COMP9032 Project

Design Manual

Kun Zhang (5086704)

2015/10/26

Content

Abstract

This document records the design details of the project for COMP9032. It firstly specifies the requirements of the project then illustrates the control flow of the program with explanation of each module. In addition, it includes several timing diagram to show how tasks are scheduled.

1. Introduction

The project is to design an emulator which remotely controls a toy helicopter flying in a confined space. The simulation is done on COM9032 Lab board which is based on AVR ATMega 2560 microcontroller and codes for the system are written in AVR assembly code.

2. Design Specification

2.1 Operating Requirement

The area of the space is 50*50*10 metres and the helicopter starts at location (25, 25, 0)

2.1.1 Inputs

The keys from key pad function as shown below

1 - Down	2 - Forward	3 - Up	A
4 - Left	5 - Backward	6 - Right	B
7	8	9	C - Speed Up
* - Hover/Resume	0	# - Take off/Land	D - Speed Down

- Key 1, 2, 3, 4, 5 and 6 are direction controls.
- Key C speeds up the helicopter and key D slows it down. The speed must be one of the stepped speeds: 1m/s, 2m/s, 3m/s, and 4m/s.
- Key # is used for taking off and landing. When # is first pressed, the helicopter goes upward at 2m/s. It will head down at 1m/s until it lands on the ground, if # is pressed for a second time.
- Key * is used for hovering and resuming. When * is first pressed, the helicopter doesn't change its location until * is pressed again when the flight resumes the previous flying state (speed and direction)

2.1.2 Outputs

- When the simulation starts, the string "Start:" should appear until # is pressed.
- When the helicopter is flying, the location, current direction and speed of the helicopters are shown on the LCD as shown below

```
      PO          D     SPD
(25,  25,  00)    U       2
```

where PO, D and SPD represents current position, direction and speed, respectively.

- The motor spinning speed positively correlates to simulated speed.

- When the helicopter lands safely, the total travelled distance and time is displayed on LCD.
- For crashes, LCD shows the location of crashing site and LED bar flashes.

2.2 Hardware

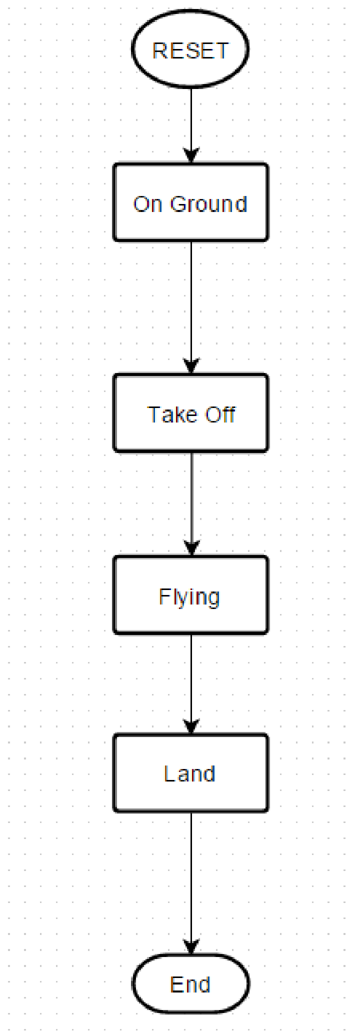
For the purpose of this project, we use COM9032 Lab board which is powered by 5V USB port or DC power supply. The wiring of the lab board can be found in Appendix I as well as User Manual.

2.3 Software

All codes are written in AVR assembly with a code size smaller than programming memory size (256kB).

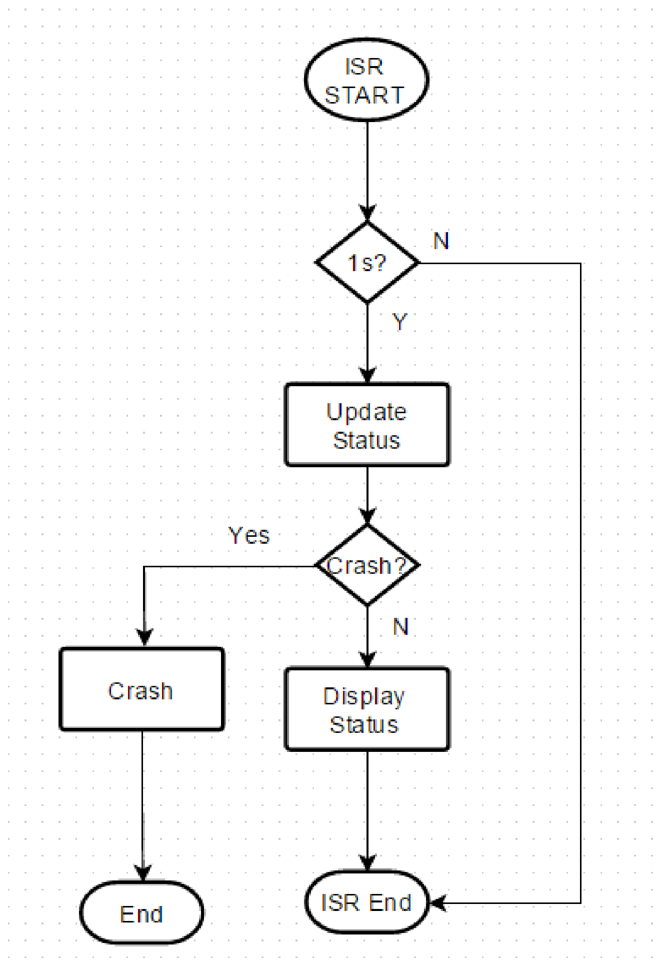
3. Control Flow and Modules

The overall system flow is shown below



After RESET, *On Ground* module runs until hash key is pressed. Then, it proceeds to *Take Off* and *Flying* module which loops if hash key is pressed again. Next, the *Land* module runs which display total travel time and distance.

Alongside of this main process, Timer3 overflow interrupt has been used for counting seconds, update the status of the helicopter and display its status. The Timer3 overflow interrupt service routine (ISR) is shown



The ISR will be called every 4 us when the counter overflows and causes an interrupt. A counter is used to determine whether 1 second has elapsed. If it is true, the *Update Status* module will update position, total time and total distance. To update the position, we simply add the current speed to current coordinate with respect to current direction, because the current speed has the same value as the distance traveled within the 1 second elapsed. For example, if the current direction is right with a speed of 4 m/s, we will add – 4 to current y coordinate to get the new position. In *Update*, we will check the new position to see if it is out of our confined space.

If it is still within bound, *Display Status* shows the current position, status and speed by reading “content” variable. If it is out of bound, we know it has crashed, and *Crash* displays the position of its crashed site. The crash site has to be a position within the confined space. So the Crash module determines which wall of the space the helicopter crashes and displays the position that is one that wall. For example, if the helicopter’s current position is evaluated to be (25, 25, 12) we will display the crash position to be (25, 25, 10)

3.1 RESET

The RESET module is called upon power on or the reset button is pressed. It sets up the stack pointers, ports for LCD, keypad and LED and initializes LCD followed by displaying “START:”

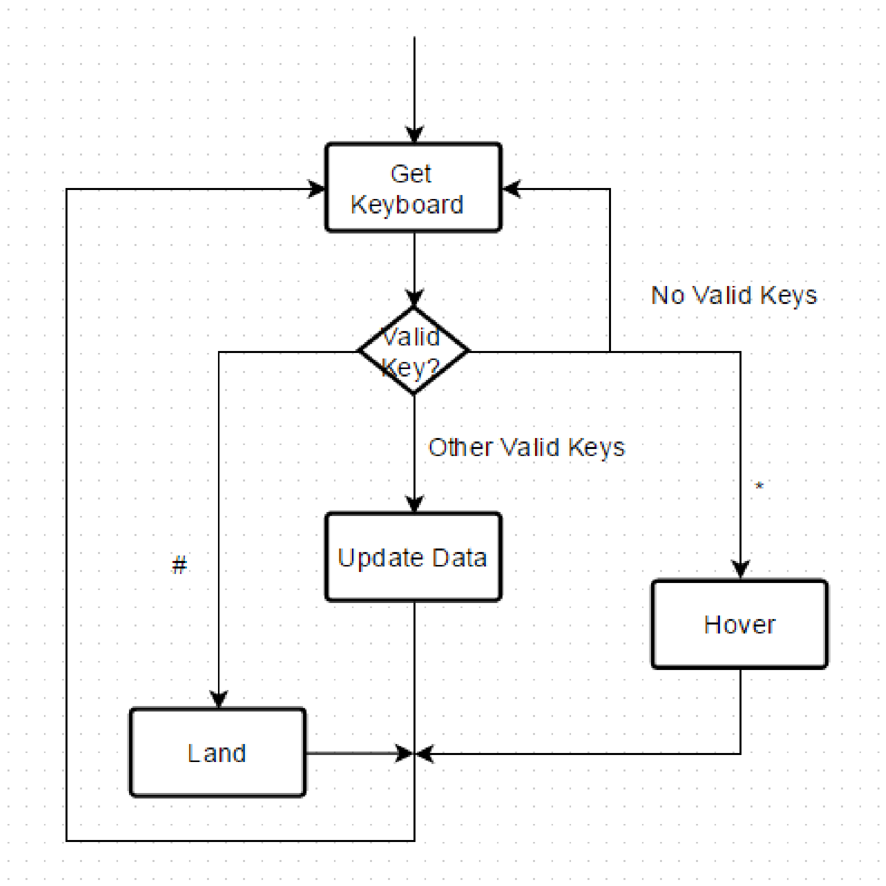
3.2 On Ground

The On Ground module simply gets keypad input. It proceeds to Take Off module upon receiving a hash key input.

3.3 Take Off

The Take Off module will save the initial position (25, 25, 0) into memory. In addition, it sets up Timer3 and enables global interrupt. We utilize two functions of Timer3: overflow interrupt and Phase Correct PWM generation. The overflow interrupt is used as 1 second timer and PWM will drive the motor.

3.4 Flying



The Flying module will end up with different modules depending on keypad input as shown above.

3.4.1 Land

If # key is received, the module saves current direction to downward, changes the speed to 1 and set the landing flag. The landing flag is used in Update module in Timer3 ISR. If the flag is set and the helicopter is still above ground, the ISR will display current status as usual. If the helicopter touches the ground, instead of going to Crash module, it will go to Land Display module which fetches the total travel distance and time and display them on LCD.

3.4.2 Hover

The Hover module executes upon * key is pressed. The hover module sets hovering flag which lets Update module in ISR know that it does not need to update any values. It will also change the direction to “H” indicating hovering status.

3.4.3 Update Data

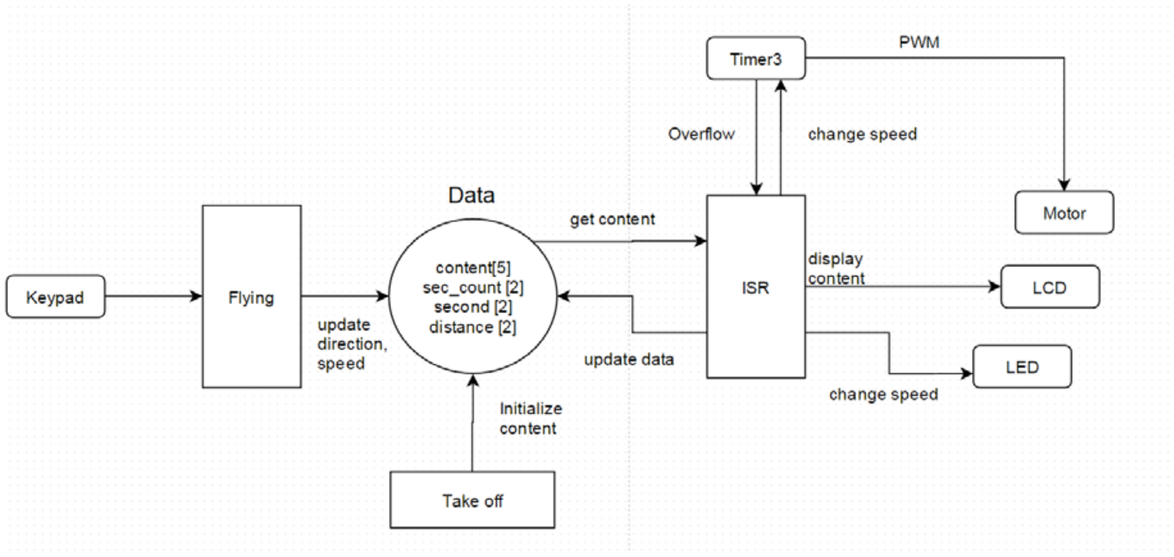
Update Data module saves updated data in memory. The updated data are stored in “content” variable as discussed below. After all data having been updated, the program will loop back to get keyboard input.

4. Data Structures

As illustrated below, the data in memory are five variables. “content” variable is with a length of 5 byte arranged as [x, y, z, direction, speed] where x, y and z are unsigned integers representing current location, speed is an unsigned integer and direction is a character; “sec_count” is a word variable which stores counters used to determine whether 1 second has elapsed. “second” and “distance” are both word variables that record total time in second and total distance in metres.

The ISR is triggered by Timer3 overflow interrupt that happens approximately every four microsecond. Timer3 also generates PWM on compare match. The compare match TOP value can be updated by ISR.

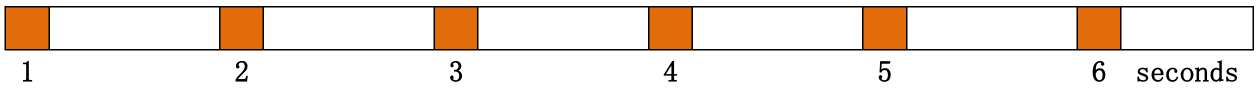
Flying module updates direction and speed in content variable upon Keypad input. In ISR, the sec_count will increment if it has reached a pre-determined value, or reset (be cleared to zero) otherwise. The Update module will calculate and update location values in content variables and check if the location is viable. If not, it will trigger LED to flash three times. Otherwise, the information in content variable will be displayed on LCD. In addition, the Update module will also change the TOP value for Timer3 so as to change the Motor’s speed.



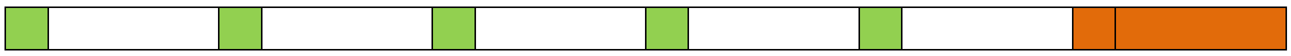
5. Timing Diagram

5.1 Basic schedule

There are two basic tasks: flying (white) and ISR when 1 second has elapsed (orange), which is allocated as below (the length of blocks are only indicative)

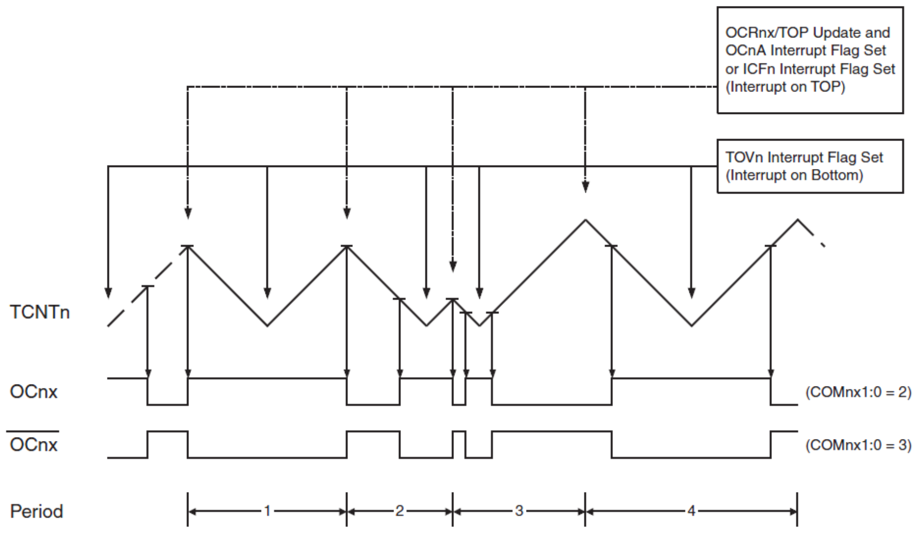


Indeed the ISR will occurs every 4 microsecond, if we magnify the period around the onset of orange block, we have a timing diagram as shown



The green blocks indicate ISR when the time is not 1 whole second. They are separated by flying (white) with an interval of 4 us. The orange is ISR when 1 second has elapsed (orange) which is longer than its green counterpart for it execute more codes.

5.2 Timer schedule


OCRnx/TOP Update and OCnA Interrupt Flag Set or ICFn Interrupt Flag Set (Interrupt on TOP)
TOVn Interrupt Flag Set (Interrupt on Bottom)
TCNTn
OCnx
(COMnx1:0 = 2)
OCnx
(COMnx1:0 = 3)
Period
1
2
3
4

The above timing diagram for Timer3 is excerpted from [1]. In the program, we use Timer Overflow (TOV) Interrupt as well as compare match (OCRA) interrupt. As can be seen from TOV interrupt happens at the bottom of the triangle wave whereas OCRA interrupt happens at the top. This indicates that these two interrupts will not overlap which ensures the timing function of the time will be rather accurate.

6. Conclusion

The project uses rather complex program in AVR assembly to simulate a controller for a toy helicopter. We have shown that the how the program is formulated and implemented. In fact, the program worked well on the lab board during demonstration. Further development is required if it is to be used in real remote controllers for toys

Reference

[1] Atmel Corporation, Atmel ATmega640/V-1280/V-1281/V-2560/V-2561/V Datasheet, San Jose, CA, USA, 2014