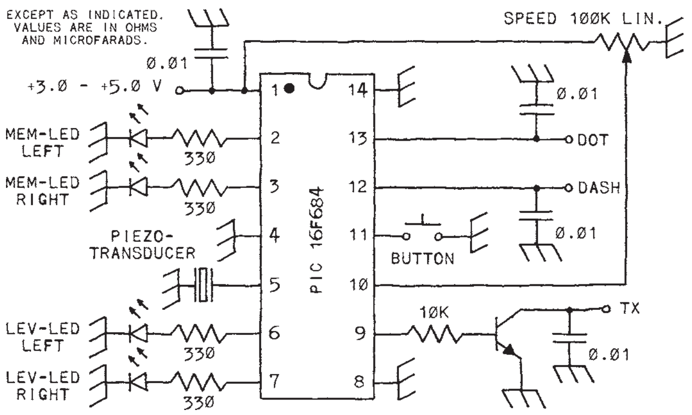
Keyrama

A unique PIC-based multi-mode Morse code keyer



Radio amateurs invented and pioneered electronic Morse code keyers, but today their knowledge of the different twin-lever keying modes is sparse. Because I could not find one single document with a competent and proper treatment of that topic I wrote *"all about squeeze-keying"* [1], and you should carefully read it if you really want to understand these modes and make the most of this keyer. The suffix "-rama" stems from the Ancient Greek word "οραμα" which means "wide view". In order to complement my treatise with a useful practical device I developed *"Keyrama"* which enables the operator to get a wide view of the different keying modes, to compare its correct emulation and timing of the original twin-lever modes with the behaviour of other keyers, to follow the visualized action of the dot/dash-memory and to find out to which extent his specific keying technique really makes use of it.

features

* emulation of the following keying modes:
  - plain iambic (K8OCO)
  - iambic type A (K6KU "Curtis-keyer")
  - iambic type B (WB4VVF "Accu-keyer")
  - ultimatic (W6SRY)
  - single-dot (W0EPV)
  - el-bug (single-lever electronic keyer)
  - bug (semi-automatic key)
  - cootie, sideswiper (double-speed key)
* dot/dash-memory
* autospace (inter-character and inter-word)
* keying speed 6 - 60 wpm
* weight 25 - 75 %
* dash-length 50 - 250 %
* two message-memories for max. 80 / 160 characters with editing function
* message loop function with adjustable delay
* dot/dash-levers reversable
* adjustable debouncing of the lever-contacts
* visualized keyer action
* highly accurate timing
* controlled by one push-button and the paddle
* supply voltage range +3.0 to +5.5 V
* low current drain (< 0.2 mA in sleep-mode)

construction

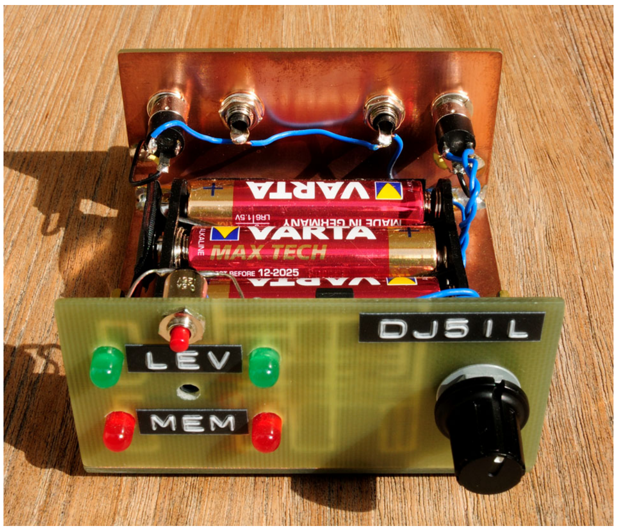
The keyer schematic is shown in the above figure. Programmed PICs 16F684 are available from the author, the firmware hex-file can be downloaded for personal non-commercial use [2]. The transistor is any general-purpose NPN type. For voltage supply I recommend 3 AA batteries giving 4.5 V, the low current drain of approx. 0.2 mA in sleep-mode makes a switch unnecessary.

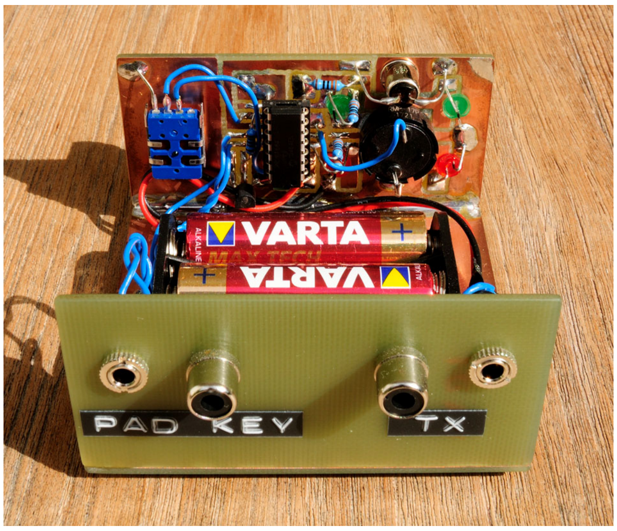
A piezo-transducer (not a piezo-buzzer !) is used because of its high impedance and low current drain, its schematic symbol is the same as for a quartz crystal. Use any transducer size that is available, the bigger the better, but do not replace it by a conventional loudspeaker ! The keyer is controlled by a single momentary push-button switch and the paddle. A linear taper potentiometer must be used for speed control. "DOT" and "DASH" go to the associated keying-lever contacts, "TX" goes to the keying jack of the transmitter.

In order to get a good visualization of keyer action the four LEDs should be arranged on the front plate like the corners of a rectangle, with the two LEV-LEDs of one color (e.g. green) in the upper two corners and the two MEM-LEDs of another color (e.g. red) in the lower two corners. Bright 5mm types are recommended. LEDs have one longer and one shorter lead, the shorter lead (cathode) goes to ground. During initial setup the LED-assignment (parameter-code "L") must be set so that the left LEV-LED lights up when the left lever is pressed.

If you are not interested in the visualization you can omit the four LEDs and 330 Ohm resistors and leave the pins # 2,3,6,7 unconnected, but do not ground these pins ! The keyer can be built and operated even without the speed pot, in that case pin # 10 must be grounded. The following chapter explains how the keying speed is varied with the push-button and paddle.

Karl Fischer, DJ5IL, Friedenstr. 42, 75173 Pforzheim, Germany, DJ5IL@cq-cq.eu, www.cq-cq.eu





Because the keyer circuit is so simple, no circuit board layout is presented here. Instead of an etched board you may use a small piece of perf-board, or you take single-sided PCB (printed circuit board) material and mill out the outlines of the copper traces with a dremel tool as I did with my prototype shown without cover on the above pictures. This is my preferred construction method for simple circuits and I also build sturdy cabinets from solder-joined plates of this material. In this case I simply used the copper-clad backside of the front plate as the circuit board.

Please note the arrangement of the four LEDs. The piezo-transducer is glued to the circuit board in the middle between the four LEDs below the red push-button. The sound is emitted through a small hole in the transducer's plastic cover, a slightly larger hole was drilled through the front plate.

operation

After power is applied, the keyer responds with "OK" and is ready for operation. The parameter- and message-memories are non-volatile. If any permanent malfunction should occur hold a lever pressed while power is applied, this procedure resets all parameters to their default values.

To **inquire or set the keying mode** press and hold the button, a "?" followed by the code of the currently selected mode is played:

```
IA = iambic type A (Curtis-keyer)
IB = iambic type B (Accu-keyer)
U  = ultimatic
S  = single-dot
E  = el-bug
B  = bug
C  = cootie, sideswiper
```

Keeping the button pressed, the next / previous mode is selected by a short tap of the dot / dash lever and after releasing the lever its code is played. Release the button to resume normal operation.

When the single-lever modes E / B / C are emulated with a twin-lever paddle and the levers are squeezed, only the lever which was pressed first is recognized as pressed because with a single-lever paddle both contacts cannot be closed at the same time. In these modes single-lever paddles work much better because twin-lever paddles provoke curtailed or even missing spaces (one lever is already pressed while the other is not yet released).

To **inquire or set a parameter** (not possible in the modes B / C) press the button and release it while a "?" is played. The dot/dash lever-assignment can only be set but not inquired. It is toggled by entering more than 7 consecutive dots or dashes, an "N" for normal or an "R" for reversed is played. To set any other parameter or to load a message into memory, enter its code according to the parameter table immediately followed by the value or message to be assigned to the parameter or memory. The pause between code and value or message must be shorter than an inter-word space (7 dot lengths). The keyer rejects invalid actions or inputs with a fast error-sign (8 dots). Valid parameter values are set and normal operation is resumed without any signal. In order to inquire a parameter or message just wait after entering its code until its value or text is played. Some examples: "W55" sets the weight to 55%, "S25" sets the keying speed to 25 wpm at the current position of the speed control, "T1" sets the monitor tone on, "A" inquires the auto-space setting, "2" inquires message-memory #2. Inquiry of a message-memory is aborted by a short tap of a lever.

The effect of **autospace** depends on the keying mode. In the modes IA / IB / U / S / E it prevents character crowding by forcing an inter-character space of at least 3 dot lengths whenever a pause longer than one dot length is detected. If autospace is set to inter-character *and* inter-word, the correct inter-character space of 3 dot-lengths may be prolonged by not more than one dot length, and during that period the space can be cut by pressing a lever. However, if it is *not* cut an inter-word space of at least 7 dot lengths is forced. Without that tolerance of one dot length the operator would be forced and hence get used to apply inter-character spaces which are too short, because otherwise they would inevitably become inter-word spaces. In the modes B / C autospace simply prevents curtailed spaces within a character by forcing at least one inter-element space (1 dot length) whenever the lever contact is changed. The speed control varies in mode C the length of the inter-element space only, in mode B also the dot length. The **dash-variator** varies the dash length, which is normally 3 dot lengths. When a lever-contact changes its state from open to closed or vice versa, the new state is maintained and subsequent changes are ignored until the **hold-time** has elapsed, which results in adjustable *debouncing* of the lever-contacts. The **LED-assignment** must be set so that the left LEV-LED is on when the left lever is pressed. The **dot/dash-memory** can be set without restrictions, but if you want to emulate the exact behaviour of the original modes IA / IB / U the dot-*and* dash-memory must be set on while in mode S *only* the dot-memory must be set on. With this setting in modes IA / IB while an element is generated directly or out of the memory one single opposite element can be stored; contrary to that in mode U one dot *and* one dash and in mode S one dot can be stored at any time, even while the same element is being generated. For plain iambic keying select mode IA or IB and turn off the dot/dash-memory ("M0"). The **speed** parameter sets the keying speed at the current position of the speed control, which allows for a speed variation of 25 wpm independent of the speed parameter setting. The speed control varies in mode C the length of the inter-element space only, in mode B the dot length as well. Take care not to set the speed too high, as you might not be able to set it back. The keying speed can also be easily varied even without the speed control pot: press and hold the button and quickly (within less than 0.4 s) press a lever, a dash/dot-sequence is played as long as the button is held pressed and the keying speed can be increased or decreased with the dot- or dash lever. Irrespective of its parameter setting, the monitor **tone** is always on while inquiring or setting the keying mode or a parameter. The keying **weight** is the duty cycle of a series of consecutive dots, which is normally 50% because dots and spaces have the same length. Higher values produce longer dots and shorter spaces ("heavier" keying), whereas lower values produce shorter dots and longer spaces ("lighter" keying). If the dash-variator is set to the normal 100% the dash length is varied by the same period as the dot length, otherwise the weight has no influence on the modified dash length. When inquiring or setting the keying speed, please note that its value in wpm is independent of weight but only valid if the dash-variator is set to 100%.

To **load a message** into memory proceed according to the following rules: characters with more than 7 elements (dot + space or dash + space) are invalid and rejected. A pause longer than an inter-character space (3 dot lengths) is recognized and stored as an inter-word space and confirmed by a short "pip". There is no limit to the elapsed time between words, so there is no need to hurry and only one inter-word space is written into memory. Entering more than 7 consecutive dots is recognized as an error-sign, the character previously stored is re-played and erased from memory. In that way the whole memory can be edited backwards, character by character. If the memory shall be played in an endless loop, enter a colon (---...) at the end of the text followed by the 3-digit delay-time in tenths of a second with leading zeros (max. 999 = 99.9 seconds). While entering, a pause of any length is allowed after the colon and between the delay-time digits. For example, entering "1CQ CQ CQ DE DJ5IL K: 035" loads the message "CQ CQ CQ DE DJ5IL K" into memory #1 and causes the message to be played in a loop with 3.5 seconds delay. When the memory is full, an error-sign and an "F" are played. The capacity is 80 characters for memory #1 and 160 characters for memory #2. Pressing the button ends message loading, after release an "R" is played and normal operation is resumed.

Start playing of message #1 with one short tap (less than 0.4 s) of the button, of message #2 with two short taps in fast succession (pause between taps less than 0.4 s). If a fast error-sign is played at the end of the message, it contains an invalid delay time. Playing is aborted either immediately by pressing a lever, or after the currently played character by pressing and holding the button. Normal operation is resumed after the lever or button is released.

debouncing

Contact "bouncing" is a common problem with all unwetted mechanical contacts, even those of high-quality keyer paddles. Because moving contacts have mass and springiness with low damping, they will be bouncy as they make and break. That is, when an open pair of contacts is closed the contacts will come together and bounce off each other several times over a period of about 2 to 10 ms before coming to a rest in a closed position. Note that contact bouncing is common on closing and uncommon but also possible on opening.

The following tests reveal the bouncing behaviour of your lever-contacts and allow for adjustment of the hold-time (parameter "H") which controls software-debouncing. Set the hold-time initially to 0 ms ("H0") so that debouncing is inactive, set the dot- and dash-memory on ("M3") and turn the speed down. Then ...

Test #1: set the keyer to mode IA, squeeze both levers and hold for a while, then release both levers while a dash (dot) is generated. If an additional dot (dash) is stored and generated the dot- (dash-) contact bounces on OPENING. Bouncing on opening of a lever-contact causes malfunction in modes IA / U / S if the associated dot/dash-memory is set on, but it does not affect mode IB.

Test #2: set the keyer to mode U and apply a short tap to the dot- (dash-) lever. If a dot (dash) is stored and therefore two dots (dashes) are generated the dot- (dash-) contact bounces on CLOSING. Bouncing on closing causes malfunction in modes U / S if the associated dot/dash-memory is on, but it does not affect modes IA / IB.

If these tests reveal that the lever-contacts are bouncing, the hold-time should be gradually increased (max. 25 ms) in small steps until the malfunction disappears. In mode B with autospace bouncing contacts cause an initial short klick and element space to be generated when the lever is pressed.

what the LEDs indicate

The keyer is basically able to generate two types of character elements: a *dot-element* (dot + space) or a *dash-element* (dash + space). A **LEV**-LED lights up when the associated LEVer is polled and in the state "pressed" (no element can be in progress at this moment), so that the generation of that element is triggered. Provided its memory is on (check the dot/dash-memory setting), a **MEM**-LED lights up when the associated MEMory is set because that lever changed its state from "unpressed" to "pressed" (modes IA / U / S / E) or because it just was in the state "pressed" (mode IB) sometime during generation of an element. A LEV-LED or MEM-LED goes out as soon as the generation of the associated element is completed.

The following interpretation of the LEDs allows to judge the timing of your keying and exhibits to what extend you really make use of and need support from the dot-/dash-memory. Please note that it refers to the currently generated element and its associated lever and vertical column of LEDs, which is either the element triggered by the left or the right lever and its associated left or right LEV-LED together with the MEM-LED below:

LEV on / MEM off: the element has been triggered directly by the pressed lever without any action of the dot- or dash-memory.

LEV on / MEM on: the lever was pressed too early, but it was held long enough so that the element would have been triggered directly even without the set memory. When this condition occurs, the operator does not really make use of the associated dot- or dash-memory.

LEV off / MEM on: the lever was pressed and released too early, so that the element has been triggered indirectly by the set memory. When this condition occurs, the operator really makes use of the associated dot- or dash-memory.

An improved PIC-based capacitive sensor-paddle without moving parts that works perfectly together with this Morse code keyer is described in my article *"CapKey+"* [3].

parameter table

Code	Description [default value]
A	**Autospace** [0] 0 = off 1 = inter-character 2 = inter-character and inter-word
D	**Dash-variator** [100] 50 - 250 (%)
H	**Hold-time** [0] 0 - 25 (ms)
L	**LED-assignment** [0] 0 = normal 1 = reversed
M	**dot/dash-Memory** [0] 0 = off 1 = dot-memory 2 = dash-memory 3 = dot- and dash-memory
S	**Speed** 6 - 60 (wpm)
T	**monitor Tone** [1] 0 = off 1 = on
V	**Volume** [5] 3 - 10
W	**Weight** [50] 25 - 75 (%)
1	**message-memory #1**
2	**message-memory #2**
	lever-assignment [N] to toggle enter more than 7 dots or dashes: N = normal, R = reversed

references

1. http://cq-cq.eu/DJ5IL_rt007.pdf (click URL to open)
2. http://cq-cq.eu/keyrama.hex (click URL to download)
3. http://cq-cq.eu/DJ5IL_rt011.pdf (click URL to open)

DJ5IL_rt008.pdf
Original version: 4.10.2016
Revisions: 20.10.2016, 3.4.2018, 1.4.2019, 30.12.2019, 24.5.2020, 8.6.2020