# Name & Student ID

Tristen Ward

20356578

# Title

Mechatronics Robot Writing Project

# Software Description

The scope of this project is to develop the software which exports G-Code to a writing robot from originally stored data in a text file. The communication device used to deliver this data is through a virtual RS232 serial port. This will be connected from a computer to the robot, allowing for the program to convert the data into the correct format, size and font. The size of the font is determined by user input but must be between 4 and 10mm.

The font data coordinates are imported from a text file into the program which provides the needed coordinates and commands. This data is stored in structs which are able to be called further in the program to use. These are key for locating which letter is assigned to which function of the font data. The font data then determines what positions are required by the robot to write the selected letters and words. This font data comes in the format of coordinates which are designated to each Ascii unit. Each Ascii unit will have a set number of strokes it will execute. Before the letters have the assigned commands, the defining user input font size is used to calculate the actual size to print. Derived from the font units of 18, the movements of the letters will be scaled to correspond to the user input size. This calculation is executed through user inputted size/18. In the program, this will alter the current coordinates stored and G-Code exported to the writing robot to achieve the correct sizing. The data which is used for the G-Code is itself also stored in a struct in the program to then be written in a G-Code format to allow the robot to read.

Another critical problem is the line space available for writing, only covers 100 units per line on the X axis. Without acting on this occurring problem, the robot would not know when to start a new line and continue to write off the page. Hence, only a certain amount of the total words will be physically recorded. Words must be fully written on the line and to not be cut off halfway through said word. To tackle this problem a record of the word coordinates are stored and constant checks are implemented to overcome the problem. However, this leads to another similar problem with the Y axis: a similar solution is applied to make sure the words stay on the paper. Other parameters which also are vital for the display of letters, include the correct coordinate positioning when the program starts. This variable must be set at 0 at the beginning of the program.

To visually display the robot has finished its writing of the words, the pen should then end up back at the beginning of the coordinates 0,0. Where the robot writer continues after each letter is crucial to store the coordinate data, as left without will see the robot writing over previous letters. Knowing when the pen must be in a pen-down position and when it is needed in the pen-up position, is important to be included in the program. This can be simply extracted from the font data values as 1, the pen is down, or 0, the pen is up.

Knowing when to stop reading the text file could cause problems for the robot program. To determine where the program stops reading the text data file, the program will check if there are more than two spaces in a row. This allows the program to acknowledge when the text file is completed, allowing to disconnect from the robot and close the data files safely. It is essential that the files are closed after use to allow them to be opened in the future. It is equally important the text files are opened with read rather than write as read will not edit or overwrite the present data.

# Project Files

(Maximum 1 page)

# Key Data Items

|  |  |  |
| --- | --- | --- |
| Name | Data type | Rationale |
| pArray | Int Struct | This Struct stores coordinates from the font data file. This data is stored as Int as only numbers are used. |
| pStrokes | Float Struct | This Struct stores edited coordinates to conform with the offsets and the font size fraction. |
| CharacterFraction | Float | Stored user input for size of characters/18. |
| WordBuffer | Char | Stores word from text file as char. |
| XCoord | Int | Stores the accumulation of maximum coordinate length of each letter. |
| YCoord | Int | Stores the accumulation of maximum coordinate height of each letter. |
| Buffer | Float | This temporally stores the G-Code data to be exported to the robot. |
| StrokeBuffer | Int | Counts how many strokes are used per word. |

# Functions

void LettersToAscii(struct point \*pArray, struct Strokes \*pStrokes, float CharacterFraction, char WordBuffer[10], float \*XCoord, float \*YCoord, char \*buffer);

Parameters:

pArray – Struct which stores the coordinates for each letter in a word.

pStrokes – Struct which stores the edited (applied Character Fraction and the X offsets) coordinates for each letter.

CharacterFraction – input of the user defined letter size as a fraction.

WordBuffer – Stores input of the Word

XCoord – used to store the max X Coordinates for each letter so they don’t overlap.

YCoord - used to store the max Y Coordinates for the whole text file so it doesn’t run out of space.

Buffer – variable buffer to send to the SendCommands function.

void FontCodeToGCode(struct Strokes \*pStrokes, char \*buffer, int StrokeBuffer)

Parameters:

pStrokes – Struct which stores the edited (applied Character Fraction and the X offsets) coordinates for each letter.

Buffer – variable buffer to send to the SendCommands function.

StrokeBuffer – Used to track how many strokes each word contained, then to convert all pStrokes lines to G-Code

# Testing Information

|  |  |  |  |
| --- | --- | --- | --- |
| Function | Test Case | Test Data | Expected Output |
| LettersToAscii | Correct | WordBuffer = “H” | Output in Struct pStrokes will look like (without size factor) =  0 0 0  0 18 1  12 0 0  12 18 1  0 9 0  12 9 1  18 0 0 |
| LettersToAscii | Incorrect | WordBuffer = “” | Program will terminate. |
| FontCodeToGCode | Correct | Struct pStrokes =  0 0 0  0 18 1  12 0 0  12 18 1  0 9 0  12 9 1  18 0 0 | Buffer =  G1 0 0  S1000  G1 X0 Y18  S0  G0 X12 Y0  S1000  G1 X12 Y18  S0  G0 X0 Y9  S1000  G1 X12 Y9  S0  G0 X18 Y0 |
| FontCodeToGCode | Incorrect | Struct pStrokes = NULL | Program will terminate. |
| Main | Height check – positive | Anything between 4 and 10 | Will set scale to the user input. |
| Main | Height check - negative | Anything outside 4 and 10 | Return “Character size of %d is not possible, please try again  .” |
| Main | Text File correct | \* Correct file format \* “Hello World” | Data from file can be read. Robot writes “Hello World”. |
| Main | Text file incorrect | \* Incorrect file format \* | Program will terminate. |
| Main | Font File path correct | \* Correct file path \* | Data from font file can be read. |
| Main | Font file path incorrect | \* Incorrect file path \* | Return “The file has not been found. Please try again.” |