Ethernet Test

Experimenting with MikroE Libraries

There is very little in the way of examples for using Mikroelektronika libraries for PIC32. This results in trial by error programming and a lot of hard work. The project represented here can serve as a skeleton for more complete development.

1/5/2013

General

Ethernet is one of the most popular interfaces in the microcontroller world. It is fast and can span long distances. It is the communications choice in small factories for interfacing tools and monitoring equipment.

The PIC32MX7 is ideally suited for this task with its built-in Ethernet controller. It has the right mix of serial interfaces and I/O for various applications in factory automation and process control.

Scope

This project attempts to construct several applications for Ethernet which run simultaneously on the microprocessor. It is a starting point for further development in implementing real world products.

Method

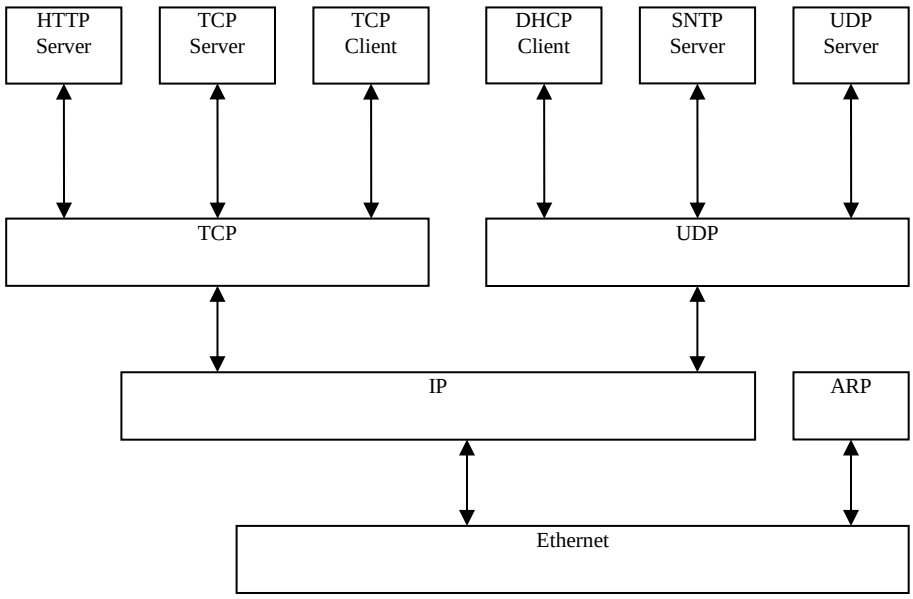
In order to test the applications, it was necessary to write test programs on the PC. I chose to write them in Delphi 7 due to its simplicity. Delphi 7 includes Indy components for Ethernet. The Serial Utility uses Async Professional for the serial interface.

Results

The implementation using MikroE Libraries seem to work reasonably well. There are a few glitches that have not been resolved at this time. E.g. – TCP Client loses the ability to make a connection after some time. The only way to overcome this difficulty is to cycle power to the board.

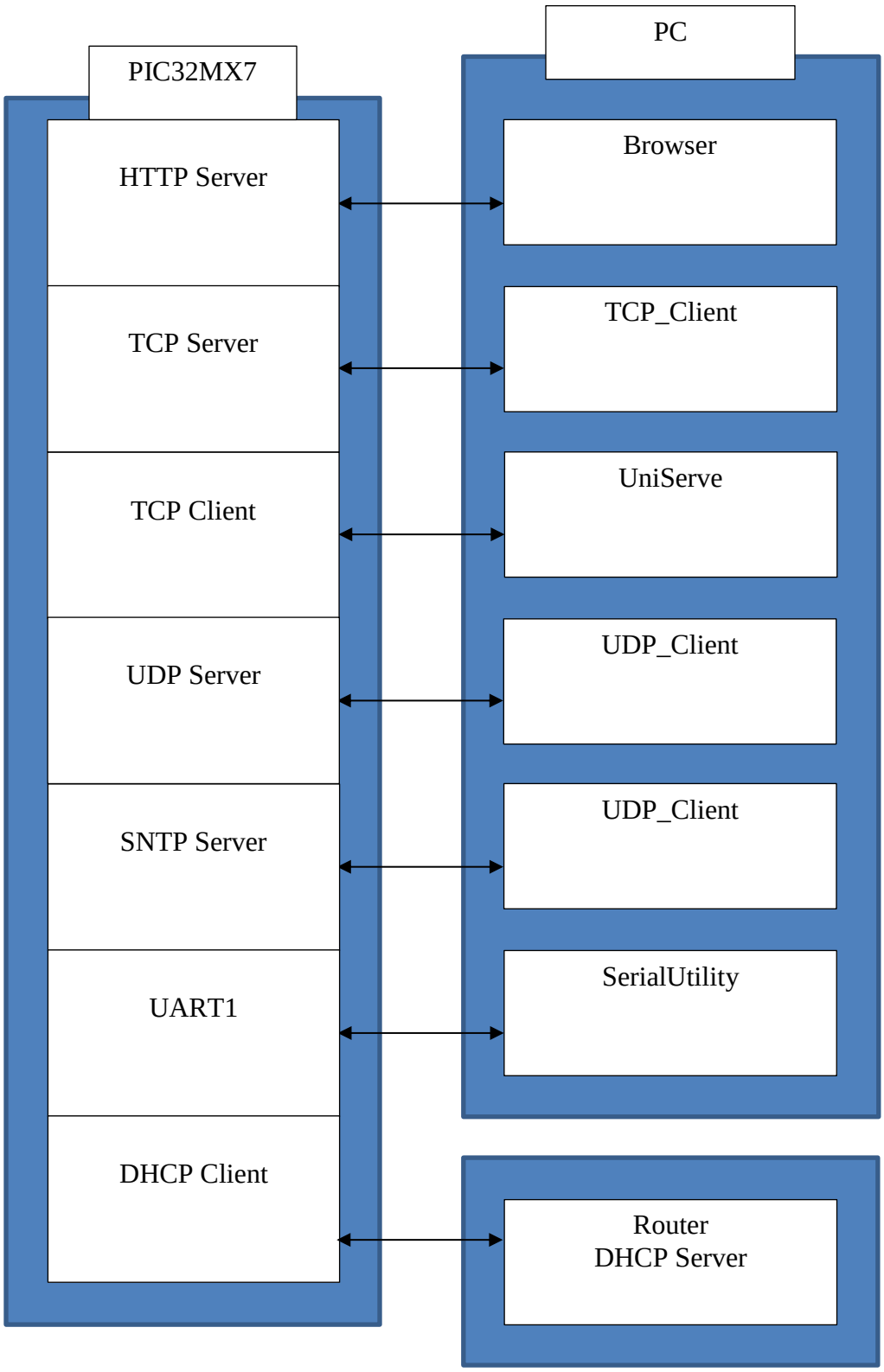
I only wish that MikroE would publish the source code for the libraries which would make the job much easier. It would also facilitate debugging when a problem arises.

Internet Protocols Running in the Board



HTTP server uses Port 80.
TCP server listens on Port 1234.
TCP client is used to push data to the TCP server (UniServe). UniServe is listening on Port 6000 and the client uses Port 6001.
DHCP uses Ports 67 and 68.
SNTP server uses Port 123 to listen for requests. Sends to remote Port as received.
UDP server listens Port 6060.

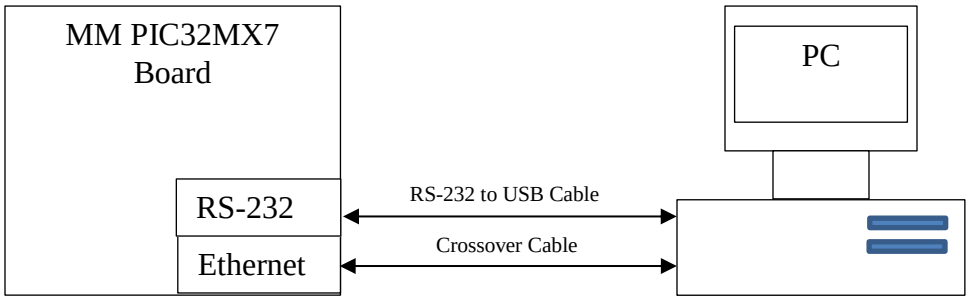
Utilities for Testing the Software



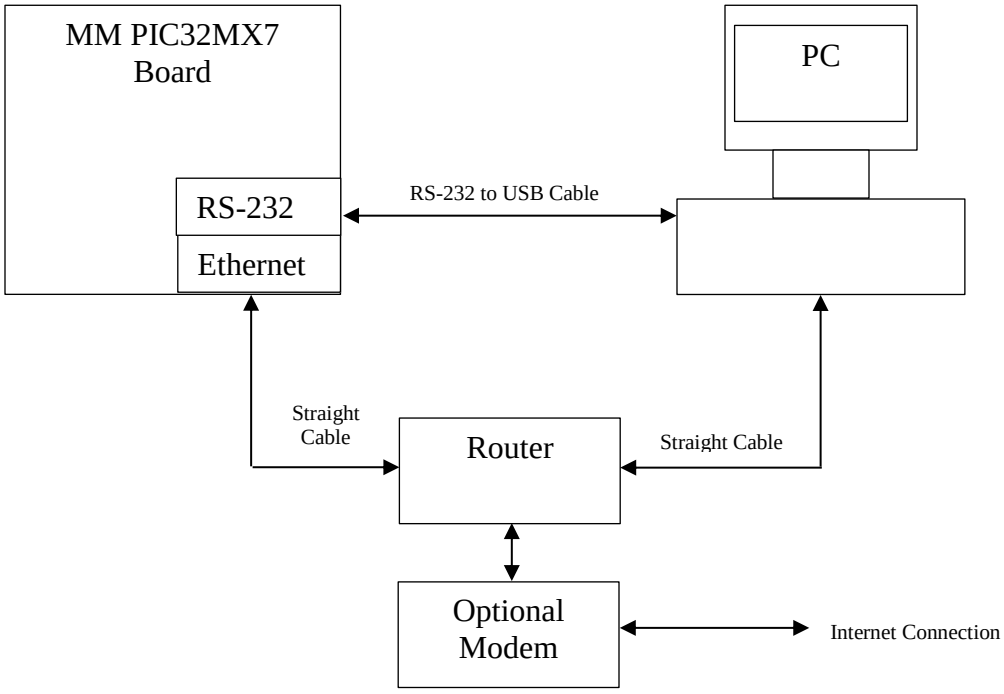
Hardware

I have used 2 configurations. Both work equally well.

Direct Connection to PC



Router



Applications

HTTP Server
The HTTP Server application is the MikroE example from "Network_Ethernet_PIC32 Library". I haven't figured out how to compile a web page into resources.h yet so I can't change the current web page.

TCP Server
This is just a stub and will respond with "ACK" to a transmission from the TCP Client.

TCP Client
This application has an input FIFO that will send data to the remote server (UniServe). UniServe saves the data to disk. It can service 8 connections simultaneously and sorts the data by creating folders with the client's IP address. The file stored in the folder has today's date for a file name. Right now a TCP "Push" event can be triggered by sending a 'c' command over the serial interface. In a real world application, the triggering event would be the closing of a switch or an A to D sample, etc.

UDP Server
The UDP Server is just another form of the serial interface. It responds to commands from the Client.

SNTP Server
This application is useful for providing time Synchronization over a local network. I have attempted to recreate the format documented in RFC4030 at http://tools.ietf.org/html/rfc4330 but I am not sure if I was entirely successful.

Serial Utility for UART1
This is an application that communicates with the serial interface. It is useful in debugging by sending out unsolicited information and status messages. It can also be used to trigger services in the micro. The debug messages have been left active. In real world applications these messages would be turned off using a compiler directive.

DHCP Client
This application communicates with a Router to obtain an IP address for the local network. It can then rout messages to and from the Internet.

Timer

This code uses the Core Timer available in PIC32MX7 to implement the clock keeping functions. It interrupts every millisecond to create a resolution of 1 millisecond.