DESIGN MANUAL

Monorail emulator

ABSTRACT

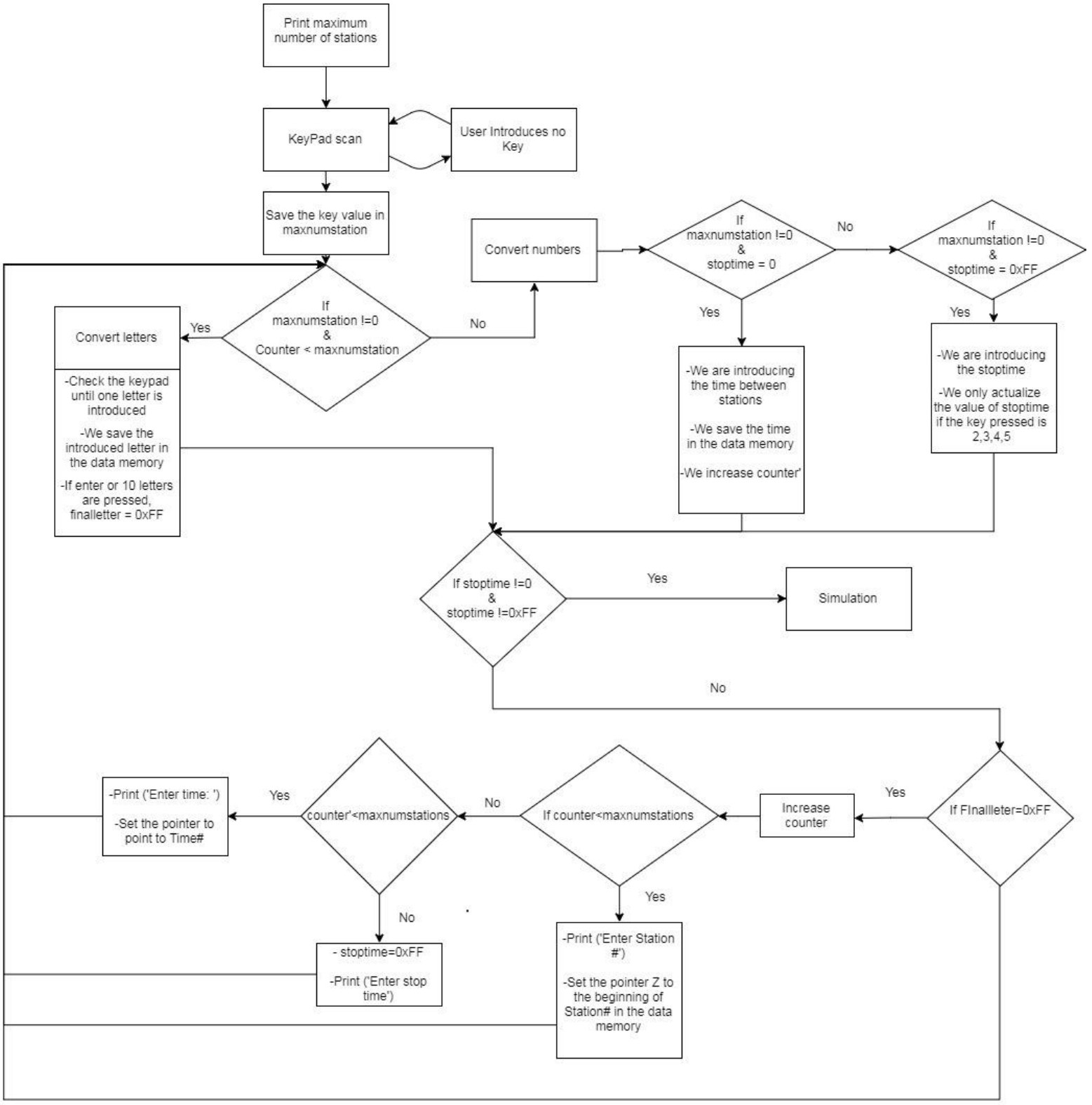
In this document we can see the main structures and control flow of the emulator.

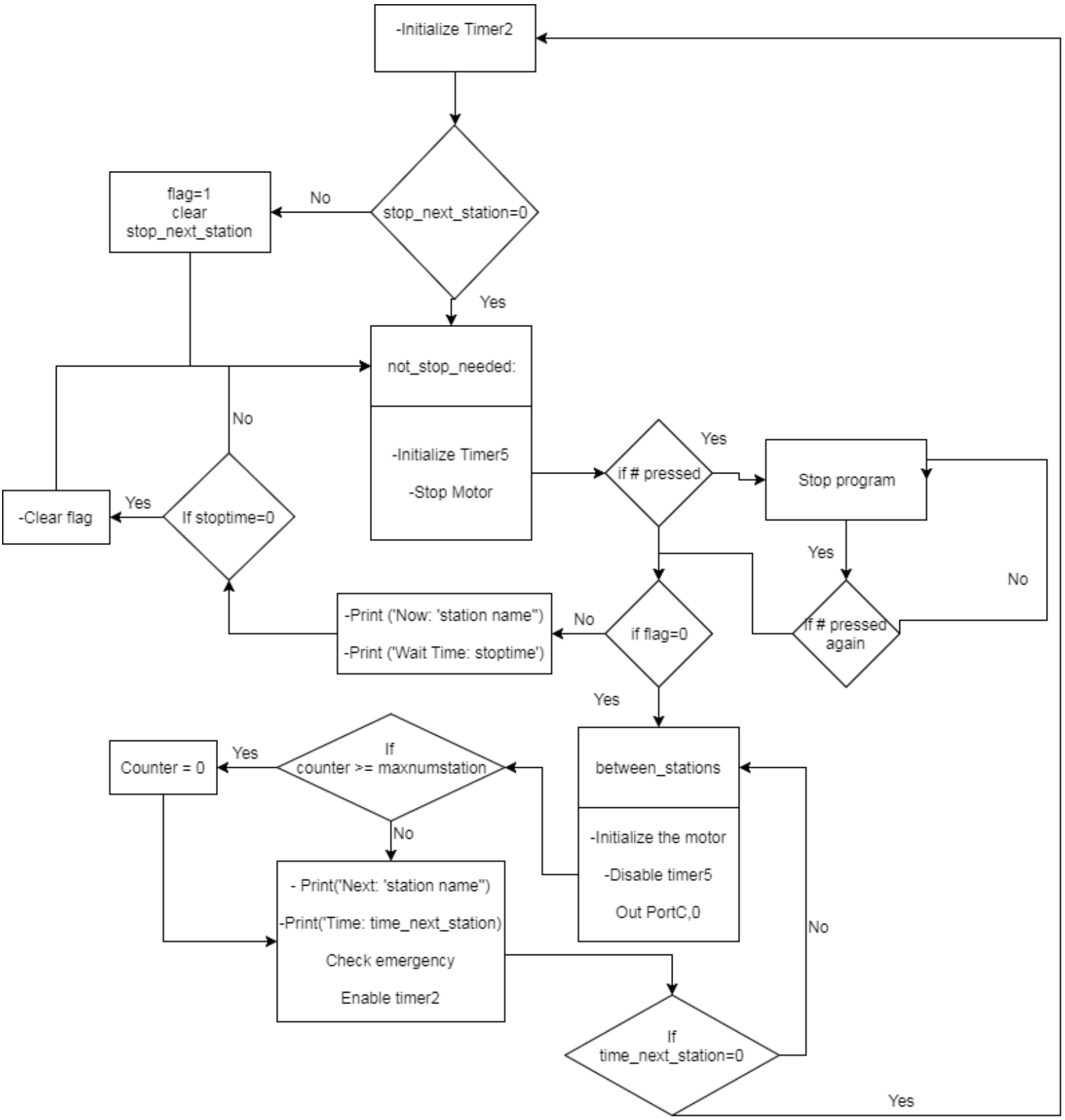
Unai Aguilera and Alejandro Hernandez

COMP2121

Control Flow

In this section, we present two diagrams showing the control flow of the emulator. Diagram 1 shows the control flow of the configuration section of the code, and Diagram 2 shows the control flow of the simulation part of the code. Both of them show the flow of the program and its main parts.




-Initialize Timer2
stop_next_station=0
No
flag=1
clear
stop_next_station
Yes
not_stop_needed:
-Initialize Timer5
-Stop Motor
if # pressed
Yes
Stop program
Yes
No
If # pressed again
No
if flag=0
No
-Print ('Now: 'station name")
-Print ('Wait Time: stoptime')
If stoptime=0
Yes
-Clear flag
No
Yes
between_stations
-Initialize the motor
-Disable timer5
Out PortC,0
If
counter >= maxnumstation
Yes
Counter = 0
No
- Print('Next: 'station name")
-Print('Time: time_next_station)
Check emergency
Enable timer2
If
time_next_station=0
No
Yes

Functions, Macros and Interrupts

Functions:

Division:

The function division is called in order to print the revolutions of the motor every second when is spinning, as it is a number higher than 9. It receives as an input the register row, which contains the value of the revolutions in one second. As row contains a number higher than 9, we obtain the digits of the number by dividing the number by ten. Making so we obtain the values in reverse order, we make it recursive, so that it prints in the LCD the digits in correct order. The only output of the function is the number printed in the LCD.

Station Name:

Station name is called when we want to print the name of the station in the LCD during the simulation. The inputs of the functions are counter and the maximum number of stations. The function basically checks the value of the counter, and sets the pointer in the corresponding data space, so that latter in the program we can print it. If the value of the counter is equal to the maximum number of stations, then the pointer is set to the stations 0. The only output of the function is the pointer set to the corresponding data address.

Time:

Time is used to when we are introducing the time between stations. It checks the value of counter in order to know the time we are introducing. We keep increasing counter, so we have to subtract the value of maximum number of stations, which are the only two input registers. After checking the value of counter, the function sets the pointer in the corresponding data address, which is the only output with the printing of it in the LCD.

Convert_numbers:

This function is called different times in the main program, but the purpose is always entering a number. The first thing this functions does is to calculate the value of key pressed by noticing the row and column the key is in. The first time is called the user has to introduce the number of stations the emulator will have. This time the range of values if 1-10, so it checks is the value of the number of stations is 0 and it makes the keys 1 to 10 available. Then the function is called when the user has to introduce the time between the stations, so this time again the values are between 1 and 10. The third time the function is called to introduce the time the train will be stopped in the stations. This time the value has to be between 2 and 5, so before entering the main program sets stoptime to 0xFF so that the function can notice if the available keys are 1 to 10 or 2 to 5. It checks the value of stoptime, and if it is 0xFF it only makes valid the keys 2-5, considering the rest as error keys.

Convert_letters:

Convert letters is used to convert the key pressed in the keypad into the desired letter. It is set so that it is called after we press a key in the keypad, then it checks the column and row the key is in. Once it locates the letter, it compares its value with the previously pressed letter. If it is the same letter and is being pressed in an interval smaller than one second, it means that the user wants to introduce the next letter available of that key. Say we are pressing the key that contains A,B and C. If we press the key the second time within a second, it will mean that the letter B was desired. If the interval of time was bigger than one second, then it means that it wants to introduce a new letter. In order to calculate the second, the function activates the Timer0 when a key is pressed, and checks if the next one is pressed in less than a second. When enter is pressed, the functions sets the value of the final letter register to 0xFF so that the main program knows the user has finished introducing the station name. The function itself handles the error by printing the error message in the display and letting the main program know that the user has to introduce the same station name again.

Macros:

The macros we have used in the program are the following ones:

do_lcd_command: This macro is used to write in the control register of the LCD. Is is used for example to clear the display, jump to the second line, move the cursor backwards and many more.

do_lcd_data and do_lcd_data1: These two macros write the desired ascii characters in the LCD, but they do it in a different way. do_lcd_data uses a register as the input, whereas do_lcd_data1 uses a character as an input. We need to use these two different registers because in the first one we move the value of the register to the register used to print, and the later one loads the data in the register.

clear: The clear macro is used to clear the TempCounter# used in the timers. Once overflow has occurred, the TheCounter#, needs to be restored to 0 in order to be able to count the time again properly. It simply loads 0 in the two bytes of the TempCounter

clear_station: Clear_station uses the same logic of clear, but in this case it is used to erase the letters stored in the name of the station. If the user is introducing the station 1 and by any change presses an error button, the name has to be introduced again from the beginning, thus, the memory has to be cleared so that the new word is introduced.

Interrupts:

RESET:

The reset interrupt creates is used to initialize the interrupts, registers, LCD, keypad, LEDs and everything that is going to be used along the program. It is done at the very beginning of the program. It sets the LEDs to output; initializes the LCD; Initialises all the Timers, setting the correct pre-scalar; it enables the external interrupts of the PB0 and PB1, as well as the one used to count the holes of the motor. It initialises all the register, clearing all of them and setting the needed valued on those who start the program with a certain value. Finally, it starts the program printing in the LCD the first instruction for the user.

Timer 0:

Timer 0 is used to measure one second. This is needed to see if the user has pressed two keys with more than one second in between or less.

Timer 2:

Timer 2 also measures one second, but this time it is used to decrease the values of stoptime and time_next_station when is convenient, as well as setting a flag to count the revolutions every second.

Timer 5

Timer5 is used to make the 3Hz blinking of the leds. In order to do so, the timer measures ⅓ of a second, and when that time passes, checks the state of the leds, and then it reverses it. That is, if they were on, it switches them off, and the other way around.

External Interrupt 0

External interrput 0 sets the stop_next_station register to 1 every time is pressed, so that latter in the program, it stops in the next stations, and switches on the lower 4 LEDs

External Interrupt 1

External interrput 1 also sets the stop_next_station register to 1, but in order to differentiate it from the external interrupt 0, switches the upper 4 LEDS on.

External Interrupt 2

External Interrupt 2 is used to count the holes that passes through the detector, when a second has passed, the interrupt measures the revolutions that the motor had in the last second and moves it to row so that the program prints them. Furthermore, it performs feedback. As the desired speed is 60 rps, it increases or decreases the Dutty cycle in order to get that speed.

Data Structures:

Registers used in the program:

maxnumstations=r2	Maxnumstations is used to store the total number of stations that we have.
numletters=r3	numletters is the number of letters that the station has. Once numletters reaches ten, we know that is time to stop introducing more letters for the name of a particular station. Basically, numletters is a counter that is increased every time a letter is introduced for the name of a station.
delay_one = r4	Low register of the delay used to stabilize the keypad
delay_two = r23	High register of the time to stabilize the keypad
counter2=r5	Used to count if we have pressed a key more than once. If the same key is pressed repeatedly, we use counter2 to know when the key is pressed three times in a row. Once a key is pressed three times in a row, we must start to display the same letter again. For example, if key 1 is pressed 4 times, A will be displayed in the LCD. We use counter2 to display A instead of D.
flag=r6	We use flag for: 1- Flag is set to 1 when a letter is pressed, and to 0 again when a second passes since the last letter was pressed. 2- In the emulation when flag is 0 we are between stations, when it is 1 we are at a stop.
stoptime=r7	This is the time the train has to stop in a station. It will be decreased every second using Timer2.
time_next_station=r8	Time left for arriving to the next station. We decrease the value of this register every second (using Timer2) if we are between stations.
auxstoptime=r9	It is used to restore stoptime to its original value. Since we decrease stoptime every second, once we reach 0 we need to restore it to its original value.
stop_next_station=r10	Used to know if a passenger wants to get off or get in in the next station. If PB0 or PB1 are pressed, this register will be written to 1, indicating that a passenger wants to get off at the next station. Otherwise, this register will contain the number 0.
flag_emergency=r11	Used to stop the train at emergency. If # is pressed, this register will contain the value 1. Otherwise it will contain the number 0.

led_state=r12	Used to know the state of the leds when blinking. If the leds are off, this register contains the number 0. Thus, if the leds are off, it means that we have to turn them on. If this register contained the value 1, it means that the leds are on and that we need to turn them off.
countholes=r13	Used to calculate the speed of the motor by counting the holes that pass through the laser.
revolutions=r14	It is used to calculate the revolutions per second of the motor. Every time we count 4 holes, we increase revolutions.
flag_timer0=r15	It is used to actualize the speed of the motor every second. Once a second passes, we clear flag_timer0 and then we know that we can print the speed of the motor in the LCD. After printing the speed of the motor, we set the value of this flag to 1, and we don't print again the speed of the motor until one second passes again.
temp =r16	Temporary register.
row =r17	Used to store the row the keypad is analysing. We also use row as a copy of revolutions in the simulation part.
col =r18	It stores the column the keypad is analysing.
mask =r19	Used to scan the keypad
temp2 =r20	Temporary register.
ascii = r21	Used to load the ascii value.
counter = r22	Used to know if we introduced the name for all stations and the time between all the stations. Once a station name is introduced, we increase the value of counter. When counter reaches the same value as maxnumstations, we stop introducing more names of stations. Then, we reuse counter. We clear it and we use counter in the same way for the times between stations. We refer to this reused counter in the control flow diagram as counter'.
prev=r24	It is used: 1-To know if we pressed the same key more than once. Prev stores the value of the previous introduces key. 2- In the simulation in order to print the 10 characters of the station names. That is, in the simulation we reuse prev and we use it to count up to ten to print all the letters of a station.
finalletter=r25	Used to store the letter we introduce when introducing the names of the stations. It is set to 0xff if enter is pressed.

Data Memory

For controlling the led blinking, the time left to arrive at a station, the time left waiting in a station, and to control different flags and delays, we used timers. We used 2 eight-bit timers (Timer0 and Timer2) and one sixteen-bit timer (Timer5). Therefore, we created different temporal counters to count to the desired time. The following counters are used:

TempCounter: .byte 2	Two byte number to count the number of overflows of timer0 until one second is reached.
TempCounter2: .byte 2	Two byte number to count the number of overflows of timer2 until one second is reached.
TempCounter5: .byte 2	Two byte number to count the number of overflows of timer5 until a third of a second is reached. Used for Led Blinking.

The names of the stations are stored in the data memory. We allocate 10 spaces in total. One for each station. We name them: Station0, Station1 ,…., Station9. Each memory space is ten bytes. One byte for each letter. If the number of stations is 5, for example, the Station5 to Station 9 will be empty.

The times between stations are stored in the data memory as well. We store each time in a memory space of one byte under the name of Time0, Time1…etc. The time to go from Station 0 to Station 1 is stored in Time0, the time to go from Station 1 Station 2 is stored in Time1, and so on. Finally, the time to go from Station n to station 0 is stored in time n. If less than 10 stations are introduced (let's say n stations), Timen to Time9 will be empty.

To run the motor at a speed of 60 rps, we used feedback. That is, we read the register revolutions, and if it is greater than 60, we decrease the duty cycle. If on the other hand the register revolutions is less than 60, we increase the duty cycle. To save the duty cycle, we allocate one byte in the data memory under the name Dutycycle.