0.1 Module Requirements: Module LCD

This section looks at the hardware interfacing of the LCD in both setting it up for output and writing strings to it. Most sections will consist of a 'Hardware' subsection to give detail of the relevant low level hardware interfacing that occurs through software and hardware, followed by 'Writing', which concerns the higher level software that is involved with creating the user output on the LCD, and will make extensive reference to the LCD_disp(x,y) function.

LCD_disp(x,y) is a Menu mode function written that takes in the current status (namely menu_ref_1 and menu_ref_2) and prints an appropriate message to the LCD.

0.1.1 Reference Document

Much of the initialisation process is informed by these external sources and libraries:

- 'LCD Initialization' Donald Weiman 2012, Alfred State.
- 'Interfacing LCD with PIC Microcontroller', Ligo George 2014, Electro-Some.

The main goal of this module is the appropriate implementation of the LCD with the system and determining how to achieve LCD functionality given the existing platform of information and the hardware we are given.

0.1.2 Functional Requirements

The LCD should gives immediate feedback and information to user of current status of system that is relevant to the context of operation. It does so with little latency and maximum clarity and understandability. In the interest of saving pins, the system uses the LCD in 4-bit mode.

Inputs The LCD hardware consists of 14-inputs given as follows:

- 1 - Ground
- 2 - Vcc
- 3 - Contrast - Connected to output of 10KOhm potentiometer connected to Vcc
- 4 - RS Register Select - 0 for LCD instruction; 1 for data read/write; PORTDbits.RD2
- 5 - RW Read/Write - Always set to ground as we never read (0 = Write)
- 6 - Enable; PORTDbits.RD3
- 7-10 - Data bits 0-3; unused in 4-bit mode

- 11-14 - Data bits 4-7; Connected to PORTDbits.RD4-7

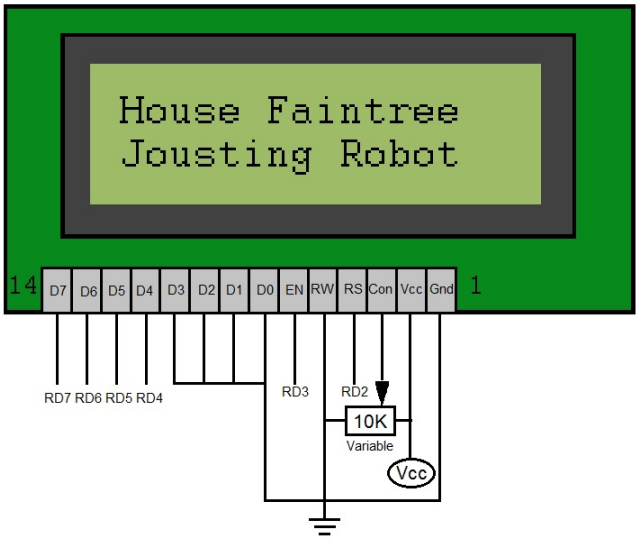

Figure 1: LCD inputs

These hardware inputs are interfaced by software commands and data through Port D by functions such as LCD_init and writing functions. The main function that uses the LCD to display a message, LCD_write_string, takes in a string, while the function LCD_disp (which utilises LCD_write_string)takes in two arguments relating to the status of the commander, menu_ref_1 and menu_ref_2.

Process and Output The LCD takes two forms of signals, Commands and Data. Commands are instructions to the LCD that do tasks such as Clear Display, Reset, and Set Cursor and are defined by a 0 value to RS, and nibbles written to RD4-RD7 corresponding to individual commands. Data inputs are read as characters to be written to the LCD and are defined by a 1 value to RS and corresponding nibble to RD4-RD7. EN in both cases is toggled on and off for the LCD to capture the input.

Before sending data to the LCD, there is some level of parsing done through LCD_disp. Through menu_ref_1 it determines the current run-time mode and writes it to the first line. Through menu_ref_2, it determines the submenu and displays it in an appropriate manner on the second line:

- If a submenu is to be displayed, it is displayed with the label of what

it is and its current value. For example, the second line will appear as `Submenu:Value`

- The submenu label string is stored in an array of strings in `stringtab[ ]` and the respective value is stored in an array of values in `values[ ]`
- Max Speed, Yaw Rate and similar are displayed as percentages; this was determined to be the most intuitive way of presenting this data to users
- IR Samples/Estimate, IR Sample Rate and similar are displayed as integers or actual values
- No value is displayed on the second line when in the Menu Mode of Assisted and Auto, as per specifications

Timing For proper initialisation of the LCD, specific delays must be implemented with each initialisation command. This is covered in greater detail in the Conceptual Design.

Post Initialisation, the LCD should have no noticeable latency that is attributed to computing latency, nor any timing requirements for basic functionality.

0.1.3 Non-Function (Quality of Service) Requirements

LCD in being the main source of data and feedback to the user in implementing desired behaviour should behave in a manner that is comfortable for use.

Performance The LCD was determined to only need to be updated (which is a process that requires a clear and then a write) when a variable has actually changed. Initial tests found that constant updating created flickering. Current means of updating LCD results only in a subtle blanking before a new write; again only occurring if a variable were to change.

Design Constraints Care should be (and was) exercised when moving software between different processors with different internal clocks. As will be outlined, the function of the LCD depends heavily on correct initialisation which in turn depends heavily on correct settings for delays between commands. Neglecting this will most likely lead to non-function in migration.

0.2 Conceptual Design: LCD

0.2.1 Assumptions Made

The necessary initialisation sequence was based largely off a pre-existing methodology of initialising the LCD; data sheet provides little information of subtle startup commands necessary for initialisation. This is shown in the following flow diagram and is accomplished through LCD_init( ).

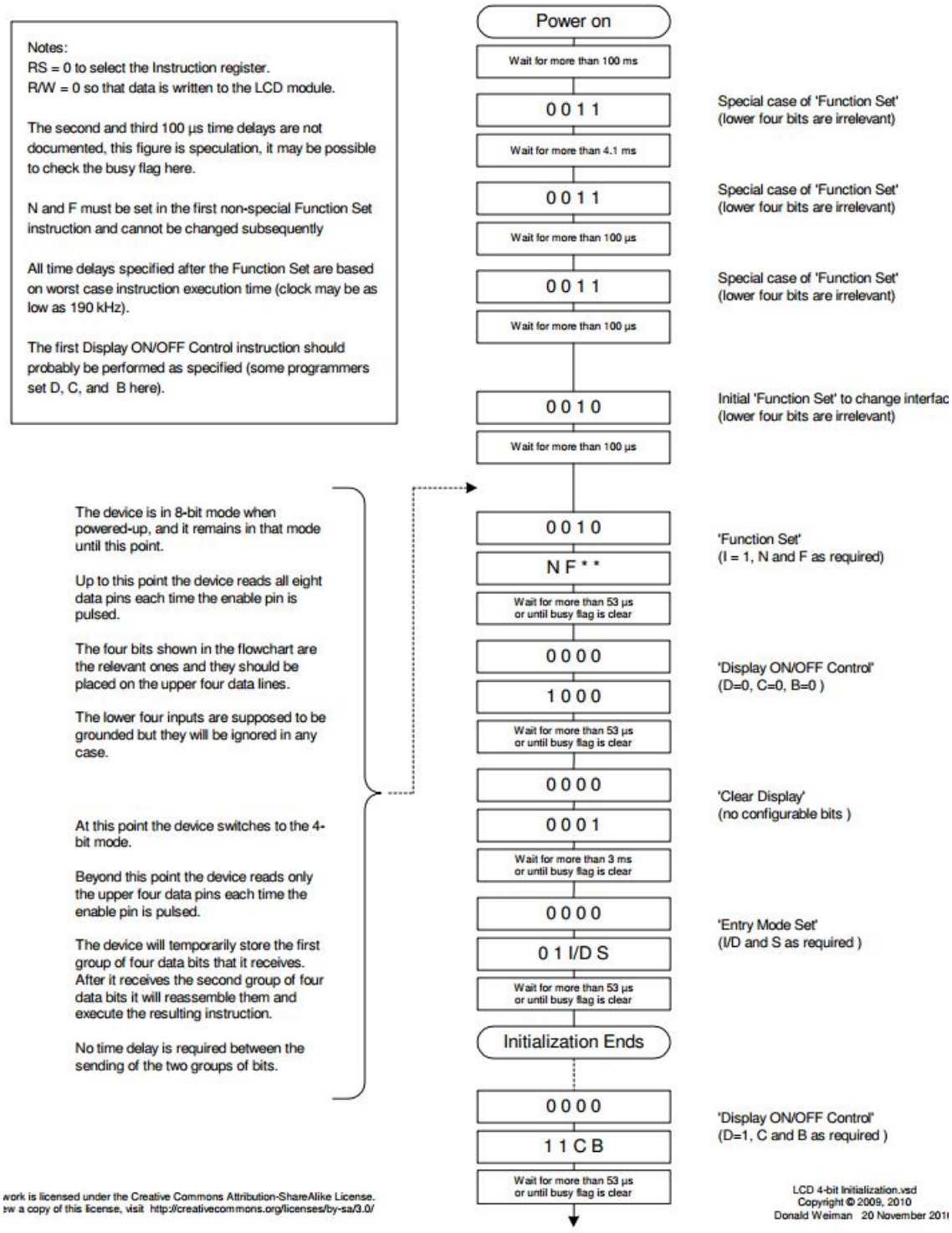

Figure 2: LCD init data flow

0.2.2 Minor LCD command functions

`LCD_Port(char data)` This function was the explicit means of setting the data pins given a char input. It takes the least significant bits and moves and sets the MSBs of the data bits on LCD (as the LSBs are grounded for 4-bit functionality).

`LCD_Cmd(char a)` This function sends commands by setting RS to 0 for LCD to anticipate a command, use `LCD_Port` to set data bits, EN to 1 to capture command, delaying momentarily, before toggling EN off again.

`LCD_Clear(void)` Simply uses `LCD_Cmd` to clear LCD with the clear command (0 then 1).

`LCD_Set_Cursor(char row, char col)` This function takes in two arguments, row and col and sets the cursor position. As their names may suggest, row determines row or line the cursor is moved to and col determines the column. After parsing arguments, function sends command to LCD.

0.2.3 LCD writing functions