003207

WIZnet iMCU Design Contest 2010
Project Number: **003207**
WIZnet Components: **W7100** on Evaluation Board for W7100 Mfr Part#: IMCU7100EVB

CDP Finder

Abstract

CDP (Cisco Discovery Protocol) is a Cisco protocol that runs between direct connected network entities (routers, switches, remote access devices, IP telephones etc.). The purpose of the protocol is to supply network devices with information about its direct connected neighbors. The Protocol sends detailed data about the including the IP address, hostname, domain, operating system version, model number, Ethernet port and other information about the switch. The CDP Finder will connect to a Cisco switch, router or other device and listen for the Cisco Discovery Protocol. The CDP Finder can be used by technicians to find unlabeled and mislabeled connections instantly.

Once connected to a live network CDP Finder opens a single macraw socket 0 and begins a loop that listens for packets. Each packet is examined for several conditions which are true for a CDP packet.

- The DSAP (Destination Service Access Point) the value must be 0xAA (SNAP) *(SubNetwork Access Protocol)*
- Then SSAP (Source Service Access Point) again the value must be 0xAA (SNAP)
- CDP Finder checks to see if the packet is encapsulated in SNAP
- CDP Finder reads the SNAP type which must be CDP 0x20 & 0x00

Once a CDP packet is confirmed CDP Finder parses the variable length data and loads each CDP Field into a variable. This design then displays the information necessary to find the device, to be precise the IP address, name and Ethernet port. The display shows the name and IP address alternately on the top line of the display and the port on the bottom line. Switch ports set to trunked mode send 4 additional bytes to negotiate 802.1Q trunking. The CDP Finder repeats the same process with a 4 byte offset. A more polished version would reuse the code; the submitted code easier to read and learn from.

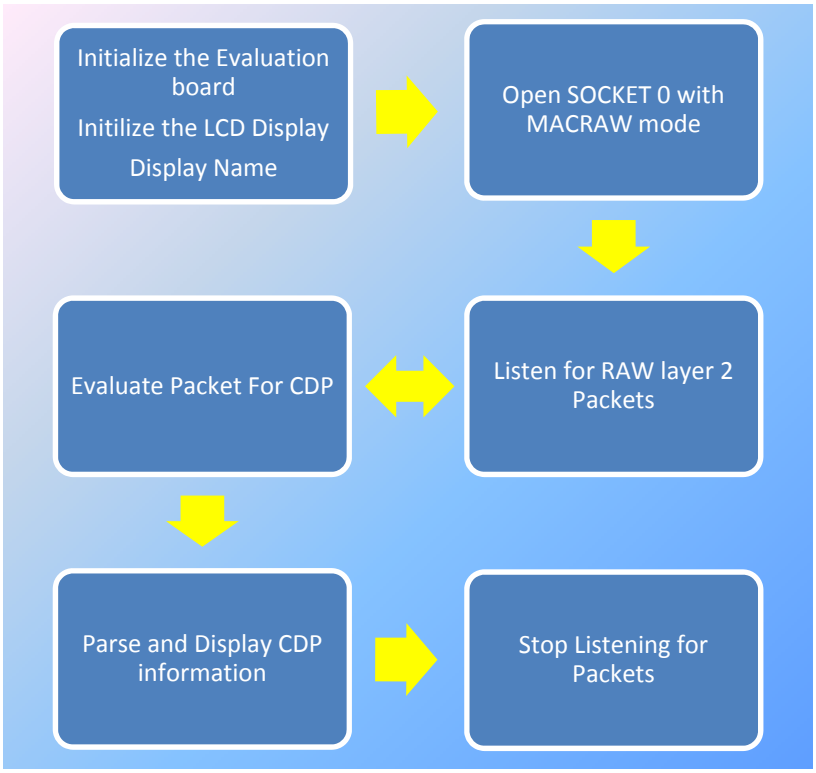
The hardware is a (functionally) unmodified W7100 Evaluation Board. There were no additional interface to the project, although a single button was considered to change the display it was

unnecessary and a timed loop serves the function of changing the display. The power switch was moved to the case front and the board was trimmed down to fit in project case. The CDP Finder has 2 small pieces of clear polycarbonate resin thermoplastic added to act as a 'light pipe' and bring the light from '*receive'* and '*link'* LED's to the front of the project case. The LCD display was relocated and arranged to fit the project case. For the sake of speed the CDP Finder's port is set to 10Mbps Half duplex that is all three switches at SW4 are set to 'on'. By eliminating the hardware negotiation the CDP Finder can usually resolve a port within 10 seconds .

Schematic

This project does not contain a schematic since the CDP Finder uses an evaluation board Evaluation Board for W7100 Mfr Part#: IMCU7100EVB with no circuitry added by the Participant.

Block Diagram



Code

The Code for this project was written in c and compiled by the Keil compiler
IDE-Version: µVision V4.02

```
void CiscoDP(SOCKET s, uint16 aPort,  uint8 count)
{
                    uint16 xdata len = 0;
    uint16 xdata   i= 0;
    uint16 xdata rlen =0;
    uint16 xdata cnt =0;

    for ( i=0 ; i<count +1; i++ ){

        switch(getSn_SR(s))
            {
                case SOCK_CLOSED:
                    close(s);                           // close the SOCKET
                    socket(s,Sn_MR_MACRAW,aPort,0);         // open the SOCKET with MACRAW mode
                break;

                case SOCK_MACRAW:
                // Raw packet captured
                CiscoDP_received     = 0;
                while(1){
                        if (CDPFound == 1){
                                        // display the ip address
                                        evb_set_lcd_text(0,"            ");
                                        evb_set_lcd_text(0,ip1);
                                        wait_1ms(3000);
                                        // display the switch's hostname
                                        evb_set_lcd_text(0,"            ");
                                        evb_set_lcd_text(0,name);
                                        wait_1ms(3000);
                                        }

                        if ( (rlen = getSn_RX_RSR(s) ) > 0){
                            CiscoDP_parse(s, rlen);
                            if (CiscoDP_received) break ;
                            } // end of if from 70
                    } //end of while from 58
            } // end of switch from line 47
    } //end of for loop from line 45
}


void CiscoDP_parse(SOCKET s, uint16 rlen)
{
    uint16 xdata mac_destport;
    uint8 xdata * data_buf = 0x007000;
    uint16 xdata len =0;
    uint8 xdata mac_destip[4];
    uint16 xdata ch =0;

    if (cdp ==0){
        len = recvfrom(s,(uint8 *)data_buf,rlen,mac_destip,&mac_destport);

        //printf("%u bytes long.",len);
        wait_1ms(50);
        LCD_CLEAR;
        evb_set_lcd_text(0,(uint8 *) "   CDP Finder   ");
        evb_set_lcd_text(1,"  Searching...  ");

                CiscoDP_received = 1;

                //              this section detects an access port

                // Is DSAP address 14 0xAA and SSAP address 15 0xAA
                // Is this a SNAP packet ? 17,18,19 = 0x00 0x00 0x0C
                // Is SNAP type CDP? 20,21 0x20 0x00


                if    (data_buf[14]== 0xAA && data_buf[15] == 0xAA && data_buf[17] == 0x00 && data_buf[18] == 0x00 && data_buf[19] == 0x0C && data_buf[20]== 0x20 && data_buf[21] == 0x00 ){

                evb_set_lcd_text(0,"   CDP Found    ");
                CDPFound =1;

                    //copy everthing into a string
                    for (ctoloop = 0 ;  ctoloop < len; ctoloop++){
                      str[ctoloop] = data_buf[ctoloop];
                    }

                    // First field starts at 26
                    Start[0] = 26;
                    Length [0] = 0;

                    // Set the starting point
                    Start[cdpfield] = 26;
                    //printf("Start[cdpfield] = %u\r\n" ,Start[cdpfield]);


                        //this number sets the number of cdp fields to examine
                        while ((cdpfield < 9) ) {
                        if (Start[cdpfield] >= len)  break;

                            //get first CDP field type
                            Type[cdpfield] = ((data_buf[Start[cdpfield]] *255) + (data_buf[Start[cdpfield] +1] ));

                            //get first data length
```

```
                    Length[cdpfield] = (data_buf[Start[cdpfield] + 2] *255) + (data_buf[Start[cdpfield] + 3]);
                    //printf("CDP field Type[cdpfield] %u\r\n" , Type[cdpfield]);
                    //printf("Field start from begining of packet Start[cdpfield] = %u\r\n" , Start[cdpfield]);
                    //printf("Field Length[cdpfield] %u\r\n" , Length[cdpfield]);

                    if ( Type[cdpfield] == 1){
                    // ADD 4 bytes to the start of the field to offset for the type and length
                        for ( ch = Start[cdpfield] + 4; ch < (Start[cdpfield] + Length[cdpfield]) ; ch++){
                            tmpstring[0] = data_buf[ch];
                            strncat(name,  tmpstring, 1);
                            printf("%c", data_buf[ch] );
                        }
                    }
                    if ( Type[cdpfield] == 3){
                    // ADD 4 bytes to the start of the field to offset for the type and length
                        for ( ch = Start[cdpfield] + 4 ; ch < (Start[cdpfield] + Length[cdpfield]) ; ch++){
                            tmpstring[0] = data_buf[ch];
                            strncat(port,  tmpstring, 1);
                            printf("%c", data_buf[ch] );
                        }

                        // starting at the last 't' and continue until the
                        // last char in string "port"
                        for (ch = strrpos (port,'t') + 1 ; ch < (strlen(port)) ; ch++){
                            tmpstring[0] = port[ch];
                            strncat(shortport,  tmpstring, 1);
                            printf("*%c", data_buf[ch] );
                        }
                    }

                    // 14 15 16 17 ip address
                    if ( Type[cdpfield] == 2){
                    //sprintf("ip %u " , (unsigned int)str[(unsigned int)startoffield]  );
                    startoffield = Start[cdpfield];
                    sprintf(ip1,"%u.%u.%u.%u \r\n" ,str[startoffield+13] ,str[startoffield+14] ,str[startoffield+15] ,str[startoffield+16]  );
                    }

                cdpfield++;
                Start[cdpfield] = Start[(cdpfield-1)] +    Length[(cdpfield-1)];
                } // end the for cdpfield loop
            cdp =1;
        }

    //change the display
        if (CDPFound == 1){
            // display the shortened port name on the bottome line of the display
            evb_set_lcd_text(1,"            ");
            evb_set_lcd_text(1,shortport);
        }
    }
}
```