RMIT

CO 438

Engineering Design 4

1986

Title : Ethernet Packet Telephone

Field of work: Communication Networks

Students : Hien Linh Nguyen 830152H
Nghiem Huu Pham 820123Q

Supervisor : Dr. Greg Egan

DEPARTMENT OF COMMUNICATION AND ELECTRONIC ENGINEERING

Royal Melbourne Institute of Technology Ltd.
124 Latrobe Street,
Melbourne, Victoria, 3000
Australia

CONTENTS LISTING

ABSTRACT 1

I. INTRODUCTION 2

II. SOME DISCUSSION ON IMPORTANT ASPECTS OF PACKET VOICE SYSTEMS .. 4

A. Basic principle of a packet voice communication system 4

B. Advantages of packet voice communication systems 4

C. Digital speech processing functions 6

1. Speech encoding algorithms 6

2. Speech activity detection 6

3. Echo control 7

D. Packet speech protocol functions 7

E. Speech packetisation and reconstitution 8

1. Choice of packet size 9

2. Time stamps and sequence numbers 10

3. Reconstitution of speech from received packets 10

F. Conferencing techniques 11

G. Statistical multiplexing of packet voice and data 12

III. THE PROJECT 14

A. Period 1: Z80 work station-based development (3/86 - 9/86). 14

1. Hardware development 15

a. Telephone line interfacing circuit 15

b. Packetising and unpacketising circuit 17

i. Packetising circuit 17

ii. Unpacketising circuit 18

c. Clock circuit 20

2. Software development 21

a. The block diagram 21

b. The Packet Telephone state 22

i. The Power-on test block 22

ii. The Programming Slic&Slac state 22

iii. The Simulation state 24

c. Notes on how to program the transmit GX and receive GR filters of the SLAC 28

B. Period 2: IBM work station-based development (9/86 - 10/86) 30

1. Hardware development 31

2. Software development 32

IV. FUTURE DEVELOPMENT 33

V. CONCLUSION 35

VI. ACKNOWLEDGEMENT 36

VII. REFERENCES 37

APPENDIX-A: POWER SUPPLIES

APPENDIX-B: CIRCUIT DIAGRAMS

APPENDIX-C: PARTS LIST

APPENDIX-D: SOFTWARE LISTINGS

APPENDIX-E: DATA SHEETS

ABSTRACT

This report describes the development and testing of hardware and software for an experimental prototype with application to packet voice communication systems. The prototype will allow bidirectional transmission of voice across an Ethernet Local Area Network (LAN) in form of packets.

I. INTRODUCTION

In many buildings, we usually find that there are two networks: one is the Local Area Network for data and the other is the traditional PABX network for voice (Figure 1). This fact may prompt us to ask questions like: "Two networks ? Well, is it a duplication of communication resources ?" or "Can the Local Area Network be used to carry both voice and data ?". And if the answer to these questions is "Yes !", it means that the Local Area Network will be singly capable of supporting both voice and data traffic and consequently, it also means that significant savings from extra communication equipment and extra cable laying can be made.

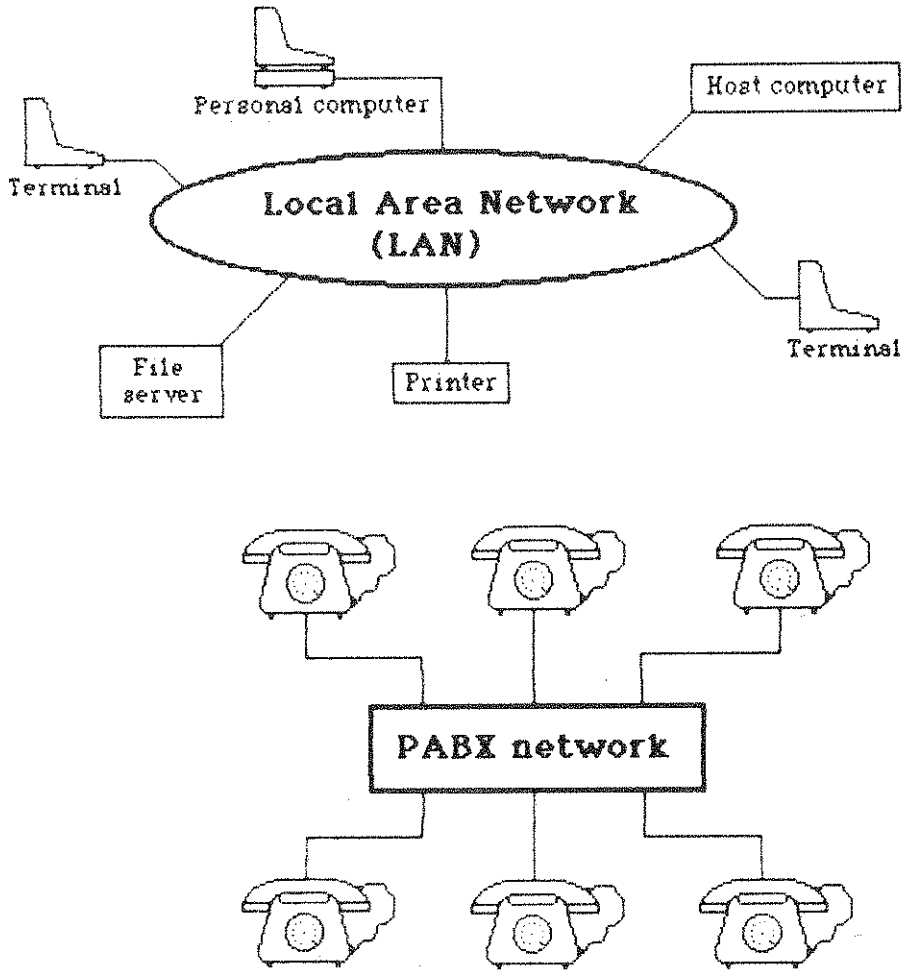


Figure 1. Seperate Local Area Network and PABX Network in one building.

Researchers have anticipated that in the long run, the two existing networks will merge and become one (Figure 2). Here, we are talking about two possibilities: either the Local Area Network will remain and the PABX network will be removed or vice versa. Only time will be able to give a definite answer to whether which network will become dominant.

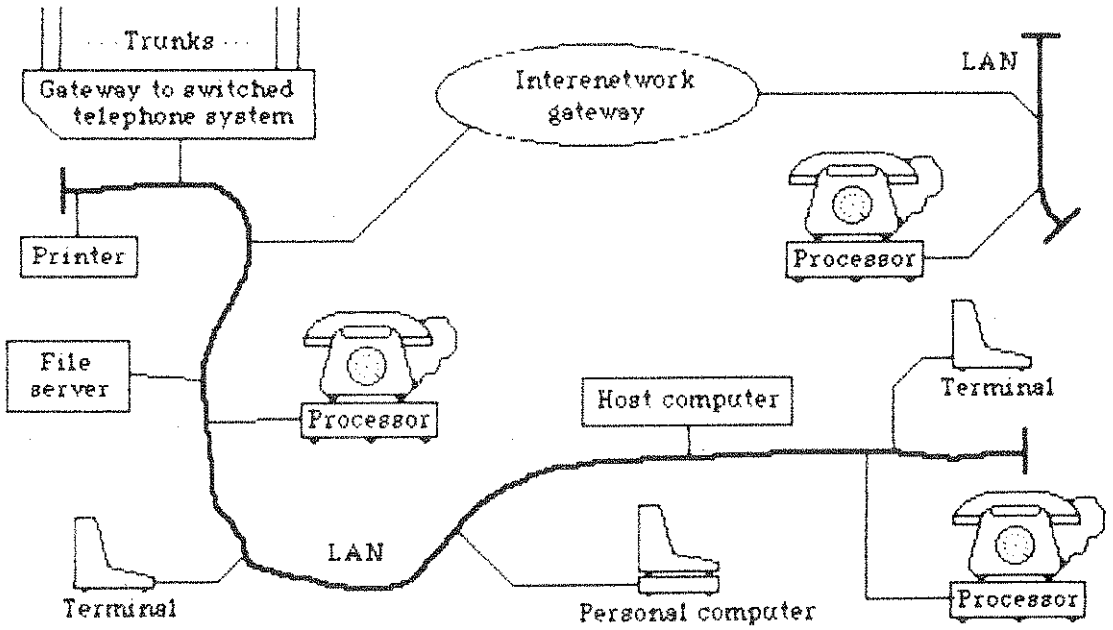


Figure 2. An integrated data and voice network.

In this project, we have investigated the feasibility of using the Local Area Network to carry both data and voice. The project can also be looked at as a preliminary experimental attempt to establish voice communication in form of packets through a Local Area Network. In this case, the Local Area Network is the Ethernet of our Communication/Electronic Engineering Department at RMIT. However, the basic principle of any packet voice communication system will be the same to a large extent, regardless of the type of Local Area Network it runs on.

II. SOME DISCUSSION ON IMPORTANT ASPECTS OF PACKET VOICE NETWORKS

The purpose of this section is to provide some background information on various important aspects which should be addressed in any packet voice system implementation.

A. Basic principle of a packet voice communication system

The basic principle of packet voice communication is quite simple. Voice message, e.g., coming from a telephone set in analog form, is first low pass filtered, then digitised and encoded into digital form using an Analog-to-Digital Converter (ADC). The digital information is then stored temporarily in some buffer and "packetised", i.e., the digital signals are assembled into packet of certain length. The packet can then be sent to the destination via the transmission medium of the Local Area Network. Depending on the type of the Local Area Network, there are certain rules and protocols that must be observed in this step. AT the destination, the received packet is stored in some buffer and "unpacketised", i.e., the digital signals are reconstituted from the packet. The digital information is next decoded and undigitised, using a Digital-to-Analog Converter (DAC), then low pass filtered and amplified to finally resemble the original voice message.

B. Advantages of packet voice communication systems

Packet voice system has many advantages to offer in comparison with traditional switched telephone system. It is particularly suitable for bursty type of traffic, which is a nature of human speech. It eleminates the need to dedicate and maintain a physical link during the whole duration of the call by transmitting

packets only when speakers are actually talking (i.e., during talkspurts). During the silence intervals, the same link may be used for other services, hence the utilisation of the communication channel can be increased. Packet networks offer significant advantages for digital voice conferencing in terms of channel utilisation (only one of the conferees needs to use the channel capacity at any given time) and in terms of control flexibility. A packet network allows convenient accomodation for voice terminals with different bit rates and data formats. Each voice encoder will use the channel capacity necessary to transmit its information rather than the fixed minimum bandwidth increment typically used in circuit-switched networks. Packet networks also provide a system environment for effective exploitation of variable-bit-rate voice transmission techniques, either to reduce average end-to end bit rate or to dynamically adapt voice bit rate to network conditions. Using digital form, packet network is also less sensitive to error and noise degradation, therefore better audio quality can be achieved. Last but not least, security of information can be readily implemented using coding techniques such as scrambling, etc.

From an integrated network point of view, the integration of voice with data in a common packet switch network offers potential cost savings through sharing of switching and transmission resources, as well as enhances services for users who require access to both voice and data communications. Packet internetworking techniques can be applied to provide intercommunications among voice users on different types of networks.

C. Digital speech processing functions

The primary speech processing function for packet voice system is speech digitisation. Two other important speech processing functions are also noted here: speech activity detection and echo control.

1. Speech Encoding Algorithms

Speech is a compressible source that can be coded at rates ranging from 64 Kbits/s to below 2.4 Kbits/s. Recent packet experiments have made use of the pulse code modulatin (PCM) widely used in digital telephony, but some earlier work used encoding techniques such as CVSD (continuously variable slope data modulation) or LPC (linear predictive coding) to provide data rates low enough for use on the networks that were available for experimentaion. Low data rates, however, usually required complex coders, or additional sacrifice of voice quality.

Packet systems offer flexibility for taking advantage of speech encoders at a variety of rates. More complex coding schemes can be applied which vary the transmission rates according to the time-varying compressibility of the speech signal. Even multirate "embedded coding" algorithms can be used to allow rapid adaptaion of voice bit rates to network conditions which may vary during a call.

Selection of a speech coding algorithm for a given application depends on many factors including network bit rate constraint, speech quality needs, noise or distortions on the input speech, and terminal cost and complexity constraints.

2. Speech Activity Detection

A key advantage of packet speech is the ability to save bandwidth by transmitting packets only during talkspurts. Therefore, accurate discrimination between speech and silence, or speech

activity detection (SAD), is an essential speech processing function. The SAD algorithm must minimize the average percentage activity, but also meets tight constraints on the fraction of lost speech. SAD, in a laboratory or quiet input speech environment, is relatively straightforward. But when the speaker is in a noisy environment, or when the speech originated in the switched telephone network, the design of effective SAD algorithm is more difficult.

3. Echo Control

Echo control is not needed in a pure packet voice system in spite of the delays that may be present since the system is fully digital and provides isolation between the two directions of voice transmission for the entire path between sending and receiving handsets. However, echo control becomes an issue if we wish to connect the packet network and the common switched telephone network. Techniques for controlling echo include 1) echo suppression, generally aimed at passing speech in only one direction at a time; and 2) echo cancellation, which attepmts to adaptively cancel echo and maintain full duplex speech. Echo cancellation is usually the preferred, but more costly technique. Echo canceller chips which reduce the cost have become available.

D. Packet speech protocol functions

The technique of protocol layering to partition and organise the task of providing various levels of communication services has been a fundamental aspect of packet communication systems. The protocols are usually designed to provide very reliable end-to-end packet delivery either at high throughput or low delay. Some protocols impose end-to-end flow restrictions which include retransmissions when necessary to reliably deliver all the packets and work against the

simultaneous achievement of high throughput and low delay. But for real-time voice communication, both high throughput and low delay are needed. Some reliability may be sacrificed, as a small percentage of lost packets is tolerable. Therefore, new protocol developments are needed for packet voice.

Some basic functions of a packet voice protocol should include:

1) Call initiation and termination, including negotiation of voice encoder compatibility and handling of ringing and busy conditions.

2) Packetisation of voice for transmission, with the time stamps and sequence numbers needed for speech reconstitution at the receiver.

3) Speech playout with buffering to smooth variable packet delays.

There is also another generation of voice protocols with a more general internetwork-oriented approach and with network-dependent aspects limited fo the lowest levels. The "higher" functions of call setup, packetisation, and reconstitution, as well as dynamic conference control features are incorporated. The lower level protocol provides and efficient internet transport mechanism for both point-to-point conversations and conferences.

E. Speech packetisation and reconstitution

Packet communication necessarily involves both fixed components of delay due to transmission and propagation, and statistically varying components such as queueing delays in network nodes or in gateways. Additional varying delay components are caused by packet retransmission to compensate for errors in delivery and by

the possibility that all packets between a particular source and destination may not follow the same route. In addition to delay effect, some packets may be lost between source and destination. In this regard, a delay versus reliability tradeoff is possible where (for example) delays due to retransmissions can be reduced at a cost of an increase in percentage of lost packets.

The purpose of speech packetisation and reconstitution algorithms is to provide speech with 1) minimum end-to-end delay and 2) any anomolies caused by lost or late packets basically imperceptible to the listener. Ideally, the overall packet network would provide high enough link bandwidths and sufficient nodal processing power to keep delay and delay dispersion within tightly controlled limits.

1. Choice of Packet Size

It is usually necessary to make compromises to resolve the issue of packet size. In order to minimize both the packetisation delay at the transmitter and the perceptual effect of lost packet anomolies at the receiver, packets should be as short as possible. Experience with packet lost anomolies indicates that individual packets should ideally contain no more than 50 ms of speech; ideally, we would like packets to be even shorter to minimize packetisation delay. On the other hand, in order to maintain high channel utilisation, we would like to keep the number of speech bits per packet as high as possible relative to the overhead which must accompany each packet.

The choice of packet size is also influenced by limitations on network throughput in packets/s. For the same user data rate, processing loads on network nodes will generally increase as packet size is decreased. This can force use of longer packets.

2. Time Stamps and Sequence Numbers

To assist in the reconstitution process, it is desirable to include a time stamp and a sequence number with each transmitted packet. The time stamp allows the receiver to reconstitute speech with accurate silent gap durations in spite of varying delays between talkspurts. Incorrect gap durations can cause significant perceptual degradation in the output speech, especiallly for short gaps between syllables, or between words in a phrase. The time stamp also allows reordering of out-of-order packets at the receiver. The time stamp is usually derived by counting every speech or silent parcel generated by the processor incorporated at the telephone set.

The sequence numver allows the receiver to detect lost packets whereas with a time stamp alone, it would not be possible to distinguish silence gaps from packet loss. The detection of lost packets can be used to inform the listener (e.g., by playing out a distinct audible signal) that some speech has been lost. This can be particularly important if packets contain enough speech to include linguistically significant utterances (such as the word "not"). Detection of lost packets can also be used to allow the terminal to adapt bit rate and/or packet rate to network conditions.

If the network provides services with very short delays and very little delay dispersion, then satisfactory speech can be produced without either time stamps or sequence numbers. However, it is experienced by researchers that both should be included.

3) Reconstitution of Speech from Received Packets

The reconstitution algorithm has two major tasks: 1) it must buffer incoming packets and decide exactly when to play them out, and 2) it must decide what to play out when it has finished playing out a packet and the next packet is not available.

The degree of complexity to be built into the reconstitution algorithm should be chosen based on the knowledge we do have of network delays. A fixed reconstitution delay would suffice if nerwork delays and delay dispersion are short. If delays are expected to be large or dispersions vary greatly with the network load, it would be desirable to use some adaptive algorithms to adjust the reconstitution delay to effect the compromise between packet loss and overall delay.

The other major reconstitution algorithm task is to decide what to play out when it has finished playing out a packet and the next packet is not available. This can result from a late or lost packet or it may simply indicate a pause in the talker's speech. Typically, the reconstitution algorithm has no way to distinguish these cases and should take the same action in either case. A number of fill-in strategies have been suggested, including 1) filling with silence, 2) filling by repeating the last segment of speech data, and 3) filling repeated frames of speech data which are made voiceless and have energy values which decay with time. The third strategy has generally been found the most effective. However, the best choice of fill-in strategy varies with encoder type, packetisation size, and statistics of gaps introduced by network.

F. Conferencing techniques

Digital voice conferencing imposes a number of requirements in addition to those required for point-to-point speech. There is a need to set up and control multiple connections and to deliver each talker's speech to multiple destinations.

Packet techniques offer advantages for digital voice conferencing in a number of areas. Since packets need be sent only

when speech is present, they can make very efficient use of network resources in conferences where typically only one participant is speaking at any given time. Because connections to packet networks are multiplexed, it is simple for speech terminals and conference controllers to exchange control information at the same time that speech is being transmitted. This out-of-band signalling capability helps in achieving effective conferencing control including the control algorithm that selects a talker to "have the floor" at any given time.

6. Statistical multiplexing of packet voice and data

An important goal for packet voice systems is to achieve efficient statistical multiplexing of multiple voice users, and of voice users with data traffic on common transmission resources. Much analysis and simulation work have been reported showing potentials and limitations of voice/data multiplexing for various system configurations. Some selective observations related to statistical multiplexing in packet voice systems are noted below.

First, packet speech multiplexing allows a straightforward utilisation of the tradeoff between delay and channel utilisation. The number of users multiplexed on a link can be increased at a cost in variable buffering delay at the multiplexer.

A second observation, based on simulation, is that interactive data traffic (characterised by Poisson packet arrival processes) can make efficient use of silence intervals in voice calls. However, the utilisation by data traffic of varying capacity due to voice calls initiation and termination is not nearly as effective due to much slower variation in channel capacity used by voice.

A third observation is that local area carrier sense multiple access (CSMA) cable networks can be used effectively for

voice. The bandwidth utilisation of such a CSMA network can be equal to or better than the efficiency obtained by using fixed time division multiple access (TDMA). CSMA cable networks have been effectively employed for packet voice systems.

Finally, variable-rate voice flow control techniques using embedded coding can be employed effectively in situations where we are attepting to maintain link loads close to capacity, and temporary overloads are inevitable. Embedded coding allows immediate response by network nodes to such overloads (by discarding packets), with minimal impact on speech users, since communications can be maintained with a temporary degradation in speech fidelity.

III. THE PROJECT

The project activities can be divided into 2 periods: period 1 from 3/86 to 9/86, Z80 Work Station-based development; and period 2 from 9/86 to 10/86, IBM Work Station-based development.

A. Period 1: Z80 Work Station-based Development (3/86 - 9/86)

In this first period, it was planned that the project would be carried out in two stages (Figure 3):

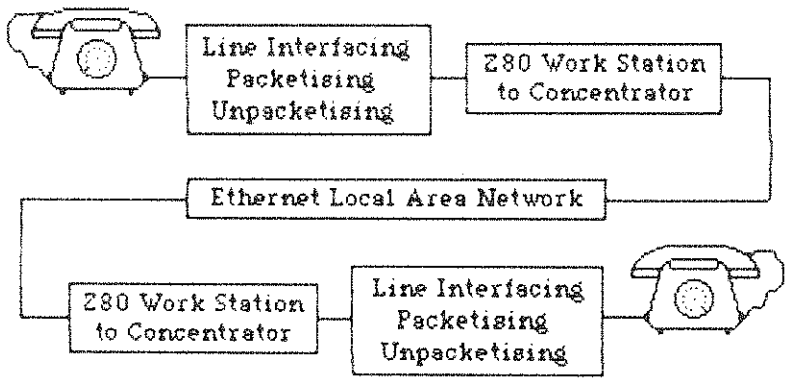


Figure 3. The Block Diagram of the Project in Period 1.

* Stage 1: In this stage, the aim was to design the front-end circuit for the Ethernet packet telephone system. The functions of the circuit would include:

- allows interfacing to any standard telephone set, e.g., Telecom 800 series.

- packetises voice information going into the network and unpacketises voice information coming out from the network, for the experiment we chose a packet size of 128 bytes.

* Stage 2: In this stage, the aim was to interface the front-end prototype developed in Stage 1 with a Department's concentrator in order to gain access to the Ethernet. This would allow voice transmission through the Ethernet in form of packets.

Unfortunately, after we finished the first stage, it turned out that no concentrator was available for our experiment. Therefore, we had to stop after Stage 1 and found some other way to gain access to the Ethernet (please refer to period 2).

In the following paragraphs, we describe all the work done in Stage 1 of this period. The hardware part is discussed first and the software part is explained after that.

1. Hardware Development

Figure 4 shows the block diagram of the hardware prototype.

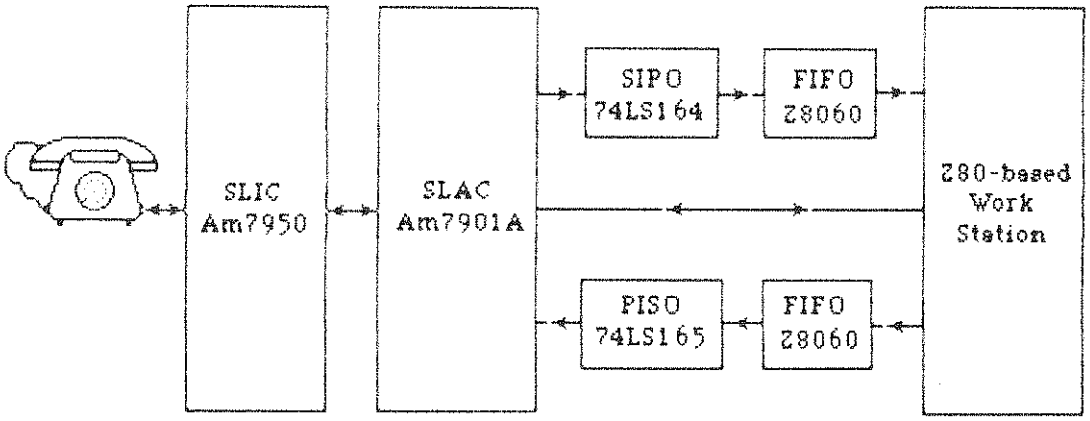


Figure 4. The Block Diagram of the Hardware.

a. Telephone Line Interfacing Circuit (Figure 5)

A device called SLIC (Subscriber Line Interface Circuit) can be used to interface to the telephone set. The SLIC Am7950 from Advanced Micro Devices is chosen for this job. The SLIC acts like a bridge between the high-voltage side, i.e., the telephone set circuit, and the low-voltage side, i.e., the prototype circuit. It also acts as a hybrid transformer which performs 2-to-4 wire and 4-to-2 wire conversions. The SLIC Am7950 provides off-hook detection and ring trip detection through a TTL compatible output. Over-voltage protection and ringing are also provided by means of some minimum amount of external components.

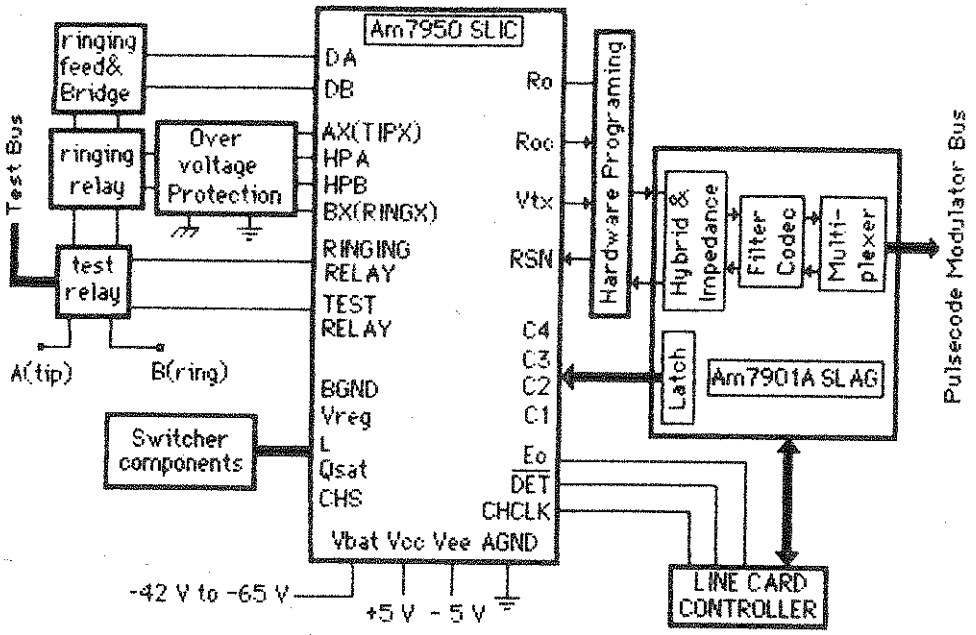


Figure 5. The Telephone Line Interfacing Circuit.

The companion of the SLIC Am7950 is the SLAC Am7901A (Subscriber Line Audio Processing Circuit). The SLAC performs the analog-to-digital and digital-to-analog conversions, encoding, decoding and signal processing functions. In the transmit path, analog voice signals received from the SLIC are converted and digitally processed to generate 8-bit Pulse Code Modulation (PCM) u-law codes. (Although A-law should be used in Australia, we could not obtain the A-law SLAC version, therefore we had to use the u-law version. However, these two versions are pin compatible, hence the u-law chip can be replaced any time with the A-law chip if available.) In the receive path, the 8-bit PCM codes are received, processed and converted back to voice analog signals. There are six digital filters in the signal processing sections, all of them are user programmable. These filters allow the user to independently modify the gain in both transmit and receive paths, provide trans-hybrid balancing in the system, and adjust two-wire line termination impedance.

The SLAC Am7901A is also the control interface to the SLIC Am7950. Both of these devices are programmable, the SLIC can be programmed via the SLAC, and the SLAC can be programmed via a serial

I/O interface. They are both conformed to CCITT specifications and together, they form the heart of the telephone line interface circuit.

b. Packetising and Unpacketising Circuit

The packetising and unpacketising circuits consist mainly of 2 First-In-First-Out (FIFO) buffers Z8060 (from Zilog), a Serial-In-Parallel-Out (SIPO) IC 74LS164, a Parallel-In-Serial-Out (PISO) IC 74LS165 and some other standard TTL devices. The SIPO converts the serial bit stream from the SLAC output into a parallel 8-bit word for the transmit FIFO buffer. On the other hand, the PISO converts the parallel 8-bit word from the receive FIFO buffer into a serial bit stream for the SLAC input. The FIFO buffers act as elastic stores. In the transmit path, the transmit FIFO buffers the digital signals for the 128-byte packet to be assembled in memory. In the receive path, the receive FIFO buffers the 128-byte packet for the digital signals to be reconstituted and processed by the SLAC.

Following, more details of the components and timing sequences of the packetising and unpacketising circuits are presented.

i. Packetising circuit

The packetising circuit accepts 8-bit serial PCM output codes from the SLAC, performs serial-to-parallel conversions, and buffers the digital signals for the 128-byte packet to be assembled in the memory of the Work Station.

Because the processing rate of the SLAC is 8 KHz, 1 burst of 8-bit serial PCM output code is available every 125 us. Hence the time taken to assemble a packet is:

128 X 125 us = 16 ms

The diagram of the packetising circuit is shown on the next page:

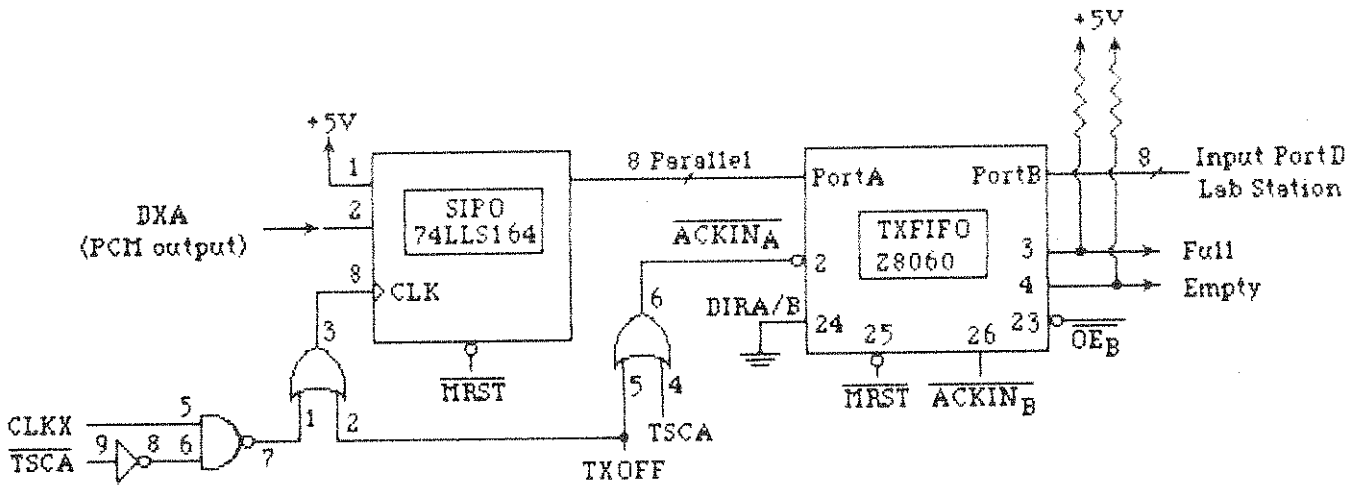


Figure 6. The Packetising Circuit.

The important timing sequence is shown below:

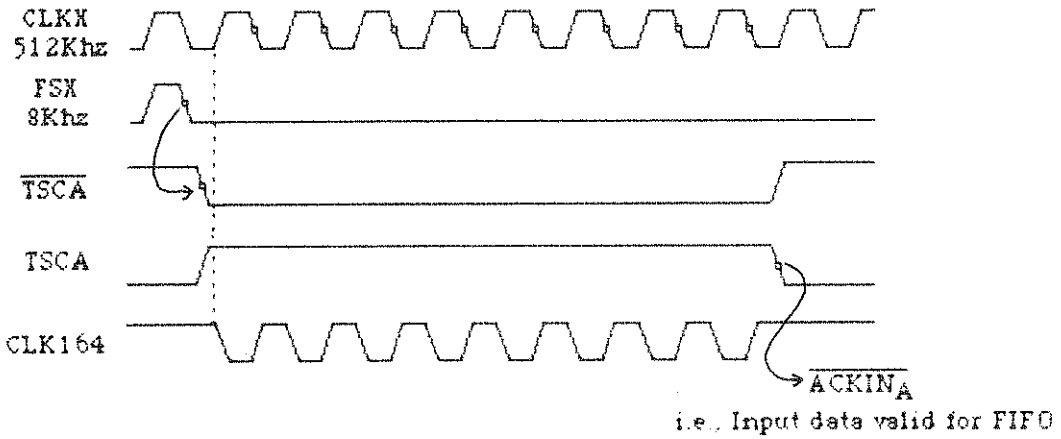


Figure 7. The Timing Sequence of the Packetising Circuit.

ii. Unpacketising circuit

The unpacketising circuit accepts and buffers 8-bit parallel words from the memory of the Work Station, performs parallel-to-serial conversions and generates PCM 8-bit serial bit streams for the SLAC input.

Similarly to the packetising circuit, the time taken to disassemble a packet is 16 ms.

The diagram of the unpacketising circuit is shown below:

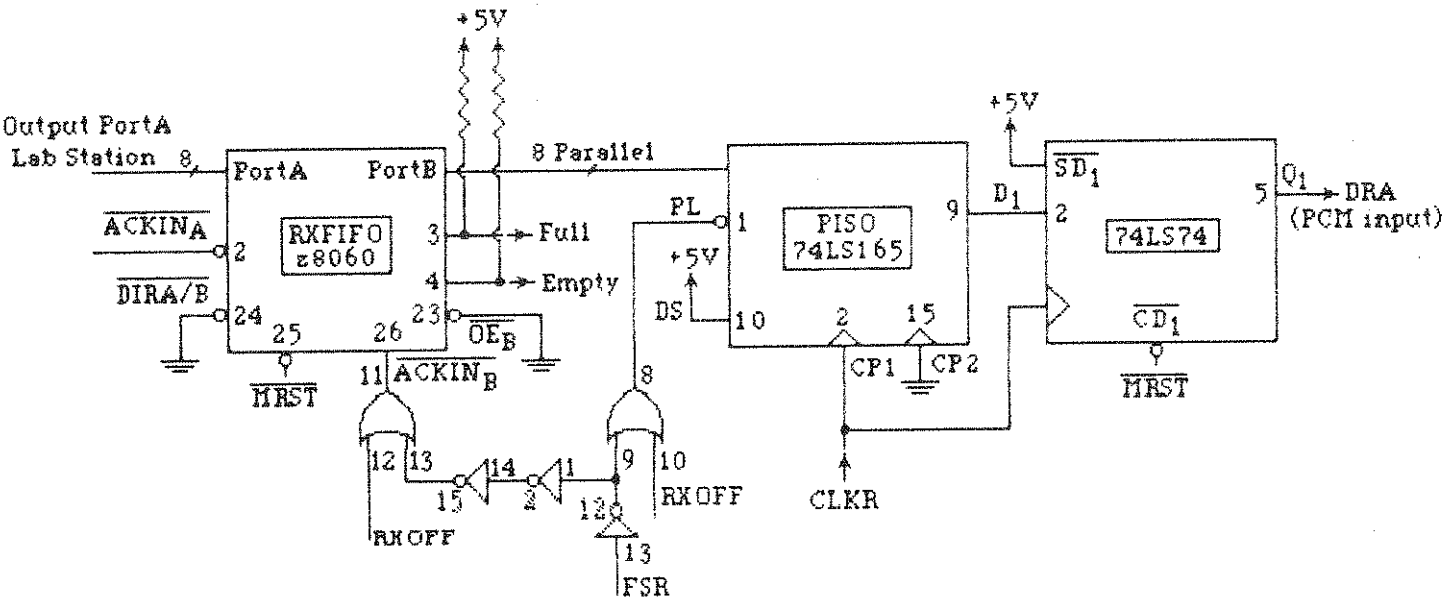


Figure 8. The Unpacketising Circuit.

The important timing sequence is shown below:

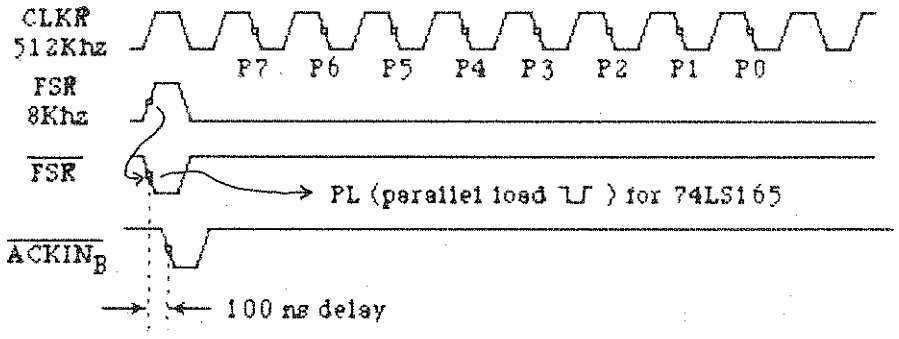


Figure 9. The Timing Sequence of the Unpacketising Circuit.

c. Clock Circuit

The clock circuit consists of a 4.096 MHz crystal oscillator, 3 ICs 74LS161 and a few other components. The clock circuit provides different clock rates for different parts of the hardware prototype. Particular care was taken to ensure that various strict clock requirements are all met.

The diagram of the clock circuit is shown below:

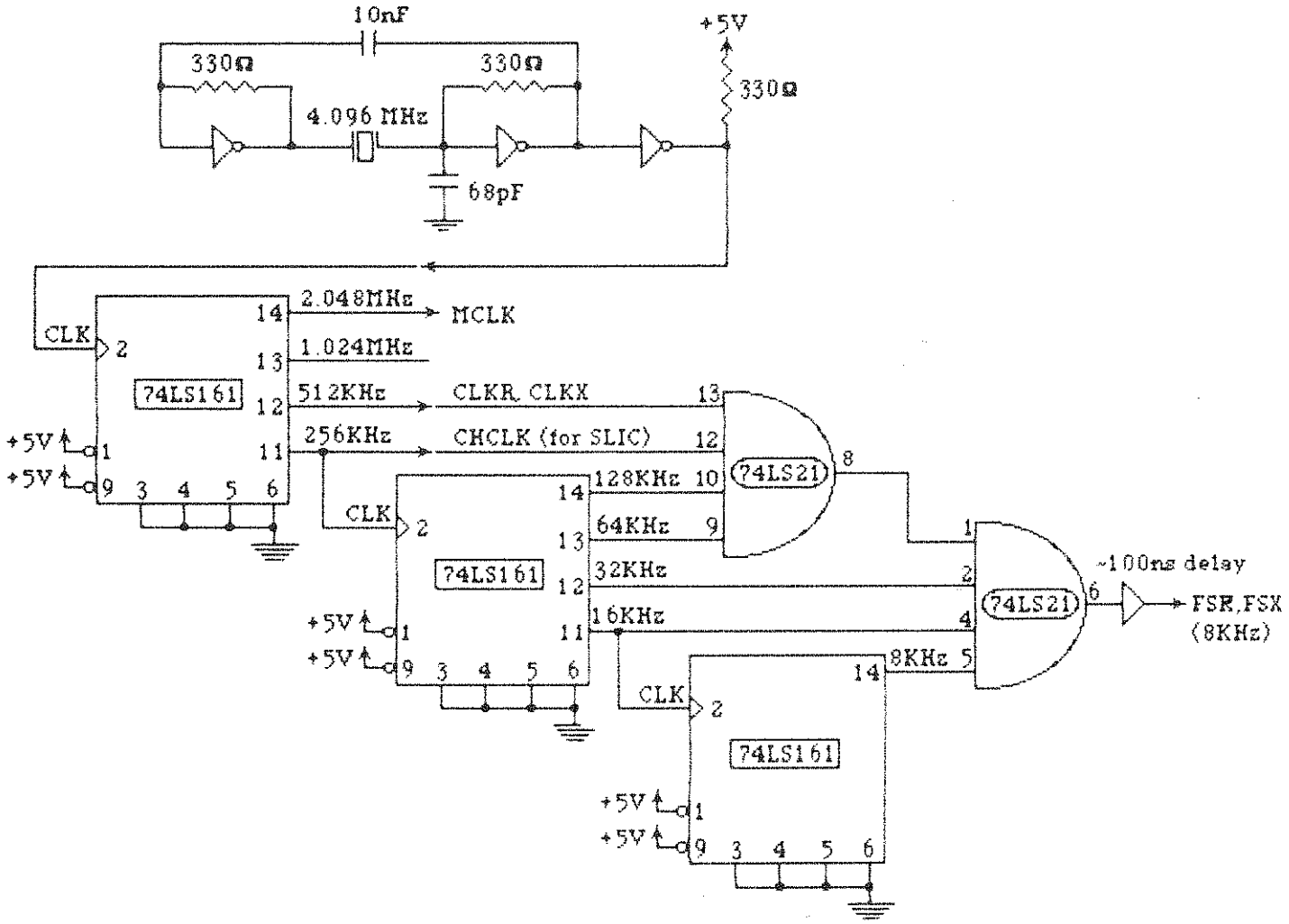


Figure 10. The Clock Circuit.

For the complete diagram of the front-end hardware, please refer to Appendix-C.

2. Software Development

All software is written in Turbo Pascal, with the current prototype card being used on a Z80-based Work Station, the supported software is a CP/M version.

a. The Block Diagram

The software is not only written for the testing, debugging of the prototype, but it also drives the Work Station to control the hardware to perform specific tasks. The tasks are allocated to the blocks (sharp corner boxes), see Figure 11. The states (round coner boxes) indicate a number of options that can be chosen to perform, thus from a state, the user can select a task, tasks or another state to be executed. From the same figure, here are some examples of blocks and states: the Packet Telephone state, the Power-on Test block, the Quit block, the Programming Slic&Slac state, the Simulation state, and so on. Each block or state can be selected by the user with just a few keystrokes from the keyboard.

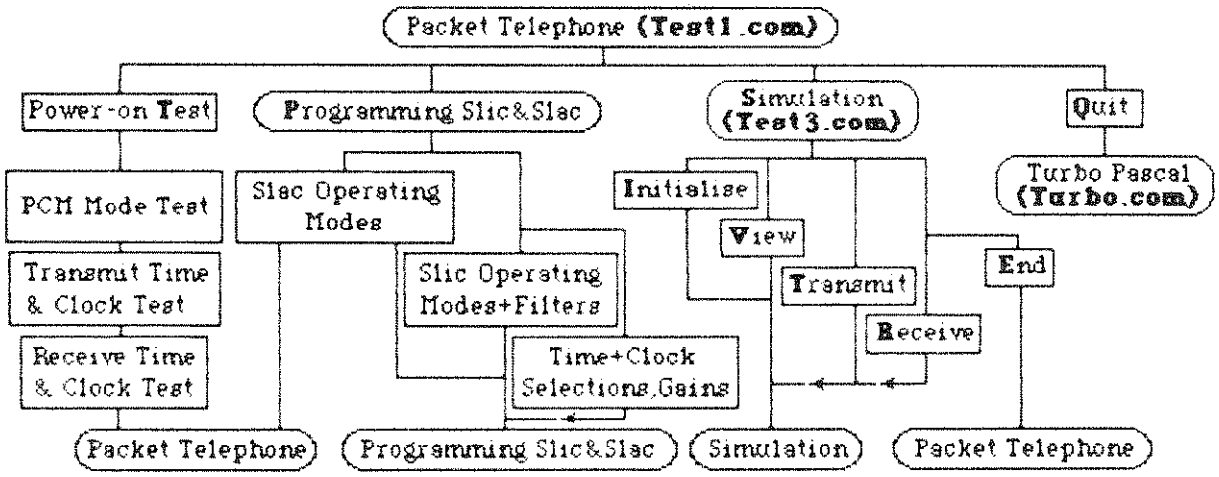


Figure 11. The Block Diagram of the Software.

b. The Packet Telephone State

The Packet Telephone State is the heading of the project, from this block the user can enter the Power_on Test block, Programming Slic&Slac state, Simulation state and Quit block (to quit the system) by entering "T", "P", "S" and "Q" respectively (for all cases, in this program, either upper or lower case letter can be used). Figure 12 shows the screen of the Packet Telephone state.

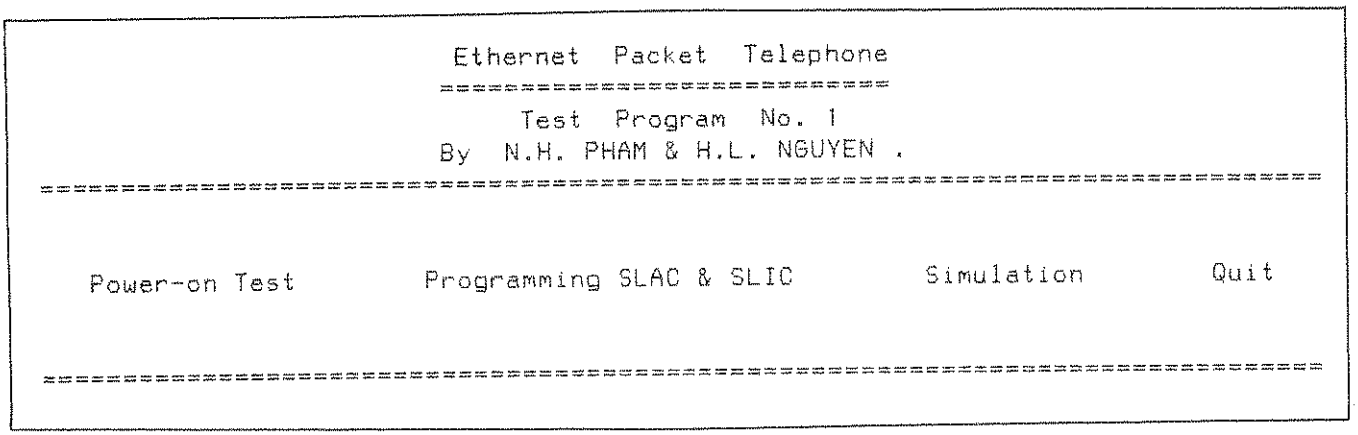


Figure 12. The Packet Telephone Screen.

i. The Power-on Test Block

The Power-on Test block can be invoked from the Packet Telephone state by entering "T". The test must be done before the execution of any block or any state . In this test, there are three sequencially executed sub-tests, the PCM mode test, the Transmit Time & Clock test and the Receive Time & Clock Test. The three tests read three 8-bit words, and compare them to the known values at the time when the power is just on, the results are known immediately after these tests, if any one fails, the user needs to check connections of the prototype.

ii. The Programming Slic&Slac State

Programming Slic&Slac state has many options, they are classified in three classes: the Slac Operating Modes, the Time & Clock Selections + Transmit & Receive Gains, and the Slic

Operating Modes + PCM Modes + Filters. To execute any of these options, the user can use one or a few keystrokes, the followings are the summary of the options and controlled keys.

Slac Operating Modes

W Power down V Add -6dB to Receive Gain E Reset to normal conditions
U Power up C Cutoff Receive Path A Analog loop-back test
I Choose linear F Disable High Pass Filter & freeze Auto Zero Circuit ^A Digital loop-back test ^I Choose micro-law
K Back to the Packet Telephone state.

Time & Clock Selections + Transmit and Receive Gains

T Transmit time slot selection	^T Receive time slot selection
L Transmit clock slot selection	^L Receive time slot selection
D Read transmit Time & Clock slot	^D Read receive Time & Clock Slot
G Transmit gain selection	^G Receive gain selection
N Read Transmit Gain	^N Read receive gain

Note : For selection options, the user needs to supply one or two 3-bit, 5-bit or 8-bit data words to program the hardware into certain mode. For read options, the program shows the data words to the user for verification.

Slic Operating Modes + PCM Modes + Filters

P PCM Mode selection B Write B coefficients ^B Read B coefficients
^P Read PCM mode R Write R coefficients ^R Read R coefficients
O Output to Slic X Write R coefficients ^X Read R coefficients
^O Operating & basic functions. Z Write R coefficients ^Z Read R coefficients

Note : For selection or write options, the user needs to supply one or two 3-bit, 5-bit or 8-bit data words to program the hardware into certain mode. For read options, the program shows the data words to the user for verification.

All the above options from the Programming Slic&Slac state again verify the working of the prototype card. For further information about these options please refer to the data sheets attached at the end of the report.

Figure 13 shows the screen when the program is in the Programming Slic&Slac state, where a number of options can be selected to test the hardware prototype.

```
  W  Power down          V  Add -6dB to Receive Gain    E  Reset to Normal Conditns
  U  Power up            C  Cutoff Receive Path         A  Analog Loop-back test
  I  Choose Linear       F  Disable High Pass Filter    ^A Digital Loop-back test
  ^I Choose u_Law           & Freeze Auto Zero Circt    K  Back to The MAIN menu
=============================================================================
  T  Transmit Time Slot Selection        ^T Receive Time Slot Selection
  L  Transmit Clock Slot Selection       ^L Receive Clock Slot Selection
  D  Read Transmit Time & Clock Slot     ^D Read Receive Time & Clock Slot
  G  Transmit Gain Selection             ^G Receive Gain Selection
  N  Read Transmit Gain                  ^N Read Receive Gain Selection
=============================================================================
  P  PCM Mode Selection            B  Write B Coeffs    ^B Read B Coeffs
  ^P Read PCM Mode                 R  Write R Coeffs    ^R Read R Coeffs
  O  Output to SLIC                X  Write X Coeffs    ^X Read X Coeffs
  ^O Operating & Basic Functions   Z  Write Z Coeffs    ^Z Read Z Coeffs
=============================================================================
```

Figure 13.The Programming Slic&Slac Screen

iii. The Simulation State

The Simulation state consists of the tasks that show how packets are contructed and how they can be converted back to voice signals. The voice signals in digital form coming from the hardware prototype are assembled into packets of 128 bytes each. There are about 330 packets being stored for about 4 seconds of a continuous speech. At any time later, the user can play the recorded speech back and evaluate the voice quality. There are five sub-tasks in the Simulation state, each of the tasks can be called by typing in the

appropriate key. After this, the user will be asked to do some specific actions like: lifting the handset up, hitting any key when ready to talk or listen etc.

The keystrokes and tasks in the Simulation state are:

I Initialize memory V View T Transmit R Receive E End

Figure 14 shows the Simulation screen.

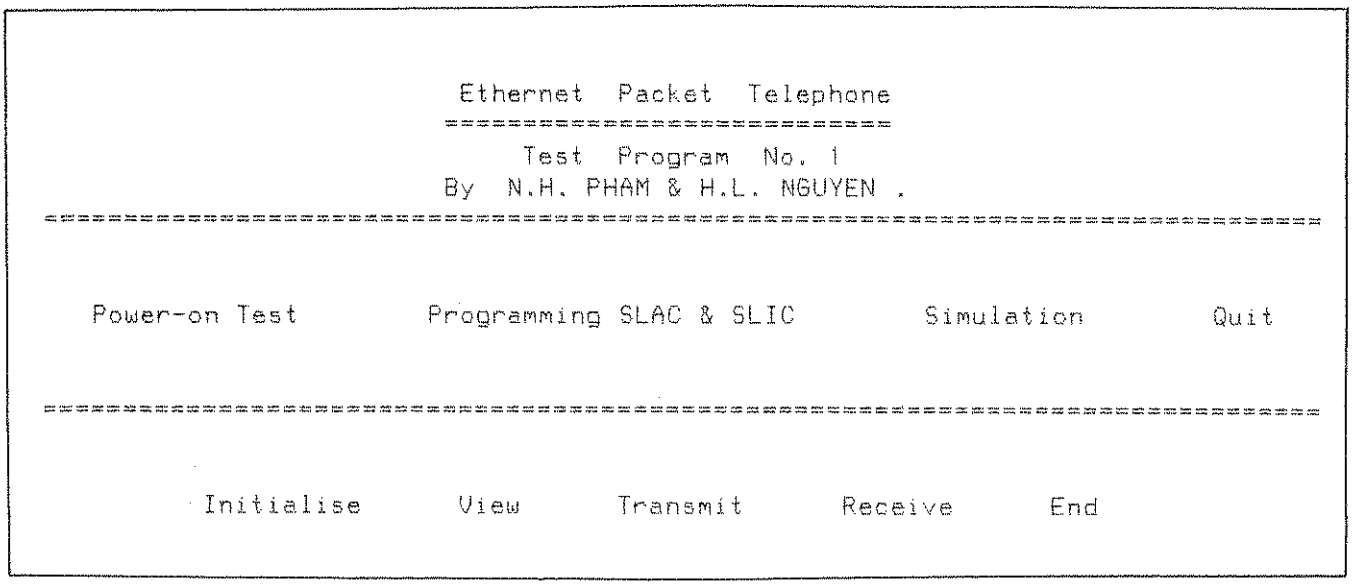


Figure 14. The Simulation Screen, the active screen below the Packet Telephone Screen

o The Initialize Task

The Initialize Memory resets the portion of memory to store the packets to zero for every byte. After this, the user can view the contents of these packets by typing "V", any he will be asked to press any key to see the next packet or to press "S" to stop.

Figure 15 shows the contents of packet number 0, after the Initialise task has been called.

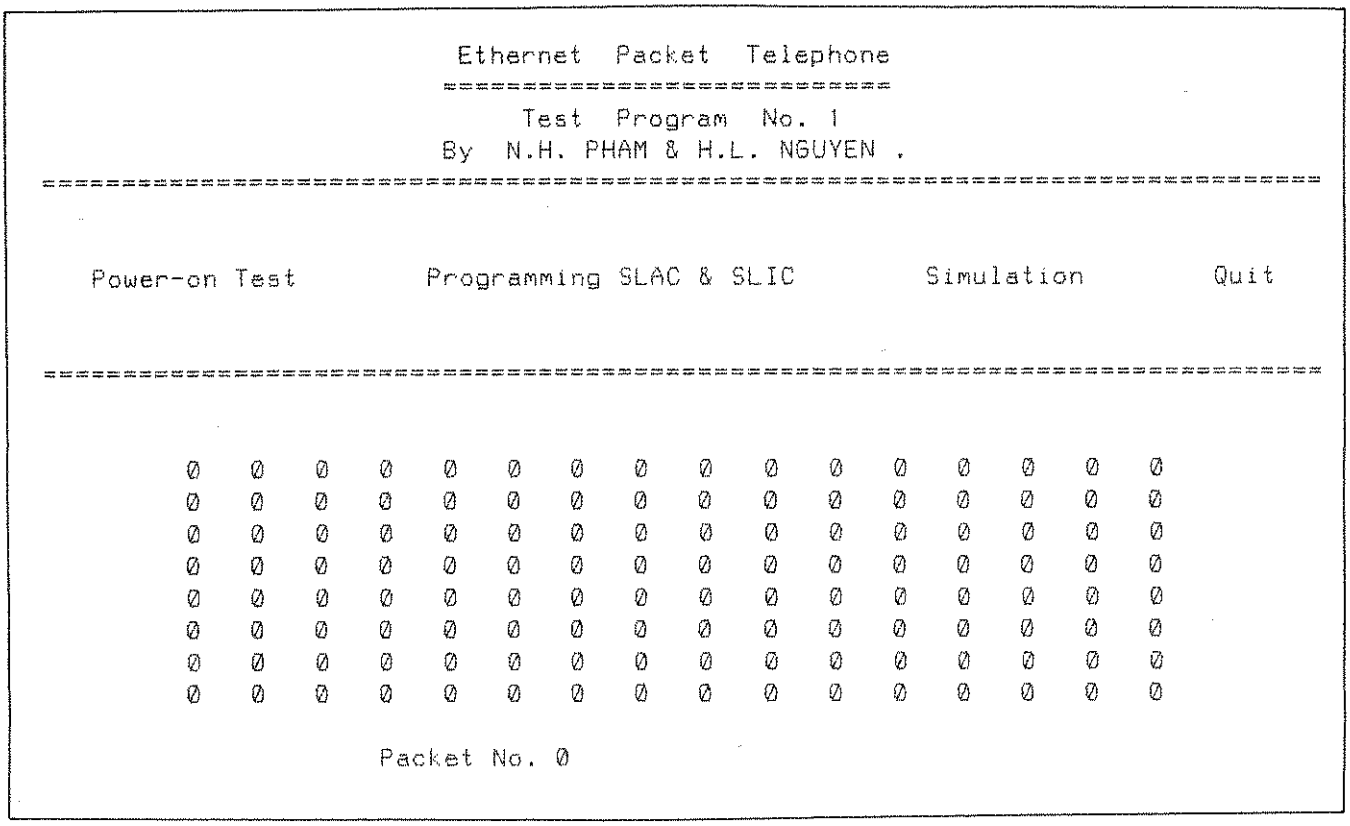


Figure 15. The active screen is showing the contents of packet number 0.

o The Transmit Task

The Transmit task is for testing the transmit path. The user can have a continuous mode or he can be asked to have a duration of speech once (i.e., only 4 seconds). In the continuous mode, he can input sinusoidal waveform and trouble-shoot the circuit if necessary. In the other case, he can see how packets are formed and displayed through the View task.

o The Receive Task

Similarly, for the Receive task, the user can listen to the transmitted speech once or repeatedly. In the former case, he can use it to evaluate the voice quality, then if

necessary, change filter parameters and gains to improve the voice. In the latter case, he can trouble-shoot the receive path of the circuit while the sinusoidal test waveform being stored is continuously playing back.

o The End Task

The End task will bring the user back to the Packet Telephone state, i.e., it clears the active screen below the Packet Telephone screen.

The simulate program is written seperately with the initilization of the Slic and Slac seperated from the Programming Slic&Slac state, i.e. the two states, the Programming Slic&Slac and the Simulation are independent from each other. It is a seperate program written in Turbo Pascal, and seperately compiled to .COM file, it is called from the program Test1.Pas by using EXECUTE statement in Turbo Pascal. Thus, in order to change operating modes, the user must exit the system and modify the program itself (Test3.Pas and compile to Test3.Com file). We suggest that for further software development, the user can modify the program so that it will start with a default operatimg mode, then he can change the operating mode through the use of the Programming Slic&Slac state . When he exits the system, he will be asked to save the operating mode as the next default mode, etc. When this is done, the two above states are no longer independent from each other.

The Quit option of the Packet Telephone state is used to quit the system, and the user will be automatically put into Turbo mode (i.e., after quitting the system, the file Turbo.Com is executed in the same manner as the Test3.Com (Simulation program) mentioned previously). This feature is very useful during software development because it brings the user directly into the Turbo Editor

after he exits from the execution, where he can modify or upgrade the program, etc.

c. Notes on how to program the transmit GX and receive GR filters of the SLAC

As mentioned previouly, the SLAC provides six user-programmable digital filters in the signal processing section. These allow the user to independently modify the gain in both the transmit and receive paths, provide trans-hybrid balancing in the system, and adjust the two-wire line termination impedance. Each programmable filter has the following type of transfer function:

$$H_z = h_0 + h_1 z^{-1} + h_2 z^{-2} + h_n z^{-n}$$

The values of the user-defined coefficients (h_n) are assigned via the serial I/O interface. The number of taps (n) provided depends on the particular filter.

In the experiment, we only had to use the transmit GX and receive GR filters. Following, we describe how these two filters' gain parameters can be derived. However, the description applies for the other four filters as well.

There is only one user-defined coefficient for the GX filter and there is also only one for the GR filter. Effectively, the coefficient is the gain parameter of the filter. The gain parameters are written in or read out as 8-bit words. The format of the parameters is shown below:

C m 40 40	C m 30 30	<--- 1st word
C m 20 20	C m 10 10	<--- 2nd word

where C_{xy} m_{xy} = C is the sign bit and m is the 3-bit code specifying the position of the one in coefficient Y (1 = most significant one, 2 = second one, etc.). and the coefficients in the Equation written previously are described by:

$$h_i = (C_{1i} .2^{-\hat{m}_{1i}} (1 + C_{2i} .2^{-\hat{m}_{2i}} (1 + C_{3i} .2^{-\hat{m}_{3i}} (1 + C_{4i} .2^{-\hat{m}_{4i}}))))$$

for GR filter and

$$h_i = 1 + (C_{1i} .2^{-\hat{m}_{1i}} (1 + C_{2i} .2^{-\hat{m}_{2i}} (1 + C_{3i} .2^{-\hat{m}_{3i}} (1 + C_{4i} .2^{-\hat{m}_{4i}}))))$$

for GX filter and where $\hat{m}_{ij} = 7 - m_{ij}$.

Now, for example, if we want a GX transmit gain of 5, we will have:

$$5 = 1 + 2^{0} (1 + 2^{0} (1 + 2^{0} (1 + 2^{0})))$$

i.e., $C_{10} = 0$, $\hat{m}_{10} = 0$ => $m_{10} = 7$ Or 111 binary

$C_{20} = 0$, $\hat{m}_{20} = 0$ => $m_{20} = 7$ Or 111 binary

$C_{30} = 0$, $\hat{m}_{30} = 0$ => $m_{30} = 7$ Or 111 binary

$C_{40} = 0$, $\hat{m}_{40} = 0$ => $m_{40} = 7$ Or 111 binary

Therefore, the format of the transmit gain GX = 5 is:

0111 0111

0111 0111

Note: The user must not program the SLAC with the codes **0000 0000** and **1111 1111** because these are reserved codes for power-down and power-up. For more detail, please refer to the SLAC data sheets.

B. Period 2: IBM Work Station-based Development (9/86 - 10/86)

Recalling that after the first period, we could not attempt to gain access to the Ethernet in order to experiment voice communication in form of packets because of the inavailability of the concentrator. Therefore, we needed to change the design environment: from CP/M Z80 Work Station to IBM Work Station. This is also necessary because the Department is being equipped with new IBM Work Stations and it seems that inevitably, these IBM Work Stations will replace the CP/M Z80 Work Stations in the long run. Another advantage of the IBM Work Stations is the fact that each Station has (or will have) an add-on Ethernet controller card which allows direct access to the Ethernet Local Area Network.

In this period, it was also planned that the project would have two stages again (Figure 16):

* Stage 1: In this stage, the aim was to transfer the front-end circuit design developed on the Z80 Work Station (in period 1) to the new IBM Work Station. This would involve the design of an Input/Output interface circuit on the IBM PC and some slight modification of the existing hardware prototype.

* Stage 2: In this stage, the aim would be to interface the circuit developed in stage 1 with the add-on Ethernet controller card installed in the IBM Work Station. This would allow voice transmission through the Ethernet in form of packets.

Following, we describe all the work that we managed to complete in this very short and hectic period (end of the year).

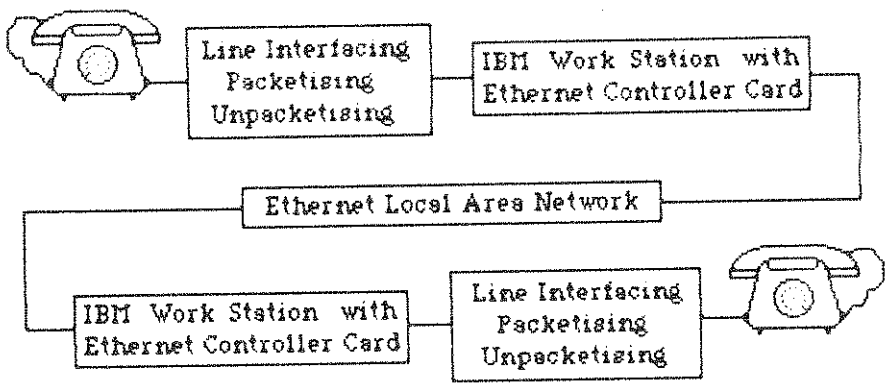


Figure 16. The Block Diagram of the Project in Period 2.

1. Hardware Development

In the very limited time that was available, we designed the Input/Output interface circuit on the IBM PC. Earlier during the first period, we found that we were more or less limited in term of design space because the Z80 Work Station provides only 2 input ports and 2 output ports. Therefore, we aimed to increase the I/O facilities for the IBM prototype design, by providing up to 4 input ports and 4 output ports. These increased I/O facilities will create more design options. For example, it is now possible to easily experiment parallel packet voice communication between 2 IBM Work Stations, something that we could not experiment before on the Z80 Work Station due to the limit of I/O ports.

Basically, the I/O interface circuit was designed and wirewrapped on a standard IBM prototype card, utilising the address decoding circuit provided. In consists mainly of standard TTL devices. For the complete circuit diagram, please refer to Appendix-B. The addresses for the I/O ports are:

Input Port A = 300 Hex

Input Port B = 301 Hex

Input Port C = 302 Hex

Input Port D = 303 Hex

Output Port E = 304 Hex

Output Port F = 305 Hex

Output Port G = 306 Hex

Output Port H = 307 Hex

2. Software Development

The software developed on the CM/P Z80 Work Station (period 1) was slightly modified for the DOS IBM Work Station. Because the software was developed mainly using Turbo Pascal, it can run directly on the IBM except for a few changes in the I/O Port Addresses and the use of OVERLAY instead of EXECUTE. After these modifications, the software was found to function correctly.

IV. FUTURE DEVELOPMENT

In the short term, the following development tasks can be considered and carried out in the order given below:

* Complete the IBM Work Station-based front-end circuit for the Ethernet packet telephone system. This will involve some wire-wrapping work, testing and debugging if necessary.

* Experiment parallel packet voice communication between two IBM Work Stations if two prototypes are available.

* Interface the front-end circuit with the Ethernet controller card .

* Carry out loop-back communication test on the Ethernet packet telephone system.

* Carry out half-duplex, and later full-duplex, communication tests on the Ethernet packet telephone system if two prototypes are available.

We suggest that the above tasks can be the goals for one or two third-year design students in their design contract periods.

In the long term, further work can be considered and carried out to investigate:

* Speech activity dectection (i.e., silence detection) algorithms.

* Packet voice protocol, which includes features such as call initiation and termination, handling of ringing and busy conditions.

* Conferencing techniques.

* Connection to the public switched telephone network or other networks via gateways.

The knowledge and results obtained from the project then may also provide interesting testbed for further experimentation on the possibility of a fully digital telephone, integrated with a microprocessor and a direct Ethernet connection (Figure 17).

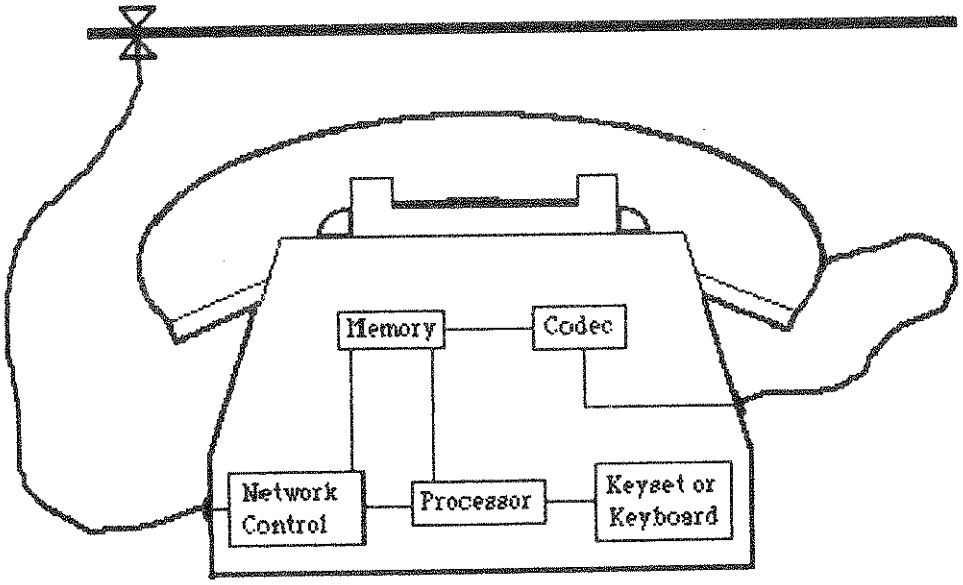


Figure 17. An integrated phone and Etherner interface.

V. CONCLUSION

The project has been very interesting and rewarding. Although we were faced with many difficult problems, especially in term of available support resources and timing and therefore did not have the opportunity to experiment packet voice communication through the Ethernet, we strongly feel that we have learnt a lot from the project and are satisfied with the results that we have achieved.

All the front-end circuit hardware and necessary software for the Ethernet packet telephone system have been completely designed, built, and demonstrated working. The documentation has been carefully written in this report so that from this point, the project can be readily continued in next year by design students or the technician staff.

VI. ACKNOWLEDGEMENT

We would like to sincerely thank Dr. Greg K. Egan for the help and support that he has given us during the whole duration of the project. We would also like to thank the Staff of the Department of Communication and Electronic Engineering RMIT for providing us the opportunity and facilities to carry out the project.

VII. REFERENCES

[1] West, A. & Tanson, P., "LOcal Networks for Computer Communications", North Holland, 1981.

[2] "SLIC Chip Shrinks Phone Line Interfaces", Electronics, June 17, 1985.

[3] Weinstein, C. J. & Forgie, J. W., "Experience with Speech Communication in Packet Networks", IEEE Journal on Selected Areas in Communications, Vol. Sac-1, No. 6, December 1983.

[4] Gruber, J. G. & Le, N. H., "Performance Requirements for Integrated Voice/Data Networks", IEEE Journal on Selected Areas in Communications, Vol. Sac-1, No. 6, December 1983.

[5] Tanenbaum, A., "Computer Networks", Prentice Hall, 1981.

[6] Stella, S., "Voice Communication Over Ethernet", Design Report, RMIT, 1984.

[7] Various Data Sheets and Data Handbooks.

APPENDIX-A: POWER SUPPLIES

Appendix-A: Power Supplies

The hardware prototype has the following power requirements:

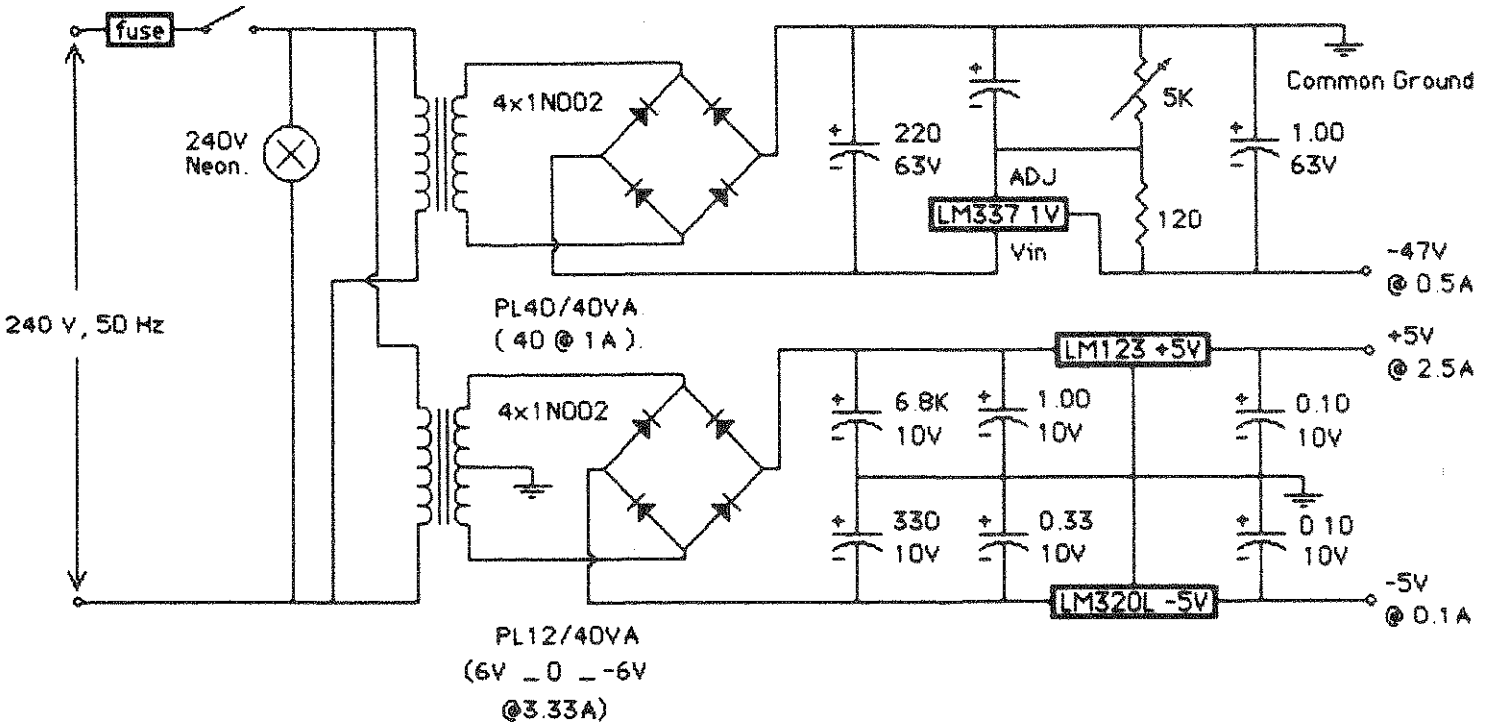
* Supply -45 V to -65 V DC @ low current (about 100 mA) for the telephone set via the SLIC.

* Supply +5 V DC at high current (about 2 A) for all the devices on the prototype board.

* Supply -5 V DC at low current (about 200 mA) for the SLIC and the SLAC.

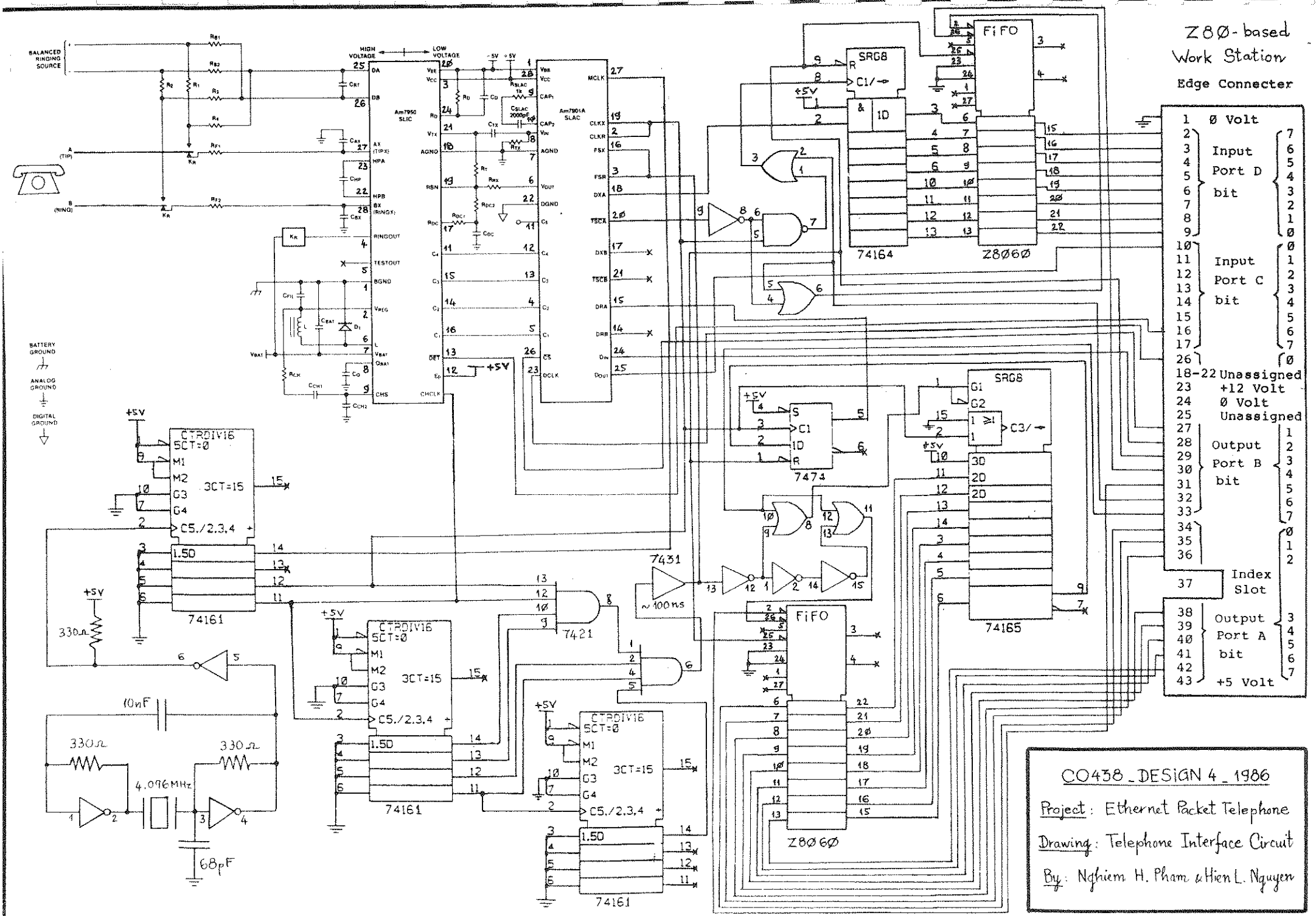
We designed the power supply to meet the above requirements, using mainly 2 Ferguson transformers PL12/40VA, PL40/40VA, 3 regulators LM337HV, LM323K+5V and LM320L-5V. All safety factors such as fuse, switch, thermal heatsinks, earthed chassis were taken into account.

Two power supply units were built and tested. They both function correctly and can be used to provide power for two hardware prototypes. For the complete circuit diagram of the power supply, please see next page.



POWER SUPPLY CIRCUIT.

APPENDIX-B: CIRCUIT DIAGRAMS


Z8Ø-based
Work Station
Edge Connecter
1 Ø Volt
2-9 Input Port D bit
10-17 Input Port C bit
18-22 Unassigned
23 +12 Volt
24 Ø Volt
25 Unassigned
27-33 Output Port B bit
34-36
37 Index Slot
38-43 Output Port A bit
43 +5 Volt
BALANCED RINGING SOURCE
HIGH VOLTAGE
LOW VOLTAGE
Am7950 SLIC
Am7901A SLAC
BATTERY GROUND
ANALOG GROUND
DIGITAL GROUND
SRG8
FIFO
74164
Z8Ø6Ø
7474
7431
~100ns
7421
74165
74161
CTRDIV16
SCT=Ø
3CT=15
C5./2.3.4
1.5Ø
330Ω
10nF
4.096MHz
68pF
CO438_DESIGN 4_1986
Project: Ethernet Packet Telephone
Drawing: Telephone Interface Circuit
By: Nghiem H. Pham & Hien L. Nguyen

Appendix-B: Circuit Diagrams

Z80 Work Station Edge Connector Pin Assignments

* Input Port D:

  8 bits from transmit FIFO

* Input Port C:

  bit 0 ..... DOUT (pin 10)

  bit 1 ..... FULLtx (pin 11)

  bit 2 ..... EMPTYtx (pin 12)

  bit 4 ..... FULLrx (pin 14)

  bit 5 ..... EMPTYrx (pin 15)

  bit 6 ..... not(DET) (pin 16)

* Output Port B:

  bit 0 ..... DCLK (pin 26)

  bit 1 ..... CS (pin 27)

  bit 2 ..... DIN (pin 28)

  bit 3 ..... not(MRST) (pin 29)

  bit 4 ..... not(ACKINb,tx) (pin 30)

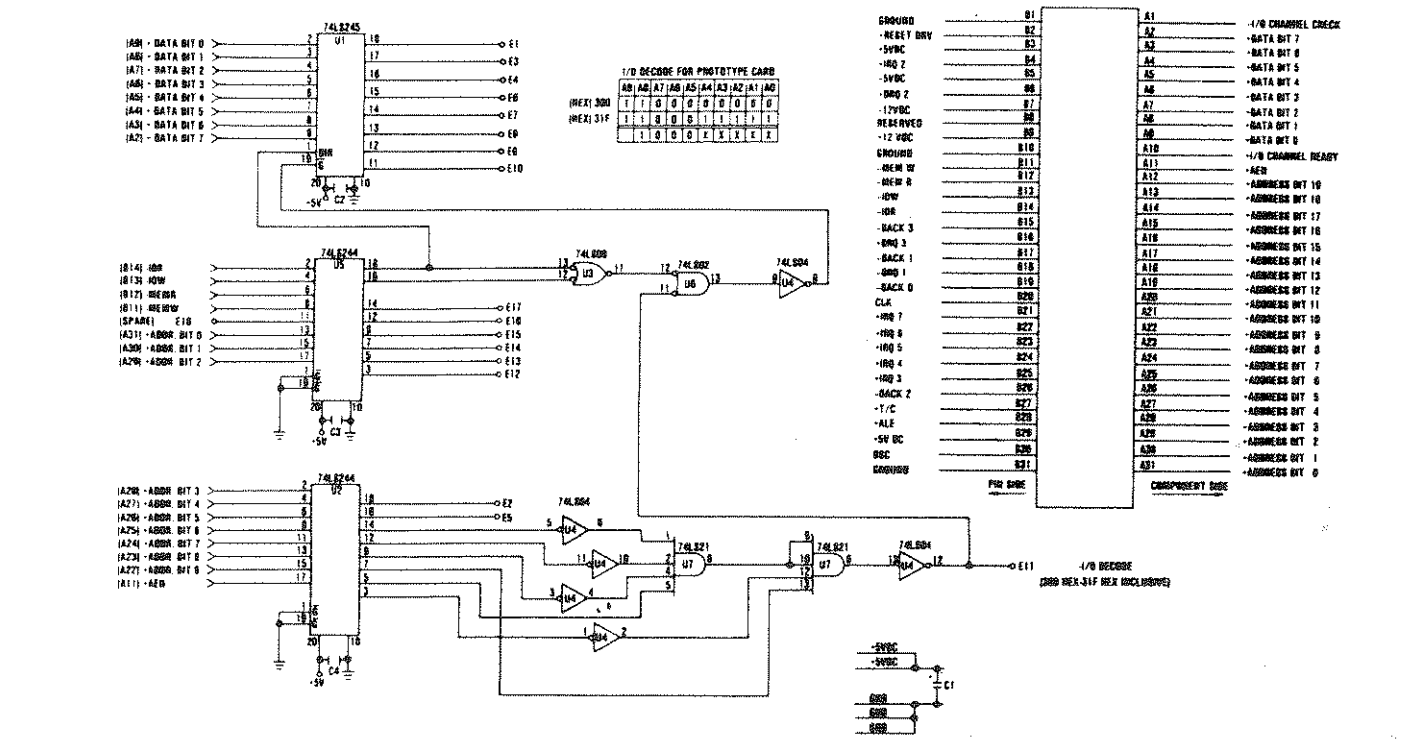
  bit 5 ..... not(ACKINa,rx) (pin 31)

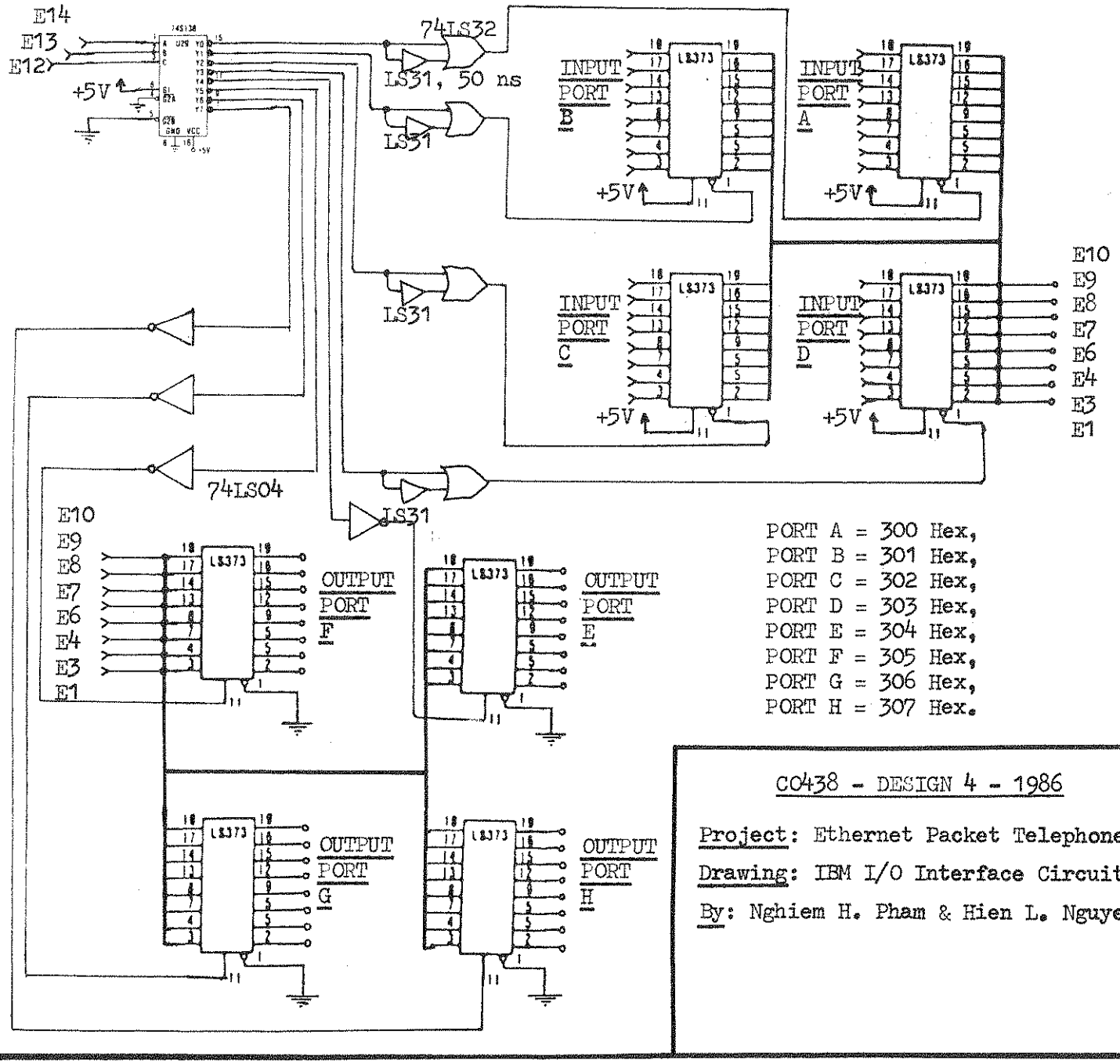
  bit 6 ..... TXOFF (pin 32)

  bit 7 ..... RXOFF (pin 33)

* Output Port A:

  8 bits to receive FIFO





C0438 - DESIGN 4 - 1986

Project: Ethernet Packet Telephone

Drawing: IBM I/O Interface Circuit

By: Nghiem H. Pham & Hien L. Nguyen

APPENDIX-C: PARTS LIST

Integrated Circuits:

1	Am7950
1	Am7901A
2	Z8060PS
5	74LS04
1	74LS08
1	74LS14
1	74LS21
3	74LS31
2	74LS32
1	74LS74
3	74LS161
1	74LS164
1	74LS165
2	74LS244
1	74LS245
8	74LS373
1	LM337HV
1	LM323K+5V
1	LM320L-5V

Diodes:

4	1N4002
4	1N5404

Resistors:

2	20 Ohm, 1%, 1/4 W
8	100 Ohm, 1%, 1 W
1	2.4 KOhm, 5%, 1/4 W
1	560 Ohm. 1%, 1/4 W
2	20 KOhm, 1%,1/4 W
1	10 KOhm. 1%, 1/4 W
1	1 KOhm, 1%, 1/4 W
5	500 KOhm trimpots
1	120 Ohm, 1%, 1 W
2	470 Ohm, 5%, 1/2 W
1	560 Ohm, 5%, 1/2 W
1	10 KOhm, 5%, 1/2 W

Capacitors:

1	330 uF, 10 V, electrolytic
1	220 uF, 63 V, electrolytic
4	1 uF, 10 V, solid tantalum
2	4.7 uF, 10 V, solid tantalum
2	0.22 uF, 20%, 100 V
2	0.002 uF, 20%, 100 V
2	0.47 uF, 20%, 100 V
1	0.33 uF, 20%, 100 V
1	0.0082 uF, 20%, 100 V
1	0.00056 uF, 20%, 100 V
1	0.15 uF, 20%, 10 V
1	3 uF, 20%, 10 V
4	0.01 uF, 20%, 10 V
1	1 uF, 20%, 10 V
20	0.1 uF, ceramic (decoupling caps)

Miscellaneous:

1	PL12/40VA transformer
1	PL40/40VA transformer
1	DS/2/M/48VDC relay
1	1 mH RF choke
4	28-pin w/w sockets
8	14-pin w/w sockets
8	16-pin w/w sockets
8	20-pin w/w sockets
2	TO3 heatsinks
1	TO202 heatsink
1	0.5 A fuse
1	fuse holder
1	Z80 prototype card (or IBM prototype card)
1	4.096 MHz crystal

On average, the cost of one hardware prototype is about **$250**.

APPENDIX-D: SOFTWARE LISTINGS

```
PROGRAM SLAC;

LABEL MAIN,RETURN;

CONST

      OUTA = $42; { Output port A }
      OUTB = $43; { Output port B }
      INC  = $41; { Input port C }
      IND  = $40; { Input port D }
      W0 = 254;
      W1 = 252;
      W2 = 251;
      W3 = 250;
      W4 = 249;
      W5 = 248;
      W6 = 242;
      W7 = 186;
      W8 = 170;
      W9 = 122;
      W10 = 90;
      W11 = 58;

TYPE

      STRING20 = STRING[12];
      STRING8 = STRING[8];
      STRING5 = STRING[5];
      STRING3 = STRING[3];
      STRING4 = STRING[4];

VAR

      I : BYTE;
      ANSWER : CHAR;
      DUMMY : STRING8;
      SIMULATE,TURBO : FILE;

PROCEDURE CLEAR;

      BEGIN
      FOR I := 0 TO 11 DO
           BEGIN
           GOTOXY(1,13 + I);
           CLREOL
           END
      END;

PROCEDURE BRIGHT(X : STRING20);

      BEGIN
      NORMVIDEO;
      WRITE(COPY(X,1,1));
      LOWVIDEO;
      WRITE(COPY(X,2,19));
      NORMVIDEO
      END;
```

```
PROCEDURE CLEARPROGRAMSCREEN;

      BEGIN
      FOR I:= 0 TO 7 DO
           BEGIN
           GOTOXY(1,17 + I);
           CLREOL
           END
      END;

 PROCEDURE MAINDISPLAY;

      BEGIN
      CLRSCR;
      GOTOXY(27,2);
      WRITE('Ethernet  Packet  Telephone');
      GOTOXY(26,3);
      WRITE('============================');
      GOTOXY(31,4);
      WRITE('Test  Program  No. 1');
      GOTOXY(26,5);
      WRITE('By  N.H. PHAM & H.L. NGUYEN .');
      GOTOXY(4,9);
      LOWVIDEO;
      WRITE('Power-on ');
      BRIGHT('Test');
      GOTOXY(25,9);
      BRIGHT('Programming');
      LOWVIDEO;
      WRITE(' SLAC & SLIC');
      GOTOXY(56,9);
      BRIGHT('Simulation');
      GOTOXY(74,9);
      BRIGHT('Quit');
      GOTOXY(1,6);
      FOR I := 1 TO 8 DO
           WRITE('==========');
      GOTOXY(1,12);
      FOR I := 1 TO 8 DO
           WRITE('==========');
      GOTOXY(1,12)
      END; { end of MAINDISPLAY }

 PROCEDURE TESTDISPLAY;

      BEGIN
      GOTOXY(11,15);
      WRITE('PCM Mode Test   __   ');
      GOTOXY(11,18);
      WRITE('Transmit Time & Clock Test  __   ');
      GOTOXY(11,21);
      WRITE('Receive Time & Clock Test  __   ')
      END;
```

```
PROCEDURE PRODISPLAY;

     BEGIN
     CLRSCR;
     GOTOXY(1,5);
     FOR I := 1 TO 8 DO
          WRITE('==========');
     GOTOXY(1,11);
     FOR I := 1 TO 8 DO
          WRITE('==========');
     GOTOXY(1,16);
     FOR I := 1 TO 8 DO
          WRITE('==========');
     NORMVIDEO;
     GOTOXY(3,1);WRITE('W');
     GOTOXY(3,2);WRITE('U');
     GOTOXY(3,3);WRITE('I');
     GOTOXY(3,4);WRITE('^I');
     GOTOXY(3,6);WRITE('T');
     GOTOXY(3,7);WRITE('L');
     GOTOXY(3,8);WRITE('D');
     GOTOXY(3,9);WRITE('G');
     GOTOXY(3,10);WRITE('N');
     GOTOXY(3,12);WRITE('P');
     GOTOXY(3,13);WRITE('^P');
     GOTOXY(3,14);WRITE('O');
     GOTOXY(3,15);WRITE('^O');
     GOTOXY(22,1);WRITE('V');
     GOTOXY(22,2);WRITE('C');
     GOTOXY(22,3);WRITE('F');
     GOTOXY(52,1);WRITE('E');
     GOTOXY(52,2);WRITE('A');
     GOTOXY(52,3);WRITE('^A');
     GOTOXY(52,4);WRITE('K');
     GOTOXY(41,6);WRITE('^T');
     GOTOXY(41,7);WRITE('^L');
     GOTOXY(41,8);WRITE('^D');
     GOTOXY(41,9);WRITE('^G');
     GOTOXY(41,10);WRITE('^N');
     GOTOXY(36,12);WRITE('B');
     GOTOXY(36,13);WRITE('R');
     GOTOXY(36,14);WRITE('X');
     GOTOXY(36,15);WRITE('Z');
     GOTOXY(56,12);WRITE('^B');
     GOTOXY(56,13);WRITE('^R');
     GOTOXY(56,14);WRITE('^X');
     GOTOXY(56,15);WRITE('^Z');
     LOWVIDEO;
     GOTOXY(6,1);WRITE('Power down');
     GOTOXY(6,2);WRITE('Power up');
     GOTOXY(6,3);WRITE('Choose Linear');
     GOTOXY(6,4);WRITE('Choose u_Law');
     GOTOXY(6,6);WRITE('Transmit Time Slot Selection');
     GOTOXY(6,7);WRITE('Transmit Clock Slot Selection');
     GOTOXY(6,8);WRITE('Read Transmit Time & Clock Slot');
     GOTOXY(6,9);WRITE('Transmit Gain Selection');
     GOTOXY(6,10);WRITE('Read Transmit Gain');
```

```
      GOTOXY(6,12);WRITE('PCM Mode Selection');
      GOTOXY(6,13);WRITE('Read PCM Mode');
      GOTOXY(6,14);WRITE('Output to SLIC');
      GOTOXY(6,15);WRITE('Operating & Basic Functions');
      GOTOXY(25,1);WRITE('Add -6dB to Receive Gain');
      GOTOXY(25,2);WRITE('Cutoff Receive Path');
      GOTOXY(25,3);WRITE('Disable High Pass Filter');
      GOTOXY(25,4);WRITE('& Freeze Auto Zero Circt');
      GOTOXY(55,1);WRITE('Reset to Normal Conditns');
      GOTOXY(55,2);WRITE('Analog Loop-back test');
      GOTOXY(55,3);WRITE('Digital Loop-back test');
      GOTOXY(55,4);
      NORMVIDEO;
      WRITE('Back to The MAIN menu');
      LOWVIDEO;
      GOTOXY(44,6);WRITE('Receive Time Slot Selection');
      GOTOXY(44,7);WRITE('Receive Clock Slot Selection');
      GOTOXY(44,8);WRITE('Read Receive Time & Clock Slot');
      GOTOXY(44,9);WRITE('Receive Gain Selection');
      GOTOXY(44,10);WRITE('Read Receive Gain Selection');
      GOTOXY(39,12);WRITE('Write B Coeffs');
      GOTOXY(39,13);WRITE('Write R Coeffs');
      GOTOXY(39,14);WRITE('Write X Coeffs');
      GOTOXY(39,15);WRITE('Write Z Coeffs');
      GOTOXY(59,12);WRITE('Read B Coeffs');
      GOTOXY(59,13);WRITE('Read R Coeffs');
      GOTOXY(59,14);WRITE('Read X Coeffs');
      GOTOXY(59,15);WRITE('Read Z Coeffs');
      NORMVIDEO
      END; { end of PRODISPLAY }

PROCEDURE OUT(CODE : STRING8);
{ write control data to the SLAC }

VAR
      X : CHAR;
      X1,X2,X3,X4,X5,X6,X7,X8 : BYTE;

      BEGIN
      FOR I := 1 TO 8 DO
            BEGIN
            X := COPY(CODE,I,1);
            IF X = '1' THEN
                  CASE I OF
                        1:X1 := W1;
                        2:X2 := W1;
                        3:X3 := W1;
                        4:X4 := W1;
                        5:X5 := W1;
                        6:X6 := W1;
                        7:X7 := W1;
                        8:X8 := W1
                  END
            ELSE
                  CASE I OF
                        1:X1 := W5;
                        2:X2 := W5;
```

```
                    3:X3 := W5;
                    4:X4 := W5;
                    5:X5 := W5;
                    6:X6 := W5;
                    7:X7 := W5;
                    8:X8 := W5
                END;
            END;
        PORT [OUTB] := W3;
        PORT [OUTB] := W2;
        PORT [OUTB] := W3;
        PORT [OUTB] := W2;
        PORT [OUTB] := W3;
        PORT [OUTB] := W5;
        PORT [OUTB] := X1;
        PORT [OUTB] := W4;
        PORT [OUTB] := W5;
        PORT [OUTB] := X2;
        PORT [OUTB] := W4;
        PORT [OUTB] := W5;
        PORT [OUTB] := X3;
        PORT [OUTB] := W4;
        PORT [OUTB] := W5;
        PORT [OUTB] := X4;
        PORT [OUTB] := W4;
        PORT [OUTB] := W5;
        PORT [OUTB] := X5;
        PORT [OUTB] := W4;
        PORT [OUTB] := W5;
        PORT [OUTB] := X6;
        PORT [OUTB] := W4;
        PORT [OUTB] := W5;
        PORT [OUTB] := X7;
        PORT [OUTB] := W4;
        PORT [OUTB] := W5;
        PORT [OUTB] := X8;
        PORT [OUTB] := W4;
        PORT [OUTB] := W5;
        PORT [OUTB] := W3;
        PORT [OUTB] := W3;
        END; { end of OUT }

PROCEDURE IINN(VAR CODE : STRING8);
{ read control data from the SLAC }

VAR
        X : ARRAY[1..8] OF BYTE;

        BEGIN
        PORT [OUTB] := W3;
        PORT [OUTB] := W2;
        PORT [OUTB] := W3;
        PORT [OUTB] := W2;
        PORT [OUTB] := W3;
        PORT [OUTB] := W5; { CS goes LOW }
```

```
      { serial I/O interface starts }
      X[1] := PORT [INC];
      PORT [OUTB] := W4;
      PORT [OUTB] := W5;
      X[2] := PORT [INC];
      PORT [OUTB] := W4;
      PORT [OUTB] := W5;
      X[3] := PORT [INC];
      PORT [OUTB] := W4;
      PORT [OUTB] := W5;
      X[4] := PORT [INC];
      PORT [OUTB] := W4;
      PORT [OUTB] := W5;
      X[5] := PORT [INC];
      PORT [OUTB] := W4;
      PORT [OUTB] := W5;
      X[6] := PORT [INC];
      PORT [OUTB] := W4;
      PORT [OUTB] := W5;
      X[7] := PORT [INC];
      PORT [OUTB] := W4;
      PORT [OUTB] := W5;
      X[8] := PORT [INC];
      PORT [OUTB] := W4;
      PORT [OUTB] := W5;
      PORT [OUTB] := W3; { CS goes HIGH }
      PORT [OUTB] := W3;
      FOR I := 8 DOWNTO 1 DO
           IF ODD(X[I]) THEN CODE := CONCAT('1',CODE)
           ELSE CODE := CONCAT('0',CODE)
      END; { end of IINN }

PROCEDURE POWERONTEST;

VAR
      R : STRING8;

      BEGIN
      CLEAR;
      TESTDISPLAY;
      GOTOXY(30,15);
      OUT('01110111');
      IINN(R);
      IF R='11110000' THEN
           WRITE('PASSED')
      ELSE
           WRITE('FAILED at ',R,' c/w 11110000');
      GOTOXY(43,18);
      OUT('01110101');
      IINN(R);
      IF R= '00000000' THEN
           WRITE('PASSED')
      ELSE
           WRITE('FAILED at ',R,' c/w 00000000');
      GOTOXY(42,21);
      OUT('01111101');
      IINN(R);
```

```
      IF R = '00000000' THEN
           WRITE('PASSED')
      ELSE
           WRITE('FAILED at ',R,' c/w 00000000')
      END; { end of POWERONTEST }

PROCEDURE PROGRAMSLICSLAC;

LABEL
      PROIO,EXITIO;

      PROCEDURE ENTRY(X:BYTE);

      VAR
           J:BYTE;

           BEGIN
           CLEARPROGRAMSCREEN;
           FOR J := 1 TO X DO
                BEGIN
                IF J <= 8 THEN
                     BEGIN
                     GOTOXY(10,16 + J);
                     WRITE('Enter Data Word No. ',J,': ');
                     END
                ELSE
                     BEGIN
                     GOTOXY(42,16 - 8 + J);
                     WRITE('Enter Data Word No. ',J,': ');
                     END;
                READ(DUMMY);
                OUT(DUMMY);
                END;
           END; { end of ENTRY }

      PROCEDURE COLLECT(X:BYTE);

      VAR
           J:BYTE;

           BEGIN
           CLEARPROGRAMSCREEN;
           FOR J := 1 TO X DO
                BEGIN
                IF J <= 8 THEN
                     BEGIN
                     GOTOXY(10,16 + J);
                     WRITE('Data Word No. ',J,': ');
                     END
                ELSE
                     BEGIN
                     GOTOXY(42,16 - 8 + J);
                     WRITE('Data Word No. ',J,': ');
                     END;
                IINN(DUMMY);
                WRITE(DUMMY);
                END
           END; { end of COLLECT }
```

```
VAR
     TERM5 : STRING5;
     TERM4 : STRING4;
     TERM3 : STRING3;

     BEGIN { PROGRAMSLICSLAC }
     PRODISPLAY;
     PROIO:
     WHILE  TRUE DO
          IF KEYPRESSED THEN
               BEGIN
               READ(KBD,ANSWER);
               IF NOT(ANSWER IN [CHR( 9), CHR( 1), CHR( 4), CHR( 7),
                           CHR( 2), CHR(20), CHR(12), CHR(14),
                           CHR(15), CHR(16), CHR(18), CHR(24),
                           CHR(26)]) THEN
                    CASE ANSWER OF
                         'W','w' : BEGIN
                                   CLEARPROGRAMSCREEN;
                                   OUT('00000000');
                                   GOTOXY(10,20);
                                   WRITE('Power Down Mode is Done.');
                                   END;
                        'U','u' : BEGIN
                                   CLEARPROGRAMSCREEN;
                                   OUT('11111111');
                                   GOTOXY(10,20);
                                   WRITE('Power Up Mode is Done');
                                   END;
                        'I','i' : BEGIN
                                   CLEARPROGRAMSCREEN;
                                   OUT('10111000');
                                   GOTOXY(10,20);
                                   WRITE('The Linear Code is Chosen');
                                   END;
                        'V','v' : BEGIN
                                   CLEARPROGRAMSCREEN;
                                   OUT('10110001');
                                   GOTOXY(10,20);
                                   WRITE('- 6 dB (minus 6 dB) is added
                                             to Receive Gain');
                                   END;
                        'C','c' : BEGIN
                                   CLEARPROGRAMSCREEN;
                                   OUT('10110010');
                                   GOTOXY(10,20);
                                   WRITE('The Receive  Path  is  Cut-
                                             off ');
                                   END;
                        'F','f' : BEGIN
                                   CLEARPROGRAMSCREEN;
                                   OUT('10110011');
                                   GOTOXY(10,20);
                                   WRITE('The  High Pass Filter & Auto
                                             Zero Circuit are Disable');
                                   END;
```

```
'E','e' : BEGIN
          CLEARPROGRAMSCREEN;
          OUT('10110000');
          GOTOXY(10,20);
          WRITE('Normal  Conditions  (  Reset
                Test Modes; Receive Path,');
          GOTOXY(30,21);
          WRITE('High Pass Filter & Auto Zero
                Circuit operate )');
          END;
'A','a' : BEGIN
          CLEARPROGRAMSCREEN;
          OUT('10110111');
          GOTOXY(10,20);
          WRITE('Analog    Loop-back     Test
                operates');
          END;
'K','k' : GOTO EXITIO;
'T','t' : BEGIN
          CLEARPROGRAMSCREEN;
          GOTOXY(10,20);
          WRITE('Time  Slot Selection,  Enter
                5-bit Word : ');
          READ(TERM5);
          OUT(CONCAT('001',TERM5));
          END;
'L','l' : BEGIN
          CLEARPROGRAMSCREEN;
          GOTOXY(10,20);
          WRITE('Clock Slot Selection,  Enter
                3-bit Word : ');
          READ(TERM3);
          OUT(CONCAT('01100',TERM3));
          END;
'D','d' : BEGIN
          CLEARPROGRAMSCREEN;
          OUT('01110101');
          GOTOXY(10,20);
          WRITE('Transmit   Time   &    Clock
                Slot : ');
          IINN(DUMMY);
          WRITE(DUMMY);
          END;
'G','g' : BEGIN
          CLEARPROGRAMSCREEN;
          OUT('01110010');
          GOTOXY(10,20);
          WRITE('Transmit   Gain   Selection,
                Enter Word No. 1 : ');
          READ(DUMMY);
          OUT(DUMMY);
          GOTOXY(35,21);
          WRITE('Enter Word No. 2 : ');
          READ(DUMMY);
          OUT(DUMMY);
          END;
```

```
'N','n' : BEGIN
          CLEARPROGRAMSCREEN;
          OUT('01110001');
          GOTOXY(10,20);
          WRITE('Transmit Gain,Word No.1:');
          IINN(DUMMY);
          WRITE(DUMMY);
          GOTOXY(25,21);
          WRITE('Word No. 2 : ');
          IINN(DUMMY);
          WRITE(DUMMY);
          END;
'B','b' : BEGIN
          CLEARPROGRAMSCREEN;
          OUT('10100000');
          GOTOXY(10,20);
          WRITE('12   8-bit  Words  expected,
                    Ready ? Any Key ');
          READ(KBD,ANSWER);
          ENTRY(12);
          END;
'R','r' : BEGIN
          CLEARPROGRAMSCREEN;
          OUT('10101000');
          GOTOXY(10,20);
          WRITE('8   8-bit  Words   expected,
                    Ready ? Any Key ');
          ENTRY(8);
          END;
'X','x' : BEGIN
          CLEARPROGRAMSCREEN;
          OUT('10100100');
          GOTOXY(10,20);
          WRITE('8   8-bit  Words   expected,
                    Ready ? Any Key ');
          ENTRY(8);
          END;
'Z','z' : BEGIN
          CLEARPROGRAMSCREEN;
          OUT('10101100');
          GOTOXY(10,20);
          WRITE('8   8-bit  Words   expected,
                    Ready ? Any Key ');
          ENTRY(8);
          END;
'P','p' : BEGIN
          CLEARPROGRAMSCREEN;
          GOTOXY(10,20);
          WRITE('PCM Mode Selection, Enter 4-
                    bit Word : ');
          READ(TERM4);
          OUT(CONCAT('1001',TERM4));
          END;
```

```
                    'O','o' : BEGIN
                              CLEARPROGRAMSCREEN;
                              GOTOXY(10,20);
                              WRITE('SLIC Operations, Enter 5-bit
                                    Word : ');
                              READ(TERM5);
                              OUT(CONCAT('110',TERM5));
                              END;
               END{ end of CASE }
          ELSE
               IF ANSWER IN  [CHR(16), CHR(15), CHR( 2), CHR(18),
               CHR(24), CHR(26)] THEN
                    IF ANSWER IN  [CHR(16), CHR(15), CHR(2)] THEN
                         IF ANSWER = CHR(16) THEN
                              BEGIN
                              CLEARPROGRAMSCREEN;
                              OUT('01110111');
                              GOTOXY(10,20);
                              WRITE('PCM Mode, 8-bit Word : ');
                              IINN(DUMMY);
                              WRITE(DUMMY);
                              END
                             ELSE IF ANSWER = CHR(15) THEN
                              BEGIN
                              CLEARPROGRAMSCREEN;
                              GOTOXY(10,20);
                              WRITE('Operating & Basic  fuctions,
                                    Enter a 4-bit Word : ');
                              READ(TERM4);
                              OUT(CONCAT('1000',TERM4));
                              END
                                  ELSE
                              BEGIN
                              CLEARPROGRAMSCREEN;
                              OUT('10100011');
                              GOTOXY(10,20);
                              WRITE('12   8-bit  Words  expected,
                                    Ready ? Any Key ');
                              COLLECT(12);
                              END
                         ELSE IF ANSWER = CHR(18) THEN
                              BEGIN
                              CLEARPROGRAMSCREEN;
                              OUT('10101011');
                              GOTOXY(10,20);
                              WRITE('8 8-bit Words expected,  Ready  ?
                                     Any Key ');
                              COLLECT(8);
                              END
                              ELSE IF ANSWER = CHR(24) THEN
                                   BEGIN
                                   CLEARPROGRAMSCREEN;
                                   OUT('10100111');
                                   GOTOXY(10,20);
                                   WRITE('8   8-bit  Words   expected,
                                         Ready ? Any Key ');
```

```
                              COLLECT(8);
                              END
                              ELSE
                              BEGIN
                              CLEARPROGRAMSCREEN;
                              OUT('10101111');
                              GOTOXY(10,20);
                              WRITE('8   8-bit  Words   expected,
                                        Ready ? Any Key ');
                              COLLECT(8);
                              END
          ELSE  IF ANSWER IN  [CHR(20),  CHR(12),  CHR(  7)]
               THEN
                    IF ANSWER = CHR(20) THEN
                              BEGIN
                              CLEARPROGRAMSCREEN;
                              GOTOXY(10,20);
                              WRITE('Receive Time Slot Selection,
                                        Enter a 5-bit Word : ');
                              READ(TERM5);
                              OUT(CONCAT('010',TERM5));
                              END
                         ELSE IF ANSWER = CHR(12) THEN
                              BEGIN
                              CLEARPROGRAMSCREEN;
                              GOTOXY(10,20);
                              WRITE('Receive   Clock  Slot  Selection,
                                        Enter a 3-bit Word : ');
                              READ(TERM3);
                              OUT(CONCAT('01101',TERM3));
                              END
                              ELSE
                              BEGIN
                              CLEARPROGRAMSCREEN;
                              OUT('01111010');
                              GOTOXY(10,20);
                              WRITE('Receive  Gain  Selection,   Enter
                                        Word No. 1 : ');
                              READ(DUMMY);
                              OUT(DUMMY);
                              GOTOXY(34,21);
                              WRITE('Enter Word No. 2 : ');
                              READ(DUMMY);
                              OUT(DUMMY);
                              END
     ELSE IF ANSWER IN  [CHR( 9), CHR( 1)] THEN
          IF ANSWER = CHR( 9) THEN
               BEGIN
               CLEARPROGRAMSCREEN;
               OUT('10111001');
               GOTOXY(10,20);
               WRITE('u-Law is chosen.');
               END
          ELSE
               BEGIN
               CLEARPROGRAMSCREEN;
               OUT('10110100');
               GOTOXY(10,20);
               WRITE('Digital Loop-back Test Operates');
               END
```

```
                ELSE IF ANSWER = CHR( 4 ) THEN
                     BEGIN
                     CLEARPROGRAMSCREEN;
                     OUT('01111101');
                     GOTOXY(10,20);
                     WRITE('Receive Time & Clock Slot : ');
                     IINN(DUMMY);
                     WRITE(DUMMY);
                     END
                     ELSE
                     BEGIN
                     CLEARPROGRAMSCREEN;
                     OUT('01111001');
                     GOTOXY(10,20);
                     WRITE('Receive Gain, Word No. 1 : ');
                     IINN(DUMMY);
                     WRITE(DUMMY);
                     GOTOXY(24,21);
                     WRITE('Word No. 2 : ');
                     IINN(DUMMY);
                     WRITE(DUMMY);
                     END;
                GOTO PROIO;
                END; { end of KEYPRESSED }
     EXITIO:
     END; { end of PROGRAMSLICSLAC }


BEGIN { Main Program }
PORT [OUTB] := W0;
MAIN:
MAINDISPLAY;
RETURN:
WHILE  TRUE DO
     IF KEYPRESSED THEN
          BEGIN
          READ(KBD,ANSWER);
          CASE ANSWER OF
               'T','t' : BEGIN
                         POWERONTEST;
                         GOTO RETURN
                         END;
               'P','p' : BEGIN
                         PROGRAMSLICSLAC;
                         GOTO MAIN
                         END;
               'S','s' : BEGIN
                         ASSIGN(SIMULATE,'TEST3.COM');
                         EXECUTE(SIMULATE);
                         END;
               'Q','q' : BEGIN
                         ASSIGN(TURBO,'TURBO.COM');
                         EXECUTE(TURBO);
                         END;
               END;
          END;
END. { end of Main program }
```

```
PROGRAM SIMULATION;

LABEL
     INTIALIZE,VIEW,TRANSMIT,RECEIVE,ARRANGE;

CONST
     OUTA = $42;
     OUTB = $43;
     INC  = $41;
     IND  = $40;
     W0 = 254;
     W1 = 252;
     W2 = 251;
     W3 = 250;
     W4 = 249;
     W5 = 248;
     W6 = 242;
     W7 = 186;
     W8 = 170;
     W9 = 122;
     W10 = 90;
     W11 = 58;

TYPE
     STRING8 = STRING[8];

VAR
     I,K,L : BYTE;
     J : INTEGER;
     ANSWER1,ANSWER2 : CHAR;
     SPEECH : ARRAY[0..330,1..128] OF BYTE;
     MAIN : FILE;

     PROCEDURE CLEAR;

          BEGIN
          FOR I := 0 TO 11 DO
               BEGIN
               GOTOXY(1,13 + I);
               CLREOL
               END
          END;

     PROCEDURE OUT(CODE : STRING8);
     { write control data to the SLAC }

     VAR
          X : CHAR;
          X1,X2,X3,X4,X5,X6,X7,X8 : BYTE;

          BEGIN
          FOR I := 1 TO 8 DO
               BEGIN
               X := COPY(CODE,I,1);
```

```
            IF X = '1' THEN
                  CASE I OF
                        1:X1 := W1;
                        2:X2 := W1;
                        3:X3 := W1;
                        4:X4 := W1;
                        5:X5 := W1;
                        6:X6 := W1;
                        7:X7 := W1;
                        8:X8 := W1
                  END
            ELSE
                  CASE I OF
                        1:X1 := W5;
                        2:X2 := W5;
                        3:X3 := W5;
                        4:X4 := W5;
                        5:X5 := W5;
                        6:X6 := W5;
                        7:X7 := W5;
                        8:X8 := W5
                  END;
            END;
      PORT [OUTB] := W3;
      PORT [OUTB] := W2;
      PORT [OUTB] := W3;
      PORT [OUTB] := W2;
      PORT [OUTB] := W3;
      PORT [OUTB] := W5;
      PORT [OUTB] := X1;
      PORT [OUTB] := W4;
      PORT [OUTB] := W5;
      PORT [OUTB] := X2;
      PORT [OUTB] := W4;
      PORT [OUTB] := W5;
      PORT [OUTB] := X3;
      PORT [OUTB] := W4;
      PORT [OUTB] := W5;
      PORT [OUTB] := X4;
      PORT [OUTB] := W4;
      PORT [OUTB] := W5;
      PORT [OUTB] := X5;
      PORT [OUTB] := W4;
      PORT [OUTB] := W5;
      PORT [OUTB] := X6;
      PORT [OUTB] := W4;
      PORT [OUTB] := W5;
      PORT [OUTB] := X7;
      PORT [OUTB] := W4;
      PORT [OUTB] := W5;
      PORT [OUTB] := X8;
      PORT [OUTB] := W4;
      PORT [OUTB] := W5;
      PORT [OUTB] := W3;
      PORT [OUTB] := W3;
END; { end of OUT }
```

```
PROCEDURE SEND;

     BEGIN
     PORT [OUTB] := W11;
     FOR J := 0 TO 330 DO
          FOR K := 1 TO 128 DO
               BEGIN
                    INLINE($00/
                           $00/
                           $00/
                           $00/
                           $00/
                           $00/
                           $00/
                           $00/
                           $00); { NOP }
                    SPEECH[J,K] := PORT [IND];
                    PORT [OUTB] := W8;
                    PORT [OUTB] := W7
               END;
     PORT [OUTB] := W3;
     END; { end of SEND }

PROCEDURE FETCH;

     BEGIN
     PORT [OUTB] := W9;
     FOR J := 0 TO 330 DO
          FOR K := 1 TO 128 DO
               BEGIN
                    INLINE($00/
                           $00/
                           $00/
                           $00/
                           $00/
                           $00/
                           $00); { NOP }
                    PORT [OUTA] := SPEECH[J,K];
                    PORT [OUTB] := W10;
                    PORT [OUTB] := W9;
               END;
     PORT [OUTB] := W3;
     END; { end of FETCH }

BEGIN { of Simulation }
OUT('10111001'); { Choose u law }
OUT('01110010'); { Transmit gain Selection }
OUT('01110111'); { Transmit gain = 5 }
OUT('01110111');
OUT('01111010'); { Receive gain Selection }
OUT('01110111'); { Receive gain = 4 }
OUT('01110111');
GOTO ARRANGE;
```

```
INTIALIZE:
FOR J := 0 TO 330 DO
     FOR K := 1 TO 128 DO
          SPEECH[J,K] := 0;
GOTOXY(11,20);
WRITE('Initialisation is done          ');
DELAY(1000);
GOTO ARRANGE;
VIEW:
CLEAR;
GOTOXY(14,24);
WRITE('<S> to Stop, Any Key to Continue');
FOR J := 0 TO 330 DO
     BEGIN
     GOTOXY(55,24);
     WRITE('Packet No. ',J);
     FOR L := 0 TO 7 DO
          FOR K := 0 TO 15 DO
               BEGIN
               GOTOXY(7 + K*4 ,15 + L);
               WRITE(SPEECH[J,L*16 + K+1]:4);
               END;
     READ(KBD,ANSWER1);
     IF ANSWER1 IN  ['S','s'] THEN
     GOTO ARRANGE;
     END;
TRANSMIT:
PORT [OUTB] := W3;
CLEAR;
OUT('00000000'); { Power Down }
OUT('11011010'); { Normal Mode }
GOTOXY(11,17);
WRITE('Continuous OR Once ? <C,O>');
READ(KBD,ANSWER1);
GOTOXY(11,19);
WRITE('Please lift the handset UP');
WHILE  (PORT [INC] AND 64)<>0 DO; { Poll For Det Low }
PORT [OUTB] := W6;
PORT [OUTB] := W3;
GOTOXY(11,21);
WRITE('Hit any KEY to Talk');
READ(KBD,ANSWER2);
OUT('11111111'); { Power Up }
IF ANSWER1 IN  ['C','c'] THEN
     WHILE  NOT KEYPRESSED DO
          SEND
ELSE
     SEND;
GOTOXY(11,23);
WRITE('330 Packets have been stored');
DELAY(3000);
GOTO ARRANGE;
RECEIVE:
PORT [OUTB] := W3;
```

```
CLEAR;
GOTOXY(11,15);
WRITE('Simulation of a call, (Ringing...)');
OUT('11001001'); { Ringing Mode }
WHILE  (PORT [INC] AND 64)<>0 DO { Poll For Det low }
     BEGIN
     FOR J := 1 TO 30 DO
          WRITE(CHR(7));
     DELAY(2000);
     END;
CLEAR;
GOTOXY(11,17);
WRITE('Continuous OR Once ? <C,O>');
READ(KBD,ANSWER1);
GOTOXY(11,19);
WRITE('Hit any Key, When ready to listen');
PORT [OUTB] := W6;
PORT [OUTB] := W3;
READ(KBD,ANSWER2);
OUT('11011010'); { Normal Mode }
OUT('11111111'); { Power Up }
IF ANSWER1 IN  ['C','c'] THEN
     WHILE  NOT KEYPRESSED DO
          FETCH
ELSE
     FETCH;
ARRANGE:
OUT('00000000'); { Power Down }
CLEAR;
GOTOXY(11,15);
WRITE('Initialise          View          Transmit          Receive
          End');
WHILE  TRUE DO
     IF KEYPRESSED THEN
          BEGIN
          READ(KBD,ANSWER1);
          CASE ANSWER1 OF
               'I','i' : GOTO INTIALIZE;
               'V','v' : GOTO VIEW;
               'T','t' : GOTO TRANSMIT;
               'R','r' : GOTO RECEIVE;
               'E','e' : BEGIN
                         ASSIGN(MAIN,'TEST1.COM');
                         EXECUTE(MAIN);
                         END
               END
          END
END. { end of Simulation }
```

APPENDIX-E: DATA SHEETS

Am7950

Subscriber Line Interface Circuit

ADVANCED INFORMATION

DISTINCTIVE CHARACTERISTICS

- Programmable line feed impedance
- Programmable loop detect threshold
- Line feed characteristics independent of battery variations
- On-chip switching regulator for low power dissipation
- Low standby power
- 2-wire impedance set by single external impedance

GENERAL DESCRIPTION

The Am7950 Subscriber Line Interface Circuit (SLIC) performs the telephone line interface functions required in both Central Office and PABX environments. The full range of signal transmission, battery feed and loop supervision functions are performed. Signal transmission performance is compatible with North American and CCITT recommendations. Overvoltage protection and ringing are provided by means of external networks.

The signal transmission functions include both 2- to 4-wire and 4- to 2-wire conversion. The 2-wire termination impedance is programmable with a single external impedance, which may be complex. The companion Am7901A SLAC (Subscriber Line Audio Processing Circuit) has a digital balancing filter that provides the trans-hybrid loss function. If the SLAC is not used, most codec/filter sets provide an uncommitted op amp which may be used for this purpose.

The battery feed architecture makes the DC feed resistance programmable with external resistors. Furthermore, the open circuit feed voltage and the feed resistance are independent of battery variations. Loop currents up to 70mA are recommended, although higher loop currents are possible.

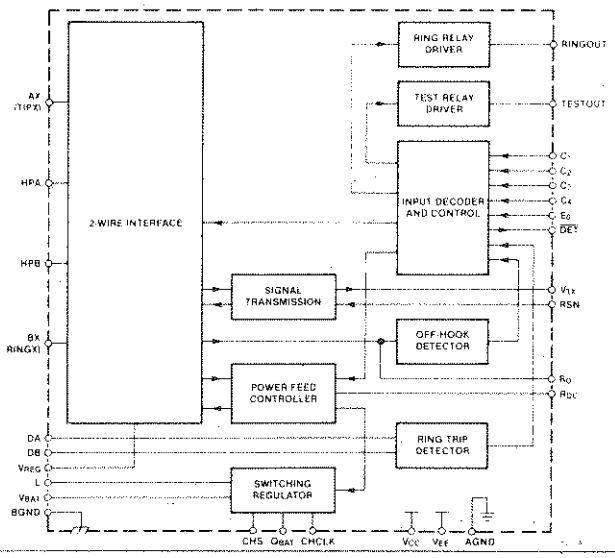
A Polarity Reversal function is provided which transposes the normal voltage sense of the A(TIP) and B(RING) leads with a controlled transition time. All transmission functions continue normally following the transition. A disable mode which limits loop current at 1.5 times the user programmable loop detector threshold current cuts power dissipation while allowing the full complement of supervisory functions to be utilized. The output amplifiers are powered by an internal switching regulator in order to also reduce power consumption.

The supervisory functions of off-hook detection and ring trip detection are read through a single, TTL compatible output. To eliminate noise induced errors, the off-hook detector signal may be filtered and has a threshold adjusted by means of external components. Additional supervisory functions put the A(TIP) lead into an open circuit or high impedance state suitable for application in ground start systems. Similarly, both the A(TIP) and B(RING) leads may be open circuited to clear relays or recover from line faults. Two relay drivers are provided for the Test and Ring relay functions.

The SLIC's user programmable states are controlled by a four-bit TTL compatible digital code. These control inputs are designed to easily interface to popular single chip microcomputers such as the AMD Am8051.

BLOCK DIAGRAM

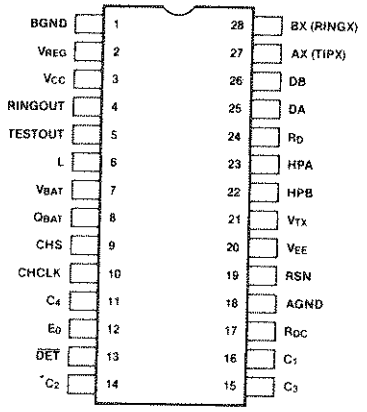
Am7950—SUBSCRIBER LINE INTERFACE CIRCUIT (SLIC)



Order #05203A

June 1984

CONNECTION DIAGRAM—Top View



05203A-14

INTERFACE SIGNAL DESCRIPTION

V_{CC}: +5V power supply

V_{EE}: −5V power supply

AGND: Analog (quiet) and digital ground

BGND: Battery (power) ground

V_{BAT}: Battery supply

AX(TIPX): Output of A(TIP) power amplifier

BX(RINGX): Output of B(RING) power amplifier

HPA: A(TIP) side of high-pass filter capacitor

HPB: B(RING) side of high-pass filter capacitor

RSN: **Receive summing node** The metallic current (both DC and AC) between A(TIP) and B(RING) is equal to 1000 times the current into this pin. The networks which program receive gain, 2-wire impedance, and feed resistance all connect to this node.

V_{TX}: **Transmit audio output** This output is a unity gain version of the AX(TIPX) and BX(RINGX) metallic voltage. The other end of the 2-wire input impedance programming network connects here.

E_0: **Read Enable** A logic high enables $\overline{DET}$. A logic low disables $\overline{DET}$.

$\overline{DET}$: **Detector out** When enabled, a logic low indicates that the selected detector is tripped. The detector is selected by the logic inputs (C_1–C_4). The output is open collector with a built-in pull-up resistor.

R_D: Threshold modification and filter point for the off-hook detector. Also sets current in disable mode to 1.5 times the off-hook threshold.

R_{DC}: Connection point for DC feed resistance programming network. The other end of the network connects to the receiver summing node (RSN). $|V_{RDC}| = (|V_{HPA} - V_{HPB}| - 50V)/20$. The sign of V_{RDC} is minus for normal polarity and plus for reverse polarity.

CHS: Chopper stabilization input.

CHCLK: Chopper clock input to switching regulator. TTL compatible. Frequency = 256kHz (nominal).

L: **Switching Regulator power transistor output** Connection point for 1.0mH inductor and anode of catch diode. This pin will have a 60V pulse waveform on it. Extreme care must be taken to keep the diode connections short because of the high currents and high di/dt.

V_{REG}: **Regulated voltage input** Provides negative power supply for power amplifiers. Connection point for inductor, filter capacitor, and chopper stabilization.

Q_{BAT}: Filtered battery supply for the signal processing circuits.

C_1, C_2, C_3, C_4: **Decoder inputs** TTL compatible. C_4 is MSB and C_1 is LSB.

RINGOUT: **Output of ring relay driver** 25mA sourcing from V_{CC}.

TESTOUT: **Output of test relay driver** 25mA sourcing from V_{CC}.

DA: Positive input to ring trip comparator.

DB: Negative input to ring trip comparator.

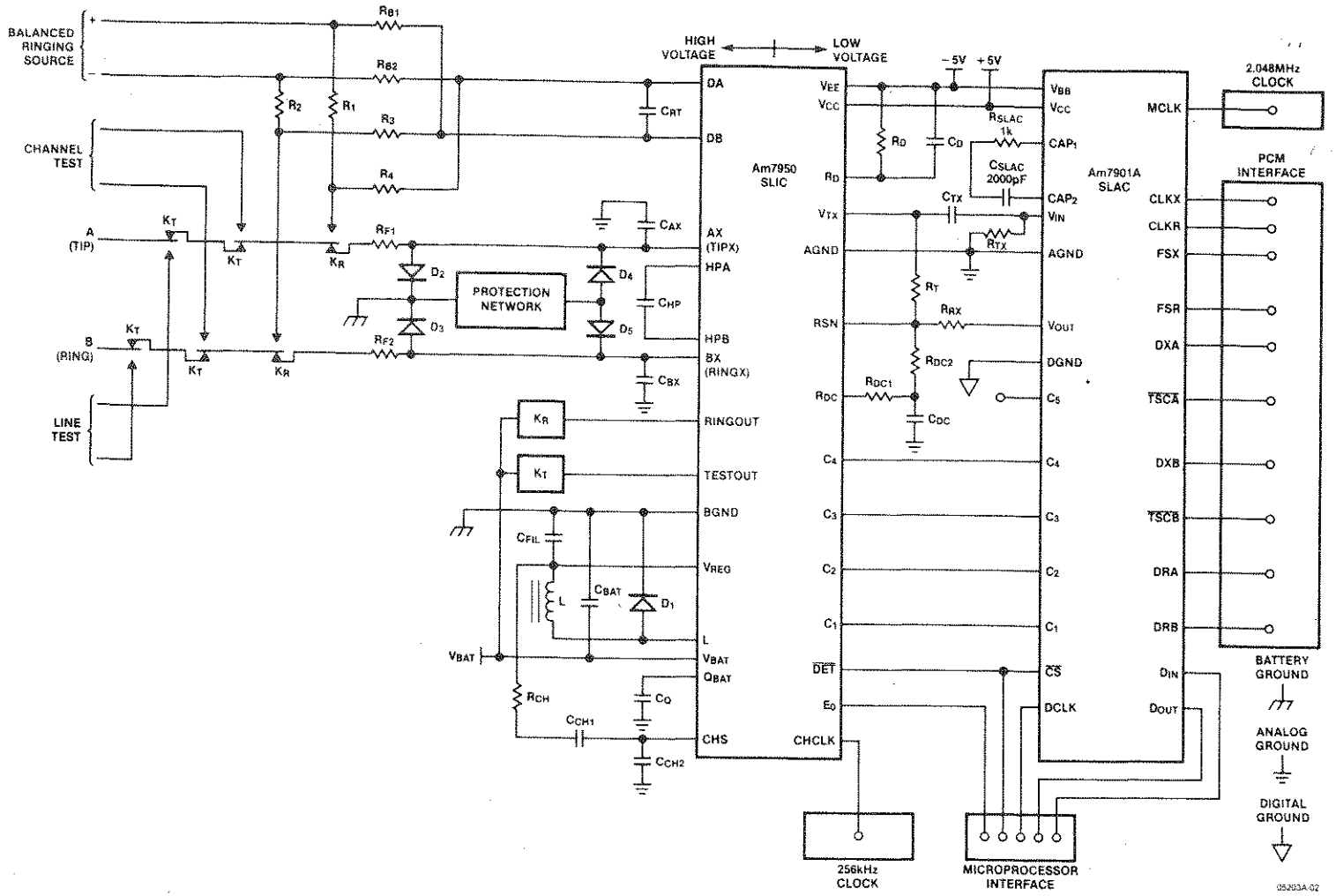


Figure 1. Single Channel Subscriber Line System (See TABLE 3 for component values.)

DEVICE OPERATION

General

The Am7950 performs the subscriber line interface functions at the 2- to 4-wire interface in Central Office and PABX environments. When used with the Am7901A SLAC, the SLIC provides a complete solution of BORSHT functions (see Figure 1). The internal operation of the SLIC is summarized in the block diagram at the beginning of this document. The following sections describe in detail the operation of each block in the block diagram.

2-WIRE INTERFACE

The 2-wire interface (see Figure 2) consists of two current mode amplifiers, voltage sensing circuits with AC/DC pass separation, and a loop current sensing circuit.

The current mode amplifiers which drive the AX(TIPX) and BX(RINGX) pins are controlled by two input signals, I_{LI} and I_{MI}. I_{LI} controls the longitudinal (common mode) current, and I_{MI} controls the metallic (transverse) current. The 2-wire currents are:

$$I_{AX} = 1000(I_{LI} + I_{MI}) \text{ and } I_{BX} = 1000(I_{LI} - I_{MI})$$

I_{MI} is equal to the current into the receive summing node (RSN), which is the terminating point for the external networks controlling 2-wire impedance, receive gain, and battery feed. These networks are described in detail in later paragraphs.

The voltage sense signal which goes to the signal transmission block (V_{ACMET}) is the AC metallic component of the AX and BX voltages.

Two voltage sense signals go to the power feed controller block. One (V_{DCMET}) is the DC metallic component of the AX and BX voltages. The other voltage sense signal (V_{LONG}) is the longitudinal component of the AX and BX voltages. An external capacitor (C_{HP}) connected between HPA and HPB separates the AC and DC components of the metallic voltage. The recommended value of capacitance is 0.22μF, which corresponds to a separation frequency of 2.25Hz. Since this frequency would be too low during polarity reversal and pulse dialing, the 2-wire interface decreases the time constant during these events.

The loop current sensing circuit produces a current output to the R_D pin which is proportional to the magnitude of the loop current. An external resistor and filter capacitor connected from R_D to V_{EE} converts this current to a filtered voltage for use by the off-hook detector. It is also used to control the AX and BX currents in the disable state.

SIGNAL TRANSMISSION

Figures 3a and 3b provide a more detailed diagram of the SLIC transmission path. This path is split between the signal transmission block and the 2-wire interface block shown previously in Figure 2.

The AC line voltage is sensed by differential amplifiers between the AX(TIPX) and HPA leads and between the HPB and BX(RINGX) leads. The outputs of these amplifiers are equal to the AC components of the line voltages. These voltages are summed and buffered by the op amp at V_{TX}.

The differential amplifiers reject the longitudinal voltages so that they do not affect V_{TX}. The external filter capacitor between HPA and HPB eliminates the DC components of the line voltage.

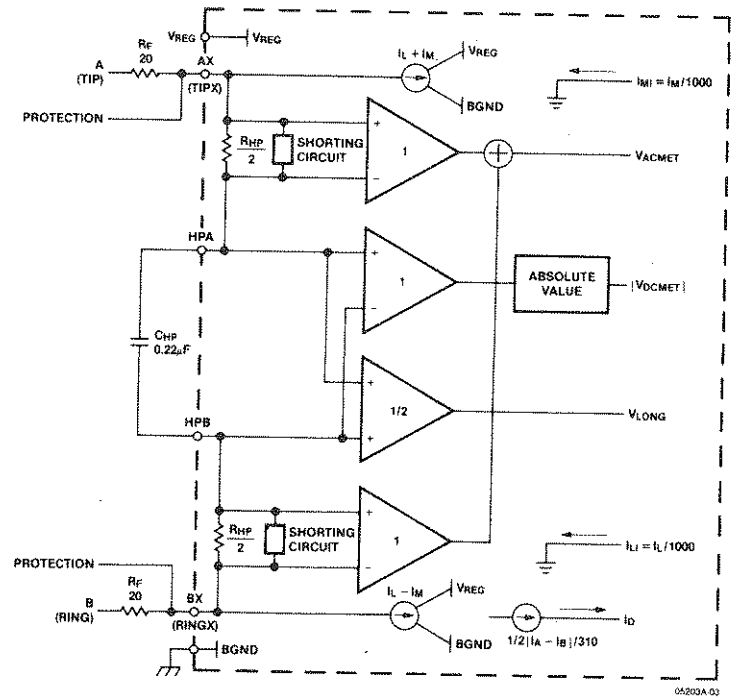


Figure 2. Am7950—SLIC 2-Wire Interface

The SLIC's 2-wire AC input impedance is programmable by means of an external impedance that is connected between RSN and V_{TX} (see Figure 3a). This impedance may be a complex R-C network and should be 1000 times the desired 2-wire input impedance minus the fuse resistors. This means that resistors get 1000 times larger while capacitors become 1000 times smaller.

$Z_T = 1000\,(Z_{2WIN} - 2R_F)$ where Z_{2WIN} = desired impedance

To ensure good insertion loss, controlled gain paths provide 2- to 4-wire and 4- to 2-wire conversion. The 4-wire to 4-wire path, or Balance Return Signal, is specified, both in amplitude and phase, to allow superior trans-hybrid loss to be realized. The Balance Return Signal on V_{TX} exhibits 180° phase shift with respect to V_{RX}. The 4-wire output is found on the V_{TX} terminal and the 4-wire input terminal is V_{RX} (see Figure 3b). Both of these ports are referred to analog ground (AGND).

Because the fuse resistors are outside the feedback loops, they influence the effective gains. These gains are as follows:

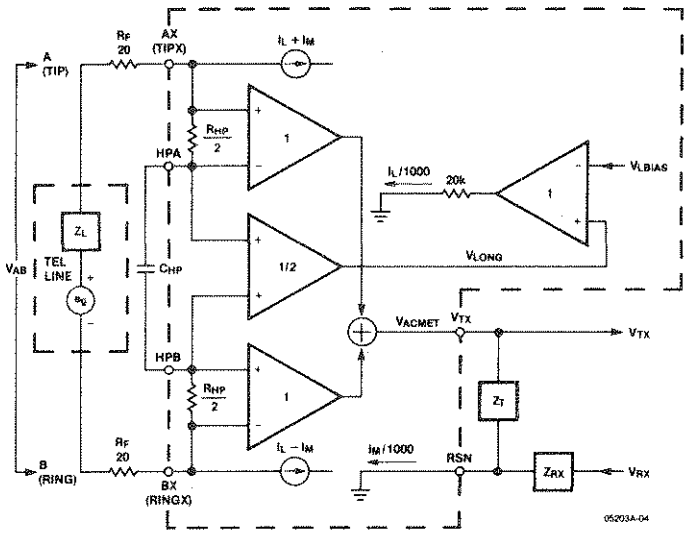
$$G_{4\text{-}2} = \frac{V_{AB}}{V_{RX}}\,[e_g = 0] = \frac{-Z_T}{Z_{RX}}\,\frac{Z_L}{Z_L + 2R_F + Z_T/1000}$$

$$G_{2\text{-}4} = \frac{V_{TX}}{V_{AB}}\,[V_{RX} = 0] = \frac{Z_T/1000}{2R_F + Z_T/1000}$$

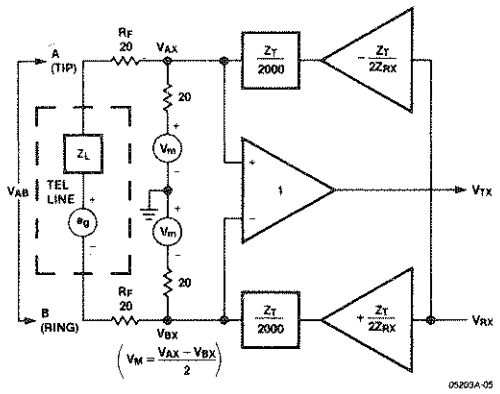
$$G_{4\text{-}4} = \frac{V_{TX}}{V_{RX}}\,[e_g = 0] = \frac{-Z_T}{Z_{RX}}\,\frac{Z_L + 2R_F}{Z_L + 2R_F + Z_T/1000}$$

Typically, $R_F = 20\Omega$ and $Z_T = 2Z_{RX}$. The Am7901A Subscriber Line Audio Processing Circuit can be used to adjust $G_{4\text{-}2}$ and $G_{2\text{-}4}$ to unity and to adjust $G_{4\text{-}4}$ to zero. The user should refer to the Am7901A data sheet for details.

The transmission circuit also contains a longitudinal feedback circuit to shunt longitudinal signals to a DC bias voltage (V_{LBIAS}) which comes from the power feed controller. Longitudinally, the SLIC appears as 20Ω resistors from the AX(TIPX) and BX(RINGX) pins to V_{LBIAS}. The longitudinal feedback circuit does not affect metallic signals.



3a). Detailed Model



3b). Simplified Model (AC Only)

Figure 3. Am7950—SLIC Signal Transmission

POWER FEED CONTROLLER

The power feed controller has three sections: (1) the battery feed circuit, (2) the polarity reversal circuit, and (3) the bias circuit. These are shown in Figure 4. The detailed model is shown in Figure 4a and the simplified model is shown in Figure 4b.

The battery feed circuit produces a voltage at the R_{DC} pin whose magnitude is equal to $(50V - |V_{DCMET}|)/20$ and whose sign depends on the battery polarity; minus for normal polarity and plus for reverse polarity. V_{DCMET} is the DC component of the voltage between AX and BX (TIPX and RINGX). The low-pass filter formed by R_{HP} and C_{HP} attenuates frequencies above 2.25Hz, so that the battery feed circuit does not affect voice band transmission. The loop current is equal to 1000 times the current into the receive summing node (RSN), which is equal to the voltage on R_{DC} divided by $R_{DC1} + R_{DC2}$. The net result is that the SLIC appears to have an open circuit voltage of 50V and a feed resistance, R_{FEEDX}, equal to $(R_{DC1} + R_{DC2})/50$; thus, the feed resistance is programmable, but the open circuit voltage is not. The feed resistance seen by the telephone line also includes the fuse resistors, R_F. The total feed resistance is then:

$$R_{FEED} = 2R_F + (R_{DC1} + R_{DC2})/50$$

For example, to achieve a feed resistance of 840Ω with 20Ω fuse resistors requires that:

$$R_{DC1} + R_{DC2} = 50(840 - 2 \times 20) = 40K\Omega$$

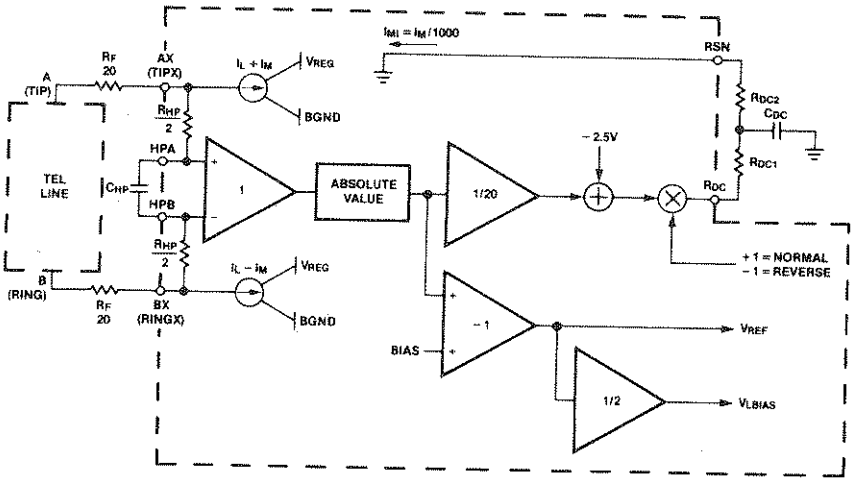
The values of the programming resistors, R_{DC1} and R_{DC2}, should be kept nearly equal. In the example, values of $R_{DC1} = 20K\Omega$ and $R_{DC2} = 20K\Omega$ could be used.

To obtain a polarity reversal, the input decoder and control circuit sends a signal which reverses the sign of the voltage on the R_{DC} pin. The circuit also reduces R_{HP} during the transition, because its time constant is too large. The transition time is actually controlled by the capacitor, C_{DC}, and the parallel resistance of R_{DC1} and R_{DC2}. This time constant should be 1.5ms. In the example where $R_{DC1} = 20K\Omega$ and $R_{DC2} = 20K\Omega$, C_{DC} should be $0.15\mu F$.

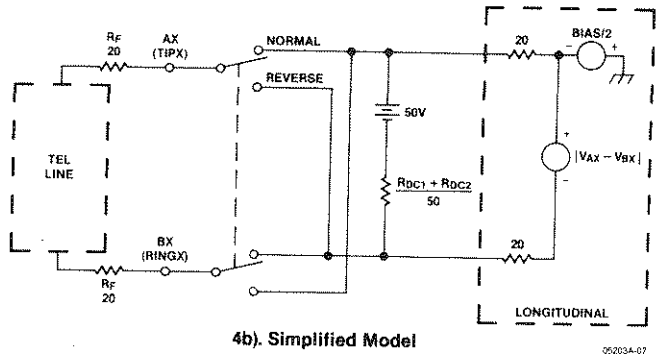
The bias circuit controls both the supply voltage and the longitudinal bias voltage of the 2-wire interface. It controls the supply voltage by sending a reference voltage (V_{REF}) to the switching regulator (next section). The switching regulator then adjusts the supply voltage of the 2-wire interface (V_{REG}) to be equal to V_{REF}. The bias circuit also sets the longitudinal bias voltage (V_{LBIAS}) of the 2-wire interface to a value equal to half the supply reference; i.e., $V_{REF}/2$. The equations for these voltages are:

$$V_{REG} = V_{REF} = -(|V_{DCMET}| + BIAS)$$
$$V_{LBIAS} = V_{REF}/2$$

The voltage represented by BIAS is needed to keep the amplifier response linear when audio signals are being transmitted.



4a). Detailed Diagram



4b). Simplified Model

Figure 4. Am7950—SLIC Power Feed Controller

The Am7950 SLIC power feed programming components are determined by:

$R_{DC1} + R_{DC2} = 50\,(R_{FEED} - 2R_F)$

$R_{DC1} = R_{DC2}$ (approx.)

$C_{DC} = (1.5ms)(R_{DC1} + R_{DC2})/(R_{DC1} \times R_{DC2})$

From these expressions, the components required for the above feed characteristics are:

$R_{DC1} = 25(840 - 40) = 20K$

$R_{DC2} = 40K - 20K = 20K$

$C_{DC} = (1.5ms)\,(40K)/(20K \times 20K) = 0.15\mu F$

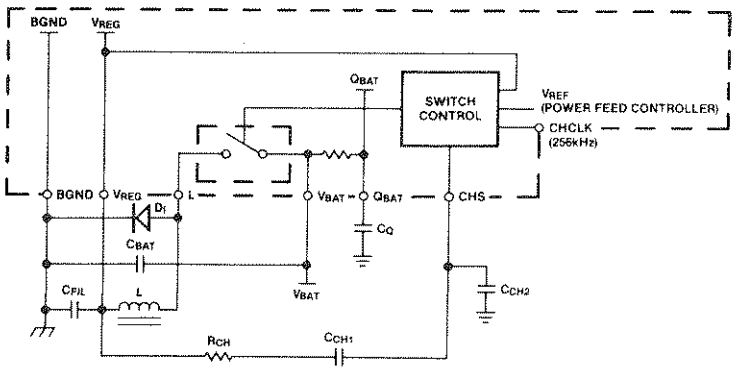
SWITCHING REGULATOR

The switching regulator supplies the operating voltage, V_{REG}, to the 2-wire interface (see Figure 5). This circuit adjusts V_{REG} to follow $V_{REF} = -|V_{AX} - V_{BX}| - BIAS$. Setting V_{REG} to the minimum voltage necessary to power the output amplifiers minimizes power consumption, particularly at high line currents. The switch control tells the switch to disconnect the L pin from V_{BAT} at the beginning of each CHCLK cycle and connect it for a time which depends on the difference between V_{REF} and V_{REG}. During this time, the current through the inductor decreases. When the switch controller connects L to V_{BAT}, the inductor current increases. The filter capacitor, C_{FIL}, on V_{REG} smooths the ripple caused by the variation in inductor current. The current from the V_{BAT} pin changes rapidly when the switch turns on or off, so a filter capacitor is needed from this pin to BGND. There is also a chopper stabilization network needed between V_{REG} and CHS.

The layout and quality of several of the external switching regulator components are very important. Extremely fast current changes occur in the catch diode, D_1, and in the V_{BAT} filter capacitor, C_{BAT}; hence, these must be low inductance components with short leads.

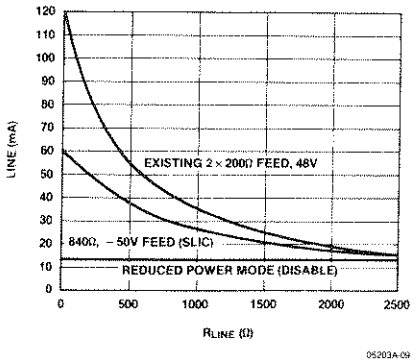
The connections from the diode to the L pin, from C_{BAT} to the V_{BAT} pin, and from the diode to C_{BAT} must all be short, low inductance connections. The L pin is subject to very fast voltage variations as the switch turns on and off, so all of the connections to this pin must be isolated from sensitive signals by means of a trace connected to BGND. All of the components connected to the regulator circuit should have voltage ratings well in excess of 70V. In addition, the diode should have a recovery time less than 10ns, and the inductor should have a series resistance less than 20Ω. All of the SLICs in a system should be synchronized to a common clock to prevent intermodulation products in the voice band.

The maximum recommended loop current is 70mA. (Higher currents can be accommodated but are not recommended.) In order to limit the maximum loop current to 70mA (required for power dissipation considerations) and still ensure that 18mA will be supplied to a 1900Ω loop, a feed characteristic of 840Ω feed resistance with an apparent battery of −50V is recommended. This feed characteristic closely simulates a 2 × 200Ω, 48V feed for lines above 1KΩ and provides adequate current for loops below 1KΩ. This feed characteristic and the power savings resulting from it are shown in Figures 6a and 6b respectively.



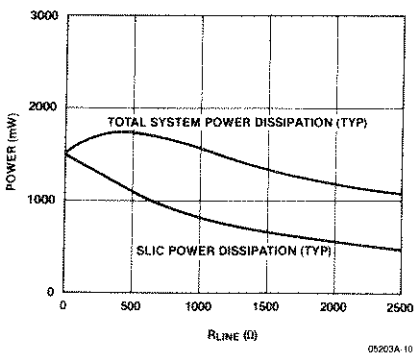
05203A-08

Figure 5. Am7950—SLIC Switching Regulator



05203A-09

Figure 6a. SLIC Battery Feed (Typical)



05203A-10

Figure 6b. SLIC Power Dissipation

INPUT DECODER AND CONTROL

Figure 7 is a simplified diagram of an eight channel subscriber line card. The Am7901A SLAC contains a serial interface port which connects to the line card controller microprocessor through the D_{IN}, D_{OUT} and D_{CLK} pins. Note that the SLAC's D_{IN}, D_{OUT} and D_{CLK} lines are common with all of the SLACs on the board. The SLAC is enabled for data input when the line card controller pulls $\overline{CS}$ low and toggles D_{CLK}. One of the SLAC's instructions enables the SLAC to accept a five-bit code that is transferred to an output latch. This five-bit field is used to drive the SLIC decoder and control inputs, C_1 through C_4 (only four bits are used).

On each channel the SLAC's $\overline{CS}$ input and the SLIC's $\overline{DET}$ output are connected together and the eight resulting lines are connected to a single 8-bit I/O port. The E_0 inputs of all eight SLICs are connected together so that all of the SLICs may be enabled simultaneously. (The SLIC's $\overline{DET}$ output is open collector when E_0 is low, thus preventing conflicts with the SLAC's $\overline{CS}$ input.) When the SLIC detectors are to be read the common enable line, E_0, is pulled high which allows each SLIC to indicate, on $\overline{DET}$, the logic state of the selected detector (selected via C_1–C_4). The SLACs will not be affected by this, even though $\overline{CS}$ may be pulled low, because the SLAC's serial I/O will accept commands or transfer data only while D_{CLK} is toggling.

This line card architecture offers the advantage that all eight channels (one SLIC and one SLAC per channel) can be controlled and/or monitored by only twelve microprocessor I/O ports. Furthermore, the detectors of all eight SLICs can be read simultaneously as a single byte.

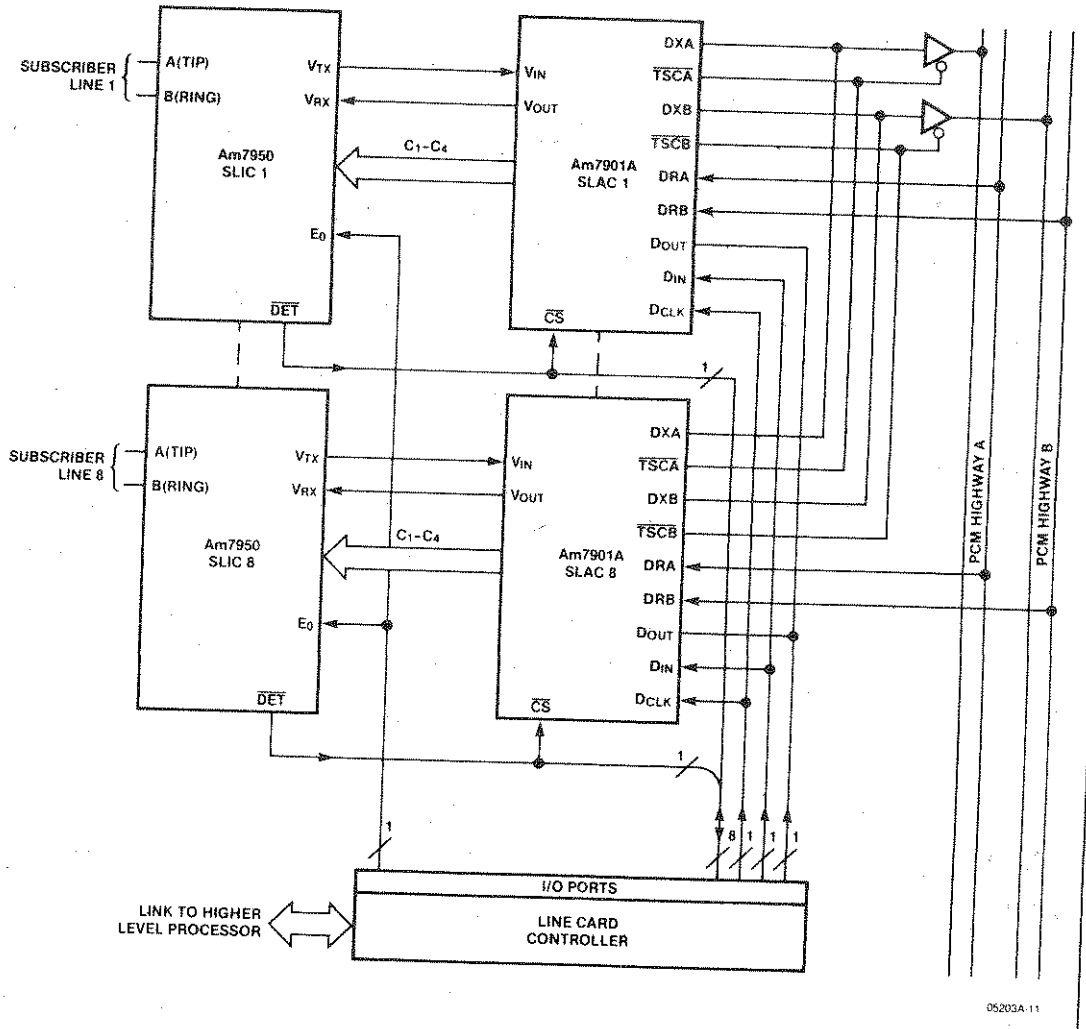


Figure 7. AMD Subscriber Line System 8-Channel Line Card

OFF-HOOK DETECTION

The first, and most important, loop monitoring function is off-hook detection. The block diagram of this detector is shown in Figure 8.

The 2-wire interface produces a current equal to the magnitude of the loop current divided by 310 and sends it out of the R_D pin. An external resistor and filter capacitor connect the R_D pin to −5V. The value of the voltage at the R_D pin is R_D times the current. The off-hook detector compares this voltage to a threshold voltage of 1.25V and generates a logic low output on $\overline{DET}$ when this voltage rises above the threshold.

The resulting relationship between threshold loop current (I_{THRESH}) and the external programming resistor (R_D) is given by:

$$R_D I_{THRESH}/310 = 1.25V$$

A filter capacitor (C_D) should be connected in parallel with R_D so that $R_D C_D = 0.5ms$.

RING TRIP DETECTOR

The suggested Ring Trip network for balanced ringing is shown in Figure 9. During ringing, the Ring relay driver is activated and the AX(TIPX) and BX(RINGX) leads are placed in the open circuit state. The ring feed source is connected by the ring relay to the line through the feed resistors, R_1 and R_2. For balanced ringing the feed resistors are equal, $R_1 = R_2$. The bridging resistors, R_{B1}, R_{B2}, R_3 and R_4, are used to produce a voltage between DA and DB whose sign changes when the telephone goes off-hook.

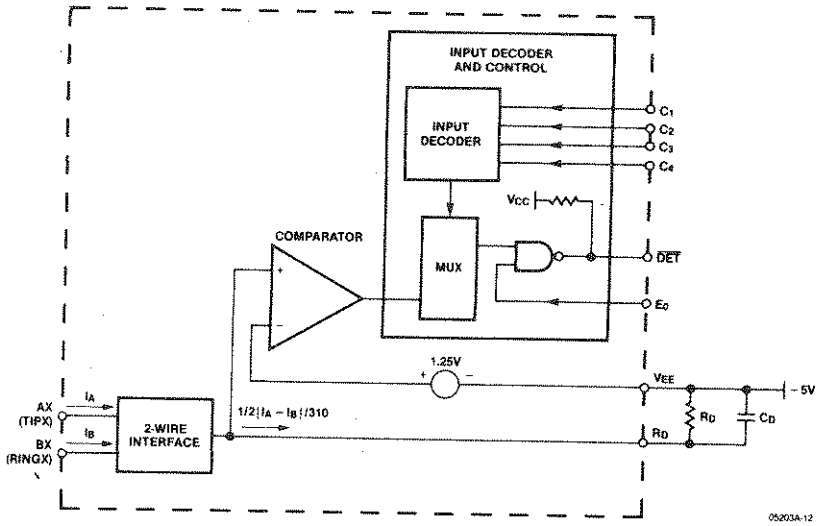


Figure 8. Am7950—SLIC Signaling Off-Hook Detection

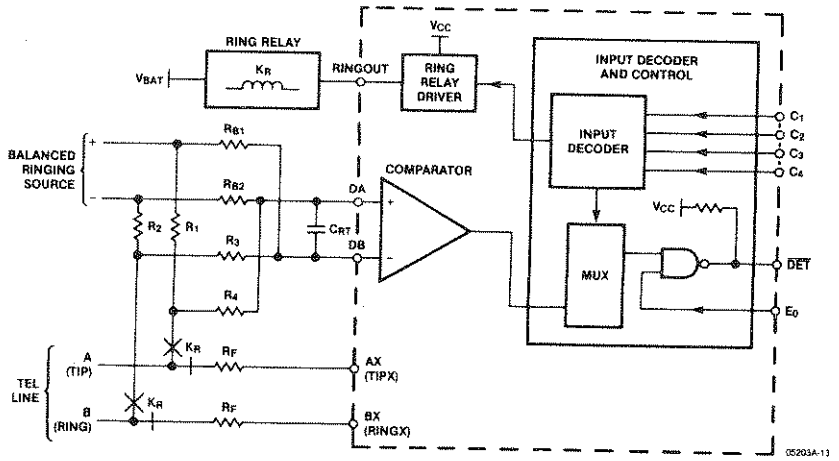


Figure 9. Am7950—SLIC Ring Trip Detector

The capacitor between DA and DB reduces the effective amplitude of the ringing signal by a factor of $1/|1+j2\pi f_r t|$ where $t=(R_3||R_{B2}+R_4||R_{B1})C$. For $f_r=20Hz$ ringing, C should be chosen to give a value of $t=50ms$. This will reduce the ringing by a factor of 6.4 and allow detection within 2 ringing cycles.

If R_M is the maximum line resistance which is to be considered an off-hook, the bridging resistors should be chosen so that $R_3=R_4$ and $R_{B1}=R_{B2}=R_3\times(R_M+R_{FEED})/R_M$ where $R_{FEED}=R_1+R_2$.

TABLE 1. SLIC DECODING

State	C_3	C_2	C_1	2-Wire Status	Detector Armed
0	0	0	0	Open Circuit	Ring Trip
1	0	0	1	Ringing	Ring Trip
2	0	1	0	Normal	Loop Det.
3	0	1	1	A, B Disable	Loop Det.
4	1	0	0	A(TIP) Open Circuit	Loop Det.
5	1	0	1	Reserved	—
6	1	1	0	Polarity Reversal, Active	Loop Det.
7	1	1	1	Polarity Reversal, Disable	Loop Det.

RELAY DRIVERS

A Test Relay driver is provided in order to facilitate line testing. During testing the SLIC operates as it normally would during the selected state, except that the test relay is activated. The test relay is designed to source up to 25mA from V_{CC} with less than 2V drop. The relay driver has an internal protection diode to V_{BAT} and can operate with 5V or 50V relays.

The Ring Relay driver is electrically identical to the Test Relay driver.

OTHER OPERATING MODES

An A(TIP) Open mode is provided in order to facilitate ground start signaling. In this state the A(TIP) power amplifier is open circuit and presents a high impedance to the line.

An Open Circuit function is provided to allow line powered relays to collapse, as well as to allow clearing of line faults. In this state both the A(TIP) and B(RING) power amplifiers are placed into a high impedance state.

For those systems that require it, Polarity Reversal is provided. In this state the normal voltages of A(TIP) and B(RING) are reversed. That is, B(RING) approaches ground and sources current while A(TIP) approaches battery and sinks current. A transition time of approximately 5ms is specified. All signal transmission, battery feed and loop control functions operate normally in Polarity Reversal.

A disable, or standby, feature is implemented in the SLIC to reduce power. The DC loop current is limited to 1.5 times the loop detector threshold current. Below the current limit the amplifier operates as if it were in the active state. The off-hook detector works normally.

OPERATING STATES

The SLIC has eight different operating states. These states are controlled by three TTL compatible inputs, C_1–C_3. A fourth TTL compatible input, C_4, controls the Test Relay driver. When C_4 is low, the TESTOUT pin will source current to a relay coil.

Test: When C_4 is low the test relay driver (TESTOUT) is activated, sourcing up to 25mA from V_{CC}. The operation of the SLIC's other circuits is determined by the particular operating state.

Ringing: When the SLIC is in the ringing state the ring relay driver is activated (RINGOUT) and the Ring Trip Detector is readable at $\overline{DET}$. Also, A(TIP) and B(RING) are both open circuit. While the SLIC is in the ringing state, signal transmission is inhibited.

Normal: In states where normal mode operation is indicated, the standard battery feed convention applies; A(TIP) is near ground and sources current and B(RING) is near V_{BAT} and sinks current. During normal mode operation all signal transmission and loop supervision functions operate and the off-hook detector is gated to $\overline{DET}$.

Disable: The Disable operating state is the SLIC's low power mode, in which the battery feed circuit limits the maximum DC loop current to 1.5 times the loop detector threshold current. The A(TIP) and B(RING) power amplifiers are still capable of handling at least 20mA longitudinals.

Open Circuit: When the SLIC is in the Open Circuit state both the A(TIP) and B(RING) power amplifiers are switched off and present a high impedance to the line. The open circuit state is the lowest power dissipation state. The loop detector is not armed in this state.

Tip Open: When the SLIC is in the Tip Open state, the A(TIP) power amplifier is switched off so that it presents a high impedance to the line.

Polarity Reversal: When the SLIC is in the Polarity Reversal state the normal battery feed convention is reversed with B(RING) approaching ground and sourcing current while A(TIP) approaches battery and sinks current. The transition time is specified to be approximately 5ms in order to prevent injection of noise into adjacent cables. While A(TIP) and B(RING) are in transition the off-hook function is meaningless since the loop current must pass through zero.

TABLE 2. Am7950—SLIC USER PROGRAMMABLE COMPONENTS

$Z_T = 1000\ (Z_{2WIN} - 2R_F)$	where Z_T is connected between the V_{TX} and RSN pins, the fuse resistors are R_F, and Z_{2WIN} is the desired 2-wire input impedance. When the SLIC is used with the Am7901A SLAC, Z_T can be a simple resistor.
$Z_{RX} = 1/2\ Z_T$	where Z_{RX} is connected from V_{RX} to the RSN pin and Z_T is defined above. Z_T/Z_{RX} sets the receive gain.
$R_{DC1} + R_{DC2} = 50(R_{FEED} - 2R_F)$ $C_{DC} = (1.5ms)(R_{DC1} + R_{DC2})/(R_{DC1}R_{DC2})$	where (R_{DC1}, R_{DC2}, C_{DC}) is the network connected to the R_{DC} pin. R_{DC1} and R_{DC2} are approximately equal.
$R_D = (390V)/I_T$ $C_D = (0.5ms)/R_D$	where (R_D, C_D) is connected from R_D to −5V and I_T is the threshold current between on-hook and off-hook.

TABLE 3. PARTS LIST—SINGLE CHANNEL SUBSCRIBER LINE SYSTEM (Figure 1)

U1	Am7950, SLIC (Subscriber Line Interface Circuit)
U2	Am7901A, SLAC (Subscriber Line Audio Processing Circuit)
K_R	Relay, 48V coil, 2C contacts, 1500V rating
K_T	Relay, 48V coil, 4C contacts, 1500V rating
L	Choke, 1mH, RF, 10Ω max
D_1	Diode, 100V, 100mA, 10ns
D_2–D_5	Diode, 100V, 3A
R_{F1}, R_{F2}	Resistor, fuse, 20Ω, 1% match
R_1, R_2	Resistor, 400Ω, 3%, 3W (sets ring feed resistance)*
R_{B1}, R_{B2}	Resistor, 249K, 1%, 1/4W (sets ring trip threshold)*
R_3, R_4	Resistor, 205K, 1%, 1/2W
R_{CH}	Resistor, 2.4K, 5%, 1/4W
R_D	Resistor, 51.1K, 1%, 1/4W (sets off-hook threshold and disable current)*
R_T	Resistor, 562K, 1%, 1/4W (sets 2-wire impedance)*
R_{RX}	Resistor, 280K, 1%, 1/4W (sets receive gain)*
R_{DC1}, R_{DC2}	Resistor, 20K, 1%, 1/4W (sets battery feed resistance)*
R_{TX}	Resistor, 10K, 5%, 1/4W
R_{SLAC}	Resistor, 1K, 5%, 1/4W
C_{RT}	Capacitor, 0.22μF, 20%, 100V
C_{HP}	Capacitor, 0.22μF, 20%, 100V
C_{AX}, C_{BX}	Capacitor, 2200pF, 20%, 100V
C_{FIL}	Capacitor, 0.47μF, 20%, 100V
C_{BAT}	Capacitor, 0.47μF, 20%, 100V
C_Q	Capacitor, 0.33μF, 20%, 100V
C_{CH1}	Capacitor, 8200pF, 20%, 100V
C_{CH2}	Capacitor, 560pF, 20%, 100V
C_D	Capacitor, 0.01μF, 20%, 10V (sets off-hook filtering)*
C_{DC}	Capacitor, 0.15μF, 20%, 10V
C_{TX}	Capacitor, 3μF, 20%
C_{SLAC}	Capacitor, 2000pF, 20%, 10V

Note: The parts marked by an asterisk (*) are user programmable. The values shown can be altered to suit the application; see the text for details.

ABSOLUTE MAXIMUM RATINGS

beyond which the life of the unit may be impaired

Storage Temperature	−55°C to +150°C
Ambient Temperature, Operating	0°C to +70°C
V_{CC} with Respect to AGND	−0.4V to +7V
V_{EE} with Respect to AGND	+0.4V to −7V
V_{BAT} with Respect to AGND	+0.4V to −70V
AGND with Respect to BGND	±0.3V
AX(TIPX) or BX(RINGX) to AGND (Note 1)	−70V to +0.4V
AX(TIPX) or BX(RINGX) to V_{BAT} (Note 1)	−0.4V to +70V
Current from AX(TIPX) or BX(RINGX)	±150mA
Voltage on TESTOUT, RINGOUT	V_{BAT} to V_{CC}
Current through Relay Drivers or Diodes	30mA
Voltage on Ring Trip Inputs	V_{BAT} to 0V
Peak Current through Regulator Switch	150mApk
Switcher Transient Peak Off Voltage	+1V
Chopper Stabilization Voltage (V_{CHS})	V_{BAT} to 0V
C_1, C_2, C_3, C_4, E_0, CHCLK to AGND	−0.4V to V_{CC}
Maximum Power Dissipation, $T_A = 70°C$	1.5W

Stresses above those listed under ABSOLUTE MAXIMUM RATINGS may cause permanent device failure. Functionality at or above these limits is not implied. Exposure to absolute maximum ratings for extended periods may affect device reliability.

Notes: 1. External diodes from V_{REG} to AX(TIPX) and BX(RINGX) may be required to protect against shorts to V_{BAT} and surges more negative than V_{REG}.

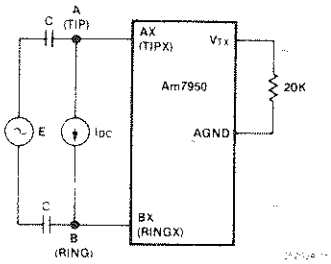
2. Thermal limiting circuitry on-chip will limit chip temperature to about 150°C. However, the device should never see this temperature. Operation above 150°C junction temperature can degrade device reliability.

OPERATING RANGE

Ambient Temperature	$0°C < T_A < 70°C$
AGND	0V
BGND	0mV ± 100mV
V_{CC}	+5.0V ± 5%
V_{EE}	−5.0V ± 5%
V_{BAT}	−64V to −40.5V

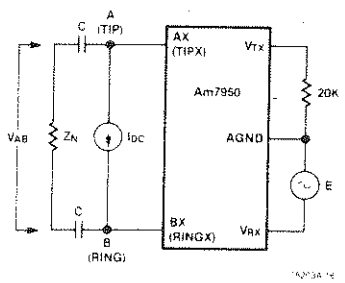
Operating ranges define those limits over which the functionality of the device is guaranteed.

TEST CIRCUITS



$1/\omega C << 600\Omega$ $\quad IL_{2,4} = 20 \log (V_{TX}/E)$

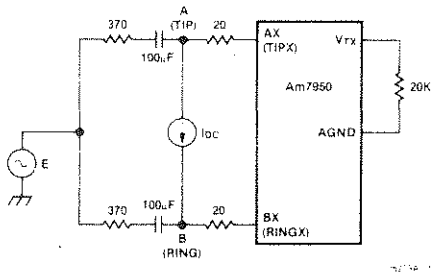
Figure 10. 2-to-4-Wire Insertion Loss Test Circuit



$1/\omega C << Z_N$ $\quad IL_{4,2} = 20 \log (V_{AB}/E)$

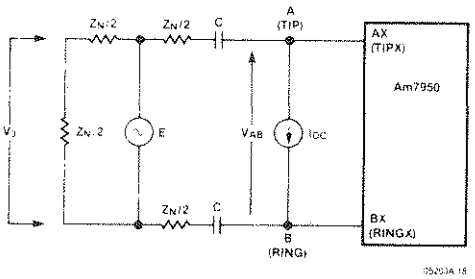
$BRS = 20 \log (V_{TX}/E)$

Figure 11. 4-to-2-Wire Insertion Loss and Balance Return Signal Test Circuit



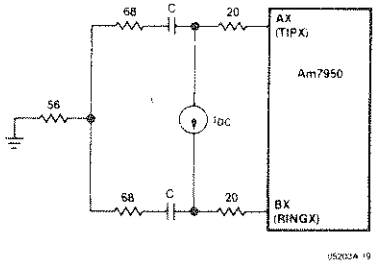
$$\text{Long. Bal} = 20 \log \left(\frac{V_{AB}}{E}\right)$$

Figure 12. Longitudinal Balance Test Circuit (IEEE Test Circuit)



$$1/\omega C << Z_N/2 \qquad RL = 20 \log \left| \frac{V_{AB} + V_3}{V_{AB} - V_3} \right|$$

Figure 13. 2-Wire Return Loss Test Circuit



$1/\omega C << 90\Omega$

Figure 14. Single Frequency Noise Test Circuit

Am7901A/B

Subscriber Line Audio-Processing Circuit
WORLD-CHIP™
PRELIMINARY

DISTINCTIVE CHARACTERISTICS

- Combination CODEC and Filter
- No trimming or adjustments required
- Uses digital signal processing
- Six user-programmable digital filters
- Dynamic Time Slot assignment
- Only 2 external components (non-precision)
- Dual PCM ports
- 4.096 MHz, 64-channel expanded mode operation
- Built-in test modes
- Microprocessor-compatible Serial Interface
- Control interface to SLIC
- Low standby power
- Selectable linear, μ-law (Am7901A) or μ-law, A-law (Am7901B)

GENERAL DESCRIPTION

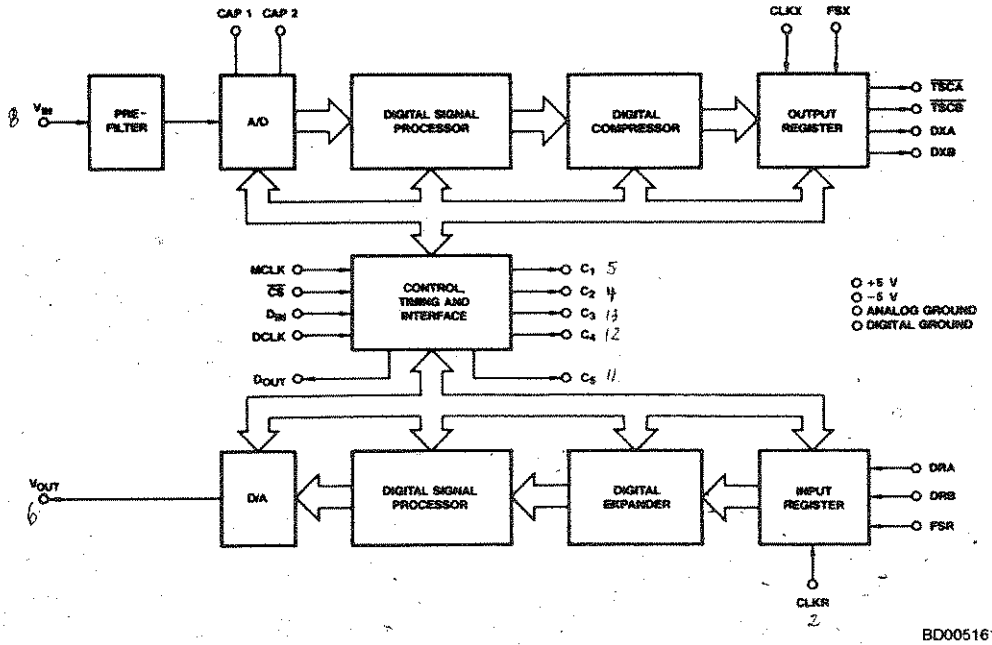
The Subscriber Line Audio-Processing Circuit (SLAC) performs the codec and filtering functions necessary in digital voice switching machines. In this application, the SLAC processes voiceband analog signals into Pulse-Code Modulated (PCM) outputs and processes PCM inputs into analog outputs. The SLAC's performance is compatible with applicable AT&T and CCITT specifications. The device consists of three main sections: transmit processor, receive processor, and control logic.

The transmit section contains an anti-aliasing filter, an interpolative A/D converter and a digital signal processor. The analog signals received are converted and digitally processed to generate either 16-bit linear or 8-bit μ-law codes (Am7901A), or 8-bit μ-law or A-law codes (Am7901B). Either one of two output ports may be selected for PCM data transmission.

The receive section contains a digital signal processor and a D/A converter. Either 16-bit linear or 8-bit μ-law codes (Am7901A), or 8-bit μ-law or A-law codes (Am7901B) are received, processed and converted to analog signals. Either one of two input ports may be selected for reception of PCM data.

The control I/O provides a microprocessor-compatible serial interface and allows the user bi-directional access to many programmable features and the capability to completely control the operation of the device via a comprehensive set of 32 commands.

BLOCK DIAGRAM



BD005161

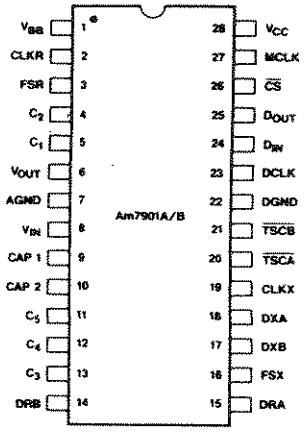
WORLD-CHIP is a trademark of Advanced Micro Devices, Inc.

September 1985

Order # 01520D

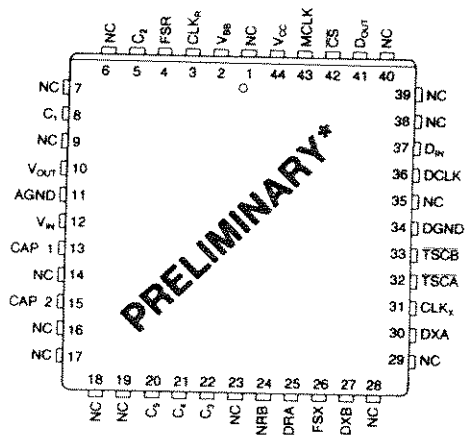
CONNECTION DIAGRAM
Top View

DIPs



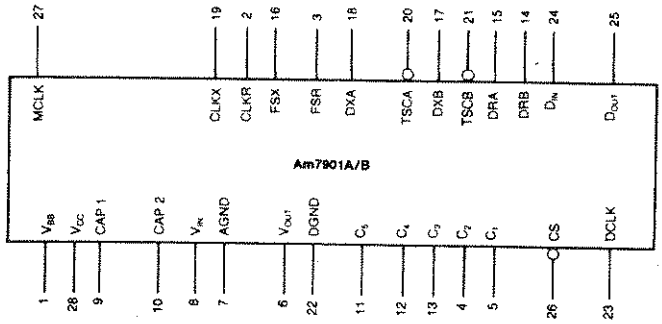
CD005871

PLCC



CD007041

LOGIC SYMBOL
DIPs



LS001891

ORDERING INFORMATION

AMD products are available in several packages and operating ranges. The order number is formed by a combination of the following: Device number, speed option (if applicable), package type, operating range and screening option (if desired).

Am7901A D C

Optional Processing
Blank = Standard Processing

Temperature Range
C = Commercial (0°C to +70°C)

Package Type
D = 28-Pin Ceramic DIP (CD 028)
P = 28-Pin Plastic DIP (PD 028)
J = 44-Pin Plastic Leaded Chip Carrier (PL 044*)

AMD Device Type
Am7901A (Linear, μ-Law)
Am7901B (A-Law, μ-Law)
Subscriber Line Audio-Processing Circuit (SLAC) WORLD-CHIP

Valid Combinations	
Am7901A	DC, PC*, JC*
Am7901B	DC, PC*, JC*

Valid Combinations

Consult the local AMD sales office to confirm availability of specific valid combinations, to check on newly released valid combinations, and to obtain additional data on AMD's standard military grade products.

*Preliminary. Subject to change.

PIN DESCRIPTION

V_{CC}: +5-V Power Supply

V_{BB}: −5-V Power Supply

DGND: Digital Ground

AGND: Analog Ground

Analog Input **(V_{IN})** The analog input is applied to the transmit path of the SLAC. The signal is sampled, digitally processed and encoded for the PCM output.

Analog Output **(V_{OUT})** The received-PCM data is digitally processed and converted to an analog signal at the V_{OUT} pin.

CAP 1, CAP 2 An external series resistor and capacitor are connected to these pins. These components are part of the integrator in the A/D converter. The recommended values of these non-precision components are 1 kΩ ±5% and 2000 pF ±20%.

Master Clock **(MCLK)** The Master Clock must be a 2.048 MHz ±100 ppm clock input. MCLK is used by the digital signal processors and is not dependent on the PCM input and output clocks.

PCM Outputs **(DXA, DXB)** The transmit-PCM data is serially fed out to either the DXA or the DXB port. The port selection is under user program control. For μ-law and A-law, 8 bits are transmitted and for linear code, 16 bits are transmitted. The output is available every 125 μs and the data is shifted out in 8/16-bit bursts at the CLKX rate. DXA and DXB are high impedance in between bursts and also in the stand-by mode.

Time Slot Control **($\overline{TSCA}$, $\overline{TSCB}$)** The Time Slot Control outputs are open drain outputs and are normally HIGH. $\overline{TSCA}$ is LOW when PCM data is present on the DXA output and $\overline{TSCB}$ is LOW when PCM data is present on the DXB output.

PCM Inputs **(DRA, DRB)** The receive-PCM data is serially received from either the DRA or the DRB port. The port selection is under user program control. For μ-law and A-law, 8 bits are received and for linear code, 16 bits are received. The data is received in 8 or 16-bit bursts every 125 μs at the CLKR rate.

Frame Sync **(FSX, FSR)** The Frame Sync pulse is an 8-kHz signal which identifies the beginning of a frame. The SLAC references individual time slots with respect to the Frame Sync pulse. FSX is the transmit-PCM Frame Sync and FSR is the receive-PCM Frame Sync. The FSX pulse must not be longer than 8 clock periods when companded code is used, and 16 clock periods when linear code is use.

PCM Clocks **(CLKX, CLKR)** The PCM Clocks determine the rate at which PCM data is serially shifted in to or out of the PCM ports. The maximum clock frequency is 4.096 MHz and the minimum clock frequency is 128 kHz. CLKX determines the rate at which PCM data is transmitted. CLKR determines the rate at which PCM data is received.

Chip Select **($\overline{CS}$)** The Chip Select input enables the device to either input or output control data.

Data Input **(D_{IN})** Control data is serially written via the Data Input port. The input rate is determined by the Data Clock.

Data Output **(D_{OUT})** Control data is serially read via the Data Output port. The output rate is determined by the Data Clock. D_{OUT} is HIGH-impedance when control data output is completed and $\overline{CS}$ is HIGH.

Data Clock **(DCLK)** The Data Clock shifts control data either in to or out of the SLAC. The maximum clock rate is 2.048 MHz.

Latched Outputs **(C_1-C_5)** The serial interface may be used to write data to a register whose outputs are brought out to C_1-C_5. These 5 lines are TTL-compatible and may be used to control the operation of a SLIC or any other device associated with the subscriber line.

FUNCTIONAL DESCRIPTION

Device Operation

General

The Am7901A/B performs the codec and filtering functions associated with the 4-wire section of the subscriber line circuitry in a digital switch. When used with the Am7950/7953 Subscriber Line Interface Circuit (SLIC), the pair provide a complete solution to the BORSCHT functions (Figure 1).

The SLAC contains auto-zeroed A/D and D/A converters. A microprocessor-compatible interface is provided to program the device into a variety of modes. These operating modes include, but are not limited to, companded or linear-code operation, dynamic time-slot assignment, and PCM-port selection.

The SLAC samples the analog signal at the V_{IN} pin and digitally processes it to produce either a linear or companded PCM code at the DXA or DXB output (Figure 2). Conversely, it receives either a linear or companded PCM code at the DRA or DRB input and digitally processes it to produce an analog output at the V_{OUT} pin. The processing is accomplished at the frame rate (8 kHz), and the digital output/input is available for transmission/reception every 125 μs.

Transmit Signal Processor

In the transmit path (Figure 3), the analog signal is converted, filtered, compressed, and made available for output.

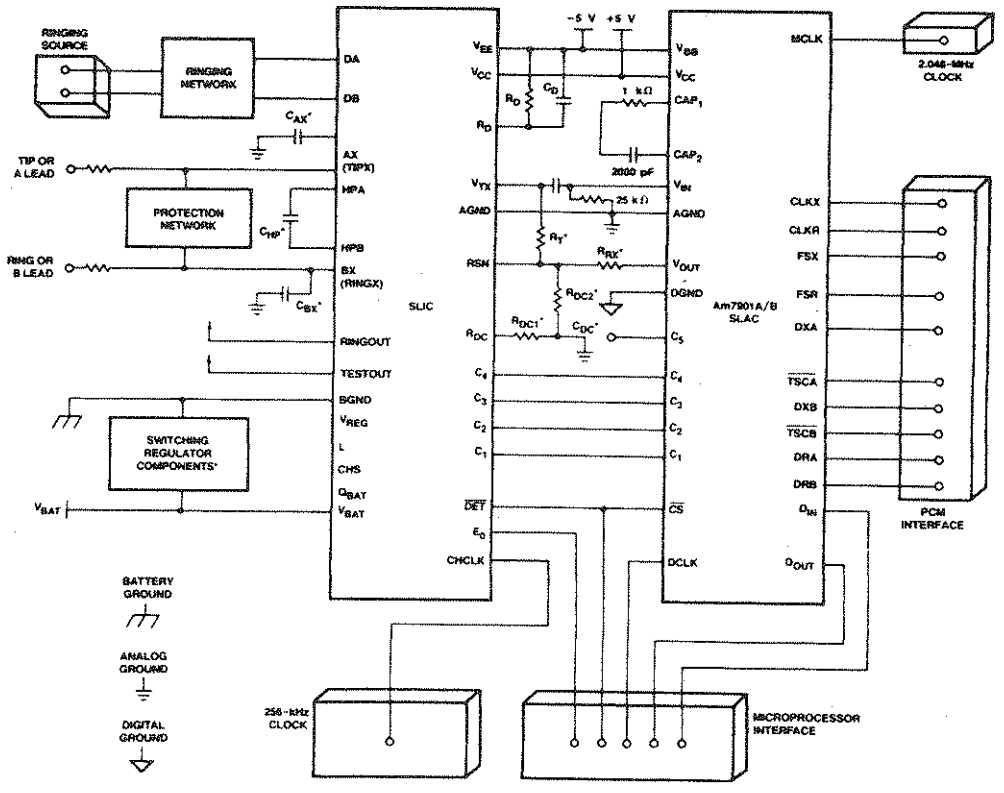
The prefilter is an integrated anti-aliasing filter which prevents signals near the sample rate from folding back into the voiceband during decimation. The A/D is designed to have a wide dynamic range and excellent signal-to-noise performance. It uses a modified sigma delta loop with a D/A converter to track the input signal at a 512-kHz sampling rate.

The Signal Processor contains an ALU, RAM, ROM and control logic to implement the filter sections. The B, X and GX blocks shown in Figure 3 are user-programmable filter sections and their coefficients are stored in the Coefficient RAM. These filters may be made transparent when not required in a system. The digital compressor may be bypassed when linear-code operation is desired.

The decimator reduces the high input sample rate. The X filter is a 4-tap Finite Impulse Response (FIR) section and is part of the frequency response correction network. The GX filter allows the user to program up to 12-dB gain in 0.1-dB steps in the transmit path. The B filter has 8 taps and operates on samples input from the Receive Signal Processor in order to provide trans-hybrid balancing in the loop. The low-pass filter limits the output bandwidth to meet the transmission requirements. The high-pass filter rejects 15-Hz and 50/60-Hz frequencies, and may be disabled during idle periods to allow low-frequency leakage testing on the 2-wire line.

Transmit PCM Interface

The Transmit PCM Interface receives either a 16-bit linear code (for linear operation) or an 8-bit compressed code (for μ-law and A-law operation) from the digital compressor. This code is loaded into the output register. The Transmit PCM Interface logic (Figure 4) controls the transmission of data onto the PCM highway through the output port-selection circuitry and the Time Slot Control block.



AF003643

*Component values are user-programmable. Refer to SLIC product specification.

Figure 1. Single-Channel Subscriber Line System

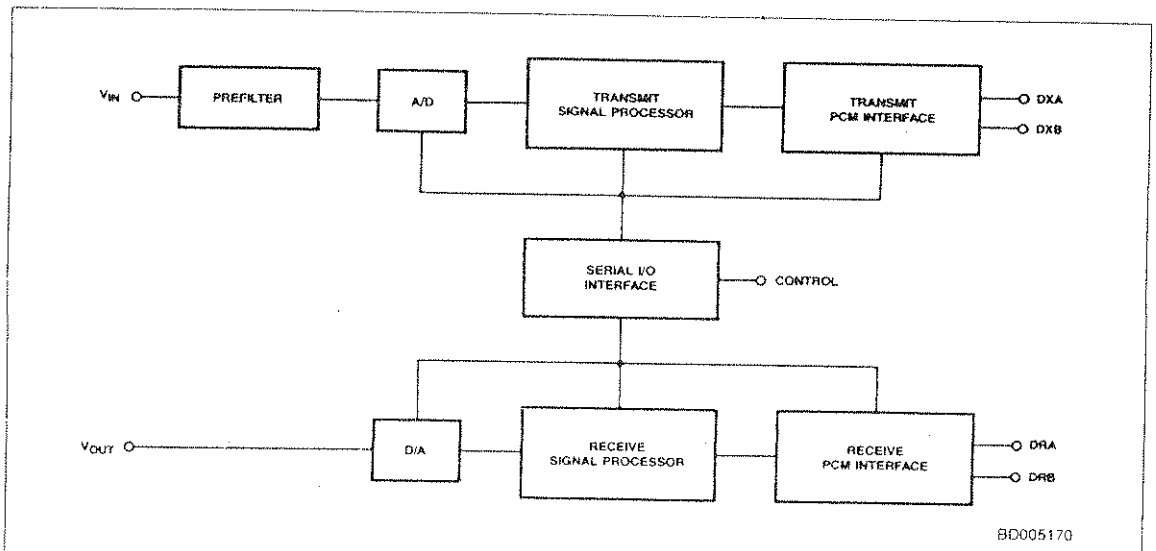
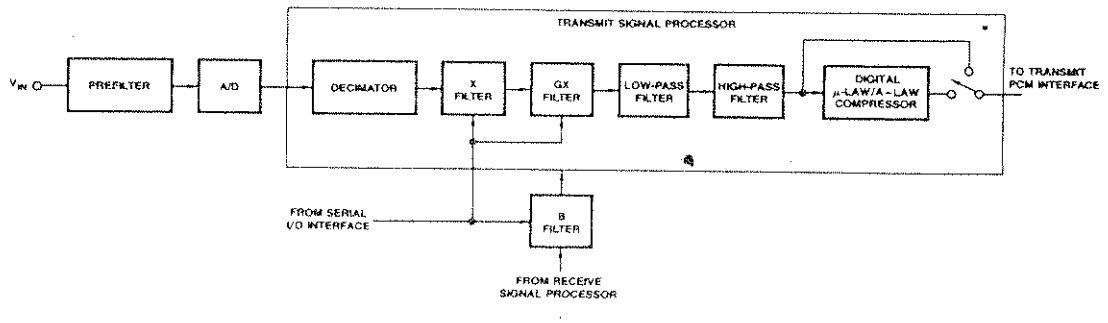


Figure 2. SLAC Block Diagram



*For Am7901B, the Digital Compressor cannot be bypassed.

Figure 3. Transmit Signal Processor

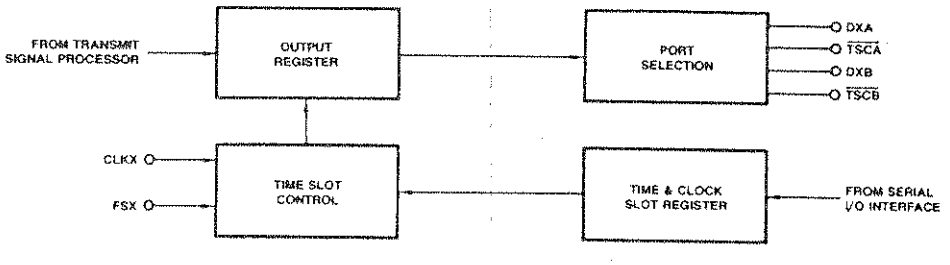


Figure 4. Transmit PCM Interface

The Frame Sync (FSX) pulse identifies the beginning of a Transmit frame and all channels (time slots) are referenced to it. The logic contains user-programmable Transmit Time Slot and Transmit Clock Slot registers. The Time Slot register is normally 5 bits wide and allows up to 32 8-bit channels or 16 16-bit channels (using CLKX = 2.048 MHz) in each frame. But in the expanded mode, 6 bits may be programmed to give 32 16-bit channels or 64 8-bit channels (using CLKX = 4.096 MHz) in each frame. The expanded mode bit becomes the sixth bit of the Time Slot register. If this bit is low, one of channels 0 to 31 is selected and if it is high, one of channels 32 to 64 is selected. This feature allows any combination of channel assignments and clock frequencies (over a range of 128 kHz to 4.096 MHz) in a system. For μ-law and A-law operation, 8 bits/channel are output and for linear code operation, 16 bits/channel are output. The data is transmitted Most Significant Bit (MSB) first. The Clock Slot register is 3 bits wide and may be programmed to offset the Time Slot assignment by 0 to 7 CLKX periods to eliminate any clock skew in the system (Figure 5).

In the Am7901A/B, the PCM data may be user-programmed to be output onto one of two ports, DXA or DXB. Correspondingly, either $\overline{TSCA}$ or $\overline{TSCB}$ is also low.

Receive PCM Interface

The Receive PCM Interface logic (Figure 7) controls the reception of data from the PCM highway and transfers it for expansion (μ-law or A-law) to the Receive Signal Processor. The operation of this interface is identical to the Transmit section.

The Frame Sync (FSR) pulse identifies the beginning of a Receive frame and all channels (time slots) are referenced to it. The logic contains user-programmable Receive Time Slot and Receive Clock Slot registers. The Time Slot register is normally 5 bits wide and allows up to 32 8-bit channels (using CLKR = 2.048 MHz) in each frame. But in the expanded mode, 6 bits may be programmed to give 32 16-bit channels or 64 8-bit channels (using CLKR = 4.096 MHz) in each frame. The expanded mode bit becomes the sixth bit of the Time Slot register. If this bit is low, one of channels 0 to 31 is selected and if it is high, one of channels 32 to 64 is selected. This feature allows any combination of clock frequencies (over a range of 128 kHz to 4.096 MHz) and channel assignments in a system. For μ-law and A-law operation, 8 bits/channel are input and for linear code, 16 bits/channel are input. The MSB of the code must be received first. The Clock Slot register is 3 bits wide and may be programmed to offset the Time Slot assignment by 0 to 7 CLKR periods to eliminate any clock skews in the system (Figure 8).

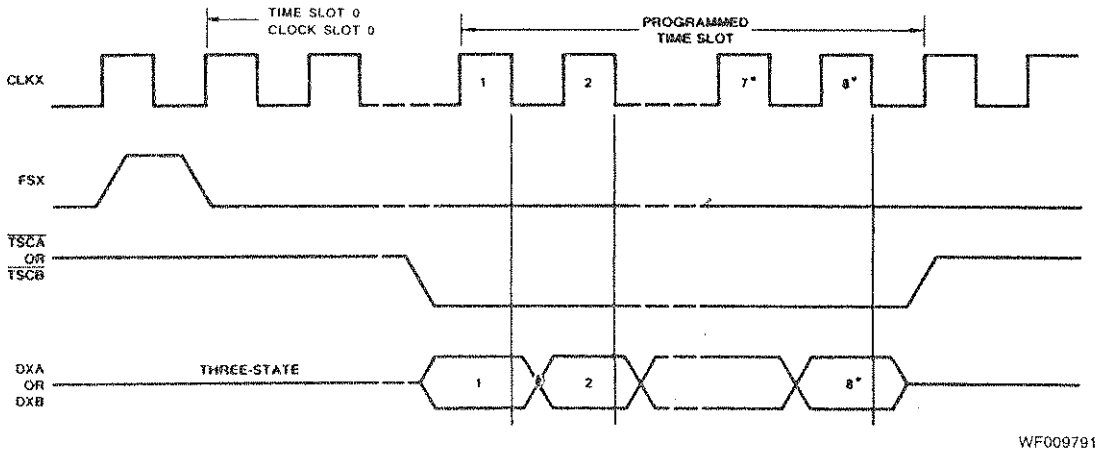
In the Am7901A/B, the PCM data may be user-programmed to be input from one of two ports, DRA or DRB.

Receive Signal Processor

In the receive path (Figure 6), the digital signal is expanded, filtered, converted to analog, and output onto the V_{OUT} pin.

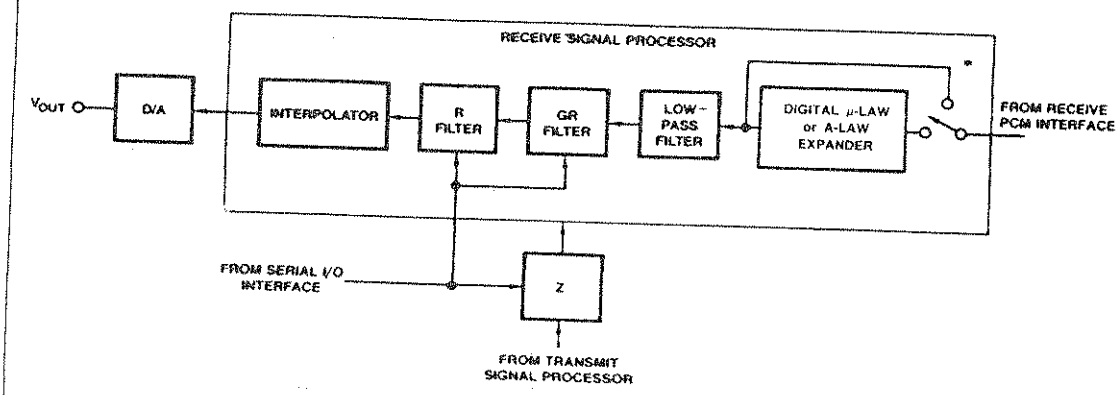
The Signal Processor contains an ALU, RAM, ROM and control logic to implement the filter sections. The Z, R and GR are user-programmable (through the Serial I/O Interface) filter sections and their coefficients are stored in the coefficient RAM. These filters may be made transparent when not required in a system.

The low-pass filter band-limits the signal. The GR filter allows the user to program a loss of up to 12 dB in 0.1-dB steps. The R filter is a 4-tap FIR section and is part of the frequency response correction network. The Z filter provides feedback from the Transmit Signal Processor to the Receive Signal Processor and is used to modify the effective input impedance to the system. The interpolator provides the higher sample rate to the D/A converter.



*For Linear Code, the 7th and 8th Clock Cycles Correspond to the 15th and 16th Clock Cycles.

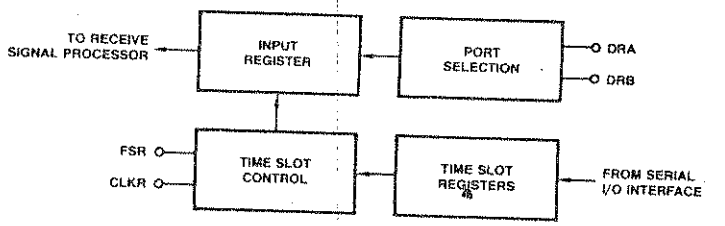
Figure 5. Transmit PCM Timing Diagram



BD005202

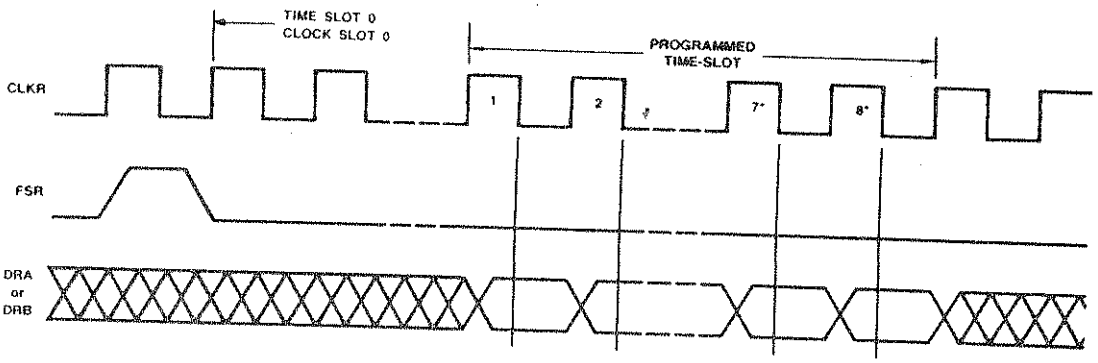
*For Am7901B, the expander cannot be bypassed.

Figure 6. Receive Signal Processor



BD005210

Figure 7. Receive PCM Interface



WF009801

*For Linear Code, the 7th and 8th Clock Cycles Correspond to the 15th and 16th Clock Cycles.

Figure 8. Receive PCM Timing Diagram

Serial I/O Interface

A microprocessor may be used to program the SLAC and control its operation using the Serial I/O Interface (Figure 9). Additionally, data programmed previously may be read out for verification. The control word format is shown in Table 1. Commands are provided to:

- Set power-up/power-down modes
- Set up test functions
- Set up operating functions
- Program filter coefficients
- Assign time slots and port selection
- Write to the SLIC latch
- Enable/Disable each user-programmable filter

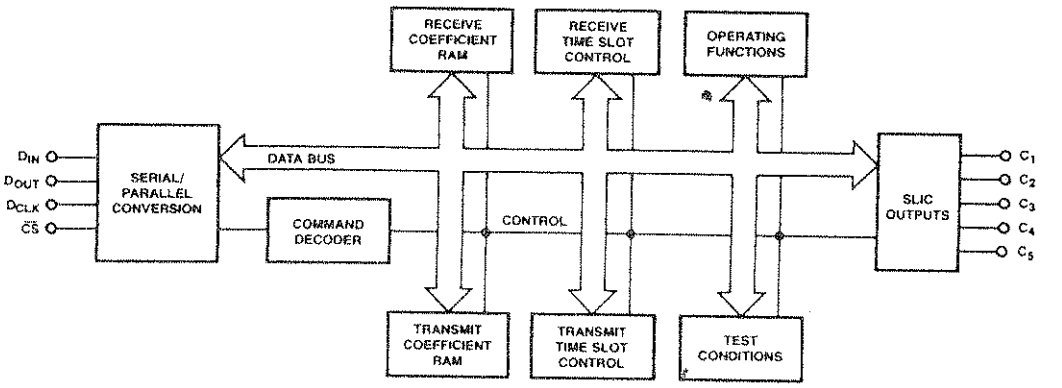
The interface consists of 4 pins, $\overline{CS}$, DCLK, D_{IN} and D_{OUT}. The device is accessed by $\overline{CS}$ and data is serially loaded-in on D_{IN}, or read-out on D_{OUT} under control of DCLK. Either commands or data words may be written to the SLAC, but only data words can be read out. All words are 8 bits wide and are written or read MSB first (Figure 10).

For both reception or transmission of words, exactly 8 Data Clock cycles must be received after $\overline{CS}$ goes LOW. $\overline{CS}$ must stay HIGH (off period) for a minimum time period before it can go LOW again (see Note 4 under Switching Characteristics). During this off-period, the logic decodes and executes the command. All reading of data must be preceded by an input command requesting the data. Once control data transmission has begun, no new input commands will be accepted until control data transmission is completed.

A Serial I/O cycle is defined by transitions of $\overline{CS}$ and DCLK. Upon proper application of power supplies and MCLK, the device expects the first word to be a command. A number of commands require additional data words to be input or output. The SLAC will not accept new commands until all this data has been transferred. But in the read mode, a data word of all zeroes is equivalent to the power-down command and the device resets to the stand-by mode and is ready to receive a new command.

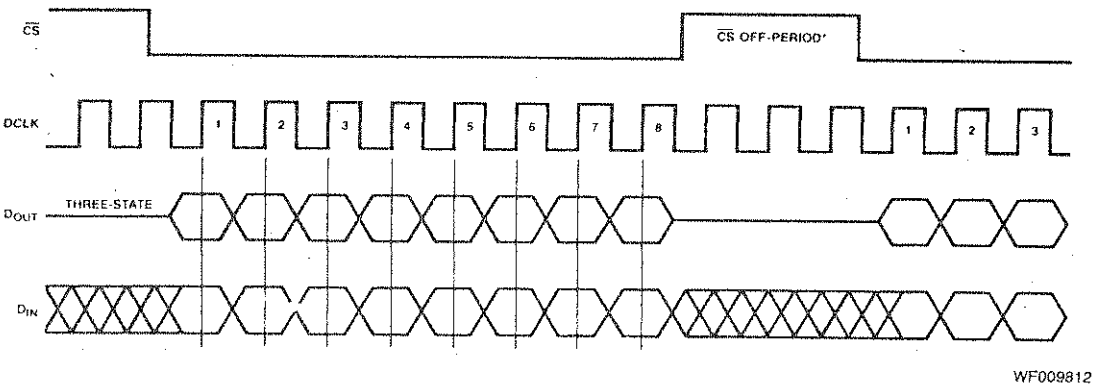
There are two possible operations of DCLK and $\overline{CS}$ for the SLAC to function correctly. If the $\overline{CS}$ is held in the HIGH state between accesses, the DCLK may free run with no change to the internal control data. Using this method, the same DCLK may be run to a number of SLACs and individual $\overline{CS}$ lines will select the appropriate device to access. If the DCLK is held in the LOW state between accesses, the $\overline{CS}$ line may make multiple transitions between accesses for a particular SLAC. This allows running one $\overline{CS}$ line to all SLACs and selecting a particular device through enabling or disabling its DCLK.

It should be noted that the DCLK can stay in the LOW state indefinitely with no loss of internal control information. However, it should not be held in the HIGH state for more than 20 μs to ensure proper operation as indicated by the Switching Characteristics Table.



BD005221

Figure 9. Serial I/O Interface



*See Note 4 under Switching Characteristics

Figure 10. Serial I/O Interface Timing Diagrams

Digital Filters

The SLAC uses digital signal processing to implement the various filters (Figure 11).

The advantages of digital filters are:

- High reliability
- No drift with time or temperature
- Unit-to-unit repeatability
- Superior transmission performance

Six of the digital filters in the signal processing sections are user-programmable. These allow the user to independently modify the gain in both the transmit and receive paths, provide trans-hybrid balancing in the system, and adjust the two-wire line termination impedance. The programming capability feature allows the user to optimize the performance of the SLAC for his system. Each programmable filter section has the following type of transfer function:

$$H_Z = h_0 + h_1 z^{-1} + h_2 z^{-2} + \ldots\ldots + h_n z^{-n} \qquad \text{(Eq. 1)}$$

The values of the user-defined coefficients (h_n) are assigned via the Serial I/O Interface (Table 1). The number of taps (n) provided depends on the particular filter.

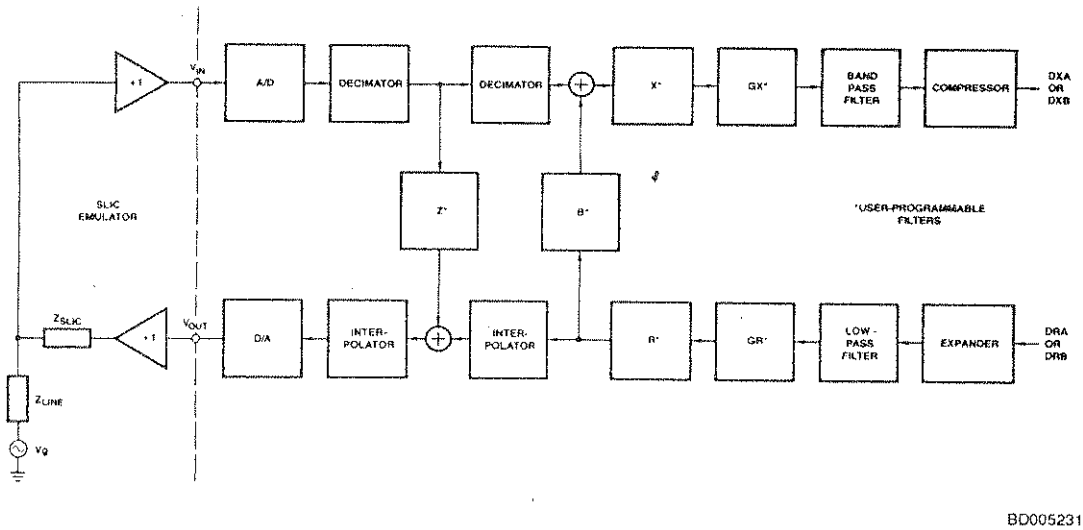


Figure 11. SLAC Signal Processing Flow

The filter function is performed by a series of multiplications and accumulations. A multiply is accomplished by shifting the multiplicand and summing the result with the previous value at that summation node. For example, a one-bit multiply is a shift of M bits where M is related to the position of the binary one in the multiplier (h_i) as expressed in the following equation:

$$h_i = B_1\ 2^{-M_1} + B_2\ 2^{-M_2} + \ldots B_N\ 2^{-M_N} \qquad \text{(Eq. 2)}$$

where: $M_i \leq M_{i+1}$
$B_i = \pm 1$

The subscript N is limited to 4 for the GR, GX, R, X and Z filters, and N is 3 for the B filter. The multiply is done from the Least Significant Bit (LSB) to the Most Significant Bit (MSB). Notes 11 and 12 explain the encoding of the shift codes.

The B, X, R, Z and Gain Parameters are written in or read out as 8-bit words. The format of the parameters is shown below:

A. B Coefficients

7 4	3 0	
$C_{30}m_{30}$	$C_{20}m_{20}$	← 1st word
$C_{10}m_{10}$	$C_{31}m_{31}$	← 2nd word
$C_{21}m_{21}$	$C_{11}m_{11}$	
$C_{32}m_{32}$	$C_{22}m_{22}$	
$C_{12}m_{12}$	•	
•	•	
•	•	
•	$C_{37}m_{37}$	
$C_{27}m_{27}$	$C_{17}m_{17}$	← 12th word

B. X Coefficients

7 4	3 0	
$C_{40}m_{40}$	$C_{30}m_{30}$	← 1st word
$C_{20}m_{20}$	$C_{10}m_{10}$	← 2nd word
$C_{41}m_{41}$	$C_{31}m_{31}$	
$C_{21}m_{21}$	$C_{11}m_{11}$	
$C_{42}m_{42}$	$C_{32}m_{32}$	
$C_{22}m_{22}$	$C_{12}m_{12}$	
$C_{43}m_{43}$	$C_{33}m_{33}$	
$C_{23}m_{23}$	$C_{13}m_{13}$	← 8th word

C. R, Z Coefficients

7 4	3 0	
$C_{43}m_{43}$	$C_{33}m_{33}$	← 1st word
$C_{23}m_{23}$	$C_{13}m_{13}$	
$C_{42}m_{42}$	$C_{32}m_{32}$	
$C_{22}m_{22}$	$C_{12}m_{12}$	
$C_{41}m_{41}$	$C_{31}m_{31}$	
$C_{21}m_{21}$	$C_{11}m_{11}$	
$C_{40}m_{40}$	$C_{30}m_{30}$	
$C_{20}m_{20}$	$C_{10}m_{10}$	← 8th word

D. Gain Coefficients

7 4	3 0	
$C_{40}m_{40}$	$C_{30}m_{30}$	← 1st word
$C_{20}m_{20}$	$C_{10}m_{10}$	← 2nd word

$C_{xy}\ m_{xy}$ = C is the sign bit and m is the 3-bit code specifying the position of the 1s.
y is the coefficient number
x specifies the relative position of the one in coefficient Y (1 = most significant one, 2 = second one, etc.).
and the coefficients in Equation 1 shown above are described by:

$$h_i = (C_{1i} \cdot 2^{-\hat{m}_{1i}}\ (1 + C_{2i} \cdot 2^{-\hat{m}_{2i}}\ (1 + C_{3i} \cdot 2^{-\hat{m}_{3i}}\ (1 + C_{4i} \cdot 2^{-m_{4i}}))))$$

except for the G_X filter where

$$h_i = 1 + (C_{1i} \cdot 2^{-\hat{m}_{1i}}\ (1 + C_{2i} \cdot 2^{-\hat{m}_{2i}}\ (1 + C_{3i} \cdot 2^{-\hat{m}_{3i}}\ (1 + C_{4i} \cdot 2^{-\hat{m}_{4i}}))))$$

where $\hat{m}_{ij} = 7 - m_{ij}$

Two-Wire Impedance Matching

A feedback path is provided from the transmit to the receive section via the Z filter. This filter may be programmed to modify the effective termination impedance (Z_{SLIC}) of a SLIC or a transformer hybrid to a desired value. The desired impedance may be complex. This feature allows the user to terminate each SLIC in a Subscriber Line System with a fixed resistor and digitally modify their impedance using the Z filter.

The X and R filters are the Transmit and Receive attenuation distortion correction filters. These filter sections are programmed to compensate the attenuation distortion caused by the Z filter.

Trans-Hybrid Balance

In a traditional line card system, a balance network is used with the SLIC to achieve trans-hybrid balancing. If the balance network perfectly matches the subscriber's line, infinite trans-hybrid balancing is achieved. But in general, the matching in traditional systems is poor and trans-hybrid balancing is not very good. Some systems have up to 2 or 3 compromise networks per line that must be selected semi-automatically or manually to provide the balance.

In the SLAC, a feedback path is provided from the receive to the transmit section via the B filter. This filter may be programmed to cancel the received signal from the transmit signal path and achieve a significantly improved level of trans-hybrid balance.

Gain Adjustment

Signal levels in the transmit and receive paths may be modified by programming the GX and GR filters. The GX filter allows the user to add up to 12 dB of gain (in 0.1-dB steps) in the transmit path. The GR filter allows the user to add up to 12 dB of loss (in 0.1-dB steps) in the receive path.

Test Features

The SLAC simplifies system testing by providing both digital and analog loop-back paths. Under program control, either the DRA or DRB input is looped to the DXA or DXB output (digital loop-back) through a path from the output of the interpolator in the receive path to the input of the decimator in the transmit path, or the V_{IN} input is looped to the V_{OUT} output (analog loop-back) through the Z filter. To allow testing of the subscriber loop cabling for leakage, the transmit high pass filter may be disabled and auto zero operation interrupted. The receive analog output may be programmed to cut off. This receive cut-off command may be used to stop oscillations in the four-wire side of the telephone network.

Stand-by Mode

The SLAC is forced into the stand-by mode either by power-on clear or by reception of the power-down code. In this mode, power is switched off from all circuitry that can be turned off. No transmission or reception of PCM data takes place. However, the circuits which contain programmed information retain their data. The Serial I/O interface remains active to receive new commands.

Power-On Clear

Proper operation of power-on clear requires sequenced application of V_{CC}, MCLK then V_{BB}.

Stand-Alone Mode

In the stand-alone mode, the serial interface is not used. The DCLK and D_{IN} pins may be used to control the device. Applying −5 V to the DCLK pin resets the device and the D_{IN} pin can subsequently be used to power-up or power-down the SLAC.

DCLK	D_{IN}	
0	X	Normal mode
1	X	Normal mode
−5 V	0	Reset and Power-Down
−5 V	1	Reset and Power-Up

Reset State

The Reset State of the device is:

a) Both Transmit and Receive Time and Clock Slots are set to zero.
b) μ-law is selected for Am7901A. A-law is selected for Am7901B
c) B, X, R, Z filters are disabled
d) Both Transmit (GX) and Receive (RX) gains are set to unity
e) SLIC outputs are set high
f) Normal conditions are selected (see Note 9 – Command Word Format)
g) DXA/DRA ports are selected

μ-LAW: POSITIVE INPUT VALUES

1	2	3	4	5	6	7	8
Segment Number	Number of Intervals X Interval Size	Value at Segment End Points	Decision Value Number n	Decision Value x_n (1)	Character Signal (5) Bit Number 1 2 3 4 5 6 7 8	Value at Decoder Output y_n (3)	Decoder Output Value Number
		8159	(128)	(8159)	- - - - - - - -		
8	16 x 256		127	7903	1 0 0 0 0 0 0 0	8031	127
			⋮	⋮	⋮ (2)	⋮	⋮
			113	4319			
		4063	112	4063	1 0 0 0 1 1 1 1	4191	112
7	16 x 128		⋮	⋮	⋮ (2)	⋮	⋮
			97	2143			
		2015	96	2015	1 0 0 1 1 1 1 1	2079	96
6	16 x 64		⋮	⋮	⋮ (2)	⋮	⋮
			81	1055			
		991	80	991	1 0 1 0 1 1 1 1	1023	80
5	16 x 32		⋮	⋮	⋮ (2)	⋮	⋮
			65	511			
		479	64	479	1 0 1 1 1 1 1 1	495	64
4	16 x 16		⋮	⋮	⋮ (2)	⋮	⋮
			49	239			
		223	48	223	1 1 0 0 1 1 1 1	231	48
3	16 x 8		⋮	⋮	⋮ (2)	⋮	⋮
			33	103			
		95	32	95	1 1 0 1 1 1 1 1	99	32
2	16 x 4		⋮	⋮	⋮ (2)	⋮	⋮
			17	35			
		31	16	31	1 1 1 0 1 1 1 1	33	16
1	15 x 2		⋮	⋮	⋮ (2)	⋮	⋮
			2	3			
	1 x 1		1	1	1 1 1 1 1 1 1 0	2	1
			0	0	1 1 1 1 1 1 1 1	0	0

Notes: 1. 8159 normalized value units correspond to T_{MAX} = 3.17 dBm0.
2. The character signal corresponding to positive input values between two successive decision values numbered n and $n+1$ (see column 4) is $(255-n)$ expressed as a binary number.
3. The value at the decoder is $y_0 = x_0 = 0$ for $n = 0$, and $y_n = \frac{x_n + x_{n+1}}{2}$ for $n = 1, 2, \ldots, 127$.
4. x_{128} is a virtual decision value.
5. Bit 1 is a 0 for negative input values.

A-law, positive input values

1	2	3	4	5	6	7	8
Segment Number	Number Of Intervals X Interval Size	Value At Segment End Points	Decision Value Number n	Decision Value x_n (1)	Character Signal Before Inversion Of The Even Bits Bit Number 1 2 3 4 5 6 7 8	Value at Decoder Output y_n (3)	Decoder Output Value Number
		4096	(128)	(4096)	- - - - - - - -		
			127	3968	1 1 1 1 1 1 1 1	4032	128
7	16 x 128		⋮	⋮	⋮ (2) ⋮	⋮	⋮
			113	2176	1 1 1 1 0 0 0 0	2112	113
		2048	112	2048			
6	16 x 64		⋮	⋮	⋮ (2) ⋮	⋮	⋮
			97	1086	1 1 1 0 0 0 0 0	1056	97
		1024	96	1024			
5	16 x 32		⋮	⋮	⋮ (2) ⋮	⋮	⋮
			81	544	1 1 0 1 0 0 0 0	528	81
		512	80	512			
4	16 x 16		⋮	⋮	⋮ (2) ⋮	⋮	⋮
			65	272	1 1 0 0 0 0 0 0	264	65
		256	64	256			
3	16 x 8		⋮	⋮	⋮ (2) ⋮	⋮	⋮
			49	136	1 0 1 1 0 0 0 0	132	49
		128	48	128			
2	16 x 4		⋮	⋮	⋮ (2) ⋮	⋮	⋮
			33	68	1 0 1 0 0 0 0 0	66	33
		64	32	64			
1	32 x 2		⋮	⋮	⋮ (2) ⋮	⋮	⋮
			1	2			
			0	0	1 0 0 0 0 0 0 0	1	1

Notes: 1. 4096 normalized value units correspond to $T_{max} = 3.14$ dBm0.
2. The character signals are obtained by inverting the even bits of the signals of column 6. Before this inversion, the character signal corresponding to positive input values between two successive decision values numbered n and $n + 1$ (see column 4) is $(128 + n)$ expressed as a binary number.
3. The value at the decoder output is $y_n = \frac{x_{n-1} + x_n}{2}$ for $n = 1, \ldots, 127, 128$.
4. x_{128} is a virtual decision value.
5. Bit 1 is a 0 for negative input values.

TABLE I: CONTROL WORD FORMAT

The Control Interface consists of Data Input, Data Output, Data Clock and $\overline{CS}$ Input. Data is read in (read out) on the Serial Data Input (output). The Serial Input consists of 8-bit (byte) command words which may be followed with additional bytes of input data or may be followed by the SLAC outputting bytes of data. All words are input with MSB (D_7) first and LSB (D_0) last. All outputs are output with the MSB (D_7) first and the LSB (D_0) last. Words are written or read one at a time, with $\overline{CS}$ going high for at least the minimum off-period (see Note 4 under Switching Characteristics) before the next read or write operation. The first 3 bits of the command word indicate the type of command and the last 5 bits contain either data or further information about the command. The classes of command are:

D_7	D_6	D_5	
0	0	0	Power Down/No Operation
0	0	1	Transmit Time Slot Selection
0	1	0	Receive Time Slot Selection
0	1	1	Clock Slot and Gain Selection Read Slot, Gain and PCM Mode
1	0	0	Set Basic and Operating Functions and PCM Modes
1	0	1	Read/Write Coefficients, Set Test Modes, Select μ-law/A-law/linear
1	1	0	Data for SLIC Interface
1	1	1	Power Up/No Operation

MSB	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	LSB	
	0	0	0	0	0	0	0	0	Power Down[1]	
	0	0	0	X	X	X	X	X	Reserved[2]	
	0	0	1	Y	Y	Y	Y	Y	Transmit Time Slot Selection[3]	Choose 1 of 32 Time Slots
	0	1	0	Y	Y	Y	Y	Y	Receive Time Slot Selection[3]	Choose 1 of 32 Time Slots
	0	1	1	0	0	Y	Y	Y	Transmit Clock Slot Selection[3]	Choose 1 of 8 Clock Slots
	0	1	1	0	1	Y	Y	Y	Receive Clock Slot Selection[3]	Choose 1 of 8 Clock Slots
	0	1	1	1	0	0	1	0	Transmit Gain Selection	Followed by 2 Bytes of Data[4]
	0	1	1	1	1	0	1	0	Receive Gain Selection	Followed by 2 Bytes of Data[4]
	0	1	1	1	0	1	0	1	Read Transmit Time and Clock Slot[5]	Followed by 1 Byte of Data[4]
	0	1	1	1	0	0	0	1	Read Transmit Gain	Followed by 2 Bytes of Data[4]
	0	1	1	1	1	1	0	1	Read Receive Time and Clock Slots[5]	Followed by 1 Byte of Data[4]
	0	1	1	1	1	0	0	1	Read Receive Gain	Followed by 2 Bytes of Data[4]
	0	1	1	1	0	1	1	1	Read PCM Mode	Followed by 1 Byte of Data[4, 6]
	1	0	0	0	A	B	C	D	Operating & Basic Function[7]	
	1	0	0	1	E	F	G	H	PCM-Mode Selection[8]	
	1	0	1	0	0	0	0	0	Write B Coefficients	Followed by 12 Bytes of Data[4, 12]
	1	0	1	0	0	1	0	0	Write X Coefficients	Followed by 8 Bytes of Data[4,11]
	1	0	1	0	1	0	0	0	Write R Coefficients	Followed by 8 Bytes of Data[4,11]
	1	0	1	0	1	1	0	0	Write Z Coefficients	Followed by 8 Bytes of Data[4, 11]
	1	0	1	0	0	0	1	1	Read B Coefficients	Followed by 12 Bytes of Data[4, 12]
	1	0	1	0	0	1	1	1	Read X Coefficients	Followed by 8 Bytes of Data[4, 11]
	1	0	1	0	1	0	1	1	Read R Coefficients	Followed by 8 Bytes of Data[4, 11]
	1	0	1	0	1	1	1	1	Read Z Coefficients	Followed by 8 Bytes of Data[4, 11]
	1	0	1	1	0	0	0	0	Reset to normal conditions[9]	
	1	0	1	1	0	0	0	1	Add −6 dB to receive gain	
	1	0	1	1	0	0	1	0	Cutoff receive path	
	1	0	1	1	0	1	1	1	Test mode–analog loop-back	
	1	0	1	1	0	1	0	0	Test mode–digital loop-back[13]	
	1	0	1	1	0	0	1	1	Disable High-Pass Filter (set to 1) and freeze auto zero circuit	
	1	0	1	1	1	0	0	0	Choose Linear code (Am7901A)/Choose A-law code (Am7901B)	
	1	0	1	1	1	0	0	1	Choose μ-law	
	1	1	0	I	J	K	L	M	Outputs to SLIC[10]	
	1	1	1	X	X	X	X	X	Reserved[2]	
	1	1	1	1	1	1	1	1	Power Up[1]	

NOTES:

1. During power-down the control information is not changed. The Serial I/O remains active, the SLIC control outputs remain valid, the PCM outputs are high impedance, the PCM inputs are disabled and the analog output is set to zero with a moderate series impedance to analog ground. Upon power-up, all data RAMs except the coefficient RAMs are powered up in a cleared state (set to all zeroes). No PCM data is transmitted until after the second FSX pulse is received following the execution of the power-up command.
2. These reserved codes are all codes beginning with 000 and 111 except for 00000000 (power-down) and 11111111 (power-up). These codes may be used by future members of this product family.

3. The Ys are binary codes which program the time slots for transmission and reception of PCM data. Five bits are available for time-slot selection which allow one of 32 time slots to be programmed. The three bits of the clock-slot selection allow 0 to 7 clock offsets within the time slot to be programmed.

4. All commands that are followed by additional input data to the device (transmit-gain selection, receive-gain selection, write B, Z, X or R coefficients) must have the input data as the next N words (N = 1, 2, 8, 12) written to the device (framed by the next N transitions of $\overline{CS}$). All commands that are followed by output data (read transmit time and clock slot, read transmit gain, read receive time and clock slot, read receive gain, read PCM mode, read B, Z, X or R coefficients) will cause the device to output data for the next N (N = 1, 2, 8, 12) transitions of $\overline{CS}$ going low and will not accept any input commands until all the data has been output. When in an input mode, data word of 00000000 will automatically power-down the device.

5. Time and clock slots are read out time slot first, followed by clock slot.

6. The PCM Modes are read out as the least significant 4 bits of data. The most significant 4 bits are set to 1. The least significant 4 bits contain the following data:

   BIT 3: Data Receive select bit
   BIT 2: Data Transmit select bit
   BIT 1: Receive Expanded Mode bit
   BIT 0: Transmit Expanded Mode bit

   The Data Receive/Transmit select bits define which port is used to receive/transmit data. A 0 means port A has been selected. A 1 means port B has been selected.

   The Receive/Transmit Expanded Mode bits allow up to 64 channels in a Receive/Transmit frame.

7. The operating function command has four 1-bit fields:
   A: A = 1 enables B filter, A = 0 disables B (sets B = 0)
   B: B = 1 enables X filters, B = 0 disables X (sets X = 1)
   C: C = 1 enables R filter, C = 0 disables R (sets R = 1)
   D: D = 1 enables Z filter, D = 0 disables Z (sets Z = 0)

8. The transmit PCM data may be output onto either the DXA or the DXB port. Either $\overline{TSCA}$ or $\overline{TSCB}$ is correspondingly output. The receive PCM data may be input onto either the DRA or the DRB port. The Transmit/Receive Expanded Mode bits allow up to 64 channels in Transmit/Receive frame.
   E: E = 1 chooses DRB, E = 0 chooses DRA
   F: F = 1 chooses DXB ($\overline{TSCB}$), F = 0 chooses DXA ($\overline{TSCA}$)
   G: G = 1 sets Receive Expanded Mode bit
      G = 0 clears Receive Expanded Mode bit
   H: H = 1 sets Transmit Expanded Mode bit
      H = 0 clears Transmit Expanded Mode bit

9. Normal conditions are receive gain set to value stored in the receive gain control words, the receive path and high-pass filter are enabled and the auto-zero-circuit operates, Z filter coefficients are the value set by the basic and operating function bit D and the device is not in a test mode (no loop-back). The test modes are mutually exclusive. Entering a command to set one test mode clears the other test mode (if set). "Reset to normal conditions" does reset a test mode.

10. The outputs to the SLIC are defined below:
    I = C5 L = C2
    J = C4 M = C1
    K = C3

11. X, R, and Z coefficients are allowed to have only 1 to 4 ones. Each coefficient is encoded in a 4-bit code where the lower three bits represent the number of shifts to the next higher one in the coefficient and the first bit (MSB) defines the coefficient sign. Each one can be either positive or negative (0 = positive, 1 = negative). The maximum number of shifts allowed is six. The lower three bits are encoded for 0(111), 1(110), 2(101), 3(100), 4(011), 5(010) or 6(001) shifts. A code of 1000 implies 0 shifts and no addition and a code of 0000 is not allowed (See note 4). The four coefficients use sixteen 4-bit codes which are input as eight 8-bit words starting with coefficients 0 and ending with coefficient 3 for the X coefficients. The R and Z filter coefficient data starts with coefficient 3 and ends with coefficient 0.

12. B coefficients are allowed to have only 1 to 3 ones. Each coefficient is encoded in a 4-bit code where the lower three bits represent the number of shifts to the next higher one in the coefficient and the first bit (MSB) defines the coefficient sign. Each one can be either positive or negative (0 = positive, 1 = negative). The maximum number of shifts allowed is six. The lower three bits are encoded for 0(111), 1(110), 2(101), 3(100), 4(011), 5(010) or 6(001) shifts. A code of 1000 implies 0 shifts and no addition and a code of 0000 is not allowed (See note 4). The eight coefficients use twenty-four 4-bit codes which are input as twelve 8-bit words starting with coefficient 0 and ending with coefficient 7.

13. Digital loop-back provides 6 dB of gain.

ABSOLUTE MAXIMUM RATINGS

Storage Temperature ..... −60 to 125°C
Ambient Temperature, under Bias ..... 0 to 70°C
V_{CC} with Respect to DGND ..... −0.4 to +6.0 V
V_{BB} with Respect to DGND ..... +0.4 to −6.0 V
V_{IN} with Respect to AGND ..... V_{BB} to V_{CC}

Stresses above those listed under ABSOLUTE MAXIMUM RATINGS may cause permanent device failure. Functionality at or above these limits is not implied. Exposure to absolute maximum ratings for extended periods may affect device reliability.

OPERATING RANGES

Part Number	Ambient Temperature	V_{CC}	V_{BB}	DGND	AGND
Am7901A/BDC	0°C ≤ T_A ≤ 70°C	+5.0 V±5%	−5.0 V±5%	0 V	0 V±100 mV

Operating ranges define those limits over which the functionality of the device is guaranteed.

DC SPECIFICATIONS

ELECTRICAL CHARACTERISTICS over operating range (Note 1) unless otherwise specified

Parameters	Description	Test Conditions	Min.	Typ.	Max.	Units
Z_{IN}	Analog Input Impedance	−3.2 V < V_{IN} < 3.2 V	20			kΩ
Z_{OUT}	Analog Output Impedance	−3.2 V < V_{OUT} < 3.2 V			20	Ω
V_{IOS}	Offset Voltage Allowed on V_{IN}				±5	mV
V_{OOS}	Analog Output Offset Voltage				±200	mV
V_{IR}	Analog Input Voltage Range				±3.2 V	V
V_{OR}	Analog Output Voltage Range	R_L ≥ 10 kΩ, C_L ≤ 50 pF			±3.2 V	V
I_{OUT}	Analog Output Current		350			μA
V_{IL}	Input Low Voltage (All Digital Inputs Except DCLK in Stand Alone Mode)		−0.5		0.8	V
V_{IH}	Input High Voltage (All Digital Inputs)		2.0		V_{CC}	V
V_{OL}	Output Low Voltage (All Digital Outputs)	I_{OL} = 2 mA			0.45	V
V_{OH}	Output High Voltage (All Outputs Except TSC)	I_{OH} = 400 μA	2.4			V
I_{OL}	Output Leakage Current				±10	μA
I_{IL}	Input Leakage Current				±1	μA
I_{IL} (V_{IN})	Input Leakage Current on V_{IN} Pin				±0.2	μA
I_{CC} (S)	V_{CC} Supply Current (Standby)	V_{CC} = 5.25 V V_{BB} = −4.75 V			15	mA
I_{BB} (S)	V_{BB} Supply Current (Standby)				10	mA
I_{CC} (A)	V_{CC} Supply Current (Active)				60	mA
I_{BB} (A)	V_{BB} Supply Current (Active)				20	mA
PSRR (V_{CC})	V_{CC} Power Supply Rejection Ratio	200 mV p-p @ 1.02 kHz on the appropriate supply	35			dB
PSRR (V_{BB})	V_{BB} Power Supply Rejection Ratio		30			dB
C_I	Input Capacitance (Digital)			5		pF
C_O	Output Capacitance (Digital)			8		pF

Notes: 1. Typical values are for T_A = 25°C and nominal supply voltages. Min. and max. specifications are over the temperature and supply voltage ranges shown in the above table entitled "Operating Ranges."

TRANSMISSION CHARACTERISTICS

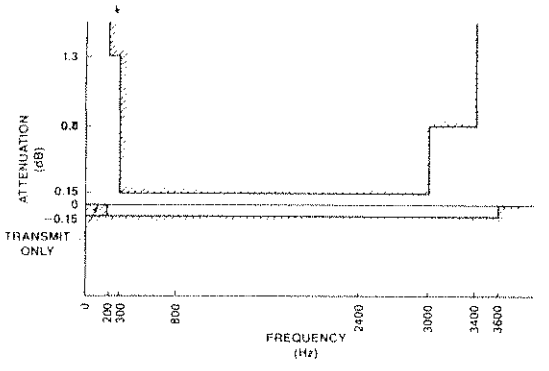
(All measurements are made end-to-end with GX = GR = 0 dB and A-law or μ-law companded PCM unless otherwise specified.)

A 0-dBm0 signal at V_{IN} is equivalent to 1.57 V_{RMS}. A 3-dBm0 signal at V_{IN} is equivalent to 2.22 V_{RMS} which corresponds to the overload point of 3.14 volts.

A 0-dBm0 signal at V_{OUT} is equivalent to 1.6 V_{RMS}. A 3-dBm0 signal at V_{OUT} is equivalent to 2.260 V_{RMS} which corresponds to the overload point of 3.196 volts.

Description	Test Conditions	Min	Typ	Max	Units
Attenuation Distortion	800 Hz at 0 dBm0, or 1000 Hz at 0 dBm0		See Fig 12		dB
Gain[1] (either path) a) deviation from ideal value b) deviation from initial value	800 Hz at 0 dBm0, or 1000 Hz at 0 dBm0	 −0.2 −0.2		 +0.2 +0.2	 dB dB
Group Delay Distortion (either path)	0-dBm0 signal		See Fig 13		μs
Group Delay (either path)			150		μs
Harmonic Distortion	(Note 2)			−40	dB
Intermodulation Distortion	a) (Note 3) b) (Note 4)			−35 −49	dB dBm0
Crosstalk a) Go-to-Return path b) Return-to-Go path	300-3400 Hz 0 dBm0 300-3400 Hz 0 dBm0			−70 −70	dB dB
Gain Tracking (either path)			See Fig 14		dB
Signal to Total Distortion (either path)			See Fig 15		dB
μ-Law Companded PCM					
Idle Channel Noise (weighted)	(Note 5)			19	dBrnc0
Idle Channel Noise (weighted, receive only)				15	dBrnc0
Idle Channel Noise (single frequency)				−50	dBm0
A-Law Companded PCM					
Idle Channel Noise (weighted)	(Note 5)			−71	dBm0p
Idle Channel Noise (weighted, receive only)				−75	dBm0p
Idle Channel Noise (single frequency)				−50	dBm0

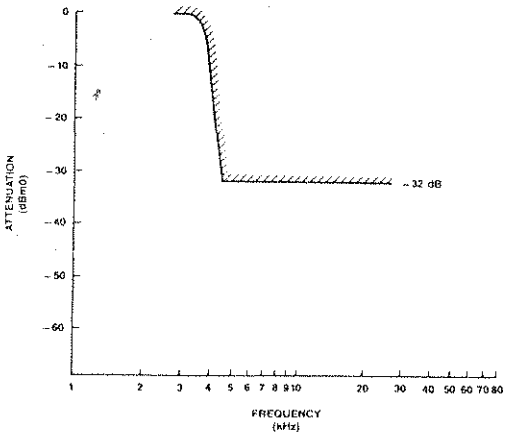
Notes: 1. The device gains are adjusted during manufacture to guarantee a ±0.4 − dB maximum deviation over lifetime of device.
2. Applied signal is a 0-dBm0 sine wave within 300 to 3400 Hz. The signal measured is any frequency in the range 300 to 3400 Hz.
3. Two different frequencies f_1 and f_2 in the range 300-3400 Hz and of equal levels in the range −4 to −21 dBm0 are applied. $2f_1-f_2$ products are measured relative to the level of either f_1 or f_2.
4. Any intermodulation product due to a signal in the range 300-3400 Hz with input level −9 dBm0 and a 50-Hz signal with input level −23 dBm0.
5. Noise is measured at the analog output, with the analog input zero and the digital PCM output connected to the digital PCM input.



OP001700

Figure 12a. Attenuation Distortion (Single Ended)

Notes: 1. The frequency is 800/1000 Hz.
2. Input signal level is 0 dBm0.



OP001712

Figure 12b. Out of Band Signals (End-to-End)

Notes: 1. The frequency is 800/1000 Hz.
2. Input signal level is 0 dBm0.

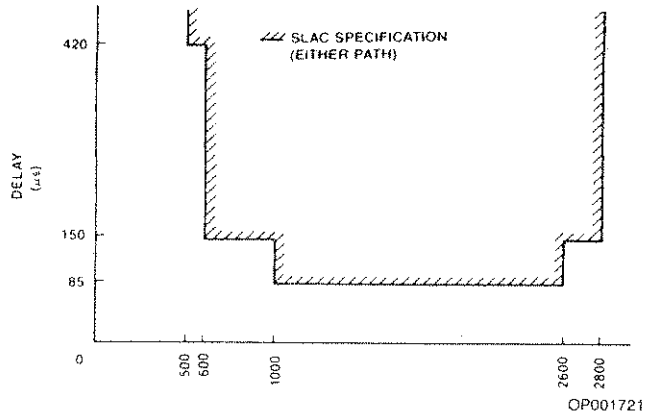


Figure 13. Group Delay Distortion (Either Path)

Notes: 1. Input signal is 0 dBm0.
2. Minimum value of group delay is taken as reference.

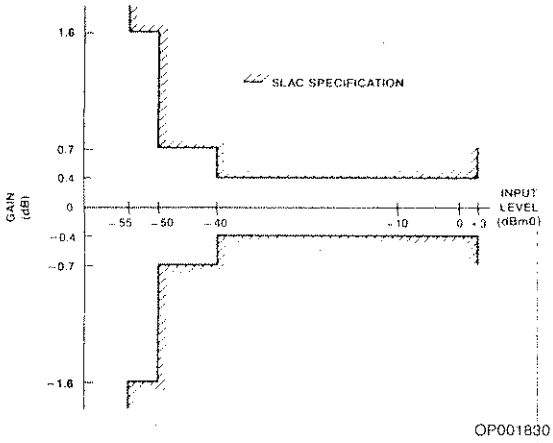


Figure 14. Gain Tracking with Tone (Either Path)

Notes: 1. The input signal is a sine wave in the range of 700 to 1100 Hz, (excluding submultiples of 8 kHz).
2. The gain variation is relative to the gain at −10 dBm0.

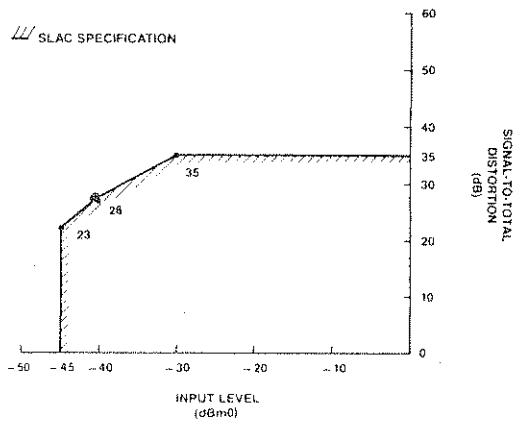


Figure 15. Signal-to-Total Distortion with Tone (Either Path)

Note: The input signal is a sine wave in the range of 700 to 1100 Hz, (excluding submultiples of 8 kHz).

SWITCHING CHARACTERISTICS over operating range unless otherwise specified

$T_A = 0$ to 70°C, $V_{CC} = +5$ V ±5%, $V_{BB} = -5$ V ±5% (See Notes 1, 7, 8)

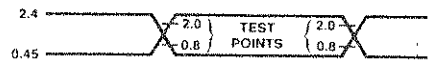
No.	Parameters	Description	Min.	Typ.	Max.	Units
Serial Interface Input Mode						
1	t_{DCH}	Data Clock High Pulse Width (Note 2)	0.220		20	µs
2	t_{DCL}	Data Clock Low Pulse Width (Note 2)	0.220			µs
3	t_{DCR}	Rise Time of Clock	5		50	ns
4	t_{DCF}	Fall Time of Clock	5		50	ns
5	t_{ICSS}	Chip Select Setup Time	150			ns
6	t_{ICSH}	Chip Select Hold Time	50			ns
7	t_{ICSL}	Chip Select Pulse Width (Notes 3 & 9)		8 t_{DCY}		ns
8	t_{ICSO}	Chip Select Off Time (Note 4)				ns
9	t_{IDS}	Input Data Setup Time	50			ns
10	t_{IDH}	Input Data Hold Time	30			ns
11	t_{OLH}	Output Latch Propagation Delay	0.75		1.9	µs
Serial Interface Output Mode						
12	t_{OCSS}	Chip Select Setup Time	150			ns
13	t_{OCSH}	Chip Select Hold Time	50			ns
14	t_{OCSL}	Chip Select Pulse Width (Notes 3 & 9)		8 t_{DCY}		ns
15	t_{OCSO}	Chip Select Off Time (Note 4)				
16	t_{ODD}	Output Data Turn on Delay			100	ns
17	t_{ODH}	Output Data Hold Time	30			ns
18	t_{DDOF}	Output Turn off Delay			100	ns
19	t_{ODC}	Output Data Valid	30		150	ns
PCM Interface						
20	t_{PCY}	PCM Clock Period (Note 5)	0.244		7.8	µs
21	t_{PCH}	PCM Clock Pulse Width (Note 5)	110			ns
22	t_{PCL}	PCM Clock Low Pulse Width (Note 5)	110			ns
23	t_{PCF}	Fall Time of Clock	5		15	ns
24	t_{PCR}	Rise Time of Clock	5		15	ns
25	t_{FSS}	Frame Sync Setup Time	50		(t_{PCY} - 30)	ns
26	t_{FSH}	Frame Sync Hold Time (Companded Mode)	30		(8 t_{PCY} - 50)	ns
		Frame Sync Hold Time (Linear Mode)	30		(16 t_{PCY} - 50)	ns
27	t_{TSD}	Delay to TSC Valid (Note 6)	(N t_{PCY} + 30)		(N t_{PCY} + 150)	ns
28	t_{TSO}	Delay to TSC Off	30			ns
29	t_{DXD}	PCM Data Output Delay	80		150	ns
30	t_{DXH}	PCM Data Output Hold Time	30		100	ns
31	t_{DXZ}	PCM Data Output Delay to High Z	40		75	ns
32	t_{DRS}	PCM Data Input Setup Time	50			ns
33	t_{DRH}	PCM Data Input Hold Time	30			ns
Master Clock						
34	t_{MCY}	Master Clock Period	488.23	488.28	488.33	ns
35	t_{MCH}	Master Clock High Pulse Width	220			ns
36	t_{MCL}	Master Clock Low Pulse Width	238			ns
37	t_{MCR}	Rise Time of Clock	5		15	ns
38	t_{MCF}	Fall Time of Clock	5		15	ns

Notes:
1. Min. and Max. values are valid on all digital outputs except C_1–C_5 with a 150-pF load. C_1–C_5 outputs are valid with a 30-pF load.
2. The Data Clock may be stopped in the Low state indefinitely without loss of information. Data will not be clocked in or out while the clock is in the low state.
3. Chip Select Pulse Width is nominally 8 Data Clock Cycles with a minimum value of 7 Data Clock Cycles + t_{ICSH} + t_{ICSS} and a maximum value of 9 Data Clock Cycles – t_{ICSH} – t_{ICSS}
4. Chip Select Off Time is defined by the type of command being executed. Commands attempting access to the coefficient RAMs, i.e., Read or Write B, Z, X, R or gain coefficients must have a minimum Chip Select Off Time of:
   7 t_{MCY} – if device is in power-down mode.
   32 t_{MCY} – if device is in power-up mode.
   For all other commands, Chip Select Off Time is defined as a minimum of:
   7 t_{MCY} – for both power-up or power-down modes.
5. The maximum allowed PCM clock frequency is 4.096 MHz. The actual PCM clock rate is dependent on the number of channels allocated within a frame. The minimum clock frequency is 128 kHz.
6. TSC is delayed from FS by a typical value of N t_{PCY}, where N is the value stored in the Time/Clock Slot register.
7. The Frame Sync pulses (FSX, FSR) repeat at an 8-kHz rate.
8. FSR, FSX, CLKR, CLKX and MCLK all must be synchronized and exactly 256 cycles of MCLK must be guaranteed between Frame Syncs. All five clocks must not be interrupted to assure proper operation.
9. t_{DCY} is 1 Data Clock Cycle.

SWITCHING WAVEFORMS

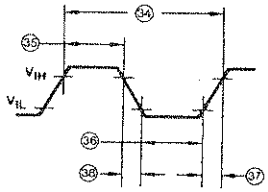
TIMING DIAGRAMS

INPUT AND OUTPUT WAVEFORMS FOR AC TESTS



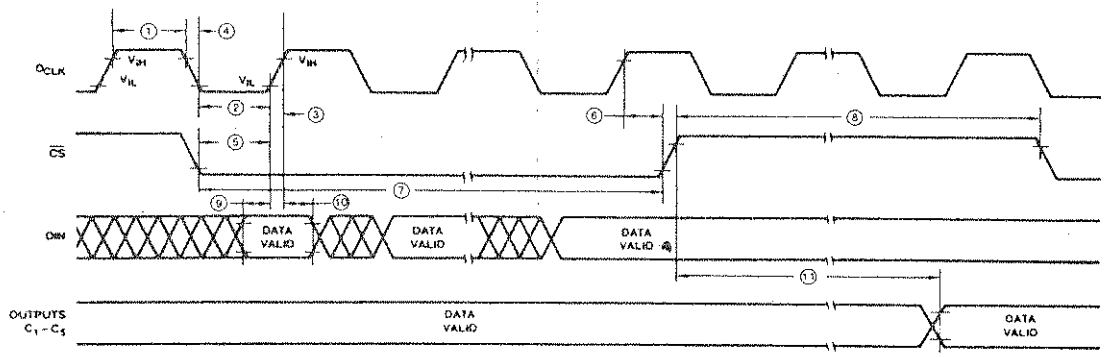
WF009820

MASTER CLOCK TIMING



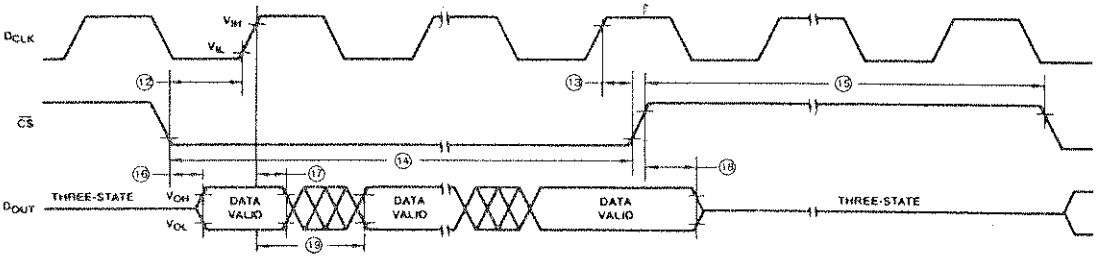
WF009832

SERIAL I/O INTERFACE (INPUT MODE)



WF009841

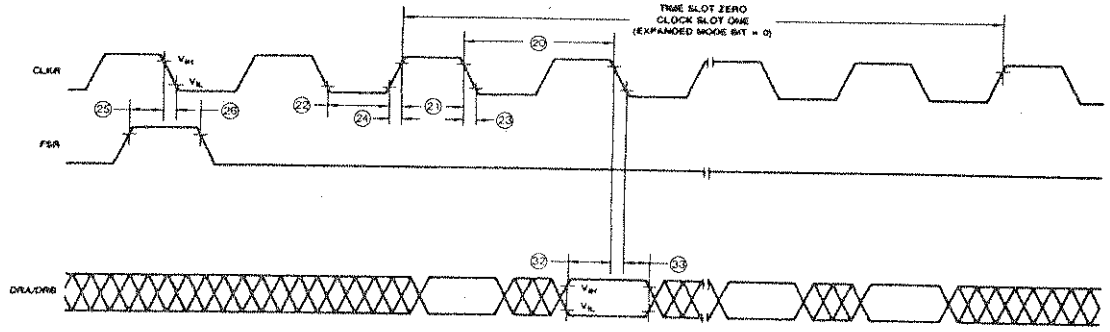
SERIAL I/O INTERFACE (OUTPUT MODE)



WF009852

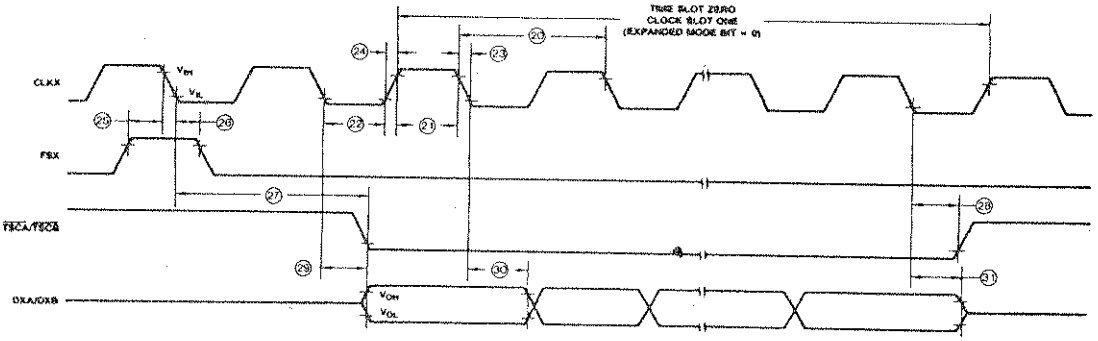
SWITCHING WAVEFORMS (Cont.)

PCM INPUT TIMING



WF009862

PCM OUTPUT TIMING



WF009872

Z8060
Z8000™ Z-FIFO Buffer Unit and Z-FIO Expander

Zilog

Product Specification

September 1983

Features

- Bidirectional, asynchronous data transfer capability
- Large 128-bit-by-8-bit buffer memory
- Two-wire, interlocked handshake protocol
- Wire-ORing of empty and full outputs for sensing of multiple-unit buffers
- 3-state data outputs
- Connects any number of FIFOs in series to form buffer of any desired length
- Connects any number of FIFOs in parallel to form buffer of any desired width

General Description

The Z8060 First-In First-Out (Z-FIFO) Buffer Unit consists of a 128-bit-by-8-bit memory, bidirectional data transfer and handshake logic. The structure of the Z-FIFO unit is similar to that of other available buffer units. Z-FIFO is a general-purpose unit; its handshake logic is compatible with that of other members of Zilog's Z8 and Z8000 Families.

Z-FIFOs can be cascaded end-to-end without limit to form a parallel 8-bit buffer of any desired length (in 128-byte increments). Any number of single- or multiple-unit Z-FIFO serial buffers can be connected in parallel to form buffers of any desired width (in 8-bit increments).

The Z-FIFO buffer units are available as 28-pin packages. Figures 1 and 2 show the pin functions and pin assignments, respectively, of the Z-FIFO device. A block diagram is shown in Figure 3.

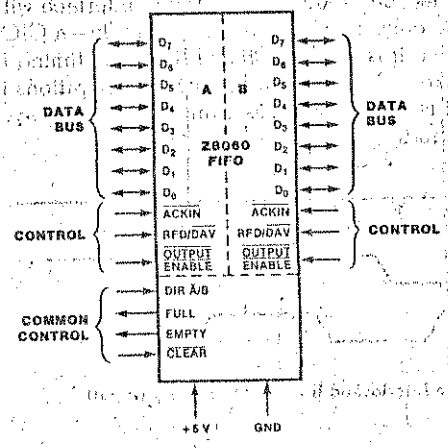


Figure 1. FIFO Pin Functions

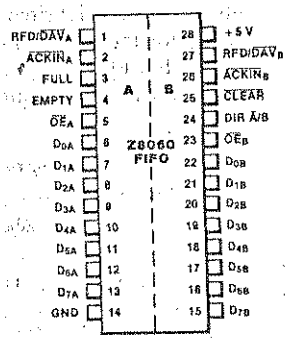


Figure 2. FIFO Pin Assignments

General Description (Continued)

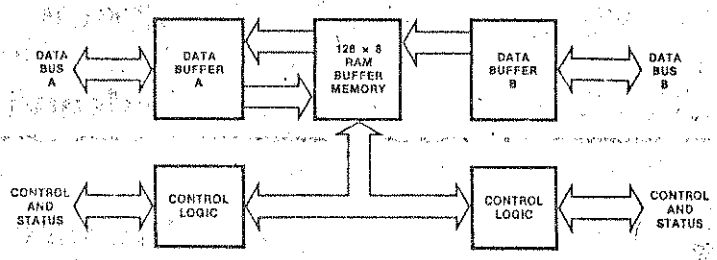


Figure 3. Functional Block Diagram

Pin Descriptions

$\overline{ACKIN}$. *Acknowledge Input* (input, active Low). This line signals the FIFO that output data has been received by peripherals or that input data is valid.

$\overline{CLEAR}$. *Clear Buffer* (input, active Low). When set to Low, this line causes all data to be cleared from the FIFO buffer.

D_0-D_7. *Data Bus* (inputs/outputs, bidirectional). These bidirectional lines are used by the FIFO to receive and to transmit data.

DIR $\overline{A}$/B. *Direction Input A/B* (input, two control states). A High on this line signals that input data is to be received at Port B. A Low on this line signals that input data is to be received at Port A.

EMPTY. *Buffer Status* (output, active High, open-drain). A High on this line indicates that the FIFO buffer is empty.

FULL. *Buffer Status* (output, active High, open-drain). A High on this line indicates that the FIFO buffer is full.

$\overline{OEA}$, $\overline{OEB}$. *Output Enable A, Output Enable B* (inputs, active Low). When Low, $\overline{OEA}$ enables the bus drivers for Port A; when High, $\overline{OEA}$ causes the bus drivers to float to a high-impedance level. Input $\overline{OEB}$ controls the bus drivers for Port B in the same manner as $\overline{OEA}$ controls those for Port A.

RFD/$\overline{DAV}$. *Ready-for-Data/Data Available* (outputs RFD, active High; $\overline{DAV}$ active Low). RFD, when High, signals to the peripherals involved that the FIFO is ready to receive data. $\overline{DAV}$, when Low, signals to the peripherals involved that FIFO has data available to send.

Functional Description

Interlocked 2-Wire Handshake. In interlocked 2-wire handshake operation, the action of FIFO must be acknowledged by the other half of the handshake before the next action can occur. In an Output Handshake mode, the FIFO indicates that new data is available only after the external device has indicated that it is ready for the data. In an Input Handshake mode, the FIFO does not indicate that it is ready for new data until the data source indicates that the previous byte of the data is no longer available, thereby acknowledging the acceptance of the last byte. This control feature allows the FIFO, with no external logic, to directly interface with the port of any CPU in the Z8 Family—a CIO, a UPC, an FIO, or another FIFO. The timing for the input and output handshake operations is shown in Figures 4 and 5, respectively.

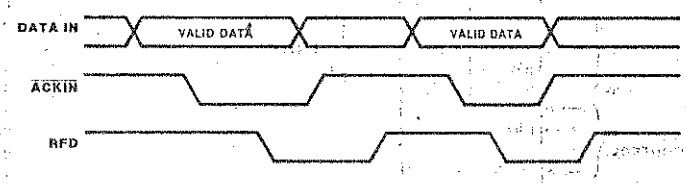


Figure 4. Two-Wire Interlocked Handshake Timing (Input)

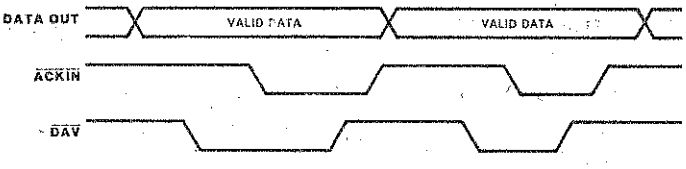


Figure 5. Two-Wire Interlocked Handshake Timing (output)

Functional Description (Continued)

Resetting or Clearing the FIFO. The $\overline{CLEAR}$ input is used to initialize and clear the FIFO. A Low level on this input clears all data from the FIFO, allows the EMPTY output to go High and forces both outputs $RFD/\overline{DAV}_A$ and $RFD/\overline{DAV}_B$ High. A High level on $\overline{CLEAR}$ allows the data to transfer through the FIFO.

Bidirectional Transfer Control. The FIFO has bidirectional data transfer capability under control of the DIR $\overline{A}$/B input. When DIR $\overline{A}$/B is set Low, Port $\overline{A}$ becomes input handshake and Port B becomes output handshake; data transfers are then made from Port A to Port B. Setting DIR $\overline{A}$/B High reverses the handshake assignments and the direction of transfer. This bidirectional control is illustrated in Table 1.

DIR $\overline{A}$/B	Port A Handshake	Port B Handshake	Transfer
0	Input	Output	A to B
1	Output	Input	B to A

Table 1. Bidirectional Control Function Table

The FIFO buffer must be empty before the direction of transfer is changed; otherwise, the results of the change will be unpredictable. If FIFO status is unknown when a transfer direction change is to be made, the recommended procedure is:

(1) Force and hold $\overline{CLEAR}$ Low.

(2) Set DIR $\overline{A}$/B to the level required for the desired direction.

(3) Force $\overline{CLEAR}$ High.

Empty and Full Operation. The EMPTY and FULL output lines can be wire-ORed with the EMPTY and FULL lines of other FIFOs and FIOs. This capability enables the user to determine the empty/full status of a buffer consisting of multiple FIFOs, FIOs, or a combination of both. Table 2 shows the various states of EMPTY and FULL.

Number of Bytes in FIFO	EMPTY	FULL
0	High	Low
1-127	Low	Low
128	Low	High

Table 2. Signals EMPTY and FULL Operation Table

Interconnection Example. Figure 6 illustrates a simplified block diagram showing the manner in which FIFOs can be interconnected to extend a FIO buffer.

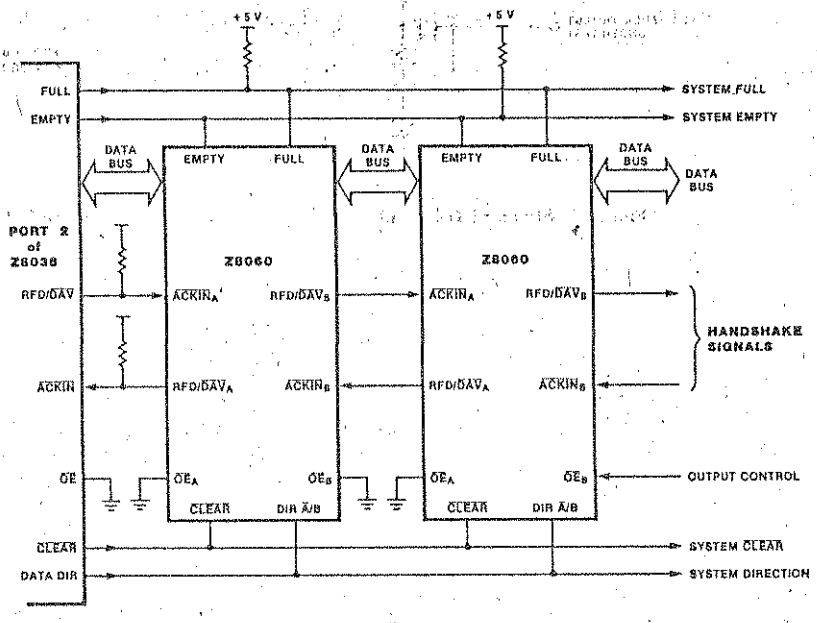


Figure 6. Typical Interconnection (Simplified Diagram)

Functional Description (Continued)

Output Enable Operation. The FIFO provides a separate Output Enable ($\overline{OE}$) signal for each port of the buffer. An $\overline{OE}$ output is valid only when its port is in the Output Handshake mode. The control of this output function is shown in Table 3. Signal $\overline{OE}$ operates with lines DIR $\overline{A}/B$. A High on a valid $\overline{OE}$ line 3-states its port's data bus but does not affect the handshake operation. A Low level on a valid $\overline{OE}$ enables the data bus outputs if its port is in the Output Handshake mode. Note that the handshake operation is unaffected by the Output Enable pin.

DIR $\overline{A}/B$	$\overline{OE}_A$	$\overline{OE}_B$	Function
0	X	0	Disable Port A Output Enable Port B Output
0	X	1	Disable Port A Output Disable Port B Output
1	0	X	Enable Port A Output Disable Port B Output
1	1	X	Disable Port A Output Disable Port B Output

NOTE: X = Don't care.

Table 3. Output Control Function Table

Absolute Maximum Ratings

Voltages on all inputs and outputs with respect to GND .....-0.3 V to +7.0 V
Operating Ambient Temperature.....As specified in Ordering Information
Storage Temperature.....-65° to +150°C

Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Standard Test Conditions

The characteristics below apply for the following standard test conditions, unless otherwise noted. All voltages are referenced to GND. Positive current flows into the referenced pin. Standard conditions are as follows:

- $+4.75\ V \le V_{CC} \le +5.25\ V$
- GND = 0 V
- T_A as specified in Ordering Information. All ac parameters assume a load capacitance of 50 pF max.

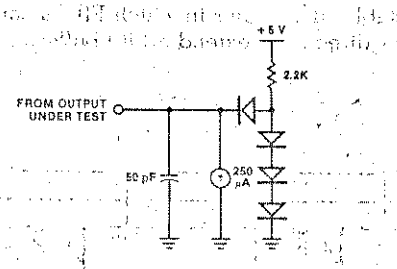


Figure 7. Standard Test Load

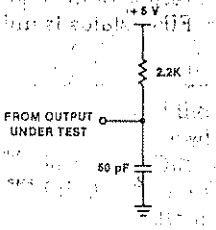


Figure 8. Open-Drain Test Load

DC Characteristics

Symbol	Parameter	Min	Max	Unit	Condition
V_{IH}	Input High Voltage	2.0	$V_{CC}+0.3$	V	
V_{IH}	Input Low Voltage	-0.3	0.8	V	
V_{OH}	Output High Voltage	2.4		V	I_{OH} = -250 A
V_{OL}	Output Low Voltage		0.4	V	I_{OL} = +2.0 mA
			0.5	V	I_{OL} = +3.2 mA
I_{IL}	Input Leakage		±10	μA	$0.4 \le V_{IN} \le +2.4$ V
I_{OL}	Output Leakage		±10	μA	$0.4 \le V_{OUT} \le +2.4$ V
I_{CC}	V_{CC} Supply Current		200	mA	

NOTE: V_{CC} = +5 V ±5% unless otherwise specified over specified temperature range.

Capacitance

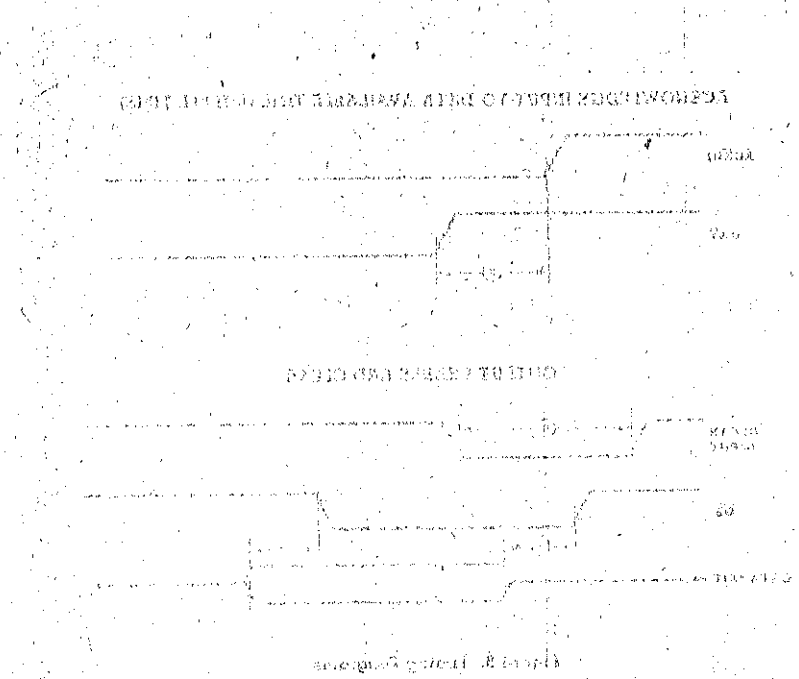
Symbol	Parameter	Min	Max	Unit	Test Condition
C_{IN}	Input Capacitance		10	pF	Unmeasured pins
C_{OUT}	Output Capacitance		15	pF	returned to ground
$C_{I/O}$	Bidirectional Capacitance		20	pF	
Input					
tr	Any input rise time		100	ns	
tf	Any input fall time		100	ns	

NOTE: f = 1 MHz over specified temperature range.

Ordering Information

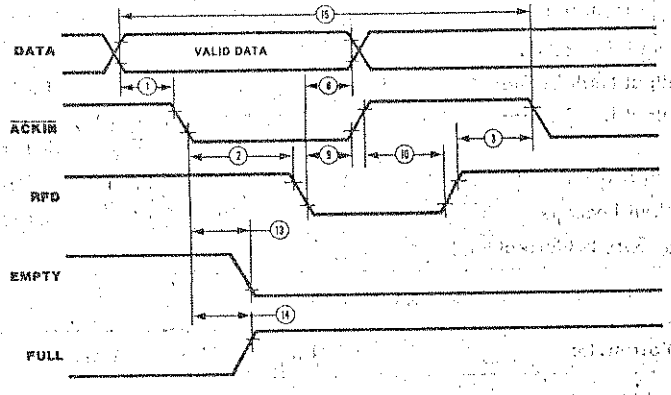
Product Number	Package/ Temp	Speed	Description	Product Number	Package/ Temp	Speed	Description
Z8060	CE	4.0 MHz	FIFO (28-pin)	Z8060	DS	4.0 MHz	FIFO (28-pin)
Z8060	CS	4.0 MHz	Same as above	Z8060	PE	4.0 MHz	Same as above
Z8060	DE	4.0 MHz	Same as above	Z8060	PS	4.0 MHz	Same as above

NOTES: C = Ceramic, D = Cerdip, P = Plastic; E = -40°C to +85°C, S = 0°C to 70°C

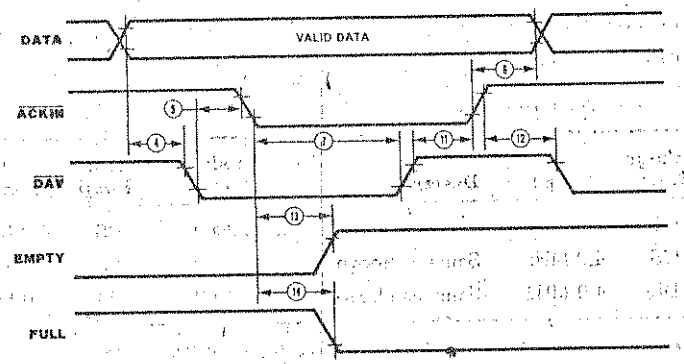


2-Wire Interlocked Handshake Timing

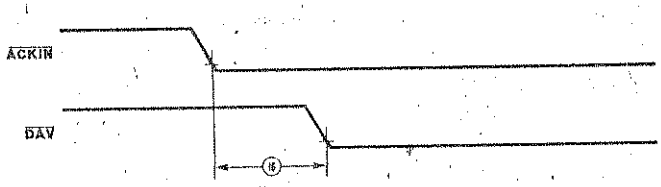
INPUT TIMING



OUTPUT TIMING



ACKNOWLEDGE INPUT TO DATA AVAILABLE TIME (BUBBLE TIME)



OUTPUT ENABLE AND CLEAR

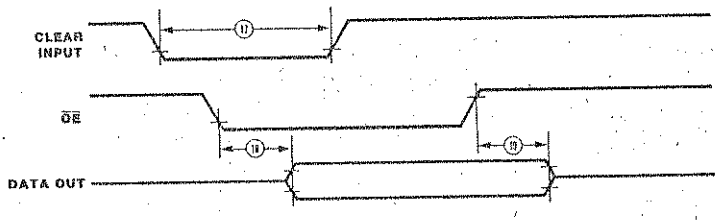


Figure 9. Timing Diagrams

FIFO 2-Wire Handshake Timing. Timing for 2-wire interlocked handshake operation is shown in Figure 9. The symbol, description and values for the numbered parameters (Figure 9) are given in AC Characteristics.

AC Characteristics

No.	Symbol	Parameter	Min	Max	Units*
1	TsDI(ACK)	Data Input to $\overline{ACKIN}$ ↓ to Setup Time	50		ns
2	TdACKI(RFD)	$\overline{ACKIN}$ ↓ to RFD ↓ Delay	0		ns
3	TdRFDr(ACK)	RFD ↑ to $\overline{ACKIN}$ ↓ Delay	0		ns
4	TsDO(DAV)	Data Out to $\overline{DAV}$ ↓ Setup Time	50		ns
5	TdDAVI(ACK)	$\overline{DAV}$ ↓ to $\overline{ACKIN}$ ↓ Delay	0		ns
6	ThDO(ACK)	Data Out to $\overline{ACKIN}$ ↑ Hold Time	50		ns
7	TdACK(DAV)	$\overline{ACKIN}$ ↓ to $\overline{DAV}$ ↑ Delay	0		ns
8	ThDI(RFD)	Data Input to RFD ↓ Hold Time	0		ns
9	TdRFDI(ACK)	RFD ↓ to $\overline{ACKIN}$ ↑ Delay	0		ns
10	TdACKr(RFD)	$\overline{ACKIN}$ ↑ to RFD ↑ Delay	0		ns
11	TdDAVr(ACK)	$\overline{DAV}$ ↑ to $\overline{ACKIN}$ ↑	0		ns
12	TdACKr(DAV)	$\overline{ACKIN}$ ↑ to $\overline{DAV}$ ↓	0		ns
13	TdACKINI(EMPTY)	(Input) $\overline{ACKIN}$ ↓ to EMPTY ↑ Delay (Output) $\overline{ACKIN}$ ↓ to EMPTY ↑ Delay			
14	TdACKINI(FULL)	(Input) $\overline{ACKIN}$ ↓ to FULL ↑ Delay (Output) $\overline{ACKIN}$ ↓ to FULL ↓ Delay			
15	ACKIN Clock Rate	(Input or Output)	1.0		MHz
16	TdACKINI(DAVI)	(Bubble Time)			ns
17	TwCLR	Width of Clear to Reset FIFO	700		ns
18	TdOE(DO)	$\overline{OE}$ ↓ to Data Bus Driven	0		ns
19	TdOE(DRZ)	$\overline{OE}$ ↑ to Data Bus Float			ns

NOTES:
* All timing references assume 2.0 V for a logic 1 and 0.8 V for a logic 0. Timings are preliminary and subject to change.