

Introduction

The Enigma was a WWII 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 Rotor 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 1 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 rotate its Alphabet letters to align with different 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 extended the Enigma Machines combinations.

Code Breaking

The breaking of the Enigma Machine Code by members of the Bletchley Park British WWII Secret Code Centre was one of its greatest achievements. The electro-mechanical technique of the Enigma Machine Code Generation was cracked using a form of reverse engineering. The Invented Decipher machines were built with a series of revolving drums that would click around seeking out code combinations that matched. Later development went on to create a Programmable Code Braking Machine the Colossus, the forerunner of today's modern computer.

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

QBITS EnigmaSE

Display shows the components Plug-Board, Rotors, Lamp outputs with workings plus a space for a typed in Message and an area for the Cipher CODE output.

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 - Plug-board and Rotors.

- (1) Plug-Board Setup - Yellow or Green [Active/Set]
- (2) Rotor Setup - Yellow or Green [Active/Set]
- (3) Clear Settings - Yellow [Active]
- (4) Load Configuration - Yellow Enter Filename
- (5) Save Configuration - Red/Yellow Enter Filename
- (6) Encipher/Decipher - Red/Yellow Enter Message
- (7) Rotor Reset - Red /Yellow [Active]
- (8) Quit
- (?) Selected Action – Displays Menu Number



QBITS EnigmaSE – (1) Plug-Board Setup

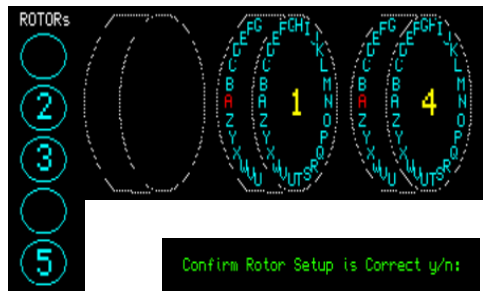
This requires 10 Plug-Board settings, select two unused [different] Letters for each. The Letters change colour to Red with a line draw between upper and lower row set with crossover Letter. The visa versa crossover Letter position is also set. The 10 Plug ups are therefore shown as twenty entries on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



QBITS EnigmaSE – (2) Rotor Setup

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

Select three Rotors, 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.



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.

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

Type a Filename ie. 'win1_EnigmaConfig01' this is entered via INPUT#0 into the QL Interpreter. Warning the results can be precarious, **WHEN ERROR** and **IF** statements cover some of the unwanted outcomes, but not all. If unavailable to Load, '**File NOT found**' should be displayed.

(4) LOAD - Select (N)ew for just Plug and Rotor settings, additionally send (M)essage or (C)ode string into Message buffer. The Cypher Code and Morse output is generated.

(5) SAVE File includes Plug-Board, Rotor settings, Message and CODE Strings. **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 active and changes to [Yellow/White]. 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 is set at 250 Letters. When message row is full, Characters PAN left to show last entry. As CODE generated exceeds display area, rows SCROLL up.

[**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

When active (7) clears any CODE and Message displayed and resets Rotors back to last Settings. Type CODE as Message and Decipher should display the original Letters.

QBITS EnigmaSE – (9) DEMO

Option (9) 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, the **Rotors** turn and the corresponding **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 'dit' & 'dah' Beep 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.

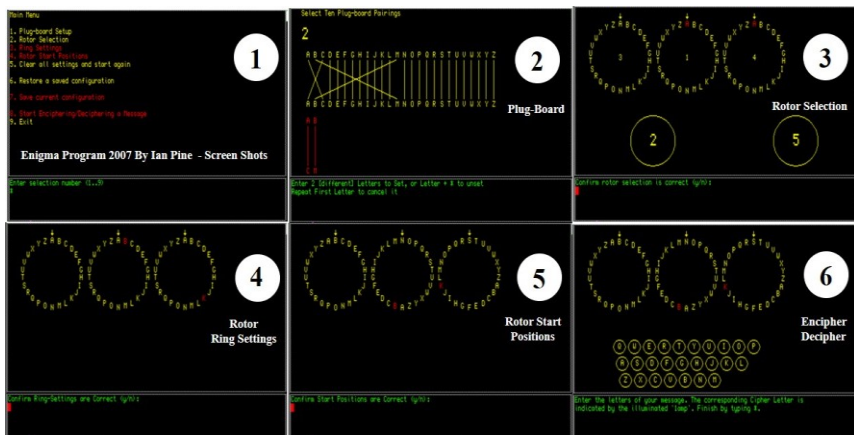
The two most fundamental features of the original system have been retained:

- (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

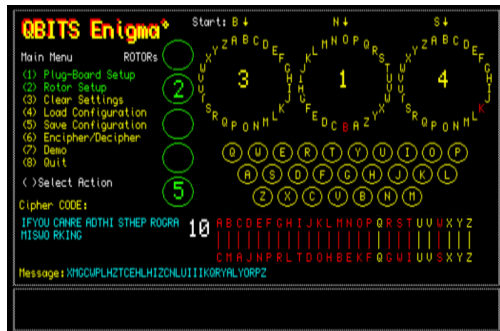
Having Downloaded and scanned a pdf copy of Ian Pine’s QL Today article, after several attempts I finally made the lines of Program code into a plain text format that the QL Interpreter accepted as a SuperBASIC Program. However, a number of lines with ‘MISTAKE’ still needed correcting. The problem was the Optical Character Recognition (OCR) interpretation and myself in scanning the articles print. The Font had L in lower case ‘l’, ‘I’ and numeral one ‘1’, all looking very similar.

After some hours 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 the Lamps set 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 buffer.



QBITS Enigma Plug-Board

The display uses an upper and lower line of A..Z each linked with a [Yellow/White] vertical line. As a Plug-up pair is chosen ie. A-C the colour changes **A** to **Red** with lower line printed with the crossover Letter in this case a **C** with a vertical line [Red] draw between. The corresponding pair **C-A** is changed likewise. The unused Plug positions ie 'Q UV XYZ as shown in Demo remain directly connected ie Q-Q. Once 10 pairs have been chosen a y/n Prompt appears, accept or clear. Pressing [**Esc**] while entering Plug-Board Letters will also abort and clear entries. Select (**1**) to Set a new arrangement.

QBITS Enigma Rotors

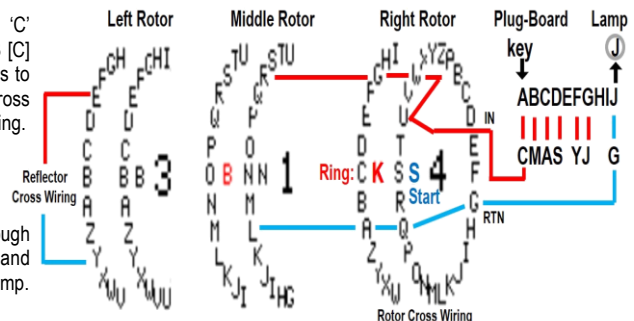
For a combined screen, I felt the Setup display of the Rotor Ring and Start positions were not truly represented. This prompted some further investigation into their actions.

The Rotors are constructed as two Rings, The Rotors right-hand inputs represent the letters of the Alphabet A..Z or positions 1 to 26 which 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: Plug-Bboard and Rotors. 'C' connected to Right Rotor pin 03 [C] but Start position has moved this to 'U' on Rotor, this is internally cross wired to W then set to G of 2nd Ring. Middle, Left 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, 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.

```
1439 REMark *** Pre-calculate Coordinates For Rotor Alphabet ***
1440 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:

```
1141 DEFINE PROCEDURE DrawRotor(n)
1142 LOCAL i:INK 6:BEEP 1000,100,20,8,3,0,12,9
1143 FOR i=0 TO 25
1144   INK 5:CURSOR 104+36*n+cx(i),85+cy(i),-3,-5:PRINT CHR$(65+(cp%(n)+i) MOD 26)
1145   CURSOR 95+36*n+cx(i),85+cy(i),-3,-5
1146   IF n>0 AND rs%(n)=(cp%(n)+i+rs%(n)) MOD 26:INK 2
1147   IF i<7 OR i>19 AND i<26:PRINT CHR$(65+(cp%(n)+i+rs%(n)) MOD 26)
1148 END FOR i
1149 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

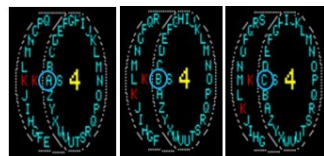


Ring & Start

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.

```
1194 DEFINE PROCEDURE AdvanceRotor(n)
1195 IF n<0 OR n>2:RETURN
1196 cp%(n)=(cp%(n)+1) MOD 26:DrawRotor n
1197 IF n>0:IF rs%(n)=(cp%(n)-1) MOD 26:AdvanceRotor n-1
1198 END DEFINE AdvanceRotor
```



Stepping A

to B

through to C

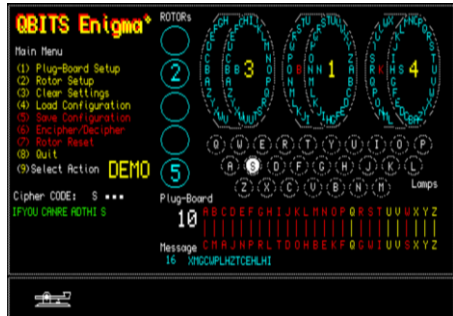
QBITS EnigmaSE Encipher/Decipher

Once the Plug-Board and Rotors are set a Message or Code can be typed to Encipher or Decipher. For example, type a simple message and exit with [**Esc**] then use **(5)** to Save. Clear settings with **(3)** then Load with **(4)** and select **(C)**ode. This types the CODE into Message buffer and Deciphers to original Message as Cipher Code output.

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 limit is 250 characters. The Message below the Plug-Board shows characters PAN left across the screen. The CODE is set out in rows of five-character blocks that SCROLL up.



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 ... 5

```

1135 DEFine PROCedure CopyRotor(n,k)
1136 LOCAl i
1137 FOR i=0 TO 25:ri%(n,i)=ro%(k,i):rr%(n,ro%(k,i))=i
1138 sp%(n)=0:cp%(n)=0:rs%(n)=0
1139 END DEFine CopyRotor

```

QBITS EnigmaSE Reset

Menu item (7) Rotor Reset – checks if Plug-Board setup has not changed i.e. $pcb=10$ then sets Rotor setting back to last start position.

Menu item (3) Clear Settings – turns Yellow Plug-Board (1) and Rotors (2) ready for Setup and deactivates turns Red (5) Save (6) Encipher/Decipher (7) Rotor Reset.

QBITS EnigmaSE Code

1000 REMark **QBITS_EnigmaSE_bas** [QBITS Enigma SE 2024 QL40th - QPC2] vM30

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 Makeover of QL Enigma Prog by Ian Pine 2007**

1011 REMark adds Message Code Displays & Morse Code Playback

1012 REMark Load (**N**ew) (**M**)essage (**C**)ode +Select (9) for DEMO

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:CUSOR 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(**AllDone**):INK 2

1027 PRINT ' (5) Save Configuration': INK 6:IF NOT(**AllDone**):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:CUSOR 12,128:PRINT r\$

1032 **SELECT ON** r

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

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

1035 =3:**ResetPlugBoard**:**ResetRotors**:xs=0:xc=0:CLS#7:CLS#8

1036 =4:**LoadConfig**

1037 =5:IF **AllDone**:**SaveConfig**

1038 =6:IF **AllDone**:**Cipher**

1039 =7:IF **AllDone**:**LoadSetup** 0 :BLOCK 20,10,172,202,0

1040 =8:INK 6:CLS#0:PRINT#0,\, 'Program Ends':PAUSE 50:LRUN dn\$:STOP

1041 =9:**EnigmaDEMO**:PAUSE 30:BLOCK 48,20,112,124,0

1042 **END SELECT**

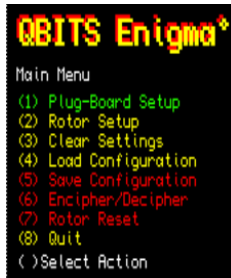
1043 **END REPEAT** EnigmaMenu

1044 **END DEFINE**



Note: Startup

Note: Set Plug-Board & Rotors



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

1048 DEFine PROCEDURE ResetPlugBoard

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

1050 BLOCK#1,24,20,190,164,0:BLOCK#1,276,38,218,164,0:INK 6

1051 FOR i=0 TO 25

1052 CURSOR i*4+84,28,0,0:PRINT CHR\$(65+i)

1053 CURSOR i*4+84,14,0,0:PRINT CHR\$(65+i)

1054 LINE i*4+85,22 TO i*4+85,15:END FOR i

1055 END FOR i

1056 BLOCK#1,80,10,90,150,0: BLOCK 18,10,172,202,0

1057 END DEFine



1059 DEFine PROCEDURE PlugBoard

1060 LOCAL k1,k2:CLS#0:INK#0,4

Select TEN Pairings of Two unused Letters

[Esc] Reset

1061 REPEAT PBloop1

1062 CURSOR#0, 32,10:PRINT#0,'Select TEN Pairings of ';

1063 PRINT#0,'Two unused Letters [Esc] Reset'

1064 REPEAT PBloop2

1065 IF pbcc=10:CSIZE#0,0,0:EXIT PBloop2

1066 REPEAT key1:k1=GetLetter(1):IF k1=27 OR pb%(k1)=k1:EXIT key1

1067 IF k1=27:ResetPlugBoard:RETurn

1068 INK 2:CURSOR k1*4+84,28,0,0:PRINT CHR\$(65+k1)

1069 REPEAT key2:k2=GetLetter(0):IF k1<>k2 AND pb%(k2)=k2:EXIT key2

1070 pb%(k1)=k2:pb%(k2)=k1:pbcc=pbcc+1:ShowPlugUp

1071 END REPEAT PBloop2

1072 IF Confirm('Confirm Plug-Board Setup Correct'):EXIT PBloop1

1073 CLS#0:ResetPlugBoard

Confirm Plug-Board Setup Correct y/n

1074 END REPEAT PBloop1

1075 pbdone=1:FOR i=0 TO 25:ts%(i)=pb%(i)

1076 END DEFine

1078 DEFine FuNction GetLetter(f)

1079 LOCAL k

1080 REPEAT GL_Lp

1081 k=CODE(INKEY\$(-1))

1082 IF f=1 AND k=27 :RETurn k

1083 IF f=2 AND k=27 OR f=2 AND k=35 : RETurn k

1084 IF k>96 AND k<123:k=k-32

1085 IF k>=65 and k<=90 : RETurn k-65

1086 END REPEAT GL_Lp

1087 END DEFine



1089 DEFine PROCEDURE ShowPlugUp

1090 INK 2:BEEP 300,80,20,5,8,0,0

1091 CURSOR k1*4+84,28,0,0:PRINT CHR\$(65+k1)

1092 CURSOR k2*4+84,28,0,0:PRINT CHR\$(65+k2)

1093 CURSOR k2*4+84,14,0,0:PRINT CHR\$(65+k1)

1094 CURSOR k1*4+84,14,0,0:PRINT CHR\$(65+k2)

1095 LINE k1*4+85,22 TO k1*4+85,15:PAUSE 5

1096 LINE k2*4+85,22 TO k2*4+85,15:INK 7

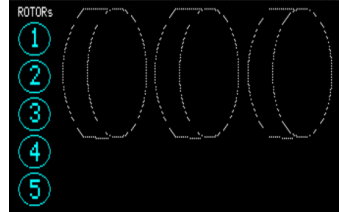
1097 CURSOR 190,164:CSIZE 2,1:PRINT FILL\$(" ",2-LEN(pbc));pbc:CSIZE 0,0

1098 END DEFine

```

1100 DEfINE PROCEDURE ResetRotors
1101 sp%(0)=0:sp%(1)=0:sp%(2)=0:rs%(1)=0:rs%(2)=0
1102 cp%(0)=0:cp%(1)=0:cp%(2)=0
1103 CURSOR 0,0:CSIZE 2,1:INK 5
1104 FOR i=0 TO 4
1105   gr%(i)=-1:CIRCLE 72,98-i*14,6:CURSOR 72,98-i*14,-6,-9:PRINT i+1
1106 END FOR i
1107 CURSOR 0,0:CSIZE 0,0:INK 248:BLOCK 290,96,210,4,0:rodone=0
1108 FOR n=0 TO 2
1109   LINE 91+36*n,109 TO 101+36*n,109:CIRCLE 104+36*n,85,24,-48,PI
1110   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
1111 END FOR n
1112 END DEfINE

```



```

1114 DEfINE PROCEDURE SetRotors
1115 LOCAL i,k,n:CLS#0:INK#0,4
1116 REPEAT SRloop1
1117   CURSOR#0,32,10:PRINT#0,'Select in the Sequence:';
1118   PRINT#0,' RIGHT MIDDLE LEFT: Enter Rotor Number [1..5]'
1119   FOR n=2 TO 0 STEP -1
1120     REPEAT SRloop2
1121       k=CODE(INKEY$(-1))
1122       SElect ON k=49 TO 53:k=k-49:IF gr%(k)=-1:EXIT SRloop2
1123     END REPEAT SRloop2
1124     INK 0:FILL 1:CIRCLE 72,98-k*14,5:FILL 0:CopyRotor n,k
1125     INK 6:CSIZE 2,1:CURSOR 103+n*36,87,-3,-5:PRINT k+1:CSIZE 0,0
1126     gr%(k)=n:DrawRotor n
1127   END FOR n
1128   IF Confirm('Confirm Rotor Setup is Correct'):EXIT SRloop1
1129   ResetRotors
1130 END REPEAT SRloop1
1131 FOR i=0 TO 4:ts%(i+26)=gr%(i):END FOR i:IF pbc=10:pbdone=1
1132 SetRotorRings:ts%(34)=rs%(1):ts%(35)=rs%(2):SetRotorStarts:rodone=1
1133 END DEfINE

```

Select in the Sequence RIGHT MIDDLE LEFT :Enter Rotor Number [1..5]

Confirm Rotor Setup is Correct y/n

```

1135 DEfINE PROCEDURE CopyRotor(n,k)
1136 LOCAL i
1137 FOR i=0 TO 25:ri%(n,i)=ro%(k,i):ri%(n,ro%(k,i))=i
1138 sp%(n)=0:cp%(n)=0:rs%(n)=0
1139 END DEfINE

```



```

1141 DEfINE PROCEDURE DrawRotor(n)
1142 LOCAL i:INK 6:BEEP 1000,100,20,8,3,0,12,9
1143 FOR i=0 TO 25
1144   INK 5:CURSOR 104+36*n+cx(i),85+cy(i),-3,-5:PRINT CHR$(65+(cp%(n)+i) MOD 26)
1145   CURSOR 95+36*n+cx(i),85+cy(i),-3,-5
1146   IF n>0 AND rs%(n)=(cp%(n)+i+rs%(n)) MOD 26:INK 2
1147   IF i<7 OR i>19 AND i<26:PRINT CHR$(65+(cp%(n)+i+rs%(n)) MOD 26)
1148 END FOR i
1149 END DEfINE

```

```

1151 DEFine PROCEDURE SetRotorRings
1152 LOCal k,n : INK#0,4
1153 REPEAT RSloop1
1154 FOR n=2 TO 1 STEP -1
1155   CLS#0:PRINT#0,\,'Select Ring Setting for the ';
1156   IF n=2:PRINT#0,"Righthand";
1157   IF n=1:PRINT#0,'Middle';
1158   PRINT#0,' Rotor [A..Z]: ' :k=GetLetter(0):rs%(n)=k:DrawRotor n
1159   INK 2:CURSOR 234+n*94,47:PRINT CHR$(65+k)
1160 END FOR n
1161 IF Confirm('Confirm Ring-Settings are Correct'):EXIT RSloop1
1162 END REPEAT RSloop1
1163 END DEFine

```

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



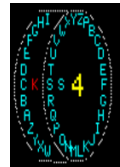
Confirm Ring-Settings are Correct y/n:

```

1165 DEFine PROCEDURE SetRotorStarts
1166 LOCAl k,n,tp%(2) : INK#0,4
1167 REPEAT SPloop1
1168 FOR n=2 TO 0 STEP -1
1169   CLS#0:PRINT#0,\,'Select Start Position for the ';
1170   IF n=2:PRINT#0,'Righthand ';
1171   IF n=1:PRINT#0,'Middle ';
1172   IF n=0:PRINT#0,'Lefthand ';
1173   PRINT#0,'Rotor [A. . Z]: ' :k=GetLetter(1):tp%(n)=k
1174   REPEAT SPloop2
1175     IF cp%(n)=k:CURSOR 257+n*94,47:PRINT CHR$(65+k):EXIT SPloop2
1176     AdvanceRotor n:PAUSE 6
1177   END REPEAT SPloop2
1178 END FOR n
1179 IF Confirm('Confirm Start Positions are Correct'):EXIT SPloop1
1180 END REPEAT SPloop1
1181 sp%(0)=tp%(0):sp%(1)=tp%(1):sp%(2)=tp%(2):STRIP 0:INK 6
1182 ts%(31)=sp%(0):ts%(32)=sp%(1):ts%(33)=sp%(2):STRIP 0:INK 6
1183 END DEFine

```

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



Confirm Start Positions are Correct y/n:

```

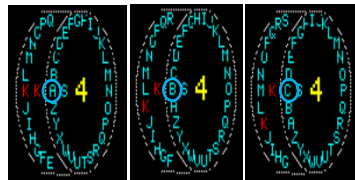
1185 DEFine FuNction Confirm(m$)
1186 LOCAl r$
1187 REPEAT Cloop1
1188   CLS#0:CURSOR#0,24,10:PRINT#0,m$,' y/n: ':r$=INKEY$(#0,-1):PRINT#0,r$
1189   PAUSE 20:IF r$=='y' OR r$=='yes':CLS#0:RETURN 1
1190   PAUSE 20:IF r$=='n' OR r$=='no' :CLS#0:RETURN 0
1191 END REPEAT Cloop1
1192 END DEFine Confirm

```

```

1194 DEFine PROCEDURE AdvanceRotor(n)
1195 IF n<0 OR n>2:RETURN
1196 cp%(n)=(cp%(n)+1) MOD 26:DrawRotor n
1197 IF n>0:IF rs%(n)=(cp%(n)-1) MOD 26:AdvanceRotor n-1
1198 END DEFine

```

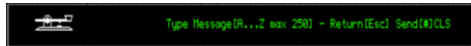


1200 REMark Enigma CODEing

```

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>250:EXIT Cipher_lp
1209 STRIP 0:INK 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

```



```

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=r%(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(INK1,INK2)
1223 IF lk>25:RETURN
1224 INK INK1:STRIP INK1:FILL 1:CIRCLE kcx%(lk),kcy%(lk),2.8:FILL 0
1225 INK INK2:CURSOR kcx%(lk),kcy%(lk),-3,-4:PRINT CHR$(65+lk)
1226 END DEFine

```

```

1228 DEFine PROCEDURE CipherCODE
1229 IF lk>25:RETURN :END IF :Code$=code$&CHR$(65+lk):xc=xc+6
1230 num=LEN(Code$):CURSOR#7,xc,yc:PRINT#7,Code$(num)
1231 CURSOR#1,172,202:PRINT#1,FILL$(' ',3-LEN(num))&num
1232 IF num MOD 5=0:xc=xc+4
1233 IF num MOD 25=0:xc=-6:yc=yc+10:END IF :IF yc>40:SCROLL#7,-10:yc=40
1234 END DEFine

```

```

1236 DEFine PROCEDURE CipherMess
1237 IF nk>25:RETURN :END IF :IF xs>292:PAN#8,-6:ELSE xs=xs+6
1238 CURSOR#8,xs,0:PRINT#8,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 mx=110:CURSOR#1,94,150:PRINT#1,CHR$(ML+64);' '
1249 FOR a=1 TO 4:mw=Morse%(lk,a):BLOCK#1,mw,3,mx,154,7:mx=mx+mw+4
1250 END DEFine

```

```

1252 DEFine PROCEDURE Beep_Morse_key(lk%,sp)
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

```

Note: sp single pause length

1300 REMark Enigma File Access

1302 DEFINE PROCEDURE SaveConfig

```
1303 LOCAL i:CLS#0:INK#0,7:f$="":eck=0:f$="
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:CLS#0: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
1309 PRINT#3,str$(Code$:CLOSE#3
1310 END DEFINE
```

Note: SAVE / LOAD uses Interpreter INPUT#0 for Filename entry, which must include device name. Save/Load File holds Settings: Message: Code:

Where the local setup has device and Directory re-routes in place. Saved Files could be sent there instead of shown as a 'DEVICE ERROR'



1312 DEFINE PROCEDURE LoadConfig

```
1313 LOCAL i:CLS#0:INK#0,7:f$="":eck=0:f$=" Note: If LOAD successful Select (N)ew (M)essage or (C)ode
1314 INPUT#0,\,'Enter Filename: ','f$,' ':IF f$="":eck=1:END IF :OPEN_IN#3,f$
1315 IF eck=1:PRINT#0,'...File NOT Found':CLOSE#3:PAUSE 50:eck=0:CLS#0:RETURN
1316 FOR i=0 TO 35:INPUT#3,ts%(i)\:END FOR i:INPUT#3,str$(Code$:CLOSE#3
1317 CLS#0:PRINT#0,\,'(N)ew (M)essage (C)ipher':PAUSE:CLS#0
1318 IF KEYROW(7)=64:LoadSetup 1:Cipher:RETURN
1319 Mes$=str$:IF KEYROW(2)=8:Mes$=Code$:END IF :LoadSetup 1:EnigmaCODE
1320 END DEFINE
```

Note: EnigmaPCB WELCOME TO QBITS ENIGMA MACHINE IN HONOUR OF BLETCHLEY PARK CODE BREAKERS

1322 DEFINE PROCEDURE LoadSetup (sm)

```
1323 IF sm=1
1324   ResetPlugBoard:ResetRotors:FOR i=0 TO 25:pb%(i)=ts%(i)
1325   pbc=10:FOR i=0 TO 25:k1=i:k2=pb%(i):IF k1<>k2:ShowPlugUp
1326 END IF
1327 IF sm=2:ResetRotors:PAUSE 20
1328 FOR i=0 TO 4:gr%(i)=ts%(i+26):IF gr%(i)<>-1:CopyRotor gr%(i),i
1329 FOR i=2 TO 0 STEP -1
1330   r=-1:REPEAT Rotorlp:r=r+1:IF gr%(r)=i:EXIT Rotorlp
1331   INK 0:FILL 1:CIRCLE 72,98-r*14,5:FILL 0:DrawRotor i:PAUSE 10
1332   INK 6:CURSOR 103+i*36,88,-3,-5:CSIZE 2,1:PRINT r+1:CSIZE 0,0
1333 END FOR i
1334 rs%(1)=ts%(34):rs%(2)=ts%(35)
1335 FOR i=2 TO 1 STEP -1:DrawRotor i:INK 2:CURSOR 234+i*94,47:PRINT CHR$(65+rs%(i))
1336 sp%(0)=ts%(31):sp%(1)=ts%(32):sp%(2)=ts%(33):pbdone=1:rodone=1
1337 FOR i=2 TO 0 STEP -1
1338   cp%(0)=sp%(0):cp%(1)=sp%(1):cp%(2)=sp%(2):DrawRotor i
1339   CURSOR 257+i*94,47:PRINT CHR$(65+sp%(i)):PAUSE 5
1340 END FOR i
1341 str$=":xs=0:Code$=":xc=-6:yc=0:CLS#7:CLS#8:BLOCK 80,10,90,150,0
1342 END DEFINE
```



1420 REMark *** Establish Keyboard Layout ***

1421 RESTORE 1434:L=LANGUAGE:IF L<>33 AND L<>44 AND L<>49 :L=44

1422 REPEAT Getlang

1423 READ I1,r1\$,r2\$,r3\$:IF I1=0:EXIT Getlang

1424 IF I1=L:kr\$(0)=r1\$:kr\$(1)=r2\$:kr\$(2)=r3\$

1425 END REPEAT Getlang

1426 FOR i=0 TO 2

1427 FOR j=0 TO LEN(kr\$(i))-1

1428 p=CODE(kr\$(i,j))-65:kcx%(p)=90+j*10+i*6:kcy%(p)=54-i*9

1429 INK 248:CIRCLE 90+j*10+i*6,54-i*9,3.8

1430 INK 7:CORSOR 90+j*10+i*6,54-i*9,-3,-4:PRINT kr\$(i,j+1)

1431 END FOR j

1432 END FOR i

1433 :

1434 DATA 33,'AZERTYUIOP','QSDFGHJKLM','WXCVBVN'

1435 DATA 44,'QWERTYUIOP','ASDFGHJKL','ZXCVBNM'

1436 DATA 49,'QWERTZUIOP','ASDFGJKL','YXCVBNM'

1437 DATA 0, " , "

1438 :

1439 REMark *** Pre-calculate Coordinates for Rotor Alphabet ***

1440 FOR i=0 TO 25:cx(i)=-9*COS(PI*i/13):cy(i)=21*SIN(PI*i/13)

1441 :

1442 REMark *** Load 'Wiring' Definitions for the Five Rotors ***

1443 RESTORE 1445:FOR i=0 TO 4:FOR j=0 TO 25 : READ ro%(i,j):END FOR j:END FOR i

1444 :

1445 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

1446 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

1447 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

1448 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

1449 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

1450 :

1451 REMark *** Load 'Wiring' Definition for the Reflector ***

1452 RESTORE 1454:FOR i=0 TO 25:READ rf%(i)

1453 :

1454 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

1455 :

1456 REMark *** Load Morse code dots & dashes ***

1457 RESTORE 1459:FOR k=1 TO 26:FOR c=1 TO 4:READ Morse%(k,c):END FOR c:END FOR k

1458 :

1459 DATA 4,12,0,0,12,4,4,4,12,4,12,4,12,4,4,0,4,0,0,4,4,12,4,12,12,4,0

1460 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

1461 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

1462 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

1463 END Define

1465 DEFine PROCEDURE Draw_Morse_Key(ch,col,mx,my)

1466 INK#ch,7:LINE#ch,mx-10,my+5 TO mx+6,my+5 TO mx+5,my+7 TO mx+9,my+7

1467 LINE#ch TO mx+8.5,my+4 TO mx-10,my+4 TO mx-10,my+5

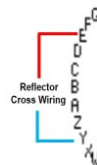
1468 LINE#ch,mx-10,my TO mx+9,my TO mx+7,my+1 TO mx-8,my+1 TO mx-10,my

1469 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

1470 LINE#ch,mx+1,my+1 TO mx+1,my+2 TO mx+3,my+2 TO mx+3,my+1

1471 FILL#ch,1:CIRCLE#ch,mx-4,my+4.4,1.2:FILL#ch,0:CIRCLE#ch,mx+2,my+3.5,.5

1472 END Define



A	---	N	---
B	---	O	---
C	---	P	---
D	---	Q	---
E	---	R	---
F	---	S	---
G	---	T	---
H	---	U	---
I	---	V	---
J	---	W	---
K	---	X	---
L	---	Y	---
M	---	Z	---



QBITS EnigmaSE - Encryption Notes

The Enigma machine could process only A...Z letters, therefore numbers were written out in full ZERO, ONE, TWO, THREE ... TEN. To reduce multiple Zeros, CENTA-(00) MILLE-(000) ; MYRIA-(0000) were used. For punctuations rare letter combinations, Comma [ZZ], Full Stop [XX] and names defined by [X] such as XLONDONX. There was no letter(s) for SPACE they were usually omitted, sometimes an [X] might be used.

Note: Two Additional Programs [amended] from Ian Price's QL Today Enigma Article.

Make **Five New Rotors** LRUN program then MERGE the _dat file with QBITS_EnigmaSE_bas.

```
100 REMark New Rotors
110 DIM r%(25),str$(50) : RANDOMISE : Inum=1445
120 OPEN_NEW#9,Dev$ _Enlgn_Rotors_dat Note: Dev$ default Drive
130 FOR rotor=1 TO 5
140   FOR i=0 TO 25 : r%(i)=i
150   str$=Inum&' DATA ' : last=25
160   REPEAT rlp1
170     IF last=0 THEN EXIT rlp1
180     a=RND(last-1) : str$=str$&r%(a)&' ' : r%(a)=r%(last) : last=last-1
190   END REPEAT rlp1
200   str$=str$&r%(0) : Inum=Inum+1 : PRINT#9,str$
210 END FOR rotor
220 CLOSE#9
```

Make **New Reflector** LRUN program then MERGE the _dat file with QBITS_EnigmaSE_bas.

```
100 REMark New Reflector
110 DIM r%(25),str$(50) : RANDOMISE : str$='1454 DATA '
120 FOR i=0 TO 25 : r%(i)=i
130 FOR i=0 TO 25
140   IF r%(i)=i THEN
150     REPEAT lp1 : a%=RND(25) : IF a%<>i AND r%(a%)=a% : EXIT lp1
160     r%(i)=a% : r%(a%)=i
170   END IF
180 END FOR i
190 FOR i=0 TO 24 : str$=str$&r%(i)&' ' : END FOR i : str$=str$&r%(25)
200 OPEN_NEW#9,dev$ _Enigma_Reflector_dat : PRINT#9,str$ : CLOSE #9
```

Note: Messages/Coding generated with new Rotors and Reflector Data will not be recoverable unless the _dat files are saved and can Merge with the Enigma Program as required in future use.