University of
HUDDERSFIELD

Name: Sibtain Ali

Student number: U1970762

Contents

1 Background:

This project aims to demonstrate skills for developing and maintaining an embedded system. This project was based on the PIC16F877A microcontroller. In the bigger picture, working code for a clock and a temperature monitoring system was written in the C language.

1.1 Environment used:

The Code was run on MpLabX for the IDE and Proteus was used to simulate the hardware which largely included 18 LEDs, 16 buttons spread across the PORTB and PORTD for I/O, an analogue to digital convertor was used along with a potentiometer, a 16*2 LCD display and the PIC16F877A itself running on a quartz crystal with 3.2768 MHz clock rate. A feature table for the controller is given below:

Program Memory		Data SRAM (Bytes)	EEPROM (Bytes)	I/O	10-bit A/D (ch)	CCP (PWM)	MSSP		USART	Timers 8/16-bit	Comparators
Bytes	# Single Word Instructions						SPI	Master I^2C			
14.3K	8192	368	256	33	8	2	Yes	Yes	Yes	2/1	2

2 Functionality:

2.1 Clock

The main functionality was to achieve a working clock. The developed clock can have two working modes: 24- hour format and 12-hour format. This was achieved using the Timer 1 module on the microcontroller by calculating overflow time and the number of required overflows. The implemented delay and the actual delay on running the simulation may differ. This could be because of the hardware running on a simulation or other factors. This could be easily avoided by modifying the Pre-scalar and Post-scalar values to run in a much more accurate manner, however this was not executed to keep the code functioning normally.

2.2 Weather station

This was introduced as the secondary function. It was achieved using an Analogue-to-Digital convertor and a potentiometer to simulate a temperature sensor. This can run in two modes, Celsius and Fahrenheit.

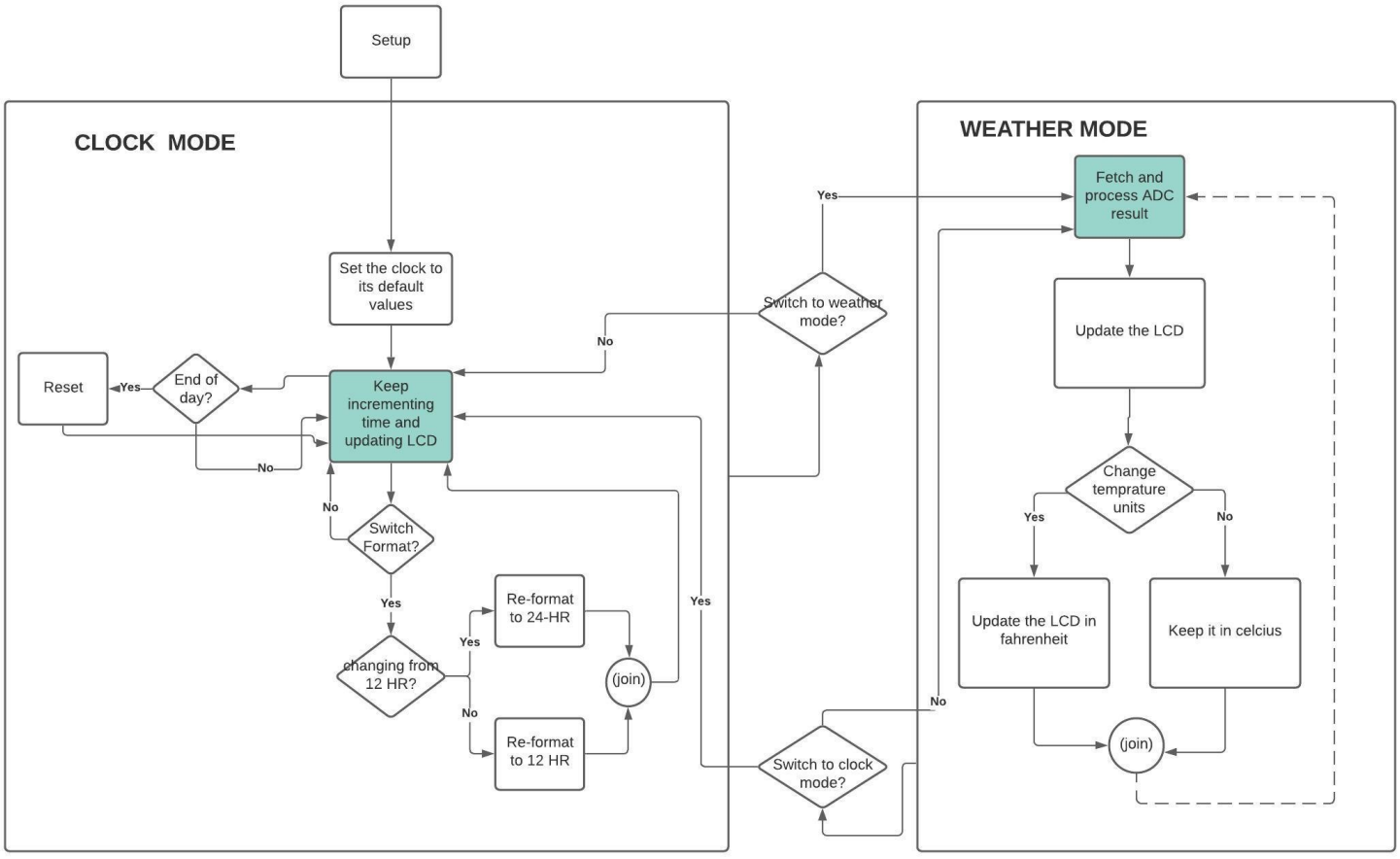
2.3 Frost Alarm

This function demonstrated the use of interrupts, where, when the temperature dropped below 4^o C, the LED on RB7 would toggle every 250ms to simulate an alarm. This can be triggered from any mode on the embedded system and would resume once the temperature would raise above the specified limit.

3 The code:

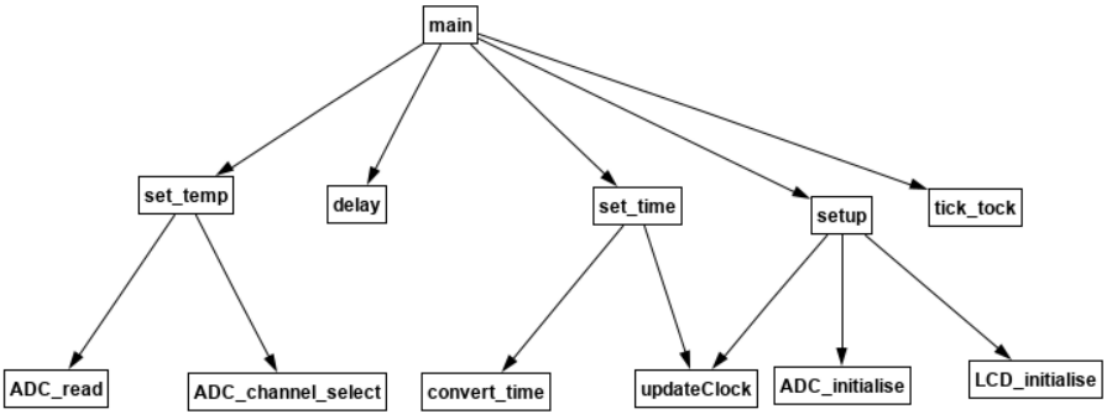
The code can be represented in two basic states, clock-mode and weather-mode, both of these states can switch between each other at any given time. Both can be condensed into an activity diagram show below:

(The states in green are constantly updating values regardless of what mode the embedded system is in)



3.1 Main function

This has been broken down into 5 basic methods, including a delay method to keep the program running forever. The call graph for the same is given below:



3.2 Code Structure

- The main function has been kept as “clean” with almost no code outside the mentioned methods. This was done to improve readability and understanding.
- The variables are global were used according to the required datatypes.
- Every function has a description and commenting is used to add to the documentation, focusing on the registry code.
- Multi parameter functions were used where required.
- Functions are reused wherever possible.

4 Efficiency:

The code runs with a maximum CPU usage of 5% on an intel i7-8750H. Booleans were used where possible. The declaration of variable datatypes was prioritised to *char* focusing on minimising the space taken up by the program. 76% of program space and 41% of data space was used.

5 Short comings:

Only the suggested used of interrupts is shown, a much more extensive use of interrupts can be used. The clock stops to display anything while in the interrupt service routine which can be improved.