

Generating a program for a specified embedded requirement in C.

by Adam Ibrahim

Submission date: 17-Apr-2026 02:47AM (UTC+0100)

Submission ID: 280019301

File name:

57247_Adam_Ibrahim_Generating_a_program_for_a_specified_embedded_requirement_in_C_728569_364913870.docx
(1.07M)

Word count: 2098

Character count: 11106



Prifysgol Cymru

Y Drindod Dewi Sant

University of Wales

Trinity Saint David

**Programme: B.Eng. (Hons) Electrical and Electronic
Engineering**

Module: Embedded C

Module code: ACDA4013

Student name: Adam Ibrahim

Student ID Number: 2406034

Table of Contents

Introduction	2
Background Research	2
Embedded Systems.....	2
PIC16F877A Microcontroller	2
LCD Display (16x2)	3
UART Communication	3
DC Motor (Cooling Fan Simulation).....	3
Components Used	3
System Operation	3
Circuit Diagram	4
System Design / Circuit Design.....	4
Program Implementation	5
Full Code Explanation	5
Pseudocode.....	7
Pseudocode.....	7
Simulation Screenshots	9
Conclusion.....	12
Reference's.....	13

Introduction

Many electronic and mechanical applications use a cooling system to keep parts at an optimal operating temperature. These automated cooling systems monitor the status of the cooling system and automatically turn on cooler units that prevent components from burning out. Our cooling systems improve safety, efficiency and reliability.



This project explains how to design and simulate Cooling System Status Simulator using PIC16F877A microcontroller. The system consists of various states and functions like start, check coolant, turn ON fan, stop and so on. Push buttons are used as sensors. Feedback to the user is provided through LEDs on the front panel, an LCD which displays messages such as the current processing stage and feedback through the virtual terminal in simulation. All the system's behaviour is observable in real time. The system was designed and assessed using Proteus simulation software. The software program of the system was written using CCS C language.

Background Research

Embedded Systems

An embedded system is a small computer system with a fixed set of instructions and is designed to perform a specific function. Unlike general-purpose computers, embedded systems are typically configured to perform a subset of the tasks that a regular computer can perform. Embedded systems are everywhere. You will find them embedded in a range of products such as modern vehicles, washing machines, dryers and any type of industrial plant or machine [1].

PIC16F877A Microcontroller

The PIC16F877A is an 8-bit microcontroller widely used in embedded system applications. It contains input/output ports, timers, analogue-to-digital converters, and communication modules such as UART. It is suitable for this project because it can easily interface with switches, LEDs, LCDs, and motors [2].

LCD Display (16x2)

A 16x2 LCD display is used to show messages to the user. It can display sixteen characters per line across two lines. In this project, the LCD operates in 4-bit mode, which reduces the number of pins required for communication with the microcontroller.

UART Communication

UART communication is used in a lot of actual world designs to communicate between various parts of a design. In this project we will use UART communication to transfer system messages (messages created by switch statements) to the simulation software called Proteus. There we can monitor the messages in a virtual console [1].

DC Motor (Cooling Fan Simulation)

A DC motor is used to simulate a radiator fan in an automotive cooling system. The motor is activated when the system detects that cooling is required, representing real-world fan operation in vehicles.

Components Used

- PIC16F877A Microcontroller
- 3 Push Buttons (SW1, SW2, SW3)
- 3 LEDs (LED1, LED2, LED3)
- 16x2 LCD Display
- DC Motor
- Resistors
- Power Supply
- Virtual Terminal (UART)



System Operation

The system operates in a sequence:

1. The system starts in a ready state.
2. Pressing SW1 starts the engine and turns on LED1.
3. Pressing SW2 confirms coolant status and turns on LED2.
4. Pressing SW3 activates the fan and runs the motor.
5. The motor runs for 3 seconds.

6. The alert LED blinks three times.
7. The system shuts down.

Each stage is displayed on the LCD and sent to the UART terminal.

Circuit Diagram

System Design / Circuit Design

This project simulates Engine Coolant Temperature System using PIC16F877A microcontroller. The circuit also includes a 5v dc voltage source with excellent quality earth for power supply. A crystal oscillator with 8mhz crystal is used for microcontroller. The MCLR/VPP pin is connected through a 10k ohm resistor to VDD for reset function. Three push buttons are connected between VDD and ground through 10k ohm pull down resistors for simulating engine start switch, coolant sensor switch and fan switch on. Three LEDs are used for displaying engine running, normal engine cooling, and overheating condition. The LEDs are connected through current limiting resistors of 220 ohms to corresponding output pins of microcontroller. A 16x2 LCD is interfacing is done using 4-bit data transfer method. The data pins D0-D7 of LCD are connected to PORTD (RD0-RD7) of PIC16F877A. The control pins E and RS of LCD are connected to PORTC (RC0-RC7) of PIC16F877A. A potentiometer is connected between VDD and GND via 10k ohm resistor for adjusting the LCD contrast. System messages will be displayed in LCD at all stages of system. UART communication is also implemented using TX (RC6) and RX (RC7) pins of PIC16F877A. The communication will be done with virtual terminal open in Proteus software. We also utilize a DC motor as our cooling fan, which is controlled off the microcontroller's output pins through a motor driver circuit. As with the rest of the lab, everything is tied to a ground point.

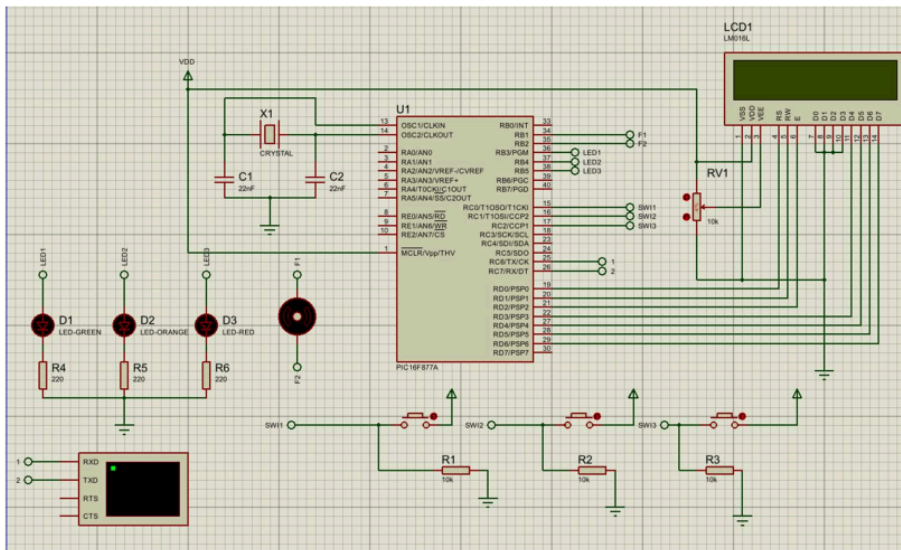


Figure 1: Circuit Diagram

Program Implementation

This program will read buttons and output to LEDs, LCD, DC motor, and UART. It will use branching conditions to step through the program in the correct order and delays to control timing. In this example the DC motor will run for 3 seconds, and the LEDs will blink on and off.

Full Code Explanation

The program begins with `#include <main.h>`, which imports the main configuration file for the **PIC16F877A** microcontroller. This file usually contains important setup details such as clock frequency, fuse settings, and device configuration required for the program to run correctly.

The line `#use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7)` configures UART communication. It sets the communication speed to 9600 bits per second and assigns pin RC6 as the transmit (TX) pin and RC7 as the receive (RX) pin. This allows the microcontroller to send system messages to the virtual terminal in Proteus.

The next set of lines uses `#define` statements to assign specific microcontroller pins to the LCD control and data lines. These include the enable pin, register select pin, read/write pin, and data pins D4 to D7. This configuration enables the LCD to operate in 4-bit mode, reducing the number of pins required for communication.

The line `#include <lcd.c>` includes a pre-written LCD library that provides functions such as `lcd_init()` and `lcd_putc()`. These functions simplify the process of displaying messages on the LCD without needing to write low-level control code.

The void `main()` function is the starting point of the program where all operations are executed. Inside this function, the first step is `lcd_init()`, which initialises the LCD display and prepares it for use [3].

The lines `set_tris_b(0b00000000)`, `set_tris_c(0b00000111)`, and `set_tris_d(0x00)` configure the direction of the microcontroller pins. `PORTB` is set as output for LEDs, `PORTC` has its first three pins configured as inputs for switches and the remaining pins as outputs for motor control and UART, and `PORTD` is set as output for the LCD.

The program then ensures a safe starting state by turning off all LEDs using `output_low(LED1)`, `output_low(LED2)`, and `output_low(LED3)`. Similarly, the motor is turned off by setting both motor control pins low using `output_low(MOTOR_IN1)` and `output_low(MOTOR_IN2)` [4].

Next, the system displays “System Ready” on the LCD using `lcd_putc("\fSystem Ready")`, where `\f` clears the display before printing. At the same time, the message is sent to the virtual terminal using `printf("System Ready\r\n")`. A short delay is added to allow the message to be clearly visible [5].

The program then enters a waiting state using `while(!input(SWI1))`, where it continuously checks if switch SW1 has been pressed. This ensures that the system only proceeds when the correct input is given. Once SW1 is pressed, LED1 is turned on to indicate the engine start stage, and the LCD and UART display the message “Engine Started.”

The program then waits for the second input using `while(!input(SWI2))`. When SW2 is pressed, LED2 is turned on to indicate coolant confirmation, and the system displays “Coolant OK” on both the LCD and the virtual terminal.

Next, the program waits for SW3 using `while(!input(SWI3))`. This input triggers the fan activation stage. Before starting the motor, the program ensures that previous LEDs are turned off. The LCD and UART then display “Motor Running” (and optionally “Fan Activated” if included).

The motor is activated by setting one control pin high and the other low using `output_high(MOTOR_IN1)` and `output_low(MOTOR_IN2)`. This causes the motor to rotate in one direction. The motor continues running for 3 seconds, controlled by the `delay_ms(3000)` function.

After the delay, the motor is stopped by setting both control pins low. The system then enters the alert phase, where the LCD displays “ALERT!” and the same message is sent via UART.

A for loop is used to blink LED3 three times. Inside the loop, the LED is turned on and off with a 100 ms delay between each state, creating a visible alert signal to indicate that the system is shutting down.

Finally, the system displays “Shutdown Done” on the LCD and sends the same message to the virtual terminal, indicating that the full operation cycle has been completed successfully.

The program ends with an infinite loop while(TRUE) which keeps the system in a stopped state. This prevents the program from restarting automatically unless modified, ensuring that the sequence only runs once per execution.

Pseudocode

Pseudocode

START

Display "System Ready" on LCD

Send "System Ready" via UART

WAIT until SW1 is pressed.

Turn ON LED1

Display "Engine Started" on LCD

Send "Engine Started" via UART

WAIT until SW2 is pressed.

Turn ON LED2

Display "Coolant OK" on LCD

Send "Coolant OK" via UART

WAIT until SW3 is pressed.

Turn ON DC Motor

Display " Motor Running " on LCD

Send "Motor Running" via UART

Delay for 3 seconds

REPEAT 3 times:

Turn ON LED3

Delay 100 ms

Turn OFF LED3

Delay 100 ms

Display "Shutdown Done" on LCD

Send "Shutdown Done" via UART

RETURN to ready state

END

Simulation Screenshots

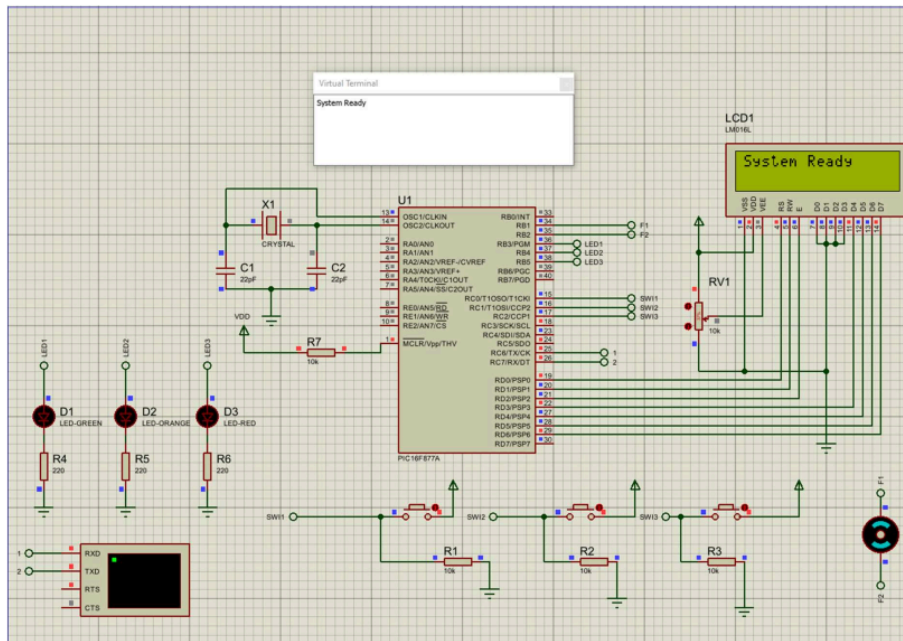


Figure 2: when program has run showing system Ready.

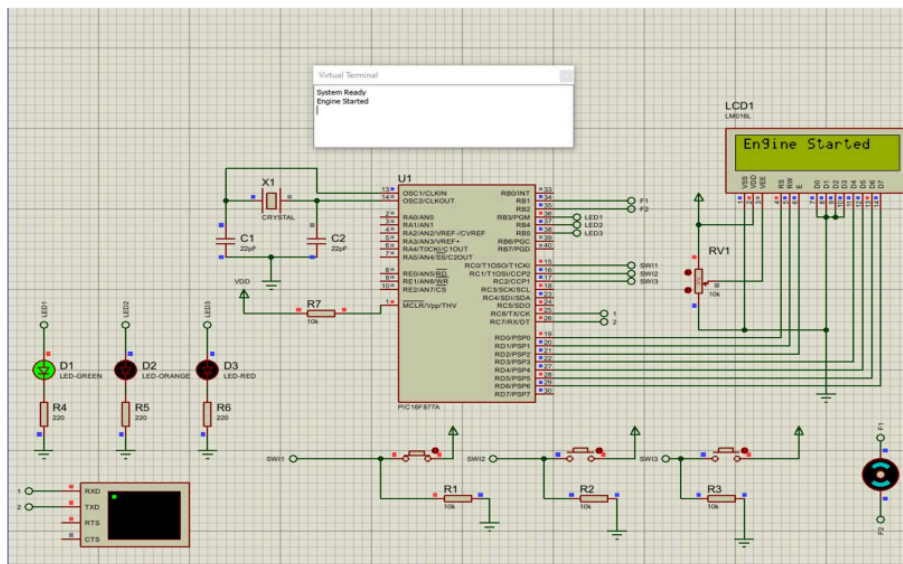
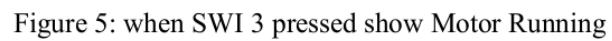
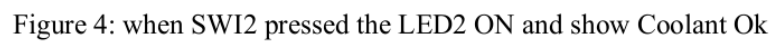


Figure 3: when SW11 pressed the LED1 ON and show Engine Started



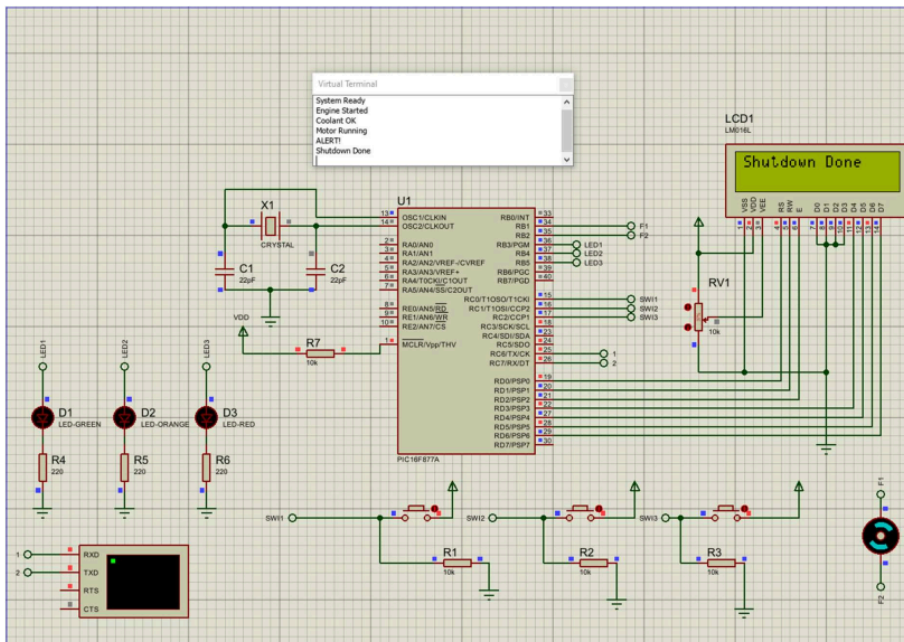


Figure 6: shows LED3 blinking three times and shutdown the system.

Testing and Results

Test Case	Input Action	Expected Output	Actual Output
1	Run program	System ready UART & LCD message	Same as expected
2	Press SW1	LED1 ON, LCD: Engine Started, UART message	Same as expected
3	Press SW2	LED2 ON, LCD: Coolant OK, UART message	Same as expected
4	Press SW3	Motor ON, LCD: Fan Activated	Same as expected
5	After 3 seconds	LED3 blinks three times	Same as expected
6	End of cycle	LCD: Shutdown Done, UART message	Same as expected

Conclusion

This automotive cooling system simulator program utilizes embedded system control to monitor and control multiple stages with a single microcontroller. The program monitors the start-up/shut-down engine states and fulfils user commands throughout these states. LEDs, LCD and UART are utilized to verify user commands and system states during all phases of operation.

OUTPUT OBSERVATION: The system does what it is supposed to do, responding to its input appropriately. The DC motor serves as the fan connected to the radiator. The alert LED turns ON as required and the delays are timed appropriately. The system responds to all of the other components (the push buttons, the alert LEDs, LCD, and the UART) as expected, allowing the microcontrollers to process real-time functions.

This project completes all requirements for the assignment and demonstrates well the processes of the automotive cooling system. Future improvements to the project would be to substitute the push buttons for appropriate temperature sensors and to implement automatic control rather than manual. Additionally, safety features could be implemented for further protection of the system and authenticity of the project.



Reference's

- [1] M. P. Bates, "USART," in *Programming 8-bit PIC Microcontrollers in C : With Interactive Hardware Simulation*, 2008, pp. 19-77.
- [2] J. S. F. L. J. L. W. W. a. L. G. Wenqiang Li, " μ AFL: Non-intrusive Feedback-driven Fuzzing for Microcontroller Firmware," New York, NY, USA, May 21–29, 2022.
- [3] T. B. J. r. B. a. F. P. L. homas Nu" rnberg, "An introductory microcontroller programming laboratory course for first-year students," *International Journal of Electrical Engineering Education*, pp. 1-15, 2016.
- [4] P. Marwedel, "Embedded Systems," in *Embedded System Design*, ISBN 978-3-030-60910-8 , 2021, pp. 21-161.
- [5] I. Gbati, "General-Purpose Timers (TIM)," in *Bare-Metal Embedded C Programming*, Birmingham, Packt Publishing Ltd., 2024, pp. 200-395.

Generating a program for a specified embedded requirement in C.

GRADEMARK REPORT

FINAL GRADE

GENERAL COMMENTS

67 /100

PAGE 1

PAGE 2

PAGE 3

 **Comment 1** |  Engagement with Literature(10)

A brief discussion of the cooling system is included, which is good. However, the concept of the cooling system should be explained in more detail.

 **Comment 2** |  Cognitive & Intellectual (25)

It is good to see brief details on the selection of the microcontroller. However, it is essential to explain why it was selected, including factors such as speed performance, low power consumption, and cost-effectiveness.

PAGE 4

 **Comment 3** |  Knowledge & Understanding(20)

The selection of components and their functions are briefly explained, which is good.

 **Comment 4** |  Cognitive & Intellectual (25)

The LED state sequence is provided clearly which is commendable.

**Comment 5** | Knowledge & Understanding(20)

The circuit construction is well explained, which is appreciable.

**Comment 6** | Knowledge & Understanding(20)

It is necessary to include CCS C structures such as header files, pin definitions, functions used, and the overall program structure.

**Comment 7** | Cognitive & Intellectual (25)

A good pseudocode is provided, which is appreciated.

**Comment 8** | Practical Skills (30)**Practical Observations**

The system design is clean, and the schematic has been completed properly with all values accurately included. The design also works perfectly as intended. However, no additional innovations have been incorporated into the program or the design. The overall project could be improved further by adding some innovative features and enhancements to the design.

**Comment 9** | Transferable Skills (15)

The testing results presented in the table are clear and well organised.

**Comment 10** | Cognitive & Intellectual (25)

Future improvements are discussed; however, it is also necessary to include the limitations and challenges faced during the design process.

**Comment 11** | Transferable Skills (15)

- The structure of the report is neat and well presented. It could be further enhanced by including a critical review, justification for the selection of the PIC microcontroller, and the limitations of the design.

-

**Comment 12** | Transferable Skills (15)

Limited referencing is observed. The design should be critically evaluated by comparing it with journal papers and reference books.

**Comment 13** | Transferable Skills (15)

The code should be included in the appendix, but it is currently missing.

GRADING FORM: EMBEDDED C

ADAM IBRAHIM

67

ENGAGEMENT WITH LITERATURE(10)

Research supports embedded automotive cooling logic and PIC-based simulation. Uses scholarly/technical sources and correct referencing.



Some relevant sources seen but weak synthesis; referencing inconsistent.

5

KNOWLEDGE & UNDERSTANDING(20)

Demonstrates knowledge of sequencing logic, timers/delays, I/O interfacing, LCD 4-bit mode, UART monitoring and real-world cooling workflow.



Satisfactory understanding; basic explanations provided but limited depth in some areas.

13

COGNITIVE & INTELLECTUAL (25)

Evaluates design approach and makes justified decisions using logical reasoning. Demonstrates structured thinking and reflective judgement.



Some reasoning shown; limited justification and evaluation of limitations.

15

PRACTICAL SKILLS (30)

Implements and tests a PIC16F877A Proteus simulation meeting : switch sequencing, LED states, fan motor control, alert blink, LCD and UART messaging.



Mostly functional system; Core requirements met; good integration and testing evidence.

25

TRANSFERABLE SKILLS (15)

Communicates work clearly and professionally through a structured report and well-documented code. Submits required files and evidence reliably.



Adequate report and presentation; code partially documented.

9