EMBEDDED-SYSTEMS

PIC16F877A

This document provides a concise guide to the PIC16F877A microcontroller, detailing its pin descriptions and interfacing capabilities to aid in designing embedded systems.

PIC16F877A

Introduction to Embedded Systems

Embedded systems are specialized computing systems that perform dedicated functions within larger mechanical or electrical systems. Unlike general-purpose computers, embedded systems are designed for specific tasks, which allows them to be optimized for performance, reliability, and efficiency. These systems are found in a wide range of applications, from household appliances like microwaves and washing machines to sophisticated devices like medical equipment and industrial machines. Embedded systems often include microcontrollers or microprocessors, sensors, and actuators, and they operate in real-time to ensure timely and precise responses to external inputs.

❖ Task 1

1) Pin Description of PIC16F877A

The PIC16F877A is a versatile and widely-used 40-pin microcontroller from Microchip. Here's a detailed description of its pins to help you interface it with other hardware:

1. **Power Pins**:
    - **VDD (Pins 11 and 32)**: Connect to the positive supply voltage (+5V).
    - **VSS (Pins 12 and 31)**: Connect to the ground (0V).
2. **Oscillator Pins**:
    - **OSC1/CLKIN (Pin 13)**: Input for the external clock or crystal oscillator.

- **OSC2/CLKOUT (Pin 14)**: Output for the external clock or crystal oscillator.

3. **Reset Pin**:
    - **MCLR (Pin 1)**: Master Clear (Reset) pin. Used to reset the microcontroller when a low voltage is applied.
4. **Input/Output Ports**:
    - **PORTA (Pins 2-7)**: 6-bit wide, bidirectional port. Each pin can be used as an input or output.
    - **PORTB (Pins 33-40)**: 8-bit wide, bidirectional port. Each pin can be used as an input or output.
    - **PORTC (Pins 15-18, 23-26)**: 8-bit wide, bidirectional port. Each pin can be used as an input or output.
    - **PORTD (Pins 19-22, 27-30)**: 8-bit wide, bidirectional port. Each pin can be used as an input or output.
    - **PORTE (Pins 8-10)**: 3-bit wide, bidirectional port. Each pin can be used as an input or output.
5. **Analog Pins**:
    - **AN0 to AN7 (PORTA pins and RE0-RE2)**: These pins can be used for analog-to-digital conversion (ADC).
6. **Serial Communication Pins**:
    - **TX (Pin 25)**: Transmit pin for UART communication.
    - **RX (Pin 26)**: Receive pin for UART communication.
    - **SCK/SCL (Pin 18)**: Clock pin for SPI or I2C communication.
    - **SDI/SDA (Pin 23)**: Data input pin for SPI or I2C communication.
    - **SDO (Pin 24)**: Data output pin for SPI communication.
7. **Special Function Pins**:
    - **SS (Pin 5)**: Slave select pin for SPI communication.

- **CCP1 (Pin 17)**: Capture/Compare/PWM pin 1.
- **CCP2 (Pin 16)**: Capture/Compare/PWM pin 2.
- **T1OSI and T1OSO (Pins 33 and 34)**: Timer1 oscillator pins.

Summary

- **Power and Ground Pins**: Provide power to the microcontroller.
- **Oscillator Pins**: Control the clock frequency.
- **Reset Pin**: Used to reset the microcontroller.
- **I/O Ports**: Used for digital input/output.
- **Analog Pins**: Used for reading analog signals.
- **Communication Pins**: Used for serial communication (UART, SPI, I2C).
- **Special Function Pins**: Used for specific tasks like PWM and timers.

2) Functions of the Main Blocks in PIC16F877A

1. **Arithmetic Logic Unit (ALU)**:
    - The ALU performs all arithmetic and logical operations, such as addition, subtraction, and logical operations like AND, OR, and NOT.
2. **Status and Control Registers**:
    - The Status Register contains bits indicating the outcome of ALU operations, like zero or carry. Control Registers manage the microcontroller's features and functions, such as enabling/disabling interrupts and configuring I/O ports.
3. **Program Counter (PC)**:
    - The Program Counter keeps track of the address of the next instruction to be executed, ensuring the correct sequence of instruction execution.

4. **Flash Program Memory**:
    - This non-volatile memory stores the program code, retaining it even when the power is off. It holds 14-bit wide instructions for efficient code storage.
5. **Instruction Register (IR)**:
    - The Instruction Register temporarily holds the current instruction fetched from the program memory before it is decoded and executed.
6. **Instruction Decoder**:
    - The Instruction Decoder interprets the instruction in the Instruction Register and translates it into signals that control other parts of the microcontroller to perform the desired operation.

3) Troubleshooting LED Functionality on RA4 Pin:

1. **RA4 Functionality**:
    - **RA4** is also known as **T0CKI** (Timer0 Clock Input) on the PIC16F877A. This pin can be used for Timer0 input and might not be available for general-purpose I/O operations in some configurations. Check if RA4 is configured as an output and not being used for its alternative function.
2. **Pin Configuration**:
    - Ensure that RA4 is properly configured as a digital output. If RA4 is set up as an analog input or is configured for another function, it may not output the correct signal for the LED.
3. **Drive Capability**:
    - The RA4 pin might not have enough current drive capability to power the LED directly. In some cases, the output current of a pin

might be insufficient to light up the LED. You might need to use a transistor or a current-limiting resistor to drive the LED.

4. **Incorrect Code**:
    - Verify that the microcontroller code is correctly setting RA4 to output and toggling it as intended. Ensure that the code properly initializes RA4 and controls it to flash the LED.
5. **Hardware Issues**:
    - Check the physical connections and components. Ensure the LED is correctly connected with the right polarity, and verify that the resistor (if used) is of the correct value. Inspect the connections for any loose or faulty wiring.
6. **Power Supply**:
    - Ensure that the microcontroller and the LED are receiving the correct voltage. An unstable or incorrect power supply can cause the LED not to function properly.

Summary

- **RA4 Functionality**: RA4 may serve a different function like Timer0 input.
- **Pin Configuration**: Ensure RA4 is configured as a digital output.
- **Drive Capability**: RA4 might not supply enough current for the LED.
- **Incorrect Code**: Verify the microcontroller code is correctly toggling RA4.
- **Hardware Issues**: Check connections and components.
- **Power Supply**: Ensure stable and correct voltage supply.

4) Comparison of ATMega328P and PIC16F877A

1. Memory Size:

- **ATMega328P**:
    - **Flash Memory**: 32 KB
    - **SRAM**: 2 KB
    - **EEPROM**: 1 KB
- **PIC16F877A**:
    - **Flash Memory**: 14 KB
    - **SRAM**: 368 Bytes
    - **EEPROM**: 256 Bytes

2. Power Consumption:

- **ATMega328P**:
    - Typically consumes around 0.2 mA to 2 mA in active mode at 8 MHz.
    - Lower power consumption in sleep modes.
- **PIC16F877A**:
    - Typically consumes around 1.5 mA to 10 mA in active mode at 4 MHz.
    - Also has power-saving sleep modes but generally consumes more power compared to ATMega328P.

3. Pin Count:

- **ATMega328P**:
    - 28 pins (in the DIP package).
- **PIC16F877A**:
    - 40 pins.

4. Additional Features:

- **ATMega328P**:

- Includes 6 analog input pins.
- Has built-in UART, SPI, and I2C for communication.
- Built-in watchdog timer and various power-saving modes.

- **PIC16F877A**:
  - Includes 8 analog input pins.
  - Has UART, SPI, and limited I2C support.
  - Advanced timer modules and PWM outputs.

Examples of Embedded Systems Where ATMega328P is a Better Choice

1. **Arduino Projects:**
   - The ATMega328P is the core of many Arduino boards (e.g., Arduino Uno). Its large Flash memory, efficient power consumption, and rich feature set make it ideal for hobbyist and educational projects, where ease of programming and power efficiency are crucial.
2. **Battery-Powered Devices:**
   - For applications like portable sensors or wearable electronics that need to operate on battery power for extended periods, the ATMega328P's lower power consumption and efficient sleep modes make it a better choice compared to the PIC16F877A, which generally consumes more power.

Summary

- **Memory Size**: ATMega328P has more Flash and SRAM, useful for larger programs and data.
- **Power Consumption**: ATMega328P generally consumes less power.
- **Pin Count**: ATMega328P has fewer pins compared to PIC16F877A.
- **Embedded Systems**: ATMega328P is ideal for Arduino-based projects and battery-powered devices due to its power efficiency and memory size.

❖ Task 2

- **Task Description: Traffic Light Controller**

  Objective: Develop a traffic light control system using the PIC16F877A microcontroller, designed to operate in both automatic and manual modes. The system will manage traffic lights at an intersection with two streets: West and South.

  Features:

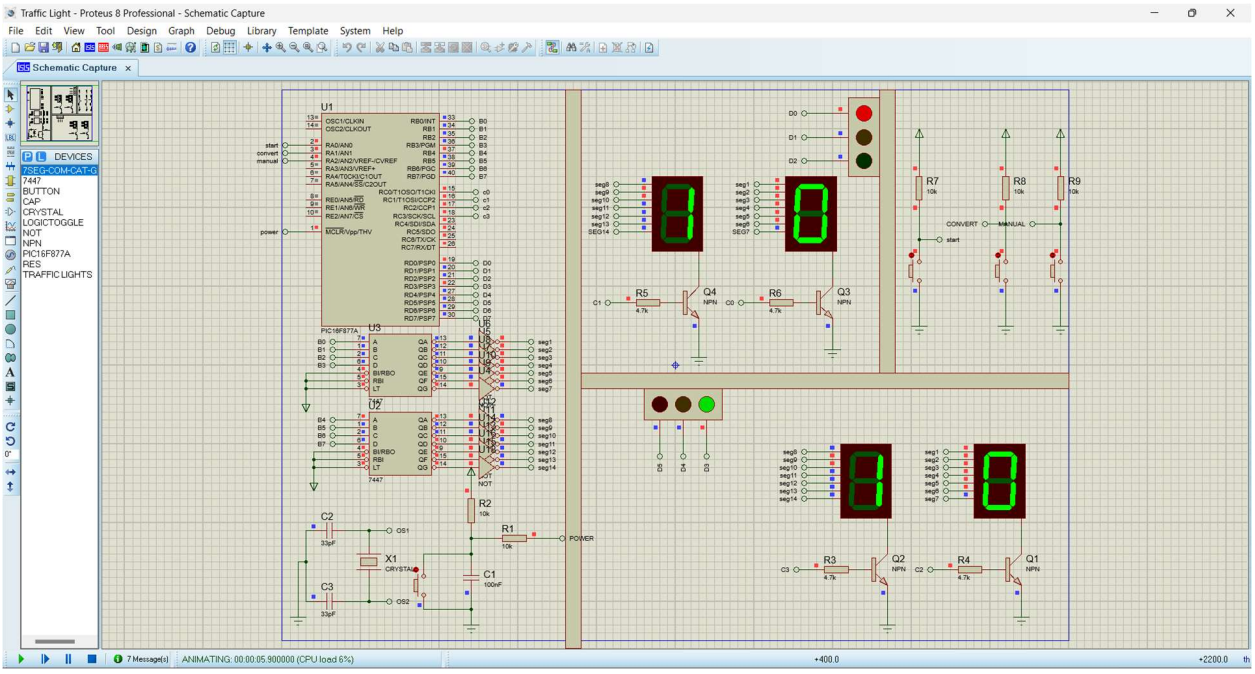
  1. **Modes of Operation:**
     - **Automatic Mode:** The traffic lights switch according to pre-defined timing:
       - **West Street:** 15 seconds Red, 3 seconds Yellow, 20 seconds Green.
       - **South Street:** 23 seconds Red, 3 seconds Yellow, 12 seconds Green.
     - **Manual Mode:** Allows manual switching between streets and toggling the traffic lights if needed.
  2. **Control Mechanism:**
     - **Switches:** Two switches are used:
       - One for toggling between Manual and Automatic modes.
       - Another for selecting between the two streets in Manual mode.
  3. **Timing in Manual Mode:** Even in Manual mode, a 3-second Yellow light period is required.

4. **Display:** Use 7-segment displays at each corner of the intersection to show the remaining time. BJTs and a 7447 IC are recommended to manage these displays and reduce the number of microcontroller pins used.

Code Overview:

- **automatic() Function:** Handles traffic light timing and transitions in Automatic mode.
- **manual() Function:** Controls the traffic lights in Manual mode, allowing manual toggling of lights.
- **Initialization:** Configures ports and settings, including ADC conversion and pin directions.
- **Main Loop:** Continuously checks mode and executes corresponding functions.

The provided code snippet includes functions to manage the traffic light timings and transitions, as well as to handle switching between Automatic and Manual modes.



GitHub Link : https://github.com/Abanoub756/Embedded-Project

YouTube Video Link : https://youtu.be/wVQSlNve3hU