Erklärungen zur NEMA

Explanation for NEMA

Martin Käslin

November 22, 2017

Contents

Figure 1: That's what a NEMA looks like.

1 Historical

NEMA is an acronym for NEw MAchine.
NEMA is a Swiss equivalent to the well known German Enigma.
It is used to encrypt and decrypt texts.
It weights 10.7 kg and has a dimension of 332 x 384 x 148 mm.
There were different versions of NEMA:

- The school-machines have been in use from about 1947 at different troops, e.g. at aviator troops (until ca. 1950) and at the emergency radio of the radio police (ca. 1976).
- The combat-machines became sealed and locked away.
- The NEMA was also used at the embassy radio (1947-1976).

In 1992, NEMA became declassified and from 1994 sold by the Swiss Army.

In the manual NEMA was also called „T-D“ for „Tasten-Drücker“ which means literally „button-presser“.

2 Known Rotors

We differ:

- stepping rotor (SR, German: Fortschaltwalze - FW) and
- contact rotor (CR, German: Kontaktwalze KW).

The stepping rotors are primarily responsible for the rotors' rotation. Different SR are distinct by a number.
Contact rotors conduct electric impulses from the entry point to an exit point on the other side - representing a shift or a substitution of a letter. Different CR are distinct by a character.

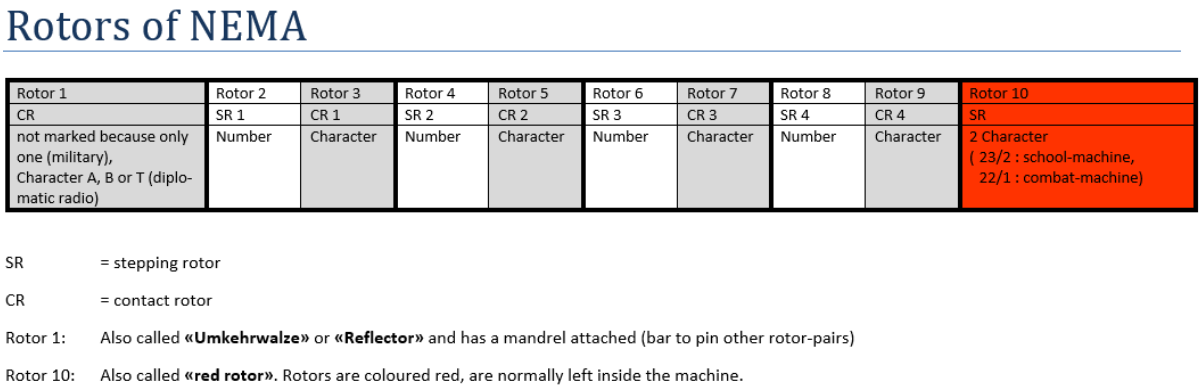

Figure 2: The ten rotors of NEMA

The right end is a SR, composed of two rings. Both rings are red coloured for optical marking. This SR is marked with a double number (e.g. 2/23). This red rotor is – watching the NEMA in operational mode – the one on the very right. (See Figure 3.)
There follow four pairs composed of each one stepping rotor (SR) and one contact rotor (CR). The ending (left side) forms the reflector, which is likewise a CR. On this reflector is fixed a bar to stick together the other rotors.

Military school-machine (220 ex):

- UKW^1, A, B, C, D, 16, 19, 20, 21 (2/23)

Military combat-machine (320 ex):

- UKW^1, A, B, C, D, E, F, 12, 13, 14, 15, 17, 18 (1/22)

Embassy-machine (100 ex, with 3 rotor sets):

- A, C, D, E, F, H, 27, 28, 29, 30, 31, 32
- B, J, K, L, N, O, 38, 39, 41, 42, 43, 44
- T, U, V, W, X, Y, Z, 25, 26, 33, 35, 37
- UKW^1 A, UKW^1 B, UKW^1 C

[1] UKW = Umkehrwalze [ger.] = reflector [eng.]

There is an other rotor G, where I don't know at which set of the embassy radio it belongs.
I don't know if the reflectors belong to a specific set of rotors or if they are to be combined freely.
(The embassy rotor's wiring is unknown to me, so that they can't be implemented in this program.)

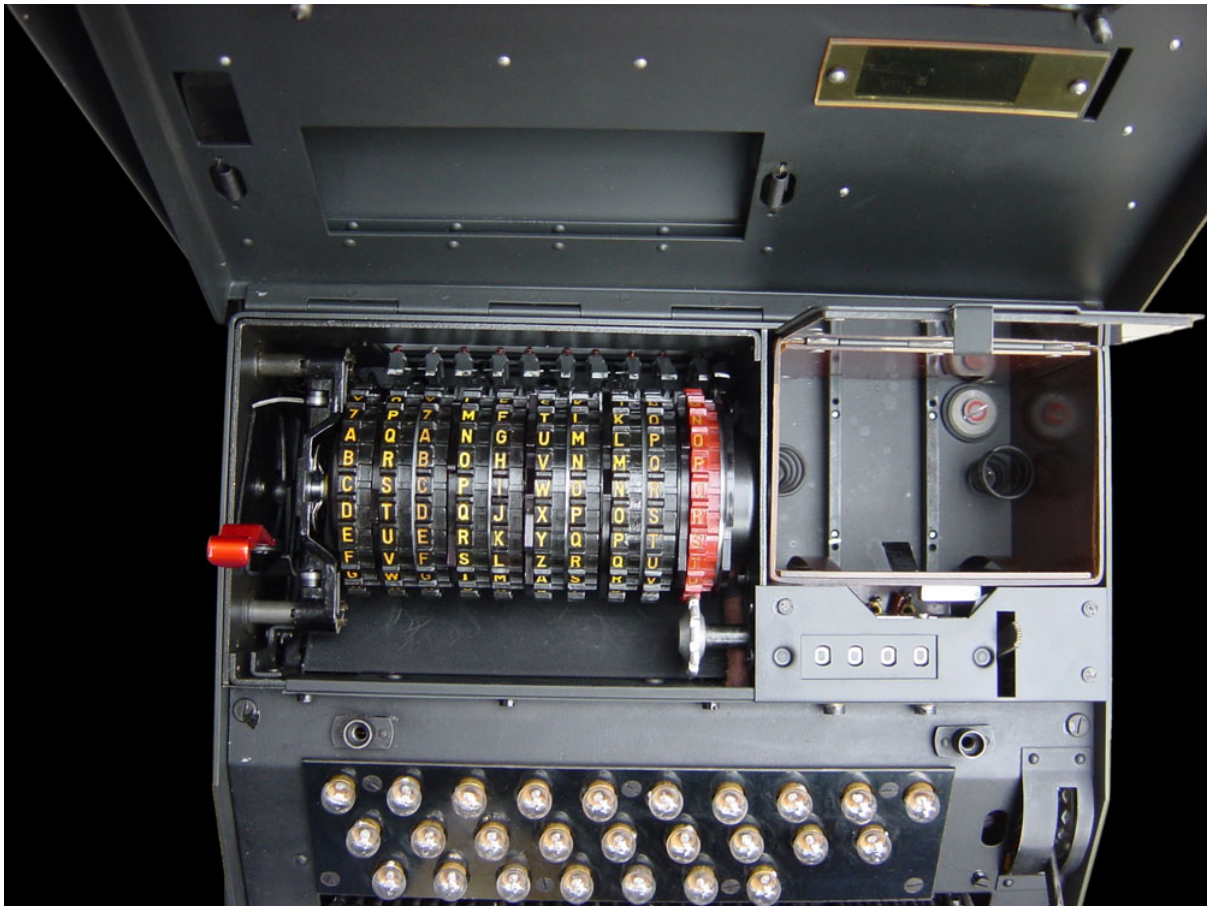

Figure 3: The original uploader was Matt Crypto at English Wikipedia – Transferred from en.wikipedia to Commons. CC BY-SA 3.0, Link

3 Usage

3.1 Digits with NEMA

Normally, numbers are written out with characters, thus 2 as „TWO“ or 22 as „ZWEIZWEI“ or 11 as „ELEVEN“. Additionally they define a virtual numeral layout. It is in the top row of the keyboard from left to right (Q=1, W=2, E=3, R=4, . . . , O=9, P=0). See also Figure 1.
Switch between normal character layout and virtual number layout with YX. Y to start with numeral representations, X to switch back. Like YQWX for 12. The characters Y and X are added to the plain text and cyphered. An error in the numeral layout is hard to detect and could have severe consequences caused by an simple typo or by a transmission error.
Y and X are both rare letters in German. But they could appear without switching to the virtual layout. So dealing with virtual numeral layout is quite tricky.

3.2 Theory: Crypt „daily key method“

1. The inner key was stuck together according to the key order / daily key. The validity period should be about 100 messages (could be equal for several messages).
2. The outer key was turned into position, so that it is equals to the daily key. One outer key per message - you never should use the same outer key twice.
3. The coder thinks of 10 random characters (military), respectively 5 random characters which get doubled (embassy radio). Theses 10 characters are the message's beginning. (e.g. EXTERSNHIK at military or EXTEREXTER at embassy radio)
4. These 10 characters (e.g. EXTERSNHIK) are cyphered with the daily key and you get the message key. (e.g. KKVTOBUQLE)
5. The outer key is reset to the new message key. (New outer key is now KKVTOBUQLE.)
6. Now, the message is typed character by character. After each character shines the cyphered character. These are attached to the text from item 3.
7. The ten character from item 3 are additionally attached to the message's end to detect transmission errors.

3.3 Theory: Decrypt „daily key method“

1. The coder sets up inner and outer key according to the key order / daily key.
2. The first 10 characters are typed (e.g. EXTERSNHIK) and the lighted up characters give you the message key. Set the new inner key to the message key (e.g. KKVTOBUQLE).
3. You can now decrypt the message from the 11th character. The last 10 characters are to check the correctness of the message key (e.g. KKVTOBUQLE).

3.4 Military practice: Crypt „session key method“

1. The inner key was stuck together according to the key order / daily key. The validity period should be about 100 messages (could be equal for several messages).
2. The outer key was turned into position, so that it is equals to the daily key. One outer key per message - you never should use the same outer key twice.
3. Now, the message is typed character by character. After each character shines the cyphered character. They are the cyphered message.

3.5 Military practice: Decrypt „session key method“

1. The coder sets up inner and outer key according to the key order / daily key.
2. You can now decrypt the message character by character.

3.6 Conventions

1. Important punctuation marks are written out: LEFT PARENTHESIS, COMMA, PARAGRAPH, QUESTION MARK ...

2. Whenever possible numbers are written out instead of switching to numeral layout. Errors in numeral layout are hard to detect (see: 3.1 Digits with NEMA) and asking back is time consuming.

3. The cyphered text is split into groups of five characters because these are internationally conventional for radio operators. A group of five characters is called word, 10 words give a line. (In the program I only group in words because of variable window size.)

4. If a plain text doesn't end in a group of 5 characters, there will be added nonsense-characters at the end to match group size (embassy radio). The real end is signed with „STOP“ (also used as synonym for „full stop“).
   Military allowed an incomplete group at the end.

3.7 Embassy radio: various security enhancements

The embassy radio know further security enhancements as character tripling, blending, splitting and codewords for set phrases. These were applied before cyphering to the plain text so that you get a new plain text which will be cyphered.

3.7.1 Character tripling

This makes it harder to assume a plain-text-to-secret-text correlation (masking). A plain text should have about 2 to 3 character tripling per line (10 words).
So becomes: MISTERPRESIDENT → MISSSTERPRRRESIDENNNT.

3.7.2 Blending

Also used to mask, because the secret text becomes longer. Blender are words – normally in a foreign language and of a specific, predefined group like animals, trees, car labels and so on. Blender can be everywhere; at the beginning of a word, in a word (MISTER → MISAUDIRR), at the end or between words.

3.7.3 Splitting

The text is split into any sized paragraphs. Every paragraph starts with a numeral word of a different language. The paragraphs with their numeral word at the beginning are shuffled.
Underline only for simpler recognition in this ultra short example (normally several words):

1. CHANCELLOR

2. UNCHANDEUXCELTROILOR

3. DEUXCELTROILORUNCHAN

3.7.4 Codewords for set phrases

The typical style (header, footer, greetings, ...) of a telegram is destroyed while adding a three digit number code at predefined place.

3.8 Important rules

An identical telegram is not allowed to be cyphered with two different key. Should the same message go to different receiver, the message is to be changed! Best practice is by splitting, character tripling at different characters or with (other) blenders.

4 Known wiring

Pieces of code that might be useful for reproduction (manually or for own programming)

- *bool* is a logical value with 0=false and 1=true.
- *char* is a single character.
- *int* is an integer literal.
- characters can also represented a number. Where $A = 0$, $B = 1$, $C = 2$, $D = 3$, ... so the number is the offset to character A.

The tables are written in the following schema: On the first line is the name of the table, followed by the arguments in squared brackets. The return type is given after the right arrow and its interpretation in parenthesis.
On the second line is the return value, its name followed by the dimension (number of entries) per argument in squared brackets. After the equals sign follow the entries in curly braces.
Everything after „//“ is a comment and doesn't belong to the effective entry.

4.1 Cams to revolve rotors

Gives the cams' position on the stepping rotor which cause a movement of the rotors.

Cams[„stepping rotor“][„offset to A“] ⇒ bool

```
bool Cams[24][26] = {
    {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},          // rotor 0   (unused)
    {0,1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,1,1},          // rotor 1
    {0,1,0,1,1,0,0,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0},          // rotor 2
    {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},          // rotor 3   (unused)
    {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},          // rotor 4   (unused)
    {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},          // rotor 5   (unused)
    {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},          // rotor 6   (unused)
    {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},          // rotor 7   (unused)
    {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},          // rotor 8   (unused)
    {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},          // rotor 9   (unused)
    {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},          // rotor 10  (unused)
    {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},          // rotor 11  (unused)
    {0,1,1,1,1,1,1,1,1,1,1,0,0,0,1,1,1,1,0,1,1,1,1,1,1},            // rotor 12
    {1,1,0,1,1,1,1,0,0,1,1,0,1,1,0,1,1,1,0,1,1,1,1,1,1,0},          // rotor 13
    {0,0,1,0,1,1,1,1,0,1,1,1,1,1,1,1,1,0,1,0,0,1,0,1,0,1},          // rotor 14
    {1,0,0,1,1,0,1,0,0,0,0,0,1,0,1,1,1,1,1,1,0,1,0,1,1,1},          // rotor 15
    {1,1,1,1,1,1,0,1,1,1,1,1,1,1,1,0,1,1,1,1,1,1,1,1,1,0},          // rotor 16
    {0,1,0,0,0,0,0,1,1,1,1,0,0,0,0,0,1,0,1,0,1,1,0,1,1,0},          // rotor 17
    {1,1,1,1,1,1,1,1,1,1,1,1,1,0,1,0,1,1,1,1,1,1,1,0,1,1},          // rotor 18
    {1,1,1,0,1,1,1,1,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,0,1,1,1},        // rotor 19
    {1,1,1,1,1,1,0,1,1,1,0,1,0,1,0,1,0,1,0,1,1,0,1,1,1,1},          // rotor 20
    {1,0,1,1,1,0,1,1,1,1,0,1,1,1,1,0,1,1,1,0,1,0,0,1,0,0},          // rotor 21
    {1,1,0,0,1,0,1,1,0,0,1,0,1,1,0,1,1,1,1,1,0,0,1,1,1,0,0},        // rotor 22
    {1,0,1,1,1,1,1,1,1,1,1,1,0,1,1,1,1,1,1,1,1,1,1,1,1,1,0}         // rotor 23
};
```

4.2 Connection plate „in“

Which key is linked to which contact at the first rotor.

ConnectionIn[„offset to A“] ⇒ int (new „offset to A“)

```
int ConnectionIn[26] = {14,1,3,12,22,11,10,9,17,8,7,6,25,0,16,15,24,21,13,20,18,2,23,4,5,19};
```

4.3 Contact right wired to contact left „in“

Electric flow for characters from right to left through the rotor.

ContactIn[„contact rotor“][„offset to A“] ⇒ char

```
char ContactIn[6][26] = {
    {N,S,K,I,T,C,O,Y,M,V,W,A,U,J,D,R,L,Z,X,H,F,Q,E,G,P,B},    // rotor A
    {K,J,Y,N,T,M,E,H,L,O,Z,Q,B,W,P,S,X,I,R,F,A,G,U,D,V,C},    // rotor B
    {P,N,F,U,T,E,D,I,Z,Y,A,H,V,R,W,O,J,S,G,B,Q,M,K,C,X,L},    // rotor C
    {W,J,B,E,Y,F,U,C,M,D,T,A,Z,K,X,P,I,Q,H,S,V,L,G,O,N,R},    // rotor D
    {H,R,Q,T,Y,V,X,M,N,A,C,F,U,J,E,S,W,L,Z,I,G,D,P,O,K,B},    // rotor E
    {Z,V,G,E,Q,M,U,T,W,L,N,S,H,P,O,A,F,Y,I,X,K,B,D,R,J,C}     // rotor F
};
```

4.4 Reflector

The reflector (ger. Umkehrwalze) is also a contact rotor and shifts therefore the character. The return value is the offset to the input character.

Reflect[„offset to A“] ⇒ int

```
int Reflect[26] = {10,21,13,10,2,2,24,24,13,10,16,14,8,16,10,13,7,1,25,16,18,13,5,19,16,12};
```

4.5 Contact left wired to contact right „out“

Electric flow for characters from left to right through the rotor.

ContactOut[„contact rotor“][„offset to A“] ⇒ char

```
char ContactOut[6][26] = {
    {L,Z,F,O,W,U,X,T,D,N,C,Q,I,A,G,Y,V,P,B,E,M,J,K,S,H,R},    // rotor A
    {U,M,Z,X,G,T,V,H,R,B,A,I,F,D,J,O,L,S,P,E,W,Y,N,Q,C,K},    // rotor B
    {K,T,X,G,F,C,S,L,H,Q,W,Z,V,B,P,A,U,N,R,E,D,M,O,Y,J,I},    // rotor C
    {L,C,H,J,D,F,W,S,Q,B,N,V,I,Y,X,P,R,Z,T,K,G,U,A,O,E,M},    // rotor D
    {J,Z,K,V,O,L,U,A,T,N,Y,R,H,I,X,W,C,B,P,D,M,F,Q,G,E,S},    // rotor E
    {P,V,Z,W,D,Q,C,M,S,Y,U,J,F,K,O,N,E,X,L,H,G,B,I,T,R,A}     // rotor F
};
```

4.6 Connection plate „out“

Which contact at the red rotor is linked to which lamp/character on the output key.

ConnectionOut[„offset to A“] ⇒ char

```
char ConnectionOut[26] = {N,B,V,C,X,Y,L,K,J,H,G,F,D,S,A,P,O,I,U,Z,T,R,E,W,Q,M};
```

5 Example tasks

5.1 Example A

Combat machine, hint: session key method

inner key:	13 C 15 B 14 A 12 D
outer key:	DISTELFINK
ciphered text:	WTQLL GHPGP YLTXL GTMOA QHGHW USJBL QMABY KLZDK VONXJ AIXAI

5.2 Example B

School machine, hint: session key method

inner key:	19 B 16 C 20 A 21 D
outer key:	DISTELFINK
ciphered text:	YIJVE KKEGD WZBTE IQXOE PXCQT DYUJX IEMMC IEHLW HECVT

5.3 Example C

School machine, hint: daily key method

inner key:	19 B 16 C 20 A 21 D
outer key:	FELDWEIBEL
ciphered text:	EXTER EXTER LXHPN BHWLJ YFCDZ QESXP PSYNU JVOVX CLYZP LILAL YPIKN PRHPH ODAZC

5.4 Example D

Combat machine, hint: daily key method

inner key:	13 C 15 B 14 A 12 D
outer key:	FELDWEIBEL
ciphered text:	EXTER EXTER OABBX YBLZH RRCPX TGOCF PEMRI XCZQY GHRXW UHIYR JOOKJ YTUUI WUFDE

Solutions on the next page.

6 Solutions to example tasks

6.1 Solution to example A

The decrypted text says:
ICHGR ATULI EREST OPSIE HABEN DIENE MAVER STAND ENSTO PENDE
Or a bit more readable.
Ich gratuliere. Sie haben die NEMA verstanden. Ende

6.2 Solution to example B

The decrypted text says:
GRATU LATIO NKOMM ASIEH ABEND IENEM AVERS TANDE NSTOP Or a bit more readable.
Gratulation, Sie haben die NEMA vertanden.

6.3 Solution to example C

The decrypted text says:
BRRRA VOSIE BEHER RSCHE NDIEN EMAYQ PPXPR OZENT IGAUS RUFZE ICHEN
Or a bit more readable.
Bravo Sie beherrschen die NEMA 100-prozentig!
Hint: 100 is in numeral layout QPP. Therefore is YQPPX equals to 100.

6.4 Solution to example D

The decrypted text says:
BRAAA VOSIE BEHER RSCHE NDIEN EMAYQ PPXPR OZENT IGAUS RUFZE ICHEN
Or a bit more readable.
Bravo Sie beherrschen die NEMA 100-prozentig!
Hint: 100 is in numeral layout QPP. Therefore is YQPPX equals to 100.

References

[1] Schmid, Walter: *Die Chiffriermaschine Nema.* 3. überarbeitete und erweiterte Auflage, ca. März 2005

[2] Hirt, Armin: *Dorfleben*, in Ährenpost 35 (Jul./Aug. 2015), S. 2–3.
auch: *http://www.hombrechtikon.ch/documents/2015_07_08_Aehrenpost_def.pdf*

[3] Internet: *http://ilord.com/nema.html* besucht am: 1. Okt. 2017

[4] Internet: *http://ilord.com/nema-manual-scans.html* besucht am: 1. Okt. 2017

[5] Internet: *https://de.wikipedia.org/wiki/Nema_(Maschine)* besucht am: 4. Okt. 2017

[6] Internet: *https://de.wikipedia.org/wiki/Nema_(Maschine)* besucht am: 4. Okt. 2017

[7] Internet: *http://www.cryptocellar.org/pubs/NEMA.pdf* besucht am: 4. Okt. 2017

List of Figures