Using a Nextion Display to Update a Classic Keyer

An exploration of new technology modernizes a keyer design.

Rick Dubbs, WW9JD

Kit building is one of my favorite ham radio activities. In 1984, I purchased the Heathkit SA5010 µMatic Memory Keyer. It's pictured in the splash screen of my updated keyer in the lead photo. The µMatic had a "huge" 240-character command memory and a 22-key membrane keypad. It could use up to 10 variable-length memories and generated repeatable random CW for code practice. Its main flaw was its capacitive touch paddles, which seemingly worked well for only a few hams.

I've been wanting to build an updated keyer with more memory, a physical iambic paddle, and a touch-sensitive LCD screen. I recently found the Nextion NX4832K035 enhanced 3.5-inch HMI touch display LCD Module (available from **www.nextion.tech**). It connects to an Arduino or Raspberry Pi with just a pair of serial wires, and provides a user interface (UI) design environment that is independent from the main program code (known as the Arduino *sketch*). This was the basis for my keyer. Figure 1 shows my updated µMatic2 keyer, with paddles attached, next to the µMatic keyer with its capacitive paddles.

The original µMatic Keyer had 10 memories. My updated version has 110 memories. The Nextion display can have as many as 256 controls on each screen, and as many as 256 screens. This project doesn't need to have a large enclosure to accommodate buttons, switches, dials, and displays. The single Nextion display touchscreen will suffice (see Figure 2).

Nextion Editor

Use the *Nextion Editor* software to create the different screens, or *Pages*, and populate them with controls. The display has the ability to move between the different Pages on its own, or the Pages can be selected under Arduino sketch control. The same goes for the controls themselves. They can get or send their information from or to the sketch. If the **SEND COMPONENT ID** box is checked for a particular control's action (like **TOUCH PRESS** or **TOUCH RELEASE**), then your sketch will be able to make use of that event to perform some sketch function.

I've also created a Page to enable several Pages for reference information. There's a Page with Q signals, a Page with a US call areas map, and some Pages with keyer instructions.

Building Nextion Pages

The *Nextion Editor* is made up of multiple panes (see Figure 3). Start by clicking on the **NEW** icon (or **FILE|NEW**) on the quick access bar. Give your project a name, select which model you're using, and choose the direction in which the display will be oriented.

I recommend you start with the pane in the lower left-hand side. It's a tabbed pane, so its name changes depending on which tab is active. Click on the **FONTS** tab. If you click on the **+** sign to add a font, you'll be presented with an empty **OPEN FILE** dialog box). Click the **TOOLS** menu, and then the **FONT GENERATOR** item. You may create several fonts, but start with one right now. I used a 16-point Arial anti-aliasing font and named it Arial16AA. Click **GENERATE FONT**, name it again, click **OK**, and then **YES** to actually add it to the **FONTS** pane. You can now close the **FONT GENERATOR**.

To add functionality, click on the control you want from the **TOOLBOX** pane. Position it on the Page, resize it as needed, and then adjust its attributes and/or write any needed code to use the control within the display itself.

To try this out, click on the **BUTTON** control and drag it away from the upper left-hand corner. That's where all new controls start. Next, click on the **TEXT** control. Left-click and hold on the right-hand border of the rectangle and drag it over to the other side of the Page. In the **t0(Text) Attribute** pane, scroll down to the **tst_maxl** attribute, then click on the 10 and change it to 50. Now, click on the button you previously added, click on its **txt_maxl** attribute and change it to 30, then click on the **txt attribute** and change **newtxt** to **My Button**. In the **EVENT** pane, be sure that the **BUTTON TOUCH PRESS EVENT** is selected, then in the text box underneath enter: **t0.txt="The button is pressed"**. Select and copy that line, select the **TOUCH RELEASE EVENT**, and paste the line into the text box. Change the word **PRESSED** to **RELEASED**.

Generally, if you want to use a control in your sketch, you just need to be sure the **SEND COMPONENT ID** box is checked for the Event you want to use. A **PUSH** Event is triggered when you first tap a control (**TOUCH PRESS EVENT**), and a **POP** Event triggers when you release a control (**TOUCH RELEASE EVENT**). Different attributes of a control — like size, location, color, min/max, and value — are usually available to customize from the sketch itself. You will also need to hook the control into your sketch, if you are going to make use of **PUSH/POP** Events. The sketch identifies your control based on its Page and ID numbers. All controls on the first Page have a Page number of "0." Each control has an ID number based on the order in which it was added to the Page, or its arrangement in layers. This last feature caused me some trouble when I moved a control to an upper layer and the editor then re-indexed (re-numbered) several controls. The re-numbered controls then no longer responded as expected in the sketch. You can give each Page a name, which the display can use, but the sketch can refer to each Page only by its number. If you reorder the pages, their numbers change accordingly.

Figure 1 — The WW9JD Arduino-based μMatic2 Memory Keyer next to the original Heathkit SA-5010 μMatic Memory Keyer.

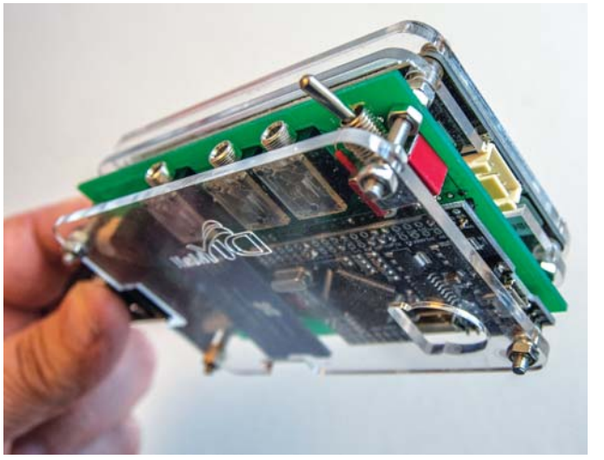

Figure 2 — The keyer board is mounted behind the Nextion display and is sandwiched by the acrylic case.

To add a new Page to the display, click on the **ADD PAGE** icon at the top-left of the Page pane. The Editor will display a list of the Pages that have been created. Initially, each Page is named with the consecutive number of its creation, but you may rename Pages to something more meaningful. The other controls at the top of the Page pane let you delete, insert, reorder, copy, import/export, and delete all the pages at once. After you have created a Page, click on it in order to

view it in the **DISPLAY** pane and begin working with controls.

After populating your Pages with controls, you can compile and test your design right in the Editor. Granted, it knows nothing of input from your sketch, but you can test location and operation of controls, the size of text, and the commands you've programmed to be used by the display itself before taking the time to load it on the display. Click **COMPILE** on the quick access bar, then click **DEBUG** and click on your button to your heart's content.

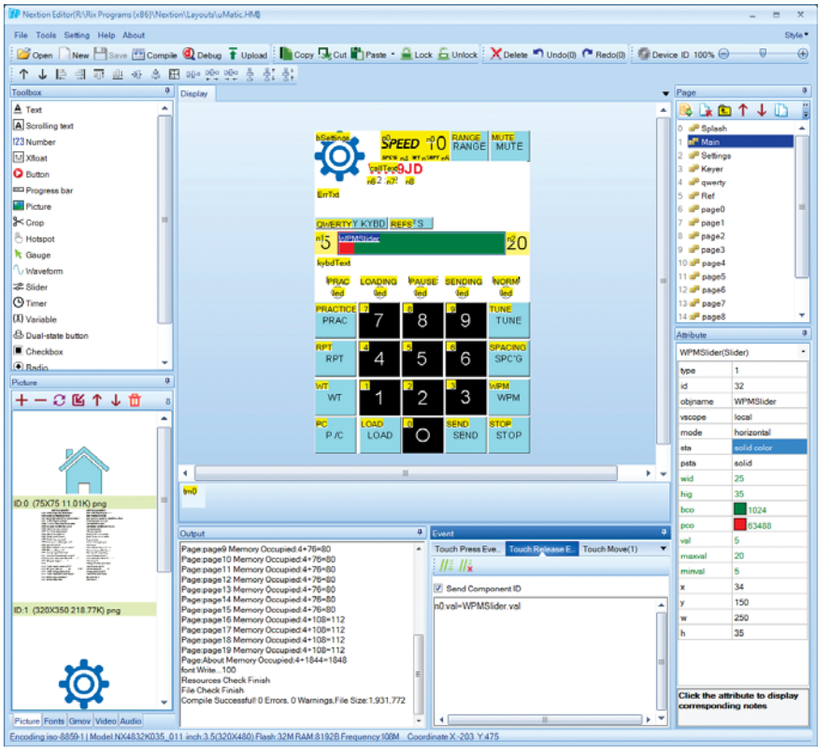

Figure 3 — *Nextion Editor* showing the keyer **MAIN PAGE**.

You will need a microSD card to load your design to the display itself. Use **File|TFT file output** to compile and save the design to the card. When the destination folder is displayed, you can turn off power to the display and insert the card in the slot on the display. When you supply power to the display, it will look for the design file and try to load it. The display screen will show that the upload was successful. Power down, remove the card, and power up to see your design or design changes. As far as I know, the microSD card reader on the display can be used only to upload display designs.

Details about using the display in the sketch are on the **www.arrl.org/qst-in-depth** web page. More project resources, including code and PCB layouts, are available from the author at **www.ww9jd.net**.

Microprocessor Board Selection and PCB Design

One of my goals was to make a smaller, lighter device than the Heathkit original. I wanted to use an Arduino Nano board, but my sketch exceeded its memory limits. I ended up doing much of my prototyping work on a breadboard using an Arduino Mega. I settled on the RobotDyn Mega 2560 PRO CH340G/ ATmega2560-16AU perfboard for my final design.

I wanted the keyer board to be installed behind the display. The Nextion included an acrylic case that is just a frame for the display: a flat back with cutouts for the display connector, microSD card, and the real-time clock battery, some plastic spacers, and hardware.

I ordered another case to use for mounting the keyer board behind the display.

There are several PCB manufacturers that will produce a small run of boards quickly, and at low cost. I used KiCad (**www.kicad-pcb.org**) to design the PCB, and PCBWay (**www.pcbway.com**) to make the actual boards. The total cost was about $30.

My final version includes provisions for attaching the QRPGuys paddle (**https://qrpguys.com/kx-iambic-mini-paddle-kit**), shown in Figure 1. I used epoxy putty to attach a pair of 6-32 standoffs directly to the PCB, placed by using the paddle itself for the proper locations. I also use the μMatic2 keyer with Vibroplex iambic paddles (see the lead photo).

The keyer program, or sketch, mostly replicates the functions of the original μMatic, and then extends that functionality.

Keyer Operation

The Fritzing diagram in Figure 4 includes a microSD memory card that can store gigabytes of data, compared to the 240 characters of memory in the Heathkit original. The Nextion board is in the upper right. The keyer program, or sketch, mostly replicates the functions of the original μMatic and then extends that functionality. The sketch is open source, and I expect others to add features.

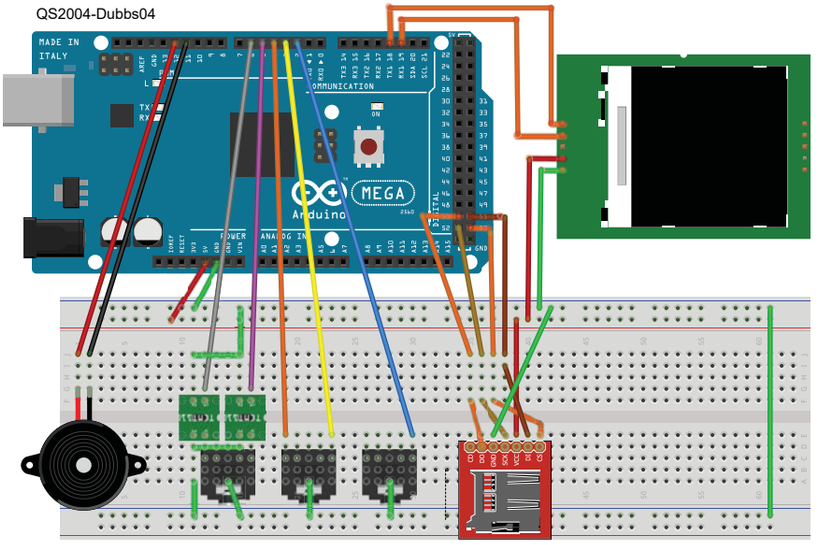

Figure 4 — The diagram for the μMatic2 Memory Keyer shows the Nextion display on the top right and the microSD card on the bottom right.

When you turn the keyer on, you first see a splash (opening) Page that shows the time and has access to an **ABOUT** button that credits some of the folks who posted material online that helped me with this project. The splash Page clears itself after about 5 seconds, or you can tap it to clear it, and then it displays the **MAIN PAGE** seen in Figure 1. The bottom half of the main screen duplicates the keypad layout of the Heathkit original. Most of the function keys work after you have pressed one or two number keys. The function keys on the right-hand side perform the same as the originals. However, both the **WPM** and **SPC'G** may be further modified.

I created several different files with the word "Paris" repeated 5, 10, 13, 16, 18, 20, 25, 30, and 35 times. Use the **TIMING CONSTANT** slider on the Keyer Setup Page to assure that the words per minute (WPM) and Farnsworth settings are accurate. On the left-hand side, the **WT** (weight) control and the **P/C** (pause/continue) work the same, but get used differently. **RPT** (repeat) sets the number of times a file is sent. **LOAD** is used to load or save characters to a file, but not with the paddles. **PRAC** (practice) merely stops the keyer from operating the transmitter. There is no current function in the sketch that produces random CW for practice, but you can copy any text to a file and run it with the keyer.

On the top of the **MAIN PAGE**, there is a large display area that shows you which of the number buttons you have pressed. It displays **OFF** whenever it's not showing recent number button presses. Tapping that display (with or without **OFF** showing) puts the Nextion and the microprocessor board to sleep. Tap the dit paddle to wake it up. There is a display area that will show error conditions, as well as display the name of the file currently being sent. Tap the **MUTE** button in the upper right-hand corner to mute the sidetone, and again to unmute it. The WPM, Farnsworth WPM, weight (if any), and repeat count are shown at the top. The upper left-hand corner has a settings button that gets you to the Nextion settings page. There are two more keys just above the status LEDs to open the QWERTY keyboard Page and the References Index Page.

Conclusion

I now have the μMatic2 update I've wanted for so long. Not only was it well worth the wait, but the process of bringing it to fruition was more enjoyable and educational than I would have imagined. I hope that you will be able to put some of what I learned to use in your next project.

All photos by the author.

ARRL Life Member Rick Dubbs, WW9JD, earned his Novice- and Technician-class licenses in 1977 while in the US Navy nuclear propulsion program. He earned his Advanced-class license in 1987. Rick graduated Ball State University in 1988 with a BS in Middle School Math/Science Education and became a teacher. In 1994, he graduated from Indiana Wesleyan University with an MA in Education, and began teaching in the graduate education program. He retired from teaching in 2017, and currently works part-time with woodworking tools and supplies. Rick also has a photography business (**www.soaringeagle.photography**). You can reach Rick at **umatic2@ww9jd.net**.

For updates to this article, see the *QST* Feedback page at www.arrl.org/feedback.