COM2108 – Autumn Functional Programming Assignment

Thomas Pearson

# The Enigma Machine Design

#### First Design Document

Upon reading the documentation on encoding the enigma simulation, I created Figure 1 (below), a simplified flow chart in order to visualise the operation of the enigma machine.

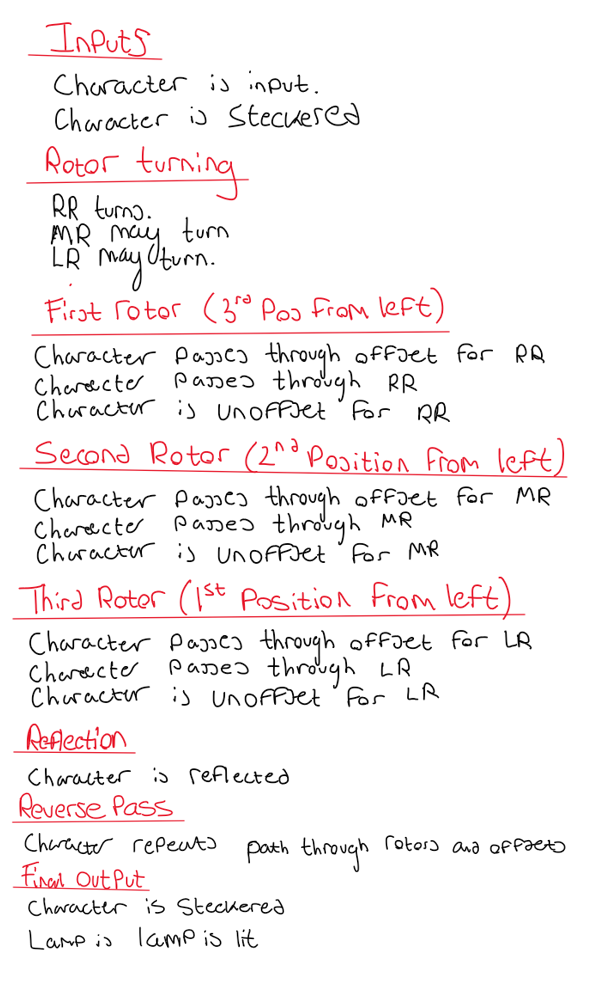


Figure 1 – Design First Draft

#### Rotor Turning

In this implementation the offset of the rotors is shifted after a character is pressed, but before it is ciphered (as shown in Figure 1). The knock-on offset position is also triggered upon reaching it. For example, if the knock-on position was position 17, the knock-on event (turning the connecting rotor) would be triggered upon the rotation from position 16 to position 17. However, in this implementation the characters are offset not the actual rotors. This is discussed further below.

#### Character Offsets

As demonstrated by Figure 1, instead of turning the rotors, which would be very resource intensive, I have decided to offset the input by the rotor amount. This way it is easier to keep track of the position of the current rotor and therefore the needed offset. It also ensures that the connecting rotor receives the correct character when the letter is passed to the next rotor.

For example, if the “Rotor I” was offset by three positions and the character “A” was input it would process the information as follows:

1. Shift “A” along the alphabet by three position to get “D”.
2. Pass letter “D” through the rotor to get “F”
3. Shift “F” backwards by three positions through the alphabet to get character “C”
4. Pass character “C” on to the next rotor and repeat the same process

This approach simulates the rotor having shifted three positions physically as the connecting rotor is unaware of the shifted previous rotor. It is only away of the position of the incoming signal in its own rotor. If we were to not “unshift” the encrypted value it would be receiving the incorrect value from the previous rotor.

#### Second Design Iteration

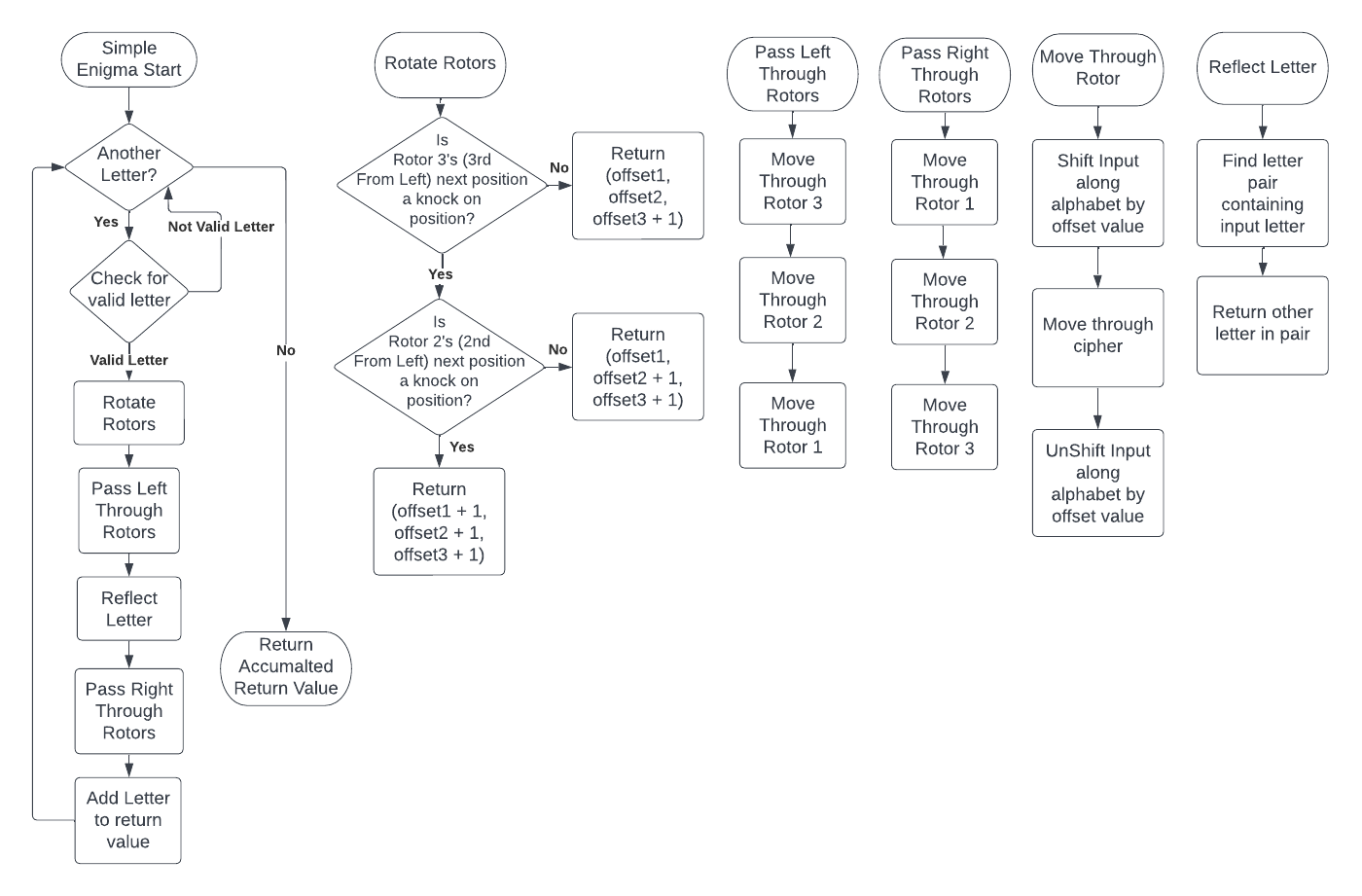
Once the main parts of the design were created it was divided further into smaller functions. The flow of these proposed functions is shown in Figures 2 and 3.

Figure 2 - Simple Enigma Operation

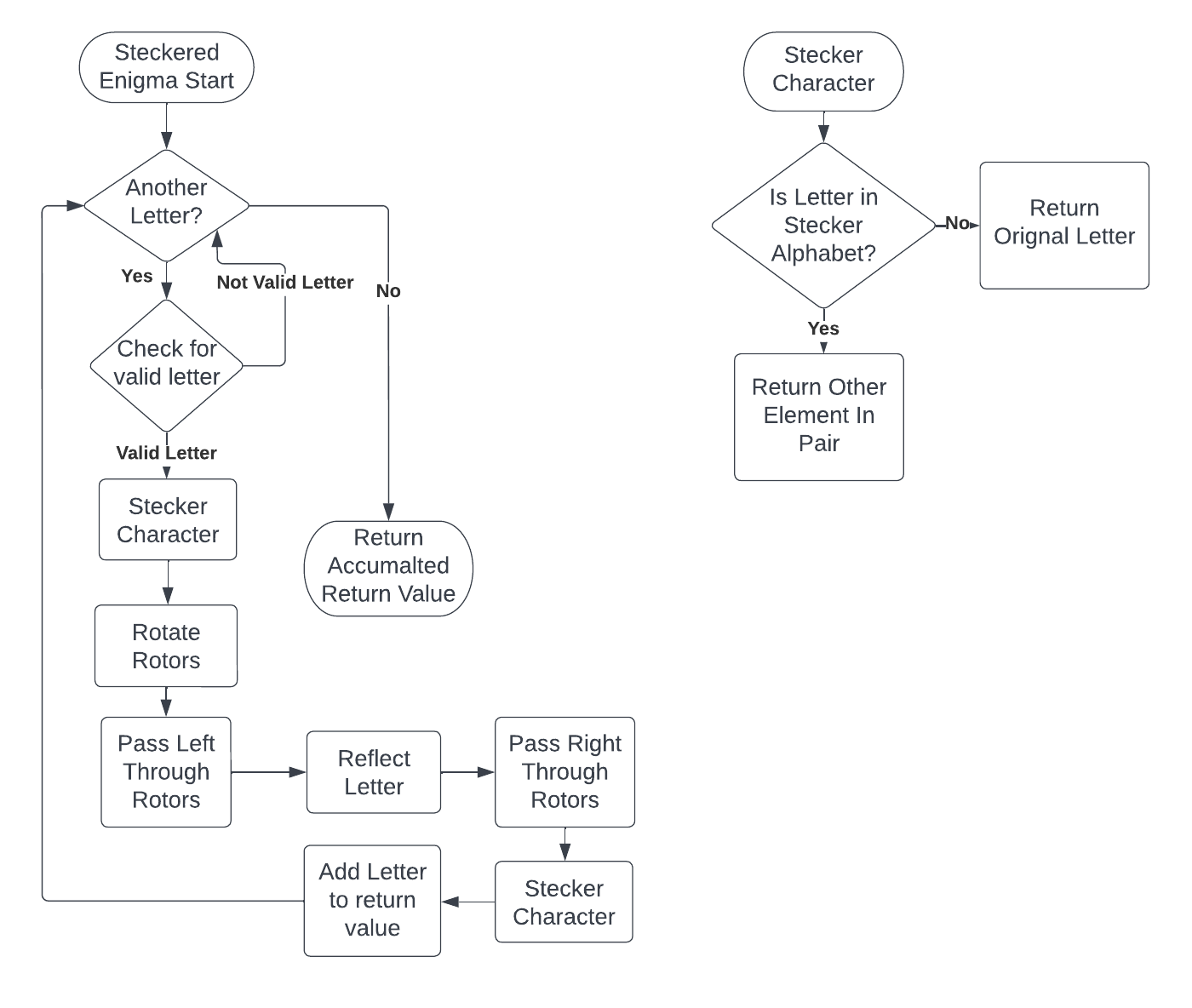


Figure 3 - Steckered Enigma Operation

# Enigma Machine Testing

The first functions that were implemented were the ones that had the least dependencies on other functions. The table below contains the first implemented functions and their tests.

#### Reflector Function

Named “reflectorFunction” in the implementation, it simply takes in a character and list of reflector pairs to find the input character’s paired letter.

|  |  |  |  |
| --- | --- | --- | --- |
| Test Input | Rational Behind Values | Expected | Output |
| ‘A’, reflectorB | Test for values input on the left of the pair are reflected | ‘Y’ | ‘Y’ |
| ‘B’, reflectorB | Test for element other than first | ‘R’ | ‘R’ |
| ‘W’, reflectorB | Test for values input on the right of the pair reflected. | ‘V’ | ‘V’ |

#### Rotate Rotors

Named “rotateRotors” in the implementation. It takes in three separate rotors and an array of their current offsets. If any rotor’s next position is a knock-on position the next rotor in the series will turn. If any rotor’s increment hits 26 it will return to position 0 (rotor has returned to starting position).

|  |  |  |  |
| --- | --- | --- | --- |
| Test Input | Rational Behind Values | Expected | Output |
| rotor1, rotor2, rotor3, [0,0,0] | First shift for starting input | [0,0,1] | [0,0,1] |
| rotor1, rotor2, rotor3, [0,1,25] | Checks for return to starting value | [0,1,0] | [0,1,0] |
| rotor1, rotor2, rotor3 [0,0,21] | Checks for knock on position for third rotor | [0,1,22] | [0,1,22] |
| rotor1, rotor2, rotor3 [0,4,21] | Checks for dual turning of rotors | [1,5,22] | [1,5,22] |
| rotor1, rotor2, rotor3 [16,4,21] | Ensures first rotor has no impact on other rotors | [17,5,22] | [17,5,22] |

#### Integer to Letter Conversion

|  |  |  |  |
| --- | --- | --- | --- |
| Test Input | Rational Behind Values | Expected | Output |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Test Input | Rational Behind Values | Expected | Output |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |