Computer Technology I

Lab. 6 : CyberTech Wall Display

Author: ANAS KWEFATI
Supervisor: ANDERS HAGGREN
Semester: Autumn 2019
Area: Computer Science
Course code: 1DT301

Contents

1 Task 1

```
/*>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
; 1DT301, Computer Technology I
; Date: 2016-09-15
; Author:
;     Anas Kwefati
;
; Lab number: 6
; Title: CyberTech Wall Display
;
; Hardware: STK600, CPU ATmega2560
;
; Function: Program that writes a character on the CyberTech Display.
;
; Input ports: none
;
; Output ports: CyberTech Display.
;
; Subroutines:
; Included files: <avr/io.h>
;
;Other information: Display is connected to the serial port (RS232) on the STK600.
; Communication speed is 2400bps.
;Changes in program: (Description and date)
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<*/

#include <avr/io.h>
#include <stdio.h>
#include <string.h>
//#include <util/delay.h>
#define FCPU 1000000// Clock Speed
#define BAUD 2400 //Communication Speed Display rate 2400
#define MYUBBRR (FCPU/16/BAUD-1) //UBBRR = 25 -> osc = 1MHz and UBRR = 47 -> osc = 1,843200MHz

void uart_int(void);
void toPutty(unsigned char data);

int main(void)
{
    uart_int();

    char* txt = "\rAO0001Hi How are you ? :)";
    int checksum =0;
    //We make sure that everything is in it
    for(int i =0; i<strlen(txt);i++){
        checksum += txt[i];
    }

    checksum\%=256;

    char toDisplay [strlen(txt)+3];
    sprintf(toDisplay, "\%s\%02X\n", txt, checksum); //\%02x means print at least 2 digits, prepends it with 0's if there's less.
    //\%02x is used to convert one character to a hexadecimal string

    for (int i = 0; i<strlen(txt)+3;i++){
```

```
55          toPutty(toDisplay[i]);
56      }
57
58      txt = "\rZD0013C\n";
59      for(int i = 0; i<strlen(txt);i++){
60          toPutty(txt[i]);
61      }
62
63      return 0;
64  }
65
66  //INITALIZATION OF THE DISPLAY
67
68  void toPutty(unsigned char data){
69      //WAIT FOR DATA TO BE RECEIVED
70      while(!(UCSR1A & (1<<UDRE1)));
71      UDR1 = data;
72  }
73
74  void uart_int(void) {
75      UBRR1L = MYUBBRR; //25 because we are setting the board at 1MHz
76      /*Enable receiver and transmitter*/
77      UCSR1B = (1<<RXEN1|1<<TXEN1); // Receive Enable (RXEN) bit //
        Transmit Enable (TXEN) bit
78  }
```

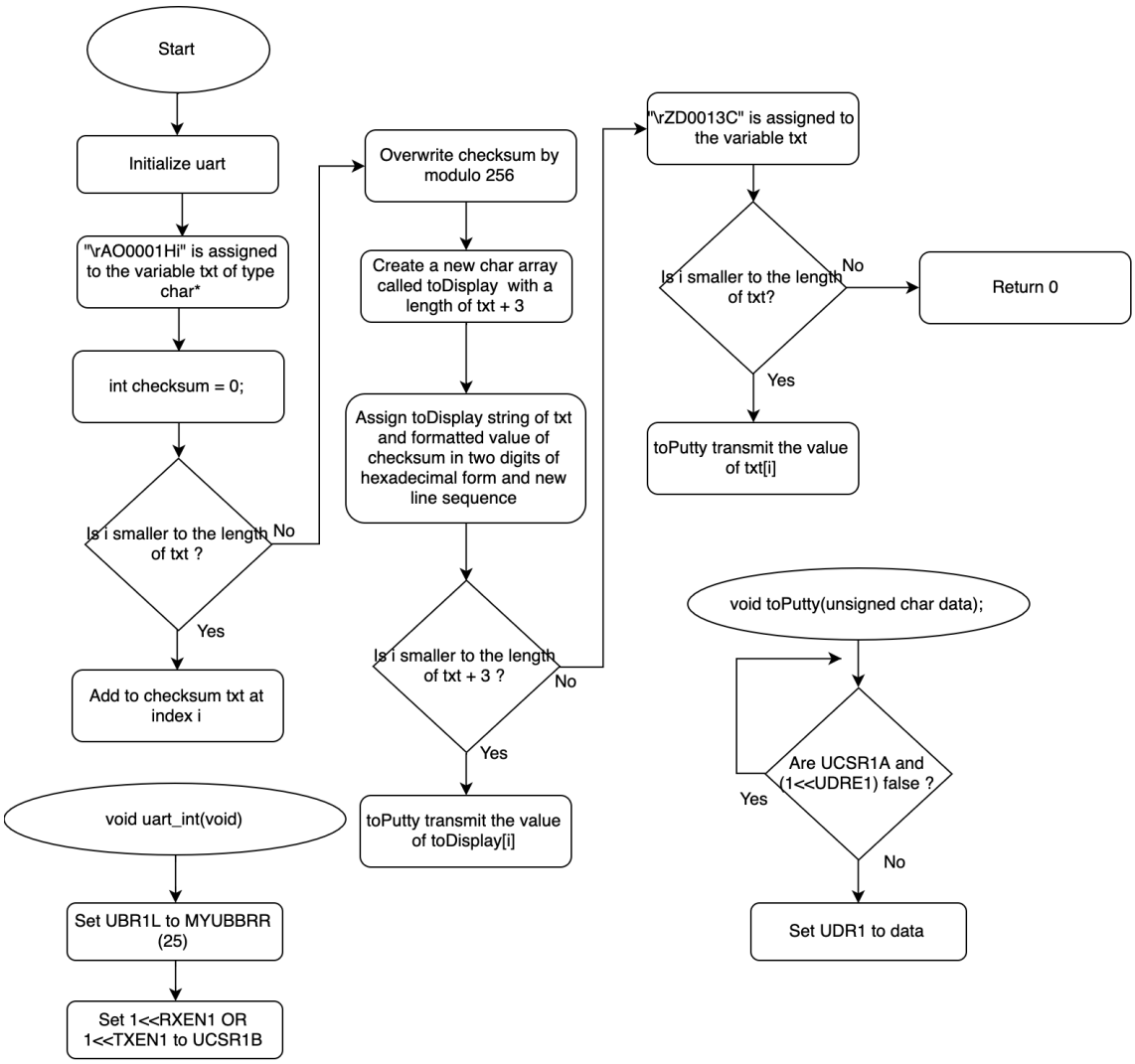

Figure 1: Task 1 flowchart

2 Task 2

In this task we used the same methods for uart_int(void) and toPutty(unsigned char data).

```
/*>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
; 1DT301, Computer Technology I
; Date: 2016-09-15
; Author:
;     Anas Kwefati
;
; Lab number: 6
; Title: CyberTech Wall Display
;
; Hardware: STK600, CPU ATmega2560
;
; Function: Program that writes characters on all text lines on the
  CyberTech Display.
; The program will write to all 3 rows.
;
; Input ports: none
;
```

```
; Output ports: CyberTech Display.
;
; Subroutines:
; Included files: <avr/io.h>
;
;Other information: Display is connected to the serial port (RS232) on the STK600.
; Communication speed is 2400bps.
;Changes in program: (Description and date)
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<*/

#include <avr/io.h>
#include <stdio.h>
#include <string.h>
//#include <util/delay.h>
#define FCPU 1000000// Clock Speed
#define BAUD 2400 //Communication Speed Display rate 2400
#define MYUBBRR (FCPU/16/BAUD-1) //UBBRR = 25 -> osc = 1MHz and UBRR = 47 -> osc = 1,843200MHz

void uart_int(void);
void toPutty(unsigned char data);
void toDisplayOnLCD(char* stringChar);

int main(void)
{
  uart_int();

  char* txt = "\rAO0001First Line              Second Line";

  toDisplayOnLCD(txt);



  txt = "\rBO0001Third Line";
  toDisplayOnLCD(txt);

  txt = "\rZD0013C\n";
  toDisplayOnLCD(txt);

  return 0;
}


//METHOD TO DISPLAY ON THE SCREEN
void toDisplayOnLCD(char* stringChar){

  int checksum = 0;
   //We make sure that everything is in it
   for(int i =0; i<strlen(stringChar);i++){
     checksum += stringChar[i];
   }

   checksum\%=256;

   char toDisplay [strlen(stringChar)+3];
   sprintf(toDisplay, "\%s\%02X\n", stringChar, checksum); //\%02x means print at least 2 digits, prepends it with 0's if there's less.
```

```
72     //\%02x is used to convert one character to a hexadecimal string
73
74   for (int i = 0; i<strlen(stringChar)+3;i++){
75     toPutty(toDisplay[i]);
76   }
77 }
78
79 //INITIALIZATION OF THE DISPLAY
80
81 void toPutty(unsigned char data){
82   //WAIT FOR DATA TO BE RECEIVED
83   while(!(UCSR1A & (1<<UDRE1)));
84   UDR1 = data;
85 }
86
87 void uart_int(void) {
88   UBRR1L = MYUBBRR; //25 because we are setting the board at 1MHz
89   /*Enable receiver and transmitter*/
90   UCSR1B = (1<<RXEN1|1<<TXEN1); // Receive Enable (RXEN) bit //
       Transmit Enable (TXEN) bit
91 }
```

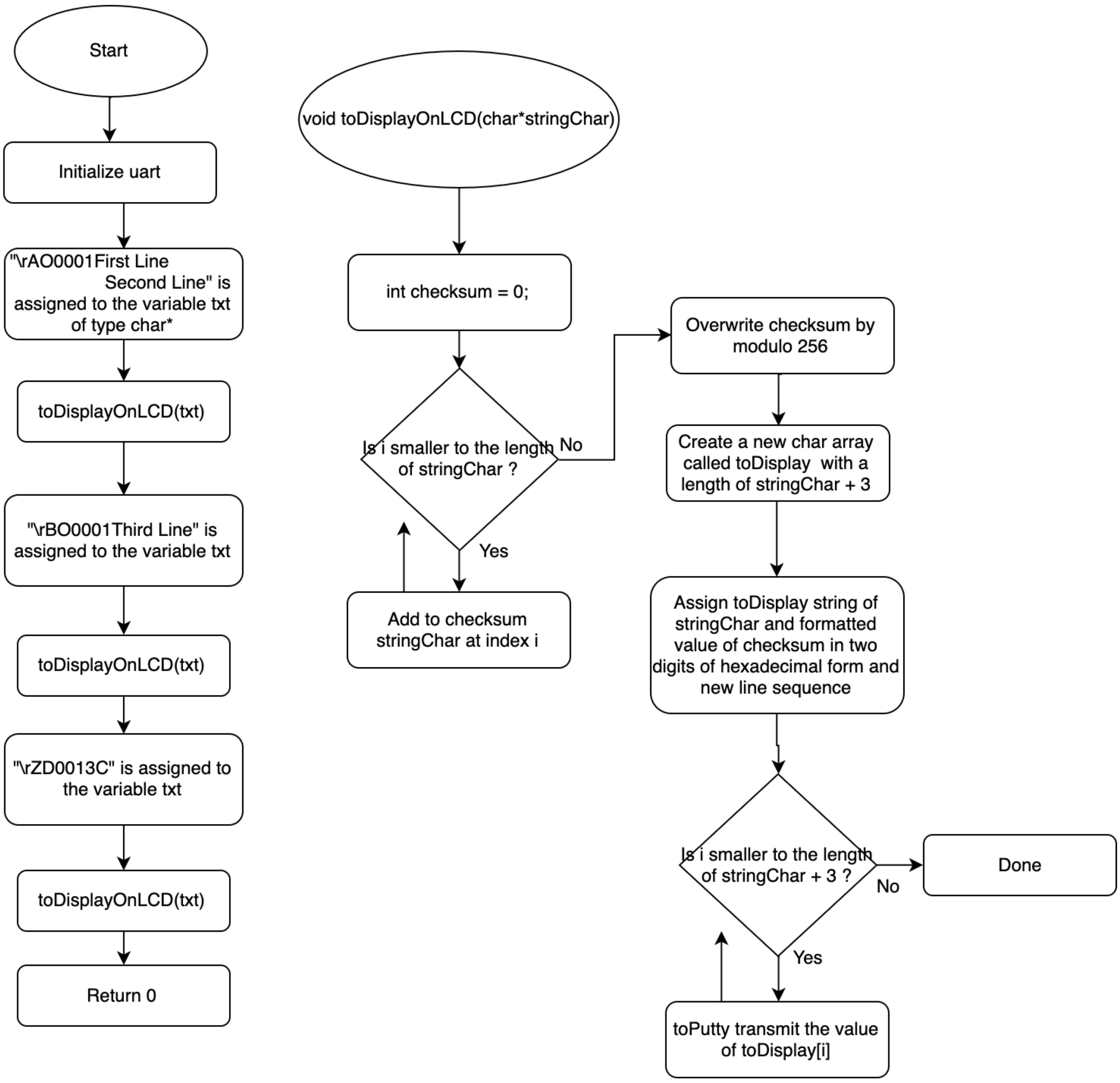

Figure 2: Task 2 flowchart

3 Task 3

In this task we used the same methods for void uart_int(void), void toPutty(unsigned char data) and void toDisplayOnLCD(char* stringChar).

```
/*>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
; 1DT301, Computer Technology I
; Date: 2016-09-15
; Author:
;     Anas Kwefati
;
; Lab number: 6
; Title: CyberTech Wall Display
;
; Hardware: STK600, CPU ATmega2560
;
; Function: Program that changes text strings on the display.
;
; Input ports: none
;
; Output ports: CyberTech Display.
```

```
;
; Subroutines:
; Included files: <avr/io.h> and <util/delay.h>
;
;Other information: Display is connected to the serial port (RS232) on
    the STK600.
; Communication speed is 2400bps.
;Changes in program: (Description and date)
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<*/
#include <avr/io.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

#define F_CPU 1000000// Clock Speed
#include <util/delay.h>
#define BAUD 2400 //Communication Speed Display rate 2400
#define MYUBBRR (F_CPU/16/BAUD-1) //UBBRR = 25 -> osc = 1MHz and UBRR =
    47 -> osc = 1,843200MHz

void uart_int(void);
void toPutty(unsigned char data);
void toDisplayOnLCD(char* stringChar);

int main(void)
{
  uart_int();


  char* data = "abc";
  char *txt = "\rAO0001";

  for(int i =0;i<strlen(data);i++){
    //The idea is to take char by char and add it one by one to str2
    char c = data[i];
    size_t len = strlen(txt); //take the length of txt
    char *str2 = malloc(len + 1 + 1); //give a length of len and
    allocate a bit more memory with malloc in case
    strcpy(str2, txt); // copy txt to str2
    str2[len] = c;  //create an array of str2 with a length of len for
    the char c
    str2[len + 1] = '\0'; // we add 1 to len and add the end char \0
    toDisplayOnLCD(str2); //call display
    free(str2); //free str2 deallocate the space used by malloc()

    str2 = "\rZD0013C";
    toDisplayOnLCD(str2);
    _delay_ms(5000); //wait 5s
  }


  return 0;
}



//METHOD TO DISPLAY ON THE SCREEN
void toDisplayOnLCD(char* stringChar){

```

```
  int checksum = 0;
   //We make sure that everything is in it
   for(int i =0; i<strlen(stringChar);i++){
     checksum += stringChar[i];
   }

   checksum\%=256;

   char toDisplay [strlen(stringChar)+3];
   sprintf(toDisplay, "\%s\%02X\n", stringChar, checksum); //\%02x
   means print at least 2 digits, prepends it with 0's if there's less
   .
   //\%02x is used to convert one character to a hexadecimal string

  for (int i = 0; i<strlen(stringChar)+3;i++){
    toPutty(toDisplay[i]);
  }
}

//INITIALIZATION OF THE DISPLAY

void toPutty(unsigned char data){
  //WAIT FOR DATA TO BE RECEIVED
  while(!(UCSR1A & (1<<UDRE1)));
  UDR1 = data;
}

void uart_int(void) {
  UBRR1L = MYUBBRR; //25 because we are setting the board at 1MHz
  /*Enable receiver and transmitter*/
  UCSR1B = (1<<RXEN1|1<<TXEN1); // Receive Enable (RXEN) bit //
    Transmit Enable (TXEN) bit
}
```

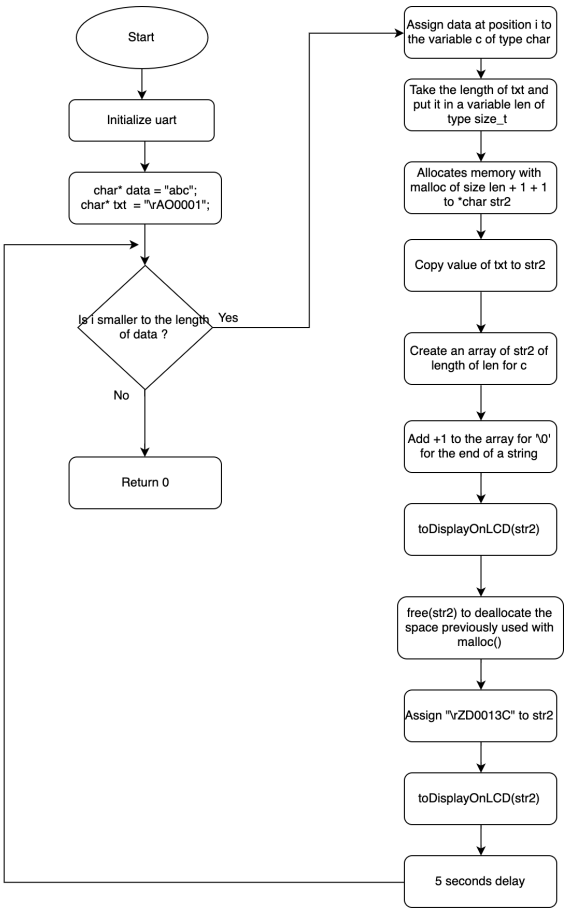

Figure 3: Task 3 flowchart

4 Task 4

```
/*>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
; 1DT301, Computer Technology I
; Date: 2016-09-15
; Author:
;     Anas Kwefati
;
; Lab number: 6
; Title: CyberTech Wall Display
;
; Hardware: STK600, CPU ATmega2560
;
; Function: Program that communicates with the terminal and the display
    .
; It is also possible to enter the address on the screen to display
    text.
;
; Input ports: Serial Port that is connected to the terminal (PuTTY)
    takes input from keyboard.
;
; Output ports: CyberTech Display.
;
; Subroutines:
; Included files: <avr/io.h> and <util/delay.h>
;
;Other information: Display is connected to the serial port (RS232) on
```

```
   the STK600.
; Communication speed is 2400bps.
; TASK 5 is the same as TASK 4 but with more address to choose (1-9)
;Changes in program: (Description and date)
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<*/
#include <avr/io.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>

#define F_CPU 1000000// Clock Speed
#include <util/delay.h>
#define BAUD 2400 //Communication Speed Display rate 2400
#define MYUBBRR (F_CPU/16/BAUD-1) //UBBRR = 25 -> osc = 1MHz and UBRR =
    47 -> osc = 1,843200MHz
#define MAX_LINES 3 // Possible lines TASK4
#define POSSIBLE_DIGIT_TO_CHOOSE_LINE "123" //TASK4

//#define POSSIBLE_DIGIT_TO_CHOOSE_LINE "123456789" //TASK 5
//#define MAX_LINES 9 //TASK 5 and uncomment the 2 other for TASK 4

void uart_int(void);
void toPutty(unsigned char data);
char getChar();

void improvedtoDisplayOnLCD(char address, char* specialCommand, char*
   stringChar);
void endToDisplayOnLCD();
void setUpToDisplayOnLCD();

void changeLine (int targetLine);
char possibleCharacter(char* possibleLine, char c);

//SOME GLOBAL VARIABLES
int currentLine = 0;
bool selectionOfLine = false;
char lines[8][24] = { "", "", "", "", "", "", "", "" };//8ROWS 24
   COLUMNS


int main(void)
{
  uart_int();
  setUpToDisplayOnLCD();

  while (1) {
    char character = getChar(); // We get the input from terminal

        //IF SELECT A LINE WE WAIT FOR A VALID DIGIT that is >= 1
    if (selectionOfLine == true) {
            if (character < '1') {
                continue; // If character <1 do nothing
            }
      // We check if the digit that the user has put is in what is
   possible
      if (possibleCharacter(POSSIBLE_DIGIT_TO_CHOOSE_LINE, character))
   {
                int convertCharToInt = character - '0'; //convert char
```

```
    to int
                /*LOGIC behind the conversion :
                 char c = '2';
                 int i = c - '0';
                 it will take the ASCII value of charactter 2 (50) and
    0 (48)
                 Then it will substract them -> int i = 50 - 48 which
    gives us 2
                 */
        changeLine(convertCharToInt); //call the method to choose the
    possible line. Change currentLine by the targetLine (here "
    convertCharToInt")

                selectionOfLine = false; //False
      }
    }
    else {

            if (character == '>'){

                selectionOfLine = true; //Make selectionOfLine true

            } else if (character == 13 ){
                //Else if character == 'enter'
                changeLine(-1); // We increment currentLine and we go
    to the next line

            } else {
        // Else add character to end of selected line
        char* line = lines[currentLine];
        sprintf(line, "\%s\%c", line, character);
      }
    }
    setUpToDisplayOnLCD(); // Update the screen
  }

  return 0;
}


//INITALIZATION OF THE DISPLAY

void toPutty(unsigned char data){
  //WAIT FOR DATA TO BE RECEIVED and SHOW IT
  while(!(UCSR1A & (1<<UDRE1)));
  UDR1 = data;
}

void uart_int(void) {
  UBRR1L = MYUBBRR; //25 because we are setting the board at 1MHz
  /*Enable receiver and transmitter*/
  UCSR1B = (1<<RXEN1|1<<TXEN1); // Receive Enable (RXEN) bit //
    Transmit Enable (TXEN) bit
}

char getChar(){
  //WAIT FOR THE CHARACTER TO BE RECEIVED THEN RETURN IT
  while(!(UCSR1A & (1<<RXC1)));
  return UDR1;
```

```
}


//METHOD TO DISPLAY ON THE SCREEN
void improvedtoDisplayOnLCD(char addressLine, char* specialCommand,
    char* stringChar){

  //we get the length of special command which is "O0001"
    //and we get the length of the specific message in stringChar
  int specialCommandLen = sizeof(specialCommand);
  int stringCharLen = sizeof(stringChar);

  char* toDisplay = malloc(1 + specialCommandLen + stringCharLen + 3);
    //give a length of specialCommand and stringCharLen and allocate a
    bit more memory with malloc for the end case. We allocate memory
    for toDisplay

  // ADD EVERYTHING TOGETHER in toDiplsay
    //So we get \rADDRESSLINE_SPECIALCOMMAND_STRINGCHAR
    //For example \rAO0001a"
  sprintf(toDisplay, "\r\%c\%s\%s", addressLine, specialCommand,
    stringChar);


    int checksum = 0;
    // We calculate checksum to make sure that everything is in it
  for (int i = 0; (toDisplay[i] != 0); i++){
    checksum += toDisplay[i];
  }

  checksum \%= 256;

  sprintf(toDisplay, "\%s\%02X\n", toDisplay, checksum);//\%02x means
    print at least 2 digits, prepends it with 0's if there's less.
    //\%02x is used to convert one character to a hexadecimal string

  for (int i = 0; (toDisplay[i] != 0); i++){
    toPutty(toDisplay[i]);
  }


  free(toDisplay); // free toDisplay deallocate the space used by
    malloc()
}

void endToDisplayOnLCD(){
    char* txt = "\rZD0013C\n";
    for(int i = 0; i<strlen(txt);i++){
        toPutty(txt[i]);
    }
}



//METHOD TO SETUP CHARACTERS FOR EACH LINE
void setUpToDisplayOnLCD(){

  //Take currentLine and see if it is <1 if yes then increment
    lineToDisplay
```

```
179   int lineToDisplay = currentLine;
180
181     if (lineToDisplay < 1){
182         lineToDisplay++;
183     }
184
185
186   //Set up for first and second rows
187   char memory_space_A[48] = "";
188
189   //char line_selected = selectionOfLine ? '_' : (currentLine + '0');
190
191     //If ChooseALine is True then add '_' to choosenLine otherwise the
      currentLine
192
193     char choosenLine = "";
194     if(selectionOfLine == true){
195         choosenLine = '_';
196     } else {
197         choosenLine = currentLine + '0';
198     }
199
200     //Add everything in the array of char.
201   sprintf(memory_space_A, "Choose input: \%c          \%s", choosenLine,
      lines[lineToDisplay-1]);
202
203   // Set up for third row
204   char memory_space_B[48] = " ";
205     if (lines[lineToDisplay][0] == true){
206         continue; //check if the selected like is '\0', if yes then do
      nothing
207
208     }
209
210     for (int i = 0; i < 48; i++){
211         //Send data from third row to the memory
212         //memory_space_B[i] = lines[1][i]
213         memory_space_B[i] = lines[lineToDisplay][i];
214     }
215
216   // Send everything to improvedtoDisplayOnLCD to do the calculation
      with checksum then it will send it to the screen
217   improvedtoDisplayOnLCD('A', "O0001", memory_space_A);
218   improvedtoDisplayOnLCD('B', "O0001", memory_space_B);
219     endToDisplayOnLCD();
220 }
221
222
223
224
225 //METHOD TO CHECK IF THE USER INPUT IS ALLOWED TO CHOOSE LINE
226 char possibleCharacter(char* possibleLine, char c){
227     char tempChar;
228
229     //While will continue until the character is \0
230     while (*possibleLine){
231         tempChar = *possibleLine; //We take char by char from
      possibleLine and we add it to the char t
232
```

```
233         if (tempChar == c) {
234             //In this condidition, we check if tempChar is equal to the
    user input c
235             return 1; //return 1 if yes
236         }
237         possibleLine++; //increment possibleLine to go to the next
    possible char defined by us at the beginning
238     }
239 
240     return 0; //0 if not
241 }
242 
243 
244 
245 // METHOD TO CHOOSE LINE BY INCREMENTS OR NOT
246 void changeLine (int targetLine){
247 
248     //if targetLine == -1 then we increase currentLine by 1
249     //By default currentLine == 0
250     //So if we press 'enter' it will increase currentLine by 1
251     //Hence currentLine == 1. And it does this as long as we don't
    reach the maximum of line possible
252   if (targetLine == -1) {
253     currentLine++;
254         if (currentLine >= MAX_LINES){
255             currentLine = 0; //We reset the currentLine to 0 when it
    exceeds the maximum possible lines.
256         }
257 
258   }
259     else {
260         // change to selection
261         currentLine = targetLine;
262     }
263 }
```

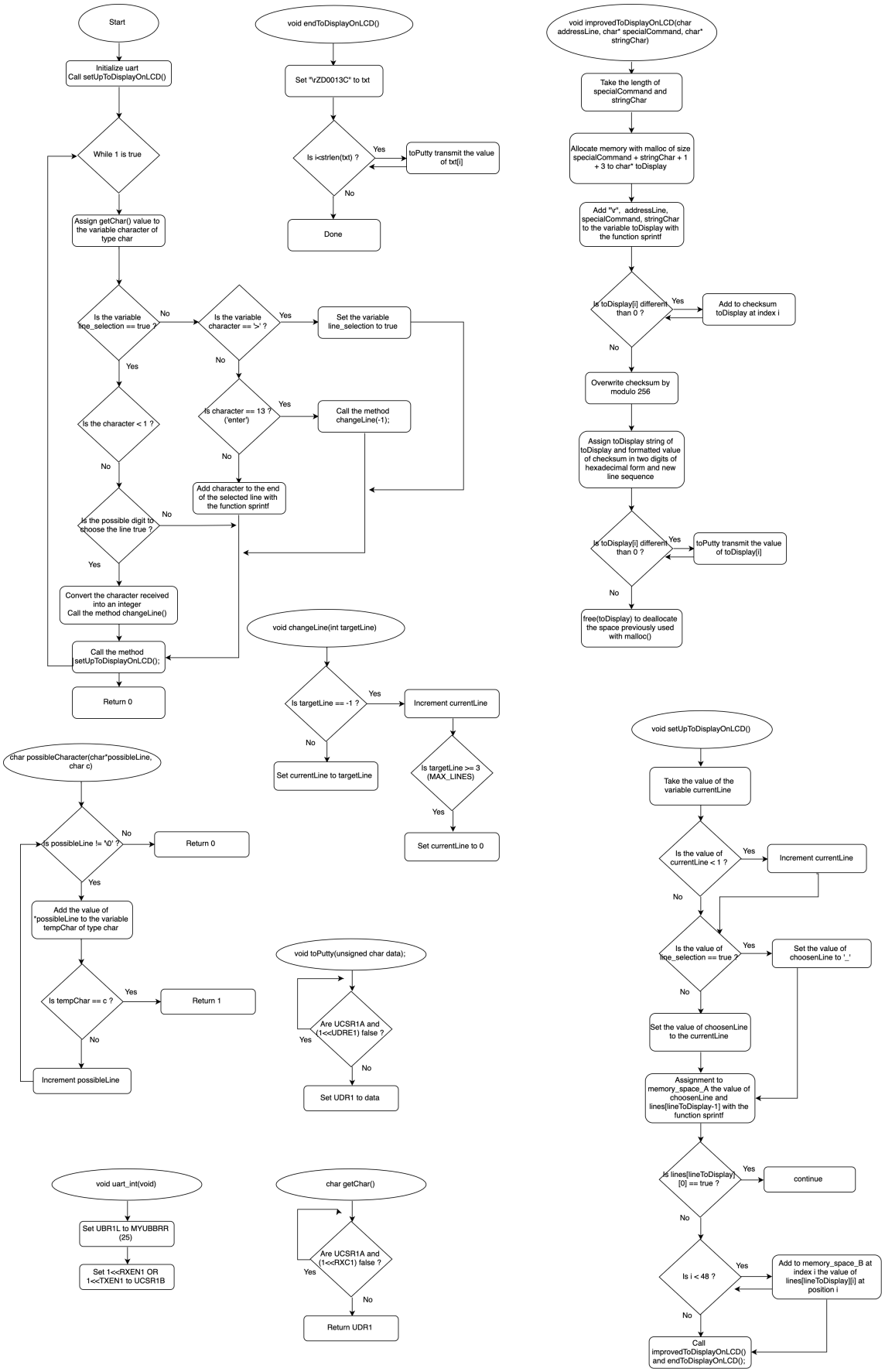

Figure 4: Task 4 flowchart

5 Task 5

Task 5 is the same as Task 4 but with more addresses to choose. (1 to 9). Therefore the flowchart of task 5 is the same as task 4.