School of Computer Science and Engineering

Faculty of Engineering

UNSW Sydney

COMP2121 Project:

Monorail Emulator Design Manual

by

Omar Al-Ouf & Ihor Balaban

Report submitted as a requirement for the Microprocessors and Interfacing course (COMP2121)

Submitted: October 26, 2018

Student ID: z5037275
z5229133

Contents

Chapter 1

1. Introduction

This paper provides a design overview of the program implemented for COMP2121 2018 second session's project. Fig. 1 illustrates a flow chart of the program.

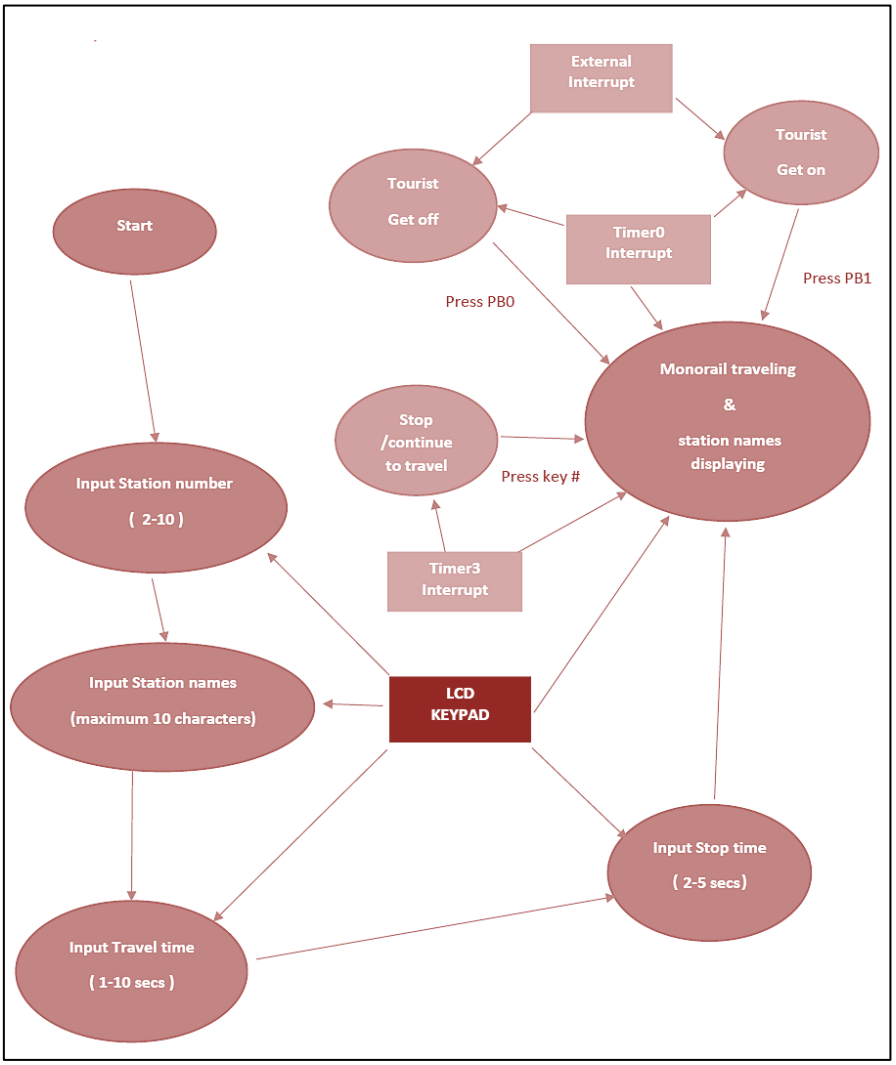

Figure 1. Flow chart of the functions.

Chapter 2

2. Data Structure

The registers are defined as below, using .def:

Table 1. Data structure.

Name	Register	Function
sec_left	R6	Travel time left before reaching next station
temp_counter	R7	Counter for the stop times
max_stations	R8	Stores the number of maximum stations
shift	R9	Which letter is selected in step2 and also used as a flag in the emulator (to move on to next station)
n_length	R10	Stores current length of name
stop_time	R11	Stores the stop time between stations
current_station	R12	Stores current station number
temp3	R13	Temporary register
flag	R14	Flag register to check if value is being displayed
char_buff	R15	Buffer that stores numbers, symbols and letters from the keypad
temp	R16	Temporary register
row	R17	The current row number
col	R18	The current column number
mask	R19	Mask for column/row during scanning
temp2	R20	Temporary register
treg	R21	Used as a status register with 8 separate flags
n_chars	R22	Stores number of characters on screen
lcd_temp	R23	Stores the ASCII value of the character displayed.
NPL	R24	The low byte of the address that stores the travel time between stations

NPH	R25	The high byte of the address that stores the travel time between stations

The addresses are defined as below, using .equ:

Table 2. Addresses.

Name	Address	Function
PORTLDIR	0xF0	PL7-4 for the output, PL3-0 for the input
INITCOLMASK	0xEF	To scan from the left-most column
INITROWMASK	0x01	To scan from the top row
ROWMASK	0x0F	To obtain input from Port L
LCD_RS	7	Different pins for the LCD
LCD_E	6	
LCD_RW	5	
LCD_BE	4	

The data segment is defined as below, using .byte:

Table 3. Data segments.

Name	Bytes	Function
stat_names	100	To store the names of ten stations, 10 bytes each, with a maximum number of 10 characters
stat_times	10	To store the travel time between two stations, 1 byte each, with a maximum number of 10
stat_char_nums	10	To store how many characters are in each station name
stations	1	To store the number of stations
TC	2	The time counter in Timer0

Table 4 shows the representation of each bit in the *treg* status register.

Table 4. *treg* status register.

Bit number	Function (if set)
1	Stop at next station
2	Motor running
3	# pressed (which means stop immediately)
4	Emulator running
5	Step4 running (which is the input stop time)
6	Step3 running (which is the input stop between stations)
7	Step2 running (which is input station names)
8	Step1 running (which is input maximum number of stations)

Chapter 3

3. Functions and Algorithms

3.1 Procedure

Step 1: Entering the number of stations.

This step obtains the number of stations. At first, the function ‘step1’ is called to display the instruction on the LCD screen. The main function continuously scans for a key press. When a key is pressed, the program will check the status and handle it. Since the data type required in this step is of a numeric type, all other keys do not work. After entering the value, the hash key indicates the end of input.

It is also important to handle any errors, such as inputs under 2, above 10 and any other irregularities. When the hash key is pressed, the function will check whether the value is valid. If nothing is entered or if the station number is 0 or 1, 10 will be stored. On the other hand, if the value is larger than 10, the last digit will be read as the input. For example, the input would be 3 if 123 is typed. There is no limit to the number of digits which can be entered; the hash indicates the end of the input and the is stored into the data segment *stations*. The program will then automatically move to the next step, with the function ‘step2’ being called to display the next instruction.

Table 5. Number of stations.

Button	Function
0-9	Enter number of stations
#	Finish inputting and store the number of stations
A	Will not work
B	Will not work
C	Will not work
D	Will not work
*	Will not work

Step 2: Entering the name of each station.

This step obtains the names of the stations. The chosen procedure for this step follows that of a phonetic keyboard. This enables the keys A, B, C and D to each represent a character on a number key (A for the 1st character, B for the 2nd, C for the 3rd and in some cases D for the 4th). After pressing one of these keys, followed by a numeric key, the desired character will be displayed on the LCD screen. For example, if the letter 'D' is required to be outputted, the A key must first be pressed, followed by 3. Similarly, for output 'Y', C must be pressed followed by 9. When a key is pressed, the program will check the status and display the letter number (*shift* variable) on the LED. In this step, since the data type required is of a characteristic value, key D will only work when keys 7 or 9 (letters S and Z) are pressed. Keys 0 and 1 will not work in this step. White spaces are inserted by pressing the star key (*). Every time a character is inputted, it will be stored into the data segment *stat_names*. After entering the station name and pressing the hash key, the amount of characters of the name is stored in *stat_char_nums*, and the name of the next station can be entered. The maximum number of characters for any name is 10. If the length of *stat_char_nums* is 0, then the station number is outputted when the emulator is running. The function will increment the variable *current_station* and call the function 'step2' to display the instruction. Once the name of the last station has been inputted, the program will automatically move to the next step and 'step3' will be called. The detailed algorithm for the character inputs will be introduced later.

Table 6. Station names.

Button	Function
2-9	Enter characters
#	Finish inputting and store the name of station
A	Represents the first character on the number key
B	Represents the second character on the number key
C	Represents the third character on the number key
D	Represents the fourth character on the number key
*	White space
0/1	Will not work

Step 3: Entering the travel time from one station to another.

This step obtains the time for the monorail to travel from one station to the next excluding stops. As in the previous step, the main function constantly scans for a key press. When a key is pressed, the program will check the status and handle it. Since the data type required for this step is of a numeric value, pressing other keys will not work. After entering a value, the hash key indicates the end of input.

As per the specifications [1], the maximum travel time is 10, thus any values greater than this must be handled accordingly. Upon pressing the hash key, the function will check whether the value is valid. If nothing is entered before pressing the hash key, the travel time will also be stored as 5 seconds. Entering 0 will store a travel time of 1 second; otherwise, any input over 10 will take its last digit to be the input. There is no limit to the number of digits which can be entered; the hash indicates the end of the input and the is stored into the data segment *stat_times*. After storing all travel times, the program will continue to the next step by calling the function ‘step4’.

Table 7. Travels times.

Button	Function
0-9	Enter number of stations
#	Finish inputting and store the number of stations
A	Will not work
B	Will not work
C	Will not work
D	Will not work
*	Will not work

Step 4: Entering the stop time.

This step acquires the stop time of the monorail at any station. As with steps 3 and 4, t the main function constantly scans for a key press. When a key is pressed, the program will check the status and handle it. Since the data type required for this step is of a numeric value, pressing other keys will not work. After entering a value, the hash key indicates the end of input.

The minimum stop time is 2 seconds while the maximum is 5 seconds. When the hash key is pressed, the function will check whether the value is valid. If nothing is entered, the default stop time is 5; if the value entered is either 0 or 1, the stop time will be stored as 2 seconds. If a single digit entered, and is larger than 5, 5 seconds will be stored. If the value is larger than 9, only the last digit will be considered as the input. There is no limit to the number of digits which can be entered; the hash indicates the end of the input. The value will be stored into the variable *stop_time* and the function ‘step5’ will be called to display the instruction.

Table 8. Stop times.

Button	Function
0-9	Enter number of stations
#	Finish inputting and store the number of stations
A	Will not work
B	Will not work
C	Will not work
D	Will not work
*	Will not work

Step 5: Monorail emulation.

During this step, the monorail emulation begins. The initialisation for the Timer0 interrupt will start and a few variables will be reset. After a waiting time of 5 seconds, the motor will begin spinning at a speed of 60 revolutions per second, which means the monorail has started travelling. During the emulation, the screen continually displays the name of the next station.

PB0 and PB1 are to simulate if a tourist wants to get off or get on at the next station. If one of them is pressed, the monorail will stop at the next station with the stop time you entered. Whenever the monorail is stopped, two LEDs will blink at a frequency of 3 Hz. The hash key simulates the case where the monorail stops half way between two stations. When the hash key is pressed again, the monorail will continue travelling and the LEDs will discontinue blinking.

3.2 Algorithms

3.2.1 Character Generation

Keypad scanning:

- Pressing the star key (*) will store white space into the data segment *stat_names* and display it on the LCD screen.
- Pressing the character keys will change the value of the variable *shift* (A:1, B:2, C:3,

D:4)

- Pressing the number keys will have a range of functions, depending on the current step.
- The function *convert_to_letter* converts a pressed key into a ASCII character.
- The function *convert_key* outputs the character inputted onto the LCD.
- The function *num_key* converts a number to either a number or character, depending on the current step.

3.2.2 Emulating the behaviour of the monorail

This is implemented in the Timer0OVF procedure:

1. Start Timer0OVF.
2. Load and increase the temporary counter.
3. Check if the emulator is running.
4. Check if the motor is spinning.
5. If the motor is not spinning, blink two LEDs at 3 Hz.
6. Check if 1 second has passed.
7. Check if the hash key is pressed.
8. If the motor is running, decrement the travel time (*sec_left*) by 1 second.
9. If *sec_left* reaches 0, check if a person wants to get off/on.
10. If a person wants to get off, stop the motor and decrement the stop time (*temp_counter*) by 1 until it reaches 0. Otherwise move on to next station.
11. End Timer0OVF.

3.2.3 Number Operations

1. Check which step the program has reached.
    a. If step 1:
        i. Handle error inputs.
        ii. Store the number of stations.
        iii. Call the function ‘step1’ to display the instruction.
        iv. Initialise step 2.
    b. If step 2:
        i. Store station names.

ii. Handle error inputs.
iii. Call the function ‘step2’ to display the instruction.
iv. Check if all station names have been entered.
v. Initialise step 3.

c. If step 3:
i. Handle error inputs.
ii. Call the function ‘step3’ to display the instruction.
iii. Check if all travel times have been entered.
iv. Initialise step 4.

d. If step 4:
i. Handle error inputs.
ii. Store the stop time.
iii. Call the function ‘step4’ to display the instruction.
iv. Initialise step 5.

e. If step 5:
i. Start Timer0OVF.
ii. Loads the pointers for the emulator.
iii. Increment the pointers accordingly for the step.
iv. Displays ‘wait’ message for 5 seconds.
v. Calls *print_station* function, which prints the next station name, increments the pointer accordingly and begins the motor if it’s the first station.

3.3 Macros

Macros were used for LCD commands, clearing 2-byte variables, checking various flags in treg, multiplying by 10 and loading a data segment into a pointer.

3.4 Interrupts

EXT_INT0:

1. Get status from SREG.
2. Change flag variable for stopping at the next station.

3. Write status back to SREG.

EXT_INT1:

1. Get status from SREG.
2. Change flag variable for stopping at the next station.
3. Write status back to SREG.

References

[1] Wu, H. (2018). COMP2121 Microprocessors and Interfacing Notes. Accessed 17 Oct. 2018, from WebCMS3 CSE UNSW.