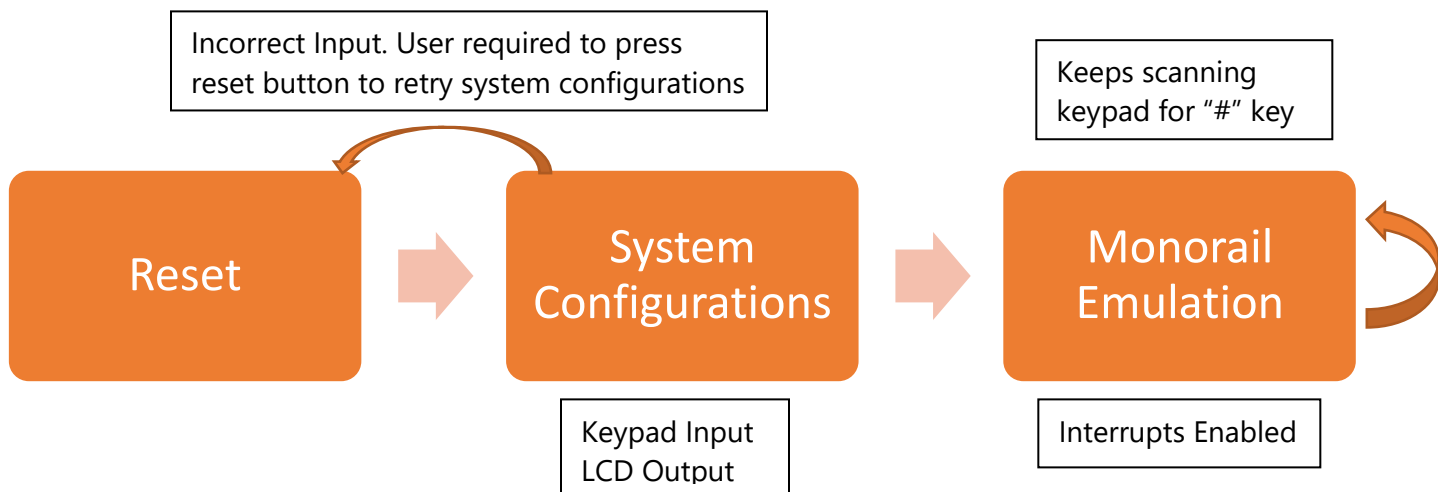


# A MONORAIL EMULATOR

## DESIGN MANUAL

### SYSTEM FLOW CONTROL



1. The program starts off in RESET, which initialises the stack pointer, data segment variables, ports and their directions, the LCD, keypad.
2. The program then goes through system configurations through taking input from the keypad storing variables into the data segment, and displaying this on the LCD. If user input is incorrect, the user is required to press the reset button to pass the system configurations.
3. The program then decides what to do with the variables in the data segment created from system configurations and enters monorail emulation. In monorail emulation, the timer, LED and motor ports are initialised here and the interrupts for the buttons and holes. The monorail emulation also keeps scanning the keypad for a "#" key user input. The other actions that are performed on the monorail are performed through interrupts which are supported through calling functions in other modules.
4. For additional details on how the actions are performed see the module specification below. The modules are source files in our project which contain functions related to the module name

## MODULE SPECIFICATION

- The module "main.asm" contained the main project file where the whole program runs.
- The module "init.asm" contains all the definitions, constants, LCD settings, macros and data segments
- The module "Sysconfig\_test.asm" contains the demo test where it displays on LCD confirmation of user input and requires any key to be pressed on keypad to confirm

| MODULE          | FUNCTIONS             | INPUT  | OUTPUT   |
|-----------------|-----------------------|--|--|
| <b>Buttons</b>  | Initialise buttons    |  | Trigger interrupt on falling edges<br>Enable external interrupts             |
|                 | Increases tourist on  | Data Segment<br>byte: tourist on               | Writes to data segment<br>tourist on and tourist count                       |
|                 | Increases tourist off | Data Segment<br>byte: tourist off<br>and count | Writes to data segment<br>tourist off and tourist count                      |
| <b>DC Motor</b> | Initialise motor      |  | Initialise PWM in Timer 3<br>Initialise DDRE<br>Initialise related variables |
|                 | Measure RPS           | Data Segment<br>byte: holes                    | Writes to data segment<br>current RPS  |
| <b>Delay</b>    | Sleep 1ms             |  | Software delay of 1ms  |
|                 | Sleep 5ms             |  | Software delay of 5ms  |
|                 | Sleep 20ms            |  | Software delay of 20ms   |
|                 | Sleep 1000ms          |  | Software delay of 1000ms   |
|                 | Sleep 5000ms          |  | Software delay of 5000ms   |
| <b>Keypad</b>   | Initialise Keypad     |  | Initialise Keypad<br>Initialise DDRL   |
|                 | Keypad Scan           | Keypad input                                   | Stores in register keypad input  |
|                 | Read Character        | Keypad input                                   | Converts digit to character A-Z  |
|                 | Read Station Number   | Keypad input                                   | Write to LCD Station number<br>Write to data segment station                 |

## Assembly

|            |                       |   |   |
|------------|-----------------------|---|---|
|            |                       |   | number  |
|            | Read Station Name     | Keypad input  | Write to LCD Station name<br>Write to data segment station name |
|            | Read Travel Time      | Keypad input  | Write to LCD Travel Time<br>Write to data segment Travel Time   |
|            | Read Stop Time        | Keypad input  | Write to LCD Stop Time<br>Write to data segment Stop Time       |
| <b>LCD</b> | Initialise LCD        |   | Initialise LCD  |
|            | LCD Output One        |   | Writes to LCD   |
|            | LCD Output Two        |   | Writes to LCD   |
|            | LCD Output Three      |   | Writes to LCD   |
|            | LCD Output Four       |   | Writes to LCD   |
|            | Print Number Station  |   | Writes to LCD   |
|            | Print Incorrect Input |   | Writes to LCD   |
|            | Print Finish          |   | Writes to LCD   |
|            | Print Name Station    |   | Writes to LCD   |
|            | Emulator LCD Display  | Data Segment<br>byte: Station<br>Name, Measured<br>RPS, Target RPS,<br>Tourist Count,<br>Tourist On, Tourist<br>Off, Seconds<br>Counted, Monorail<br>Status | Writes to LCD   |

## Assembly

|              |                          |  |   |
|--------------|--------------------------|--|---|
| <b>LED</b>   | Initialise LED           |  | Initialise LED<br>Sets all LED bit to 0   |
|              | LED ON                   |  | Sets all LED bit to 1   |
| <b>Timer</b> | Initialise Timer         |  | Writes to data segment temp<br>counter and second counter<br><br>Initialise TIMSK0  |
|              | Increment seconds passed |  | Writes to data segment third<br>of second passed, seconds<br>passed   |
|              | Check order stop         | Data Segment byte:<br>order to stop, third of<br>second passed,<br>current time to next<br>station   | Writes to data segment target<br>RPS, order to stop, monorail<br>stopping or moving   |
|              | Stop Station             | Data Segment byte:<br>monorail stopping or<br>moving, stopping<br>time, seconds passed   | Writes to data segment target<br>RPS, monorail stopping or<br>moving, third of second<br>passed, seconds passed   |
|              | Update Station           | Data Segment byte:<br>monorail stopping or<br>moving, current time<br>to next station,<br>seconds passed,<br>tourist on, tourist off,<br>tourist count, next<br>station, number of<br>stations | Writes to data segment<br>monorail stopping or moving,<br>current time to next station,<br>seconds passed, tourist on,<br>tourist off, tourist count, next<br>station |

## Assembly

### INTERRUPT SPECIFICATION

| INTERRUPTS       | INPUT   | OUTPUT  |
|------------------|---|---|
| Reset            |   | Initialises stack pointer<br>Initialises data registers and clears ports<br>Initialises LCD Initialises keypad<br>Initialises variables and data segments |
| Right button     |   | Increases tourist off and decrease tourist count  |
| Left button      |   | Increases tourist on and increase tourist count   |
| Interrupt 2      | Holes   | Writes to data segment holes counted  |
| Timer 0 Overflow | Local counter stored in data segment<br>Power level | If local counter reaches 2604 (0.33s): Update stations, LEDs, measured RPS, Motor speed, LCD display.   |

---

### DATA STRUCTURES

---

#### REGISTERS

Registers were used for storing values of temporary variables for the keypad along with loading, storing the values of the data segment variables. Three registers (R17-19) are specifically used as storing the current row and column when keypad scans and for holding the values of the row mask. Register 16 (R16) was reserved for storing and writing outputs. Four registers (R20-23) were defined for temporary registers for holding temporary variables to load and store data segment variables and to pass values to other functions. Two registers (R24:25) were used to store a two-byte number to allow for easy output address from calculation subroutines or for general temporary storage.

---

#### DATA SEGMENT VARIABLES

Data segment variables stored the values of variables amongst timer counters, button de-bouncing state, hole count and monorail emulation variables. These monorail emulation variables included the number, names, travel times, stop time of stations, tourist on, off and count, current RPS and target RPS and general helper variables. These general helper variables included the next station, monorail status, third of second passed and second passed, the current time to next station, and the order to stop the monorail.

---

#### CONSTANTS

Constants were set (set/equ) to make the code maintainable and readable. Constants were used for LCD settings and instructions, keypad variables (I/O port masks) and delay constants.