

## Introduction

The Enigma Machine is an Electrical-Mechanical device used to Encipher/Decipher Messages. It consists of a keyboard, Plug-Board and three or four Rotors one or more of which move on a keystroke and a set of lamps labelled A to Z.

The Plugboard allows rewiring of the connection between keyboard and first Rotor. The connection then feeds through the cross wiring of Three or Four Rotors to the Reflector, which then links back through the Rotors via a different route to the Plugboard and to light one of the A...Z lamps. Note that depressing a key first steps the right Rotor and any subsequent engagements of other rotors upon completing each cycle of steps.

The Rotors are selected from a group of Five and inserted to any arrangement. Each rotates through the letters A...Z or as number 01 to 26. The Rotors fixed Ring on the right side has 26 contacts which are scramble wired to spring loaded contacts on the left. The second Ring can change the Alphabet letters with its notch to align with other positions (letters) of the first Rings cross wired outputs. This represents a substitution encryption and can be different for each rotor set. The combination of several rotors, in ever-changing positions, is what extends the Enigma Machines combinations.

## Code Breaking

The breaking of the Enigma Machine Code by members of Bletchley Park British WWII secret code breaking centre was one of its greatest achievements. The electrotechnical nature of the Enigma Machine Code generation was cracked using a form of reverse engineering. These invented and built decipher machines used a series of revolving drums that would click around seeking out code combinations that matched. Later development went on to create a programable code braking machine the Colossus, the forerunner to the modern computer.

It is a sobering thought as to where we might be today without our modern computers. Perhaps we owe a greater depth of gratitude than we realise to those who invented, built and worked on these codebreaking machines.



## QBITS EnigmaSE

The Display shows all the components together and their settings and workings plus a space for a typed in Message and an area for the Cipher CODE output.

Top left the Title and below a Menu that reflects the actions to take place. A Plug-board to set A...Z crossovers. The Rotor Setup to Select Rotors, Ring Setting and Rotor Start positions. Clear Settings to reset the Plugboard, Rotors or Both. File management actions for Load & Save Configurations. Encipher/Decipher of Messages. Reset Rotors, to set Rotors back to Initial Setup for a new Message Encipher or Decipher a Coding.

### QBITS EnigmaSE - MainMenu

Menu items are shown available if coloured Yellow (White) or unavailable in Red or change to Green as a Setup is completed - the Plug-board and Rotors.

- (1) Plug-Board Setup - Yellow or Green [Locked/Set]
- (2) Rotor Setup - Yellow or Green [Locked/Set]
- (3) Clear Settings - Yellow Available
- (4) Load Configuration - Yellow Enter Filename
- (5) Save Configuration - Red/Yellow Enter Filename
- (6) Encipher/Decipher - Red/Yellow Enter Message
- (7) Rotor Reset - Red /Green [unlocked]
- (8) Quit
- ( ? ) Select Action – Displays number of Menu Item.



### QBITS EnigmaSE – (1) Plug-Board Setup

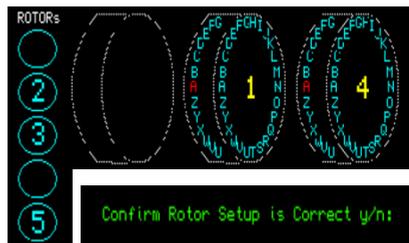
This requires 10 Plug-Board settings, select two unused [different] Letters for each. The Letters change colour to Red with a line draw between upper and lower row now set with crossover Letter. The visa versa crossover Letter position is also set. The 10 Plug ups are therefore shown as twenty entries on the Plugboard. Upon completion confirm y/n: Prompt is displayed.



### QBITS EnigmaSE – (2) Rotor Setup

The Rotors available are displayed on the left as Cyan/Green Circles labelled (1 to 5)

Select three Rotors in order, starting with the **Right** then **Middle** then **Left**. As a Rotor is selected the alphabet letters are displayed starting with 'A' centre left.



Each Rotors Input to Output is uniquely cross wired. Select from those shown available. The Rotor and its positioning give the Enigma Machine its many different combinations. Once the three Rotors are installed a y/n: Prompt allows a change if not correct.

### QBITS EnigmaSE – (2) Ring Settings

Choose a Letter when prompted for **Right** and **Middle Rotors**, these are identified as **RED** Letters shown on the Right and Middle Rings. A Correct y/n Prompt then allows you to complete or make a change if necessary.

### QBITS EnigmaSE – (2) Start Position

Again, choose a Letter when prompted for the **Right Middle Left Rotors**. The Rotor circles around until Letter shown left of centre matches the chosen Letter. On completion a y/n: Prompt then allows a change if not correct.



### QBITS EnigmaSE – (3) Clear Settings

This resets (1)**Plug-Board** and (2)**Rotor** Setups so they can be changed. (5)(6)(7) are disabled [Red]. If only Rotor Setup is actioned and Plug-Board is still set both (1)&(2) are set back to [Green] and (5)(6)(7) are reactivated [Yellow].

### QBITS EnigmaSE – (4) Load (5) Save

For Load and Save Configuration Enter Filename ie. 'win1\_EnigmaConfig01'. This is entered via INPUT#0 directly into the QL Interpreter. The results can be precarious. The use of **WHEN ERROR** and **IF** statements cover some of the unwanted outcomes, but not all. If unavailable to Load, '**File NOT found**' should be displayed. **Save** to an unavailable Device or Directory, should return '**DEVICE ERROR**'. If the File exists the Interpreter returns an 'OK to overwrite... Y or N'. Note: If unconnected devices have been redirected to active Directories, they might end up with unwanted files.

### QBITS EnigmaSE – (6) Encipher/Decipher

Once **Plug-Board Setup** and **Rotor Setup** are complete [shown as Green] (6) becomes available and changes to [Yellow]. A typed in Message will turn the Rotors and light the Cipher Lamps with the Cipher CODE printed to screen in five letter groups. The limit per individual Message is set at 75 Letters. For longer dialogs use multiple Messages, but the Plug-Board and Rotor settings must be Setup and correspond to the typed changes for the Rotors to correctly Decipher each Message.

[Esc] returns to the main menu, Select (6) again to continue Message. [#] Sends the CODE output as Morse dots and dashes, clears the CODE and Message displays, then resets the Rotors before returning to Main Menu.

### QBITS EnigmaSE – (7) Reset Rotors

This clears any CODE and Message displayed then resets the Rotors back to last Setup. Type the Message output CODE and the Decipher should display the original Letters.

### QBITS EnigmaSE – (9) DEMO

Option (9) the Special Edition is an accolade to the **Bletchley Park** codebreakers. **DEMO** displays the setting up of the **Plug-Board** and **Rotors**, a **Message** is then typed Letter by Letter to screen, the **Rotors** turn and **Cipher Lamp** glows. The **CODE** output is printed to screen in five letter groups and each Letter Sent has its **Morse Code** dots & dashes displayed with accompanying 'di' & 'dah' sounds.

## QL Enigma Program 2007

Here I'd like to thank Ian Pine for his SuperBASIC Enigma program and his informative accompanying article. However, as he stated, 'My knowledge of the machine is limited to the brief description contained in a booklet I bought when I visited Bletchley Park - where the main wartime efforts to break the cipher were made - the components described have been implemented, though some of the internal 'wiring' has been subject to 'educated' guesswork.

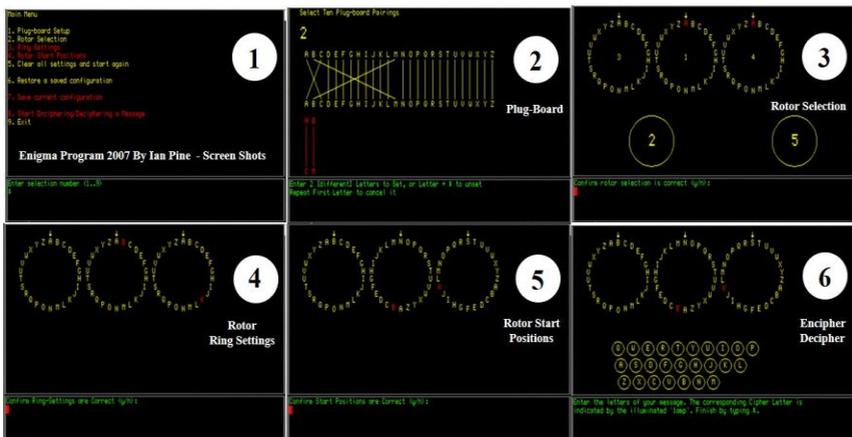
The two most fundamental features of the original system have been retained:

- (a) it is a 'self-inverting' cipher ie. the original plain-text can be recovered simply by typing the cipher-text back into the machine set to the same initial state;
- (b) the plain-text letter and the cipher-letter can never be the same.

The main components of the machine are'

- (1) a keyboard comprising the 26 alphabetic characters
- (2) a Plug-board
- (3) a Set of three Rotors
- (4) a 'Reflector' and
- (5) a Set of 26 lamps in the same layout as the keyboard

## QL Enigma 2007 Screen Shots



## QBITS QL Enigma Review

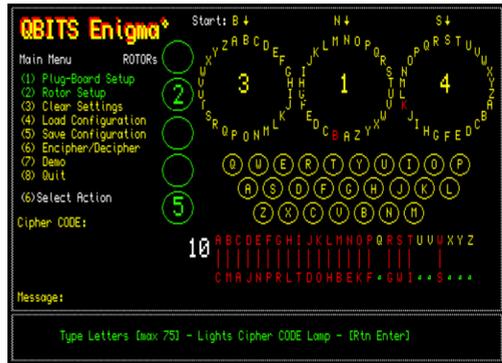
Downloading a pdf copy of Ian Pine's QL Today article, I finally managed after several attempts, to convert the lines of Program code it into a plain text format that the QL Interpreter would accept as a SuperBASIC Program. However, a number of lines with 'MISTAKE' needed to be corrected. The problem was the Optical Character Recognition (OCR) interpretation and also myself in scanning the articles print. The Font used in the scanned article had lower case 'l', 'I' and '1' numeral one, all looking very similar.

After some hours spent, and in some cases just by trial and error, the corrections were finally made and I got to see the program run. It did what Ian Pine had set out to do, but I was somewhat underwhelmed with the display arrangements for a 2007 written Program.

## QBITS Enigma Makeover

The combining of screens began by repositioning and resizing the Rotor displays and Plug-Board with Lamps between.

That left space to fit a Title and Menu with a place for the Cipher CODE output and at the bottom of the screen to display a line of text for the Message.



## QBITS Enigma Plug-Board

The arrangement chosen displays an upper line with the Alphabet A..Z and below this a line of small circles. As a Plug-up pair is chosen i.e. A-C the **A** colour changes to **Red** with the lower circle over printed with the crossover Letter in this case a **C** and a line draw between upper and lower row. The corresponding pair **C-A** is changed likewise. Once 10 pairs have been chosen a y/n Prompt appears, accept or Plug-Board is cleared for a retry. Pressing [Esc] will abort and clear entries then Select (1) to Set a new arrangement.

## QBITS Enigma Rotors

For a combined screen, I felt the Setup display of the Rotor Ring and Start positions were not truly represented. This prompted some further investigation into their actions.

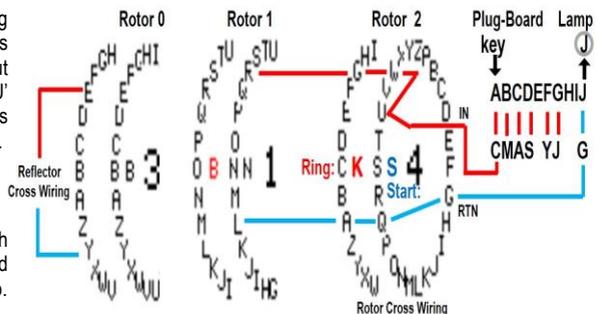
The Rotors are constructed as two Rings, The Rotors right-hand inputs represent the letters of the Alphabet A..Z or positions 0 to 25 and are cross wired to different outputs. These have spring contacts that connected to a second Ring. The positioning of the Second Rings contacts with respect to the first further redirects alphabetically adding to the cipher computation of a Rotors outputs. The procedure followed is for Rings to be Set on the Right and Middle Rotors.

All three Rotors can now be rotated to have a different Start position so as to add a further layer to the ciphering process. Last but not least is the Reflector which cross wires the output of the third/last Rotor to be fed back through all three Rotors and finally the cross wiring of the Plug-Board to light one of the cipher lamps.

**Note:** Diagram depicts cross wiring off Plug-Board and Rotors. 'C' is connected to Rotor 02 pin 02 [C] but Start position has moved this to 'U' on Rotor, which is internally cross wired to W then G via Ring settings.

Rotor 1 & 0 input <-> outputs are cross wired in a similar fashion.

The Reflector redirects back through Rotors via an alternative route and finally via Plug-Board to light a lamp.



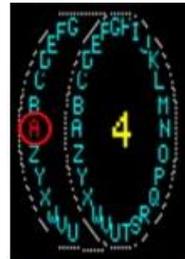
## QBITS EnigmaSE Rotor Display

Displaying a secondary Rotor Ring presented a challenge. Firstly, the Rotor screen space did not allow any expansion. Solving the problem, it seemed was to make the Rotors elliptical, keep the same height, but reduce their width. The first result was not aesthetically appealing. Overlapping the second Ring and keeping a readable aspect was the one finally taken. Adding an ellipse around the Rotor and an arc to the Ring lettering joined top and bottom by horizontal lines just enhanced the Rotors appearance.

```
1437 REMark *** Pre-calculate Coordinates For Rotor Alphabet ***
1438 FOR i=0 TO 25:cx(i)=-9*COS(PI*i/13):cy(i)=21*SIN(PI*i/13)
```

Character **cx**, **cy** values are calculated using COS and SIN with value 9 for width, 21 for height. These are used with the graphics Cursor positioning to Draw lettering of both Rotor and Ring. The **Rotor** displays all **26 Letters**, the **Ring** is reduced to those visible and achieved by the IF statement: IF i<7 OR i>19 AND i<26:

```
1137 DEFine PROCEDURE DrawRotor(n)
1138 LOCAL i:INK 6:BEEP 1000,100,20,8,3,0,12,9
1139 FOR i=0 TO 25
1140 INK 5:CURSOR 104+36*n+cx(i),85+cy(i),-3,-5:PRINT CHR$(65+(cp%(n)+i) MOD 26)
1141 CURSOR 95+36*n+cx(i),85+cy(i),-3,-5
1142 IF n>0 AND rs%(n)=(cp%(n)+i+rs%(n)) MOD 26:INK 2
1143 IF i<7 OR i>19 AND i<26:PRINT CHR$(65+(cp%(n)+i+rs%(n)) MOD 26)
1145 END FOR i
1146 END DEFine DrawRotor
```



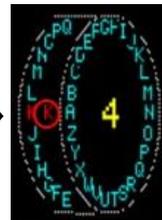
Rotor & Ring Aligned

## QBITS EnigmaSE Rotor Ring & Start

The number displayed in Yellow(White) at the centre of each Rotor is the Selected Rotor [ie. 1...5]. The current Letter position of the Rotor is at centre left.



The Ring Setting is shown in **RED**. The Rotor Start position Cyan (Green) is shown to the right of the current Rotor Letter position.



Ring Offset to Rotor

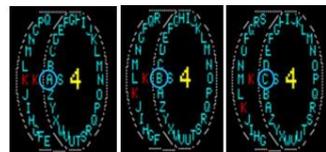


Ring & Start

## QBITS EnigmaSE Rotor Stepping

Each time a key is pressed the Rotor advances one step. On its 26 step it will advance the next Rotor one step and so on. By simply adding 1 to the current position  $cp\%(n)$  value and re-Drawing the Rotor the lettering is shown to advance. When Rotor Step  $rs\%(n)$  aligns with the penultimate Current step  $cp\%(n)-1$  the next step advances both Rotors.

```
1190 DEFine PROCEDURE AdvanceRotor(n)
1191 IF n<0 OR n>2:RETurn
1192 cp%(n)=(cp%(n)+1) MOD 26:DrawRotor n
1193 IF n>0:IF rs%(n)=(cp%(n)-1) MOD 26:AdvanceRotor n-1
1194 END DEFine AdvanceRotor
```



Stepping A

to B

through to C

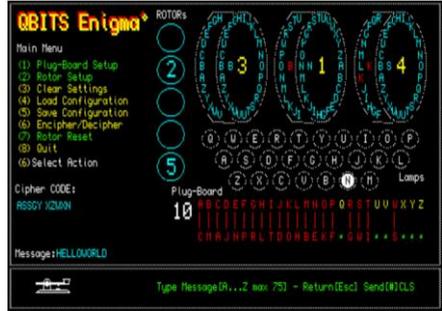
## QBITS EnigmaSE Encipher/Decipher

Once the Plug-Board and Rotors are set a Message can be typed in to create a coded output or Decrypt the Code of a Message already made. Using the Test Example for setup, type a simple message and write down the codes. Exit with [Esc] and use (7) Rotor Reset option from the Menu, then go back to (6) Encipher/Decipher and type in the codes. All being well the original Message should appear.

### Test Setup Example:

Plug board:	AC BM DJ EN FP GR HL IT K0 SW
Rotors:	3 (Left), 1 (Middle), 4 (Right)
Rings:	B K
Start:	B N S

The Message and CODE display is limited to 75 characters. The CODE is set out in five-character blocks in three rows. The Message is shown as a single row of characters no spaces that runs across the screen below the Plug-Board.



## QBITS EnigmaSE Message Coding

The Plug-Board is represented by  $pb\%(k)$  where  $k$  is 0 to 25, the output is held by 'O'. Taking **Rotors** positions from 2 through to 0 the variable 'O' is recalculated with the **Current Rotor**  $cp\%(i)$  and relative **Ring In**  $ri\%(i)$  positions. The output is changed by the cross wired **Reflector**  $O=rf\%(O)$  and then fed back through the **Rotors 0 to 2** with the use of **Ring Return**  $rr\%(i)$  and  $cp\%(i)$  **Current positions** to return an output  $pb\%(O)$ .

```

1214 DEFine FuNction ENcode(k)
1215 LOCAL i,O :O=pb%(k)
1216 FOR i=2 TO 0 STEP -1:O=(ri%(i,(cp%(i)+O) MOD 26)-cp%(i)) MOD 26
1217 O=rf%(O)
1218 FOR i=0 TO 2:O=(rr%(i,(cp%(i)+O) MOD 26)-cp%(i)) MOD 26
1219 RETURN pb%(O)
1220 END DEFine ENcode

```

**Note:** For the Enigma CODE to work the relative Rotor crossover wiring  $ro\%(k,n)$  has to be applied,  $n$  being the Rotor position 2 to 0 and  $k$  being the Rotor selected 1 of 5.

```

1131 DEFine PROCedure CopyRotor(n,k)
1132 LOCAL i
1133 FOR i=0 TO 25:ri%(n,i)=ro%(k,i):rr%(n,ro%(k,i))=i
1134 sp%(n)=0:cp%(n)=0:rs%(n)=0
1135 END DEFine CopyRotor

```

## QBITS EnigmaSE Reset

Menu item (7) Rotor Reset is only active if both Plug-Board and Rotors are setup. This is shown by Menu items (1),(2) and (7) tuning Green. (1) & (2) are then inactive until (3) Clear Settings is activated, which will then make (7) inactive [turn Red].

(7) Rotor Reset – turns the Rotors back to their Initial Start position for the beginning of a message and checks if Plug-Board setup has not changed i.e.  $pcb=10$ .

## QBITS EnigmaSE Code

1000 REMark **QBITS\_EnigmaSE\_bas** [QBITS Enigma SE 2023 review - QPC2]

1002 dev\$=win1\_':MODE 4:gx=0:gy=0 :REMark Screen Settings

1004 **WHEN ERROR:IF NOT ERR\_EX :eck=1:CONTINUE:END IF :END WHEN**

1006 REMark **Import QBITSconfig Settings - QPC2**

1007 OPEN \_IN#9,dev\$&'QBITSConfig':INPUT#9,gx\gy\dn\$:CLOSE#9

1010 REMark **QBITS REMake of QL Enigma Prog by Ian Pine 2007**

1011 REMark SE adds Message Code displays & Morse Code Playback

1012 REMark SE For DEMO Select (9)

1014 REMark **ARRAYS for Plug-Board & Rotor Configurations**

1015 DIM pb%(25),ro%(4,25),ri%(2,25),rr%(2,25),rf%(25),rs%(2),gr%(4),ts%(35)

1016 DIM sp%(2),cp%(2),cx(25),cy(25),kcx%(25),kcy%(25),kr\$(2,10),Morse%(26,4)

1018 **Init\_win:Enigma\_Menu**

1020 **DEFine PROCEDURE Enigma\_Menu**

1021 **REPEAT EnigmaMenu**

1022 CLS#0:INK 6:CURSOR 0,46 :IF pbdone=1:INK 4

1023 PRINT ' (1) Plug-Board Setup' :INK 6:IF rodone=1:INK 4

1024 PRINT ' (2) Rotor Setup' :INK 6

1025 PRINT ' (3) Clear Settings'

1026 PRINT ' (4) Load Configuration' :IF NOT(AIIDone):INK 2

1027 PRINT ' (5) Save Configuration': INK 6:IF NOT(AIIDone):INK 2

1028 PRINT ' (6) Encipher/Decipher' :INK 2:IF rodone:INK 4

1029 PRINT ' (7) Rotor Reset' :INK 6

1030 PRINT ' (8) Quit' :BLOCK 6,10,13,129,0

1031 r\$=INKEY\$(-1):r=CODE(r\$)-48:CURSOR 12,128:PRINT r\$

1032 **SELECT ON r**

1033 =1:IF pbdone=0:**ResetPlugBoard:PlugBoard**

1034 =2:IF rodone =0:**ResetRotors :SetRotors**

1035 =3:pbdone=0:rodone=0

1036 =4:**LoadConfig**

1037 =5:IF **AIIDone:SaveConfig**

1038 =6:IF **AIIDone:Cipher**

1039 =7:IF **AIIDone:LoadSetup 0**

1040 =8:INK 6:CSIZE 0,0:CLS:CLS#0:PRINT#0,'Program Ends':STOP

1041 =9:**EnigmaDEMO:PAUSE 30:BLOCK 48,20,112,142,0**

1042 **END SELECT**

1043 **END REPEAT EnigmaMenu**

1044 **END DEFINE**



1046 DEFine FuNction AllDone:RETurn pbdone AND rodone:END DEFine

1048 DEFine PROCEDURE ResetPlugBoard

1049 LOCAL i:FOR i=0 TO 25 :pb%(i)=i

1050 pbc=0:pbdone=0:BLOCK 312,38,176,164,0

1051 INK 6:FOR i=0 TO 25:CURSOR i\*4+84,28,0,0:PRINT CHR\$(65+i) **Note:** CURSOR Graphic Coordinates

1052 INK 4:FOR i=0 TO 25:CIRCLE i\*4+85,12,,6

1053 END DEFine



1055 DEFine PROCEDURE PlugBoard

1056 LOCAL k1,k2:CLS#:INK#0,4

1057 REPEAT PBloop1

1058 CURSOR#0, 32,10:PRINT#0,'Select TEN Pairings of '

1059 PRINT#0,'Two unused Letters [Esc] Reset'

1060 REPEAT PBloop2

1061 IF pbc=10:CSIZE#0,0,0:EXIT PBloop2

1062 REPEAT key1:k1=GetLetter(1):IF k1=27 OR pb%(k1)=k1:EXIT key1

1063 IF k1=27:ResetPlugBoard:RETurn

1064 INK 2:CURSOR k1\*4+84,28,0,0:PRINT CHR\$(65+k1)

1065 REPEAT key2:k2=GetLetter(0):IF k1<>k2 AND pb%(k2)=k2:EXIT key2

1066 pb%(k1)=k2:pb%(k2)=k1:pbc=pbc+1:ShowPlugUp

1067 END REPEAT PBloop2

1068 IF Confirm('Confirm Plug-Board Setup Correct'):EXIT PBloop1

1069 CLS#:ResetPlugBoard

1070 END REPEAT PBloop1

1071 pbdone=1:FOR i=0 TO 25:ts%(i)=pb%(i)

1072 END DEFine



1074 DEFine FuNction GetLetter(f)

1075 LOCAL k

1076 REPEAT GL\_Lp

1077 k=CODE(INKEY\$(-1) )

1078 IF f=1 AND k=27 :RETurn k

1079 IF f=2 AND k=27 OR f=2 AND k=35 : RETurn k

1080 IF k>96 AND k<123:k=k-32

1081 IF k>=65 and k<=90 : RETurn k-65

1082 END REPEAT GL\_Lp

1083 END DEFine



1085 DEFine PROCEDURE ShowPlugUp

1086 INK 2:BEEP 300,80,20,5,8,0,0,0

1087 CURSOR k1\*4+84,28,0,0:PRINT CHR\$(65+k1)

1088 CURSOR k2\*4+84,28,0,0:PRINT CHR\$(65+k2)

1089 CURSOR k2\*4+84,14,0,0:PRINT CHR\$(65+k1)

1090 CURSOR k1\*4+84,14,0,0:PRINT CHR\$(65+k2)

1091 LINE k1\*4+85,22 TO k1\*4+85,15:PAUSE 5

1092 LINE k2\*4+85,22 TO k2\*4+85,15:INK 7

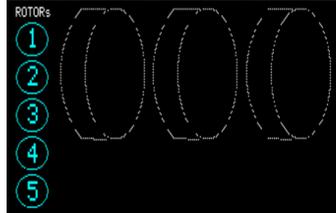
1093 CURSOR 190,164:CSIZE 2,1:PRINT FILL\$(' ',2-LEN(pbc));pbc:CSIZE 0,0

1094 END DEFine

```

1096 DEFINE PROCEDURE ResetRotors
1097 sp%(0)=0:sp%(1)=0:sp%(2)=0:rs%(1)=0:rs%(2)=0
1098 cp%(0)=0:cp%(1)=0:cp%(2)=0
1099 CURSOR 0,0:CSIZE 2,1:INK 5
1100 FOR i=0 TO 4
1101   gr%(i)=-1:CIRCLE 72,98-i*14,6:CURSOR 72,98-i*14,-6,-9:PRINT i+1
1102 END FOR i
1103 CURSOR 0,0:CSIZE 0,0:INK 248:BLOCK 290,96,210,4,0:rodone=0
1104 FOR n=0 TO 2
1105   LINE 91+36*n,109 TO 101+36*n,109:CIRCLE 104+36*n,85,24,-48,PI
1106   LINE 91+36*n,61.5 TO 101+36*n,61:ARC 91+36*n,109 TO 91+36*n,61.5,PI/2.4
1107 END FOR n
1108 END DEFINE

```



```

1110 DEFINE PROCEDURE SetRotors
1111 LOCAL i,k,n:CLS#:INK#0,4
1112 REPEAT SRloop1
1113   CURSOR#0,32,10:PRINT#0,'Select in the Sequence:;'
1114   PRINT#0,' RIGHT MIDDLE LEFT: Enter Rotor Number [1..5]'
1115   FOR n=2 TO 0 STEP -1
1116     REPEAT SRloop2
1117       k=CODE(INKEY$(-1))
1118       SElect ON k=49 TO 53:k=k-49:IF gr%(k)=-1:EXIT SRloop2
1119     END REPEAT SRloop2
1120     INK 0:FILL 1:CIRCLE 72,98-k*14,5:FILL 0:CopyRotor n,k
1121     INK 6:CSIZE 2,1:CURSOR 103+n*36,87,-3,-5:PRINT k+1:CSIZE 0,0
1122     gr%(k)=n:DrawRotor n
1123   END FOR n
1124   IF CONFIRM('Confirm Rotor Setup is Correct'):EXIT SRloop1
1125   ResetRotors
1126 END REPEAT SRloop1
1127 FOR i=0 TO 4:ts%(i+26)=gr%(i):END FOR i:IF pbc=10:pbdone=1
1128 SetRotorRings:ts%(34)=rs%(1):ts%(35)=rs%(2):SetRotorStarts:rodone=1
1129 END DEFINE

```

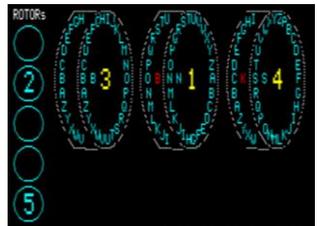


```

1131 DEFINE PROCEDURE CopyRotor(n,k)
1132 LOCAL i
1133 FOR i=0 TO 25:n%(n,i)=ro%(k,i):rr%(n,ro%(k,i))=i
1134 sp%(n)=0:cp%(n)=0:rs%(n)=0
1135 END DEFINE

1137 DEFINE PROCEDURE DrawRotor(n)
1138 LOCAL i:INK 6:BEEP 1000,100,20,8,3,0,12,9
1139 FOR i=0 TO 25
1140   INK 5:CURSOR 104+36*n+cx(i),85+cy(i),-3,-5:PRINT CHR$(65+(cp%(n)+i) MOD 26)
1141   CURSOR 95+36*n+cx(i),85+cy(i),-3,-5
1142   IF n>0 AND rs%(n)=(cp%(n)+i+rs%(n)) MOD 26:INK 2
1143   IF i<7 OR i>19 AND i<26:PRINT CHR$(65+(cp%(n)+i+rs%(n)) MOD 26)
1144 END FOR i
1145 END DEFINE

```



1147 DEFINE PROCEDURE SetRotorRings

1148 LOCAL k,n : INK#0,4

1149 REPEAT RSloop

1150 FOR n=2 TO 1 STEP -1

1151 CLS#0:PRINT#0,','Select Ring Setting for the ';

1152 IF n=2:PRINT#0,"Righthand";

1153 IF n=1:PRINT#0,'Middle';

1154 PRINT#0,' Rotor [A..Z]: ' : k=GetLetter(0):rs%(n)=k:DrawRotor n

1155 INK 2:CURSOR 234+n\*94,47:PRINT CHR\$(65+k)

1156 END FOR n

1157 IF Confirm('Confirm Ring-Settings are Correct'):EXIT RSloop

1158 END REPEAT RSloop

1159 END DEFINE

Select Ring Setting for the Middle Rotor (R..Z):



Confirm Ring-Settings are Correct y/n:

1161 DEFINE PROCEDURE SetRotorStarts

1162 LOCAL k,n,tp%(2) : INK#0,4

1163 REPEAT SPloop

1164 FOR n=2 TO 0 STEP -1

1165 CLS#0:PRINT#0,','Select Start Position for the ';

1166 IF n=2:PRINT#0,'Righthand ';

1167 IF n=1:PRINT#0,'Middle ';

1168 IF n=0:PRINT#0,'Lefthand ';

1169 PRINT#0,'Rotor [A..Z]: ' : k=GetLetter(1):tp%(n)=k

1170 REPEAT SPloop2

1171 IF cp%(n)=k:CURSOR 257+n\*94,47:PRINT CHR\$(65+k):EXIT SPloop2

1172 AdvanceRotor n:PAUSE 6

1173 END REPEAT SPloop2

1174 END FOR n

1175 IF Confirm('Confirm Start Positions are Correct'):EXIT SPloop

1176 END REPEAT SPloop

1177 sp%(0)=tp%(0):sp%(1)=tp%(1):sp%(2)=tp%(2):STRIP 0:INK 6

1178 ts%(31)=sp%(0):ts%(32)=sp%(1):ts%(33)=sp%(2):STRIP 0:INK 6

1179 END DEFINE

Select Start Position for the Righthand Rotor (R..Z):



Confirm Start Positions are Correct y/n:

1181 DEFINE FUNCTION Confirm(m\$)

1182 LOCAL r\$

1183 REPEAT Cloop1

1184 CLS#0:CURSOR#0,24,10:PRINT#0,m\$,' y/n: ':r\$=INKEY\$(#0,-1):PRINT#0,r\$

1185 PAUSE 20:IF r\$='y' OR r\$='yes':CLS#0:RETURN 1

1186 PAUSE 20:IF r\$='n' OR r\$='no':CLS#0:RETURN 0

1187 END REPEAT Cloop1

1188 END DEFINE Confirm

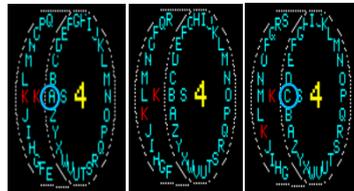
1190 DEFINE PROCEDURE AdvanceRotor(n)

1191 IF n<0 OR n>2:RETURN

1192 cp%(n)=(cp%(n)+1) MOD 26:DrawRotor n

1193 IF n>0:IF rs%(n)=(cp%(n)-1) MOD 26:AdvanceRotor n-1

1194 END DEFINE



Stepping A

to B

through to C

## 1200 REMark Enigma CODE

### 1202 DEFine PROCEDURE Cipher

```
1203 LOCAL lk:lk=0:num=0:CLS#0,:lNK#0,4
1204 CURSOR#0,172,10:PRINT#0,'Type Message[A...Z max 75] - Return[Esc] Send[#]CLS'
1205 Draw_Morse_Key 0,2,24,8:nk=GetLetter(2)
1206 REPEAT Cipher_lp
1207 IF nk=35:Send_Morse:LoadSetup 0:EXIT Cipher_lp
1208 IF nk=27 OR num>75:EXIT Cipher_lp
1209 STRIP 0:lNK 5:CipherMess:lk=Encode(nk):CipherCODE:num=num+1
1210 CipherLamp 7,0:nk=GetLetter(2):CipherLamp 0,7:AdvanceRotor 2
1211 END REPEAT Cipher_lp
1212 END DEFine
```

Type Letters (max 75) - Lights Cipher CODE Lamp - (Rtn Enter)

### 1214 DEFine FuNction Encode(k)

```
1215 LOCAL i,O :IF >25:RETURN: ELSE O=pb%(k)
1216 FOR i=2 TO 0 STEP -1:O=(r%(i,(cp%(i)+O) MOD 26)-cp%(i)) MOD 26
1217 O=rf%(O)
1218 FOR i=0 TO 2:O=(r%(i,(cp%(i)+O) MOD 26)-cp%(i)) MOD 26
1219 RETURN pb%(O)
1220 END DEFine
```



### 1222 DEFine PROCEDURE CipherLamp(lnk1,lnk2)

```
1223 IF lk>25:RETURN
1224 lNK lnk1:STRIP lnk1:FILL 1:CIRCLE kcx%(lk),kcy%(lk),2,8:FILL 0
1225 lNK lnk2:CURSOR kcx%(lk),kcy%(lk),-3,-4:PRINT CHR$(65+lk)
1226 END DEFine
```

### 1228 DEFine PROCEDURE CipherCODE

```
1229 IF lk>25:RETURN
1230 Code$=Code$&CHR$(65+lk):xc=xc+6:num=LEN(Code$)
1231 CURSOR xc,yc:PRINT Code$(num)
1232 IF num MOD 5=0:xc=xc+4
1233 IF num MOD 25=0:xc=0 :yc=yc+10
1234 END DEFine
```

```
Cipher CODE:
IF YOU CAN

Message: XMGCVPLH
```

### 1236 DEFine PROCEDURE CipherMess

```
1237 IF kk>25 : RETURN
1238 xs=xs+6:CURSOR 48+xs,208:PRINT CHR$(65+nk):str$=str$&CHR$(65+nk)
1239 END DEFine
```

### 1241 DEFine PROCEDURE Send\_Morse

```
1242 FOR i=1 TO LEN(Code$)
1243 lk=CODE(Code$(i))-65:Write_Morse_key lk+1::Beep_Morse_key lk+1,1
1244 END FOR i
1245 END DEFine
```

### 1247 DEFine PROCEDURE Write\_Morse\_key(lk)

```
1248 CURSOR#0,84,10:PRINT#0,CHR$(ML+64);'
1249 mx=100:FOR a=1 TO 4:mw=Morse%(lk,a):BLOCK#0,mw,3,mx,14,7:mx=mx+mw+4
1250 END DEFine
```

### 1252 DEFine PROCEDURE Beep\_Morse\_key(lk%,sp)

Note; sp single pause length

```
1253 FOR a=1 TO 4:ps=Morse%(lk,a):IF ps>0:BEEP 250*ps*sp,5,0,0,0,0,0:PAUSE 8*sp
1254 PAUSE 12*sp
1255 END DEFine
```

### 1300 REMark Enigma File Access

#### 1302 DEFine PROCEDURE SaveConfig

```
1303 LOCAL i:CLS#0:eck=0:f$=""
1304 INPUT#0,\\Enter Filename: ';f$;' ';:OPEN_NEW#3,f$
1305 IF KEYROW(7)=64:RETURN
1306 IF eck=1:PRINT#0,'...DEVICE ERROR':CLOSE#3:PAUSE 50:eck=0:CLS#0:RETURN
1307 IF eck=0:CURSOR#0,140,10:PRINT#0,' Saving.:',CLS#0,2:CLS#0,4
1308 FOR i=0 TO 35:PRINT#3,ts%(i):PRINT#0,!.':;PAUSE 2:END FOR i:CLOSE#3
1309 END DEFINE
```

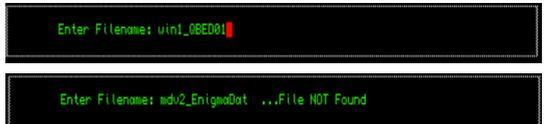
**Note:** New venture SAVE and LOAD use IINPUT#0 for Filename entry, which includes the device name.

If the local setup has device and Directory reroutes in place. Saved Files could be sent there instead of being shown as a 'DEVICE ERROR'



#### 1311 DEFine PROCEDURE LoadConfig

```
1312 LOCAL i:CLS#0:f$=""eck=0:f$=' '
1313 INPUT#0,\\Enter Filename: ';f$;' ';:IF f$=""eck=1:END IF :OPEN_IN#3,f$
1314 IF eck=1:PRINT#0,'...File NOT Found':CLOSE#3:PAUSE 50:eck=0:CLS#0:RETURN
1315 FOR i=0 TO 35:INPUT#3,ts%(i):END FOR i:CLOSE#3:pbk=10:LoadSetup 1
1316 END DEFINE
```



#### 1318 DEFine PROCEDURE LoadSetup (sm)

```
1319 IF sm=1
1320 ResetPlugBoard:ResetRotors:FOR i=0 TO 25:pb%(i)=ts%(i)
1321 pbk=10:FOR i=0 TO 25:k1=i:k2=pb%(i):IF k1<>k2:ShowPlugUp
1322 END IF
1323 IF sm=2:ResetRotors:PAUSE 20
1324 FOR i=0 TO 4:gr%(i)=ts%(i+26):IF gr%(i)<>-1:CopyRotor gr%(i),i
1325 FOR i=2 TO 0 STEP -1
1326 r=-1:REPEAT Rotorlp:r=r+1:IF gr%(r)=i:EXIT Rotorlp
1327 INK 0:FILL 1:CIRCLE 72,98-r*14,5:FILL 0:DrawRotor i:PAUSE 10
1328 INK 6:CURSOR 103+i*36,88,-3,-5:CSIZE 2,1:PRINT r+1:CSIZE 0,0
1329 END FOR i
1330 rs%(1)=ts%(34):rs%(2)=ts%(35)
1331 FOR i=2 TO 1 STEP -1:DrawRotor i:INK 2:CURSOR 234+i*94,47:PRINT CHR$(65+rs%(i))
1332 sp%(0)=ts%(31):sp%(1)=ts%(32):sp%(2)=ts%(33):pbdone=1:rodone=1
1333 FOR i=2 TO 0 STEP -1
1334 cp%(0)=sp%(0):cp%(1)=sp%(1):cp%(2)=sp%(2):DrawRotor i
1335 CURSOR 257+i*94,47:PRINT CHR$(65+sp%(i)):PAUSE 5
1336 END FOR i
1337 CURSOR 54,204:CLS 4:xc=0:yc=164:str$=":BLOCK 170,34,6,162,0:Code$=":xs=0
1338 END DEFINE
```

1350 REMark **Enigma Machine Simulator: View Working DEMO**

1352 **DEFine PROCEDURE EnigmaDEMO**

1353 **ResetPlugBoard:ResetRotors**

1354 BLOCK 170,34,6,162,0:CURSOR 52,204:CLS#1,4

1355 INK 6:CURSOR 112,142:CSIZE 2,1:PRINT 'DEMO':CSIZE 0,0

1356 lk=0:num=0:sx=0:x=0:y=164:Code\$=":str\$=":CLS#0

1357 :

1358 REMark **\*\*\* PlugBoard Setup \*\*\***

1359 **RESTORE 1365**:FOR i=0 TO 25:pb%(i)=i

1360 FOR i=1 TO 10

1361 READ P\$:k1=CODE(P\$(1))-65:k2=CODE(P\$(2))-65

1362 pb%(k1)=k2:pb%(k2)=k1:pbc=i:**ShowPlugUp**:PAUSE 8

1363 END FOR i

1364 REMark **\*\*\* PlugBoard Data \*\*\***

1365 DATA 'AC','BM','DJ','EN','FP','GR','HL','IT','KO','SW'

1366 :

1367 REMark **\*\*\* Rotors Setup \*\*\***

1368 FOR n=2 TO 0 STEP -1

1369 READ k:k=k-1

1370 INK 0:FILL 1:CIRCLE 72,98-k\*14,5:FILL 0:**CopyRotor n,k**

1371 INK 6:CURSOR 103+n\*36,88,-3,-5:CSIZE 2,1:PRINT k+1:CSIZE 0,0

1372 gr%(k)=n : **DrawRotor n**:PAUSE 10

1373 END FOR n

1374 FOR i=2 TO 1 STEP -1

1375 READ P\$:rs%(i)=CODE(P\$)-65:**DrawRotor i**

1376 INK 2:CURSOR 234+i\*94,47:PRINT CHR\$(rs%(i)+65)

1377 END FOR i

1378 FOR i=2 TO 0 STEP -1

1379 READ P\$:sp%(i)=CODE(P\$)-65:cp%(i)=sp%(i)

1380 **DrawRotor i**:CURSOR 256+i\*94,47:PRINT CHR\$(65+sp%(i)):PAUSE 10

1381 END FOR i

1382 REMark **\*\*\* Rotor Settings [Note: Rotors READ Right to Left] \*\*\***

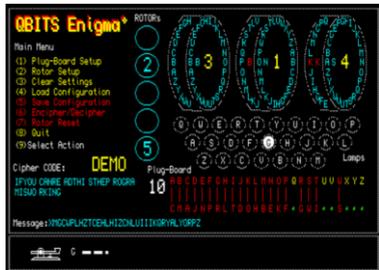
1383 DATA 4,1,3,'K','B','S','N','B'

1384 :

```

Test example2
Plug board:  AC BM DJ EN FP GR HL IT K0 SW
Rotors:      3 (Left), 1 (Middle), 4 (Right)
Rings:       B      K
Start:       B      N      S

XMGCW PLHZZ CEHLH IZCNT UIITK QRYAI YORPZ
    
```



1385 REMark **\*\*\* Cipher CODE Display \*\*\***

1386 Mes\$='XMGCWPLHZZCEHLHIZCNLUIIIKQRYALYORPZ'

1387 **Draw\_Morse\_key 0,2,2,4,8**

1388 FOR i=1 TO 35

1389 nk=CODE(Mes\$(i))-65 : STRIP 0:INK 5:**CipherMess**

1390 lk=**Encode**(nk):**CipherCODE**

1391 **Write\_Morse\_key lk+1 : CipherLamp 7,0**

1392 **Beep\_Morse\_key lk+1,1:CipherLamp 0,7:AdvanceRotor 2**

1393 END FOR i

1394 **END DEFINE**

1400 REMark **Enigma Screen Setup**

1402 **DEFine PROCEDURE Init\_win**

1403 WINDOW#1,512,256,gx,gy :PAPER#1,0:CLS#1:SCALE#1,112,0,0

1404 WINDOW#2,512,220,gx,gy

1405 WINDOW#1,512,220,gx,gy :PAPER#1,0:BORDER#1,1,255:INK#1,6:CSIZE#1,0,0:CLS#1

1406 WINDOW#0,512,34,gx,gy+222:PAPER#0,0:BORDER#0,1,255:INK#0,4:CSIZE#0,0,0:CLS#0

1407 CSIZE#1,2,1:OVER#1,1

1408 INK#1,2:FOR i=0 TO 1:CURLSOR#1,5+i,4:PRINT#1,'QBITS Enigma'

1409 INK#1,6:FOR i=0 TO 1:CURLSOR#1,7+i,6:PRINT#1,'QBITS Enigma'

1410 CSIZE#1,0,0:OVER#1,0:INK 7:**RESTORE 1311**

1411 :

1412 CURSOR 6,32:PRINT 'Main Menu' :CURSOR 172,4:PRINT 'ROTORS'

1413 CURSOR 6,128:PRINT '( )Select Action ':CURSOR 460,140:PRINT 'Lamps'

1414 CURSOR 4,150:PRINT 'Cipher CODE:' :CURSOR 4,208:PRINT 'Message:'

1415 CURSOR 190,152:PRINT 'Plug-Board' : **ResetPlugBoard** : **ResetRotors**

1416 pbdone=0 : rodone=0 : str\$="" : xs=0 : Code\$="" : xc=0 : yc=164

1417 :

1418 REMark **\*\*\* Establish Keyboard Layout \*\*\***

1419 **RESTORE 1432** : L=LANGUAGE:IF L<>33 AND L<>44 AND L<>49 :L=44

1420 **REPeat Getlang**

1421 **READ I1,r1\$,r2\$,r3\$**:IF I1=0:**EXIT Getlang**

1422 IF I1=L:kr\$(0)=r1\$:kr\$(1)=r2\$:kr\$(2)=r3\$

1423 **END REPeat Getlang**

1424 FOR i=0 TO 2

1425 FOR j=0 TO LEN(kr\$(i))-1

1426 p=CODE(kr\$(i,j+1))-65:kcx%(p)=90+i\*10+i\*6:kcy%(p)=54-i\*9

1427 INK 248:CIRCLE 90+j\*10+i\*6,54-i\*9,3.8

1428 INK 7:CURLSOR 90+j\*10+i\*6,54-i\*9,-3,-4:PRINT kr\$(i,j+1)

1429 END FOR j

1430 END FOR i

1431 :

1432 DATA 33,'AZERTYUIOP','QSDFGHJKLM','WXCVRN'

1433 DATA 44,'QWERTYUIOP','ASDFGHJKL','ZXCVRNM'

1434 DATA 49,'QWERTZUIOP','ASDFGJKL','YXCVRNM'

1435 DATA 0, ", "

1436 :

1437 REMark **\*\*\* Pre-calculate Coordinates for Rotor Alphabet \*\*\***

1438 FOR i=0 TO 25:cx(i)=-9\*COS(PI\*i/13):cy(i)=21\*SIN(PI\*i/13)

1439 :

1440 REMark **\*\*\* Load Wiring Definitions for the Five Rotors \*\*\***

1441 **RESTORE 1443**:FOR i=0 TO 4:FOR j=0 TO 25 : **READ ro%(i,j)**:END FOR j:END FOR i

1442 :

1443 DATA 13,6,1,8,19,2,14,16,9,10,3,11,4,0,22,20,5,7,23,15,21,12,18,25,24,17

1444 DATA 19,23,17,13,5,11,15,16,7,0,14,3,12,2,21,22,1,6,9,10,4,8,20,18,25,24

1445 DATA 18,9,0,14,22,11,8,10,12,15,13,19,25,1,7,3,2,23,17,20,21,16,6,4,5,24

1446 DATA 10,1,5,2,23,3,13,6,8,25,20,19,21,11,7,0,14,12,9,16,18,22,17,24,15,4

1447 DATA 12,22,13,7,15,2,25,17,3,1,14,18,0,4,5,11,6,19,23,20,21,9,16,8,10

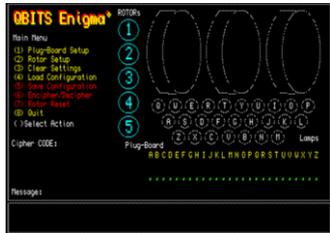
1448 :

1449 REMark **\*\*\* Load 'Wiring' Definition for the Reflector \*\*\***

1450 **RESTORE 1452**:FOR i=0 TO 25:**READ rf%(i)**

1451 :

1452 DATA 6,24,16,19,9,7,0,5,15,4,18,25,17,23,21,8,2,12,10,3,22,14,20,13,1,11



```

1454 REMark *** Load Morse Code dots & dashes ****
1455 RESTORE 1457:FOR k=1 TO 26:FOR c=1 TO 4:READ Morse%(k,c):END FOR c:END FOR k
1456 :
1457 DATA 4,12,0,0, 12,4,4,4, 12,4,12,4, 12,4,4,0, 4,0,0,0, 4,4,12,4, 12,12,4,0
1458 DATA 4,4,4,4, 4,4,0,0, 4,12,12,12, 12,4,12,0, 4,12,4,4, 12,12,0,0, 12,4,0,0
1459 DATA 2,12,12,0, 4,12,12,4, 12,12,4,12, 4,12,4,0, 4,4,4,0, 12,0,0,0
1460 DATA 4,4,12,0, 4,4,4,12, 4,12,12,0, 12,4,4,12, 12,4,12,12, 12,12,4,4
1461 END DEFine

1463 DEFine PROCEDURE Draw_Morse_Key(ch,col,mx,my)
1464 INK#ch,col:LINE#ch,mx-10,my+5 TO mx+6,my+5 TO mx+5,my+7 TO mx+9,my+7
1465 LINE#ch TO mx+8.5,my+4 TO mx-10,my+4 TO mx-10,my+5
1466 LINE#ch,mx-10,my TO mx+9,my TO mx+7,my+1 TO mx-8,my+1 TO mx-10,my
1467 LINE#ch,mx-6,my+1 TO mx-6,my+5:ARC#ch TO mx-2,my+5,-PI:LINE#ch TO mx-2,my+1
1468 LINE#ch,mx+1,my+1 TO mx+1,my+2 TO mx+3,my+2 TO mx+3,my+1
1469 FILL#ch,1:CIRCLE#ch,mx-4,my+4,1.2:FILL#ch,0:CIRCLE#ch,mx+2,my+3.5,.5
1470 END DEFine

```

### QBITS EnigmaSE – Encryption Notes

The Enigma machine could process letters only, therefore, numbers were written out in full ZERO, ONE, TWO, THREE ... TEN, to reduce multiple Zeros, CENTA(00) MILLE(000) ; MYRIA(000) were used. For punctuations rare letter combinations, Comma [ZZ], Full Stop [XX] and names defined by [X] such as XLONDONX.

**Note:** Two Additional Programs from Ian Price's QL Today Enigma Article

Make a **New Reflector** Load the main program then MERGE the data file.

```

110 OPEN-NEW#3, win3_enigma_reflector_dat
120 DIM r%(25) : RANDOMJSE
130 FOR i=0 TO 25 : r%(i)=i
140 FOR i=0 TO 25
150 IF r%(i)=i THEN
160 REPEAT lp1 :a%-RIID(25) :IF a%<>i AND r%(a%)=a% :EXIT lp1 : r%(i)=a% : r%(a%)=i
170 END IF
180 END FOR i
190 PRINT#3,'5040 DATA ' :FOR i=0 TO 24 : PRINT#3,r%(i),' ' :END FOR i : PRINT#3,r%(25)
200 CLOSE #3

```

Make **Five New Rotors** Load the main program then MERGE the data file.

```

110 OPEN_NEW#3,win3_enigma_rotors_dat
120 DIM r%(25) : RANDOMISE : Inum=4980
130 FOR rotor=1 TO 5
140 FOR i=0 TO 25 : LET r%(i)=i
150 PRINT #3,Inum; ' DATA ' :last=25
160 REPEAT rlp1
170 IF last=0 THEN EXIT rlp1
180 a=RND(last-1) : PRINT#3,r%(a); ' ' : r%(a)=r%(last) : last=last-1
200 END REPEAT rlp1
210 PRINT#3,r%(0) : Inum=Inum+10
220 END FOR rotor
230 CLOSE#3

```