



# Mighty Macros: Powerful Commands to Pump up Productivity

Sam Lucido – Haley and Aldrich, Inc.

## AC1337

This class is filled with all the tips and tricks you'll need to create macros in Autodesk® AutoCAD®. Macros are small groups of commands and routines that can be any length, without requiring any special characters at the end of the line. In this class, you learn to create custom macros that apply to your business needs and enhance your productivity on designs. You will develop a custom tool palette placing over 50 of your new macros on the palette for quick access during project design. Most importantly, what you learn will bring value to your employer and give you knowledge that can help you excel within your field.

### Learning Objectives:

At the end of this class, you will be able to:

- Identify, change and create macros.
- Use the action recorder to create custom macros.
- Create macros to automate multiple commands.
- Create a tool palette with custom macros enabled.

### About the Speaker:

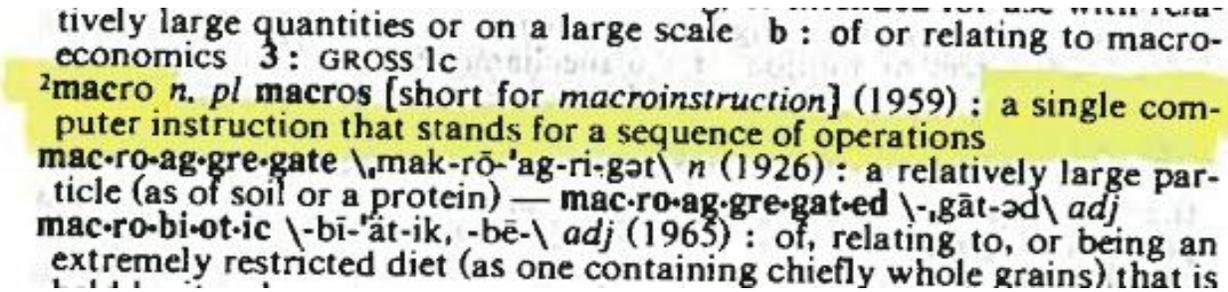
Sam Lucido is a sought out CAD designer and trainer at Haley and Aldrich, Inc. As a CAD Services Manager he presents workshops on CAD productivity to managers and users in both corporate and classroom settings. He provides support on a wide variety of architectural, civil, mechanical, and structural design projects. Sam has over 20 years of experience involving production design and drafting, user support and standards coordination. He continues to be very self-motivated and enjoys working in a team environment to accomplish project objectives and create high quality deliverables. Sam is the owner and operator of CADProtips.com and is professionally certified in AutoCAD 2011-2014.

[slucido@haleyaldrich.com](mailto:slucido@haleyaldrich.com)  
<http://www.haleyaldrich.com>  
<http://www.cadprotips.com>  
[@CADProTips](#)

Let's begin by defining the three important topics of this class: A Macro, the Action Recorder, and Tool Palettes. This is not a tutorial on tool palettes or the action recorder but we will need to quickly review these topics as we are going to use both during this lesson. We will use the action recorder to create macros to swap our tool palette paths to help make it easier to standardize and create our new palette. We are also going to explore macros within the CUI to show how you can use existing commands to help create a new more efficient process.

**What is a Macro?**

A macro (in many different programs) can be defined as a way to automate a task that you perform repeatedly with more than one command or keystroke. In AutoCAD macros can be shortcuts to a series of commands to help make the process of design more efficient. You can use the action recorder to record a series of commands and build a macro then run it automatically to repeat a series of steps. To write a macro, you type the commands in the macro properties section as you'd type them in at the command line. If a command displays a dialog box, you would place a dash in front of the command to suppress the dialog box. We will cover special characters as we begin to build our macros.

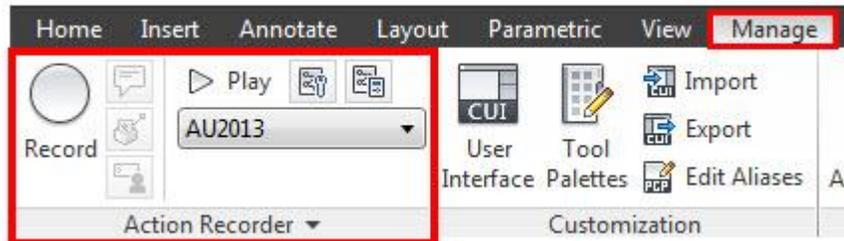


**From Autodesk Help**

A macro can contain commands, special characters, DIESEL (Direct Interpretively Evaluated String Expression Language) or AutoLISP programming code. (AutoLISP is not supported by AutoCAD LT)  
 Note: As AutoCAD-based products are revised and enhanced, the sequence of prompts for various commands (and sometimes command names) might change. Therefore, your custom macros might require minor changes when you upgrade to a new release of you application. You add macros to interface elements by using the Customize User Interface (CUI) Editor. Select an existing command or create a new command in the Command List pane. Enter macros in the Macros section of the Properties pane. There are no length limitations for macros. However, you do need to know how specific characters are used in macros and be aware of other considerations or limitations.

**What is the Action Recorder?**

The action recorder is simply a tool to record macros for later use. AutoCAD will read the commands you input then save as a macro. You can include requests for user input and messages to make the macro work interactively. When you save a macro, it has an ACTM filename extension. You'll find it in your Support\Actions folder of your AutoCAD installation. You can share ACTM files with others users on your team. You'll find the Action Recorder on the Tools tab, in the Action Recorder panel as shown in the image to the right. You can also choose Menu Browser (the A button)> Tools> Action Recorder.



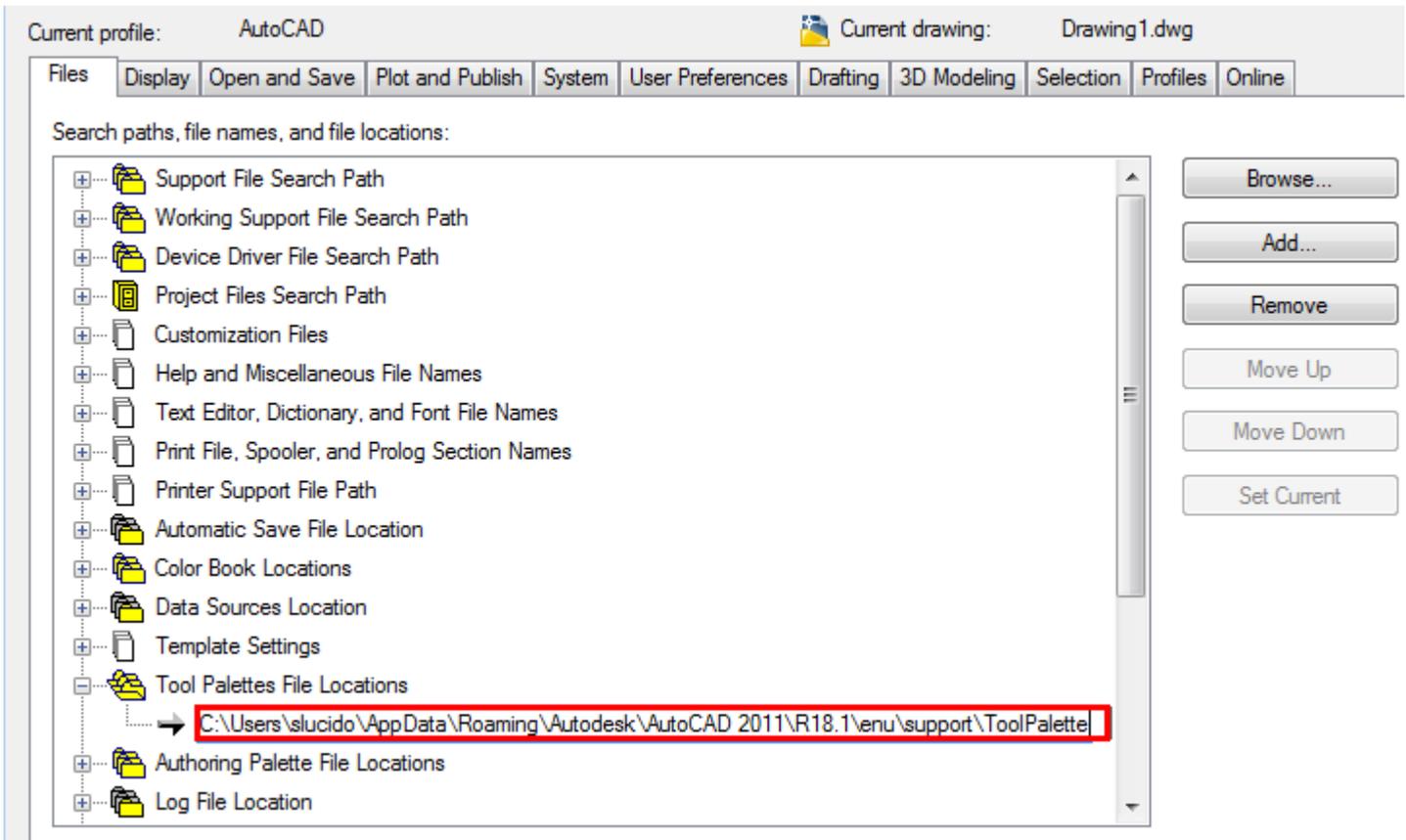
### What is a Tool Palette?

Palettes provide the shortest quickest way to increase productivity without programming. A tool palette is a free-floating tab that you can bring up on screen and keep active while you work in your drawing, so you have quick access to common symbols, commands, and most any other tool you need to complete the design. You can add any command from AutoCAD to a palette, creating quick access to all of your favorites and shortcuts.

Let's begin by using the action recorder to create a macro that will change the tool palette path to a new location where we will build our macros. We do not want to alter the default AutoCAD palette path so we will first make a macro to restore that and flip between our new palette and the default version. We are going to create a new blank tool palette so we can begin placing our macros on it. First, I would like to refer to *Matt Murphy's* class at AU2012:

**AC3441: The Productivity Power of AutoCAD® Tool Palettes – Revealed!**  
(This class was presented at Autodesk University 2012)

I used Matt's technique of creating a macro to locate the tool palette path which gives the illusion of groups. This works great when switching between different groups of blocks, command tools and engineering disciplines. We will go through this briefly to assist you in getting things setup. Just like managing your support path statements in AutoCAD, you can set a Tool Palette path location and not use the group feature.



**HINT:** if you ever delete the path accidentally, just leave it blank and AutoCAD will replace it with the default path as shown above.

We want to set this up prior to creating our new palette. Let's first create a macro to save the default path (We want to be able to go back to that later). We will name the macro TPD (tool palette default). Go to the manage tab on the ribbon and select record for the action recorder. Select the Record button as shown in Figure 1 and 1a. Type `*_TOOLPALETTEPATH` at the command prompt. Next we need to go out to the options menu and find the tool palette default path and double-click and copy the default path. Paste into the command line.

Figure 1

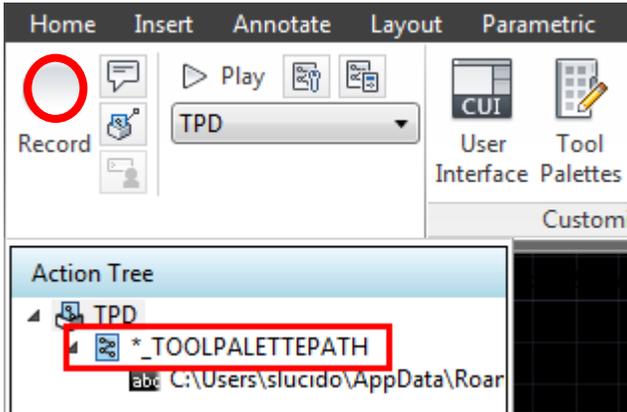


Figure 2

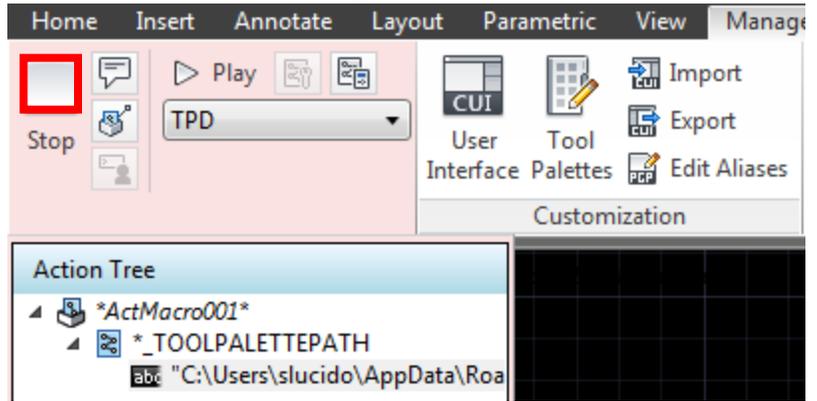
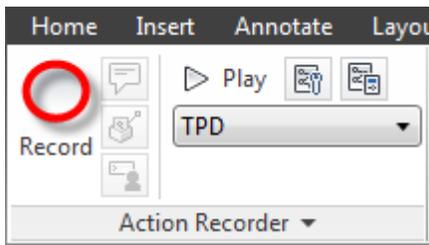


Figure 3

Stop the action recorder (Figure 2) and save the file as shown in Figure 3. You will define the file location in the options dialog box as shown on the previous page. Set that do a default location for your macros. Be sure to add a description as well so other user may know what exactly the function of the macro is.

Figure 1a



Now let's go out to our hard drive and create a folder called AC1337 (or whatever you prefer). I wanted to use the name of the class (the letter "A" will sort to the top of my folder structure making it easy to navigate to). Copy that address as shown in Figure 4 and go back to your AutoCAD session. Your path will should look something like "C:\AU2013\Mighty Macros\AC1337".

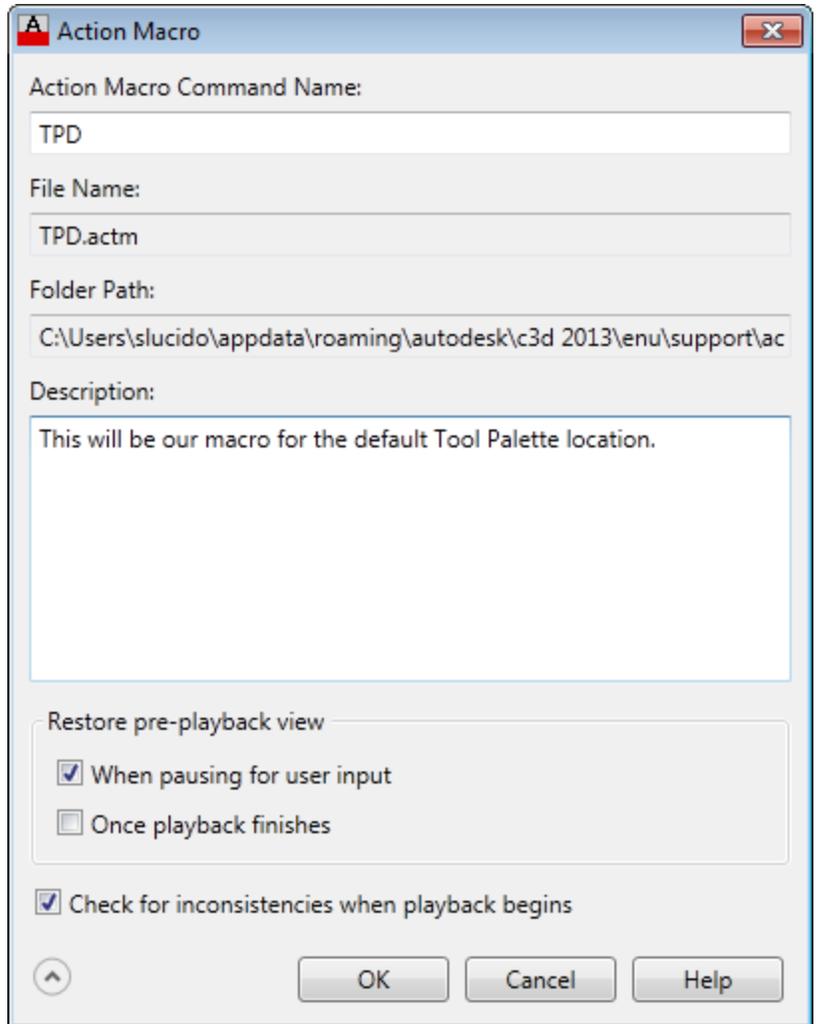
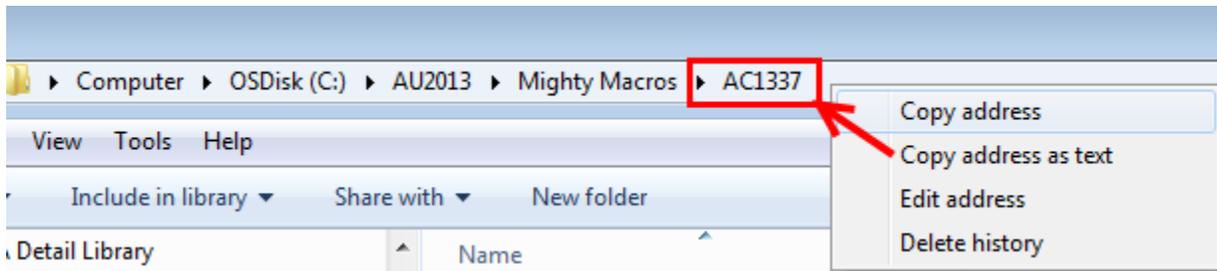
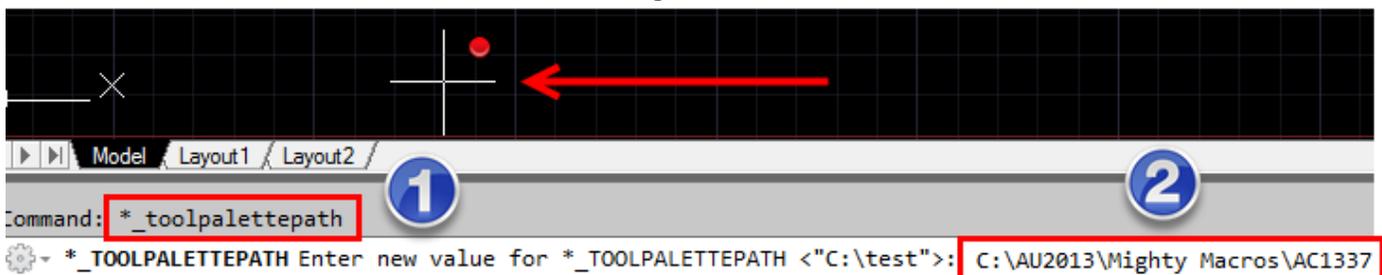


Figure 4



Go back to the Action Recorder and follow the same steps as we did before. Type **\*\_TOOLPALETTEPATH** at the command prompt; then on the command line when it asks you for a location paste in your path for your new folder as shown in steps 1 and 2 on Figure 5. I put an arrow next to the cursor, notice the red dot, which is to indicate that you are recording the actions of your macro.

Figure 5



Hit Stop. Name the macro and save the file (i.e. Figures 2 and 3). When you type the macro at the command prompt you will see a blank new palette for you to build your library as shown below. This is a great way to keep things organized.



Remember, you can always type **TPD** at the command prompt and you will get right back to the AutoCAD default palettes. If you copy your new tool palette folder to a secure location on your network, you can use a macro just like above to switch to that location. You can also copy the AutoCAD default tool palette folder to the network and complete the same task. This is a great way to share Tool Palette content while allowing users to add and manage their own content by having multiple tool palette paths. Using the Action Recorder to record action macros can be simple but you need to stay organized. After an action macro is recorded, you save the recorded commands and input to an action macro, which has the file extension ACTM. The Action Recorder contains the tools to record, play back, and modify an action macro. You can also set the preferences for the Action Recorder from the Action Recorder Preferences dialog box. During the playback, editing, or recording of an action macro, you can expand the Action Recorder panel to access the individual actions of the current action macro from the Action tree.

The Action recorder is a very powerful tool where you just have to type in commands at the command prompt and have little or no knowledge at all about programming. I encourage you to test it out with new commands and make sure you save and enter a good description of your command. How about one to insert your company title block, logo or scale bar? Just make sure you save those descriptions so other users can benefit from your hard work. Now that we have discussed the action recorder and tool palettes we will not begin to create our macros. Let's first rename our blank palette to My Macros and start to build our library. Right-click on the palette name and type in My Macros.

Figure 6

To start writing out macros we will need to think about what it is we want to do then take a look at the command line and see what happens as you perform the commands. Let's start with a few basic macros to get the hang of how this works. We need to get a command into the palette.

We will first start by creating a macro that changes a text string to uppercase. A common scenario we might encounter while receiving drawings from a sub-contractor and all the text is sentence case. You would issue the textedit command, select the text, right click and change to uppercase. Add a command to our blank palette by typing CUI at the command prompt and dragging the command into the palette as shown in Figure 6 to the right.

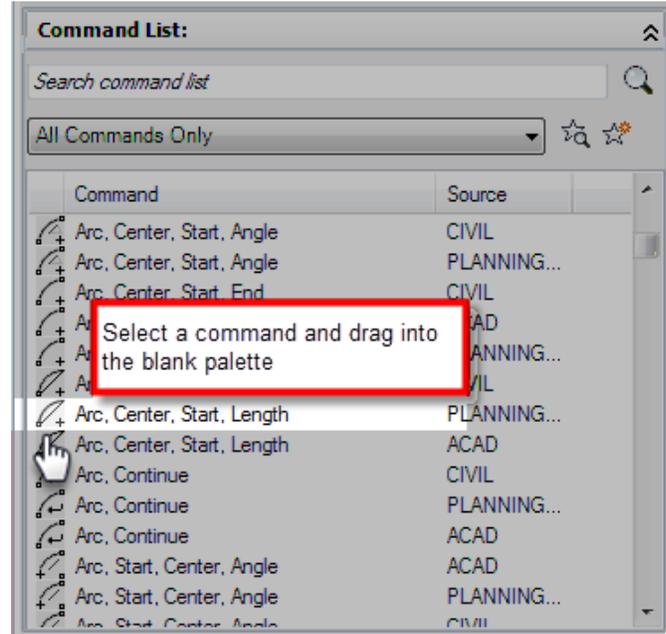


Figure 7

Now that our command is in there we will right-click and bring up the image dialog box menu as shown in Figure 7. This is where you can rename the command on the palette and even specify an image to use. You will notice that your button already contains an image as other buttons do not have any image assigned to them. You can use the button editor to create your new image. Simply go to the CUI and select any command on the command list. The Button image editor will be displayed and you can choose any button that has been supplied by Autodesk or you can clear the image and create your own. We are going to choose one that looks like a piece of text to start with (shown above) then add a red arrow indicating changing to uppercase as shown in Figure 8.



We will then export our button image to an image file on our hard drive that we can reference to the palette

Figure 8

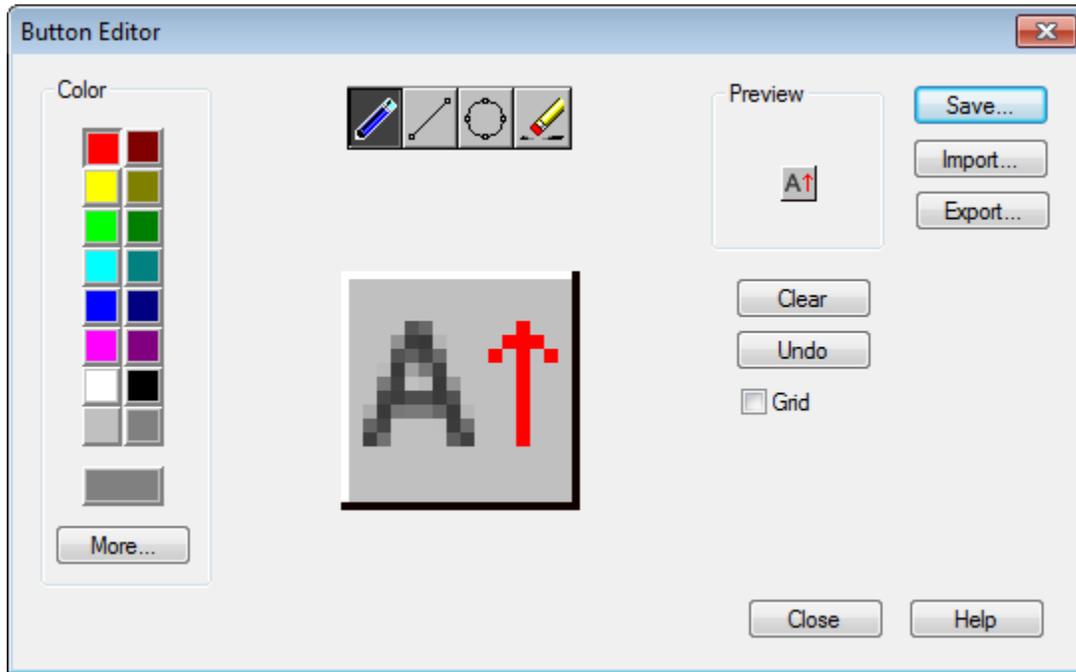


Figure 8a

Finally, right click on our command in the palette and choose properties as shown to the right in Figure 8a. In the command string sequence is where we will enter our macro information as shown in Figure 8b. We need to follow the command line and figure out what happens when we run the command. Notice on the command string we have some special characters. Table 1 shows the special characters you will use when creating a macro. Notice how the forward slash is a pause for user input and a semi-colon represents a return or enter on the keyboard.

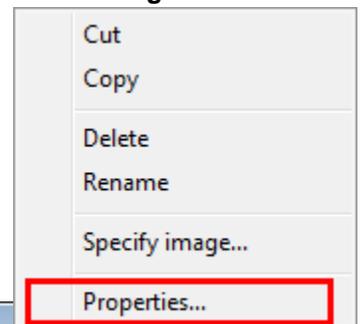


Figure 8b

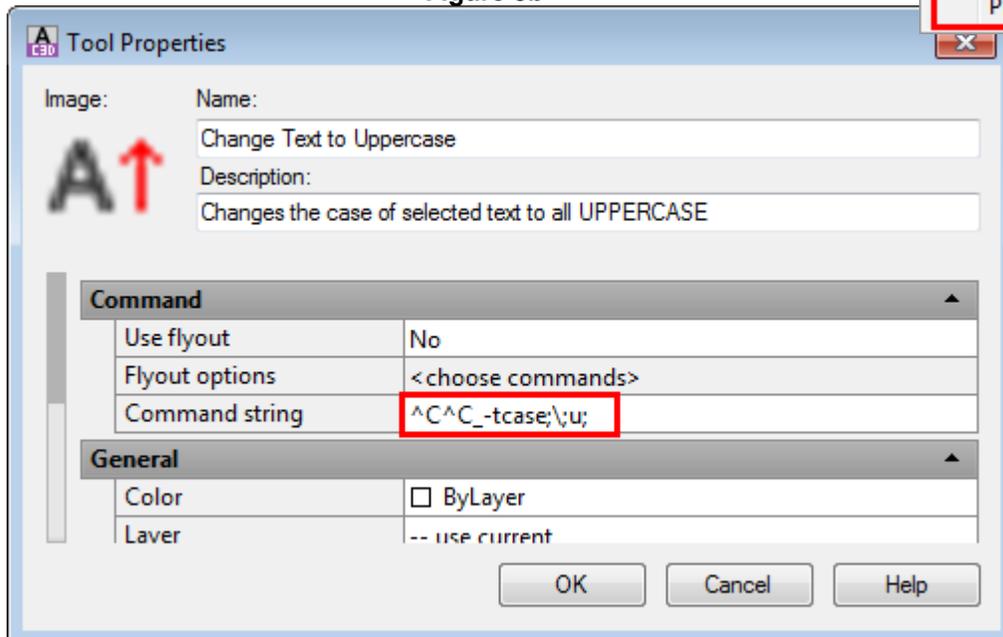


Table 1 - Special Characters in Macros	
Character	Description
;	Issues Enter
^M	Issues Enter
^I	Issues Tab
[blank space]	Enters a space; a blank space between command sequences in a command is equivalent to pressing the Spacebar
\	Pauses for user input (cannot be used with accelerators)
. (period)	Allows you to access a built-in command even if it was undefined using the UNDEFINE command. (Not Available in AutoCAD LT)
_	Translates commands and options that follow
=*	Displays the current top-level pull-down, shortcut, or image menu
*^C^C	Repeats a command until another command is chosen
\$	Introduces a conditional DIESEL macro expression (\$M=)
^B	Turns Snap on or off (equivalent to Ctrl+B)
^C	Cancel the active command or command option (equivalent to Esc)
^D	Turns Dynamic UCS on or off (equivalent to Ctrl+D)
^E	Sets the next isometric plane (equivalent to Ctrl+E)
^G	Turns Grid on or off (equivalent to Ctrl+G)
^H	Issues Backspace
^O	Turns Ortho on or off
^P	Turns MENU ECHO on or off
^Q	Echoes all prompts, status listings, and input to the printer (equivalent to Ctrl+Q)
^R	Turns command versioning on or off. Command versioning is required for some commands to ensure command macros written in an older release work properly in the latest release.
^T	Turns tablet on or off (equivalent to Ctrl+T)
^V	Changes the current viewport
^Z	Null character that suppresses the automatic addition of Spacebar at the end of a command

Let's take a look at our macro to change the text case and those commands and special characters that make up the change text case macro, one step at a time as shown in Table 2 below.

Table 2 - Change Text Case	
Command	Description
^C^C	Cancel any previous action
_-TCASE	Runs the Tcase command. Be sure to include the hyphen in front to suppress the dialog box.
;	The semi colon represents a return on the keyboard
\	Pauses for user input (select the text object)
;	Ends the "select objects" mode (Issues a return or enter)
U	Uppercase selection
;	Issues an enter and ends the command

**Macro: ^C^C\_-Tcase;\;U;**

Now we will test out the command and see how it works. You could make a few additional ones to accommodate for sentence case and lower case. Moving on we will now start building our palette according to work flow: Resources, Setup, Production and Finalize. Throughout this tutorial you will see the palette image to the right that will indicate what commands we are going to create.

**Online Resources:**

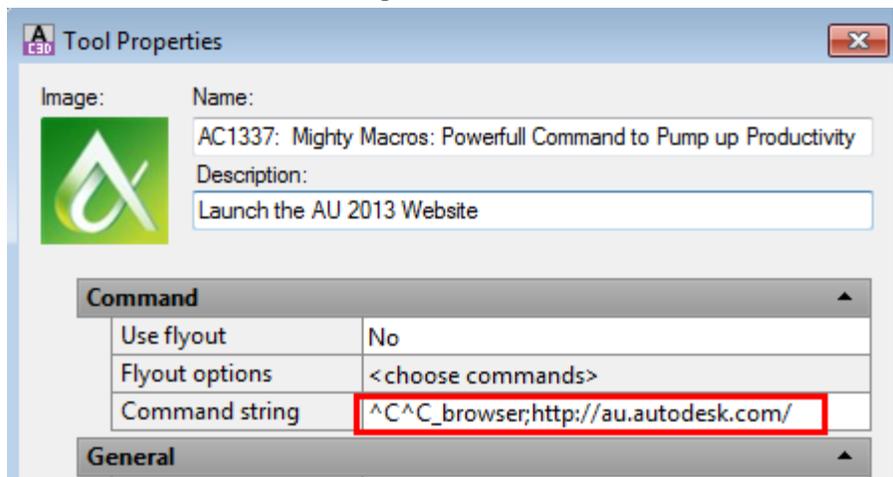
Out next two macros will launch a web page from the browser command in AutoCAD. The first button will take us to the Autodesk University website and the second to the Venetian Hotel in Vegas.



**Figure 9**

Yes, you can easily access web sites through favorites in your browser but why not have the resources you need for your particular discipline put on a palette.

Notice instead of using the CUI button editor I have images placed within the tool palette as shown in Figure 9. You can reference any image into a tool palette for the first example we used the AU logo as shown in Figure 9.



The command sequence for the macro can be found in Table 3 below.

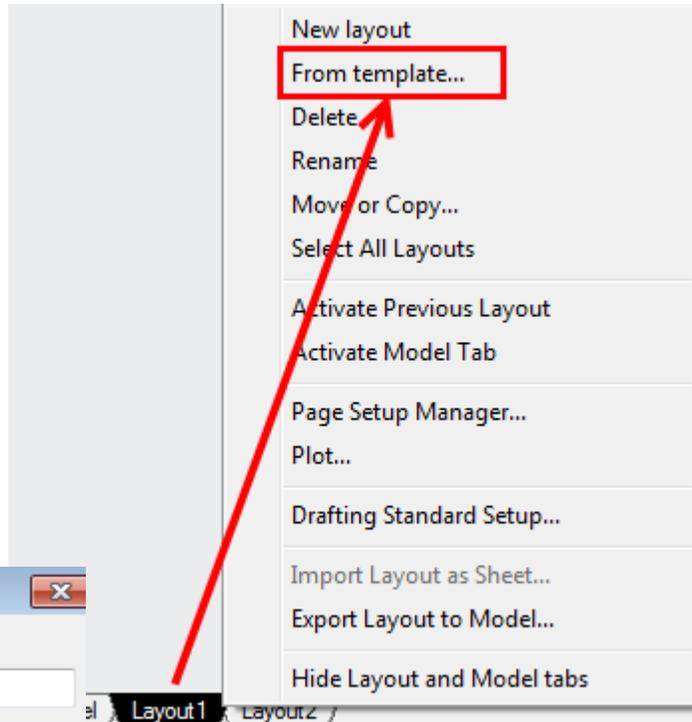
Table 3 - Browser command	
Command	Description
<b>^C^C</b>	Cancels any previous action.
<b>_browser</b>	Runs the Browser command.
<b>;</b>	The semi colon represents a return on the keyboard.
<b>http://au.autodesk.com/</b>	Launches default browser pointing to Autodesk University webpage

**Macro: ^C^C\_Browser;http://au.autodesk.com/**

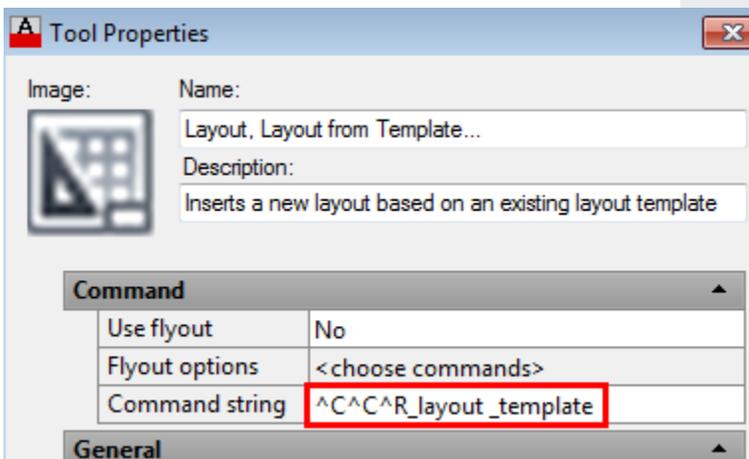
**Project Setup:**

We will now use a macro to setup our title blocks from a template for a new project. What do we need? We just need the template file that contains all of your layouts. Many companies use this method for inserting title blocks yet have them placed within individual icons on the palette or they load the template and have several layouts will all of the different title blocks within. We are going to add a button that will simulate the from layout command as shown in Figure 10.

**Figure 10**



**Figure 11**



Launch the CUI to find the layout from template command as a starting point. We are simply going to add to that command and insert the path to our template file as shown in Figure 11.

All we need to do now is add the path to our template file. Keep in mind that if you copy the path change the slashes from backwards to forward as shown in Figure 12 below (if you just paste the path in there AutoCAD will pause the macro at the first “\” waiting for user input. Table 4 shows the command sequence, one step at a time.

Figure 12

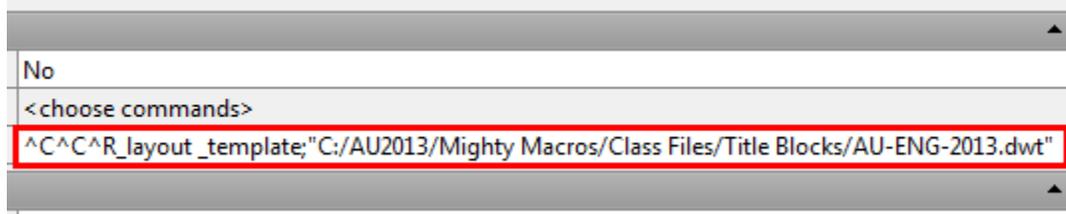


Table 4 - Title Block Template	
Command	Description
<code>^C^C</code>	Cancel any previous action.
<code>^R</code>	The ^R will issue the new version of the command.
<code>_layout_template</code>	This command is taken from the cui to insert a layout from a template
<code>;</code>	Issues a return on the keyboard
<code>"C:/path"</code>	Path to your template file. Notice that the slashes are forward and not the traditional backward slash. That is because the backward slash pauses for user input in a macro.

**`^C^C^R_layout_template;"C:/AU2013/Mighty Macros/Class Files/Title Blocks/AU-ENG-2013.dwt"`**

The next three macros go along with project setup. We are now going to use the same technique to import page setups into drawings then multileader styles. As a CAD manager I find it very helpful to drive customization and standards from a single place or file. In our firm we have multiple page setups for different disciplines. We can create a template file with all the page setups and styles then reference that from a macro. The image below shows the next section of our customized palette. Again we can just copy and paste these as we go then change the image and code to our preference.

For the page setups we will use the same code as the title block macro with one exception. For some reason using the command to invoke the psetupin command launches a dialog box. Even with the special character (-) to suppress dialog boxes this command will still enable the box. The fix is to turn off the filedia system variable prior to the command then turn it back on when you are finished. Our multileader styles both standard and annotative will be inserted from a drawing template then launching the Multileader style command to view. See Table 5 and 6 for the Page Setup Import and The Import Multileader Styles respectively.

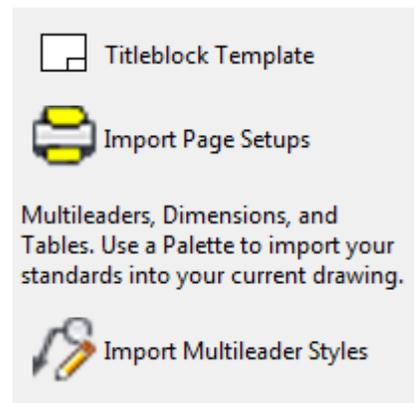


Table 5 - Import Page Setups	
Command	Description
<b>^C^C</b>	Cancel any previous action.
<b>filedia</b>	System variable to turn off dialog boxes
<b>;</b>	Issues a return on the keyboard
<b>0</b>	Turn off dialog box
<b>;</b>	Issues a return on the keyboard
<b>^R</b>	The ^R will issue the new version of the command.
<b>_psetupin</b>	Import Page setup command
<b>;</b>	Issues a return on the keyboard
<b>"C:/AU2013/..."</b>	Path to your template file. Notice that the slashes are forward and not the traditional backward slash. That is because the backward slash pauses for user input in a macro.
<b>;</b>	Issues a return on the keyboard
<b>filedia</b>	System variable to turn off dialog boxes
<b>1</b>	Turn those dialog boxes back on!

**^C^C\_filedia;0;^R\_psetupin;"C:/AU2013/Mighty Macros/Class Files/Title Blocks/AU-ENG-2013.dwt";filedia;1**

Table 6 – Multileader Styles	
Command	Description
<b>^C^C</b>	Cancel any previous action.
<b>_insert</b>	Insert command with the special character (-) preceding to suppress the dialog box.
<b>"C:/path"</b>	Path to your multileader template file. Notice that the slashes are forward and not the traditional backward slash. That is because the backward slash pauses for user input in a macro.
<b>;</b>	Issues a return on the keyboard
<b>^C^C</b>	At this point you want to cancel the command since the drawing settings have already been inserted within the drawing
<b>_multileaderstyle</b>	Runs the multileader style command. You certainly do not have to run the style command. This has been added to show the class the styles that have been imported into our file
<b>;</b>	Issues a return on the keyboard

**^C^C\_insert;"C:/AU2013/Mighty Macros/Class Files/03 Multileaders.dwg";^C^C\_mleaderstyle;**

The last button on our setup section is system variables. Sometimes we receive drawings from other sources and settings can be off or not to our liking. This macro loads some favorite settings that we use as our standard. See Table 7 below for the command sequence. This can become a very long macro, for the example I have only shown a portion of the entire macro, you can adjust and enter the system settings for your company standard.

<b>Table 7 - System Variables Setup</b>	
<b>Command</b>	<b>Description</b>
<b>^C^C</b>	Cancel any previous action.
<b>cecolor</b>	cecolor system variable
<b>;</b>	Issues a return on the keyboard
<b>bylayer</b>	Sets the variable to bylayer
<b>;</b>	Issues a return on the keyboard
<b>msltscale</b>	This variable scales linetypes displayed on the model tab by the annotation scale. This is a great way to see your linetypes as they are displayed both in paper and model space
<b>;</b>	Issues a return on the keyboard
<b>1</b>	Enters the value for the system variable
<b>Continue to enter all of the system variables that you typically set within your drawing file. Use the same format and input the variable followed by a semi-colon</b>	
<b>mtjigstring</b>	This variable sets the content of the sample text displayed at the cursor location when the MTEXT command is started
<b>alert</b>	Issues the alert command to let the operator know that the commands have been loaded

```
^C^C_cecolor;bylayer;celtype;bylayer;cmdecho;0;mirrtext;0;menuecho;0;olestartup;1;
peditaccept;1;plinegen;1;lwdefault;0;visretain;1;psltscale;1;msltscale;1;imageframe;0;
savetime;5;mtjigstring"(getvar "loginname")); (alert (strcat "System Variables Set!"))
```

**Note:** I have added an alert lisp function at the end to alert the user that the variables have been changed. You can delete that but nothing will show up when you select the button, sometimes it's good to be sure we made the right pick!

**Quick Editing with Macros**

Next up we will go through several editing commands and how you can automate functions with macros. The first will be the rotate command. Both of these commands will use a special character (\*) to repeat the macro. For example, say you want to rotate a text object 90 degrees then repeat. How about rotate an object 1 degree then continually repeat till you get things lined up? Don't forget about the torient command (located in express tools). Our two new palette items will look similar to what is shown to the right. Table 8 and Table 9 below shows the command sequence for the macro(s).

This section will serve as miscellaneous commands that help you become more productive.



Rotate object 90d then repeat

Rotate object 1d RIGHT then repeat

For the 1 degree macro we simply need to copy and paste and change the number from 90 to 1. Remember that if you want to rotate counterclockwise you will have to enter a negative number in the macro.

Table 8 - Rotate 90d then repeat	
Command	Description
*	Repeat the macro when action has been performed
^C^C	Cancel any previous action.
_rotate	Starts the Rotate command
;	Issues a return on the keyboard
\	Pause for user input
;	Issues a return on the keyboard
@	The @ symbol in AutoCAD will retrieve the last picked point
;	Issues a return on the keyboard
90	Enters the rotation angle of 90d
;	Issues a return on the keyboard

```
*^C^C_rotate;\;@;90;
```

Table 9 - Rotate 1d then repeat	
*^C^C_rotate;\;@;1;	

Continuing on with our editing section we are going to create 11 new commands all using AutoCAD commands and special characters. For the next 11 macros we will just list the command in sequence and not separate out into a table until we get to the clean screen swap. We will begin this section drawing a pline and a specified width. Maybe you use a pline with a .1 width to specify flow in a process diagram or outline a designated area of a site plan. Eliminate an extra step by putting this in a macro. Customize your macros by simply changing the width of the variable or even add the layer make function in there to have this draw on a particular layer. Both scenarios are shown in Tables 10 and 11 below.

<b>Table 10 - Polyline with a specified width</b>
<code>^^^C_PLINE;W;.1;;</code>

How would you change this sequence to make a new layer then go back to the previous layer you were on?

<b>Table 11 - pline with a specified width and layer</b>
<code>^^^C_-layer;m;newline;c;2;;;_PLINE;W;.1;;</code>

The snap angle macro will select a line using the object snap nearest point(s) and changing the cursor angle to that point. This can be very helpful when drawing things to a specific angle. We need to reset the snap angle back to 0 which is why the following button is directly below it. These macros use the object snap code of 512 which is the nearest (yes you can use the transparent command near). You can enter these values in a macro or create your own by simply entering the correct integer. For the object snap mode value refers to Figure 13 (OSMODE table) to the right. Both commands are referenced in Tables 12 and 13 as shown below.

<b>Table 12 - Set the snap angle to an object</b>
<code>^^^C_OSMODE;512;_SNAPANG;\;;</code>

Let's run another macro to reset it back.

<b>Table 13 - Reset back to 0</b>
<code>^^^C_SNAPANG;0;</code>

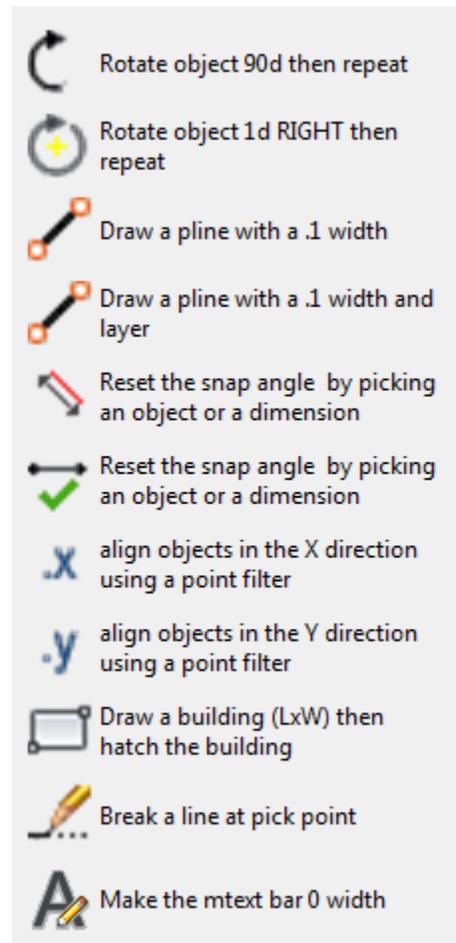


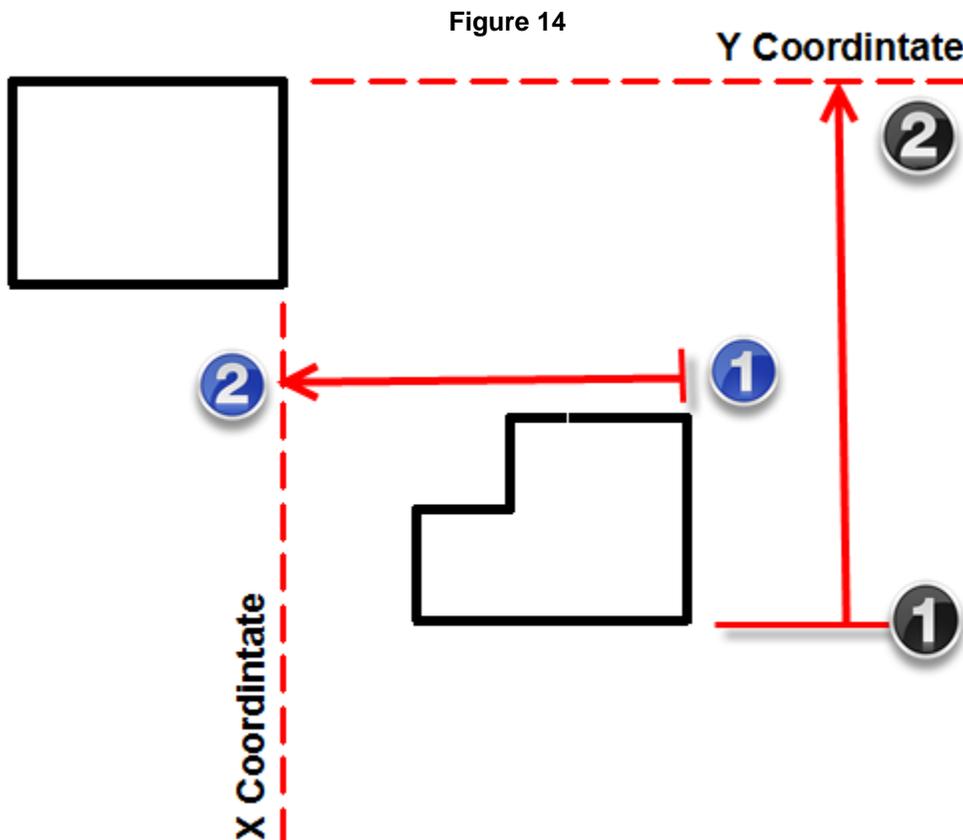
Figure 13

OSMODE	
Integer	Description
0	NONE
1	ENDpoint
2	MIDpoint
4	CENter
8	NODE
16	QUADRant
32	INTersection
64	INSertion
128	PERpendicular
256	TANGent
512	NEARest
1024	Clears all object snaps
2048	APParent Intersection

The next two commands use point filters. Most of us who do not work in the 3D world sometimes forget about how valuable point filters can be in 2D. Point filters are used in conjunction with base point selections for the move, rotate and scale commands as well as any other command requiring a base point. There are times when the point you want to use for the base point may not be easily found through the standard object snaps. For example, you may need to move an object to a new elevation but maintain its current xy location. The simple way to do that is with point filters. You would start the move command, select your object to move, and, at the first base point request, select any end point or snap point on your object. When it asks for the base point to move to, you would enter select the other object and the macro will retrieve the .x or .y value. AutoCAD will store the x or y coordinate and you will only be able to move your object in one or the other direction. A great way to line up objects in AutoCAD. These macros make it easy by letting you pick the x or y filter then hit enter to line the object up. Let's take a look at the example in Figure 14.

In the example below (Figure 14) we are going to select our macro for the x filter. It will prompt us for the end point and we are going to select the blue 1. Then the command will issue the .x filter, simply select a the end point or the object near the blue 2. Notice what happens in AutoCAD, you can move the object in the y direction but the x coordinate is locked. Point filters can be very useful in lining objects up.

- .x** align objects in the X direction using a point filter
- .y** align objects in the Y direction using a point filter

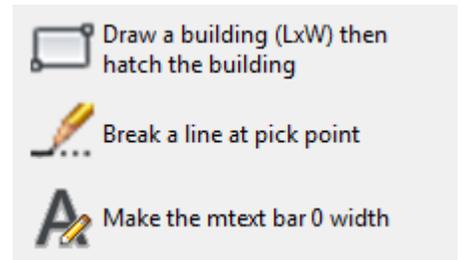


Both macros show in Tables 14 and 15 start off by changing the object snap mode to an endpoint. You could issue a few osnaps to be sure you are grabbing the correct point. Select your object to move along with the x coordinate. The end of these use the object snap code of 1024 to clear all snap settings.

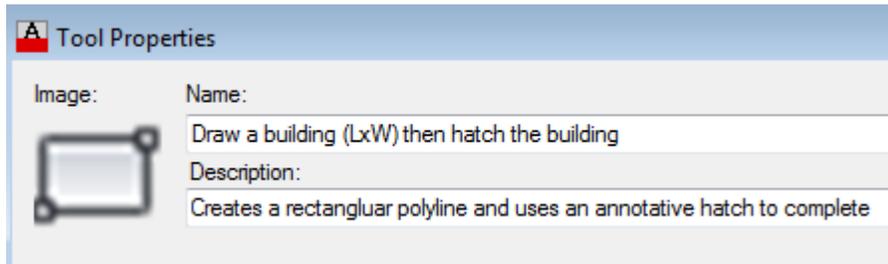
<b>Table 14 - Move an object using X point filter</b>
<b><code>^C^Cosmode;1;_move; \.x;\OSMODE;1024</code></b>

<b>Table 15 - Move an object using Y point filter</b>
<b><code>^C^Cosmode;1;_move; \.y;\OSMODE;1024</code></b>

The next three macros on our list try to accommodate solutions for a few different scenarios. The first is to draw a building using the rectangle command with dimensions, and then hatch that building with an annotative hatch. The second will break a line at a pick point and change the first part of that line to hidden. The last one is a quick macro on how to make the mtext editor just a step quicker by entering a 0 mtext width.

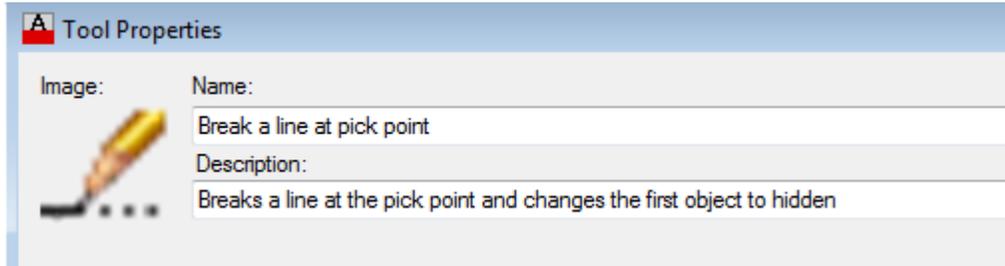


Let's take a closer look at the draw building macro and the commands associated with it. This one uses several different commands in AutoCAD to create the box then hatch the box with an annotative hatch. This macro shown in Table 16 shows you how to incorporate several different options to fit your business needs.



<b>Table 16 - Draw a building then hatch the building</b>
<b><code>^C^C_layer;m;C-BLDG;c;5;;;_rectang;\d;\-layer;m;C-BLDG-HTCH;c;8;;;-bhatch;s;l;an;y;p;ansi31;1;0;;</code></b>

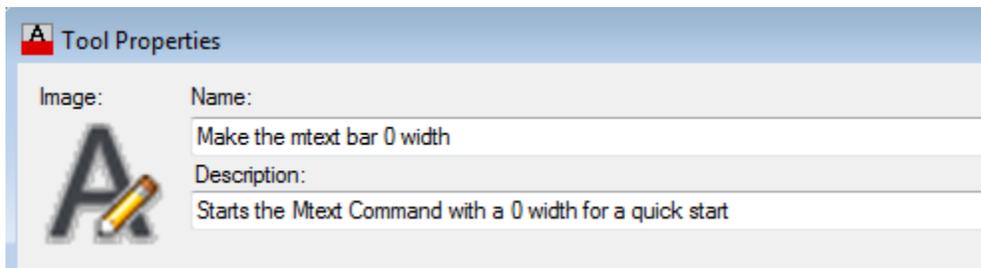
Next up is changing or breaking a line at a point using the nearest osnap then taking that line and making the original section hidden. You could break this macro down further to change layers. Many times in the design process you will need to keep a line on the same layer and just change it to hidden, it does not happen often but it does happen. The command sequence is shown on the next page in Table 17.



**Table 17 - Breaks a line at a point then changes the original line to hidden**

**`^C^C_OSMODE;512;_break;\f;\@;chprop;l;;lt;hidden;;OSMODE;1024`**

This next command is a launch the mtext command with a 0 width. It just gives the operator more of a dtext feel if they still are having a hard time transitioning from single line text. See Table 17a below for the command string.



**Table 17a - Quick Mtext Start**

**`^C^C_Mtext;\w;0;`**

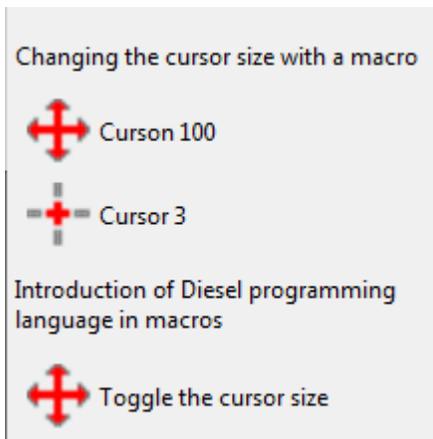
### Changing the Cursor Size with a Macro

For years I have been using a lisp routine to switch the cursor size from 3 to 100, flipping back and forth. Sometimes we need that cursor size to be small so some areas of our drawing are not obstructed. I decided to try this same concept by using a macro. The first macro will set the cursor size to 100 and the one below will change back to a 3. Table 18 shows how this simple macro can change that system variable quickly.

**Table 18 - Change the cursor from 100 to 3**

**`^C^C_CURSORSIZE;100`**

**`^C^C_CURSORSIZE;3`**



We now move on to the toggle cursor size which I mentioned I use in a lisp file. I came across this Macro while reading [CP214-1 Kate Morrival's Into to Macros Writing class from AU2009](#) (look up this class under Autodesk University 2009). This macro does exactly what my lisp routine does but with the use of diesel programming. First we will explain a little history of the diesel programming language.

## What is Diesel?

DIESEL (Direct Interpretively Evaluated String Expression Language) expressions take strings as input and generate string results. DIESEL is not like LISP or VBA since it's completely built into AutoCAD and you do not have to load or create any external files to use it. You can use DIESEL in menu macros or to display text on the status line. You can use DIESEL to alter the status line through the MODEMACRO system variable. Let's take a look at our next three commands which all use the DIESEL programming language.

### The History of Diesel by Ralph Grabrowski

Then with AutoCAD Release 12 for DOS, Autodesk introduced the fully customizable status line. Unfortunately, the user couldn't simply select options from a dialog box. Instead, the user needed to learn *Yet Another Programming Language*, this one called "Diesel" and the sixth different programming interface added to AutoCAD at the time.

Short for "direct interactively evaluated string expression language," the programming logic of **Diesel** is as clear as the acronym's meaning. Despite the word "string," Diesel mostly operates on numbers, not strings. While its purpose is to customize the status line, Diesel has found its way into menu macros and became the most powerful programming environment available in AutoCAD LT -- much to the chagrin of Autodesk, who deliberately disabled the AutoLISP that was supposed to ship with LT because its retailer were worried LT's low price would cannibalize sales of full-blown AutoCAD. Despite the handicap, European programmers have done some amazing things for LT third-party software with Diesel's limited facilities.

Is Diesel a true programming language? For me, the line of differentiation between a macro language and a programming language is whether there is logical functions, such as **If**, **While**, or even **Greater Than**, etc. (Logic functions make it possible for the program to make decisions.) Diesel has logic functions but the syntax is so obscure that it begs to be known as a simple macro language -- and that's how I'll refer to it from now on.

## What Diesel Does

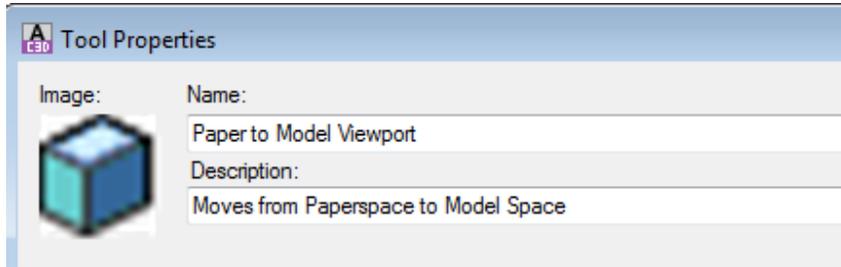
Diesel allows me to change AutoCAD's status line so that it displays other useful information, such as the z-coordinate, the DWG filename, and the time. There is a limitation, though: the text displayed by Diesel is truncated after 32 characters, no matter how big I make the window (39 characters in Release 13 for Windows and LT for Windows 95). Diesel has an unusual format for its macro language. Every function begins with the dollar sign and a bracket:

`$(function,variable)`

No doubt, the purpose of the \$-sign is to alert the AutoCAD command processor that a Diesel expression is on the way, just as the (-symbol alerts AutoCAD that an AutoLISP expression is coming up. The opening and closing parentheses signals the beginning and end of the function. This allows Diesel functions to be nested, where the variable to one function is another function. The parentheses also allow Diesel to work on more than one variable at a time; the closing parenthesis alerts Diesel that there aren't any more variables.



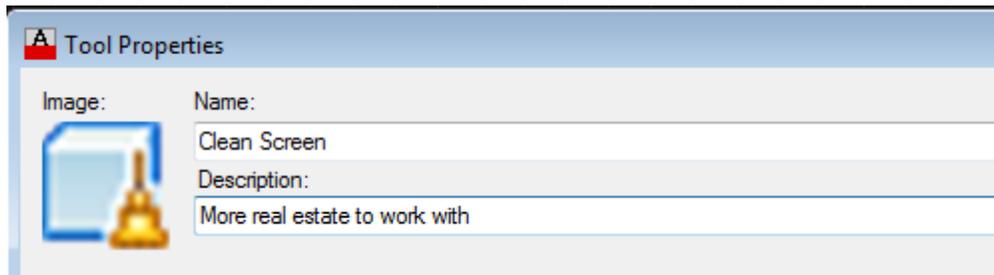
The next command will toggle between paper space and model space. This specific macro is designed to be used with one viewport and also can be found in the **AutoCAD help file**. Much like if you double click inside the view to activate the viewport. This macro will save you a keystroke by switching from paper to model quickly.



**Table 20 - DIESEL to toggle between paper and model space**

`^^C^P$M=$(if,$(=,$(getvar,cvport),1),mspace,pspace)`

The clean screen command (CTRL+0) cleans the screen in AutoCAD. Take a look at the code as shown below in Table 21. Go to the CUI and find the clean screen command which is shown in Figure 17 on the next page. Grab that code and let's alter it. We want a clean screen but we also want the command line off, menubar off, and the status bar off. One exception we still want to see out tool palettes so we can toggle back. With some altering this can be accomplished in a macro as shown below in Table 21.



**Table 21 - Clean Screen with Options**

`$M=$(if,$(and,$(getvar,CleanScreenState),1),^^C_CleanScreenOFF;statusbar;1;menubar;1;commandline,^^C_CleanScreenON;statusbar;0;menubar;0;commandlinehide);tp;`

Within the CUI you can grab all sorts of code for macros. Figure 17 (next page) shows the clean screen macro as it is listed in the CUI indicating the use of diesel, we modified that slightly as to accommodate some extra settings. Figures 17a and 17b show two separate examples, the group selection and layer manager. This gives you a good reference of how to use diesel in creating your own macros.

Figure 17

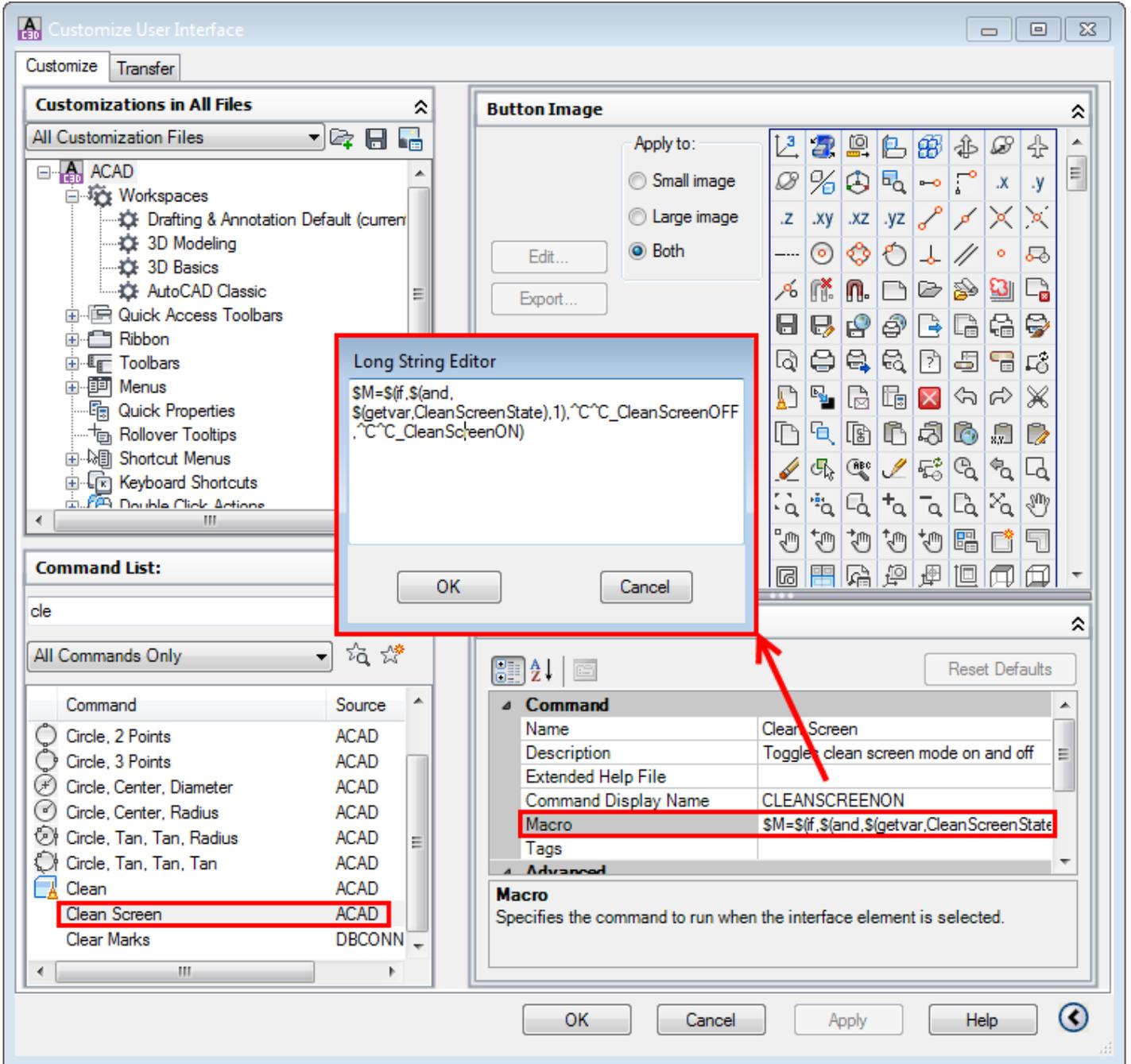


Figure 17a

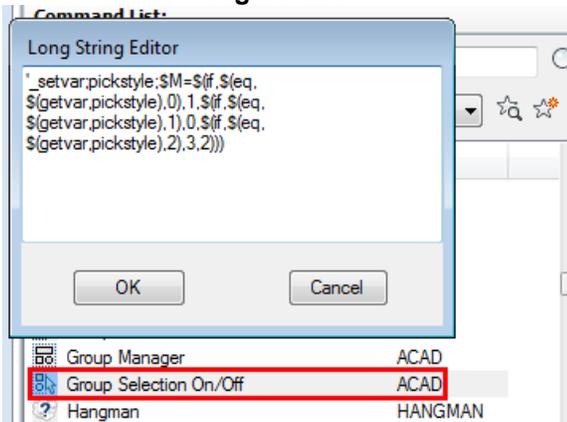
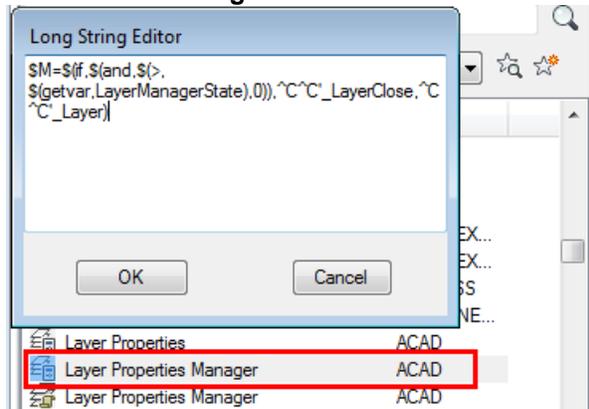


Figure 17b



### Load and Unload Xrefs quickly

A couple of the first macros I created are the following two and passed these out at AU2012 during my class on Linetypes. I use xrefs all of the time and some fairly large image and or civil site plans that I want to load and unload quickly. With a click of the button we can load and unload these files with a simple macro as shown in Table 22.

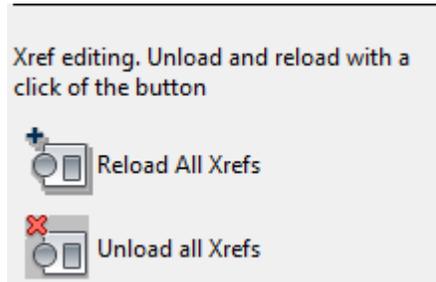


Table 22 - Unload and Reload Xrefs
<code>^^C-xref;u;*</code>
<code>^^C-xref;r;*</code>

### Cool tricks with Layers

I bet it's safe to say that all of us have used some sort of trick with the layer command or with layers. Many people forget to look at the command prompt when using AutoCAD. With the release of AutoCAD 2013 there have been significant improvements to the command line and how you can select the options. The first macro will let you select a layer by picking then it brings up the dialog box (same as hitting N or Name at the command prompt) for you to select another layer to merge. Many times I am working and I know the layer I want to merge with but not the name of the other. This is a great way to select it by visually looking at the list. Simply type in the macro as shown in Table 23, notice the special character ^R, this is necessary to bring up the dialog box.

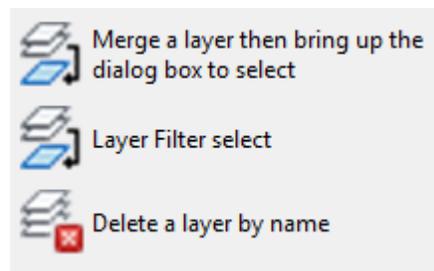


Table 23 - Layer Merge with Name
<code>^^C^R_laymrg;\;n</code>

Do you use layer filters? A layer filter limits the display of layer names in the Layer Properties Manager, and in the Layer control on ribbon and the Layers toolbar. In a large drawing, you can use layer filters to display only the layers that you need. Shown below are examples from Autodesk help on what layer filters really are.

There are two kinds of layer filters:

**Layer property filter.** Lists the layers that have portions of their names or properties in common. For example, you can define a property filter that lists all layers that include the letters *mech* and are set to the color red. Layer property filters can include nested layer property filters.

**Layer group filter.** Lists the layers that you assign to the group, regardless of their names or properties. You can add layers to a layer group filter by dragging them from the layer list onto the group filter. Layer group filters can include both nested layer property filters and layer group filters.

There are five predefined filters:

**All.** Lists all the layers in the current drawing.

**All Used.** Lists all the layers on which objects in the current drawing are drawn.

**Xref.** If xrefs are attached to the drawing, lists all the layers being referenced from other drawings.

**Viewport Overrides.** If there are layers with overrides for the current viewport, lists all the layers containing property overrides.

**Unreconciled New Layers.** If new layers were added since the drawing was last opened, saved, reloaded, or plotted, lists all new unreconciled layers.

Table 24 shows a macro brings up the filter named AU. Yes, you need to have a name of the filter to bring it up using a macro. Maybe your company uses a prefix to define layers. You can then use the macro with the filter name to bring up just those layers quickly.

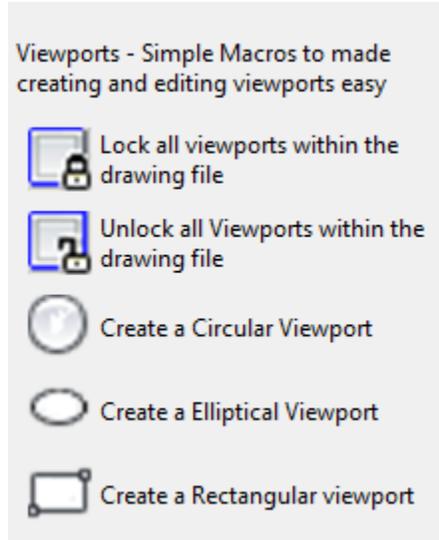
Table 24 - Load a layer filter
<code>^C^C+layer;AU;</code>

This next macro as shown in Table 25 I added in there since I like to see what layer I am deleting. Deleting a layer but with a dialog box. I want to know the name of the layer I am deleting and confident that my file structure is intact. In this instance you will have to use the ^R to get the name dialog box to appear. This is similar to the layer merge command simply just bringing up the list for you to view.

Table 25 - Layer delete Name
<code>^C^C^R_laydel;n</code>

## Viewports

You can create a single layout viewport that fits the entire layout or create multiple layout viewports in the layout. Once you create the viewports, you can change their size, their properties, and also scale and move them as needed. With MVIEW, you have several options for creating one or more layout viewports. You can also use COPY and ARRAY to create multiple layout viewport. Viewports are essential to the drawing process; you may use a rectangle, ellipse, or even a circle. Good management would be to lock all viewports when you are finished working in a drawing session. This next set of macros deals with viewports, creating and locking.



One thing that can frustrate a user is to go into a drawing and zoom in and out and loose the view. If you do not have a view set or your drawing limits defined you will not know where the last view was other than the scale. Locking viewports is key to keeping good file management. These next two commands will lock all the viewports in all layouts. I issued a lisp function to call an alert box to notify the user that the task has been completed. Simply take this out or use language of your own, many times I feel it is necessary to let the user know the task has been completed.

Table 26 - Lock Viewports in a Layout
<code>^C^C_-vports;l;on;all;;(alert (strcat "All viewports have been locked"))</code>

Table 27 - Unlock Viewports in a Layout
<code>^C^C_-vports;u;on;all;;(alert (strcat "All viewports have been locked"))</code>

The next three are fun ways of using objects to create a viewport. You can create a new viewport with nonrectangular boundaries by converting an object drawn in paper space into a layout viewport using the MVIEW command. With the Object option, you can select a closed object, such as a circle or closed polyline created in paper space, to convert into a layout viewport. The object that defines the viewport boundary is associated with

the viewport after the viewport is created. The macro simply has the user draw the object then converts that object to a viewport. First, we will look at the command string of all three then point out how to make sure these are created on the right layer. There is one thing different between the three, can you see it?

Table 28 - Create a Circular Viewport
^C^C_circle;\_mview;o;l

Under this section we show three “\” forward slashed. It only takes 2 picks to make a circle and a rectangle for the ellipse we need three therefore we have three pauses for user input as shown below

Table 29 - Create an Elliptical Viewport
^C^C_ellipse;\_\_mview;o;l

Table 30 - Create a rectangular Viewport
^C^C_rec;\_mview;o;l;

**Bonus tip:** For the viewport we could put in the macro to create a layer to put that on but why not just add it to the tool palette settings; which is one advantage to using tool palettes. Create a layer in your current drawing as shown in Figure 18 and 18a and set the properties, for the viewport we do not want that to plot.

Figure 18

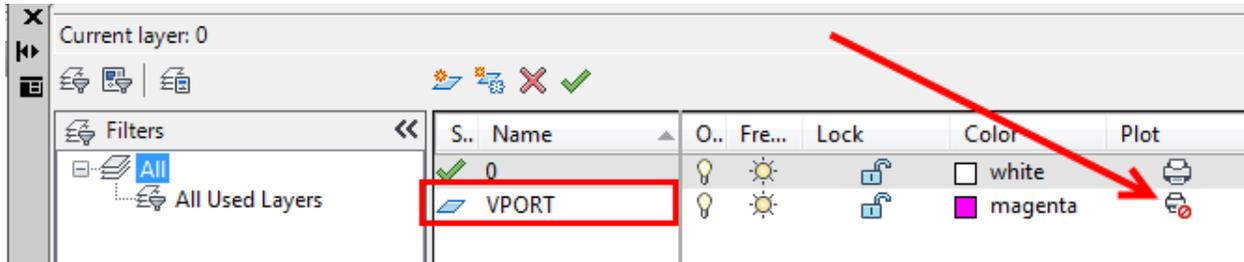
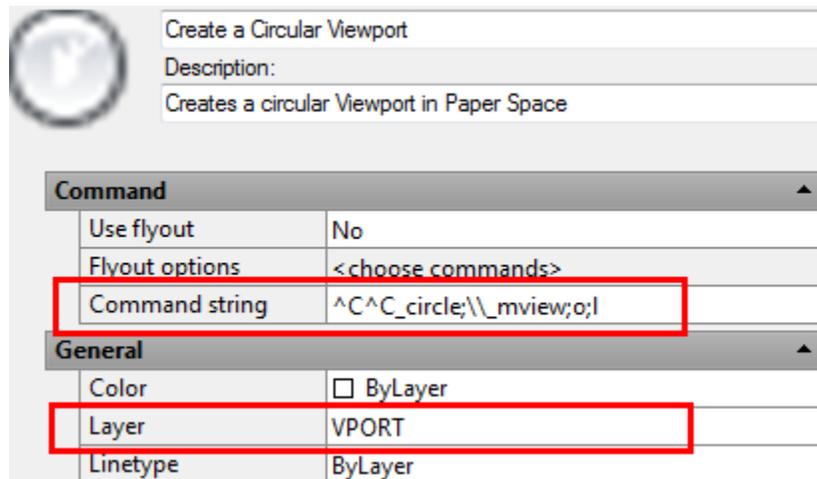


Figure 18a



### Altering the UCS with Macros

Originally I was going to leave these out but they seem to be rather popular with some of my coworkers. The UCS is the active coordinate system that establishes the XY plane (work plane) and Z-axis direction for drawing and modeling. Control the UCS origin and orientation to make drawing more convenient as you specify points, enter coordinates. Many users rely on the UCS for 3D but it can be helpful in 2D as well. I wanted to give credit to Michael Beal and his book "The AutoCAD® Workbench". Both of these macros can be found in his book as well. Michael can be found at [www.cadtutor.com](http://www.cadtutor.com) and you can purchase the book through his site. Michael's book provides hundreds of tips and tricks including some macros in there to help increase productivity.

Macros to align the UCS to plan then back again.

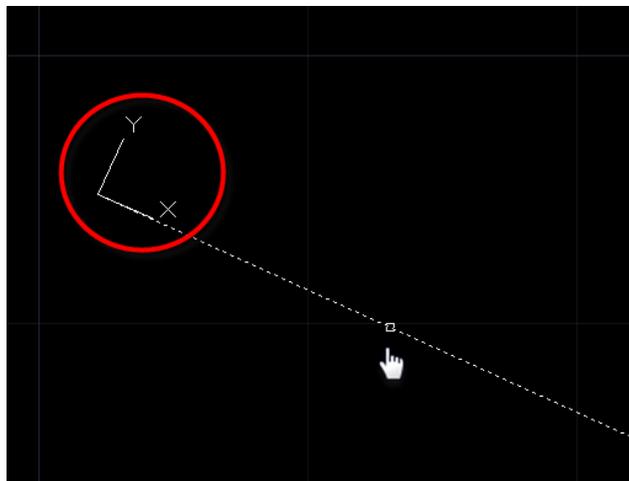
 UCS Object View

 UCS Back (WORLD)

Table 31 - Align the UCS to an Object
<code>^^C_UCS;_OB;\_PLAN;;</code>

The first of these two lets you select a line then rotates the drawing to plan view. Notice in Figure 19 below when I select the line the UCS icon appears with the new X and Y axis.

Figure 19



The next one simply resets the UCS back to world.

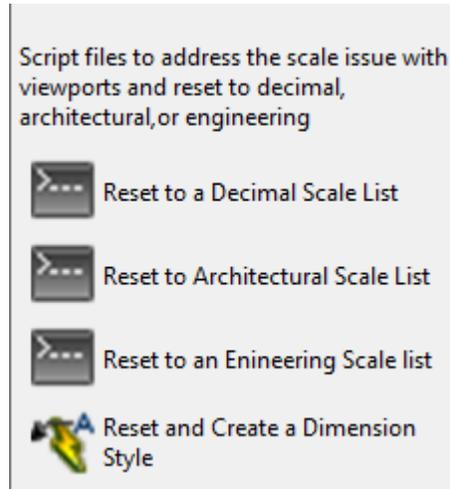
Table 32 - Reset the UCS back to World
<code>^^C_UCS;_W;\_PLAN;;</code>

**Script files vs. Macros**

A script is a text file with one command on each line. Similar to a macro except that script files are saved outside of the drawing environment. You can invoke a script at startup, or you can run a script during a work session by using the SCRIPT command. You create script files outside the program using a text editor (such as Microsoft® Windows® Notepad) or a word processor (such as Microsoft Word) that can save the file in ASCII format. The file extension must be .scr. Like macros a script can execute any command at the command prompt except a command that displays a dialog box. Our next section covers scripts to reset the scale list and also place in a macro to create a dimension style. The first three simply run a long script and the Reset the Dimension settings we will place that scrip in a long macro to complete the same task.

For the first three we are simply going to run the script from a macro.

Have you ever entered a drawing and looked at the scale list and it's so large it goes off the screen? These script files will do the trick. Script files need to have the filedia system variable turned off to start then find the script and run it.



**Table 33 - Load a Script file to reset the Scale List**

```
^C^C_filedia;0;_SCRIPT;"C:/AU2013/Mighty Macros/Class Files/Script files/AU-scale-dec.scr";filedia;1
```

Let's take a look at the actual script file as shown below in Figure 20a, 20b and 20c (portions). We are only going to show a few lines of each section since scripts are line by line (it would take up way too many pages!!) then we will show you how to use those same commands in a macro. The script file first deletes all of the default scales (Figures 20a and 20b) then adds the scales from the selection as shown in Figure 20c. Just type the commands in at the command prompt to see the results.

**Figure 20a**

```
-scalelistedit
Reset
Yes
Exit
;
Delete the decimal scales
-scalelistedit
delete
1:1
delete
1:2
delete
1:4
delete
1:5
delete
1:8
delete
1:10
delete
1:16
```

**Figure 20b**

```
Delete the Architectural scales
;
delete
1/128" = 1'-0"
delete
1/64" = 1'-0"
delete
1/32" = 1'-0"
delete
1/16" = 1'-0"
delete
3/32" = 1'-0"
delete
1/8" = 1'-0"
delete
3/16" = 1'-0"
delete
1/4" = 1'-0"
delete
3/8" = 1'-0"
delete
```

**Figure 20c**

```
;This section will add the decimal
scales to the list.
;
-Scalelistedit
Add
1=10
1:10
Add
1=20
1:20
Add
1=30
1:30
Add
1=40
1:40
Add
1=50
1:50
Add
1=60
```

Notice the path on the next two macros, we are simply going to point to another script file to change the units of the drawing (i.e. dec=decimal, arc=architectural, eng=engineering). I have included these script files with the class under additional class materials.

**Table 34 - Load a Script file to reset and load an Engineering Scale List**

```
^C^C_filedia;0;_SCRIPT;"C:/AU2013/Mighty Macros/Class Files/Script files/AU-scale-eng.scr";filedia;1
```

**Table 35 - Load a Script file to reset and load the Architectural Scale List**

```
^C^C_filedia;0;_SCRIPT;"C:/AU2013/Mighty Macros/Class Files/Script files/AU-scale-arc.scr";filedia;1
```

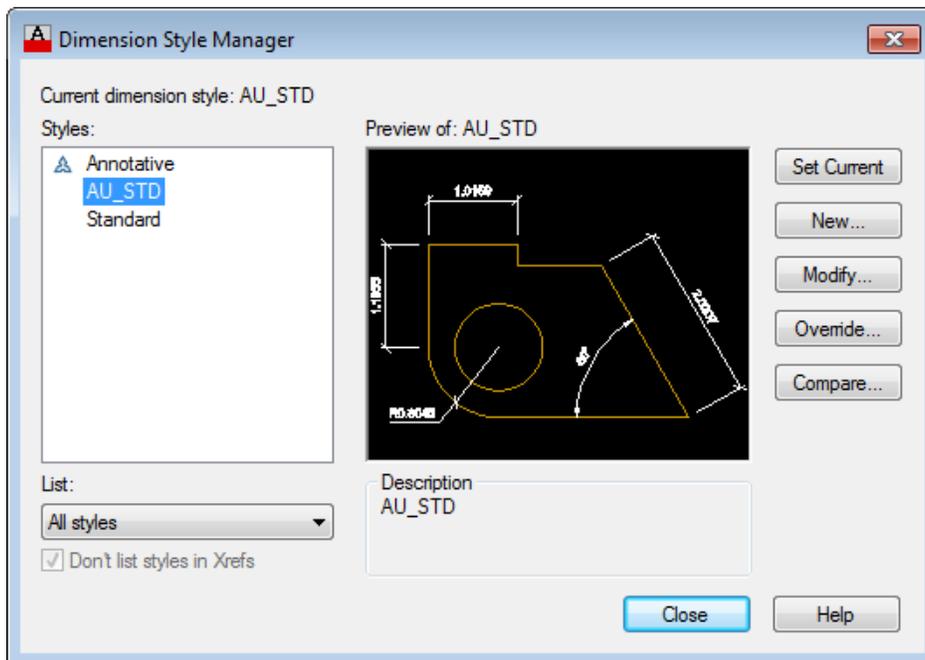
Let's see how this would look if we took the same concepts and or commands and put them on a macro string. This time we will set our Company's dimension standards and load up the ddim command at the end just to show the results of the new style created.

**Table 36 - Set Dimension standards and set the style with a macro**

```
dim;dimasz;.14;dimtsz;.05;dimexe;.05;dimcen;.05;dimgap;.06;dimdli;.08;dimdle;.00;dimexo;.12;dimtp;.06;dimtm;.06;dimdle;.00;dimtxt;.10;dimtad;1;dimtih;0;dimclrt;bylayer;dimclre;bylayer;dimclrd;bylayer;dimsho;1;dimtofl;1;dimaso;1;_sav;AU_STD;Y;exit;redraw;ddim;
```

Figure 21 below shows the new style AU\_STD created in our current drawing with all of those variables set.

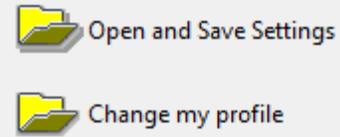
**Figure 21**



**Macros to access Folders and Options**

Did you know there are command that begin with a “+” ?  
 Simply type + at the command prompt and take a look at autocomplete and some of the command that are listed. In this section we are going to create a macro to access some settings with the +options command. The last two on this section we are going to access shell command to move out to the drawing folder as well as our automatic backup folder.

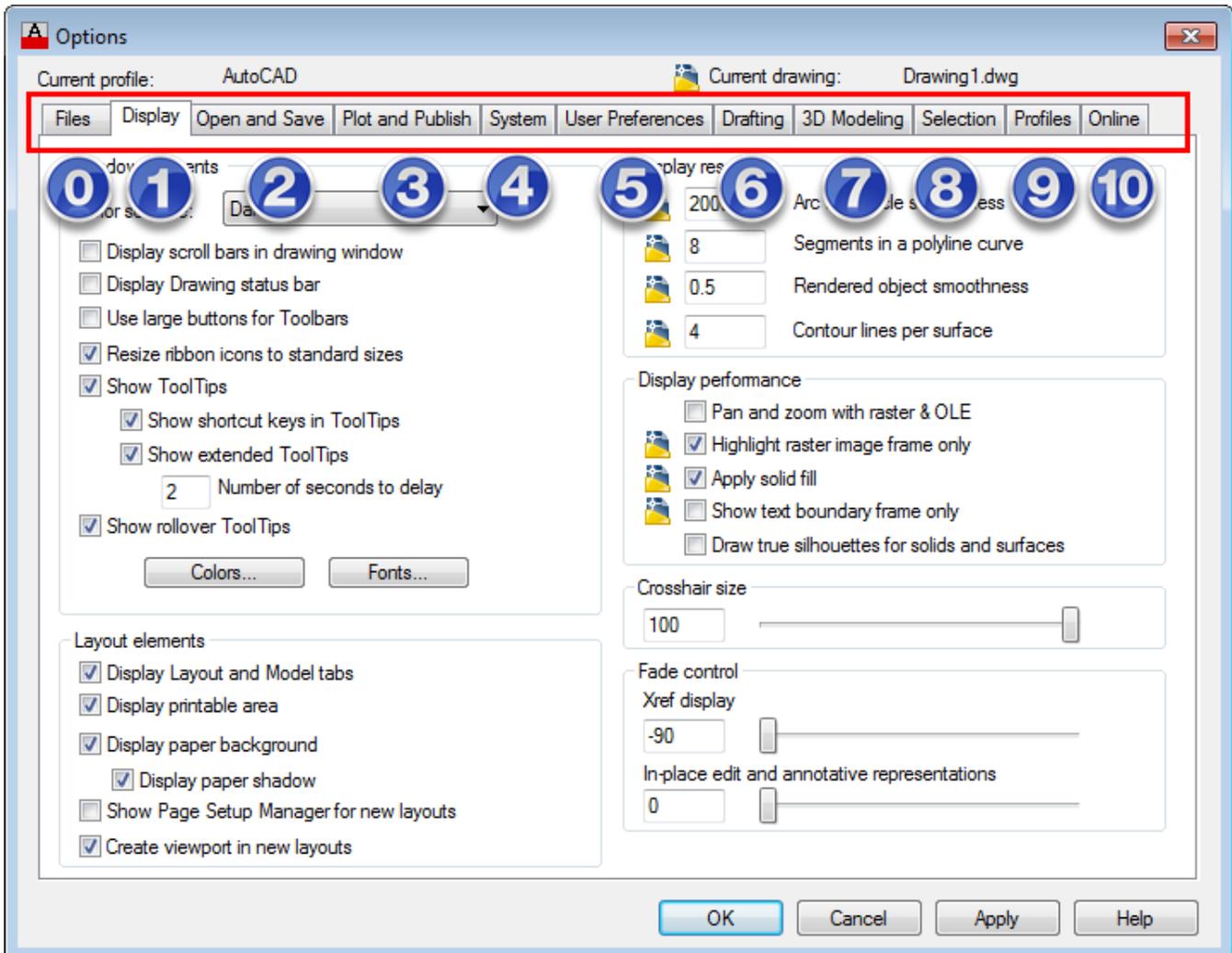
Using the +Options command to access the tabs and get to settings quickly



<b>Table 37 - Accessing Open Save Tab using Options</b>
<b>^C^C+options;2;</b>

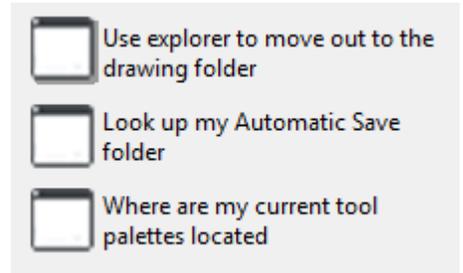
Take a look at the options dialog box as shown below in Figure 22. Notice in the macro above we precede the options command with a “+” then the number 2. That number indicates the tab you would like to open up.

**Figure 22**



### Accessing Important Folders

For many years I have been using a lisp file to access my current project folder. I simply type expl at the command prompt and AutoCAD opens up explorer and takes me where I need to go. I bet it's safe to say at one time we all need to look at the Automatic save folder as well or maybe the tool palette path. With lisp this can be done and it also can be done by using a macro. Let's first take a look at the lisp file shown in Figure 23 that performs this function then the macros following in Tables 38, 39 and 40 (There really is not much of a difference).



I attempted to make the names make some sense and keep the keys close together for quick access. So it takes me 4 keyboard clicks to punch this in and with a macro I can just select. Quicker? You be the judge.

Figure 23

```

;;; Sam Lucido - Access folders from lisp
;;; Current drawing/project folder, Automatic Save, Tool Palette
;;;
(defun c:expl ()
  (startapp "explorer" (strcat "/n,/e," (getvar "dwgprefix")))
  (princ)
)
(defun c:exps ()
  (startapp "explorer" (strcat "/n,/e," (getvar "savefilepath")))
  (princ)
)
(defun c:expt ()
  (startapp "explorer" (strcat "/n,/e," (getvar "_toolpalettepath")))
  (princ)
)

```

<b>Table 38 - Bring up the current drawing folder</b>
<b>^^^P(startapp "explorer" (strcat "/n,/e," (getvar "dwgprefix")))</b>

<b>Table 39 - Bring up the current drawing folder</b>
<b>^^^P(startapp "explorer" (strcat "/n,/e," (getvar "savefilepath")))</b>

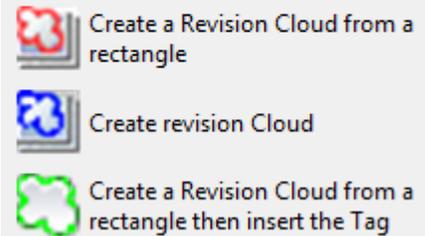
<b>Table 40 - Bring up the current drawing folder</b>
<b>^^^P(startapp "explorer" (strcat "/n,/e," (getvar "_toolpalettepath")))</b>

## Revision Clouds

During the course of 2013 I was asked to assist at the end during the as-built drawing phase of a large design project. The chief engineer on the project ask me “Sam, why are the revision clouds all different sizes and different line weights on the drawing”? With several people working on the same design and some in model space some in paper space we did not have a consistent result. This is where these following macros came into play.

The code below starts out the revcloud command then we call on a lisp function. AutoCAD looks to see if we are in a layout tab or in model space then takes the dimscale variable and multiplies that by .3 which is our standard length for the revision cloud. Keep in mind if the dimscale variable is not set then the model space calculation will be off as well. We typically put all our revclouds in paperspace at a .3 arc length and this way we can control the value if someone uses model space. Just don't forget to lock those viewports!

Revision Clouds. Several different options on creating and using the revcloud command on design projects



**Table 41 - Create a Revision Cloud**

```
^C^C_revcloud;a;(if (= (getvar "CTAB") "Model") (* (getvar "dimscale") 0.3) 0.3);;s;n;layerp;
```

The second version of this (my favorite) is to draw a rectangle first then convert that rectangle to a revcloud.

**Table 42 - Create a Revision Cloud from a Rectangle**

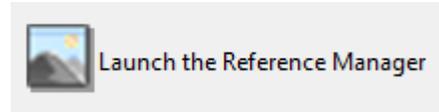
```
^C^C_RECTANGLE;\_revcloud;a;(if (= (getvar "CTAB") "Model") (* (getvar "dimscale") 0.3) 0.3);;S;N;O;;L;N;layerp;
```

Lastly, we draw the rectangle convert to a revcloud then insert a revision marker. For this version notice we created a new layer named “rev-marker” so it would be independent of the revcloud.

**Table 43 - Create a Revision Cloud from a Rectangle and insert a Marker**

```
^C^C_RECTANGLE;\_revcloud;a;(if (= (getvar "CTAB") "Model") (* (getvar "dimscale") 0.3) 0.3);;S;N;O;;L;N;_insert;"C:/AU2013/Mighty Macros/Class Files/01 Revision Cloud Marker.dwg";^C^C_layer;m;Rev-marker;c;2;;;-insert;revm;s;(if (= (getvar "CTAB") "Model") (* (getvar "dimscale") 1));\;;^C^Clayerp;;
```

**Launch the Xref Manager**



**Figure 24**

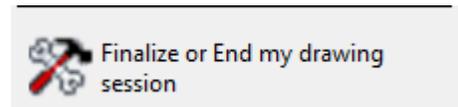
How can you update and re-path drawing reference files (*other drawing files, fonts, or plot configuration files*) after a path has changed? The reference manager makes it easy to add drawings and see where reference files were found or not. You can search, select all, invert selections, edit selected paths and even use find and replace paths in batch. The reference manager as shown in Figure 24 provides tools to list referenced files in selected drawings and to modify the saved reference paths without opening the drawing files in AutoCAD. With Reference Manager, drawings with unresolved references can be easily identified and fixed files required. Reference Manager is a stand-alone application that you can access from the Autodesk program group under Programs in the Start menu (Windows) as shown. I want the ability to open this program from within AutoCAD.



The macro below uses the shell startapp command to look for the reference manager executable file to load. Once found; loads the file and the reference manager will pop up during your current drawing session.

<b>Table 44 - Launch the Reference Manager</b>
<b><code>^C^C^P(startapp "adrefman" (strcat "C:/Program Files/Autodesk/AutoCAD 2014/AdRefMan.exe"))</code></b>

**End my Drawing Session**



Good file management is to clean up your drawing when you are finished; leaving things in a good way so the next person who works on the project can open up a clean drawing and have no problems. This macro is one of my favorites; again I started with a lisp file then realized I could place the same code on a macro so the user would not have to load the lisp file. The macro does all those file management tasks you need to clean up your drawing. Set the layer to 0, run an audit, purge the drawing, zoom to the extents, then run a qsave. As with some of the other macros I run a lisp function to alert the user that the drawing has been finalized and it's OK to quit. It is important to run the purge command after the audit, if the audit finds anything that does not belong it will be purged.

<b>Table 45 - Finalize my Drawing Session</b>
<b><code>^C^C_-layer;s;0;;_-audit;y;-purge;a;;n;_zoom;e;qsave;(alert (strcat "Drawing Saved...you may now EXIT"))</code></b>

## Linesanity 2: Linetypes in Macros

My class at AU2012 was named **Linesanity: A Journey from Simple to Complex Linetypes**. While writing this paper I notice how I could improve on some of the macros I used from last year or maybe I just wanted to teach the class again! We loaded our text styles from a template and while doing that, we all know, if you don't have the template the style is not there and the linytype will not work. We are going to review the steps to see how the linytype macro works and how we can improve it. Let's review steps 1 through 5 from Linesanity last year. Steps 1 through 5 are the ones we are going to change to get our linytype to work. Figure 24a and 24b shows the images into two parts so it would be clear on what to edit in the palette.

1. The image. This can be done through the standard cui button images or you can simply take a screen shot (as I have done) of your linytype and place in a folder to reference.
2. Type in the name of your linytype. This is what you will see on the palette next to the image.
3. Enter a description for the linytype. It is very important to inform the user of how this linytype is to be used or what it is used for. **Do not just ignore this section.**
4. This is the command string that we will use to enter our linytype. We will break this down on the next page.
5. The layer we would like the linytype to be placed on. Create a layer (in your current drawing) and select this from the pull down. It will now be created each time you use the linytype.

Figure 24a

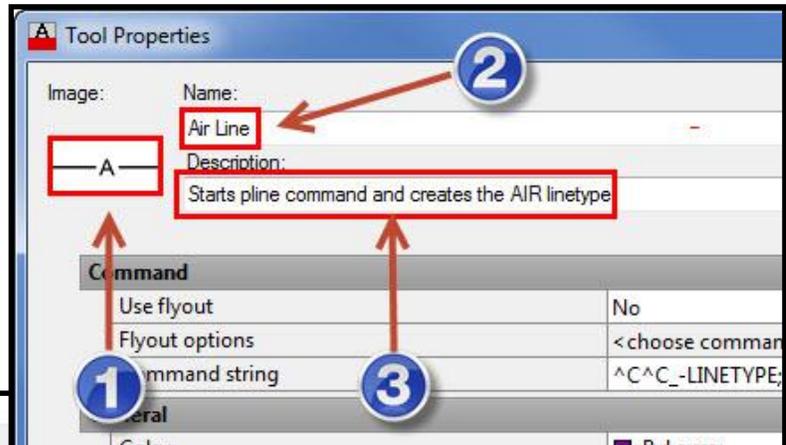
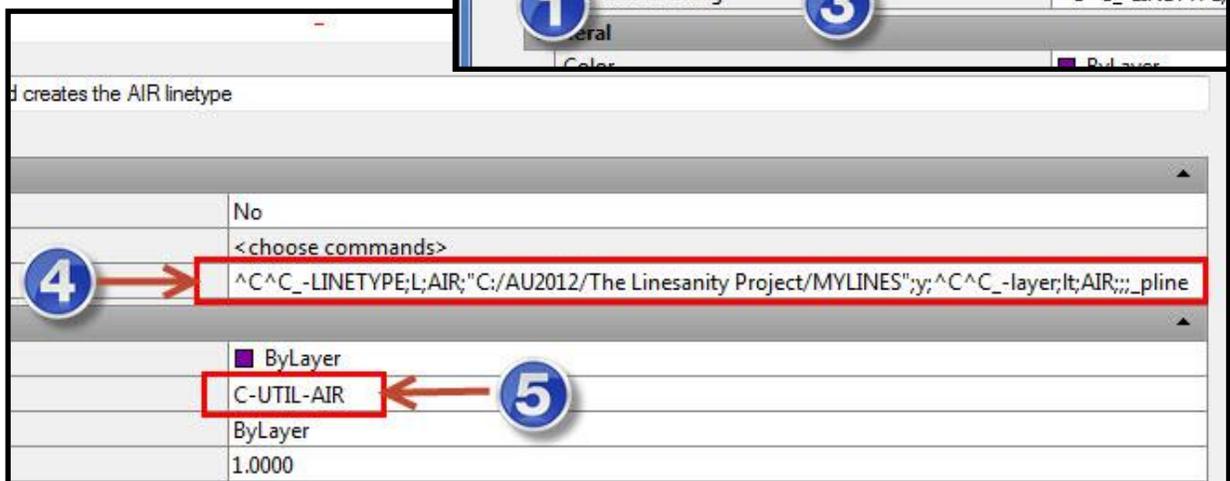


Figure 24b



All of these concepts still apply to our new macro with the exception of loading the style first. With that addition we do not have to be concerned about someone not having the style loaded or a template file needed. For this long macro we will take a look at the code, line by line, and see the results shown below in Table 46

<b>Table 46 - Linesanity! Load my linetypes with a macro</b>	
<b>Command</b>	<b>Description</b>
<b>^C^C</b>	Cancel any previous action
<b>_-STYLE</b>	Create the style or verify that the style is in the current drawing (this has been added since Linesanity at AU2012)
;	The semi colon represents a return on the keyboard
<b>linetypes</b>	Create a new unique style named "linetypes"
;	The semi colon represents a return on the keyboard
<b>Arial</b>	Set the font to true type Arial
;	The semi colon represents a return on the keyboard
<b>0;1;0;n;n;</b>	This next sequence I grouped together. These are the parameters that set the style. Height, Width, Oblique angle, backwards, upside down. Type -style at the command prompt and follow the sequence.
<b>_-LINETYPE</b>	The command to start loading the linetypes. The hyphen will disable any dialog boxes
;	The semi colon represents a return on the keyboard
<b>L</b>	Selects the option "Load"
<b>AIR</b>	The name of the linetype to be loaded
;	The semi colon represents a return on the keyboard
<b>.....path/Mylines.lin</b>	Linetype file name and path. Remember the slash will be a backward one instead of the normal forward path. A forward slash will indicate a pause for user input and will cancel the command string.
;	The semi colon represents a return on the keyboard
<b>y</b>	Reload the linetype if it already has been loaded
;	The semi colon represents a return on the keyboard
<b>^C^C</b>	Cancel any previous action
<b>_-LAYER</b>	Selects the layer command

```
^C^C_style;linetypes;arial;0;1;0;n;n;_-LINETYPE;L;AIR;"C:/AU2012/The Linesanity
Project/MYLINES.LIN";y;^C^C_-layer;lt;AIR;;;_pline
```

## Macros and the CUI

This was not one of the objectives but I wanted to cover the CUI briefly due to the hundreds of commands that contain macros and there is a section in there that says Macro, it just seemed necessary. We are going to create a panel then place that panel on a tab. Type CUI at the command prompt to bring up the CUI editor. Move to the Ribbon section and right-click panel to create a new panel as shown in Figure 25a. Name your new panel Mighty Macros as shown in Figure 25b

Figure 25a

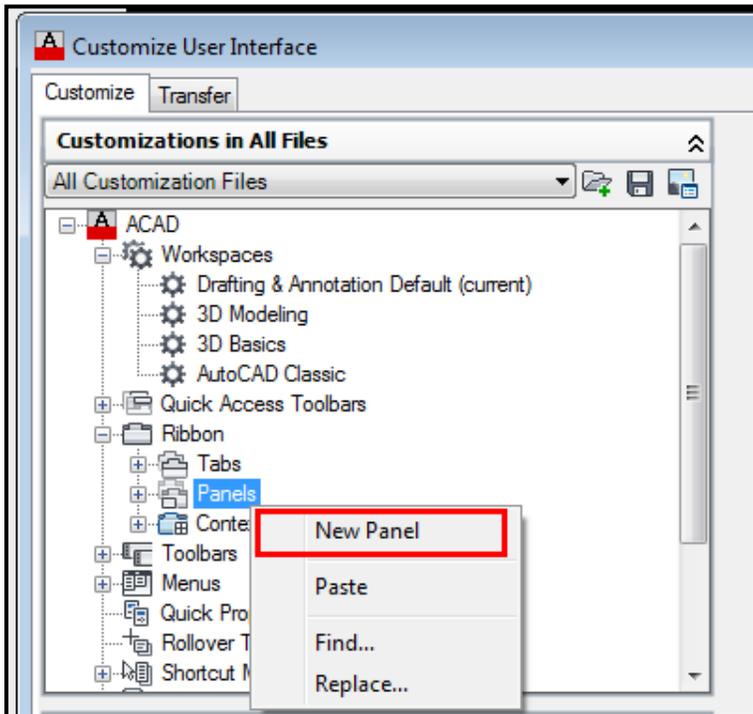


Figure 25b

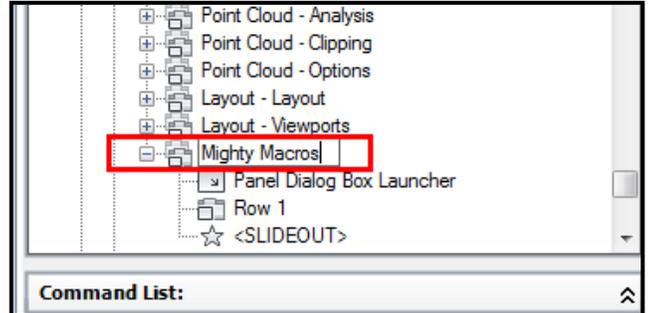
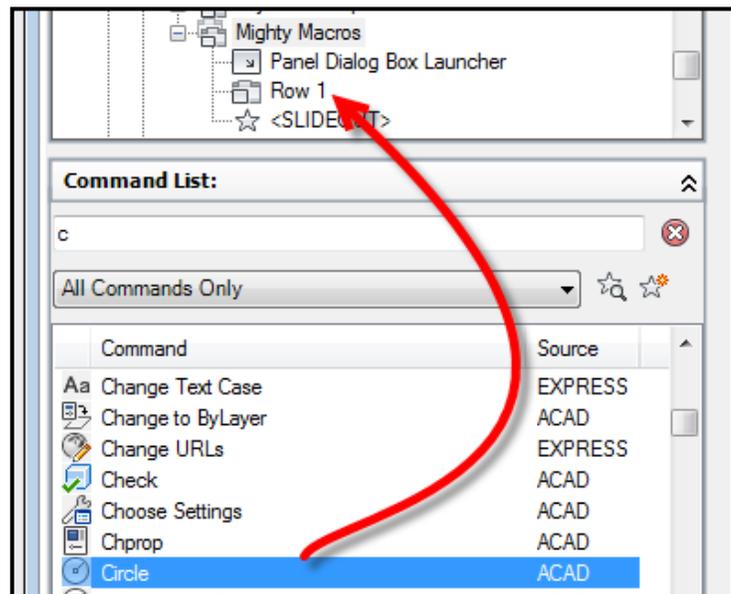


Figure 26

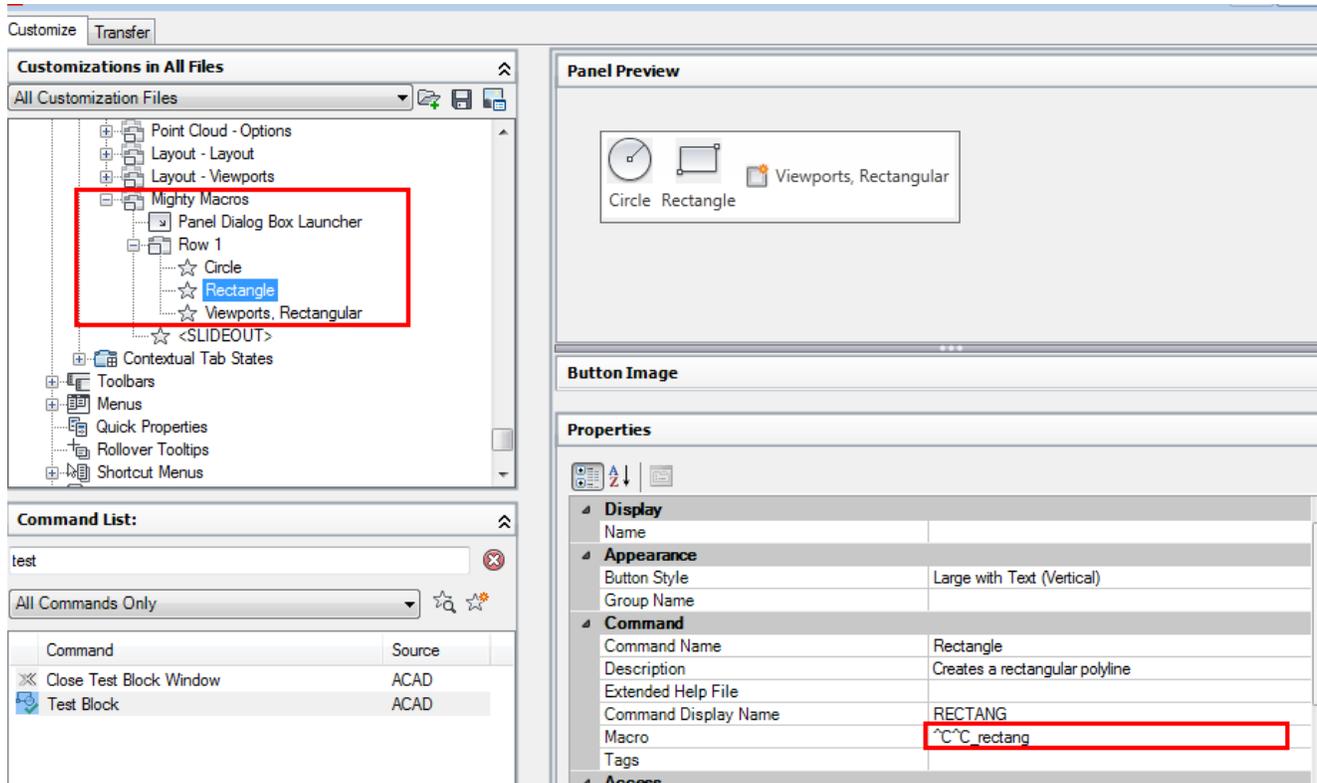
Move to the command section of the CUI as shown to the right in Figure 26. We are going to add commands to the panel or our custom macros. Move down the command list and search for a command and drag it into Row 1 (we used circle as the example). For this example we are simply going to add three existing commands to the panel.

In Figure 27 we are going to first place our commands on the Mighty Macro panel then copy that panel to the Mighty Macro Tab. We will explain how this is accomplished in the next step.



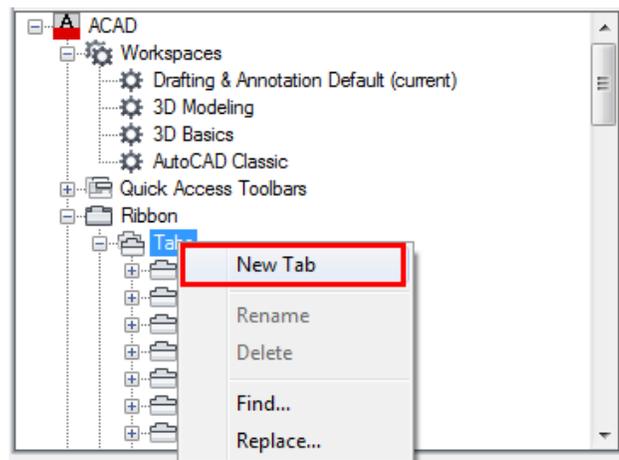
Three commands randomly picked from the command list as shown in Figure 27. We dragged each of those commands up to Row 1 as shown.

Figure 27



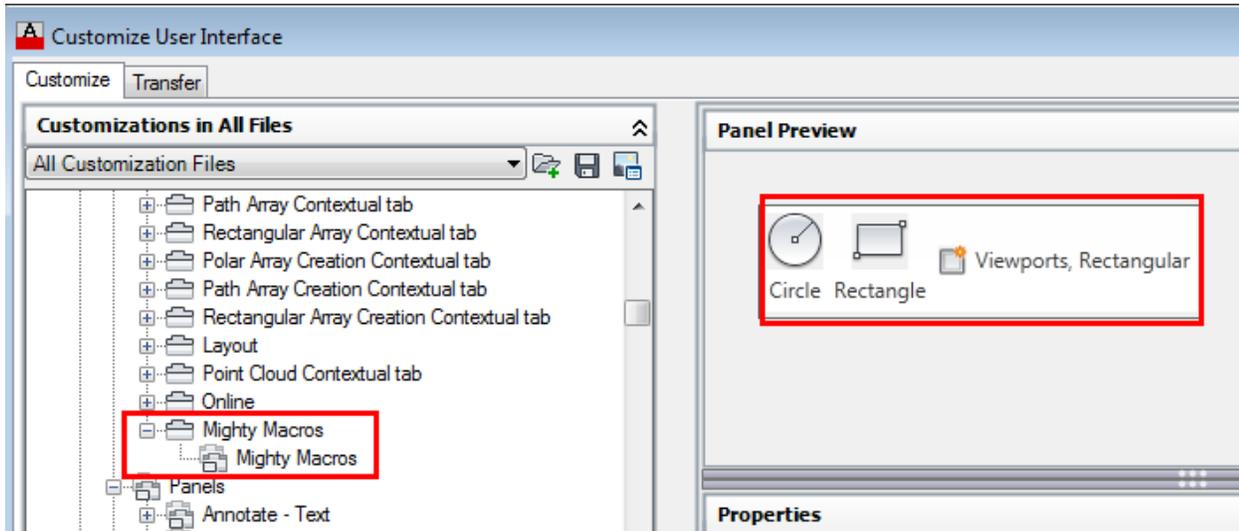
Go back up to the Ribbon and let's create a new tab and call it Mighty Macros as shown in Figure 28.

Figure 28



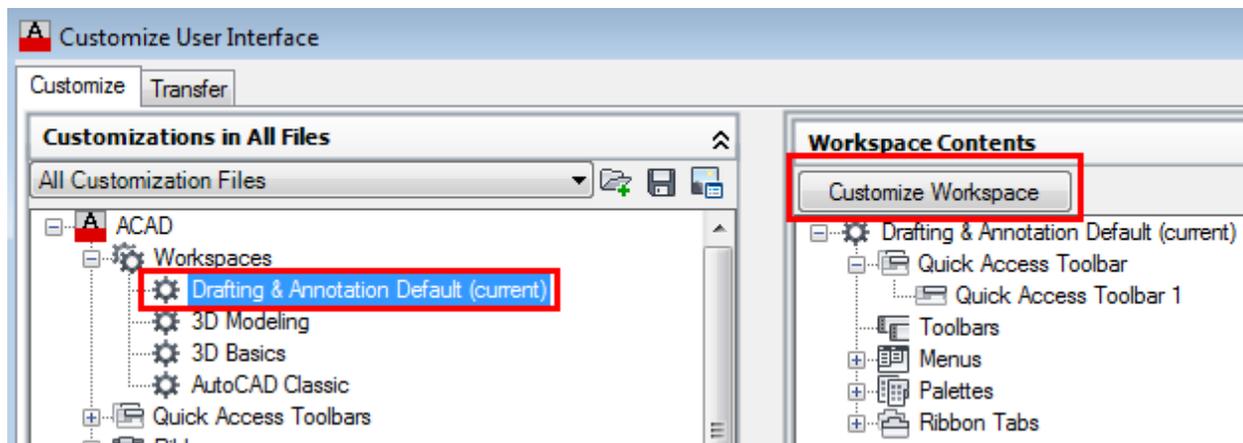
Next we need to add that new panel to our tab. We are going to right click our panel and paste into the new tab as shown in Figure 29 (otherwise you have to scroll up through many panels). It seems like we are duplicating our effort but we need to create the panel first before placing on the tab.

Figure 29



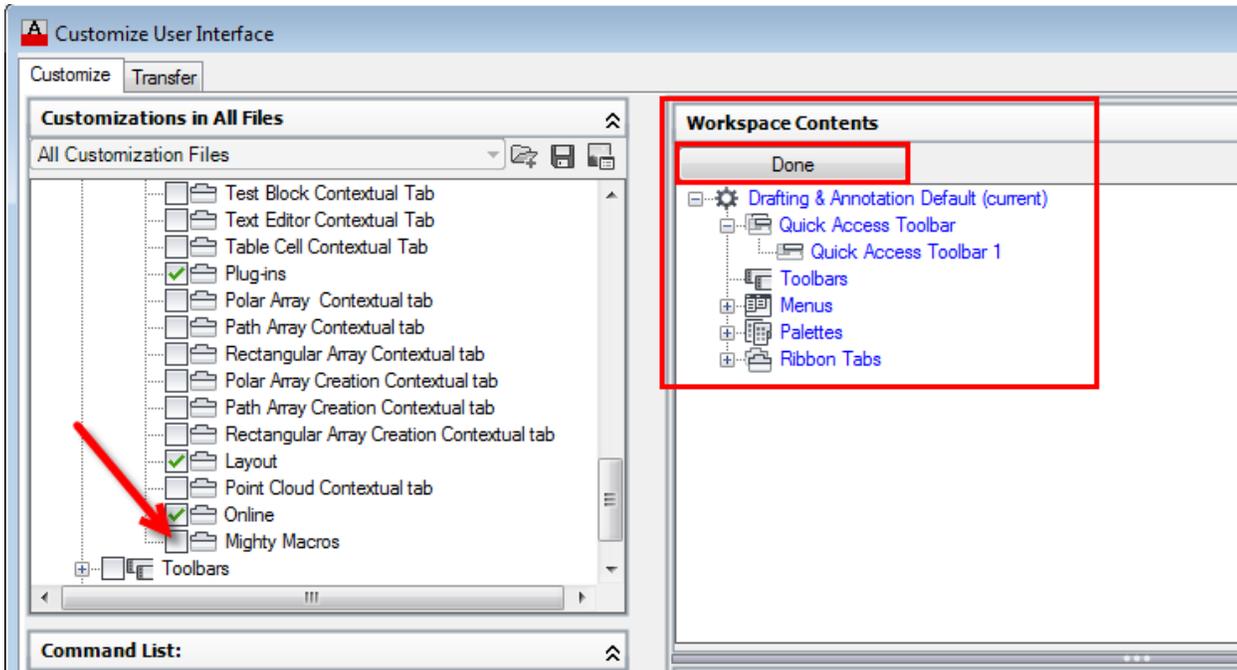
Lastly, we need to add the panel to our workspace. Click on your workspace and hit customize workspace as shown in Figure 30 below.

Figure 30

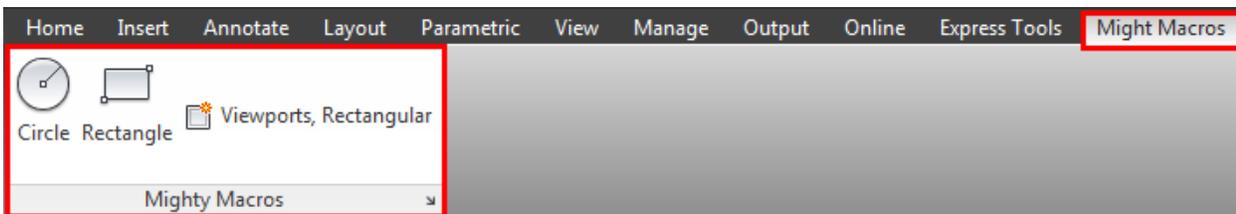


Almost done! Simply check the box next to the Mighty Macros as shown in Figure 31 and hit Done.

Figure 31



Take a look at the Ribbon and now you have your new Tab showing the new panel as shown below. As I mentioned earlier this was not part of the objectives but I felt it necessary to cover the material. Many commands in AutoCAD contain macros and this way is an easy way to create a custom tab for the commands you use most.

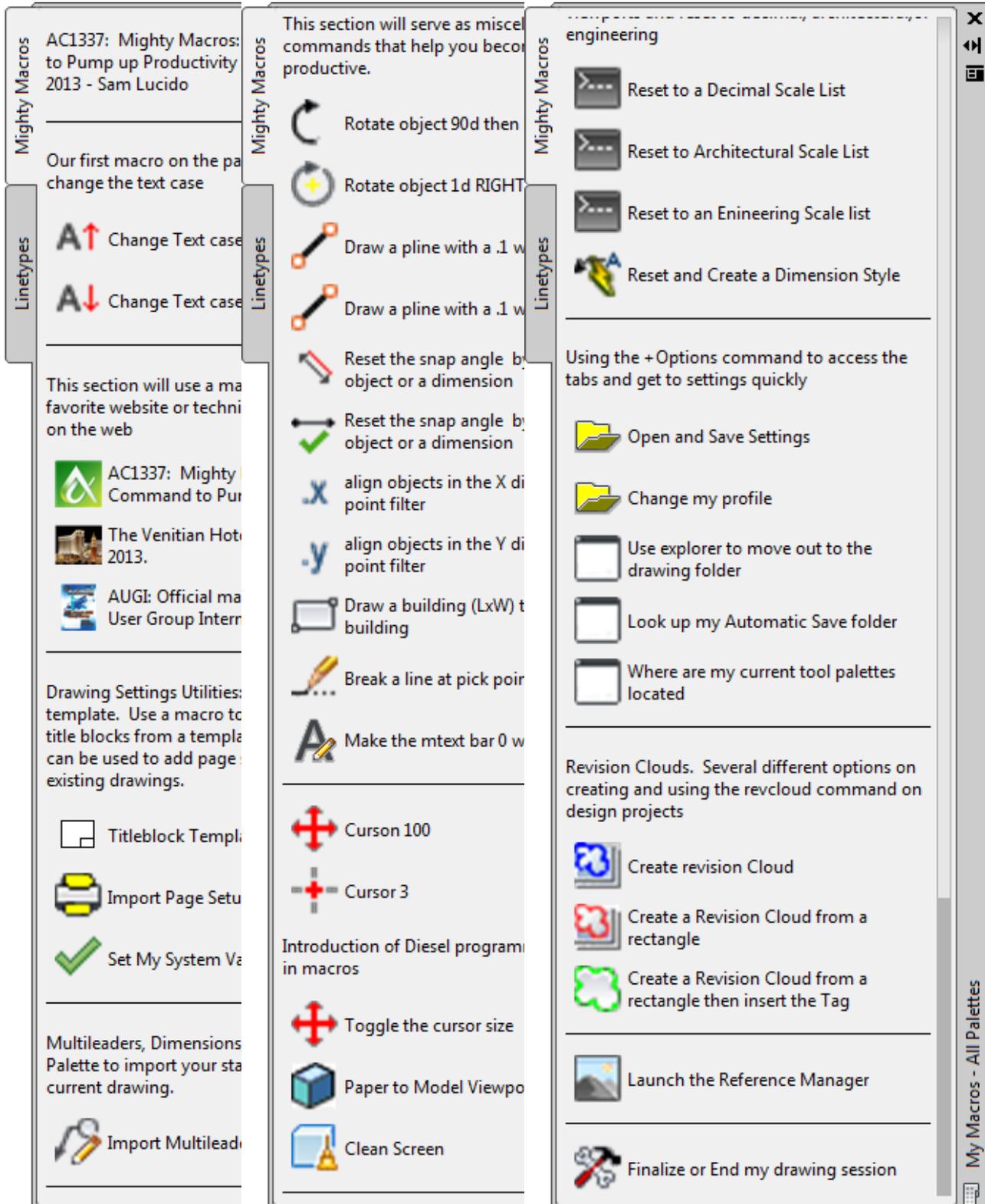


I would recommend creating a partial CUI then loading that each time. It would be much easier to customize and transfer to other users or to yourself when you upgrade. Two additional classes you may want to take a look at our [“AutoCAD Customization Boot Camp”](#) where [Lee Ambrosius](#) explains this at the end of his class and [“Making the CUI work for you”](#) where [Jeanne Aarharus](#) shows you how to create a custom CUI for your company standards.

**The End Result**

We now have built the tool palette containing over 50 macros to be used each and every day to increase productivity. Figure 32 shows some of the commands we covered during this class and throughout this document.

**Figure 32**



## Conclusion

A macro (in many different programs) can be defined as a way to automate a task that you perform repeatedly with more than one command or keystroke. In my opinion macros are the easiest way to program with little knowledge of programming. In AutoCAD® macros can be shortcuts to a series of commands to help make the process of design more efficient. In this class we used the Action Recorder, Tool Palettes, and Command strings to create macros. All of these tools help us as professionals become more efficient and perform our job at a high level. Think about workflow and what you do every day at work. Can you take four clicks and turn those into one? Think about how many times you apply that task in one day, then in one year. Being efficient and productive will only make you more valuable and provide profit to the company you work for.

Time is money and you have just learned how to save a little of both.



**Thank you!**

*“Well I tell them there’s no problem, only solutions” John Lennon*

*Enjoy the rest of your time at Autodesk University 2013!*