243-848-92 – Computer Project

Progress Report #1

Calvin Ouellet-Ference
A711643
Mr. Markou
Submitted on the 18th of February 2010

1. Objectives

The objectives of of the last four weeks was to get the Ethernet controller working.

2. Progress

2.1 Ethernet Controller

The Ethernet controller was redesigned yet again. The (hopefully) final design for it is using the CS8900A-H pre-assembled board. It is bus compatible with the x86 bus. It can operate in different modes but because it is pre-assembled, only one mode can be used and that is IO Mode. It has the option of 8-bit and 16-bit operation (even 32-bit) but in my case only the 8-bit operation is used. Some problems have occurred in both the transmission and the reception of packets. The transmission worked fine in the fact that all the data bytes were present as seen through a wireshark capture. However for some reason wireshark would put a note on the packet as “Malformed UDP” which I find odd as I've built it correctly as through the RFC-768. However the Ethernet frame and the IP frame both seem to work perfectly. To resolve the problem of the “Malformed UDP” were to use the pcap library as it was more intuitive to use this library then to use Unix raw sockets. However I did not verify if the packets I would receive through a standard port but I assumed that I would not work as wireshark had problems with the packets itself, though my assumption may prove incorrect, the way I'm using works fine. As for the reception well the Ethernet controller on the minimal system would detect a packet that would arrive but be unable to read anything from it except on the occasion a snippet of data here and there. The solution to this would to be in my protocol for issuing commands I would sent an certain amount of pings to do one command to the minimal system would count these and acknowledge them as well.

2.2 Coffee Machine

The major work for the coffee machine side of the project is done which really was not that much. It is just a simple on/off switch using the PPI of the minimal system to a voltage follower then to a transistor acting as a switch to drive on or off the coil of a 12V relay.

2.3 Computer Application

As mention above, the application is actually using raw sockets for receiving packets and filtering through them to pin point exactly which packet is from the coffee machine. This will change later on, but at this moment I'm monitoring the port numbers to pick up upon. I will change this later on to not only monitor the port number but also the source and destination MAC addresses and the IP header. IP header would be a good idea as it would also be “proper” decapsulation instead of skipping from MAC to Ports and totally ignoring the network layer. As for transmitting packets, at first I was thinking of using pings however I've failed to remember that ping requests actually wait for a reply... so to fix this I added a 1 second timeout which seem to have fixed the problem but does not work properly when it does not pick up any reply from the minimal system. It assume it's just my loops that are not done properly which is the only thing I can come up with.

2.4 Power Supply

The power supply was designed and just needs to get verified to see if I'm not forgetting anything. It was designed to have +5V and +12V which is what is required by this project.

3. Circuits Schematics

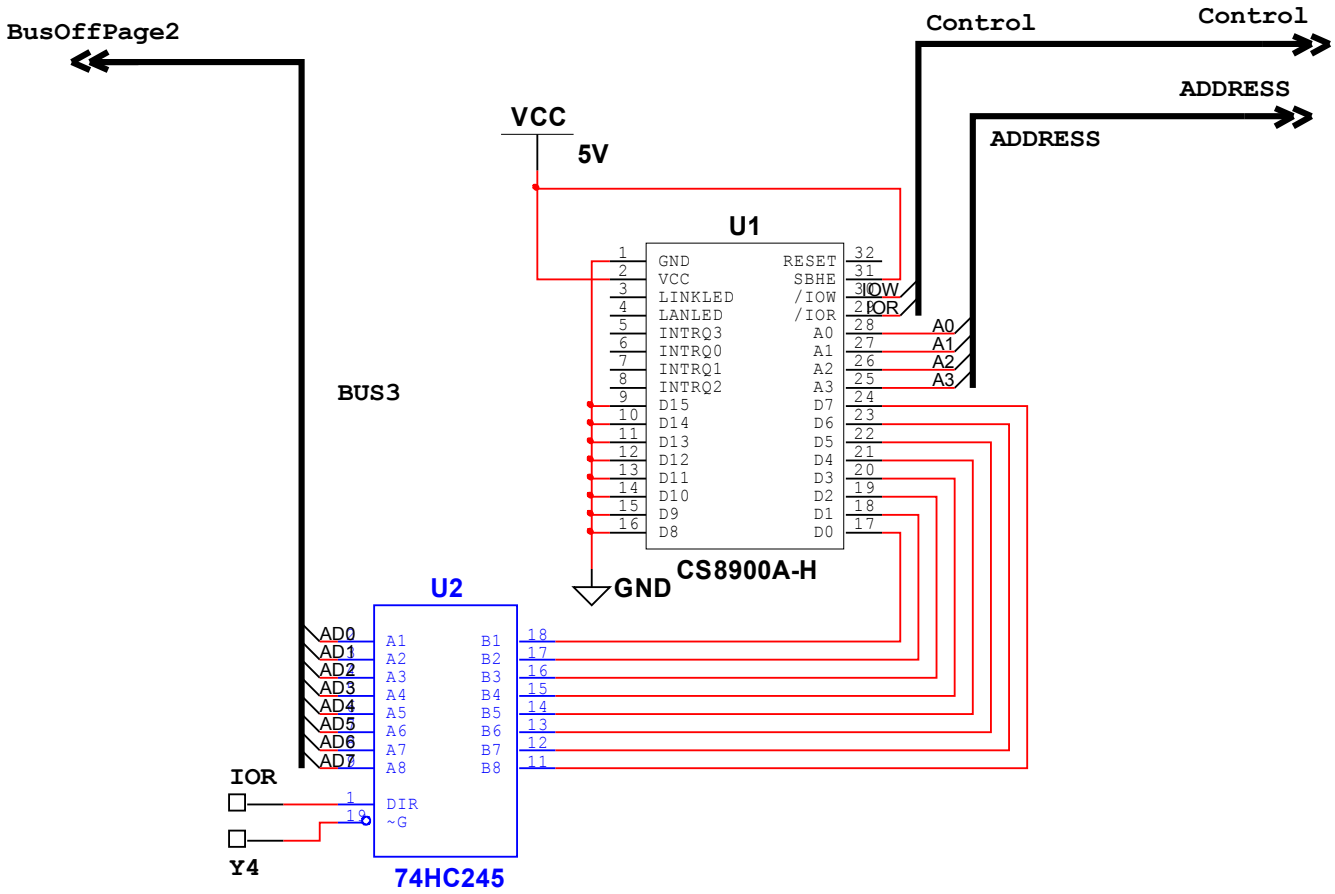

Figure 3.1 – Schematic of the Ethernet Controller (CS8900A-H)

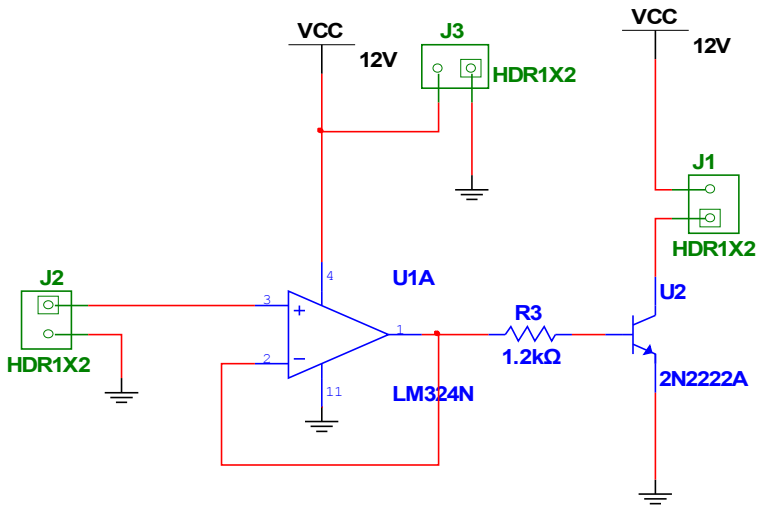

Figure 3.2 – Schematic of the Control Switch

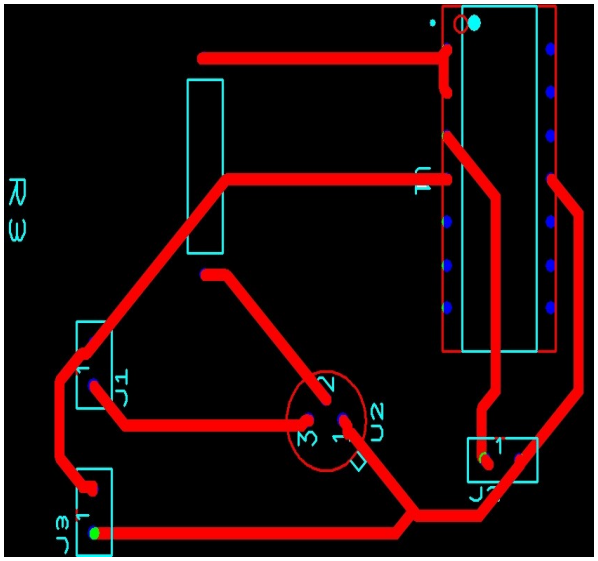

Figure 3.3 – PCB Layout of the Control Switch

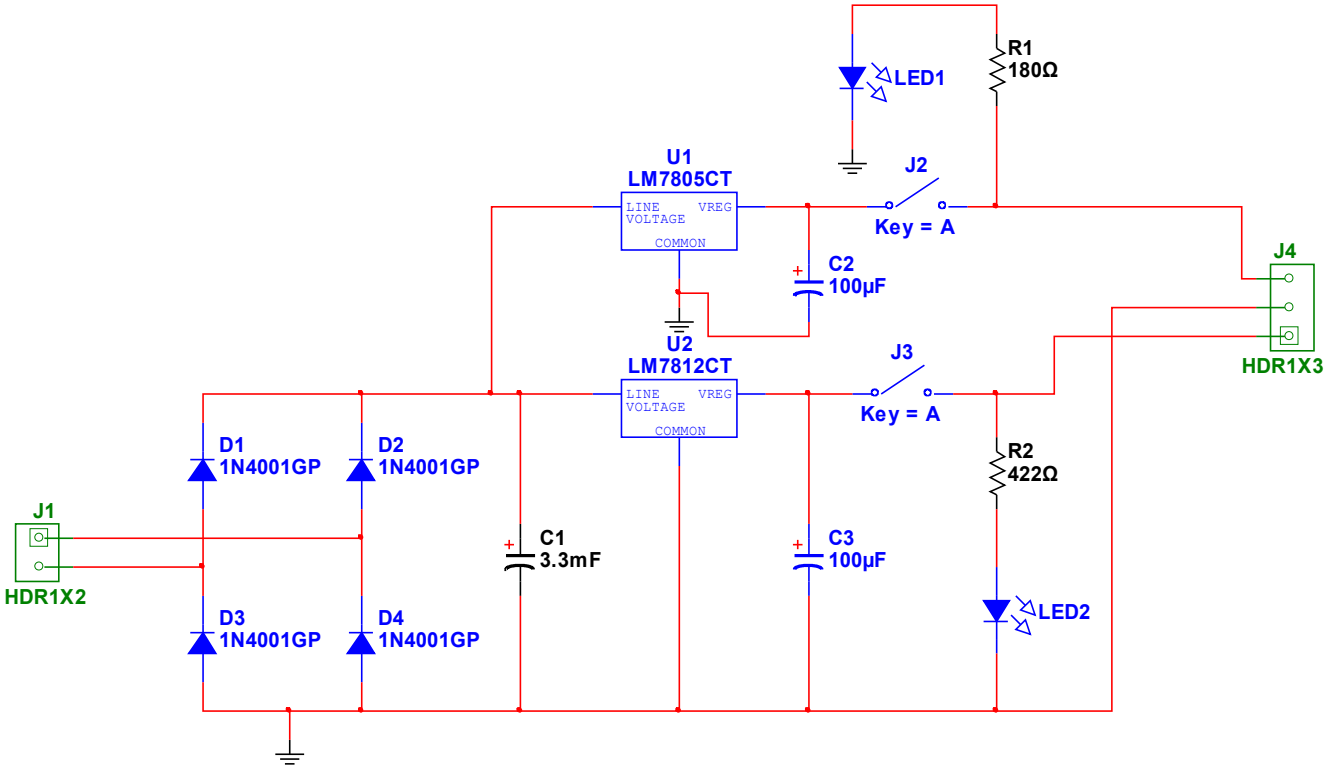

Figure 3.4 – Power Supply Unit Schematic

4. Code

Instead of just putting on big block of code, I will briefly describe each function or subroutine of the code as it goes. It is still in order just cut up in blocks.

4.1 Minimal System Protocol

The following is the ASM pre-processor and essentially the boot-up sequence, where it initializes the PPI, the PIC and the Ethernet Controller. PPI and PIC are standard initialization as for the Ethernet controller (eth0) it is initializing the MAC address and the registers for receiving and transmitting correctly (LineCTL and TestCTL).

```
.model tiny
.code
; external commands (m88io.obj) preprocessor
EXTRN        newline:NEAR, outbyte:NEAR, outword:NEAR, getc:NEAR, outc:NEAR,
outstr:NEAR
;eth0 preprocessor
ETH0          equ    0A00h
RXTX0L              equ          ETH0+00h
RXTX0H              equ          ETH0+01h
RXTX1L              equ          ETH0+02h
RXTX1H              equ          ETH0+03h
TXCMDL              equ          ETH0+04h
TXCMDH              equ          ETH0+05h
TXLENGTHL           equ          ETH0+06h
TXLENGTHH           equ          ETH0+07H
ISQL          equ            ETH0+08h
ISQH          equ            ETH0+09H
PPPL          equ            ETH0+0AH
PPPH          equ            ETH0+0BH
PPD0L         equ            ETH0+0CH
PPD0H               equ          ETH0+0DH
PPD1L         equ            ETH0+0EH
PPD1h         equ            ETH0+0FH
; I/O preprocessor
LED                 equ          0F00h
PPI                 equ          0200h
PORTA         equ    PPI+0
PORTB         equ    PPI+1
PORTC         equ    PPI+2
PPICTL        equ    PPI+3
; PIC preprocessor
PIC           equ    400h
ICW1    equ    PIC
ICW2    equ    PIC+1
ICW4    equ    PIC+1
OCW1    equ    PIC+1
ICW1B   equ    00010011b   ;edge trig
```

```
ICW2B   equ  01000000b   ;vec. no. 40h = 100h
ICW4B   equ  00000011b
OCW1B   equ 11111110b
FREQ equ     2400

      org    0800h
main   proc
init_ppi:
                cli
                mov             ax,0
      mov    al,10010000b    ;configuration word for the 8255
                        ;both group A and B = mode 0
                        ;port A = input
                        ;port B = output
                        ;port C = output
      mov    dx,PPICTL
      out    dx,al        ;send the configuration word
                mov             dx,0
                mov             cx,0
init_pic:
                lea     di,rtc
                mov     ds:[100h],di
                mov     ax,0
                mov     ds:[102h],ax

                mov     dx,ICW1
                mov     al,ICW1B
                out     dx,al

                mov     dx,ICW2
                mov     al,ICW2B
                out     dx,al

                mov     dx,ICW4
                mov     al,ICW4B
                out     dx,al

                mov     dx,OCW1
                mov     al,OCW1B
                out     dx,al

reset_wait:                             ; Just to give eth0 enough time to
                inc             cx              ; init internally
                nop
                nop
                nop
                nop
                nop
                nop
```

```
                nop
                nop
                cmp             cx,07fffh
                jbe             reset_wait
                mov             dx,LED
                mov             al,01h
                out             dx,al
eth0_init:
;MAC_INIT
                mov             dx,PPPL
                mov             al,12h
                out             dx,al
                mov             dx,PPPH
                mov             al,01h
                out             dx,al
                mov             dx,PPD0L
                mov             al,0D3h
                out             dx,al
                mov             dx,PPD0H
                mov             al,00h
                out             dx,al
                mov             dx,PPPL
                mov             al,58h
                out             dx,al
                mov             dx,PPPH
                mov             al,01h
                out             dx,al
                mov             dx,PPD0L
                mov             al,000h
                out             dx,al
                mov             dx,PPD0H
                mov             al,6Fh
                out             dx,al
                mov             dx,PPPL
                mov             al,5Ah
                out             dx,al
                mov             dx,PPPH
                mov             al,01h
                out             dx,al
                mov             dx,PPD0L
                mov             al,066h
                out             dx,al
                mov             dx,PPD0H
                mov             al,66h
                out             dx,al
                mov             dx,PPPL
                mov             al,5Ch
                out             dx,al
                mov             dx,PPPH
```

```
                mov             al,01h
                out             dx,al
                mov             dx,PPD0L
                mov             al,065h
                out             dx,al
                mov             dx,PPD0H
                mov             al,65h
                out             dx,al
;TestCTL
                mov             dx,PPPL
                mov             al,18h
                out             dx,al
                mov             dx,PPPH
                mov             al,01h
                out             dx,al
                mov             dx,PPD0L
                mov             al,00h ;10011001b
                out             dx,al
                mov             dx,PPD0H
                mov             al,00h ;01000000b
                out             dx,al
; LineCTL
                mov             dx,PPPL
                mov             al,12h
                out             dx,al
                mov             dx,PPPH
                mov             al,01h
                out             dx,al
                mov             dx,PPD0L
                mov             al,0D0h
                out             dx,al
                mov             dx,PPD0H
                mov             al,00000000b
                out             dx,al

                mov             dx,PPPL
                mov             al,04h
                out             dx,al
                mov             dx,PPPH
                mov             al,01h
                out             dx,al
                mov             dx,PPD0L
                mov             al,01000000b
                out             dx,al
                mov             dx,PPD0H
                mov             al,00111001b
                out             dx,al

                call    display_mac
```

```
poll:
                mov             dx,PPPL
                mov             al,24h
                out             dx,al
                mov             dx,PPPH
                mov             al,01h
                out             dx,al
                mov             dx,PPD0L
                in              al,dx
                mov             tmp,al
                mov             dx,PPD0H
                in              al,dx
                mov             ah,al
                mov             al,tmp
                mov             wtmp,ax

                and             ax,0ff0fh
                cmp             ax,2304h
                jne             poll
                ;mov    ax,wtmp
                ;call   outword
        ;       call    newline
                ;call   send
                call    recv
                cmp             flag,1
                je              poll
                call    send
here:   jmp             poll

rtc:
                inc             tick
                cmp             tick,FREQ
                jbe             idone
                mov             tick,0
                inc             time
idone:
                iret
main    endp
```

The following code is the receive subroutine which when a packet has been received by the MAC and needs processing. It uses the RTC as the main indicator of when to get out of the subroutine, however it probably needs to be increased from 10 seconds to something higher. When the minimal system is executing this subroutine, it is counting the amount of pings it receives (not fully implemented, which is why I need to increment the timer) so it can return it back to the main for further processing.

```
recv    proc
                sti
```

```
                inc             p_cnt
recv_poll:
                cmp             time,10 ; 10 Seconds
                jae             shi

                mov             dx,PPPL
                mov             al,24h
                out             dx,al
                mov             dx,PPPH
                mov             al,01h
                out             dx,al
                mov             dx,PPD0L
                in              al,dx
                mov             tmp,al
                mov             dx,PPD0H
                in              al,dx
                mov             ah,al
                mov             al,tmp
                and             ax,0f0ffh
                cmp             ax,2044h
                jne             recv_poll
                inc             p_cnt
                mov             time,0
                jmp             recv_poll
shi:
                cli
                call    newline
                lea             di,recv_msg
                call    outstr
                mov             al,p_cnt
                call    outbyte
                call    newline
                mov             p_cnt,0
                mov             time,0

                mov             data,0A0h
                mov             data+1,0A0h
;               call    send
                mov             flag,0
                ret
recv    endp
```

This is just a simple subroutine to display the MAC address of the system, it was originally designed to test out if writing to the internal registers of the CS8900 worked out correctly but I've kept it and using it as a display prompt for the hyperterminal session. However due to some problem with loopings which I really did not take the time to figure out as I had more pressing matters to attend to, no loops were used in the subroutine and the next because I had no time to figure out. So the long way of going at it was done...

```
display_mac  proc
             lea        di, mac
             mov        dx,PPPL
             mov        al,58h
             out        dx,al
             mov        dx,PPPH
             mov        al,01h
             out        dx,al
             mov        dx,PPD0L
             in         al,dx
             mov        [di],al
             inc        di
             mov        dx,PPD0H
             in         al,dx
             mov        [di],al
             inc        di
             mov        dx,PPPL
             mov        al,5Ah
             out        dx,al
             mov        dx,PPPH
             mov        al,01h
             out        dx,al
             mov        dx,PPD0L
             in         al,dx
             mov        [di],al
             inc        di
             mov        dx,PPD0H
             in         al,dx
             mov        [di],al
             inc        di
             mov        dx,PPPL
             mov        al,5Ch
             out        dx,al
             mov        dx,PPPH
             mov        al,01h
             out        dx,al
             mov        dx,PPD0L
             in         al,dx
             mov        [di],al
             inc        di
             mov        dx,PPD0H
```

```
                in              al,dx
                mov             [di],al
                inc             di
                lea             di, mesg_mac
                call    outstr
                lea             di,mac
                mov             al,[di]
                call    outbyte
                mov             al,3Ah
                call    outc
                inc             di
                mov             al,[di]
                call    outbyte
                mov             al,3Ah
                call    outc
                inc             di
                mov             al,[di]
                call    outbyte
                mov             al,3Ah
                call    outc
                inc             di
                mov             al,[di]
                call    outbyte
                mov             al,3Ah
                call    outc
                inc             di
                mov             al,[di]
                call    outbyte
                mov             al,3Ah
                call    outc
                inc             di
                mov             al,[di]
                call    outbyte
                ret
display_mac     endp
```

This is my subroutine to send out an UDP packet. It's very straightforward but as mentioned above, some sort of problem occurred when I was looping so I just went at it the long way. The routine essentially starts off by setting the TxCMD register to begin transmission after the last byte has been written then the TxLENGTH register specified the size of the packet and then the process of moving one byte at a time through the RxTxData port 0 starts until all the frame has been transferred to the CS8900.

```
send    proc
;setting up the TxCMD
                mov             dx,TXCMDL
                mov             al,0C0h
                out             dx,al
```

```
                mov             dx,TXCMDH
                mov             al,00h
                out             dx,al
;setting up the TxLength
                mov             dx,TXLENGTHL
                mov             al,78h ;2Bh             ;[LENGTH!!!!!]
                out             dx,al
                mov             dx,TXLENGTHH
                mov             al,00h
                out             dx,al
;Packet Page Pointer Set-up
PPP:
                mov             dx,PPPL
                mov             al,38h
                out             dx,al
                mov             dx,PPPH
                mov             al,01h
                out             dx,al
;Reading the Packet Page Pointer Data

                mov             dx,PPD0H
                in              al,dx
                and             al,01h
                cmp             al,01h
                jne             PPP
        ;       lea             di,udp_hdr
                mov             cl,0
;start moving data
tx_data:
                ;destination MAC
                mov             dx,RXTX0L
                mov             al,00h
                out             dx,al
                mov             dx,RXTX0H
                mov             al,26h
                out             dx,al
                mov             dx,RXTX0L
                mov             al,2dh
                out             dx,al
                mov             dx,RXTX0H
                mov             al,7ch
                out             dx,al
                mov             dx,RXTX0L
                mov             al,073h
                out             dx,al
                mov             dx,RXTX0H
                mov             al,0b5h
                out             dx,al
                ;Source MAC
```

```
mov             dx,RXTX0L
mov             al,43h
out             dx,al
mov             dx,RXTX0H
mov             al,6fh
out             dx,al
mov             dx,RXTX0L
mov             al,66h
out             dx,al
mov             dx,RXTX0H
mov             al,66h
out             dx,al
mov             dx,RXTX0L
mov             al,65h
out             dx,al
mov             dx,RXTX0H
mov             al,65h
out             dx,al
;type
mov             dx,RXTX0L
mov             al,08h
out             dx,al
mov             dx,RXTX0H
mov             al,00h
out             dx,al
;ip hdr
;version, header length
mov             dx,RXTX0L
mov             al,45h
out             dx,al
;services
mov             dx,RXTX0H
mov             al,00h
out             dx,al
mov             dx,RXTX0L
mov             al,00h
out             dx,al
;total length
mov             dx,RXTX0H
mov             al,14h
out             dx,al
;ID
mov             dx,RXTX0L
mov             al,01h
out             dx,al
mov             dx,RXTX0H
mov             al,40h
out             dx,al
;Flags
```

```
mov             dx,RXTX0L
mov             al,00h
out             dx,al
;Fragment Offset
mov             dx,RXTX0H
mov             al,00h
out             dx,al
;TTL
mov             dx,RXTX0L
mov             al,05h
out             dx,al
;protocl
mov             dx,RXTX0H
mov             al,11h
out             dx,al
;Checksum
mov             dx,RXTX0L
mov             al,031h
out             dx,al
mov             dx,RXTX0H
mov             al,0b8h
out             dx,al
;Source Address
mov             dx,RXTX0L
mov             al,0c0h
out             dx,al
mov             dx,RXTX0H
mov             al,0a8h
out             dx,al
mov             dx,RXTX0L
mov             al,00h
out             dx,al
mov             dx,RXTX0H
mov             al,0e1h
out             dx,al
;Destination Address
mov             dx,RXTX0L
mov             al,0c0h
out             dx,al
mov             dx,RXTX0H
mov             al,0a8h
out             dx,al
mov             dx,RXTX0L
mov             al,00h
out             dx,al
mov             dx,RXTX0H
mov             al,0b0h
out             dx,al
;udp hdr
```

```
                ;souce port
                mov     dx,RXTX0L
                mov     al,26h
                out     dx,al
                mov     dx,RXTX0H
                mov     al,17h
                out     dx,al
                ;destination port
                mov     dx,RXTX0L
                mov     al,26h
                out     dx,al
                mov     dx,RXTX0H
                mov     al,17h
                out     dx,al
                ;length
                mov     dx,RXTX0L
                mov     al,00h
                out     dx,al
                mov     dx,RXTX0H
                mov     al,52h
                out     dx,al
                ;chksum
                mov     dx,RXTX0L
                mov     al,chk_sum
                out     dx,al
                mov     dx,RXTX0H
                mov     al,chk_sum+1
                out     dx,al
                ;data
                mov     dx,RXTX0L
                mov     al,data
                out     dx,al
                mov     dx,RXTX0H
                mov     al,data+1
                out     dx,al
                mov     cl,0
pad:
        ;       ;padding
                mov     dx,RXTX0L
                mov     al,0aah
                out     dx,al
                mov     dx,RXTX0H
                mov     al,0bbh
                out     dx,al
                inc     cl
                cmp     cl,23h
                jbe     pad
                mov     dx,RXTX0L
                mov     al,0aeh
```

```
                out             dx,al
                mov             dx,RXTX0H
                mov             al,058h
                out             dx,al
                mov             dx,RXTX0L
                mov             al,0cdh
                out             dx,al
                mov             dx,RXTX0H
                mov             al,073h
                out             dx,al
                mov             flag,1
                ret
send    endp
```

Data Section

```
; Messages
mesg_mac        db "The MAC Address of this System is: ",04
msg                     db "Frame captured.",04
end_msg                 db "Press any key to send another packet... ",04
recv_msg        db "I've received: ",04

; eth0 Data
mac                     db 00,00,00,00,00,00
data            db 15h,51h
chk_sum         db 00, 00

;RTC
tick            dw      0
time            db  0

;flags
flag            db      0

; Temporary Storage Area
tmp             db  00h
wtmp            dw  0000h
p_cnt           db      00h             ;ping counter
                end
```

4.2 PC Client

I've stripped out from the header files the functions I'm not using in my actual client application.The functions from both header files are from a book on computer security programming, however the ones from the wrapper.h file were slightly edited by me.

wrapper.h

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

// This displays an error message before exiting
void fatal(char *message)
{
	char error[100];
	strcpy(error, "[ERROR] Fatal Error ");
	strncat(error, message, 80);
	perror(error);
	exit(1);
}

// checking errors for malloc, usefull wrapper function
void *Malloc(unsigned int size)
{
	void *ptr;
	ptr = malloc(size);
	if(ptr == NULL)
		fatal("in Malloc() while allocating memory");
	return ptr;
}
```

networking,h

```
#include <sys/socket.h>
#include <netinet.h>
#include <arpa/inet.h>
#include <netdb.h>

// This is used to send all bytes pointed by ptr, returns 1 on success, 0 on failure
int Send(int sockfd, unsigned char *buffer)
{
	int sent, left;
	left = strlen(buffer);
	while(left > 0){
		sent = send(sockfd, buffer, left, 0);
		if(sent == -1)
			return 0;4
```

```
                left -= sent;
                buffer += sent;
        }
        return 1;
}

int recv_l(int sockfd, unsigned char *dest_buffer)
{
#define EOL "\r\n"
#define EOL_SIZE 2
        unsigned char *ptr;
        int eol_matched = 0;
        ptr = dest_buffer;
        while(recv(sockfd, ptr, 1, 0) == 1) {
                if(*ptr == EOL[eol_matched]) {
                        eol_matched++;
                        if(eol_matched == EOL_SIZE) {
                                *(ptr+1-EOL_SIZE) = '\0';
                                return strlen(dest_buffer);
                        }
                } else {
                        eol_matched = 0;
                }
                ptr++;
        }
        return 0;
}
```

brew.c

```
#include <pcap.h>
#include "wrapper.h"
#include <arpa/inet.h>
#include <time.h>
#define WAIT_T 10*CLOCKS_PER_SEC
```

This function is just a simple wrapper function for calling showing the error from fatal_in then exits with an error as no more actions can be taken.

```
void pcap_fatal(const char *fatal_in, const char *errbuf) {
   printf("Fatal Error in %s: %s\n", fatal_in, errbuf);
   exit(1);
}
```

This function is to call the system ping command and wait for the ACK from the minimal system. However it seems when the way it is currently written, the program will go in an infinite loop when of essentially not picking up anything. Once I remove the else statement and the return 1 from the code, it seems to work but I don't get to see if I actually acquired anything. Some debugging is still in order...

```
int ping(void) {
   clock_t current, saved;
   struct pcap_pkthdr header;
   const u_char *packet;
   char errbuf[PCAP_ERRBUF_SIZE];
   char *device;
   pcap_t *pcap_handle;
   int i;

   device = pcap_lookupdev(errbuf);
   if(device == NULL)
         pcap_fatal("pcap_lookupdev", errbuf);

   pcap_handle = pcap_open_live(device, 4096, 1, 0, errbuf);
   if(pcap_handle == NULL)
     pcap_fatal("pcap_open_live", errbuf);

   system("ping -c 1 -W 1 192.168.0.225");
   saved = clock() + WAIT_T;
   while(clock() < saved){
   //for(;;){
     packet = pcap_next(pcap_handle, &header);
     if((*(packet+35) == 0x17) && (*(packet+34) == 0x26)){                  // Little-Endian type thingy
          if((*(packet+37) == 0x17) && (*(packet+36) == 0x26)){
           // if(*(packet+42) == 0xA0 || *(packet+43) == 0xA0){
              printf("ACK\n");
              printf("%x%x\n", *(packet+42), *(packet+43));
              pcap_close(pcap_handle);
              return 0;
            //}
          }
     }/*
     else{
          saved = (time(NULL)) - current;
          if(saved >= 20)
            printf("NACK\n");
            pcap_close(pcap_handle);
            return 1;
     }*/

   }
   pcap_close(pcap_handle);
   return 1;
```

```
}

int main(int argc, char *argv[]) {

  int n, x;

  printf("How many cups? >> ");
  scanf("%i", &x);

  if(x == 2){
    for(n=0;n<5;n++){
      if(ping())
           n--;
    }
  }
  else{
    printf("This feature is not available yet!\n");
  }
  exit(0);
}
```

5. Conclusion

A lot of progress was done however not in the expected route. Once the debugging on the client side of the protocol is done and the switch controller circuit is built and wired up the coffee machine will be functional for basic tasks. What will be missing is the tidying up the protocol and seeing if the receiving and transmission can't be fixed. Also the water meter circuitry will need to be added and possibly a GUI for the application as for now it's all terminal based.