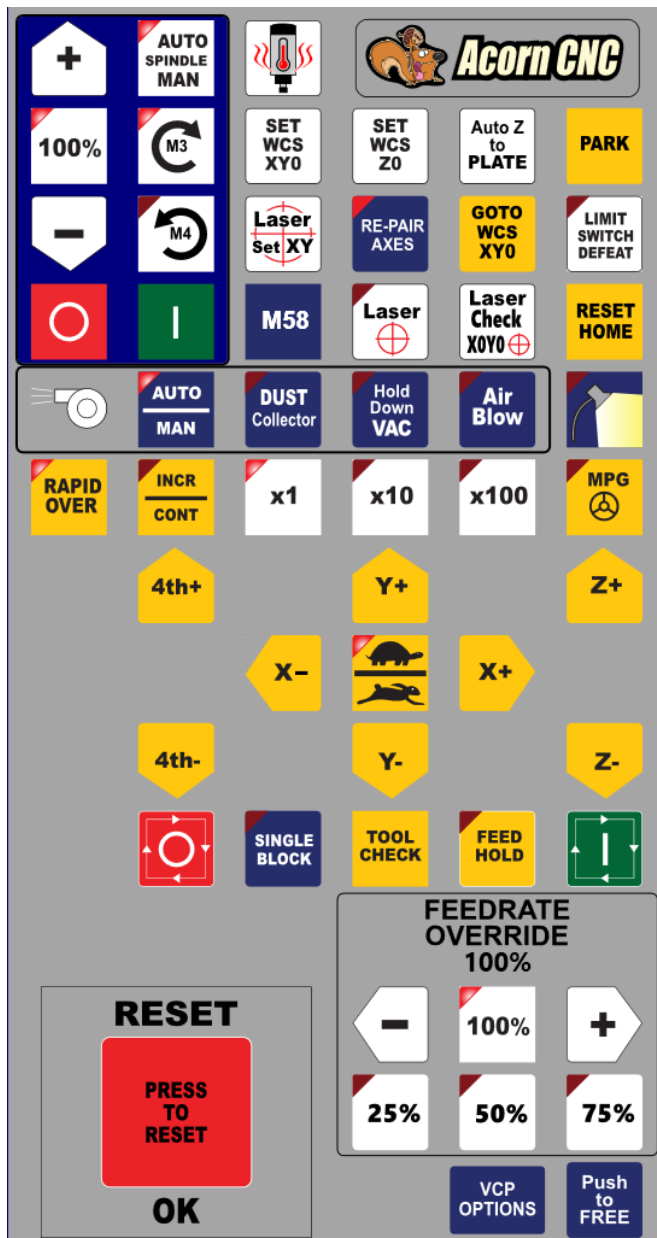
Acorn Default Mill VCP skin



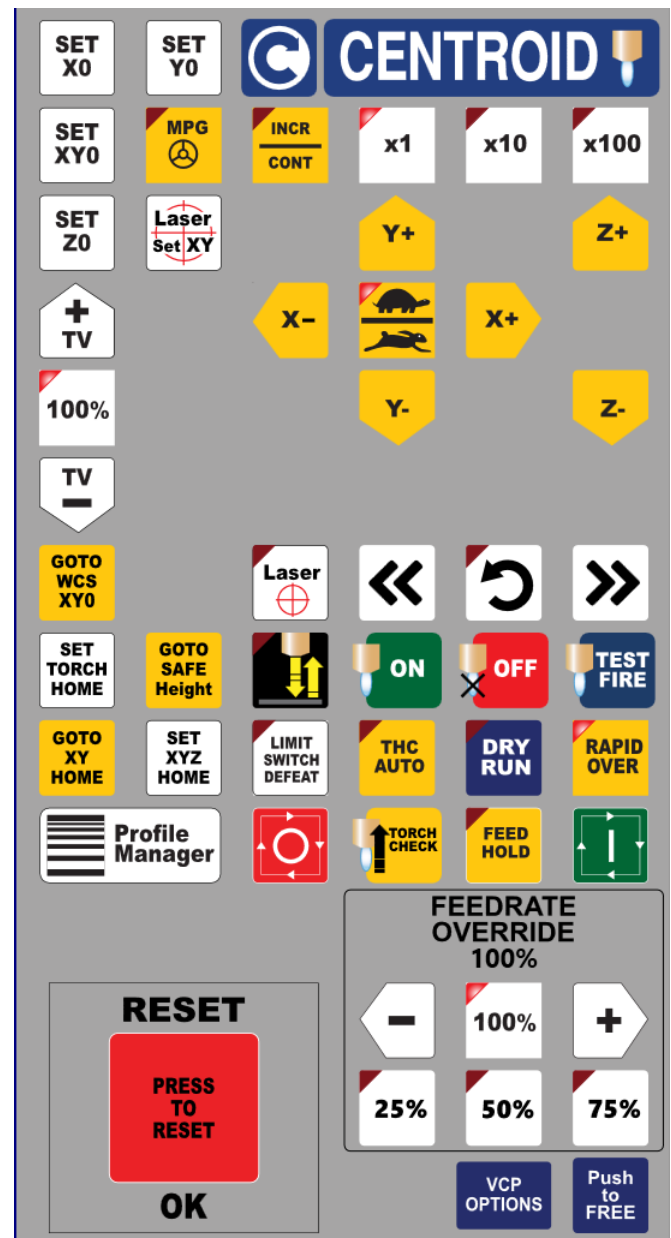
Acorn Default Lathe Front Mount Tooling VCP skin



Acorn Default Router VCP skin



Acorn Default Plasma VCP skin



The VCP is available to all current Centroid CNC controller hardware platforms, as the time of this writing they are: Acorn, AcornSix, Acorn Plasma, Oak, Allin1DC, Hickory and MPU11 based CNC control systems.

It is interesting to note that the VCP can be used in conjunction with the Centroid “hard” operator control panel commonly used on Oak, Allin1DC, AcornSix, Hickory and MPU11 systems and It can also be used to ‘replace’ the hard operators control panel in these system as well. So, for Oak, Allin1DC, AcornSix, Hickory and MPU11 the VCP can be used independently as a replacement of the hard operators panel, or it can be used in conjunction with the hard operators panel or the VCP doesn’t have to be used at all (hard panel only).

For Acorn systems the VCP is the primary operator control interface but, it can also work in conjunction with common DIY hard wired operator control buttons such as Cycle Start, Feed Hold, Tool Check, Cycle Cancel and any other hard wired button. The VCP is also compatible with the upcoming line of Centroid USB Operator Control Panels that have digital Rapid Override, Feedrate Override, and Spindle Speed Override knobs.

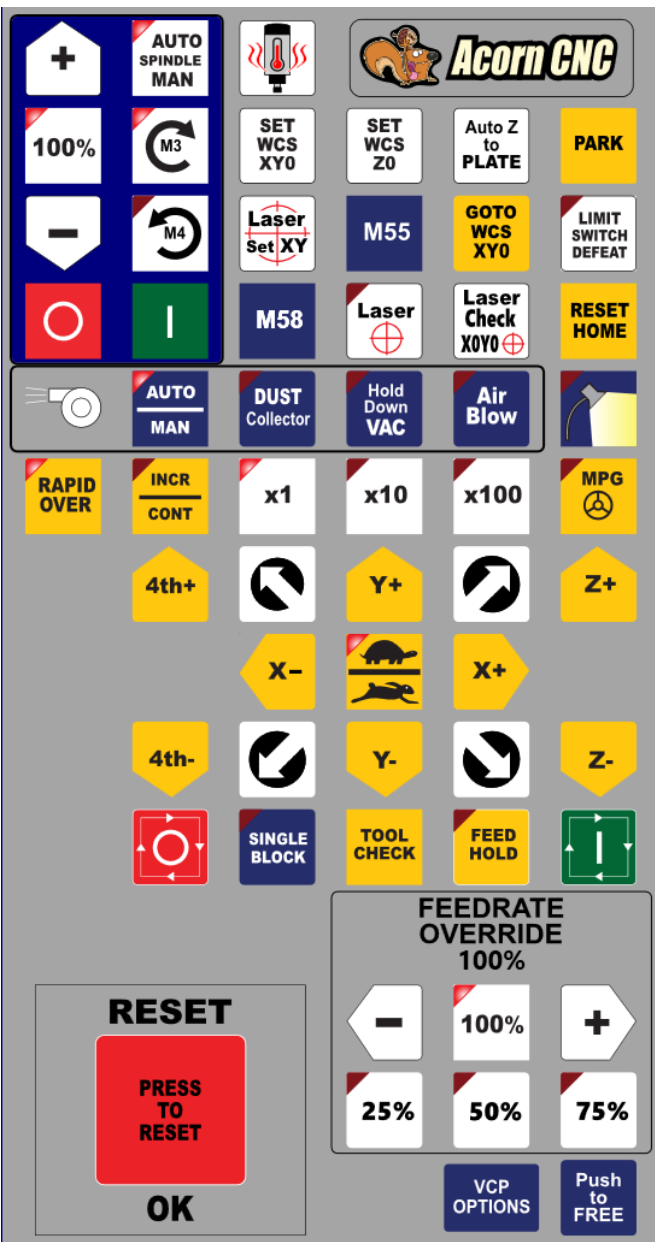
The VCP is also compatible with the [Centroid Wireless MPG](#) and can be used in conjunction with all the various operator controls found on the Wireless MPG control pendant.

There are several ‘stock’ (stock = Centroid provided VCP Skins) VCP skins with any one CNC installation. These skins are very similar but, with a few differences. For instance there are stock VCP skins with Diagonal Jog Key functionality.

Example of one of the “stock” Plasma VCP skins with diagonal jog keys



Example of one of the “stock” Router VCP skin with diagonal jog keys



With Acorn and AcornSix choosing which stock VCP skin to use (or start with) can be selected with the drop down menu in the Wizard VCP Preferences menu. The image below shows the Centroid stock skins that are available in the Acorn Lathe Wizard.

VCP (Virtual Control Panel) Preferences

Centroid "Stock" Virtual Operator Control Skins
Choose a configuration

acorn_lathe_vcp_diagonal_skin.vcp

acorn_lathe_vcp_diagonal_rapid_skin.vcp

acorn_lathe_vcp_diagonal_skin.vcp

acorn_lathe_vcp_legacy_skin.vcp

acorn_lathe_vcp_rapid_skin.vcp

acorn_lathe_vcp_skin.vcp

VCP jogging state on Acorn power up

Custom VCP skin in use

No

Enable Rapid Override

No

Stock VCP skins can be used as-is or edited to suit the users taste and the machines functionality requirements. This manual details the Centroid VCP features that can be used by anyone to create a custom VCP skin for thier particular application.

The VCP buttons are laid out in a even repeating grid pattern which is used to identify the button location.

This grid is used to identify the location of the buttons on the VCP layout skin. For instance, the Cycle Start button is located on Row 10 and Column 6, the X positive jog key is located on Row 8 Column 5, the Park button is Row 2 Column 6, etc... All the locations on the grid are completely open for use with any button and function.

	Column 1	Column 2	Column 3	Column 4	Column 5	Column 6
Row 1						
Row 2						
Row 3						
Row 4						
Row 5						
Row 6						
Row 7						
Row 8						
Row 9						
Row 10						
Row 11						
Row 12						
Row 13						
Row 14						

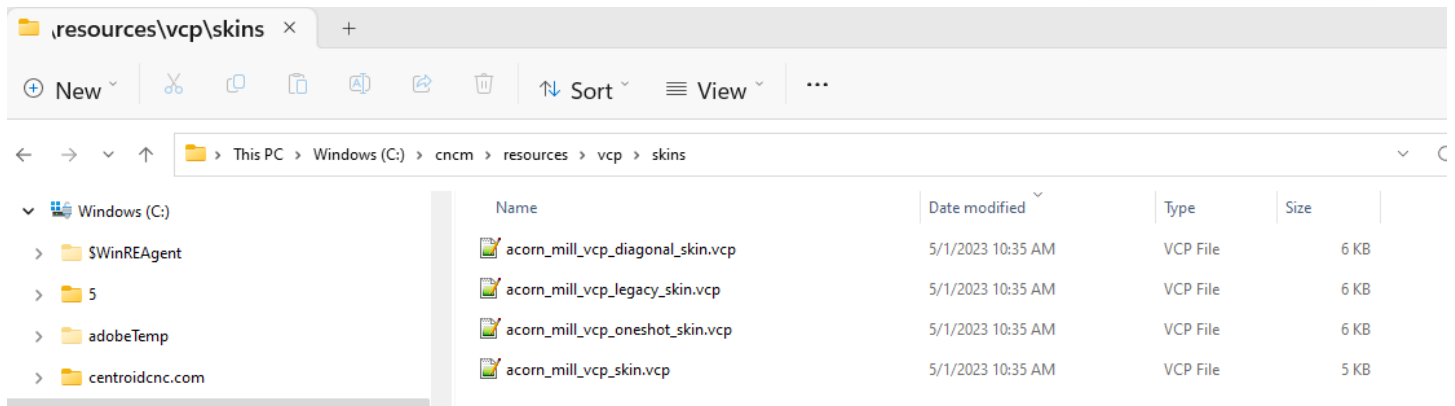
VCP user editable files

The Virtual Control Panel (VCP) buttons can be user edited and modified to display new graphics or perform a different functions. Customization of the VCP is done mainly through a set of .XML files, macros and .SVG image files for the graphics.

The VCP reads several Extensible Markup Language (XML) files that define the VCP layout, look, feel and function. These XML files are both machine readable, user editable and simple to understand. (There is also a wealth of information on XML file formatting and rules on the Internet.)

The Acorn Mill Centroid Default VCP skin seen on page 2 is primarily defined by two XML files:

- 1.) The Acorn VCP skin XML file used located in the c:\cncm\resources\vcp\skins folder and in this example is called: acorn_mill_vcp_skin.VCP
- 2.) The Individual Button XML files located in c:\cncm\resources\vcp\Buttons\name of button folder
The button XML files are typically named the same as the button itself. For example, the Cycle Start button is located in the folder: c:\cncm\resources\vcp\Buttons\cycle_start and is called cycle_start.XML



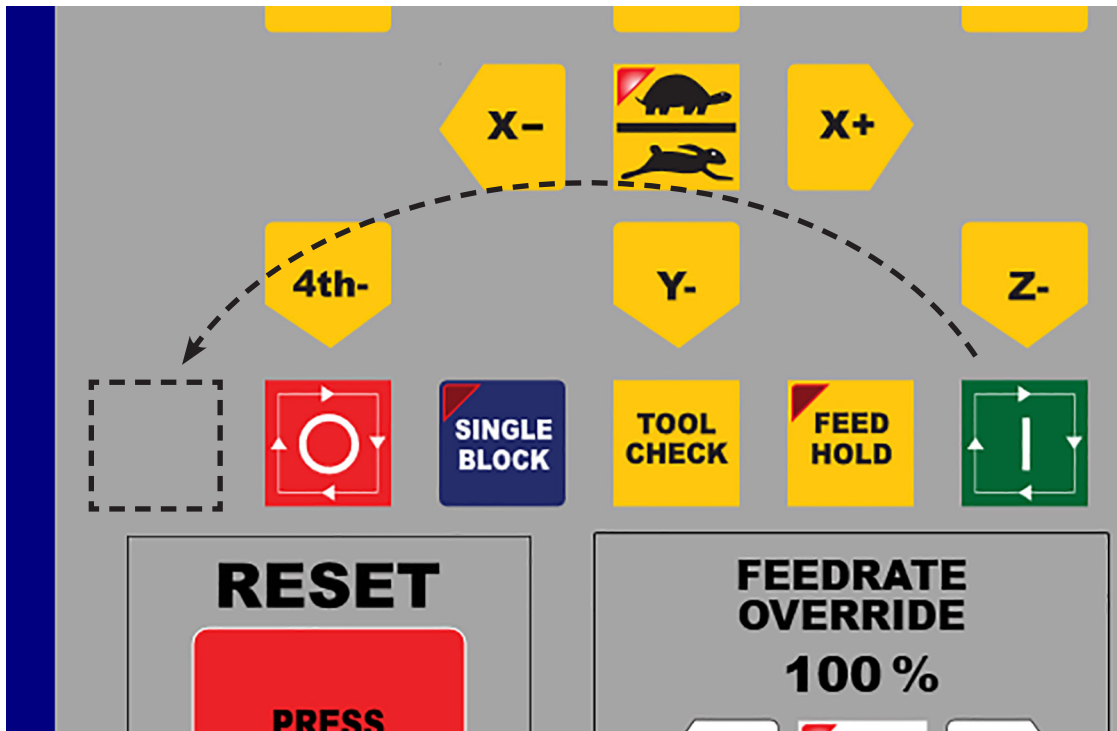
You can have any number of VCP Skins.

On startup the VCP is commanded to use a particular skin with the options.xml file located in the c:\cncm\resources\vcp\ folder. To change to a skin with a different name, simply edit the options.xml file with the name of the skin to be used.

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <ArrayOfVcpOption xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
3   <VcpOption>
4     <Name>Skin</Name>
5     <Value>acorn_mill_vcp_skin</Value>
6   </VcpOption>
7   <VcpOption>
8     <Name>KeepOnTop</Name>
9     <Value>False</Value>
10  </VcpOption>
11  <VcpOption>
12    <Name>DisableSnap</Name>
13    <Value>False</Value>
14  </VcpOption>
15  <VcpOption>
16    <Name>AdjustTransparency</Name>
17    <Value>False</Value>
18  </VcpOption>
19  <VcpOption>
20    <Name>Transparency</Name>
21    <Value>100</Value>
22  </VcpOption>
23  <VcpOption>
24    <Name>AutoSize</Name>
```

Make your first VCP customization: How to move a button

One of the simplest VCP customizations is to move a button to a different location on the VCP. So lets move the Cycle Start button from its default location to the empty space just to the left of the Cycle Cancel button.



Make a copy of acorn_mill_vcp_skin.VCP , something like.... acorn_mill_vcp_skin_backup.VCP

Open acorn_mill_vcp_skin.VCP using Notepad++ (<https://notepad-plus-plus.org/>)

Scroll down to the line with the Cycle Start button on it. (It is the next to last line in the Acorn Mill default skin)

Here are the six lines of the file. The highlighted line is the line that defines the cycle start button location.

Row 10, Column 6.

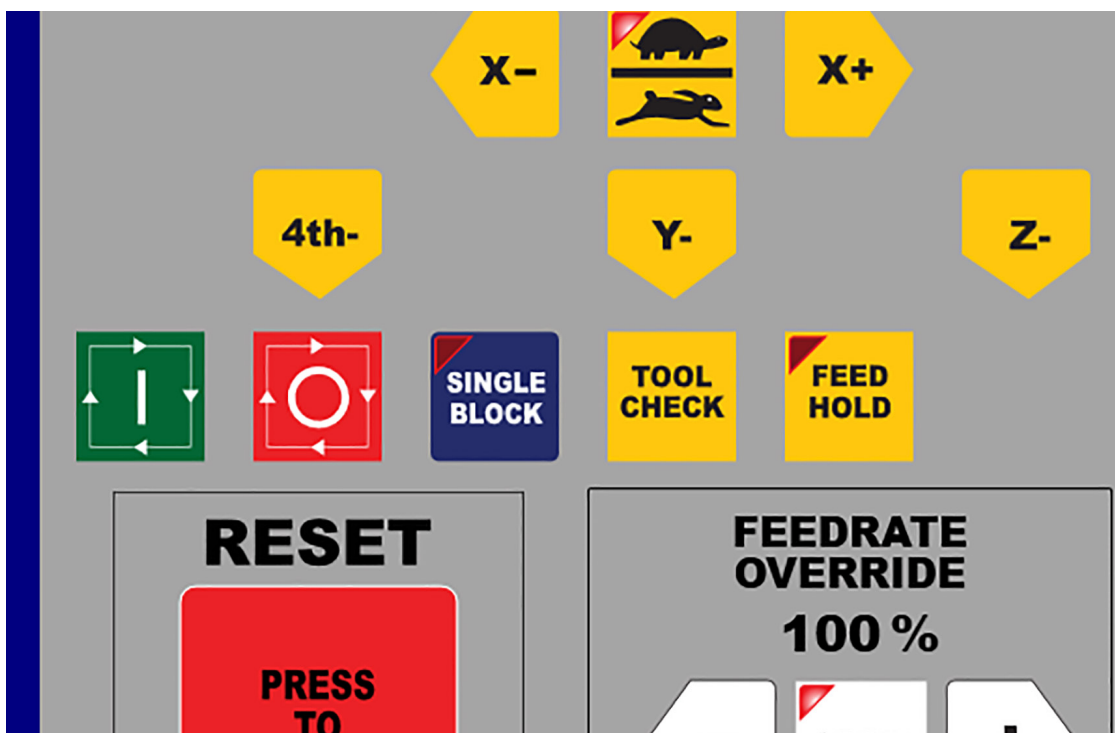
```
<button row="9" column="6">z_negative</button>
  <button row="10" column="2">cycle_cancel</button>
  <button row="10" column="3">single_block</button>
  <button row="10" column="4">tool_check</button>
  <button row="10" column="5">feed_hold</button>
  <button row="10" column="6">cycle_start</button>
</vcp_skin>
```

Now looking at the grid layout on page 3 determine the desired new button position, in this example the new button location is the space to the left of Cycle Cancel: Row 10, Column 1. Now edit button row line for cycle_start to match the new position change the column value from 6 to 1.

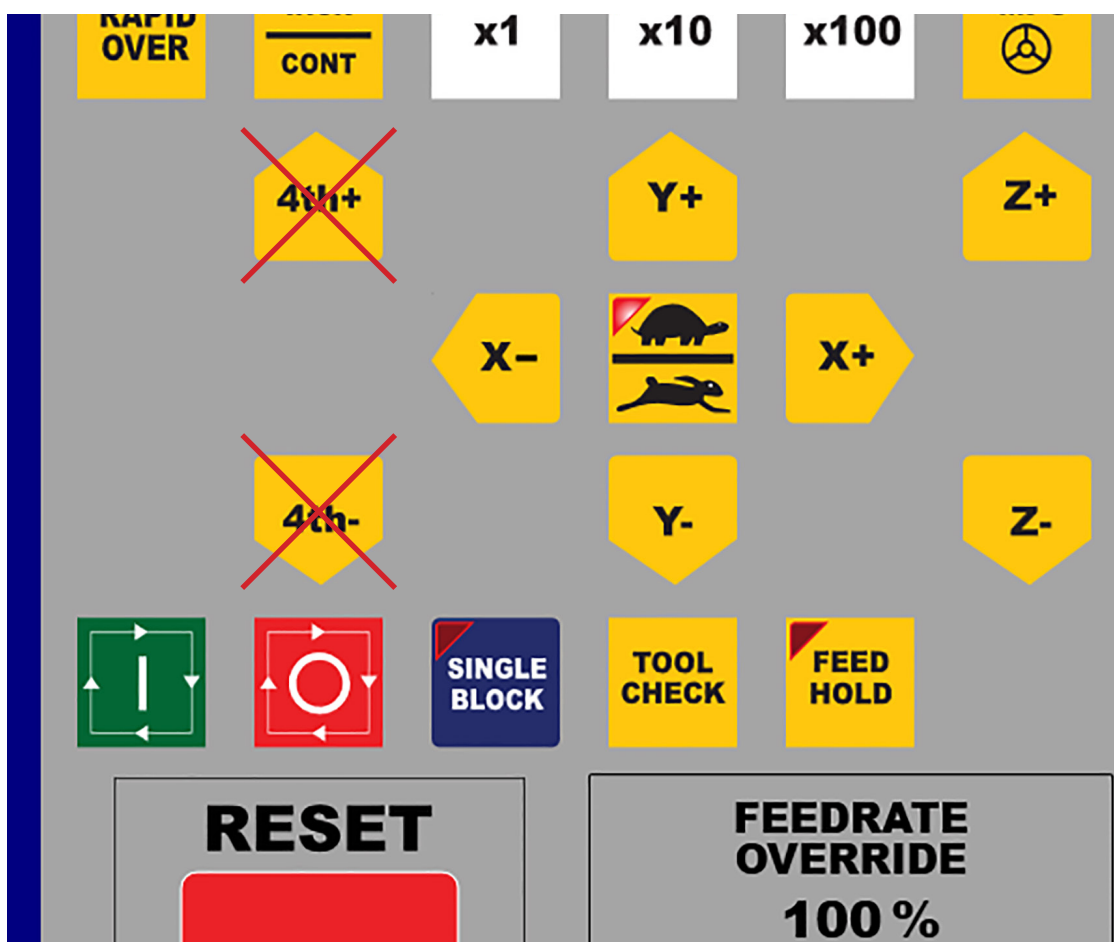
```
<button row="9" column="6">z_negative</button>
  <button row="10" column="2">cycle_cancel</button>
  <button row="10" column="3">single_block</button>
  <button row="10" column="4">tool_check</button>
  <button row="10" column="5">feed_hold</button>
  <button row="10" column="1">cycle_start</button>
</vcp_skin>
```

File, Save, Close CNC12 if it was open, wait a few seconds and then restart CNC12 to see the changes.

Cycle Start button has now been moved to Row 10, Column 1.



How to delete buttons:

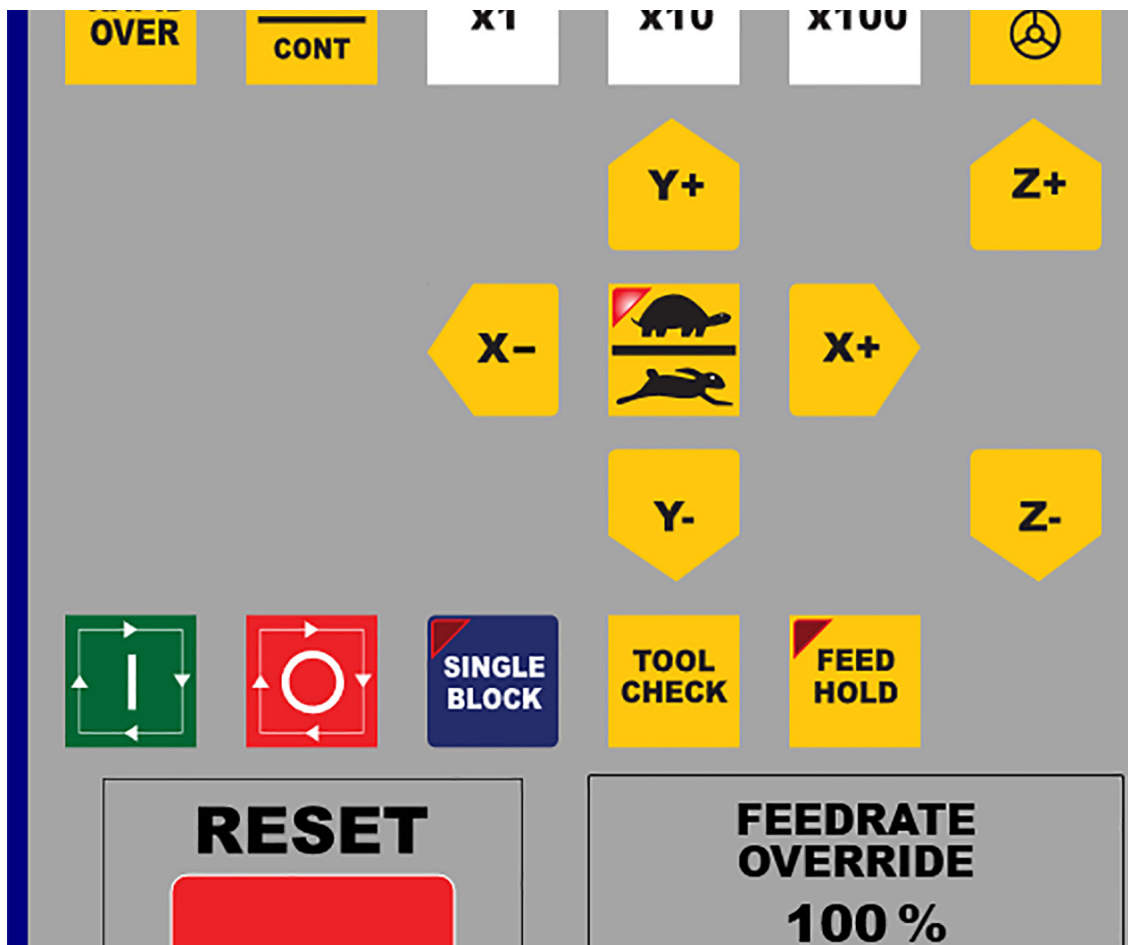


Deleting a button is just as simple as moving one. Open the acorn_mill_skin.vcp file and locate the button(s) to be deleted. In this example we will delete both the 4th axis + and the 4th axis - buttons.

Delete the lines highlighted in yellow below.

```
<button row="6" column="5">x100</button>
<button row="6" column="6">mpg</button>
<del><button row="7" column="2">4th_positive</button></del>
<button row="7" column="4">y_positive</button>
<button row="7" column="6">z_positive</button>
<button row="8" column="3">x_negative</button>
<button row="8" column="4">tortoise_hare</button>
<button row="8" column="5">x_positive</button>
<del><button row="9" column="2">4th_negative</button></del>
<button row="9" column="4">y_negative</button>
<button row="9" column="6">z_negative</button>
<button row="10" column="2">cycle_cancel</button>
<button row="10" column="3">single_block</button>
<button row="10" column="4">tool_check</button>
<button row="10" column="5">feed_hold</button>
<button row="10" column="6">cycle_start</button>
</vcp_skin>
```

File, Save and restart CNC12 to see the changes.



Button Graphics

Button Graphics are located in the 'c:\cncm\resources\vcpl\Buttons\' directory.

Each button has its own folder. For example the X axis positive jog button files are located in the "x_positive" folder here. 'c:\cncm\resources\vcpl\Buttons\x_positive'

Open the x_positive folder and you will find two files: 1.) x_positive.SVG and 2.) x_positive.XML

x_positive.SVG is the graphics file used for the X axis Positive Jog button that you see on the screen.

Any browser can be used to view a .SVG file. So, if you'd like to view one of the button graphics in the button folder simply right click on it and use "Open" or "Open with" and pick the browser of your choice to view the .SVG image.

x_positive.SVG looks like this:



Why SVG?

"SVG" stands for "Scalable Vector Graphics", SVG is an open standard Extensible Markup Language based VECTOR graphic format which is widely adopted and supported, all modern web browsers can render SVG – and most vector drawing programs export it. Centroid chose this format for a number of reasons. 1.) It scales with NO resolution loss. 2.) It is lightweight which keeps the VCP running smooth and reliable. 3.) The format meets the demands of scalability, responsiveness, interactivity, programmability, performance, and accessibility. 4.) SVG's are easy to create and modify. 5.) SVG has a large Online support community with many DIY How-To videos and web pages on how to create SVG files.

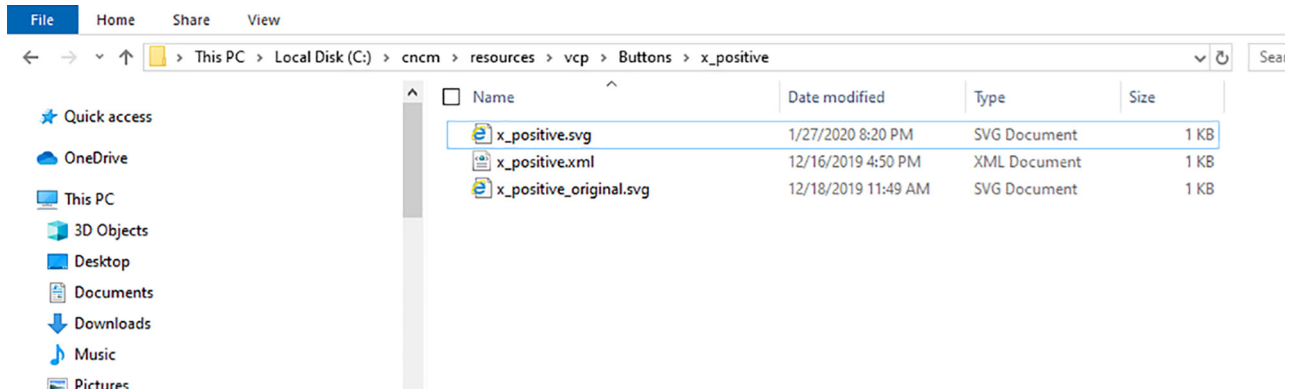
SVG documents are nothing more than simple plain text files that describe lines, curves, shapes, colors, and text. The SVG format is human and machine readable, easily understood and modified, SVG code can be manipulated via graphics programs such as Adobe Illustrator, Corel Draw or Inkscape (a free open source software <https://inkscape.org/>). All of the Centroid supplied SVG images can be opened and edited using Inkscape (or even some basic document editing software like Libre Office Draw and a few Microsoft programs). Inkscape is an open source vector editor that uses SVG as its native format. It can read and write SVG flawlessly. The Inkscape user interface is not as polished as some other higher priced solutions, but once learned, Inkscape is a very powerful tool. It is also totally free and actively developed, so we recommend Inkscape to create trouble free SVG files for the VCP. There are many free online video tutorials on how to use Inkscape.

See the Trouble Shooting section of this manual for tips on good practices to use when creating VCP button graphics with any graphics program.

Changing the Button Graphics

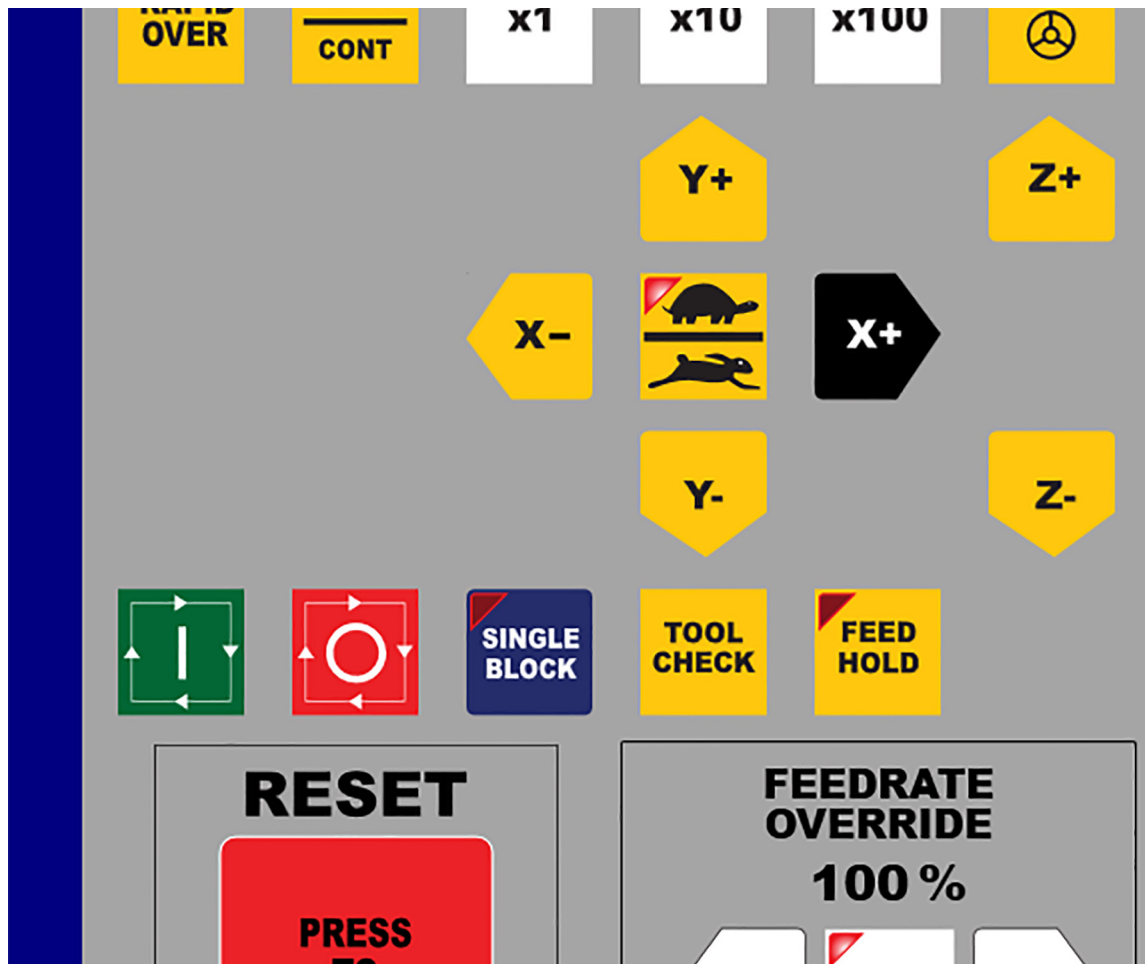
Lets say we want to change the Jog button graphics to suit our taste and instead of the classic safety yellow background with black text we would like a black background and white text. To do this....

Navigate to the x_positive jog button folder and make a backup copy of x_positive.svg.

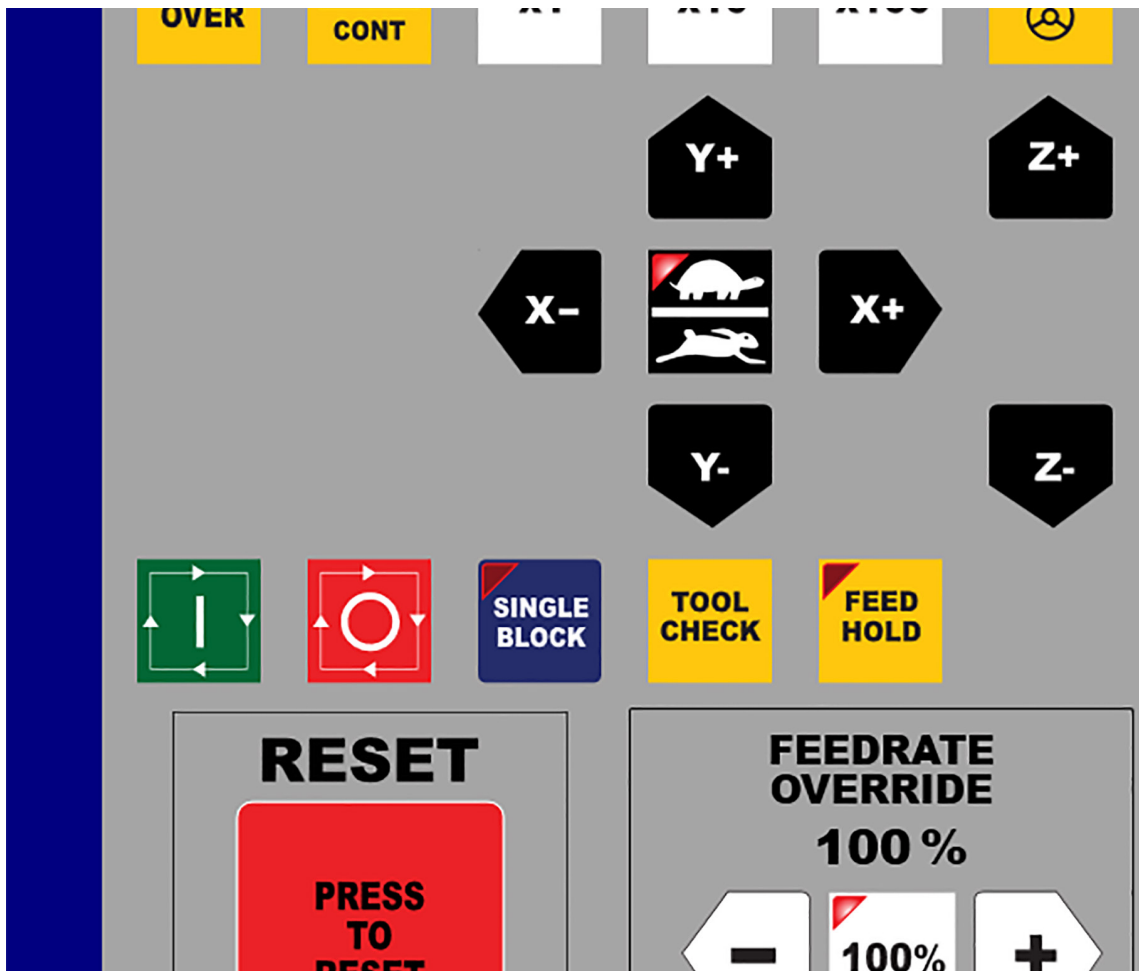


Now that we have a backup copy, open x_positive.svg with a vector graphics editing software such as Inkscape and change the background color to black and the text color to white, be sure to save the edited file in the proper folder, in this case: c:\cncm\resources\vcp\Buttons\x_positive.

Restart CNC12 to see the changes.



Repeat the process for the other buttons.



Change the VCP background color

The VCP background color defaults to the medium grey color.

The background color can be changed to any color simply by adding a line defining the desired VCP background color to the VCP skin XML file.

Navigate to c:\cncm\resources\vcp\skins and open the .vcp file being used. in our example below the file we are editing is "acorn_mill_vcp_skin.VCP"

Below is the first 14 lines of acorn_mill_vcp_skin.vcp

```
<vcp_skin>
  <border>
    <column_span>2</column_span>
    <column_start>1</column_start>
    <fill>#00007F</fill>
    <row_span>4</row_span>
    <row_start>1</row_start>
    <outline_color>#000000</outline_color>
    <outline_thickness>2</outline_thickness>
  </border>
  <border>
    <column_span>5</column_span>
    <row_start>5</row_start>
    <outline_color>#000000</outline_color>
```

To change the VCP background color:

Insert this line `<background># E9E0B7 </background>` just after the first line, like this..

```
<vcp_skin>
  <background># E9E0B7 </background>
  <border>
    <column_span>2</column_span>
    <column_start>1</column_start>
    <fill>#00007F</fill>
    <row_span>4</row_span>
    <row_start>1</row_start>
    <outline_color>#000000</outline_color>
    <outline_thickness>2</outline_thickness>
  </border>
  <border>
    <column_span>5</column_span>
```

New HEX
background color
E9E0B7

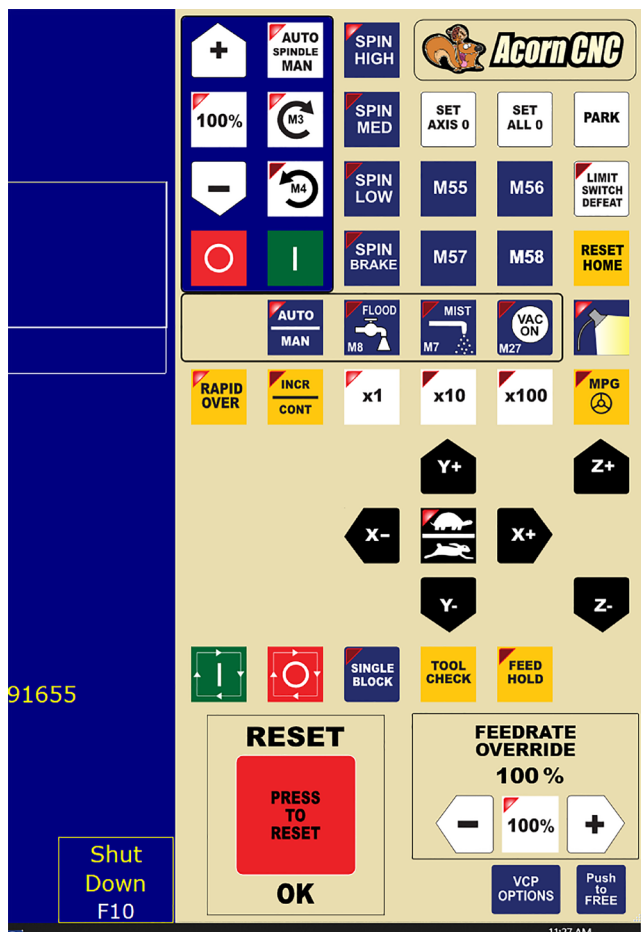
In this example I have changed the background color to a soft Tan color. HEX # E9E0B7

Colors in both the Skin and the Button XML files are expressed in HEX color code. A HEX color is expressed as a six-digit combination of numbers and letters defined by its mix of red, green and blue (RGB). Basically, a HEX color code is short-hand for its RGB values with a little conversion gymnastics in between. No need to sweat the conversion. It is easy to pick out and even create your own HEX color codes using InkScape or any of the free Hex color picker/generators online.

Note: The HEX color code for the default VCP background grey is A6A5A0.

If the background color definition line `<background>#HEXCODE</background>` is missing from the Skin, the VCP defaults to the gray background color #A6A5A0.

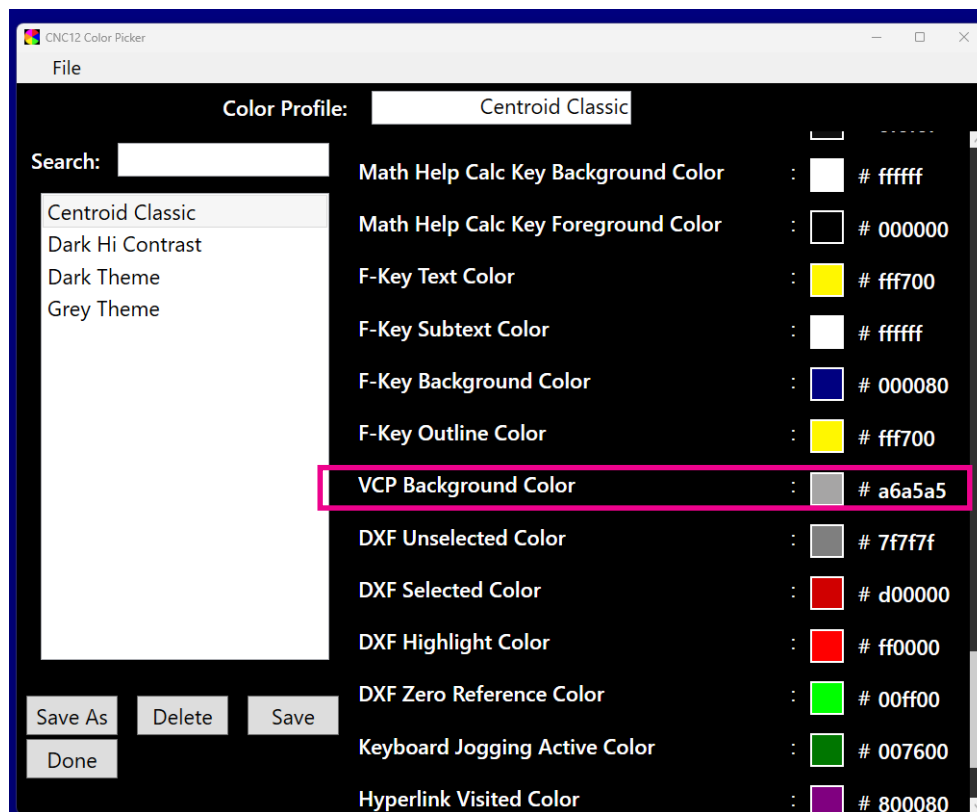
Change the VCP background color



<background># A6A5A0 </background>

HEX
E9E0B7

For those that don't want to edit any XML files alternatively the Centroid CNC12 color picker can also be used to change the VCP background Color.



Change the LED indicator light color

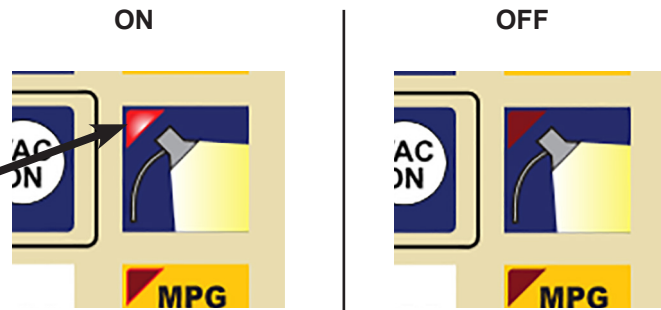
Some VCP buttons have LED indicator lights. These are primarily used to indicate that the function of the button is on or activated for example the Rapid Over and the Feed Hold buttons, LED indicator lights are also used to indicate which choice is currently selected for example the jog speed Fast/Slow (Tortoise and Hare), the Jog Incremental / Continuous selection buttons. The LED's are not part of the button .SVG graphics. The VCP will overlay the LED's onto the button graphics automatically.

Lets take a look at the XML file for the Work Light button.

The two highlighted lines below control the color of the LED in its ON and OFF state.

Work Light XML button code

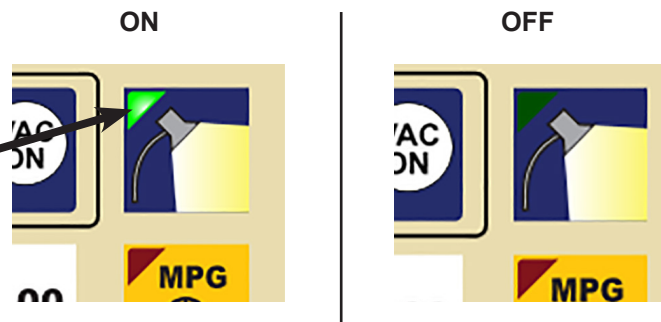
```
<vcp_button>
  <skin_event_num>20</skin_event_num>
  <plc_output>
    <number>1076</number>
    <color_on>#EC1C24</color_on>
    <color_off>#81151C</color_off>
  </plc_output>
</vcp_button>
```



I want to change the LED color to a bright green (#00FF00) so that when the work light is active the LED will be bright green and when work light is OFF the LED will be black. Using Inkscape i choose a bright green HEX color #00FF00 and decided to use black #000000 for the LED OFF color.

Now edit the work_light.XML file as seen below.

```
<vcp_button>
  <skin_event_num>20</skin_event_num>
  <plc_output>
    <number>1076</number>
    <color_on>#00FF00</color_on>
    <color_off># 004F00 </color_off>
  </plc_output>
</vcp_button>
```



Note: The VCP hard code automatically puts the radial gradient effect on the LED.

How to remove the LED entirely from a button

Delete the highlighted lines below to remove the LED indicator light from a button.

```
<vcp_button>
  <skin_event_num>20</skin_event_num>
  <plc_output>
    <number>1076</number>
    <color_on>#00FF00</color_on>
    <color_off>#000000</color_off>
  </plc_output>
</vcp_button>
```

LED indicator removed.

```
<vcp_button>
  <skin_event_num>20</skin_event_num>
</vcp_button>
```



Swap Button Image when button is pressed.

Sometimes it is desired that while a button is being pressed to have a completely different image for that button be displayed while the button is being clicked on. This “image swap” while clicking the button can be used to create a variety of UI effects. The XML tag called an “on_click_swap” as the name implies will swap the button image with another when a VCP button is being clicked on when this tag is in use.

For example: Lets use the on_click_swap tag to change the X axis positive jog button to another image when the X axis positive jog button is being pressed/clicked on.

1.) To start, lets create a new “swap” image .SVG file. Start Inkscape and open the x_positive.svg image file located in the c:\cncm\resources\vcp\buttons\x_positive folder and save it as a new file, in this case “x_positive_swap.svg” and save it in the same x_positive button folder. Now edit the image with Inkscape to make x_positive_swap.svg image look different than the x_positive.svg image. For this example i decided to change the background color of the swap image from yellow to green.



original x_positive.svg image file



save file as x_positive_swap.svg and edit
, in this case background color to green

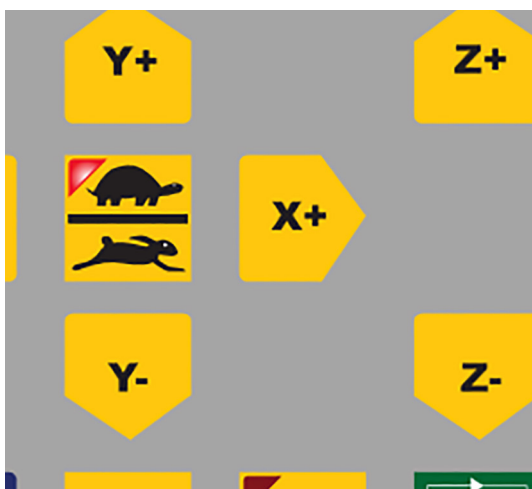
2.) Now that we have the swap image prepared, open “x_positive.xml” located in the c:\cncm\resources\vcp\buttons\x_positive folder and you will see these three lines which tell the VCP what the x_positive button does.

```
<vcp_button>  
  <skin_event_num>39</skin_event_num>  
</vcp_button>
```

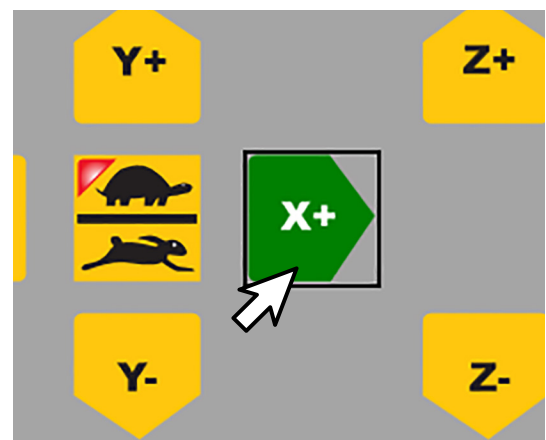
Now, add the swap image tag <on_click_swap> line with the name of the swap image file in between the tag and the closing tag as seen in yellow highlight below

```
<vcp_button>  
  <skin_event_num>39</skin_event_num>  
  <on_click_swap>x_positive_swap.svg</on_click_swap>  
</vcp_button>
```

and save the file. Restart CNC12 and the result is as follows.



X Positive Button when not pressed



X Positive Button graphic when being pressed or clicked on

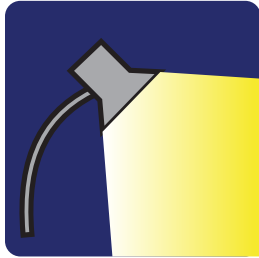
Swap Button Image when button PLC Output function is activated.

On some occasions it is desirable to have the VCP to use one button graphic for when the function of the button is activated and another graphic for when the button is deactivated. Said another way the VCP will swap back and forth between two different .SVG button graphics depending on the state of the button function. This works exactly like the LED indicator light but instead of having the LED to indicate the state of a button function the VCP will use two distinct button graphics for the ON and OFF states. For our example lets make the VCP Work Light button swap between two different Work Light graphics depending on whether or not the Work Light Output is ON or OFF.

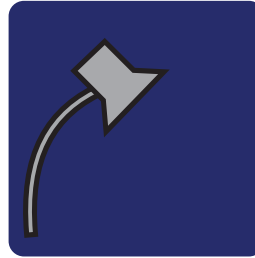
Start by creating the Work Light OFF image and saving it in the same Work Light Button folder.

For this example i saved the original 'work_light.SVG' as 'work_light_off.SVG' and edited it to create the Work Light OFF image i deleted the yellow light from the original image so the light is dark to represent the light being in the OFF state.

Original work_light.SVG



Edited work_light_off.SVG



Starting with the original Work Light XML button code with red LED overlay
Delete the two highlighted LED control lines

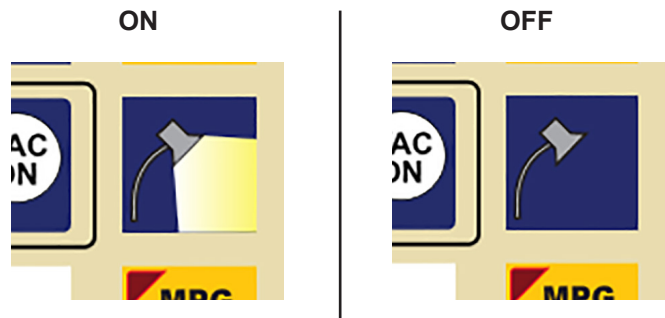
```
<vcp_button>
  <skin_event_num>20</skin_event_num>
  <plc_output>
    <number>1076</number>
    <color_on>#EC1C24</color_on>
    <color_off>#81151C</color_off>
  </plc_output>
</vcp_button>
```



Original Work Light w/ LED

Add the two highlighted image_on and image_off command lines seen highlighted below.

```
<vcp_button>
  <skin_event_num>20</skin_event_num>
  <plc_output>
    <number>1076</number>
    <image_on>work_light.svg</image_on>
    <image_off>work_light_off.svg</image_off>
  </plc_output>
</vcp_button>
```



Swap Button Image when an Input is activated.

Often it is useful to have a VCP button act as an indicator light. A light that comes on when an input is active. This effect can be achieved by swapping the Button image from one to another based on the state of an input.

For Example: Let's create a Virtual Probe indicator light that lets us know if the Probe is clear or tripped. Create a new button folder "probe_indicator" and then place in that folder the two button graphics, one for when the input is activated and the other image for when the input is off.

I created two SVG files, one called "probe_trip.svg" and the other called "probe_clear.svg"



Now create a button XML file called "probe_indicator.xml" and save in the same folder and edit as follows:

```
<vcp_button>
  <plc_input>
    <number>7</number>
    <image_on>probe_trip.svg</image_on>
    <image_off>probe_clear.svg</image_off>
  </plc_input>
</vcp_button>
```

"use this tag to let the VCP know we are linking this button to a plc input"

"use this tag to let the VCP know which input to watch"

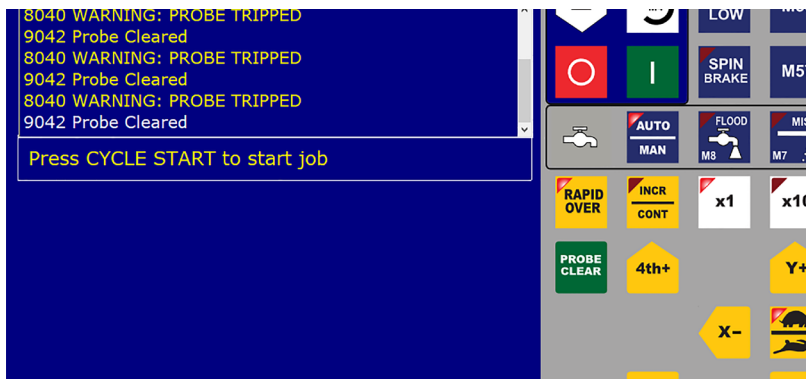
"use this tag to let the VCP know which images to use"

Note: The input definition "Probe Tripped" has been assigned using the Acorn Wizard to input #7

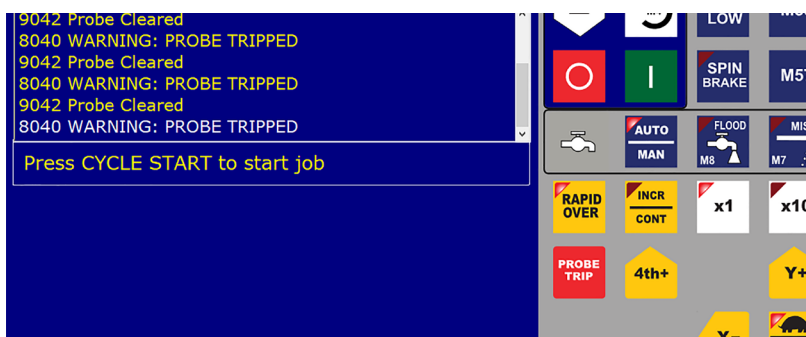
Now place the new button anywhere on the VCP grid by adding this line to the VCP skin.

```
<button row="7" column="1">probe_indicator</button>
```

I choose to place it on Row 7 and Column 1, save and restart CNC12 and here is the result.



Probe Clear (not tripped)
same as Input 7 open/not active



Probe Tripped
same as Input 7 closed/activated.

Create a new button folder with a new button image(.svg) and button XML file.

Each button has its own folder. To create an all new button, create a new folder. Navigate to c:\cncm\resources\vcp\Buttons and simply right click "New", "Folder".

In this example I'll create a new button to run a Cross Hair Laser to set the X0 and Y0 position.

- 1.) Create a folder called 'laser_set_xy'
- 2.) Copy the M56 button XML file into the laser_set_xy folder and rename it laser_set_xy.XML
- 3.) Create the graphic(s) for the Laser Set XY button with Inkscape and place a copy of the .SVG graphic into the 'laser_set_xy' folder and name the SVG file 'laser_set_xy'. below is the Laser Alignment Graphic I created.



- 4.) Edit the VCP skin XML file, in this example 'acorn_mill_skin.VCP' and insert (or modify an existing) a button line and assign the location in which you wish the new button to appear on the VCP.

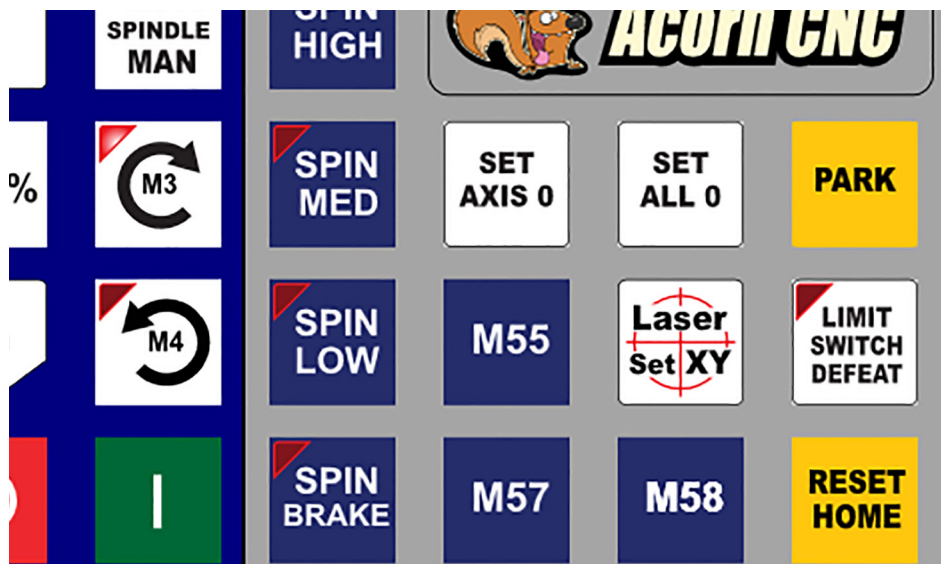
In this example I want the Laser Set XY button to appear on the VCP in the same location that is currently occupied by M56, so, all i need to do is locate and edit the M56 line. Below is an excerpt of the VCP skin that contains the M56 line.

```
<button row="3" column="2">spindle_ccw</button>
<button row="3" column="3">spin_low</button>
<button row="3" column="4">m55</button>
<button row="3" column="5">m56</button>
<button row="3" column="6">limit_switch_defeat</button>
<button row="4" column="1">spindle_cancel</button>
<button row="4" column="2">spindle_start</button>
```

Now change the button name from 'm56' to 'laser_set_xy' and keep the location (Row 4 , Column 4) the same. This is effectively replacing the M56 button graphics and XML file with the Laser Set XY graphic and XML file.

```
<button row="3" column="2">spindle_ccw</button>
<button row="3" column="3">spin_low</button>
<button row="3" column="4">m55</button>
<button row="3" column="5">laser_set_xy</button>
<button row="3" column="6">limit_switch_defeat</button>
<button row="4" column="1">spindle_cancel</button>
<button row="4" column="2">spindle_start</button>
```

Now save and re-start CNC12, the new Laser Set XY Graphic will appear on the VCP,



But, there is a problem! When the new Laser Set XY button is pressed, the new Laser Set XY button runs the stock M56 macro. No problem. Edit the M56 macro! the file is 'mfunc56.mac' (stands for M function number 56. macro, located in the c:\cncm folder) with the commands necessary to run the Laser and Set the X Y zero location.

below is a typical Laser Set X0Y0 macro:

```
;Set X0 and Y0 position to the Laser cross hairs
;
; Use and edit the values of the variables below to adjust this macro to suit your particular application
;
#101 = 5      ; Edit this value to set the length of time in seconds to wait for the M225 message
#102 = 3.654  ; Edit this value to set the X offset of Laser from the spindle center line (pos or neg)
#103 = 5.000  ; Edit this value to set the Y offset of Laser from the spindle center line (pos or neg)
#104 = 0      ; Edit this value to set the Length of time to wait for 2nd M225 message
#105 = -3.65  ; Edit this value to set the ABS coordinate to position Z for Laser focus

G53 G90 Z [#105]      ; Set to Laser focus height
M94/29                ; Turn on the Laser, activates the "LaserAlignActivate" output
M225 #101 ("#)**MOVE CROSSHAIRS TO DESIRED LOCATION**\nPRESS CYCLE START when ready to set XY0"
M0                    ; Stop and wait for operator to press cycle start to continue.
G91 G0 X [#102] Y [#103] ; Move the centerline of the spindle to the cross hair location incrementally
G90                    ; Return to absolute positioning
G92 X0 Y0              ; Set the current WCS X and Y position to X0, Y0
G4 P 0.5               ; 1/2 second dwell
M25                    ; Retract Z to Z home position
M95/29                ; Turn off the Laser, De-activates the "LaserAlignActivate" output
```

(See "Introduction to Centroid Acorn CNC12 Macros" and the Mill and Lathe operator manuals for more information on macro programming.)

Save the mfunc56.mac file, and re-start CNC12 and now the Laser Set XY button will run the Laser macro above.

Note: A requirement for this macro to work is one of the outputs must have been assigned in the Wizard to be 'LaserAlignActivate' and the Laser wired to that output.

Mill CNC Control Configuration Wizard

Primary System

- Axis Drive Type
- Input Definitions
- Output Definitions**

Axis

- Configuration
- Homing and Travel
- Axes Pairing
- Advanced

Spindle

- Setup

Touch Devices

- Probe
- Tool Touch Off

Control Peripheral

- Input Devices
- Wireless MPG

DB25 Connector

Output Type: General Purpose

LubePump
SpindleBrakeRelease
SpinFWD
SpinREV
Flood
TurnClampOn
G540SpinRevOff
G540SpinFwdOff
Mist
VacuumOn
DustCollectionOn
OUTPUT1
OUTPUT2
OUTPUT3
OUTPUT4
OUTPUT5
OUTPUT6
OUTPUT7
OUTPUT8
CutOff
PartChute
Axis1BrakeRelease

Acorn Integrated Outputs 1-8

		Definition
1	OUT1	NoFaultOut
2	OUT2	DriveResetOut
3	OUT3	ChargePump
4	OUT4	VFDResetOut
5	OUT5	RouterDustCollection
6	OUT6	AirBlowActivate
7	OUT7	PopUpPins
8	OUT8	LaserAlignActivate

Click and Drag an Output function definition from list to the Output number Definition box to assign a function to an output

Ethe

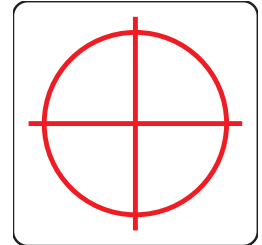
Add a simple ON/OFF button that turns on an Output and Turns Off and Output

It might be nice to have in addition to the Laser Set XY button to also have a button that simply turns ON the Laser cross hair device when pressed and then turns it OFF when pressed again. This would be nice for several reasons mostly allowing the operator to easily turn on the laser when it is desired to have it ON when not running the Laser Set XY macro. So lets use this as an example of how to create a new VCP button that acts as a simple ON/OFF switch.

Create a new button folder navigate to c:\cncm\resources\vcp\Buttons and simply right click "New", "Folder". In this example I'll create a new button folder called 'laser_on_off'

- 1.) Create a folder called 'laser_on_off'
- 2.) Copy the M57 button XML file into the laser_on_off folder and rename it laser_on_off.XML
- 3.) Create the graphic(s) for the Laser ON/OFF button with Inkscape and place a copy of the .SVG graphic into the 'laser_on_off' folder and name the SVG file 'laser_on_off'. below is the Laser ON/OFF Graphic I created.

- 4.) Edit the VCP Skin XML file, in this example 'acorn_mill_skin.VCP' and insert (or modify an existing) a button line and assign the location in which you wish the new button to appear on the VCP.

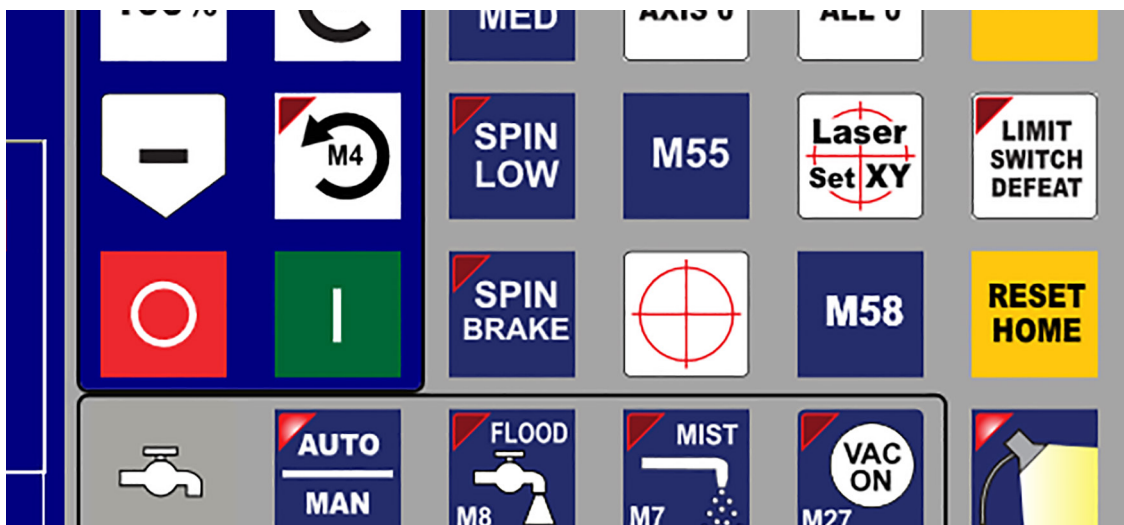


In this example I want the Laser ON/OFF button to appear on the VCP in the same location that is currently occupied by M57, so, all i need to do is locate and edit the M57 line. Below is an excerpt of the VCP skin that contains the M57 line.

```
<button row="3" column="4">m55</button>
<button row="3" column="5">laser_set_xy</button>
<button row="3" column="6">limit_switch_defeat</button>
<button row="4" column="1">spindle_cancel</button>
<button row="4" column="2">spindle_start</button>
<button row="4" column="3">spin_brake</button>
<button row="4" column="4">m57</button>
<button row="4" column="5">m58</button>
<button row="4" column="6">reset_home</button>
<button row="5" column="2">coolant_auto_man</button>
<button row="5" column="3">flood_coolant</button>
<button row="5" column="4">mist_coolant</button>
```

now edit the m57 line to look like this `<button row="4" column="4">laser_on_off</button>`

save the XML file and restart CNC12 and laser_on_off.svg will appear on the VCP.



Now, if you press the button at this point, it will run the M57 macro 'mfunc57.mac' file since we copied the M57 XML file. Well we don't want it to do that so lets modify the laser_on_off.XML file, currently it looks like this.

```
<vcp_button>
  <skin_event_num>67</skin_event_num>
</vcp_button>
```

The skin event number '67' has been assigned by the Acorn PLC program for the m-code M57. This way when you press the M57 button the Acorn PLC program knows that you want skin event #67 and then CNC12 follows the PLC programming for that event when this VCP button is pressed. In this case, running the M57 macro mfunc57.mac.

So, lets edit 'laser_on_off.XML' with the Skin Event number associated with the Laser Output Relay on the Acorn.

Wizard Acorn Output Name	Skin Event Number	Associated LED output number
LaserAlignActivate	207	1127

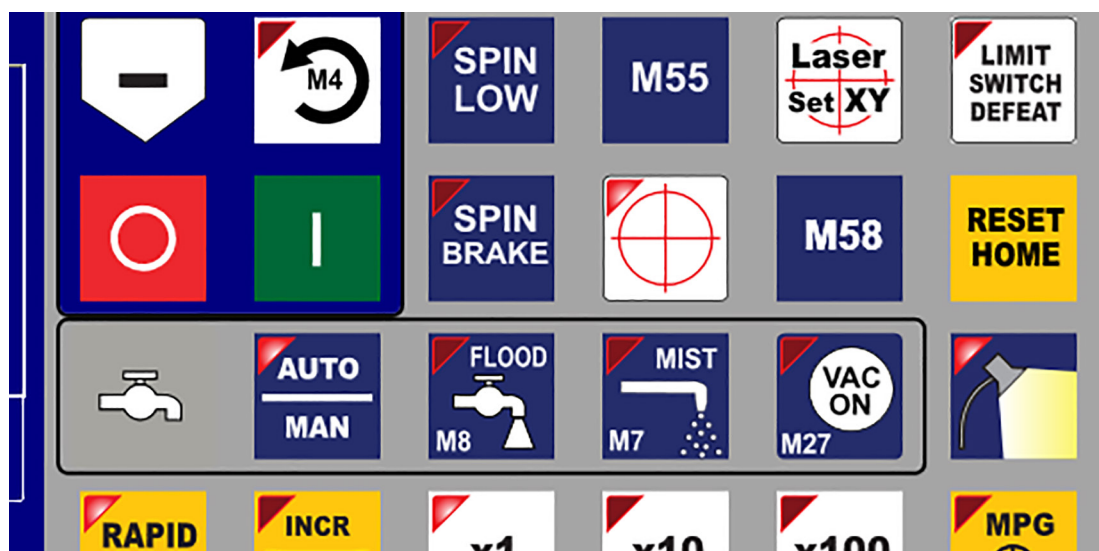
```
<vcp_button>
  <skin_event_num>207</skin_event_num>
</vcp_button>
```

I knew to use Skin Event #207 for the Laser by looking at the Acorn Output Skin Event table below or you can find all skin event numbers in the Acorn PLC source file (acorn_mill_plc.src)

Save the file, restart CNC12 and now the Laser ON/OFF button will turn on and off the output assigned in the Wizard as "LaserAlignActivate". Cool! but the LED is missing that indicates the state of the button and laser. So lets add the indicator LED into the laser_on_off.XML file. Add the lines highlighted below.

```
<vcp_button>
  <skin_event_num>207</skin_event_num>
  <plc_output>
    <number>1127</number>          (LED number associated with the 'LaserAlignActivate' output)
    <color_on>#EC1C24</color_on>   (LED ON color)
    <color_off>#81151C</color_off> (LED OFF color)
  </plc_output>
</vcp_button>
```

Now the Laser Align button will Activate the LaserAlignActivate output when pressed and deactivate it when pressed again. a simple output ON/OFF button.



Button Functionality assignments with Skin Event numbers

Outputs defined in the Wizard have an associated skin event number, you will notice that the VCP button XML files for the stock VCP buttons use these Skin Event numbers to link the button to the functionality defined in the PLC program.

All skin_events numbers can be viewed in the PLC source file that the Acorn Wizard creates based on the Wizard choices. For the examples used in this document that file is: `acorn_mill_plc.src` located in the `cncm` directory it is a text file, open it with Notepad ++. Select “Search” then “Find” and search for “skin_event”. that will bring you to this section of the PLC source file. Below is example with the first 23 skin events of the complete list contained in the plc program source file.

```
-----  
; System variables: Virtual Control Panel Events  
-----  
;Maximum Number of Skin Events is 255  
  
SkinSpinOverPlus_M      IS SV_SKIN_EVENT_1 ; Row 1 Column 1  
SkinSpinAutoMan_M       IS SV_SKIN_EVENT_2 ; Row 1 Column 2  
SkinAux1_M              IS SV_SKIN_EVENT_3 ; Row 1 Column 3  
SkinAux2_M              IS SV_SKIN_EVENT_4 ; Row 1 Column 4  
SkinAux3_M              IS SV_SKIN_EVENT_5 ; Row 1 Column 5  
SkinSpin100_M           IS SV_SKIN_EVENT_6 ; Row 2 Column 1  
SkinSpinCW_M            IS SV_SKIN_EVENT_7 ; Row 2 Column 2  
SkinAux4_M              IS SV_SKIN_EVENT_8 ; Row 1 Column 6  
SkinAux5_M              IS SV_SKIN_EVENT_9 ; Row 2 Column 3  
SkinAux6_M              IS SV_SKIN_EVENT_10 ; Row 2 Column 4  
SkinSpinOverMinus_M     IS SV_SKIN_EVENT_11 ; Row 3 Column 1  
SkinSpinCCW_M           IS SV_SKIN_EVENT_12 ; Row 3 Column 2  
SkinAux7_M              IS SV_SKIN_EVENT_13 ; Row 2 Column 5  
SkinAux8_M              IS SV_SKIN_EVENT_14 ; Row 2 Column 6  
SkinAux9_M              IS SV_SKIN_EVENT_15 ; Row 3 Column 3  
SkinSpinStop_M          IS SV_SKIN_EVENT_16 ; Row 4 Column 1  
SkinSpinStart_M         IS SV_SKIN_EVENT_17 ; Row 4 Column 2  
SkinAux10_M             IS SV_SKIN_EVENT_18 ; Row 3 Column 4  
SkinAux11_M             IS SV_SKIN_EVENT_19 ; Row 3 Column 5  
SkinAux12_M             IS SV_SKIN_EVENT_20 ; Row 3 Column 6  
SkinCoolAutoMan_M       IS SV_SKIN_EVENT_21 ; Row 5 Column 1  
SkinCoolFlood_M         IS SV_SKIN_EVENT_22 ; Row 5 Column 2  
SkinCoolMist_M          IS SV_SKIN_EVENT_23 ; Row 5 Column 3.....
```

To observe what Skin Event Number (`skin_event_num`) is being utilized by any VCP button open the button XML file.

For example, open the folder `c:\cncm\resources\vcp\buttons\flood_coolant\flood_coolant.xml`

22 is the number assigned to the Flood Pump button logic in the PLC program

Also note the associated Flood Pump LED output number is **1078**. The LED logic in the PLC program determines whether the LED is ON or OFF when the Flood Pump is ON or OFF, assigning 1078 to the Flood LED in the XML file tells the VCP to use the LED logic associated with 1078: Flood button LED. This is how you get a button to use logic contained in the PLC program.

```
<vcp_button>  
  <skin_event_num>22</skin_event_num>  
  <plc_output>  
    <number>1078</number>  
    <color_on>#EC1C24</color_on>  
    <color_off>#81151C</color_off>  
  </plc_output>  
</vcp_button>
```

What exactly is a Skin Event Number?

The 'skin_event_num' is the way PLC Logic can be assigned to a VCP button. The Skin Event Number is associated with a piece of logic used to create the desired action of a machine tool function. This logic is described in the PLC program and the Skin Event Number is also in the PLC program, Skin Event number is used to assign that function to a particular VCP button. Said another way...The CNC12 knows what function you want for a particular VCP button from this number.

For example: Lets take a look at the Work Light button XML file.

skin_event_num_73 Tells CNC12 to use the logic that has been associated with skin_event_num 73 in the PLC program. So when we use the number 73 in a Button XML file CNC12 knows we want the Work Light Logic. skin_event_num_73 = the Work Light ON / OFF output logic and corresponds to the WorkLight Output that was selected in the Acorn Wizard.

Here is the Work Light button XML file

```
<vcp_button>
  <skin_event_num>73</skin_event_num>
  <plc_output>
    <number>1112</number>
    <color_on>#EC1C24</color_on>
    <color_off>#81151C</color_off>
  </plc_output>
</vcp_button>
```

Wizard Acorn Output Name	Skin Event Number	Associated LED output number
Worklight	73	1112

So, stick with me here, the next natural question is “then how do i know what the Skin Event Numbers are and what they relate to?”

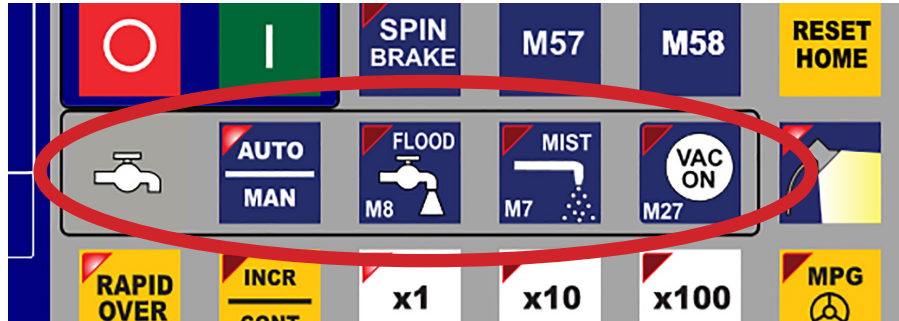
Open the button XML file to see the skin event number and then using that info open the PLC program source and search for the skin event number and read the related PLC logic. The skin event number 73 in the example above has been assigned by the Acorn PLC program to the Work Light logic contained in the stock PLC program and Worklight output is controlled by this logic. This way when you press the Work Light button the Acorn PLC program knows that you want skin event #73 and then CNC12 follows the PLC programming for that event when this VCP button is pressed. In this case, turning the Work Light Output ON and then OFF on each press of the Work Light VCP button. (The Work Light Output itself, where you actually wire up the work light to, is chosen by the user in the Wizard, the Skin Event Number for a particular function stays the same no matter what physical output number is used in the Wizard by the user)

Note: You can edit the Centroid provided PLC program and create your own custom logic and associate a skin event number with that logic but, this is a discussion for another document. Even though the Centroid PLC program is open source and fully customizable most users will not need to do this and the few that do will typically hire a pro to make those types customizations for them. That being said Centroid provides the PLC language, compiler and tools such as the PLC detective (live PLC program monitoring) for free so any type of customizations can be accomplished to meet the CNC machine tool application by anyone that wishes to do so. Most DIY users do not need to learn or perform this type of customization, I just mention it here to let the reader know that the VCP and the PLC program are designed to work together and both are editable. Most user can heavily edit the VCP to suit there application with button graphics and function changes and by editing any associated macros.

If you are interested in also learning Centroid PLC programming there is a users manual on our website and a introduction to Centroid PLC programming video series on Centroid's Tech Support YouTube channel.

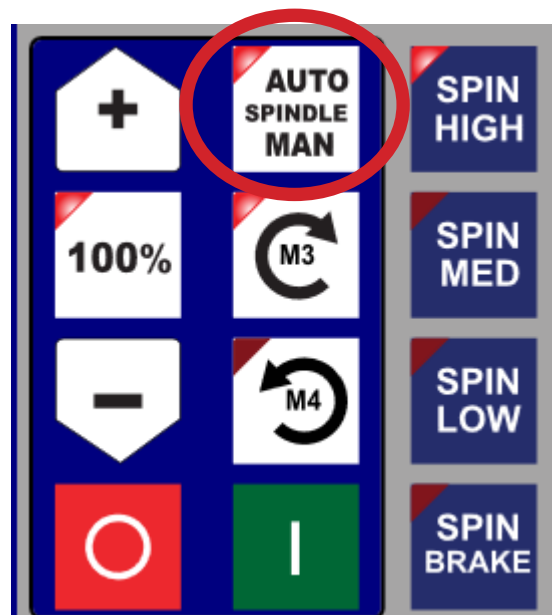
VCP buttons that are interlocked

The VCP buttons Flood, Mist, Vacuum on/off, Router Dust Collection, Router Vac Hold down work in conjunction with the Auto / Man button this is an interlock so when in Auto mode the G code program is in control of these functions, unless the user selects MAN and overrides Flood, Mist, Vac. Manual is also used to activate these function when not running a G-code program.



The Auto/Man button is required for Flood, Mist, Vacuum on/off, Router Dust Collection, Router Vac Hold down functions to work as designed (standard cnc controller logic), so don't delete it unless you really don't need these functions.

Same with the Spindle Auto/Man button.



Adding Static Graphics such as Logos and Icons

Static Graphics can be used as icons or logos with the VCP. The VCP will overlay these graphics on top of VCP itself.

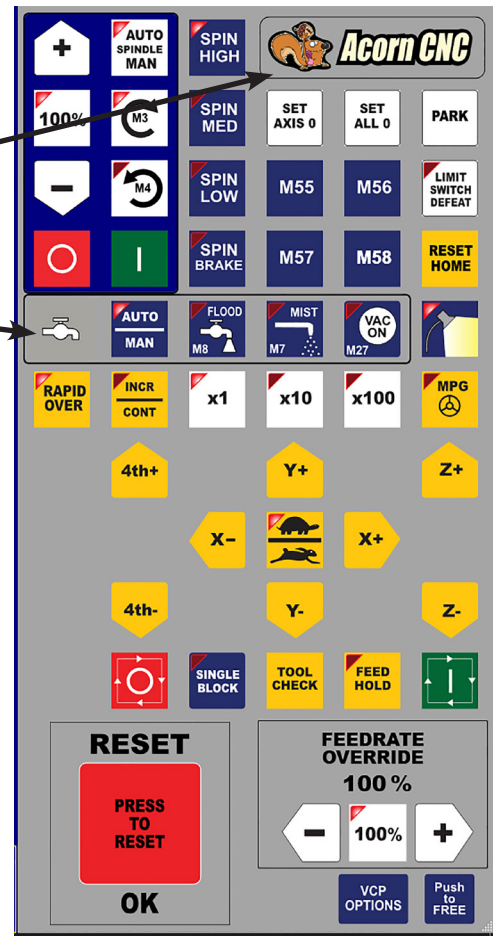
Two examples of VCP static graphics are the Acorn Logo
and the Coolant icon.

These static images are in the same .SVG format as the button images.
The static images live in the following directory:
c:\cncm\resources\vcp\images\

'coolant.svg' and 'acornlogo.svg'

The static images are controlled in the VCP Skin.
Let's take a look at the default Acorn Mill Skin XML file.
Below are the first 33 lines of 'mill_vcp_skin.VCP'
Yellow highlight = Coolant Icon static image
Blue highlight = Acorn Logo Static image

```
vcp_skin>
<border>
  <column_span>2</column_span>
  <column_start>1</column_start>
  <fill>#00007F</fill>
  <row_span>4</row_span>
  <row_start>1</row_start>
  <outline_color>#000000</outline_color>
  <outline_thickness>2</outline_thickness>
</border>
<border>
  <column_span>5</column_span>
  <column_start>1</column_start>
  <fill>Transparent</fill>
  <row_span>1</row_span>
  <row_start>5</row_start>
  <outline_color>#000000</outline_color>
  <outline_thickness>2</outline_thickness>
</border>
<image>
  <column_span>1</column_span>
  <column_start>1</column_start>
  <row_span>1</row_span>
  <row_start>5</row_start>
  <path>C:\cncm\resources\vcp\images\coolant.svg</path>
</image>
<image>
  <column_span>3</column_span>
  <column_start>4</column_start>
  <row_span>1</row_span>
  <row_start>1</row_start>
  <path>C:\cncm\resources\vcp\images\acornlogo.svg</path>
</image>
```



Coolant Static Image

```
<image>  
  <column_span>1</column_span> ; column span is the distance in columns that the image will overlay  
  <column_start>1</column_start> ; column start is the column number to start the span  
  <row_span>1</row_span> ; row span is the distance in rows that the image will overlay  
  <row_start>5</row_start> ; row start is the position that the image will start the row span  
  <path>C:\cncm\resources\vcpl\images\coolant.svg</path> ; the image filename and path  
</image>
```

The coolant image only takes up one button space to it has a span start and stop of 1 for both the Row Span and the Column Span.

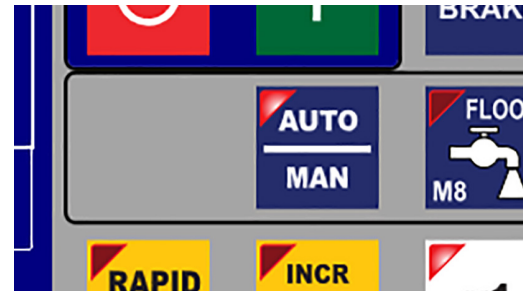
To replace this image with another simply edit the filename and point the VCP to a different image

```
<image>  
  <column_span>1</column_span>  
  <column_start>1</column_start>  
  <row_span>1</row_span>  
  <row_start>5</row_start>  
  <path>C:\cncm\resources\vcpl\images\vacuum_pump.svg</path>  
</image>
```



To delete this image simply delete these lines from the skin. This space is now available for a button if desired.

```
<image>  
  <column_span>1</column_span>  
  <column_start>1</column_start>  
  <row_span>1</row_span>  
  <row_start>5</row_start>  
  <path>C:\cncm\resources\vcpl\images\coolant.svg</path>  
</image>
```

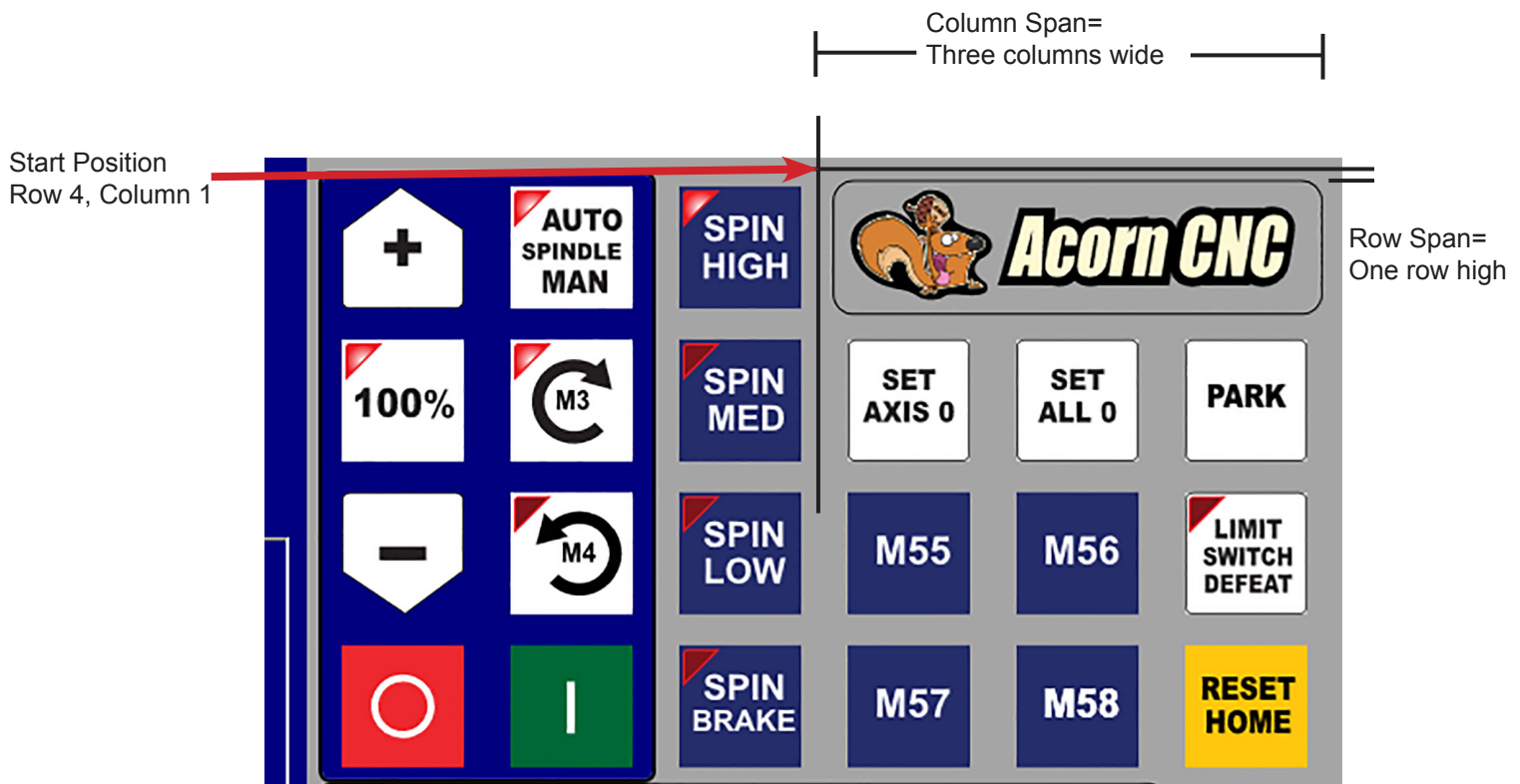


Acorn Logo Static Image

```

<image>
  <column_span>3</column_span>    ; column span is three button lengths
  <column_start>4</column_start>    ; column start is the position to start the span
  <row_span>1</row_span>            ; row span is one row high
  <row_start>1</row_start>          ; row start is the position that the image will start the row span
  <path>C:\cncm\resources\vcpl\images\acornlogo.svg</path> ; the image filename and path
</image>

```



Change the Logo to another image

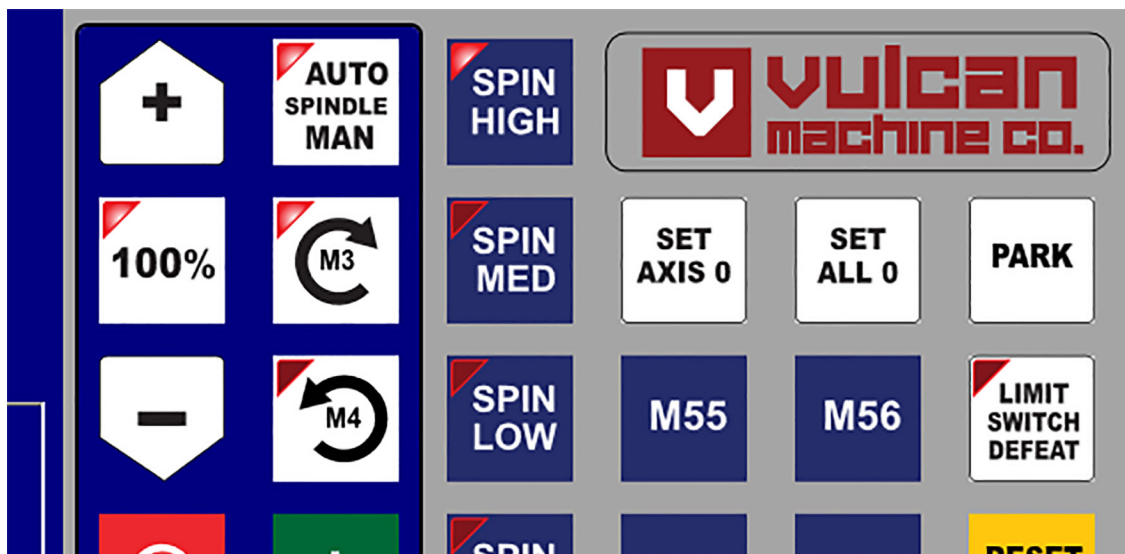
Create a new image and save into the VCP images folder C:\cncm\resources\vcpl\images

Edit the Skin XML image file name, for this example we created logo for Vulcan Machine Tool

```

<path>C:\cncm\resources\vcpl\images\vulcan_machine_logo.svg</path>

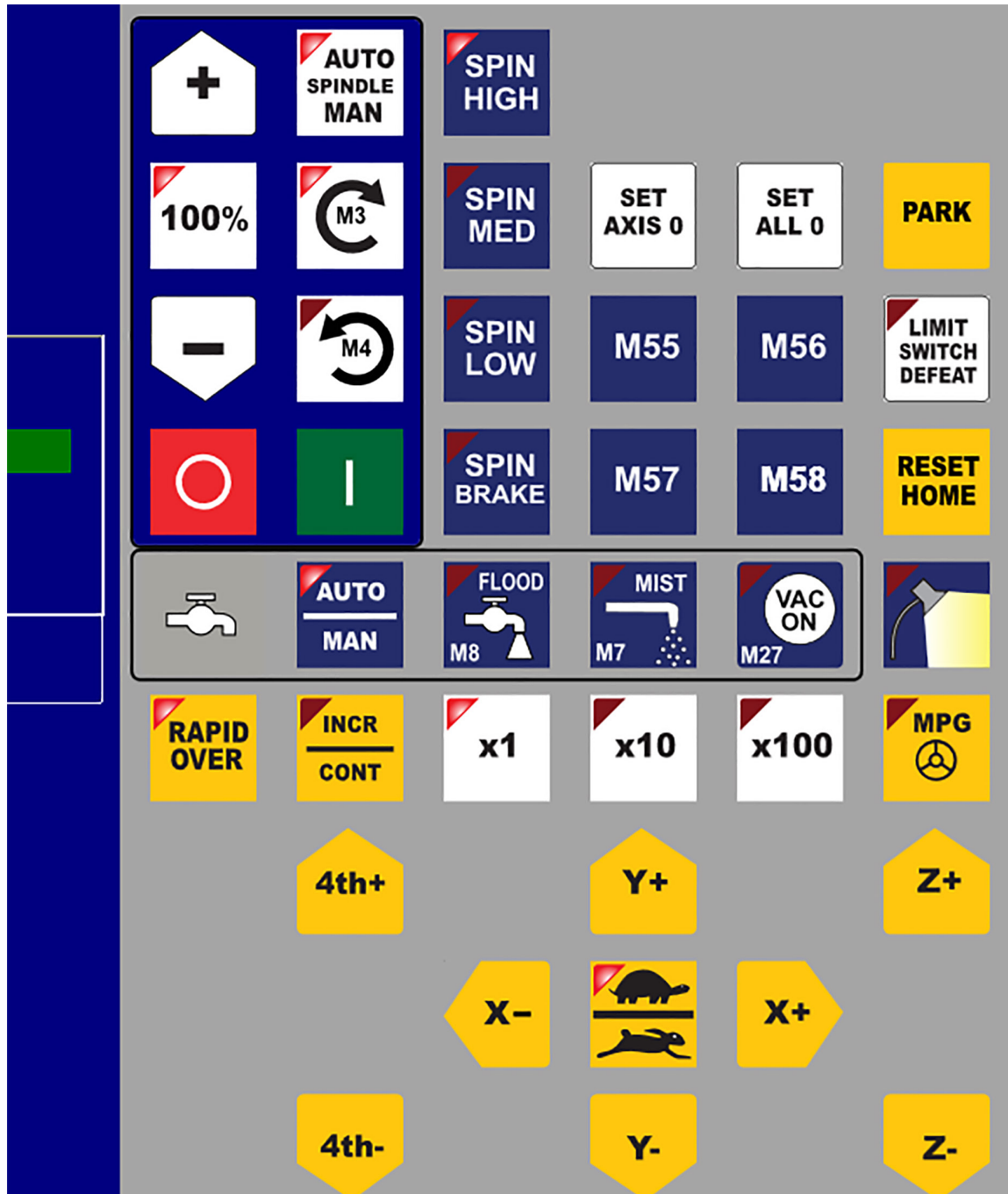
```



Delete the Logo to make room for more buttons

Delete the code that specifies the logo in the skin XML file.

```
<image>  
  <column_span>3</column_span>  
  <column_start>4</column_start>  
  <row_span>1</row_span>  
  <row_start>1</row_start>  
  <path>C:\ncm\resources\vcpl\images\acornlogo.svg</path>  
</image>
```



Create outline borders and solid fill backgrounds to signify related groups of buttons or other effects.

In the Skin XML file, both outline and solid fill backgrounds can be created with the <border> control.

Example from acorn_mill_vcp_skin.vcp

Solid Blue Fill background with Black outline around the Spindle Control Buttons.

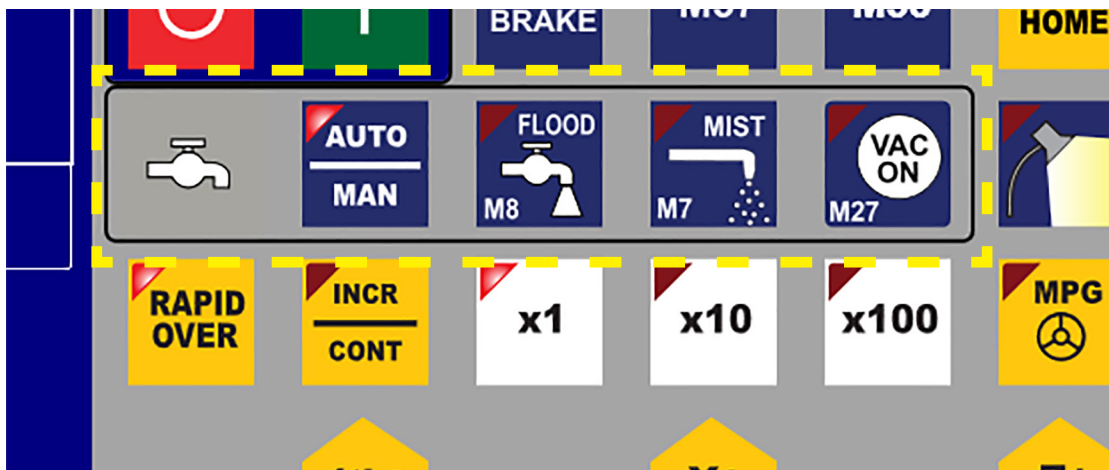
```
<border>
  <column_span>2</column_span>
  <column_start>1</column_start>
  <fill>#00007F</fill>
  <row_span>4</row_span>
  <row_start>1</row_start>
  <outline_color>#000000</outline_color>
  <outline_thickness>2</outline_thickness>
</border>
```



and another Example from acorn_mill_vcp_skin.vcp

Black outline around the Coolant controls.

```
<border>
  <column_span>5</column_span>
  <column_start>1</column_start>
  <fill>Transparent</fill>
  <row_span>1</row_span>
  <row_start>5</row_start>
  <outline_color>#000000</outline_color>
  <outline_thickness>2</outline_thickness>
</border>
```

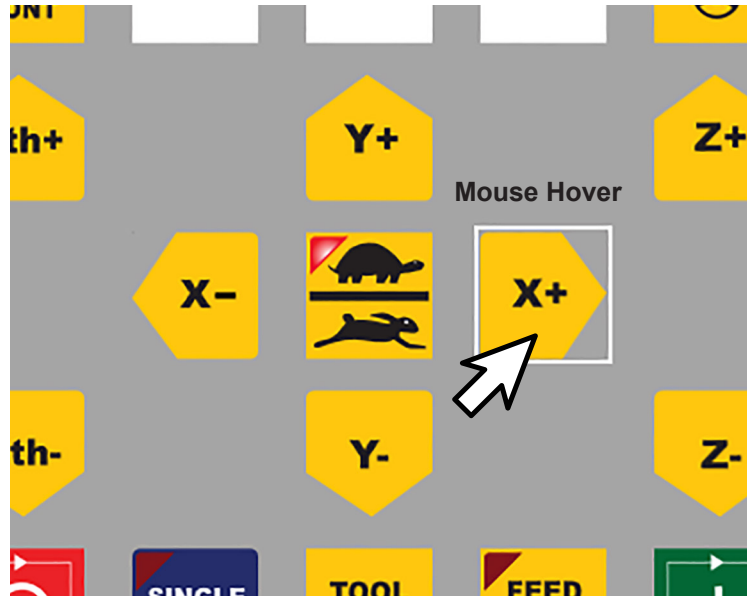


Button hover effect controls

When the mouse cursor is hovering over a button, the Button Mouse Hover effect give the user an indication and confirmation visually that the mouse cursor is indeed on the button location. The default VCP Button Hover Effect is a simple white outline around the button, it is clear concise and unmistakable and gives the VCP a nice interactive feel providing the user with reassuring feedback as the button about to be pressed is the right one. In the VCP Skin the Button Hover Effect is referred to as “on_hover”. The color and opacity can easily be changed and the effect can also be eliminated if desired by using the word “Transparent” for the color.

Example from acorn_mill_vcp_skin.vcp

```
<on_hover>  
    <opacity>100</opacity>  
    <outline_color>#ffffff</outline_color>  
</on_hover>
```

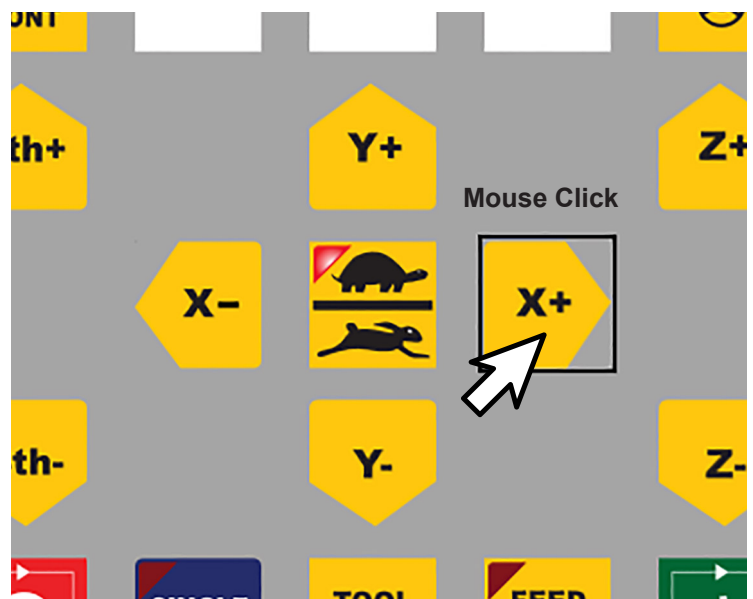


Button Click effect controls

Button click effects give the user an indication and confirmation visually of that a VCP button has been pressed. The default VCP Button Click/Finger Press effect is a black outline around the button. The opacity control also effect the button graphic and can be used to create a darkening effect on the button itself when clicked. This gives the VCP a nice interactive feel providing the user with reassuring feedback when the button is pressed. The Button Click Effect is referred to as “on_click”. The color and opacity can easily be changed and the effect can also be eliminated if desired by using the word “Transparent” for the color.

Example from acorn_mill_vcp_skin.vcp

```
<on_click>  
    <opacity>100</opacity>  
    <outline_color>#000000</outline_color>  
</on_click>
```



Mapping a macro to a VCP button.

16 Skin Events are used to create what are called Auxiliary Keys. (the words 'keys' and 'buttons' are used in this document interchangeably) These Auxiliary keys are mapped to a macro using the Wizard VCP Aux key configuration menu or CNC12 parameters. Use the 16 Aux key Skin Event numbers turn any button into a Auxiliary key that then in turn will run the assigned macro when that Auxiliary key is pressed/clicked on.

Virtual Control Panel Auxiliary Key Macro Assignments	
Default VCP Auxillary key assignments are set automatically. No need to modify this menu if you are using the default VCP Skin. This menu is provided for those that want to customize VCP Auxillary Key assignments to correspond to a custom VCP Skin. See the VCP Users Manual for more information on how to use this menu. *VCP User Manual Documentation (PDF)	
Auxillary Keys	M Function
Auxillary Key 1 / Skin Event #3	None
Auxillary Key 2 / Skin Event #4	None
Auxillary Key 3 / Skin Event #5	None
Auxillary Key 4 / Skin Event #8	None
Auxillary Key 5 / Skin Event #9	None
Auxillary Key 6 / Skin Event #10	mfunc49.mac
Auxillary Key 7 / Skin Event #13	mfunc50.mac
Auxillary Key 8 / Skin Event #14	mfunc60.mac
Auxillary Key 9 / Skin Event #15	None
Auxillary Key 10 / Skin Event #18	mfunc55.mac
Auxillary Key 11 / Skin Event #19	mfunc56.mac
Auxillary Key 12 / Skin Event #20	None
Auxillary Key 13 / Skin Event #24	None
Auxillary Key 14 / Skin Event #25	mfunc57.mac
Auxillary Key 15 / Skin Event #68	mfunc58.mac
Auxillary Key 16 / Skin Event #69	mfunc59.mac

There are several buttons on the stock Centroid VCP that are already defined as Auxiliary keys. On the Acorn Mill VCP they are M55,56,57,58, Rest Home, Park, Set Axis 0, and Set All 0. Any button that runs a macro is a Auxiliary key and is defined as such by using one of the skin event numbers above.

So, lets take a look at what a typical Auxiliary key #11 as an example. Opening the M56 button XML file contains only these three lines.

```
<vcp_button>
  <skin_event_num>19</skin_event_num>
</vcp_button>
```

The node <skin_event_num>19</skin_event_num> line tells the VCP that the M56 button is an Auxiliary key and when pressed will run the macro that is assigned to Skin Event .

Stock Acorn Mill VCP Auxiliary Key button macro assignments.

The Acorn Wizard menu allows the user to assign a macro program to an Auxiliary Key. (Note: Macros are typically used for more complex actions than just using a button to turn and output on or off. If you want to use a VCP button as a output on/off switch use anyone of the many preprogrammed outputs such as worklight on/off, vacuum on/off, laser on/off etc and their corresponding button folder in the VCP resources folder.) If you need more complex action from a button this is when a macro is use. There are several Auxiliary buttons that are preassigned with specific macros with the 'stock' Acorn VCP skins. These can be used as-is or modified.

If you would like make a Auxiliary button run a particular macro program often it is easiest to simply edit the existing macro that is already assigned to a particular Auxiliary button, these buttons are in place and predefined ready to use. For Example, the M55, M56, M57 and M58 buttons are simply VCP Auxiliary buttons that have been assigned to run the macros M55,M56,M57 and M58 respectively. So, no need to reassign these buttons to another macro, simply open the M55 macro file 'mfunc55.mac' with Notepad ++ and edit the macro to you liking and you are done. An examples of editing a preassigned Auxiliary key macro is on page 16 that shows editing both the macro and the button graphic of an Aux key.

If you have already customized the preassigned Auxiliary key macros M55,56,57 and 58 and need even more macros that corresponding to VCP buttons (or would like to reassign an auxiliary key to another macro for another reason) then read on.

Use the Acorn Wizard Preferences "VCP Aux Keys" menu to assign a macro to an Auxiliary button on the VCP. When a macro (mfuncXX.mac) is assigned to a Aux key then that button will run the macro when pressed/clicked on. The default Acorn Mill Assignments are shown below.

Virtual Control Panel Auxiliary Key Macro Assignments



































Default VCP Auxiliary key assignments are set automatically. No need to modify this menu if you are using the default VCP Skin. This menu is provided for those that want to customize VCP Auxiliary Key assignments to correspond to a custom VCP Skin. See the VCP Users Manual for more information on how to use this menu.
[*VCP User Manual Documentation \(PDF\)](#)

Auxiliary Keys	M Function
Auxiliary Key 1 / Skin Event #3	None
Auxiliary Key 2 / Skin Event #4	None
Auxiliary Key 3 / Skin Event #5	None
Auxiliary Key 4 / Skin Event #8	None
Auxiliary Key 5 / Skin Event #9	None
Auxiliary Key 6 / Skin Event #10	mfunc49.mac
Auxiliary Key 7 / Skin Event #13	mfunc50.mac
Auxiliary Key 8 / Skin Event #14	mfunc60.mac
Auxiliary Key 9 / Skin Event #15	None
Auxiliary Key 10 / Skin Event #18	mfunc55.mac
Auxiliary Key 11 / Skin Event #19	mfunc56.mac
Auxiliary Key 12 / Skin Event #20	None
Auxiliary Key 13 / Skin Event #24	None
Auxiliary Key 14 / Skin Event #25	mfunc57.mac
Auxiliary Key 15 / Skin Event #68	mfunc58.mac
Auxiliary Key 16 / Skin Event #69	mfunc59.mac

"None" means there is no macro assigned to the VCP Aux button. When none is used, most likely this Aux button is being used in conjunction with a different PLC Skin Event (a non macro related PLC function such as turning an output ON or OFF) or it is not in use at all. Keep in mind the Skin Event number determines whether a button is a Aux key or not, Not the position of the button!

Acorn Mill VCP Auxiliary Key button Default locations.

The VCP buttons are on a grid. This grid is used to identify the location of the buttons on the VCP layout skin. VCP Auxiliary keys 1 through 16 are will be located on the VCP grid beginning with Row 1, Column 3 through Row 4 Column 6. While all button locations on the VCP can be modified in both look and function, these 16 Auxiliary keys have been designed to work in conjunction with a macro program assignment via the Wizard and CNC12 parameters. In other words if you want a VCP button to run a macro use one of these 16 Auxiliary keys and its related Skin Event Number. It is also interesting to note that the Auxiliary keys can be moved to any other location on the entire VCP grid. Said another way, the location doesn't define the Aux key, the Skin Event Number in the Button XML and file does. These 16 Aux keys are placed in this location as a default position, use them in there original position or move them where you want.

			Column 3	Column 4	Column 5	Column 6
Row 1						
Row 2						
Row 3						
Row 4						
						
						

Move an Auxiliary Key to another location.

Moving an Aux button to a new location is just like moving any other button. Edit the VCP Skin as described on page 5.

Can you move and Auxiliary key outside of the stock 16 Auxiliary keys area?

Answer: Yes, Simply move any Aux button to the desired location on the VCP. Aux keys are not fixed in location. While it might appear that the Auxiliary buttons 1-16 are fixed in their default locations (Row 1, Column 3 through Row 4 Column 6) this is not true. These are just the default positions. You can move an Auxiliary button to any other button location on the VCP. When you move a button as shown on page 5 the function of the button goes along with it.

Hijack an existing Auxiliary key and change its macro and graphic and then move it.

So for example; if you would like a macro to run with a button location, lets say for instance in the location of the spindle CCW button, simply delete the Spindle CCW button and move the Auxiliary button to that location. Lets run through that as an example.

A spindle warm up macro is a common function on a CNC Router. Lets edit Aux key 11, both the graphics and the macro assigned to Aux 11. (in the Acorn Mill VCP, Aux 11 has been preassigned to the macro M56) We will edit the M56 (mfunc56.mac) macro to warm up the spindle automatically and change the button graphic for Aux 11.

1.) Create a Spindle Warm up graphic and replace the stock M56 graphic with it. Open the Button folder M56, make a backup copy of the M56.SVG file. Copy the new Router Spindle Warm Up graphic into the M55 button folder and rename it m56.SVG. Restart CNC12 and the VCP Aux 11 key will now look like this:

Default M56 Graphic on a typical Paired Axes Router VCP



Spindle Warm Graphic renamed to "m56.SVG"



2.) Now edit the macro file for M56: mfunc56.mac and insert the Spindle Warm Up commands. Here is a typical Spindle Warm Up Macro, edit it to suit your application.

```
N100                ;Insert your code between N100 and N1000
#101 = 5
M225 #101 "***Beginning spindle warm-up***"
M25
M3 S6000
G4 P180
S8000
G4 P120
S10000
G4 P120
S12000
G4 P120
S14000
G4 P120
M5
M225 #100 "***Spindle warm-up finished, now go cut something!***"
N1000              ;End of Macro
```

Hijack an existing Auxiliary key and change its macro and graphic and move it.

3.) Now everything is working at this point, when you press the Spindle Warm Up Button the Spindle Warm Up macro runs and all is good. But, we want the Spindle Warm up button to take the place of the Spindle CCW button on the VCP. Easy! Delete the Spindle CCW button from the VCP skin and move the Spindle Warm Up Button to that location.

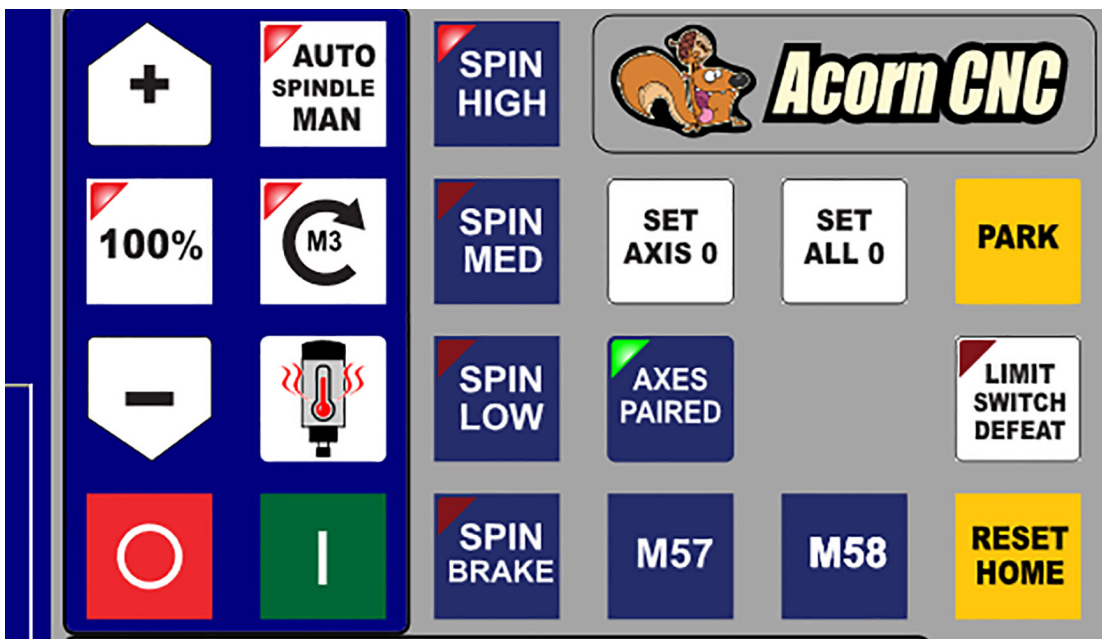
Edit the VCP skin and delete the spindle_ccw button line

```
<button row="2" column="2">spindle_cw</button>
<button row="2" column="3">spin_med</button>
<button row="2" column="6">park</button>
<button row="3" column="1">spindle_minus</button>
<button row="3" column="2">spindle_ccw</button> Delete this line to remove the spindle_ccw button
<button row="3" column="3">spin_low</button>
<button row="3" column="4">axes_paired</button>
<button row="3" column="5">m56</button>
<button row="3" column="6">limit_switch_defeat</button>
<button row="4" column="1">spindle_cancel</button>
<button row="4" column="2">spindle_start</button>
<button row="4" column="3">spin_brake</button>
<button row="4" column="4">m57</button>
<button row="4" column="5">m58</button>
```

Move the Spindle Warm Up Button (remember we hijacked the M56 button for this use) to the old spindle_ccw location

```
<button row="2" column="2">spindle_cw</button>
<button row="2" column="3">spin_med</button>
<button row="2" column="6">park</button>
<button row="3" column="1">spindle_minus</button>
<button row="3" column="3">spin_low</button>
<button row="3" column="4">axes_paired</button>
<button row="3" column="2">m56</button> change the M56 button position from column 5 to column 2
<button row="3" column="6">limit_switch_defeat</button>
<button row="4" column="1">spindle_cancel</button>
<button row="4" column="2">spindle_start</button>
<button row="4" column="3">spin_brake</button>
<button row="4" column="4">m57</button>
<button row="4" column="5">m58</button>
```

Save the changes and restart CNC12 and this is the result.



The purpose of this example was to demonstrate how to edit and existing Auxiliary Key for a new purpose and move it to a new location. The process described above is just one way of doing things for instance you could have just as easily made a copy of the entire M56 button folder and renamed the copied M56 folder and the files contained within to “spin_warm”, “spin_warm.xml”, “spin_warm.svg” effectively creating a copy of M56 / Aux 11 key with a new name. Then simply edit the VCP skin with the new name in place of ‘spindle_ccw’ like so:

```
<button row="2" column="2">spindle_cw</button>
<button row="2" column="3">spin_med</button>
<button row="2" column="6">park</button>
<button row="3" column="1">spindle_minus</button>
<button row="3" column="2">spin_warm</button> rename 'spindle_ccw' to 'spin_warm'
<button row="3" column="3">spin_low</button>
<button row="3" column="4">axes_paired</button>
<button row="3" column="4">m56</button> Delete this line to remove the M56 button
<button row="3" column="6">limit_switch_defeat</button>
<button row="4" column="1">spindle_cancel</button>
<button row="4" column="2">spindle_start</button>
<button row="4" column="3">spin_brake</button>
<button row="4" column="4">m57</button>
<button row="4" column="5">m58</button>
```

And then edit the M56 macro (mfunc56.mac) and the Spindle Warm Graphics file spin_warm.svg to your desire look and functionality.

<run> a new way to directly run a Macro with any VCP button was introduced in v5.08

In addition to the skin event method described above,
a new way to run a macro directly from any VCP button was introduced: "<run>"

<run> allows a VCP button run either a single line of G-code or a Macro immediately,
with or without the need for a cycle start button press.

Insert these commands directly into the button .xml file

To run a line of g-code with a VCP button specify <line> and type the line:

```
<run>  
    <line>G0 X0 Y0</line>  
</run>
```

To run a macro directly with a VCP button, Specify <macro> and the path and filename:

```
<run>  
    <macro>C:\cncm\ncfiles\myMacro.cnc</macro>  
</run>
```

Notes:

- a.) These types of VCP buttons will only work when being pressed from the main menu of cnc12.
- b.) Both a macro and a g code line cannot be run at the same time.

Example of what not to do:

```
<run>  
    <line>G0 X0 Y0</line>  
    <line>G1 X1 Y1</line>  
</run>
```

In the example below, after pressing the VCP button the first line (G0X0Y0) would be run and the second line would be ignored.

In this case, the two lines should be put into a macro and the macro option used.

Like so:

```
<run>  
    <macro>C:\cncm\ncfiles\myMacro.cnc</macro>  
</run>
```

Where "myMacro.cnc" contains:

```
G0 X0 Y0  
G1 X1 Y1
```

<run> a new way to directly run a Macro with any VCP button was introduced in v5.08

Another example:

What not to do: Don't mix together <line> and <macro>

```
<run>  
  <line>G0 X0 Y0</line>  
  <macro>C:\cncm\ncfiles\myMacro.cnc</macro>  
</run>
```

When pressing the button the first line (G0X0Y0) would be ignored completely and the macro would be run instead.

In this case, the two lines should be put into a macro and the macro option used.

Like so:

```
<run>  
  <macro>C:\cncm\ncfiles\myMacro.cnc</macro>  
</run>
```

Create VCP buttons larger than the stock button sizes.

Sometimes it is desired to have a button on the VCP be larger in size than the stock square buttons. It could be a button with an important feature that is used frequently in a special application or as simply as making a button large and easy to read and use.

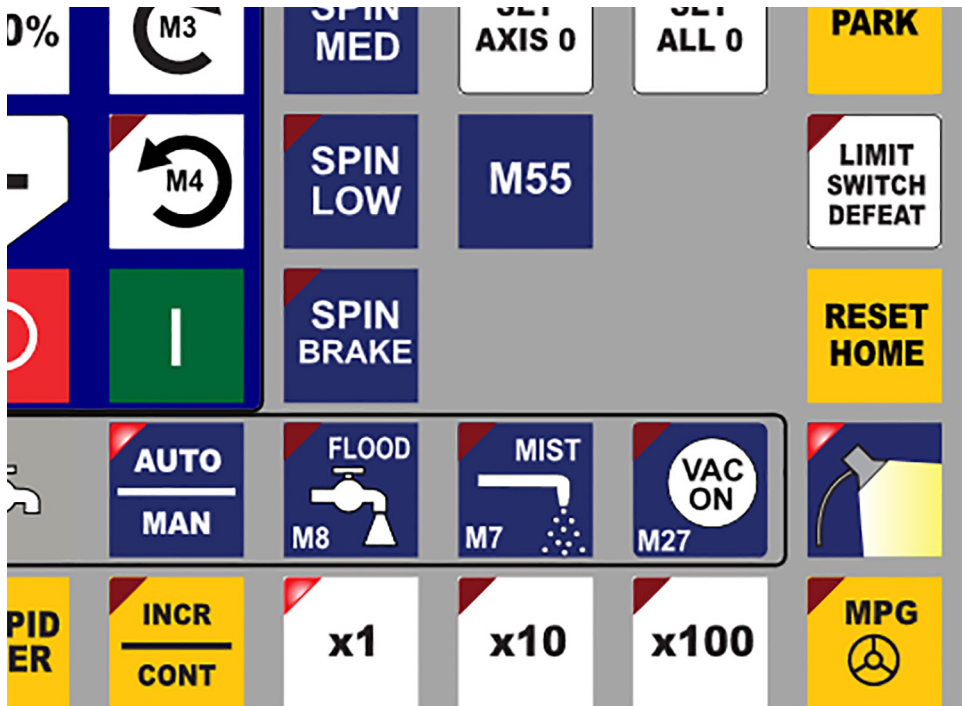
Similar to how the Logos and Icon position and size is controlled on the VCP grid so can the individual buttons. row span and column span can be use to let the VCP know to scale the button graphic larger that just one button space.

I'll demonstrate with a simple example. Lets make the M55 button two times as wide and two times as high as it currently is.

Edit acorn_skin.vcp. Make room for the new larger M55 button and delete the M56, M57, M58 buttons

```
<button row="3" column="1">spindle_minus</button>
<button row="3" column="2">spindle_ccw</button>
<button row="3" column="3">spin_low</button>
<button row="3" column="4">m55</button>
<button row="3" column="5">m56</button>
<button row="3" column="6">limit_switch_defeat</button>
<button row="4" column="1">spindle_cancel</button>
<button row="4" column="2">spindle_start</button>
<button row="4" column="3">spin_brake</button>
<button row="4" column="4">m57</button>
<button row="4" column="5">m58</button>
<button row="4" column="6">reset_home</button>
<button row="5" column="2">coolant_auto_man</button>
<button row="5" column="3">flood_coolant</button>
<button row="5" column="4">mist_coolant</button>
```

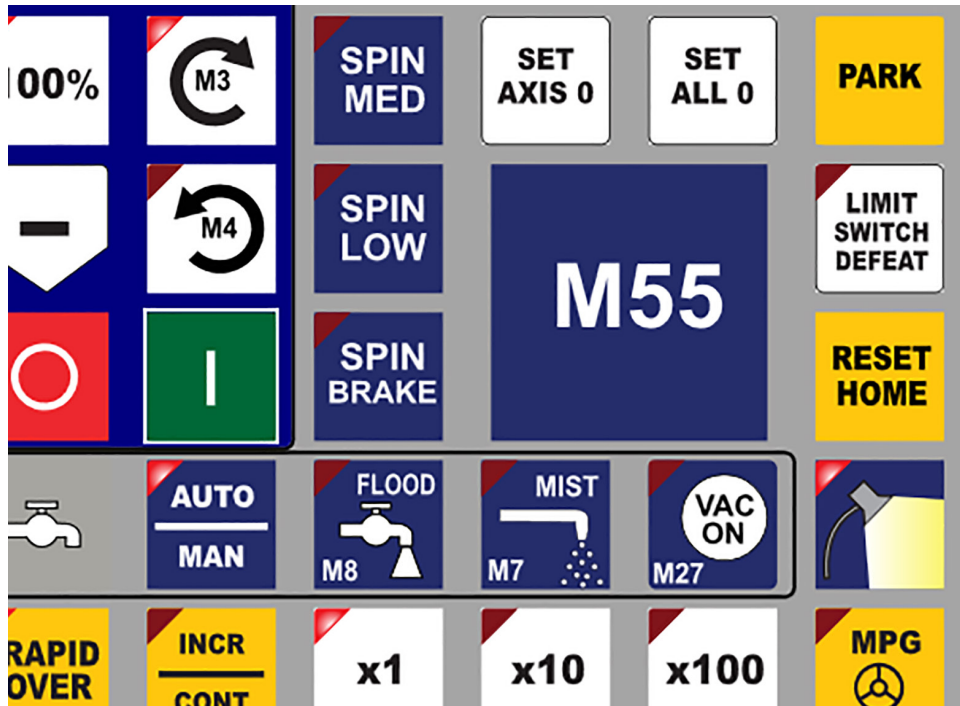
Save and this is the result.



Edit the acorn_mill_vcp_skin.vcp xml file and add to the m55 line the row span and column span as seen below.

```
<button row="3" column="1">spindle_minus</button>
<button row="3" column="2">spindle_ccw</button>
<button row="3" column="3">spin_low</button>
<button row="3" column="4" row_span="2" column_span="2">m55</button>
<button row="3" column="6">limit_switch_defeat</button>
```

and the results looks like this.



VCP button as a indicator light for a memory location

Sometimes it is useful to have a VCP button act as an indicator light. A light that comes on when an memory location is active. This effect can be achieve by swapping the Button image from one to another based on the state of the memory location. The button XML tag is: `<plc_memory>` and is used in the button xml file like this.

```
<vcp_button>
  <plc_memory>           "use this tag to let the VCP know we are linking this button to memory location"
    <number>7</number>    "use this tag to let the VCP know which memory location to watch"
    <image_on>mem_on.svg</image_on> "use this tag to let the VCP know which images to use"
    <image_off>mem_off.svg</image_off>
  </plc_memory>
</vcp_button>
```

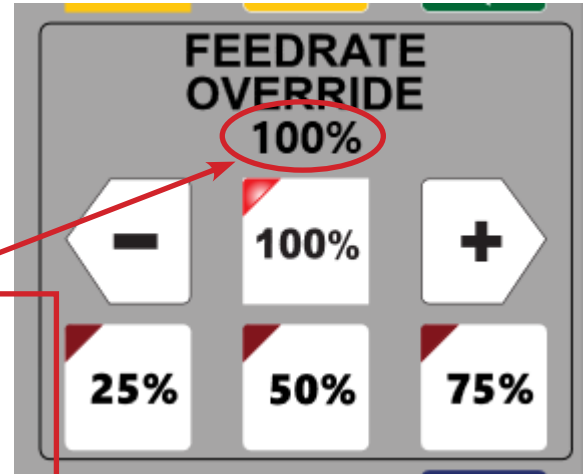
Display CNC Data Gathered from the PLC program on the VCP: “PLC Words”

New for CNC12 VCP v5.0 +: The VCP now supports a new feature called “PLC Words”. PLC Words allows the VCP to display CNC data on the VCP in real time gathered from the CNC’s PLC program.

This new technique can be used for a wide variety of applications to display useful information in real time on the VCP. The first use of the new “PLC words” feature was to display the Feedrate Override percentage value on the VCP. (Note: This was previously hard coded in Acorn VCP 4.82 and earlier and Oak/Allin1DC VCP 4.22 and earlier). So, we will use the new Feedrate Override number display as an example of how to use a PLC word to display a value on the VCP.

Below is a sample of the stock Acorn Mill VCP skin XML code using PLC words to display in real time on the VCP the current Feedrate Override Percentage value.

```
<border>
  <column_span>3</column_span>
  <column_start>4</column_start>
  <fill>Transparent</fill>
  <row_span>1</row_span>
  <row_start>11</row_start>
  <outline_color>Transparent</outline_color>
  <outline_thickness>1</outline_thickness>
  <plc_word>
    <number>31</number>
    <color>#000000</color>
    <fontsize>22</fontsize>
    <font>Sergio UI</font>
    <fontstyle>bold</fontstyle>
    <verticalalignment>bottom</verticalalignment>
    <horizontalalignment>center</horizontalalignment>
    <marginbottom>-5</marginbottom>
    <percentage>true</percentage>
  </plc_word>
</border>
<image>
  <column_span>1</column_span>
  <column_start>1</column_start>
  <row_span>1</row_span>
  <row_start>5</row_start>
  <path>C:\cncm\resources\vcp\images\coolant.svg</path>
</image>
```



This new feature allows users to display PLC Words directly on top of any element of the VCP. PLC Words can be displayed on top of any VCP button or on the VCP borders.

To display a PLC word add the plc_word node within a border node (as seen above) or a vcp_button node.

The plc_word has several control settings that the user can use to style and place the PLC word. Color, Font, Font Style, alignment, margin and even a percentage sign.

In order for the VCP to display a PLC word value use the number node `<number>xx</number>` to identify which PLC word to display.

In the example above `<number>31</number>`.

These PLC Word numbers are found in the PLC program itself.

Many PLC words are predefined in the Centroid provided and Wizard Generated PLC programs. They can be found under the Word Definitions section of the PLC program source file. cncm\XXXX.src which is a text file, open it with **Notepad ++** https://www.centroidcnc.com/dealersupport/tech_bulletins/uploads/294.pdf

There is a number assigned to each of the PLC values.
For example "W31" is Word #31 which equals the current Feedrate Override Percentage value.

Commonly used stock PLC Word values are:

Spindle Speed Override %	(SpinOverride_W is W19)
Feedrate Override %	(FinalFeedOverride_W is W31)
Target Voltage Override %	(TargetVoltage_W is W7)
Current Carousel Position	(CurrentCarouselPosition_W is W54)
Current Turret Position	(CurrentTurretPosition_W is W52)

Below is an excerpt from a typical Acorn Mill PLC program source file showing the stock PLC Word Definitions within that PLC program. (cncm/acorn_mill_plc.src)

```

;-----
;                               Word Definitions
;-----
SkinFeedOverride_W1           IS W1
TotalOffsetAppliedFW          IS W2
JogIncMultiplier              IS W3
ZJogOffset                    IS W4
ZAxisEncoder                  IS W5
TorchVoltage_W                IS W6
TargetVoltage_W               IS W7
Last_TorchKnob_W              IS W8

ShuttleAxisSelect_W           IS W10
ShuttleResumeFeedOver_W       IS W11
SkinningInt12_W               IS W12
TwelveBitSpeed_W              IS W13
FaultMsg_W                    IS W14
ErrorMsg_W                    IS W15
InfoMsg_W                     IS W16

SpinOverride_W                 IS W19
Mem_W                          IS W20
Out_W                          IS W21
MpgOffsetWord                  IS W23

SpindleRange_W                 IS W24 ; 1 = low ... 4 = high
PrevFeedOverride_W             IS W25

UsbAxisMonitor_W               IS W26
UsbScaleMonitor_W              IS W27

AtcType_W                      IS W28
KbOverride_W                   IS W29
FeedrateKnob_W                 IS W30
FinalFeedOverride_W            IS W31
Last_FeedrateKnob_W            IS W32
LastKbOverride_W               IS W33
LastSkinFeedOverride_W         IS W34
UsbJog_W                       IS W35

```

MiniPLCStatus_W	IS W36
JogKeyCfg_W	IS W37
UsbButtonMonitor	IS W38
UsbWheelCurrent	IS W39
UsbWheelLast	IS W40
UsbWheelDelta	IS W41
UsbMpgActiveAxes_W	IS W42
DefaultJogging_W	IS W43
LastCarouselDir_W	IS W49
PutBackPosition_W	IS W50
CurrentTurretPosition_W	IS W52
RequestedTurretPosition_W	IS W53
CurrentCarouselPosition_W	IS W54
RequestedCarouselPosition_W	IS W55
CycloneStatus_W	IS W56
THCEncoderStatus_W	IS W9;57
PLC_Fault_W	IS W61
PLCFaultAddr_W	IS W62
BadMsg_W	IS W63
Lube_W	IS W65
P148Value_W	IS W66
P146Value_W	IS W67
P170Value_W	IS W68
LubeAccumTime_W	IS W69
LubeM_W	IS W70
LubeS_W	IS W71
P985Value_W	IS W72
P415Value_W	IS W73
Inputs1_16_W	IS W74
Inputs33_48_W	IS W75
Inputs49_64_W	IS W76
Inputs65_80_W	IS W77
;Inputs81_96_W	IS W78
;Inputs97_112_W	IS W79
InvertedInputs911_W	IS W80
InvertedInputs913_W	IS W81
InvertedInputs914_W	IS W82
InvertedInputs915_W	IS W83
;InvertedInputs9??1_W	IS W84
;InvertedInputs9??2_W	IS W85
Limit_Invert_W	IS W86
ETHER1616_ALIVE_W	IS W87
P419Value_W	IS W88
P523Value_W	IS W89

Users can also create their own Word Values.

Below is an example of: How to Display a Lathe Turret Position on the VCP using PLC Words.

Lets create a PLC Word value that will “track” a 4 Tool Turret Location and then display Turret Position word value on the VCP to track turret location.

We will use a Typical Lathe Turret which has 4 separate inputs for each position. First we must define our word value in the PLC program and add the logic for that word value. Edit the PLC program to add the new word value 52, add the logic to feed the data to #52 (CurrentTurretPosition_W) and compile the PLC program.

Find the Word Definition section of the PLC program source file and add in the turret word definitions. Add the PLC word definition for the Turret Position: CurrentTurretPosition_W IS W52

```
-----  
;  
; Word Definitions  
;  
-----  
CurrentTurretPosition_W IS W52
```

Next we will add the logic that will assign values to our new word value (W52). Add the following code to set the Word value from 1 to 4 based on the Inputs.

```
IF TurretInput1 THEN CurrentTurretPosition_W = 1  
IF TurretInput2 THEN CurrentTurretPosition_W = 2  
IF TurretInput3 THEN CurrentTurretPosition_W = 3  
IF TurretInput4 THEN CurrentTurretPosition_W = 4
```

Complie the PLC program. https://www.centroidcnc.com/centroid_diy/downloads/centroid_plc_programming_manual.pdf
videos. https://youtube.com/playlist?list=PLXhs2C5No0_gFS_RmKNo7hii2WKledQIQ

Now when Input 2 is ON, the word value will be set to 2.

To see the turret current tool # dispaly on top of a VCP button

Add the PLC Word value 52 to the VCP button XML file of the button that we want this value to be displayed on top of

```
<plc_word>  
  <number> 52</number>  
</plc_word>
```

Save all files, shut down and restart CNC12.

Now when the tool turret changes tools, the current turret position will be displayed on that VCP button.

This is a basic example, however any value in the PLC when assigned a Word value can be displayed on the VCP.

More complicated applications for instance would be: displaying an analog input from a pressure gauge, displaying calculated math done by the PLC, or even displaying system variables.

The PLC word “type” node sets what type of plc word the user wants displayed. This is defaults to Interger type if no other types are set. Types Are:

- Int (Int = Interger such as 10)
- Float (Float – Floating Point vaule such as 10.1234). valid values are in the range of -2147483648 to 2147483647
- Also available but typically not used: Double (64 bit) and DoubleFloat (64 bit) which doubles the range values (-9223372036854775808 to 9223372036854775807)

The significant node is for all plc word types except Int. This tells the VCP how many digits after the decimal point a user wants displayed. This is defaulted to 2 if the significant node is not used.

See example below of PLC word set to Interger value and 5 decimal places.

```
<plc_word>
  <type>float</type>
  <significant>5</significant>
  <number>31</number>
  <color>#000000</color>
  <fontsize>22</fontsize>
    <font>Sergio UI</font>
    <fontstyle>bold</fontstyle>
    <verticalalignment>bottom</verticalalignment>
    <horizontalalignment>center</horizontalalignment>
    <marginbottom>-5</marginbottom>
    <percentage>true</percentage>
</plc_word>
```



PLC word Color

The color node sets the color of the text displayed. Color is specified in hex code as seen in the example. If no color is spiecified then the default color is black: #000000.

PLC word Font

The font node allows the user to set the font of the text. This is defaulted to Segoe UI. The list allowed fonts: <https://learn.microsoft.com/en-us/typography/font-list/>. Note: Users must put the font name in exactly as it is displayed in the list, spaces and all.

PLC word Font Size

The font size node sets the font size for the text. Font size is an integer value. If no size is specified the default font size is 16

PLC word Font Style

The font style node sets the style that the text is displayed in. This is defaulted to normal. The user can also set it to be bold, italics, normal or oblique.

PLC word Vertical Alignment

The vertical alignment node sets the vertical alignment of the text displayed. The default is bottom. Vertical alignment can be set it to be top, center, or bottom.

PLC word Horizontal Alignment

The horizontal alignment node sets the horizontal alignment of the text. If not alignment is given, default is left. Horizontal alignment can set it to be left, right, center.

PLC word Margin

The margin bottom, top, left, and right nodes sets the margins on the text. This allows fine tuning where the text is displayed. This must be an integer and it is defaulted to 0.

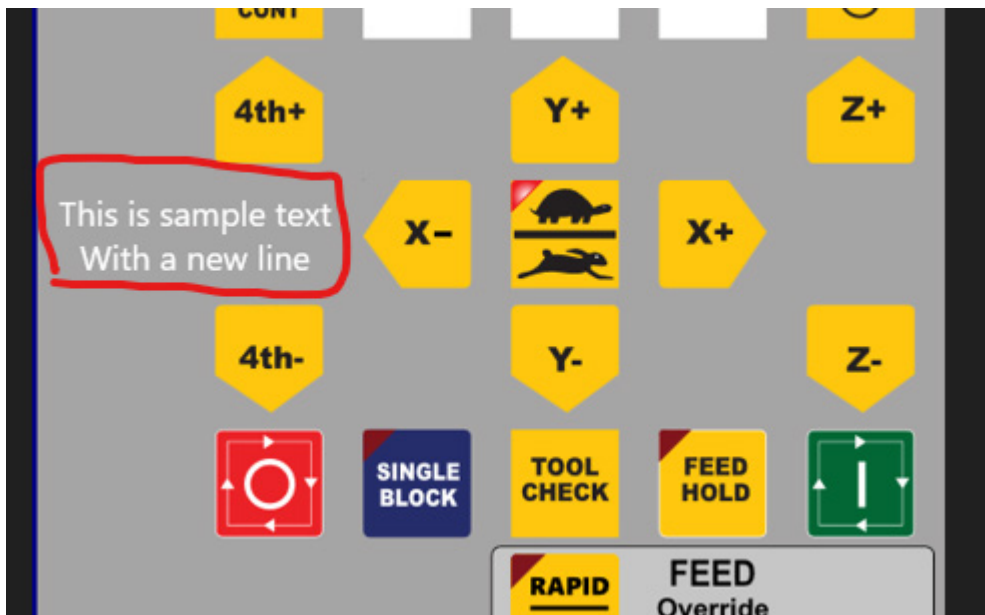
PLC percentage sign

Adds a percentage sign to the end of the word value. True = display the % sign.

<text> a way to directly display text onto the VCP WITHOUT the need of an image.

For cases where it is desirable to display text on the VCP when not using a image or button svg <text> is used in the VCP skin (.vcp) xml file.

```
<text>
  <content>This is sample text&#13;With a new line</content>
  <fontsize>20</fontsize>
  <color>#ffffff</color>
  <font>Segoe UI</font>
  <horizontalalignment>center</horizontalalignment>
  <verticalalignment>center</verticalalignment>
</text>
```



Example code from the skin.vcp file for the image above.

```
<border>
  <column_span>2</column_span>
  <column_start>1</column_start>
  <fill>Transparent</fill>
  <row_span>1</row_span>
  <row_start>8</row_start>
  <text>
    <content>This is sample text&#13;With a new line</content>
    <fontsize>20</fontsize>
    <color>#ffffff</color>
    <font>Segoe UI</font>
    <horizontalalignment>center</horizontalalignment>
    <verticalalignment>center</verticalalignment>
  </text>
</border>
```

(how many columns wide?)
(which column to start on?)
(any background color?)
(how many rows?)
(which row to start on?)
(start of the text definition)
(the text to display)
(fontsize)
(color)
(font)
(horizontal alignment within the space)
(vertical alignment withing the space)

Notes:

- "" is a Carriage Return (adds a new line)
- This feature functions in a similar fashion to the VCP feature called plc_word.

Switching Functionality: Buttons, Borders, and Images can be swapped out when a VCP button is pressed.

This feature gives the ability for the VCP to have different buttons, borders, and images defined to occupy the same space but only appear when we want.

To use the switching function each button, border, and image (referred to collectively henceforth as an 'object') requires a group to be defined in the .vcp xml file. This group can then be called on when we want it to be displayed.

Groups for borders and images that we intend to display together or swap are defined via a child node, example:

```
<border>
  <group>group_name</group>          (this border is part of this group name)
</border>
```

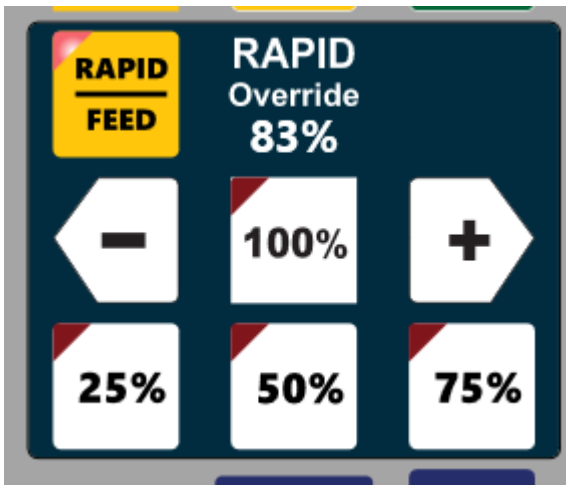
```
<image>
  <group>group_name</group>          (this image is part of this group name)
</image>
```

Groups for buttons are defined via an attribute

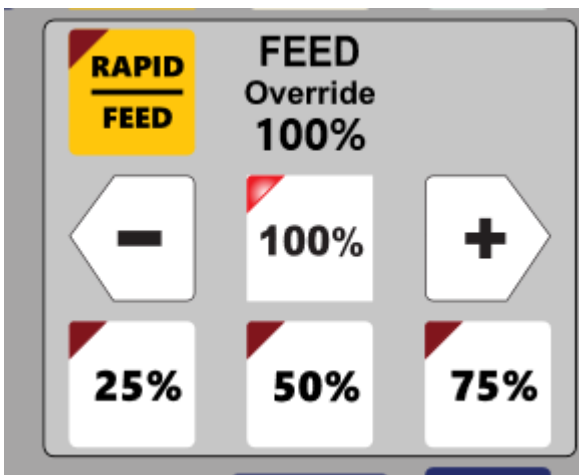
```
<button row="Y" column="X" group="group_name"></button>          (this button is part of this group name)
```

Any border, button or image can be defined as a group by simply adding the child node into the button/border/image node in the skin.VCP xml file. For example a common use of the switching function is the Rapid to Feedrate control swap.

Rapid Override ON = Blue background for Override Controls and Swap to Rapid Override % display

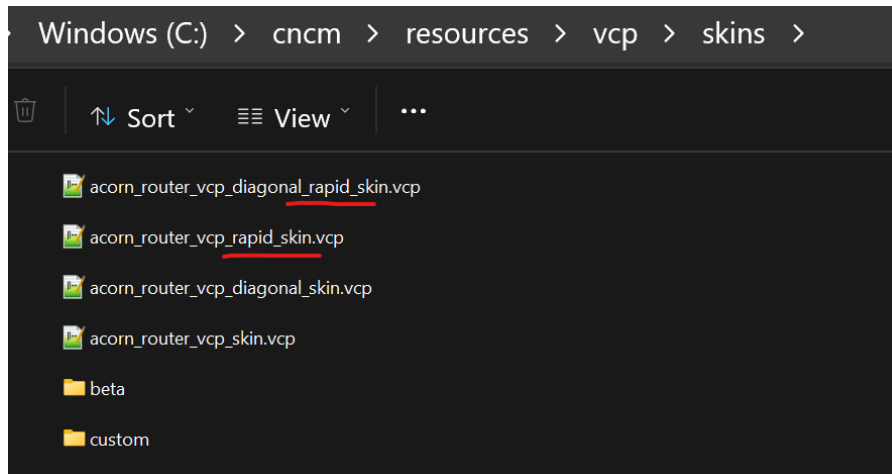


Rapid Override OFF = Grey background for Override Controls and Swap to Feedrate Override % display

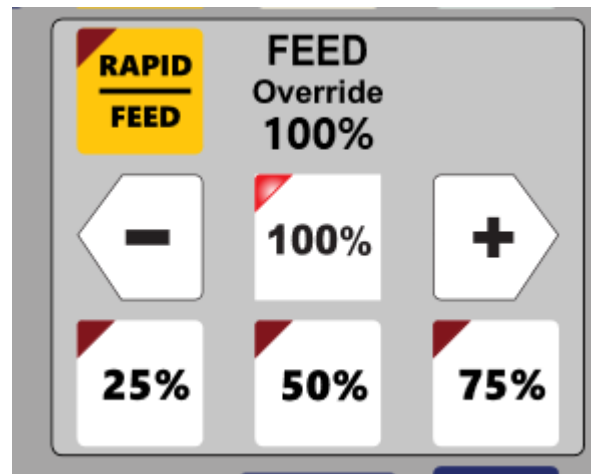
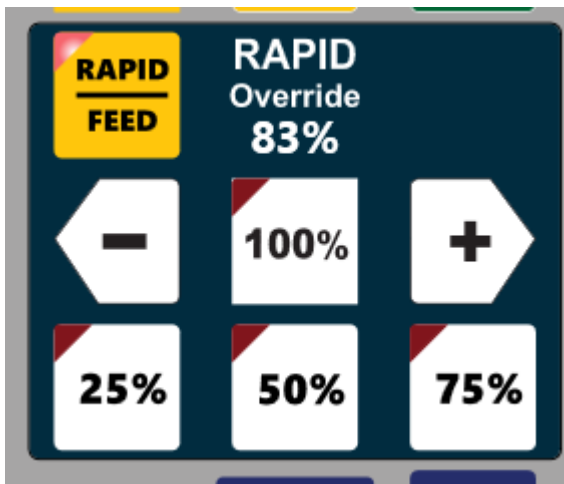


Switching Functionality: Buttons, Borders, and Images can be swapped out when a VCP button is pressed.

A great example of this feature in action is contained within the Centroid stock skins that contain the word “rapid” in them.



These rapid skins swap the background image, override text and PLC word depending on the Rapid/Feed selection being made with the Rapid Feed toggle button.



The sample code below is from acorn_router_vcp_skin.vcp showing how to add a border to a group.

```
<border>
  <column_span>3</column_span>
  <column_start>4</column_start>
  <fill>#012c3f</fill>
  <row_span>3</row_span>
  <row_start>11</row_start>
  <outline_color>#000000</outline_color>
  <outline_thickness>1</outline_thickness>
  <group>rapid_group</group> (add this border to the group named rapid_group)
</border>
```

Adding this child node within <border> and </border> adds this particular border to a group called “rapid_group” this group can then be switched on/off along with the button press.

Switching Functionality: Buttons, Borders, and Images can be swapped out when a VCP button is pressed.

Multiple borders, images and buttons can be added to any given group.

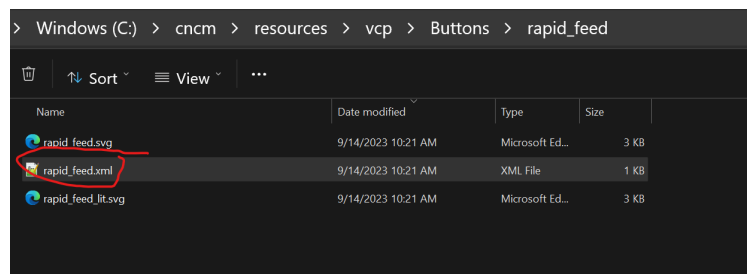
acorn_router_vcp_skin.vcp has examples of this as well, below you see the border for rapid override and the % value of the rapid override being included in the group call “rapid_group”.

```
<border>
  <column_span>3</column_span>
  <column_start>4</column_start>
  <fill>#012c3f</fill>
  <row_span>3</row_span>
  <row_start>11</row_start>
  <outline_color>#000000</outline_color>
  <outline_thickness>1</outline_thickness>
  <group>rapid_group</group>
</border>
<border>
  <column_span>3</column_span>
  <column_start>4</column_start>
  <fill>Transparent</fill>
  <row_span>1</row_span>
  <row_start>11</row_start>
  <outline_color>Transparent</outline_color>
  <outline_thickness>1</outline_thickness>
  <plc_word>
    <number>58</number>
    <color>#ffffff</color>
    <fontsize>22</fontsize>
    <font>Sergio UI</font>
    <fontstyle>bold</fontstyle>
    <verticalalignment>bottom</verticalalignment>
    <horizontalalignment>center</horizontalalignment>
    <marginbottom>-5</marginbottom>
    <percentage>>true</percentage>
  </plc_word>
  <group>rapid_group</group>
</border>
```

Once groups are defined in the .VCP skin. What to do with those groups is defined in the button .XML itself.

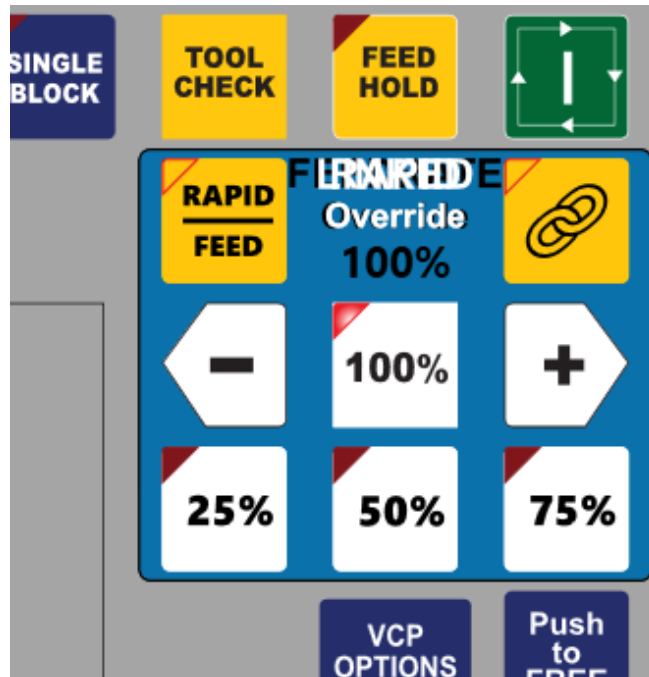
Example code from the new button called “rapid_feed.xml” file that is using this feature:

```
<vcp_button>
  <switch>
    <switch_on>
      <remove>feed_group</remove>
      <remove>feed_button_group</remove>
      <add>rapid_group</add>
      <image_on>rapid_feed_lit.svg</image_on>
    </switch_on>
    <switch_off>
      <remove>rapid_group</remove>
      <add>feed_group</add>
      <add>feed_button_group</add>
      <image_off>rapid_feed.svg</image_off>
    </switch_off>
    <remove>linked_group</remove>
    <add>unlinked_group</add>
  </switch>
</vcp_button>
```



Switching Functionality: Buttons, Borders, and Images can be swapped out when a VCP button is pressed.

For VCP start up the <hide_group> is often used: This node is inserted into the skin (.vcp file) to hide all of the groups that are unwanted at start up. If this is not done, all objects will be visible at VCP startup resulting in a mess as seen below.



In our example file `acorn_router_vcp_rapid_skin.vcp` you can find an example of hiding the rapid group on start up beginning on line 121.

```
117     <opacity>100</opacity>
118     <outline_color>#ffffff</outline_color>
119     </on_hover>
120
121     <hide_group>
122         <group>rapid_group</group>
123     </hide_group>
124
125     <button row="1" column="1">spindle_plus</button>
126     <button row="1" column="2">spindle_auto_man</button>
```

The rapid group is being hidden on start up since the VCP defaults to Feedrate Override control being displayed on start-up so, we don't want the Rapid Override value, border or background color to be displayed on startup therefore we used `<hide_group>`.

Convention to hide on start up is simply:

```
<hideGroup>
    <group>group_name_1</group>
    <group>group_name_2</group>
</hideGroup>
```

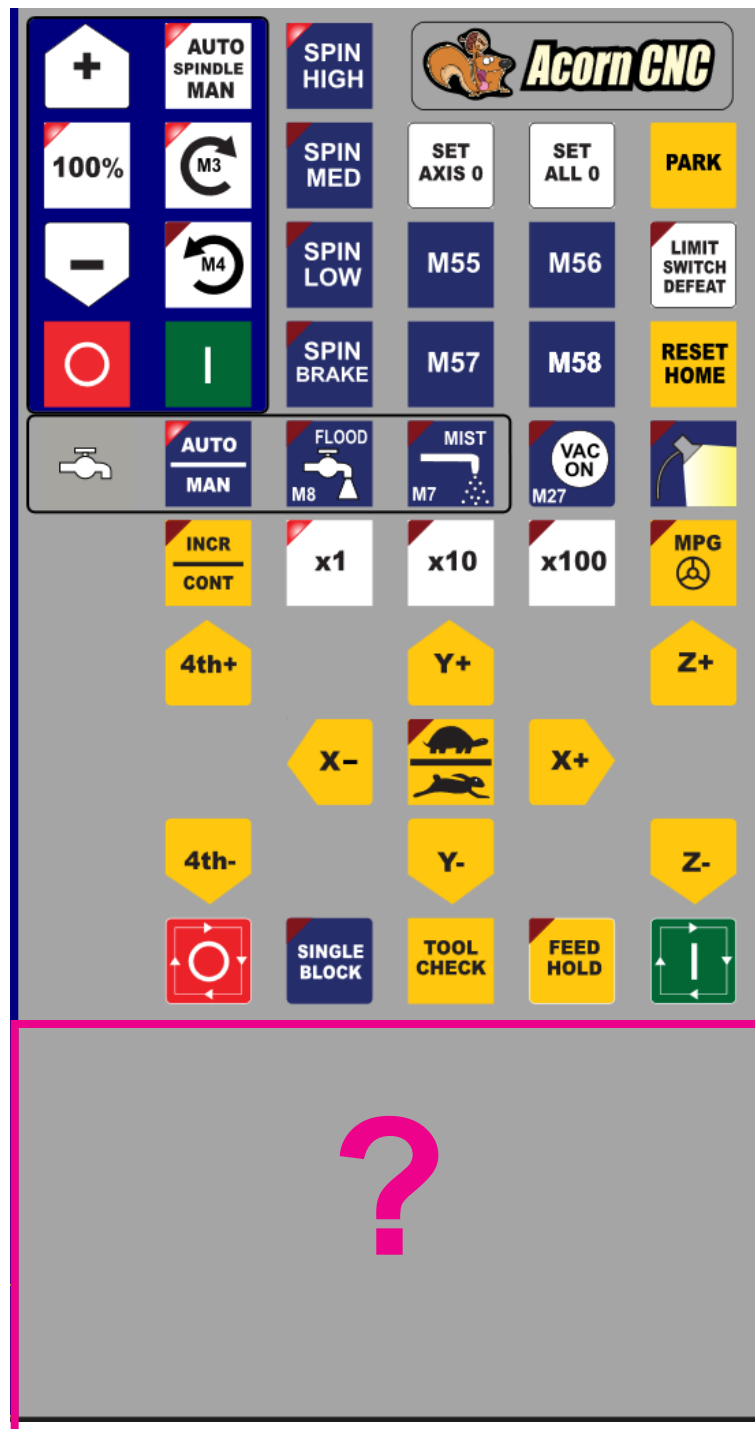
Note: The `hideGroup` node has no limit on how many groups can be hidden. While there is no limit on how many objects can be added to the VCP, it is important to note that performance may suffer as more objects are added and we estimate that decreased performance would start to become visible at approximately 1000 objects.

Trouble Shooting

1.) Missing lower third of the VCP when installing a new version of CNC12. This is caused by using the CNC12 utility feature “restore report” and using a report.zip file from an older version of CNC12. Restore Report works within a given version number of CNC12. For instance, if you are building a new CNC PC for a machine that has an existing CNC PC you can use the ‘report.zip’ to automatically copy all the settings from the CNC12 installation on the old computer to the new computer with the SAME version of CNC12 installed.

See the CNC12 installation instructions found on the CNC12 downloads page for more details.
https://www.centroidcnc.com/centroid_diy/centroid_cnc_software_downloads.html

and video discussion about copying old VCP's to new version of CNC12.
<https://youtu.be/MKzClzk8WEs?si=zFSjHv0XVoC3CBDs>



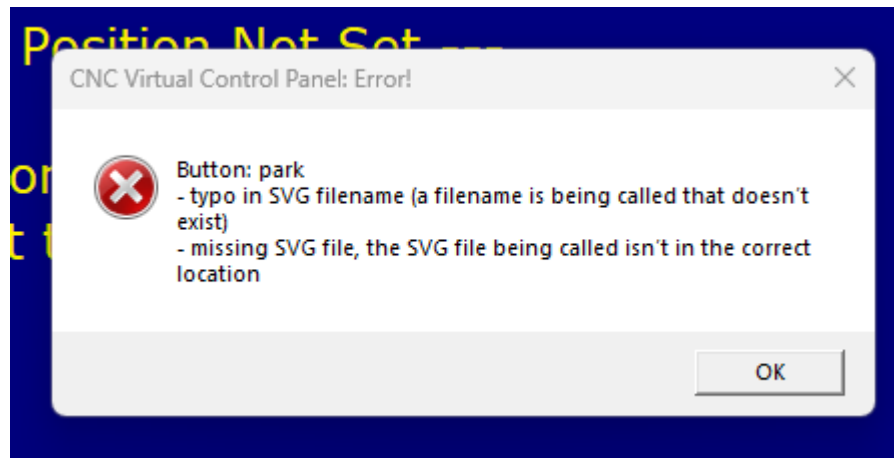
Trouble Shooting

2.) VCP won't start after editing an XML file or SVG file

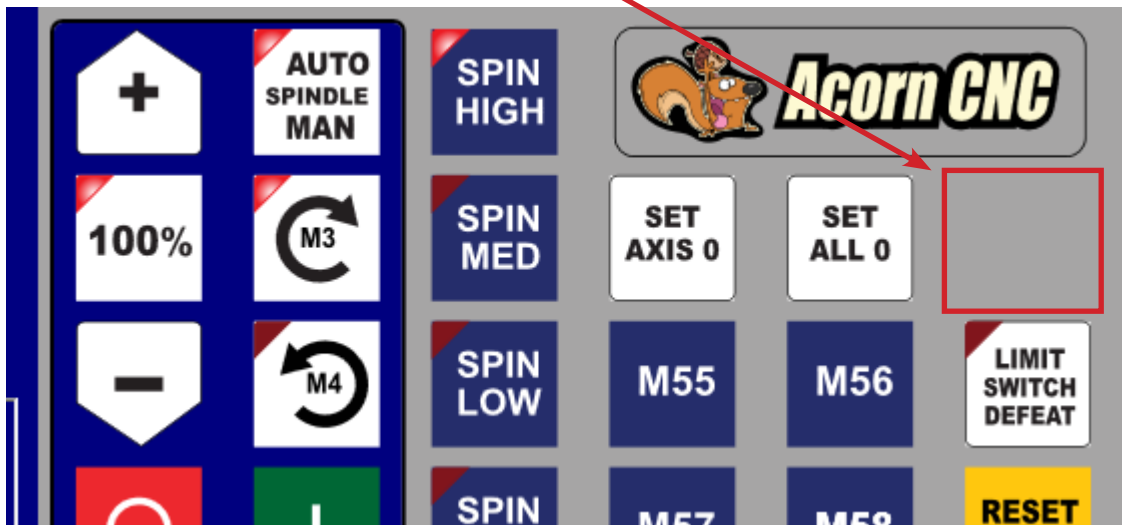
The VCP will fail to start due to a number of reasons. For beginners I recommend making one change at a time and keep backup copies of files so it is easy to revert to a working setup. Here are some common reasons why the VCP will fail to start.

- typo in SVG filename (a filename is being called that doesn't exist)
- missing SVG file, the SVG file being called isn't in the correct location
- Incompatible SVG file see below for more info.
- missing XML file for a button
- typos in the button XML file

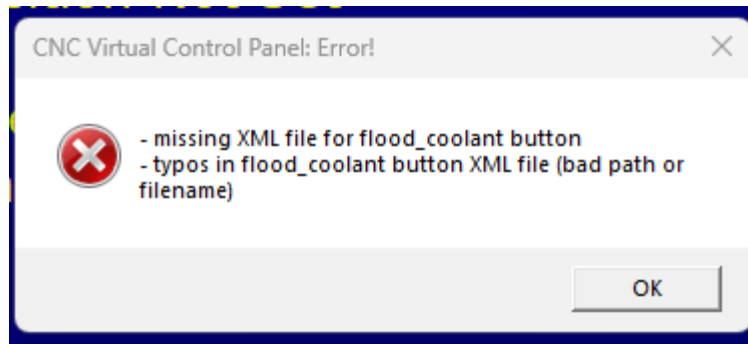
Note: The VCP has error and warning messages for common items to give some indication of why the VCP didn't start due to problems like the ones listed above.) I purposefully changed the name of the SVG file to produce this message below, moving it or not having the svg file in the correct button folder will result in the same message.



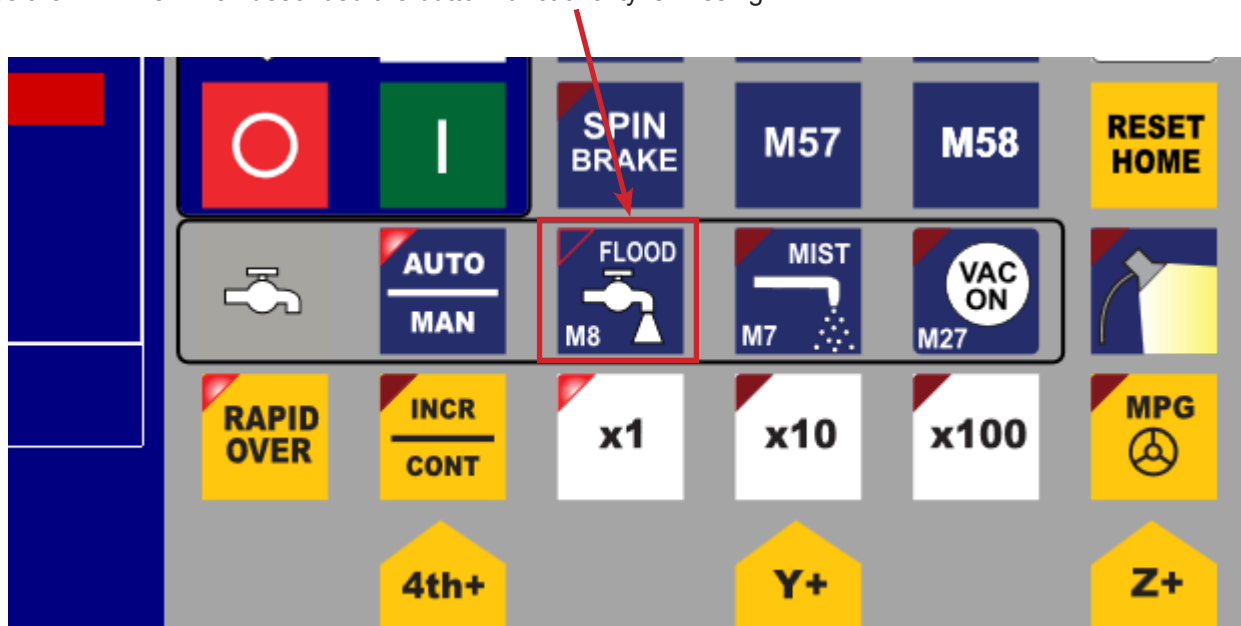
Often the VCP will still start and run for simple errors such as this. In this case the Park Button is missing when the VCP starts.



A missing XML file for a button produces this message. In this example I deleted the Flood Coolant XML file from the Flood Coolant button folder. Incorrect file names, missing file, incorrect path, typos withing the XML file will all generate this warning message.



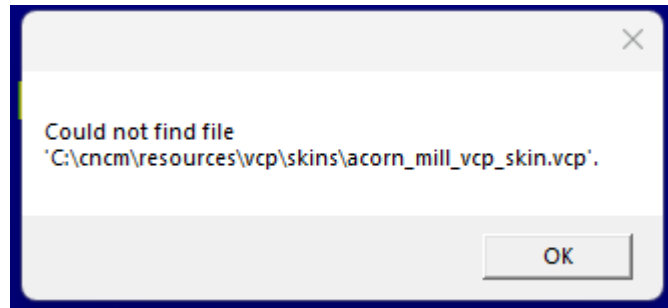
In the case of the missing button xml file the button graphic still displays but the functionality of the button doesn't work since the XML file which described the button functionality is missing.



3.) VCP doesn't show up at all.

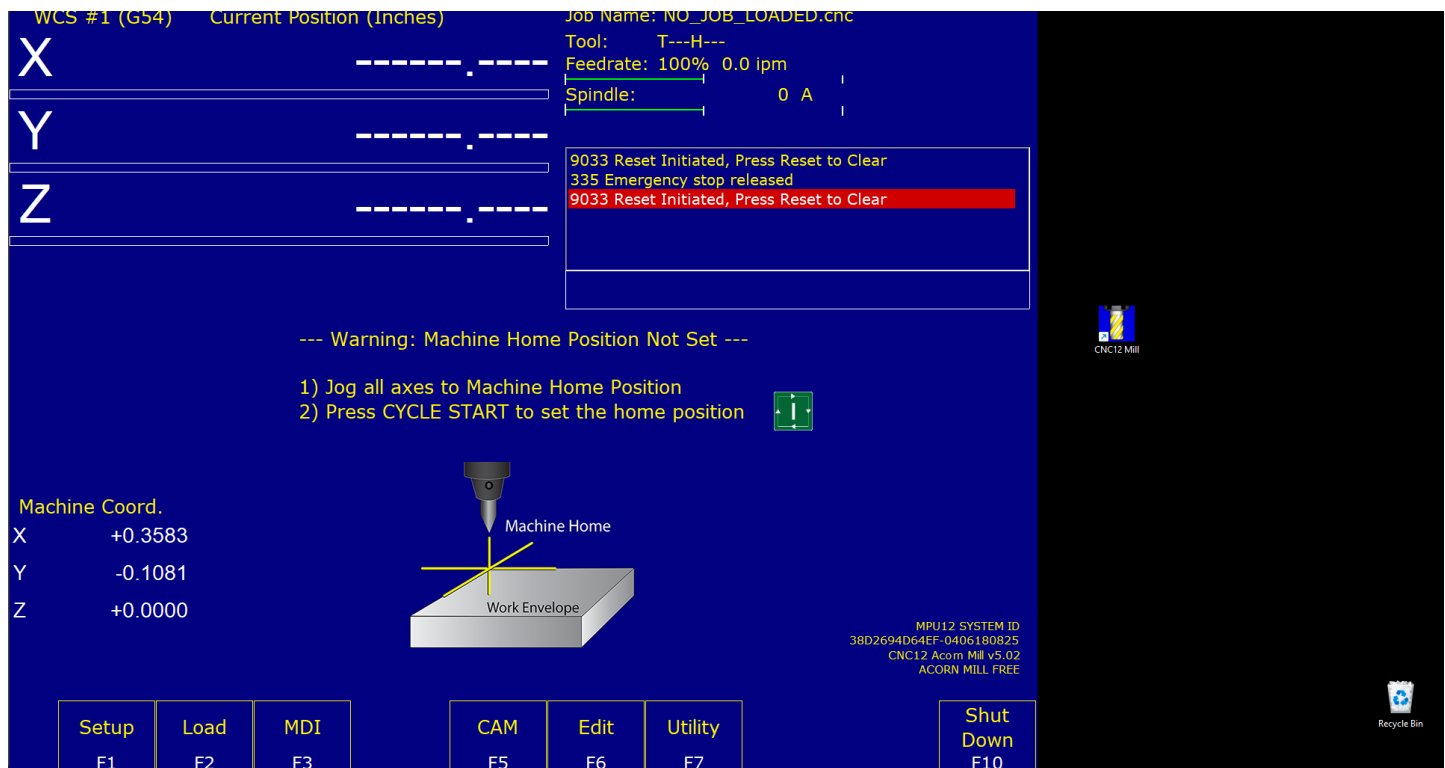
- skin has bad filename, issue could be either in the options.xml file or the skin file name itself has a typo
- skin is not in the location that the VCP is expecting (options.xml), could be options.xml is being told to look in the wrong place (bad path) or the skin itself is not in the same place that the options.xml is being told to look.

These types of errors produce this message.

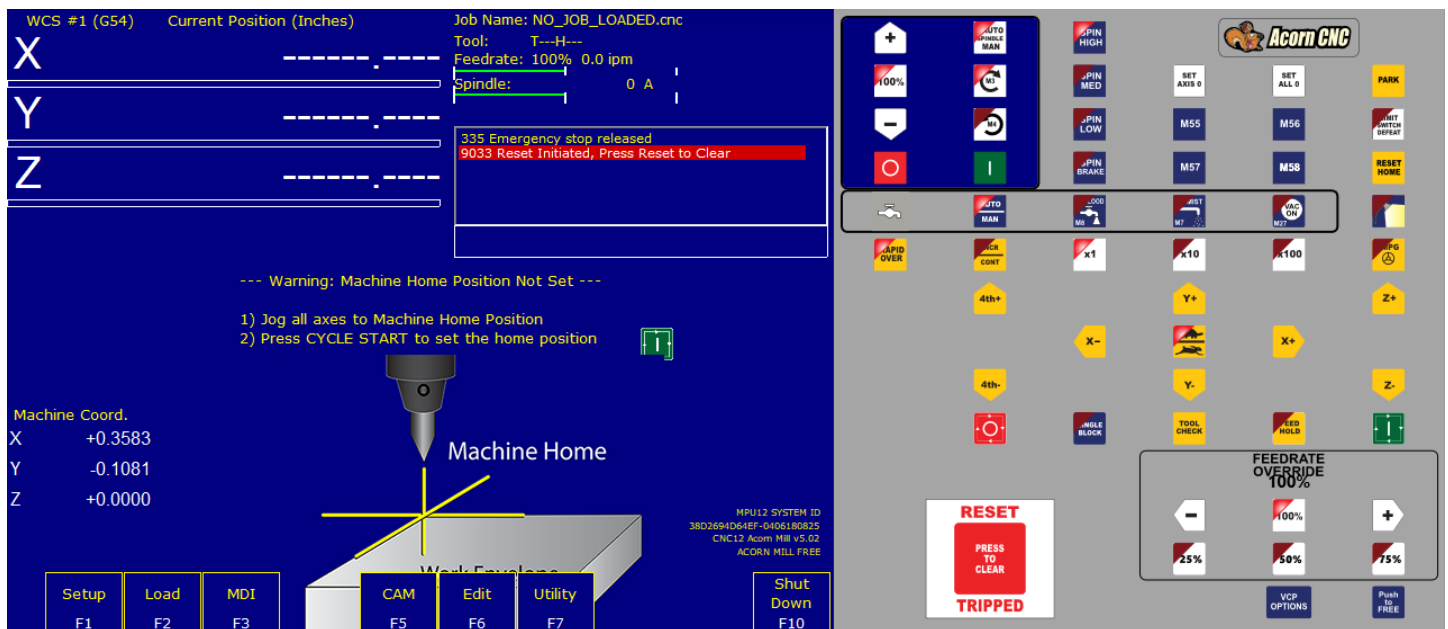


4.) VCP doesn't show up at all and there is no warning message.

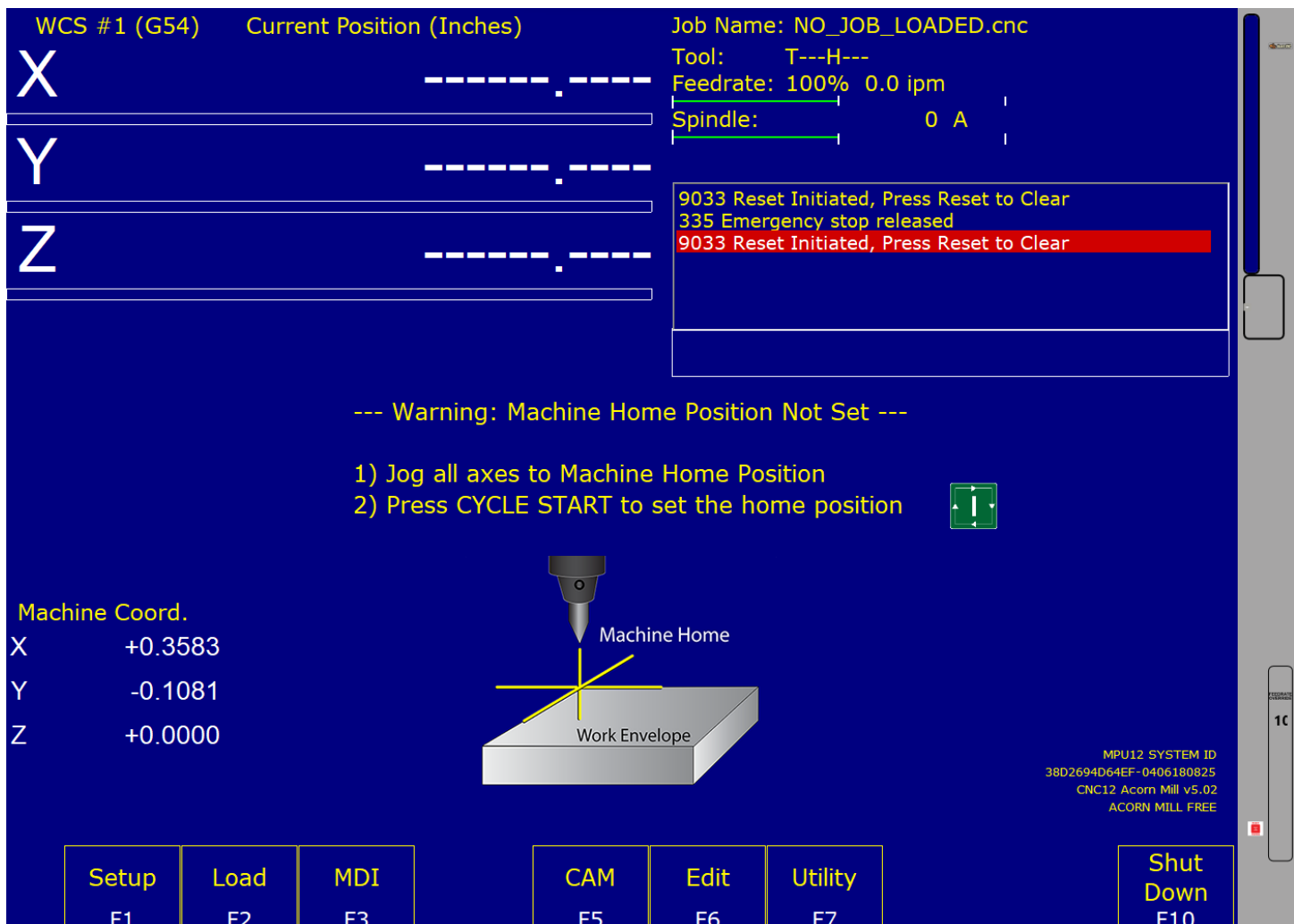
- Typos in the skin xml file. Solution: review the xml and fix the typos.
- The required windows .NET framework is not installed on the PC. Solution: run windows update and pick the .NET framework updates and install them.
- PC is running in Windows Tablet mode. Solution: set Windows to desktop mode.
- Windows "Region" is not set to United States. Solution: set Windows "Region" to USA



5.) VCP looks skewed or chopped off



Solution: Set Windows Display resolution to 1920x1080



Solution: Set Windows Display resolution to 1920x1080

6.) My SVG image file displays just fine in a browser but won't display on the VCP or causes the VCP to not start.

The SVG library that the VCP employs supports a wide variety of SVG graphic features but there are limits. One of the more common newbie mistakes is embedding a font or opening a bitmap image (.JPG, .PNG, .BMP etc) and saving it as a .SVG. This does not magically convert a Bitmap image or the font into a Vector image file, all this does is creates a Bitmap image or font that is embedded into the .SVG file. Sometimes this will actually work but this method does not scale or display well and is often fuzzy in appearance. The VCP wants a clean SVG that contains ONLY Vectors (lines and arcs), Colors and Gradients. Avoid embedded bitmap images and fonts. See the below for information on how to convert a bitmap image to a vector for use with the VCP and information on fonts.

1.) I have a logo or button image that i want to use but it is in Bitmap/Raster format (.JPG, .PNG, .BMP etc), how can i use this image with the VCP?

Bitmap images (.jpg,.png,.bmp etc) should be properly converted to Vector format (.svg) for use with the VCP. There are many ways to convert bitmaps to vectors ranging from manually tracing the bitmap with lines and arcs to full automatic bit-map to vector conversion programs and methods. You will find a wide variety of information on this subject online as it is a common procedure used by graphic artists in the Print and Web site industry as well as CNC programmers that convert art drawings and photos to G code. Inkscape has the necessary free tools for both manual and automatic conversion of a bitmap to vector image. Search on YouTube Inkscape "bitmap to vector" or 'convert image to vector' Here are a couple of links to videos showing the process of creating a Vector SVG from a bitmap image.

Auto conversion method https://youtu.be/XNEqW_rOGw

Auto conversion method <https://youtu.be/R6dgW5J0Osw>

Auto conversion method <https://youtu.be/1PX3KrwgLNc>

Manual method. more work but produces very clean vector image. <https://youtu.be/s-kPg4vYKfk>

2.) Convert all Fonts to Lines and Arcs!

Use any font you like, size and shape the word(s) anyway you like, but before finally saving the .SVG file for use in the VCP, convert all fonts (words) to lines and arcs.

- In Inkscape click on/select the word(s) and use "Object to Path" to convert the words to lines and arcs.
- In Adobe Illustrator use "create/convert to outlines" to convert fonts to lines and arcs.

Video Demonstration.

Use Inkscape to create a button with words using "Object to Path" <https://youtu.be/6ya4P2RmFHU>

3.) General SVG Graphics advice.

- Open the existing button .svg file, rename it and then modify it (this way the size/artboard is correct). All the stock button images are located in the button folders in the VCP folder.
- Delete all unnecessary art (lines, arcs, words etc..)
- Delete all unused/unnecessary/hidden Layers
- Keep all art within the art board
- "Grouping" of elements of the SVG in general works but some real convoluted grouping may cause issues so, in general if you are having issues with a graphic try ungrouping everything, you might find there are groups within groups within groups etc... keep ungrouping until the .svg displays on the VCP.
- Look for elements with no color or no line thickness and delete them.
- Make sure all elements have a color assigned to them. Sometime elements display as white on the screen as a default if no color is selected for the element and then will not display properly in the VCP since no color was assigned to the element in Inkscape.

Special Cases

1.) Installing a new version of CNC12 for Acorn and AcornSix when a Custom VCP is in use.

New versions of CNC12 will have new features and bug fixes. Check the release notes available on the download page for more information.

In general, the desire to use a new version of CNC12 software stems from wanting to make use of a new feature. In regards to the VCP if you have never customized the VCP the 'upgrading' process is simple as a new stock VCP with more features will automatically be installed and configured. However if a custom VCP is in use you can update to the new version of CNC12 to get the new features you want while keeping the features and functionality of the old custom VCP. It just takes a bit of hand editing like you did when you created the custom VCP in the first place.

The "restore report" feature can not be used to update to a newer version of CNC12. (Note: restore report can be used within the same version of CNC12 for instance, if you installed the same version of CNC12 on a new computer you can use restore report to restore all the VCP and the rest of the CNC configuration in just seconds.) But don't despair it is fairly simple to install a new version of CNC12 and migrate any exiting old custom VCP buttons and functions to the new version.

General strategy is:

- 1.) Install new version of CNC12, the installer will make a backup of the existing/working CNC12 version.
- 2.) For Acorn and AcornSix use the Wizard to configure the installation as normal.
- 3.) Use and Edit any of the stock (centroid provided) VCP skins with your 'old' VCP customizations. Edit the new version skin.xml file and copy over any custom buttons and macros from the old install to the new one. This way you will have both the new features of the new VCP with your old customizations. Video explanation with details.
https://youtu.be/MKzC1zk8WEs?si=ctPCJoPL8b_odCGP

2.) Using a Custom PLC program from old version of Acorn CNC12.

When using a custom PLC program (a plc program that has been hand edited and compiled such as a custom Lathe Turret program or mill umbrella ATC program) Edit the new latest Wizard or Centroid provided CNC12 PLC program with your customizations and use the latest VCP skin and enjoy all the new features of the new VCP and the new PLC program.

- a.) For Acorn and AcornSix users: Hand edit the new Wizard auto generated PLC program with your old PLC customizations and recompile. Customize the VCP 2.0 skin to match your application (if necessary).
- b.) For Oak, Allin1DC and MPU11 users Hand edit any of the new stock centroid provided PLC program with your old PLC customizations and recompile. Customize the VCP 2.0 skin to match your application (if necessary).

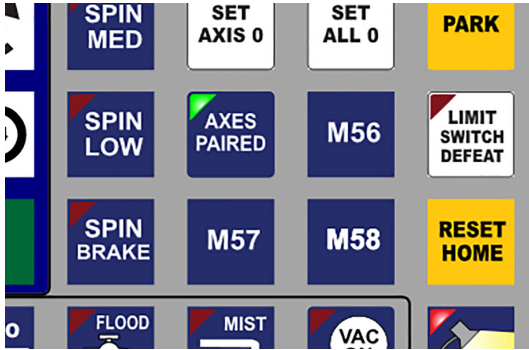
Video overall discussion of installing a new version of CNC12

<https://youtu.be/j3Kr8CI-nx4?si=hFibH95rdMBI4VMG>

Special Cases

4.) Axes Paired/Re-pair Axes button.

The Axes Paired button is automatically injected into the VCP button layout by the Acorn Wizard when any two axes are software paired together using the Wizard. (See the Acorn CNC12 Axis Pairing users manual for more info on axis pairing. https://www.centroidcnc.com/centroid_diy/downloads/acorn_documentation/paired_axes_acorn_user_guide.pdf) The default location for the Axes Paired button is Row 3 / Column 4 and uses the same skin event number as Auxiliary key 10.



Said another way the Wizard “takes over” the Auxiliary 10 button (which is typically pre assigned to M55) and uses the M75 macro for the Re-pairing code.

So, if you are using software paired axes the Auxiliary 10 Button will be dedicated to the Axes Paired Functionality and can not be used for any other use.

You can move the axes paired button just like any other VCP button by editing the VCP skin and choose a new row and column space for the ‘axes_ paired’ button.

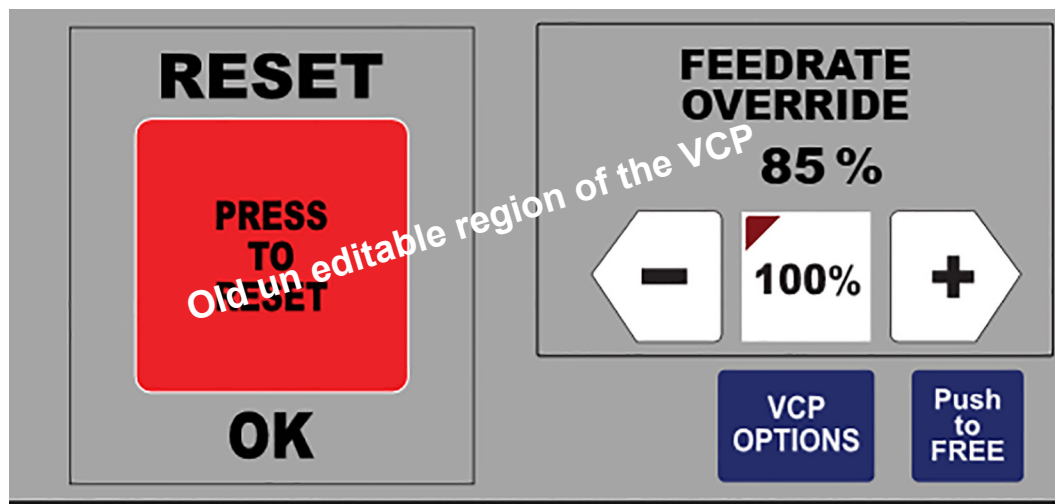
When software pairing is selected in the Wizard pairing menu, the Wizard will remove which ever button is at Row 3, Column 4 and replace it with the Axes Pairing button or the M55 button if axes pairing is off unless a custom VCP is in use.

5.) In the old VCP (any version before v5.0) these buttons were fixed in size and location.

The Reset, Feedrate Override, VCP options and Push to Free buttons are fixed in size and position in versions v4.82 Acorn and the v4.22 Oak/Allin1DC and older.

You could change the images and even edit the xml file but you can not change the size of the button box or the position or delete these buttons. Now with the new VCP v5.0+ this entire region is user editable just like the rest of the VCP.

This area and these buttons are now fully editable just like the rest of the VCP in CNC12 v5.0+



Edit and Create new VCP Skins and Graphics Offline (on another computer that is not the CNC control PC)

Starting with CNC12 v5.04 To make it a bit easier to check out if your VCP graphic customizations are working or not, it is possible to run the VCP without being connected to a CNC control board. It is always ideal to be working on the actual control system when building a new VCP button and feature since you can actually run a macro and or interact with the PLC program but, that is not always practical certainly for the Graphics design part of any new VCP skin.

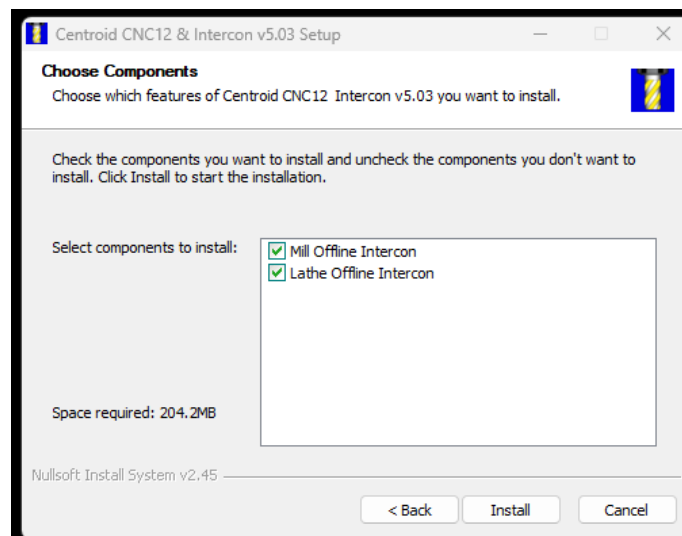
To 'install' the VCP on an 'offline' computer (not the CNC control computer) that is not connected to any CNC controller.

1.) Download and unzip the "Offline Mill and Lathe installer. here: https://www.centroidcnc.com/centroid_diy/centroid_cnc_software_downloads.html

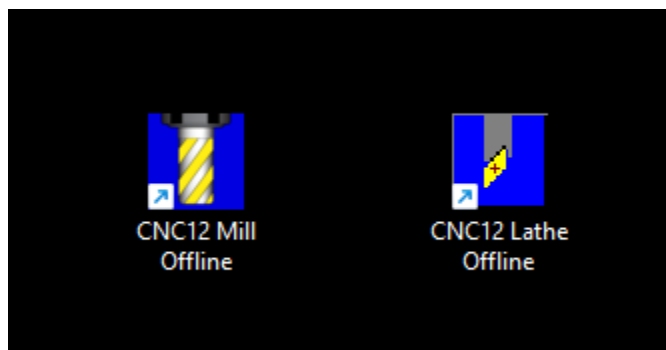
Offline Mill and Lathe Intercon: Interactive Conversational programming software for use on a Windows 10/11 PC (not the CNC controller PC)

[Download the 32 bit v5.08 Offline Mill and Lathe Intercon installer\(1-5-24\)](#)

2.) Run the offline Intstaller. Choose Mill or Lathe or install both. Follow instructions on the screen.



These desktop icons appear. You can use the Mill offline version to test any Mill, Router or Plasma VCP graphics set.



and their corresponding installed directories are:

c:\Centroid_Mill_Intercon_Offline

c:\Centroid_Lathe_Intercon_Offline

The VCP folder is in the same location as the CNC controller online system.resources\VCP

Resources

- **Notepad ++:** <https://notepad-plus-plus.org/>

A free powerful text editor used for editing VCP XML files. Also useful for editing G and M code programs and macros.

- **InkScape:** <https://inkscape.org/>

A free Vector Drawing program. Great for editing and creating VCP buttons and graphics. Also useful for CNC Art work.

- **YouTube:** YouTube.com

Search on “Inkscape for Beginners”, “InkScape bitmap to vector” or “Inkscape convert image to vector”

- **Hex code Color Picker:** w3schools.com

Hex color picker website. While not necessary as Inkscape has similar feature built in this page is a good reference.

- **Centroid Technical Support Forum:**

[Custom VCP thread specifically discusssing custom VCP's with examples.](#)

Sign up for Free Tech Support.

<https://centroidcncforum.com/index.php>

- **Centroid CNC12 software download page.**

https://www.centroidcnc.com/centroid_diy/centroid_cnc_software_downloads.html