Lab 6
Flow Free®

CSE 379

Introduction To Microprocessors

Jesse Both
Nicholas Anzalone
Lab Section: R1

April 27, 2021

Contents

Program Overview

The purpose of this program was to implement the game Flow Free in ARM Assembly. Some of the major tasks include:

- utilizing the UART0, SW2 and LED
- breaking the correct links when links are crossed
- determining the correct link character that should be displayed
- determining when to increment or decrement completed
- outputing a random board
- hiding the board when the game is paused

Division of Work

-

Jesse

Implemented the main functionality of the game.

Nick

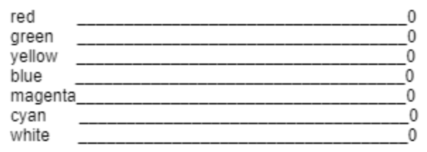
Implemented the functionality such as the random board, pause screen and timer.

Subroutines

-

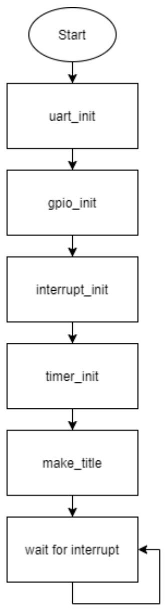
Lab6.s

This file does a lot of miscellaneous tasks, but a major part of it is how the link coordinates are stored. Each color has 30 available bytes in memory to place coordinates. The reason 30 was chosen is because it is the maximum number of links that needs to be stored (+starting O). The links are stored like this:



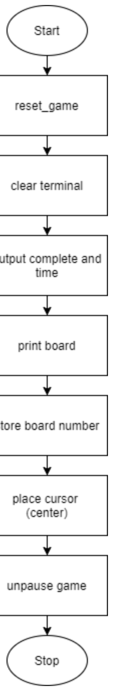
lab6

This is the main routine of the program. It initializes everything and print the title screen to Putty. It then loops until an interrupt happens.



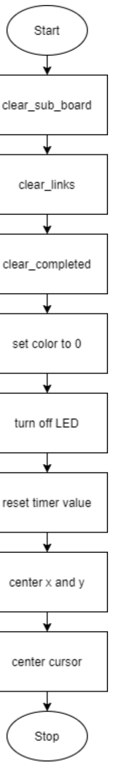
init_game

This routine sets up the game to be played. First it resets everything then it prints all the necessary elements of the game to Putty.



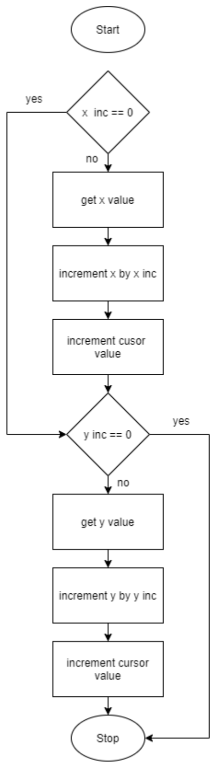
reset_game

This subroutine resets each element of the game. It removes all links, turns off led and sets the timer and completed to 0



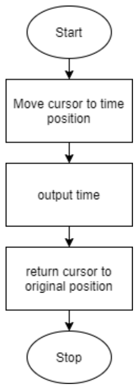
move_cursor

This subroutine is responsible for moving the cursor after WASD is pressed. It increments the cursor coordinates to move relative to the key that was pressed



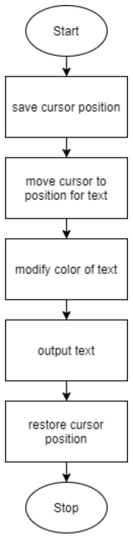
put_time

This subroutine places the correct updated time to the correct position on the screen.


Start
Move cursor to time position
output time
return cursor to original position
Stop

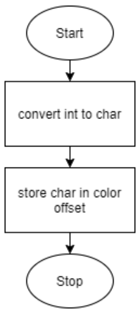
place_text

This subroutine receives the inputs of a char and cursor pointer. First it places the char in the correct position in the string. It then moves the cursor to the input position and prints the char in that location. The cursor is restored to its previous position.



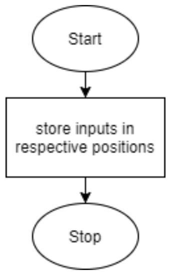
color_text

This subroutine takes a input of 0-7 for the new color. It converts the int to char and places it in the foreground position.



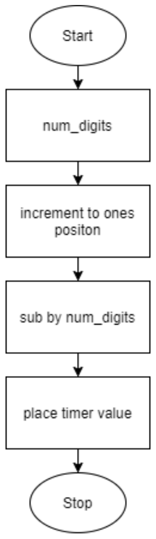
modify_cursor_two

This routine changes the x and y values of cursor_two. This relies on the other subroutine get_cursor_pos to get those x and y values



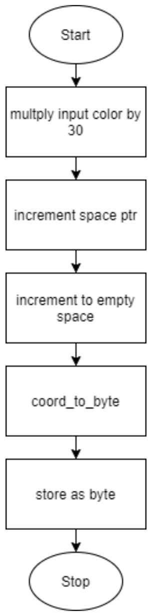
timer_to_string

This routine takes in an int for the timer and converts it it to the correct ansi string that it needs to be placed in the correct position of the screen.



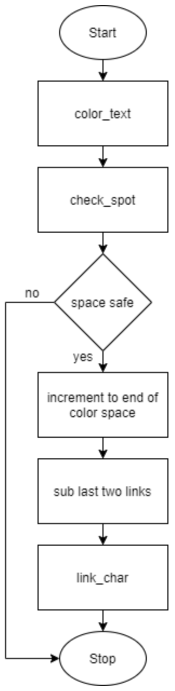
store_pos

This subroutine has inputs for x, y and the current active color. It then stores the coordinates in memory as a byte at the end of the colors space.



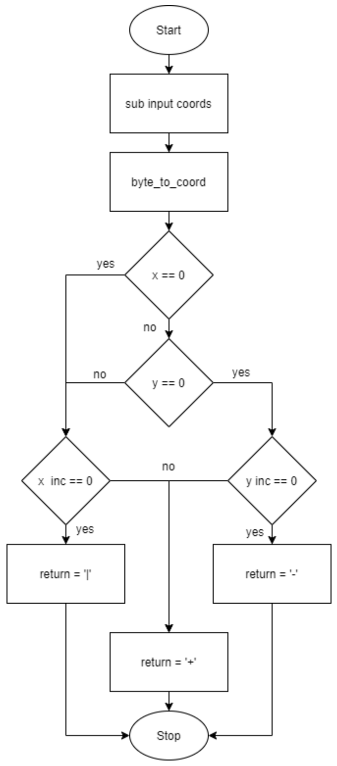
show_link

This subroutine is responsible for placing the correct link on the board in the correct location. Its input is a color to determine what color the link should be.



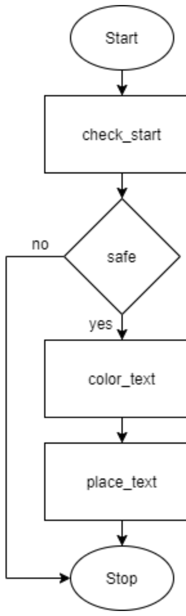
link_char

This subroutine determines what the correct type of link is required. It returns a '+', '-' or 'I'



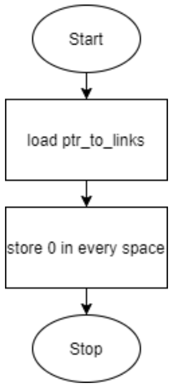
show_plus

This subroutine outputs a plus in the current position. This is used when space is pressed to deactivate the links.



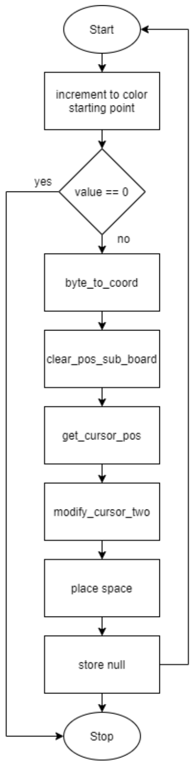
clear_links

This subroutine clears all the link coordinates from memory. This is used to to within reset_game.



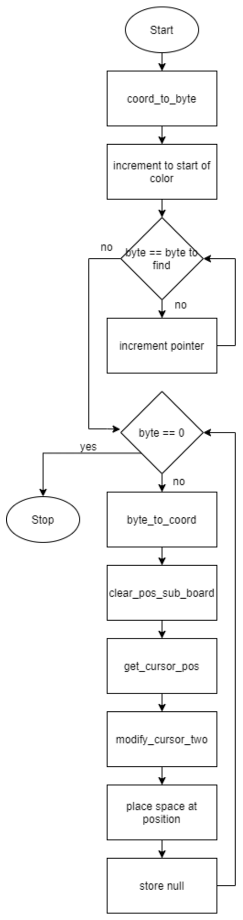
clear_color_links

This subroutine clears all the links of a specific color. This routine is used when a link is already started and an O is selected.



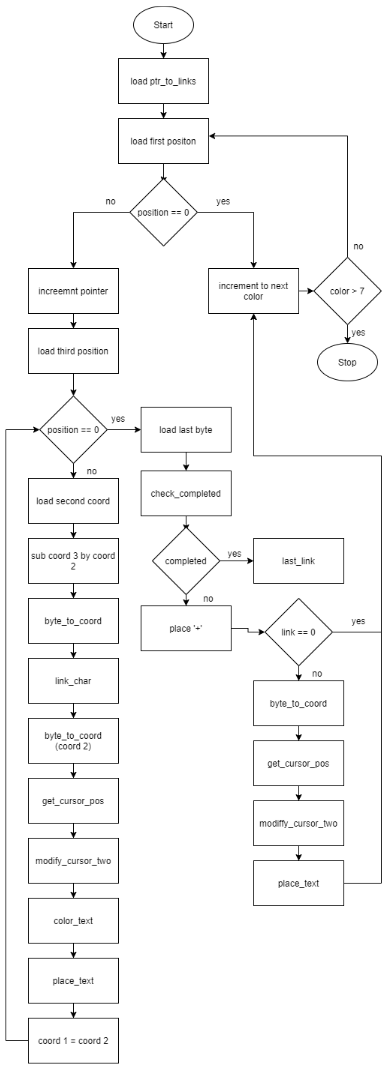
clear_pos_links

This subroutine takes in the color of links that needs to be broken. It uses the current x and y coordinates to determine where to start breaking the links. Once it finds those coordinates it breaks all the links after it for that specific color.



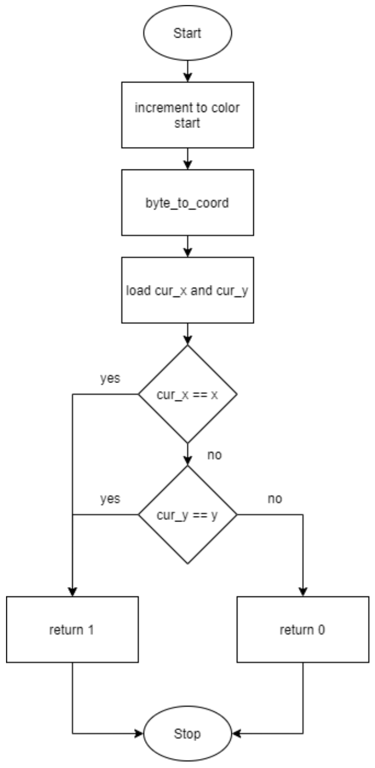
replace_links

This routine is used when the game is unpaused. Due to poor planning each link has to be recalculated to determine what char should be used in the link.



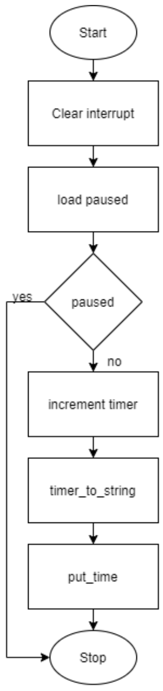
check_start

This subroutine determines if the O that the cursor is currently on was the start link or an end link. Its input is the current active color.



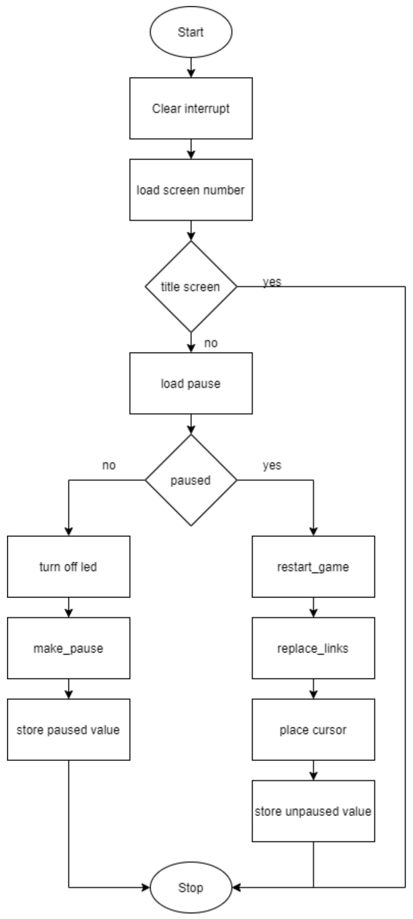
Timer_Handler

This interrupt handler is responsible for incrementing the timer when the game is not paused.



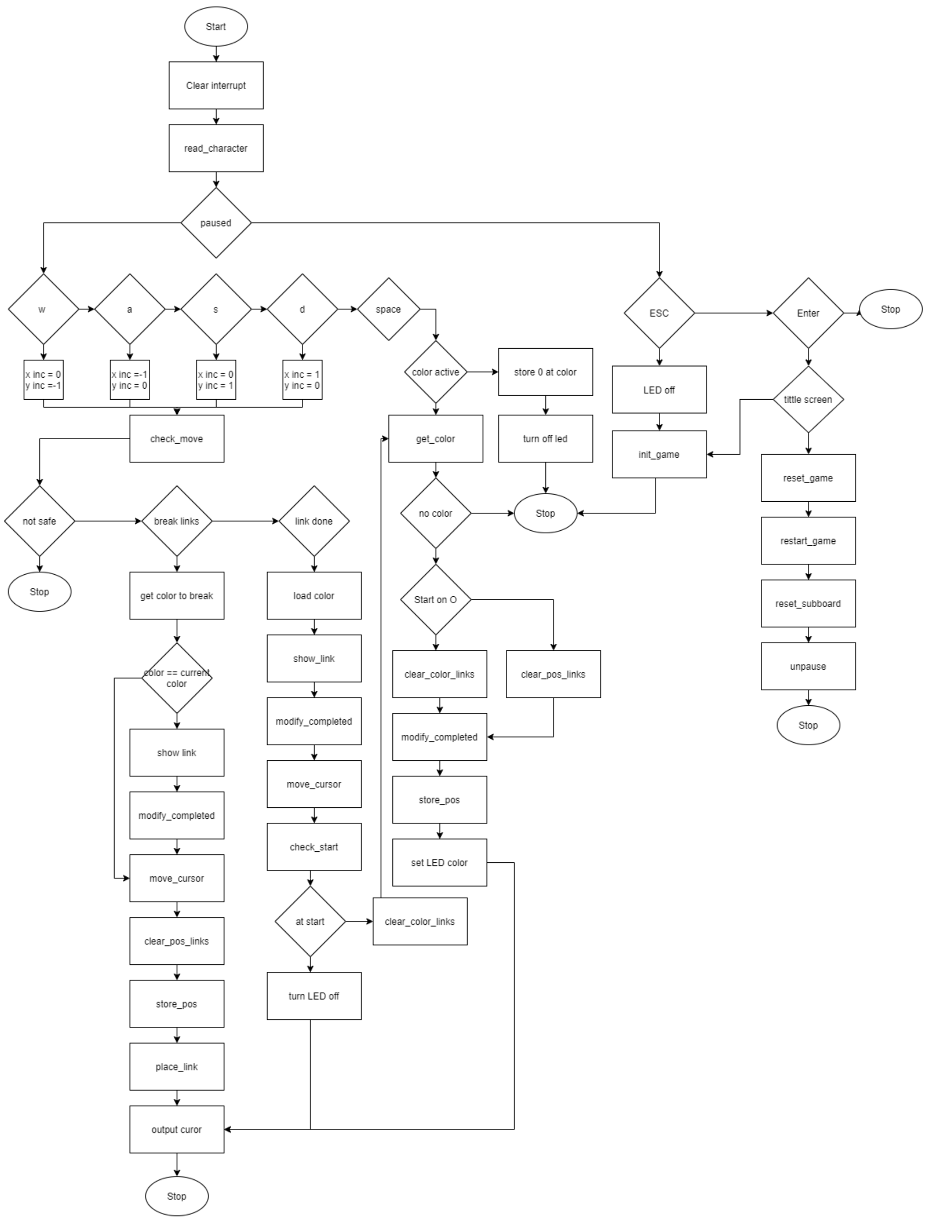
Switch_Handler

This interrupt handler is responsible for pausing and unpausing the game when it is able.



UART0_Handler

This interrupt handler is responsible for all key presses within the game. It does what ever operation is necessary regaurding moving, making links, breaking links, triggering a completed link and triggering a completed board.

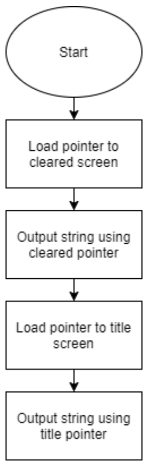


Title.s

This file does is responsible for changing the screen to the title screen, end screen or pause screen.

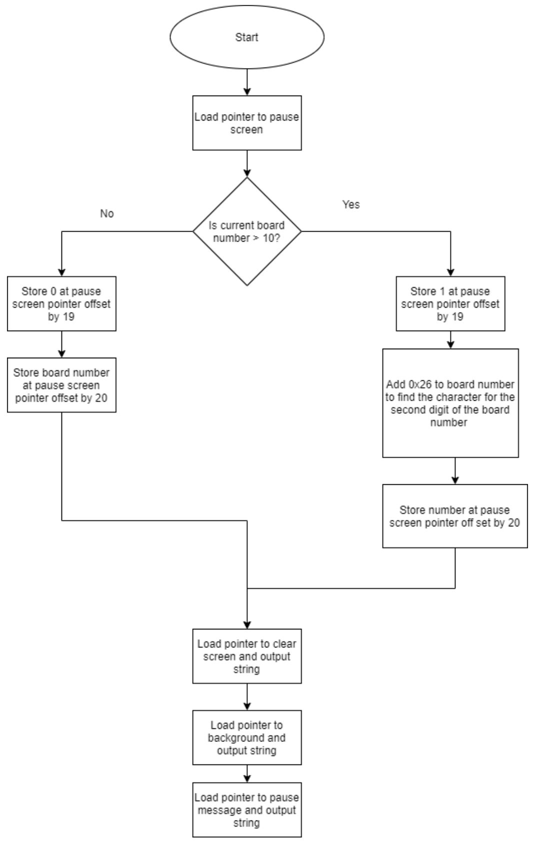
make_title

This routine will create the title menu that is to be displayed while the game first starts and print that menu to the screen.


Start
Load pointer to cleared screen
Output string using cleared pointer
Load pointer to title screen
Output string using title pointer

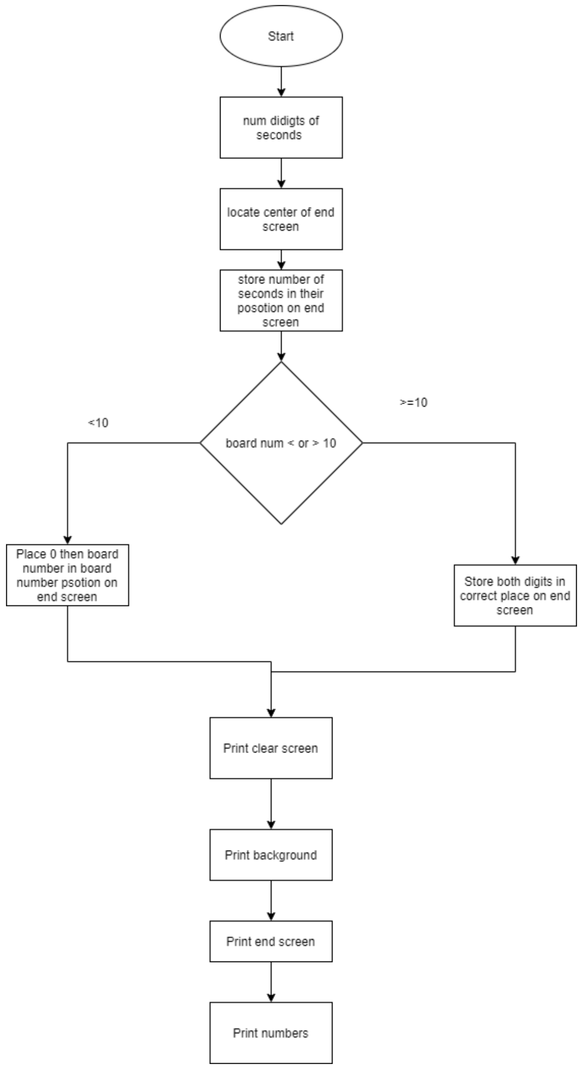
make_pause

This routine will create the pause menu that is to be displayed while the game is paused and print that menu to the screen.


Start
Load pointer to pause screen
Is current board number > 10?
No
Yes
Store 0 at pause screen pointer offset by 19
Store board number at pause screen pointer offset by 20
Store 1 at pause screen pointer offset by 19
Add 0x26 to board number to find the character for the second digit of the board number
Store number at pause screen pointer off set by 20
Load pointer to clear screen and output string
Load pointer to background and output string
Load pointer to pause message and output string

make_end

This routine will create the end game menu that is to be displayed when a game has been socessfully completed.


Start
num didigts of seconds
locate center of end screen
store number of seconds in their posotion on end screen
board num < or > 10
<10
>=10
Place 0 then board number in board number psotion on end screen
Store both digits in correct place on end screen
Print clear screen
Print background
Print end screen
Print numbers

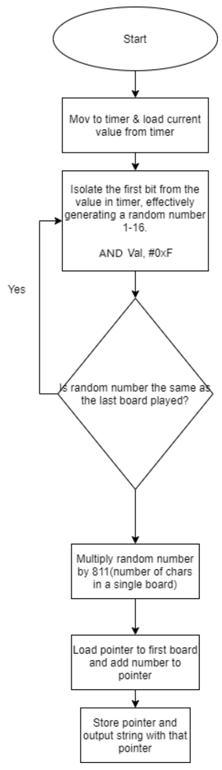
Board.s

The purpose of this file is to do the board output operations. This is where the random board is determined. Each boards string is in memory like so:

```
board:    .string 9,9,9,9,32,32,32,27,"[30;40;1mXXXXXXXXX",0xA,0xD
        .string 9,9,9,9,32,32,32,27,"[30;40;1mX",27,"[30;40;1m ",27,"[35;40;1mO",27,"[31;40;1mO",27,"[30;40;1m ",27,"[30;40;1m ",27,"[30;40;1m ",27,"[30;40;1m ",27,"[30;40;1mX",0xA,0xD
        .string 9,9,9,9,32,32,32,27,"[30;40;1mX",27,"[30;40;1m ",27,"[31;40;1m ",27,"[37;40;1mO",27,"[34;40;1mO",27,"[30;40;1m ",27,"[34;40;1mO",27,"[30;40;1m ",27,"[30;40;1mX",0xA,0xD
        .string 9,9,9,9,32,32,32,27,"[30;40;1mX",27,"[30;40;1m ",27,"[30;40;1m ",27,"[32;40;1mO",27,"[30;40;1m ",27,"[30;40;1m ",27,"[30;40;1m ",27,"[30;40;1m ",27,"[30;40;1mX",0xA,0xD
        .string 9,9,9,9,32,32,32,27,"[30;40;1mX",27,"[30;40;1m ",27,"[30;40;1m ",27,"[36;40;1mO",27,"[33;40;1mO",27,"[30;40;1m ",27,"[32;40;1mO",27,"[30;40;1m ",27,"[30;40;1mX",0xA,0xD
        .string 9,9,9,9,32,32,32,27,"[30;40;1mX",27,"[30;40;1m ",27,"[30;40;1m ",27,"[30;40;1m ",27,"[30;40;1m ",27,"[33;40;1mO",27,"[36;40;1mO",27,"[30;40;1m ",27,"[30;40;1mX",0xA,0xD
        .string 9,9,9,9,32,32,32,27,"[30;40;1mX",27,"[30;40;1m ",27,"[30;40;1m ",27,"[37;40;1mO",27,"[30;40;1m ",27,"[30;40;1m ",27,"[30;40;1m ",27,"[30;40;1m ",27,"[30;40;1mX",0xA,0xD
        .string 9,9,9,9,32,32,32,27,"[30;40;1mX",27,"[30;40;1m ",27,"[30;40;1m ",27,"[35;40;1mO",27,"[31;40;1mO",27,"[30;40;1m ",27,"[30;40;1m ",27,"[30;40;1m ",27,"[30;40;1mX",0xA,0xD
        .string 9,9,9,9,32,32,32,27,"[30;40;1mXXXXXXXXX",0
```

board_out

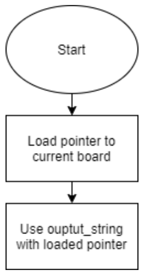
This routine chooses a random board from boards 1-16, and prints the board to the screen. In order to generate the random number, the value from the internal timer is pulled and the first bit from the timer is isolated. Effectivley randomly generating a number 1-16.



Start
Mov to timer & load current value from timer
Isolate the first bit from the value in timer, effectively generating a random number 1-16.
AND Val, #0xF
Yes
Is random number the same as the last board played?
Multiply random number by 811(number of chars in a single board)
Load pointer to first board and add number to pointer
Store pointer and output string with that pointer

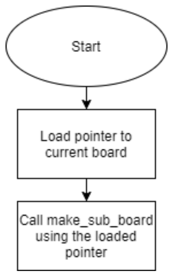
reprint_board

This routine prints out the current board in the current state it is in. It is used when going from paused state to unpaused state.


Start
Load pointer to current board
Use ouptut_string with loaded pointer

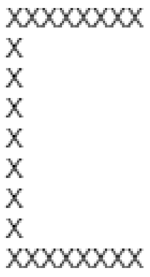
reset_subboard

This routine will reset the sub board to a clear sub board with no links. This can be used when restarting a level.


Start
Load pointer to current board
Call make_sub_board using the loaded pointer

Subboard.s

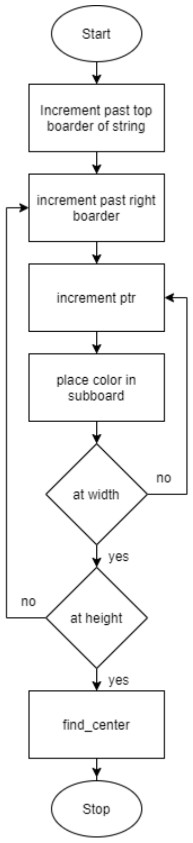
The purpose of this file is to simplify the board and remove all of the ansi characters from the board string. How the board looks is seen below.



```
XXXXXXXX
X
X
X
X
X
X
X
XXXXXXXX
```

make_sub_board

This subroutine converts the board string into a more manageable size. The purpose of the subboard is to remove all ansi characters.

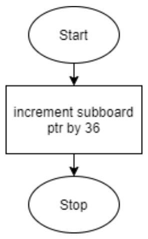


test_sub_board

This routine was used to test that the subboard was doing what it was supposed to.

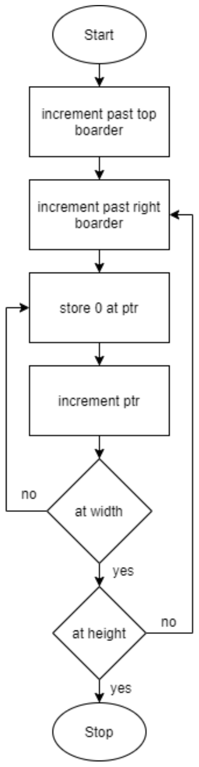
find_center

The subroutine incrments the subboards ptr to the center position to be used in rouines like check_move.



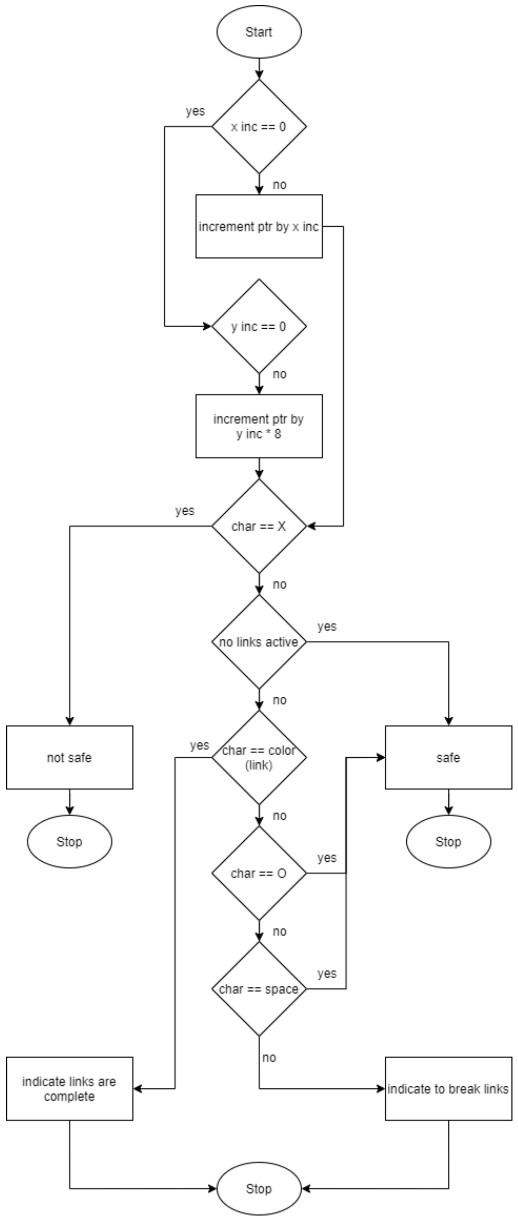
clear_sub_board

This subroutine clears the subboard to its default state.



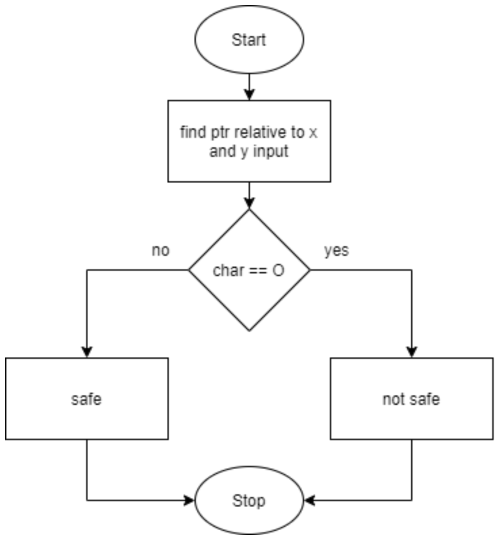
check_move

This routine determines if the move is 'safe'. It also is responsible for telling the program if links need to be broken of if the link is completed.



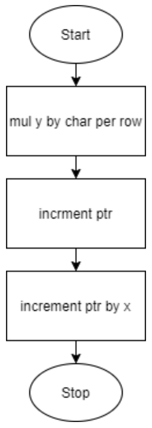
check_spot

This subroutine determines if the position is safe to put a link. If the current position is an O, you don't want to overwrite the O with a link.



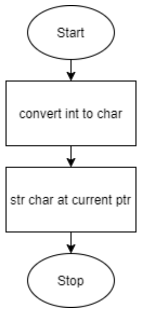
get_subboard_ptr

This routine takes in x and y coordinates and returns the subboard location replative to the input.



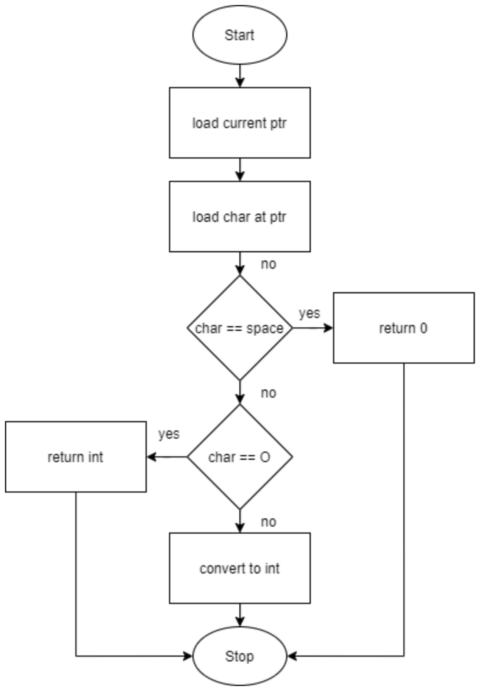
place_link

This subroutine takes a color 1-7 as an input at places the int as a char into the subboard.



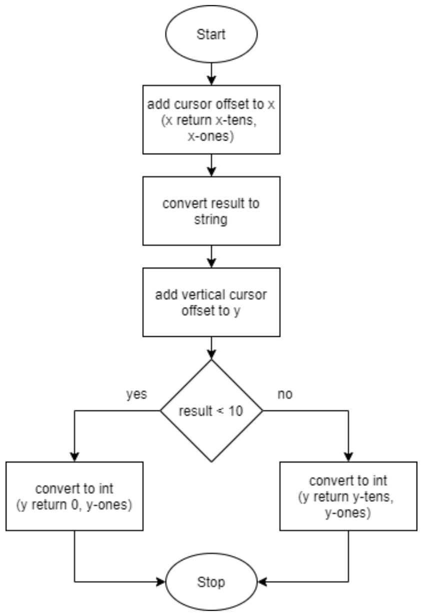
get_color

This routine is utilized when space is pressed to determine what the active color should be. If no color is present it returns 0.



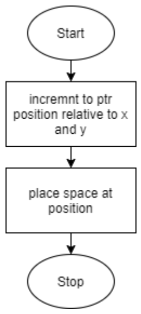
get_cursor_pos

This subroutine gets the correct cursor position within the board relative to the inputs x and y.


Start
add cursor offset to x (x return x-tens, x-ones)
convert result to string
add vertical cursor offset to y
result < 10
yes
no
convert to int (y return 0, y-ones)
convert to int (y return y-tens, y-ones)
Stop

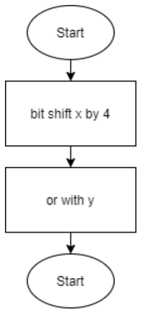
clear_pos_sub_board

This subroutine moves to the ptr relative to x and y within the subboard and places a space into that position.



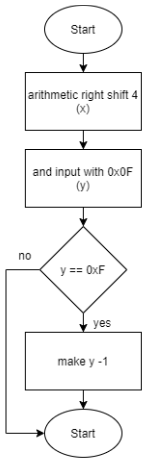
coord_to_byte

This routine encodes x and y coordintes to be store within one byte.



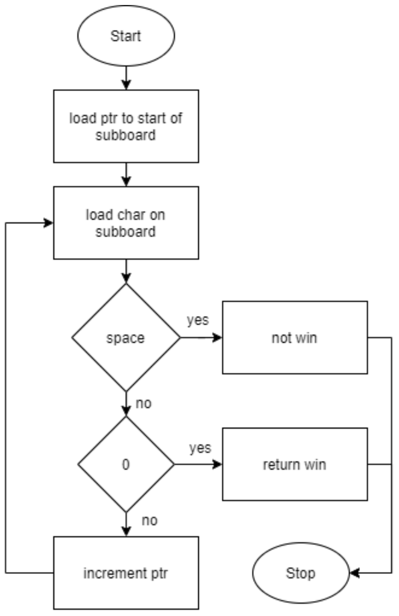
byte_to_coord

This routine reverses the encoding to output x and y coordinates.



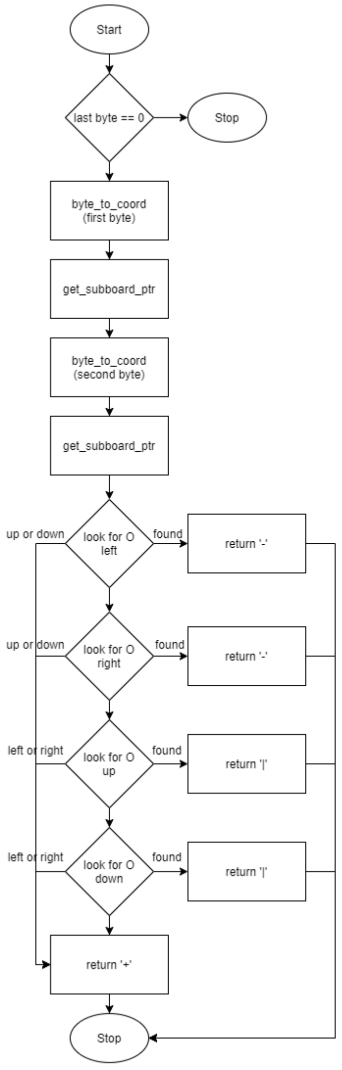
check_win

This subroutine determines if there are still any spaces on the board. If there are not, the board is completed.



last_link

Theh routine is utilized when unpausing. Due to the way link coordinates are stored this routine is required to try to find the O that is nearby.

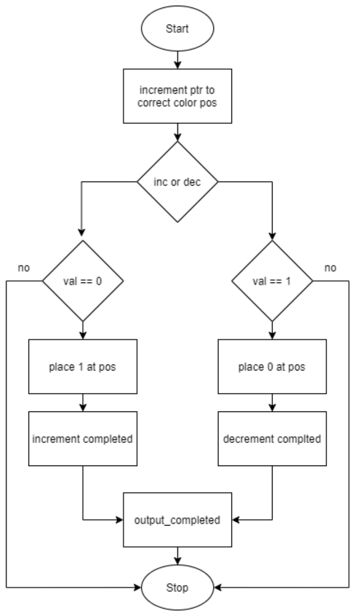


Complete.s

This file is responsible for incrementing or decrementing the completed number or determining if the game has been won. Inorder to determine if a specific color link is allowed to be changed 7 bytes of memory were used for each color. If a 1 was in space 3 (blue-start at 0) that means that link was already completed. If it was 0 it means that there is no link and it can't be decremented.

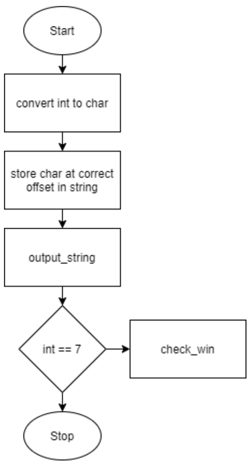
modify_completed

This subroutine determines if it is okay to increment or decrement the completed number. The inputs are the number to increment by and the color that was been broken or completed.



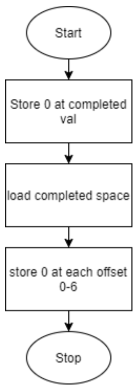
output_completed

This routine outputs the string for completed. The string contains both the value for completed and the cursor position. It takes in an int for the completed value and returns a 0 or 1 for game done.



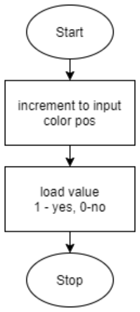
clear_completed

This subroutine sets each byte in memory to 0 for the completed values. It also sets the completed int to 0.



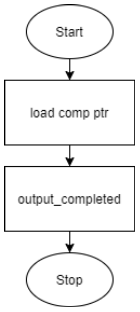
check_completed

This routine increment to the correct positon in memory and returns whether or not the input color link has been completed.



reprint_completed

This subroutine reprints the completed string for cases where the completed value does not need to be changed.

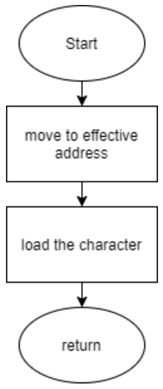


Library.s

This file contains all of the generalized subroutines that we have made througout the semester.

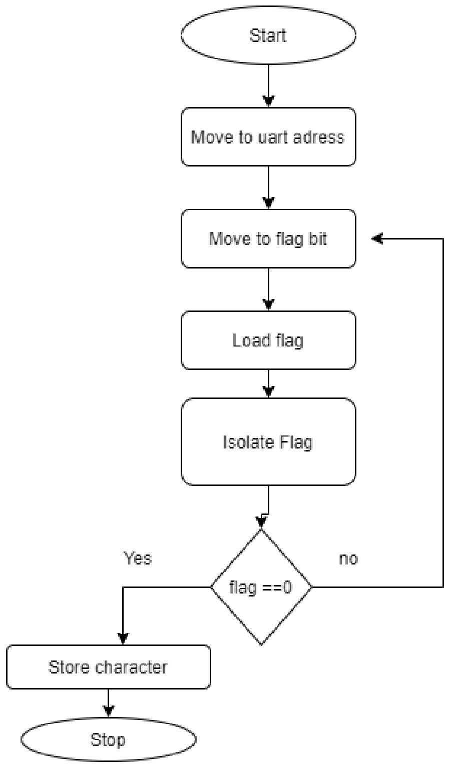
read_character

read_character first sets the uart address then increments to the flag bit for input. It loads the flag bit to determine if a key was pressed and if it was, then it loads the character.



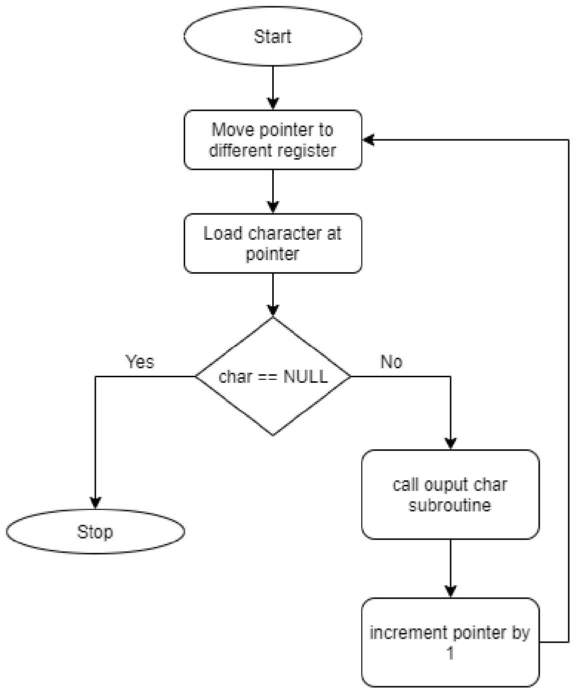
output_character

output_character will output a character that is typed to the putty screen. This is done by moving to the uart adress and visting the flag bit. It isolates the flag bit and stores its data (the character pressed).



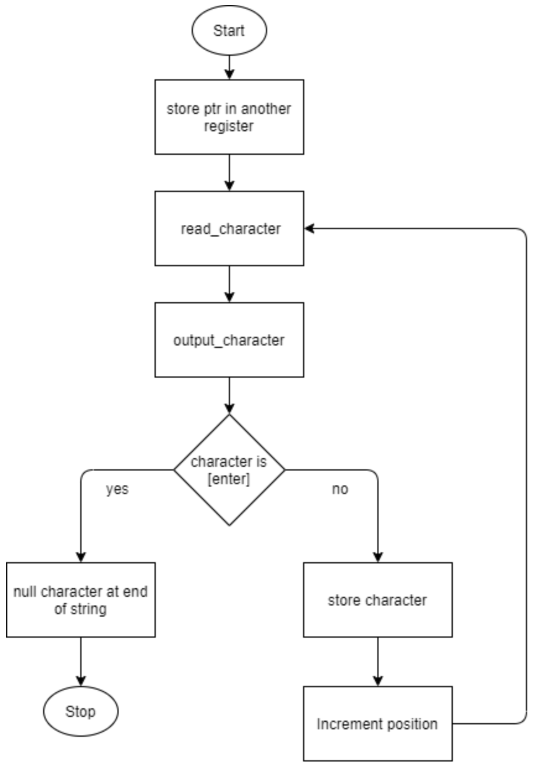
output_string

output_string is given a string and calls output_character on each specific character on the string until it reaches a null character. When it reaches a null character the output_string function stops.


Start
Move pointer to different register
Load character at pointer
char == NULL
Yes
No
Stop
call ouput char subroutine
increment pointer by 1

read_string

read_string calls read_character to accept input and then output_character to display the read character on the screen. It continues to read until the enter key is pressed and null terminates the string.

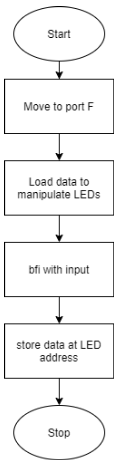


uart_init

Ths subroutine initializes the uart in order for it to be used. This is done by isolating specific peices of memory and storing a specific value at those peices of memory.

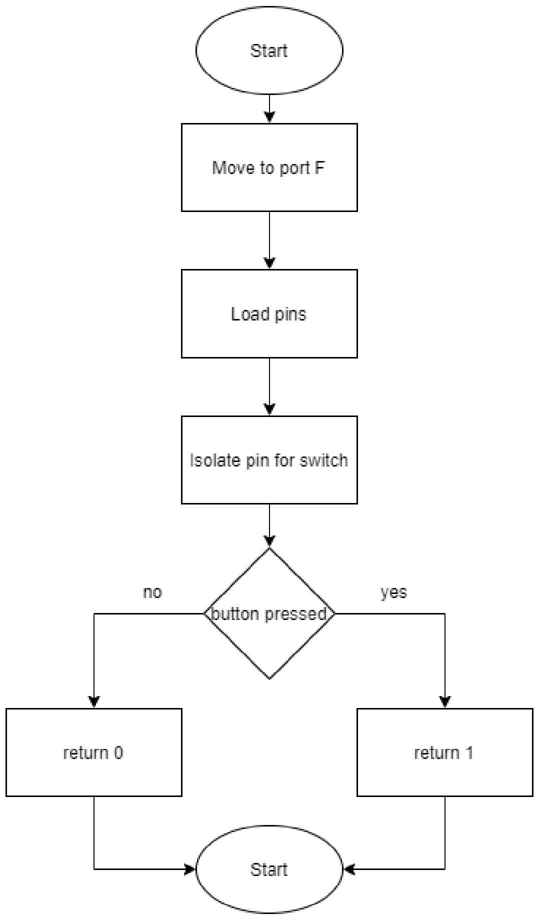
gpio_init

This subroutine initializes the tiva board for use with gpioo



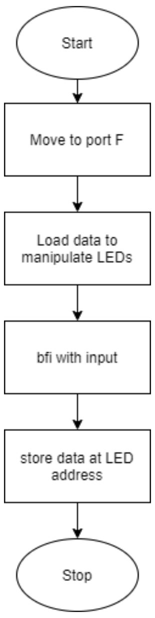
read_from_push_btn

This routine checks if the button is currently being pressed



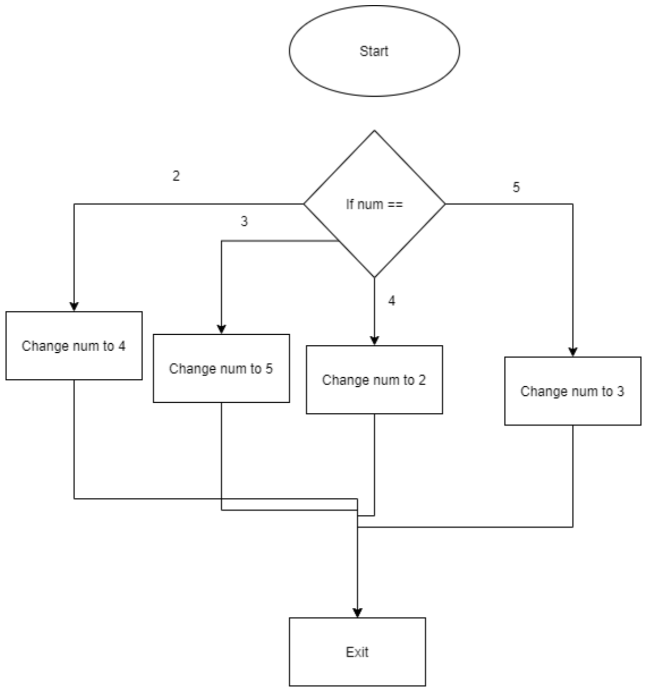
illuminate_RGB_LED

This routine changes the color of the LEDs based on the input 1-7



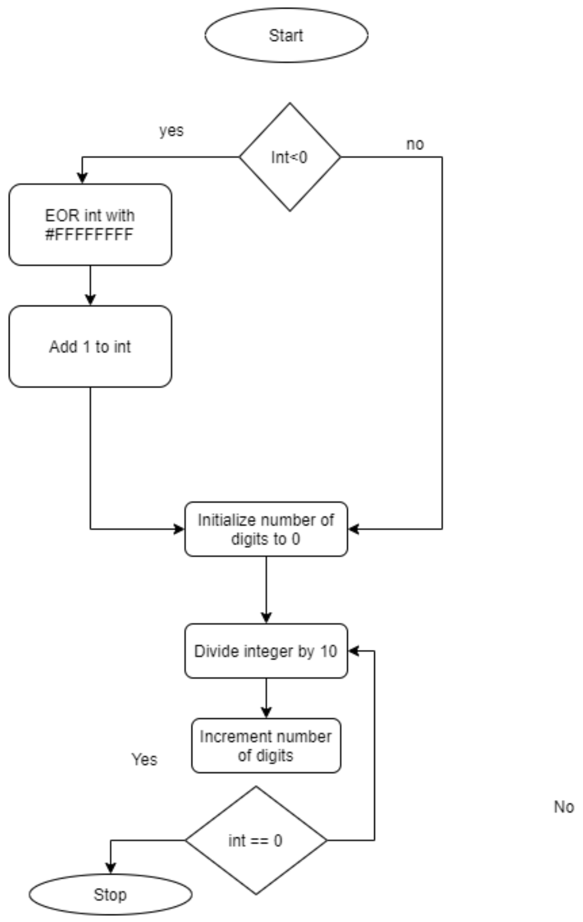
correct_num

This routine will achange numbers taken from subroutines in Lab6 and make the number taken from the ANSI Escape Sequences and correctly correlate them to the number that needs to be passed into illuminate_RGB_LED.



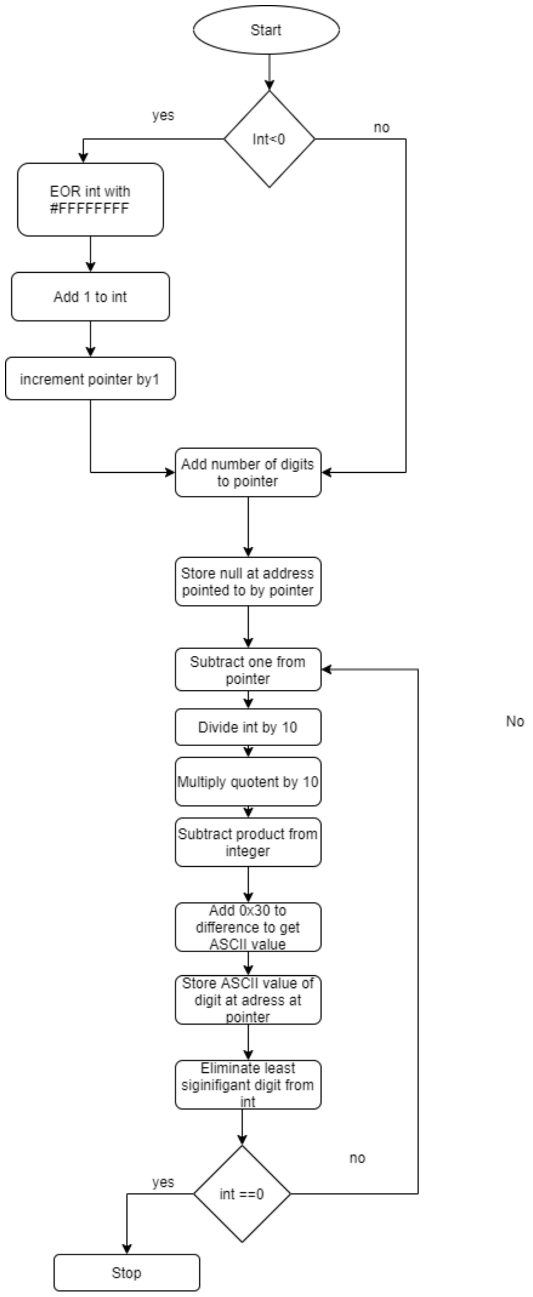
num_digits

num_digits will calculate the number of digits in a string. This is done by dividing by 10 and incrementing the number of digits each time the divison is done until the int is equal to 0.



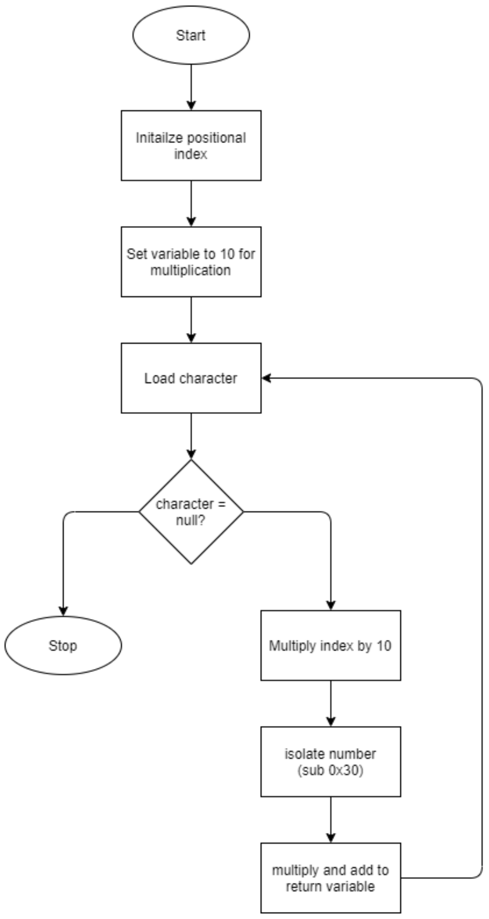
int2str

int2str will turn an integer value into a string and output that string. This is done by calculating the value at each digit in the int and then finding the ASCII value for that digit and storing that value.


Start
Int<0
yes
no
EOR int with #FFFFFFFF
Add 1 to int
increment pointer by1
Add number of digits to pointer
Store null at address pointed to by pointer
Subtract one from pointer
No
Divide int by 10
Multiply quotent by 10
Subtract product from integer
Add 0x30 to difference to get ASCII value
Store ASCII value of digit at adress at pointer
Eliminate least siginifigant digit from int
no
yes
int ==0
Stop

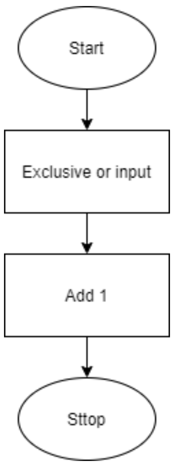
str2int

str2int initializes a posistional register to 0 then loops through each character until the null terminator. At each character it multiplies the posistional int by 10 and subtracts 0x30 from the character to isolate the int. It adds that value to the return value and increments the posistional int.


Start
Initailze positional index
Set variable to 10 for multiplication
Load character
character = null?
Stop
Multiply index by 10
isolate number (sub 0x30)
multiply and add to return variable

negate_int

This routine performs a bit flip to negate the input number and adds 1.


Start
Exclusive or input
Add 1
Sttop