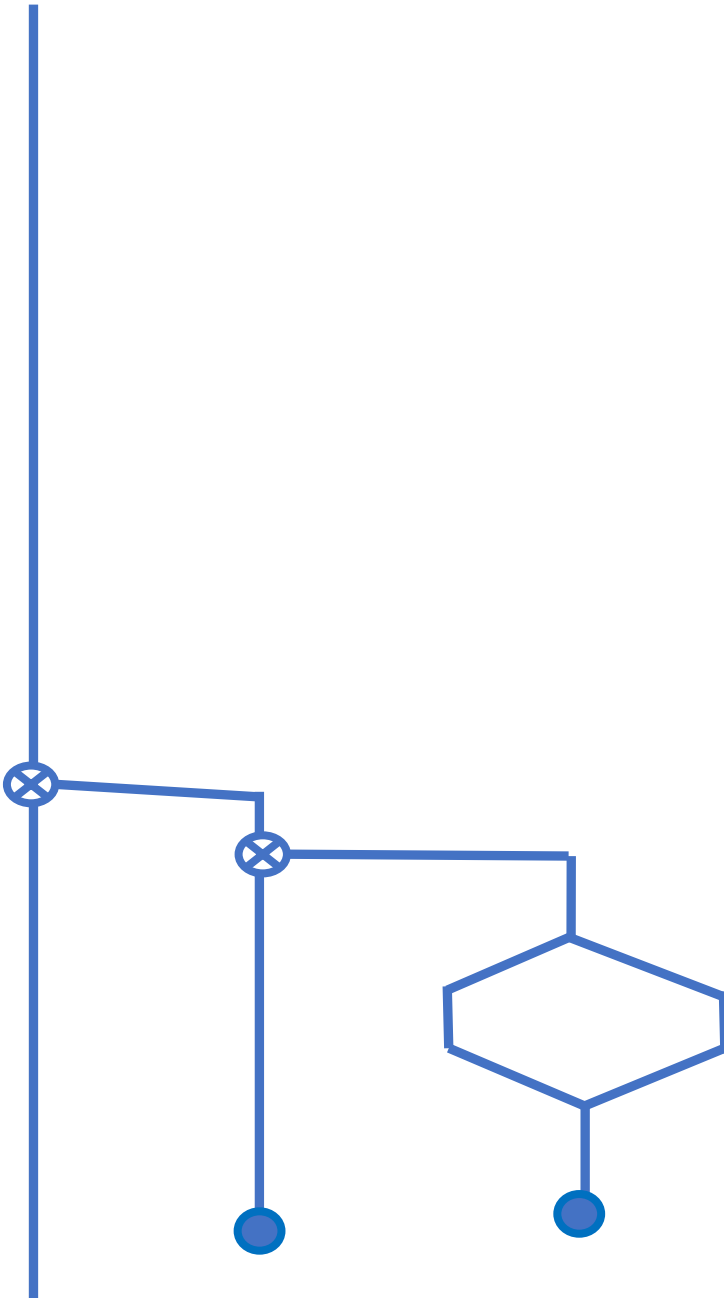


The Enigma of Forth

SVFIG

May 22, 2021

Bill Ragsdale



Today . . .

The Enigma cyphering machine was invented in 1920s with limited commercial acceptance.

In the 1930s it was adopted by the German government for military communications.

Through the joint effort of Poland and England many/most of German radio communications were decoded.

Original Enigma, 3 rotors



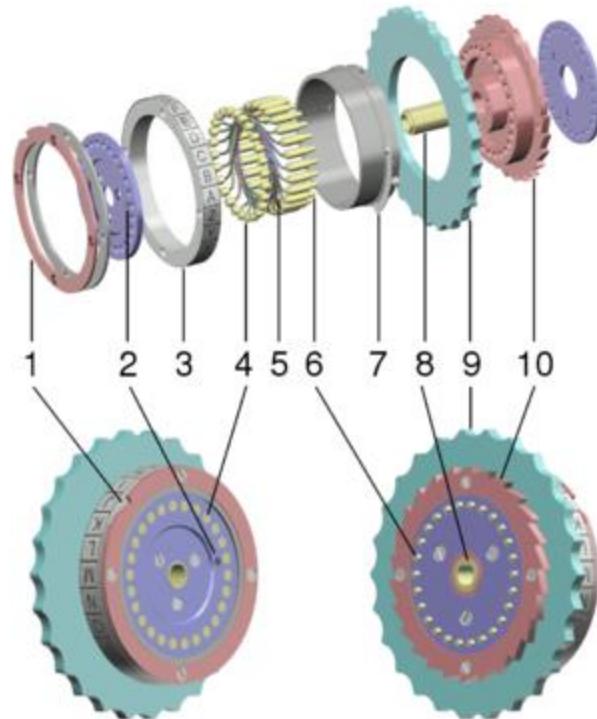


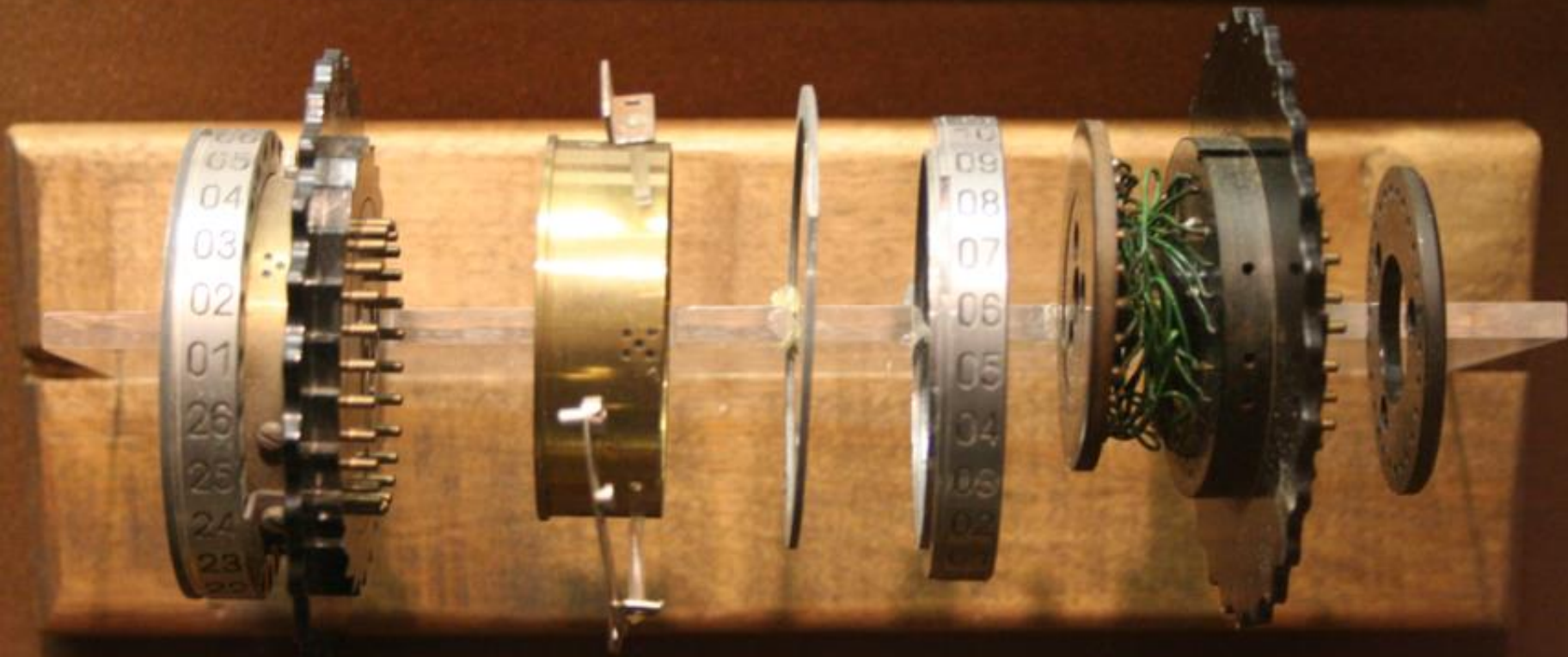
DOYLE

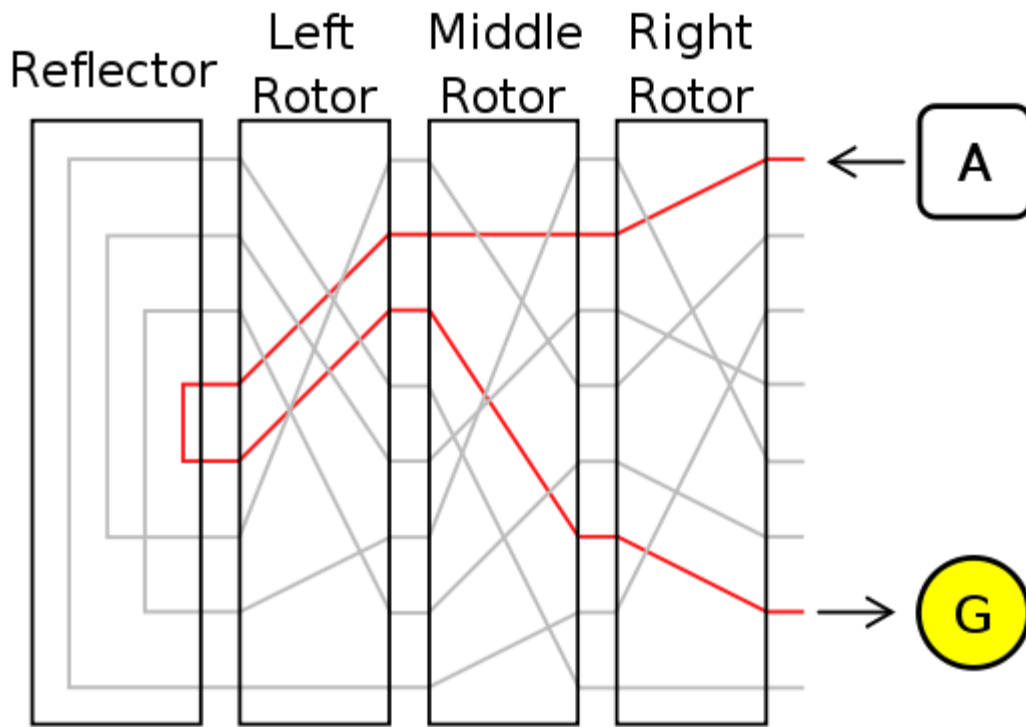
The Rotor Assembly



Rotor Exploded View







Letter A
Encrypts
To
Letter G

Manual Analysis, one rotor

m i s t e r
F V F G R E

OFF	1	1	1	1	1	-5	1	1	1	1	1	-5	1	1	1	1	1	1	1	1	1	1	-5	1	1	1	1	-5	1	-1
OFF	5	-1	-1	-1	-1	-1	5	-1	-1	-1	-1	-1	5	-1	-1	-1	-1	5	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	1	-1

IN	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

0	1	2	3	4	5	0	7	8	9	10	11	6	13	14	15	16	17	12	19	20	21	22	23	18	25	24
1	1	2	3	4	25	6	7	8	9	10	5	12	13	14	15	16	11	18	19	20	21	22	17	24	23	0
2	1	2	3	24	5	6	7	8	9	4	11	12	13	14	15	10	17	18	19	20	21	16	23	22	25	0
3	1	2	23	4	5	6	7	8	3	10	11	12	13	14	9	16	17	18	19	20	15	22	21	24	25	0
4	1	22	3	4	5	6	7	2	9	10	11	12	13	8	15	16	17	18	19	14	21	20	23	24	25	0
5	21	2	3	4	5	6	1	8	9	10	11	12	7	14	15	16	17	18	13	20	19	22	23	24	25	0
6	1	2	3	4	5	0	7	8	9	10	11	6	13	14	15	16	17	12	19	18	21	22	23	24	25	20

REF	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
	13	14	15	16	17	18	19	20	21	22	23	24	25	0	1	2	3	4	5	6	7	8	9	10	11	12

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	
0	5	0	1	2	3	4	11	6	7	8	9	10	17	12	13	14	15	16	23	18	19	20	21	22	25	24
1	25	0	1	2	3	10	5	6	7	8	9	16	11	12	13	14	15	22	17	18	19	20	21	24	23	4
2	25	0	1	2	9	4	5	6	7	8	15	10	11	12	13	14	21	16	17	18	19	20	23	22	3	24
3	25	0	1	8	3	4	5	6	7	14	9	10	11	12	13	20	15	16	17	18	19	22	21	2	23	24
4	25	0	7	2	3	4	5	6	13	8	9	10	11	12	19	14	15	16	17	18	21	20	1	22	23	24
5	25	6	1	2	3	4	5	12	7	8	9	10	11	18	13	14	15	16	17	20	19	0	21	22	23	24
6	5	0	1	2	3	4	11	6	7	8	9	10	17	12	13	14	15	16	19	18	25	20	21	22	23	24

OUT	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z

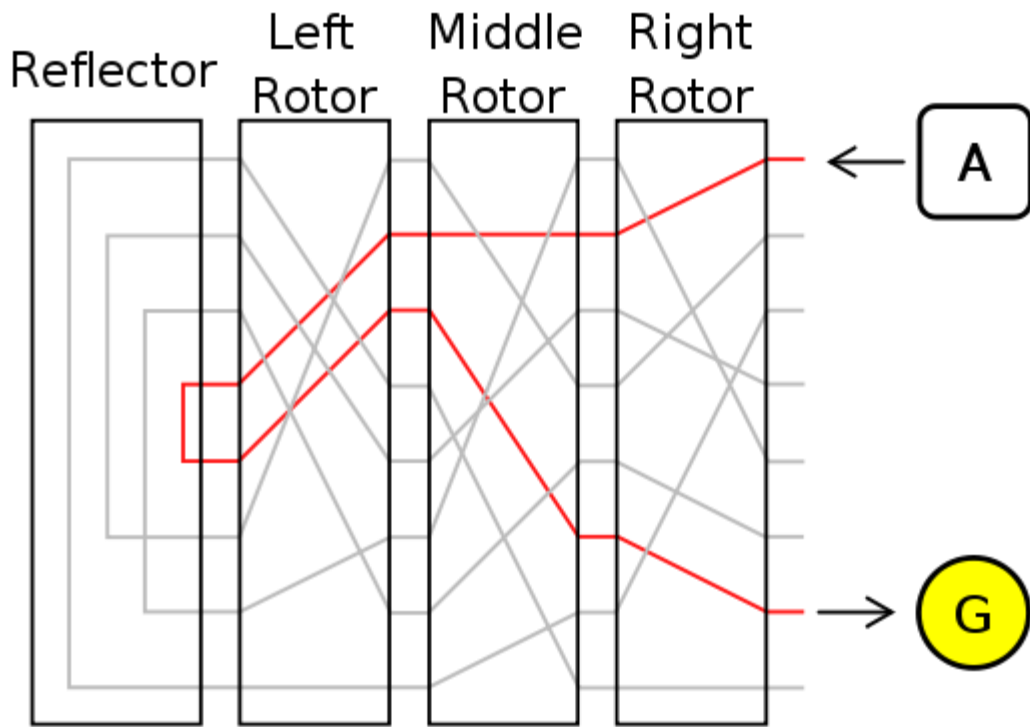
One Letter Through One Rotor

Output letter =

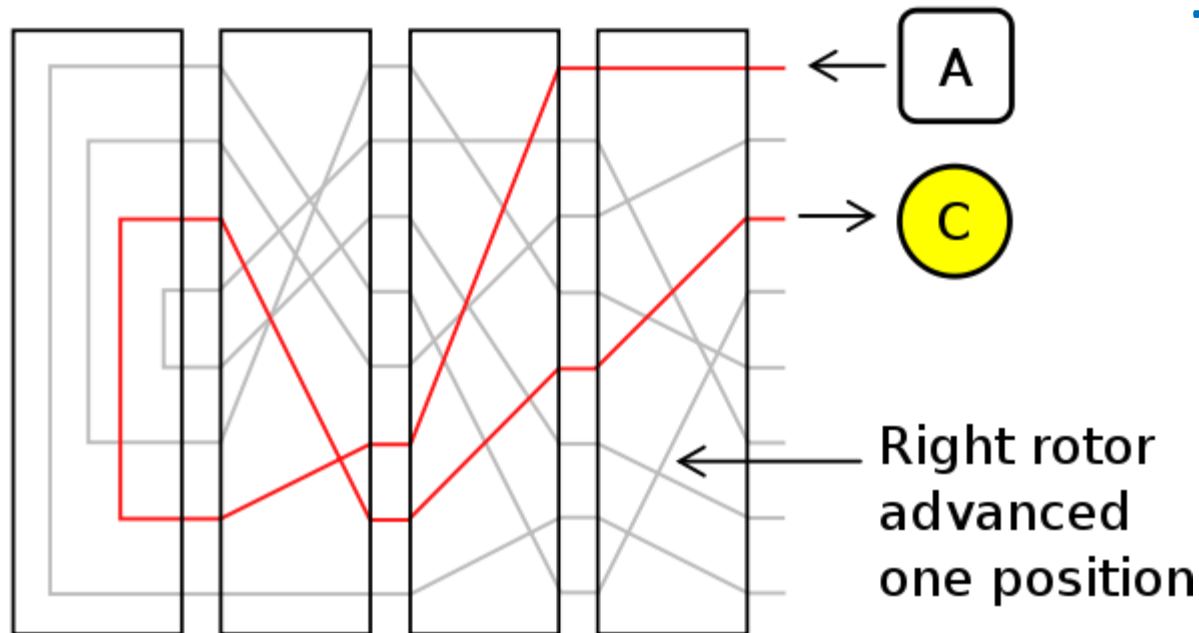
f(Rotor, Rotor Position, Input Letter, Transfer(+/-))

: Through-A-Rotor

```
RotorForward SlotIII      RotorPosition SlotIII
@ 2 pick + #letters mod ( position)
swap @ + Sc@              ( transfer)
+ #letters mod ;          ( output)
```



After
Rotor
Advance
Letter A
Encrypts
To Letter C





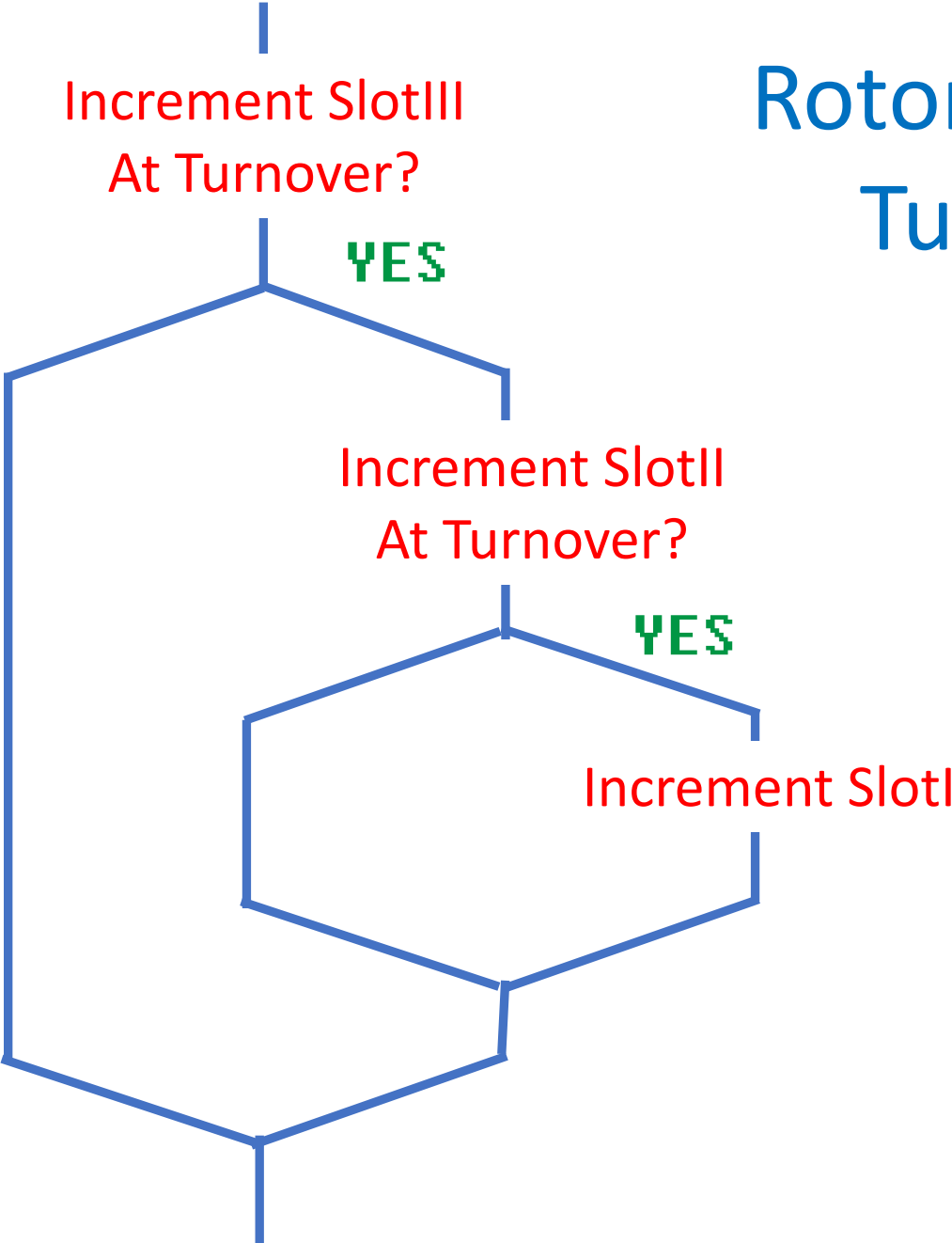
Turnover SlotIII to SlotII to SlotI

I	II	III
3	25	20
3	25	21
3	25	22
4	0	23
4	0	24
4	0	25
4	0	0
4	0	1
4	0	2



```
: EntryComplete \ Do turnover
RotorPosition SlotIII @ >step dup
RotorPosition SlotIII !
RotorTurnover SlotIII @ =
  if RotorPosition SlotII @ >step dup
    RotorPosition SlotII !
    RotorTurnover SlotII @ =
      if RotorPosition SlotI @ >step
        RotorPosition SlotI !
      then then ;
```

Rotor to Rotor Turnover



Rotor A with +/- Offsets

```
CREATE RotorA-Forward #letters allot
```

```
\  0  1  2  3  4  5  6  7  8  9 10 11 12 letter in  
   1  1  1  1  1 -5  1  1  1  1  1 -5  1  
  
\ 13 14 15 16 17 18 19 20 21 22 23 24 25 letter in  
   1  1  1  1 -5  1  1  1  1  1 -5  1 -1
```

```
CREATE RotorA-Reverse #letters allot
```

```
\  0  1  2  3  4  5  6  7  8  9 10 11 12 letter in  
   5 -1 -1 -1 -1 -1  5 -1 -1 -1 -1 -1  5  
  
\ 13 14 15 16 17 18 19 20 21 22 23 24 25 letter in  
  -1 -1 -1 -1 -1  5 -1 -1 -1 -1 -1  1 -1
```

The Reflector with +/- Offsets

```
CREATE Reflector #letters allot
```

```
\  0  1  2  3  4  5  6  7  8  9 10 11 12  
   13 13 13 13 13 13 13 13 13 13 13 13 13
```

```
\ 13 14 15 16 17 18 19 20 21 22 23 24 25  
   -13 -13 -13 -13 -13 -13 -13 -13 -13 -13 -13 -13 -13
```


A Slot Definer

```
: Define-Slot ( RotorxFwd, RotorxRev, Its_Position, Its_Turnover )  
CREATE 4 cells allot  
DOES> + ; \ Yield the field address within this array's data.
```

```
Define-Slot SlotI  
Define-Slot SlotII  
Define-Slot SlotIII  
Define-Slot ReflectorI  
Define-Slot Plug Board (omitted)  
Define-Slot Keyboard (omitted)
```

```
0 CONSTANT RotorForward  
1 CELLS CONSTANT RotorReverse  
2 CELLS CONSTANT RotorPosition  
3 CELLS CONSTANT RotorTurnover
```

Daily Setup By An Enigma Operator

Which rotor is in which slot.

Initial position of each rotor.

Turnover point of each rotor.

Plugboard settings. (omitted here)

Assign Rotors to Slots and Reset

The daily setup by the Enigma operator.

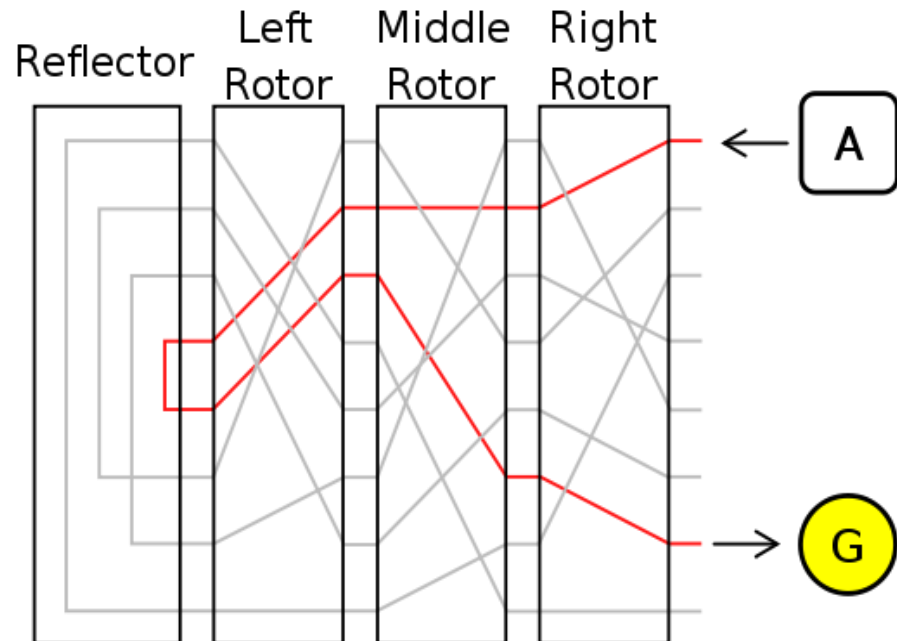
: Start

```
RotorA-Fwd  RotorForward  SlotI      †
RotorA-Rev  RotorReverse   SlotI      †
      0      RotorPosition SlotI      †
      0      RotorTurnover SlotI      †
RotorB-Fwd  RotorForward  SlotII     †
RotorB-Rev  RotorReverse   SlotII     †
      0      RotorPosition SlotII     †
      0      RotorTurnover SlotII     †
RotorC-Fwd  RotorForward  SlotIII    †
RotorC-Rev  RotorReverse   SlotIII    †
      0      RotorPosition SlotIII    †
      0      RotorTurnover SlotIII    †
Reflector   RotorForward  ReflectorI †
Reflector   RotorReverse   ReflectorI †
      0      RotorPosition ReflectorI †
      0      RotorTurnover ReflectorI † ;
```

Encrypting One Letter

: **A-letter** (letter_in --- letter_out)

```
RotorForward SlotIII   RotorPosition SlotIII   one-level
RotorForward SlotII    RotorPosition SlotII    one-level
RotorForward SlotI     RotorPosition SlotI     one-level
RotorForward ReflectorI RotorPosition ReflectorI one-level
RotorReverse SlotI     RotorPosition SlotI     one-level
RotorReverse SlotII    RotorPosition SlotII    one-level
RotorReverse SlotIII   RotorPosition SlotIII   one-level
EntryComplete          ;
```



Setup For A Test

start

```
5 RotorPosition SlotIII ? 12 RotorTurnover SlotIII ?  
5 RotorPosition SlotII ? 6 RotorTurnover SlotII ?  
20 RotorPosition SlotI ? 21 RotorTurnover SlotI ?  
Full-Alpha-Test
```

Encoding A 26 Letter Message “The Alphabet”

In	#	III	II	I	Re	I	II	III	OUT
A	0	21	22	23	10	9	8	7	H
B	1	2	3	24	11	10	9	8	I
C	2	3	4	5	18	23	22	21	U
D	3	24	25	0	13	12	11	10	K
E	4	5	6	7	20	19	20	19	T
F	5	6	1	2	15	14	13	12	M
G	6	1	2	3	16	15	14	13	N
H	7	8	9	10	23	2	1	0	A
I	8	9	10	5	18	17	16	15	P
J	9	4	5	6	19	18	19	18	S
K	10	9	10	5	18	17	16	15	P
L	11	12	13	14	1	0	5	4	E
M	12	13	14	15	2	1	0	25	Z
N	13	8	9	10	23	2	1	0	A
O	14	15	16	11	24	23	22	21	U
P	15	16	17	18	5	10	9	8	I
Q	16	11	6	7	20	19	18	17	R
R	17	18	19	20	7	6	11	10	K
S	18	19	18	19	6	5	4	3	D
T	19	14	15	16	3	4	3	2	C
U	20	21	22	17	4	3	2	1	A
V	21	22	23	24	11	16	15	14	O
W	22	17	12	13	0	25	24	23	X
X	23	22	23	24	11	16	15	14	O
Y	24	25	20	21	8	7	6	5	F
Z	25	0	1	2	15	14	13	12	M

IN = 15 = P

OUT = 8 = I

I	II	III	In	Out	Check
20	5	5	0	7	0
20	5	6	1	8	1
20	5	7	2	21	2
20	5	8	3	10	3
20	5	9	4	19	4
20	5	10	5	12	5
20	5	11	6	13	6
21	6	12	7	0	7
21	6	13	8	15	8
21	6	14	9	18	9
21	6	15	10	15	10
21	6	16	11	4	11
21	6	17	12	25	12
21	6	18	13	0	13
21	6	19	14	21	14
21	6	20	15	8	15
21	6	21	16	17	16
21	6	22	17	10	17
21	6	23	18	3	18
21	6	24	19	2	19
21	6	25	20	1	20
21	6	0	21	14	21
21	6	1	22	23	22
21	6	2	23	14	23
21	6	3	24	5	24
21	6	4	25	12	25 ok

In Action Summary

Encrypting a 26
letter message:

00 = A

to

25 = Z

Small Words

- `: ASCII>Integer` ASCII A - ;
- `: Integer>ASCII` ASCII A + ;
- `: SignExtend` \ extend 8 bits to 32
dup 128 and
if -256 (0xFFFFFFFF00) or then ;
- `: bounded` \ keep in 0..25 range
#letters mod ;

Forth Code

: encode

```
sample-out 200 erase start
sample-out sample-in count
0 do dup i + c@ A-letter
      2 pick i + 1+ c!
      i 1+ 2 pick c! loop 2drop ;
```

: decode

```
sample-check 200 erase start
sample-check sample-out count
0 do dup i + c@ A-letter
      2 pick i + 1+ c!
      i 1+ 2 pick c! loop 2drop ;
```

Encryption & Decryption

Plain Text

MISTER WATSON COME HERE I WANT TO SEE YOU

For Encryption, adding X word separators

MISTERXWATSONXCOMEXHEREXIXWANTXTOXSEEYOUYXXX

Encrypted

FXFGLKIJNGFZEIUBZLIURQLOXIUNAGCMBCFRRRCNBHICCC

As transmitted

FXFGL KIJNG FZEIU BZLIU RQLOX IUNAG CMBCF RRCNB HICCC

Decrypted

MISTERXWATSONXCOMEXHEREXIXWANTXTOXSEEYOUYXXX

Performance

FXFGL KIJNG FZEIU BZLIU RQLOX IUNAG CMBCF RRCNB HICCC

= 62 microseconds

MISTERXWATSONXCOMEXHEREXIXWANTXTOXSEEYOUYXXX

= 61 microseconds

Use In Attack

Time to decrypt all of the basic 3 rotor Enigma
1,054,450 combinations: 1 minute 34 seconds

Time to decrypt the 4 rotor Naval Enigma:
 $26^6 \times 336 = 1.037 \cdot 10^{11}$ combinations

= 1.47 million minutes = 2.80 years

With analysis, lots of messages, electromechanical aids and German operator errors (most) messages were read through the entire war. Ultimately, about 30 cypher systems were in use.

Comments

The original Bletchley Park attack used sliding rods with inscribed alphabets. It depended on technical discoveries and German operator errors.

The expansion of Enigma to four slots and eight rotors blocked decryption for seven months (April, 1942 to October, 1942).

Development of the electromechanical 'bombes' led by Alan Turing delivered decryption ranging from one hour to 60 hours.

Summary

Enigma existed in about 20 production variations from 1923 to 1942.

Enigma encryption was in use until about 1960.

Its decryption was still a British Top Secret until 1974.

Many software implementations exist for personal use of this technology.

Recovered February, 2021



References

- https://github.com/BillRagsdale/Forth_Projects
- https://en.wikipedia.org/wiki/Enigma_machine