Robot Writer Software System Manual

Chengyu ZHANG (20124992)

Software Description

This software is aimed to generate a G-code to the writing robot for writing the text which is read from a file. The G-code will be executed by the writing robot and draw the text on the paper.

The text which needs to be ‘draw out’ is input to the software by saving as a file. The output G-code will be sent to the control unit of the writing robot (Arduino in this case) using a virtual RS232 serial port.

In this software, some font data and open-source libraries will be used, for example, an additional font file contains the pre-defined G code for each character will be required. The RS232 serial port communication will be handled by a RS232 library.

For the serial port communication, before sending more G-code, the software will wait until receiving the acknowledge signal from the Arduino.

The software interface with user will be a console window. The communication code and possible errors during the process will be displayed on this console window.

Project Files

File Structure

|- src	source code and build path, code files are directly in this folder
|- include	the head files which are included
|- build	store the object files after compile
|- asset	
|- font	store the static font g-code files
|- input	the relative path for input.txt
|- bin	the final output file for exactable program
|- docs	store the necessary documents

Head Files

utility.h	some marcos to control the debug/release mode
main.h	head file for main, defined the basic functions
rs232.h	head file provided by rs232 library
serial.h	head file provided as a packed interface for rs232

Code Files

main.c	main file code, execute the necessary functions
serial.c	some functions which packed for rs232 serial connection
rs232.c	code file provided by rs232 library

Possible changes needed during further development: Some functions for load static files can be encapsulated into some functions.

Pack for release: pack the bin, asset and input folders after the compile.

Function Declarations

int CanRS232PortBeOpened()

Function: initial the rs232 library

Return value: always 0.

int WaitForWakeUp(void);

Function: pause and wait for the first reply

Return value: always 0.

int WaitForReply(void);

Function: pause the g-code output to wait for the receiving command from serial port

Return value: always 0.

int convertCharArrayToInt(char numarray[], int *startPosition, int charLength, int *returnValue)

Function: convert the char in a char array to integer

Parameters:

char numarray[]: file pointer of the font g code file;

int *startPosition: from which point of the string to convert the int number

int charLength: the max length of this conversion, must larger than length+1

int *returnValue: return the integer after this conversion

Return value: 0 for failed and 1 for success.

returnValue will handle the result as a pointer

int generateFontIndex(FILE *filePointer, struct FontIndex fontGcodeLineIndex[]);

Function: generate an index to make g-code generation much easier

Parameters:

filePointer: file pointer of the font g-code file.

fontGcodeLineIndex[]: an array stores the start line and end line number of each character G-code;

Return value: 1 for success, 0 for failed.

Index will be handle and returned by the Pointer fontGcodeLineIndex.

int initializeWritingMachine();

Function: send the initialize g-code for writing robots.

Return value: 1 for success, 0 for failed

int createFontDataCache(FILE *filePointer, int fontGcodeData[]);

Function: convert the plain text g code to a more executable data saved in memory

Parameters:

filePointer: file pointer of the font g-code file.

fontGcodeData[]: the pointer point to the pre allocated memory for the storage;

Return value: 1 for success, 0 for failed.

Data after convert will be saved to the memory block allocated before and pointed by the int pointer fontGcodeData.

int generateCharGcodeCommand(int charAsciiNum, double *selectedOffsetX, double *selectedOffsetY, char commandBuffer[], int fontDataCache[], struct FontIndex fontIndexArray[], double Scaler);

Function: use the offset and scaler to adjust the font g-code for writing robot to execute. This function will be called for each character.

Parameters:

charAsciiNum: the ASCII number of current character need to be printed;

*selectedOffsetX: the offset position X for this character;

*selectedOffsetY: the offset position Y for this character;

commandBuffer[]: the commandBuffer of serial port, which will be send;

fontDataCache[]: the pointer for the memory block which stores the font g-code cache;

fontIndexArray[]: an array stores the start line and end line number of each character G-code;

Scaler: the global scaler for this writing task;

Return Value: 1 for success and 0 for failed.

G-code command will be saved to command buffer and send through serial port.

int updateCharactorOffsetPosition(double *selectedOffsetX, double *selectedOffsetY, double commandWidthChange, double commandHeightChange, double globalScaler);

Function: change the offset for next character.

Parameters:

*selectedOffsetX: the offset position X needs to update;

*selectedOffsetY: the offset position Y needs to update;

commandWidthChange: required move on x axis for next character’s offset;

commandHeightChange: required move on y axis for next character’s offset;

globalScaler: the scale set for the offset change.

Return value: 1 for success, 0 for failed.

The offset after updated will be returned through the change on the variables pointer point to.

Key Data Items

Name	Data type	Rationale
fpFont	FILE *	File pointer for font data
fontIndexArray[128]	FontIndex Self-defined struct	An array with size of 128, start line for each character is .start_line, the length of g-code is .line_num. both are int, because all of these are line numbers and will normally in the range of the int
memForFontData	int *	Point to a block of memory assigned after the index function sort out the size required to store the g-code data for each character. All position data in font file are integers, but the file size is not determined, thus the dynamic allocation of memory will be more suitable
fpText	FILE *	File pointer for text file
outputOffsetX	double	The x offset of current character. Because a scaler will be multiplied, thus double is quired
outputOffsetY	double	The y offset of current character. Because a scaler will be multiplied, thus double is quired
generalScaler	double	A variable to determine size scale of the output text. This needs to be a double because the decimal might be used
charReadyToWrite	char	The next char read out from the input text file, which will be ready to print by sending into a function. The char is used because it use much smaller memory space, and can be easily convert to int type.

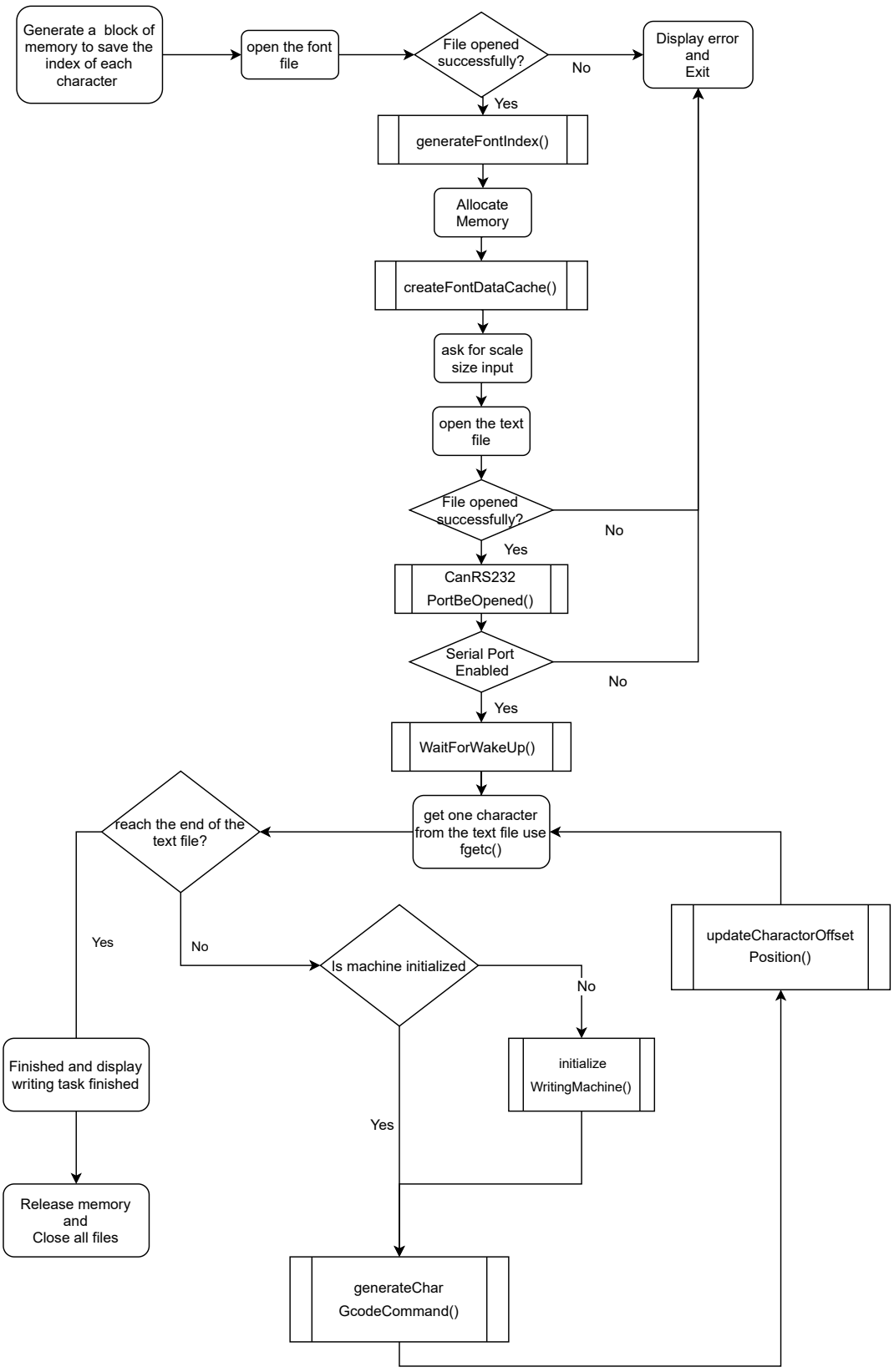
Test Information

Function	Test Case	Test Data	Expected Output
convertCharArrayToInt	Test the convert of a normal number string	numarray[] = “999 10 1”; startPosition = 0; charLength = 253; returnValue = 0;	returnValue send in with value of 999, startPosition send in with value of 3, the function returns 1;
convertCharArrayToInt	Test the convert of a non-number string	numarray[] = “As 99 00”; startPosition = 0; charLength = 253; returnValue = 0;	returnValue send in with value of 0, startPosition send in with value of 0, the function returns 0;
convertCharArrayToInt	Test the convert of a number string start with negative number	numarray[] = “-3 10 1”; startPosition = 0; charLength = 253; returnValue = 0;	returnValue send in with value of -3, startPosition send in with value of 2, the function returns 1;
convertCharArrayToInt	Test the convert of the number string ‘s second number	numarray[] = “-3 10 1”; startPosition =3; charLength = 253; returnValue = 0;	returnValue send in with value of 10, startPosition send in with value of 5, the function returns 1;
generateFontIndex	Load a correct font data, with a line number of 1900.	Font data file path, An empty array.	The function returns 1900, With all the font index loaded, this can be checked by a looped printf function
generateFontIndex	Load a false font data with error at line 150, which is “999 300 0”	Font data file path, empty array.	The function returns -150, with the index before 150 lines loaded.
CreatFontDataCache	Load a correct font data into cache, with line number of 1900	Font data file path, memory block allocated.	The function returns with 1900 and the memory data can be check manually
CreatFontDataCache	Load a font data with format error into cache, with	Font data file path, memory block allocated.	The function returns with -1 and the memory

	error in line 320, which is “-5 ERR 20”.		data before sizeof(int)*319 is set, console output of “Format Error Found in Line 320”
updateCharactorOffsetPosition	Move one default width. Also test if the scaler working	tempOffsetX = 0, tempOffsetY = 0, commandWidthChange = 16, commandHeightChange = 0, globalScaler = 0.4 Macro _MAX_LINE_WIDTH_ is 150;	tempOffsetX = 6.4, tempOffsetY = 0,
updateCharactorOffsetPosition	Send command with height change. Also test if the scaler working	tempOffsetX = 0, tempOffsetY = 0, commandWidthChange = 0, commandHeightChange = 30, globalScaler = 0.6 Macro _MAX_LINE_WIDTH_ is 150;	tempOffsetX = 0, tempOffsetY = 18,
updateCharactorOffsetPosition	Move one width which exceed the max line width limit	tempOffsetX = 146, tempOffsetY = 0, commandWidthChange = 16, commandHeightChange = 0, globalScaler = 1 Macro _MAX_LINE_WIDTH_ is 150; _LINE_HEIGHT_OFFSET_ is 30	tempOffsetX = 0, tempOffsetY = -30,
generateCharGcodeCommand	Check if it is working with random character, With a scaler applied	tempOffsetX and tempOffsetY both 0, FontCache and FontArray loaded using pervious fully tested function, Input char wich is ‘(int)H’, Scaler is 0.4	With a display of the g code on console output, which is correct as the one inside example and with a 0.4 large size scaled. Offset is set to (6.4, 0)
generateCharGcodeCommand	Check if it is working with no output ASCII number, With a scaler applied	tempOffsetX and tempOffsetY both 0, FontCache and FontArray loaded, Input char wich is ‘3’, Scaler is 0.4	Only gcode output is the move to offset, the offset is then set to (6.4, 0)

Flowcharts

Main Function Process



Flowcharts

generateCharGcodeCommand Function Process