Embedded project

Traffic light

Name: Youssef Rashad Farouk

Group: 3

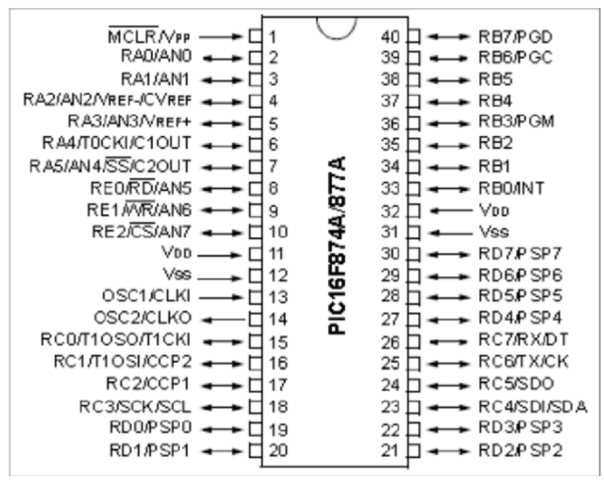
B.N: 59

2nd year communications & computer science

PIC16F877A

Description: PIC16F877A is one of microchip 8-bit microcontrollers and the most popular one that is used in many applications such as robotics, embedded systems, and industrial automation due to its versatility and robust feature set. The term of PIC16F indicates that it belongs to the PIC16 series, which is classified as mid-range in Microchip's lineup of microcontrollers, the PIC16 series includes microcontrollers with 8-bit data paths and a 14-bit wide instruction word, which makes it suitable for a wide range of applications with moderate complexity.

<u>Task1:</u>



a) Power Supply Pins

1. **VDD (Pins 11 and 32)**: Positive supply for logic and I/O pins.
2. **VSS (Pins 12 and 31)**: Ground reference for logic and I/O pins.

Oscillator Pins

3. **OSC1/CLKIN (Pin 13)**: Oscillator crystal or external clock input.
4. **OSC2/CLKOUT (Pin 14)**: Oscillator crystal or clock output.

Reset Pin

5. **MCLR/Vpp (Pin 1)**: Master Clear (input) or programming voltage (output).

I/O Port Pins

Port A

6. **RA0/AN0 (Pin 2)**: Digital I/O, Analog input 0.
7. **RA1/AN1 (Pin 3)**: Digital I/O, Analog input 1.
8. **RA2/AN2/VREF-/CVREF (Pin 4)**: Digital I/O, Analog input 2, A/D reference voltage (Low) input, Comparator VREF output.
9. **RA3/AN3/VREF+ (Pin 5)**: Digital I/O, Analog input 3, A/D reference voltage (High) input.
10. **RA4/T0CKI/C1OUT (Pin 6)**: Digital I/O – Open-drain when configured as output, Timer0 external clock input, Comparator 1 output.
11. **RA5/AN4/SS/C2OUT (Pin 7)**: Digital I/O, Analog input 4, SPI slave select input, Comparator 2 output.

Port B

12. **RB0/INT (Pin 33)**: Digital I/O, External interrupt.
13. **RB1 (Pin 34)**: Digital I/O.
14. **RB2 (Pin 35)**: Digital I/O.
15. **RB3/PGM (Pin 36)**: Digital I/O, Low-voltage ICSP programming enable pin.
16. **RB4 (Pin 37)**: Digital I/O.
17. **RB5 (Pin 38)**: Digital I/O.
18. **RB6/PGC (Pin 39)**: Digital I/O, In-circuit debugger and ICSP programming clock.
19. **RB7/PGD (Pin 40)**: Digital I/O, In-circuit debugger and ICSP programming data.

Port C

20. **RC0/T1OSO/T1CKI (Pin 15)**: Digital I/O, Timer1 oscillator output, Timer1 external clock input.
21. **RC1/T1OSI/CCP2 (Pin 16)**: Digital I/O, Timer1 oscillator input, Capture2 input, Compare2 output, PWM2 output.
22. **RC2/CCP1 (Pin 17)**: Digital I/O, Capture1 input, Compare1 output, PWM1 output.
23. **RC3/SCK/SCL (Pin 18)**: Digital I/O, Synchronous serial clock input/output for SPI mode, Synchronous serial clock input/output for I2C mode.
24. **RC4/SDI/SDA (Pin 23)**: Digital I/O, SPI data in, I2C data I/O.
25. **RC5/SDO (Pin 24)**: Digital I/O, SPI data out.
26. **RC6/TX/CK (Pin 25)**: Digital I/O, USART asynchronous transmit, USART1 synchronous clock.
27. **RC7/RX/DT (Pin 26)**: Digital I/O, USART asynchronous receive, USART synchronous data.

Port D

28. **RD0/PSP0 (Pin 19)**: Digital I/O, Parallel Slave Port data.
29. **RD1/PSP1 (Pin 20)**: Digital I/O, Parallel Slave Port data.
30. **RD2/PSP2 (Pin 21)**: Digital I/O, Parallel Slave Port data.
31. **RD3/PSP3 (Pin 22)**: Digital I/O, Parallel Slave Port data.
32. **RD4/PSP4 (Pin 27)**: Digital I/O, Parallel Slave Port data.
33. **RD5/PSP5 (Pin 28)**: Digital I/O, Parallel Slave Port data.
34. **RD6/PSP6 (Pin 29)**: Digital I/O, Parallel Slave Port data.
35. **RD7/PSP7 (Pin 30)**: Digital I/O, Parallel Slave Port data.

Port E

36. **RE0/RD/AN5 (Pin 8)**: Digital I/O, Read control for Parallel Slave Port, Analog input 5.
37. **RE1/WR/AN6 (Pin 9)**: Digital I/O, Write control for Parallel Slave Port, Analog input 6.

38. **RE2/CS/AN7 (Pin 10)**: Digital I/O, Chip select control for Parallel Slave Port, Analog input 7.

b)

1. **Arithmetic Logic Unit (ALU)**: The ALU performs arithmetic and logical operations. It is capable of handling operations like addition, subtraction, bitwise AND, OR, XOR, increment, decrement, and various shift operations.
2. **Status and Control**: This block includes the STATUS register and other control registers that monitor and control the operation of the microcontroller. The Status register contains the arithmetic status of the ALU, the Reset status and the bank select bits for data memory. The Status register can be the destination for any instruction, as with any other register. If the Status register is the destination for an instruction that affects the Z, DC or C bits, then the write to these three bits is disabled. These bits are set or cleared according to the device logic. Furthermore, the TO and PD bits are not writable, therefore, the result of an instruction with the Status register as destination may be different than intended.
3. **Program Counter (PC)**: The program counter holds the address of the next instruction to be executed from the Flash program memory. Ensures the sequential execution of instructions. It increments after fetching each instruction, but can also be modified by jump, call, and return instructions to change the execution flow.
4. **Flash Program Memory**: The flash program memory is readable and writable during normal operation (over the full VDD range). This memory is not directly mapped in the register file space. Instead, it is indirectly addressed through the Special Function Registers. There are six SFRs used to read and write this memory:
    - EECON1 • EECON2
    - EEDATA
    - EEDATH
    - EEADR
    - EEADRH

5. **Instruction Register**: It's also called buffer register, temporarily holds the current instruction fetched from the program memory before it is decoded and executed.
6. **Instruction Decoder**: Interprets the instruction held in the Instruction Register and generates the necessary control signals to execute it, converts the binary instruction code into a series of control signals that drive the various parts of the microcontroller to perform the specified operation.

c) The RA4/T0CKI pin is a Schmitt Trigger input and an open-drain output. In an open-drain configuration, the output transistor can pull the pin to ground (logic 0) but cannot drive it to a high level (logic 1). To achieve a high level, an external pull-up resistor is needed to pull the pin up to the supply voltage (VDD). All other PORTA pins have TTL (Transistor-Transistor Logic) input levels and full CMOS output drivers.

d)

1. Memory Size

- Program Memory (Flash): ○ **ATMega328P**: 32 KB ○ **PIC16F877A**: 14 KB
- RAM:
  ○ **ATMega328P**: 2 KB ○
  PIC16F877A: 368 bytes
- EEPROM:
  ○ **ATMega328P**: 1 KB ○
  PIC16F877A: 256 bytes

2. Power Consumption

- Operating Voltage:

○ **ATMega328P**: 1.8V to 5.5V ○
PIC16F877A: 2.0V to 5.5V

• Power Consumption:

○ **ATMega328P**: Typically lower power consumption in both active and sleep modes compared to PIC16F877A. ○ **PIC16F877A**: Higher power consumption in comparison.

3. Pin Count

- **ATMega328P**: 28 pins (PDIP)
- **PIC16F877A**: 40 pins (PDIP)

4. Clock Speed

- **ATMega328P**: Up to 20 MHz
- **PIC16F877A**: Up to 20 MHz

5. Peripheral Features

• **ATMega328P**:

○ 1x 16-bit Timer/Counter
○ 2x 8-bit Timer/Counters ○
6-channel 10-bit ADC ○
USART, SPI, I2C (TWI) ○
PWM channels

• **PIC16F877A**:

- ○ 3x Timers (one 8-bit, two 16-bit) ○ 8-channel 10-bit ADC ○ USART, SPI, I2C ○ PWM channels
- ○ CCP (Capture/Compare/PWM) modules

6. Development Ecosystem

• **ATMega328P**:

○ Widely supported by the Arduino ecosystem, making it very beginner-friendly. ○ Extensive community support and libraries.

- **PIC16F877A**:
  - Supported by Microchip's MPLAB ecosystem. ○ Good community support but generally considered less beginner-friendly compared to Arduino.

Examples of Embedded Systems Where ATMega328P is a Better Choice:

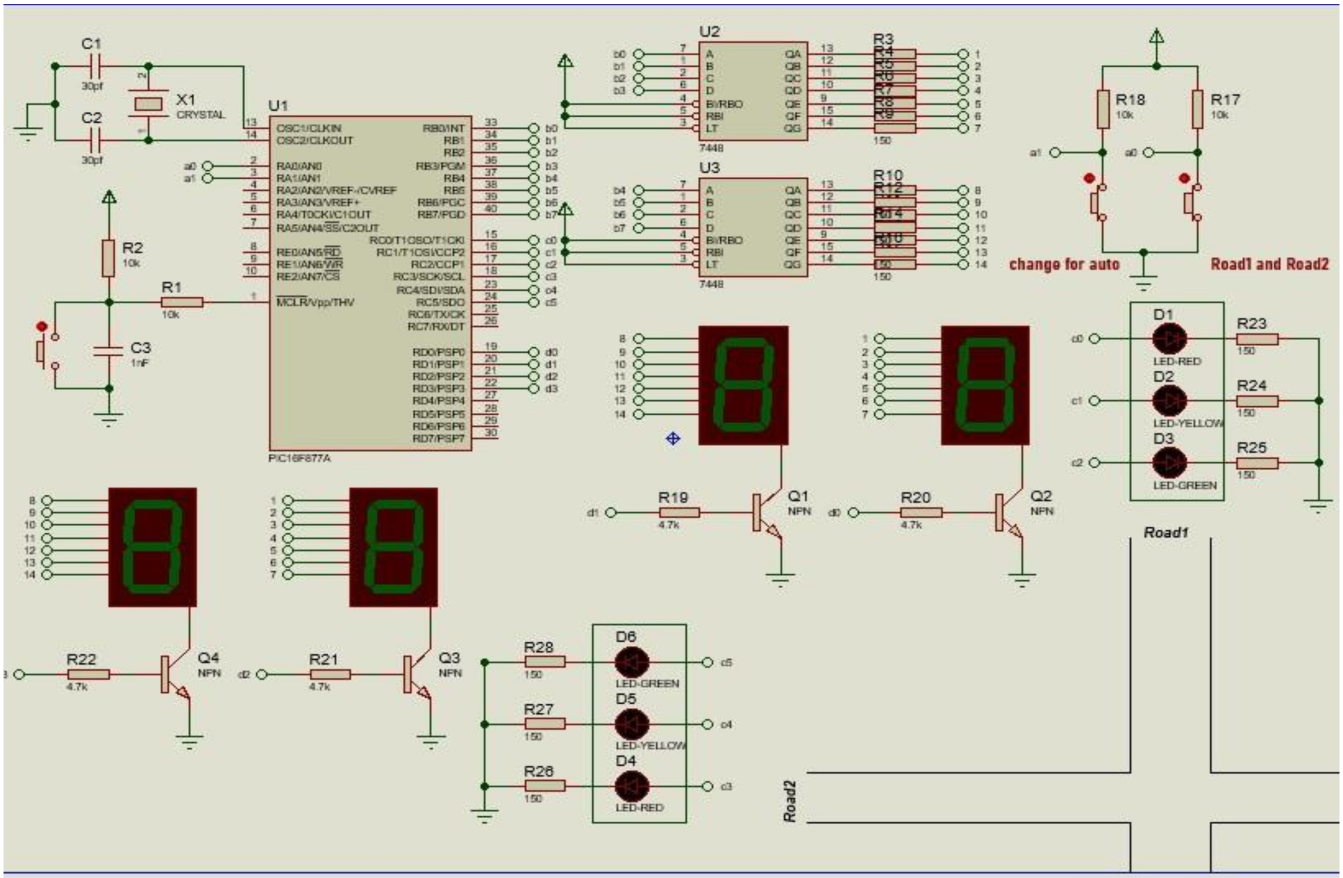
2. Arduino-Based Projects:

- Reason: The ATMega328P is the core of the Arduino Uno board, which has extensive support, libraries, and documentation. It is ideal for rapid prototyping and projects where ease of use and community support are critical.
- Example: A DIY home automation system using various sensors and actuators, where quick development and a plethora of ready-made libraries are advantageous.

3. Battery-Powered IoT Devices:

- Reason: Due to its lower power consumption and sleep modes, the ATMega328P is better suited for battery-powered applications. Its efficiency in sleep mode extends battery life significantly.
- Example: A remote environmental monitoring system with temperature, humidity, and pressure sensors, designed to run for extended periods on battery power.

Task2:



Circuit:

- The circuit diagram is a traffic light control system using a microcontroller (PIC16F877A), seven-segment displays, and LEDs to represent traffic lights. Here is a general description of the connections:

Port B (RB0-RB7):

Connected to seven-segment displays via 74LS48 BCD to seven-segment decoder/drivers (U2, U3).

- **Port D (RD0-RD7):**

Connected to the control transistors (Q1, Q2, Q3, Q4) which likely drive the seven-segment displays.

- **Port C (RC0-RC3):**

Connected to the LEDs (D1-D6) representing the traffic lights. These are connected via current-limiting resistors (R23-R28).

- **Seven-Segment Displays:**
  There are four seven-segment displays, each with connections for digits. These displays are likely used to show countdown timers for traffic light changes.
  The displays are connected to the outputs of the BCD to seven-segment decoders (U2, U3).

- **BCD to Seven-Segment Decoders (U2, U3):**

U2 and U3 are likely 74LS48 chips, which decode the binary input from the microcontroller into a seven-segment display output.

The outputs of U2 and U3 are connected to the corresponding seven-segment displays.

Each BCD decoder has its inputs connected to PORTB of the microcontroller.

- **Transistors (Q1-Q4):**

These are NPN transistors used to switch the seven-segment displays and LEDs on and off.

The bases of these transistors are connected to PORTD of the microcontroller via resistors (R19-R22).

The collectors are connected to the ground through the seven-segment displays or LEDs.

- **LEDs (D1-D6):**

These LEDs represent traffic lights for two roads (Road1 and Road2).

The LEDs are connected to the microcontroller via resistors (R23-R28) to limit current.

Red, Yellow, and Green LEDs are used to represent different traffic signals.

- ***Summary:***

The microcontroller controls the traffic light sequence using its I/O ports.

The BCD to seven-segment decoders convert binary input to display the countdown.

LEDs represent the traffic lights, controlled by the microcontroller through transistors.

The seven-segment displays show the countdown timers for each traffic light.

This circuit is likely designed for a simple traffic light system with timers for two roads.

- **Code Explanation:**

Checking porta.b1: If porta.b1 is high (button pressed), the system enters a sequence controlled by the for loop, adjusting the lights based on the countdown values and controlling the 7-segment display on port B.

Inner For Loops:

- The first loop cycles through the array arr[], adjusting the light behavior depending on the current traffic light state.

- The countdown is managed by count, count2, and count3, decrementing until the desired condition is met.

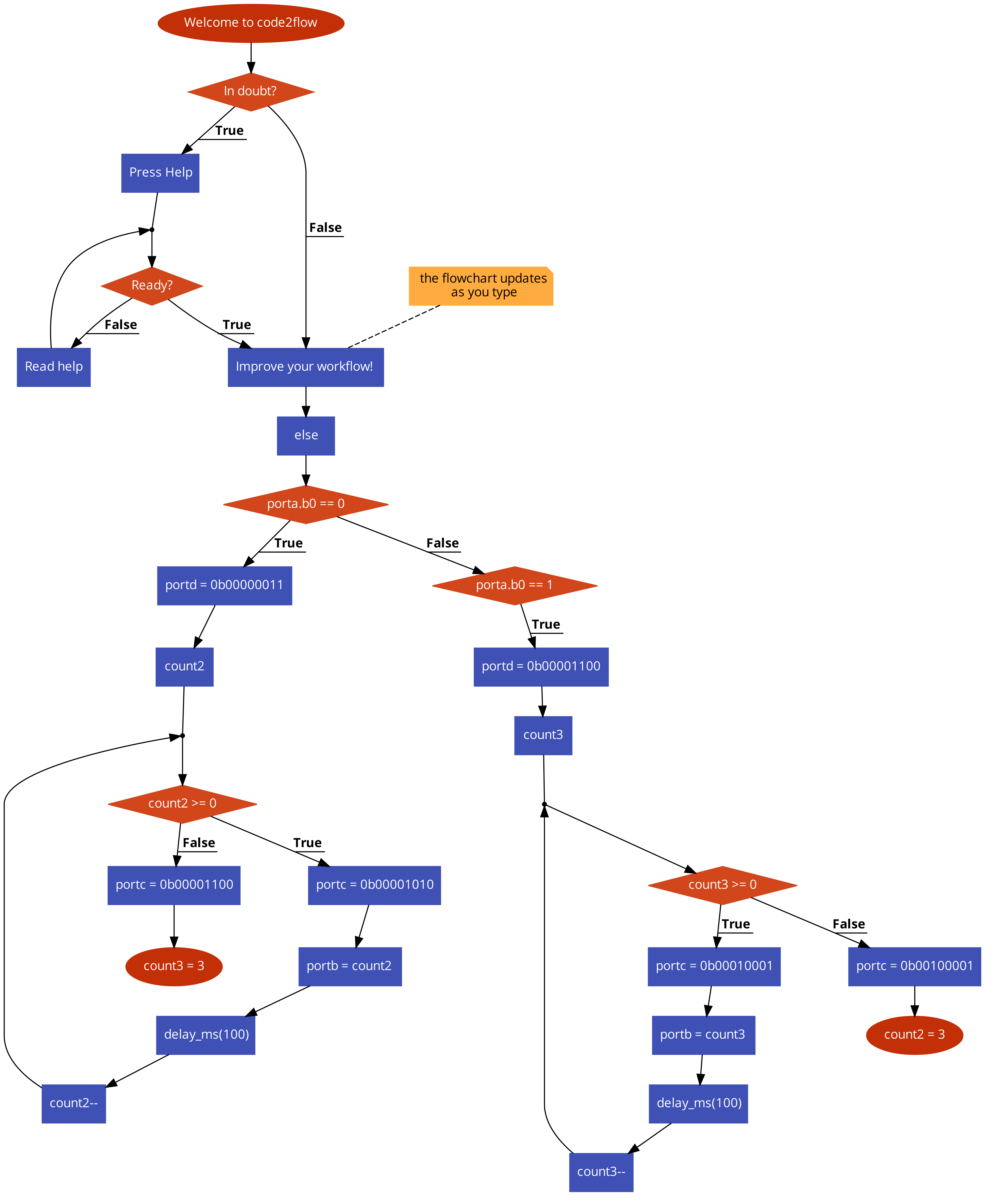
- Traffic light states are controlled using portc, which switches between different binary values to represent different colors (e.g., 0b00010001 for green, 0b00001010 for yellow, etc.).

Checking porta.b0:

- If porta.b1 is not pressed, the system checks porta.b0 to decide the alternate states of the system, which manage the sequence of the lights differently.

The project in YouTube:(https://youtu.be/rjRAKrR4yls?si=u682k-6jDI_bd40k)


Welcome to code2flow
In doubt?
True
False
Press Help
Ready?
False
True
Read help
the flowchart updates as you type
Improve your workflow!
True
porta.b1 == 1
True
False
portd = 0b11111111
i = 3
i
i >= 0
True
False
count = 9
count
count >= 0
True
False
porta.b1 == 0
True
False
portb = 0
count2 = 3
count3 = 3
i--
(arr[i] + count) == 57
(arr[i] + count) > 53
(arr[i] + count) == 52
(arr[i] + count) == 35
(arr[i] + count) == 32
portc = 0b00010001
portc = 0b00100001
portc = 0b00001010
portc = 0b00001100
portb = arr[i] + count
delay_ms(100)
count--


Welcome to code2flow
In doubt?
True
False
Press Help
Ready?
False
True
the flowchart updates as you type
Read help
Improve your workflow!
else
porta.b0 == 0
True
False
portd = 0b00000011
porta.b0 == 1
True
count2
portd = 0b00001100
count3
count2 >= 0
False
True
portc = 0b00001100
portc = 0b00001010
count3 = 3
portb = count2
delay_ms(100)
count2--
count3 >= 0
True
False
portc = 0b00010001
portc = 0b00100001
portb = count3
count2 = 3
delay_ms(100)
count3--