# KB

(KeyBoard)

Bill Mahaffey W6EET
w6eet@cox.net

KB is a keyboard program written to interface to the WinKey keyer.
See http://k1el.tripod.com/wkinfo.html for information on the WinKey keyer.

KB also optionally interfaces to MacLoggerDX for logging purposes.
MacLoggerDX is highly recommended even if you don't use this program!
See http://www.dogparksoftware.com/MacLoggerDX.html

KB provides the following major features:

Programmable or "Action" buttons.
(Includes a contest sequence number feature)

Interfaces with MacLoggerDX

Virtually unlimited "memories."

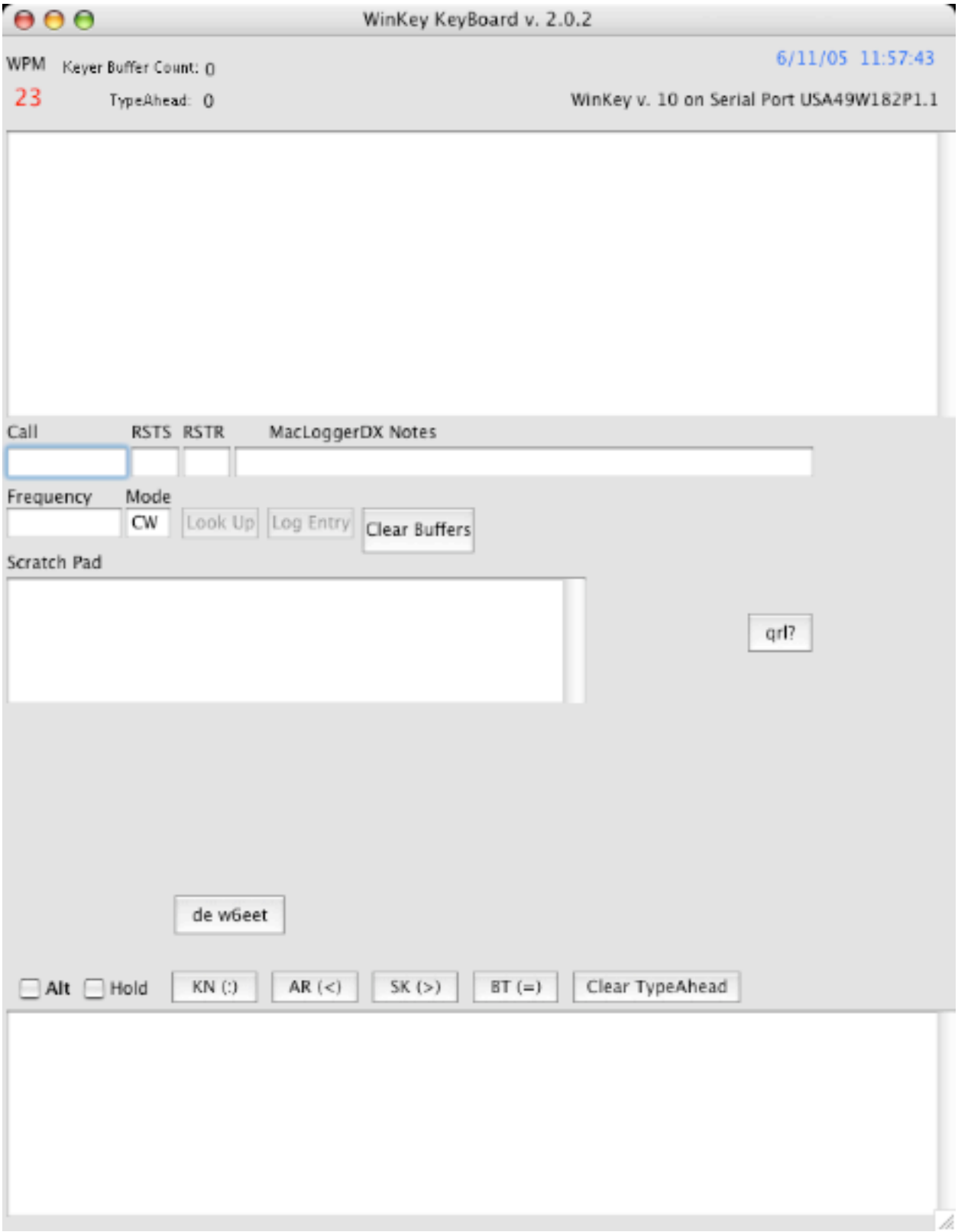
Multiple typeahead buffers.

A text window available for those that like to use a "mill" for copying with the capability of saving the entered data.

Key Mapping

Scripting capability
(See the "Script capability description" document)

Extensive flexibility for color selection.




The above is the window you see during normal operation before user color selection.

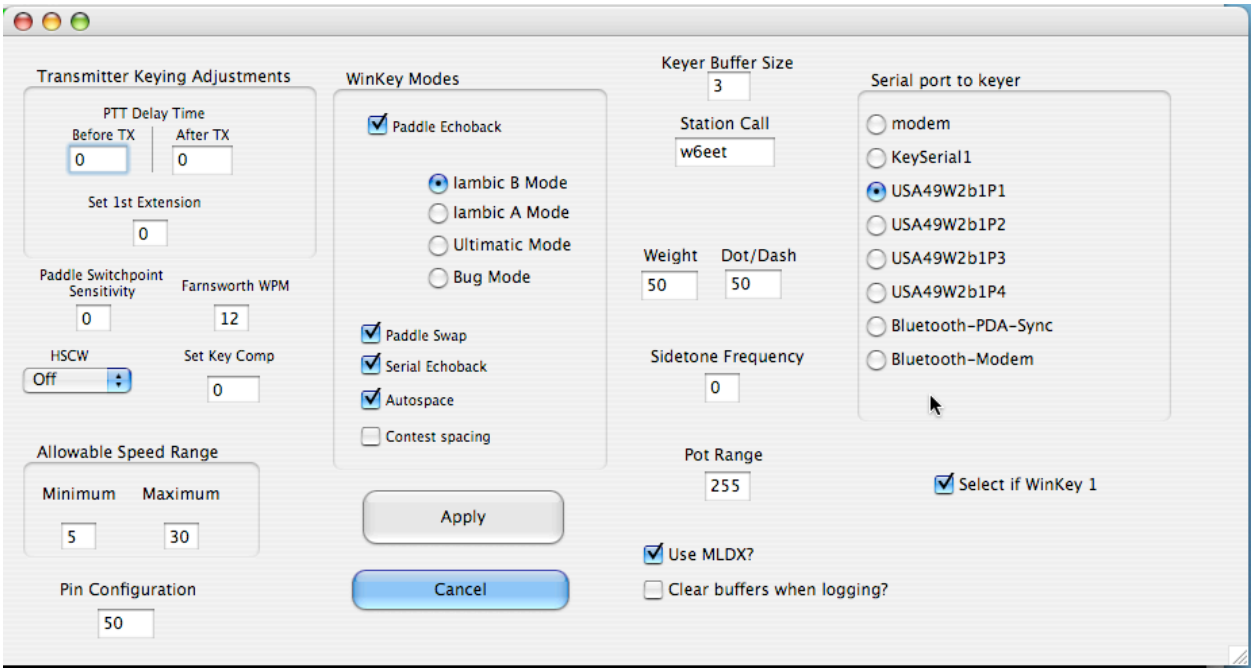

The above is the same window after a user went hog wild with color selection. Your taste may vary.

One major point is the color selections are saved in the preferences file and are thus semi-permanent.

# Getting Started

If you are going to be using  MacLoggerDX please make sure that you have configured MacLoggerDX correctly. If you are able to successfully use MacLoggerDX manually for logging and call sign lookups then KB should be able to interface with it with no problems.

After the program has started select the preferences menu item and the following window will open:




Much of this window is best described in the WinKey documentation and I refer you to that as I certainly could not do it justice. The only comment I have to make about the WinKey fields are that “Serial Echoback” cannot be deselected as I need that to keep track of what the keyer has sent so I can keep the buffer count correct. There are some fields, however, that relate only to KB and those follow:

The most important, of course, is the selection of the serial port that the keyer is connected to and that must be selected of course. Secondly, the original WinKey required the use of DTR, and WinKey 2 does not work if DTR is selected, so set “Select if WinKey 1” accordingly.

“Keyer Buffer Size” specifies how many characters to buffer in the keyers memory. We want to keep some number of characters in the buffer for smooth sending. A tradeoff must be made here between keeping the keyer “fed” so it has something to do and having what you have entered available for error correction. If you make a typo and that character has been sent to the keyer you won’t be able to correct it, so you will want the buffer size small enough to keep the data on this computer long enough for you to recognize that an error has been made and to correct it, but large enough so that the keyer isn’t wanting for something to do. This is going to vary from person to person of course, but I have found that I am most comfortable with a buffer size of 6, and that 3 really doesn’t cause any problems. If your computer can get very busy doing other tasks at the same time a larger buffer size might be needed. If your typing (like mine) is error prone then you will want a smaller buffer size to keep the data around long enough for you to correct it.

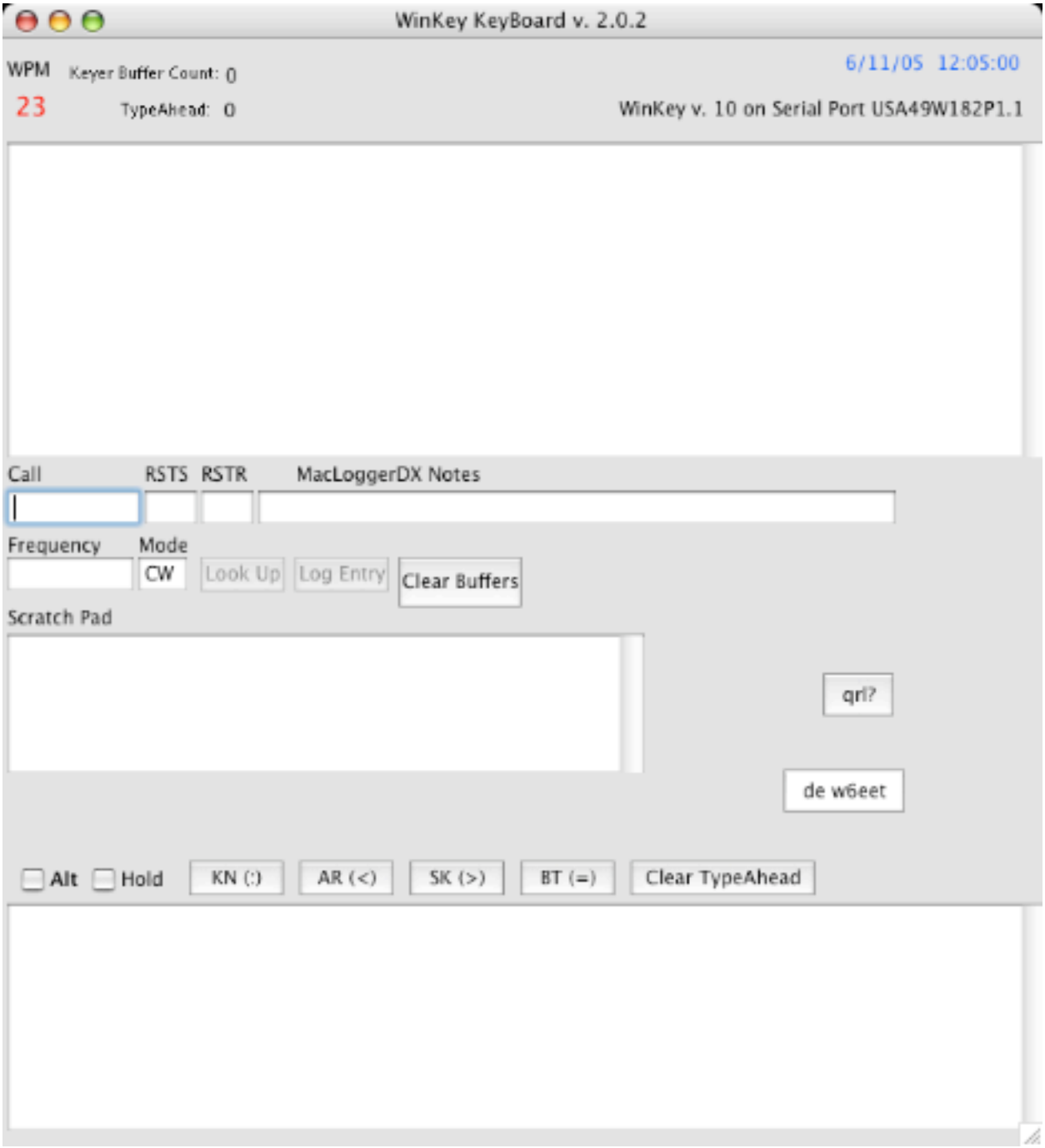
“Station Call” is your Stations call sign. During a QSO and after you have entered the other stations call into the field designated for it (described later), a command-B will send something like the following:
“ xx6xx de mycall “
This is often used at the beginning and/or the end of a transmission so it is a keystroke saver. Note that a space is at the beginning and the end.

“Use MLDX?” specifies if you want to interface to MacLoggerDX. If selected, then four buttons are enabled during a qso to optionally do a lookup (if enabled for MacLoggerDX) and to log the QSO. More on this later. In addition, “Clear buffers when logging” optionally clears all buffers after sending the log to MLDX.

After making your selections click on APPLY.

If you have selected correctly, and if the keyer is in good working condition, and if the stars are aligned correctly then you will see something like what is shown on the following page.

If you indicated that you want to interface with MLDX it will be starting up now.




The blue characters at the top right is the UTC date and time.
Below it indicate the WinKey version and the serial port it is found on.
The red characters to the left indicate the sending speed set by the keyer pot.

If you do not see the WinKey version number line and you think you have selected the correct serial port, restarting KB at this time might be helpful.

# Using it

The basic operation at this point is straightforward. Select the code speed you want by turning the pot on the keyer and watching the speed in the window, then select the bottom edit field by clicking on it, make sure that “Hold” is not selected, and start typing. Assuming the keyer is correctly connected to your rig, etc., you should be transmitting.

So, with that out of the way, I will describe some of the less obvious points. I will not insult your intelligence by describing the use of the “AR(<)” button, for example.

At the top of the window are the Keyer Buffer Count and the TypeAhead Count. The total of those two indicate the number of characters to be sent, including the one currently being sent. After the keyer finishes sending a character, it sends that character back. I can thus keep track of what is being sent, yet to be sent, etc. Do you recall what I said about Serial Echoback on the preferences window?

The area below where the count fields are is where the transmitted and returned characters are displayed. I store the characters here that the keyer returns to me.

Below this area are 6 fields mainly, but not entirely, used for interfacing with MacLoggerDX, referred to as MLDX from now on.

If MLDX was selected the “LookUp” and “Log Entry” buttons are visible but initially disabled. Frequency and Mode are also made available if MLDX is selected. “CW” is the default mode. If MLDX is **not connected** to your rig enter the frequency to be logged. MLDX will obtain the frequency if connected.

The “Call” field is where the other stations call is entered. “return” must be entered to indicate that the call has been completely entered and to get the QSO start date and time. Once the call is entered command-B is setup to send the “ hiscall de mycall ” text.

“Alt” and “Hold”

Consider the following scenario...
You’re involved in a slow speed ragchew that enables you to consider carefully what you want to say during for your next transmission along with the time to enter most if not all of it prior to the other station turning it over to you. So, you have selected “Hold” so that what you are typing will not be sent, and have typed in a good deal of your experiences during your long and extensive ham radio life, when you hear the other station send something like...”i missed ur name es qth last tx, pse agn om bk”. Now what? You certainly don’t want to delete everything and retype that book again so instead you select “Alt’ and the alternate typeahead buffer is made available so that you can respond to the query. When finished you simply click on “Alt” again and you are back to the primary typeahead buffer.

Admittedly this does not occur often but when it does this feature is appreciated.

“Clear Buffers” clears everything. This button can be placed where you want (in the middle of the window) by Control-Clicking and moving the button while holding the mouse button down.

“Clear TypeAhead” clears the typeahead buffer currently in use.

Back to the Look Up and Log Entry buttons.

Once again, MLDX must have been configured to work correctly using manual operation. if it isn't working correctly for you manually it sure ain't gonna work for me either!

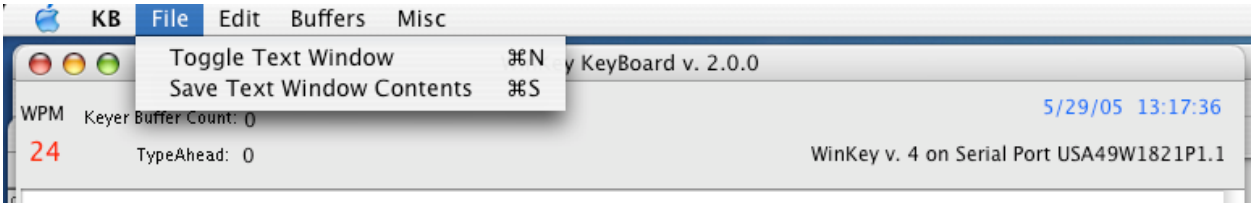
When using MLDX (and especially relevant if MLDX is configured to be able to do a Call Lookup) once a call is entered these buttons are enabled. Selecting the Look Up button will tell MLDX to get the information on that call that is available on the specified data base. Prior to doing this, however, the keyer buffer size limitation is increased to 20 and 20 characters are sent to the keyer (assuming that many are still in the typeahead buffer.) The thought here is that if you are doing a lookup with MLDX you are not going to be doing any error corrections and that switching from one program to another is time consuming so we want the keyer to have sufficient data to work with before starting. Switching back to KB from MLDX will reset the buffer limitation back to what you specified in the preferences. Switching to any other program should have the same effect. I just wait for the mac to tell me that the switch is going to be made instead of my knowing before hand that it is going to occur.

Much of the same buffer size processing occurs when you log the QSO by selecting the "Log Entry" button.

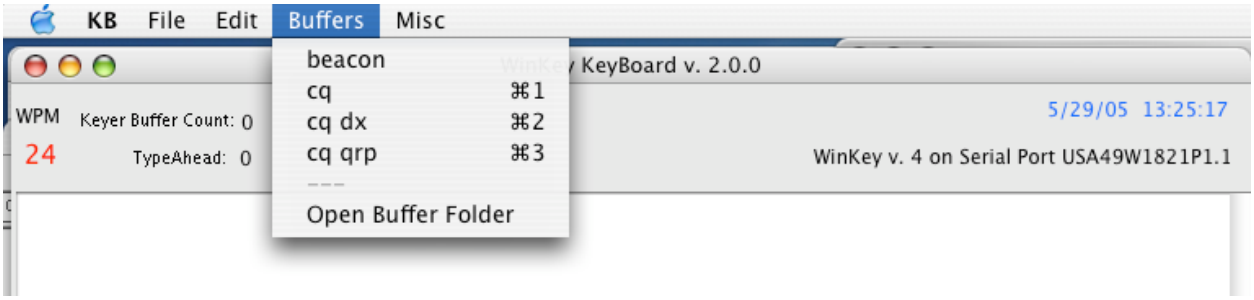
Logging entails getting the current date and time (UTC), gathering up the other stations call, the RST data, the comments, the previously stored start date and time (stored when "return" entered in the "Call" field), the mode and frequency, and sending that all off to MLDX with the command to MLDX to log all this. You will see a short temporary switch to MLDX while this occurs and, unlike call lookup, control returns to KB.

That just about finishes the description of basic operation.

On to menus and what they provide.




For those that like to copy using a “mill” and for whom the scratch pad is insufficient, an additional text only window can be opened for that purpose. Note that the contents can be saved in a file when this feature is used.




The description of Buffers is tied somewhat to the file about scripting titled “Script capability description.” When noted you might want to refer to it.

The Buffers menu lists the files that are in the “Buffers” folder that is located in the KB preferences folder. The “Open Buffer Folder” menu item will open the folder in the finder to make storing/removing files from it easier.

These buffers are simply files that you have previously created with your text editor of choice that contain scripts and/or data that is processed by KB when selected via this menu.

The files listed in the menu shown above are shown as examples only.

When “Buffers” is selected the directory is read, and all files in it have the first line in it read. A discussion of the “cq” file will show why.

The “cq” file consists of:
#1
cq cq cq de w6eet w6eet cq cq cq de w6eet w6eet < k

If the first line of a file consists of a # plus some character, such as the 1 above, then that character is used as a menu selection character to be used with the command key for menu selection.

Note that I do absolutely no checking for conflicts with existing menus and duplicate specifications will probably give surprising results.

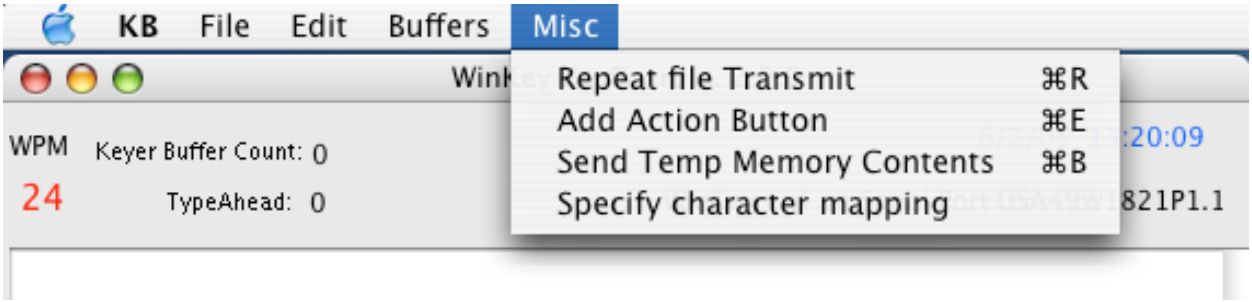
During normal operation entering command-1 will result in
cq cq cq de w6eet w6eet cq cq cq de w6eet w6eet < k
being stashed in the typeahead buffer as if it were typed in.

The “beacon” file has no command key indication and consists of:
\chgspeed 13\loopde w6eet escondido ca near san diego\pause 05\keydown 05\pause 05

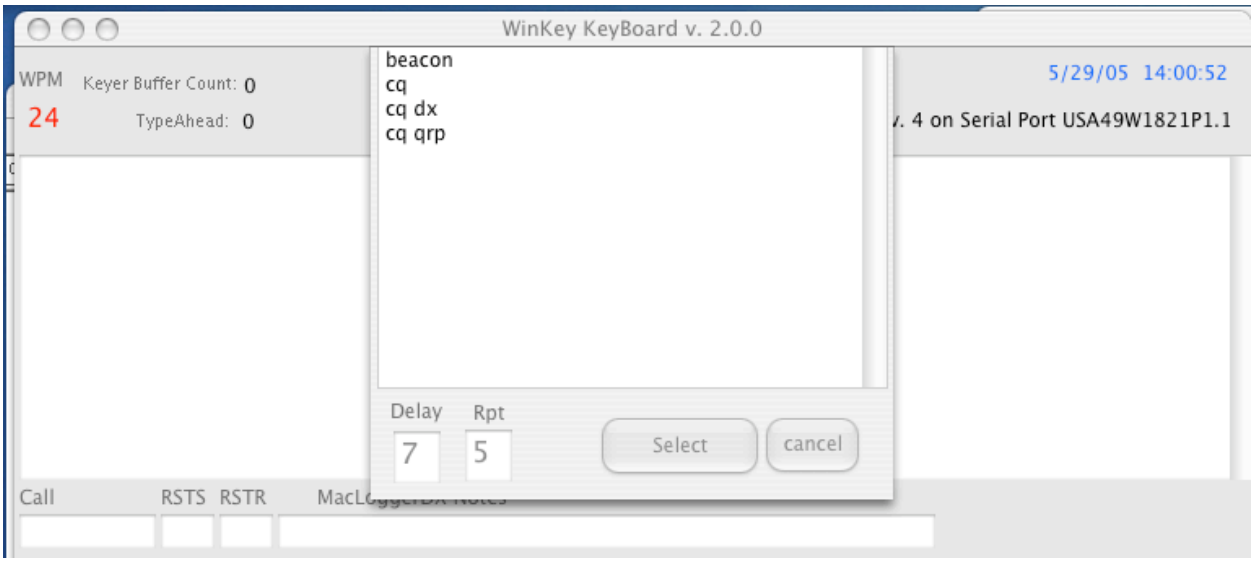
This menu item would have to be selected with the menu since the first line does not consist of a #somecharacter.

The files in the buffer folder are essentially scripts and this is where you would turn to the “Script capability description” file for additional information on what the files contain. An example of what can be done is included in the “Example Keyer Commands” folder.

Note that since the “Buffer” menu is built after it is selected you can modify the file contents as well as the directory contents (adding/deleting files for example) and not have to restart the program to see the changes.

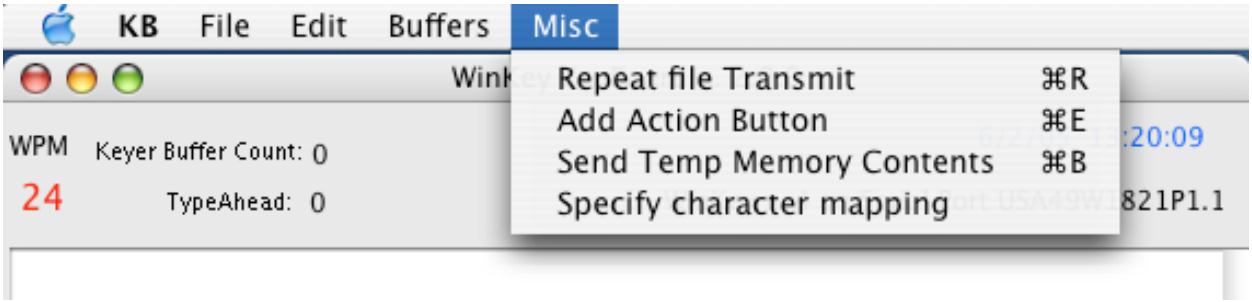



“Repeat File Transfer” permits you to send one of the files in the “Buffers” folder X number of times with Y number of seconds between transmissions. When selected you will see something like the following:

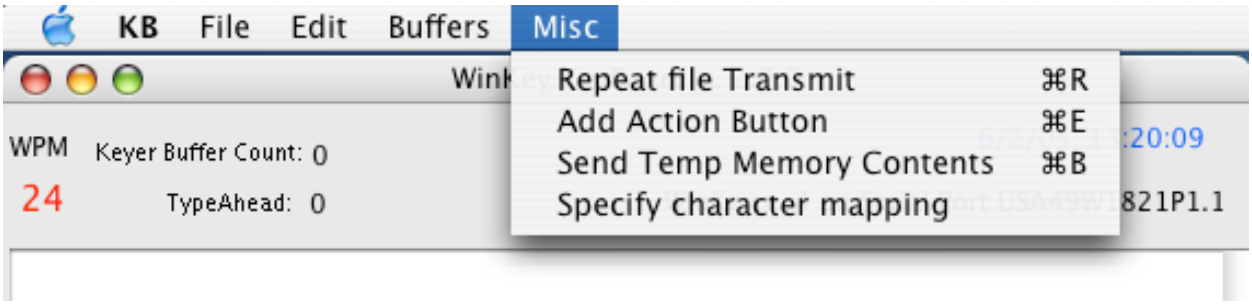



Double click or click then select the file you want to send. Once started the “Cancel” button

changes to “Stop”. If Stop is selected during a transmission all buffers, including the keyers, are cleared, so if someone responds to a cq, for example, clicking on “Stop” will clear the buffers and close the Repeat window and you are then ready to respond.




“Send Temp Memory Contents” sends the “ hiscall de urcall “ message described earlier.



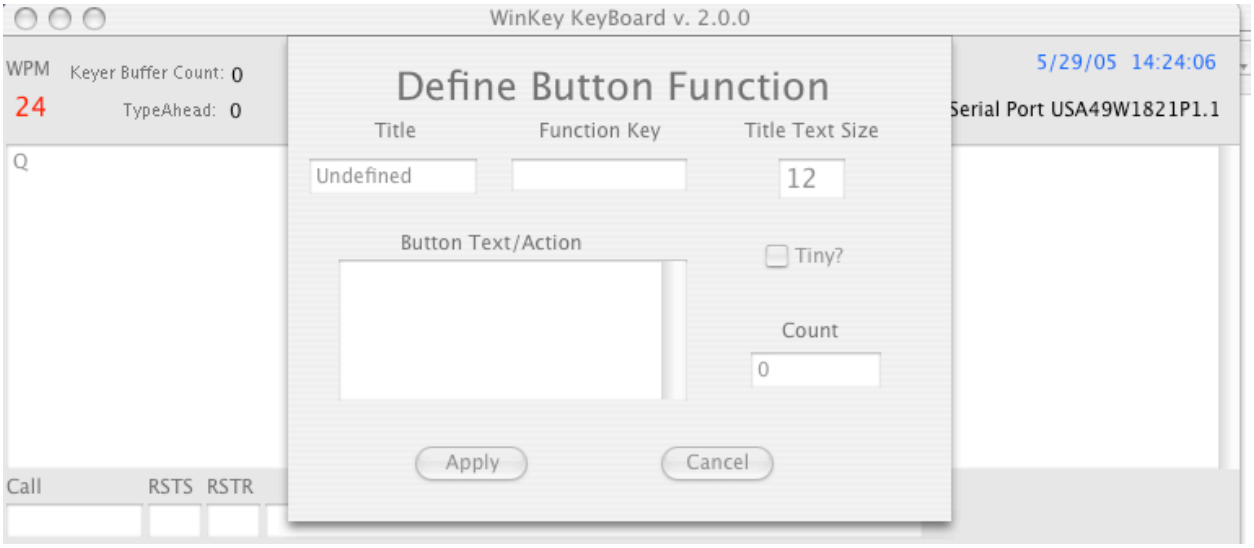

# Action Buttons

“Action Buttons” are brought over from my Jupiter Remote Control application where their usefulness is more pronounced, but I think they do have some usefulness here as well.

“Action Buttons” provide the means of executing scripts by clicking on a button instead of selecting a file from the “Buffers” menu by either specifying the name of the file within the “Button” or by including the contents of what you want to occur within the button itself.

Additionally, “Action Buttons” provide a rudimentary contest capability in that it can keep a qso sequence number for you. I am not a contester so it’s actual usefulness is assuredly debatable but I gave it a shot anyway.

When “Add Action Button” is selected the following window appears:




“Title” is part or all of the buttons caption or title. It is only part of the caption if you also specify something in the “Function Key” field and/or include within the buttons “Text/Action” field that you are using a contest sequence counter.

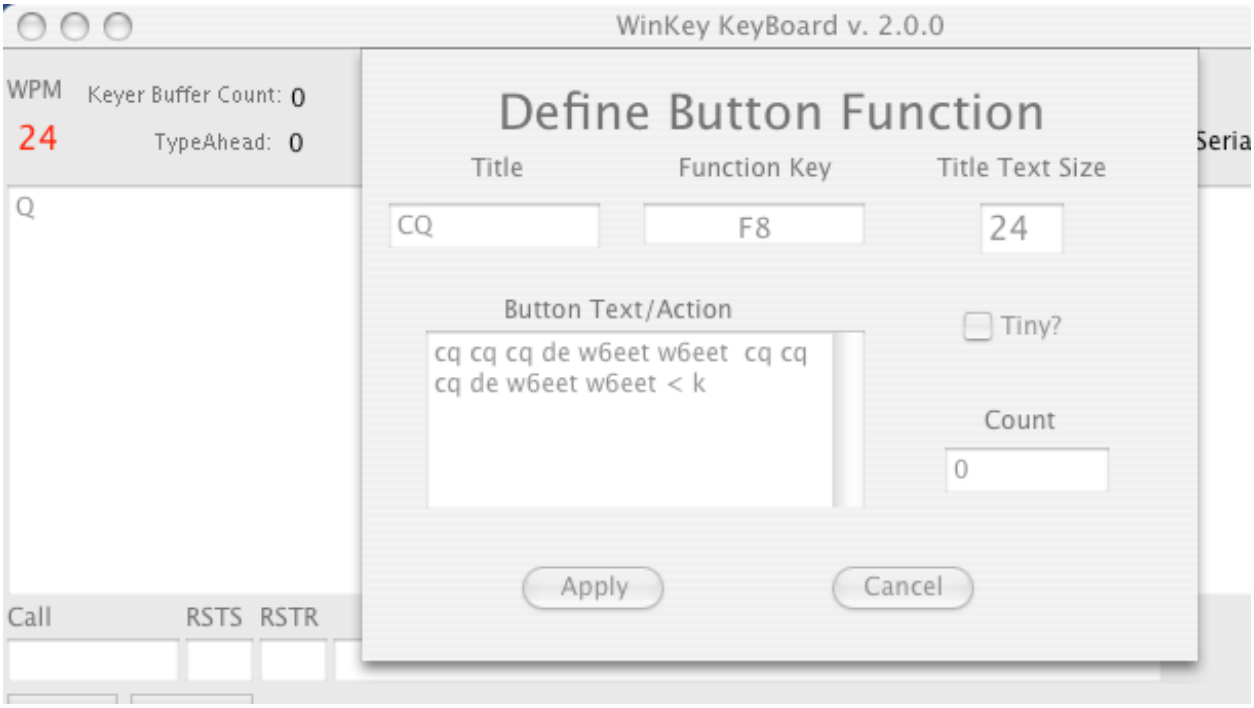
If a key sequence is desired to select the button in addition to clicking on it then select the “Function Key” field and then enter the control key combination or function(F) key that you want to use. Note that I do absolutely no checking for conflicts with existing buttons and duplicate specifications will probably give surprising results.

The size of the button is determined by the combination of title (including the counter and function key indication), the specified text size, and the selection of “Tiny.” Experimenting with different values of these is easy.

“Count” refers to the value of the counter kept internally during a contest. It is included here so that in the case of a “disaster” the button can be recreated with the starting count instead of zero.

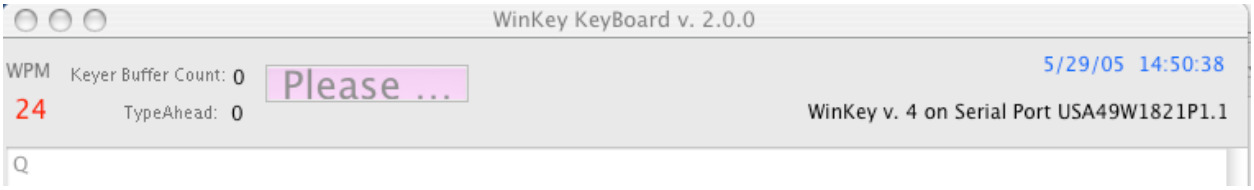
“Text/Action” contains essentially what you could put in a script/buffer file. For example you could put your cq sequence in here, label it as “CQ”, specify a function key or control key to be used to kick it off, and a large text size so the button will be large. The result would be a big button labeled “CQ” that contains your cq call and the button is located in the window where you put it.

Let’s go and create that button.



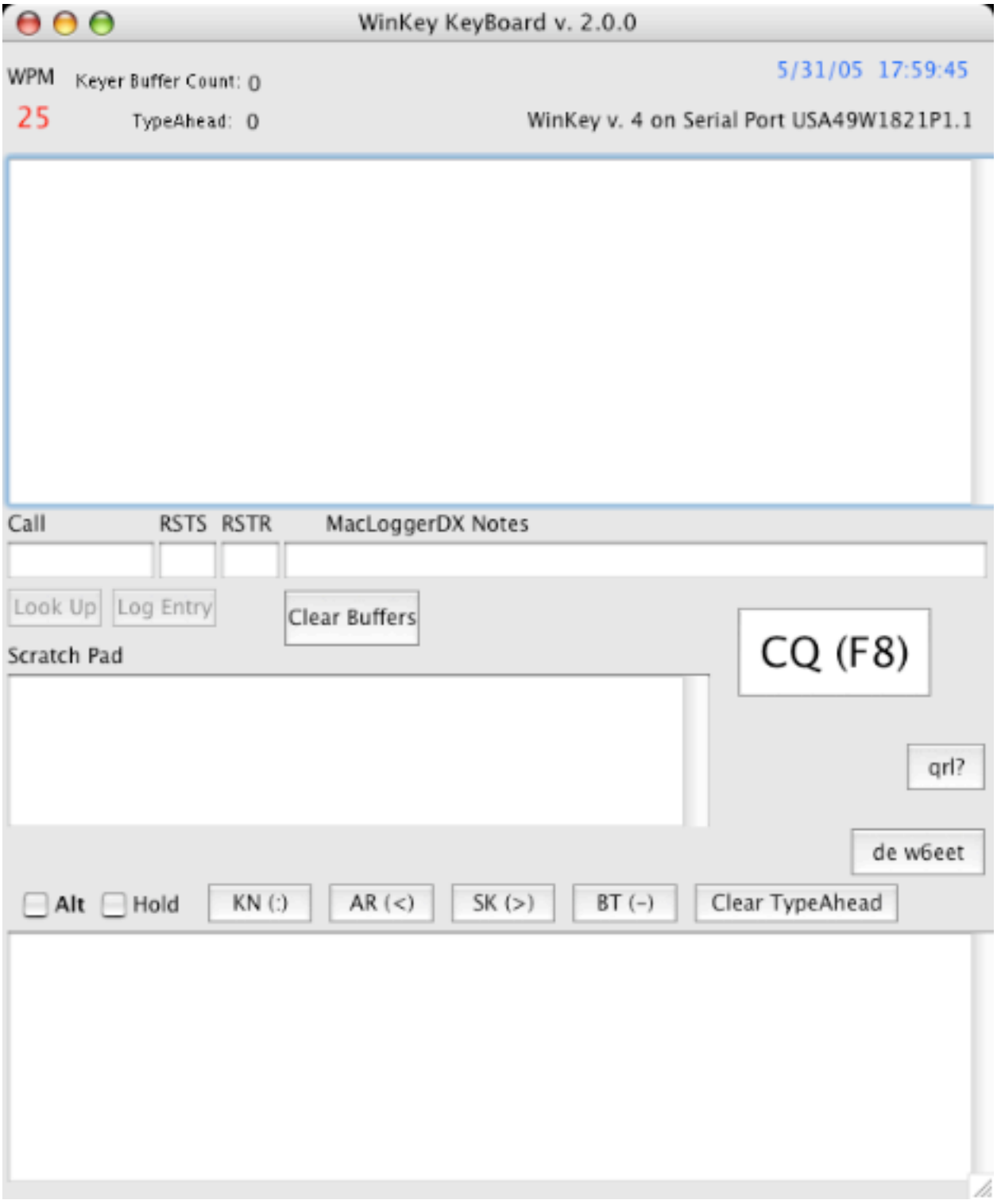

I am now ready to click on reply and then move the button where I want it to be. Note the Function Key field contains “F8” and that is because I pressed the F8 key after selecting that field.

After pressing Apply this window closes and you see:




If a smaller text size were chosen you should see “Please move me” instead of just “Please..” but I wanted to show what would happen if that message were truncated.

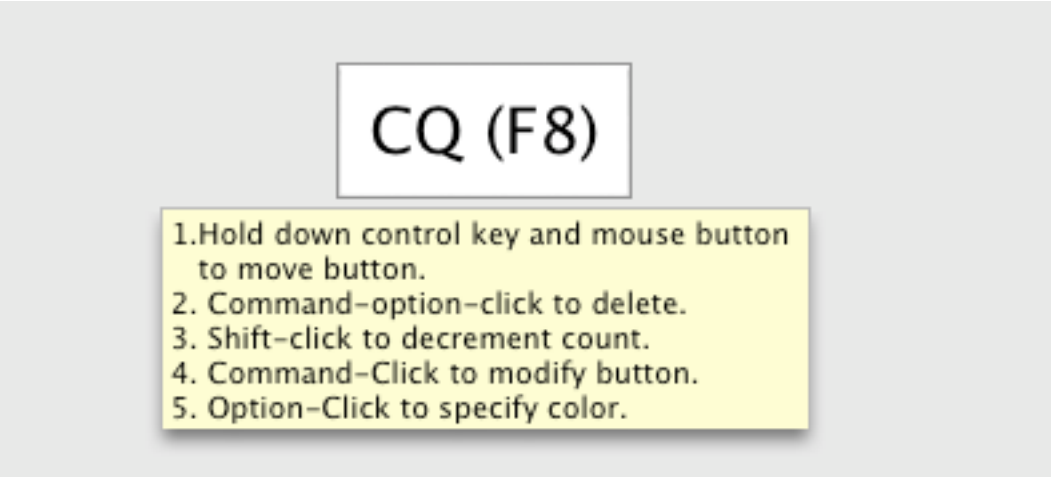
The next thing to do is move the button to wear you want it to be. The is done by pressing and holding the control key while clicking and holding the mouse on the button. Then you move the mouse to where you want the button to be. An outline of the button in the eventual size of the button will follow the mouse to assist in placing the button. You're just going to have to do this yourself to understand, I think. Anyway, the end result, depending on where you choose to locate the button, would be something like what you see on the next page:




Either clicking on the button or pressing F8 will activate the script defined within the button. Since “Undefined commands” are sent as data, the contents of the script are then transmitted.

But what if you want to change what you specified in the button?

Easy, but first look at this.




If you let the mouse pointer “sit” on the button for a second or two the above will appear.

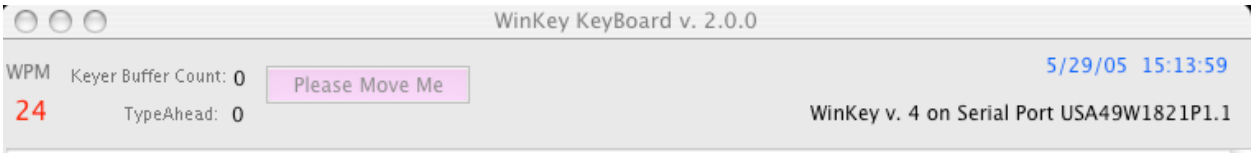
I’ll explain a couple of these and then we will go modify the button.

If, during a contest, the counter was incremented and then you realized you had a dup, the counter can be decremented by shift-clicking on the button.

Option-click to bring up a standard color selection dialog to select a color for the button.

Ok, let’s modify the button by doing a command-click of the button. You will then see the following window..


This is very similar to the button creation window so we will simply change the Title Text Size from 24 to 12 and click on Apply. You will then see..




With the smaller text size you now see the complete “move” message. After moving the button to the desired location you will see something akin to:

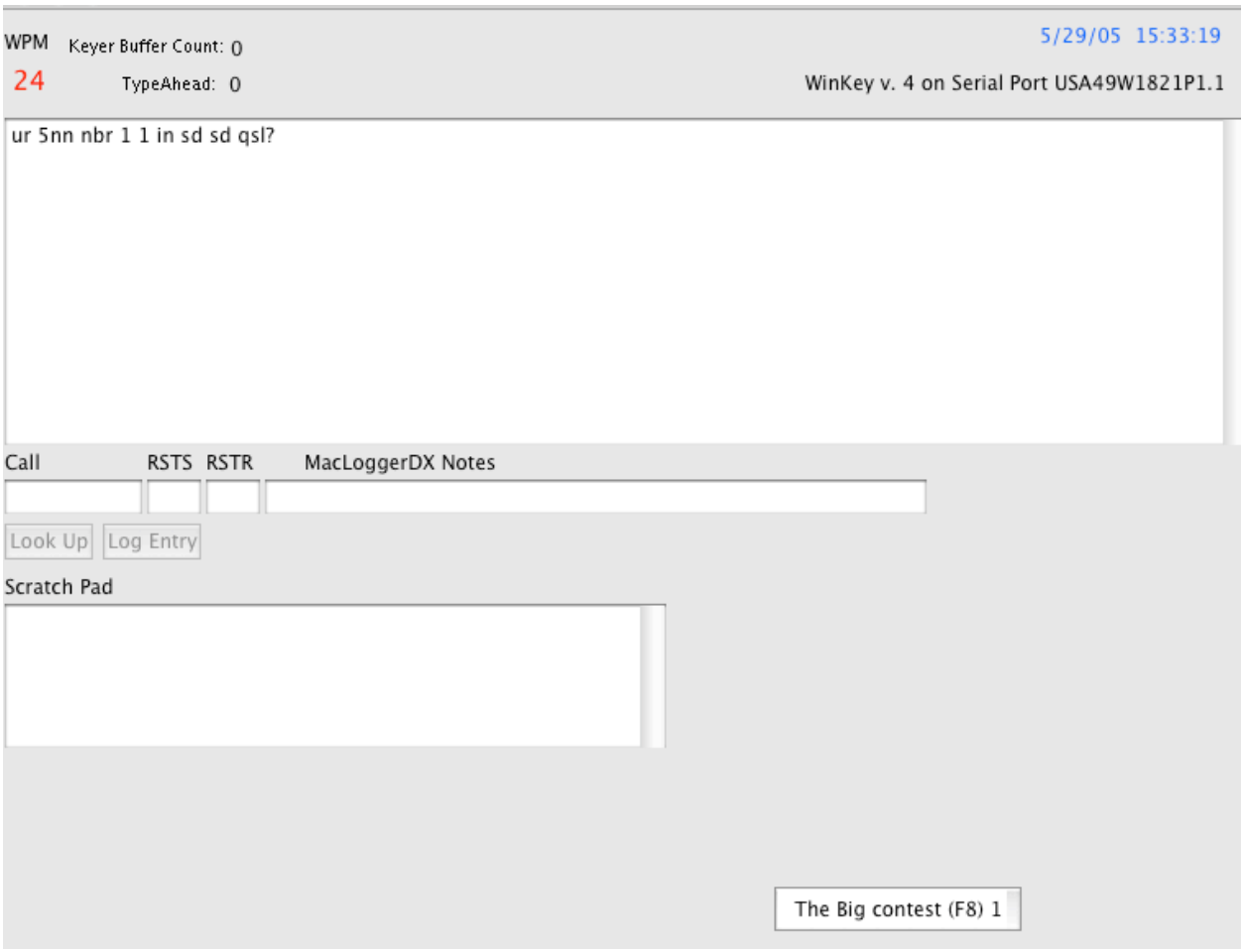

And, it is now smaller.

Now, about that contest thing...

Let's set one up...

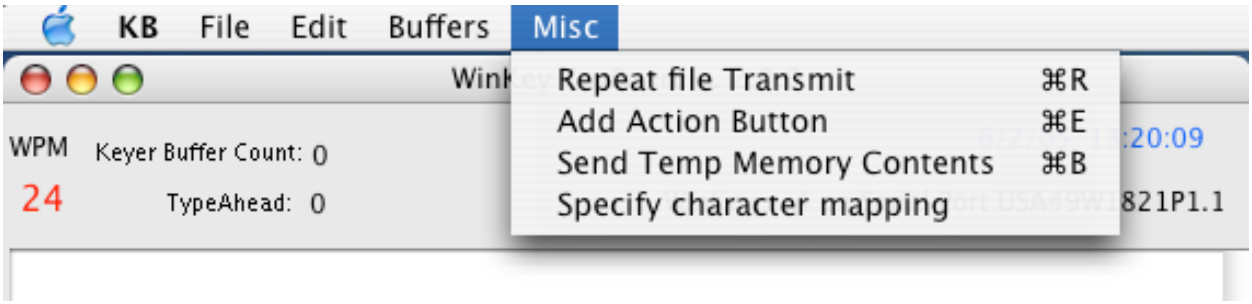
During the Button creation you enter into the text field something like...


Then after "Apply", moving, and the first "execution" of the button you will see...
( On the next page)




In the upper part of the window you can see that the number 1 was substituted for the 2 occurrences of % and that the caption of the button was changed to include the qso count.

The buttons are saved in the preferences file and reloaded so they are “semi” permanent.




**Specify character mapping**

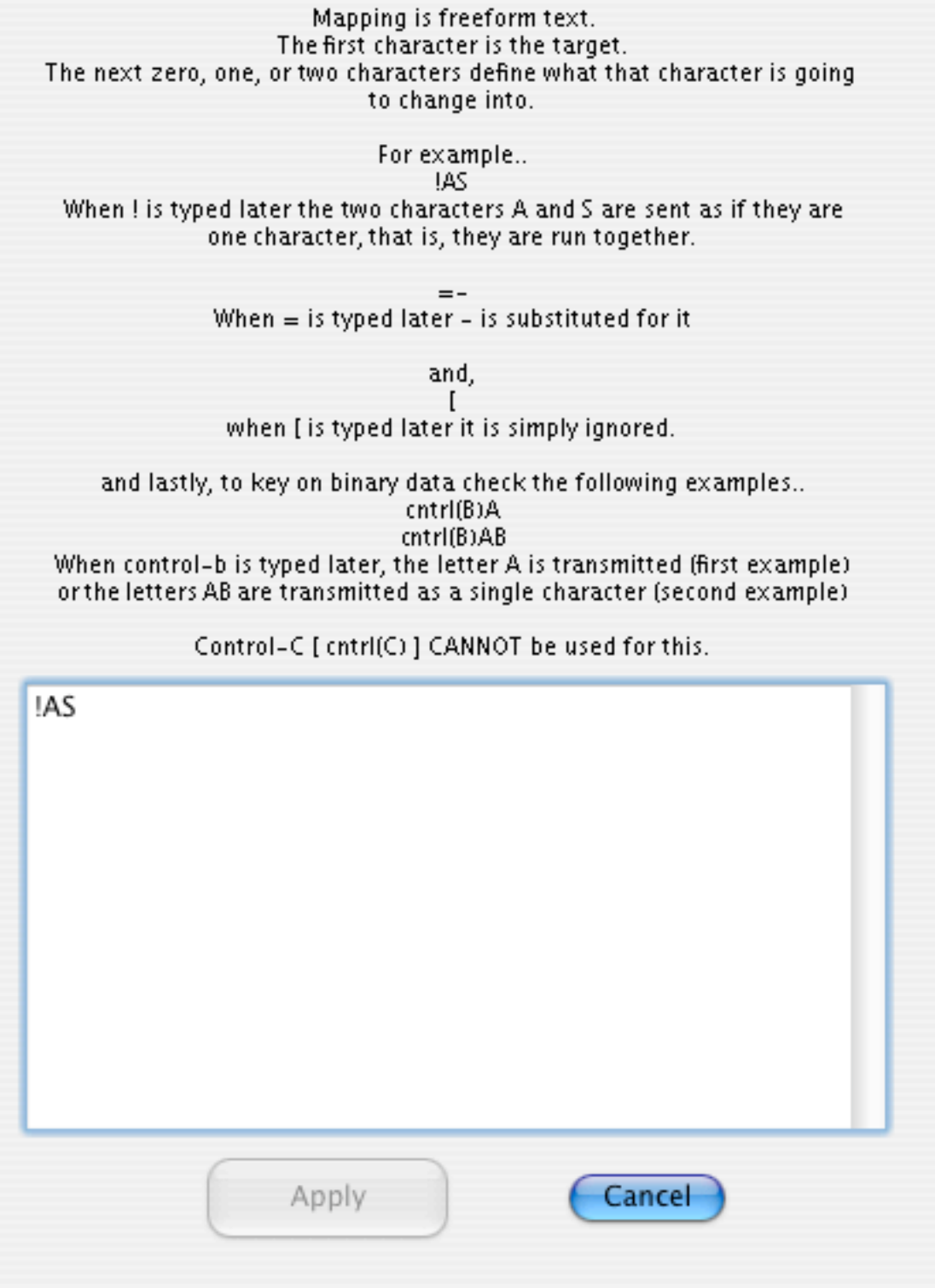
3 capabilities are provided:

1. One-to-one character substitution
format: XY where Y is the character you want to be used in place of X.
Example: =-
The effect would be that everytime a = is entered a - is substituded for it.

2. Ignore a character
Format: X where X is the character to be ignored

3. Merge two characters into one continuous character
Format: Xyz Where X is the "KEY" character and yz are the two characters to be sent as one.
Example: !as
The effect of above is that when ! is seen "as" is sent as one continuous character.

When the “Specify character mapping” menu item is selected the mapping file in the preferences folder is read and made available to be changed via the window on the following page.




Make the desired changes, click Apply, and the changes are available immediately.

# Misc

With the exception of menus  items such as buttons and text areas that can have a color specified can have a color assigned to it by option-clicking on that item. Command-Option clicking in text entry fields to change the text color. The color for static text fields can be changed by option-clicking on the text.

The window background color can be specified as well.

### Resizing input/output buffer areas

If you place the pointer on the top of the input text field, or the bottom of the output text field, you will see the cursor change to a “split” cursor. If you then click the mouse you can alter the size of these two edit fields by moving the mouse up or down. When you have the size you desire another click of the mouse will set the size at that point. The field sizes are saved in the preferences.

### Specifying input/output text color and size

Command-Option click in the input/output buffer areas brings up the color and text size selection dialogs.