David M Collins, AD7JT
22111 N San Ramon Dr., Sun City West, AZ 85375; **ad7jt@cox.net**

The Ultimate Keyer

I have been interested in electronic keyers since I was first licensed (WNØLSH) as a teenager in the fifties. Of course, back then, all "electronic keyers" were "vacuum tube keyers". One popular keyer of the day, the Hallicrafters HA-1 Keyer, used four dual-triode, 12AU7 vacuum tubes to perform a basic set of logic functions to generate "self-completing" dots and dashes as dictated by the "key lever" contact closures. With only the income from my paper route there was no way I could afford the $79.95 sticker price on the HA-1 nor did I have the wherewithal to design and build my own version.

During the seventies (as WØLSH), the time demands of starting a career and family left little time for amateur radio and I let my license lapse. I retired in 2005 and my wife and I moved to our retirement home in Arizona. Having never lost my interest in the hobby, I got relicensed in 2006 as AD7JT and started to familiarize myself with all the changes made to the hobby over the intervening years. I learned to use the keyers built into a couple transceivers and purchased a stand-alone keyer, but was never completely satisfied with it. I had spent my thirty-year career working in the computer industry in both hardware and software development and started to think about applying some of that experience to creating the ***ultimate keyer***.

I got my first opportunity to implement a keyer while working with George Herron, N2APB, of Midnight Design Solutions[1], on the NUE-PSK digital modem. I had just completed adding a couple new features to the modem firmware when George asked me if I would be interested in adding CW to the modem's PSK and RTTY operating modes. I implemented the new mode using the keyboard for transmit and displaying the transmitted and received text on the modem's LCD display. Next, I added a connector for a paddle and implemented my first electronic keyer in the modem's firmware so text could be entered either with the keyboard or with an iambic paddle.

About that time I started to develop an interest in QRP rigs. I noticed that most QRPers had several QRP rigs and virtually all of them were CW rigs. A few rigs had built-in keyers but they were usually a one-chip, bare bones hardware design. Since the firmware in these keyers can only be controlled using the paddle, or possibly a single COMMAND push button switch, setup usually involves a series of tedious text entry operations using special character sequences. Commercial keyers could be used on these rigs but there are some restrictions there too that keeps that solution from being ***ultimate***. I started thinking seriously about what the ***ultimate keyer*** should be.

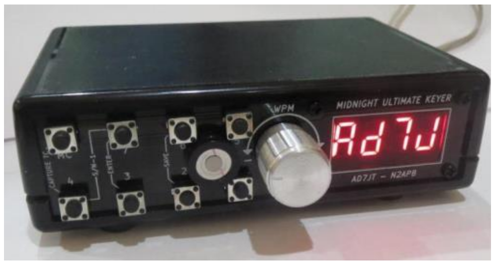


Figure 1 -- Midnight Ultimate Keyer, Front View

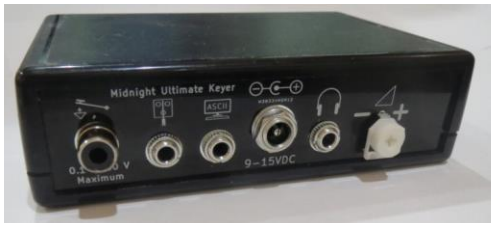


Figure 2 Midnight Ultimate Keyer, Rear View

Figure 1b -- Midnight Ultimate Keyer, Rear View

The Ultimate Requirement List

After building a few feasibility models of not-so-ultimate keyers, I finally settled for the following list of basic requirements for my ultimate keyer:

- Small size, very portable
- Built-in visual display to aid setup and operation
- Simple, intuitive, and flexible user interface
- Stand-alone setup and operation (no PC required)
- Multiple keyer modes
  - Iambic A
  - Iambic B
  - Ultimatic
  - Dot-preferred
  - Dash-preferred
  - Semi-automatic (Bug)
  - Manual (Straight)
- Macros
  - Pushbutton activation
  - Non-volatile storage
  - Special accommodation for **My Call** and **Their Call**
  - Macro chaining, looping, and nesting
  - Keyer speed (WPM) control sequences
  - Time delay insertion (for beacon mode)
  - Some editing support for paddle input
- Serial interface to an ASCII terminal/terminal emulator
  - Add convenience to setup (e.g., entering and editing macro text)
  - Display and/or log transmitted text
  - Keyboard input of transmitted text
- Side tone generator
  - Built-in speaker (mutable)
  - Phone jack with volume control
  - Variable audio frequency
- Practice modes

I also compiled the following nice-to-have list:

- High-current keying (up to 2A)
- Boat anchor keying (grid blocking and cathode)
- PCB-mounted connectors eliminating wiring harnesses
- Audio Channel
  - Mixes keyer side tone with received audio
  - Independent side tone and received audio volume control
  - Audio amplifier with gain control
  - Flexible configuration
- CW audio band-pass filter

The remainder of this article describes the final implementation of the basic ultimate keyer. A future article is planned to cover the extensions in the nice-to-have list.

David M Collins, AD7JT
22111 N San Ramon Dr., Sun City West, AZ 85375; **ad7jt@cox.net**

Enclosure

About this time, George (N2APB) and I were working on a GPS-disciplined clock and he had found a great enclosure for that project. It looked to me like it would be a great enclosure for the ultimate keyer as well.

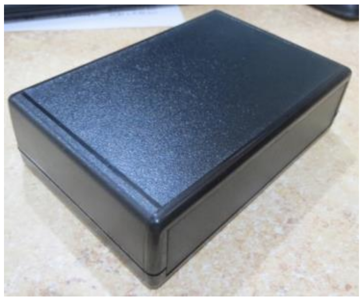


Figure 3a -- Keyer enclosure, assembled

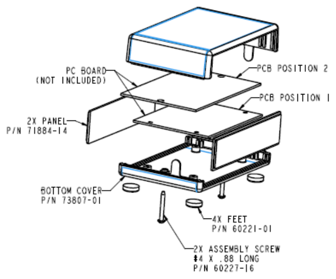


Figure 3b Keyer enclosure, exploded view.

The ABS plastic enclosure is made by PacTec[2], model LH43-100. It is available in either black or bone color, however, with a name like Midnight Design Solutions, ours had to be black. The overall dimensions are 4.5 inches wide by 2.68 inches deep by 1.25 inches high. The front and rear panels are 4.173 x 1.062 inches (105.99 x 26.96 mm) and fit in grooves in the top and bottom halves of the enclosure. The grooves are about .062-inch wide, perfect for standard, 1.6mm PCB stock. I envisioned the main keyer PCBA (PC Board Assembly) replacing the original plastic front panel and thus having it serve as the outward-facing front panel of the enclosure. Through-hole mounted switches, speaker, and a rotary control could mount on the front of the panel; all other components would be SMT devices mounted on the rear of the front panel. A second PCBA could hold the external connectors and replace the enclosure's standard rear panel.

Commented [n1]: You probably need a short sentence that describes our partnership and/or the mention of MDS before this point. P perhaps in the previous section where you first mention the "Midnight Ultimate Keyer" in the photo caption. Otherwise iIt just sort of pops up for the first time here.

Commented [n2]: Metrics may be superfluous for this article(?)

Major Components

Now that I had decided on the ultimate enclosure, the next task was to come up with the detailed ultimate keyer circuit design. The goal of course is to meet the minimum or basic requirements with a minimum of devices and cost while fitting on the chosen enclosure's approximately 4" by 1" front panel PCBA. A secondary goal was to easily accommodate extending the basic keyer capabilities to include the items in the nice-to-have list. I won't go into all the rationale behind all the decisions that went into the circuit design. Instead, I'll just talk about the *ultimate* results.

Pushbutton Switches

Eight, SPST pushbutton switches were to be located on roughly half the front panel in two rows of four buttons, they must be small size, look attractive, and easy to actuate. These switches are to be mounted on the front of the panel so they will need to have through-hole pins positioned under the switch body to hide the through-hole pads. Such a switch is made by Gravitech[3]. Fortunately the switch (or a very similar one) is readily available from multiple suppliers on eBay at very attractive prices. The switch body is only about 1/4-inch square and about 1/8-inch high and has a very positive, tactile feel to it.

Commented [n3]: Maybe your original word "activate" works better. But switches are actuated, not activated. Whichever.

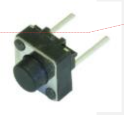


Figure 4 -- Pushbutton switch

Speaker

Again, size is critical and it must mount on the front, outward-facing side of the front panel and be attractive. Since I wanted to control the frequency of the sidetone, a real speaker is required as opposed to a buzzer (or Sonalert). The speaker I chose is a transducer speaker sold by Mouser[4]. It has two pins located under the roughly half-inch diameter round body. The speaker is about 0.2 inches high. The resonant frequency is about 2 KHz but it performs well over the range of 500 Hz to 9.5 KHz.

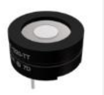


Figure 5 -- Mini-speaker

David M Collins, AD7JT
22111 N San Ramon Dr., Sun City West, AZ 85375; **ad7jt@cox.net**

Display Module

Here again I borrowed from George's GPS-disciplined clock. The display is a four-character LED clock display, each character being a seven-segment LED with a colon between the second and third characters. The .036-inch high display comes mounted on a small (1.65" x 0.5") PCBA along with a controller (TM1637) and a 4-pin interface connector. Two pins are used for voltage (3.3V or 5V) and ground, and two pins for a serial interface. The serial interface is similar to I^2C but does not use an address so an I^2C interface cannot be used to control it. Instead, the firmware "bit-bangs" out the clock and 40-bit control data word. The module is made by Catalex and is readably available from several suppliers on eBay at very reasonable prices.

Figure 6 -- LED display

Rotary Control

My original attempts at designing the ultimate keyer used a potentiometer for speed control. I didn't really like the lack of precision and the fact that it was really a single-function control so I decided to upgrade to a rotary encoder with a built-in pushbutton switch. Combined use of the encoder output and the pushbutton switch allows the firmware to navigate through an array of control states (Figure 23) for entering a number of operating parameters with digital precision.

Figure 7 -- Rotary encoder

Micro Control Unit (MCU)

Since "retiring" in 2005, I have done a fair amount of embedded system design using Microchip's eight and sixteen-bit PIC MCUs. Since I was pretty well set up with firmware and hardware design tools for the PIC lines, this was the obvious choice for the ultimate MCU family. I initially tried an eight-bit design programmed in assembly language but that turned out to be far more problematic than could be justified by the small price difference between the two MCU families. I wanted the MCU to be easily replaced so it had to be socketed which meant a 0.3-inch wide, dual inline package (DIP), not an SMT device. I also wanted an MCU with a built-in non-volatile memory (NVM or EEPROM) for storing recorded messages (or macros) and a number of operating parameters such as keyer speed and iambic type. After reviewing Microchip's online product guide[5] I zeroed in on the PIC24FV32KA301-I/P which has the following features that are used in the ultimate keyer:

Figure 8 -- 20-pin DIP

- Package: ....20-pin DIP
- Usable I/O pins.........................16
- DC Power.................................2.0V to 5.5V
- Program Memory:....................11,264 24-bit instruction flash memory
- Data RAM ....2K bytes
- EEPROM ....512 bytes
- System clock............................Internal RC oscillator
- Instruction execution rate........Up to 16 million instructions per second (MIP)
- Hardware timers/counters.......Five, 16-bit
- Serial interfaces.......................UART and I^2C
- Input pull-up resistors ....Programmable
- Field programmable ....In-Circuit Serial Programming (ICSP)
- Debug support.........................In-Circuit Debug (ICD) via two pins

MCU IC socket

There are SMT IC sockets available but they tend to be pretty pricy. They can sometimes be found at reasonable prices on eBay but the supply cannot be relied upon. As a result, I decided to modify a standard, through-hole IC socket to a gull-wing version for SMT mounting. The kit builder would just need to first bend the pins out at a 45 degree angle, then make a 90 degree bend up at the center of the pin. When the socket is soldered to the DIP SMT pads, the builder puts downward pressure on the socket to make sure all 20 pins contact all 20 SMT pads.

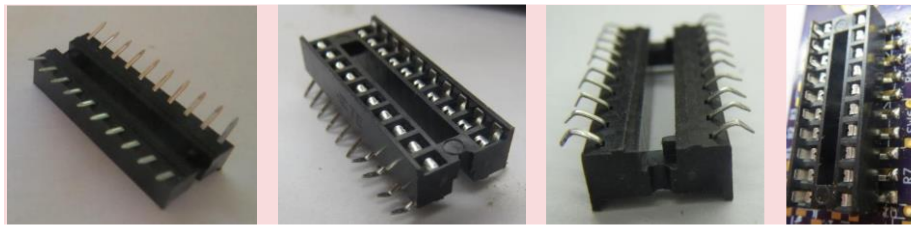


Figure 9 -- Converting 20-pin DIP socket to 20-pin SMT Socket

Interface Connector

A SMT male header mounted on the rear of the front panel connects to a female, IDC connector wired to the interface connectors on the rear panel. The connector is a 2x5 header with gull-wing legs that solder to SMT pads on the rear of the front panel PCBA. The header comes in 40-position strips notched for easy separation into smaller header strips.

Figure 10 -- SMT Header Strips

Miscellaneous Components

The remaining components are SMT devices with very small footprints to enable them all to fit in the available space. Resistors and capacitors are case code 0805 (inches). The voltage regulator and transistors have SOT-23-3 footprints.

Commented [n4]: These sections may be unnecessary and just add to the length of the paper. Better to be shorter with more interesting concepts.

Front Panel Printed Circuit Board

About the time I had started getting serious about the ultimate keyer, I was also investigating printed circuit board design software. I found the freeware versions of most of the available programs have limitations too restrictive for this project. Then I found KiCad[6], an open source freeware total design package without limitations. The board is laid out with the eight pushbutton switches in two rows on the left side of the front of the panel. The MPU and interface header are positioned on the rear side of the panel between the two switch rows. The rotary encoder is panel mounted about in the center of the PCB. The display module mounts on four standoffs behind a rectangular cutout on the right side of the panel. A semi-transparent red lens fits in the cutout. The only hardware showing are four black nylon screws holding the display standoffs to the panel. The rear of the panel has SMT-like pads for connecting to the encoder solder pins and the display module connector.

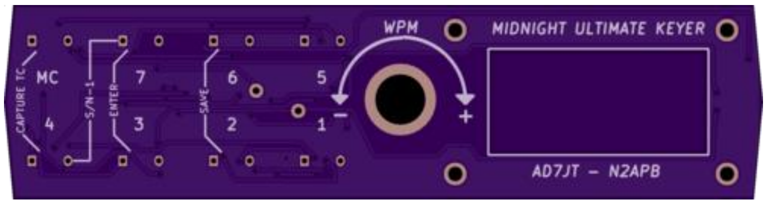


Figure 11 -- Front panel, front view.

David M Collins, AD7JT
22111 N San Ramon Dr., Sun City West, AZ 85375; **ad7jt@cox.net**

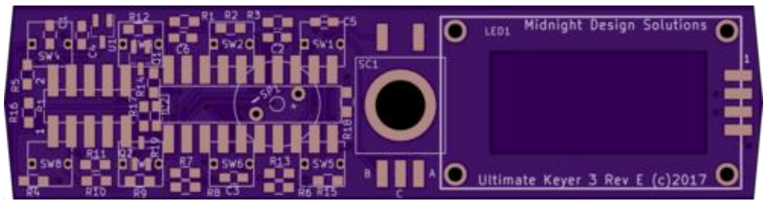


Figure 12 -- Front panel, rear view.

Rear Panel Printed Circuit Board

The rear panel mounts five connectors and a potentiometer. In most cases, the connector terminals are connected to a wiring harness made from the end of a 10-conductor ribbon cable with a 2x5 IDC connector on the other end. This connector plugs into the male header on the back of the front panel. SMD-type pads and etch connections are provided to connect the audio volume control potentiometer to the wiring harness and to the phone jack.

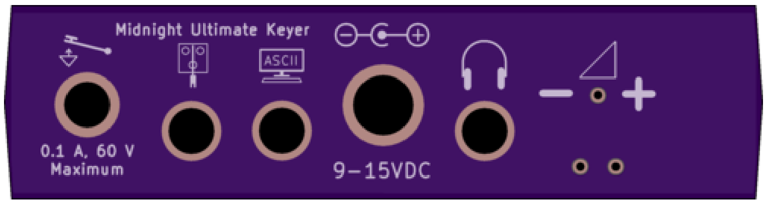


Figure 13 -- Rear panel, rear (outside) view.

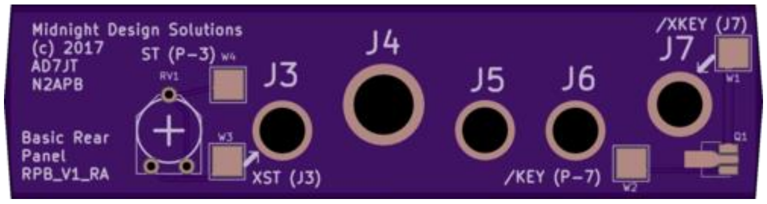
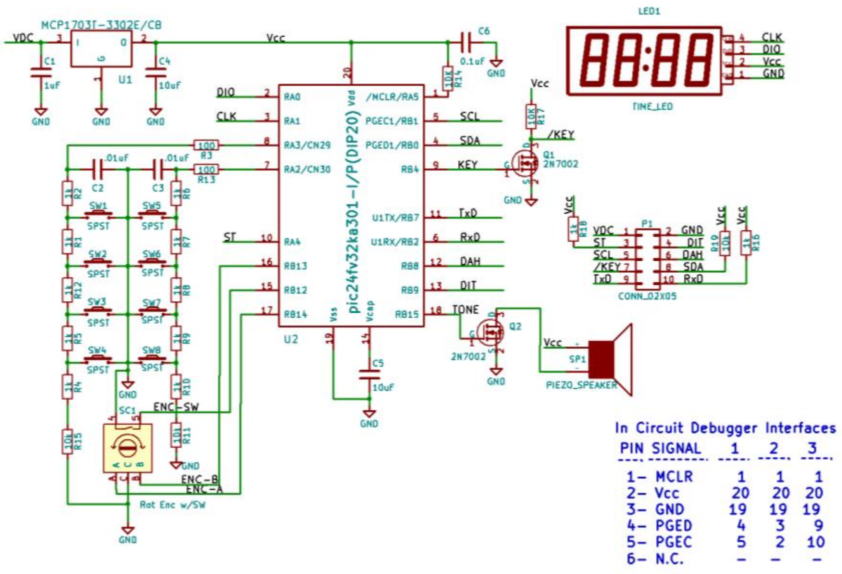


Figure 14 -- Rear panel, front (inside) view.

David M Collins, AD7JT
22111 N San Ramon Dr., Sun City West, AZ 85375; **ad7jt@cox.net**

Schematic

The next problem to solve was how to connect all those components to a 20-pin MCU. I have been told that I am pretty good at getting ten pounds of "stuff" into a five pound sack, so I applied that skill during the layout step!



In Circuit Debugger Interfaces

PIN	SIGNAL	1	2	3
1-	MCLR	1	1	1
2-	Vcc	20	20	20
3-	GND	19	19	19
4-	PGED	4	3	9
5-	PGEC	5	2	10
6-	N.C.	-	-	-

Figure 15 -- Ultimate Keyer Schematic

Pushbutton Switches

The main challenge here is the eight pushbutton switches. I must admit, the technique I used is not my invention but is something I had seen in other designs and adapted to my ultimate keyer. This technique reduces, from eight to two, the number of pins required for MCU to read the switches. Each row of switches is sensed independently so it is possible for the firmware to simultaneously sense two switch closings, one from each switch row.

The sensing is done by first charging a capacitor (C2 or C3) to the logic high level (about 3.3V) through an output pin in series with a 100-ohm resistor. The pin is then changed to an input pin and the firmware then times how long it takes for the I/O pin to switch from a logic one (high) to a logic zero (low). As soon as the I/O pin switches from output to input, the capacitor starts to discharge through the chain of resistors (e.g., C3 will discharge through R6 through R11). When no pushbutton in the chain is pushed, the discharge path is the sum of the individual resistors or 15K ohms. This is the longest possible discharge period. When one of the buttons is pressed, the discharge resistance is reduced from 15K to 1K, 2K, 3K, or 4K depending on which button is pressed. When the keyer is first powered up, the firmware determines the maximum discharge time for each row of switches and uses a pre-calculated percentage range of the maximum discharge time to determine which, if any, of the four pushbuttons is pressed. This calibrates the RC circuit and compensates for component value variations.

Commented [n5]: This is a very clever implementation Dave. Perhaps not novel (shown in app notes or elsewhere?), but such “design efficiency” is not normally seen in our QRP field. Nicely done, and good explanation.)

Once the pushbutton interface was reduced to two pins, connecting the remaining devices became pretty easy.

Rotary Encoder

The encoder used here is a mechanical encoder with 24 detents per revolution. As the knob is turned, two output pins (ENC-A and ENC-B) are momentarily connected and unconnected to a third pin (ENC-C) which is grounded. The grounding periods on the two pins (A and B) are offset so that when the control is turned **counterclockwise**, A is always grounded before B is grounded and A becomes ungrounded before B is ungrounded.

David M Collins, AD7JT
22111 N San Ramon Dr., Sun City West, AZ 85375; **ad7jt@cox.net**

When the control is turned **clockwise**, B is always grounded before A and B is ungrounded before A. By monitoring A and B, the firmware can tell when the control is being turned, which direction it is being turned, and how far it is turned. The firmware activates pull-up resistors on the I/O pins used for A and B so an ungrounded pin will raise to a high logic level.

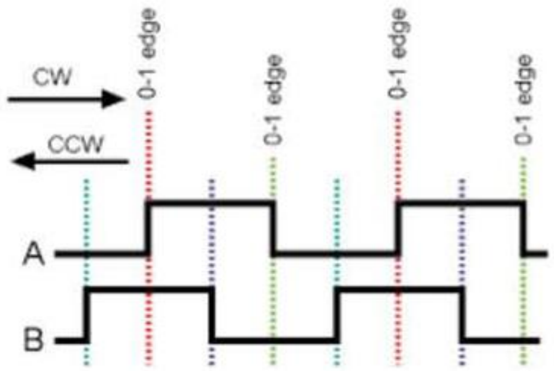


Figure 16 -- Rotary Encoder Output

The most common (and least expensive) rotary encoders are mechanical. These encoders have a simple three-wire interface (A, B, and C; no power required) and are available with and without detents. I prefer detents so the ultimate keyer uses a mechanical rotary encoder with 24 detents. The number of detents determines the number of A and B pulses per rotation of the control knob. The detents also determine the resting points for the knob. Normally the resting points are located such that neither A or B is grounded (both high or 1).

There are a couple problems with mechanical encoders with detents that need to be addressed in the firmware. The first problem is obvious, switch bounce. The second is a little more subtle, I call it "detent bounce". When you turn the control knob and release it at an arbitrary position, it may be between detents and will move to the nearest detent which may be in the direction you were turning the knob or in the opposite direction. Moving in opposite direction can cause a problem if not handled properly in the firmware. To resolvce both of these problems, the ultimate keyer firmware implements a simple state machine that essentially "knows" the direction the knob was turned and can recognize the bounce-back action and ignore it.

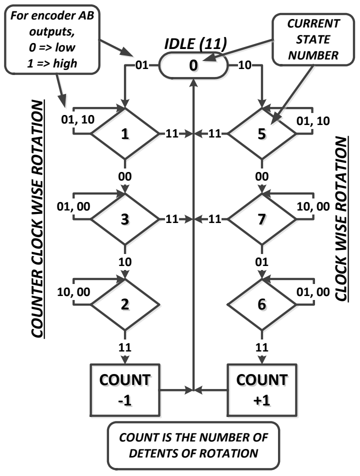


Figure 17 -- Encoder State Diagram

Commented [n6]: This is a very clever and relatively obscure solution. I like it!

David M Collins, AD7JT
22111 N San Ramon Dr., Sun City West, AZ 85375; **ad7jt@cox.net**

Front Panel Assembly

LED Display Module

Figure 18 illustrates the LED display module, the red lens, and how they are assembled on the ultimate keyer front panel. The red lens starts out as a piece of red acrylic plastic approcimately 1/8 inch thick. The plastic comes in one-foot square sheets which I cut to 3/4" strips with my table saw. I then use my router table to mill a 1/16 inch rabit along the long ends. I then cut each 12" strip into 1-3/8" pieces. I use a home made jig to hold the lenses while I mill rabits along the lens ends. The resulting raised portion of the lens fits into the rectangular cutout on the right side of the front panel.

The cutout the front panel is made with a router during the PCB fab process so the cutout will generally have rounded inside corners and may cause the lens to not lay flat all around the cutout. This can be fixed by either using a file to square the rounded corners in the cutout or to round the square corners on the raised part of the lens. The lens must also be modified by cutting 45 degree bevels at the two corners that will be on the right when the lens is mounted. This is to clear two of the sandoffs holding the LED display in place. It is not necessary to glue the lens in place. The 9mm standoffs leave less than 1/16" clearance between the display face and the lens; the lens is held in place by the display module itself.

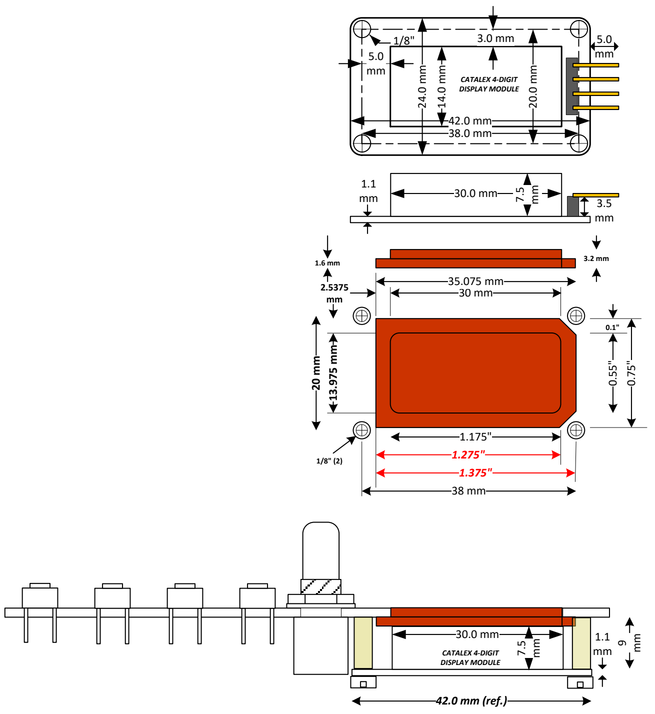


Figure 18 -- LED module, lens, and mounting detail

Once the display module is mounted, the four pins on the display connector are bent down to touch corresponding four pads on the back of the front panel. They are then soldered to the pads. See Figure 19.

Rotary Encoder

The rotary encoder mounts to the front panel with one nut and washer. The three pins for the encoder and two pins for the built-in push button must be bent down towards the SMT pads directly below the pins on the PCB. The pins are then soldered to the pads. If the pins are not long enough to reach the pads, small pieces of solid wire are used to bridge the gap. See Figures 19 and 20 for details.

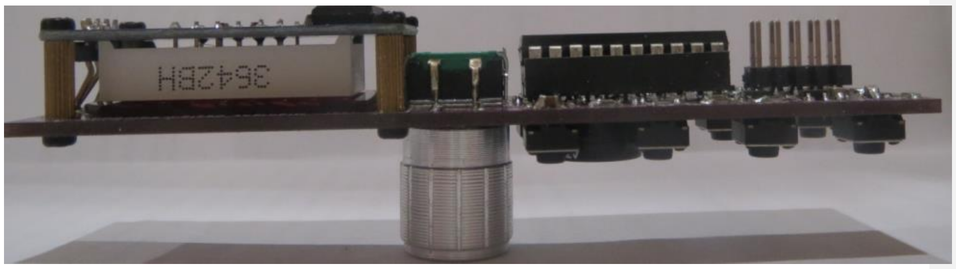


Figure 19 -- LED module and rotary encoder mountings.

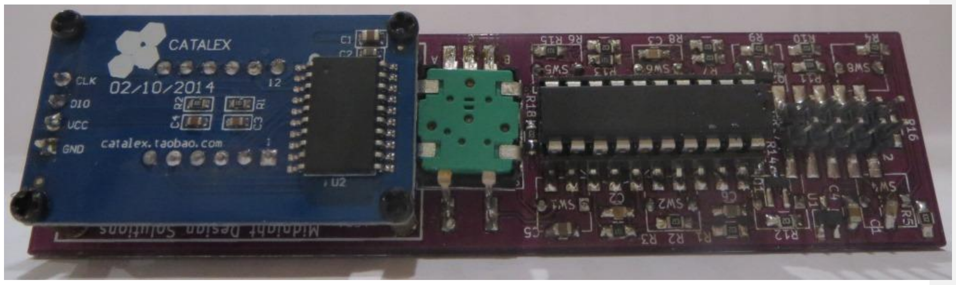


Figure 20 -- Front panel assembly, rear view.

Component Assembly Views

Figures 21 and 22 show more detail of the through-hole and SMD component mountings.

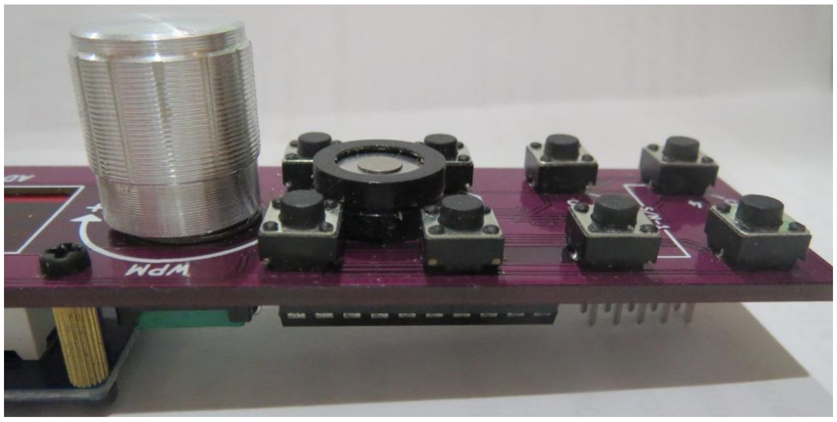


Figure 21 -- Front Panel, side view showing pushbutton switch and speaker mounting.

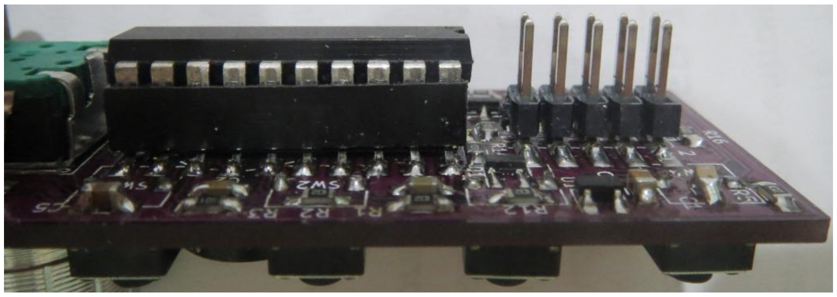


Figure 22 -- Front Panel, side view showing SMD parts mounting.

DISPLAY FONTS

Before getting into describing the ultimate keyer operation, a few words about the LED display are in order. This display is normally used in clocks to show time with its four seven-segment characters. Seven-segment displays were designed to only display decimal digits '0' through '9'. Later the "fonts" were expanded to also display the additional six characters needed to display hexadecimal numbers ('A', 'b', 'C', 'd', 'E' and 'F'). I've gone a bit further and defined seven-segment fonts for the remaining letters and several punctuation marks. Figure 21 illustrates all the decimal digits, all the letters, and a few punctuation marks. Note the font for '/' is used to represent WPM (W/M) when adjusting keyer speed.

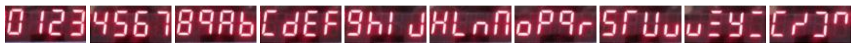


Figure 23-- 7-Segment character fonts for: 0123456789AbCdEFghIJKLmNoPqrSTUvwXyZ[/]^

The center colon is used in some displays following an abbreviation for a setting and preceding the current value for that setting. An underscore ('_') is occasionally used to represent a space. Keyed data is shown as it is being "transmitted" scrolling into the display from right to left.

OPERATION

The rotary encoder and its integral push button switch are used to setup and direct keyer operation. There is a total of 19 control states the keyer may be in. Think of an array of six columns each with three rows except one with four rows. Each point in the array represents one of the keyer's 19 control states. States are identified by name and by position in the array. The position is defined by the column number (0 through 5) and the row number (0 through 2 or 3) separated by a comma. For example, the upper left corner position is labeled "0,0", the lower right corner position is labeled "5,2". The control state diagram is shown in Figure 24.

Referring to the control state diagram, the keyer firmware always starts in control state 0,0 (first column, first row). In the first row (0), rotating the control knob will move between columns. When in this row, the display is dimmed slightly. Pressing and releasing the control knob moves down the columns, one row for each press and release. In any row except row zero, turning the control knob will generally change a parameter such as side tone frequency or keyer mode. Turning the control knob will change a parameter, the current value of the parameter is displayed, flashing, on the LED display. Pressing and releasing the control knob will confirm and capture the new setting. When the last (bottom) row in a column is reached, pressing and releasing the control knob will return to the top row (0).

At any time, in any control state, pressing the control knob and turning it at the same time will adjust the keyer's speed (WPM). The control state row is not changed when the control knob is released if the control knob has been turned to change the keyer speed.

Commented [n7]: This method only happens with the Option Card in place, or in a later FW version, right?

The following briefly describe operations available in various control states. In the display settings, underlined characters will be flashing. The displays for row zero are shown in parentheses after the column name.

COLUMN 0: OPERATE (_go_)

0,1	Display: keyed text	Rotation: adjusts keyer speed
0,2	Display: S<u>nnn</u>	Rotation: adjust QSO serial number (nnn)

COLUMN 1: MACRO ENTRY (FN:)

1,1	Display: FN:<u>nn</u>	Rotation: select macro as follows: 1 - 7, MC (My Call), or TC (Their Call)
1,2	Display: Keyed macro text	Rotation: backspace

COLUMN 2: SIDE TONE SETTINGS (ST:nn)

2,1	Display: ST:<u>nn</u>	Rotation: adjust side tone frequency (nn x 100 Hz)
2,2	Display: SP:<u>aa</u>	Rotation: turn keyer speaker on (aa = ON) and off (aa = OF)

COLUMN 3: MODES (KM:tr)

3,1 Display KM:<u>t</u>r — Rotation: change keyer mode (t) as follows:

A	Iambic A
B	Iambic B
D	Dot preferred
-	Dash preferred
U	Ultimatic

		S	Semi-automatic (bug mode)
		M	Manual (straight key)
3,2	Display KM:t<u>r</u>		Rotation: change paddle mode (r) as follows: Normal = n, Reversed = r
3,3	Display EN: <u>a</u>		Rotation: change encoder mode (Normal = n, Reversed = r)

COLUMN 4: PRACTICE MODE (PR:IN)

4,0	Display: keyed text (underscore = space)	
		Rotation: Change state column
4,1	Display: PR:<u>an</u>	Rotation: select text mode (Alpha only = A , Numeric only = N, Both = AN, Sequential alpha = Ab)
4,2	Display: FW:<u>nn</u>	Rotation: adjust keyer Farnsworth count.
		In this state, the keyer will generate Morse code in five letter groups at the current keyer rate. Except for text mode Ab, characters are selected at random from the character set selected in state 4,1. When text mode Ab is selected, only alpha characters are keyed in sequential order.

COLUMN 5: BAUDRATE (Bnnn)

5,1	Display: (B<u>nnn</u>)	Rotation: change serial port baud rate (nnn = baud rate ÷ 100 BPS)
5,2	NEW BAUD RATE IS CAPTURED AND SAVED IN EEPROM AND FIRMWARE RESTARTS.	

CONCLUSION

Currently my ultimate keyer meets the basic set of requirements listed at the start. It took me very little time to get used to the unorthodox font displayed on the seven-segement displays. The single control works well and I find its use to be very intuitive. The final configuration is small (3" x 4.5" x1.25") and light-weight (less than 5 ounces) taking up less desk space than a 3x5 index card. The small size makes it easy to press the push buttons using only one hand by pressing on the back of the enclosure with a couple fingers while pressing the button with my thumb. The tactile feel of the push button switches gives a sold feel and feedback when a switch closes and opens. When idle, the ultimate keyer draws only about 20 ma. With key down, it draws about 35 ma (plus about 100 ma if the speaker is on).

I do find that having to use something like a Bencher Iambic paddle to be inconvenient for field use. The paddle generally weighs more than the keyer and QRP rig combined. Therefore, I have added a touch key to the nice-to-have list and have been working on a simple design to mount it in/on the ultimate keyer's enclosure.

The Midnight Ultimate Keyer is available in kit form online[7] along with detailed user documentation and assembly instructions. The kit, including all components, PCB front and rear panels, and the enclosure is available for $34 plus shipping.

I am now off working on the nice-to-have list. In anticipation of required firmware changes, I have included a boot loader that can be used to update the firmware from an ASCII terminal or a PC running a terminal emulator. When I complete the nice-to-have list and, if the interest is there, I will make it the topic of a future, follow on article.

Notes:

[1] http://midnightdesignsolutions.com

[2] https://www.pactecenclosures.com/product-detail.php?productid=116&seriesid=48&classid=26

[3] http://www.gravitech.us/mipubusw2qt4.html

[4] https://www.mouser.com/ProductDetail/PUI-Audio/AT-1220-TT-2-R/?qs=%2fha2pyFadujeO6zYsUq71ZVWWcedJm5Ewkdpo0SVvj12e5V2bUhshQ%3d%3d

[5] http://www.microchip.com/ParamChartSearch/chart.aspx?branchID=8181

[6] http://kicad-pcb.org/

[7] http://midnightdesignsolutions.com/muk/index.html

David M Collins, AD7JT
22111 N San Ramon Dr., Sun City West, AZ 85375; **ad7jt@cox.net**

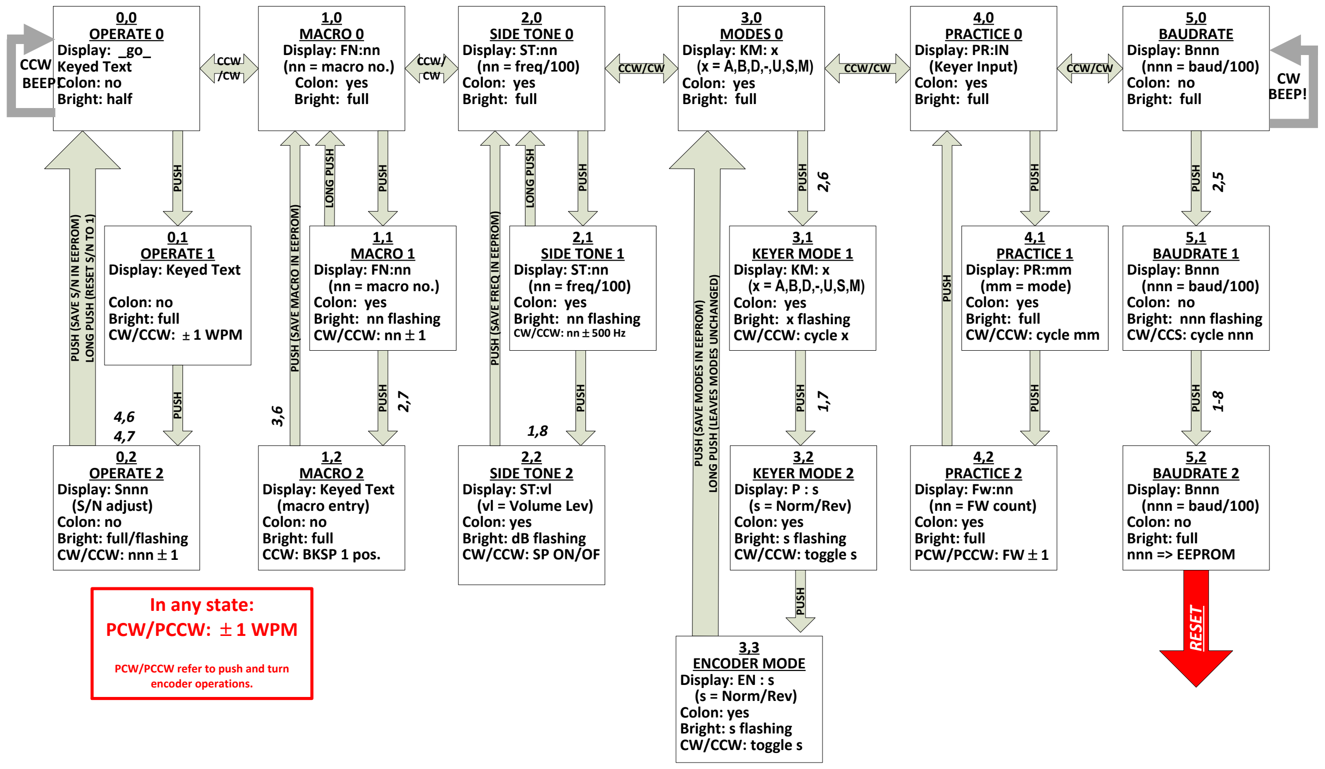

0,0 OPERATE 0 Display: _go_ Keyed Text Colon: no Bright: half
CCW BEEP!
CCW/CW
1,0 MACRO 0 Display: FN:nn (nn = macro no.) Colon: yes Bright: full
CCW/CW
2,0 SIDE TONE 0 Display: ST:nn (nn = freq/100) Colon: yes Bright: full
CCW/CW
3,0 MODES 0 Display: KM: x (x = A,B,D,-,U,S,M) Colon: yes Bright: full
CCW/CW
4,0 PRACTICE 0 Display: PR:IN (Keyer Input) Colon: yes Bright: full
CCW/CW
5,0 BAUDRATE Display: Bnnn (nnn = baud/100) Colon: no Bright: full
CW BEEP!
PUSH (SAVE S/N IN EEPROM) LONG PUSH (RESET S/N TO 1)
PUSH
0,1 OPERATE 1 Display: Keyed Text Colon: no Bright: full CW/CCW: ± 1 WPM
PUSH
4,6 4,7
0,2 OPERATE 2 Display: Snnn (S/N adjust) Colon: no Bright: full/flashing CW/CCW: nnn ± 1
PUSH (SAVE MACRO IN EEPROM)
LONG PUSH
PUSH
1,1 MACRO 1 Display: FN:nn (nn = macro no.) Colon: yes Bright: nn flashing CW/CCW: nn ± 1
PUSH
2,7
3,6
1,2 MACRO 2 Display: Keyed Text (macro entry) Colon: no Bright: full CCW: BKSP 1 pos.
PUSH (SAVE FREQ IN EEPROM)
LONG PUSH
PUSH
2,1 SIDE TONE 1 Display: ST:nn (nn = freq/100) Colon: yes Bright: nn flashing CW/CCW: nn ± 500 Hz
PUSH
1,8
2,2 SIDE TONE 2 Display: ST:vl (vl = Volume Lev) Colon: yes Bright: dB flashing CW/CCW: SP ON/OF
PUSH (SAVE MODES IN EEPROM) LONG PUSH (LEAVES MODES UNCHANGED)
PUSH 2,6
3,1 KEYER MODE 1 Display: KM: x (x = A,B,D,-,U,S,M) Colon: yes Bright: x flashing CW/CCW: cycle x
PUSH 1,7
3,2 KEYER MODE 2 Display: P : s (s = Norm/Rev) Colon: yes Bright: s flashing CW/CCW: toggle s
PUSH
3,3 ENCODER MODE Display: EN : s (s = Norm/Rev) Colon: yes Bright: s flashing CW/CCW: toggle s
PUSH
PUSH
4,1 PRACTICE 1 Display: PR:mm (mm = mode) Colon: yes Bright: full CW/CCW: cycle mm
PUSH
4,2 PRACTICE 2 Display: Fw:nn (nn = FW count) Colon: yes Bright: full PCW/PCCW: FW ± 1
PUSH 2,5
5,1 BAUDRATE 1 Display: Bnnn (nnn = baud/100) Colon: no Bright: nnn flashing CW/CCS: cycle nnn
PUSH 1-8
5,2 BAUDRATE 2 Display: Bnnn (nnn = baud/100) Colon: no Bright: full nnn => EEPROM
RESET
In any state: PCW/PCCW: ± 1 WPM
PCW/PCCW refer to push and turn encoder operations.

Figure 24 Ultimate Keyer Control State Diagram