**Data Dictionary**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Location** | **Description** |
| current\_floor | 8-bit integer | R16 | Used to store the lift's current floor level |
| col | 8-bit integer | R21 | Used to store the current column number. Used in keypad scanning |
| colmask | 8-bit integer | R23 | Mask for current column during keypad scan |
| door\_state | 8-bit integer | R19 | Used to indicate the state of the doors. Takes on 4 values:  0 (closed), 1 (opening), 2 (opened), 3 (closed) |
| eighthTimeCounter | 16-bit integer | dseg | Determines whether 1/8th of a second has passed. Used in Timer2 |
| final\_dest | 8-bit integer | R17 | Used to store the destination floor that the lift is moving towards |
| floor\_changed | flag | dseg | Used to indicate whether the floor level has changed |
| LED\_door\_state\_output | 8-bit integer | dseg | LED pattern for the door state component |
| LED\_lift\_direction\_output | 8-bit integer | dseg | LED pattern for the lift direction component |
| lift\_direction | 8-bit integer | R18 | Used to indicate the direction lift is moving. Takes on 3 values: -1 (down), 0 (stationary), 1 (up) |
| oldCol | 8-bit integer | dseg | Used for keypad debouncing |
| oldRow | 8-bit integer | dseg | Used for keypad debouncing |
| row | 8-bit integer | R20 | Used to store the current row number. Used in keypad scanning |
| rowmask | 8-bit integer | R22 | Mask for current row during keypad scan |
| stop\_at\_floor | flag | dseg | Used to indicate a "stop at current floor" request |
| stop\_at\_floor\_progress | 8-bit integer | dseg | Used to keep track of the different stages within the "stop at current floor" procedure |
| temp1 | 8-bit integer | R24 | Temporary register used for general processing. Sometimes used with temp2 for 16-bit processing. |
| temp2 | 8-bit integer | R25 | Temporary register used for general processing. Sometimes used with temp1 for 16-bit processing. |
| timer0\_TimeCounter | 8-bit integer | dseg | Used to count number of timer 0 overflows |
| timer4\_TimeCounter | 8-bit integer | dseg | Used to count number of timer 4 overflows |

Notes:

* For some reason, making "poll\_keypresses" a function will bug the entire system. Currently it is being called by "rjmp" in MAIN, which is a temporary fix. However it would be ideal to make this a function for cleaner and more consistent code.

**MODULE SPECIFICATION**

**Main**

‘Main’ is responsible for executing the lift, in both "Normal mode" and "Emergency mode". It processes of data, and prepares variables prior to appropriate procedures being carried out by other components. The keypad is responsible for acknowledging the request to visit a floor, or request for emergency.

**Timer 0**

[Diagram]

‘Timer0’ is responsible for moving the lift through the floors, by keeping track of time. Once the duration to reach a floor has been elapsed, it will set a flag indicating the floor has changed (which is to be acknowledged and reset by ‘Main’).  It uses “timer0\_TimeCounter” as a local variable to keep track of the time progressed. It will only start timing when the lift is in motion AND if a floor change hasn’t been requested, so it must check both conditions prior to carrying out its procedure. 'Timer0' has a prescaling of CLK/8.

**Timer 1**

[Diagram]

‘Timer1’ is responsible for displaying the emergency alarm signal through the LED’s. It does so by checking whether the emergency alarm flag is set, prior to displaying the alarm pattern, and flashing the appropriate strobe LED lights (see "Algorithms" section for more detail with displaying LED's). It uses "timer1\_TimeCounter" as a local variable to keep track of the timing. 'Timer1' has a prescaling of CLK/8.

**Timer 2**

[Diagram]

'Timer2' is responsible for displaying the state of the doors and the direction the lift is travelling. It takes in only two inputs from the set of global variables: "door\_state" and "lift\_direction". Based on their values, it loads two patterns into two local variables, which are then displayed through the LED's. When door is opened/closed, the strobe LED lights turn on. For closing/opening, the strobe LED lights flash (see "Algorithms" section for more detail with displaying LED's). 'Timer2' has a prescaling of CLK/8.

**Timer4**

[Diagram]

'Timer4' is responsible for carrying out the "stop at floor" procedure, by keeping track of the time elapsed and counting the progress. This procedure is only executed if there is a "stop at floor" request, indicated by the flag "stop\_at\_floor". The time is tracked using the local variable "timer4\_TimeCounter", and the progress is processed using another local variable "stop\_at\_floor\_progress". Depending on the progress, 'Timer4' changes the door state ("door\_state" in global variables). 'Timer4' can also change progress directly if it detects a door change request (through the variable "door\_state\_change\_request"). Timer4 has a prescaling of CLK/8.

**Timers 3 and 5**

[Diagram]

These two timers form the unit for controlling the motor. 'Timer3' is responsible for generating a PWM waveform to control the motor, whilst 'Timer5' is responsible for turning the motor on/off, depending on the value in "door\_state" from global variables. 'Timer5' is able to manipulate the motor by changing the value in Timer3's 16-bit output compare register B (OCR3B). 'Timer3' is configured to be in fast PWM mode.

**Push buttons**

[Diagram]

PB0 is responsible for triggering a "close door" request when pushed", and PB1 is responsible for triggering an "open door" request when pushed. Both involve changing the value of the global variable "door\_state\_change\_request", which can then read by the 'Timer4' component.

**Algorithms**

**Processing and ordering of requested floors**

**Timing with timers**

All timers that require timing do so using the number of overflows that have occurred as a counter. This counter is stored using a local variable for that timer. Prescaling is configured so that the number of overflows that have occurred will correspond to a particular amount of time (this number of overflows is determined theoretically, and is an approximate estimation). Once the number of overflows is reached, the counter is reset.

**Push buttons debouncing**

**Keypad reading and debouncing**

**Processing a "door state change" request**

When the door is requested to be opened/closed, a variable becomes loaded with a particular value. Upon detecting this request, the system determines how to process this request, with relation to the door state. When door is requested to be opened: if door is opening, the request is simply cleared. When door is already opened, the

**Displaying LED’s**

There is a variable in dseg that contains the pattern to output using the LED's. When the pattern is flashed through LED's (eg door is closing) the pattern is constantly being initialised and reset (to zero). When pattern is moving (eg displaying lift direction), the pattern is shifted (using **lsr** or **lsl**), and once it gets to 0, becomes reset to the appropriate pattern.