WiZNet iMCU Design Contest 2010

June 30, 2010

Registration No: 003051

Project:

Light Weight Embedded DHCP Server (DHCPlite) (ABSTRACT)

Featuring iMCU7100EVB and DS1306 Realtime Clock Chip

Introduction

DHCP servers are used in professional environments to automatically allocate IP address and related parameters to equipments connected in a network environment. It requires skills to configure and manage DHCP Servers. Also, DHCP services takes significant amount of CPU time. An embedded DHCP with a simplified and most commonly used features will be of interest to a large segment of user, such as School, Hospitals, House, Clubs, etc. The motivation pf the project is based on these requirements. To implement such an application, we need a device with built in TCPIP and popular CPU core. iMCU is the best choice for such application. In this project, the author tries to show how such application can be developed on iMCU.

Description of the Project:

The main goals to be met by the DHCPlite are:

1. Should be easy for the user to manage and work with.
2. Meet the DHCP Clients in a small LAN Environment such as School, Colleges, Hospitals, Home, etc.
3. Should benefit by using the inbuilt TCPIP Core of W7100
4. Should fit into the Flash and Data memory available in W7100

The design specifications are as follows:

1. Maximum of 1024 clients will be supported.
2. Maximum of 4 Subnets are supported.
3. The lease period is available in two category:
    a. 24 hours lease period
    b. Infinite lease period (Static)
    c. DHCPlite tracks the clients. Initially every client is assigned with 24 Hour lease period. Each DHCP request from the client is counted and when this count exists 10 days, then the client gets an infinite lease.
4. User configures the DHCPlite with a PS2 Keyboard using simple commands
    a. Configure DHCP IP Address, Subnet Mask, Gateway, MAC Address
    b. Manage IP Pool – Create, Modify, Delete

Set and Modify the Realtime Clock

The serial port is used as a debug console to develop the application. On demand the internal registers and memory of W7100 can be dumped to the console. Once the application is developed, we do not require this, and hence not shown in the block diagram.

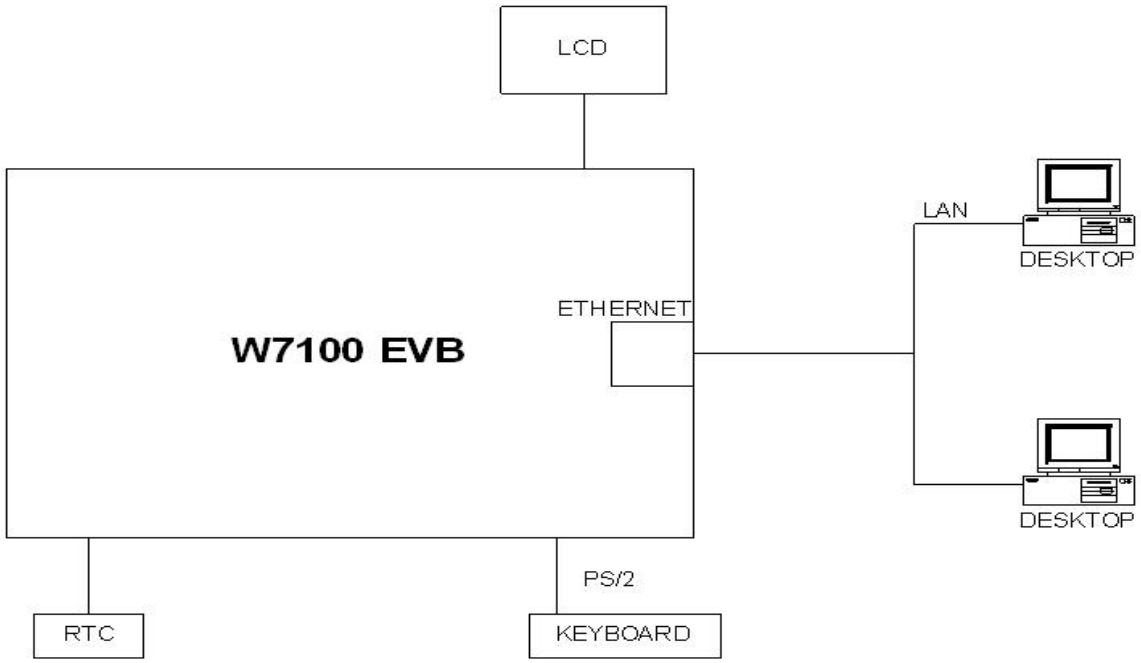
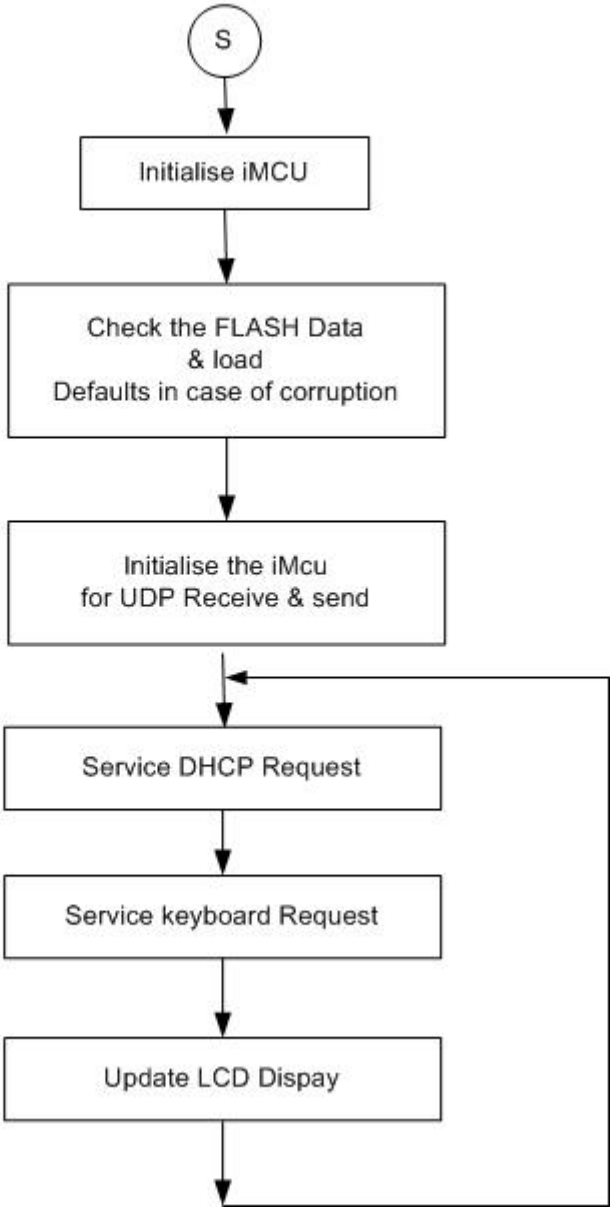


Figure 1

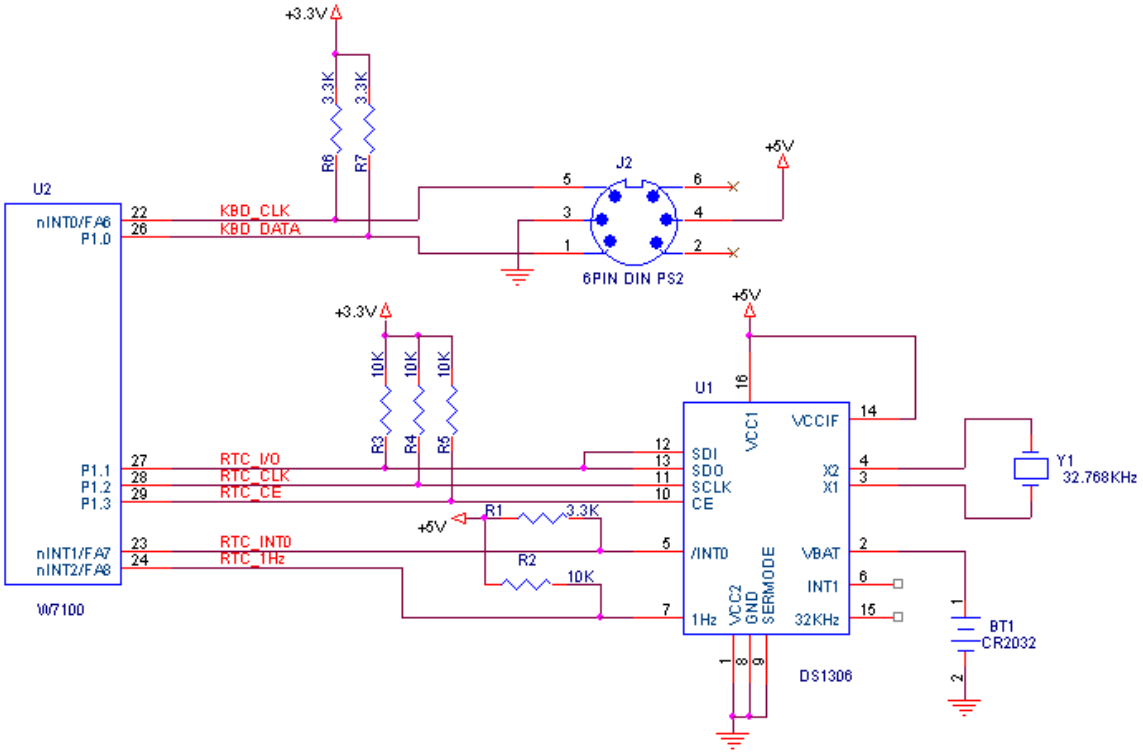
The basic flow chart for the DHCPlite is shown in Figure 2.
The main program, after initialisation performs the following tasks;

1. Service the DHCP Messages received by the clients
2. Service the Keyboard Requests to manage the DHCPlite
3. Update the LCD with relevant messages such as day and time, IP Address of the DHCPlite, and Commands to execute while configuring.

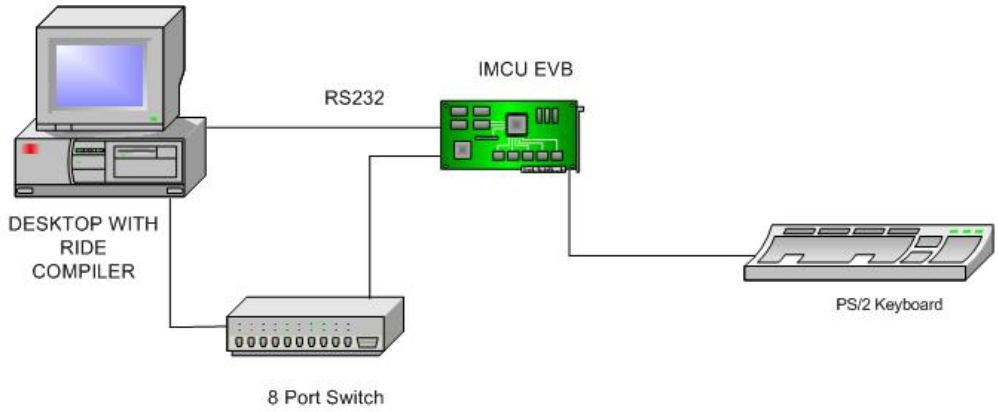
Main Flow Chart



Schematic:



DHCP lite Development System

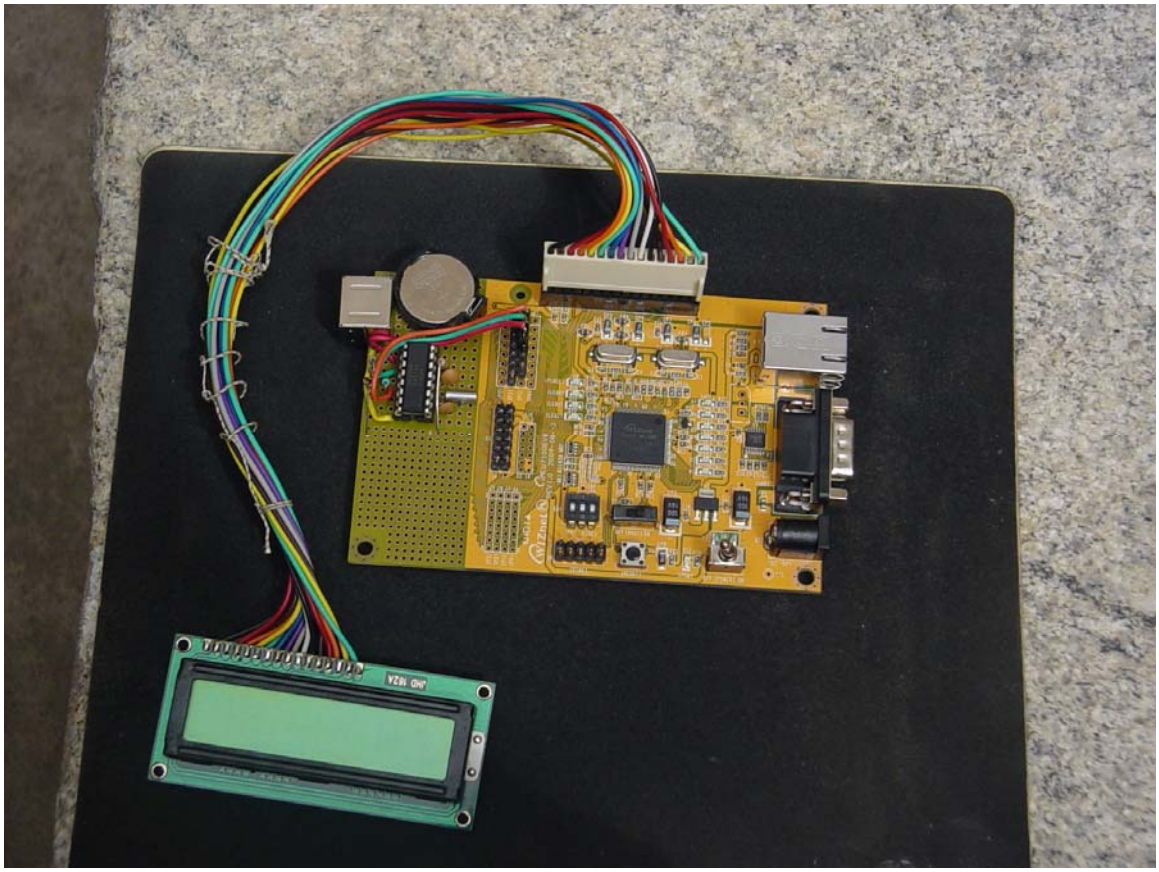


Development Environment:

The program is developed using RIDE 51(Raisonance Integrated Development Environment). However, the C source can be migrated to other environments such as Keil uVision, IAR etc. There is no specific reason to choose RIDE, other than the author's familiarity with this tool.

Sample Code and Photograph:
For a brief sample code. Please refer the file *dhcplitesample.c.*

A photograph showing the W7100EVB with RTC is shown in the *assembly.jpg*



Assembly Photo 1

Example Source Code

The sample code is of the main program is shown in Listing1.

Listing 1

```
#include "W7100.h"
#include "lcd.h"
#include "serial.h"
#include "flash.h"
#include "dhcplite.h"
#include "TCPIPCore.h"
#include "iinchip_conf.h"
#include "rtc.h"
#include "keyboard.h"
#include "socket.h"




code unsigned char InitialisingMsg[32]    = {"  Initialising      DHCPlite    "};
code unsigned char VersionMsg[32]         = {"  DHCPlite 1.0                  "};
code unsigned char ErrorMsg1[32]          = {"UDP FailTo Start   Try Again    "};
code unsigned char FailMsg[32]            = {" Write Fail                     "};
code unsigned char WIZnetMsg[32]          = {" WIZnet DHXCP Lite              "};
code unsigned char breakMsg[32]           =  {" Break Point                   "};

code unsigned char UDPInitMsg[] = "UDP Initialised \n\r";



//Function Prototypes are declared here
void dhcpserve(void);
void displayServe(void);


void initMCU(void);
void initflags(void);
void setClock(void);
void manageIPpool(void);
void checkNrestoredata();
unsigned char initUDP(void);
```

```
//void flashWrite(unsigned char* flashaddr, unsigned char*  memaddr);


unsigned char xdata inputBuffer[32];     // To store the characters from the
keyboard
unsigned char xdata tempBuff1[256];
unsigned char KBState;

code unsigned char *codememptr;

unsigned char UIflag;

int main(void)
{
    unsigned char temp;
    initRTCInterface();
     initLCD();              // LCD controller is initialised
     mov(dispBuff,(unsigned char *)InitialisingMsg,32);
     display(dispBuff);
     serialInit();           // Initialise the serial Port for 115200 baud
     enableKeyboard();       // PS2 Keyboard interface is intialised


     //Initialise the MCU
    initMCU();
    initflags();


     //Check the data stored in the Flash is not corrupted.  If corrupted then restore
the default.
     checkNrestoredata();


     //Initialise the UDP Port 67.  If intialisation fails, then Hang on
     if (initUDP() == 1){
        mov(dispBuff,(unsigned char *)ErrorMsg1,32);
        display(dispBuff);
        EA = 0;
        while(1);
     }

          mov(dispBuff,(unsigned char *)VersionMsg,32);
          display(dispBuff);
 /*The  main loop serves DHCP requests and Keyboard service */



    while(1){
       dhcpserve();
```

```
        keyboardServe();
        displayServe();
    }

        return (0);
}
```

Conclusion

The DHCPlite was developed and tested in a limited environment. The objective of this project to make use of the TCPIP Core and the most popular CPU core to demonstrate the ease of developing a very complex application was achieved.