Water Meter Reader Documentation

Neptune Pro Read Version

Software Code

Written in C++ Arduino V1.0.2 for Windows

Target Board: Ethernet with USB Serial Light Serial Adapter for programming

Release 1.0

January 14, 2013

Reference:Pin Map

Pin 0	Serial Tx
Pin 1	Serial Rx
Pin 2	
Pin 3	
Pin 4	SD card CS (SPI Select)
Pin 5	
Pin 6	Rx Read Meter Data
Pin 7	Tx Clock to Meter
Pin 8	Relay
Pin 9	LED
Pin 10	Ethernet SPI (SS)
Pin 11	Ethernet SPI (MOSI)
Pin 12	Ethernet SPI (MISO)
Pin 13	Ethernet SPI (SCK)
Analog pins 1-6	

Initialization –

Definitions, libraries incuded, Global declarations

Void (setup) –

Initialize Serial Port, SD Card, Ethernet Client, Server.

IP: 192.168.17.176 [May be DHCP from Rozeware Checkbox park intranet server] See intranet setup

Void()

Prog Exec – once every hour

- Engage Relay contact with water meter (on for < 0.5 Sec)
- Void(GetTime) Get internet time from UDP NTP
- Void(ReadMeter) Read Water Meter
- Void(SDCardWrite) Write date, time, Water Meter Reading to SD Card
- Disengage Relay contact with water meter
- Monitor Client request for data
    - Send SD Data file

```
GetTime()

Disable SD
Enable Ethernet
Set UDP listen port to 8888
Set Ethernet client as DNS Client
Request Time Server IP address

Create NTP packet request
Send NTP request to DNS Server (pool.ntp.org)
Get UDP packet from assigned  DNS  server
Decode NTP  time
    -   Add day to Microsoft date code
    -   Decode hour and minute
```

```
ReadMeter()

Put clock on Tx line for 0.3 Seconds - Then
Find frame
Decode 0x0100 as 1
Decode 0x0111 as 0
Get 4 bits as nibble
Decode 4 nibbles as digit
Read 28 digits
```

```
SDCardWrite()


Disable Ethernet
Enable SD Card
Decode Read String
    -   Find Byte start Digit
    -   Write string Start Digit + Offset
            - 7,8,9,10,5,6
Open SD
Write SD
Close SD
Disable SD
```

Client Monitor

When Client Requests data;
Enable Ethernet as server
Respond to establish handshake with content type
Open SD data file DateData.TXT
Write data to Ethernet
Close SD Card
Disconnect Client

Code on Arduino Ethernet Card

```
/*
  Web Server, Web client, water meter reader, MicroSD card
 File:  DateData.txt

 Circuit:
 Ethernet shield attached to pins 10, 11, 12, 13
 SD Enable attached to pin 4
 TxClk pin 7
 RxData pin 6
 led pin 9
Relay pin 8

 */

#include <SPI.h>
#include <Ethernet.h>
#include <EthernetUdp.h>
#include <Dns.h>
#include <SD.h>

#define RxData 6
#define TxClock 7
#define led 9
#define Relay  8
//////////////////Ethernet Setup
// Enter a MAC address for  controller .
```

```
byte mac[] = {
  0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
byte ip[] = {
  192,168,17,176 };   // ip in LAN
unsigned int localPort = 8888;      // local port to listen for UDP packets

IPAddress timeServer; // pool.ntp.org NTP server
const int NTP_PACKET_SIZE= 48; // NTP time stamp is in the first 48 bytes of the message
byte packetBuffer[ NTP_PACKET_SIZE]; //buffer to hold incoming and outgoing packets
EthernetServer server(80);
EthernetClient client;

EthernetUDP Udp;        // A UDP instance to let us send and receive packets over UDP


/////////////////////////////Read Meter declarations
int meterData[60];                          // 60 integer Read Data
int meterByte[60];                          // 60 Integer Buffer for bit read data manipulation
int mData = 0;
int nibble = 3;
unsigned long previousMillis = 0;       // For Delay between NTP reads
unsigned long interval = 3600000;      // 1 Hour Time delay between NTP reads
unsigned int Current_Date = 0;
unsigned int Current_Hour = 0;
unsigned Current_Minute = 0;
int comp = 0b0000000001;                // comparator to invert read bit
int bitRate=52;                             //  52 = clock phase rate - clock 9600 Hz
int errorcode = 0;
int j=0;




void setup() {
  Serial.begin(9600);

  pinMode (RxData,INPUT_PULLUP);      //  Meter Read Pin
  pinMode (TxClock,OUTPUT);              // clock Pin
  pinMode (led,OUTPUT);
  pinMode (Relay,OUTPUT);

  // disable w5100 while setting up SD
  pinMode(10,OUTPUT);
  digitalWrite(10,HIGH);
  Serial.print(F("Starting SD.."));
  if(!SD.begin(4)) Serial.println(F("failed"));
  else Serial.println(F("ok"));

  Ethernet.begin(mac, ip);
  digitalWrite(10,HIGH);

  //delay(2000);
  server.begin();
  Serial.println(F("Ready"));                    //write all serial from flash

}
void loop() {
  unsigned long currentMillis = millis();
  if (currentMillis - previousMillis > interval) {
    previousMillis = currentMillis;
    digitalWrite(Relay,HIGH);
    delay (200);
    GetTime();                                   // call GetTime function
    Serial.println(F("Got time"));

    MeterRead();                                 // get meter data
    Serial.println(F("Got Meter Read"));

    SDcardWrite();                           // output the 'Water Meter Reading' formatted for spreadsheet
    Serial.println(F("Wrote to SD"));
    digitalWrite(Relay,LOW);
    return;
  }

  // Create a client connection

    EthernetClient client = server.available();
  if (client) {
    while (client.connected()) {
      if (client.available()) {
        char c = client.read();
        //Serial.print(c);

        if (c == '\n') {                               //if HTTP request has ended
```

```
          client.println(F("HTTP/1.1 200 OK"));          //send new page
          // client.println("Content-Type: text/html");
          client.println(F("Content-Type: text"));
          client.println();

          File myFile = SD.open("DateData.TXT");
          delay(1);
          if (myFile) {

            while (myFile.available()) {
              client.write(myFile.read());
            }
            myFile.close();
          }
          delay(1);
          //stopping client
          client.stop();
        }
      }
    }
  }
}
```

void MeterRead()

```
{
  digitalWrite(led, HIGH);
  errorcode = 0;
  //counter =(counter++);                              // loop counter for testing

  // Generate clock prior to read data for x seconds

  for (int i=0;i<3000;i++){         // 0.312 sec pre-clock - 0.208 consistent but bad data bits 1,2 and 5

    digitalWrite(TxClock,HIGH);
    delayMicroseconds(bitRate);
    digitalWrite(TxClock,LOW);
    delayMicroseconds(bitRate);

  }



  // find frame bit
```

```
for(int i=0; i < 32; i++){                    // Compare hi -lo data until frame found

 digitalWrite(TxClock,HIGH);
 delayMicroseconds(bitRate);
 int d0 =digitalRead(RxData);

 digitalWrite(TxClock,LOW);
 delayMicroseconds(bitRate);
 int d1 =digitalRead(RxData);

 if (d0 != d1)
 {
  break;
 }
}                                             // end find frame bit


// Read 256 bits, convert to 4 and 7 nibbles, translate to 1 and 0 nibbles, invert for DEC write


for(mData=0; mData < 28; mData++){            // 28  data nibbles
 for (nibble=0; nibble <4; nibble++){         // 4 bit nibbles convert 4's to 1's; 7's to 0's
  for(int x=0; x<4; x++){                     // decode 4's and 7's

   digitalWrite(TxClock,HIGH);
   delayMicroseconds(52);
   int nibblebit =digitalRead(RxData);        // read on this phase to get 0 into frame bit - 16th.

   digitalWrite(TxClock,LOW);
   delayMicroseconds(52);

   bitWrite(meterByte[mData],x,(nibblebit ^ comp)); // invert data bit  for transistor inversion
  }

  if (meterByte[mData] == 4){                 // meterData is decoded 0 or 1
   bitWrite(meterData[mData],nibble,1);       // write MSB to left

  }
  else if (meterByte[mData] ==7){
   bitWrite(meterData[mData],nibble,0);
  }
  else {
    errorcode =1;                          // need to print error code. 0 means good data
   }

  }
 }
 digitalWrite(TxClock,HIGH);              // put low on meter
 digitalWrite(led, LOW);

}


void SDcardWrite() {                  //Write data to SDCardFile

 // disable w5100 SPI while starting SD
 pinMode(10,OUTPUT);
 digitalWrite(10,HIGH);
 Serial.print(F("Starting SD.."));
 SD.begin(4);

 String dataString = "";

 /* data format of meter is 16 nibbles. After F start 4 0xE's
  6 data and 5 0xE's. Data below is position of data bits */

 for( int i=0; i<16;i++){
  if (meterData[i]==15){
   j=i;
  }
 }
 dataString += Current_Date;
 dataString += ",";
 dataString += Current_Hour;
 dataString += ":";
 if (Current_Minute < 10 ) {    // In the first 10 minutes of each hour, we'll want a leading '0'
  dataString += "0";
 }
 dataString += Current_Minute;
 dataString += ",";
 dataString += meterData[j+7];
 dataString += meterData[j+8];
 dataString += meterData[j+9];
 dataString += meterData[j+10];
```

```
  dataString += meterData[j+5];
  dataString += (".");
  if (meterData[j+6] != 14)
  {
    dataString +=meterData[j+6];
  }
  else
  {
    dataString +="0";
  }
  dataString += "\0"; // null


  // Serial.println("Initalizing SD");
  File dataFile = SD.open("DateData.txt", FILE_WRITE);  // open the file

  if (dataFile) {                                  // if the file is available, write to it:
    dataFile.println(dataString);

    dataFile.close();                              // close Data file
    Serial.println(dataString);                   // print to the serial port too:
  }
  dataString ="";
  // disable SD SPI
  //pinMode(4,OUTPUT);
  digitalWrite(4,HIGH);
}

void GetTime(){

  // disable SD SPI while starting w5100
  pinMode(4,OUTPUT);
  digitalWrite(4,HIGH);
  Serial.println(F("Starting w5100.."));
  pinMode(10,OUTPUT);
  digitalWrite(10,HIGH);

  Udp.begin(localPort);
  DNSClient dns;
  dns.begin(Ethernet.dnsServerIP());
  if(dns.getHostByName("pool.ntp.org",timeServer)) {
    Serial.println(timeServer);
```

```
 }

 sendNTPpacket(timeServer);                    // send an NTP packet to a time server

 delay(1000);                                   // wait to see if a reply is available
 if ( Udp.parsePacket() ) {                     // We've received a packet, read the data from it
  Udp.read(packetBuffer,NTP_PACKET_SIZE);  // read the packet into the buffer
  unsigned long highWord = word(packetBuffer[40], packetBuffer[41]);
  unsigned long lowWord = word(packetBuffer[42], packetBuffer[43]);

  unsigned long secsSince1900 = highWord << 16 | lowWord;
  const unsigned long seventyYears = 2208988800UL;
  unsigned long epoch = secsSince1900 - seventyYears;

  Current_Date = (41244 +((epoch - 1354341600) / 86400));       // set for DST
  Current_Hour = (((epoch - 18000)  % 86400L) / 3600);            // print the hour -5 Hrs from UTC
  Current_Minute = ((epoch  % 3600) / 60);         // print the minute (3600 equals secs per minute)

 }
}

// send an NTP request to the time server at the given address
unsigned long sendNTPpacket(IPAddress& address)
{

 memset(packetBuffer, 0, NTP_PACKET_SIZE);                       // set all bytes in the buffer to 0
 packetBuffer[0] = 0b11100011;                                    // LI, Version, Mode
 packetBuffer[1] = 0;                                             // Stratum, or type of clock
 packetBuffer[2] = 6;                                             // Polling Interval
 packetBuffer[3] = 0xEC;                                          // Peer Clock Precision
 // 8 bytes of zero for Root Delay & Root Dispersion
 packetBuffer[12]  = 49;
 packetBuffer[13]  = 0x4E;
 packetBuffer[14]  = 49;
 packetBuffer[15]  = 52;

 // all NTP fields have been given values, now
 // you can send a packet requesting a timestamp:
 Udp.beginPacket(address, 123); //NTP requests are to port 123
 Udp.write(packetBuffer,NTP_PACKET_SIZE);
 Udp.endPacket();
}
```