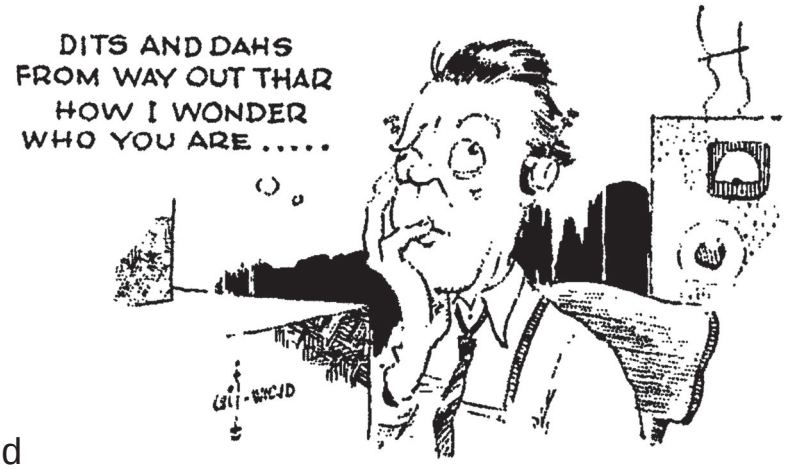
All about Squeeze-Keying



Radio amateurs invented and pioneered electronic Morse code keyers, but today their knowledge of the different twin-lever keying modes is sparse. Here is a review and thorough explanation ...

ultimatic mode

Basically an electronic Morse code keyer is able to generate two different character elements: a *dot-element* (dot + space) or a *dash-element* (dash + space). Please note that the space following a dot or dash is part of the element. In 1951an electronic single-lever key was described [1] which used 5 tubes to send self-completing dot- and dash-elements with automatic spacing between letters and words. But its continuously running time-base resulted in an uncontrollable beast so that the author himself wrote he does *"not feel that any but the most feverish electronic key enthusiasts will wish to build one of these infernal, maddening machines"*, but nevertheless he hoped that the idea might provide an inspiration for further development.

John Kaye, W6SRY, accepted that challenge and came up with a rather ingenious design which he published as the *Ultimatic* key in 1953 [2] . The circuit is based on 3 tubes and 7 relays and sticks to the basic idea of a continously running time-base, which is the only weak spot of his design: pulses from the time-base trigger the generation of dot- and dash-elements, and so they do not start immediately with the closure of a key contact but only with the next pulse.

By addition of *dot/dash-memories* he avoids dropping of leading elements and transforms the beast into a beauty: once a contact of the single-lever key has been closed, that closure is retained by a memory-relay contact parallel with the key contact and the related dot- or dash-element is properly generated as soon as the trigger pulse arrives, even if that key contact is open again or the opposite key contact is closed by then. The dot/dash-memory relays are reset and their contacts opened by the closing contacts of the related dot/dash-generator relays. The dot/dash-memories are independent of each other and because a dot and dash often are rapidly stored together before keying starts, a *sequencing* circuit retains the proper order in which the dot/dash-generators are triggered. This combination of independent *dot/dash-memories* with *sequencing* avoids dropping of leading elements by allowing the storage not only of a single dot or dash but of a whole dot + dash or dash + dot sequence before a character commences. And in addition it provides tremendous timing leeway when an alternate element is to be inserted or appended. Together with automatic letter spacing the result is an amazing ease of operation, or as W6SRY put it: *"with the key set for 10 w.p.m., you can hit a 40 w.p.m. 'N' and walk away while the key produces a slow 'daah-dit'..."*

While this initial design still used a single key lever, his next version which appeared in 1955 [3] was the first twin-lever electronic keyer and ancestor of the modern squeeze-keyers which we use today, and it is this key's action that gave the "ultimatic" mode its name. The circuit is based on 11 tubes and only one relay and the time-base, memory and sequencer are functionally identical to the first version. But because contrary to a single-lever both contacts of a twin-lever key can be closed at the same time, a *seizure* circuitry was added: whenever a lever makes contact, it seizes control and the subsequent elements correspond to that lever until the other lever makes contact or the lever is released.

While one lever contact is closed the twin-lever Ultimatic generates a string of dot- or dash-elements, exactly like any single-lever keyer does. However, when the levers are squeezed so that both contacts are closed, it generates a string of elements

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

from whichever lever was pressed last. So any closure of a lever contact guarantees at least one element of that type, generated in correct relationship to the order of closure. For example, to key an "X" press the dash-lever for the first dash and hold it, then press the dot-lever (squeeze both levers) for the middle two dots, release the dot-lever for the last dash and finally release the dash-lever. This key can be treated as if it were a semi-automatic "bug" key or a single-lever electronic keyer or with any intermediate technique. And at that time it was considered the "ultimate" key because it sent perfect code without the need for the operator to send it perfectly, or in the words of W6SRY *"a key that gives Klein output with Lake Erie input. It does everything for the operator but spell and punctuate"* [1].

The ultimatic mode is most effective for characters with leading or trailing strings of two or more identical elements. The letters B D G J U V W Z and the digits 1 to 4 and 6 to 9 can be keyed with one single squeeze of both levers just by proper timing of the second contact closure whilst keeping the first closure, no need to let it go. Let us define a stroke as the closure of a lever contact, so that a squeeze of both levers counts two strokes. Then all letters of the alphabet, except for the "C", and all digits can be generated with just one or two strokes. In 1960 Alvin F. Kanada, K0MHU, described his transistorized version [4] of that "key with a brain". The ultimatic mode has been largely eclipsed by the iambic mode, which came up in the late 1960s.

basic iambic mode

The vast majority of today's electronic twin-lever Morse code keyers operates in *iambic* mode, derived from the iambus which is a metrical foot in poetry with alternating short and long syllables like "dah-di-dah-di-dah". In 1967 Harry Gensler Jr., K8OCO, described his *Iambimatic* keying concept [5] together with an adapter for the Hallicrafters HA-1 single-lever keyer. Pressing one lever generates a string of dot- or dash-elements only, exactly as in ultimatic mode. But contrary to that, squeezing both levers generates a string of alternating dot- and dash-elements with the commencing element corresponding to the lever which was hit first. So basic iambic keying with self-completing dots and dashes can be generated by executing this simple set of instructions: *poll both levers alternately, if the lever is pressed generate the corresponding element and continue polling*.

If the squeezed levers are released while an element is in progress, a basic iambic keyer simply completes that element. So in order to key a "C" the dash-lever is pressed first, immediately followed by the dot-lever, and both squeezed levers are released sometime between the onset of the last dot and the end of the following space. The iambic mode is most effective for characters with alternating elements. All characters of the alphabet, except for the "P" and "X", and all digits can be generated with just one or two strokes. However, only the "C" needs less strokes than in ultimatic mode.

In 1967 Hermann Samson, DJ2BW, started to manufacture his ETM series of electronic keyers, developed by radio telegraphy instructor Klaus Duhme from the maritime school in Elsfleth / Germany. Their first twin-lever squeeze keyer was the model ETM-3.

photo DC7XJ

ETM keyers became very popular among radio amateurs in Germany and neighbouring countries and were also used by the maritime as well as other "special" radio services. They had no dot/dash-memory until 1983 when the model ETM-5C appeared, and so I learned the basic iambic mode in the early 1970s as a teenaged novice radio amateur. It forced me to ingrain proper element sequence timing within characters and also helped me to develop the feeling for proper intercharacter and interword spacing. However, asking fellow radio telegraphy operators it turns out that today this seems to be a most unusual keying mode (see the survey presented later).

Comparing the number of strokes necessary to key all letters of the alphabet and all digits with different keys and keyers yields the following result, provided the operator is consistently squeezing with

1. The meaning of **"Klein output"** was a mystery to me until Mort Mortimer, G2JL, mentioned the name "Kleinschmidt" which was the crucial hint. "Klein" is short for *Edward Ernst Kleinschmidt* who was born in Germany in 1876 and immigrated to the United States at the age of 8. While still in his teens, he first patented a Morse keyboard transmitter in 1895, and so "Klein output" obviously stands for perfectly timed Morse code. In 1924 Kleinschmidt and the Morkrum Company decided to merge, later the company name was changed to "Teletype Corporation" and in 1930 it was sold to the American Telephone and Telegraph Company for $30 million. Kleinschmidt, a prolific inventor who obtained 118 patents, died in Canaan, Connecticut, in 1977 at the age of 101. The so-called **"Lake Erie Swing"** was a typical semi-automatic "bug" keying style of marine operators on the Great Lakes, characterized by short dots and dashes of exaggerated and varying length. This made for a somewhat melodic and musical sound, quite pleasant to copy once one got the hang of it. It was later adopted by many airline and police CW operators.

twin-lever keyers:

```
straight key ............... 132
sideswiper or cootie key ... 132
semi-automatic "bug" key ... 100
single-lever keyer .........  73
iambic keyer ...............  65
ultimatic keyer ............  64
```

the Curtis-keyer

The first iambic keyer "Electronic Fist" EK-38 by John Curtis, K6KU, which appeared on the market in 1969, already extended that basic iambic logic by a *dot-memory*. As we already know, this feature was originally developed by W6SRY, but his very ambitious Ultimatic key did not gain too much popularity. Then it was reinvented by Dave Muir, W2YVO, who recognized the problem of dropping single embedded or final dots e.g. in letters like "K" or "G" because the operator is too quick (continuously running time-bases were outdated and hence dropping of leading elements was no more a problem). In 1962 he filled the gap between simple circuits and the Ultimatic with his *Penultimatic* single-lever electronic keyer [6]. Because it is more likely to press and release the short dot too early during the long dash than the long dash during the short dot, the first Curtis-keyer had a dot-memory only but no dash-memory exactly like the single-lever keyer by W2YVO. In 1973 John Curtis brought out the 8043 CMOS chip, the first integrated-circuit iambic keyer with dot-memory.

Of course no logic is able to foresee and hence it is impossible to compensate for our finger movement being too slow - but logic can remember and so it can compensate for being too quick. The behaviour of the Curtis-keyer can be emulated by the basic iambic set of instructions together with the following simple dot-memory rule: *if anytime during generation of a dash-element the dot-lever was hit, generate one extra dot-element.* This rule forces one single extra dot-element after completion of a dash-element if the dot-lever was hit and released too early before the dash-element was completed. To accomplish this, the dot-lever is polled not just for its state and not only when no element is in progress as in basic iambic mode, but for a *change of state* and constantly during generation of a dash-element. That change of state is remembered until the dash-element is completed, and then if the memory is set one extra dot-element is generated and the memory is cleared. If the dot-lever is hit but not released during the dash-element, the memory is unnecessarily set because the following dot-element is still triggered by the pressed lever.

The sole purpose of the dot-memory is to increase the dot-lever timing tolerance by allowing its too early pressure and release when a single dot is to be inserted into or appended to a string of dashes. But when the keyer is manipulated with proper timing, it should behave exactly like a basic iambic keyer. To show you how the Curtis dot-memory works and that it perfectly meets this requirement, suppose you want to key an "N": You press the dash-lever first to start the dash-element. Then you can either release it and press the dot-lever (single-lever keying) or hold it and press the dot-lever (squeeze keying). Then you can either release the lever(s) before the dash-element is completed and the keyer will append a dot-element to key the "N" by utilizing its dot-memory. Or you can hold the lever(s) until the dot-element is in progress and then release to key the "N" with proper timing as in basic iambic mode. So there are four possible keying methods to get an "N" out of the Curtis-keyer.

the Accu-keyer

The *Accu-keyer* by James Garrett, WB4VVF, featuring dot- and dash-memory as well as automatic character spacing, was published [7] shortly after John Curtis' 8043 chip appeared. The dot/dash-memories of the original circuit are TTL latches which are set *always* when the related lever is pressed and cleared *only* when that lever is released *and* the related element is in progress. Not the levers directly but only their memories are polled in order to decide which element is to be generated. Therefore the whole operation of the Accu-keyer is based on its memory logic, and that's why it has both a dot- and a dash-memory. When an element is completed, the logic assumes that another element must always follow if a memory is set. So pressing and holding only one lever produces a string of corresponding elements, when it is released its memory is immediately cleared by the element in progress and no additional element is generated after its completion. However, if both squeezed levers are released the memory opposite to the current element remains set and one surplus alternate element is generated.

Because while an element is generated the state of the corresponding lever is absolutely irrelevant for the behaviour of the Accu-keyer, it can be emulated by the basic iambic set of instructions together with the following simple dot/dash-memory

rule: *if anytime during generation of an element the opposite lever was pressed, generate one extra alternate element*. So neglecting the fact that the Accu-keyer has both a dot- and a dash-memory, the only procedural difference is that it just remembers the *state "pressed"* while the Curtis-keyer remembers the *change of state or transient from "unpressed" to "pressed"* of an opposing lever during generation of an element, and if that happened both keyers generate one extra alternate element.

Though this subtle difference between both memory rules at first glance seems negligible, it has a profound and detrimental side-effect which I already explained before: while the Accu-keyer offers the same timing tolerance as the Curtis-keyer when a single alternate element is to be inserted, contrary to the Curtis-keyer it does not allow to hold the squeeze as long as necessary with proper timing in basic iambic mode. Because if you do so, it remembers a pressed lever and generates an unwanted extra alternate element.

Before we continue, let's have a look at the correct timing of Morse code. The keying speed can be expressed either in WPM (words per minute) or in CPM (characters per minute) with CPM = 5 x WPM. From that we can calculate the dot length, which is the basic timing unit, and deduce the space (pause between dots or dashes within the same character), intercharacter space (pause between characters) and interword space (pause between words) in milliseconds (ms) as follows:

```
dot length = 1200 ms / WPM
           = 6000 ms / CPM
space = dot length
dash length = 3 x dot lenght
intercharacter space = dash length
interword space = 7 x dot length
```

The problematic memory logic of the Accu-keyer affects all characters ending on dot + dash or dash + dot. With a basic iambic or Curtis-keyer both squeezed levers must be released only before the space following the last dot or dash ends, and the higher the keying speed the more beneficial this large timing tolerance is. However, with an Accu-keyer the squeeze must be released much earlier before the last dot or dash starts. The most problematic of these characters is the "A" which frequently becomes an "R" if the dot-lever is not released during the short initial dot-element. Less frequently - because there is more time to release the dash-lever during the long initial dash-element - an "N" becomes a "K".

Comparing the maximum time window allowed to hold a squeeze yields an interesting result: it is identical for the basic iambic mode and for the Curtis-keyer, but for the Accu-keyer it is reduced by the length of the last element of the character. For example, at a speed of 30 WPM a dot or space is 40 ms and a dash 120 ms long. Squeezing an "A" you have 240 ms (dot + space + dash + space) to release both levers in basic iambic mode and with the Curtis-keyer but only 80 ms (dot + space) or 33% thereof with the Accu-keyer, otherwise you get an "R". Squeezing a "K" you have 400 ms to release both levers in basic iambic mode and with the Curtis-keyer but only 240 ms or 60% thereof with the Accu-keyer, otherwise you get a "C".

It follows that the Accu-keyer does not behave like a basic iambic keyer when it is manipulated with proper iambic timing. And its dot/dash-memory logic in fact does not increase timing tolerance, but instead even boosts the possibility for errors with increasing speed because it requires a much faster release of the squeeze. In view of this analysis, I think it is fair to call the Accu-keyer's dot/dash-memory logic a major design flaw. Nevertheless WB4VVF has sold thousands of printed circuit boards and his keyer became so popular - especially in the USA - that its behaviour was adopted for keyers made by major manufacturers of amateur radio equipment.

iambic type A and B

In 1975 the 8044 chip was introduced by John Curtis, an improved version of the 8043 with dot- and dash-memory. At that time most telegraphy operators already used iambic keyers - but scarcely anybody in basic iambic mode without dot/dash-memory, because neither the Curtis-keyer nor the Accu-keyer allowed to disable that feature. So over the years two schools of iambic keying developed, differing only in the dot/dash-memory logic which the operators initially learned but rarely scrutinized or even changed: Curtis-keyer and Accu-keyer. In light of that fact, Curtis named his own logic iambic *type A* and that of the Accu-keyer iambic *type B* and in 1986 he introduced the 8044ABM chip which offered selectable A or B type of iambic keying.

Those who learned type A (Curtis-keyer) are usually unable to master type B (Accu-keyer) and vice versa. And while both groups are usually unable to master basic iambic keying, those who learned it should have absolutely no problem with type A. When

both squeezed levers are released, a type A keyer simply completes the element in progress whereas a type B keyer generates an extra alternate element. That's how the difference between the two iambic types is usually explained and this simple explanation is true. But it is incomplete, because it merely describes an effect of type B without explaining its single cause: the inferior dot/dash-memory logic which was described in detail before. This annoying surplus element can be avoided by not squeezing or by using a single-lever paddle, but of course that is not the intended mode of operation for any iambic keyer. If the dot/dash-memory in type A and type B could be disabled, which is normally not the case, both types would behave absolutely identical and boil down to basic iambic keying.

To test a keyer for dot/dash-memory and its iambic type, set the speed as low as any possible and key an "N" as fast as possible - both levers must be released before the dash-element is completed ! With dot-memory the dash is always followed by a dot and you get the "N", without dot-memory the dot is lost and you get a "T" instead. Now key an "A" as fast as possible - again, both levers must be released before the dot-element is completed ! With dash-memory the dot is always followed by a dash and you get the "A", without dash-memory the dash is lost and you get an "E" instead. Without dot- and dash-memory the keyer works in plain iambic mode, with dot-memory squeeze a "K" and release both levers only during the second dash. If you get the "K" the keyer works in iambic type A (Curtis-keyer) mode, if you get a "C" instead it works in iambic type B (Accu-keyer) mode.

variants and phonies

As described before, the short squeeze release time window of iambic type B is most problematic for the dot-memory and makes it almost impossible to squeeze the character "A" at high speed. The creators of the *CMOS Super Keyer*, which appeared in 1981 [8] and became very popular, were well aware of that fact and therefore they implemented this timing variant: *the dot- and dash-memories generally work according to type B, but during the first dot-length (first third) of a dash the dot-memory is disabled so that it cannot be set by a pressed dot-lever*. This gives the operator more time to release a squeeze than in true type B, but the the drawback is that the insertion of a dot by a short tap of the dot-lever may fail: key an "N" as described in the test before and you will often get a "T" instead, indicating a disabled dot-memory. From the firmware version 2.0 on, which appeared in 1991, the Super Keyer can be set to emulate other modes in addition to that default setting "V0" which is designated as Super Keyer (or Logikey) timing w/dot and dash memory. Unfortunately, in contrast to the Super Keyer with its unmistakable mode designations there are also quite a number of phonies, keyers which pretend to work in type A (Curtis-keyer) or type B (Accu-keyer) mode but which actually do not ...

One prominent example is the internal keyer of all *Elecraft* transceivers and what is called "mode A" and "mode B" are in fact both timing variants and mixtures of the true type A and type B modes. *In Elecraft's "mode B" the dot- and dash-memories generally work according to type B, but during the first dot-length (first third) of a dash the dot-memory works according to type A so that it is set only if the dot-lever changes its state from "unpressed" to "pressed"*. So this mode follows the Super Keyer timing scheme but with type A instead of disabled dot-memory during the first third of a dash, with the result that the insertion of a dot by a short tap of the dot-lever can not fail. *And in Elecraft's "mode A" the period during which the dot-memory works according to type A is simply extended from the first third to the whole length of a dash.* So this mode gives the operator even more time to release a squeeze than the Super Keyer timing. Most Elecraft operators are very pleased with one or the other of these two modes, which is not surprising since they are more tolerant than the true type B (Accu-keyer) mode which according to my survey the vast majority of these operators initially learned. However, both modes are problematic for all those who are used to true type A or basic iambic mode.

Another example is the *PicoKeyer* which according to its specs features dot/dash-memory. The manual for the Ultra PicoKeyer (firmware V2.1, 13 January 2016) states: *"Modes A & B are simply a matter of when the keyer checks for input from the paddles. In iambic mode A, the keyer only checks for paddle inputs after the end of each dot or dash. In iambic mode B, on the other hand, the keyer will check for paddle input during each dot or dash"*. Of course this explanation is wrong, because a correctly programmed iambic keyer with dot/dash-memory checks for paddle inputs during each dot or dash in type A as well as in type B mode. But in type A (Curtis-keyer) mode it checks for a *transient* from unpressed to pressed whereas in type B (Accu-keyer) it checks just for the *state* pressed. And even in basic iambic mode without dot/dash-memory it does not check after the end of each dot or dash, but after the end of each dot- or dash-*element* which contains the following space. But since the PicoKeyer indeed works according to its wrong explanation, when it is set to mode A it does not recognize paddle inputs during each dot or dash but only during the following space: key an "N" as described in the test before, if you release the dot-lever while the dash is heard the dot-memory does not work and you get a "T" instead. But if you release it a bit later during the space following the dash, the dot-memory works and you get the

"N". So the PicoKeyer in fact works neither as a true type A nor as a basic iambic keyer but instead it behaves like a strange hybrid of both. The dot/dash-memories do not work properly in Ultimatic mode either, only in mode B.

OZ mode

Last but not least a very unusual keying mode which has its origin in Denmark. It has been brought to my knowledge by Steen Wichmand, OZ8SW, and for lack of an established name I will call it "OZ mode". During the 1970s the Danish company "Dansk Radio Aktieselskab" manufactured its squeeze-keyer

type MSK5 based on a circuit with 10 transistors and 2 ICs. If the dot-lever is pressed to start a character, the keyer works in ultimatic mode but without dot/dash-memory: as long as it is held pressed, dots are produced. But when the dash-lever is also pressed it overrides the dot-lever and subsequent dashes are produced until it is released. However, if the dash-lever is pressed to start a character the keyer works in a different way utilizing its so-called "single dot memory and injection system": as long as it is held pressed, dashes are produced. But when the dot-lever is also pressed during a dash for the first time during the character, it inserts one single dot and then reverts to dashes even if the dot-lever is held pressed. Then, when the dash-lever is released while the dot-lever is held pressed, only dots are produced until a subsequent pressure of the dash-lever overrides the dot-lever and dashes are produced again.

All letters of the alphabet, except for the "X", and all digits can be generated with just one or two strokes and the number of strokes necessary to key all letters of the alphabet and all digits is 64, the same as for ultimatic mode. But though the description how it works might sound rather complicated, in fact it combines the merits of the basic iambic and the ultimatic mode. An operator who is used to basic iambic mode should have no problem to operate in OZ mode if he only minds to release the dash-lever before the final dot of the "C" starts, while in ultimatic mode the letters "C" and "K" require a different keying technique which is difficult to manage at high speeds. And contrary to the iambic modes the OZ mode just like the ultimatic mode for many characters allows to hold a lever pressed with no need to let it go in time.

The MSK5 became very popular and widespread in Scandinavia among radio amateurs as well as professionals. It was also used at the coastal radio station "Blaavand Radio" with the callsign OXB, located on the west coast of Jutland and noted for its characteristic low-pitched modulated CW (mode A2) signal. The following picture shows radio telegraphy operator Connie Nielsen at her OXB operating position in 1980. Note the MSK5 keyer on the table together with a straight key ...

electronic keying modes - a survey

In 2015 I asked an international group of dedicated and proficient amateur radio telegraphy operators for their preferred electronic keying mode and received 68 answers, here is the result of my survey:

```
25 = 36.8% iambic type B (Accu-keyer)
21 = 30.9% single-lever
14 = 20.6% iambic type A (Curtis-keyer)
 7 = 10.3% basic iambic
 1 =  1.5% ultimatic
```

Interestingly most operators seem to prefer iambic type B (Accu-keyer) despite its problematic dot/dash-memory logic. And it turned out that the preference

of most of them is indeed not the result of an experimenting process with a final decision for the personally most suitable and effective mode, but it is simply the mode they initially learned and never changed since then. Once a certain keying mode is ingrained, it is really hard to change ...

Keyrama

The suffix "-rama" stems from the Ancient Greek word "ορα$\mu\alpha$" which means "wide view". In order to complement this explanation with a useful practical device, I developed the unique PIC-based multi-mode Morse code keyer *"Keyrama"* [9] which enables you to get a wide view of the different keying modes, to compare their proper logic and accurate timing with the logic and timing of other keyers, to follow the visualized action of dot/dash-memory and to find out to which extent your personal keying technique really makes use of it.

Because I could not find one single document with a proper in-depth explanation of the different twin-lever keying modes, I wrote this article to fill the gap. You don't have to know how the internal logic of your electronic keyer works to use it, but getting a handle on this subject can deepen your understanding for the process by which it generates those dits and dahs. And maybe it can excite your curiosity to try another keying mode.

references

1. Jack W. Herbstreit, W4JNX: "Automatic Spacing of Letters and Words for the Electronic Key", QST, April 1951, p. 46

2. John Kaye, W6SRY: "The 'Ultimatic' - The Key with a Memory", QST, February 1953, p. 11

3. John Kaye, W6SRY: "The All-Electronic 'Ultimatic' Keyer", QST, April 1955, p. 11

4. Alvin F. Kanada, K0MHU: "The 'Ultimatic' - Transistorized", QST, September 1960, p. 27

5. Harry Gensler Jr., K8OCO: "The 'Iambimatic' Concept", QST, January 1967, p. 18

6. Dave Muir, W2VYO: "The Penultimate Electronic Key", QST, March 1962, p. 15

7. James M. Garrett: "The WB4VVF Accu-Keyer", QST, August 1973, p. 19

8. Jeffrey D. Russell, KC0Q, and Conway A. Southard, N0II: "The CMOS Super Keyer", QST, October 1981, p. 11

9. http://cq-cq.eu/DJ5IL_rt008e.pdf (English)
   http://cq-cq.eu/DJ5IL_rt008d.pdf (German)

DJ5IL_rt007.pdf
Original version: 3.10.2016
Revisions: 7.10.2016, 20.10.2016, 7.6.2017