

PROJECT REPORT

Gaurav Doshi
CSCI 339

PROJECT ABSTRACT:

To Add a wireless network interface to the Z8 board. That is I am basically implementing the 802.11b protocol on the Z8 microcontroller, so that it is connected to the Access point available in the area and it can talk to other stations in the area.

IEEE 802.11, the Wi-Fi standard, denotes a set of Wireless LAN/WLAN standards developed by working group 11 of the IEEE LAN/MAN Standards Committee IEEE 802. The term 802.11x is also used to denote this set of standards. 802.11b has a maximum raw data rate of 11 Mbit/s and uses the same CSMA/CA media access method defined in the original standard. Due to the CSMA/CA protocol overhead, in practice the maximum 802.11b throughput that an application can achieve is about 5.9 Mbit/s over TCP and 7.1 Mbit/s over UDP.

802.11b products appeared on the market very quickly, since 802.11b is a direct extension of the DSSS(Direct-sequence spread spectrum) modulation technique defined in the original standard. Technically, the 802.11b standard uses Complementary code keying (CCK) as its modulation technique, which is a variation on CDMA. Hence, chipsets and products were easily upgraded to support the 802.11b enhancements. The dramatic increase in throughput of 802.11b (compared to the original standard) along with substantial price reductions led to the rapid acceptance of 802.11b as the definitive wireless LAN technology.

It is based on a cellular architecture which is subdivided into cells , where each called the Basic Service Set is controlled by an access point .

Strategy:

Components Used :

The microcontroller used for this project is the zilog **Z8F6403** microcontroller . According to the references “Implementing 802.11 with microcontrollers “ The project can be implemented using a wide variety of 8 bit microcontrollers including the Z8 . Also we would have been well acquainted with this microcontroller.

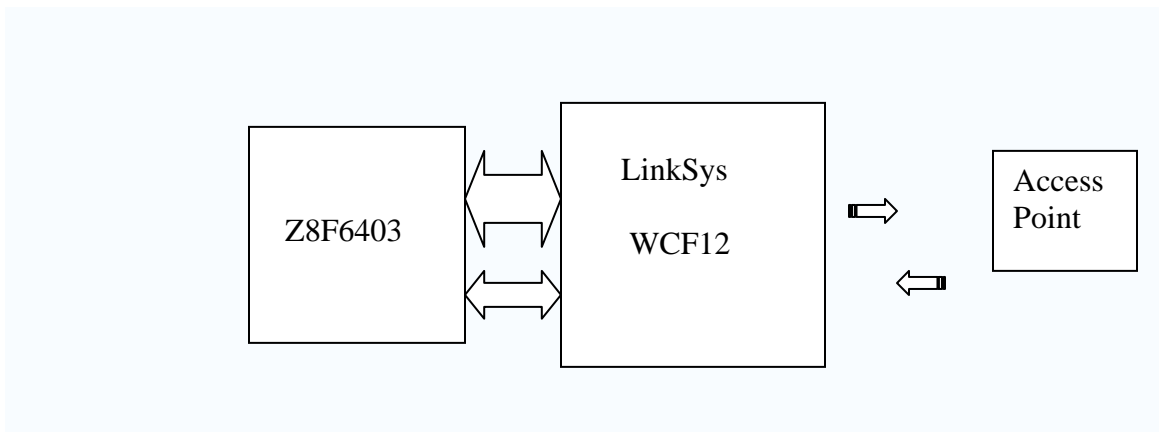
The entire 802.11b design is dependent on a **NIC** ,(network interface card) which can be obtained commercially . I have selected the **Linksys WCF12** NIC which is a compact flash network interface card which operates in the unlicensed 2.4 Mhz ISM band It is the most widely available compactflash card for PDA's

For connecting the Z8 microcontroller to the **Compact Flash card connector** . The compact Flash card connector is a 16 bit device as implied by the Compact Flash card data I/O Pins D0- D15 .however it allows for the use of 8 bit data .

OVERALL DESIGN

Hence the overall design consists of 2 Z8 boards , each interfaced to the NIC through a Compact Flash card connector .

In order to show that the network is working I intend to pass a simple string of data . I will need an access point to configure the microcontrollers



SOFTWARE CODES

The Software Code that I have to write is :

- The WCF12 initialization code.
- That is the code to interface the NIC to the microcontroller to tour the card information structure
- The Zilog 802.11 driver source code This is simply a combination of various Compact Flash card I/O routines coupled with PRISM commands
- Communicating with the Access Point in the Area and receiving an acknowledgement frame from the 505N Access Point

Implementation Plan

1. Initialize the Zilog features required for communication with the NIC
2. Initialize and configure the NIC for BSS operation, desired SSID, max frame data length, transmission rates and retrieve and format the NIC's MAC address
3. Format the IP address
4. Allocate the NIC transmit buffers
5. Enable the CompactFlash NIC's MAC
6. Probe requests will be transmitted by the 802.11b NIC
7. The AP will acknowledge the 802.11b NIC's probe requests
8. The 802.11 b NIC will request authentication
9. The AP will authenticate the 802.11b NIC
10. the 802.11 b NIC will request association
11. The AP will grant the 802.11b NIC association request
12. The Zilog chip will join the wireless LAN

Resources :

The Z8 micro controller is provided by GWU

The Linksys WEC12 CompactFlash Card

The CompactFlash Card Connector with the Breakout Board

A lot of wires

Status :

Out of the implementation steps I had completed were

- Initialized the Zilog features required for communication with the NIC .
- Initialized and configured the NIC for BSS operation, desired SSID, max frame data length, transmission rates and retrieve and format the NIC's MAC address .
- Formatted the IP address
- Allocate the NIC transmit buffers .Enable the CompactFlash NIC's MAC.
- However Due to a hardware failure I could not continue with the project.

I need to make changes in the hardware to continue implementation of the project.

I was having many problems with the wiring too . Since there were many which had to be connected in a small space . Pre insulated wires would have been better.

Another problem I had was lack of information about the hardware ,since all the information I got was from the Implementing 802.11b with microcontrollers book. There is not much information available about the pin out of the CompactFlash NIC's

PROJECT SPECIFICATIONS:

HARDWARE :

**Microcontroller Used : Zilog Z8F6403 on the development Kit:
Features and Specifications :**

- eZ8 CPU, 20 MHz operation
- 12-channel, 10-bit analog-to-digital converter (ADC)
- 3-channel DMA
- Up to 64KB Flash memory with in-circuit programming capability
- Up to 4KB register RAM
- Serial communication protocols
 - Serial Peripheral Interface
 - I2C
- Two full-duplex 9-bit UARTs
- 24 interrupts with programmable priority
- Three or four 16-bit timers with capture, compare, and PWM capability
- Single-pin On-Chip Debugger
- Two Infrared Data Association (IrDA)-compliant infrared encoder/decoders integrated with the UARTs
- Watch-Dog Timer (WDT) with internal RC oscillator
- Up to 60 I/O pins
- Voltage Brown-out Protection (VBO)

CompactFlash NIC used : Linksys WCF12

Features:

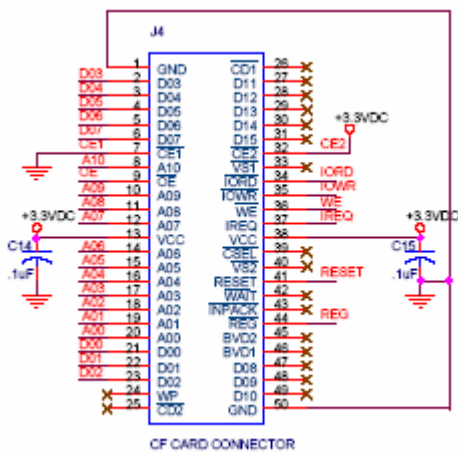
- Up to 11Mbps High-Speed Data Transfer Rate with Automatic Fallback
- Compliant with 802.11b, DSSS, 2.4GHz Standard
- Supports up to 128-bit WEP Encryption Security
- Fits in a CompactFlash (CF) Type I or Type II Slot
- Wi-Fi Compatible—Attach to a 2.4GHz, 802.11b Wireless Network
- Built-in Power Management Saves Battery Power
- Works with Most Pocket PC PDAs
- **Channels:** 11 Channels (US, Canada)

- 13 Channels (Europe)
- 14 Channels (Japan)
- **Transmit :15 dBm**
- **Receive Sensitivity: -84 dBm (typical)**
- **Modulation CCK, BPSK, QPSK**
- **Network Protocols IPX/SPX, TCP/IP, NetBEUI**
- **LED Link**



Compact Flash Card Connector with Breakout Board :

PinDiagram:



Picture:



Access Point :

I had intended to communicate with the WCG 200 access point which is actually meant for 802.11G protocol but it is backward compatible to support the 802.11b

Features of the WCG200 are :

- DOCSIS 1.1 and 2.0 certified cable modem
- Integrated Wireless-G Access Point (Also compatible with all Wireless-B devices)
- SPI firewall with Denial of Service attack prevention
- Wireless Access Lists and WPA to enhance wireless security
- USB port for systems without a wired or wireless network card
- Email and Web-based Logging of Security Events
- Supports VPN Passthrough for IPSec, PPTP, and L2TP Protocols
- MAC Address Filtering, Port Forwarding, and DMZ support



SOFTWARE MODULES

The following software modules needed to be written :

1. Initialize the UART :

For this I have just used the inbuilt API provided by Zilog to initialize the UART thus all the printf statements are directly routed to the hyper-term through the serial cable

2. Initializing the CF card :

For this First we have to read the Card Information Structure to get the Device Tuple of value 0x01 first . This memory structure consists of Device Tuples , Link Tuples and Tuple Data which contains all the information of the NIC which is passed on to the micro controller .

The Hex Dump i.e this information provided from the CIS memory for the Linksys is as follows :

```
CIS v 0.1
01
17
1D
15 LinksysWireless CompactFlash Card
20 MANFID 8A 02 73 06
21 FUNCID 06 00
22 FUNCE 01 07
22 FUNCE 02 40 42 0F 00
22 FUNCE 02 80 84 1E 00
22 FUNCE 02 60 EC 53 00
22 FUNCE 02 C0 D8 A7 00
22 FUNCE 03 07
22 FUNCE 04 06 00 0C 41 15 B4 64
22 FUNCE 05 01
1A COR 03 E0
1B C1 01 19 77 B5 1E 35 B5 3C 36 36 05 46 FF FF FF
FF
```

- The Core Base address : 0x03E0 is the information we require from scanning the CIS . This is used in the driver .
- Configure the NIC to operate in the I/O mode by enabling the IORD and IOWR pins after this is done :

The Drivers :

The driver performs the following steps :

First we send the Initialize Command :

To do this the following steps are required:

- First we retrieve the 16 bit value of cmd_data from the command register which is located at location 0x0000 . Since we perform 8 bit operations only , 2 operations are required which then merges the values into a 16 bit value
- Once this indicates that it is not busy we can start writing the command sequence .
- Then we write the command parameter to the parameter register
- This is followed by writing the command sequence to the Command register
- The Compactflash card will go into a busy status while this command is executed .
- Completion of this event is signaled by setting the bit in the Event Status register.
- Once the command complete bit materializes a command status word is posted in the status register . .
- We then set the EVStat_cmd_Bit_Mask in the event acknowledge register to acknowledge the pending command complete event

Now we have to communicate with the AP , hence for that we have to read and write data in the PRISM chipset memory , We do this with the help of a buffer access path . The PRISM chipset firmware is constantly and dynamically allocating groups of 128 byte memory blocks into the transmit and receive buffer. The dynamically allocated buffer chains are built upon linked lists of pointers to non continuous 128 byte memory locations .Now the FID is a unique physical reference to an associated dynamically allocated 128 byte per block buffer chain .A Buffer Access Path is a means of accessing the data within the dynamically allocated buffer chain that a FID identifies . The RID is used to configure the operation of an 802.11 CompactFlash NIC

Communication with the Access Point :

- For this we first tell it that it will be a part of the BSS by placing the appropriate PORT Type RID values (0xFC00) within the “ fidrid” buffer which is a 256 byte array in which we place the RID_cfg_portType_length , the porttype and the infrastructure In the first 3 elements

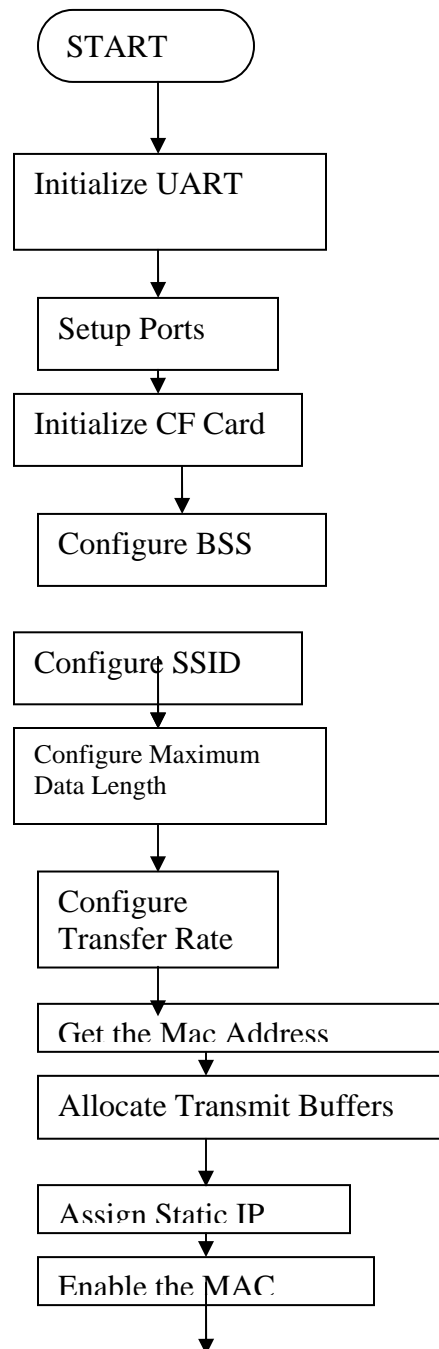
- Now first we find out which of the BAP's is free to use .
- Then the port type RID value of 0xfc00 is placed in the BAP's select register 0x0000 is placed in the offset register , thus we set up the pipeline to the port type RID , Then the Data placed in the Fidrid buffer is written the BAP register .
- Now Datax _register at 0x0036 or 0x0038 is our data portal and we have to write the data there
- An access command is then issued against the Port_type RID with the access write bit set.
- To read the data contained within the PORT Type RID the RID_read function may be used
- In a similar manner as telling the NIC that it will be a part of the BSS we pass the SSID . string . This string is stored in position 2 of the Fidrid buffer , if we pass the 0 length string the compactflash NIC will attempt to join any available BSS
- After this the maximum length of the 802.11b frame body data is passed .
- Then we similarly pass the NIC Rate data which determine the rate at which transmission is to take place
- The compactflash NIC mac Arddr is held in the fidrid_buffer is first converted intoCharacter format and then stored into the macdeerc array we also convert it onto integer format and store thm in array macaddri
- Finally we choose a static ip adderss and store it into the ipaddri variables . to complete the configuration

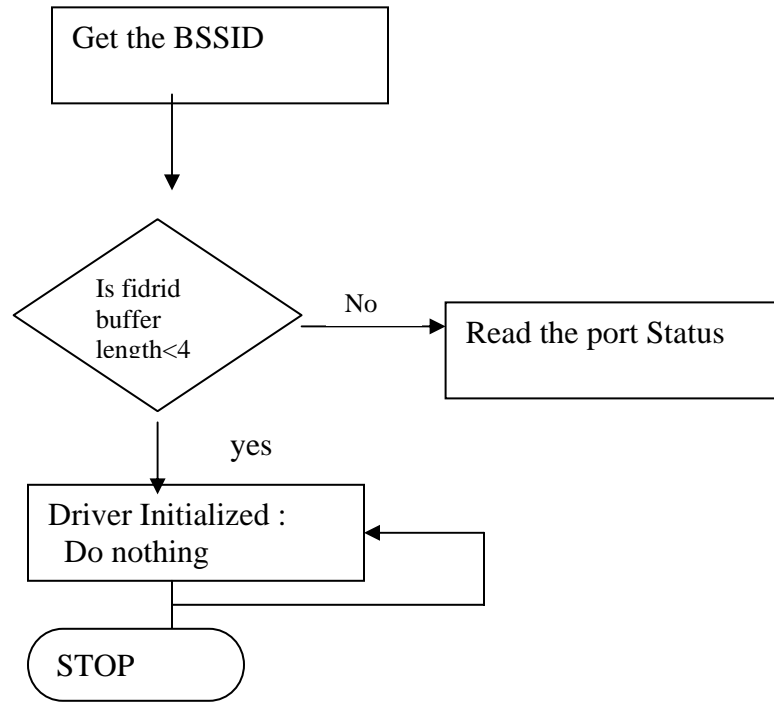
To put the Airdrop on the Lan

- First we allocate 802.11b NIC transmit buffers using the allocate_xmit_buffers function by using the Allocate _cmd which has value of 0x0A
- Ten we enable the compact flash's NIC's MAC This enables the NIC to transmit probe requests . with the help of EnableMAC_cmd
- Once we receive an acknowledgement for probe request we then request authorization .
- After authorization is received we request association and after association is received finally the microcontroller has joined the wireless LAN .

Implementation and Construction :

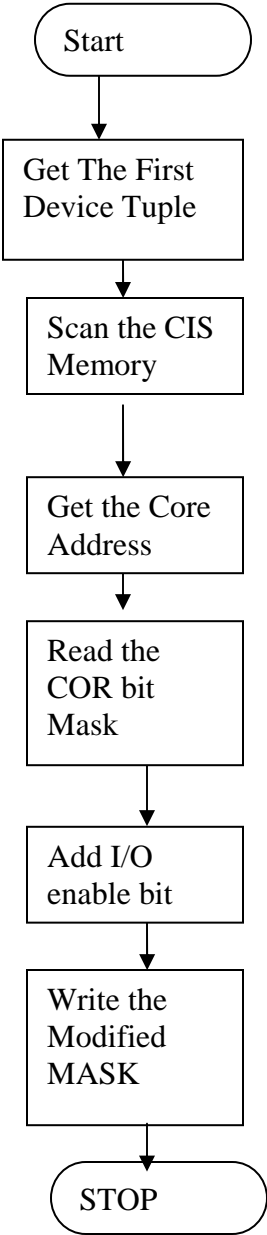
FLOW CHART
MAIN:



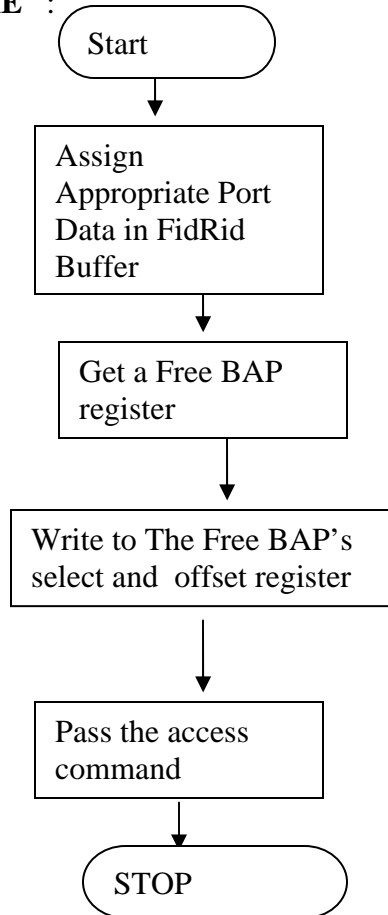


:

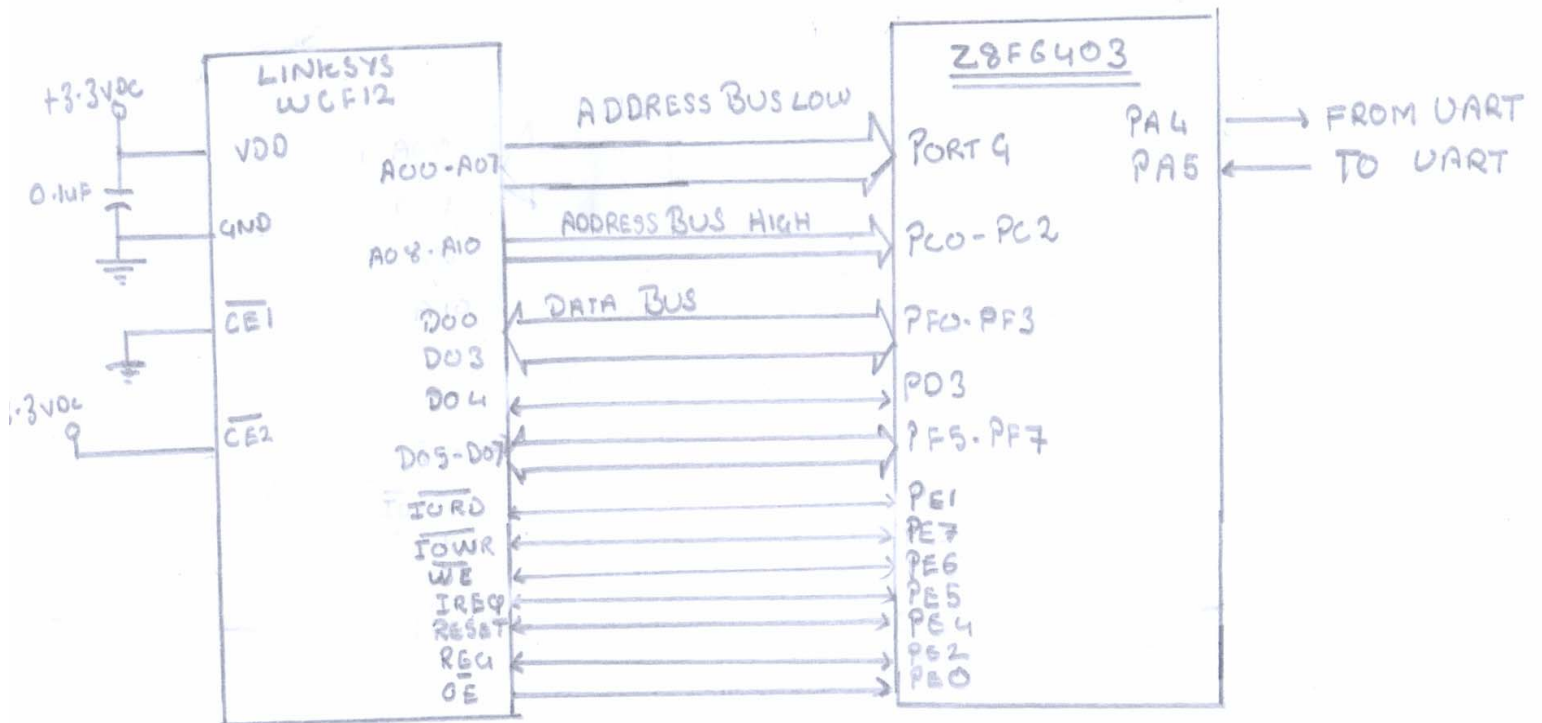
Initialize CompactFlash Card



CONFIGURE :



BLOCK DIAGRAM



Mile Stone Chart

- Initialize the Zilog features required for communication with the NIC
- Initialize and configure the NIC for BSS operation, desired SSID, max frame data length, transmission rates and retrieve and format the NIC's MAC address
- Format the IP address
- Allocate the NIC transmit buffers
- Enable the CompactFlash NIC's MAC
- Probe requests will be transmitted by the 802.11b NIC

- The AP will acknowledge the 802.11b NIC's probe requests
- The 802.11 b NIC will request authentication
- The AP will authenticate the 802.11b NIC
- the 802.11 b NIC will request association
- The AP will grant the 802.11b NIC association request
- The Zilog chip will join the wireless LAN

TESTING :

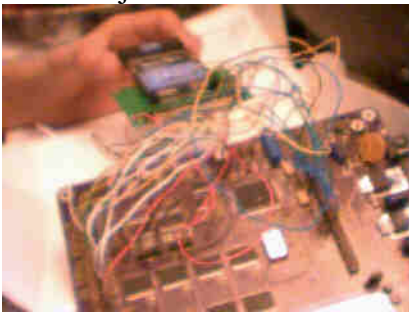
All the hardware testing was done using a CRO and multimeter .

For hardware testing First I had to check the connections using the multimeter to check for resistance .

Then I had to check the voltages at the control lines at various points in the program so I used breakpoints in the program and then checked the voltages at the CF card end to verify whether correct voltages were reaching the CF card .

The software was tested using the debugger

Final Project Picture :



Retrospective :

I need to make a lot of changes in the hardware . If I was to start over I would have started testing the hardware with the CRO from the beginning .

Also I would have searched for data bus connectors instead of connecting the wires individually ..

A lot of time has to be left to debug the hardware .

