# Design Manual for COMP9032 Helicopter Project

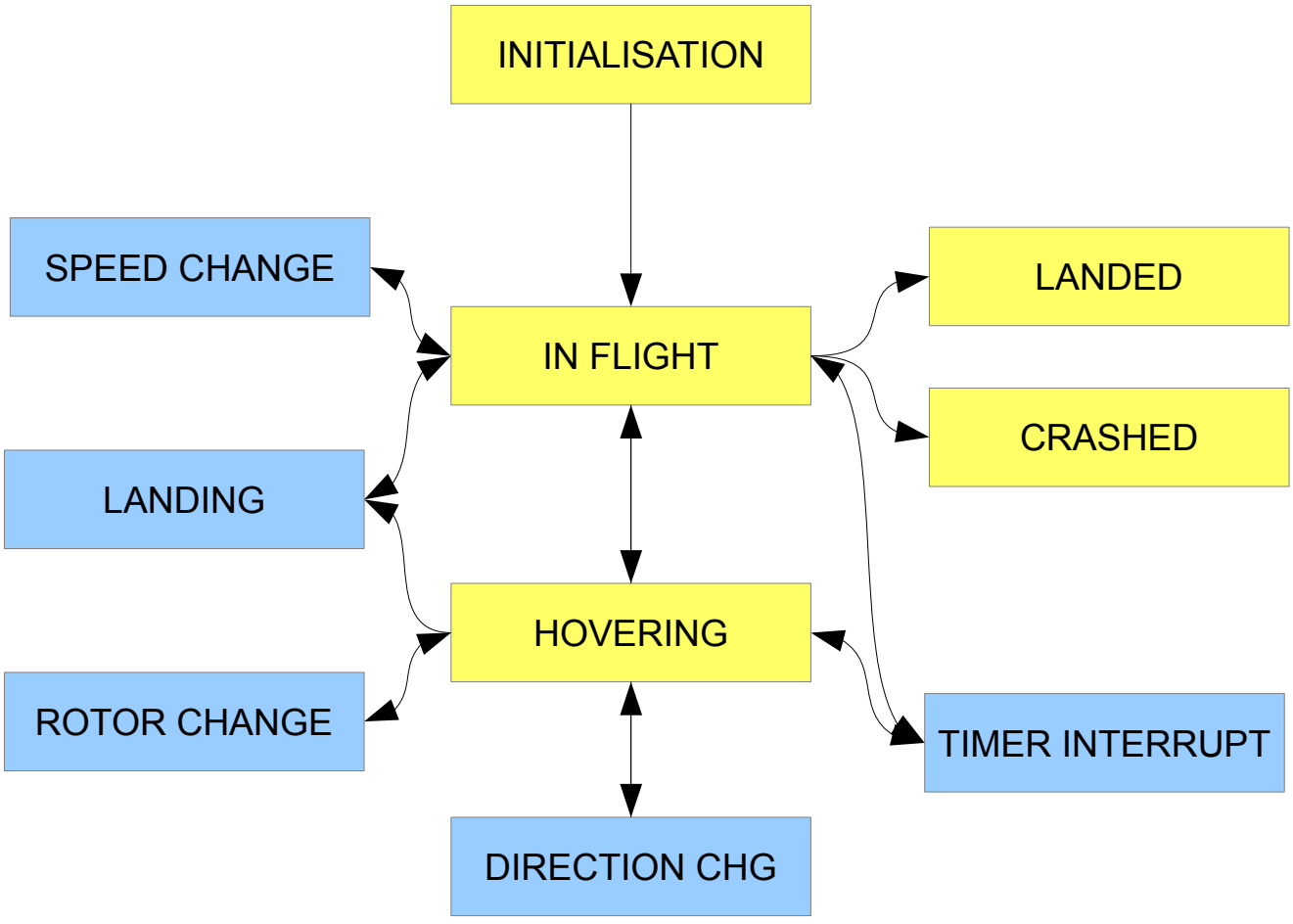
Author: Harold Jacob Hoare
StudentID: z5021639
Date: Semester 2, 2014

## 1. System Control Flow

There are 5 main states of the system :

- Initilaisation. When the '#' key is pressed the state changes to In Flight.
- In Flight. Speed can be changed. When the '*' key is pressed or the speed is reduced to zero the helicopter enters Hovering state.
- Hovering. Direction and rotor speed can be changed. When the key '*' is pressed the state changes back to In Flight.
- Landed. If the floor is reached at 2m/s or less. During flight or hovering the '#' key causes the helicopter to fly downwards at 1m/s so will land if no other commands are given.
- Crashed. If the ceiling or walls are hit, or the floor is hit at 3m/s or more.

Timer interrupts occur 450 times per second. Every 45 interrupts (0.1 seconds) the position is recalculated and checks to identify landing or crashing are performed. Every 0.5 seconds the LCD display is refreshed.
The diagram below shows states that can exist permanently in yellow and temporary states in blue.



## 2. Data Structures

The variables that describe the state of the helicopter are stored in data memory as 1 byte integers.

- (x_last, y_last, z_last) is the position vector of the helicopter at the last speed or direction change. They are integer values representing a number of meters.
- speed is an integer from 0 to 4 in m/s.
- direction is an integer from 1 to 6.
- (x_new, y_new, z_new) is the current position of the helicopter.
- flight_time is an integer representing the time in seconds since takeoff.
- flight_distance is an integer representing the distance travelled in meters since takeoff.

Registers are assigned as follows.

- r16 & r17 - del_lo & del_hi. 2 byte integer used to count loops in the 'delay' macro.
- r18 - row. The row of the keypad scan.
- r19 - col. The column of the keypad scan.
- r20 - rmask. Mask for the row keypad scan.
- r21 - cmask. Mask for the column keypad scan.
- r22 - timer_counter. Counts the number of timer2 interrupts for the position refresh.
- r23 - tenth_seconds. Counts ther number of 0.1 seconds for the LCD refresh.
- r24 & r25 - temp1 & temp2. Multi-purpose registers for temporary data transfer.
- r26 & r27 - path_time_l and _h. The number of 0.1 seconds since the last speed or direction change.
- Register pairs Y and Z are used to for memory addreses when transferring data.
- r0, r1 and r15 are used for temporary storage.

Strings used for the LCD display are stored in program memory.

5 lists of static data are defined with the .equ directive.

- LCD control paramaters.
- Room dimensions (maximum and minimum x, y and z co-ordinates).
- 3 bit (numbers 1 to 6) codes for the flight direction.
- Masks for scanning the keypad.
- Motor duty cycle control parameters.

## 3. Algorithms

The state of the helicopter is determined by the position at the start of the current path plus the distance travelled on the current path. The path is the travel since the previous speed or direction change.
The position at the start of the current path is stored in data memory as (x_last, y_last, z_last). The 2 byte register pair path_time counts the time in 0.1 second increments since the last speed or direction change. This time multiplied by the speed and divided by 10 is the path distance in meters. The direction of travel determines whether the path distance is added to or subtracted from the x, y or z co-ordinate to refresh the position.
The data memory variables (x_new, y_new, z_new) store the updated helicopter location. This is compared to the room boundaries every 0.1 seconds to determine whether the helicopter has crashed or landed. Every 0.5 seconds the position is updated on the LCD display.
Whenever a path ends due to a speed or direction change the vecor (x_new, y_new, z_new) is copied over to become the new (x_last, y_last, z_last). The total flight_distance is also updated.

All keypad and pushbutton presses are identified by polling. Which pins are being polled and their actions depends upon the helicopter state.
Before takeoff only the '#' key is polled. During flight PB0, PB1, '*' and '#' are polled. During hovering the the first 3 columns of the keypad and the pushbuttons are polled.
If the pushbuttons cause a speed change there is a delay of 0.2 seconds introduced before thie takes effect.

The rotor is simulated during flight by setting the motor duty factor to be equal to 50 * (1 + speed). Whilst hovering the pushbuttons increase or decrease the duty factor between 100 and 250 in steps of 50. When flight resumes this duty factor is converted back to determine the new flight speed.

The helicopter variable ranges can be summarised as follows :

| | |
|---|---|
| Position (x,y,z) | 0 to 50 (x and y), 0 to 10 (z) in steps of 1 meter |
| Path time | 0 to 65535 in steps of 0.1 seconds |
| Speed | 0 to 4 in steps of 1m/s |
| Flight distance | 0 to 255 in steps of 1 meter |
| Flight time | 0 to 255 in steps of 1 second |
| Duty factor | 100 to 250 in steps of 50 (unless 0 if pre-start, crash or land). |

## 4. Module Specification

The main code is organised as follows:

Reset

Initilaise stack.
Set port input and output directions.
Initialise the LCD and display the start message.
Poll the '#' key for takeoff.
Initialise Timer0 PWM for motor control.
Initialise helicopter position, speed and direction.
Display starting helicopter variables.
Initialise Timer2 interrupt with prescale of 64 so interrupts occur every
64*256 / 7.3728 MHz = 2.22 milliseconds = 450 per second.

Main (flight)

Poll the '*' key, indicating hover.
Poll the '#' key to commence landing.
Poll PB0 to increase the flight speed.
Poll PB1 to decrease the flight speed.
Repeat.

Main (hovering)

Poll PB0 to increase the rotor speed.
Poll PB1 to decrease the rotor speed
Poll keypad for a new direction setting, landing or to resume flight.
Repeat.

Timer Interrupt

Push registers to stack.
If 0.1 seconds has passed update the position and test for crash or landing.
Crash - Calculate and display the position. Flash the LED bar.
Landed - Calculate and display the total time and distance.
If 0.5 seconds has passed update the LCD display.
Pop registers from stack.

Specific module descriptions are given in the tables below. Please see comments in the code for more detail.

| **Macro** | **Description** | **Input Data** | **Output Data** | **Registers amended** |
|---|---|---|---|---|
| delay | creates a 10.308 microsecond delay (76 clock cycles) used for LCD setup and LED flashing. | del_lo, del_hi | | temp2 |
| lcd_write_com | write a command to the LCD (RS = 0, RW = 0) | temp2 | | temp1 |
| lcd_write_data | write data to the LCD (RS = 1, RW = 0) | temp2 | | temp1 |
| lcd_wait_busy | wait until busy flag is clear so data can | | | temp1 |

| | | | | |
|---|---|---|---|---|
| | be written to the LCD | | | |
| lcd_initialise | delay then function set repeated 3 times. 2 line display, 5x7 font and no cursor. | | | temp1, temp2 |
| lcd_display | displays a string of text on the LCD | ZH:ZL | | temp2 |
| display_new_ position | writes the latest position to the LCD including brackets and commas | | | temp1, temp2 |
| display_dir_speed | writes the latest direction and speed to the LCD | | | temp1, temp2 |
| display_triple _digit | display a triple digit integer on the LCD. leading zeros are not displayed. | temp1 | | temp2, r0, r1, r15 |
| scan_hash | poll the keypad for the '#' key | | | temp1, cmask |
| takeoff_position | initialise position, speed, direction and timers at takeoff | | | temp1 |
| path_distance | multiplies speed by path time in 0.1s and divides by 10 | | r0 | temp1, temp2, r1, r2 |
| copy_positions | transfers last position vector to new position vector | | | temp1, temp2 |
| copy_back | transfers new position vector to last position vector | | | temp1, temp2 |
| update_new_ positions | adds the path distance in the current direction to the new position | r0 | | temp1 |
| update_total_ distance | adds the path distance to the total flight distance | r0 | | temp1 |
| fly_to_land | sets the speed to 1m/s and flight direction to downwards | | | temp1, temp2, r0 |

| **Function** | **Description** | **Input Data** | **Output Data** | **Registers amended** |
|---|---|---|---|---|
| divide_by_10 | divides a 2 byte number by 10 and outputs the 1 byte result and 1 byte remainder | temp2, temp1 | r0, r1 | |