



## Center for Computer Information Systems Engineering

Session 2000-2004

# Voice over Internet Protocol (VoIP) Media Gateway

### Group Members

1. Aamir Mehmood (Team Leader)
2. Muhammad Tufail
3. Yasir Feroz

Project Advisor  
Engr. Jawad Hafeez

NWFP University of Engineering and Technology Peshawar

CENTER for COMPUTER INFORMATION SYSTEMS ENGINEERING

NWFP University of Engineering and Technology (U.E.T) Peshawar

February 2004

This report is submitted to Center for Computer Information Systems Engineering as partial fulfillment of Bachelors of Science degree in Computer Information Systems Engineering.

Approved By

1. Director CCISE

---

2. Project Advisor

---

3. Examiner

---

Submitted By

1. Aamir Mehmood
2. Muhammad Tufail
3. Yasir Feroz

## **DEDICATION**

To our parents and our teachers whose love and affection has been  
inspirational throughout our lives

## ACKNOWLEDGEMENT

For the completion of our project we are thankful to plenty of people for their very sincere cooperation that they extended to us at various stages. First and foremost problem that each group faces is the selection of a project and the selection of supervisor which was solved by **Mr. Umer Anwar Sheikh**. We would like to acknowledge our project supervisor **Engr. Jawad Hafeez**, for his invaluable contribution and encouragement. During the execution of our project we had a real chance to learn from the experience and immense knowledge of our advisor. We also wish to thank Engr. Tariq Saeed for providing us all the required equipments through all possible means.

We again would like to thank our teachers Mr. Jawad Hafeez and Mr. Umer Sheikh for providing us with the valuable resource of their personal private laboratory without which this project would never had been realized. It is the devoted contribution of all the above mentioned persons, our devoted hard work and the prayers of our parents that this project was able to win the **FIRST PRIZE** in “**National Exhibition on Embedded Systems and Microcontrollers**” held at the COMSATS Institute of Information Technology, Abbottabad on the 12<sup>th</sup> and 13<sup>th</sup> of March, 2004.

Aamir Mehmood

Muhammad Tufail

Yasir Feroz

## TABLE OF CONTENTS

<b>SUMMARY .....</b>	<b>6</b>
<b>INTRODUCTION.....</b>	<b>7</b>
<b>PART ONE “HARDWARE DESIGN” .....</b>	<b>9</b>
<b>CHAPTER 1 Power Supply .....</b>	<b>10</b>
1.1 Background .....	10
1.2 Power Supply Elements .....	10
1.3 Functional Description .....	10
<b>CHAPTER 2 Subscriber Line Interface Circuit (SLIC).....</b>	<b>14</b>
2.1 Background .....	14
2.2 Pin Configuration .....	15
2.2 Functional Data .....	16
<b>CHAPTER 3 DTMF Receiver .....</b>	<b>19</b>
3.1 Background .....	19
3.2 Pin Configuration .....	19
3.3 Functional Data .....	20
<b>CHAPTER 4 CODEC (ADC / DAC) .....</b>	<b>22</b>
4.1 Background .....	22
4.2 Pin Configuration .....	22
4.2.1 Analog to Digital Converter .....	22
4.2.2 Digital to Analog Converter .....	23
4.3 Functional Data .....	24
<b>CHAPTER 5 Main <math>\mu</math>Controller, Memory Unit and NIC.....</b>	<b>28</b>
5.1 Background .....	28
5.2 Microcontroller Description .....	28
5.3 Memory Chip Description.....	30
5.4 Network Interface Card Description .....	32
5.4.1 Physical Design.....	33
5.4.2 Signal Descriptions .....	35
<b>CHAPTER 6 Ringer <math>\mu</math>Controller &amp; Amplifier Circuit.....</b>	<b>38</b>
6.1 Pin Configuration .....	38
6.2 Functional Description .....	38

<b>PART TWO “NETWORK INTERFACE CARD INTERNAL DESIGN” .....</b>	<b>42</b>
<b>CHAPTER 7 Network Interface Card Internal Design .....</b>	<b>43</b>
7.1 General Description .....	43
7.2 Features.....	43
7.3 Transmit/Receive Packet.....	44
7.4 Packet Reception.....	46
7.4.1 Initialization of the Buffer Ring.....	48
7.4.2 Beginning of Reception.....	49
7.4.3 End of Packet Operations .....	49
7.4.4 Successful Reception.....	49
7.4.5 Removing Packets from the Ring .....	50
7.4.6 Storage Format for Received Packets .....	52
7.5 Packet Transmission.....	53
7.5.1 Transmit Packet Assembly.....	53
7.5.2 Transmission .....	53
7.5.3 Conditions Required To Begin Transmission .....	54
7.5.4 Collision Recovery .....	54
7.5.5 Transmit Packet Assembly Format.....	54
7.6 Remote DMA.....	55
7.6.1 Remote Write .....	55
7.6.2 Remote Read.....	56
7.6.3 Send Packet Command.....	56
7.7 Internal Registers.....	57
7.8.1 Command Register (CR) 00h (Read/Write).....	60
7.8.2 Interrupt Status Register (ISR) 07h (Read/Write) .....	62
7.8.3 Interrupt Mask Register (IMR) 0fh (Write) .....	63
7.8.4 Data Configuration Register (DCR) 0EH (Write) .....	64
7.8.5 Transmit Configuration Register (TCR) 0dh (Write).....	65
7.8.6 Transmit Status Register (TSR) 04h (Read) .....	66
7.8.7 Receive Configuration Register (RCR) 0ch (Write).....	67
7.8.8 Receive Status Register (RSR) 0CH (Read) .....	68
7.9 DMA REGISTERS .....	69
7.10 Transmit DMA Registers .....	70
7.10.1 Transmit Page Start Register (TPSR).....	70
7.10.2 Transmit Byte Count Register 0,1 (TBCR0, TBCR1) .....	70
7.11 Local DMA Receive Registers.....	71
7.11.1 Page Start Stop Registers (PSTART, PSTOP).....	71
7.11.2 Boundary (BNRY) Register .....	71
7.11.3 Current Page Register (CURR).....	71
7.11.4 Current Local DMA Register 0,1 (CLDA0,1) .....	72
7.12 Remote DMA Registers .....	72
7.12.1 Remote Start Address Registers (RSAR0,1) .....	72
7.12.2 Current Remote DMA Address (CRDA0, CRDAL) .....	73

7.13 Physical Address Registers (PAR0-PAR5) .....	73
7.14 Initialization Procedures .....	74
7.14.1 Initialization Sequence.....	74
<b>PART THREE “SOFTWARE DESCRIPTION” .....</b>	<b>76</b>
<b>CHAPTER 8 Main Microcontroller Software .....</b>	<b>77</b>
8.1 General Description .....	77
8.2 System Software Flow Chart .....	77
8.3 System Command Flow Diagram.....	81
<b>CHAPTER 9 Ringer Microcontroller Software.....</b>	<b>84</b>
9.1 General Description .....	84
9.2 Software Description .....	84
<b>APPENDIX A .....</b>	<b>86</b>
<b>APPENDIX B .....</b>	<b>119</b>
<b>REFERENCES.....</b>	<b>123</b>

## LIST OF FIGURES

Figure 0.0 Big Picture .....	8
Figure 1.1 Voltage Regulator.....	11
Figure 1.2 Power Supply Schematic .....	13
Figure 2.1 SLIC IC Pin Configuration.....	14
Figure 2.2 SLIC Schematic .....	18
Figure 3.1 DTMF IC Pin Configuration .....	20
Figure 3.2 DTMF Receiver Schematic .....	21
Figure 4.1 ADC / DAC Pin Configuration .....	23
Figure 4.2 CODEC Schematic .....	27
Figure 5.1 The VoIP System Big Picture.....	29
Figure 5.2 Memory Chip Pin Configuration.....	30
Figure 5.3 Memory Module Circuitry .....	32
Figure 5.4 ISA Card.....	33
Figure 5.5 ISA Connector .....	34
Figure 5.6 Main Microcontroller and NIC Schematic .....	37
Figure 6.1 Ringer $\mu$ Controller Circuit .....	40
Figure 6.2 Ring Amplifier Circuit .....	41
Figure 7.1 Standard Ethernet Packet.....	44
Figure 7.2 Dual Bus System .....	46
Figure 7.3 NIC Receiver Buffer.....	47
Figure 7.4 Receiver Buffer Ring at Initialization .....	48
Figure 7.5 Received Packet Enters Buffer Pages.....	49
Figure 7.6 Packets Accepted.....	50
Figure 7.7 First Received Packet Removed By Remote DMA .....	52
Figure 7.8 Storage Format .....	52
Figure 7.9 General Transmit Packet Format.....	53
Figure 7.10 Transmit Packet Assembly Format .....	55
Figure 7.11 Remote DMA Auto Initialization.....	57
Figure 7.12 Register Address Mapping .....	58
Figure 7.13 DMA Registers.....	69
Figure 8.1 System Software Flow Chart.....	78
Figure 8.2 System Command Flow Diagram .....	83



## LIST OF TABLES

Table 2.1 SLIC IC Pin Description.....	15
Table 5.1 Memory Chip Function Table.....	31
Table 6.1 Telephone Tone Standards Used .....	39
Table 7.1 Register Address Assignments .....	59
Table 7.2 Command Register Description.....	61
Table 7.3 Interrupt Status Register Description.....	62
Table 7.4 Interrupt Mask Register Description.....	63
Table 7.5 Data Configuration Register Description .....	64
Table 7.6 Transmit Configuration Register Description.....	65
Table 7.7 Transmit Status Register Description .....	66
Table 7.8 Receive Configuration Register Description .....	67
Table 7.9 Receive Status Register Description.....	68

## SUMMARY

There is a general consensus that in years to come more and more Internet devices will be embedded and not PC oriented. Just one such prediction is that by 2010, 70% of Internet-connected devices will not be computers. So if they are not computers, what will they be?

### **“Embedded Internet Devices”**

VoIP Media Gateway is a complex project of transferring packetized voice over an IP based network. This is achieved by interfacing the digital telephone exchange to an Intel 8051 microcontroller which is also connected to a Realtek RTL8019AS network controller. The result is a very low-cost LAN based telephone network. It is an embedded system that transfers the voice over the packet based network. The basic idea is to make such telephone exchanges that act as a liaison between the telephone sets and the packet based network (Internet or LAN). Hence by adopting such idea, the networks will be used to transfer the data as well as the voice on the same channel based on the time sharing of the medium.

Voice from the telephone sets is taken as analogue and then converted to the digital form. Some storage mechanism for voice is used so that we send the voice data in the form of chunks of packets. An end to end delay of the duration of storing of one packet (almost 25ms) is created in such a system.

## INTRODUCTION

VoIP Media Gateway is an embedded system which uses the existing LAN facilities to make a Private Branch Exchange (PBX). LANs transfer both the voice and data between the computers and those embedded devices.

To help grasp the concept of this complex project, the block diagram on the next page will be very useful. **Subscriber Line Interface Circuitry** (SLIC) is a small IC from the Mitel Semiconductor that gives all of the possible functionalities to the telephone set along with receiving and transmitting the analogue voice from the set. It also detects the hook status and numeric pad keys pressed by the user. On the other end of the SLIC is the Single Board Computer (SBC) that comprises of CODEC, DTMF Detector and two Microcontroller ICs from the Atmel. The analogue voice coming out of the SLIC is converted to digitized voice by passing through the **Analogue to Digital Converter** (ADC). The conversion rate of the ADC is 8000 samples per second. This digital voice is then fed to the **Main Microcontroller** that stores it temporarily in a RAM for a fixed length of duration. The other end of the SBC is interfaced to an ISA based **Network Interface Card**. This NIC provides a medium of interaction between the microcontroller and the packet based network. Packets are prepared in the microcontroller and are then written in the internal memory of the interface card. Similarly, the packets received from the NIC are read directly from its internal memory and the digital data is extracted, which is then stored in the RAM attached to the Microcontroller. This data is then placed on the **Digital to Analogue Converter** (DAC) byte by byte after every 125us. The DAC gives its output to the SLIC, which in turn plays it on the telephone set. The second microcontroller also called the **Ringer** is used to produce the different melodies related to the telephony. These may include dial tone, engage tone etc. All these tones are produced continuously and an analogue multiplexer is used to choose one of them at a time. This multiplexer is also controlled by the Main Microcontroller.

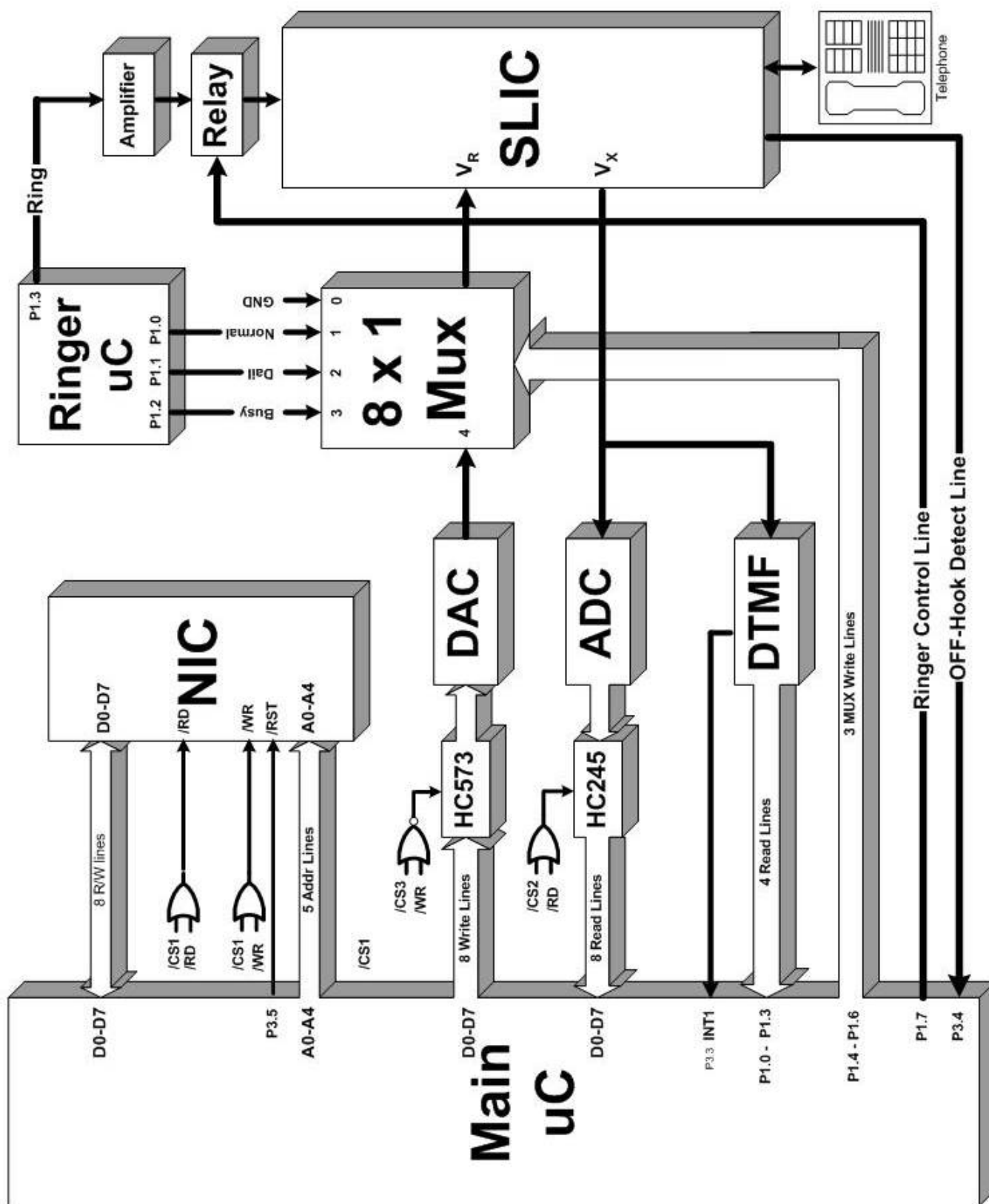


Figure 0.0 Big Picture

# **PART ONE**

## **HARDWARE DESIGN**

This project consists of nine different subsystems that work independent of each other. These subsystems are then integrated with each other to make the whole embedded system. Each of these subsystems is further explained in separate chapters in this section. Each subsystem takes some inputs and gives some output services which then serve as input to other subsystems or to the end user.

1. Power Supply
2. SLIC
3. DTMF Receiver
4. CODEC (ADC / DAC)
5. Main Microcontroller, Network Interface Card & Memory Unit
6. Ringer Microcontroller & Amplifier Circuitry

## CHAPTER 1

# Power Supply

### 1.1 Background

Power supply is the basic component of the system. It is the first one of all the subsystems to be made as it serves the purpose of stable voltage supply to all the other subsystems during the whole project. This project uses versatile range of voltages given as +12v, +5v, Gnd, -5v, -11v and -28v in the laboratories, there are no such supply elements giving such range of supply voltages, so it was to be made as the first phase of the project.

### 1.2 Power Supply Elements

The power supply consisted of series of different voltage regulators along with bridge rectifiers, capacitors, resistors and a transformer. These included regulators LM 337, LM 7805 and LM 7812, bridge rectifier D3SBA60, transformer with many output loops, electrolytic capacitors 470uF 50v, tantalum capacitors 1uF 50v, various resistors of  $\frac{1}{4}$ ,  $\frac{1}{2}$  and 5 watts.

### 1.3 Functional Description

Firstly the AC input is directly fed to the **transformer**. Transformer used in the project is one of the common available in the market with having more than 4 output loops. It means that it could provide a variety of output voltages like 3v, 6v, 9v, 12v, 18v and 25v. The two loops of these available loops were used by the project, one for creating negative voltage supply of -5v, -11v and -28v and one for creating positive voltage supply of +5v, +12v. The loop originally providing the 25v AC was used for the negative half, as its RMS value increased up to 35v when it was passed through the bridge rectifier. This is well above the limit of the rectifier that the input voltage must be at least 2 volts greater than the output voltage. Here the care to be taken was in the upper voltage limit of the rectifier which is 38v. to avoid any voltage surge which may take it above 38v, especially

when the transient current is flowing through the transformer coils; we introduced a 5W **resistor** in the power supply circuit that will ensure a minimum flow of current through the circuit all the time.

Another important component coming before the regulator is the **set of capacitors** that are used to convert the pulsating DC into a pure DC signal with very less amplitude variations. The set of capacitors used includes a large value electrolyte capacitor along with a very small value tantalum capacitor. As per formula:

$$f = 1 / 2\pi RC$$

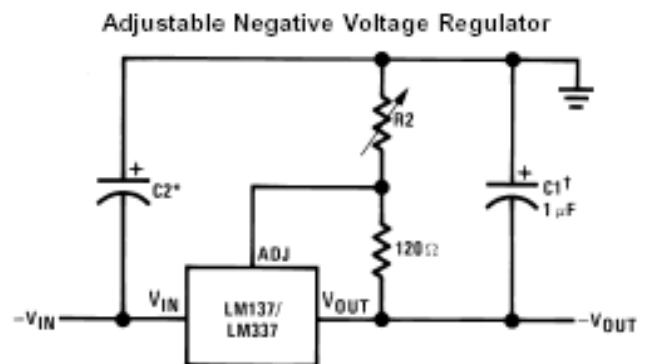
The large capacitor is used to ground the low frequency signal present in the pulsating DC signal while the small value capacitor is to ground the high frequency noise. The optimum values for the electrolyte capacitor are of 470uF where as of tantalum capacitor are of 0.1uF to 1uF.

Among all, the main component is the regulator. The three regulators used in the power supply are LM 337, LM 7805 and LM 7812. LM 337 shown in the diagram is a 3-Terminal Adjustable Negative Regulators and can give different voltage levels depending upon the values of resistors used. Pin 1 is common for both input and output.

TO-220



1. Adj 2. Input 3. Output



**Figure 1.1 Voltage Regulator (a) LM 337 (b) –ve Supply Circuit**

For the adjustment of voltage, two resistors are used, which provide a reference voltage at the adjust pin. One of the two resistors is fixed at 120 Ω while other can be changed to

get the desired voltage level. All the negative voltages -5v, -11v and -28v are created using this regulator. The value of R2 can be calculated through a simple formula given as:

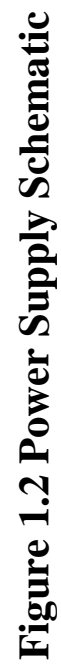
$$-V_{OUT} = -1.25V \left( 1 + \frac{R2}{120} \right) + \left( -I_{ADJ} \times R2 \right)$$

Rest of the circuit is simple and schematic created in Orcad is shown on the next page. Transformer used, consists of two separate windings, one of which is used for +ve voltage supply while other is used for the -ve one. One winding provides the 18v AC while the other provides the 35v AC. This AC voltage is then fed to the bridge rectifier which converts it to the pulsating DC. Capacitors are used extensively at the input and the output. At input, they smooth out the incoming pulsating DC while at the output they provide stability to the supply. Tantalum capacitors of 1uF, 16v are used for high performance and stability of voltage.

LM 7805 and LM 7812 are comparatively simple regulators having three pins. The central pin is common to input and output, called as ground. There are no external resistors used and these regulators provide fixed voltage levels of 5v and 12v respectively.

At the output of the regulators, a set of capacitors with the same settings is again used along with a resistor. The sole purpose of the resistor is to keep a minimum flow of current through the circuit even when the other circuit is not drawing the current. If this setting is not prevailed, there are chances of the regulator to heat up. Both the negative and positive voltage supply are made separately and then the negative terminal of the positive supply and the positive terminal of the negative supply are shorted to create a common ground of the whole circuit. Any voltage levels taken are based on the consideration of this common ground.



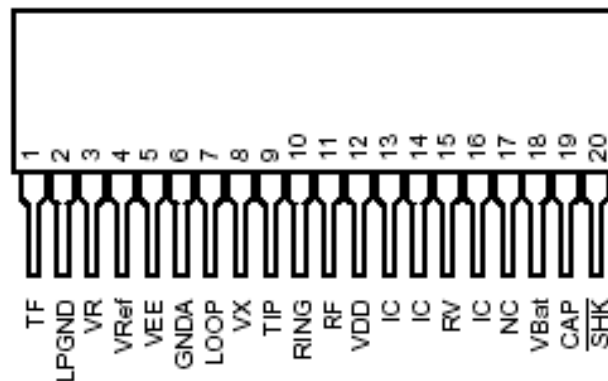


## CHAPTER 2

# Subscriber Line Interface Circuit (SLIC)

### 2.1 Background

The Mitel MH88610 Subscriber Line Interface Circuit provides a complete interface between a switching system and a subscriber loop. It acts as a liaison between the telephone set and the microcontroller circuit. The telephone set functions on the operating voltage of 28 volts DC and of 90 volts AC when the ring is coming. Functions provided include battery feed and ringing feed to the subscriber line, 2-Wire to 4-Wire hybrid interfacing, constant current, loop length and dial pulse detection. The device is fabricated using thick film hybrid technology in a 20-pin single in-line package.



**Figure 2.1 SLIC IC Pin Configuration**

## 2.2 Pin Configuration

List of all the pins of the Mitel SLIC MH88610 along with a brief description is shown in the table below.

### Pin Description

Pin #	Name	Description
1	TF	<b>TF Tip Feed.</b> Internal connection. Normally connects a pair of external diodes for protection.
2	LPGND	<b>Loop Ground.</b> is the system ground reference with respect to VBat.
3	VR	<b>Voice Receive (input)</b> is the 4 wire analog signal to the SLIC.
4	V <sub>Ref</sub>	<b>Voltage Reference (Input)</b> to set the line current feed to the subscriber line.
5	V <sub>EE</sub>	<b>Negative Power Supply Voltage (-5V).</b>
6	GND A	<b>Analog Ground (0V).</b>
7	LOOP	<b>Loop Monitor Voltage (Output).</b> is proportional to the loop length.
8	VX	<b>Voice Transmit (Output)</b> is the 4-wire analog signal from the SLIC.
9	TIP	Connects to the "Tip" lead of the telephone line .
10	RING	Connects to the "Ring" lead of the telephone line.
11	RF	<b>Ring Feed (Input)</b> is normally connected to Ring relay for negative battery feed voltage and ringing voltage input.
12	V <sub>DD</sub>	<b>Positive Power Supply Voltage.</b>
13	IC	<b>Internal Connection.</b> Pin cut short.
14	IC	<b>Internal Connection.</b> Pin cut short.
15	RV	<b>Ring Feed Voltage</b> connects to pin 11 (RF) through a normally closed ring relay.
16	IC	<b>Internal Connection.</b> Pin cut short.
17	NC	<b>No Connection.</b>
18	V <sub>Bat</sub>	<b>Negative Battery Feed Supply Voltage.</b>
19	CAP	Connects external capacitor and resistor to ground for ring trip filter control.
20	SHK	<b>Switch Hook Detect (Output).</b> Digital output of an open-collector comparator. This output will go low (V <sub>EE</sub> ) when the subscriber line resistance falls below a set threshold value indicating that the telephone set has gone off-hook. This output can be monitored for dial pulse collection.

**Table 2.1 SLIC IC Pin Description**

## **2.2 Functional Data**

The Schematic of the SLIC consists of four main portions:

1. SLIC IC
2. Hook Detection Circuit
3. Analogue Multiplexer
4. Ringer Circuit.

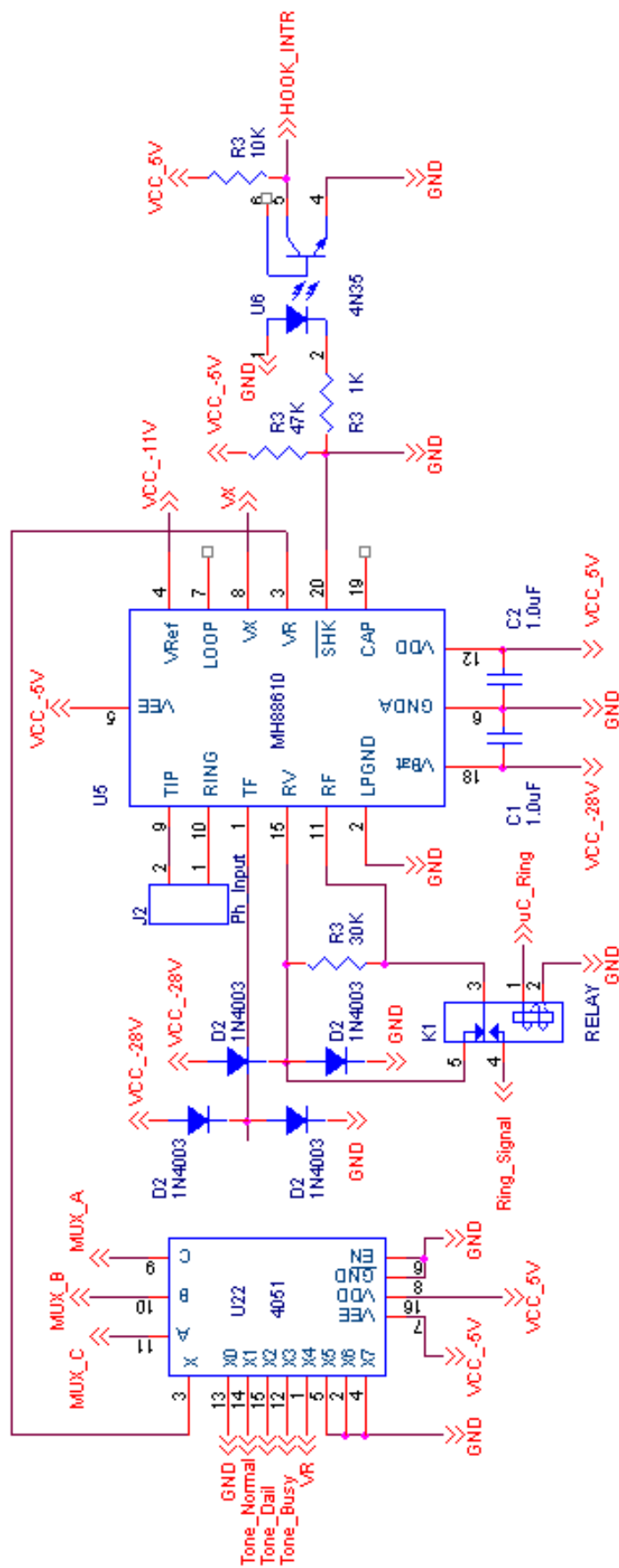
The **SLIC IC** is provided with all the 5 different voltage levels that were created in the power supply circuit. Tip and Ring pins are used to connect the telephone set to the IC. VR and VX are used to receive and transmit voice from the IC. One of the primary functions of the SLIC, to make the 2 wire to 4 wire conversion, is clear from the pins. Pins Tip and Ring contain the differential voice, i.e. both the incoming and the outgoing voice are on one wire, while the other is used as the reference. The two signals on the same wire can't be further processed in this form and hence are separated by the SLIC as the VR for the incoming voice and VX for the transmitted voice. The TF and RV pins are used by the IC for its internal functionality and extra diodes are connected at these pins to provide the necessary protection to the IC. These diodes restrict the pin voltage in between -28v and 0v.

The SHK pin gives the status of the telephone set and is at -5v when the telephone is "On Hook" and at Gnd level when the set is "Off Hook". The **Hook Detection Circuit** is used to make it TTL compatible and gives +5v and Gnd for On and Off Hook respectively. The optocoupler device 4N35 is used to achieve the goal. When the phone is On Hook, the SHK pin of the SLIC is at -5v and it causes no flow of current from the outer pull-up voltage. So the conventional current flows from the Gnd to the -5v source. This causes the transistor in the optocoupler to trigger and it shows 0v at its output. But when the phone set is Off Hook, the

SHK goes to 0v level and the current path build between the SHK pin and the -5v source. The optocoupler remains off and the output shows +5v.

**Analogue Multiplexer** is a component that is used to select one analog signal from the many signals through the control pins. The purpose of its use is that to provide the SLIC with different tones that many be played over the telephone as per needed. These tones can be dial tone, ring tone, busy tone and no sound, as when we feel after pressing a single button of a number. CD4051, a CMOS IC is used in the project and is an 8 input multiplexer. It uses two power supply voltages +5v and -5v, so that the whole signal in between this range may be processed. It takes in four tones of different frequencies and one ground signal to show silence and selects only one of them at a time, depending upon the control pins A, B and C.

**Ringer Circuit** is used in the project to provide Ring voltage of 90Vrms and 25 Hz to the SLIC IC through pin 11. The ringer circuit basically consists of a relay, which is constantly fed with the ring coming from the ringer microcontroller. The relay is also controlled through the ringer microcontroller via a pin. If the telephone set is not to be rung, then the pins 11 and 15 of the SLIC IC must be kept short which is also the default position of the relay. Whenever the telephone ring is to be made, the pin 11 is disconnect from pin 15 and is connected to 90v/25Hz signal. This can be done by only providing +5v to the relay.



**Figure 2.2 SLIC Schematic**

## CHAPTER 3

# DTMF Receiver

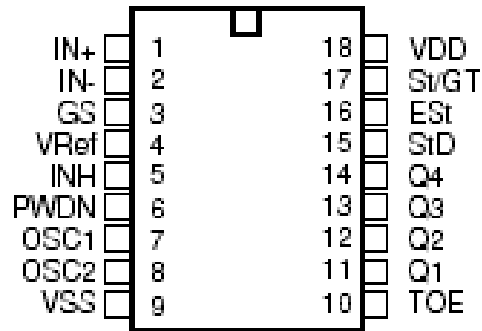
### 3.1 Background

When once the Analogue telephone exchange is created in the form of SLIC and its accessories, the tone corresponding to the key pressed by the user can be received on the VX pin. It can easily be seen on the oscilloscope in the form of sharp band of waves as the key is pressed. This signal is called as the **Dual Tone Multi Frequency** signal as it contains two different frequencies. This DTMF signal comes on the same single wire along with the analogue transmitted voice. A DTMF receiver circuitry is needed to intelligently detect the DTMF tone out of the analogue signal and to correctly recognize which key is pressed. In an ordinary telephone set, there are 12 different keys which can be represented by four bits of data. So, this IC takes in the DTMF tone and gives out the corresponding binary code on the **four output pins**. As soon as the pins represent the valid key pressed, the **interrupt pin** goes high and then again to low to show that the key was pressed. This pin can be polled continuously or used to interrupt the microcontroller to look for the key pressed.

### 3.2 Pin Configuration

The DMTF Receiver, another IC from the Mitel Semiconductor, is provided with the power supply of +5v from the VDD and VSS pins. TOE pin is the Tristate Output Enable pin and can be used to make the data pins as tristate. If this pin is kept low, the output will be high impedance logic. In this case, the TOE pin is permanently tied to the VDD (+5v). ESt pin is an output pin which presents the logic high when a valid tone pair is detected on the VX pin of the SLIC. It remains at the high logic till the key is kept pressed. StD pin presents logic high when a valid tone pair is registered and the output is updated. This pin can be used by the microcontroller to check for any new key being

pressed. Pins Q1 to Q4 are the data pins showing the code of the valid tone pair detected with Q1 being the most significant.



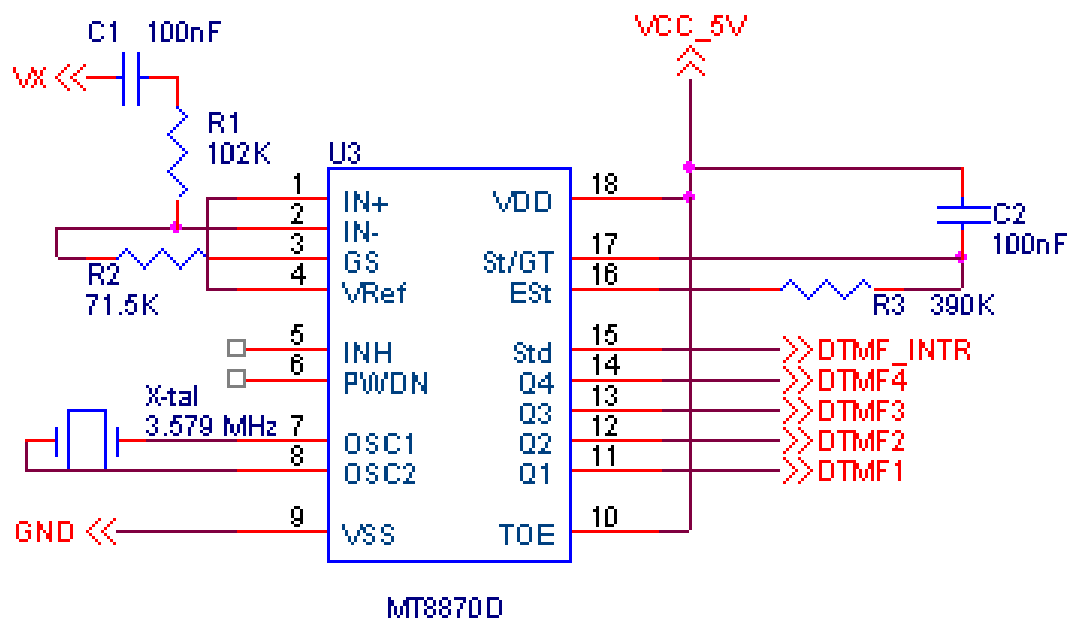
18 PIN PLASTIC DIP/SOIC

**Figure 3.1 DTMF IC Pin Configuration**

### **3.3 Functional Data**

The MT8870D/MT8870D-1 is a complete DTMF receiver integrating both the bandsplit filter and digital decoder functions. The filter section uses switched capacitor techniques for high and low group filters; the decoder uses digital counting techniques to detect and decode all 16 DTMF tone pairs into a 4-bit code. External component count is minimized by on chip provision of a differential input amplifier, clock oscillator and latched three-state bus interface. The DMTF Receiver has at the total of eight pins which have interaction with other subsystems. Two of the pins are the power supply pins while one is the input to the IC and 5 are output from the IC. The input to the IC is made from the VX pin of the SLIC. The analogue voice comes on this pin with occasional presence of DTMF tone pairs. The output pins of the IC are Q1 to Q4 and the StD pin. The StD pin tells about the detection of a valid tone pair and the Q1 to Q4 data pins give the code of the corresponding key pressed.





**Figure 3.2 DTMF Receiver Schematic**

## **CHAPTER 4**

# **CODEC (ADC / DAC)**

### **4.1 Background**

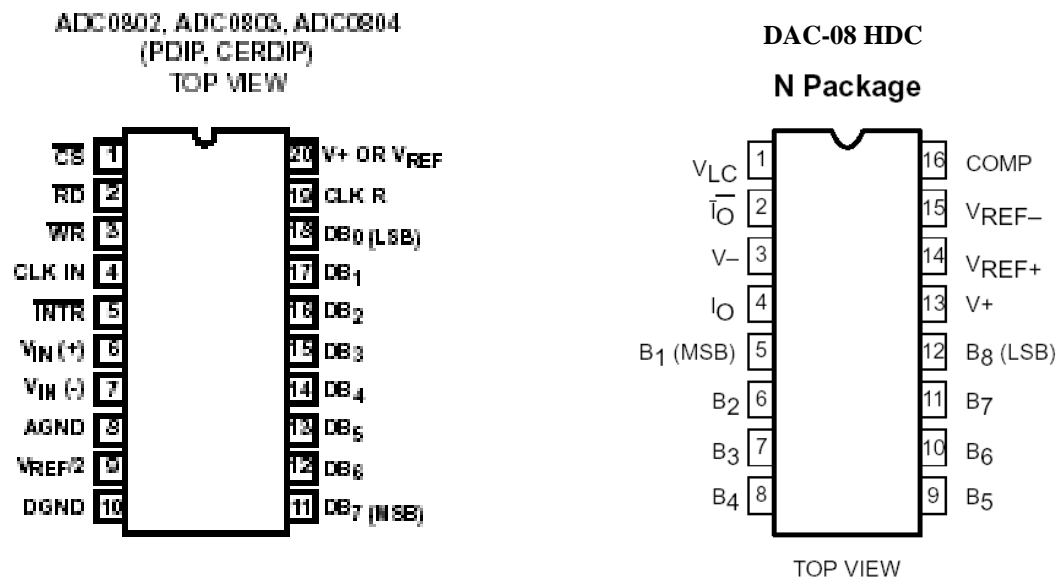
When once the Analogue telephone exchange is created in the form of SLIC and its accessories, there is a need to convert all the analogue voice to the digital and vice versa. All this conversion is needed because the data transferred over the network is only in the form of zeros and ones. Once the voice data is in the digital form, it is then converted in to the packets that are used to transfer it at the other end network card. This conversion takes place with the help of CODEC, derived from the word Coder and Decoder.

### **4.2 Pin Configuration**

#### **4.2.1 Analog to Digital Converter**

Discussing ADC first, it comes in lots of variations and with different accuracy levels and conversion times. ADC0804 used in this project is in Dual in line Package (DIP) and consists of 20 pins. It has got four control pins of CS, RD, WR and INTR. CS is the chip select and is used to activate or deactivate the IC, so it is always kept low. The RD pin is the control pin for the tri-state 8 output pins. RD pin is active low; meaning that it makes the tri-state latches to be transparent when kept low. So both the CS and RD pins are permanently kept low. WR is pin which is used to tell the IC to take the sample from the analogue signal and when the sampling conversion is completed, the INTR pin is taken low by the ADC. In this project we are using the loop back mode in which the WR and INTR pins are tied together such that as soon as one sample takes place another sampling starts. In this mode, the signal from the INTR is used to trigger the WR pin. So when on sample is completed, the INTR pin goes low and as a result the WR pin also goes low which causes the ADC to start preparing the next sample. In this way, the ADC works at its fastest speed. ADC updates its output in this mode after every 110 us, which is the conversion time taken by the ADC to convert the analog signal into its corresponding code. Vin(+) and Vin(-) are the input pins which are used to provide the differential

input, the pin Vin(-) is normally grounded while the pin Vin(+) is used to give the analogue signal. AGND and DGND are the analog and digital ground pins and in this project, are treated as the same and are grounded. The Vref/2 pin is provided by the user with the half of the total voltage, but is not used when we are working over the full range of 0 to 5volts. V+ or Vref pin is provided with the voltage, that is the maximum expected voltage (in our case its +5v). CLK and CLK R pins are used to provide the internal oscillator of the circuit with an external resistor. DB0 to DB7 are the data bits and show the valid code for the sample taken after the conversion is done.



**Figure 4.1 ADC / DAC Pin Configuration**

#### 4.2.2 Digital to Analog Converter

The DAC used in the circuitry is DAC-08HDC which comes in the DIP package with 16 pins. The B1 to B8 pins of DAC are for the data bits that are taken in. The bit B1 is the most significant and the bit B8 is the least. The V+ and V- pins are the power supply pins and are provided with the +5v and -5v respectively. The VR+ and VR- pins are used to provide the reference voltages and these are provided with voltage levels of +5v and 0v. The two IO pin show the output in terms of current and are differential output of each

other. So if one pin contains the current of 0.5A then the other pin must contain the 1.5A of current. The internal structure of the DAC is made up of R2R ladder which allows a specific value of current for each bit. So the DAC provides only the appropriate amount of current which is equivalent to the bits having 1. This current value can be converted to the corresponding value of voltage if a resistor is used across one of the IO pin. The two voltages at the two IO pins will be the reciprocal of each other. VLC is the logic threshold pin and is grounded in the circuit.

### **4.3 Functional Data**

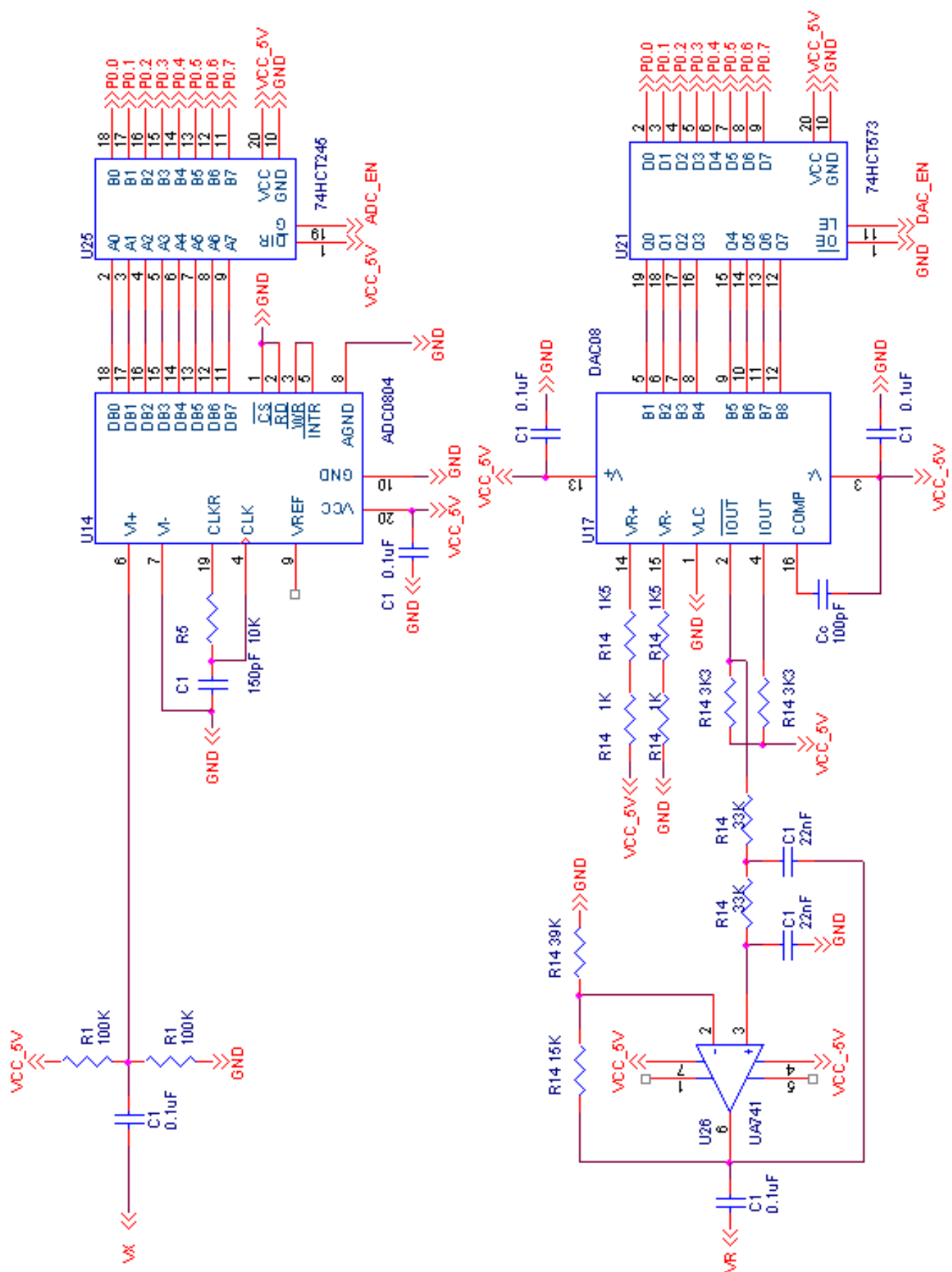
The **ADC0804** series of A/D converters operates on the successive approximation principle. Analog switches are closed sequentially by successive-approximation logic until the analog differential input voltage [ $V_{in(+)} - V_{in(-)}$ ] matches a voltage derived from a tapped resistor string across the reference voltage. The most significant bit is tested first and after 8 comparisons (64 clock cycles), an 8-bit binary code (1111 1111 = full scale) is transferred to an output latch. The normal operation proceeds as follows. On the high-to-low transition of the WR input, the internal SAR latches and the shift-register stages are reset, and the INTR output will be set high. As long as the CS input and WR input remain low, the A/D will remain in a reset state. Conversion will start from 1 to 8 clock periods after at least one of these inputs makes a low to high transition. After the requisite number of clock pulses to complete the conversion, the INTR pin will make a high-to-low transition. This can be used to interrupt a processor, or otherwise signal the availability of a new conversion. A RD operation (with CS low) will clear the INTR line high again. The device may be operated in the free-running mode by connecting INTR to the WR input with CS = 0. To ensure start-up under all possible conditions, an external WR pulse is required during the first power-up cycle. A conversion-in-process can be interrupted by issuing a second start command.

The **DAC-08HDC** series of 8-bit monolithic multiplying Digital-to-Analog Converters provide very high-speed performance coupled with low cost and outstanding applications flexibility. Advanced circuit design achieves 70 ns settling times with very low glitch and at low power consumption. Monotonic multiplying performance is attained over a wide 20-to-1 reference current range. Matching to within 1 LSB between reference and full-scale currents eliminates the need for full-scale trimming in most applications. Direct interface to all popular logic families with full noise immunity is provided by the high swing, adjustable threshold logic inputs. Dual complementary outputs are provided, increasing versatility and enabling differential operation to effectively double the peak-to-peak output swing. True high voltage compliance outputs allow direct output voltage conversion and eliminate output op amps in many applications. All DAC-08 series models guarantee full 8-bit monotonicity and linearities as tight as 0.1% over the entire operating temperature range. Device performance is essentially unchanged over the  $\pm 4.5$  V to  $\pm 18$  V power supply range, with 37 mW power consumption attainable at  $\pm 5$  V supplies. The compact size and low power consumption make the DAC-08 attractive for portable and military aerospace applications.

Both the ADC and DAC are isolated from the data bus of the main microcontroller through the latch HCT 573 and the buffer IC HCT 245. These isolations are necessary because we have many other components directly related to the Bus. If ADC is directly connected to the bus, it will capture the bus all the time and will not allow any other communication through the bus. The ADC provides the output which is present all the times and hence is separated from the other circuitry through the buffer. The purpose of the buffer is to separate the ADC data from the data bus to which the buffer is attached too. Hence, on need basis when the ADC data is needed, the data bus is cleared and the buffer is made transparent.

Similarly, if DAC is not isolated from the bus through the buffer then the other data present on the bus will affect the input of the DAC causing the unpredictable output results.

External to the ADC, the input provided to the ADC0804 is firstly filtered out with the capacitor such that any DC components in the voice signal are eliminated. Then the signal again biased as desired at the level of 2.5 volts. This is achieved by using large but equal valued resistors. The output of the DAC is passed through an Op-Amp filter, which is used to provide isolation to DAC circuitry. If this filter is not used, the circuit being droved by the DAC, may cause an extra flow of current through the DAC output pins which will cause the DAC to change its characteristics and behave unsuspected. The filter circuit used here is a two pole filter, which is merely used to increase the efficiency to filter out high frequency signals. The wave produced by the DAC is somewhat squared and contains unnecessary high frequency signals. The good quality two pole filter works well to eliminate these unwanted signals and works efficiently to reproduce a replica of the original voice signal.



### Figure 4.2 CODEC Schematic

## CHAPTER 5

# Main $\mu$ Controller, Memory Unit and NIC

### 5.1 Background

Technically, Main Microcontroller is the most interactive and the most important part of the whole system. It works as the brain of the whole system and makes all the decisions. On one side, it detects the hook status of the telephone set, and selects the proper tone for the telephone and on the other side; it transfers the packets and does the initial negotiation for the recognition of the network card over the LAN. The main microcontroller is having the most interfaces in the whole system. It has got the interface to almost all the other devices, as it controls all other subsystems.

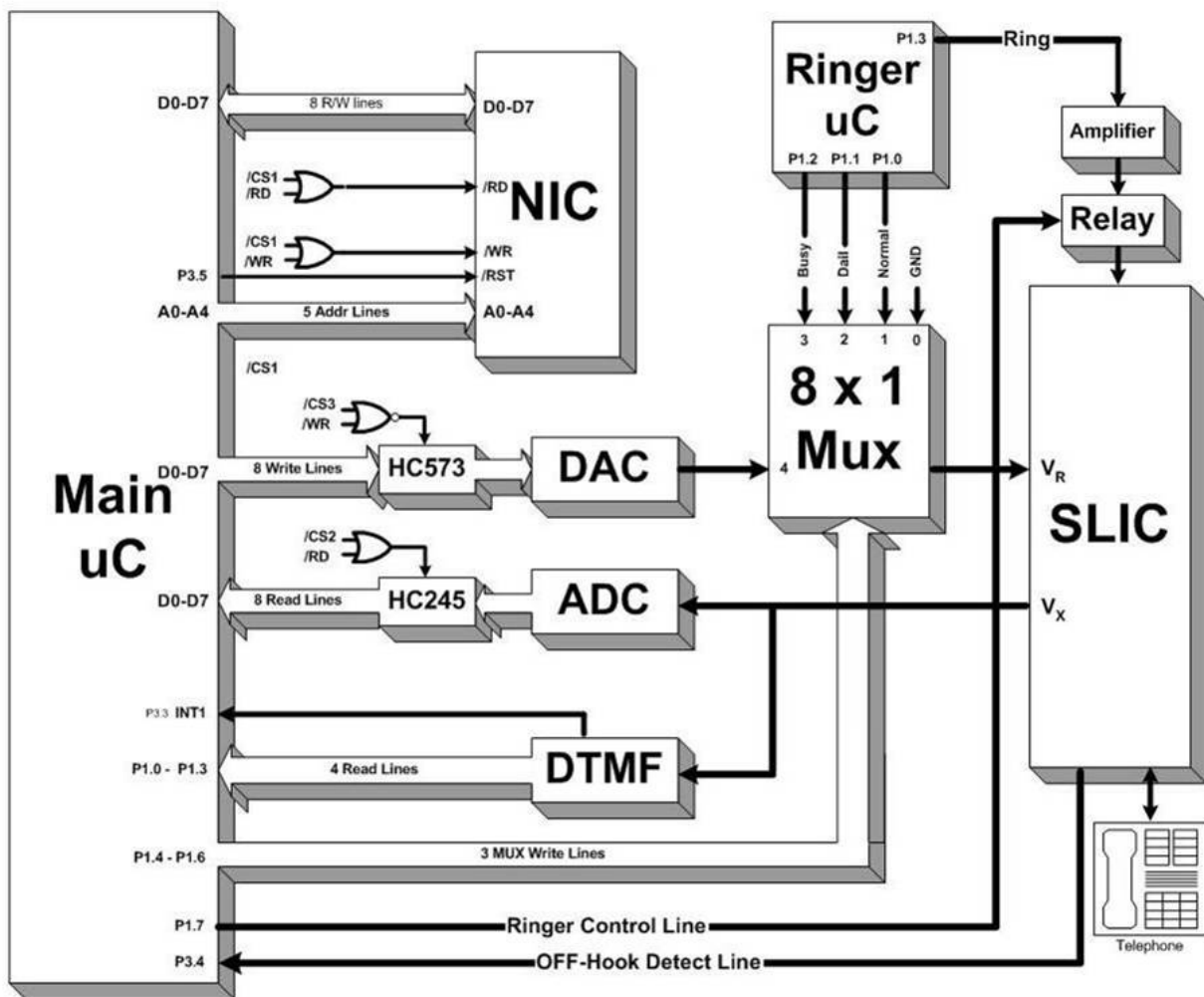
But on the hardware side, the microcontroller is only having many interfaces; else the circuitry is not so complex. All of the tough things related to the main microcontroller are in the software, as the whole NIC driver is written in it. Interface of the NIC is directly to the main microcontroller with the help of the ISA card. ISA card is a 62 + 36 pin slot in which the old designed network card **RTL8019** is fitted in. A total of 16 selected wires from this ISA slot are taken to the main microcontroller.

### 5.2 Microcontroller Description

All the pins of the **Microcontroller** are used in some sort of data acquisition from its surroundings and controlling based on this data. **Port 0** of the microcontroller is used as the Data bus, and **Port 2** is used occasionally as the Address bus. The Memory Mapping architecture is used such that all the devices like ADC, DAC, Memory Unit and NIC can be accessed in the software through the specific memory addresses and the same data and address buses are used for both. Similarly, the NIC interfaced through the ISA slot is also connected to the Ports P0 and P2 of the microcontroller. **Port 1** is partially used in different jobs; Pins P1.0 to P1.3 are used to get the key codes from the DTMF receiver. Pins P1.4 to P1.6 are used to control the output of the analogue multiplexer IC's CD 4051. The last Pin P1.7 is used to turn the Ringer of the telephone ON and OFF by



controlling the Relay. Port 2, as already told is used for the memory mapping architecture by enabling the various devices from the address bus pins and for the addressing of the NIC. **Port 3** of the microcontroller is involved in some special tasks like receiving the interrupts and for polling the status pins. Pin P3.3 is used for the receiving the interrupt from the DTMF receiver IC that a new has been pressed on the telephone keypad. Pin P3.4 is used to check the status of the Hook obtained directly from the SLIC IC that the telephone has been pick up or not. This status doesn't trigger any interrupt and is polled all the time. Pin P3.5 is used to reset the NIC at the start of the System. Every time driver



**Figure 5.1 The VoIP System Big Picture**

is reinitialized, the Reset pin is also used. Pins P3.6 and P3.7 are used to complete the memory mapping architecture, as these pins inform the attached peripherals that the command issued is of read or write.

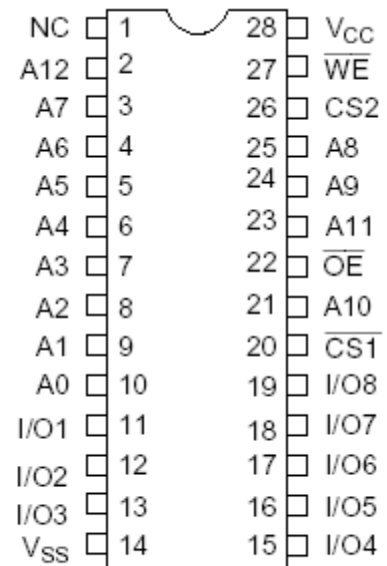
A number of **Logic Gates** are used with inputs having the combinations from the RD/WR and the ADC, DAC and NIC address pins. These combinations are ORed to produce the signal that can uniquely select these devices. The output from the ORing of DAC and the WR is inverted as the DAC is in active low logic. The ALE pin is used to latch the address lines and is used specifically in the case of Memory Interfacing. A close look at the big picture in figure 5.1 makes it all understandable easily.

### **5.3 Memory Chip Description**

The External **Memory Module HY6264AL** used with the microcontroller is due to the reason that the internal memory consists only of 128 bytes and is totally inadequate to store the packets ready to send and those just received. These

#### **Pin Description**

Pin name	Function
A0 to A12	Address input
I/O1 to I/O8	Data input/output
$\overline{CS1}$	Chip select 1
CS2	Chip select 2
$\overline{WE}$	Write enable
$\overline{OE}$	Output enable
NC	No connection
$V_{CC}$	Power supply
$V_{SS}$	Ground



(Top view)

**Figure 5.2 Memory Chip Pin Configuration**

packets contain 200 voice samples which are later played on the DAC and samples are taken from the ADC after the regular intervals of 125us. The packets saved in this memory are in the form of arrays and at the total four packets are placed, consuming a total of almost 400 bytes of memory. On the hardware level, the memory unit contains a total of 13 address lines which can address a total of 8 KB of memory. Two of the MSB lines are hard coded to Gnd and the rest 11 lines are used making 2 KB of memory accessible to us. The chip is enabled by two chip select pins CS1 and CS2, one on logic high and other on logic low. To enable the chip, both the pins are given with the proper combination. The OE pin is used to read the data from the memory unit and the WR to write the data on the memory.

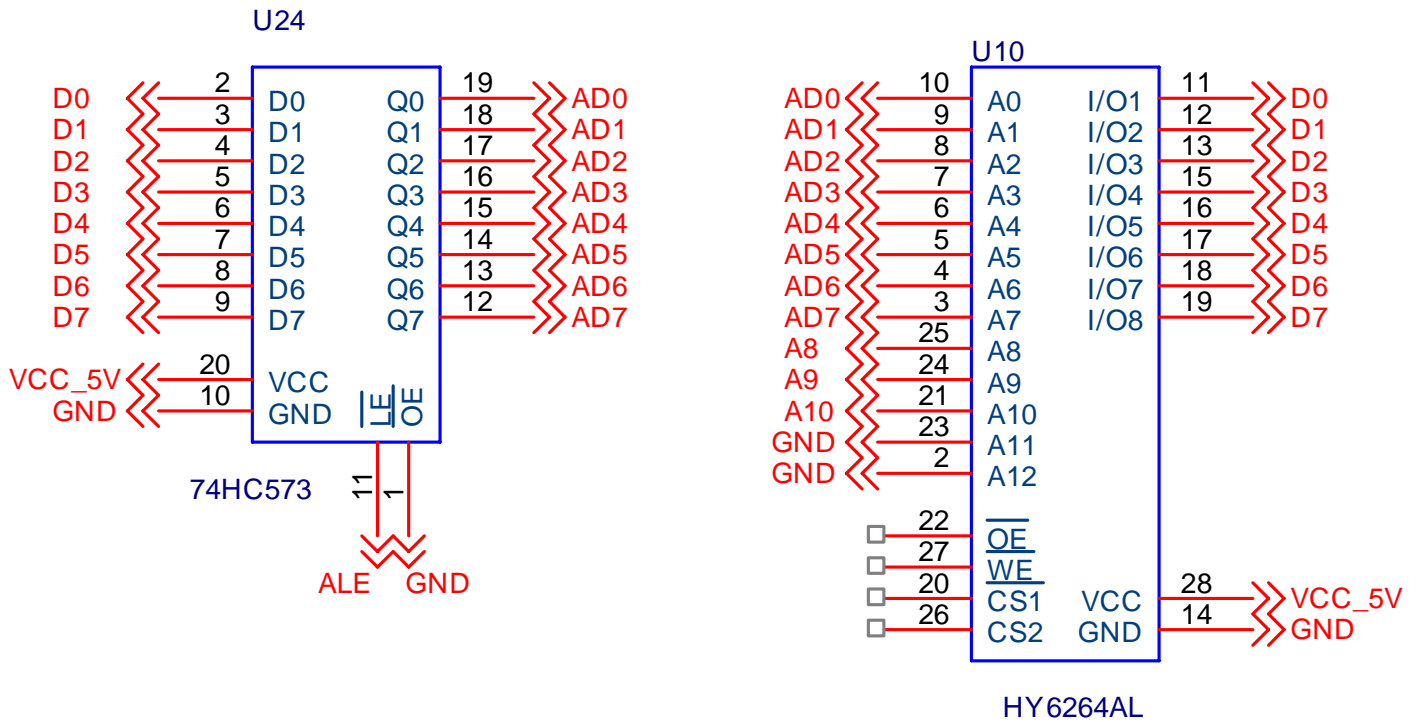
**Function Table**

$\overline{WE}$	$\overline{CS1}$	$CS2$	$\overline{OE}$	Mode	$V_{CC}$ current	I/O pin	Ref. cycle
x	H	x	x	Not selected (power down)	$I_{SB}, I_{SB1}$	High-Z	—
x	x	L	x	Not selected (power down)	$I_{SB}, I_{SB1}$	High-Z	—
H	L	H	H	Output disable	$I_{CC}$	High-Z	—
H	L	H	L	Read	$I_{CC}$	Dout	Read cycle (1)–(3)
L	L	H	H	Write	$I_{CC}$	Din	Write cycle (1)
L	L	H	L	Write	$I_{CC}$	Din	Write cycle (2)

Note: x: H or L

**Table 5.1 Memory Chip Function Table**

The memory unit is provided with 8 data wires and 11 address wires. The lower bits of the address comes on the same data bus, just before the data comes. This address is latched by using the HCT573 using the ALE pin. The ALE pin goes to logic high till the address is valid on the bus and goes to low again before the data comes. The two bits of the address come directly from the Port 2 and are not latched at all. The ALE signal comes from the microcontroller directly and is only used with the external memory.



**Figure 5.3 Memory Module Circuitry**

#### **5.4 Network Interface Card Description**

Computer components send and receive data between different devices by the use of a bus. The design and type of the bus therefore has a crucial effect on how well a computer system will operate. If you have a high-speed drive and a fast CPU, but a slow bus, data will be held up and the individual components will not operate at the speed they are capable.

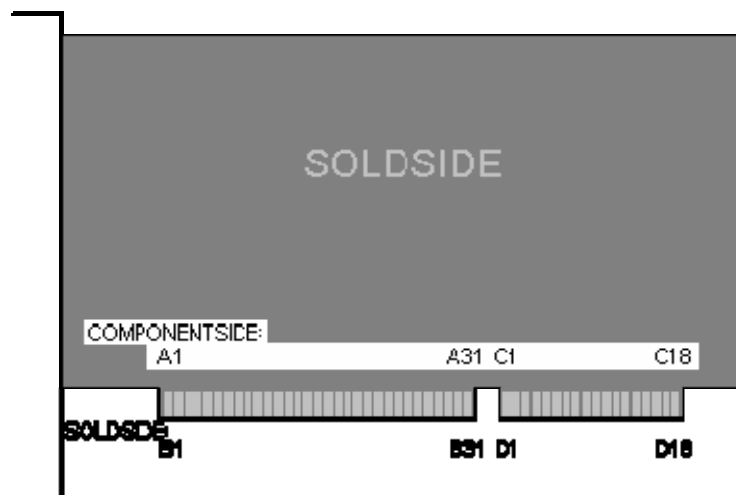
The Industry Standard Architecture (ISA) bus was originated in the early 1980s at an IBM development lab in Boca Raton, Florida. The original IBM Personal Computer introduced in 1981 included the 8-bit subset of the ISA bus. This bus was produced for many years without any formal standard. In recent years, a more formal standard called the ISA bus (Industry Standard Architecture) has been created. In 1984, IBM introduced

the PC-AT, which was the first full 16-bit implementation of the ISA bus called the EISA (Extended ISA) bus.

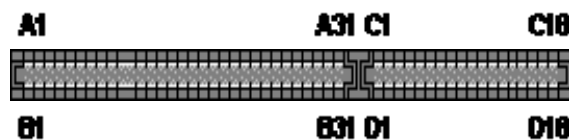
Interfacing with I/O space 64k bytes of ISA I/O are available on the ISA bus while interfacing with memory spaces ISA bus memory has 20 address bits, giving it a total range of 1MB. There are 98 pins in 16-bit ISA Card, out of which we used 5-pins as address lines, 3-pins as control Bits (Read, Write and Reset) and 8-pins as data bits as shown in figure 5.6.

#### **5.4.1 Physical Design**

ISA cards can be either 8-bit or 16-bit. An 8-bit card only uses the first 62 pins and 16-bit cards uses all 98 pins.

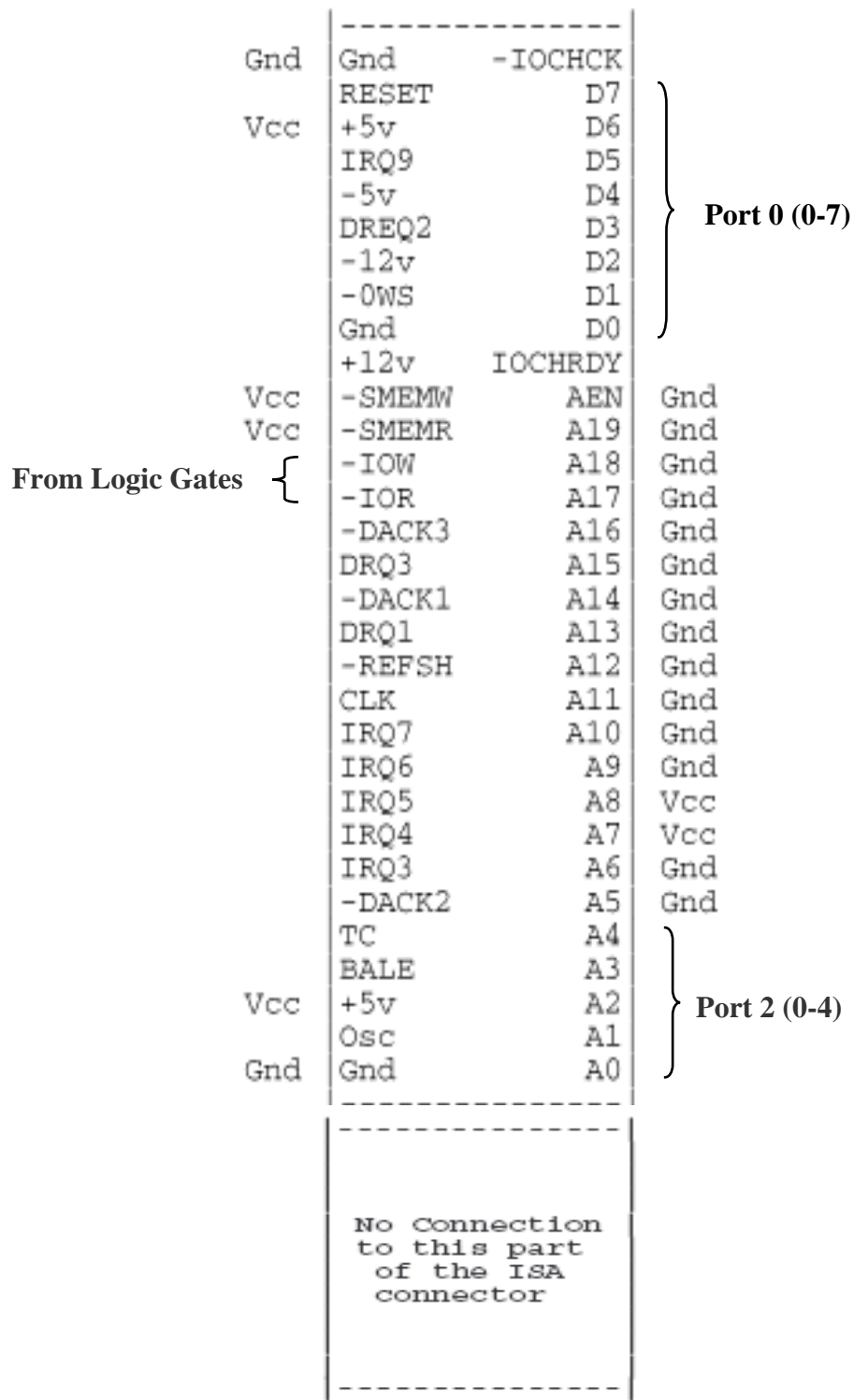


**(a) At the Card**



**(b) At the Computer**

**Figure 5.4 ISA Card**



**Figure 5.5 ISA Connector**

### **5.4.2 Signal Descriptions**

A brief description of the pins used in the project is given below:

#### **IOR**

The I/O Read is an active-low signal, which instructs the I/O device to drive its data onto the data bus, D0-D7.

#### **IOW**

The I/O Write is an active-low signal, which instructs the I/O device to read data from the data bus, D0-D7.

#### **RESET**

This signal goes low when the machine is powered up. Driving it low will force a system reset. This signal goes high to reset the system during power up, low line-voltage or hardware reset. This pin is attached with Port1.

#### **A0-A19**

System Address Lines. The System Address lines run from bit 0 to bit 19. We have used five address lines (from A0 to A4).

#### **D0-D7**

System Data lines or Standard Data Lines. They are bidirectional and tri-state. On most systems, the data lines float high when not driven. These 8 lines provide for data transfer between the processor, memory and I/O devices.

#### **SMEMR**

System Memory Read Command line. Indicates a memory read in the lower 1 MB area. This System Memory Read is an active-low signal, which instructs memory devices to drive data onto the data bus D0-D7. This signal is active only when the memory address is within the lowest 1MB of memory address space.

## **SMEMW**

System Memory Write Command line. Indicates a memory write in the lower 1 MB area. The System Memory Write is an active-low signal, which instructs memory devices to store data preset on the data bus D0-D7. This signal is active only when the memory address is within the lowest 1MB of memory address space.

Only the three pins (SMEMW, SMEMR, +5V) are kept high (VCC) while all other pins that are not used in our project are grounded (Gnd).





## CHAPTER 6

# Ringer $\mu$ Controller & Amplifier Circuit

Just like the main microcontroller the ringer microcontroller has also got a simple circuitry and all the logic is in the software. But on the other hand, the amplifier circuit is complex and uses hard electronics concepts.

### 6.1 Pin Configuration

The **Ringer Microcontroller** is **89C2051**, a smaller version of the 8051 microcontroller from the Atmel with only 20 pins. It is used in the circuit, where the hardware is minimally used. This mini-microcontroller is provided with only two ports P1 and P3 with P3.6 not present. The two pins P1.0 and P1.1 are open collector, meaning that the external Pull-up source is needed to use the pins. Avoiding complexity, the pins P1.0 and P1.1 are not used. Only four pins from P1.2 to P1.5 from the whole microcontroller are used. The three pins P1.2 to P1.4 are used to produce various tones that can be heard when the phone receiver is picked up.

### 6.2 Functional Description

The tones created by the 2051 are provided to the analog multiplexer CD4051 which switches to one of them as needed. The three tones created include the Normal Tone, Dial Tone and the Busy Tone. These tones are produced constantly and not on demand basis. The tone signals generated from the microcontroller has two problems; the signal oscillates between 0 and 5v with an offset of 2.5v and the signals produced are purely square waves containing high frequency harmonics. The DC biasing of 2.5v of the signal is undesired as the SLIC IC takes in only analog signal with no DC components. This purpose is achieved by adding a 2uF electrolytic capacitor. The high frequency harmonics make the tone very shrill, and is very unpleasant to be heard. These components are removed by using a simple RC filter circuit. This filter is of low efficiency but is just as desired in our case. The output signal after passing these two stages is just as rounded square wave. The pure sine wave gives the best desired audio

results but takes too much circuitry and hence, is avoided here. The Tone Ring is not passed through any filter as it is filtered in the Amplifier Circuitry. The frequencies used for the tones along with their cadence are given in the table below. Cadence is period of time for which the wave is kept on and off. These standards are taken from the ones used in Pakistan and are similar to American Standards.

Category	Frequency	Cadence
Normal Tone	560 Hz	None
Dial Tone	560 Hz	2 sec ON, 4 sec OFF
Busy Tone	600 Hz	0.5 sec ON, 0.5 sec OFF
Ring Tone	25 Hz	2 sec ON, 4 sec OFF

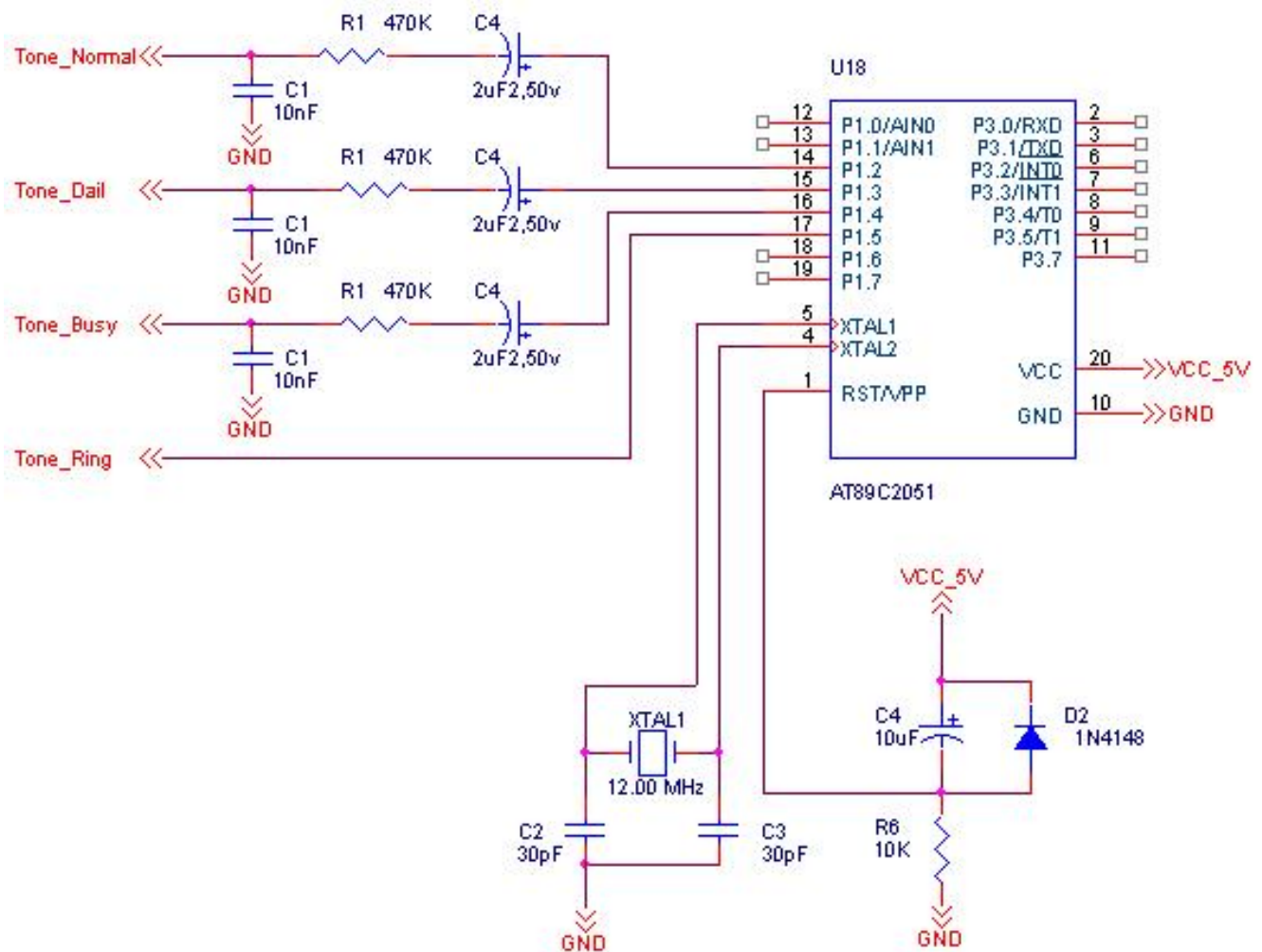
**Table 6.1 Telephone Tone Standards Used**

The **Amplifier Circuit** has got the basic the basic need from the ring voltage standards which is provided to the telephone set through the SLIC IC. The ring signal is of 25 Hz and of 90 volts RMS. So, the amplifier is used solely to step-up the signal of 5v peak to peak coming from the microcontroller to the voltage of 110 volt peak to peak, which will count to 90 v RMS. Firstly in the circuit, a pair of two RC filters is used to get the desired wave. Then the wave is inverted by putting the signal at the –ve input of the op-amp. Now two versions of the source signal exist with a shift of  $180^0$ . These two signals are used to turn ON two transformers each one in different half cycle. The signal is passed through the unity gain Op-Amps, so that the original signal is not distorted by the draining of the current by power transistors. A diode is used just at the input of the coupling op-amp, which cuts down the negative half cycle of the wave. So, after these steps the wave produced is a just a positive half cycle, one on upper half circuit and the

other on lower half circuit. These two distinct signals are then provided at the base of the power transistors such that they produce a large current amplification at the emitter. This current amplification is used on the principle that if we are going to step up the voltage of a signal, its current will be decreased proportionally as per formula:

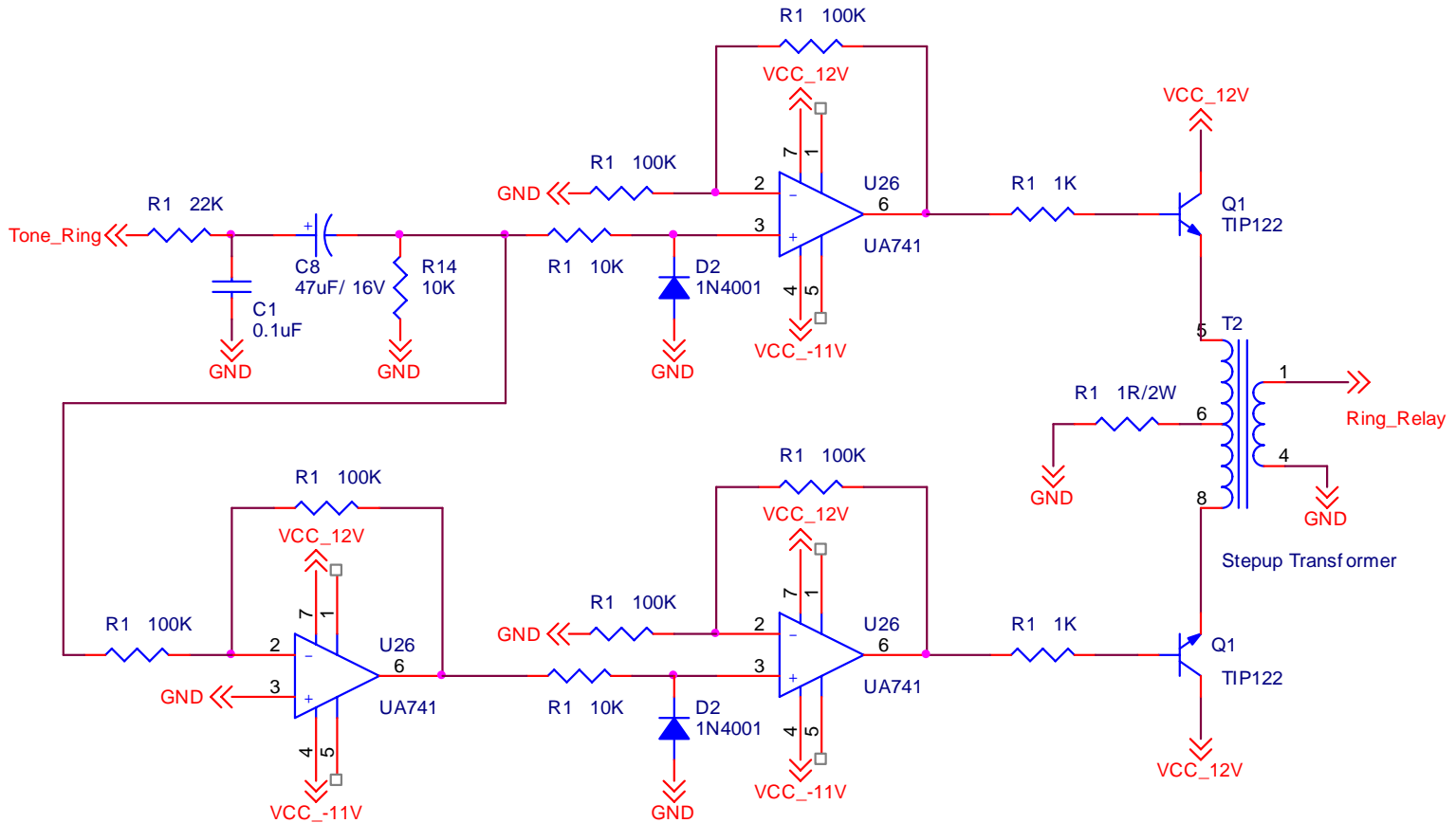
$$V1 * I1 = V2 * I2$$

So the current is amplified such that it might not get extremely feeble after the signal is amplified. The power transistors used are the TIP 122 which can tolerate high flows of currents. The current amplified signal is then applied to the center taped transformer,



**Figure 6.1 Ringer  $\mu$ Controller Circuit**

having the windings ratio of 1:20 so that we may get the desired output voltage amplification. The output of the transformer is fed directly to the relay in the SLIC circuit. Ringing signal is also continuously produced and is just connect to the SLIC when needed.



**Figure 6.2 Ring Amplifier Circuit**

## **PART TWO**

# **NETWORK INTERFACE CARD** **INTERNAL DESIGN**

RTL8019AS is an ISA slot based network card from the Realtek Semiconductor. It is based on the standard chip NE 2000 from the National Semiconductor. This chip is an open architecture chip, with the details regarding its internal design, registers, memory arrays all given by the National Semiconductor in its datasheet D8390 being referred at the last. By continuous and close contact with the above mentioned datasheet one can easily understand the NIC architecture.

Most of the registers in the NIC are not used in this project. These registers deal mainly when the NIC is operated in the interrupt mode and interrupts are being generated and checked. Moreover, the several error conditions like buffer overflow, buffer overrun are not treated in the software of this project as the necessary precautions are taken already to prevent such happenings.

## **CHAPTER 7**

# **Network Interface Card Internal Design**

### **7.1 General Description**

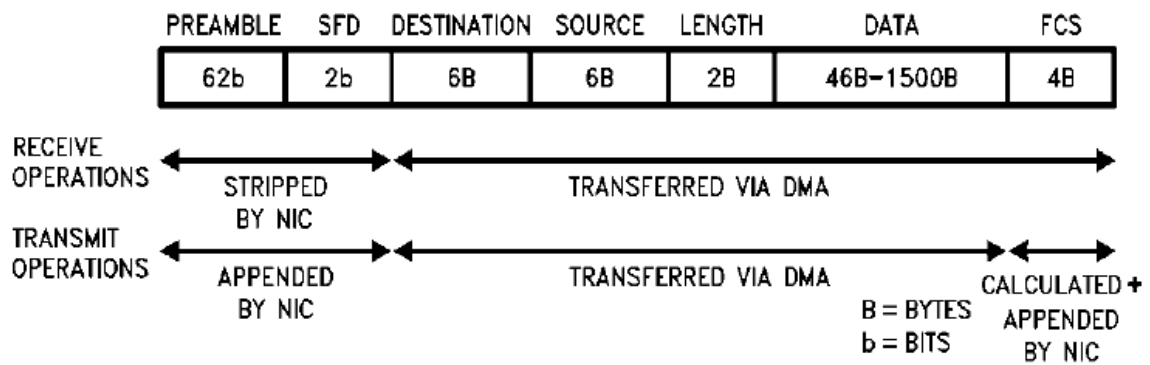
The Network Interface Card (NIC) is a MicroCMOS VLSI device designed to ease interfacing with CSMA/CD type local area networks including Ethernet, Thin Ethernet (Cheapernet) and StarLAN. The NIC implements all Media Access Control (MAC) layer functions for transmission and reception of packets in accordance with the IEEE 802.3 Standard. Unique dual DMA channels and an internal FIFO provide a simple yet efficient packet management design. To minimize system parts count and cost, all bus arbitration and memory support logic are integrated into the NIC.

### **7.2 Features**

- Compatible with IEEE 802.3/Ethernet II/Thin Ethernet/Star LAN
- Interfaces with 8-, 16- and 32-bit microprocessor systems
- Implements simple, versatile buffer management
- Requires single 5V supply
- Utilizes low power microCMOS process
- Includes
  - Two 16-bit DMA channels
  - 16-byte internal FIFO with programmable threshold
  - Network statistics storage
- Supports physical, multicast, and broadcast address filtering
- Provides 3 levels of loopback
- Utilizes independent system and network clocks

### **7.3 Transmit/Receive Packet**

A standard IEEE 802.3 packet consists of the following fields: Preamble, Start of Frame Delimiter (SFD), Destination address, Source address, Length, Data and Frame Check Sequence (FCS). The typical format is shown in Figure 3.1. The packets are Manchester encoded and decoded by the DP8391 SNI and transferred serially to the NIC using NRZ data with a clock. All fields are of fixed length except for the data field. The NIC generates and appends the preamble, SFD and FCS field during transmission. The Preamble and SFD fields are stripped during reception. (The CRC is passed through to buffer memory during reception.)



**Figure 7.1 Standard Ethernet Packet**

#### **PREAMBLE AND START OF FRAME DELIMITER (SFD)**

The Manchester encoded alternating 1,0 preamble field is used by the SNI to acquire bit synchronization with an incoming packet. When transmitted each packet contains 62 bits of alternating 1,0 preamble. Some of this preamble will be lost as the packet travels through the network. The preamble field is stripped by the NIC. Byte alignment is performed with the Start of Frame Delimiter (SFD) pattern, which consists of two consecutive 1's. The NIC does not treat the SFD pattern as a byte, it detects only the two



bit pattern. This allows any preceding preamble within the SFD to be used for phase locking.

### **DESTINATION ADDRESS**

The destination address indicates the destination of the packet on the network and is used to filter unwanted packets from reaching a node. There are three types of address formats supported by the NIC: physical, multicast, and broadcast. The physical address is a unique address that corresponds only to a single node. All physical addresses have an MSB of ``0". These addresses are compared to the internally stored physical address registers. Each bit in the destination address must match in order for the NIC to accept the packet. Multicast addresses begin with an MSB of ``1". If the address consists of all 1's it is a broadcast address, indicating that the packet is intended for all nodes. A promiscuous mode allows reception of all packets: the destination address is not required to match any filters. Physical, broadcast, multicast, and promiscuous address modes can be selected.

### **SOURCE ADDRESS**

The source address is the physical address of the node that sent the packet. Source addresses cannot be multicast or broadcast addresses. This field is simply passed to buffer memory.

### **LENGTH FIELD**

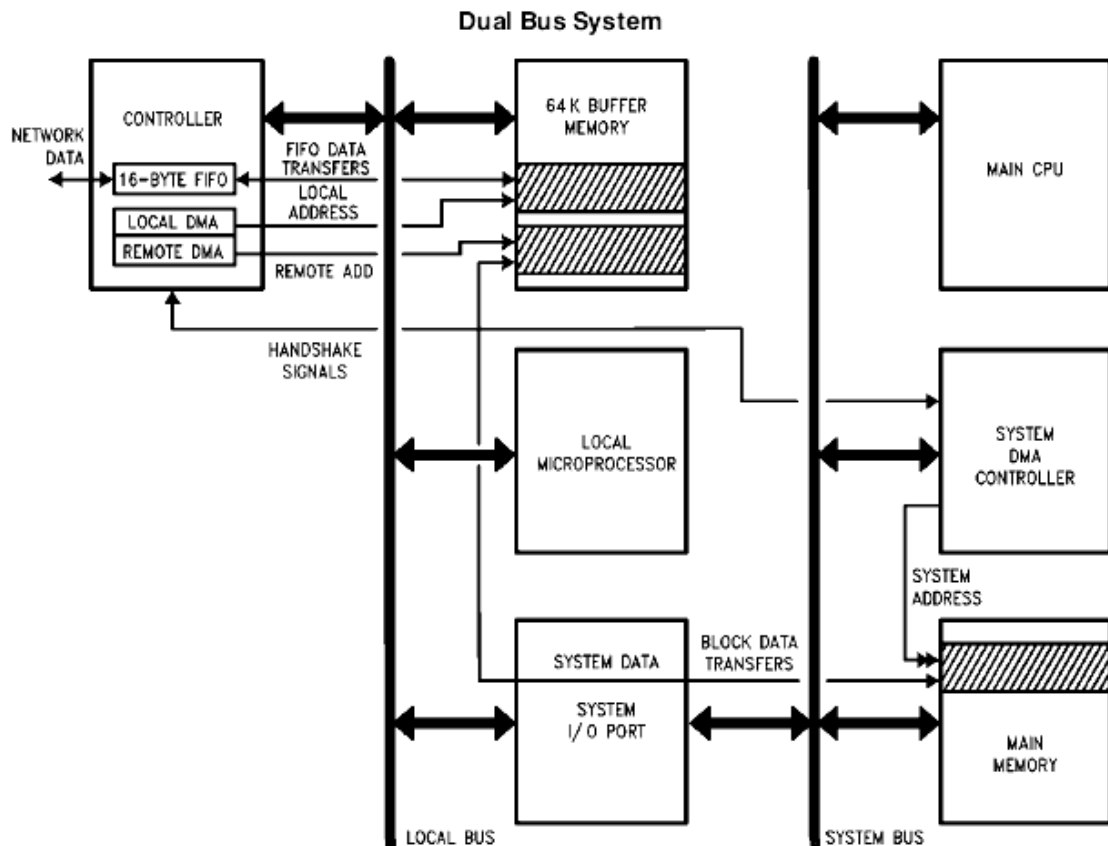
The 2-byte length field indicates the number of bytes that are contained in the data field of the packet. This field is not interpreted by the NIC.

### **DATA FIELD**

The data field consists of anywhere from 46 to 1500 bytes. Messages longer than 1500 bytes need to be broken into multiple packets. Messages shorter than 46 bytes will require appending a pad to bring the data field to the minimum length of 46 bytes. If the data field is padded, the number of valid data bytes is indicated in the length field. The NIC does not strip or append pad bytes for short packets, or check for oversize packets.

## FCS FIELD

The Frame Check Sequence (FCS) is a 32-bit CRC field calculated and appended to a packet during transmission to allow detection of errors when a packet is received. During reception, error free packets result in a specific pattern in the CRC generator. Packets with improper CRC will be rejected.



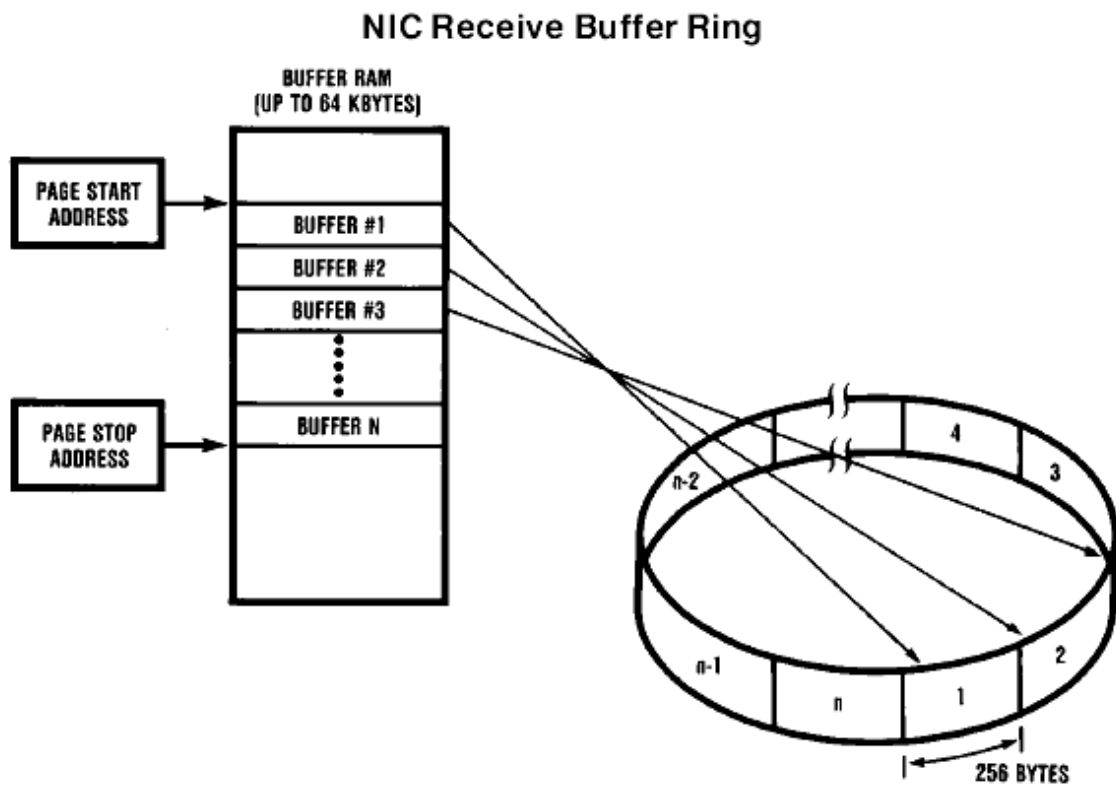
**Figure 7.2 Dual Bus System**

### 7.4 Packet Reception

The Local DMA receive channel uses a Buffer Ring Structure comprised of a series of contiguous fixed length 256 byte buffers for storage of received packets. The location of the Receive Buffer Ring is programmed in two registers, a Page Start and a Page Stop Register. Ethernet packets consist of a distribution of shorter link control packets and

longer data packets, the 256 byte buffer length provides a good compromise between short packets and longer packets to most efficiently use memory. In addition these buffers provide memory resources for storage of back-to-back packets in loaded networks. The assignment of buffers for storing packets is controlled by Buffer Management Logic in the NIC. The Buffer Management Logic provides three basic functions: linking receive buffers for long packets, recovery of buffers when a packet is rejected, and recirculation of buffer pages that have been read by the host.

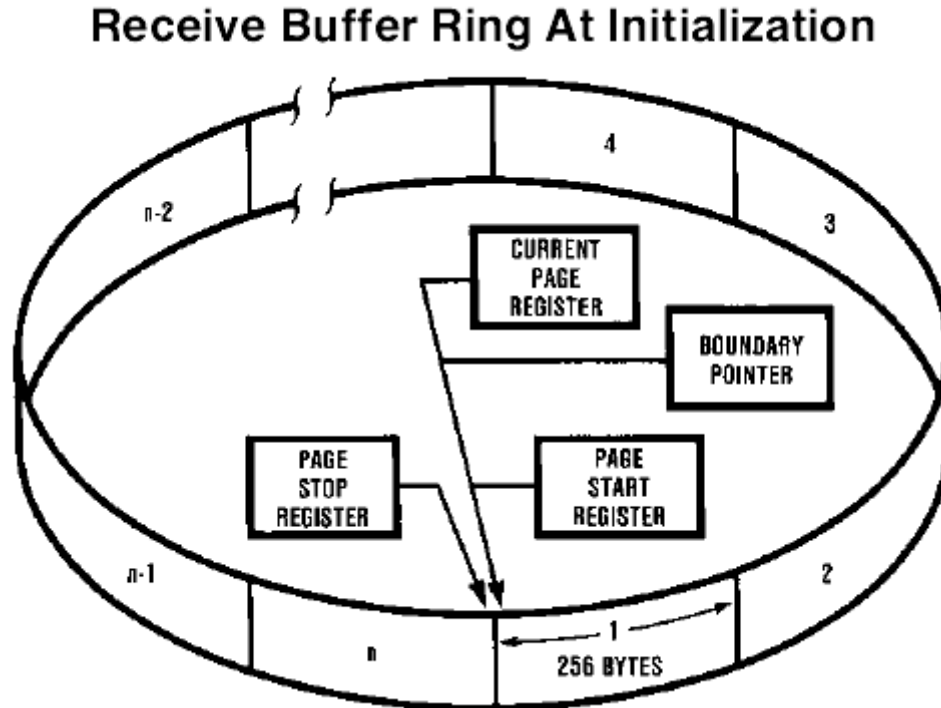
At initialization, a portion of the 64k byte address space is reserved for the receive buffer ring. As shown in Figure 3.4 two eight bit registers, the Page Start Address Register (PSTART) and the Page Stop Address Register (PSTOP) define the physical boundaries of where the buffers reside. The NIC treats the list of buffers as a logical ring; whenever the DMA address reaches the Page Stop Address, the DMA is reset to the Page Start Address.



**Figure 7.3 NIC Receiver Buffer**

### **7.4.1 Initialization of the Buffer Ring**

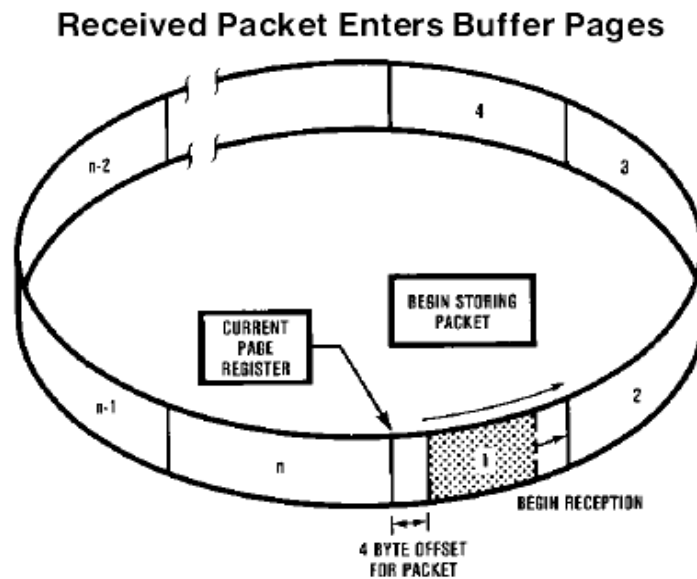
Two static registers and two working registers control the operation of the Buffer Ring. These are the Page Start Register, Page Stop Register, the Current Page Register and the Boundary Pointer Register as shown in Figure 3.5. The Current Page Register points to the first buffer used to store a packet and is used to restore the DMA for writing status to the Buffer Ring or for restoring the DMA address in the event of a Runt packet, a CRC, or Frame Alignment error. The Boundary Register points to the first packet in the Ring not yet read by the host. If the local DMA address ever reaches the Boundary, reception is aborted. The Boundary Pointer is also used to initialize the Remote DMA for removing a packet and is advanced when a packet is removed. A simple analogy to remember the function of these registers is that the Current Page Register acts as a Write Pointer and the Boundary Pointer acts as a Read Pointer.



**Figure 7.4 Receiver Buffer Ring at Initialization**

### **7.4.2 Beginning of Reception**

When the first packet begins arriving the NIC begins storing the packet at the location pointed to by the Current Page Register. As shown in Figure below an offset of 4 bytes is saved in this first buffer to allow room for storing receive status corresponding to this packet.



**Figure 7.5 Received Packet Enters Buffer Pages**

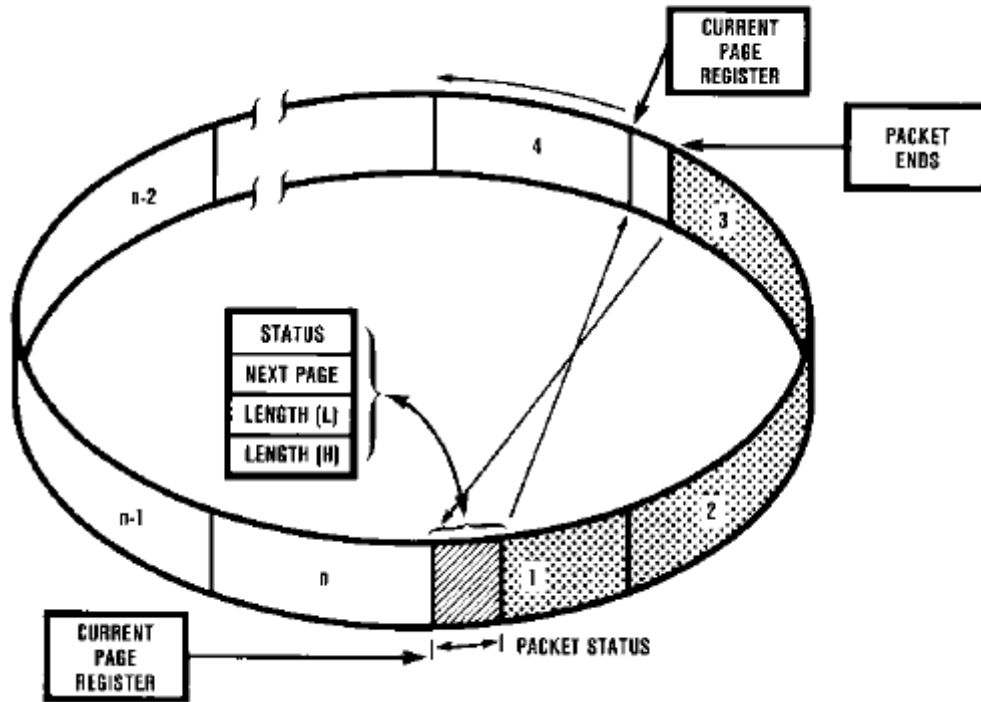
### **7.4.3 End of Packet Operations**

At the end of the packet the NIC determines whether the received packet is to be accepted or rejected. It either branches to a routine to store the Buffer Header or to another routine that recovers the buffers used to store the packet.

### **7.4.4 Successful Reception**

If the packet is successfully received, the DMA is restored to the first buffer used to store the packet (pointed to by the Current Page Register). The DMA then stores the Receive Status, a Pointer to where the next packet will be stored (Buffer 4) and the number of received bytes. This is shown the Figure 7.10.

### Termination of Received Packet—Packet Accepted



**Figure 7.6 Packets Accepted**

The remaining bytes in the last buffer are discarded and reception of the next packet begins on the next empty 256-byte buffer boundary. The Current Page Register is then initialized to the next available buffer in the Buffer Ring.

#### **7.4.5 Removing Packets from the Ring**

Packets are removed from the ring using the Remote DMA or an external device this is shown in Figure 3.12 on the next page. When using the Remote DMA the Send Packet command can be used. This programs the Remote DMA to automatically remove the received packet pointed to by the Boundary Pointer. At the end of the transfer, the NIC

moves the Boundary Pointer, freeing additional buffers for reception. The Boundary Pointer can also be moved manually by programming the Boundary Register. Care should be taken to keep the Boundary Pointer at least one buffer behind the Current Page Pointer. The following is a method for maintaining the Receive Buffer Ring pointers.

1. At initialization, set up a software variable (next\_pkt) to indicate where the next packet will be read. At the beginning of each Remote Read DMA operation, the value of next\_pkt will be loaded into RSAR0 and RSAR1.

2. When initializing the NIC set:

$$\text{BNDRY} = \text{PSTART}$$

$$\text{CURR} = \text{PSTART} + 1$$

$$\text{Next\_pkt} = \text{PSTART} + 1$$

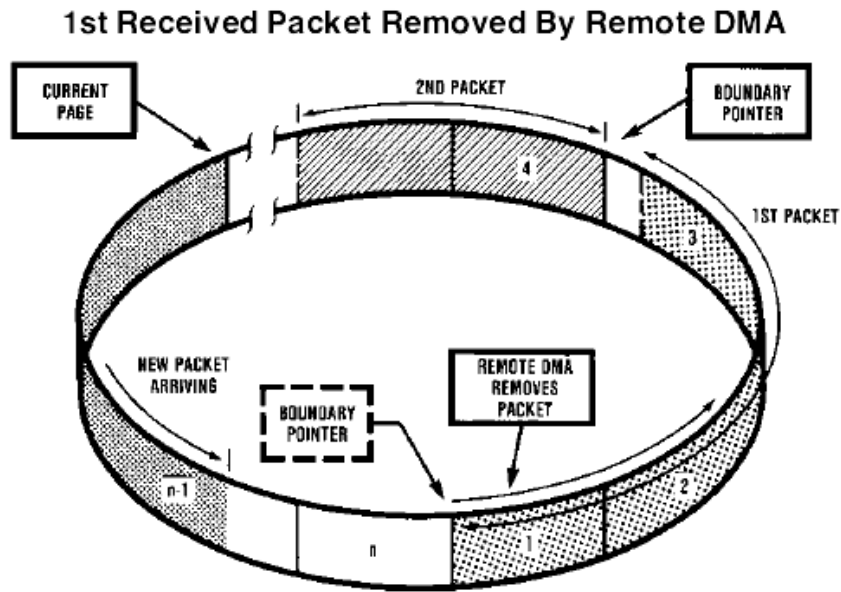
3. After a packet is DMAed from the Receive Buffer Ring, the Next Page Pointer (second byte in NIC buffer header) is used to update\_BNDRY and next\_pkt

$$\text{Next\_pkt} = \text{Next Page Pointer}$$

$$\text{BNDRY} = \text{Next Page Pointer} - 1$$

$$\text{If } \text{BNDRY} < \text{PSTART} \text{ then } \text{BNDRY} = \text{PSTOP} - 1$$

Note the size of the Receive Buffer Ring is reduced by one 256-byte buffer; this will not, however, impede the operation of the NIC.



**Figure 7.7 First Received Packet Removed By Remote DMA**

#### **7.4.6 Storage Format for Received Packets**

The following diagrams describe the format for how received packets are placed into memory by the local DMA channel. These modes are selected in the Data Configuration Register. This format used with general 8-bit CPUs.

AD7	AD0
Receive Status	
Next Packet Pointer	
Receive Byte Count 0	
Receive Byte Count 1	
Byte 0	
Byte 1	

BOS = 0, WTS = 0 in Data Configuration Register.

**Figure 7.8 Storage Format**

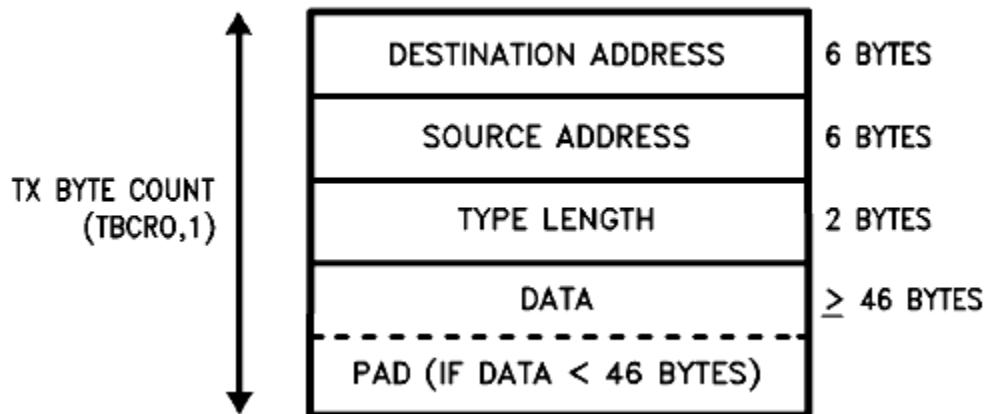


## **7.5 Packet Transmission**

The Local DMA is also used during transmission of a packet. Three registers control the DMA transfer during transmission, a Transmit Page Start Address Register (TPSR) and the Transmit Byte Count Registers (TBCR0,1). When the NIC receives a command to transmit the packet pointed to by these registers, buffer memory data will be moved into the FIFO as required during transmission. The NIC will generate and append the preamble, synch and CRC fields.

### **7.5.1 Transmit Packet Assembly**

The NIC requires a contiguous assembled packet in a specific format. The transmit byte count includes the Destination Address, Source Address, Length Field and Data. It does not include preamble and CRC. When transmitting data smaller than 46 bytes, the packet must be padded to a minimum size of 64 bytes.



**Figure 7.9 General Transmit Packet Format**

### **7.5.2 Transmission**

Prior to transmission, the TPSR (Transmit Page Start Register) and TBCR0, TBCR1 (Transmit Byte Count Registers) must be initialized. To initiate transmission of the packet the TXP bit in the Command Register is set. The Transmit Status Register (TSR)

is cleared and the NIC begins to pre-fetch transmit data from memory (unless the NIC is currently receiving). If the interframe gap has timed out the NIC will begin transmission.

### **7.5.3 Conditions Required To Begin Transmission**

In order to transmit a packet, the following three conditions must be met:

1. The inter-frame Gap Timer has timed out the first 6.4 ms of the inter-frame Gap
2. At least one byte has entered the FIFO. (This indicates that the burst transfer has been started)
3. If the NIC had collided, the back off timer has expired.

In typical systems, the NIC has already pre-fetched the first burst of bytes before 6.4 ms timer expires. The time during which NIC transmits preamble can also be used to load the FIFO.

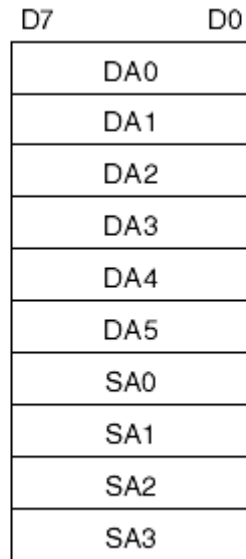
**Note:** If carrier sense is asserted before a byte has been loaded into the FIFO, the NIC will become a receiver.

### **7.5.4 Collision Recovery**

During transmission, the Buffer Management logic monitors the transmit circuitry to determine if a collision has occurred. If a collision is detected, the Buffer Management logic will reset the FIFO and restore the Transmit DMA pointers for retransmission of the packet. The COL bit will be set in the TSR and the NCR (Number of Collisions Register) will be incremented. If 15 retransmissions each result in a collision the transmission will be aborted and the ABT bit in the TSR will be set.

### **7.5.5 Transmit Packet Assembly Format**

The following diagrams describe the format for how packets must be assembled prior to transmission for different byte ordering schemes. The various formats are selected in the Data Configuration Register. This format is used with general 8-bit CPUs.



BOS = 0, WTS = 0 in Data Configuration Register.

**Figure 7.10 Transmit Packet Assembly Format**

## **7.6 Remote DMA**

The Remote DMA channel is used to both assemble packets for transmission, and to remove received packets from the Receive Buffer Ring. It may also be used as a general purpose slave DMA channel for moving blocks of data or commands between host memory and local buffer memory. There are three modes of operation, Remote Write, Remote Read, or Send Packet. Two register pairs are used to control the Remote DMA, a Remote Start Address (RSAR0, RSAR1) and a Remote Byte Count (RBCR0, RBCR1) register pair. The Start Address Register pair points to the beginning of the block to be moved while the Byte Count Register pair is used to indicate the number of bytes to be transferred. Full handshake logic is provided to move data between local buffer memory and a bidirectional I/O port.

### **7.6.1 Remote Write**

A Remote Write transfer is used to move a block of data from the host into local buffer memory. The Remote DMA will read data from the I/O port and sequentially write it to local buffer memory beginning at the Remote Start Address. The DMA Address will be

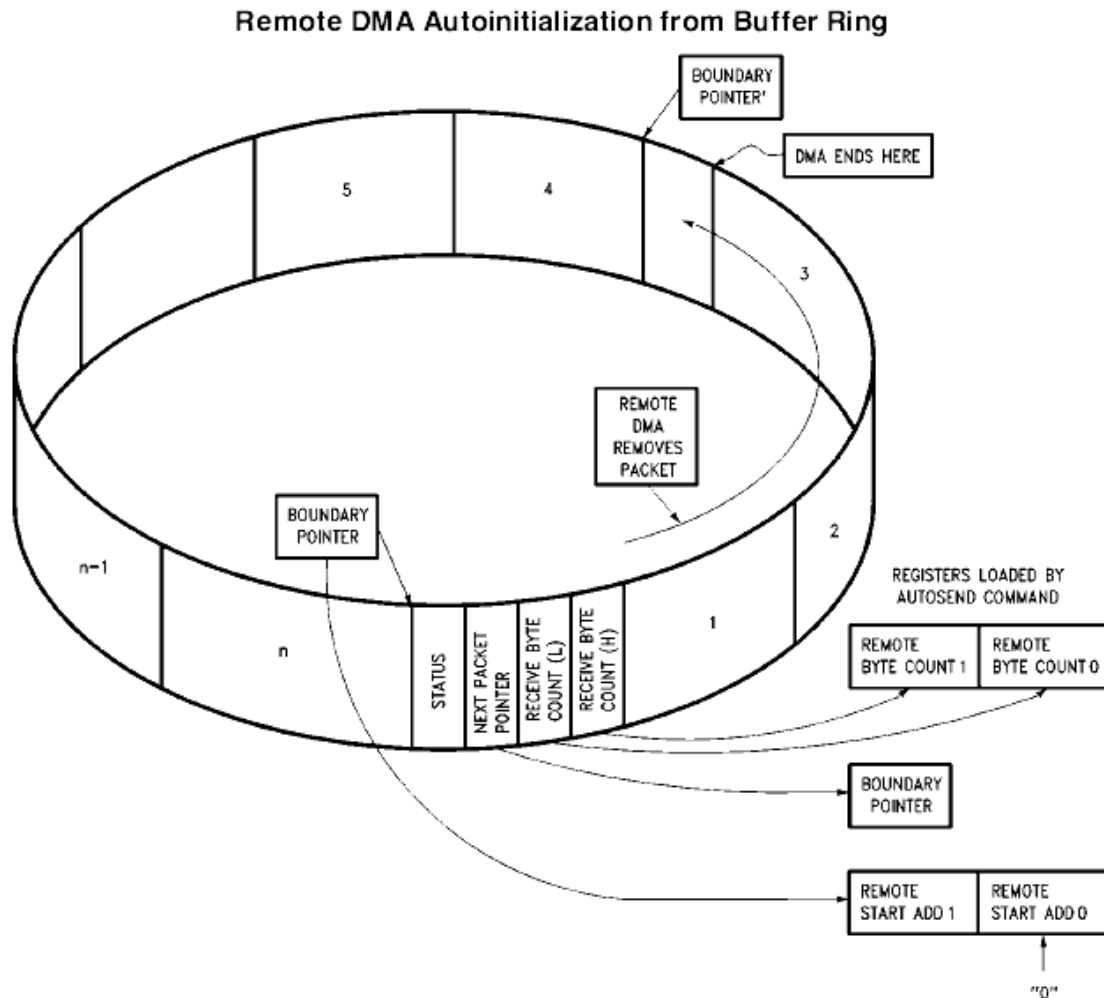
incremented and the Byte Counter will be decremented after each transfer. The DMA is terminated when the Remote Byte Count Register reaches a count of zero.

### **7.6.2 Remote Read**

A Remote Read transfer is used to move a block of data from local buffer memory to the host. The Remote DMA will sequentially read data from the local buffer memory, beginning at the Remote Start Address, and write data to the I/O port. The DMA Address will be incremented and the Byte Counter will be decremented after each transfer. The DMA is terminated when the Remote Byte Count Register reaches zero.

### **7.6.3 Send Packet Command**

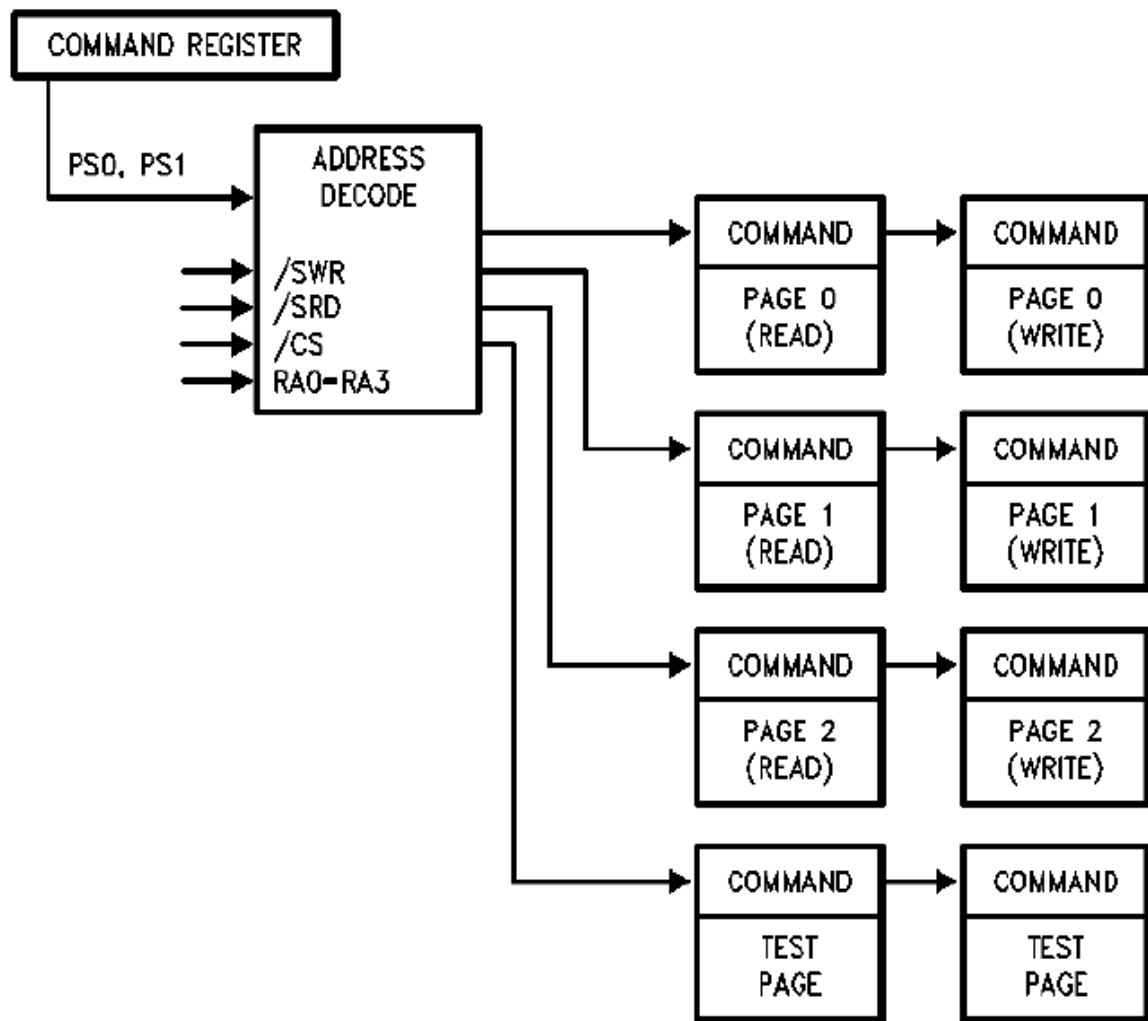
The Remote DMA channel can be automatically initialized to transfer a single packet from the Receive Buffer Ring. The CPU begins this transfer by issuing a 'Send Packet' Command. The DMA will be initialized to the value of the Boundary Pointer Register and the Remote Byte Count Register pair (RBCR0, RBCR1) will be initialized to the value of the Receive Byte Count fields found in the Buffer Header of each packet. After the data is transferred, the Boundary Pointer is advanced to allow the buffers to be used for new receive packets. The Remote Read will terminate when the Byte Count equals zero. The Remote DMA is then prepared to read the next packet from the Receive Buffer Ring. If the DMA pointer crosses the Page Stop Register, it is reset to the Page Start Address. This allows the Remote DMA to remove packets that have wrapped around to the top of the Receive Buffer Ring. In order for the NIC to correctly execute the Send Packet Command, the upper Remote Byte Count Register (RBCR1) must first be loaded with 0FH.



**Figure 7.11 Remote DMA Auto Initialization**

### **7.7 Internal Registers**

All registers are 8-bit wide and mapped into two pages which are selected in the Command Register (PS0, PS1). Pins RA0-RA3 are used to address registers within each page as shown in Figure 3.21. Page0 registers are those registers which are commonly accessed during NIC operation while page1 registers are used primarily for initialization. The registers are partitioned to avoid having to perform two write/read cycles to access commonly used registers.



**Figure 7.12 Register Address Mapping**

**Table 7.1 Register Address Assignments**

**Page 0 Address Assignments (PS1 = 0, PS0 = 0)**

RA0-RA3	RD	WR
00H	Command (CR)	Command (CR)
01H	Current Local DMA Address 0 (CLDA0)	Page Start Register (PSTART)
02H	Current Local DMA Address 1 (CLDA1)	Page Stop Register (PSTOP)
03H	Boundary Pointer (BNRY)	Boundary Pointer (BNRY)
04H	Transmit Status Register (TSR)	Transmit Page Start Address (TPSR)
05H	Number of Collisions Register (NCR)	Transmit Byte Count Register 0 (TBCR0)
06H	FIFO (FIFO)	Transmit Byte Count Register 1 (TBCR1)
07H	Interrupt Status Register (ISR)	Interrupt Status Register (ISR)
08H	Current Remote DMA Address 0 (CRDA0)	Remote Start Address Register 0 (RSAR0)
09H	Current Remote DMA Address 1 (CRDA1)	Remote Start Address Register 1 (RSAR1)
0AH	Reserved	Remote Byte Count Register 0 (RBCR0)
0BH	Reserved	Remote Byte Count Register 1 (RBCR1)
0CH	Receive Status Register (RSR)	Receive Configuration Register (RCR)
0DH	Tally Counter 0 (Frame Alignment Errors) (CNTR0)	Transmit Configuration Register (TCR)
0EH	Tally Counter 1 (CRC Errors) (CNTR1)	Data Configuration Register (DCR)
0FH	Tally Counter 2 (Missed Packet Errors) (CNTR2)	Interrupt Mask Register (IMR)

**Page 1 Address Assignments (PS1 = 0, PS0 = 1)**

RA0-RA3	RD	WR
00H	Command (CR)	Command (CR)
01H	Physical Address Register 0 (PAR0)	Physical Address Register 0 (PAR0)
02H	Physical Address Register 1 (PAR1)	Physical Address Register 1 (PAR1)
03H	Physical Address Register 2 (PAR2)	Physical Address Register 2 (PAR2)
04H	Physical Address Register 3 (PAR3)	Physical Address Register 3 (PAR3)
05H	Physical Address Register 4 (PAR4)	Physical Address Register 4 (PAR4)
06H	Physical Address Register 5 (PAR5)	Physical Address Register 5 (PAR5)
07H	Current Page Register (CURR)	Current Page Register (CURR)
08H	Multicast Address Register 0 (MAR0)	Multicast Address Register 0 (MAR0)
09H	Multicast Address Register 1 (MAR1)	Multicast Address Register 1 (MAR1)
0AH	Multicast Address Register 2 (MAR2)	Multicast Address Register 2 (MAR2)
0BH	Multicast Address Register 3 (MAR3)	Multicast Address Register 3 (MAR3)
0CH	Multicast Address Register 4 (MAR4)	Multicast Address Register 4 (MAR4)
0DH	Multicast Address Register 5 (MAR5)	Multicast Address Register 5 (MAR5)
0EH	Multicast Address Register 6 (MAR6)	Multicast Address Register 6 (MAR6)
0FH	Multicast Address Register 7 (MAR7)	Multicast Address Register 7 (MAR7)

## **7.8 Register Descriptions**

### **7.8.1 Command Register (CR) 00h (Read/Write)**

The Command Register is used to initiate transmissions, enable or disable Remote DMA operations and to select register pages. To issue a command the microprocessor sets the corresponding bit(s) (RD2, RD1, RD0, TXP). Further commands may be overlapped, but with the following rules:

- 1) If a transmit command overlaps with a remote DMA operation, bits RD0, RD1, and RD2 must be maintained for the remote DMA command when setting the TXP bit. Note if a remote DMA command is re-issued when giving the transmit command, the DMA will complete immediately if the remote byte count register have not been reinitialized.
- 2) If a remote DMA operation overlaps a transmission, RD0, RD1, and RD2 may be written with the desired values and a 0 written to the TXP bit. Writing a 0 to this bit has no effect.
- 3) A remote write DMA may not overlap remote read operation or visa versa. Either of these operations must either complete or be aborted before the other operation may start.

Bits PS1, PS0, RD2, and STP may be set any time.

7	6	5	4	3	2	1	0
PS1	PS0	RD2	RD1	RD0	TXP	STA	STP



**Table 7.2 Command Register Description**

Bit	Symbol	Description																								
D0	STP	<b>STOP:</b> Software reset command, takes the controller offline, no packets will be received or transmitted. Any reception or transmission in progress will continue to completion before entering the reset state. To exit this state, the STP bit must be reset and the STA bit must be set high. To perform a software reset, this bit should be set high. The software reset has executed only when indicated by the RST bit in the ISR being set to a 1. <b>STP powers up high.</b> <b>Note:</b> If the NIC has previously been in start mode and the STP is set, both the STP and STA bits will remain set.																								
D1	STA	<b>START:</b> This bit is used to activate the NIC after either power up, or when the NIC has been placed in a reset mode by software command or error. <b>STA powers up low.</b>																								
D2	TXP	<b>TRANSMIT PACKET:</b> This bit must be set to initiate transmission of a packet. TXP is internally reset either after the transmission is completed or aborted. This bit should be set only after the Transmit Byte Count and Transmit Page Start registers have been programmed. <b>Note:</b> Before the transmit command is given, the STA bit must be set and the STP bit reset.																								
D3, D4, D5	RD0, RD1, RD2	<b>REMOTE DMA COMMAND:</b> These three encoded bits control operation of the Remote DMA channel. RD2 can be set to abort any Remote DMA command in progress. The Remote Byte Count Registers should be cleared when a Remote DMA has been aborted. The Remote Start Addresses are not restored to the starting address if the Remote DMA is aborted. <table><tr><td>RD2</td><td>RD1</td><td>RD0</td><td></td></tr><tr><td>0</td><td>0</td><td>0</td><td>Not Allowed</td></tr><tr><td>0</td><td>0</td><td>1</td><td>Remote Read</td></tr><tr><td>0</td><td>1</td><td>0</td><td>Remote Write (Note 2)</td></tr><tr><td>0</td><td>1</td><td>1</td><td>Send Packet</td></tr><tr><td>1</td><td>X</td><td>X</td><td>Abort/Complete Remote DMA (Note 1)</td></tr></table> <b>Note 1:</b> If a remote DMA operation is aborted and the remote byte count has not decremented to zero, PRQ (pin 29, DIP) will remain high. A read acknowledge (RACK) on a write acknowledge (WACK) will reset PRQ low. <b>Note 2:</b> For proper operation of the Remote Write DMA, there are two steps which must be performed before using the Remote Write DMA. The steps are as follows: <ul style="list-style-type: none"><li>i) Write a non-zero value into RBCR0.</li><li>ii) Set bits RD2, RD1, RD0 to 0, 0, 1.</li><li>iii) Set RBCR0, 1 and RSAR0, 1</li><li>iv) Issue the Remote Write DMA Command (RD2, RD1, RD0 = 0, 1, 0)</li></ul>	RD2	RD1	RD0		0	0	0	Not Allowed	0	0	1	Remote Read	0	1	0	Remote Write (Note 2)	0	1	1	Send Packet	1	X	X	Abort/Complete Remote DMA (Note 1)
RD2	RD1	RD0																								
0	0	0	Not Allowed																							
0	0	1	Remote Read																							
0	1	0	Remote Write (Note 2)																							
0	1	1	Send Packet																							
1	X	X	Abort/Complete Remote DMA (Note 1)																							
D6, D7	PS0, PS1	<b>PAGE SELECT:</b> These two encoded bits select which register page is to be accessed with addresses RA0–3. <table><tr><td>PS1</td><td>PS0</td><td></td></tr><tr><td>0</td><td>0</td><td>Register Page 0</td></tr><tr><td>0</td><td>1</td><td>Register Page 1</td></tr><tr><td>1</td><td>0</td><td>Register Page 2</td></tr><tr><td>1</td><td>1</td><td>Reserved</td></tr></table>	PS1	PS0		0	0	Register Page 0	0	1	Register Page 1	1	0	Register Page 2	1	1	Reserved									
PS1	PS0																									
0	0	Register Page 0																								
0	1	Register Page 1																								
1	0	Register Page 2																								
1	1	Reserved																								

### **7.8.2 Interrupt Status Register (ISR) 07h (Read/Write)**

This register is accessed by the host processor to determine the cause of an interrupt. Any interrupt can be masked in the Interrupt Mask Register (IMR). Individual interrupt bits are cleared by writing a 1 into the corresponding bit of the ISR. The INT signal is active as long as any unmasked signal is set, and will not go low until all unmasked bits in this register have been cleared. The ISR must be cleared after power up by writing it with all 1's.

7	6	5	4	3	2	1	0
RST	RDC	CNT	OWW	TXE	RXE	PTX	PRX

**Table 7.3 Interrupt Status Register Description**

Bit	Symbol	Description
D0	PRX	<b>PACKET RECEIVED:</b> Indicates packet received with no errors.
D1	PTX	<b>PACKET TRANSMITTED:</b> Indicates packet transmitted with no errors.
D2	RXE	<b>RECEIVE ERROR:</b> Indicates that a packet was received with one or more of the following errors: —CRC Error —Frame Alignment Error —FIFO Overrun —Missed Packet
D3	TXE	<b>TRANSMIT ERROR:</b> Set when packet transmitted with one or more of the following errors: —Excessive Collisions —FIFO Underrun
D4	OWW	<b>OVERWRITE WARNING:</b> Set when receive buffer ring storage resources have been exhausted. (Local DMA has reached Boundary Pointer).
D5	CNT	<b>COUNTER OVERFLOW:</b> Set when MSB of one or more of the Network Tally Counters has been set.
D6	RDC	<b>REMOTE DMA COMPLETE:</b> Set when Remote DMA operation has been completed.
D7	RST	<b>RESET STATUS:</b> Set when NIC enters reset state and cleared when a Start Command is issued to the CR. This bit is also set when a Receive Buffer Ring overflow occurs and is cleared when one or more packets have been removed from the ring. Writing to this bit has no effect. <b>NOTE:</b> This bit does not generate an interrupt, it is merely a status indicator.

### 7.8.3 Interrupt Mask Register (IMR) 0fh (Write)

The Interrupt Mask Register is used to mask interrupts. Each interrupt mask bit corresponds to a bit in the Interrupt Status Register (ISR). If an interrupt mask bit is set an interrupt will be issued whenever the corresponding bit in the ISR is set. If any bit in the IMR is set low, an interrupt will not occur when the bit in the ISR is set. The IMR powers up all zeroes.

7	6	5	4	3	2	1	0
—	RDCE	CNTE	OVWE	TXEE	RXEE	PTXE	PRXE

**Table 7.4 Interrupt Mask Register Description**

Bit	Symbol	Description
D0	PRXE	<b>PACKET RECEIVED INTERRUPT ENABLE</b> 0: Interrupt Disabled 1: Enables Interrupt when packet received.
D1	PTXE	<b>PACKET TRANSMITTED INTERRUPT ENABLE</b> 0: Interrupt Disabled 1: Enables Interrupt when packet is transmitted.
D2	RXEE	<b>RECEIVE ERROR INTERRUPT ENABLE</b> 0: Interrupt Disabled 1: Enables Interrupt when packet received with error.
D3	TXEE	<b>TRANSMIT ERROR INTERRUPT ENABLE</b> 0: Interrupt Disabled 1: Enables Interrupt when packet transmission results in error.
D4	OVWE	<b>OVERWRITE WARNING INTERRUPT ENABLE</b> 0: Interrupt Disabled 1: Enables Interrupt when Buffer Management Logic lacks sufficient buffers to store incoming packet.
D5	CNTE	<b>COUNTER OVERFLOW INTERRUPT ENABLE</b> 0: Interrupt Disabled 1: Enables Interrupt when MSB of one or more of the Network Statistics counters has been set.
D6	RDCE	<b>DMA COMPLETE INTERRUPT ENABLE</b> 0: Interrupt Disabled 1: Enables Interrupt when Remote DMA transfer has been completed.
D7	reserved	reserved

### 7.8.4 Data Configuration Register (DCR) 0EH (Write)

This Register is used to program the NIC for 8- or 16-bit memory interface, select byte ordering in 16-bit applications and establish FIFO thresholds. The DCR must be initialized prior to loading the Remote Byte Count Registers. LAS is set on power up.

7	6	5	4	3	2	1	0
—	FT1	FT0	ARM	LS	LAS	BOS	WTS

**Table 7.5 Data Configuration Register Description**

Bit	Symbol	Description																				
D0	WTS	<b>WORD TRANSFER SELECT</b> 0: Selects byte-wide DMA transfers 1: Selects word-wide DMA transfers  ; WTS establishes byte or word transfers for both Remote and Local DMA transfers <b>Note:</b> When word-wide mode is selected, up to 32k words are addressable; A0 remains low.																				
D1	BOS	<b>BYTE ORDER SELECT</b> 0: MS byte placed on AD15–AD8 and LS byte on AD7–AD0. (32000, 8086) 1: MS byte placed on AD7–AD0 and LS byte on AD15–AD8. (68000)  ; Ignored when WTS is low																				
D2	LAS	<b>LONG ADDRESS SELECT</b> 0: Dual 16-bit DMA mode 1: Single 32-bit DMA mode  ; When LAS is high, the contents of the Remote DMA registers RSAR0,1 are issued as A16–A31 Power up high.																				
D3	LS	<b>LOOPBACK SELECT</b> 0: Loopback mode selected. Bits D1, D2 of the TCR must also be programmed for Loopback operation. 1: Normal Operation.																				
D4	AR	<b>AUTO-INITIALIZE REMOTE</b> 0: Send Command not executed, all packets removed from Buffer Ring under program control. 1: Send Command executed, Remote DMA auto-initialized to remove packets from Buffer Ring. <b>Note:</b> Send Command cannot be used with 68000 type processors.																				
D5, D6	FT0, FT1	<b>FIFO THRESHHOLD SELECT:</b> Encoded FIFO threshold. Establishes point at which bus is requested when filling or emptying the FIFO. During reception, the FIFO threshold indicates the number of bytes (or words) the FIFO has filled serially from the network before bus request (BREQ) is asserted. <b>Note:</b> FIFO threshold setting determines the DMA burst length. <b>RECEIVE THRESHOLDS</b> <table><tr><td>FT1</td><td>FT0</td><td>Word Wide</td><td>Byte Wide</td></tr><tr><td>0</td><td>0</td><td>1 Word</td><td>2 Bytes</td></tr><tr><td>0</td><td>1</td><td>2 Words</td><td>4 Bytes</td></tr><tr><td>1</td><td>0</td><td>4 Words</td><td>8 Bytes</td></tr><tr><td>1</td><td>1</td><td>6 Words</td><td>12 Bytes</td></tr></table> During transmission, the FIFO threshold indicates the numer of bytes (or words) the FIFO has filled from the Local DMA before BREQ is asserted. Thus, the transmission threshold is 16 bytes less the receive threshold.	FT1	FT0	Word Wide	Byte Wide	0	0	1 Word	2 Bytes	0	1	2 Words	4 Bytes	1	0	4 Words	8 Bytes	1	1	6 Words	12 Bytes
FT1	FT0	Word Wide	Byte Wide																			
0	0	1 Word	2 Bytes																			
0	1	2 Words	4 Bytes																			
1	0	4 Words	8 Bytes																			
1	1	6 Words	12 Bytes																			

### 7.8.5 Transmit Configuration Register (TCR) 0dh (Write)

The transmit configuration establishes the actions of the transmitter section of the NIC during transmission of a packet on the network. LB1 and LB0 which select loopback mode power up as 0.

7	6	5	4	3	2	1	0
—	—	—	OFST	ATD	LB1	LB0	CRC

**Table 7.6 Transmit Configuration Register Description**

Bit	Symbol	Description																				
D0	CRC	<b>INHIBIT CRC</b> 0: CRC appended by transmitter 1: CRC inhibited by transmitter ; In loopback mode CRC can be enabled or disabled to test the CRC logic.																				
D1, D2	LB0, LB1	<b>ENCODED LOOPBACK CONTROL:</b> These encoded configuration bits set the type of loopback that is to be performed. Note that loopback in mode 2 sets the LPBK pin high, this places the SNI in loopback mode and that D3 of the DCR must be set to zero for loopback operation.  <table><tr><td></td><td>LB1</td><td>LB0</td><td></td></tr><tr><td>Mode 0</td><td>0</td><td>0</td><td>Normal Operation (LPBK = 0)</td></tr><tr><td>Mode 1</td><td>0</td><td>1</td><td>Internal Loopback (LPBK = 0)</td></tr><tr><td>Mode 2</td><td>1</td><td>0</td><td>External Loopback (LPBK = 1)</td></tr><tr><td>Mode 3</td><td>1</td><td>1</td><td>External Loopback (LPBK = 0)</td></tr></table>		LB1	LB0		Mode 0	0	0	Normal Operation (LPBK = 0)	Mode 1	0	1	Internal Loopback (LPBK = 0)	Mode 2	1	0	External Loopback (LPBK = 1)	Mode 3	1	1	External Loopback (LPBK = 0)
	LB1	LB0																				
Mode 0	0	0	Normal Operation (LPBK = 0)																			
Mode 1	0	1	Internal Loopback (LPBK = 0)																			
Mode 2	1	0	External Loopback (LPBK = 1)																			
Mode 3	1	1	External Loopback (LPBK = 0)																			
D3	ATD	<b>AUTO TRANSMIT DISABLE:</b> This bit allows another station to disable the NIC's transmitter by transmission of a particular multicast packet. The transmitter can be re-enabled by resetting this bit or by reception of a second particular multicast packet. 0: Normal Operation 1: Reception of multicast address hashing to bit 62 disables transmitter, reception of multicast address hashing to bit 63 enables transmitter.																				
D4	OFST	<b>COLLISION OFFSET ENABLE:</b> This bit modifies the backoff algorithm to allow prioritization of nodes. 0: Backoff Logic implements normal algorithm. 1: Forces Backoff algorithm modification to 0 to $2^{\min(3+n,10)}$ slot times for first three collisions, then follows standard backoff. (For first three collisions station has higher average backoff delay making a low priority mode.)																				
D5	reserved	reserved																				
D6	reserved	reserved																				
D7	reserved	reserved																				

### 7.8.6 Transmit Status Register (TSR) 04h (Read)

This register records events that occur on the media during transmission of a packet. It is cleared when the next transmission is initiated by the host. All bits remain low unless the event that corresponds to a particular bit occurs during transmission. Each transmission should be followed by a read of this register. The contents of this register are not specified until after the first transmission.

7	6	5	4	3	2	1	0
OWC	CDH	FU	CRS	ABT	COL	—	PTX

**Table 7.7 Transmit Status Register Description**

Bit	Symbol	Description
D0	PTX	<b>PACKET TRANSMITTED:</b> Indicates transmission without error. (No excessive collisions or FIFO underrun) (ABT = "0", FU = "0").
D1	reserved	reserved
D2	COL	<b>TRANSMIT COLLIDED:</b> Indicates that the transmission collided at least once with another station on the network. The number of collisions is recorded in the Number of Collisions Registers (NCR).
D3	ABT	<b>TRANSMIT ABORTED:</b> Indicates the NIC aborted transmission because of excessive collisions. (Total number of transmissions including original transmission attempt equals 16).
D4	CRS	<b>CARRIER SENSE LOST:</b> This bit is set when carrier is lost during transmission of the packet. Carrier Sense is monitored from the end of Preamble/Synch until TXEN is dropped. Transmission is not aborted on loss of carrier.
D5	FU	<b>FIFO UNDERRUN:</b> If the NIC cannot gain access of the bus before the FIFO empties, this bit is set. Transmission of the packet will be aborted.
D6	CDH	<b>CD HEARTBEAT:</b> Failure of the transceiver to transmit a collision signal after transmission of a packet will set this bit. The Collision Detect (CD) heartbeat signal must commence during the first 6.4 $\mu$ s of the Interframe Gap following a transmission. In certain collisions, the CD Heartbeat bit will be set even though the transceiver is not performing the CD heartbeat test.
D7	OWC	<b>OUT OF WINDOW COLLISION:</b> Indicates that a collision occurred after a slot time (51.2 $\mu$ s). Transmissions rescheduled as in normal collisions.

### 7.8.7 Receive Configuration Register (RCR) 0ch (Write)

This register determines operation of the NIC during reception of a packet and is used to program what types of packets to accept.

7	6	5	4	3	2	1	0
—	—	MON	PRO	AM	AB	AR	SEP

**Table 7.8 Receive Configuration Register Description**

Bit	Symbol	Description
D0	SEP	<b>SAVE ERRORED PACKETS</b> 0: Packets with receive errors are rejected. 1: Packets with receive errors are accepted. Receive errors are CRC and Frame Alignment errors.
D1	AR	<b>ACCEPT RUNT PACKETS:</b> This bit allows the receiver to accept packets that are smaller than 64 bytes. The packet must be at least 8 bytes long to be accepted as a runt. 0: Packets with fewer than 64 bytes rejected. 1: Packets with fewer than 64 bytes accepted.
D2	AB	<b>ACCEPT BROADCAST:</b> Enables the receiver to accept a packet with an all 1's destination address. 0: Packets with broadcast destination address rejected. 1: Packets with broadcast destination address accepted.
D3	AM	<b>ACCEPT MULTICAST:</b> Enables the receiver to accept a packet with a multicast address, all multicast addresses must pass the hashing array. 0: Packets with multicast destination address not checked. 1: Packets with multicast destination address checked.
D4	PRO	<b>PROMISCUOUS PHYSICAL:</b> Enables the receiver to accept all packets with a physical address. 0: Physical address of node must match the station address programmed in PAR0–PAR5. 1: All packets with physical addresses accepted.
D5	MON	<b>MONITOR MODE:</b> Enables the receiver to check addresses and CRC on incoming packets without buffering to memory. The Missed Packet Tally counter will be incremented for each recognized packet. 0: Packets buffered to memory. 1: Packets checked for address match, good CRC and Frame Alignment but not buffered to memory.
D6	reserved	reserved
D7	reserved	reserved

### 7.8.8 Receive Status Register (RSR) 0CH (Read)

This register records status of the received packet, including information on errors and the type of address match, either physical or multicast. The contents of this register are written to buffer memory by the DMA after reception of a good packet. If packets with errors are to be saved the receive status is written to memory at the head of the erroneous packet if an erroneous packet is received. If packets with errors are to be rejected the RSR will not be written to memory. The contents will be cleared when the next packet arrives. CRC errors, Frame Alignment errors and missed packets are counted internally by the NIC which relinquishes the Host from reading the RSR in real time to record errors for Network Management Functions. The contents of this register are not specified until after the first reception.

7	6	5	4	3	2	1	0
DFR	DIS	PHY	MPA	FO	FAE	CRC	PRX

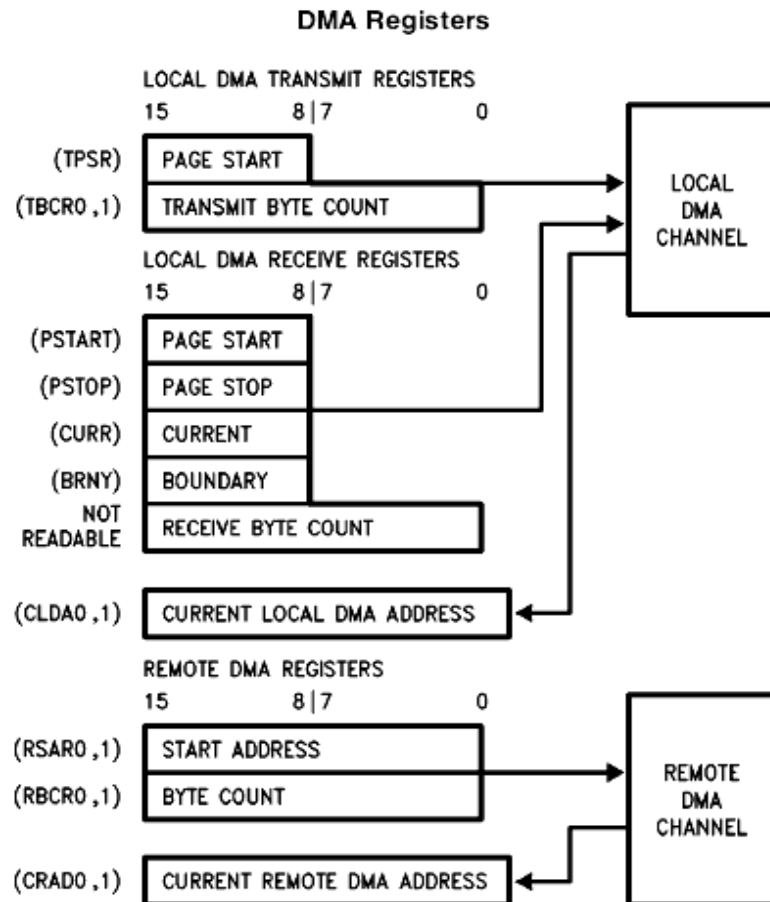
**Table 7.9 Receive Status Register Description**

Bit	Symbol	Description
D0	PRX	<b>PACKET RECEIVED INTACT:</b> Indicates packet received without error. (Bits CRC, FAE, FO, and MPA are zero for the received packet.)
D1	CRC	<b>CRC ERROR:</b> Indicates packet received with CRC error. Increments Tally Counter (CNTR1). This bit will also be set for Frame Alignment errors.
D2	FAE	<b>FRAME ALIGNMENT ERROR:</b> Indicates that the incoming packet did not end on a byte boundary and the CRC did not match at last byte boundary. Increments Tally Counter (CNTR0).
D3	FO	<b>FIFO OVERRUN:</b> This bit is set when the FIFO is not serviced causing overflow during reception. Reception of the packet will be aborted.
D4	MPA	<b>MISSED PACKET:</b> Set when packet intended for node cannot be accepted by NIC because of a lack of receive buffers or if the controller is in monitor mode and did not buffer the packet to memory. Increments Tally Counter (CNTR2).
D5	PHY	<b>PHYSICAL/MULTICAST ADDRESS:</b> Indicates whether received packet had a physical or multicast address type. 0: Physical Address Match 1: Multicast/Broadcast Address Match
D6	DIS	<b>RECEIVER DISABLED:</b> Set when receiver disabled by entering Monitor mode. Reset when receiver is re-enabled when exiting Monitor mode.
D7	DFR	<b>DEFERRING:</b> Set when CRS or COL inputs are active. If the transceiver has asserted the CD line as a result of the jabber, this bit will stay set indicating the jabber condition.



## 7.9 DMA REGISTERS

The DMA Registers are partitioned into three groups; Transmit, Receive and Remote DMA Registers as shown in the Figure below. The Transmit registers are used to initialize the Local DMA Channel for transmission of packets while the Receive Registers are used to initialize the Local DMA Channel for packet Reception. The Page Stop, Page Start, Current and Boundary Registers are used by the Buffer Management Logic to supervise the Receive Buffer Ring. The Remote DMA Registers are used to initialize the Remote DMA.



**Figure 7.13 DMA Registers**

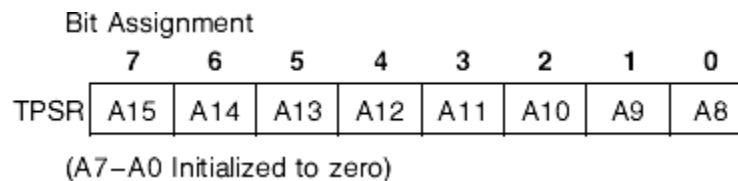
**Note:** The 16-bit Transmit Byte Count Register is broken into two 8-bit registers, TBCR0 and TBCR1. Also TPSR, PSTART, PSTOP, CURR and BNRY only check or control the upper 8 bits of address information on the bus.

## **7.10 Transmit DMA Registers**

### **7.10.1 Transmit Page Start Register (TPSR)**

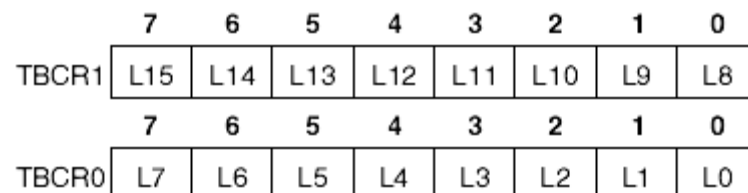
This register points to the assembled packet to be transmitted. Only the eight higher order addresses are specified since all transmit packets are assembled on 256-byte page boundaries. The values placed in bits D7-D0 will be used to initialize the higher order address (A8-A15) of the Local DMA for transmission.

The lower order bits (A7-A0) are initialized to zero.



### **7.10.2 Transmit Byte Count Register 0,1 (TBCR0, TBCR1)**

These two registers indicate the length of the packet to be transmitted in bytes. The count must include the number of bytes in the source, destination, length and data fields. The maximum number of transmit bytes allowed is 64k bytes. The NIC will not truncate transmissions longer than 1500 bytes. The bit assignment is shown below:



## **7.11 Local DMA Receive Registers**

### **7.11.1 Page Start Stop Registers (PSTART, PSTOP)**

The Page Start and Page Stop Registers program the starting and stopping address of the Receive Buffer Ring. Since the NIC uses fixed 256-byte buffers aligned on page boundaries only the upper eight bits of the start and stop address are specified.

	7	6	5	4	3	2	1	0
PSTART, PSTOP	A15	A14	A13	A12	A11	A10	A9	A8

### **7.11.2 Boundary (BNRY) Register**

This register is used to prevent overflow of the Receive Buffer Ring. Buffer management compares the contents of this register to the next buffer address when linking buffers together. If the contents of this register match the next buffer address the Local DMA operation is aborted.

	7	6	5	4	3	2	1	0
BNRY	A15	A14	A13	A12	A11	A10	A9	A8

### **7.11.3 Current Page Register (CURR)**

This register is used internally by the Buffer Management Logic as a backup register for reception. CURR contains the address of the first buffer to be used for a packet reception and is used to restore DMA pointers in the event of receive errors. This register is initialized to the same value as PSTART and should not be written to again unless the controller is Reset.

	7	6	5	4	3	2	1	0
CURR	A15	A14	A13	A12	A11	A10	A9	A8

#### **7.11.4 Current Local DMA Register 0,1 (CLDA0,1)**

These two registers can be accessed to determine the current Local DMA Address.

	7	6	5	4	3	2	1	0
CLDA1	A15	A14	A13	A12	A11	A10	A9	A8
	7	6	5	4	3	2	1	0
CLDA0	A7	A6	A5	A4	A3	A2	A1	A0

### **7.12 Remote DMA Registers**

#### **7.12.1 Remote Start Address Registers (RSAR0,1)**

Remote DMA operations are programmed via the Remote Start Address (RSAR0,1) and Remote Byte Count (RBCR0,1) registers. The Remote Start Address is used to point to the start of the block of data to be transferred and the Remote Byte Count is used to indicate the length of the block (in bytes).

	7	6	5	4	3	2	1	0
RSAR1	A15	A14	A13	A12	A11	A10	A9	A8
	7	6	5	4	3	2	1	0
RSAR0	A7	A6	A5	A4	A3	A2	A1	A0
6.4.3.2 REMOTE BYTE COUNT REGISTERS (RBCR0,1)								
	7	6	5	4	3	2	1	0
RBCR1	BC15	BC14	BC13	BC12	BC11	BC10	BC9	BC8
	7	6	5	4	3	2	1	0
RBCR0	BC7	BC6	BC5	BC4	BC3	BC2	BC1	BC0

**Note:** RSAR0 programs the start address bits A0-A7.

RSAR1 programs the start address bits A8-A15.

Byte Count decremented by two for word transfers and by one for byte transfers.

RBCR0 programs LSB byte count.

RBCR1 programs MSB byte count.

### **7.12.2 Current Remote DMA Address (CRDA0, CRDAL)**

The Current Remote DMA Registers contain the current address of the Remote DMA.

	7	6	5	4	3	2	1	0
CRDAL	A15	A14	A13	A12	A11	A10	A9	A8
	7	6	5	4	3	2	1	0
CRDA0	A7	A6	A5	A4	A3	A2	A1	A0

### **7.13 Physical Address Registers (PAR0-PAR5)**

The physical address registers are used to compare the destination address of incoming packets for rejecting or accepting packets. Comparisons are performed on a byte-wide basis. The bit assignment shown below relates the sequence in PAR0-PAR5 to the bit sequence of the received packet.

	D7	D6	D5	D4	D3	D2	D1	D0
PAR0	DA7	DA6	DA5	DA4	DA3	DA2	DA1	DA0
PAR1	DA15	DA14	DA13	DA12	DA11	DA10	DA9	DA8
PAR2	DA23	DA22	DA21	DA20	DA19	DA18	DA17	DA16
PAR3	DA31	DA30	DA29	DA28	DA27	DA26	DA25	DA24
PAR4	DA39	DA38	DA37	DA36	DA35	DA34	DA33	DA32
PAR5	DA47	DA46	DA45	DA44	DA43	DA42	DA41	DA40
Destination Address								Source
P/S	DA0	DA1	DA2	DA3	.....	DA46	DA47	SA0 ...

Note:

P/S = Preamble, Synch

DA0 = Physical/Multicast Bit

## **7.14 Initialization Procedures**

The NIC must be initialized prior to transmission or reception of packets from the network. Power on reset is applied to the NIC's reset pin. This clears/sets the following bits:

Register	Reset Bits	Set Bits
Command Register (CR)	TXP, STA	RD2, STP
Interrupt Status (ISR)		RST
Interrupt Mask (IMR)	All Bits	
Data Control (DCR)		LAS
Transmit Config. (TCR)	LB1, LB0	

The NIC remains in its reset state until a Start Command is issued. This guarantees that no packets are transmitted or received and that the NIC remains a bus slave until all appropriate internal registers have been programmed. After initialization the STP bit of the command register is reset and packets may be received and transmitted.

### **7.14.1 Initialization Sequence**

The following initialization procedure is mandatory

1. Program Command Register for Page 0 (Command Register = 21H)
2. Initialize Data Configuration Register (DCR)
3. Clear Remote Byte Count Registers (RBCR0, RBCR1)
4. Initialize Receive Configuration Register (RCR)
5. Place the NIC in LOOPBACK mode 1 or 2 (Transmit Configuration Register = 02H or 04H)
6. Initialize Receive Buffer Ring: Boundary Pointer (BNDRY), Page Start (PSTART), and Page Stop (PSTOP)
7. Clear Interrupt Status Register (ISR) by writing 0FFh to it.
8. Initialize Interrupt Mask Register (IMR)
9. Program Command Register for page 1 (Command Register = 61H)
  - I. Initialize Physical Address Registers (PAR0-PAR5)

II. Initialize Multicast Address Registers (MAR0-MAR7)

III. Initialize CURRENT pointer

10. Put NIC in START mode (Command Register = 22H). The local receive DMA is still not active since the NIC is in LOOPBACK.

11. Initialize the Transmit Configuration for the intended value.

The NIC is now ready for transmission and reception. Before receiving packets, the user must specify the location of the Receive Buffer Ring. This is programmed in the Page Start and Page Stop Registers. In addition, the Boundary and Current Page Registers must be initialized to the value of the Page Start Register. These registers will be modified during reception of packets.

## **PART THREE**

### **SOFTWARE DESCRIPTION**

The key to creating a VoIP system is more in its software than in its hardware. As the working goes near to the LAN interface through NIC, the work on software increases whereas the work on hardware decreases. Seeing this, the NIC has got a very simple 16 pins hardware but its software one the other hand consists of many pages.

The VoIP media gateway system consists of two distinct software codes. One of them is the **main microcontroller software** that controls all other activities and works as the brain of the system. It also works as a liaison between the network card and the digital telephone exchange. Moreover, it also buffers, creates, sends and receives the packets over the LAN through the NIC. As a central controlling entity, it detects the HOOK status, sends the appropriate bits to the analog multiplexer and connects/disconnects the ringer voltage to the SLIC to make the phone ring. Also, it checks if any key on the keypad of the telephone is pressed and if so it detects its code and processes for making peer to peer connection.

The second portion is the **ringer microcontroller software** that is a small two page code, just to create various tones for the telephone exchange having varying telephone cadences. These tone and cadence standards were discussed in Chapter 6 already.



## **CHAPTER 8**

# **Main Microcontroller Software**

### **8.1 General Description**

The main microcontroller consists of many subroutines. The “main” subroutine is the start up of all these subroutines. The flow diagram of the main microcontroller software is shown in figure 8.1. The software code is also included in Appendix A for correspondence.

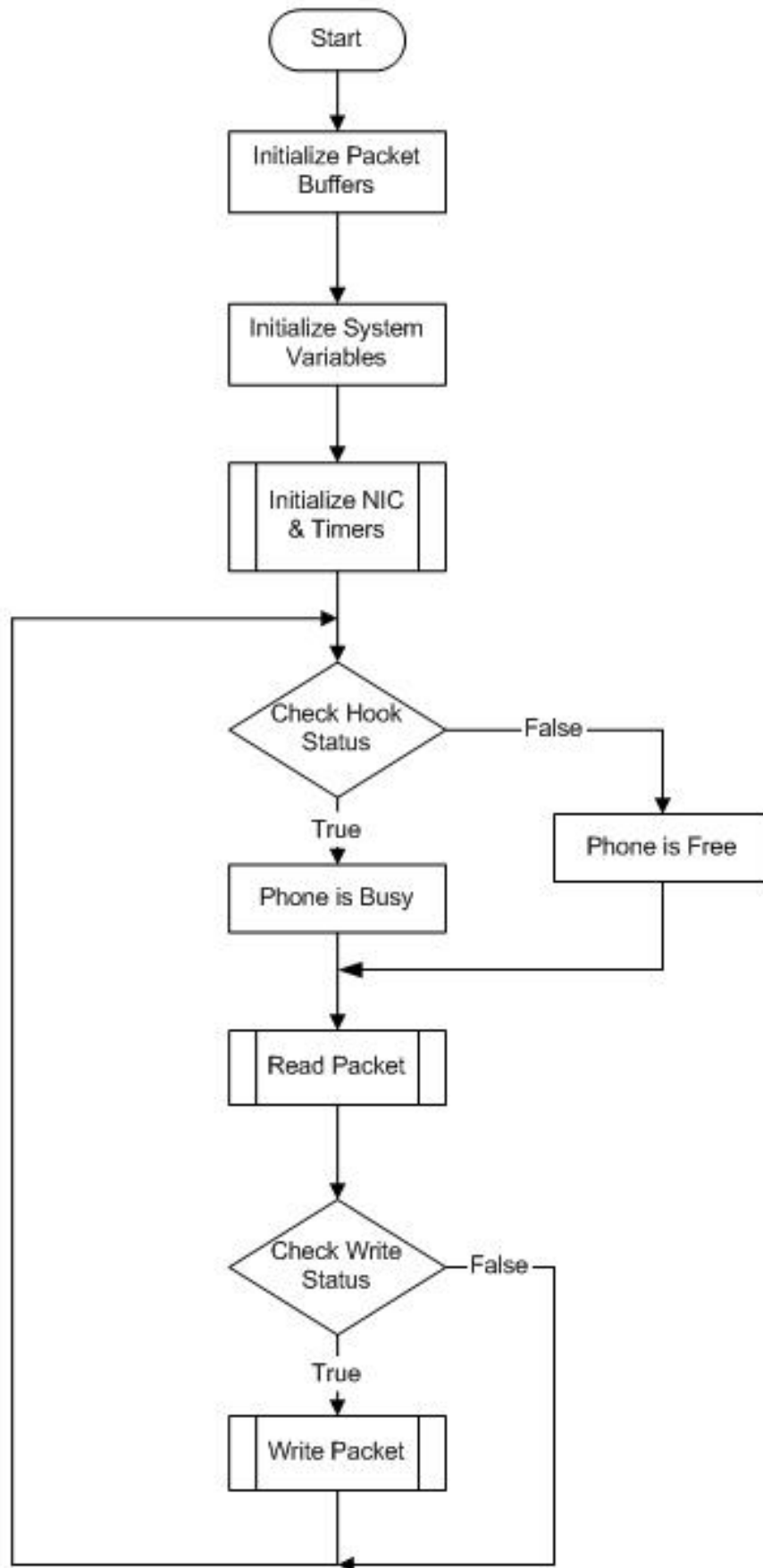
All the NIC registers defined are already explained in the Part II. Overall, there are three distinct tasks with the main microcontroller to perform:

1. Control all of the hardware signaling.
2. Interchange data with the digital telephone exchange.
3. Interchange data with the LAN through NIC interface.

Of these three tasks, the last two only take place when the call is taking place or is going to take place through negotiation process. All the time, the microcontroller has to poll various pins and to check interrupts to confirm that certain events have occurred. Based on these activities, a flow of data occurs in a specific pattern and the events occur in an order.

### **8.2 System Software Flow Chart**

According to software flow, as soon as the microcontroller is started along with the whole system is powered, the initialization of the system occurs. Firstly, the four buffers in the external RAM, 200 bytes each in size are initialized and set to a mid value of 7FH. These buffers are ADC\_Read0, ADC\_Read1, DAC\_Write0 and DAC\_Write1. The idea before initializing these buffers is that they contain garbage value by default as the system is started and will be played on ADC if not deleted. Next, the system variables are reset which are used to control the work of the buffers, packet writing, packet reading, the hook status, DTMF status etc. All these are set to their default values.



**Figure 8.1 System Software Flow Chart**

After the system variables initialized, the NIC initialization routine is started. The NIC is initialized as is stated in the National Semiconductor datasheet or in the chapter 7 of this book. The NIC initialization consists of about 11 steps which deal with the internal registers of the NIC and make it into functioning mode for the rest of the operations. Effectively, after the NIC initialization routine finishes, the NIC is in working mode and ready to send and receive any packets. Moreover, the timers used in uController at the delay of 125usec are also initialized. These timers are used to sample data into memory from the DAC and to play the stored bytes in the memory on the ADC. Also another timer of 25msec is initialized which is used to transfer data from the buffers over the network.

Next in the Flow Chart, the ON\_HOOK variable is polled to check that the receiver is picked or not. If the receiver of the phone is not picked (**ON\_HOOK is True**), then the phone status is set to free or terminating based on the previous state, in which the phone was .i.e., was anyone taking on the phone (busy status) and he terminated or else the phone is already on hook. In the later case (already **free status**), the phone is set to the free status based if the phone is on hook for a long time as the user may be tapping the phone or accidentally tapped the phone which should not instantly cut the phone call. This is just the same as in normal phoning occurs. If a user disconnected the phone after talking, the status is not instantly set to free but the existing packets in the buffer are first sent to the network. The **terminating status** is just to ensure the sending of remaining packets and a proper negotiated end of the call. The packet based call is not just the same as the normal call which is disconnected suddenly. Here certain information has to be exchanged before the ending of the call session.

On the other hand, if the receiver is picked (**ON\_HOOK is False**), the status is checked by the software to see if this phone picking is the result of the phone ringing previously or the user intends to dial a new call. If the phone was ringing and the user picked up the receiver, then the software sets the status variable to Talking, which means that the user is busy. But if the user has picked up the phone to dial a number, software has to uniquely identify this state as the Dialing state and wait for the three key input from the user. The

whole main microcontroller software consists of many sub-routines which have their own functionality. These sub-routines are:

- void NIC\_Write(uchar Address,uchar Data);
- uchar NIC\_Read(uchar Address);
- void NIC\_Hard\_Reset();
- void NIC\_Init();
- void NIC\_Read\_MAC();
- void Packet\_Read();
- void Packet\_Write(uchar Type);
- void Delay(uchar Value);
- void Mux(uchar Value);

The main() subroutine is the starting point of all the activities, and it directly or indirectly calls all other routines. Here in the above mentioned routines, the NIC\_Write and NIC\_Read routines are used to write and read a single byte of data to and from the buses of the NIC. These are related to the hardware used to access the NIC and make the hardware complexities invisible to other routines in the software. The NIC\_Hard\_Reset is called at the start of the system initialization to reset the NIC, after which the NIC\_Init routine is called, completing the initialization of the NIC. The NIC\_Reset routine just toggles Reset pin of the NIC. These two routines are designed just as stated in the datasheet of the D8390 chip of National Semiconductor. The NIC\_Read\_MAC is used in the NIC initialization to retrieve the MAC address from the EPROM of the NIC and to save for future usage. It is also a one time used routine. The Packet\_Read and Packet\_Write routines are the main routines to interact with the other peers in the network. The routines write and read different sort of packets and gets the data out of them. These packets may be ARP, IP or UDP as discussed in the 7<sup>th</sup> Chapter. The Delay and the Mux routines are used to create required delays and to select the desired pins of

the analog multiplexer. Other than these routines, there are also some other routines related to timers and external interrupts. These two routines are:

- T0ISR() interrupt 1
- E1ISR() interrupt 2

Timer T0ISR routine is called after every 125usec when the conversation or call negotiation is going on with some peer. This routine uses counter variable to save and retrieve the samples from the buffers. It also checks for the buffer to be fully populated such that the other buffer may be started and it be used later. This buffering technique is used to lessen the jitter in the voice which may occur when a single storage array is used. The E1ISR is used to detect the pressing of keys on the telephone keypad. The external interrupt pin 1 is mapped with the DTMF interrupt and when the user starts dialing, it detects the keys and stores them until the required number of keys are pressed. This number finally is used to make peer communication. Initial ARP request to the peer system is also sent through this routine.

### **8.3 System Command Flow Diagram**

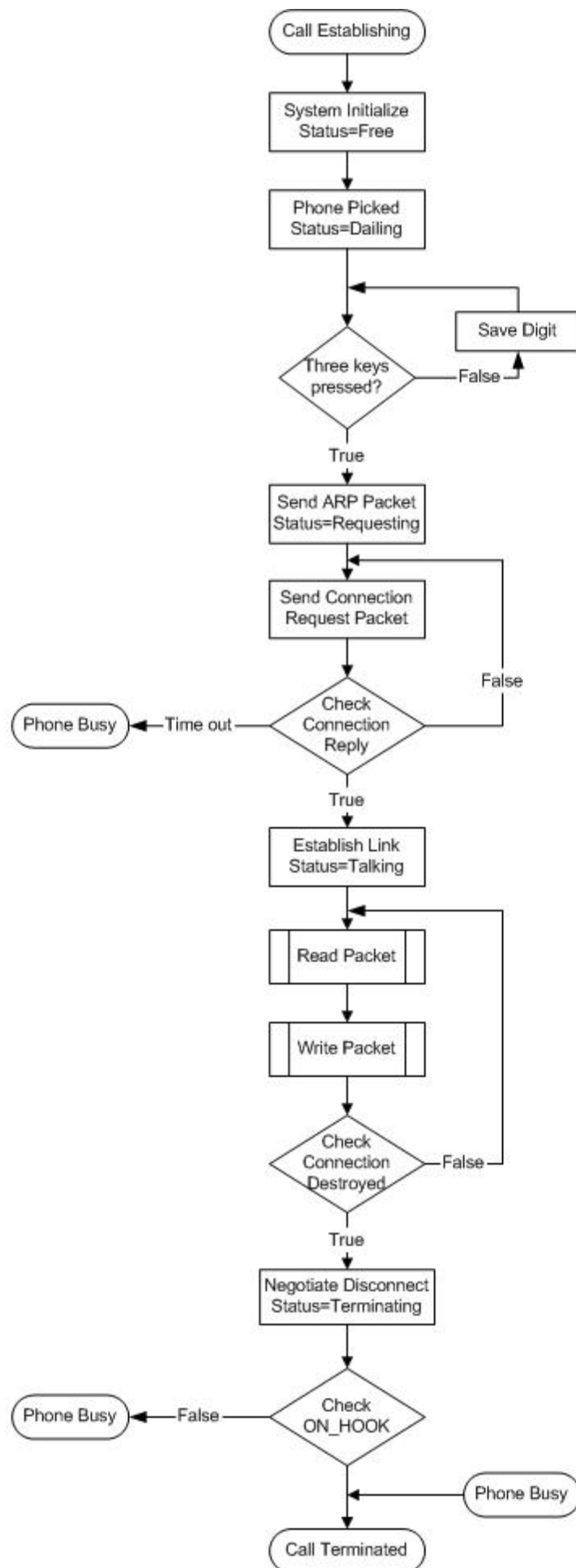
The diagram shown in figure 8.2 shows the flow of command when a call is being made through this system and the order of events that happen. Considering that the system itself has initialized, the behavior of the system is observed when the user makes a call. The system software contains a variable named **“Status”** that is used to track the current status of the system.

Until the user hasn't picked the phone, in its normal condition and the status is set to be in **“Free mode”**. As soon as the user picks the phone receiver, the phone gives a proper tone and sets itself into the **“Dialing mode”**. Here, as the diagram shows, the system polls for the three keys pressed by the user to form a valid no. Until these keys are pressed the system waits for the input. As soon as the number is completed, the status of the system is set as **“Requesting mode”**. The number entered is directly mapped into corresponding IP Address and an ARP Request is sent to this IP Address. A specific time interval is waited for the ARP Reply, failing upon which causes the phone to be busy.

On ARP Reply being received, a UDP Packet with a connection request is sent. This packet in fact tells the peer system that a caller wants to connect to his system. If the peer's status is already busy or dialing or terminating, it just replies telling his status. This effectively causes the call to be busy for the dialing party. The call also goes busy if the peer doesn't reply in a specific time limit.

If the peer replies with a positive reply, telling that it's ringing, the peer's status is set to **"Ringing Mode"** while the caller is stilling requesting mode. As soon as the phone is picked at the peer's end, both the systems go into **"Talking Mode"**. Here the series of packet reads and writes take place after specific intervals to transfer the voice. These packets also contain a byte of status to tell the peer of the current status of the system.

If the system doesn't get any packets from the other end for a longer duration, it goes into **"Busy Mode"** and an engage tone is heard on the phone set. Also, if the user goes on hook, the system status variable becomes in **"Terminating Mode"** and it sends final negotiation packets to the peer to end the conversation. As soon as the negotiation packets are transferred, the system goes into Free State again completing a phone call.



**Figure 8.2 System Command Flow Diagram**

## CHAPTER 9

# Ringer Microcontroller Software

### 9.1 General Description

The ringer microcontroller is AT89C2051 20 pin microcontroller chip from Atmel having a small hardware interface using only 4 pins. The sole purpose of this microcontroller is to create tones of various frequencies. These tones must also be switched on and off after regular intervals called as cadence of the tones. These produced tones are then transferred to the system through filters. In principle, there are four tones:

- Normal Tone
- Dial Tone
- Busy Tone
- Ringing Tone

Normal tone is the one which is heard as soon as the phone receiver is picked up. The dial tone is version of the same tone but having a specific cadence of 2 sec On and 4 sec Off. Busy tone is shrill then the normal and dial tone and has a lesser cadence of 0.5 sec On and Off each. The last tone is not played on the phone receiver, but is the one to produce the ringing from the phone when it is on hook. This ringing tone has the same cadence as the dial tone but its frequency is much lesser than then dial tone.

### 9.2 Software Description

The ringer microcontroller software contains a small number of variables to control the frequency and cadence parameters of the above mentioned four tones. The dial tone contains two separate and distinct on and off periods, so it uses two separate variables for its calculation.

The software uses two timers, one for shorter delays of high frequencies while the other for larger delays of low frequency ringing tone and various cadences. Initially, when the



microcontroller is switched on, the variables are initialized with their respective values. The timers are loaded with the appropriate counts. Both the timers are running in 8-bit auto reload mode. At the last of the main routine, the timers are set on and an infinite do nothing loop is entered.

There are three variables which are controlling the frequency of the three tones. These variables include:

- Bit\_Normal
- Bit\_Busy
- Bit\_Ring

The timer 0 interrupt routine T0ISR is used to toggle the values of these variables as soon as their respective counters reset. The cadence variables used are:

- Cadence\_Dail
- Cadence\_Busy

These variables are toggled in the timer 1 interrupt routine T1ISR based on the counters when they overflow at distinct intervals. No variable is used to store the status of the dial tone as it is of the same frequency as is the normal tone. For dial tone, the Bit\_Normal is directly applied with the dial cadence.

For combining the effects of the frequency and cadence variables, both the values are ANDed and the result is applied to the respective output pin. Hence, if the busy bit is showing a high voltage level while the busy cadence variable is showing the low, the result will be the low signal at the output.

## APPENDIX A

## Source Code of the Main Microcontroller

[illegible]

```

// "Page 0 Read-Only Registers"
    // +0: CR is read/write
    #define NE2K_CLDA0 0x01           // current local DMA address 0
    #define NE2K_CLDA1 0x02           // local dma 1
    // +3: BNRY is read/write
    #define NE2K_TSR 0x04              // transmit status register
    #define NE2K_NCR 0x05              // number of collisions register
    #define NE2K_FIFO 0x06             // FIFO
    // +7: ISR is read/write
    #define NE2K_CRDA0 0x08            // current remote DMA address 0
    #define NE2K_CRDA1 0x09            // remote DMA 1
    #define NE2K_RESV1 0x0A            // reserved
    #define NE2K_RESV2 0x0B            // reserved
    #define NE2K_RSR 0x0C              // receive status register
    #define NE2K_CNTR0 0x0D            // tally counter 0 (frame alignment)
    #define NE2K_CNTR1 0x0E            // tally counter 1 (CRC errors)
    #define NE2K_CNTR2 0x0F            // tally counter 2 (missed packet)

// "Page 0 Write-Only Registers"
    // +0: CR is read/write
    #define NE2K_PSTART 0x01           // page start register
    #define NE2K_PSTOP 0x02           // page stop register
    // +3: BNRY is read/write
    #define NE2K_TPSR 0x04             // transmit page start address
    #define NE2K_TBCR0 0x05            // transmit byte count register 0
    #define NE2K_TBCR1 0x06            // transmit byte count register 1
    // +7: ISR is read/write
    #define NE2K_RSAR0 0x08            // remote start address register 0
    #define NE2K_RSAR1 0x09            // remote start address register 1
    #define NE2K_RBCR0 0x0A            // remote byte count register 0
    #define NE2K_RBCR1 0x0B            // remote byte count register 1

```

```

#define NE2K_RCR 0x0C           // receive configuration register
#define NE2K_TCR 0x0D           // transmit configuration register
#define NE2K_DCR 0x0E           // data configuration register
#define NE2K_IMR 0x0F           // interrupt mask register

// Page 1 registers
// +0: CR spans pages 0,1, and 2
#define NE2K_PAR0 0x01           // physical address register 0
#define NE2K_PAR1 0x02           // physical address register 1
#define NE2K_PAR2 0x03           // physical address register 2
#define NE2K_PAR3 0x04           // physical address register 3
#define NE2K_PAR4 0x05           // physical address register 4
#define NE2K_PAR5 0x06           // physical address register 5
#define NE2K_CURR 0x07           // current page register
#define NE2K_MAR0 0x08           // multicast address register 0
#define NE2K_MAR1 0x09           // multicast address register 1
#define NE2K_MAR2 0x0A           // multicast address register 2
#define NE2K_MAR3 0x0B           // multicast address register 3
#define NE2K_MAR4 0x0C           // multicast address register 4
#define NE2K_MAR5 0x0D           // multicast address register 5
#define NE2K_MAR6 0x0E           // multicast address register 6
#define NE2K_MAR7 0x0F           // multicast address register 7

// Page 2 registers
// ... not implemented ...

// Page 3 registers
// ... not implemented ...

// Other Special Locations

```

```

#define NE2K_DATAPORT 0x10
#define NE2K_RESET 0x1F

// Bits in Various Registers

#define NE2K_CR_STOP 0x01           // stop card
#define NE2K_CR_START 0x02          // start card
#define NE2K_CR_TRANSMIT 0x04        // transmit packet
#define NE2K_CR_DMAREAD 0x08         // remote DMA read
#define NE2K_CR_DMAWRITE 0x10        // remote DMA write
#define NE2K_CR_NODMA 0x20           // abort/complete remote DMA
#define NE2K_CR_PAGE0 0x00           // select register page 0
#define NE2K_CR_PAGE1 0x40           // select register page 1
#define NE2K_CR_PAGE2 0x80           // select register page 2
#define NE2K_RCR_BCAST 0x04
#define NE2K_RCR_MCAST 0x08
#define NE2K_RCR_PROMISCUOUS 0x10
#define NE2K_RCR_MONITOR 0x20
#define NE2K_DCR_BYTEDMA 0x00
#define NE2K_DCR_WORDDMA 0x01
#define NE2K_DCR_NOLPBK 0x08
#define NE2K_DCR_FIFO2 0x00
#define NE2K_DCR_FIFO4 0x20
#define NE2K_DCR_FIFO8 0x40
#define NE2K_DCR_FIFO12 0x60
#define NE2K_TCR_NOLPBK 0x00          // mode 0: normal operation
#define NE2K_TCR_INTLPBK 0x02         // mode 1: internal loopback
#define NE2K_TCR_EXTLPBK 0x04         // mode 2: external loopback
#define NE2K_TCR_EXTLPBK2 0x06        // mode 3: external loopback
#define NE2K_TRANSMIT_BUFFER 0x40     // transmit buffer from 4000 - 5FFF.
#define NE2K_START_PAGE 0x46          // receive buffer ring from
#define NE2K_STOP_PAGE 0x60           // 0x6000-0x7FFF

//-----

```

```

// Public Declarations.

//-----

#define IP1 192
#define IP2 168
#define IP3 0
#define IP4 114


#define Free            0x00
#define Dailing         0x01
#define Requesting      0x02
#define Talking         0x03
#define Busy            0x04
#define Ringing         0x05
#define Terminating   0x06


#define False           0x00
#define True            0x01
#define Count_Max      200


/* Global Variables */
uchar xdata NIC_Addr[32]  _at_ 0x7800;           //CS1
uchar xdata ADC_Addr     _at_ 0xB800;           //CS2
uchar xdata DAC_Addr     _at_ 0xD800;           //CS3
uchar xdata ADC_Read0[0xFF] _at_ 0xE000;
uchar xdata ADC_Read1[0xFF] _at_ 0xE200;
uchar xdata DAC_Write0[0xFF] _at_ 0xE400;
uchar xdata DAC_Write1[0xFF] _at_ 0xE600;
uchar xdata DTMF[3]      _at_ 0xE7F0;
uchar CODEC_Count;
uchar DTMF_Count;
uchar Hook_Count;
uchar Status;

```

```

uchar Count;
uchar Temp;
uint Timeout;

bit MAC_Valid;
bit Write_Data;
bit Use_Array;
bit Snd_Array;
bit Rcv_Array;
bit Resend;

uchar MAC[6];

uchar PEER_MAC[6];
uchar PEER_IP[4];

uchar CONN_MAC[6];
uchar CONN_IP[4];

/* Global Port Bits */
sbit Mux2=P1^4;
sbit Mux1=P1^5;
sbit Mux0=P1^6;
sbit Ringer=P1^7;
sbit RST=P3^5;
sbit ON_HOOK=P3^4;
sbit Timer_Test=P3^1;
sbit NIC_Test=P3^0;

/*

```

	0	1	2	3	4	5	6	7
PORT 3	N/A	N/A	RST	WRR	RDD	ADR_EN	N/A	N/A

\*/

```

void NIC_Write(uchar Address,uchar Data);
uchar NIC_Read(uchar Address);
void NIC_Hard_Reset();
void NIC_Init();
void NIC_Read_MAC();
void Packet_Read();
void Packet_Write(uchar Type);
void Delay(uchar Value);
void Mux(uchar Value);
//-----
// Public Functions
//-----

```

```

void main()
{
    uchar Temp;
    for(Temp=0;Temp<=Count_Max;Temp++)
    {
        ADC_Read0[Temp]=0x7F;
        ADC_Read1[Temp]=0x7F;
        DAC_Write0[Temp]=0x7F;
        DAC_Write1[Temp]=0x7F;
    }
}

```

```

// Reset the System.

```



```

RST=0;
IT1=1;
IE=0x86;
TMOD=0x02;
TH0=0x72;
IP=0x02;

CODEC_Count=0x00;
Hook_Count=0x00;
Timeout=0x00;
Use_Array=0;
Snd_Array=1;
Rcv_Array=1;
Write_Data=0;
MAC_Valid=0;
Ringer=0;
// Initialization Process
NIC_Init();
Mux(0);
TR0=0;
// Packet Reading Loop
while(1)
{
    if(ON_HOOK==True)
    {
        if(Hook_Count>=20)
        {
            if(Status!=Ringing)
            {
                Status=Free;
                Timeout=0x00;
                TR0=0;
            }
        }
    }
}

```

```

    }
    }
    else
        Hook_Count++;
}
else if(ON_HOOK==False)                // When OFF-HOOK Then...
{
    Hook_Count=0x00;
    if(Status==Ringing)
    {
        Ringer=0;
        Mux(4);
        Status=Talking;
        CODEC_Count=1;
    }
    else if(Status==Free)
    {
        Status=Dailing;
        Mux(1);
// Give the Normal Tone
        DTMF_Count=0;
    }
}
Packet_Read();
if(Status==Talking) Mux(4);
if(Write_Data==1)
{
    if(MAC_Valid==1)
    {
        Write_Data=0;
        Packet_Write(3);
    }
}

```





```

        if(DTMF[DTMF_Count]==0x0A) DTMF[DTMF_Count]=0x00;
        DTMF_Count++;
        if(DTMF_Count==0x03)
        {
            CONN_IP[0] = 192;
            CONN_IP[1] = 168;
            CONN_IP[2] = 0;
            CONN_IP[3] = 0x100*DTMF[0]+0x10*DTMF[1]+DTMF[2]-
0xA2;

            Packet_Write(1);
            Status=Requesting;
            TR0=1;
        }
    }
}

// *_ *_ *_ *_ *_ *_ *_ *_ *_ *_ *_ *_ *_ *_ *_ *_ *_ *_ *_ *_ *_ *_ *_ *_ *_ *_ *_ *_

```

```

void Control(uchar Peer_Status)
{
    Timeout=0x00;
    switch(Peer_Status)
    {
        case Requesting:
            if(Status==Free)
            {
                Ringer=1;
                Status=Ringling;
                TR0=1;
            }
            break;
        case Talking:
            if(Status!=Free)

```









```

        NIC_Write(NE2K_RCR,NE2K_RCR_MONITOR);

// Step 5A: Place the NIC in loopback mode
        NIC_Write(NE2K_TCR,NE2K_TCR_INTLPBK);

// Step 5B: Initialize the transmit buffer start page
        NIC_Write(NE2K_TPSR,NE2K_TRANSMIT_BUFFER);

// Step 6: Initialize receive buffer ring (256 byte blocks)
        NIC_Write(NE2K_PSTART,NE2K_START_PAGE);

        NIC_Write(NE2K_BNRY,NE2K_START_PAGE);

        NIC_Write(NE2K_PSTOP,NE2K_STOP_PAGE);

// Step 7: Clear interrupt status register
        NIC_Write(NE2K_ISR,0xFF);

// Step 8: Initialize the interrupt mask register
        // Disable Interrupts
        NIC_Write(NE2K_IMR,0);

// Step 9A: Go to register page 1
        // NE2K_CR_PAGE1 | NE2K_CR_STOP | NE2K_CR_NODMA
        NIC_Write(NE2K_CR,0x61);

// Step 9B: Initialize hardware address
        NIC_Write(NE2K_PAR0,MAC[0]);
        NIC_Write(NE2K_PAR1,MAC[1]);
        NIC_Write(NE2K_PAR2,MAC[2]);
        NIC_Write(NE2K_PAR3,MAC[3]);
        NIC_Write(NE2K_PAR4,MAC[4]);

```



```

NIC_Write(NE2K_RBCR1,0);

NIC_Write(NE2K_IMR,0);

NIC_Write(NE2K_ISR,0xFF);

// Receive off
NIC_Write(NE2K_RCR,NE2K_RCR_MONITOR);

// Transmit off
NIC_Write(NE2K_TCR,NE2K_TCR_INTLPBK);

// Intend to read 32 bytes
NIC_Write(NE2K_RBCR0,32);
NIC_Write(NE2K_RBCR1,0);

// Low byte of start address (0x0000)
NIC_Write(NE2K_RSAR0,0);

// High byte of start address
NIC_Write(NE2K_RSAR1,0);

// NE2K_CR_PAGE0 | NE2K_CR_START | NE2K_CR_DMAREAD
NIC_Write(NE2K_CR,0x0A);

// Now Read the MAC Address.
MAC[0]=NIC_Read(NE2K_DATAPORT);
MAC[0]=NIC_Read(NE2K_DATAPORT);

MAC[1]=NIC_Read(NE2K_DATAPORT);
MAC[1]=NIC_Read(NE2K_DATAPORT);

```



```

// NE2K_CR_PAGE0 | NE2K_CR_START | NE2K_CR_NODMA
NIC_Write(NE2K_CR,0x22);

// Read the boundary pointer
Reg1=NIC_Read(NE2K_BNRY);

// Compare(CURR and BNRY Pointers) to Check that New Data has Arrived.
if(Reg0==Reg1) return; // No New Data, Exit Routine.
NIC_Test=~NIC_Test;
// There is data in the NIC's Rx buffer which we need to read out
// I don't know how many bytes i intend to read, so just set this to the maximum
NIC_Write(NE2K_RBCR0,0xFF);
NIC_Write(NE2K_RBCR1,0x00);

// Low byte of start address (0)
NIC_Write(NE2K_RSAR0,0);

// High byte of start address (BNRY)
NIC_Write(NE2K_RSAR1,Reg1);

// Begin the dma read
// NE2K_CR_PAGE0 | NE2K_CR_START | NE2K_CR_DMAREAD
NIC_Write(NE2K_CR ,0x0A);

// First Four Bytes are not part of the actual received ethernet packet.
// These Contain some status information about the packet.

Temp=NIC_Read(NE2K_DATAPORT); // Receive status code
NPP =NIC_Read(NE2K_DATAPORT); // Next packet pointer
Temp=NIC_Read(NE2K_DATAPORT); // Receive byte count low
Temp=NIC_Read(NE2K_DATAPORT); // Receive byte count high

```

```

// Now start reading the actual ethernet frame.
Temp=NIC_Read(NE2K_DATAPORT);           // Destination MAC Address
Temp=NIC_Read(NE2K_DATAPORT);           // My Device's MAC Address
Temp=NIC_Read(NE2K_DATAPORT);
Temp=NIC_Read(NE2K_DATAPORT);
Temp=NIC_Read(NE2K_DATAPORT);
Temp=NIC_Read(NE2K_DATAPORT);

// The next "6 bytes" are the Source MAC address.
PEER_MAC[0]=NIC_Read(NE2K_DATAPORT);
PEER_MAC[1]=NIC_Read(NE2K_DATAPORT);
PEER_MAC[2]=NIC_Read(NE2K_DATAPORT);
PEER_MAC[3]=NIC_Read(NE2K_DATAPORT);
PEER_MAC[4]=NIC_Read(NE2K_DATAPORT);
PEER_MAC[5]=NIC_Read(NE2K_DATAPORT);

// Look at the "Type" field in the ethernet frame.
// 0x0800 = IP  and 0x0806 = ARP
Reg0=NIC_Read(NE2K_DATAPORT);
if(Reg0!=0x08) goto Packet_Clean;        // Type high byte

Reg0=NIC_Read(NE2K_DATAPORT);            // Type low byte
switch(Reg0)
{
case 0x00:
// *****
// 0x0800: IP (Data Packet)      *
// *****

        // Fixed Length Data.
        for(Temp=0;Temp<=Count_Max;Temp++)
        {
                if(Rcv_Array==0)

```

```

DAC_Write0[Temp]=NIC_Read(NE2K_DATAPORT);
else
DAC_Write1[Temp]=NIC_Read(NE2K_DATAPORT);
}
if(Rcv_Array==0)
{
if(DAC_Write0[0]>0x08) goto Packet_Clean;
Control(DAC_Write0[0]);
}
else
{
if(DAC_Write1[0]>0x08) goto Packet_Clean;
Control(DAC_Write1[0]);
}
Rcv_Array=~Rcv_Array;
break;
case 0x06:
// *****
// 0x0806: ARP *
// *****

// Confirm hardware type 0x0001
Reg0=NIC_Read(NE2K_DATAPORT); // Hardware type high byte
if(Reg0!=00) goto Packet_Clean;

Reg1=NIC_Read(NE2K_DATAPORT); // Hardware type low byte
if(Reg1!=01) goto Packet_Clean;

// Confirm protocol type 0x0800(Subset of IP)
Reg0=NIC_Read(NE2K_DATAPORT); // Protocol type high byte
if(Reg0!=0x08) goto Packet_Clean;

```

```

        Reg1=NIC_Read(NE2K_DATAPORT);           // Protocol type low byte
        if(Reg1!=0x00) goto Packet_Clean;

// Confirm hardware size 6
        Reg0=NIC_Read(NE2K_DATAPORT);
        if(Reg0!=6) goto Packet_Clean;

// Confirm protocol size 4
        Reg0=NIC_Read(NE2K_DATAPORT);
        if(Reg0!=4) goto Packet_Clean;

// Confirm OP Code (0x0001/02,request/reply)
        Reg0=NIC_Read(NE2K_DATAPORT);
        if(Reg0!=00) goto Packet_Clean;

        Reg1=NIC_Read(NE2K_DATAPORT);
        if(Reg1!=0x01)
        if(Reg1!=0x02) goto Packet_Clean;

// Ignore sender's hardware address (we already recorded it)
        Temp=NIC_Read(NE2K_DATAPORT);
        Temp=NIC_Read(NE2K_DATAPORT);
        Temp=NIC_Read(NE2K_DATAPORT);
        Temp=NIC_Read(NE2K_DATAPORT);
        Temp=NIC_Read(NE2K_DATAPORT);
        Temp=NIC_Read(NE2K_DATAPORT);

// Record sender's IP address
        PEER_IP[0]=NIC_Read(NE2K_DATAPORT);
        PEER_IP[1]=NIC_Read(NE2K_DATAPORT);
        PEER_IP[2]=NIC_Read(NE2K_DATAPORT);
        PEER_IP[3]=NIC_Read(NE2K_DATAPORT);

```



```

// Ignore target hardware address (Meaningless: 00:00:00:00:00:00)
    Temp=NIC_Read(NE2K_DATAPORT);
    Temp=NIC_Read(NE2K_DATAPORT);
    Temp=NIC_Read(NE2K_DATAPORT);
    Temp=NIC_Read(NE2K_DATAPORT);
    Temp=NIC_Read(NE2K_DATAPORT);
    Temp=NIC_Read(NE2K_DATAPORT);

// Compare target IP address to our own.
    Reg0=NIC_Read(NE2K_DATAPORT);
    if(Reg0!=IP1) goto Packet_Clean;

    Reg0=NIC_Read(NE2K_DATAPORT);
    if(Reg0!=IP2) goto Packet_Clean;

    Reg0=NIC_Read(NE2K_DATAPORT);
    if(Reg0!=IP3) goto Packet_Clean;

    Reg0=NIC_Read(NE2K_DATAPORT);
    if(Reg0!=IP4) goto Packet_Clean;
//ARP Request/Reply is to this machine, take necessary action.

// End (abort) the DMA transfer
    // NE2K_CR_PAGE0 | NE2K_CR_START | NE2K_CR_NODMA
    NIC_Write(NE2K_CR,0x22);
// Update the BNRY (receive buffer ring boundary) pointer.
    NIC_Write(NE2K_BNRY,NPP);

// If its ARP Request, Send Reply; if its ARP Reply, save the Mac+IP Address.
    CONN_MAC[0]=PEER_MAC[0];
    CONN_MAC[1]=PEER_MAC[1];

```



```

{

// Set the Byte Count to Maximum(Don't know how much to send)
    NIC_Write(NE2K_RBCR0,0xFF);
    NIC_Write(NE2K_RBCR1,0);

// Start address
    NIC_Write(NE2K_RSAR0,0);
    // LOW byte
    NIC_Write(NE2K_RSAR1,NE2K_TRANSMIT_BUFFER);
    // HIGH byte

// Begin DMA write
    // NE2K_CR_PAGE0 | NE2K_CR_START | NE2K_CR_DMAWRITE
    NIC_Write(NE2K_CR,0x12);

switch(Type)
{
case 1:
    // *****
    // ARP Request Packet          *
    // *****

// Network Layer
    // Destination hardware address
        NIC_Write(NE2K_DATAPORT,0xFF);
        NIC_Write(NE2K_DATAPORT,0xFF);
        NIC_Write(NE2K_DATAPORT,0xFF);
        NIC_Write(NE2K_DATAPORT,0xFF);
        NIC_Write(NE2K_DATAPORT,0xFF);
        NIC_Write(NE2K_DATAPORT,0xFF);

    // Source hardware address

```

```

        NIC_Write(NE2K_DATAPORT,MAC[0]);
        NIC_Write(NE2K_DATAPORT,MAC[1]);
        NIC_Write(NE2K_DATAPORT,MAC[2]);
        NIC_Write(NE2K_DATAPORT,MAC[3]);
        NIC_Write(NE2K_DATAPORT,MAC[4]);
        NIC_Write(NE2K_DATAPORT,MAC[5]);

// "Ethernet" (not 802.3) Type 0x0806 (=ARP)
        NIC_Write(NE2K_DATAPORT,0x08);
        NIC_Write(NE2K_DATAPORT,0x06);

// IP Layer
// Hardware Type 0x0001
        NIC_Write(NE2K_DATAPORT,0x00);
        NIC_Write(NE2K_DATAPORT,0x01);

// Protocol Type 0x0800
        NIC_Write(NE2K_DATAPORT,0x08);
        NIC_Write(NE2K_DATAPORT,0x00);

// Hardware size 6
        NIC_Write(NE2K_DATAPORT,0x06);

// Protocol size 4
        NIC_Write(NE2K_DATAPORT,0x04);

// OP Code 0x0001 (ARP Request)
        NIC_Write(NE2K_DATAPORT,0x00);
        NIC_Write(NE2K_DATAPORT,0x01);

// Source Hardware Address
        NIC_Write(NE2K_DATAPORT,MAC[0]);

```

```
// *_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*
```

case 2:

```
// *****  
// ARP Reply Packet *  
// *****
```

// Network Layer

// Destination hardware address

```
NIC_Write(NE2K_DATAPORT,PEER_MAC[0]);  
NIC_Write(NE2K_DATAPORT,PEER_MAC[1]);  
NIC_Write(NE2K_DATAPORT,PEER_MAC[2]);  
NIC_Write(NE2K_DATAPORT,PEER_MAC[3]);  
NIC_Write(NE2K_DATAPORT,PEER_MAC[4]);  
NIC_Write(NE2K_DATAPORT,PEER_MAC[5]);
```

// Source hardware address

```
NIC_Write(NE2K_DATAPORT,MAC[0]);  
NIC_Write(NE2K_DATAPORT,MAC[1]);  
NIC_Write(NE2K_DATAPORT,MAC[2]);  
NIC_Write(NE2K_DATAPORT,MAC[3]);  
NIC_Write(NE2K_DATAPORT,MAC[4]);  
NIC_Write(NE2K_DATAPORT,MAC[5]);
```

// "Ethernet" (not 802.3) Type 0x0806 (=ARP)

```
NIC_Write(NE2K_DATAPORT,0x08);  
NIC_Write(NE2K_DATAPORT,0x06);
```

// IP Layer

// Hardware Type 0x0001

```
NIC_Write(NE2K_DATAPORT,0x00);  
NIC_Write(NE2K_DATAPORT,0x01);
```

// Protocol Type 0x0800

```

        NIC_Write(NE2K_DATAPORT,0x08);
        NIC_Write(NE2K_DATAPORT,0x00);

// Hardware size 6
        NIC_Write(NE2K_DATAPORT,6);

// Protocol size 4
        NIC_Write(NE2K_DATAPORT,4);

// OP Code 0x0002 (ARP reply)
        NIC_Write(NE2K_DATAPORT,0x00);
        NIC_Write(NE2K_DATAPORT,0x02);

// Source Hardware Address
        NIC_Write(NE2K_DATAPORT,MAC[0]);
        NIC_Write(NE2K_DATAPORT,MAC[1]);
        NIC_Write(NE2K_DATAPORT,MAC[2]);
        NIC_Write(NE2K_DATAPORT,MAC[3]);
        NIC_Write(NE2K_DATAPORT,MAC[4]);
        NIC_Write(NE2K_DATAPORT,MAC[5]);

// Source IP Address
        NIC_Write(NE2K_DATAPORT,IP1);
        NIC_Write(NE2K_DATAPORT,IP2);
        NIC_Write(NE2K_DATAPORT,IP3);
        NIC_Write(NE2K_DATAPORT,IP4);

// Target Hardware Address
        NIC_Write(NE2K_DATAPORT,PEER_MAC[0]);
        NIC_Write(NE2K_DATAPORT,PEER_MAC[1]);
        NIC_Write(NE2K_DATAPORT,PEER_MAC[2]);
        NIC_Write(NE2K_DATAPORT,PEER_MAC[3]);

```





```

        NIC_Write(NE2K_DATAPORT,MAC[5]);

// "Ethernet" (not 802.3) Type 0x0800 (=IP)
        NIC_Write(NE2K_DATAPORT,0x08);
        NIC_Write(NE2K_DATAPORT,0x00);

// Fixed Length Data.
        for(Temp=0;Temp<=Count_Max;Temp++)
        {
                if(Snd_Array==0)
                        NIC_Write(NE2K_DATAPORT,ADC_Read0[Temp]);
                else
                        NIC_Write(NE2K_DATAPORT,ADC_Read1[Temp]);
        }

        Snd_Array=~Snd_Array;
        Temp=0xD6;

break;
}

// End the DMA Transfer
        // NE2K_CR_PAGE0 | NE2K_CR_START | NE2K_CR_NODMA
        NIC_Write(NE2K_CR,0x22);

// Bytes to Send.
        NIC_Write(NE2K_TBCR0,Temp);
        NIC_Write(NE2K_TBCR1,0x00);

// Start Position.
        NIC_Write(NE2K_TPSR,NE2K_TRANSMIT_BUFFER);

// Clear Transmit status Register.
        NIC_Write(NE2K_TSR,0x00);

```



## APPENDIX B

## Source Code of the Ringer Microcontroller

[illegible]



[illegible]

```
{
    TR1=0;
    TH1=0x3C; // A Base Delay.
    TL1=0xAF;

    if(--Delay_DailOFF==0) // Dail Cadence.
    {
        Delay_DailOFF=0x1E;
        if(--Delay_DailON==0)
        {
            Delay_DailON=0x03;
        }
    }
}
```

```

        Cadence_Dail=0;                /* ON Delay = 1.5 sec.*/
    }
    else
    {
        Cadence_Dail=1;                /* OFF Delay = 3 sec.*/
    }
}
if(--Delay_Busy==0)                  // Busy Cadence.
{
    Delay_Busy=0x08;
    Cadence_Busy=~Cadence_Busy;      /* Busy Delay = 0.5 sec.*/
}
Bit_Ring=~Bit_Ring;
Tone_Ring=Bit_Ring | Cadence_Dail; // Ring Tone(with Dail Cadence).
TR1=1;
}

```

## REFERENCES

### Books:

8051 Microcontroller by Scott Machenzie 2<sup>nd</sup> Edition

The 8051 Microcontrollers and Embedded Systems by Muhammand Ali Mazidi

Electronic Devices & Circuit Theory by Boylestad 8<sup>th</sup> Edition

Digital Logic Design by Morris Mano

Computer Networks by Behrouz A. Ferouzan

### Internet:

Various IC Datasheets

[www.Fairchildsemi.com](http://www.Fairchildsemi.com)


[www.nationalsemiconductor.com](http://www.nationalsemiconductor.com)

[www.zarlink.com](http://www.zarlink.com) for SLIC, DTMF Datasheets

[www.atmel.com](http://www.atmel.com) for 89C51 and AT89C2051 Datasheets

[www.datasheetarchive.com](http://www.datasheetarchive.com)

[www.semicon.mitel.com](http://www.semicon.mitel.com)

D8390 chip datasheet by  *National Semiconductor*

Keil Software and Emulator Hardware Specifications

[www.keil.com](http://www.keil.com) and software manual

Ethereal Network Monitoring Software

[www.ethereal.com](http://www.ethereal.com)

#### Embedded Ethernet links

[www.3beans.com/ether.html](http://www.3beans.com/ether.html)

[www.8052.com](http://www.8052.com)

[www.embeddedether.net](http://www.embeddedether.net)

[www.ee.ucr.edu/~phoang/index.htm](http://www.ee.ucr.edu/~phoang/index.htm)

[www.realtek.com.tw](http://www.realtek.com.tw) for NIC configuration setting

[www.kano.org.uk](http://www.kano.org.uk) for a similar project overview

#### RFCs Used

RFC 768 User Datagram Protocol

RFC 1071 Computing the Internet Checksum

RFC 791 Internet Protocol

RFC 826 Address Resolution Protocol

#### ISA Bus Technical Summary

<http://www.techfest.com/hardware/bus/isa.htm>

<http://users.supernet.com/sokos/ISA.HTM>

[http://sunsite.tut.fi/hwb/co\\_ISA\\_Tech.html](http://sunsite.tut.fi/hwb/co_ISA_Tech.html)