EMBEDDED SYSTEMS
TASK 1

Zeyad Mohamed Abdelfatah/زياد محمد عبدالفتاح فهيم

SEC : 2 B.N : 27

كلية الهندسة بشبرا

FACULTY OF ENGINEERING AT SHOUBRA

TASK_1:

• DESCRIBE ALL THE PINS OF PIC16F877A.

• THE FUNCTIONS OF THE MAIN BLOCKS IN PIC16F877A.

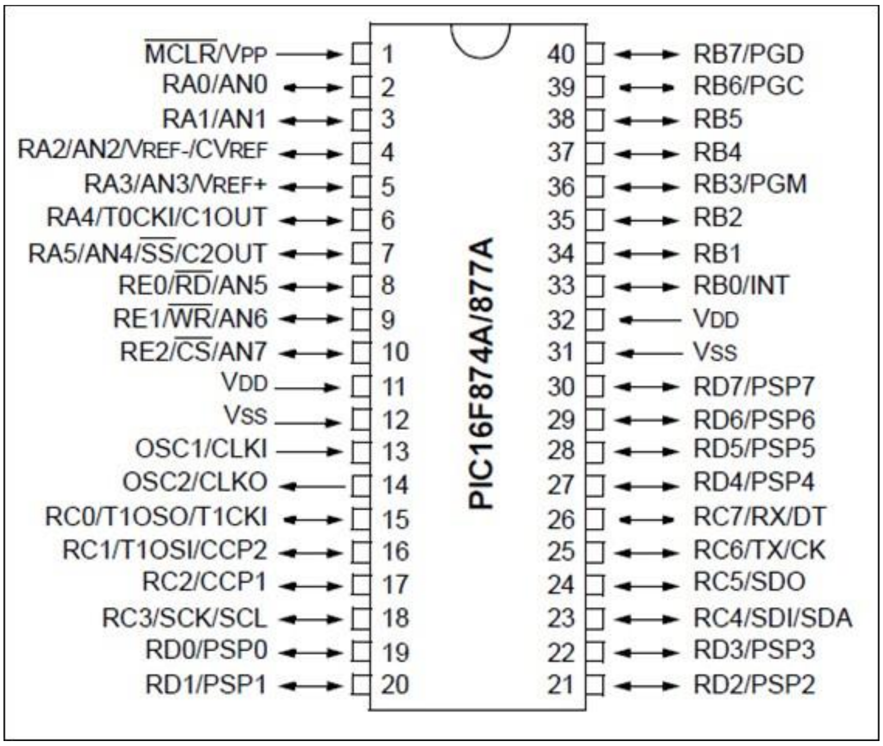
• A LED, WHICH IS CONNECTED TO RA4.

• THE CHARACTERISTICS OF ATMEGA328P VERSUS PIC16F877A.

• 2 EXAMPLES OF EMBEDDED SYSTEMS WHERE ATMEGA328P IS A BETTER CHOICE THAN PIC16F877A.

1- PINS OF PIC16F877A :

- THE PIC16F877A IS A VERSATILE 8-BIT MICROCONTROLLER WITH 40 PINS. IT'S WIDELY USED IN VARIOUS EMBEDDED SYSTEMS DUE TO ITS RICH FEATURE SET AND EASE OF PROGRAMMING. LET'S EXPLORE ITS PIN CONFIGURATION:



1-PIN GROUPS

THE PINS ARE ORGANIZED INTO FIVE PORTS: A, B, C, D, AND E. EACH PORT SERVES SPECIFIC FUNCTIONS, BUT MANY PINS HAVE MULTIPLE ROLES DEPENDING ON THE CONFIGURATION.

PORT A (RA0-RA5)

- PRIMARILY ANALOG INPUTS, BUT CAN BE CONFIGURED AS DIGITAL I/OS.
- RA0-RA4 ARE PART OF THE ADC (ANALOG-TO-DIGITAL CONVERTER).
- RA5 HAS ADDITIONAL FUNCTIONS LIKE VREF+ AND C1OUT.

PORT B (RB0-RB7)

- VERSATILE PORT WITH MULTIPLE FUNCTIONS:
  - DIGITAL I/OS
  - INTERRUPT SOURCES (RB0, RB4-RB7)
  - PROGRAMMING PINS (RB6, RB7)
  - TIMER INPUTS (RB0, RB2, RB3)
  - COMPARATOR INPUTS (RB0, RB1)

PORT C (RC0-RC7)

- PRIMARILY DEDICATED TO PERIPHERALS:
  - CCP1 (CAPTURE/COMPARE/PWM)
  - TMR0
  - USART (UNIVERSAL SYNCHRONOUS ASYNCHRONOUS RECEIVER TRANSMITTER)
  - SPI (SERIAL PERIPHERAL INTERFACE)
  - I2C (INTER-INTEGRATED CIRCUIT)
  - ANALOG INPUTS (RC0, RC1)

PORT D (RD0-RD7)

- PRIMARILY DIGITAL I/OS, BUT ALSO SERVES AS:
  - SLAVE PORT FOR I2C
  - COMPARATOR INPUTS (RD0, RD1)

PORT E (RE0-RE2)

- PRIMARILY DIGITAL I/OS, BUT ALSO SERVES AS:
  - MASTER OUTPUT ENABLE FOR I2C
  - COMPARATOR INPUTS (RE0, RE1)

ADDITIONAL PINS

- **MCLR (MASTER CLEAR):** RESETS THE MICROCONTROLLER.
- **OSC1 AND OSC2:** OSCILLATOR INPUTS FOR CLOCK SOURCE.
- **VDD:** POWER SUPPLY VOLTAGE.
- **VSS:** GROUND.

IMPORTANT NOTES

- MANY PINS HAVE MULTIPLE FUNCTIONS. THE SPECIFIC FUNCTION IS DETERMINED BY THE CONFIGURATION OF THE MICROCONTROLLER'S REGISTERS.
- CAREFUL CONSIDERATION SHOULD BE GIVEN TO PIN ASSIGNMENTS TO AVOID CONFLICTS AND ENSURE PROPER OPERATION.
- REFER TO THE PIC16F877A DATASHEET FOR DETAILED PIN DESCRIPTIONS, REGISTER DEFINITIONS, AND TIMING SPECIFICATIONS

2-FUNCTIONS OF THE MAIN BLOCKS IN PIC16F877A :

THE PIC16F877A MICROCONTROLLER IS A VERSATILE 8-BIT MICROCONTROLLER WITH A RICH FEATURE SET AND EASE OF PROGRAMMING:

THE MAIN BLOCKS OF THE PIC16F877A MICROCONTROLLER ARE:

- **ALU (ARITHMETIC LOGIC UNIT)**
- **STATUS AND CONTROL REGISTER**
- **PROGRAM COUNTER**
- **FLASH PROGRAM MEMORY**
- **INSTRUCTION REGISTER**
- **INSTRUCTION DECODER**

1) **ALU (ARITHMETIC LOGIC UNIT)**

THE ALU IS THE HEART OF THE MICROCONTROLLER, RESPONSIBLE FOR PERFORMING ARITHMETIC AND LOGICAL OPERATIONS ON DATA. IT CAN ADD, SUBTRACT, MULTIPLY, DIVIDE, COMPARE, AND PERFORM BITWISE OPERATIONS ON DATA.

2) **STATUS AND CONTROL REGISTER**

THE STATUS AND CONTROL REGISTER (SFR) IS A SPECIAL REGISTER THAT STORES THE STATUS OF THE MICROCONTROLLER AND CONTROLS ITS OPERATION. IT CONTAINS FLAGS THAT INDICATE THE RESULTS OF ARITHMETIC AND LOGICAL OPERATIONS, AS WELL AS BITS THAT CONTROL THE OPERATION OF THE MICROCONTROLLER'S PERIPHERALS.

3) **PROGRAM COUNTER**

THE PROGRAM COUNTER (PC) IS A REGISTER THAT HOLDS THE ADDRESS OF THE NEXT INSTRUCTION TO BE EXECUTED. IT IS INCREMENTED AFTER EACH INSTRUCTION IS EXECUTED, SO THAT THE MICROCONTROLLER CAN EXECUTE INSTRUCTIONS IN SEQUENCE.

4) **FLASH PROGRAM MEMORY**

THE FLASH PROGRAM MEMORY IS A NON-VOLATILE MEMORY THAT STORES THE MICROCONTROLLER'S PROGRAM CODE. IT IS USED TO STORE THE INSTRUCTIONS THAT THE MICROCONTROLLER EXECUTES.

5) **INSTRUCTION REGISTER**

THE INSTRUCTION REGISTER (IR) IS A REGISTER THAT HOLDS THE CURRENT INSTRUCTION BEING EXECUTED. IT IS LOADED WITH THE INSTRUCTION FROM THE PROGRAM MEMORY BY THE PROGRAM COUNTER.

6) **INSTRUCTION DECODER**

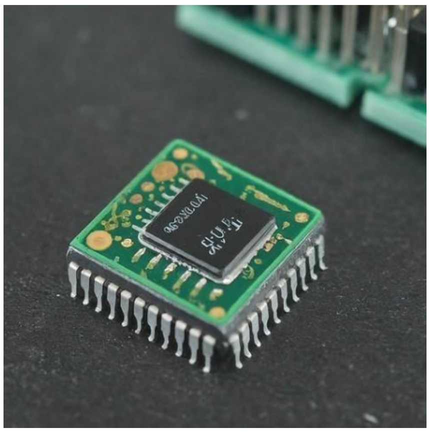
THE INSTRUCTION DECODER IS A CIRCUIT THAT DECODES THE INSTRUCTIONS STORED IN THE INSTRUCTION REGISTER. IT INTERPRETS THE INSTRUCTIONS AND GENERATES CONTROL SIGNALS THAT TELL THE OTHER BLOCKS OF THE MICROCONTROLLER HOW TO EXECUTE THE INSTRUCTION.

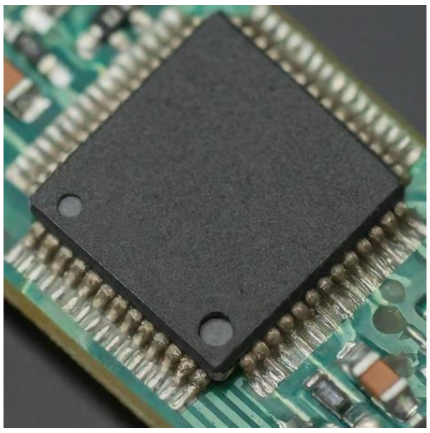
THE PIC16F877A MICROCONTROLLER ALSO HAS A NUMBER OF OTHER BLOCKS, INCLUDING:

- **CLOCK GENERATOR**
- **INTERRUPT CONTROLLER**
- **TIMER/COUNTERS**
- **SERIAL COMMUNICATION PERIPHERALS**

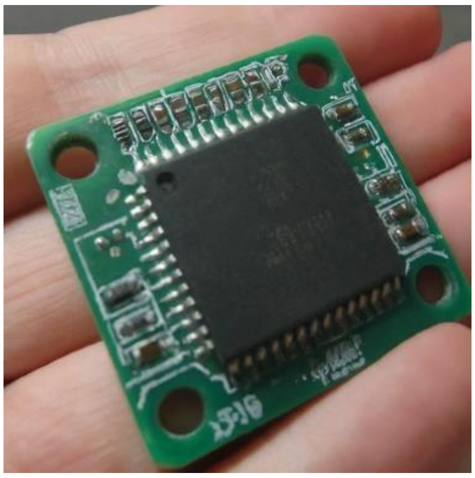
- **ANALOG-TO-DIGITAL CONVERTER (ADC)**

ALU
FLASH





PROGRAM COUNTER



STATUS AND CONTROL REGISTER

3-A LED CONNECTED TO RA4:

OPEN-DRAIN CONFIGURATION OF RA4 ON PIC16F877A

- **UNDERSTANDING OPEN-DRAIN CONFIGURATION**

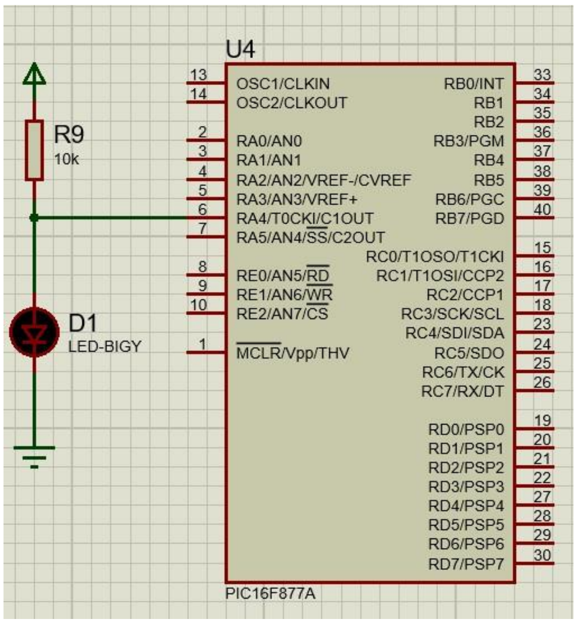
AN OPEN-DRAIN OUTPUT ACTS AS A SWITCH TO GROUND. WHEN THE OUTPUT IS ACTIVE, IT PULLS THE PIN LOW. TO ACHIEVE A HIGH OUTPUT, AN EXTERNAL PULL-UP RESISTOR IS REQUIRED TO CONNECT THE PIN TO THE POWER SUPPLY (VDD).

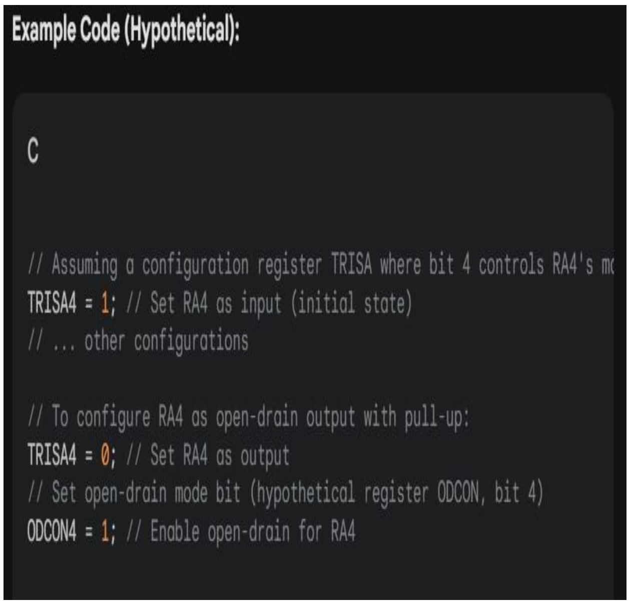
- **RA4 AS AN OPEN-DRAIN OUTPUT**

WHILE THE PIC16F877A DATASHEET DOESN'T EXPLICITLY MENTION RA4 AS AN OPEN-DRAIN OUTPUT, IT'S GENERALLY POSSIBLE TO CONFIGURE ANY DIGITAL I/O PIN AS OPEN-DRAIN BY SETTING THE APPROPRIATE BITS IN THE CONFIGURATION REGISTERS.

- **STEPS TO CONFIGURE RA4 AS OPEN-DRAIN:**

  1. **CHECK THE DATASHEET:**
     - VERIFY IF RA4 HAS ANY SPECIFIC LIMITATIONS OR REQUIREMENTS FOR OPENDRAIN OPERATION.
     - IDENTIFY THE CONFIGURATION REGISTER(S) CONTROLLING THE PIN'S MODE.
  2. **SET THE OPEN-DRAIN MODE:**
     - CONSULT THE DATASHEET FOR THE SPECIFIC BIT(S) CONTROLLING THE OPENDRAIN MODE.
     - SET THESE BITS TO ENABLE OPEN-DRAIN OPERATION FOR RA4.
  3. **ADD A PULL-UP RESISTOR:**
     - CONNECT A RESISTOR BETWEEN RA4 AND VDD. THE RESISTOR VALUE DEPENDS ON THE DESIRED CURRENT AND VOLTAGE LEVELS.




Example Code (Hypothetical):

C

```c
// Assuming a configuration register TRISA where bit 4 controls RA4's m
TRISA4 = 1; // Set RA4 as input (initial state)
// ... other configurations

// To configure RA4 as open-drain output with pull-up:
TRISA4 = 0; // Set RA4 as output
// Set open-drain mode bit (hypothetical register ODCON, bit 4)
ODCON4 = 1; // Enable open-drain for RA4
```

4- ATMega328P vs. PIC16F877A: A Comparison

FEATURE	PIC16F877A	ATMEGA328P
ARCHITECTURE	AVR	PIC
Memory (Flash/RAM/EEPROM)	32KB/2KB/1KB	14KB/368B/256B
CLOCK SPEED	UP TO 16 MHZ	UP TO 20 MHZ
ADC CHANNELS	8-CHANNEL, 10-BIT	8-CHANNEL, 10-BIT
PWM CHANNELS	6	2
TIMERS	3	1
I/O PINS	32	40
POWER CONSUMPTION	GENERALLY HIGHER	GENERALLY LOWER

TWO EXAMPLES WHERE ATMEGA328P OUTSHINES PIC16F877A

1. DRONE FLIGHT CONTROLLER

A DRONE FLIGHT CONTROLLER REQUIRES SIGNIFICANT PROCESSING POWER TO HANDLE SENSOR DATA, MOTOR CONTROL, AND STABILIZATION ALGORITHMS. THE ATMEGA328P, WITH ITS HIGHER CLOCK SPEED AND LARGER MEMORY, IS BETTER SUITED FOR THESE COMPLEX CALCULATIONS AND REAL-TIME CONTROL DEMANDS. ADDITIONALLY, THE AVAILABILITY OF MULTIPLE PWM CHANNELS FOR MOTOR CONTROL IS A SIGNIFICANT ADVANTAGE.

2. DATA LOGGER WITH DISPLAY

A DATA LOGGER THAT COLLECTS SENSOR DATA AND DISPLAYS IT ON AN LCD REQUIRES A MICROCONTROLLER WITH SUFFICIENT MEMORY TO STORE DATA AND HANDLE THE DISPLAY INTERFACE. THE ATMEGA328P'S LARGER PROGRAM AND DATA MEMORY, ALONG WITH ITS ABILITY TO DRIVE LCD DISPLAYS DIRECTLY, MAKE IT A MORE SUITABLE CHOICE COMPARED TO THE PIC16F877A, WHICH MIGHT STRUGGLE WITH THE PROCESSING AND MEMORY REQUIREMENTS OF SUCH A SYSTEM.

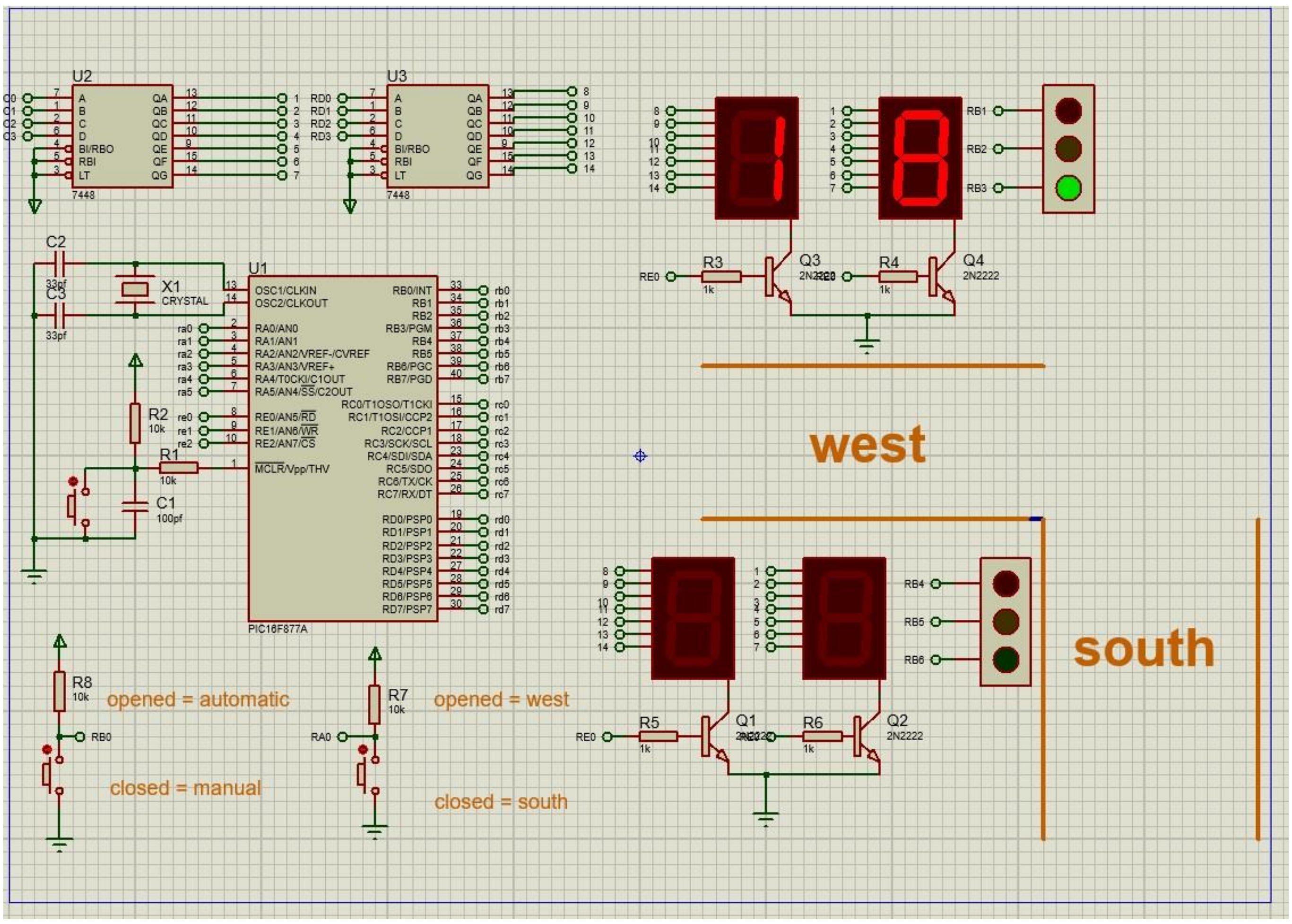
IN BOTH THESE EXAMPLES, THE ATMEGA328P'S SUPERIOR PROCESSING CAPABILITIES, LARGER MEMORY, AND ADDITIONAL PERIPHERALS PROVIDE A CLEAR ADVANTAGE OVER THE PIC16F877A.

EMBEDDED SYSTEMS
TASK 2(DIGITAL TRAFFIC)

Zeyad Mohamed Abdelfatah/زياد محمد عبدالفتاح فهيم

SEC : 2 B.N : 27

The circuit of the system using proteus:



Code Analysis: Traffic Light Controller Purpose:

- Controls a two-way traffic intersection with LEDs for traffic signals and 7-segment displays for countdown timers.
- Offers both automatic and manual modes of operation.

Hardware:

- Microcontroller: PIC16F877A
- LEDs: Six LEDs for traffic signals (red1, yellow1, green1, red2, yellow2, green2) connected to RB1-RB6.
- Buttons: Two buttons connected to RA0 and RB0 for manual control.
- Displays: Two 7-segment displays connected to PortC and PortD for displaying the countdown timer.
- 2N3220 (Transistor) : Used as a switch for the LEDs.
- Resistors.
- Capacitors.
- Crystal**:** Used as a clock source for the microcontroller.
- Integrated Circuits (ICs).

Code Structure:

- **Variable declarations:** Defines variables for LED control, display data, and loop counters.
- **Function prototypes:** Declares functions for traffic light control, display update, and mode handling.
- **traffic(signed char counter):** Updates the 7-segment displays with the given counter value.
- **main():** Initializes hardware, creates an infinite loop for continuous operation, and handles mode selection based on button presses.
- **automatic():** Implements the automatic traffic light sequence with countdown timers.
- **south()** and **west():** Handle manual control for the respective directions.

Code Logic

Automatic Mode:

- Cycles through a fixed sequence of green, yellow, and red lights for both directions.
- Uses a countdown timer displayed on the 7-segment displays.

- Checks for manual mode activation during the sequence.

Manual Mode:

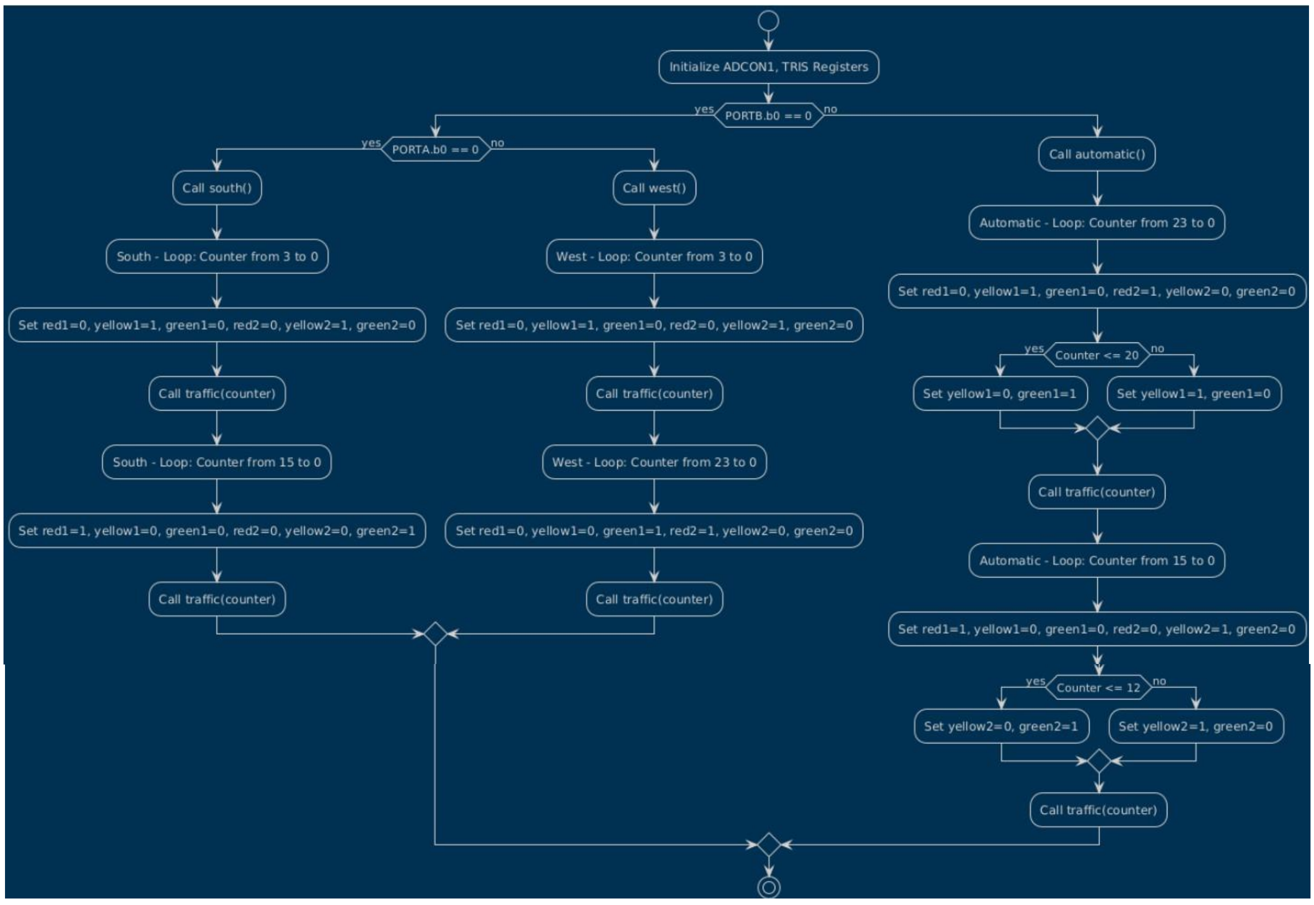
- Prioritizes the direction corresponding to the pressed button.
- Displays a countdown timer for the green light and then switches to the opposite direction's green light.

Code:

```
sbit red1 at rb1_bit;

sbit yellow1 at rb2_bit;

sbit green1 at rb3_bit;

sbit red2 at rb4_bit;

sbit yellow2 at rb5_bit;

sbit green2 at rb6_bit;

char left, right, i;

signed char counter;

void south();

void west();

void automatic();

void traffic(signed char counter) {

   porte.b0 = 1;

   left = counter / 10;

   right = counter % 10;

   portc = right;

   delay_ms(10);

   portd = left;

   delay_ms(10);

}

void main() {

   ADCON1 = 0x07;

   trisb = 0b00000001;
```

```
    trisc = 0x00;

    trisd = 0x00;

    trisa = 0xff;

    trise = 0x00;

    while (1) {

        if (portb.b0 == 0){

         if (porta.b0 == 0){south();}

                    else{west();}

        }


        else

       automatic();

    }

}


void automatic() {

    for (;;) {

        for (counter = 23; counter >= 0; counter--) {

            if (portb.b0 == 0) {

                    if (porta.b0 == 0){south();}

                      else{west();}

            }

            red1 = 0;yellow1 = 1;green1 = 0;

            red2 = 1;yellow2 = 0;green2 = 0;

            if (counter <= 20) {yellow1 = 0;green1 = 1;}

            for (i = 0; i <= 50; i++) {

                traffic(counter);

            }

        }
        for (counter = 15; counter >= 0; counter--) {
           if (portb.b0 == 0) {
               if (porta.b0 == 0){south();}
               else {west();}
           }
           red1 = 1;yellow1 = 0;green1 = 0;
           red2 = 0;yellow2 = 1;green2 = 0;
           if (counter <= 12) {yellow2 = 0;green2 = 1;}
           for (i = 0; i <= 50; i++) {
              traffic(counter);
           }
        }
     }
  }
  void south() {
           for (counter = 3; counter >= 0; counter--) {
        red1 = 0;yellow1 = 1;green1 = 0;
        red2 = 0;yellow2 = 1;green2 = 0;
        for (i = 0; i <= 50; i++) {
           traffic(counter);
        }
     }
        for (counter = 15; counter >= 0; counter--) {
           red1 = 1;yellow1 = 0;green1 = 0;
           red2 = 0;yellow2 = 0;green2 = 1;
           if (portb.b0!=0) break;
           for (i = 0; i <= 50; i++) {
              traffic(counter);
           }
        }
}
void west() {
        for (counter = 3; counter >= 0; counter--) {
        red1 = 0;yellow1 = 1;green1 = 0;
        red2 = 0;yellow2 = 1;green2 = 0;
        for (i = 0; i <= 50; i++) {
            traffic(counter);
        }
    }
        for (counter = 23; counter >= 0; counter--) {
            red1 = 0;yellow1 = 0;green1 = 1;
            red2 = 1;yellow2 = 0;green2 = 0;
            if (portb.b0!=0) break;
            for (i = 0; i <=50; i++) {
                traffic(counter);
            }
        }
}
```

Flow chart:


Initialize ADCON1, TRIS Registers
yes
PORTB.b0 == 0
no
yes
PORTA.b0 == 0
no
Call south()
South - Loop: Counter from 3 to 0
Set red1=0, yellow1=1, green1=0, red2=0, yellow2=1, green2=0
Call traffic(counter)
South - Loop: Counter from 15 to 0
Set red1=1, yellow1=0, green1=0, red2=0, yellow2=0, green2=1
Call traffic(counter)
Call west()
West - Loop: Counter from 3 to 0
Set red1=0, yellow1=1, green1=0, red2=0, yellow2=1, green2=0
Call traffic(counter)
West - Loop: Counter from 23 to 0
Set red1=0, yellow1=0, green1=1, red2=1, yellow2=0, green2=0
Call traffic(counter)
Call automatic()
Automatic - Loop: Counter from 23 to 0
Set red1=0, yellow1=1, green1=0, red2=1, yellow2=0, green2=0
yes
Counter <= 20
no
Set yellow1=0, green1=1
Set yellow1=1, green1=0
Call traffic(counter)
Automatic - Loop: Counter from 15 to 0
Set red1=1, yellow1=0, green1=0, red2=0, yellow2=1, green2=0
yes
Counter <= 12
no
Set yellow2=0, green2=1
Set yellow2=1, green2=0
Call traffic(counter)