User Manual for TCP/IP Library with ENC28J60 Ported to XC8

Overview

This manual provides detailed instructions for setting up, configuring, and using the TCP/IP library with the ENC28J60 Ethernet controller and Microchip's XC8 compiler. The library supports UDP communication and includes all necessary drivers for the ENC28J60.

Prerequisites

Hardware Requirements

- **Microcontroller**: PIC16F877A (or compatible PIC microcontrollers)
- **Ethernet Controller**: ENC28J60
- **Clock Frequency**: Minimum 20 MHz
- **Power Supply**: Ensure a stable 3.3V power source for ENC28J60

Software Requirements

- **Compiler**: XC8 Compiler (compatible with MPLAB X IDE)
- **IDE**: MPLAB X IDE

Setup Instructions

Step 1: Hardware Connections

1. **ENC28J60 Connections**:
    - **SCK**: Connect to SPI clock pin (RC3 on PIC16F877A)
    - **SDI**: Connect to SPI data input pin (RC4)
    - **SDO**: Connect to SPI data output pin (RC5)
    - **CS**: Connect to a GPIO pin (RD3)
    - **RESET**: Connect to a GPIO pin (RD2)
    - **INT**: Connect to a GPIO pin (optional for interrupts)
2. **Power Supply**:
    - Provide a stable 3.3V supply to ENC28J60.
    - Ensure proper grounding between ENC28J60 and the microcontroller.
3. **Crystal Oscillator**:
    - Use a 25 MHz crystal oscillator for ENC28J60.

Step 2: Software Configuration

1. **Clone the Repository**:
   git clone https://github.com/yourusername/tcpip-library-enc28j60-xc8.git
2. **Include Header Files**: Add the following to your main file:
   #include "enc28j60.h"

Set Up Configuration Bits: Use the following configuration for your microcontroller:
#pragma config FOSC = HS // High-speed oscillator
#pragma config WDTE = OFF // Watchdog Timer disabled
#pragma config PWRTE = OFF // Power-up Timer disabled
#pragma config BOREN = ON // Brown-out Reset enabled

3. #pragma config LVP = OFF // Low-Voltage Programming disabled
4. **Set the Clock Frequency**: Define the crystal frequency:
   #define _XTAL_FREQ 20000000

Initialization

ENC28J60 Initialization

1. Configure the SPI module:
   configurar_SPI();
2. Perform a soft reset on the ENC28J60:
   soft_reset();
3. Initialize the Ethernet buffer:
   eth_buffer_init();

Configure MAC and PHY registers:
mac_init();

4. phy_init();

USART Initialization

Set up the USART module for debugging and communication:

USART_Init(9600); // Set baud rate to 9600

Sending a UDP Packet

1. Set up the Ethernet frame in the transmission buffer:
   write_buffer_memory();

Send the packet:
uint8_t status = dhcp_discover();
if (status == 'G') {
USART_Transmit("Packet Sent Successfully\n");

2. }

Receiving Packets

1. Check for available packets in the RX buffer:
   uint8_t pkt_count = read_register(EPKTCNT);

Process received packets:
if (pkt_count > 0) {
read_receive_buffer_pkt();

2. }

Debugging

Use the USART module for debugging:

- **Send Hexadecimal Data**:
  usart_tx_byte_to_hex(data);
- **Insert a New Line**:
  usart_newline();

Print Register Values:
uint8_t reg_value = read_register(ESTAT);

- usart_tx_byte_to_hex(reg_value);

Best Practices

1. **Error Handling**: Check for errors in transmission and reception by monitoring the `ESTAT` and `EIR` registers.
2. **Memory Management**: Ensure proper buffer management by updating `ERXRDPT` after processing a received packet.
3. **Interrupt Handling** (Optional): Use the INT pin of ENC28J60 for efficient packet handling in interrupt-driven applications.

Troubleshooting

Common Issues

1. **No Response from ENC28J60**:
    - Verify SPI connections.
    - Check the power supply and crystal oscillator.
2. **Packet Loss**:
    - Increase the clock frequency if possible.
    - Check buffer initialization.
3. **Debugging Errors**:
    - Use USART for step-by-step verification of register values.

Additional Information

Refer to the ENC28J60 datasheet for detailed register descriptions and operational guidelines. For further assistance, open an issue in the GitHub repository.