THE UNIVERSITY OF ALBERTA

DIGITAL IMPLEMENTATION OF A TRANSMIT LINE BUILD-OUT CIRCUIT FOR THE DSX-1 INTERFACE

by

MYRON BORYS

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES AND RESEARCH

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE

OF MASTER OF SCIENCE

DEPARTMENT OF ELECTRICAL ENGINEERING

EDMONTON, ALBERTA

FALL 1988

ISBN 0-315-45631-0

THE UNIVERSITY OF ALBERTA

RELEASE FORM

NAME OF AUTHOR: MYRON BORYS

TITLE OF THESIS: DIGITAL IMPLEMENTATION OF A TRANSMIT LINE BUILD-OUT CIRCUIT FOR THE DSX-1 INTERFACE

DEGREE: MASTER OF SCIENCE

YEAR THIS DEGREE GRANTED: FALL 1988

Permission is hereby granted to THE UNIVERSITY OF ALBERTA LIBRARY to reproduce single copies of this thesis and to lend or sell such copies for private, scholarly or scientific research purposes only.

The author reserves other publication rights, and neither the thesis nor extensive extracts from it may be printed or otherwise reproduced without the author's written permission.

Myron Borys
(Author's Signature)

8703 - 42 Avenue
Edmonton, Alberta, Canada
T6K 1E8

Date: 20 Sept 1988

THE UNIVERSITY OF ALBERTA

FACULTY OF GRADUATE STUDIES AND RESEARCH

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled DIGITAL IMPLEMENTATION OF A TRANSMNT LINE BUILD-OUT CIRCUIT FOR THE DSX-1 INTERFACE submitted by MYRON BORYS in partial fulfillment of the requirements for the degree of MASTER of SCIENCE.

W. [illegible]
(Supervisor)

Keith A. Stromme

A.S. [illegible]

Date 14 Sept. 1988

ABSTRACT

This thesis describes the design and construction of an all digital DS-1 transmitter line build-out circuit (LBO), which eliminates manual control of LBO settings. The application of the LBO is to compensate for the length variations, ranging from 0 m to 200 m, of 22 American Wire Gauge (AWG) twisted pair cable between a DS-1 line card and the DSX-1 cross-connect. At the cross-connect, the signal must fit into a DSX-1 standard pulse shape template. Current technology uses linear equalizer networks to pre-distort the DS-1 pulse shape at the transmitter so that the pulse shape at the cross-connect fits inside the DSX-1 pulse mask. The equalizers require several settings to meet the standard over the full range of allowable cable lengths, with each setting valid for a limited range of cable length. This thesis reports a technique which generates the pre-distorted DS-1 pulse shape digitally from numerical samples stored in an EPROM. The samples are passed to a digital to analog converter which generates the bipolar DS-1 signal. To meet the DSX-1 standard, it is sufficient to synthesize the waveform with 8 samples per DS-1 symbol interval (for a 12.352 MHz sampling rate), with 5 bits of resolution for each sample (for 32 quantization levels). One unique version of the pre-distorted DS-1 pulse has been found which meets the DSX-1 standard for the full 200 m range of 22 AWG cable. This solution eliminates record keeping and the need for human intervention associated with multiple, manually controlled LBO settings, resulting in cost savings for telephone companies. The solution also simplifies 1:N protection because all LBOs are identically set, which may not be the case with current LBOs. The low circuit speed and the all digital nature of the LBO mean that the circuit can be easily integrated on a single IC, and possibly multiple LBOs could be placed on one chip for application in DS-1 demultiplexers. The digital pulse shape generation technique has extended applications as well, such as allowing the use of 24 and 26 AWG cables, and for advanced test equipment.

ACKNOWLEDGEMENTS

I wish to thank the following for their financial assistance:

- Alberta Telecommunications Research Centre (ATRC)
- University of Alberta
- Natural Sciences and Engineering Research Council of Canada
- Alberta Heritage Scholarship Fund

I also thank my supervisor, Dr. Witold Krzymien, for his guidance and many fruitful discussions, and Wayne Grover for the initial idea of the project.

I am indebted to the ATRC for allowing me to use their excellent facilities to conduct my research, and to the technical and secretarial staff of the ATRC for their help throughout the project.

Finally, I would like to thank my beautiful wife, Linda, for her patience and understanding in allowing me to fulfil my goals.

TABLE OF CONTENTS

LIST OF TABLES

Table Page

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

This thesis describes the design and construction of a DSX-1 line build-out circuit (LBO) which uses digital pulse generation, with the intention to eliminate the need for manual control of the LBO. The LBO predistorts a DS-1 signal before transmission over a 22 American ire Gauge (AWG) cable such that the signal meets certain electrical specifications at a reference point called a cross-connect, which may be located up to 200 m from the signal source. The proposed LBO design generates the pre-distorted waveform by passing samples stored in an EPROM to a digital to analog converter (D/A) at a fixed rate several times the DS-1 data rate.

Current LBO technology has resulted in a single chip LBO circuit. Internal, analog equalizer networks are used to pre-distort the DS-1 signal. Up to six different equalizer settings, each valid for a limited range of line lengths, are needed to meet the DSX-1 electrical specifications at the cross-connect for the full 200 m range of possible line lengths. Each setting must be selected manually for the particular length of cable connected between the LBO and the cross-connect. The manual control requires that records be kept, and complicates the process of changing equipment configurations. These factors increase the operating costs of a telephone company, and it would be beneficial if manual control were eliminated. The proposed digital technique can generate pulse shapes which are impossible to achieve using linear, analog equalizers. This factor promises to reduce the number of LBO settings required, and, in turn, to facilitate the removal of manual control.

In Chapter 2, background information on the DS-1 signal, and the DSX-1 standard is presented. In this chapter, current LBO technology is reviewed, and the details of the proposed solution are introduced. The design process began with computer simulations of the transmission of digitally generated pulses over various lengths of cable. The simulation was used to determine the sampling rate, the number of quantization levels for each sample, and the number of unique pulse shapes (LBO settings) required to meet the electrical specifications for 0 m to 200 m of 22 AWG cable. Chapter 3 describes the simulation program and the procedure used to determine the unique pulse shapes. A hardware prototype of the LBO circuit was built to generate the pulse shapes identified in the simulation phase, and to test transmission of these pulses over real cables., Chapter 4 describes the LBO circuit. The results of the hardware measurements were compared to the operation of a single-chip, analog LBO, to compare the performace of the digital LBO with the best of today's technology. Next, two techniques were examined to solve the manual control problem, but only one was successful. These topics are also discussed in Chapter 4. Finally, in Chapter 5, other applications, besides the LBO, of the waveform generation technique are also examined. Chapter 6 summarizes the important results of the research presented in this thesis.

CHAPTER 2

BACKGROUND AND PROBLEM DEFINITION

This chapter explains what the Line Build-Out circuit (LBO) is and where it is positioned in the overall telecommunications network. Digital transmission in the network occurs at various signal rates. Interconnection of equipment operating at the same signal rate, and located in a telephone company central office, is achieved with a cross-connect device. The hierarchy of digital transmission rates and cross-connects are described in general terms. This thesis deals with the DS-1 signal and the DSX-1 cross connect point, which are described in detail in this chapter. The LBO is also defined, and LBOs using the foremost in today's technology are reviewed. The chapter concludes by outlining the new LBO solution that is the object of this thesis. The remainder of the thesis deals with the design of the proposed LBO.

2.1 Transmission Signal Hierarchy

Transmission of digital signals within the North American telecommunications network takes place at various bit rates. Each signal has a well defined structure that has been accepted as a standard both nationally and internationally.[1] These bit rates form a hierarchical structure whereby lower rate signals are time multiplexed into higher rate signals. The lowest rate standard transmission signal is the DS-1 signal. It is made up of 24 voice channels (DS-0 signals) which have been digitized at an 8 kHz sampling rate with 8 bits of resolution for each sample. The 24 voice channels are byte interleaved, and a single framing bit is added as overhead. This forms a total signal with 193 bits per frame (24 x 8 bits plus 1 bit overhead) transmitted serially at an 8 kHz frame rate to form a 1.544 Mbit/s signal. More detail on the exact structure of the DS-1 signal can be found in any one of many books describing the digital transmission standards.[2] The next two signals in the hierarchy are the DS-2 signal at 6.312 Mbit/s (a multiplex of 4 DS-1s plus overhead), and the DS-3 signal at 44.736 Mbit/s (a multiplex of 7 DS-2s plus overhead). A fourth signal, the DS-4, has been defined for higher rate signals, but is not yet an accepted industry standard.

A digital signal cross-connect point is defined for each of the standard transmission signals. It is denoted DSX-n, where "n" corresponds to the level in the multiplex hierarchy. Standards also define the electrical characteristics of each signal at the cross-connect. This thesis deals only with the DS-1 signal, and specifically, the DSX-1 standard specification.

A digital cross-connect is the termination point for digital transmission equipment located on the telephone company's premises. The purpose of having the DSX-1 electrical standard at the cross-connect point is to provide a common reference for all DS-1 transmission equipment manufacturers. This ensures compatibility between equipment from different manufacturers. The DSX-1 standard also provides a reference point for international boundaries. Troubleshooting and maintenance of interconnected equipment is simplified by having all manufacturers adhere to one common electrical standard. This scenario is similar to the computer industry, where components from a multitude of suppliers can all interface with each other because of the existence of a standard bus.

The actual cross-connect consists of two sets of terminating sockets for attaching cables from DS-1 equipment located on either side of the cross-connect, plus cross-connect sockets for making connections between the two sets of terminating sockets. Generally, there is also provision for bypassing failed lines, monitoring suspect lines, and inserting test signals. The DSX-1 standard gives the electrical specifications that signals must meet at the cross-connect.

2.2 DSX-1 Standard

Table 1 shows the standard specifications for the DSX-1 signal.[3] The pulse characteristics are defined for the signal measured at the cross-connect with the line terminated at the cross-connect by a 100 ohm resistor. The 100 ohm resistor is very close to the characteristic impedance, Z_0, of a 22 gauge twisted pair transmission line, and will suppress nearly all reflections. The DSX-1 standard was created and defined by the American Telephone and Telegraph Company (AT & T).[4] The AT & T standard definition does not include a specification for cable type or for cable length restrictions. However, the industry practise is to use 22 AWG, shielded, twisted pair cable, and to allow a maximum cable length of 200 m between the cross-connect and the transmission equipment. This short distance is characteristic of the distances found inside a central office of a telephone company. Transmission on a line is in one direction only, therefore a bidirectional link requires separate transmit and receive cables.

Bit Rate:	1.544 Mbit/s ± 130 ppm
Line Code:	Bipolar AMI (B8ZS for DS-0 transparency)
Line Impedance:	100 ohms ± 5 % resistive, balanced

Pulse Characteristics for +0000000-0000000+...sequence into a 100 ohm resistive termination:

Amplitude:	2.4 to 3.6 V
Width at Half Pulse Amplitude:	294 to 404 ns
Ratio of Power Between Positive and Negative Amplitude Pulses:	<0.5 dB
Ones Density (Minimum Average):	1 in 8 with no more than 15 consecutive zeros (does not apply with B8ZS line coding)

The voltage within a time slot containing a ZERO shall be no greater than either the value produced in that time slot by other pulses in adjacent time slots within the mask of Figure 1 or +/- 0.1 of the peak pulse amplitude, whichever is greater in magnitude.

Power with a +-+...sequence measured in a 2 kHz bandwidth into a 100 ohm resistive termination:

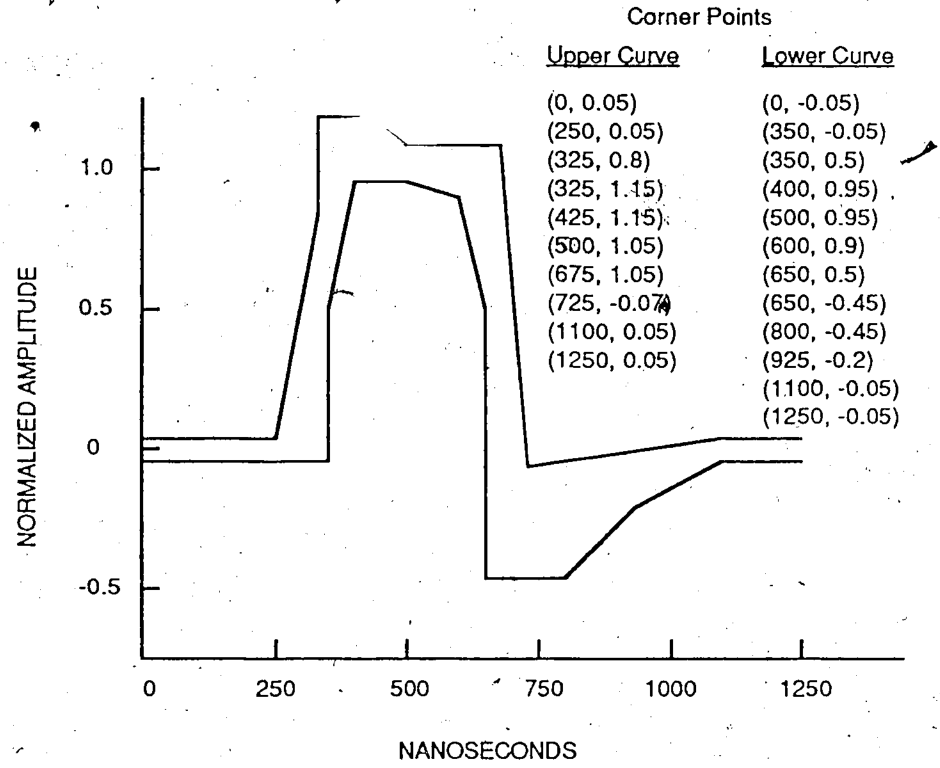
772 kHz:	+12.6 to +17.9 dBm
1.544 MHz:	At least 29 dB below that at 772 kHz

Cross-Connect Cabling and Patching:

DSX-1 Cable Type:	22 AWG shielded twisted pair cable
Maximum Cabling Distance:	200 m

Table 1: DSX-1 Standard Specifications

Perhaps the single most important specification defined by the DSX-1 standard is the pulse shape specification shown in Figure 1.[5] A valid DSX-1 pulse is one that, with the amplitude scaled appropriately, falls between the two curves shown in Figure 1. If this specification and the pulse amplitude specification are met, then the pulse width and power specifications are automatically met. It is the function of the LBO circuit to ensure that signals meet the shape and amplitude specifications at the cross-connect.



Corner Points

Upper Curve	Lower Curve
(0, 0.05)	(0, -0.05)
(250, 0.05)	(350, -0.05)
(325, 0.8)	(350, 0.5)
(325, 1.15)	(400, 0.95)
(425, 1.15)	(500, 0.95)
(500, 1.05)	(600, 0.9)
(675, 1.05)	(650, 0.5)
(725, -0.07)	(650, -0.45)
(1100, 0.05)	(800, -0.45)
(1250, 0.05)	(925, -0.2)
	(1100, -0.05)
	(1250, -0.05)

Figure 1: DSX-1 Pulse Shape Specification

2.3 LBO Definition

The purpose of an LBO is to ensure that signals transmitted to the cross-connect will meet the DSX-1 standard after travelling over some length of cable. Since the physical length of cable between equipment and the DSX-1 panel may vary from 0 m to 200 m, the distortion of the transmitted pulse caused by the cable must be compensated accordingly, so that, regardless of physical distance, the pulse shape at the cross-connect always meets the DSX-1 standard. Consequently, to obtain the standard pulse shape at the cross-connect, the DS-1 line card must be able to predistort the pulse in a way dependent on the length of the terminating cable.

In summary, the LBO predistorts a bipolar DS-1 signal to compensate for a particular length of cable between the DS-1 signal source and the DSX-1 cross-connect.

2.4 Today's Solution

Currently, analog pulse shaping techniques using linear equalizer networks are used to pre-distort the DS-1 signal. The pulse shape specification places contradictory requirements on the equalizer network.. On the one hand, a fast rising edge with little overshoot is desired (see Figure 1). On the other hand, a large undershoot that decays to 0 V slowly is desirable on the pulse's trailing edge. These requirements cannot easily be met simultaneously by a linear circuit. Hence, one equalizer setting is insufficient to maintain conformity to the DSX-1 pulse shape standard over the entire 0 m to 200 m range of line lengths. A compromise solution is to have several equalizer settings, with each setting valid for a limited range of line lengths.

The most advanced LBO technology has produced a single-chip transmit LBO and receiver circuit. Table 2 shows the various LBO integrated circuits (ICs) available today. The devices use various fabrication technologies to implement the analog equalizer networks. Manual switches are used to select the equalizer settings. Each setting produces a pulse shape that will meet the DSX-1 standard at the cross-connect when transmitted over a limited range of line lengths, as shown in the rightmost column of Table 2.

The major disadvantage of this solution is the manual control required to set the LBO for each range of line lengths. This means that each time equipment configurations are changed, the appropriate LBO setting must be chosen for the length of line between the signal source and the cross-connect. The extra record keeping and manual intervention that this requires increase the operating cost to a telephone company.

DEVICE	TECHNOLOGY	TRANSMIT EQUALIZER SETTINGS
Silicon Systems, Inc. SSI 78P-233 DS-1 Line Interface	Analog Bipolar 24 Pin Dual In-Line Package	Setting: 1: 0 to 40 m 2: 40 to 80 m 3: 80 to 120 m 4: 120 to 160 m 5: 160 to 200 m
Rockwell R8069 Line Interface Unit	Analog CMOS 28 Pin Dual In-Line Package	Setting: 1: 0 to 33.5 m 2: 33.5 to 67 m 3: 67 to 100.5 m 4: 100.5 to 134 m 5: 134 to 167.5 m 6: 167.5 to 201 m
Mitel Semiconductor MH89750 DS-1 / T1 Digital Trunk Interface	Thick Film Hybrid 40 Pin Dual In-Line Hybrid Module	Setting: 1: 0 to 45.7 m 2: 45.7 to 137 m 3: 137 to 228.5 m

Table 2: Current LBO Technology Summary

2.5 Proposed Solution and Thesis Objectives

This thesis describes a digital method for creating pre-distorted DS-1 pulse shapes. Digitized voltage samples which define a distorted DS-1 pulse shape are stored in an EPROM. The samples are read out of the EPROM at a fixed rate several times the DS-1 data rate, and applied to a digital to analog converter (D/A). The output of the D/A is an analog, bipolar, DS-1 data stream of pre-distorted pulses. The EPROM stores positive and negative pulse shapes, plus a value corresponding to a zero pulse. A binary, NRZ DS-1 data stream and a 1.544 MHz clock signal are encoded to form the address inputs to the EPROM, and thus cause the correct pulse (either a +1, a -1, or a O) to appear at the D/A output.

The prime motivation for using this digital technique was to eliminate the manual LBO settings. It was felt that digital techniques could produce a superior pulse shape, and digital circuits could possibly be set using computer control, rather than manual switches. The EPROM and D/A approach realizes a non-linear LBO circuit which, unlike a linear circuit, can predistort the transmitted pulse so that it will meet the contradictory specifications described in the previous section, over a much broader range of line lengths. Fast rise times and long trailing edge undershoot decay can be achieved simultaneously by chosing appropriate sample values. The ability to easily generate complex, non-linear pulse shapes, and to alter the pulse shapes easily through editing the EPROM contents, were the primary reasons that this technique was chosen.

Two major objectives of the project were decided before the research began. First, the feasibility of the EPROM and D/A approach to LBO design was to be evaluated. If the outcome of this evaluation was positive, then an examination of techniques to eliminate the manual switch settings would take place. The rest of this thesis deals with how these two objectives were addressed, beginning with the design of the LBO, followed by the elimination of manual LBO control. Once these two major objectives were addressed, other applications of the pulse shaping method were examined. The design of the digital LBO began with a software simulation, which is described in the following chapter.

CHAPTER 3

SIMULATION PHASE

A simulation program was used to determine the minimum number of different pulse shapes needed to satisfy the DSX-1 standard over cable lengths ranging from 0 m to 200 m, and to determine the minimum required time and amplitude quantization resolution that would be necessary. The time resolution refers to the number of samples, or segments, that the DS-1 symbol interval is divided into. The amplitude resolution refers to the number of bits used to describe each segment. The computer simulation was used to test the feasibility of the proposed pulse shaping technique before a hardware prototype was built. During the simulation phase, only the DSX-1 pulse shape specification was used to test the validity of a particular pulse shape. The amplitude specification was addressed at the hardware stage. The simulation was written in Turbo Pascal on an IBM-AT compatible personal computer. This software/hardware combination was chosen due to the availability of various mathematical and graphics software packages for the IBM personal computer. The processing power of the AT sufficed for this application. To obtain frequency domain representations of the DS-1 pulses, fast Fourier transforms (FFTs) were used. The FFT method was chosen because of its speed and the availability of packaged routines to perform the transformation.

This chapter consists of two major sections. The first describes the computer simulation program in detail. The simulation program was used as a tool in an evaluation process to determine which pulse shapes the LBO should generate. The second part of this chapter describes the iterative process used to evaluate various pulse shapes.

3.1 Simulation Program

Before coding began, requirements on the program's functionality were established. Generally, the program was to simulate the transmission of predistorted DS-1 pulses from a DS-1 signal source to a DSX-1 cross-connect over a length of cable. A desirable feature was to graphically display the pulse shape and the pulse mask at the cross-connect. In addition, time base shifting and amplitude scaling of the pulse should be possible to test for fit inside the DSX-1 pulse mask. The pulse shape to be tested should be stored in a computer disk file to allow shape modifications, and switching between several different pulse shapes to be done easily. It was also desired that the line length be easily modified, for example by keyboard entry at run time. The pulse mask should also be stored in a disk file in case compliance with a mask other than the DSX-1 mask was to be tested. Parameters for the line model were also to reside in a disk file to allow testing of lines with different gauges, and hence, different transfer functions. Once these details of the high-level functionality of the simulation were established, a computer program was written.

The simulation program was written entirely in Turbo Pascal, version 3.0. The hardware configuration consisted of an IBM-AT compatible computer, with 640 Kbytes of RAM, an Intel 80287 math-co-processor, and an EGA monitor. Two commercial software packages were used in conjunction with Turbo Pascal: the Turbo Numerical Toolbox (for FFT routines), and the Turbo Halo graphics package.[6,7] Routines from both packages must be included with the simulation program during compilation. These routines must reside in the same directory as the simulation program, along with other files. The program listing, and the contents of all the user-created files that must reside in the same directory as the simulation program are found in Appendix C. The program is compiled to a file with a ".com" suffix so that it can be run directly from the operating system, without invoking Turbo Pascal.

Figure 2 shows a block diagram of the major functional modules of the program. The interconnecting arrows indicate the flow of program control. The program begins by initializing several variables, and by reading in the necessary simulation parameters. Next, the pre-distorted DS-1 signal is transformed to the frequency domain, filtered, passed through a transmission line, and then transformed back to the time domain. The resulting pulse is then displayed graphically on the screen. The simulation program can be divided into three major functional areas: data entry, the actual simulation of transmission, and a graphics display mode. The remainder of this section describes each area in detail.

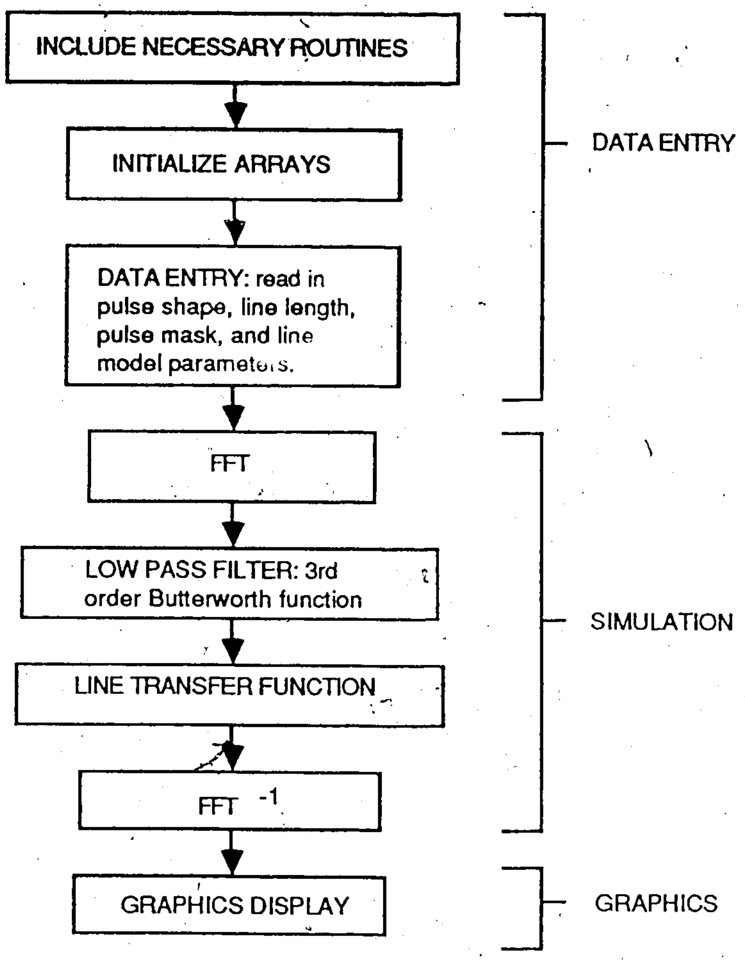

INCLUDE NECESSARY ROUTINES
INITIALIZE ARRAYS
DATA ENTRY: read in pulse shape, line length, pulse mask, and line model parameters.
DATA ENTRY
FFT
LOW PASS FILTER: 3rd order Butterworth function
LINE TRANSFER FUNCTION
SIMULATION
FFT -1
GRAPHICS DISPLAY
GRAPHICS

Figure 2: Simulation Software Block Diagram

3.1.1 Data Entry

Pascal array variables used by the FFT routines are defined as pointers to arrays of real numbers. Arrays must be passed between procedures, and the use of pointers reduces the memory requirements. All data read in by the program are stored in array variables. All arrays are of the same type, and are initialized using the Pascal "NEW" command at the start of the program.

Following initialization, program variables are entered in the following order: the stimulus vector defining the pre-distorted pulse shape, the pulse mask, the line length, and the Z_0 and γ values for the line model. The program queries the user for the file names and line length, which are entered via the keyboard. The

The pulse mask file name is not entered by keyboard; the program reads the pulse mask from a file called "pmask.dat". The internal structure of each file must follow a certain format, otherwise I/O errors may occur, or the program will fail later due to incorrect data entry. File I/O errors are handled by monitoring a return code after each file operation. If the return code indicates that an error has occurred, the program prints out an appropriate error message, and the user can re-enter the file name. The error messages will only be correct for Turbo Pascal version 3.0.

The DS-1 pulse stimulus vector describes a bipolar waveform that is 8 DS-1 symbols long, with the pattern +1, 3 zeros, -1, 3 zeros. This pattern produces a balanced signal with 0 V dc offset. The program uses fast Fourier transforms (FFTs) to get a frequency domain representation of the input waveform. Therefore, the stimulus vector must be one cycle of a repetitive waveform, and be described by a finite number of discrete samples. Sufficient time base resolution is provided by 32 samples per DS-1 symbol interval, and the stimulus waveform is described by a file that contains 256 samples, arranged one sample per line. The samples are Pascal type "real" numbers, and represent voltage values. When using FFTs, the value of the first and last sample of the stimulus waveform must be equal, otherwise problems with the frequency domain representation will occur due to the discontinuities at the boundary of each cycle of the repetitive waveform. The calculation of sample values and file creation must be done manually before the program is invoked.

The stimulus pattern used in the simulation is different from the DSX-1 test stimulus in which bipolar "1" pulses are separated by 7 zeros. This was done to reduce the length of the stimulus vector, and hence the memory requirements and run time of the simulation program. The FFT routines can work with up to 4096 points, but the program execution would be very slow. The length of the simulation was to be minimized due to the large number of simulations that were expected to be run. In reality, it would be more difficult to meet the DSX-1 specification with the shorter stimulus, because the "1"s are closer together, and interference between consecutive "1"s could be a larger problem. If the specification could be met for the short stimulus vector, then it could also be met for the true DSX-1 test pattern.

The vertices of the pulse mask are read in next. These must be stored in a file called "pmask.dat" which is located in the same directory as the simulation program. The vertices are stored one to a line, as an x and y coordinate separated by a space. Values for the x and y coordinates are shown in Figure 1. The pulse mask is treated as a closed polygon, and the points are stored in order starting with the leftmost coordinate of the upper curve, and going clockwise around the entire mask, ending with the leftmost coordinate of the lower curve. With the input file structured in this way, the pulse mask can be drawn on the computer screen using a single Halo graphics command statement.

Following the pulse mask entry, the line length is read from the keyboard. The value can be any integer value between 0 and 400 (the line length is in metres). The reason for this wide range of allowable line lengths when the maximum cable length to the cross-connect is 200 m lies in the structure of the cable model. In the simulation, the cross-connect is modeled as being located at the midpoint of a line that is terminated at one end by a resistor. Therefore, the distance between the cross-connect and the signal source is actually one half the line length entered. Figure 3 shows the simulated structure. This representation was used initially due to the lack of a proper description of the DSX-1 standard. However, since 22 gauge cable exhibits a characteristic impedance very close to 100+j0 ohms for frequencies over approximately 100 kHz, the location of the termination has very little effect on the shape of the pulse at the cross-connect. Measurements made at the hardware stage over real 22 gauge cable confirmed these results - the pulse looks the same whether the line is terminated at the cross-connect, or at any point past it.

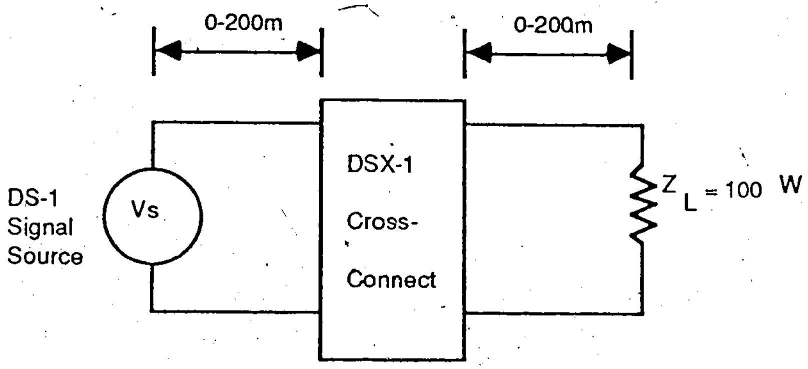


Figure 3: Simulated Structure

Finally, data describing the line model are read in. The name of the file containing the data is entered by the user. The file format consists of four values per line, separated by spaces. The first two numbers on each line are the real and imaginary parts of Z_0. The third and fourth numbers are the real and imaginary parts of γ. Significant preparation is required to generate the contents of this file. The FFT routine generates frequency domain components of a signal at discrete frequencies. The frequency spacing of these components is determined by the period of the repetitive time waveform being analyzed. In this case, the period $T = 8 \div 1.544 \times 10^6$ ns, and the frequency spacing is $1/T = 193$ kHz. Since Z_0 and γ are functions of frequency, they must be evaluated at 193 kHz intervals. Z_0 and γ values for $f = 0$ Hz are excluded, since the input signal is balanced, and has a DC component equal to 0 V. Furthermore, measured RLGC parameters for twisted pair cable were only available for frequencies up to 5 MHz.[8] To obtain a description of the line for frequencies above 5 MHz, the RLGC parameters were extrapolated to 10 MHz. In the frequency domain, all signal components above 10 MHz are set to zero. Appendix A describes the cable model in detail and explains the interpolation and extrapolation methods used. Since truncation over 10 MHz occurs, only 51 values of Z_0 and γ are needed - the fifty-second point corresponds to a frequency just over 10 MHz. The program only reads in the first 51 points in the cable model file. Once the cable parameters have been read in, the program has all the information it needs to perform the simulation.

3.1.2 Simulation Steps

The first step after data entry is to convert the time domain input waveform to the frequency domain by calling the FFT procedure. Samples in the input file form the real part of a vector of complex points, with the imaginary part set to zero. Real parts are kept in one array, and the imaginary parts in a second array. The result of the FFT is a vector of 256 complex numbers (ie. with non-zero imaginary parts), which contains the frequency domain components at 193 kHz intervals. Components for negative frequencies are returned as well. The relationship between array subscripts, "X", and FFT coefficient indices, "N", is as follows: for $0<X<127$, $X = N$; for $128<X<255$, $-128<N<-1$. The frequency of the Nth FFT coefficient is given by N times 193 KHz. The Turbo FFT routine uses this convention because, in general, much of the high frequency information can be ignored with minimal effects on the time domain representation of the signal.

Initially, the frequency domain signal was passed directly through the line transfer function. All Fourier coefficients of the resulting signal above 10 MHz were set to zero. The resulting time domain signal contained high frequency ringing superimposed on the DSX-1 pulses. This effect is known as Gibbs' phenomenon, and is caused by the frequency domain truncation.[9] Gibbs' phenomenon was pronounced because the frequency components of the signal over 10 MHz are quite strong. To decrease Gibbs' phenomenon, either the signal or the line transfer function must roll-off smoothly to a value near zero at 10 MHz. Then, subsequent truncation will not result in any significant ringing in the time domain. Since the transfer function of the line is a function of line length, changing its roll-off would require a different approach for each length of line. It is easier to add a low-pass filter (LPF) transfer function before the line transfer function, to decrease the high frequency components of the signal. After experimenting with several filter transfer functions, a third order Butterworth LPF with a 3 dB passband cut-off of $f_c = 5$ MHz was chosen. The transfer function for this filter is shown below.

$$H(f) = \frac{1}{\left(1 - 2\left(\frac{f}{f_c}\right)^2\right) + j\,\frac{f}{f_c}\left(2 - \left(\frac{f}{f_c}\right)^2\right)}$$

Addition of this filter transfer function to the program eliminated the ringing in the time domain.

The program multiplies the Fourier coefficients corresponding to the first 51 positive frequencies by the filter transfer function evaluated at each frequency. The 51 negative frequency components are multiplied by the complex conjugate of the filter transfer function, evaluated at the positive frequency of the same magnitude.

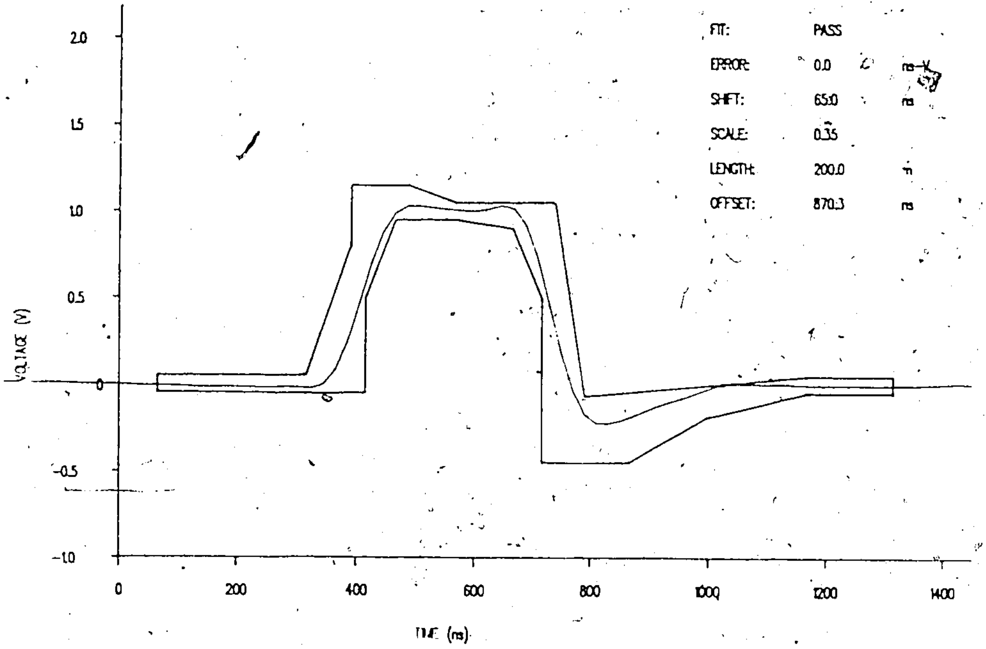
The next task is to multiply the frequency domain representation of the filtered signal by the transfer function of the line model. The transfer function for the line is derived in Appendix A. The transfer function is evaluated for each of the first 51 positive frequencies, and multiplied by the appropriate frequency component of the test signal. The test signal components are multiplied by the complex conjugate of the positive frequency transfer function for the first 51 negative discrete frequencies. All calculations are made using complex number multiplication, division, and hyperbolic sine and cosine routines. Also, all FFT coefficients with |N| > 51 are set to zero. The 0 Hz, or DC, component of the test signal is zero, and is not altered by the filter or line transfer functions.

The resulting vector of complex numbers is transformed back to the time domain by a single procedure call. The graphics mode is entered next.

3.1.3 Graphics Mode

Once the mathematical manipulations of the test signal are complete, the resulting time domain signal is displayed graphically on the computer screen. After transmission over a length of line, the DS-1 pulse experiences a time delay. Thus the first function performed is to find the peak of the positive pulse, and shift this point to the centre of the display window. Next, the Halo graphics routines are initialized, and a short automatic analysis phase is entered.

The automatic analysis checks for fit of the positive pulse of the test signal inside the DSX-1 mask. The only allowed operation on the pulse is amplitude scaling. The pulse is actually held stationary, in the centre of the screen, and the pulse mask is shifted to test for compliance with the standard. The pulse mask can be moved both vertically and horizontally on the screen. In the automatic analysis, the movement of the pulse mask is restricted to the horizontal direction only. The program loops through 3 settings of time shift for the mask. For each setting of time shift, it loops through 3 settings of the scale factor. At each combination of time shift and scale factor, an error calculation is made. The mean square error is calculated by taking the sum of the product of the time between sample instants (1÷(32 x 1.544 MHz) = 20.2 ns) times the absolute value of the amount that the pulse exceeds the mask by, over the duration of the pulse mask. The sample values used in the error calculation are from the 256 samples defining the time domain signal after transmission over the cable. When the pulse fits entirely inside the mask, the error will be equal to zero. For each setting of time shift and scaling facto-, the pulse and mask are displayed along with some simulation parameters. Figure 4 shows a sample of the computer display. The "fit" parameter is either pass("error" = 0) or fail ("error" = 0). The "shift" is the time shift of the pulse mask, "scale" is the scale factor, and "length" is the total line length (up to 400m). The "offset" variable shows the amount that the positive pulse was shifted in time to bring it into the display window. The analysis phase performs a search for the minimum error setting, and redisplays this setting when the analysis is complete.



FIT: PASS
ERROR: 0.0 ns
SHIFT: 65.0 ns
SCALE: 0.35
LENGTH: 200.0 m
OFFSET: 870.3 ns

Figure 4: Example of Simulation Output

This automatic analysis only coarsely scales the pulse and shifts the pulse mask, and therefore can be eliminated. To be helpful, a 1 ns time shift resolution, and an increment of .01 on the scale factor would be necessary over a wide range of values. This would enable the program to automatically determine accurately if the pulse fits inside the mask or not. However, testing each combination of time shift and scaling takes approximately 1 second. At fine resolution, and over a wide range of values, the automatic process would take several hundred seconds. The mask may also be shifted vertically, which adds a third variable to the automatic analysis, and would further increase the overall time to complete the process. It is faster to shift and scale manually, and eliminate the automatic analysis completely.

In the manual control mode, 3 options are displayed at the bottom of the screen:

1	Move Mask and Scale Pulses
2	Plot Screen
3	Quit

Options are entered by the user through the keyboard. When a "1" is entered, the program prompts the user for a scale increment and a time increment. The scale increment is also the increment in volts used to shift the mask in the vertical direction. After entry, left and right arrow keys are used to move the pulse-mask by the time increment in either direction, and the up and down arrow keys are used to scale the pulse, increasing or decreasing the scale factor by the scale increment each time they are pressed. The "page up" and "page down" keys are used to move the mask up or down one increment, respectively. Pressing the escape key returns to the main menu. From the main menu, entering a "2" causes the screen to be redrawn, and a plot file is generated in the same directory as the program with the name "list.out". If a "3" is entered from the main menu, the graphics routines are shut down, and the program ends. One option not shown on the screen is to press "5", which draws a compressed display of the entire test signal on the screen. This option was added a late stage in the simulation process to aid in the development of the different pulse shapes. To plot, the user must exit the program, and invoke the Halo plot command to dump the file "list.out" to a plotter. The plotter must be a Hewlett-Packard model 7475 plotter, or any device that supports the HP-GL language.

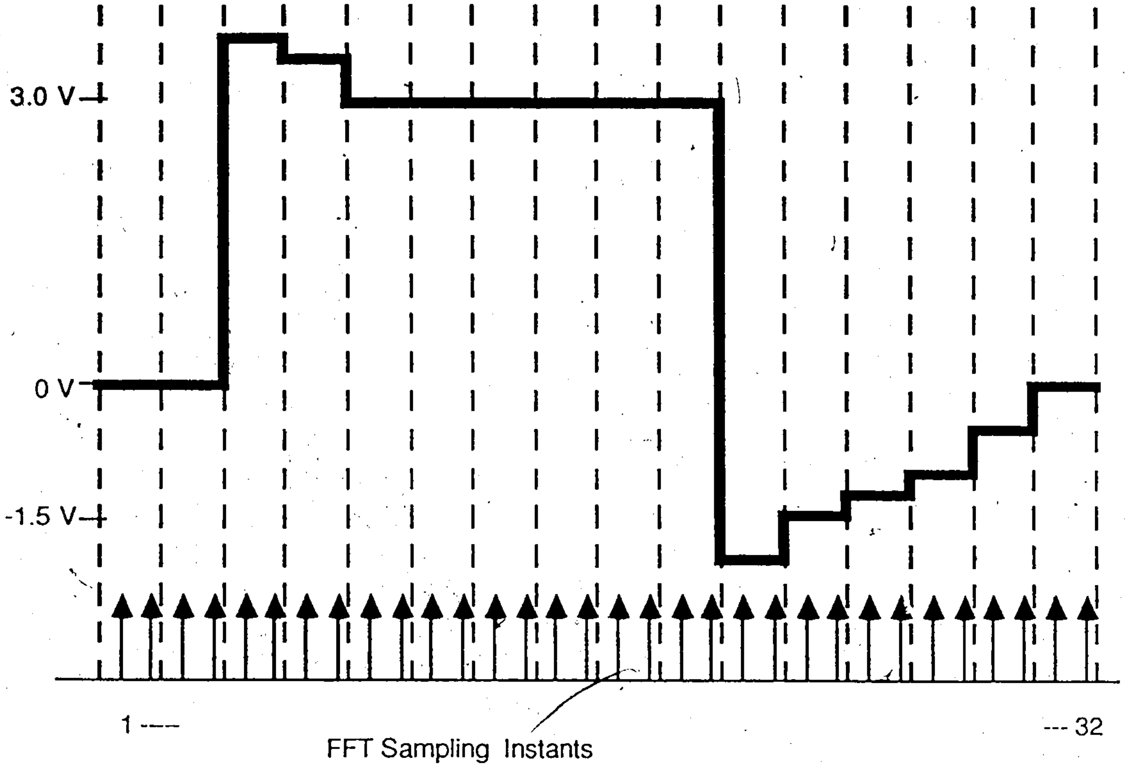
This concludes the description of the Pascal simulation program. The simulation was used as a tool in an iterative procedure to define pre-distorted pulse shapes. This procedure is described in the next section.

3.2 Iterative Pulse Shape Design Procedure

The computer simulation program was used as a tool in an iterative procedure which determined the minimum number of pulse shapes needed to satisfy the DSX-1 standard over the 0 m to 200 m range of line lengths. Only the pulse shape specification was used to determine if a given pulse met the DSX-1 standard. Since the pulse amplitude may be scaled to fit the normalized DSX-1 pulse mask, the amplitude specification can be addressed separately at the hardware stage by transmit buffer drive level adjustments. A minimal number of individual pulse shapes is desirable to minimize the number of LBO settings. It was felt that fewer settings would simplify the receiver design. In addition, the iterative process was to determine the number of time and amplitude quantization levels necessary to generate distorted pulse shapes. These parameters were also to be minimized to simplify the hardware design. Few time quantizations are desirable because the hardware could be clocked at a lower speed, thus allowing the use of inexpensive, low-speed TTL or CMOS technology. Less importantly, fewer bits per sample are desired (few amplitude quantizations) because less memory for sample storage, and a smaller, and therefore simpler, D/A would be required. This section describes the process used to evaluate various combinations of time and amplitude quantization resolution, with the goal of finding the minimum number of unique pulse shapes that the LBO would generate.

Pulse shapes tested were stored in a computer file, as described in the previous section. This file contains 256 voltage samples which represent 8 DS-1 symbol intervals with a resolution of 32 samples per DS-1 symbol. To simulate the output of a D/A converter, a working voltage range of -3.6 V to +3.6 V was chosen based on the maximum amplitude allowed by the DSX-1 specification. The voltage step, ΔV, of a linear D/A converter is obtained by dividing the 7.2 V range by 2^n-1, where "n" is the number of bits of resolution for each sample. All valid samples were calculated as integer multiples of ΔV. Up to $2^n/2$ levels are allowed below zero, and $2^n/2$ - 1 levels above zero. The simulation only checks compliance with the DSX-1 pulse shape specification. The exact amplitude of the simulated pulses is not important, because the amplitude specification can be addressed separately; only the values of the samples relative to other samples need to be correct.

At this point some clarification is necessary. When considering the structure of the stimulus signal file, there are two sampling rates that must be kept separate in the reader's mind to avoid confusion. One is the sampling rate used to generate the pre-distorted DS-1 pulses, which refers to the number of segments which each DS-1 symbol interval is divided into. This is the rate that samples are read from an EPROM and passed to a D/A converter. The other sampling rate is the one used to create discrete samples for proper input to the FFT routines. The latter sampling rate refers to the number of samples per DS-1 symbol interval taken at the output of the D/A. For example, consider that the DS-1 pulse may be generated by, say, 16 segments per DS-1 symbol. This pulse appears as an analog signal at the output of the D/A. To take the FFT of this analog signal, it must be decomposed into discrete samples, taken at a rate of 32 samples per DS-1 symbol interval. Figure 5(a) shows a DS-1 pulse generated by 16 segments per DS-1 symbol, as it appears at the D/A output. The arrows along the horizontal axis show the 32 sampling instants used to generate appropriate input for the FFT routines. Figure 5(b) shows the values of the resulting samples.



(a) D/A Output

Sample #	Value	Sample #	Value
1	0	17	3.0
2	0	18	3.0
3	0	19	3.0
4	0	20	3.0
5	3.6	21	-1.8
6	3.6	22	-1.8
7	3.5	23	-1.5
8	3.5	24	-1.5
9	3.0	25	-1.2
10	3.0	26	-1.2
11	3.0	27	-0.9
12	3.0	28	-0.9
13	3.0	29	-0.5
14	3.0	30	-0.5
15	3.0	31	0
16	3.0	32	0

(b) FFT Sample Values

Figure 5: Example of Stimulus Vector Structure

It was desirable to keep the FFT sampling rate at 32 samples per DS-1 symbol, but to somehow simulate pulses generated by fewer than 32 segments per DS-1 symbol interval. Figure 5 shows that the FFT samples for a pulse generated by 16 segments per DS-1 symbol are grouped in pairs of samples of the same value. Similarly, to simulate a DS-1 pulse generated by 8 segments per DS-1 symbol, the FFT samples can be entered into the file in groups of 4; for a pulse generated by 4 segments per DS-1 symbol, the FFT samples can be considered as groups of 8, and so on. Thus to simulate the use of different numbers of segments per DS-1 symbol interval, the samples in the file are simply treated as groups of 2, 4, or 8 samples.

The computer simulation was used to iteratively explore the various discrete design spaces created by each combination of time and amplitude quantization. To start the process, two pulse shapes where found which fit the mask at the two extreme lengths of physical cable: 0 m and 200 m. These initial shapes were then incrementally modified using an iterative procedure to maximize the range of line lengths for which each shape was valid. If the ranges did not overlap, a third pulse shape could be developed. The procedure used to determine the range of line lengths for each pulse shape was as follows, using pulse shape 1 (valid for shorter lines) as an example.

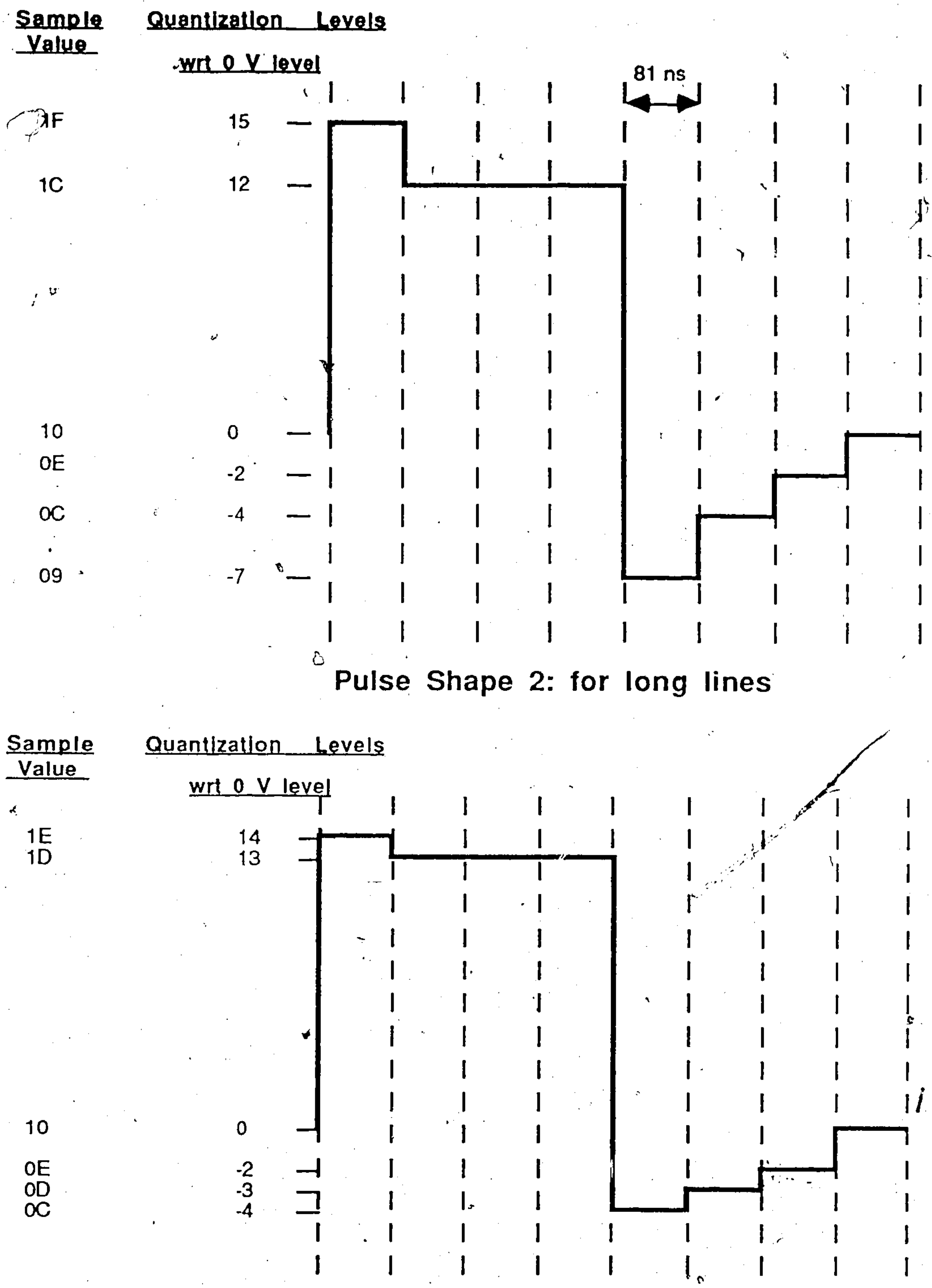
First, the current number of segments per DS-1 bit, and the number of quantization levels were chosen. Next, the D/A voltage step, ΔV, was calculated for the fixed working voltage range of the model D/A. The samples describing the pulse were calculated as integer multiples of ΔV, subject to the number of quantizing levels being considered at that time. Pulse shape 1 was created and the appropriate sample values were entered into a file. The simulation was run at 0 m to check if the pulse did indeed fit the mask. If it did not fit, the points where the pulse violated the mask were noted, and the necessary samples in the source file were changed to remove this violation. The initial fit at the starting extreme of a range was usually achieved in one or two iterations.

Upon finding a shape which fit the DSX-1 pulse mask at 0 m, simulations were run at gradually increasing line lengths until an exact fit of the pulse within the mask was not possible, at length L1, say. Again the location of the mask violations were noted; usually the trailing edge undershoot was too small, or the rising edge of the pulse clipped the upper left inside mask edge. The source file was intuitively altered, and the simulation run at length L1. This was repeated until the pulse fit the mask at length L1. Then, simulations were run again at lengths from 0 m to L2 > L1 m. If modification of the pulse shape caused problems at lengths where the pulse fit the mask before, an assessment was made as to whether to terminate the process or continue. In most cases, L could be extended several times with slight modifications to the source file, and without affecting the fit at shorter lengths. When modifications to the pulse shape to make it fit at some length Ln caused problems at other lengths, the process was terminated. This procedure was repeated with pulse shape 2 working back from a 200 m starting length. As anticipated, the ranges for the two pulse shapes overlapped, and no more than two pulse shapes were necessary.

Although this iterative manual approach sufficed here, it is of interest to consider the prospect of totally automating the search of each discrete design space. This approach was considered, and results showed that iterative human design using the simulation was much faster than the exhaustive space search approach. For example, consider the simple case of 8 time segments and 8 quantization levels. There are 8^8= 16,777,216 possible pulse shapes. With a simulation and automated mask fitting test execution time of 5 seconds per tested pulse shape, it would take about 23,302 hours to test each possibility. With the human approach, several combinations of the number of segments and number of levels were examined in about 10 working days, or about 70 hours.

All combinations of 3 sampling rates (4, 8 and 16 segments per DS-1 symbol) and 2 amplitude quantization levels (16 and 32 levels) were examined. The combination of 8 segments per DS-1 symbol and 32 quantization levels was the coarsest resolution that produced satisfactory results. Below these values, the DS-1 pulse shape could not be reconstructed with acceptable accuracy. Two unique pulse shapes were obtained. Pulse shape 1 fit the mask for the range of line lengths between the cross-connect and DS-1 signal source of from 0 m to 150 m. The range for pulse shape 2 was 75 m to 200 m. Figure 6 shows the two pulse shapes resulting from the simulation phase. Figures 7, 8, 9, and 10 show the results of the computer simulation at the endpoints of each range. The overlap of the ranges was expected to be a significant factor in eliminating manual control of the LBO settings.

The simulation phase of the project was considered complete when the two pulse shapes were found. The next step was to build a hardware prototype which would generate these pulse shapes and transmit them over a real 22 AWG twisted pair cable.



Pulse Shape 2: for long lines

Pulse Shape 1: for short lines

Figure 6: Simulated Pulse Shapes

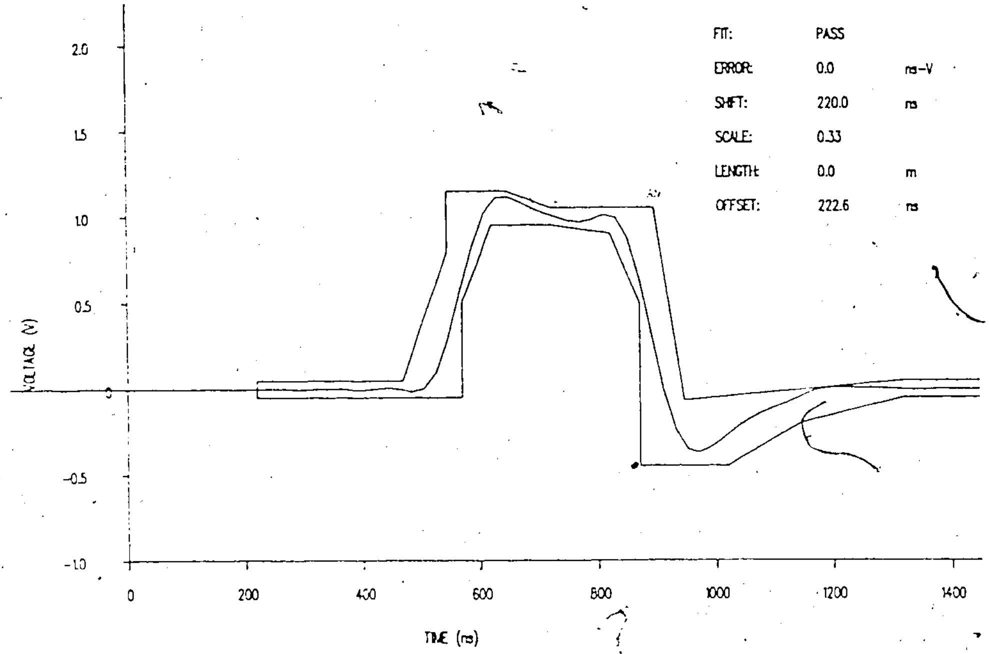


Figure 7: Simulated Pulse Shape 1, 0 m

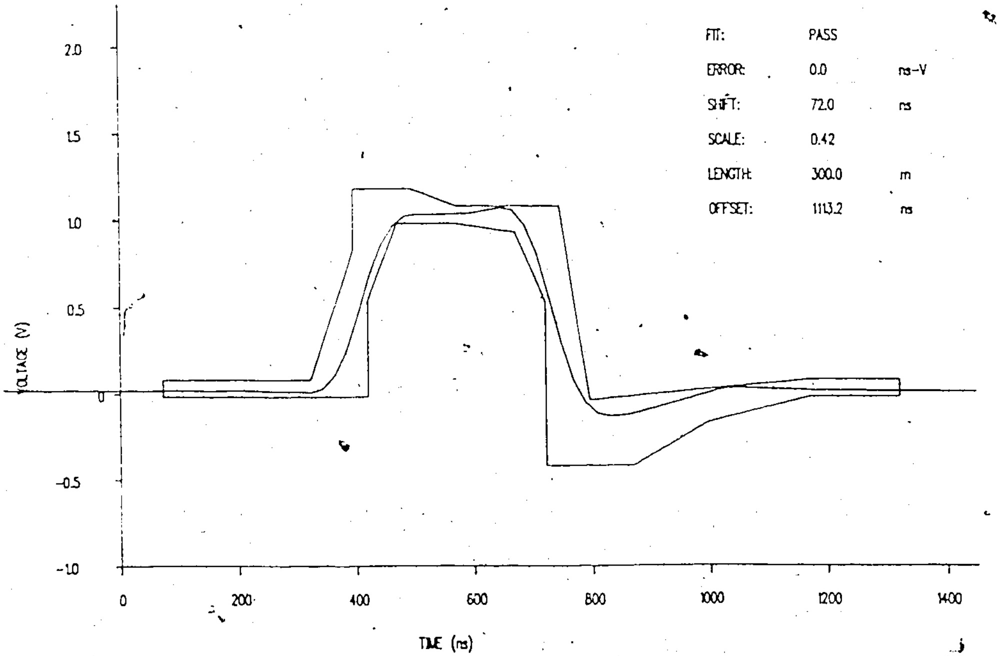


Figure 8: Simulated Pulse Shape 1, 150 m

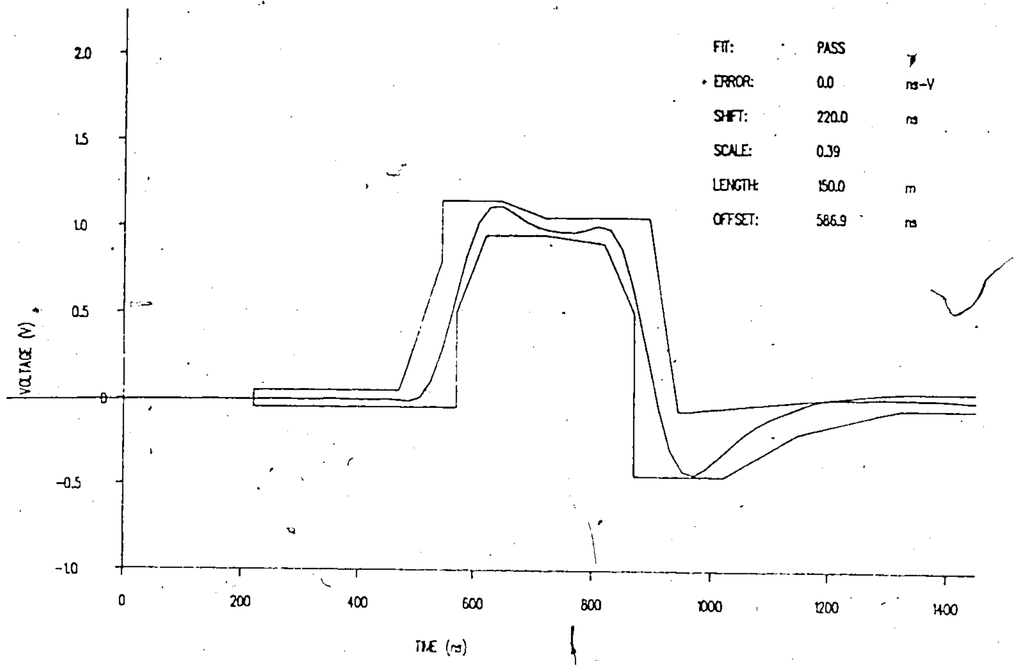


Figure 9: Simulated Pulse Shape 2, 75 m

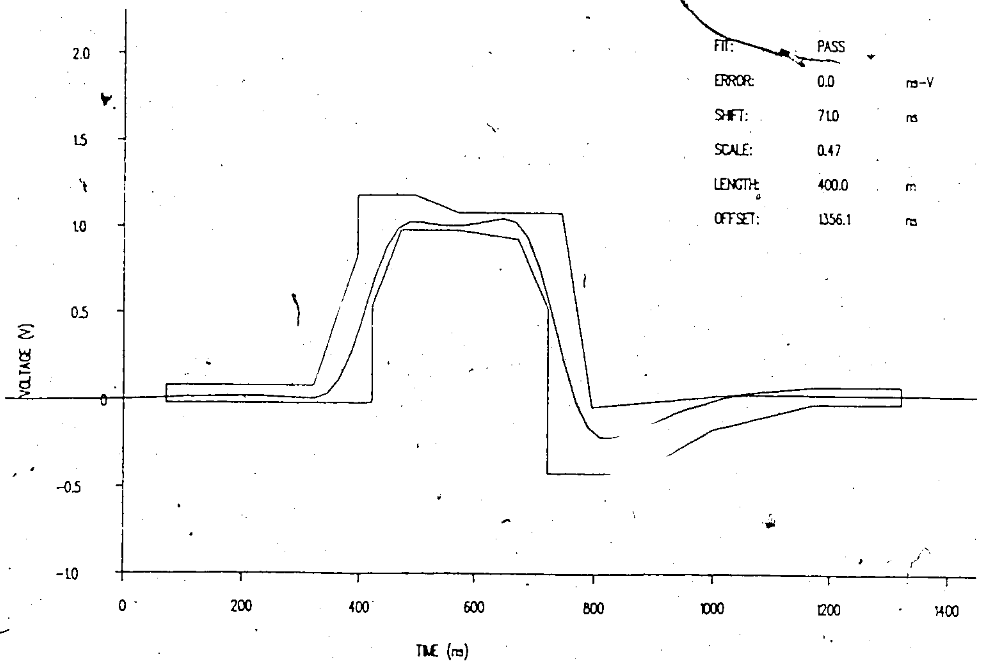


Figure 10: Simulated Pulse Shape 2, 200 m

CHAPTER 4

HARDWARE DESIGN AND TESTING

This chapter describes a hardware prototype of the proposed LBO circuit. The results of the simulation phase were used as the starting point for the hardware design. The circuit was designed to generate the two different pulse shapes derived in the simulation phase. The pulses are formed using 8 segments per DS-1 symbol interval, and each segment is described by a 5 bit wide sample stored in an EPROM. This chapter describes the operation of the LBO circuit in detail. The 22 AWG cable test facility on which transmission tests were carried out is also described, and the results of measurements of transmitted pulse shapes are presented. Next, an unsuccessful technique for automating the pulse shape selection is described, which fails due to a fundamental characteristic of the DSX-1 cross-connect. The final section of the chapter describes how the manual switches were eliminated using a single pulse shape solution.

4.1 LBO Circuit Description

The primary function of the LBO circuit is to generate the pre-distorted DS-1 pulse shapes defined during the simulation phase. The samples are stored in an EPROM, and read out and passed to a D/A converter. The D/A output is the bipolar, analog DS-1 signal, which is coupled to a transmission line via a pulse transformer. The simulation phase provided values for the two key hardware parameters: sampling rate, and the size of each sample. The samples are each 5 bits wide (for 32 quantizing levels), and are passed to the D/A at a rate of 12.352 Msamples/sec., which is 8 times the DS-1 signal rate.

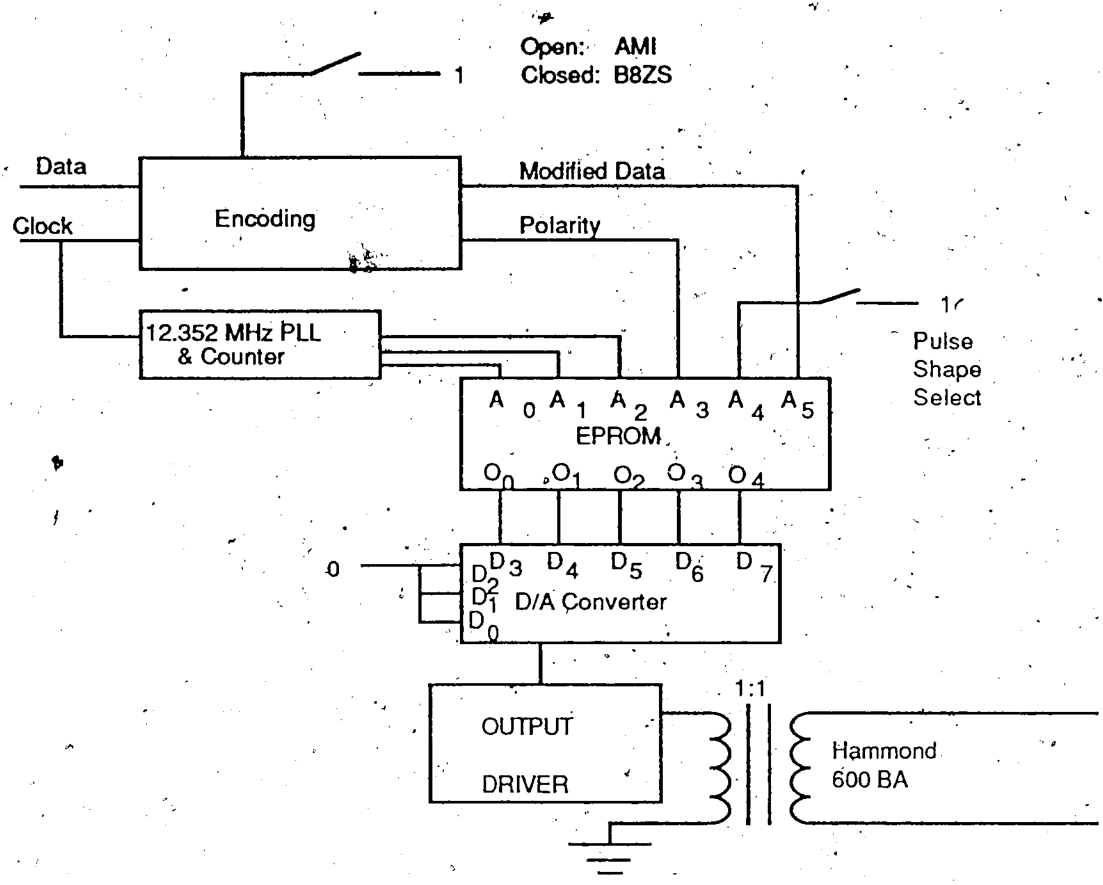


Figure 11: LBO Hardware Block Diagram

Figure 11 shows a functional block diagram o the LBO circuit; a detailed circuit schematic and timing diagrams are found in Appendix B. The circuit has 4 inputs. A DS-1, non-return to zero (NRZ) binary data signal and a 1.544 MHz clock are supplied by a DS-1 test set. Manual switches provide the other two inputs. One switch selects one of two encoding formats, and the other selects one of two pulse shapes. The LBO has one output: the bipolar, pre-distorted DS-1 signal. The encoding block and the phase-locked loop (PLL) and counter block generate the address inputs to the EPROM. Most of the LBO circuitry is devoted to the address generation. For each symbol in the DS-1 data stream, 8 samples must be read from the EPROM. The EPROM contents are divided into five groups of samples defining two different versions of both positive and negative polarity '1' pulses, and the '0' output. Generation of the EPROM addresses to cause the correct pulse shape to be read out requires approximately 75% of the LBO circuitry. The EPROM and the D/A are single ICs which require a small amount of support circuitry. The D/A output is amplified before being coupled to the transmission line with a transformer. Each of the functional blocks shown in Figure 11 will be described in detail in this section.

The LBO prototype circuit was built from discrete components and ICs. The circuit uses the 74HC family of high-speed CMOS (HCMOS) devices throughout, except for four special function ICs. The PLL uses two bipolar ICs: the MC4044 phase detector and loop filter, and the MC4324 voltage controlled oscillator. The D/A is the MC10318, which is an ECL device. Also, the MC10124 ECL level converters are needed to change the HCMOS logic levels (0 to 5V) to ECL logic levels (-1.8 to -0.8V). Other devices could be used to achieve the same functionality achieved with the devices listed above. The D/A, however, was particularly difficult to acquire. Several D/As meet the speed requirements, but they are in high demand as video D/As, and consequently are in short supply.

The encoding block has three inputs and two outputs. Two of the inputs are supplied by a DS-1 test set: the binary data stream, and the 1.544 MHz clock signal. Lines 1 and 2 of Figure 12 show the timing relationship between these two signals when they arrive at the LBO circuit. The clock is used for counters, latches, and other sequential elements in the encoding circuit. The data signal is manipulated to form twc outputs: a modified data signal, and a polarity signal. The encoding of the modified data and polarity signals is controlled by a third input: a binary switch.

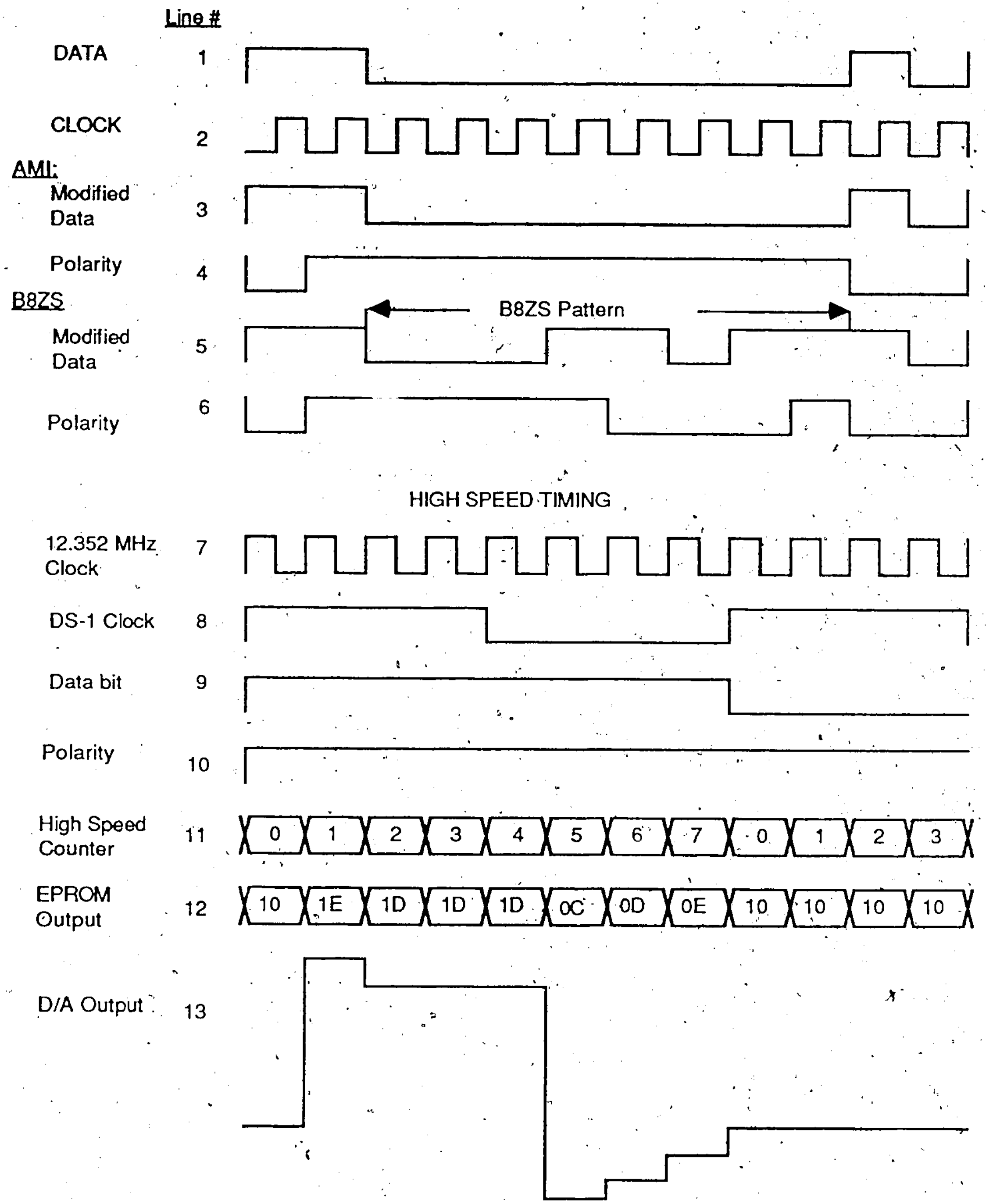


Figure 12: LBO Timing Diagram

At the output of the encoding block, the modified data and polarity signals are in binary, NRZ format. Some means is necessary to indicate the polarity of the '1' symbols in the modified data stream to ensure that the D/A output is a proper bipolar signal. This is accomplished by the polarity signal. When the switched input to the encoding block is open, the resulting bipolar data stream (D/A output) displays alternate mark inversion (AMI) coding. For AMI, the '1's in the bipolar data stream simply alternate in polarity. To indicate AMI coding, the polarity signal changes state at the start of each modified data symbol interval containing a '1'. The modified data is identical to the input data for AMI coding. Lines 3 and 4 of Figure 12 show the timing on the modified data and polarity signals when AMI encoding is selected. A '0' level on the polarity signal indicates that the '1' in the corresponding time slot in the modified data'signal will have negative polarity in the bipolar signal. A '1' level on the polarity signal indicates positive polarity. The polarity signal does not change state during modified data symbol intervals containing '0's. The concept of polarity for '0' symbols is meaningless, since '0' symbols are represented by a single level on the bipolar signal, namely 0 V.

When the switched input to the encoding block is closed, the encoding circuit performs binary with 8 zeros substitution (B8ZS) encoding on the input data. B8ZS encoding replaces strings of 8 consecutive '0's in the data stream with the pattern '000VB0VB'. The 'V's are bipolar violations, which are caused when a '1' bit in the bipolar data stream has the same polarity as the previous '1' bit. The 'B's are normal bipolar '1' bits which are the opposite polarity to the preceding '1' bit. Normally, bipolar violations are undesirable, because they cause an imbalance in the number of pulses of one polarity over those of the other polarity. This in turn causes a non-zero DC offset, which may affect the operation of receiver circuitry. However, the B8ZS pattern is such that the violations cancel each other, and the overall bipolar signal remains balanced about 0 V. The encoding circuit modifies the input data by replacing 8 consecutive zeros with the pattern '00011011'. The polarity signal is made to skip a state transition on the fourth and seventh bits of the replacement pattern. This indicates that the fourth and seventh bits will have the same polarity as the preceding '1's, thus accomplishing the B8ZS encoding. Lines 5 and 6 of Figure 12 show the outputs of the encoding circuit when B8ZS encoding is selected.

Both AMI and B8ZS encoded DS-1 signals are in use in the telecommunications network today. Thus, a DSX-1 receive circuit must be capable of operating under both types of encoding. The receiver design was to be an integral part in eliminating manual LBO switches. Therefore, the LBO circuit performs both types of encoding as a test feature: the switch controlling the encoding would not be necessary in a product implementation of the LBO circuit.

The modified data and polarity signals are two of the six address lines attached to the EPROM. The EPROM is a 16 Kbit device arranged as 2048, 8 bit words. The device used is a TMS27C292, with a 50 ns access time. The speed of this device is critical to the correct operation of the LBO. Samples are read from the EPROM every 81 ns (one period of the 12.352 MHz clock). Therefore the overall delay through a latch preceding the EPROM, plus the EPROM must be substantially less than 81 ns to meet the set-up time on a latch placed at the EPROM output. The latch placed before the EPROM retimes the address lines so that they all change simultaneously. The latch placed after the EPROM retimes the samples before they are presented to the D/A. These latches are not shown explicitly in Figure 11, but are used in the circuit.

The EPROM address lines change at a rate of 12.352 MHz. A steady stream of samples at 8 times the DS-1 symbol rate appears at the EPROM outputs, and is passed to the D/A. The 6 least significant address lines on the EPROM are used, which address $2^6 = 64$ memory locations. Table 3 shows a functional memory map of the EPROM contents. The combination of the three most significant address lines, A_5 to A_3, select an area in the EPROM where a particular pulse shape is stored. These lines remain fixed for the duration of a DS-1 symbol interval. The 3 least significant adress lines are clocked at 12.352 MHz, and address individual samples in the EPROM. The function of each address line will now be described in detail.

Data	Shape	Polarity	Counter		
A_5	A_4	A_3	A_2	A_1	A_0
0	D/A Output = 0V				
1	0	0	Negative Pulse Shape 1		
		1	Positive Pulse Shape 1		
	1	0	Negative Pulse Shape 2		
		1	Positive Pulse Shape 2		

Table 3: Functional EPROM Memory Map

The most significant address line, A_5, is the modified data signal coming from the encoding circuit. For modified data bits equal to '0', the D/A output should be OV. To achieve this, the first 32 locations of the EPROM contain the sample value 00010000 (10, hexadecimal) which corresponds to a O V D/A output. Thus, regardless of the value of the other address lines, when A_5 is '0', the D/A will generate O V. When A_5 = '1', the value of the other 5 address lines will cause the appropriate pulse shape to be read out. The pulse may be one of two shapes, each of which can have one of two polarities. Alternately, an external multiplexer could be used which routes a fixed sample value to the D/A when a '0' occurs on the modified data stream. This would halve the memory requirements, but increase the logic circuitry. Since a 2048 x 8 bit device was available, the amount of memory used was not a problem. However, if the digital LBO was integrated on an IC, the memory size would be a consideration.

The next most significant address line, A_4, is provided by an external switch. This switch controls which pulse shape is transmitted. A '0' on A_4 will select pulse shape 1 (for shorter lines), and a '1' will select pulse shape 2 (for longer lines). On the prototype circuit, this switch is controlled manually. However, it was intended that the selection of pulse shapes would be automated, as described in the next section.

The third most significant address line, A_3, is the polarity signal generated by the encoding block. A '0' on A_3 selects a negative polarity pulse, and a '1' selects a positive pulse. The 3 most significant address lines remain fixed for the entire duration of a DS-1 symbol interval. When a new DS-1 symbol interval begins, these 3 address lines immediately select the correct pulse shape of the correct polarity stored in the EPROM. During the ensuing symbol interval, the 3 least significant address lines change 8 times, thus reading the 8 samples which describe a DS-1 pulse out of the EPROM.

The function of the three least significant address lines is best described by explaining the operation of the PLL and counter block. The PLL generates a 50% duty cycle, 12.352 MHz clock that is synchronized to the 1.544 MHz DS-1 clock. The 12.352 MHz clock is divided by 8 to enable phase detection to be done at the 1.544 MHz rate. The MC4044 phase detector/loop filter chip is used in a third order PLL. The PLL was designed with a damping factor of $\zeta = 0.7$ and a natural frequency of $\omega_n = 50.7$ krads/sec. Additional filtering to that provided by the MC4044 is provided by a simple R-C low-pass filter with a 3 dB cut-off frequency of 250 krads/sec, or about 5 times ω_n. The resulting voltage is connected to the control input of the MC4324 voltage controlled oscillator (VCO). This PLL implementation produces approximately 2 ns of jitter on the 12.352 MHz clock signal. No further attempt was made to quantize this jitter because it had no effect on the operation of the circuitry. However, in a product design, the designer must ensure that all DSX-1 equipment conforms to standard jitter tolerances.

The 12.352 MHz clock drives a 3 bit binary up-counter. During each DS-1 symbol interval, the counter counts up from 000 to 111, and resets at the start of the next symbol interval. The counter cycles through these 8 states continuously, and the counter outputs form the three least significant bits of the EPROM address lines. The effect is to provide a continuous stream of evenly spaced samples at a rate of 12.352 Msamples/sec. to the D/A. Lines 7 to 12 of Figure 12 show the 12.352 MHz timing of the EPROM address lines and the EPROM output. The analog output of the D/A is shown in line 13 of Figure 12.

Table 4 shows the actual EPROM contents. Each memory location is 8 bits wide, but the samples require only 5 bits of storage each. The samples therefore occupy the 5 least significant bits in each memory location, and the 3 most significant bits are set to '0'. Only the 5 least significant EPROM output lines are connected to the D/A converter.

ADDRESS	CONTENTS	ADDRESS	CONTENTS	ADDRESS	CONTENTS
00	10	29	1E	35	17
I I	I I	2A	1D	36	14
1F	10	2B	1D	37	12
20	10	2C	1D	38	10
21	02	2D	0C	39	1F
22	03	2E	0D	3A	1C
23	03	2F	0E	3B	1C
24	03	30	10	3C	1C
25	14	31	01	3D	09
26	13	32	04	3E	0C
27	12	33	04	3F	0E
28	10	34	04		

Table 4: EPROM Contents, Two Shape Solution

The D/A used in the LBO circuit is an MC10318, ECL device. The device has a settling time to 1/2 of a least significant bit of accuracy of 15 ns., and is capable of conversion rates in excess of 25 MHz. Various other devices operate at these same speeds, but the MC10318 was one of few that were easily available in small quantities. The MC10318 is an 8 bit device capable of producing 256 unique analog quantization levels at the output. Since samples are only 5 bits wide, connecting the EPROM outputs to the 5 least significant inputs of the D/A would utilize only 1/8 of the dynamic range of the D/A output. To use as much of the D/A's dynamic range as possible, the EPROM outputs were connected to the 5 most significant inputs of the D/A. The 3 least significant D/A inputs were tied to '0'. In this configuration, the D/A will produce 32 unique output levels. The step size between levels will be equal to 8 quantization steps of the D/A. Nearly the full range of the D/A is used, from a minimum value of 00000000 to a maximum input of 11111000.

The MC10318 is a current output D/A with a full-scale output of approximately 51 mA. This amount of current was insufficient to drive the transformer and transmission line. A single transistor current driver was built to drive the line sufficiently to produce pulses which are as much as 3.6 V in amplitude. This circuit is necessary to ensure that the DSX-1 amplitude specification is met. The output driver is connected to one end of the primary coil of a Hammond 600 BA pulse transformer. The other end of the primary is grounded. The secondary winding of the transformer is connected to the transmission line. This arrangement is necessary to keep the transmission line free of any dc voltage. Only an ac, bipolar signal is present on the line. At the receiving end, the line would be connected to the primary coil of another transformer. The secondary would be grounded on one end, and connected to the receiver circuitry on the other. The line is isolated in this manner because the transmit and receive electronics cannot be guaranteed to have the same ground reference since they are physically separated.

In most D/A applications, the output of the D/A is filtered by a low-pass smoothing filter which removes the sharp edges caused by the quantization process. This is done to make the D/A output appear as a smoother, more natural analog signal. In the LBO application, no smoothing filter was used. The sharp edges on the D/A output were only apparent with 0 m of line loading the circuit. Attaching even a short length of cable to the LBO circut smoothed the signal sufficiently to make it appear as a true analog signal. However, the fast rise times and sharp edges at the D/A output imply that the signal contains very high frequency components, which may radiate and cause electromagnetic interference (EMI) with surrounding ciruitry. No EMI problems were encountered with the prototype. However, in a product implementation the use of a smoothing filter is recommended.

The LBO hardware was built and tested incrementally without connecting the circuit to a real transmission line. To thouroughly test pulse transmission over various 22 AWG line lengths ranging from 0 m to 200 m, the cable facility described in the next section was assembled.

4.2 Cable Test Facility

Several characteristics were desired in the cable test facility. Most importantly the cable length should be adjustable in small increments from 0 m to 200 m. The size of the increment should be about 15 m or so, and measurements of the pulse shape would be made at integer multiples of the line increment. An increment larger than 15 m would not provide a sufficient number of different line lengths for testing, and the greater resolution provided by a smaller increment was felt would not be necessary. The line length under test should be easily set up, and it should be simple to change between different cable gauges. The facility described below meets each of these desired characteristics.

The 22 AWG cable facility was fabricated using a 15 m length of 50 pair telephone cable. Connectors were attached to 48 pairs at both ends of the cable, and the connectors were attached to a jack panel. The small amount of cabling in the jack field panel adds 1.5 m to each pair in the 50 pair cable. Thus, each segment is 16.5 m long in total. The jack panel is configured so that all internal connections are closed when no jacks are inserted in the sockets on the front of the panel. The cable-end connectors were wired so that the twisted pair segments are connected end-to-end through the jack panel to form one, unbroken, single-pair line.

The front panel of the jack field has 48 columns of 3 sockets per column. One segment of line is connected between each column of sockets. Inserting a jack into the top socket in a column breaks the line at that point, and connects the jack to the line segment positioned to the left of the socket. The top row of sockets are used to place a termination at the end of a line. Inserting a jack in one of the middle sockets breaks the line at that point, and connects the jack to the line segment immediately to the right of that column of sockets. The middle sockets were used to inject test signals into the line. When jacks are inserted in one of the top two sockets of a column, the line breaks at the socket position only, into two pieces; all other line segments remain interconnected as before. Inserting a jack into the bottom row of sockets does not break the line, but taps into the line at the location of the socket.

Only 24 of the 48 columns of sockets were used, corresponding to a maximum total line length of 24 x 16.5 = 396 m. The bipolar DS-1 signal generated by the LBO circuit is injected into one end of the line using the centre socket in the leftmost column of the jack panel. A 100 ohm termination can be inserted into the top socket of any column to the right of the signal insertion point. The termination will be located at the end of a line with a length of 16.5 m times the number of columns between the insertion point and the termination point. The signal can be monitored across the 100 ohm termination, or at any point between the signal source and the termination using the bottom row of sockets.

Initial measurements were strongly affected by inductive coupling of the 60 Hz supply frequency onto the test cable. The 50 pair cable has an aluminum sheath that acts as a strength member. When this sheath was connected to ground, the inductive pick-up problem disappeared. Crosstalk was not noticeable due to the relatively short length of cable used. Similar multi-pair cables were available for construction of 24 and 26 AWG cable test facilities. To interchange between the different gauges, the correct cable is simply attached to the rear of the jack panel.

4.3 Measurement Method and Results

This section describes the method used to record transmitted DSX-1 pulse shapes, and discusses the results of the measurements made over 22 AWG cable. It was desirable to record the transmitted pulse shapes as they appeared at the cross-connect. As in the simulation phase, the pulse mask was to be recorded together with the pulse shape under test. To do this, a digital oscilloscope was used to observe the transmitted pulse shapes. The DSX-1 pulse mask was stored in the oscilloscope's memory, and could be displayed simultaneously with the pulse shape under observation. The oscilloscope display could also be plotted using an X-Y plotter to get a permanent record of the results. The method of creating the pulse mask and storing it in the oscilloscope memory is described in this section. Next, the method used to record transmission results is described. Finally, the results of transmission over 22 AWG cable are presented and discussed.

All measurements were made on a Tektronix 2430A digital oscilloscope. This oscilloscope has 4 memories where signal traces can be stored. A digitized pulse shape can be transferred to one of the trace memories from an external device via an IEEE-488 bus. To transfer the DSX-1 pulse mask to the 2430A, the upper and lower curves of the pulse mask are transferred into separate trace memories. The oscilloscope format requires that each trace be described by 1024 sample points. Rather than interpolating over both curves of the DSX-1 pulse mask to get 1024 discrete points for each, an indirect method was used which produced the same results much faster.

The Tektronix 7854 digital storage oscilloscope has a feature that allows a user to draw waveforms on the oscilloscope screen, and to store them for use later. This feature was used to draw the ~~top~~ and bottom curves of the DSX-1 pulse mask. Each curve was stored in a separate trace memory of the 7854 oscilloscope. This scope stores waveforms in the same format as required by the 2430A, therefore no manipulation of the data was necessary. The contents of the two 7854 trace memories were transferred to a diskette in an IBM personal computer equipped with an IEEE-488 interface card. The two disk files were subsequently transferred to the 2430A oscilloscope trace memories 1 and 2 (RF1 and RF2). The two traces are displayed on the oscilloscope simultaneously to display the full DSX-1 pulse mask. The sloping parts of the traces show jagged steps caused by the quantizing process, but this does not affect the judgement of whether a particular pulse shape fits the mask or not. The 2430A has a battery powered memory system, and the pulse mask traces remain intact after the power is shut-off.

The bipolar test pattern for all measurements was +1, 7 zeros, -1, 7 zeros, etc., which is the DSX-1 test pattern specification. All measurements were made across a 100 ohm resistive termination. Results of the measurements were compared with measurements made at the midpoint of a line twice as long, with the 100 ohm termination placed at the far end of the line. This was the structure that was simulated. The differences in the measured pulse shapes were imperceptable, and all measurements were made at the point where the cross-connect would be located, across the 100 ohm termination.

The bipolar signal travels differentially over the twisted pair cable: the full signal appears between the two wires of the pair. The signal cannot be measured directly using one oscilloscope probe, because the reference voltage for the probe is ground. Differential measurement techniques are needed to observe the full signal on the oscilloscope. The grounds of two oscilloscope probes were soldered together at the ends of the two probes. One probe was connected to one end of the 100 ohm termination and attached to channel A of the scope. The other probe was connected to the other end of the terminating resistor and attached to scope channel B. The channel B display was inverted, and the sum of channel A and inverted channel B reproduces the full signal on the cable. The total signal was displayed on the oscilloscope screen together with the DSX-1 pulse mask. Amplitude scaling and time base shifting were the only allowed operations on the signal to check if a pulse fit inside the DSX-1 pulse mask. Both negative and positive pulses were checked for fit inside the mask. Negative pulses were also checked to ensure that they were equal in amplitude to the positive pulses. To check a negative pulse for fit in the DSX-1 pulse mask, the polarity of the total signal is inverted on the scope display by reversing channel A and B. The previously negative pulses are now shown with positive polarity, and can be tested for fit inside the pulse mask. Both positive and negative pulses had the same shape and amplitude, and all further measurements were made on the positive pulse shape only.

Figures 13 to 16 show plots of the pulse shapes for four different lengths of 22 AWG cable. Figure 13 shows pulse shape 1 as seen with 0 m of line attached to the LBO circuit. At this line length, the pulse amplitude is 3.6 V, and the pulse fits well inside the DSX-1 pulse mask. The line length was increased in increments of 16.5 m until the pulse could n [illegible] within the mask. This occured at 1.65 m, and the pulse shape is shown in Figure 14. The pulse amplitude of 2.84 V at 165 m is well within the 2.4 V to 3.6 V range specified by the DSX-1 standard. Figure 15 shows pulse shape 2 as it appears after travelling over 214.5 m of line, which is just over the maximum DSX-1 line length of 200m. The pulse amplitude at this length of cable is 2.46 V, which is just within the DSX-1 standard. The line length was decreased in 16.5 m increments until pulse shape 2 could not meet the DSX-1 mask, which occured at 82.5 m, as shown in Figure 16. At this length, the pulse amplitude is 3.14 V. All measured pulses within the two ranges meet both the DSX-1 pulse shape and amplitude specifications. Other DSX-1 specifications such as signal power and pulse width were not measured, since any pulses meeting the shape and amplitude specifications will also meet the power and pulse width specifications.

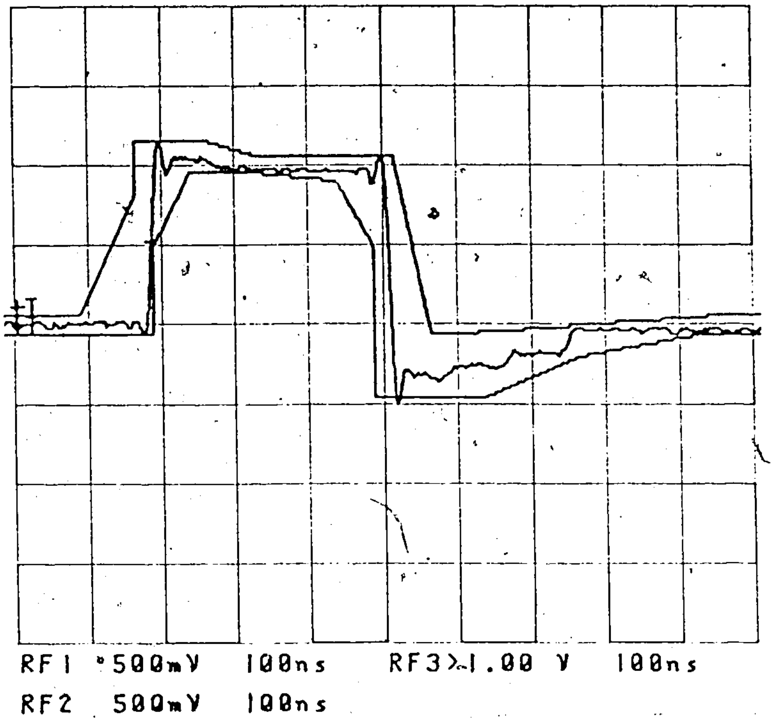
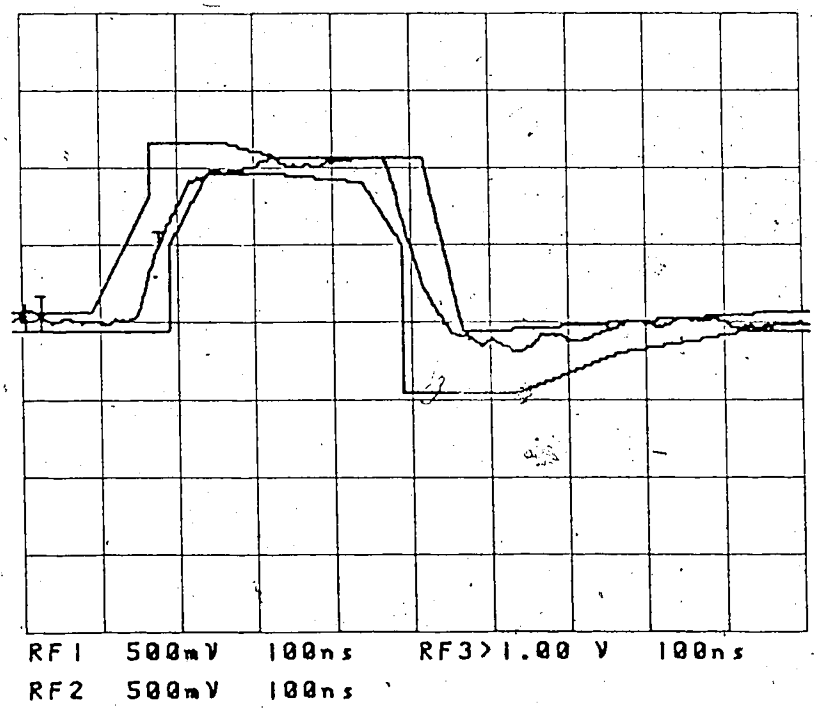


Figure 13: Measured Pulse Shape 1, 0 m of 22 AWG cable



F 14: Measured Pulse Shape 1, 165 m of 22 AWG cable

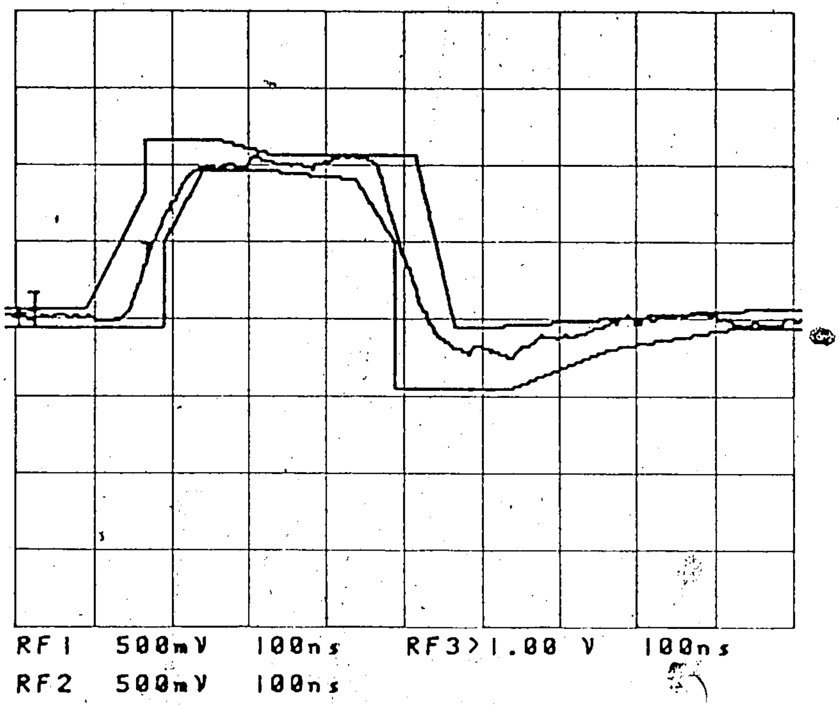


Figure 15: Meaured Pulse Shape 2, 214.5 m of 22 AWG cable

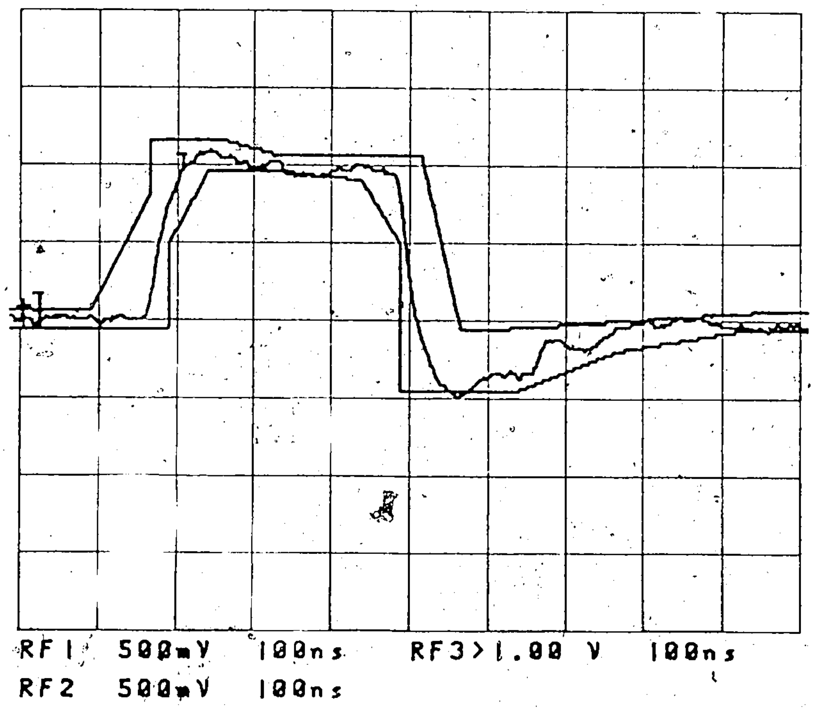


Figure 16: Measured Pulse Shape 2, 82.5 m of 22 AWG cable

Several characteristics of the measured results were noted. First, the pulse shape specification was the limiting factor deciding the valid range for each pulse shape, and not the pulse amplitude. This result proved to be very significant when eliminating manual LBO switches, as will be shown later. Second, the overlap of the valid ranges for the two different pulse shapes was thought to be significant. It was felt that the overlap would simplify the design of a receiver which could eliminate manual LBO settings. Generally, the measured and simulated ranges were quite close. The ıeasured range for pulse shape 1 is longer, and the measured range for pulse shape 2 is shorter than the simulated results. This difference is due to the transformer coupling and other real circuit effects that were not simulated. One additional observation made was on the measured results for pulse shape 1 shown in Figures 13 and 14. It seemed that it might be possible to extend the valid range beyond 165 m with some slight pulse shape modifications.

The measured results confirmed what the simulation phase had shown - that the digital LBO approach was feasible. In general, simulated and measured results agreed quite closely. The next step was to devise a method to automate the selection of an appropriate pulse shape, and eliminate manual LBO switch settings. The following two sections describe how this was accomplished. The initial method failed due to a fundamental characteristic of the DSX-1 cross-connect, but a second approach proved to be successful.

4.4 Automatic Line Length Determination

Based on the results of the last section, the following technique for automating pulse shape selection was envisaged. It was hypothesized that a simple device could monitor the received signal and perform some rudimentary frequency domain analysis. Based on the analysis, an estimate of the cable length between the cross-connect and the equipment could be made, and the appropriate LBO setting could be selected. This technique assumes that transmit and receive cables are of the same length, which is the configuration specified by the DSX-1 standard. The significant characteristics of the measured results are that only two LBO settings are needed, and the two valid ranges overlap between the line lengths of 82.5 m and 165 m. Thus, the decision as to which pulse shape to choose could have a large margin of error.

about some point in the centre of the overlap region. The large overlap could be used to add hysteresis to the decision process to prevent decisio oscillations when the line length is very close to the decision point.

More than one variation of the proposed frequency domain analysis could be used. In general, the technique filters out some portion of the received signal spectrum, and compares this to another part of the spectrum, or to the broadband signal. The filtered signals are not directly compared to each other, rather some characteristic of the signals, such as peak amplitude or signal power, is compared. A ratio would be a logical choice for the comparison technique, since a ratio would be independent of the absolute signal level, which may vary from one transmitter to the next.

The frequency domain analysis technique must be independent of both pattern and signal level. A major design difficulty is to determine which portions of the signal are pattern and level independent, and which characteristic of the signal (peak amplitude, etc.) is best suited for camparison, and what the comparison technique should be (ratio or other). The desired result of all this manipulation is a dc signal, either a voltage or current, that is proportional to the line length. The line transfer function changes as the line length is increased. For longer lines, the amplitude response rolls off at lower frequencies and has a steeper slope after roll-off begins for the longer lines. This change in frequency response should be detectable with the proper frequency domain analysis technique. It is not certain that a signal with all the necessary characteristics can be found, since this method was not thoroughly examined once it was determined that it could not work with the DSX-1 cross-connect arrangement. If such a length indicating signal can be found, however, it could be applied to a simple comparator with hysteresis, and compared to a fixed reference level. The reference level would correspond to a line length in the centre of the overlap region. The two-state comparator output would be used to select the transmit LBO pulse shape.

This proposed technique fails when used at the DSX-1 interface, due to a fundamental characteristic of the DSX-1 cross-connect. This characteristic is that the DSX-1 cross-connect is transparent to the line, except for the insertion loss of the connectors at the cross-connect panel. The cross-connect carries signals internally over 22 AWG wire, and the total line physically remains unbroken. Because of this characteristic, two different line configurations can result in the identical signal arriving at the receiver, but require that a different decision be made in each case.

Figure 17 illustrates two such configurations. In Figure 17 (a), the cross connect is 120 m from a DSX-1 signal source, which is transmitting pulse shape 1. The transmitter could have just been powered-up, and chose pulse shape 1 as a default. The receiver is located 75 m from the cross-connect. Since the cross-connect is transparent, the receiver will see a signal that has travelled over 195 m of line, assuming that the internal cross-connect wiring length is negligible. In this configuration, the receiver must choose to send pulse shape 1, since 75 m is less than 82.5 m, which is the minimum line length for which pulse shape 2 could be used. In Figure 17 (b), the signal source is again transmitting pulse shape 1, but the cross-connect is located a negligible distance from the signal source. The receiver is located 195 m from the cross-connect. The receiver will see exactly the same signal as in the configuration in Figure 17 (a). However , the correct decision in this case is to transmit pulse shape 2, since 195 m is beyond the valid range for pulse shape 1. The receiver must therefore make two different decisions when the same signal is received, which is impossible! The receiver cannot distinguish between the two configurations shown in Figure 17.

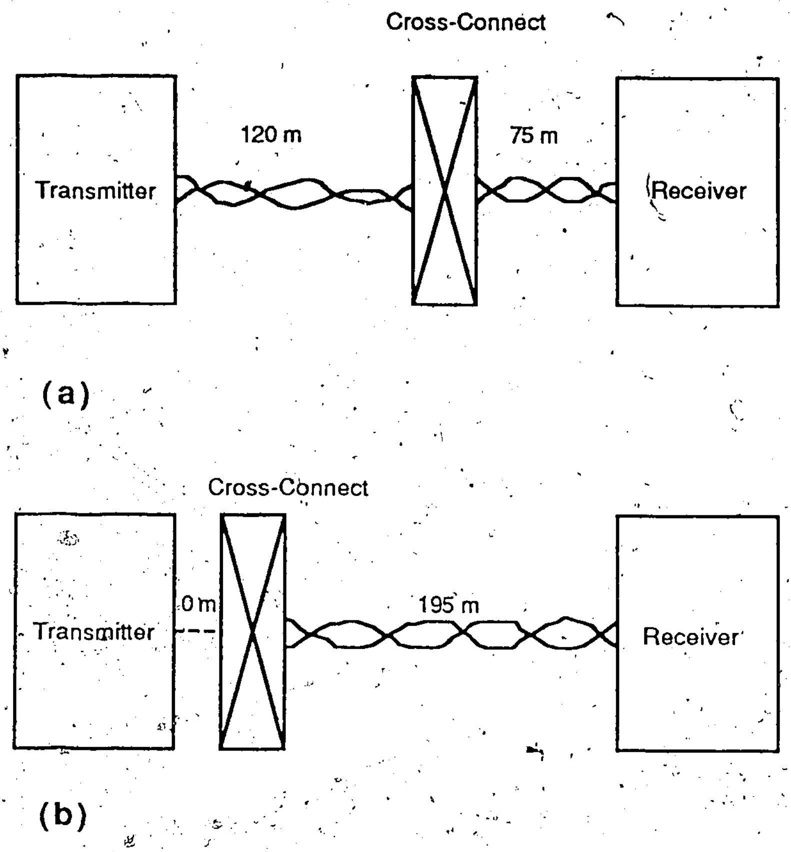


Figure 17: Two Equipment Configurations Resulting in the Same Signal at the Receiver

Having found that this initial method failed to eliminate the manual LBO switches, an alternate method for eliminating manual LBO control was needed. The next section describes how LBO switches were eliminated altogether.

4.5 Single Pulse Shape

The measured results for the transmission of the two separate pulse shapes over 22 AWG cable provided insight to the solution of the manual LBO setting problem. Pulse shape 1 was found to meet the DSX-1 standard over a range of line lengths of 0 m to 165 m. This result caused the following question to be asked: could pulse shape 1 be modified to extend the valid range to cover the full 200 m? This would eliminate the manual LBO switches completely, rather than trying to automate the pulse shape selection process. Close observation of pulse shape 1 indicated that this might be possible. This section describes how pulse shape 1 was modified, and presents the oscilloscope displays of the resulting pulse shape for 0 m and 200 m.

Figures 13 and 14 showed the plots for transmission of pulse shape 1 over 0 m and 165 m of 22 AWG line, respectively. Observing Figure 14, it can be seen that at 165 m, there is no rising edge overshoot, and the pulse is clipping the minimum curve of the mask at the top of the rising edge of the pulse. This problem can be overcome by increasing the overshoot on the transmitted pulse. Figure 14 also shows that at 165 m, the trailing edge undershoot is too small. The small undershoot, combined with the pulse spreading caused by the line, cause the pulse to exceed the upper curve of the pulse mask at the pulse's trailing edge. The remedy for this problem is to increase the undershoot of the transmitted pulse. However, any changes made to the transmitted pulse must not cause the pulse to fail to meet the DSX-1 standard at 0 m. Observation of Figure 13 shows that at 0 m, some leeway is present to increase the rising edge overshoot and the trailing edge undershoot of pulse shape 1, without exceeding the DSX-1 pulse mask. However, it is not clear whether the resulting pulse shape could extend the upper end of the valid range from 165 m to 200 m.

For these fine adjustments to the pulse shape, the simulation program was not helpful. The program was not a hardware simulation, and did not represent a real circuit implementation of the LBO. The simulation served its purpose in identifying the sampling rate and sample size, plus the initial pulse shapes. Fine tuning of the pulse shape was done by editing the EPROM contents. The following iterative process was used to arrive at the final pulse shape. Pulse shape 1 was modified by editing the EPROM contents. The resulting pulse was transmitted, and observed at line lengths over 165 m, and also at 0 m to ensure compliance with the DSX-1 standard over the full range. If the pulse did not fit the mask for line lengths up to 200 m, the pulse shape was modified and tested again. Each iteration involves erasing the EPROM under an ultra-violet lamp, and using a programming device to enter the new samples into the EPROM.

The final pulse shape was found in 2 iterations. Only 3 of the 8 segments differed from the original pulse shape 1. The second segment, which defines the rising edge overshoot, is 1 quantization level higher than the original pulse shape. The trailing edge undershoot was increased by making the sixth and seventh segments 1 quantization level lower. The remainder of the pulse remained unchanged. Table 5 shows the contents of the EPROM which achieves the single shape solution.

ADDRESS	CONTENTS	D/A OUTPUT
00 I I 1F	10 I I 10	0 Pulse Level
20	10	Negative Pulse
21	01	
22	03	
23	03	
24	03	
25	15	
26	14	
27	12	
28	10	Positive Pulse
29	1F	
2A	1D	
2B	1D	
2C	1D	
2D	0B	
2E	0C	
2F	0E	

Table 5: EPROM Contents, Single Shape Solution

Table 5 shows that only 10 unique samples, and hence 10 unique voltage levels, are used to generate the two pulse shapes plus the 0 V level. This small number of voltage levels indicates that a full-function D/A is not necessary. A simple custom D/A circuit could be designed to generate only the 10 different levels needed to create the pulse shapes. This circuit would be much smaller than a full-function D/A, and could be integrated onto the same chip as the other LBO circuitry. The D/A circuit could be made small enough to allow several complete LBO circuits to reside on a single IC. An application for a multiple LBO chip would be a DS-3 rate demultiplexer which would recover 28 separate DS-1 signals, which would be interconnected to other equipment via the DSX-1 cross-connect.

Figures 18 and 19 show the results at the two endpoints of the valid range, 0 m and 198 m, respectively. The 198 m maximum length is obtained by using 12 of the 16.5 m sections of line, and one can assume that the pulse will fit at 200 m of line as well. The pulse fits the DSX-1 pulse mask well at both endpoints, and at all points in between. At 0 m, the pulse amplitude is 3.6 V, the maximum allowed by the DSX-1 standard. By 198 m, the amplitude has deteriorated to 2.74 V, but still meets the DSX-1 specification of a 2.4 V minimum pulse amplitude. This proved the assumption made at the simulation phase that amplitude would not be a limiting factor for a particular pulse shape over the full 200 m range of line lengths.

The results presented in this section represent a significant accomplishment: DSX-1 LBO settings have been entirely eliminated. The flexibility of the digital LBO approach permitted the creation of a single pulse shape that could meet the DSX-1 standard when transmitted over any length of 22 AWG cable between 0 m and 200 m. This solution is elegant due to its simplicity. LBO settings have been eliminated without the need for complex signal processing or line length determination circuitry. The costly record keeping and inconvenience of manual LBO settings are no longer a problem. In addition, 1:N protection of LBO circuits is simplified. Since all LBOs are operating of one setting, it is straightforward to have an auxiliary LBO circuit on standby, which could protect several in-service LBOs. In addition, the low operating speed and all digital nature of the LBO open the possibility of integrating several devices on a single IC.

The results shown in Figures 18 and 19 prove that the single pulse shape approach is a valid DS-1 LBO design technique. However, a comparison to the foremost in today's technology was desired to confirm that the quality of the pulse shapes measured at various cable lengths was at least equivalent to those generated by circuits in use today. One of the LBO circuits reviewed in Table 1, in Chapter 2, was purchased, and its performance compared to that of the single-shape digital LBO.

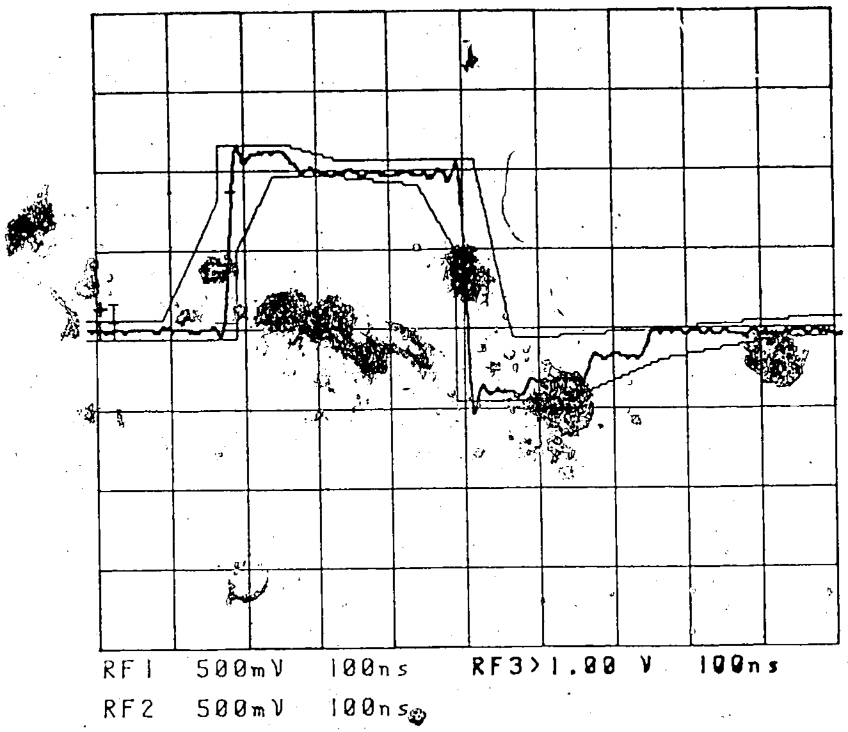


Figure 18: Single Pulse Shape Solution, 0 m of 22 AWG cable

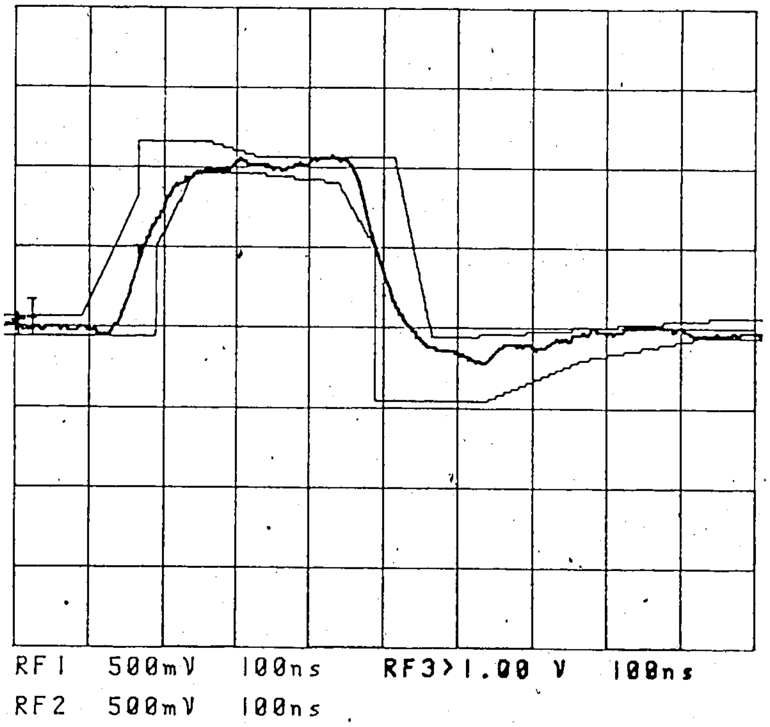
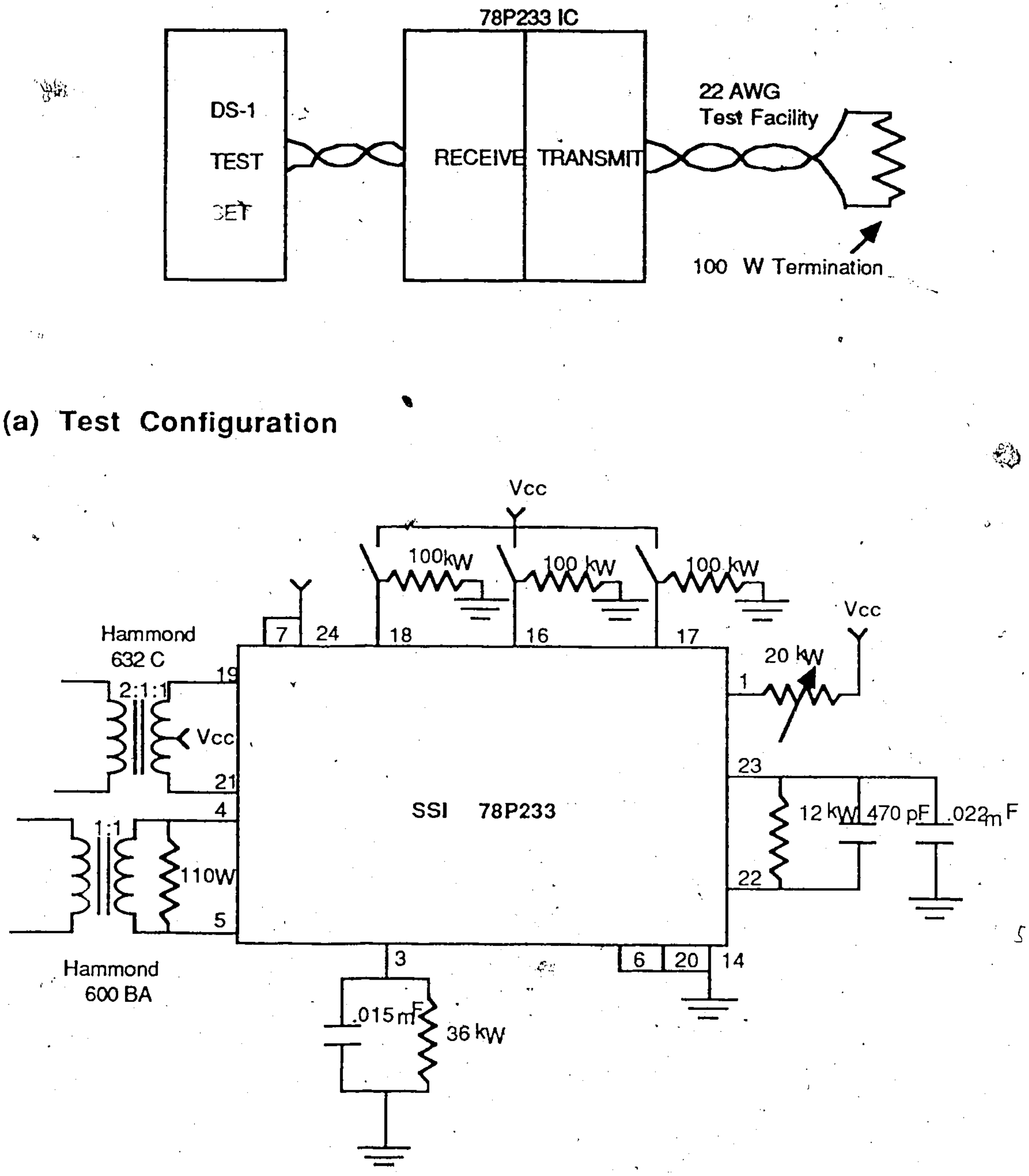


Figure 19: Single Pulse Shape Solution, 198 m of 22 AWG cable

4.6 Comparison with 78P233

This section describes the results of measurements made using the Silicon Systems 78P233 DS-1 LBO chip as the signal source. The IC was connected to supporting circuit components and attached to the 22 AWG line via a coupling transformer. This section describes the required circuit connections, and shows plots of pulse shapes transmitted by the 78P233. A short discussion compares the pulse shapes with those measured in the preceding section.

Figure 20 (a) shows the block diagram of the measurement system which uses the 78P233 to transmit a signal to the cross-connect. The transmit portion of the IC requires that the NRZ DS-1 data stream be divided into two signals: one that carries all '1' bits that will have positive polarity, and one data stream with only the negative polarity '1' bits. The two separate signals produce the total DS-1 binary signal when 'OR'-ed together. The DS-1 test set can not provide these two data streams directly. However, the 78P233 has a loop-back feature that allows the received DS-1 data stream to be used as the input to the transmit LBO circuitry. The bipolar output of the DS-1 test set meets the DSX-1 standard, and was connected directly to the receive circuitry of the 78P233, as shown in Figure 20 (a). The 78P233 was set to operate in the loop-back mode, and the received data was connected internally to the transmit circuitry. The output of the 78P233 transmitter was connected to various lengths of 22 AWG line for testing.



(a) Test Configuration

(b) Test Circuit Schematic

Figure 20: LBO Circuit using the SSI 78P233

Figure 20 (b) shows the detailed circuit connections and all component values used,for the test circuit. The components connected to pins 22 and 23 form the loop filter for the clock recovery phase-locked loop. Tying pin 14 to ground (logic '0') causes the received data and clock to be used as inputs to the transmit circuit. The 20 kohm potentiometer connected to pin 1 is used to establish the centre frequency of the oscillator in the clock recovery circuit. The two components connected to pin 3 are used by a peak detector circuit to set threshold levels for internal comparators. The input transformer can be any 1:1 transformer that is fast enough to adequately pass a DS-1 waveform. The most critical component is the output transformer which couples the 78P233 output to the 22 AWG cable. This transformer must have certain characteristics to ensure that the pulse shape generated by the IC is not affected during coupling to the actual line. The data sheets for the 78P233 specify the characteristics for the output transformer. Table 6 shows the specified characteristics in one column, and the characteristics of the Hammond 632C transformer in the adjoining column. Table 6 shows that the 632C is an acceptable choice for the coupling transformer. Unfortunately, two of the parameters for the transformer were not specified in the manufacturer's data sheets for the device. However, the 632C was the only commercially available transformer found that met most of the 78P233 requirements, and it was felt that the pulse shapes measured at various lengths of line should be representative of a typical application using the 78P233 as the LBO circuit.

Characteristic	78P233 Requirements	Hammond 632 C
Primary Open Circuit Inductance	1.25 mH, minimum	5 mH
Primary Volt-Time Product	10 V- μ sec, minimum	160 V- μ sec
Primary DC Resistance	1 Ω , maximum	0.84 Ω
Secondary DC Resistance	1 Ω , maximum	0.5 Ω
Effective Primary Distributed Capacitance	15 pF, maximum	Not Available
Primary Leakage Inductance	2 μ H, maximum	Not Available

Table 6: Output Transformer Characteristics for SSI 78P233

Figures 21 and 22 show the measured pulse shape at 0 and 198 m, respectively. Comparing Figure 18 with Figure 21, one can see that the undershoot on the pulse in Figure 21 is much smaller than that on the pulse in Figure 18. This is due to the analog equalizer network used inside the 78P233 to predistort the DS-1 pulse. It cannot meet both the fast rise time and the large slowly decaying undershoot simultaneously. Also, the amplitude of the pulse generated by the 78P233 at 0m of line is 3.92 V, which is larger than the 3.6 V maximum specified by the DSX-1 standard. However, the data sheets for the 78P233 state that the IC meets the DSX-1 standard. One possible explanation for these results is that the 632C is not truely compatible with the 78P233 specifications. Figure 22 shows that at 198 m, the pulse generated by the 78P233 just exceeds the DSX-1 mask at 3 points: at the top of the rising edge, at the peak pulse amplitude, and at the base of the trailing edge of the pulse. The amount that the pulse exceeds the mask is small, and would likely not cause problems with data and clock recovery at the receiver. A coupling transformer with a lower inductance, and thus a faster frequency response, might improve the shape slightly. By comparison, the pulse shape shown in Figure 19 fits completely inside the DSX-1 pulse mask. For both the digital LBO and the 78P233, the pulse amplitude is within specifications at 198 m: 3.6 V for the 78P233, and 2.74 V for the digital LBO. To cover the 200 m range, the 78P233 has 5 discrete equalizer settings. These are controlled by the manual switches connected to pins 16, 17, and 18, as shown in Figure 20(b). At the end of the valid range for each setting, the pulse shape had difficulty fitting inside the DSX-1 mask, similar to the problems experienced by the pulse shown in Figure 22.

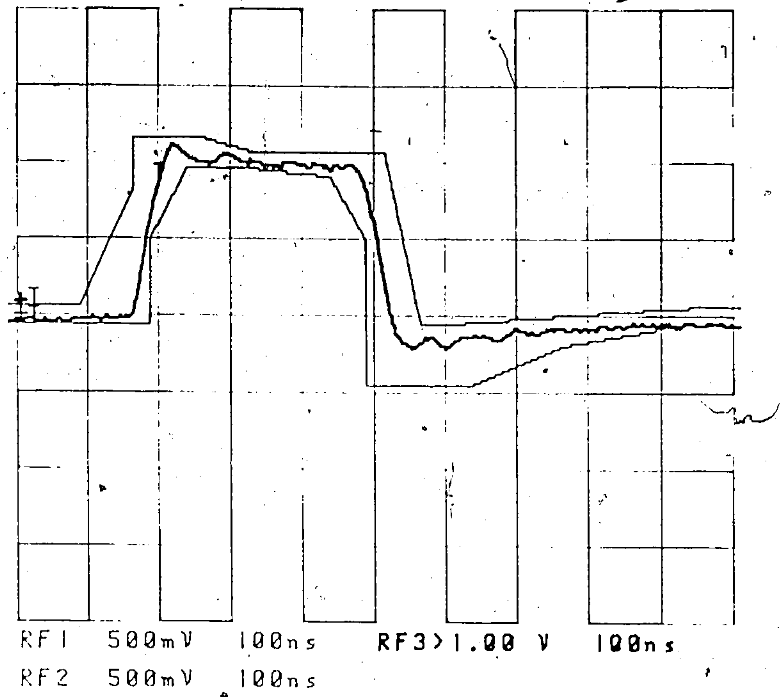


Figure 21: 78P233 Output, 0 m of cable

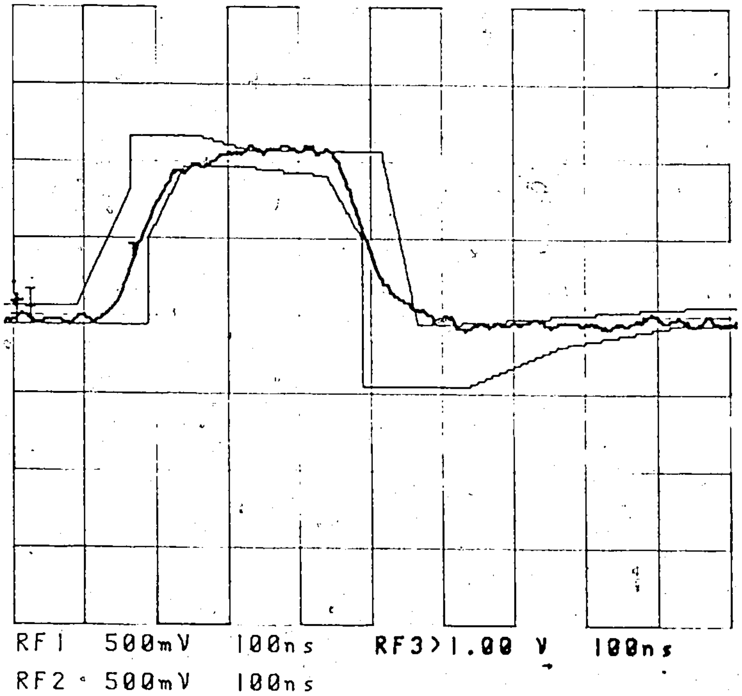


Figure 22: 78P233 Output, 198 m of 22 AWG cable

These results confirmed that the single-shape digital LBO could produce DSX-1 compatible pulse shapes with the same or better quality than today's technology, over the full 200m range of line lengths. The primary advantage of the digital LBO over today's solution is that it completely eliminates the need for manual control. This simplifies system maintenance for telephone companies since they no longer need to keep records of the LBO settings of the DS-1 equipment. At this point, the design of the LBO was complete; all objectives outlined in chapter 2 had been achieved. The digital pulse generation technique was felt could be used for DSX-1 related applications other than the LBO. Two possible applications of the digital pulse generation technique are presented in the next chapter.

CHAPTER 5

EXTENDED APPLICATIONS

This chapter examines two extended applications of the digital pulse generation technique used in the LBO design. Two main features of the EPROM and D/A combination are exploited in this section. One feature is the ability to generate complex pulse shapes not possible using linear devices. The other feature is the flexibility resulting from the use of a digital memory device: pulse shapes can be easily altered, many pulses can be stored in a single device, and devices can be interchanged easily to provide access to an unlimited choice of pulse shapes. The first application seeks to extend the operation of the LBO for transmission over 24 and 26 AWG cables. The higher gauges reduce the cabling space requirements, which can be a significant advantage to a telephone company. The second application is the design of a new DSX-1 test set.

5.1 24 & 26 AWG Results

There are several advantages to using 24 or 26 AWG cable over using 22 AWG cable. The DSX-1 standard issued by AT & T does not specify the type or length of cable used: it merely specifies the electrical characteristics that a signal must meet at the DSX-1 cross-connect panel. The telephone companies and DS-1 equipment manufacturers have extended the standard to include the 200 m maximum cabling destance and restricted the type of cable to 22 AWG. These limitations to the standard were necessary to allow DS-1 equipment manufacturers to produce inexpensive LBO and receiver circuits for the DSX-1 interface.

The advantages of 24 and 26 AWG cable are a result of the wire size. The smaller wire diameter of higher gauge wire enables more wire pairs to be connected to a single bay of equipment. Often, the wire connections, and not the size of the electronics limit the capacity of a bay. The smaller wires require less copper, and hence cost less than the 22 AWG alternative. This section begins by presenting the results of transmission over 24 and 26 AWG cable of the single pulse shape developed for 22 AWG cable. As expected, the 22 AWG solution failed for 24 and 26 AWG cables. This section describes the simulation and design procedure used to develop pulse shapes to satisfy the DSX-1 standard over 0 m to 200 m of 24 and 26 AWG cable.

Testing of 24 and 26 AWG cables began by checking the performance of the single pulse shape developed for 22 AWG cable. Since the higher gauge cables have smaller wire diameters, and hence a higher resistance per unit length than 22 AWG wire, the single pulse shape was expected to have difficulty meeting the DSX-1 specification over the full 0 m to 200 m range of cable lengths. The plot of the single-pulse shape with 0 m of line is shown in Figure 18 in the preceeding chapter. Figure 23 shows a plot of the single pulse shape measured at 181.5 m of 24 AWG cable, which is the length beyond which the pulse no longer fits the DSX-1 pulse mask. For 24 AWG cable, the DSX-1 pulse amplitude specification is met over the full 198 m range of line lengths tested. At 198 m, the pulse just meets the amplitude specification with a value of 2.42 V. The results show that the single pulse shape is valid for 24 AWG cable for nearly the entire 198 m reach, but that a second pulse shape would be needed to meet the standard at longer line lengths, up to the full 198 m reach.

Figure 24 shows a plot of the single pulse shape for 165 m of 26 AWG cable. Beyond this length, the pulse does not fit the DSX-1 pulse mask. The pulse amplitude, however, fails to meet specifications for lengths over 132 m. At 132 m of cable, the pulse amplitude is 2.44 V.

Figures 23 and 24 indicate that for longer cables, the single pulse shape must be modified if it is to fit inside the DSX-1 pulse mask. Observation of the pulse shapes indicates that the rising edge overshoot and the trailing edge undershoot need to be increased. This means that at least 2 different pulse shapes, and hence LBO settings, are needed to meet the DSX-1 specification over the full 0 to 198 m range of 24 or 26 AWG line lengths. This implies that manual switches will be necessary to control the LBO settings, which is undesirable. This is the compromise which must be made if the benefits of using the smaller gauges are to be realized. The setting for longer lines must also include additional pulse amplification, which could be controlled by the same switch that selects the appropriate pulse shape.

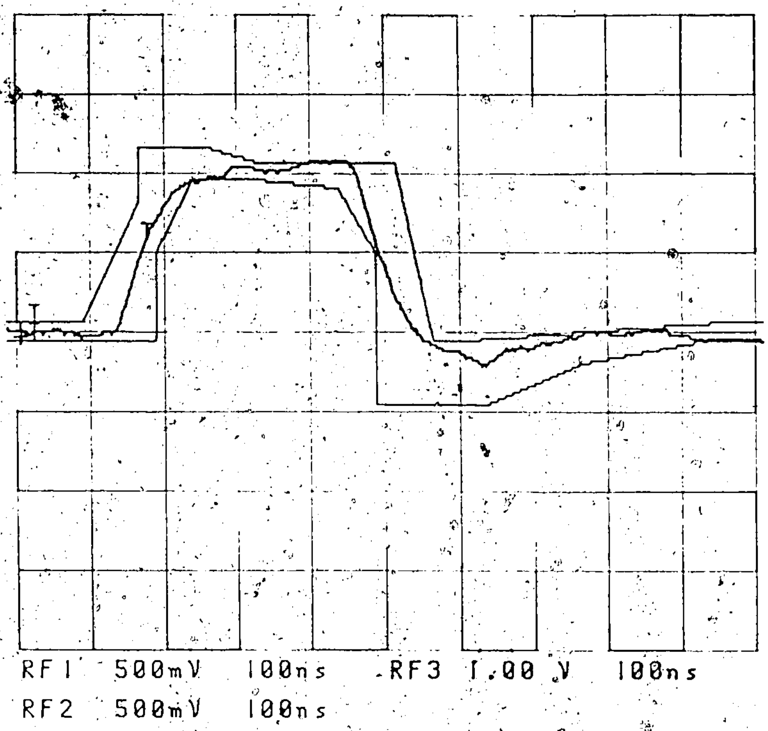


Figure 23: Single Pulse Shape Solution, 181.5 m of 24 AWG cable

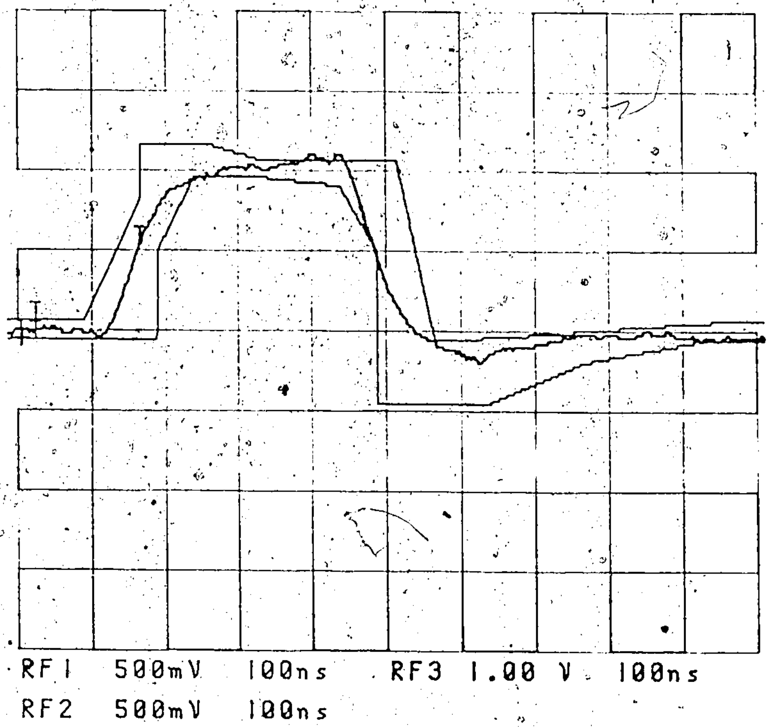


Figure 24: Single Pulse Shape Solution, 165 m of 26 AWG cable

Computer simulations failed to help in the design of pulse shapes valid for 24 and 26 AWG cable. Simulations using the models for 24 and 26 AWG cable found in Appendix A gave virtually the same results as simulations using the 22 AWG cable model. Close examination of the three models showed that tabulated values for Z_0 and γ were semilar for each wire gauge. The measured parameters from which these models were derived were then examined. They showed that for frequencies above approximately 100 kHz, the RLGC line parameters were nearly equal. The line parameters were significantly different only for frequencies below approximately 100 kHz. To make these differences noticeable in the computer simulation results, the frequency spacing of the Fourier coefficients must be decreased. This can be done by increasing the length of the stimulus vector to cover more than 8 DS-1 symbol intervals. A vector that is 128 DS-1 symbol intervals long results in a Fourier coefficient spacing of 12.0625 kHz. However, the FFT requires 32 samples per DS-1 symbol for an adequate time domain representation. The resulting stimulus vector contains 128 x 32 = 4096 samples. When a vector of this size was used as the input to the simulation, the computer program crashed due to inadequate memory space. Rather than modifying the program to be more memory efficient, it was decided to use a more direct approach, and modify the EPROM contents directly.

An iterative procedure was used to design the pulse shape which would be valid for long 24 and 26 AWG cables. This procedure was identical to that used in designing the single pulse shape for 22 AWG cable. The plots in Figures 23 and 24 indicated that the rising edge overshoot and the trailing edge undershoot needed to be larger for longer cable lengths. The EPROM contents describing the single pulse shape were modified to incorporate these changes, and the resulting pulse tested over the two cables. Two iterations of this procedure were needed to arrive at the final solution. Table 7 shows the EPROM contents for the two shape 24 and 26 AWG cable solution.

Both 24 and 26 AWG cables require two unique pulse shapes to cover the entire 200 m range of line lengths. The two shapes are the same for both cables, but result in different valid ranges for each cable. The pulse shape for short lines remains as the single pulse shape used for 22 AWG cable. For longer lines, a new pulse shape was designed. Figure 18 shows the short pulse shape with 0 m of line, and Figure 23 shows the same pulse at 181.5 m of 24 AWG line. For 24 AWG cable, the valid range of the long pulse is 99 m to 198 m. Figures 25 and 26 show plots of the long line pulse shape on 24 AWG cable at 99 and 198 m of line respectively. At 99 m, the pulse amplitude is 2.78 V, and at 198 m it is 2.4 V. Both values are within the DSX-1 amplitude specification, but an amplitude increase of 0.5 V at the signal source for the long line pulse shape would produce a stronger signal, while maintaining adherence to the DSX-1 standard.

Figure 24 shows the short pulse shape after 165 m of 26 AWG line, which is the limit of the valid range for that pulse shape. However, since the amplitude specification is not met for 26 AWG cables over 132 m long, the valid range should be restricted to 0 m to 132 m. Figures 27 and 28 show the long pulse shape at the endpoints of the valid ranges of 26 AWG cable: 82.5 and 198 m, respectively. The pulse amplitude at 82.5 m is 2.77 V, and at 198 m it is 2.15 V. To adequately meet the DSX-1 amplitude specification at 198 m, the signal source should increase the pulse amplitude by at least 0.3 V, and 0.7 V is recommended.

ADDRESS	CONTENTS	ADDRESS	CONTENTS	ADDRESS	CONTENTS
00	10	29	1F	35	17
⋮	⋮	2A	1D	36	17
1F	10	2B	1D	37	12
20	10	2C	1D	38	10
21	01	2D	0B	39	1F
22	03	2E	0C	3A	1E
23	03	2F	0E	3B	1C
24	03	30	10	3C	1C
25	15	31	01	3D	09
26	14	32	02	3E	09
27	12	33	04	3F	0E
28	10	34	04		

Table 7: EPROM Contents, 24 and 26 AWG Solution

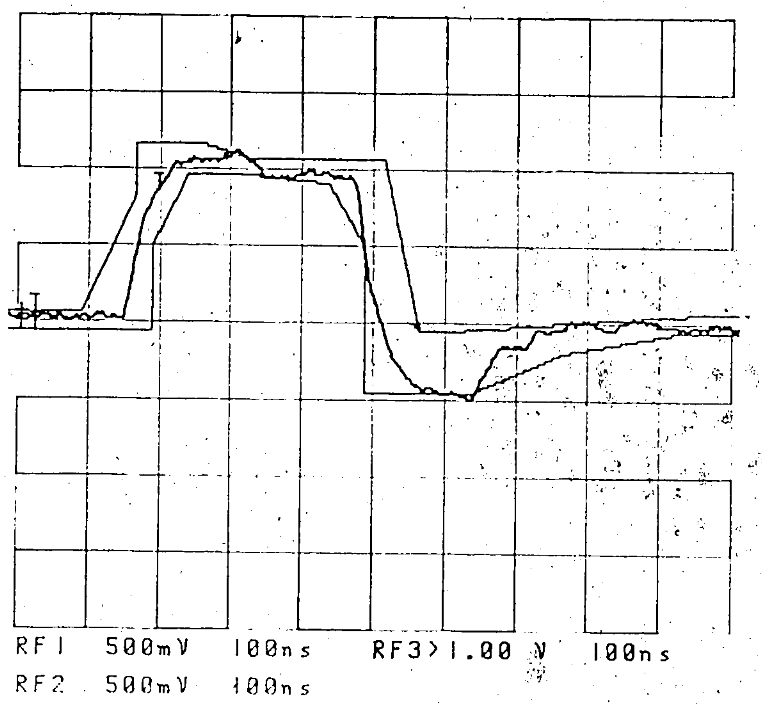


Figure 25: Long Line Pulse Shape, 99 m of 24 AWG cable

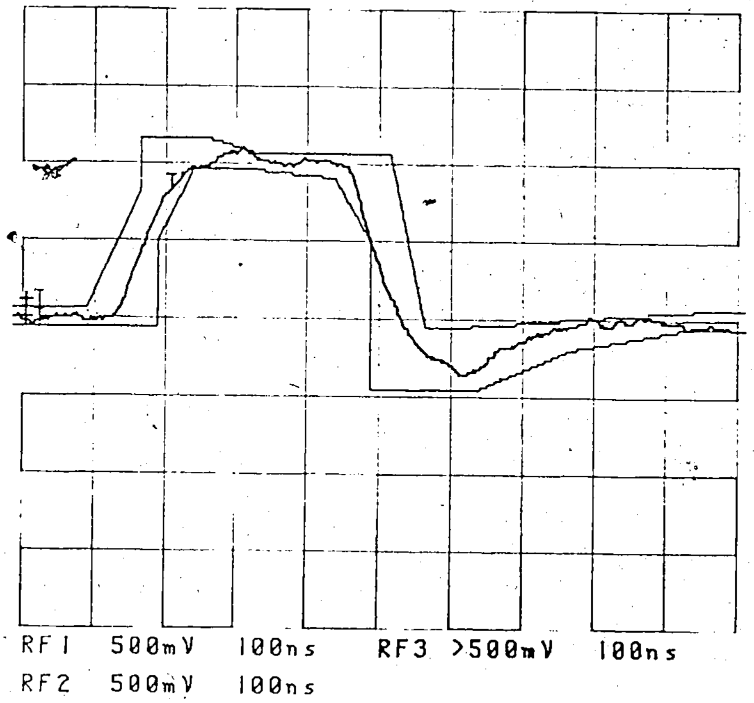


Figure 26: Long Line Pulse Shape, 198 m of 24 AWG cable

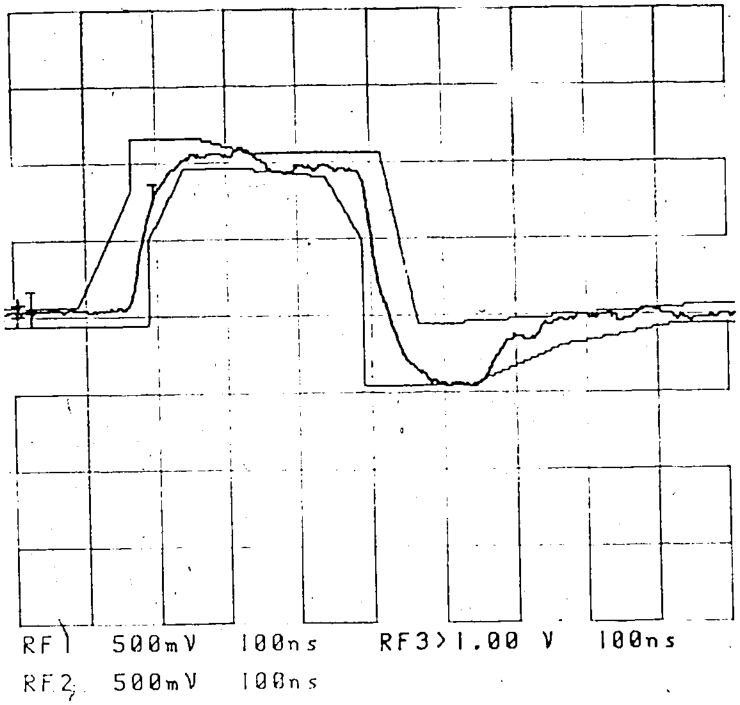


Figure 27: Long Line Pulse Shape, 82.5 m of 26 AWG cable

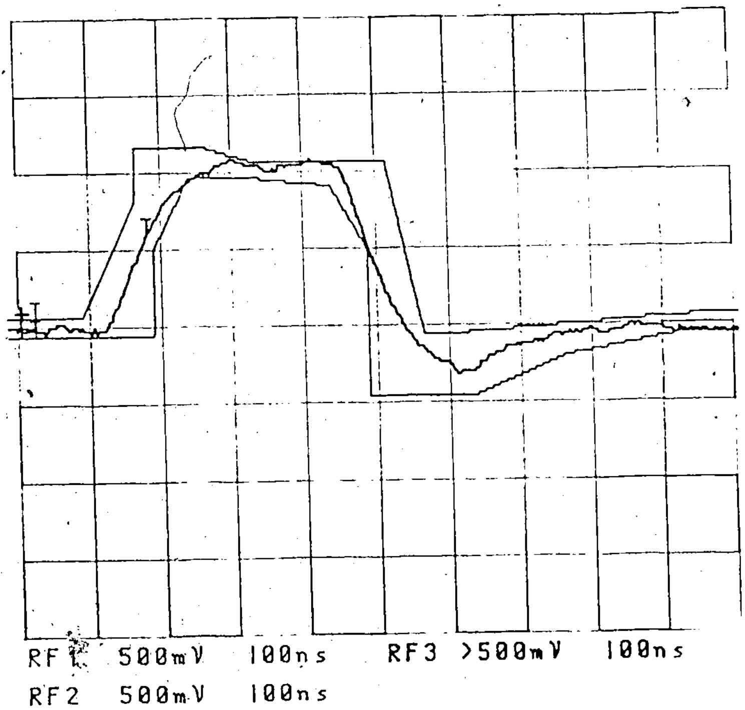


Figure 28: Long Line Pulse Shape, 198 m of 26 AWG cable

Several hurdles must be overcome before 24 and 26 AWG cable could be used in a DSX-1 environment. Although the DSX-1 standard can be met at the cross-connect with 24 or 26 AWG cables, the receivers in use today expect a signal to travel from the cross-connect to the receiver over 22 AWG cable. The higher attenuation and pulse distortion caused by the higher gauge cables may deteriorate a pulse travelling from the cross-connect sufficiently to cause the receiver to malfunction. The design of new receiver circuits would have to take into account the possibility of transmission over cable gauges other than 22 AWG. One can envisage the future network which has LBOs that have 2 modes of operation: single shape for 22 AWG cable, and a two-stage mode for other cable gauges (24 and 26 AWG). With new receiver circuits designed to accommodate various cable gauges, a mixture of cable gauges could be in use in the network, but the LBO would have to be set properly to ensure that the standard is met at the cross-connect. Also, if the cable length requirement of 200 m was relaxed, the LBO could have one setting that would work for all three gauges, but at limited lengths: up to 198 m for 22 AWG, up to 181.5 m for 24 AWG, and up to 132 m for 26 AWG. The conclusion is that 24 and 26 AWG cables can be accommodated by the digital LBO design technique with little difficulty. However, some changes to the DSX-1 requirements concerning cable lengths are needed before the use of the higher gauge cables can become commonplace.

5.2 DSX-1 Test Set Design

The design of the DSX-1 test set is the second application of the digital pulse generation technique examined. Many manufacturers produce test sets which generate a DSX-1 compatible output, plus other higher rate signals. However, these test sets only generate one DSX-1 pulse shape which falls roughly in the centre of the DSX-1 pulse mask, with an amplitude of 3.0 V. The test set design proposed in this section provides several extreme DSX-1 pulse shapes which would test various possibilities which could occur in the field. This enables much more rigorous testing of DSX-1 receiver circuits both at the factory when they are manufactured, and in the field. The test set would emulate signals as they appear at a DSX-1 cross-connect panel. This section discusses the desirable features of a DSX-1 test set, and presents a block diagram of a proposed design. Next, key parameters of amplitude and time quantization resolution are discussed. The number of segments per DS-1 symbol interval and the number of bits of resolution for each segment are different than that used for the LBO. Finally, a small set of test pulse shapes are proposed and illustrated using one possible combination of number of segments and bits per segment.

The fact that pulse shapes are stored in digital memory elements allows advanced features to be incorporated in the test set. Of course, a core EPROM chip is required to store a small set of commonly used pulse shapes. User defined pulse shapes would be desirable if a special test pulse was required. This could be achieved by using auxiliary RAM memory and allowing sample values to be entered into the RAM from a keyboard on the front panel of the test set. An advanced feature that adds to the flexibility of the test set is to have plug-in ROM cartridges which contain additional pulse shapes. A ROM cartridge could contain pulse shapes compatible with the old DSX-1 standard, or could accommodate future changes to the standard. Also, ROM cartridges could emulate pulses as they would appear after travelling over various lengths of cable. The amplitude of the output pulses should be adjustable by a front panel control knob within a 2.4 V to 3.6 V range, to produce pulses at the extremes of the amplitude specification. All functions should be accessed by keys on the front panel. As with today's test sets, a receiver would be included which could do error monitoring on incoming lines, and the test set would be capable of operating at other transmission rates, using analog pulse shaping techniques. This section only discusses the part of the test set which generates DSX-1 test signals.

Figure 29 shows a block diagram of the proposed test set. The keyboard decoding circuit determines the function selected by the front panel switches, and generates the appropriate control signals used by the rest of the circuitry. The user can load pulse shapes into the RAM by entering samples via the keyboard, or through an IEEE-488 bus. The source of the samples passed to the D/A can be one of three memories: the RAM, a plug-in ROM module, or a permanent EPROM chip. The decoding circuit controls a multiplexer which routes the outputs of the appropriate memory device to the D/A inputs. A crystal controlled PLL provides a very accurate sampling clock. This clock must be accurate enough to produce a DS-1 symbol rate of 1.544 Msymbols/sec.

± 30 parts per million. The D/A output is amplified by a driver circuit which in turn is connected via a coupling transformer to a front panel socket. The amplitude of the driver output is controllable within the 2.4 V to 3.6 V range by a front panel potentiometer.

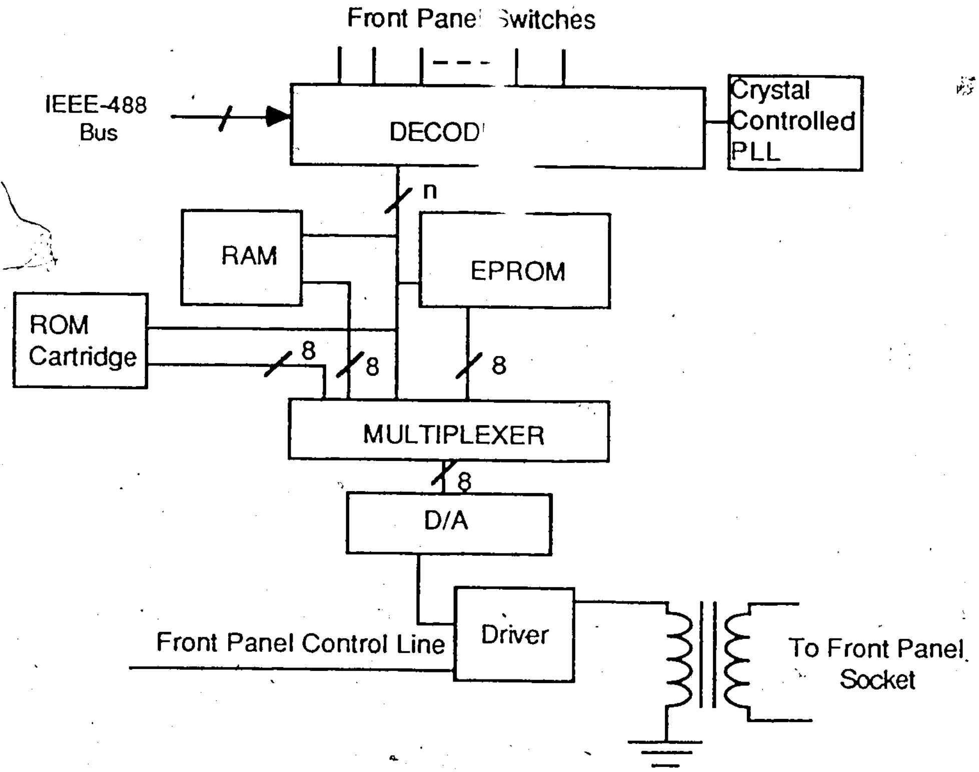


Figure 29: DSX-1 Test Set Block Diagram

The quality of the pulse shapes generated by the test set should be high. It is desirable to have an accurate representation of the pulse shapes, since the test set would be relied on for testing other equipment. The accuracy of the pulse shapes increases with increased amplitude and time base quantization resolution. The desire is to maximize the number of bits per sample (increase amplitude resolution) and the number of segments per DS-1 symbol interval (increase time resolution), within reasonable technological and cost constraints. The cost of the test set need not be minimized, as in the case of the LBO which is a high volume, low cost application. The test set is a low volume, specialized piece of equipment with advanced features, and therefore can demand a higher price for higher quality.

Current memory and D/A technology favors 8 bit data word widths. Most EPROMs are organized as banks of 8 bit words, although 1 and 4 bit word widths are also common. Word widths greater than 8 bits are not available for standard memory components. D/A technology achieves throughput rates of 200 Msamples/sec. at 8 bits/sample resolution. More than 8 bits of resoiution is deemed not necessary for this application, since the 256 quantization levels prodoced with 8 bits of resolution are the maximum used by commonly available test equipement, such as the Tektronics 2430 A digital oscilloscope. Using a resolution higher than 8 bits on the test set when oscilloscopes used to view the signal have only 8 bits of resolution would be pointless. Therefore, it is proposed that a resolution of 8 bits per sample be used for the DSX-1 test set.

A time resolution of 8 segments per DS-1 symbol interval is insufficient to describe complex pulse shapes accurately. The time resolution should be maximized, within technological constraints.. The higher time resolution requires that the memory and D/A operate at higher clock rates, but the frequency of operation of these devices is limited. The fastest EPROMs have cycle times of 15 ns (eg., the TBP38R85-15 by Texas Instruments) which results in a maximum frequency of operation of 66.7 MHz. The fastest ECL RAMs have acess times of 10 ns (eg., MBM10474-10 by Fujitsu), and can operate at 100 MHz. If 32 segments/DS-1 symbol interval are used, the resulting sample rate is 49.408 Msamples/sec. and standard EPROMs can be used. The maximum D/A operation frequency is in the 200 MHz region (eg.,HDAC 97000 by Honeywell), which means that a 49.408 MHz EPROM output can be connected directly to the D/A. With 128 segments/DS-1 symbol interval, the sample rate is 197.632 Msamples/sec., which can be supported by the D/A, but exceeds the maximum frequency of operation of the memory elements. However, several EPROMs can be operated at lower rates, and the outputs time multiplexed up to the 197.632 MHz rate.

Figure 30 shows the multiplexed memory and D/A arrangement proposed for sample rates over approximately 50 Msamples/sec.. Addresses are applied to the four EPROMs at 49.408 MHz. Each EPROM stores a different sample at the current address, with the leftmost EPROM containing the earliest occuring sample, the second from the left containing the subsequent sample, etc.. The control lines for the multiplexer are sequenced to select the outputs of each

EPROM for one quarter of the 49.408 MHz period, as shown in the timing diagram. The multiplexer output produces samples at 197.632 M amples/sec., which are applied directly to the D/A. Of course, an external ROM cartridge would be limited in speed below that of a ciruit board mounted EPROM, and memory multiplexing would have to be used to stay within the bounds of ROM cartridge technology. Or, alternately, the ROM cartridge contents could be transferred to the RAM at low speed, and subsequent operations would access the RAM to use the cartridge's pulse shapes.

The suggested test pulse shapes were drawn using 256 amplitude levels and 32 segments/DS-1 symbol interval. A higher number of segments would produce smoother, more accurate pulse shapes. Each pulse shape therefore requires 64 memory locations: 32 for the positive pulse, and 32 for the negative pulse. Five unique test shapes are suggested, which would occupy 320 memory locations in total. A typical high-speed EPROM is 2048 bytes large, and could store the positive and negative versions of 32 different pulse shapes. However, 5 extreme pulse shapes were deemed sufficient to vigorously test a DSX-1 receiver.

One of the 5 recommended test pulse shapes should pass through the centre of the mask, with a 3.0 V amplitude. This signal is shown in Figure 31, and would be used as general purpose DSX-1 test signal. The other 4 test signals represent extreme cases of pulses just fitting the DSX-1 pulse mask. Figures 32 and 33 show the two obvious extremes which are equal to the minimum and maximum curves of the pulse mask, respectively. Another extreme is the narrowest pulse possible with large overshoot and undershoot, as shown in Figure 34. Such a pulse shape could typically be encountered if the cross-connect is extremely close to the signal source. Figure 35 shows the opposite: a very wide pulse with no overshoot or undershoot. This extreme occurs when the cross-connect is very far from the transmitter. The amplitude of all pulses is fully adjustable within a 2.4 to 3.6 V range by a control knob. Many other test signals could be devised which are variations of those shown here. The test engineer has the flexibility to enter custom designed pulse shapes if those supplied with the test set or in the ROM cartridges are insufficient. This concludes the examination of alternate applications of th digital pulse generation technique.

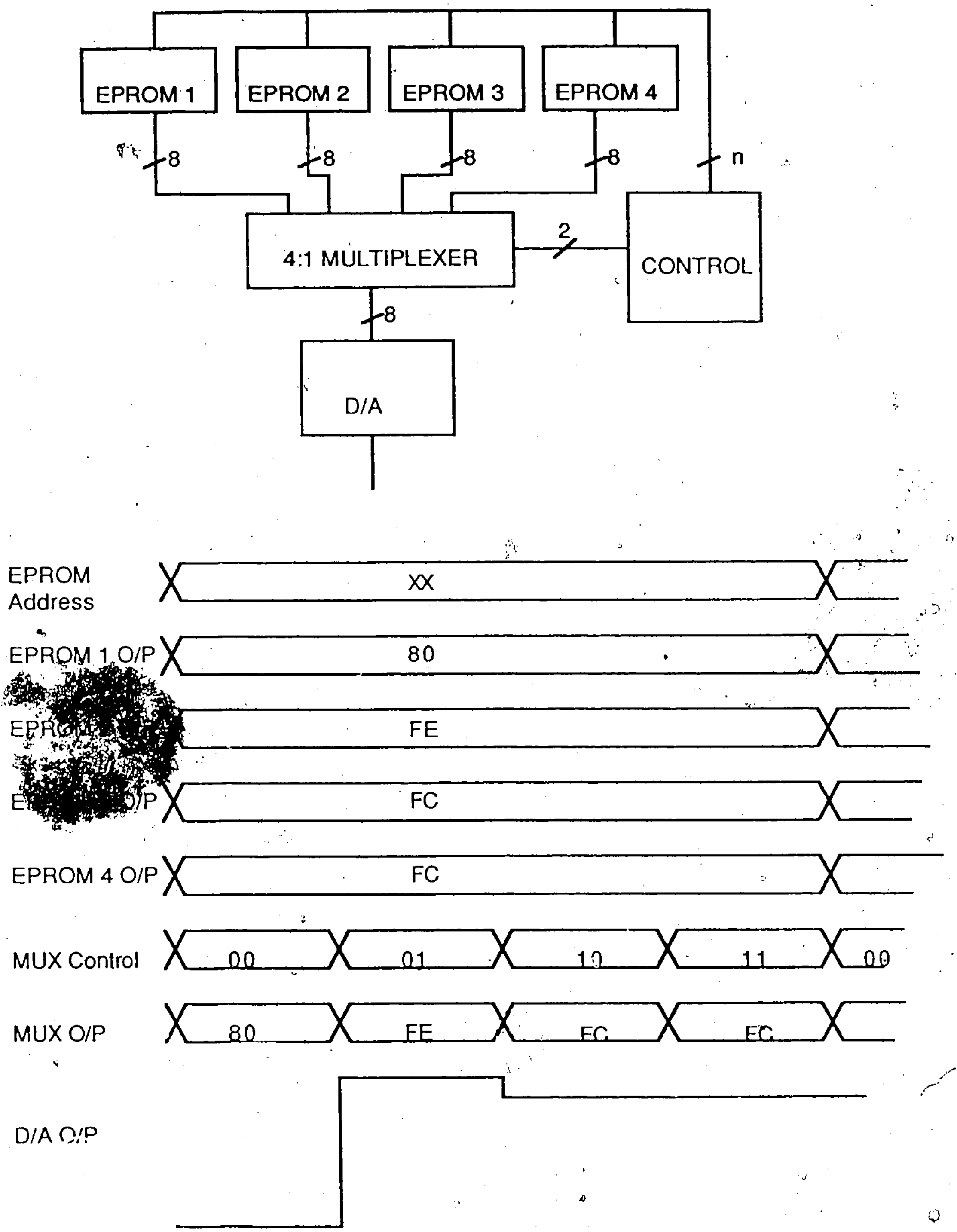


Figure 30: Multiplexed Memory Arrangement for DSX-1 Test Set

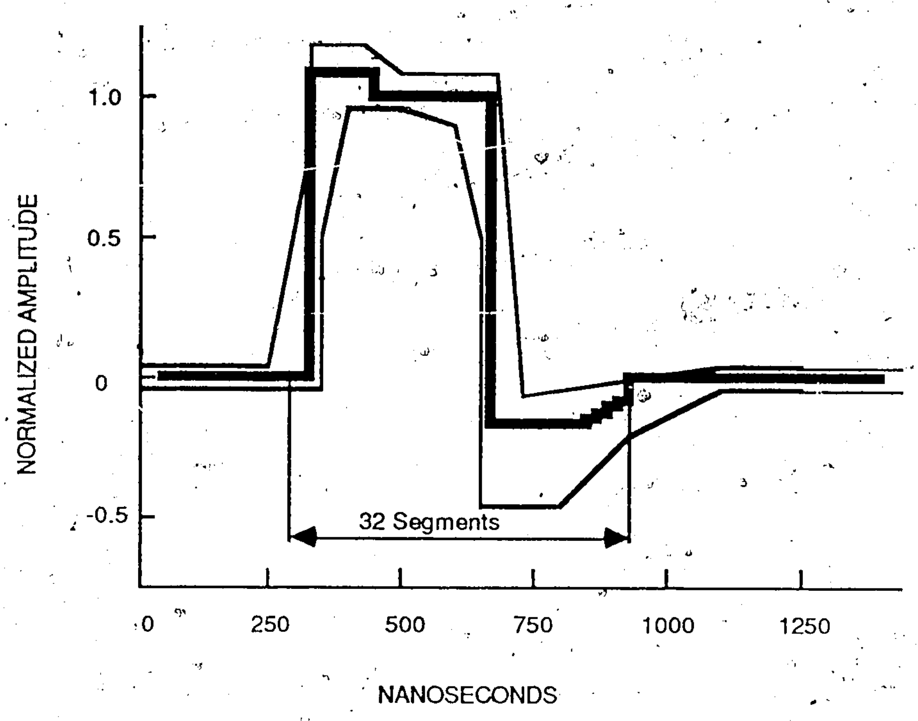


Figure 31: General Purpose Pulse Shape

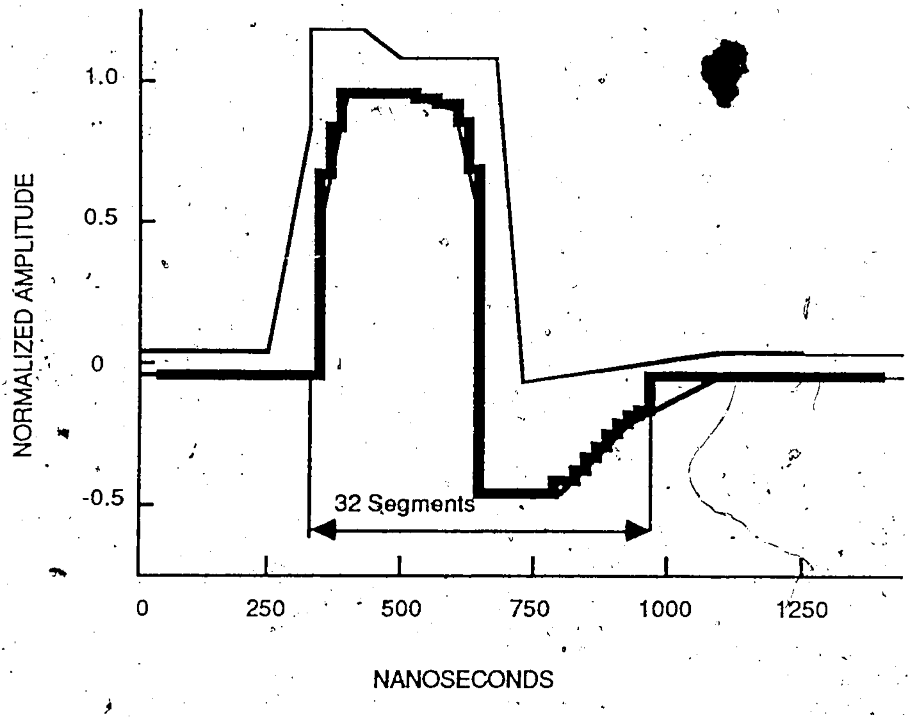


Figure 32: Minimum Pulse Shape

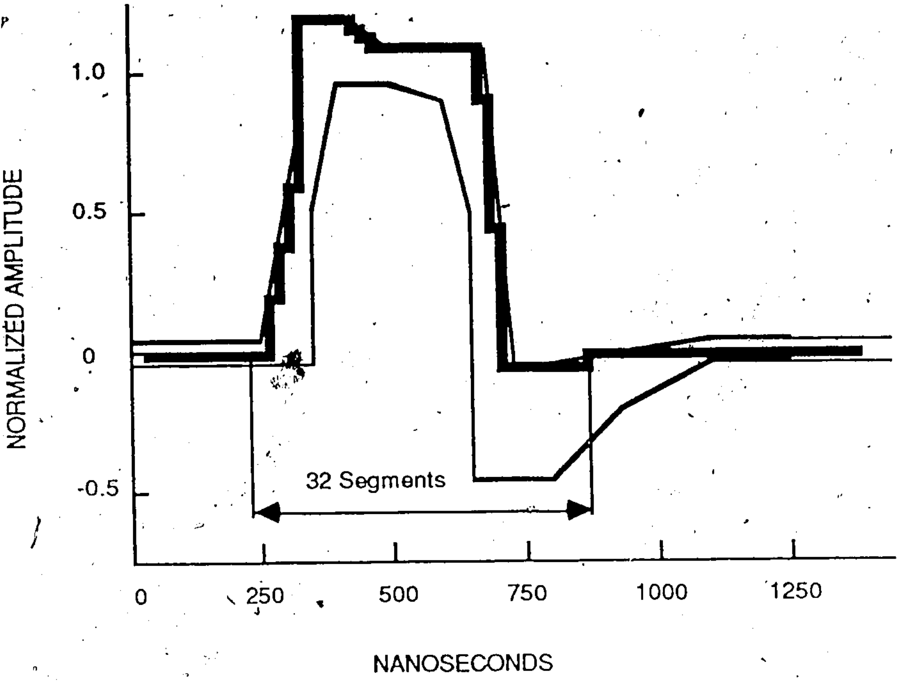


Figure 33: Maximum Pulse Shape

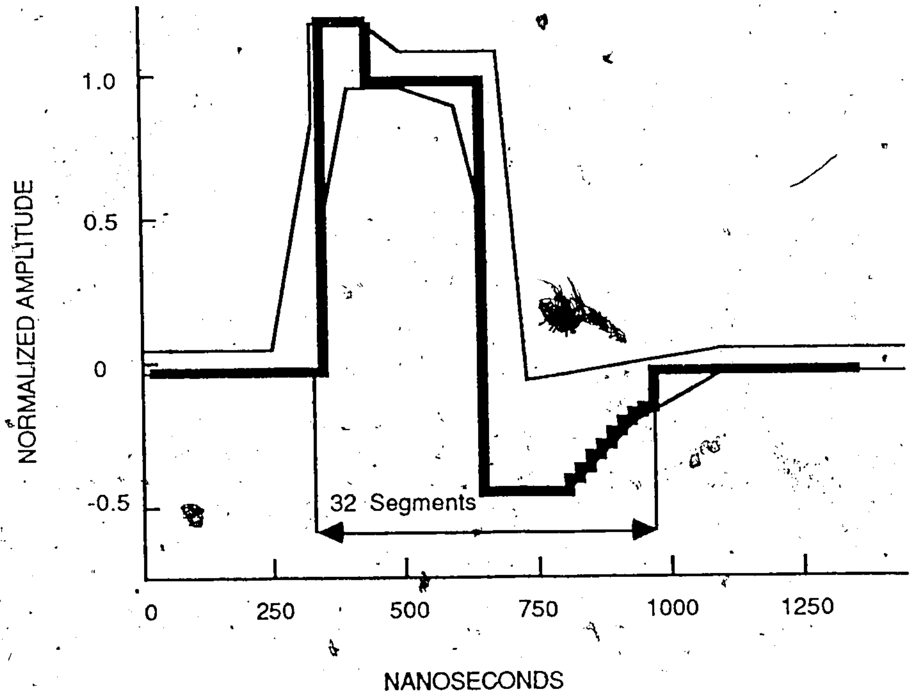


Figure 34: Narrow Pulse Shape

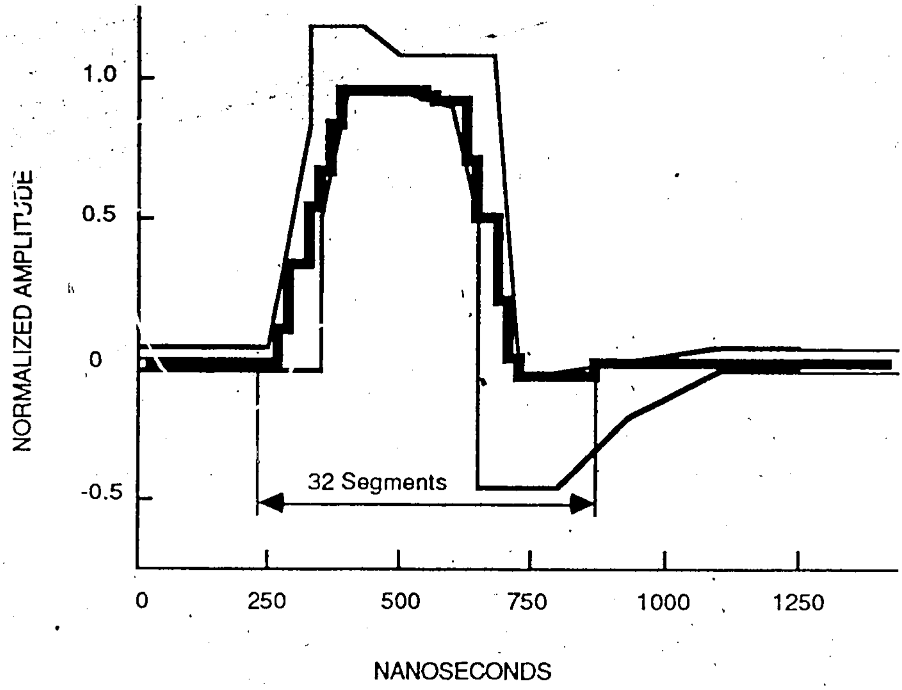


Figure 35: Spread out Pulse Shape

CHAPTER 6

CONCLUSIONS

The primary accomplishment of the research presented in this thesis is that a single pulse shape has been designed which will meet the DSX-1 standard when transmitted over 0 m to 200 m of 22 AWG cable. Since only one LBO setting is necessary, the need for manual switches or complex automatic control has been completely eliminated, and 1:N protection has been simplified. The pulses are generated at a rate of 8 samples/DS-1 symbol interval with 5 bits of resolution for each sample. Since the highest frequency of operation of any component in the LBO circuit is 12.352 MHz, the entire LBO may be integrated using low-cost TTL or CMOS technology. Furthermore, since the single shape solution utilizes only 10 unique output levels of the D/A, a full-function, 32 level D/A is not necessary. A custom D/A generating only the 10 levels needed would reduce the circuit size to the point where multiple LBO circuits could be placed on a single IC.

Two possible methods can extend the operation of the LBO to 24 and 26 AWG cables to realize the space savings provided by the smaller wires. One scenario involves a two stage solution which reintroduces manual control, but would allow the use of 24 and 26 AWG cables over the full 200 m range of line lengths. The second scenario involves limiting the maximum allowed cable length between the LBO and the cross-connect: 132 m for 26 AWG cable, and 181.5 m for 24 AWG cable. In this scenario the single shape solution would be valid for all three cable gauges. However, the use of 24 and 26 AWG cables will not become prevalent until DSX-1 receivers are altered, and the industry leaders agree to redefine the standard to include the use of all three cable gauges.

The pulse generation technique used in the LBO is not limited to this one application. A sophisticated DSX-1 test set is only one application of many which can be envisaged.

REFERENCES

1 Telecom Canada, Digital Network Notes, (1983), p. 5-9.

2 ibid, pp. 5-9 to 5-15.

3 ibid, p. 5-21.

4 American Telephone and Telegraph Company, Compatibility Bulletin No. 119: Interconnection Specifications for Digtal Cross-Connects, (Issue 3, October, 1979).

5 ibid, pp. 10,23.

6 International Computer Software, Inc., Turbo HALO Graphics Toolkit, Software User Manual, (1986).

7 Borland International Inc., Turbo Pascal Numerical Methods Toolbox, Software User Manual, (1986).

8 Bell Laboratories, Engineering Notes: Provision of Metallic Pairs to Other Common Carriers for Digital Transmission, Appendix I, pp. 3, 9, 15, (September, 1974).

9 William D. Stanley, Electronic Communications Systems, Reston Publishing Company, Inc., 1982, Reston, Virginia.

APPENDIX A

TRANSMISSION CABLE MODELS

The line model used in this thesis was developed at the Alberta Telecommunications Research Centre before the LBO project began. This appendix outlines the methods used to obtain models for 22, 24 and 26 AWG cables. Measured values for the RLGC parameters of the three line gauges were available for frequencies between 1 Hz and 5 MHz, inclusive. The model used in the simulation program for the LBO project requires that the transfer function of the line be evaluated at 193 kHz intervals, from 193 kHz to 9.843 MHz. The RLGC parameters at the 193 kHz intervals are evaluated using interpolation and extrapolation from the measured values. Transmission line parameters Z_0 and γ are evaluated from the RLGC parameters, and the transfer function can be evaluated using Z_0 and γ. The first section of this chapter describes the interpolation and extrapolation procedure used to calculate the RLGC parameters. The second section gives the derivation of the transfer function of the cable.

A.1 Calculation of Z_0 and γ

The cubic spline method was used to calculate the R, L, and G parameters at 193 kHz intervals from the measured values.[1] A cubic spline requires two sets of values. The first set consists of the tabulated function, in this case the R, L and G parameters measured at different frequencies. The second set is the tabulated second derivative of the R, L, and G parameters evaluated at each of the frequencies for which a measured value was available. Measured values for the RLGC parameters of 22, 24, and 26 AWG plastic insulated cable were available over a frequency range of 1 Hz to 5 MHz. The distributed capacitance of the line, C, was constant for all three gauges with a value of 0.052 µF/km. Standard cubic spline formulas were used to calculate the second derivatives of the R, L and G parameters at each individual frequency where a measurement had been made. In the remainder of this section, the first and second derivatives of some general function y are denoted as y' and y" respectively, and N is the number of tabulated values of y available. The second derivatives are defined by a set of N-2 linear equations in the N unknowns y_i", i=1...,N. A unique solution is obtained by specifying the values of the second derivatives at the boundaries, namely y_1'' and y_N''. One or both of the boundary conditions can be set to zero for the so-called natural cubic spline. Alternately, these boundary values can be calculated from specified values of the first derivatives at the endpoints, namely y_1' and y_N'.

The first derivatives of R, L, and G at the endpoint frequencies were estimated from a plot of the measured values for each parameter. The second derivatives are calculated from the first derivatives, and have little effect on the outcome of the interpolated values spaced 193 kHz apart, under 5 MHz. The second derivatives could be set to zero at the endpoints, and the interpolation would still give acceptable values. However, the values of the second derivative at the 5 MHz endpoint strongly affect the extrapolated values above 5 MHz. Therefore, the interpolation and extrapolation process was iterative. The first derivatives at 1 Hz and 5 MHz were estimated from plots of the measured R, L, and G parameters. The interpolation and extrapolation calculations were performed, and the resulting values for R, L, and G were plotted. If the extrapolated values over 5 MHz appeared erratic, and did not follow the trend of the measured values, the first derivative estimate at 5 MHz was altered. This process was repeated until the extrapolated values exhibited a satisfactory behaviour. Table 1 shows the resulting first derivative values used in the cubic spline calculations. The units for each of the parameters are as follows: Ω/km/kHz for R', mH/km/kHz for L', and µS/km/kHz for G'.

	VALUE AT 1 Hz			VALUE AT 5 MHz		
CABLE GAUGE	R'	L'	G'	R'	L'	G'
22 AWG	0.357902	-6.2 E -5	0.025	0.0734	-2.02 E -6	0.02564
24 AWG	0.16	-3.1 E -4	0.016	0.093	-2.45 E -6	0.0208
26 AWG	0.0648	-2.2 E -4	0.010	0.1166	-3.01 E -6	0.0208

Table A1: First derivatives of R, L, and G

A Fortran program calculates second derivatives at each of the 37 values measured at frequencies from 1 Hz to 5 MHz, for the R, L, and G parameters of the 3 different cables. The cubic spline interpolation formula shown below was used to calculate the R, L, and G parameters at 193 kHz intervals of "f", starting at f = 193 kHz and going up to f = 9.843 MHz.

$$y = Ay_j + By_{j+1} + Cy''_j + Dy''_{j+1}$$

where:

$$A = \frac{f_{j+1} - f}{f_{j+1} - f_j}, \qquad B = 1 - A$$

$$C = \frac{1}{6}(A^3 - A)(f_{j+1} - f_j)^2, \qquad D = \frac{1}{6}(B^3 - B)(f_{j+1} - f_j)^2$$

The y and y" values are the tabulated function and second derivative, respectively, of the R, L, or G parameters. The value y is the interpolated (or extrapolated) value at the frequency, f.

For frequencies between 1 Hz and 5 MHz, the tabulated values at f_{j+1} and f_j, which straddle the frequency of interest, f, are used to calculate a value for R, L, or G at that frequency. For frequencies beyond 5 MHz, the same formula is used, but the two tabulated values occuring at the two highest frequencies, in this case 3 MHz and 5 MHz, were used to calculate R, L, and G for each frequency over 5 MHz. This is why the value of the second derivative at 5 MHz strongly affected the outcome of the extrapolation.

The R, L, and G parameters calculated at 193 kHz intervals were used, along with the constant C, to calculate the transmission line parameters Z_0 and γ using standard transmission line equations. The resulting values for Z_0 and γ for 22, 24, and 26 AWG cables are shown in Tables A2, A3, and A4, respectively. The transfer function for a length of cable is completely specified if Z_0, γ, and the cable length are known. Therefore, Tables A2, A3, and A4 represent the cable models for the 3 cables.

Freq (kHz)	Z_0 (Real)	Z_0 (Imaginary)	γ (Real)	γ (Imaginary)
193	104.476000	-13.014270	1.310979	10.515410
386	101.458700	-9.162408	1.846422	20.423530
579	99.819950	-7.480834	2.261754	30.140520
772	98.840080	-6.485331	2.614773	39.792900
965	98.141460	-5.804711	2.925843	49.389590
1158	97.615020	-5.302547	3.207660	58.949620
1351	97.206770	-4.911842	3.466910	68.486950
1544	96.878300	-4.596340	3.708054	78.006340
1737	96.600480	-4.335274	3.934991	87.505500
1930	96.362340	-4.114282	4.149703	96.988660
2123	96.155490	-3.924053	4.353982	106.458500
2316	95.974130	-3.758007	4.549172	115.917500
2509	95.814170	-3.611343	4.736291	125.368100
2702	95.671650	-3.480660	4.916402	134.811000
2895	95.542680	-3.363493	5.090607	144.245500
3088	95.423180	-3.258007	5.260032	153.669600
3281	95.311230	-3.162488	5.425274	163.082400
3474	95.206640	-3.075264	5.586312	172.486000
3667	95.109440	-2.994998	5.743093	181.882700
3860	95.019660	-2.920619	5.895565	191.274700
4053	94.937320	-2.851261	6.043676	200.664400
4246	94.862460	-2.786218	6.187378	210.054100
4439	94.795120	-2.724906	6.326622	219.446200
4632	94.735350	-2.666838	6.461360	228.843000
4825	94.683210	-2.611608	6.591547	238.246900
5018	94.638750	-2.[illegible]69	6.717140	247.660400
5211	94.602010	-2.[illegible]32	6.838097	257.086000
5404	94.573040	-2.459744	6.954379	266.526100
5597	94.551900	-2.412892	7.065946	275.983200
5790	94.538640	-2.367590	7.172764	285.459800
5983	94.533300	-2.323676	7.274799	294.958500
6176	94.535930	-2.281012	7.372021	304.481800
6369	94.546590	-2.239474	7.464399	314.032200
6562	94.565310	-2.198954	7.551908	323.612500
6755	94.592150	-2.159360	7.634523	333.225000
6948	94.627130	-2.120608	7.712224	342.872500
7141	94.670300	-2.082623	7.784989	352.557500
7334	94.721720	-2.045342	7.852804	362.282800
7527	94.781390	-2.008706	7.915656	372.050800
7720	94.849370	-1.972664	7.973532	381.864300
7913	94.925700	-1.937171	8.026425	391.725800
8106	95.010380	-1.902186	8.074330	401.638100
8299	95.103450	-1.867673	8.117244	411.603800
8492	95.204960	-1.833599	8.155165	421.625500
8685	95.314900	-1.799935	8.188098	431.705800
8878	95.433300	-1.766657	8.216049	441.847500
9071	95.560200	-1.733741	8.239024	452.053200
9264	95.695590	-1.701167	8.257037	462.325500
9457	95.839510	-1.668916	8.270102	472.667000
9650	95.991960	-1.636972	8.278233	483.080500
9843	96.152940	-1.605321	8.281452	493.568500

Table A2: 22 AWG Cable Model

Freq (kHz)	Z_0 (Real)	Z_0 (Imaginary)	γ (Real)	γ (Imaginary)
193	106.105600	-17.061110	1.718394	10.679370
386	103.247600	-11.578870	2.332949	20.783580
579	101.478400	-9.449407	2.856214	30.641220
772	100.274900	-8.185961	3.299430	40.370510
965	99.452830	-7.325326	3.691000	50.049480
1158	98.834700	-6.690816	4.0[illegible]59	59.686120
1351	98.342680	-6.197187	4.372229	69.287200
1544	97.942850	-5.798717	4.675838	78.863460
1737	97.608280	-5.469121	4.961613	88.418360
1930	97.323930	-5.190176	5.232009	97.956440
2123	97.076910	-4.950112	5.489291	107.478600
2316	96.859250	-4.740612	5.735155	116.986500
2509	96.666440	-4.555592	5.970869	126.483200
2702	96.494090	-4.390749	6.197761	135.969800
2895	96.337890	-4.242963	6.417216	145.446100
3088	96.193620	-4.109917	6.630651	154.910200
3281	96.058920	-3.989447	6.838817	164.361700
3474	95.933540	-3.879440	7.041680	173.802900
3667	95.817340	-3.778206	7.239172	183.236400
3860	95.710270	-3.684393	7.431220	192.664900
4053	95.612190	-3.596910	7.617759	202.090900
4246	95.523070	-3.514865	7.798725	211.516900
4439	95.442830	-3.437520	7.974054	220.945600
4632	95.371400	-3.364264	8.143690	230.379300
4825	95.308740	-3.294584	8.307576	239.820800
5018	95.254770	-3.228046	8.465662	249.272400
5211	95.209460	-3.164284	8.617896	258.736700
5404	95.172740	-3.102985	8.764235	268.216100
5597	95.144570	-3.043877	8.904636	277.713000
5790	95.124880	-2.986730	9.039062	287.229900
5983	95.113620	-2.931342	9.167476	296.769200
6176	95.110750	-2.877538	9.289847	306.333100
6369	95.116200	-2.825165	9.406148	315.924100
6562	95.129890	-2.774091	9.516355	325.544500
6755	95.151800	-2.724197	9.620446	335.196500
6948	95.181840	-2.675381	9.718405	344.882400
7141	95.219960	-2.627551	9.810219	354.604400
7334	95.266100	-2.580626	9.895878	364.364800
7527	95.320180	-2.534536	9.975374	374.165700
7720	95.382140	-2.489216	10.048710	384.009100
7913	95.451930	-2.444610	10.115870	393.897300
8106	95.529450	-2.400669	10.176880	403.832300
8299	95.614640	-2.357346	10.231740	413.816100
8492	95.707450	-2.314603	10.280450	423.850700
8685	95.807770	-2.272403	10.323030	433.938100
8878	95.915530	-2.230714	10.359500	444.080200
9071	96.030690	-2.189508	10.389880	454.278800
9264	96.153140	-2.148759	10.414190	464.535900
9457	96.282800	-2.108444	10.432460	474.853200
9650	96.419610	-2.068544	10.444710	485.232600
9843	96.563460	-2.029038	10.450980	[illegible]

Table A3: 24 AWG Cable Model

Freq (kHz)	Z_0 (Real)	Z_0 (Imaginary)	γ (Real)	γ (Imaginary)
193	107.989400	-23.408140	2.357246	10.868910
386	104.909500	-14.923630	3.006284	21.118060
579	103.262000	-11.954880	3.612797	31.179720
772	102.025100	-10.344220	4.168413	41.075080
965	101.049600	-9.249505	4.659420	50.852950
1158	100.296400	-8.444636	5.105072	60.568760
1351	99.714830	-7.820319	5.515894	70.253850
1544	99.246640	-7.316729	5.898235	79.913170
1737	98.848760	-6.900083	6.257948	89.541950
1930	98.506100	-6.547478	6.598237	99.146180
2123	98.207790	-6.244126	6.922065	108.730600
2316	97.945970	-5.979494	7.231587	118.299000
2509	97.714810	-5.745852	7.528378	127.854800
2702	97.508670	-5.537738	7.814102	137.399300
2895	97.322010	-5.351210	8.090513	146.931800
3088	97.149380	-5.183342	8.359420	156.449300
3281	96.987790	-5.031399	8.621777	165.950900
3474	96.836940	-4.892691	8.877519	175.439400
3667	96.696750	-4.765067	9.126530	184.918000
3860	96.567140	-4.646806	9.368693	194.389600
4053	96.448070	-4.536517	9.603897	203.857500
4246	96.339530	-4.433063	9.832031	213.324600
4439	96.241490	-4.335505	10.052990	222.794300
4632	96.153950	-4.243062	10.266660	232.269500
4825	96.076930	-4.155082	10.472950	241.753600
5018	96.010400	-4.071011	10.671770	251.249700
5211	95.954400	-3.990383	10.863010	260.761000
5404	95.908920	-3.912795	11.046600	270.290700
5597	95.873990	-3.837906	11.222450	279.841900
5790	95.849590	-3.765419	11.390480	289.418000
5983	95.835760	-3.695079	11.550620	299.022200
6176	95.832490	-3.626663	11.702800	308.657500
6369	95.839780	-3.559977	11.846970	318.327300
6562	95.857660	-3.494852	11.983060	328.034800
6755	95.886100	-3.431139	12.111020	337.783100
6948	95.925120	-3.368709	12.230810	347.575400
7141	95.974700	-3.307446	12.342390	357.414900
7334	96.034860	-3.247249	12.445730	367.304900
7527	96.105580	-3.188029	12.540790	377.248400
7720	96.186820	-3.129708	12.627560	387.248600
7913	96.278610	-3.072215	12.706030	397.308600
8106	96.380910	-3.015489	12.776180	407.431500
8299	96.493690	-2.959476	12.838000	417.620400
8492	96.616950	-2.904126	12.891500	427.878400
8685	96.750650	-2.849398	12.936700	438.208500
8878	96.894750	-2.795254	12.973600	448.613700
9071	97.049250	-2.741660	13.002220	459.097000
9264	97.214080	-2.688587	13.022580	469.661400
9457	97.389220	-2.636009	13.034730	480.309800
9650	97.574630	-2.583904	13.038680	491.045100
9843	97.770260	-2.532252	13.034500	501.870200

Table A4: 26 AWG Cable Model

A.2 Transfer Function Derivation

The Z_0 and γ values calculated in the previous section were used to evaluate the transfer function of a length of line over which a pulse was to be transmitted. Figure A1 shows a schematic of the line configuration which was modeled. The line input is simply the D/A output. A transfer function for the signal as it appears at the midpoint of a line of length L, terminated by a 100 ohm resistor was desired. Obtaining this transfer function was straightforward using ABCD matrix equations, as follows.

$$\begin{bmatrix} V_i \\ I_i \end{bmatrix} = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} V_x \\ I_x \end{bmatrix}$$

where $A = D = \cosh(\gamma L_1)$

$B = Z_0 \sinh(\gamma L_1)$

$$C = \frac{\sinh(\gamma L_1)}{Z_0}$$

and

$$Z_x = \frac{V_x}{I_x} = \frac{Z_0(Z_L + Z_0 \tanh(\gamma L_2))}{Z_L \tanh(\gamma L_2) + Z_0}$$

from which:

$$H(\gamma, L) = \frac{V_x}{V_i} = \frac{Z_L \cosh(\gamma L_2) + Z_0 \sinh(\gamma L_2)}{Z_L \cosh(\gamma L) + Z_0 \sinh(\gamma L)}$$

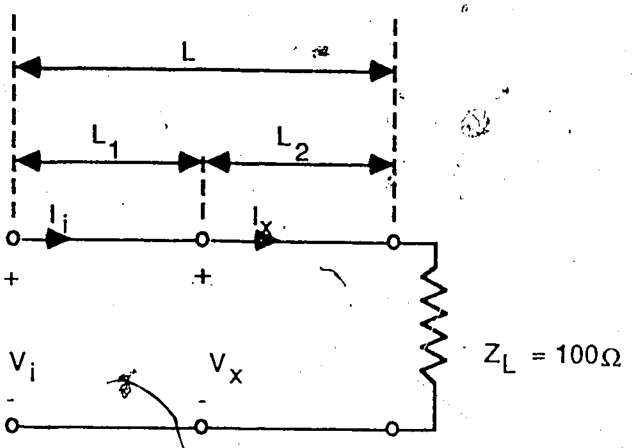


Figure A1: Cable Model Schematic

The transfer function is evaluated at 193 kHz intervals using the Z_0 and γ values given in Tables A2, A3, and A4. The Fourier coefficients of the frequency domain representation of V_i are multiplied by the transfer function to produce a frequency domain representation of the signal at the midpoint of the line. An inverse FFT of the resulting signal produces the time domain waveform as it appears at the midpoint of the line.

APPENDIX B

LBO CIRCUIT DIAGRAM AND TIMING DIAGRAMS

This Appendix contains a detailed LBO circuit schematic, and detailed timing diagrams of the circuit's operation. The circuit was built using wire-wrap techniques on a sheet of perforated fiberglass board. Each power supply was decoupled with a 4.7 mF capacitor at the board edge, then at every second IC with a 0.1 mF capacitor. This section describes the function of the various circuit components, using the timing diagrams to illustrate the relationships between the various digital signals occuring in the circuit.

B.1 DS-1 Rate Operation and Timing

Figure B1 shows the circuit schematic, and figure B2 shows the DS-1 rate timing of the circuit. The circuit schematic is found in a pocket attached to the inside back cover of the thesis. The DS-1, NRZ data and a 1.544 MHz, 50% duty cycle clock are supplied by a DS-1 digital test set. The two signals are connected to the circuit via DIP switches to avoid removing cables from the board during testing when one of the signals was not desired. The 100 kohm pulldown resistor and 200 ohm series resistor are recommended by the IC manufacturer for off-board connections when HCMOS devices are used to avoid damage which may be caused by static electricity. U_1 in Figure B1 is a monostable multivibrator (one-shot). The momentary switch can be used to generate an isolated pulse approximately 650 ns wide, and thus transmit a single DSX-1 pulse, provided the data line from the test set is disconnected. This feature was included early in the design when an unclear definition of the DSX-1 standard was available. The isolated pulses for testing were always generated with the pattern ...+10000000-10000000+1... supplied by the test set, according to DSX-1 specifications, and U_1 was not needed. U_2 is a serial shift register with parallel outputs, used to monitor the data stream for the occurence of 8 consecutive zeros. The gates attached to the U_2 outputs perform an 8 bit wide NOR function , with the output appearing at pin 10 of U_6. The signal goes high when 8 zeros have been shifted into U_2.

The timing diagram in Figure B2 shows two possibilities. Case 1 assumes a string of 8 consecutive '0's bounded by '1's. This is an extreme case where the B8ZS pattern must be switched in only once, and the normal data resumed immediately following the B8ZS pattern. Case 2 assumes a minimum of 16 consecutive '0's. In case 2, the timing of two consecutive cycles of the B8ZS pattern is examined. Timing for AMI encoding is not shown explicitly, however it is identical to the normal operation of the circuit outside of the intervals when the B8ZS pattern is being added. Each edge on the signals shown in Figures B2 and B3 has two possibilities: a minimum delay, and a maximum delay. The minimum and maximum delays for each device were obtained from the manufacturer's data sheets. The two possible delays are shown at each signal transition by numbers which indicate the delay in nanoseconds. The signal cannot be considered stable in the area between the minimum delay transition, and the maximum delay transition. The signal is only guaranteed to be stable in the unshaded areas on the timing diagrams.

U_{10} and one half of U_9, plus the surrounding logic gates, form the B8ZS pattern generation circuitry. The counter, U_{10}, will eventually settle to an all '0's output regardless of which state it was in when the circuit was powered-up. When 8 consecutive '0's are detected, the U_6, pin 10 output causes U_9, pin 9 to go high on the next clock pulse. Then, U_5, pin 8 goes high, and U_{10} counts up. When the terminal count of 111 is reached, the counter is synchronously reset to 000. At that time, if U_6, pin 10 has also gone low, the counter remains in the 000 state until another occurence of the 8 zeros is detected. However, if U_6, pin 10 is high, the counter repeats the 8 state count once again, which would be the situation if 16 consecutive zeros were present in the data stream. The counter cycles through the 8 states until it resets, and U_6, pin 10 is low at the same time. The outputs of the counter are decoded to produce the necessary B8ZS data signals.

U_5, pin 3 produces the pattern '00001001 as U_{10} counts up from 000 to 111. This pattern is routed via U_{11} to the polarity signal generator consisting of U_{13} and the AND gate from U_{12}. At U4, pin 8 the pattern '00011011' appears as the counter sequences through the 8 states. This is the B8ZS substitution pattern which replaces the 8 consecutive '0's in the data stream. When the B8ZS/AMI switch is open, U_{11}, pin 1 is low, which causes the data from U_9, pin 5 to be routed to both the polarity signal generator and to U_{14}, which retimes both signals with DS-1 clock. When the switch is closed, U_9, pin 9 is used to control the routing of the data to the polarity generator and to U_{14}. When U_9, pin 9 is high, the B8ZS patterns are substituted for the normal data by U_{11}. When U_9, pin 9 is low, the circuit operates normally, performing AMI encoding on the data until the next occurence of 8 consecutive '0's is detected.

The polarity generator consists of one D flip-flop and one AND gate. The DS-1 clock is inverted, and ANDed with the data. This produces a stream of pulses, one half of a DS-1 bit period wide, and occuring only when a '1' is present in the data stream. This signal is used as the clock input to a D flip-flop arranged as a ÷2 element. Thus, the D flip-flop output changes state each time a '1' is present in the data stream, as shown in Figure B2 for U_{13}, pin 5. The modified data and polarity signals are retimed with the DS-1 clock by U_{14}, so that the data symbol boundaries are aligned with the state changes on the polarity signal.

The Phase Locked Loop (PLL) is enclosed in a metal box to protect it from electromagnetic interference. The ÷8 element in the feedback path of the PLL, U_{19}, is located outside of the metal box. The outputs of U_{19} and U_{14} are latched by the 12.352 MHz clock into U_{15}. The 5 outputs of U_{15}, plus the pulse shape control switch are the address inputs to the EPROM, U_{16}. The EPROM outputs are retimed through U_{17} by a delayed version of the 12.352 MHz clock. The need for the delayed clock will be explained in the next section. The U_{17} outputs are converted to ECL logic levels using U_{23} and U_{24}, and are applied to the D/A converter, U_{25}. From U_{15} to the D/A, all devices operate at 12.352 MHz. The detailed high-speed timing is shown in Figure B3, and explained in the next section.

B.2 High Speed Timing

All the low speed timing shown in Figure B2 is referenced to the DS-1 clock supplied by the digital test set. For the high speed timing, the relationship between the DS-1 rate clock and the 12.352 MHz clock must be established. This relationship must be derived in an indirect manner.

When the PLL is locked, the signals at U_{21}, pins 1 and 3 will be exactly in phase, as shown in the top two line of Figure B3. This is the only known relationship between the DS-1 clock and the high speed clock. These two signals must be traced backwards through the circuit to find the relationship between the DS-1 clock and the 12.352 MHz clock. The clock signal at U_{21}, pin 1 is the inverted DS-1 clock. The actual DS-1 clock edge, at U_6, pin 9 can occur anywhere between 11ns and 24 ns before the signal at U_{21}, pin 1. Also, the signal at U_{21}, pin 3 is the output of the counter, U_{19}, which is clocked by the 12.352 MHz clock. Therefore, it can be deduced that the 12.352 MHz clock edge occurs anywhere between 17 ns and 43 ns before the signal at U_{21}, pin 3. This analysis gives a range of possibilities for the relationship between the high-speed clock and the DS-1 clock. The circuit had to be designed to work for any possible delays through U_6 and U_{19}. Four extreme possibilities were examined in Figure B3:

Case A: Maximum delay on the counter, U_{19}, and minimum delay on the inverter, U_6.

Case B: Minimum delay on the counter, maximum delay on the inverter.

Case C: Minimum delay on both devices.

Case D: Maximum delay on both devices.

Case A and B were the worst of the four possibilities. In Case A, the high speed clock edge leads the DS-1 clock edge by 32 ns. In case B, the DS-1 clock edge leads the high speed clock edge by 7 ns. The timing for both case A and case B was examined further.

The critical timing occurs at U_{15}: signals arriving from low speed circuitry, U_{14}, and from high speed circuitry, U_{19}, must be latched with the same high speed clock. The purpose of this exercise was to determine how badly the U_{14} and U_{19} outputs were misaligned with respect to each other. Figure B3 shows that the outputs of these two ICs could be skewed substantially, especially in the case A timing. Therefore, a latch, namely U_{15}, was added to retime the U_{14} and

U_{19} outputs before presenting the signals to the EPROM. In general, the 12.352 MHz clock could be used directly to latch the inputs to U_{15}. However, an additional 15 ns of delay could be added to the U_{15} clock without affecting performace of the circuit. The timing for this situation is shown at the bottom of Figure B3, but in the circuit, this extra delay was not necessary. However, the analysis showed that the clock used to latch the EPROM outputs must lag the U_{15} clock by about 30 ns to ensure that the EPROM outputs are stable when latched. This delay was achieved using U_{22}, which is an active delay line capable of delaying a signal up to 40 ns in 8 ns increments.

This concludes the description of the circuit timing. The ECL level converters and the D/A do not require a clock signal. Figure B3 shows a sample of what the D/A output would look like.

FIGURE B2 has been removed due to poor print quality

FIGURE B3 has been removed due to poor print quality.

APPENDIX C

COMPUTER FILE LISTINGS

This report contains listings of the files and the Turbo Pascal program used to simulate DS-1 pulse transmission over twisted pair transmission line for the DSX-1 LBO project studied at the ATRC. The report begins by describing the function of the files that must reside in the same directory as the simulation program. Following this, the program listing and the contents of those files required by the simulation are given.

C.1 File Description

The following files must reside in the same directory as the simulation program, otherwise the program will fail. The file name and a short description of the file's function is included.

File Name	Description
RealFFT.inc	Turbo Pascal Numerical Methods Toolbox routine. Performs FFT on a vector of Real points, as opposed to Complex points. Included in the simulation program at compilation.
CompFFT.inc	Turbo Pascal Numerical Methods Toolbox routine. Performs FFT on a vector of Complex points. Included in the simulation program at compilation.
FFTB2.inc	Turbo Pascal Numerical Methods Toolbox routine. Implements the FFT algorithm for the above two procedures. Also included in simulation program at compilation.
Haloturb.p	Turbo Halo routine. Links Turbo Pascal to the Halo graphics routines.

Haloturb.bin	Used by Haloturb.p
Haloibme.dev	Turbo Halo hardware device driver for the IBM EGA graphics card.
Halo106.fnt	Font used when plotting the screen.
Halo011.fnt	Font used for screen displays.
Halortp.exe	Turbo Halo resident driver.

Before running the simulation program, the user must type the command "halortp". This command will prepare the computer hardware to work when Turbo Halo graphics routines are called. If this command is not executed before the simulation program is invoked, the simulation will crash when it enters the graphics mode. The files listed above are provided by the manufacturers of the two commercial software packages used in conjunction with Turbo Pascal: the Turbo Numerical Toolbox and Turbo Halo. In addition to these files, the following files were created to provide input data for the simulation. All files listed below are pascal "text" files.

<u>File Name</u>	<u>Description</u>
Shortest.	Contains one number of type = real per line. Describes pulse shape that meets the DSX-1 pulse shape specification for short lengths of 22 gauge cable.
Longtest.	Same structure as "Shortest.", but describes a pulse shape that meets the DSX-1 pulse shape specification over long lengths of 22 gauge cable.
Pmask.dat	Contains two numbers of type = real per line, separated by at least one space. The first number is the X coordinate (time, in nanoseconds), and the second the Y coordinate (narmalized voltage) of a vertex of the DSX-1 pulse mask. The first 10 lines

are vertices of the upper boundary of the mask listed in order from minimum time to maximum time. The last 12 lines of the file are the vertices of the lower boundary listed in order from maximum time to minimum time.

G22T70.dat

G24T70.dat

G26T70.dat

These files contain the cable model parameters for 22, 24, and 26 AWG transmission lines. The files consist of 51 lines in total, with 4 numbers of type = real per line. The first two numbers are the real and imaginary parts of Z_0, and the last two values on each line are the real and imaginary parts of γ.

C.2 File Listings

This section contains the listing of the simulation program, plus a listing of the contents of the pulse shape, pulse mask and line model files.

C.2.1 Simulation Program Listing

```
program PulseAnalysis;

{--------------------------------------------------------------------}
{-                                                                  -}
{- A pulse in the time domain is stored as a set of 2**n points in a -}
{- text file. .This program does an FFT of these points, passes the  -}
{- transform through a transfer function representing a 22 guage    -}
{- twisted pair transmission line, of a user entered length, and does -}
{- an inverse FFT to recover the time pulse.  The recovered pulse is -}
{- compared to a DSX-1 pulse mask, and information on the status of  -}
{- the fit is displayed on the screen.  A menu is displayed, from which-}
{  the user can select from several options.                         -}
{-                                                                  -}
{-Copyright © 1987 Alberta Telecommunications Research Centre        -}
{-All rights reserved                                                -}
{-        Include Files:                                             -}
{-             RealFFT.inc      procedure   RealFFT                  -}
{-             FFTB2.inc        procedure   MakeSinCosTable          -}
{-                                          BitInvert                -}
{-                                          FFT                      -}
{-             CompFFT.inc      procedure   ComplexFFT               -}
{-             Haloturb.p       Halo/Turbo Pascal interface          -}
{--------------------------------------------------------------------}

{$I-}                       { Disable I/O error trapping }


const
  IOerr : boolean = false;  { Flags I/O errors }
  TNarraySize = 300;        { Max size of vectors}
  DisplayPoints = 64;       { # of sample points to be displayed-global }
  DS1freq = 1.544E6;        { DS - 1 frequency }
  NSampPerBit = 32;         { Number of samples per bit }
  MaskLimit = 1250;         { length of pulse mask in nanosec }
  minX = -200;              { left side of screen }
  maxX = 1450;              { right side of screen }
  minY = -2;                { bottom of screen }
  maxY = 2.25;              { top of screen }
  Bell = #7;                { ASCII bell character }
  ZL = 100;                 { terminating impedance }
  pi = 3.141592654;
  PointsNeeded = 51;        { all points > PointsNeeded means f> 10 MHz }
  ConvFactor = 621.37119E-6;{ conversion from metres to miles }
  Rzero = 170.4;            { distributed res. of line at DC }

type
  ptr_int = ^ integer;   { ptr type used in haloturb.p }

  TNvector = array [0..TNarraySize] of real;
  TNvectorPtr = ^TNvector;
  StringVar = string[60];   { for passing strings to procedures }

var
  NumPoints : integer;          { Number of points - global }
  Inverse : boolean;            { False = forward FFT }
                                { True= reverse FFT }
  XReal : TNvectorPtr;          { Real vector of points on I/P }
                                { Real vector of coeff. on O/P }
  XRealTime : TNvectorPtr;      { time of each sample point }
  XImag : TNvectorPtr;                 { vector of Imag. coeff. on O/P }
  XRealAdj : TNvectorPtr;              { modified output sample points }
  XmaskShift:TNvectorPtr;              { shifted mask x coordinates }
  Xmask,Ymask: TNvectorPtr;            { x and y vector of pulse mask }
  YmaskShift: TNvectorPtr;             {shifted y coords. of mask }
  MaskPoints: integer;                 { number of points in mask vectors-global }
  DeltaT : real;                       { time between successive samples- global }
  Error : byte;                        { Flags if something went wrong }
  i : integer;                         { dummy index  variable }
  Yshift: real;                        {Vertical shift of pulse mask}
  CurrShift: real;                     { current value of mask shift }
  CurrScale: real;                     { current scale factor value }
  CurrError: real;                     { current overlay error value }
  PassOrFail: boolean;                 { true means pulse fits in mask }
  BtextOn: boolean;                    { true means bit text, false means stroke txt}
  Z0r,Z0i: TNvectorPtr;                { characteristic impedance of line }
  GammaL2r,GammaL2i:TNvectorPtr;{ vector of gamma times L2 values }
  GammaL1r,GammaL1i:TNvectorPtr;{ vector of gamma times L1 values }
  L1,L2: real;                         { L2=line length, and L1 = 1/2*L2 }
  NShift: integer;                     { shift applied to the pulse to bring it into}
                                       { the display window}

{load algorithm procedures }
{$I FFTB2.INC}               { load radix 2, 8088 version }
{$I REALFFT.INC}             { load procedure RealFFT }
{$I COMPFFT.INC}             { load procedure ComplexFFT }
{$I haloturb.p}              { Halo/Pascal interface }



procedure IOCheck;
{ Check for an I/O error and display a message if needed.
  NOTE: This routine is only for Turbo Pascal version 3.0. }
const
  Bell = #7; { The ASCII bell character. }
type
  Prompt = string[80];
var
  IOcode : integer;

procedure Error(Msg : Prompt);
begin
  Writeln;
  Writeln(Bell; Msg);
  Writeln;
end; { procedure Error }

begin { procedure IOCheck }
  IOcode := IOresult;
  IOerr := IOcode <> 0;
  if IOerr then
    case IOcode of
      $01 : Error('File does not exist.');
      $02 : Error('File not open for input.');
      $03 : Error('File not open for output.');
      $04 : Error('File not open.');
      $10 : Error('Error in numeric format.');
      $22 : Error('Assign to standard files not allowed.');
      $91 : Error('Seek beyond end of file.');
      $99 : Error('Unexpected end of file.');
      $F0 : Error('Disk write error.');
      $F1 : Error('Directory is full.');
      $F3 : Error('Too many open files. Be sure FILES=20 in CONFIG.SYS.');
    else
      begin
        Writeln;
        Writeln(Bell);
        Writeln('Unidentified error message = ', IOcode, '. See manual.');
        Writeln;
      end;
    end; { case }
  end; { procedure IOCheck }


  procedure Initialize;

  {--------------------------------------------------------------------}
  {-                                                                  -}
  {- This procedure initializes all the pointer variables,and sets Error=0-}
  {--------------------------------------------------------------------}

  begin
    New(XRealAdj); New(XRealTime);
    New(XReal); New(Xmask); New(Ymask); New(XmaskShift);
    New(YmaskShift);
    New(XImag);
    New(Z0r); New(Z0i);
    New(GammaL2r); New(GammaL2i);
    New(GammaL1r); New(GammaL1i);
    Error := 0;
  end; { procedure Initialize }


  procedure GetData;

  {--------------------------------------------------------------------}
  {-                                                                  -}
  {-   This procedure reads data in from two files.  First, data is read  -}
  {-   from a file containing the stimulus vector, which is a real vector -}
  {-   of data points.  Next, data points are read in from a file called  -}
  {-   Pmask.dat which specifies the pulse mask.  The pulse mask points   -}
  {-   must make a closed polygon.  The user then is asked to enter the   -}
  {-   total transmission line length in metres.  Finally, the user       -}
  {-   enters the name of the file containig the Z0 and gamma values for  -}
  {-   the transmission line, and the data is read in.                    -}
  {-                                                                  -}
  {--------------------------------------------------------------------}

  var
    FileName : string[255];
    InFile : text;
    n: integer;
    GammaR, GammaI: real;

  begin
    Writeln;
    { read in stimulus vector -}

    repeat
      Write('Enter Test File Name? ');
      Readln(FileName);
      Assign(InFile, FileName);
      Reset(InFile);
      ~~IOCheck;~~
    until not IOerr;
    NumPoints := 0;
    while not EOF(InFile)  do
    begin
      Readln(InFile, XReal^[NumPoints]);
      XRealTime^[NumPoints]:= NumPoints * DeltaT;
      NumPoints := Succ(NumPoints);
      IOCheck;
    end;
    Close(InFile);

    { read in pulse template }
      FileName:= 'Pmask.dat';
      Assign(InFile,FileName);
      reset(InFile);
    MaskPoints:= 0;
    while not EOF(InFile) do
    begin
    readln(InFile,Xmask^[MaskPoints],Ymask^[MaskPoints]);
    MaskPoints:= Succ(MaskPoints);
    IOCheck;
    end;
  close(InFile);

  { get line length, and read in the Z0 and Gamma values }
  writeln;
  write('Enter total line length in metres: ');
  readln(L2);
  L2:= L2*ConvFactor;
  L1:= L2/2;
  writeln;
  repeat
    write('Enter transmission line file name: ');
    readln(FileName);
    assign(Infile,FileName);
    reset(Infile);
    IOcheck;
  until not IOerr;
  { read in the data }

  for n:=1 to PointsNeeded do
    begin
    readln(Infile,Z0r^[n],Z0i^[n],GammaR,GammaI);
    GammaL2r^[n]:= GammaR*L2;
    GammaL2i^[n]:= GammaI*L2;
    GammaL1r^[n]:= GammaR*L1;
    GammaL1i^[n]:= GammaI*L1;
    IOcheck;
    end; {for}
  close(Infile);
end; { GetData }


procedure ComplexMultiply( var R1  :  real;
                           var I1  :  real;
                               R2  :  real;
                               I2  :  real);

{-----------------------------------------------------------------------}
{- multiplies two complex numbers (R1,I1) and (R2,I2) and stores the  -}
{- result in (R1,I1).                                                  -}
{-----------------------------------------------------------------------}

var
  Rout,Iout: real;     { temporary variables }

begin { ComplexMultiply }
  Rout:= R1*R2 - I1*I2;
  Iout:= R1*I2 + I1*R2;
  R1:= Rout;
  I1:= Iout;
end;  { ComplexMultiply }

procedure ComplexDivide( var R1   : real;
                         var I1   : real;
                             R2   : real;
                             I2   : real);

{-----------------------------------------------------------------------}
{-                                                                     -}
{-  Divides two complex numbers: the numerator is (R1,I1), the denom.  -}
{-  is (R2,I2), and the result is stored in (R1,I1).                   -}
{-                                                                     -}
{-----------------------------------------------------------------------}

var
  TempR, TempI: real;   { temporary vars }

begin { ComplexDivide }
```

```
    TempR:= (R1*R2+I1*I2)/(sqr(R2)+sqr(I2));
    TempI:= (I1*R2-R1*I2)/(sqr(R2)+sqr(I2));
    R1:= TempR;
    I1:= TempI;
  end; { ComplexDivide }


  procedure coshComplex( var R : real;
                         var I : real);

  {-------------------------------------------------------------------}
  {-                                                                 -}
  {-  Determines the hyperbolic cosine of a complex number. Result is -}
  {-  stored in the input variables R and I.                         -}
  {-                                                                 -}
  {-------------------------------------------------------------------}

  var
    TempR,TempI: real;

  begin  { coshComplex }
    TempR:= cos(I)*((exp(R)+exp(-R))/2);
    TempI:= sin(I)*((exp(R)-exp(-R))/2);
    R:= TempR;
    I:= TempI;
  end;  { coshComplex }


  procedure sinhComplex(var R : real;
                        var I : real);

  {-------------------------------------------------------------------}
  {-                                                                 -}
  {-  Calculates hyperbolic sine of a complex number.  The result is -}
  {-  returned in the input variables R and I.                       -}
  {-                                                                 -}
  {-------------------------------------------------------------------}

  var
    TempR,TempI: real;

  begin { sinhComplex }
    TempR:= cos(I)*((exp(R)-exp(-R))/2);
    TempI:= sin(I)*((exp(R)+exp(-R))/2);
    R:= TempR;
    I:= TempI;
  end;  { sinhComplex }

  procedure tanhComplex(var R : real;
                        var I : real);

  {-------------------------------------------------------------------}
  {-                                                                 -}
  {-  Calculates the hyperbolic tangent of a complex number. Result is }
  {-  stored in the input variables R and I.                         -}
  {-                                                                 -}
  {-------------------------------------------------------------------}

  var
    sinR,SinI,CosR,CosI: real;  { temp. vars }

  begin  { tanhComplex }
    SinR:= R; CosR:= R;
    SinI:= I; CosI:= I;
    sinhComplex(SinR,SinI);
    coshComplex(CosR,CosI);
    ComplexDivide(SinR,SinI,CosR,CosI);
    R:= SinR;
    I:= SinI;
  end; { tanhComplex }
```

```
procedure TransferFunction;
{----------------------------------------------------------------------}
{-                                                                    -}
{-  This procedure alters the FFT of the input pulse according to the -}
{-  line's transfer function.  Those points where f > 10 MHz are set  -}
{-  to zero. Those points representing +ve frequencies are multiplied -}
{-  by the transfer function.  Those points representing -ve frequencies-}
{-  are multiplied by the complex conjugate of the transfer function.  -}
{-                                                                    -}
{----------------------------------------------------------------------}
var
  N1r,N1i,N2r,N2i: real;    { numerator temp. vars. }
  D1r,D1i,D2r,D2i: real;    { denominator temp vars }
  n: integer;
  DCvalue: real;            { transfer function at DC }
  TFcnR,TFcnI: TNvectorPtr;{ real and imag parts of trans. function }


begin  { TrnsferFunction }
  new(TFcnR); new(TFcnI);

  { calculate transfer function values }

  for n:=1 to PointsNeeded do
    begin
      { get numerator }
      N2r:= GammaL1r^[n];
      N2i:= GammaL1i^[n];
      sinhComplex(N2r,N2i);
      ComplexMultiply(N2r,N2i,Z0r^[n],Z0i^[n]);
      N1r:= GammaL1r^[n];
      N1i:= GammaL1i^[n];
      coshComplex(N1r,N1i);
      ComplexMultiply(N1r,N1i,ZL,0);
      N1r:= N1r + N2r;
      N1i:= N1i + N2i;

      { get denominator }
      D2r:= GammaL2r^[n];
      D2i:= GammaL2i^[n];
      sinhComplex(D2r,D2i);
      ComplexMultiply(D2r,D2i,Z0r^[n],Z0i^[n]);
      D1r:= GammaL2r^[n];
      D1i:= GammaL2i^[n];
      coshComplex(D1r,D1i);
      ComplexMultiply(D1r,D1i,ZL,0);
      D1r:= D1r + D2r;
      D1i:= D1i + D2i;

      { now get the final values }
      ComplexDivide(N1r,N1i,D1r,D1i);
      TFcnR^[n]:= N1r;
      TFcnI^[n]:= N1i;
  end; { for }

  { multiply voltage transform by transfer function }

  for n:=1 to NumPoints -1 -PointsNeeded do
    begin
    if n <= PointsNeeded then
      begin
        ComplexMultiply(XReal^[n], XImag^[n], TFcnR^[n], TFcnI^[n]);
        ComplexMultiply(XReal^[NumPoints-n], XImag^[Numpoints-n],
                        TFcnR^[n], -TFcnI^[n]);
      end
      else
        begin
        XReal^[n]:= 0;
        XImag^[n]:= 0;
      end;  {if}

    end;  {for}
    dispose(TFcnR); dispose(TFcnI);
  end;  { TransferFunction }


  procedure ShiftPulse;
  {------------------------------------------------------------------}
  {-                                                                -}
  {-   This procedure shifts the pulse into the display window.     -}
  {-   It locates the pulse's maximum value and shifts the entire trace -}
  {-   by the amount necessary to bring this point into the display -}
  {-   window.  The shift is accomplished by subtracting an offset value -}
  {-   from the time coordinates of the pulse.  This operation is only -}
  {-   done if the maximum value occurs more than 32 sample points away -}
  {-   from time = 0, (ie, if max value is to right of midpoint of window)-}
  {-                                                                -}
  {------------------------------------------------------------------}

  var
    MaxVal: real;        { maximum value of XReal }
    MaxN: integer;
    n: integer;

  begin { ShiftPulse }
    MaxVal:= 0; NShift:= 0;
    for n:=0 to NumPoints-1 do
      if XReal^[n] > MaxVal then
        begin
        MaxN:= n;
        MaxVal:= XReal^[n];
        end;


    if MaxN > 32 then
      begin
      { only shift those pulses that are far to the right }
      NShift:= MaxN - 32;
      for n:=0 to NumPoints-1 do
        XRealTime^[n]:= XRealTime^[n] - NShift * DeltaT;
      end; { if }
  end;   { ShiftPulse }


  procedure ScalePulse(ScaleFactor       : real;
                       var XReal         : TNvectorPtr;
                       var XRealAdj      : TNvectorPtr);

  {------------------------------------------------------------------}
  {-                                                                -}
  {-   This procedure scales the pulse by multiplying the sample values -}
  {-   by a scaling factor.  The modified pulse amplitudes are stored in -}
  {-   the array XRealAdj.                                          -}
  {-                                                                -}
  {------------------------------------------------------------------}

  var
    n : integer;       { loop index variable }

  begin  { ScalePulse }
    for n:=0 to NumPoints-1 do
      XRealAdj^[n]:= XReal^[n] * ScaleFactor;
  end;  { ScalePulse }


  function Difference(var TimeShift    : real;
                      var CurrentTime : real;
                      var VoltageValue: real):real;

  {------------------------------------------------------------------}
  {-                                                                -}
  {-   Calculates the amount in volts that the pulse misses the pulse -}
  {-   template by at time= CurrentTime.  If the point VoltageValue -}
  {-   is inside the pulse template, then 0 is returned.            -}
```

```
{-                                                                -}
{------------------------------------------------------------------}

var
  minvalue, maxvalue: real;  { pulse mask min and max value at }
                             { time=CurrentTime}

function MinMask(var TimeShift : real;
                 var time      : real):real;
{------------------------------------------------------------------}
{-  returns voltage value of minimum curve of pulse mask at time= "time"-}
{------------------------------------------------------------------}

begin  { MinMask }
  if (time >=0 + TimeShift) and (time <= 350 + TimeShift) then
    MinMask:= - 0.05;
  if (time > 350 + TimeShift) and (time < 400 + TimeShift) then
    MinMask:= 0.009 * (time - TimeShift)- 2.65;
  if (time >= 400 + TimeShift) and (time <= 500 + TimeShift) then
    MinMask:= 0.95;
  if (time > 500 + TimeShift) and (time <= 600 + TimeShift) then
    MinMask:= -0.0005 * (time - TimeShift)+ 1.2;
  if (time > 600 + TimeShift) and (time < 650 + TimeShift) then
    MinMask:= -0.008 * (time - TimeShift)+ 5.7;
  if (time >= 650 + TimeShift) and (time <= 800 + TimeShift) then
    MinMask:= -0.45;
  if (time > 800 + TimeShift) and (time <= 925 + TimeShift) then
    MinMask:= 0.002 * (time - TimeShift)- 2.05;
  if (time > 925 + TimeShift) and (time < 1100 + TimeShift) then
    MinMask:= (0.15 / 175) * (time - TimeShift)- 0.992857143;
  if (time >= 1100 + TimeShift) and (time <= 1250 + TimeShift) then
    MinMask:= -0.05;
end; { MinMask }

function MaxMask(var TimeShift  : real;
                 var time       :real):real;

{------------------------------------------------------------------}
{- Returns voltage value of maximum curve of pulse mask at time= "time" -}
{------------------------------------------------------------------}

begin  { MaxMask }
  if (time >= 0 + TimeShift) and (time <= 250 + TimeShift) then
    MaxMask:= 0.05;
  if (time > 250 + TimeShift) and (time < 325 + TimeShift) then
    MaxMask:= 0.01 * (time - TimeShift)- 2.45;
  if (time >= 325 + TimeShift) and (time <= 425 + TimeShift) then
    MaxMask:= 1.15;
  if (time > 425+ TimeShift) and (time < 500 + TimeShift) then
    MaxMask:= (0.004 / 3) * (time - TimeShift)+ 1.71666666666667;
  if (time >= 500 + TimeShift) and (time <= 675 + TimeShift) then
    MaxMask:= 1.05;
  if (time > 675 + TimeShift) and (time <= 725 + TimeShift) then
    MaxMask:= -0.0224 * (time - TimeShift)+ 16.17;
  if (time > 725 + TimeShift) and (time < 1100 + TimeShift) then
    MaxMask:= 0.00032 * (time - TimeShift)- 0.302;
  if (time >= 1100 + TimeShift) and (time <= 1250 + TimeShift) then
    MaxMask:= 0.05;
end; { MaxMask }


begin  { Difference }
  Difference:= 0;   { set default value }
  minvalue:= MinMask(TimeShift,CurrentTime) + Yshift;
  maxvalue:= MaxMask(TimeShift,CurrentTime) + Yshift;
  if minvalue > VoltageValue then
    { means point below minimum curve }
    Difference:= minvalue - VoltageValue;
  if maxvalue < VoltageValue then
    { means point above the maximum curve }
    Difference:= VoltageValue - maxvalue;
```

```
end;  { Difference }


procedure OverlayErrorCalc(var Pass         : boolean;
                           var TimeShift    : real;
                           var XRealAdj     : TNvectorPtr;
                           var XRealTime    : TNvectorPtr;
                           var OverlayError : real);

{-----------------------------------------------------------------}
{-                                                               -}
{-   This procedure calculates the overlay error between the pulse -}
{-   and the pulse template, for the current settings of scale factor -}
{-   and mask position.  The overlay error is defined as the sum over -}
{-   the extent of the pulse mask, of time spacing (DeltaT) multiplied -}
{-   by the difference between the pulse mask and the pulse. The diff. -}
{-   is calculated at the sampling times,  and only those sampling  -}
{-   points falling inside the time span of the pulse template     -}
{-   contribute to the overlay error.                              -}
{-                                                               -}
{-----------------------------------------------------------------}

var
  n : integer;  { loop index variable }

begin { OverlayErrorCalc }
  OverlayError:= 0;
  for n:=0 to NumPoints-1 do
    if (XRealTime^[n] > TimeShift) and
       (XRealTime^[n] < (MaskLimit + TimeShift)) then

      OverlayError:= OverlayError +
                     DeltaT * Difference(TimeShift,XRealTime^[n],
                                                   XRealAdj^[n]);
  if trunc(10*OverlayError) = 0 then
      Pass:= true
    else
      Pass:= false;

end; { OverlayErrorCalc }



procedure InitHalo;

{-----------------------------------------------------------------}
{-                                                               -}
{-   initialize turbo Halo environment. Using IBM EGA graphics card -}
{-   in mode 4 (640 x 350, 16 colors).  World coordinates are set up -}
{-   and a text font is loaded.                                  -}
{-   Files needed on disk:    Halo011.fnt    - bit text font     -}
{-                            Halo106.fnt    - stroke text font  -}
{-                            HaloIBME.dev                       -}
{-                            haloturb.p                         -}
{-                            halortn.exe                        -}
{-                                                               -}
{-----------------------------------------------------------------}

var
  fontname,device: string[14];  { graphics driver file name and font name }
  GrCardMode: integer;          { graphics card mode }
  ht,wd,orient,mode: integer;   { used to set text attributes }
  textclr, back: integer;       { text color and background color }
  wX1,wX2,wY1,wY2: real;        { screen corners in world coords }
  format:integer;               { floating point format }

begin { InitHalo }

  { set video card driver }
  device:= 'haloibme.dev';     { IBM EGA graphics driver }
  SetDev(addr(device));
  GrCardMode:= 4;
  InitGraphics(addr(GrCardMode));
  setieee(addr(GrCardMode));    { set for long real - 8087 version }
  { set world coordinates }
  wX1:=minX;wX2:=maxX;wY1:=minY;wY2:=maxY;
  SetWorld(addr(wX1),addr(wY1),addr(wX2),addr(wY2));

  {load font}
  fontname:= 'halo011.fnt';
  setfont(addr(fontname));
  ht:=1;wd:=1;orient:=0;mode:=1;
  settext(addr(ht),addr(wd),addr(orient),addr(mode));
  textclr:= 7;   { light grey }
  back:= 0;      { black }
  settextclr(addr(textclr),addr(back));
end; { InitHalo }


procedure DisplayText(var x,y      : real;
                      var textline : StringVar);

{-----------------------------------------------------------------------}
{-                                                                     -}
{-   displays the line of text beginning at the x,y coordinates        -}
{-                                                                     -}
{-----------------------------------------------------------------------}

begin  { DisplayText }
  movtcurabs(addr(x),addr(y));
  if BtextOn then
    btext(addr(textline))
  else
    stext(addr(textline));
  deltcur;
end;  { DisplayText }


procedure ClearText( var x,y      : real;
                     var textline  : StringVar);

{-----------------------------------------------------------------------}
{-                                                                     -}
{-   Clears a line of text starting at the x,y coordinates by printing -}
{-   a string of blanks equal in length to the line which is being     -}
{-   cleared.                                                          -}
{-                                                                     -}
{-----------------------------------------------------------------------}

var
  i: integer;             { loop variable }
  spaces: StringVar;      { holds a string of spaces as long as textline }

begin  { ClearText }
  spaces[0]:= chr(length(textline));
  for i:= 1 to length(textline) do
    spaces[i]:=' ';
  movtcurabs(addr(x),addr(y));
  if BtextOn then
    btext(addr(spaces))
  else
    stext(addr(spaces));
  deltcur;
end;   { ClearText }


procedure DrawGraph;

{-----------------------------------------------------------------------}
{-                                                                     -}
{-    Draws the X and Y axes of a graph and labels them appropriately   -}
{-    X-axis = time, Y-axis = voltage.                                  -}
{----------------------------------------------------------------------}

var
  Xtext,Ytext: real;        { x,y coords. of text cursor }
  x,y: real;                {  x,y coords. of graphics cursor }
  labels: StringVar;        { Axis labels }
  n: integer;               { loop index variable }
  color: integer;           { color of the graph lines }
  ht,wd,orient,md: integer; { used in setting bit text attributes }
  asp,rht: real;            { set stroke text attributes }


begin { DrawGraph }
  color:=7;   { light grey }
  setcolor(addr(color));

  { draw the axes }
  x:=0;y:=maxY;
  movabs(addr(x),addr(y));
  y:= minY + 1;
  lnabs(addr(x),addr(y));
  x:= maxX;
  lnabs(addr(x),addr(y));

  { print tick marks on x axis }
  n:= 0;
  Ytext:= -1.25;
  repeat
    y:= -1.0;
    x:=n;
    movabs(addr(x),addr(y));
    y:= -1.05;
    lnabs(addr(x),addr(y));
    str(n,labels);  { change axis label into string var for display }
    if BtextOn then
      Xtext:= n - (25*length(labels)/2)
    else
      {use different spacing for stroke text}
      Xtext:= n - (12.5*length(labels)/2);

    DisplayText(Xtext,Ytext,labels);
    n:= n + 200;
  until n>maxX;

  { place ticks on Y axis and label them }
  n:=-2;
  repeat
    x:=0;y:=n/2;
    movabs(addr(x),addr(y));
    x:=-10;
    lnabs(addr(x),addr(y));
    case n of
      -2 : labels:='-1.0';
      -1 : labels:='-0.5';
       0 : labels:='0';
       1 : labels:='0.5';
       2 : labels:='1.0';
       3 : labels:='1.5';
       4 : labels:='2.0';
    end; { case }
    Xtext:= -15 - 25*length(labels);
    Ytext:= n/2 -0.07;
    DisplayText(Xtext,Ytext, labels);
    n:= n + 1;
  until n > 4;

  { identify the axes }
  labels:= 'TIME (ns)';
  Xtext:= 500;Ytext:= -1.5;
  DisplayText(Xtext,Ytext,labels);
```

```
    if BtextOn then
      begin
      ht:=1;wd:=1;orient:=1;md:=1;
      settext(addr(ht),addr(wd),addr(orient),addr(md));
      end
    else
      begin
      rht:= 0.09; asp:= 1.0 ; orient:= 1;
      setstext(addr(rht),addr(asp),addr(orient));
      end;

    labels:= 'VOLTAGE (V)';
    Xtext:= -150;
    Ytext:= 0;
    DisplayText(Xtext,Ytext,labels);
    orient:= 0;
    if BtextOn then
      settext(addr(ht),addr(wd),addr(orient),addr(md))  {horiz. text}
    else
      setstext(addr(rht),addr(asp),addr(orient));

  end; { DrawGraph }


  procedure DisplayIdentifiers;

  {---------------------------------------------------------------------------}
  {-                                                                         -}
  {-   This procedure places words on the screen labelling the numbers       -}
  {-   that will be displayed indicating the status of the fit.  The         -}
  {-   following are displayed: overlay error, time shift, scale factor,     -}
  {-   a pass/fail indicator, line length, and shift offset needed to        -}
  {-   bring the pulse into the display window.                              -}
  {-                                                                         -}
  {---------------------------------------------------------------------------}

  var
    x,y: real;                       { x and y coordinates }
    word,word2: StringVar;           { current labels }
    n: integer;                      { loop variable }


  begin  { DisplayIdentifiers }
    y:= 2.0;
    for n:= 6 downto 1 do
      begin
      x:=1000;
      case n of
        3:   begin word:= 'SCALE:';word2:=' '; end;
        4:   begin word:= 'SHIFT:';word2:= 'ns';end;
        5:   begin word:= 'ERROR:';word2:= 'ns-V';end;
        6:   begin word:= 'FIT:';  word2:=' '; end;
        2:   begin word:= 'LENGTH:'; word2:= 'm'; end;
        1:   begin word:= 'OFFSET:'; word2:= 'ns'; end;
      end; { case }
      DisplayText(x,y,word);
      x:= 1325;
      DisplayText(x,y,word2);
      y:= y - 0.2;
    end; { for }
  end;  { DisplayIdentifiers }


  procedure DisplayNumber(var x            : real;
                          var y            : real;
                              Number       : real;
                              DecPts       : integer);

  {---------------------------------------------------------------------------}
  {-                                                                         -}
  {-   This procedure displays the Number at the x and y coordinates         -}
  {-   specified, with DecPts giving info about the number of digits         -}
```

```
{-  following the decimal point - 10 means one decimal pt, 100=2,etc.  -}
{-                                                                     -}
{----------------------------------------------------------------------}

var
  errortext: StringVar;   { holds string to display error }
  frac,intl: string[20];  { fraction and integer part of display }

begin  { DisplayNumber }
  errortext:='      ';
  DisplayText(x,y,errortext);
  str(round(DecPts*Number) div DecPts,intl);
  { get decimal places for fraction part }
  str(abs(round(DecPts*Number) mod DecPts),frac);
  errortext:= intl + '.' + frac;
  DisplayText(x,y,errortext);
end;   { DisplayNumber }


procedure DisplaySettings(var OverlayError   : real;
                          var TimeShift      : real;
                          var ScaleFactor    : real;
                          var Pass           : boolean);

{--------------------------------------------------------------------}
{-                                                                  -}
{-   This procedure displays the values of the above variables accurate -}
{-   to one decimal place, opposite to the appropriate label.       -}
{-                                                                  -}
{--------------------------------------------------------------------}

var
  x,y: real;              { x and y coordinates }
  n: integer;             { loop variable }
  word: StringVar;        { used to hold non-numeric display (pass/fail) }


begin  { DisplaySettings }
  y:= 2.0;
  x:= 1175;
  for n:= 6 downto 1 do
    begin

    case n of
      3:    DisplayNumber(x,y,ScaleFactor,100);
      4:    DisplayNumber(x,y,TimeShift,10);
      5:    DisplayNumber(x,y,OverlayError,10);
      6:    begin
            if Pass = true then
              begin
              word:= 'PASS';
              writeln(Bell);
              end
            else
              word:= 'FAIL';
            DisplayText(x,y,word);
            end;
      2:    DisplayNumber(x,y,L2/ConvFactor,10);
      1:    DisplayNumber(x,y,NShift*DeltaT,10);
    end; { case }
    y:= y - 0.2;
  end; { for }
  deltcur;
end;  { DisplaySettings }


procedure DisplayPulse(var YArray        : TNvectorPtr;
                       var XArray        : TNvectorPtr);

{--------------------------------------------------------------------}
{-                                                                  -}
{-   This procedure displays modified time pulse.                   -}
{-                                                                    -}
{----------------------------------------------------------------------}

var
  n,color: integer;   { number of points to be displayed }

begin  { DisplayPulse }
  color:=14;
  setcolor(addr(color));
  movabs(addr(XArray^[0]),addr(YArray^[0]));
  n:= NumPoints;
  polylnabs(addr(XArray^),addr(YArray^),addr(n));
end;   { DisplayPulse }


procedure DisplayMask(var Xmask       : TNvectorPtr;
                      var XmaskShift  : TNvectorPtr;
                      var Ymask       : TNvectorPtr;
                      var TimeShift   : real;
                      var MaskPoints  : integer);

{----------------------------------------------------------------------}
{-                                                                    -}
{- This procedure displays the pulse mask shifted by the amount equal -}
{- to "TimeShift" horisontally, and shifted vertically by "Yshift volts.-}
{-                                                                    -}
{----------------------------------------------------------------------}

var
  n: integer;    { loop index variable, and # of points displayed }
  color:integer; { color of pulse template }

begin  { DisplayMask }

  for n:= 0 to MaskPoints-1 do
    begin
    XmaskShift^[n]:= Xmask^[n] + TimeShift;
    YmaskShift^[n]:= Ymask^[n] + Yshift;
    end; {for}
  movabs(addr(XmaskShift^[MaskPoints-1]),addr(YmaskShift^[MaskPoints-1]));
  color:=12 ;
  setcolor(addr(color));
  n:= MaskPoints;
  polylnabs(addr(XmaskShift^),addr(YmaskShift^),addr(n));
end;   { DisplayMask }



procedure Analyze(var ShiftAtMinErr    : real;
                  var ScaleAtMinErr    : real;
                  var minError         : real;
                  var PassAtMinErr     : boolean);


{----------------------------------------------------------------------}
{-                                                                    -}
{-   This procedure automatically shifts the pulse mask and scales the -}
{-   test pulse.  For each setting of pulse size and template position -}
{-   the overlay error is calculted, and the displays are updated.  The -}
{-   minimum overlay error position is saved, and after the analysis is -}
{-   complete, the pulse and the template are displayed at these      -}
{-   settings.                                                        -}
{-                                                                    -}
{----------------------------------------------------------------------}

const
  minScale = 0.2;      { starting scale factor }
  maxScale = 0.5;      { stopping scale factor }
  ScaleStep = 0.1;     { increment for scale factor changes }
  TimeStep = 120;       { step in ns for moving pulse template }
  NumShifts = 2;      { number of shifts. NumShifts*TimeStep=sweep in ns }
  var
    { minError - keeps value of minmum overlay error }
    { ScaleAtMinErr - scale factor at minimum error position }
    { ShiftAtMinErr - pulse mask shift in ns at min. error position }
    { PassAtMinErr - pass or fail at min. error position }
    n: integer;              { loop index variable }
    ScaleFactor: real;       { scale factor }
    OverlayError: real;      { overlay error value }
    TimeShift: real;         { current shift applied to the pulse mask }

    color: integer;          { background color for clear }
    Pass: boolean;           { true= fit passes,ie., overlayerror=0 }


  begin  { Analyze }
    minError:= 1E6;
    for n:=-(NumShifts div 2) to (NumShifts div 2) do
      begin
      TimeShift:= n * TimeStep;
      ScaleFactor:= minScale;
      Pass:= false;
      color:=0;
      setcolor(addr(color));
      clr;
      DrawGraph;
      DisplayMask(Xmask,XmaskShift,Ymask,TimeShift,MaskPoints);
      DisplayIdentifiers;
      { loop through scale factors }
      repeat
        ScalePulse(ScaleFactor,XReal,XRealAdj);
        OverlayErrorCalc(Pass,TimeShift,XRealAdj,XRealTime,OverlayError);

        DisplayPulse(XRealAdj,XRealTime);

        if OverlayError < minError then
          begin
          minError:= OverlayError;
          ScaleAtMinErr:= ScaleFactor;
          ShiftAtMinErr:= TimeShift;
          PassAtMinErr:= Pass;
          end;
        DisplaySettings(OverlayError,TimeShift,ScaleFactor,Pass);
        ScaleFactor:= ScaleFactor + ScaleStep;
      until ScaleFactor > maxScale;
    end;  { for }
    { display minimum error pulse and mask position }

    color:= 0;
    setcolor(addr(color));
    clr;
    DrawGraph;
    DisplayIdentifiers;

    ScalePulse(ScaleAtMinErr,XReal,XRealAdj);
    DisplayMask(Xmask,XmaskShift,Ymask,ShiftAtMinErr,MaskPoints);
    DisplayPulse(XRealAdj,XRealTime);
    DisplaySettings(minError,ShiftAtMinErr,ScaleAtMinErr,PassAtMinErr);
  end;  { Analyze.}


  procedure GetIncrement(    prompt    : StringVar;
                         var Increment: real);

  {-----------------------------------------------------------------}
  {-                                                               -}
  {-   Reads in an increment value from the keyboard.  This procedure -}
  {-   is used for reading in both the scale increment and the move  -}
  {-   increment for the interactive part of the program.            -}
  {-                                                               -}
  {-----------------------------------------------------------------}
```

```
const
  ErrorPrompt = 'Non - numeric input: hit any key to continue.';

var
  x,y: real;                  { x,y coordinates of text cursor }
  code: integer;              { return code from Val procedure }
  ch: char;                   { input character }
  displaychar: StringVar;     { used to pass "ch" to display procedures }
  color: integer;             { used for inquire procedures }
  ht,wd: real;                { height and width of bit text }
  EndOfPrompt: real;          { x coord. of end of prompt }
  pos: integer;               { current position in input string }
  Inc: string[10];            { holds user entered input string }
  GoodData: boolean;          { true means value entered is OK }
  CurrPrompt: StringVar;      { sets current prompt value }
  fname: StringVar;

begin  { GetIncrement }

  repeat
    pos:= 1;
    x:= 0;  y:= -1.8;
    Inc:= '';  { empty }
    DisplayText(x,y,prompt);
    GoodData:= true;
    inqtcur(addr(EndOfPrompt),addr(y),addr(color));
    { read the input until a return is entered }
    repeat
      read(kbd,ch);
      if ch = #8 then
        {backspace: go back one character}
        begin
        displaychar:= Inc[pos-1];
        inqtsize(addr(ch),addr(wd),addr(ht));
        inqtcur(addr(x),addr(y),addr(color));
        if pos > 1 then
          { erase character }
          begin
          x:= x - 23.2;  {position cursor 1 character back }
          pos:= pos - 1;
          delete(Inc,pos,1);  {delete from input string}
          displaychar:= ' ';
          DisplayText(x,y,displaychar);
          movtcurabs(addr(x),addr(y));    {reposition cursor }
        end;  {if}
        end
      else
        if ch <> #13 then
          begin
          { ch is not the return character }
          displaychar:= ch;
          inqtcur(addr(x),addr(y),addr(color));
          DisplayText(x,y,displaychar);
          Inc:= concat(Inc,ch);
          pos:= pos + 1;
        end;
    until ch = #13; {ie., return is entered}

    { if any characters entered, get numeric value }
    if pos > 1 then
      begin  { if no characters entered, then this part skipped }
      Val(Inc,Increment,code);
      if code <> 0 then
        {entry is non-numeric}
        begin
        x:=0; y:= -1.8;
        ClearText(x,y,prompt);
        CurrPrompt:= ErrorPrompt;
        DisplayText(x,y,CurrPrompt);
        repeat
        until keypressed;
```

```
          read(kbd,ch);
          ch:= ' ';  { to avoid erroneous loop termination }
          ClearText(x,y,CurrPrompt);
          GoodData:= false;
          end; {if}
        end
      else
        GoodData:= true;
    until GoodData;
    x:=0; y:= -1.8;
    ClearText(x,y,CurrPrompt); {clear prompt line}
  end;   { GetIncrement }


  procedure DisplayScreen;

  {----------------------------------------------------------------------}
  {-                                                                    -}
  {-   redisplays all images on the screen with current settings        -}
  {-                                                                    -}
  {----------------------------------------------------------------------}

  var
    color: integer;  { background color }

  begin  { DisplayScreen }
    color:=0;
    setcolor(addr(color));
    clr;
    DrawGraph;
    DisplayIdentifiers;
    DisplaySettings(CurrError,CurrShift,CurrScale,PassOrFail);
    DisplayMask(Xmask,XmaskShift,Ymask,CurrShift,MaskPoints);
    DisplayPulse(XRealAdj,XRealTime);

  end;   { DisplayScreen }


  procedure MoveAndScale;

  {----------------------------------------------------------------------}
  {-                                                                    -}
  {-   This procedure asks the user to enter a scale increment, and a   -}
  {-   move increment.  Once these values are entered, the up/down arrows -}
  {-   are used to rescale the pulse, and the left/right arrows are used -}
  {-   to move the pulse mask.  Hitting escape terminates the session,  -}
  {-   but this can only be done after the increment values are entered. -}
  {-   Hitting return without entering any increment value sets the     -}
  {-   increment to the default value of 0.                             -}
  {-                                                                    -}
  {----------------------------------------------------------------------}

  const
    prompt1 = 'Enter Scale Increment: ';
    prompt2 = 'Enter Move Increment in nsec: ';
    msg1 = 'Use up/down arrows to scale pulse - Hit any key to continue.';
    msg2 = 'Use left/right arrows to move mask - Hit any key to continue.';
    msg3 = 'Pulse and mask may now be altered...';
    msg4 = 'Hit any key to continue.';
    msg5 = 'Press Escape to return to main menu.';

  var
    ScaleInc : real;         { Scale Increment }
    MoveInc : real;          { Move increment in ns }
    ch : char;               { used for keyboard input }
    x,y : real;              { x,y coordinates of test cursor }
    CurrLine: StringVar;     { current line to be displayed }

  begin  { MoveAndScale }
    ScaleInc:= 0;  MoveInc:= 0;  { set default values }
    x:=0;  y:= -1.8;
```

```
{ get scale increment }
GetIncrement(prompt1,ScaleInc);
CurrLine:= msg1;
DisplayText(x,y,CurrLine);
repeat
until keypressed;
read(kbd, ch);  { clear buffer }
ClearText(x,y,CurrLine);

{ get move increment }
GetIncrement(prompt2,MoveInc);
CurrLine:= msg2;
DisplayText(x,y,CurrLine);
repeat
until keypressed;
read(kbd, ch);  { clear buffer }
ClearText(x,y,CurrLine);

{ print message informing user how to continue }
CurrLine:= msg3;
DisplayText(x,y,CurrLine);
y:= -2;
CurrLine:= msg4;
DisplayText(x,y,CurrLine);
repeat
until keypressed;
read(kbd,ch);  { clear buffer }
y:= -1.8;CurrLine:= msg3;
ClearText(x,y,CurrLine);
y:= -2;CurrLine:= msg4;
ClearText(x,y,CurrLine);

{ display message 5 }
CurrLine:= msg5;
DisplayText(x,y,CurrLine);
ch:= ' ';   { to avoid accidental loop termination }
{ look for arrow keys }
repeat
  if keypressed then
    begin
    read(kbd,ch);
    if (ch = #27) and keypressed then {one more char}
      begin
      read(kbd,ch);
      case ch of
        #73:   begin {page up}
                  Yshift:= Yshift + ScaleInc;
                  OverlayErrorCalc(PassOrFail,CurrShift,XRealAdj,
                                     XRealTime,CurrError);
                  DisplayScreen;
                  DisplayText(x,y,CurrLine);
               end;

        #81:   begin {Page Down}
                 Yshift:= Yshift -ScaleInc;
                 OverlayErrorCalc(PassOrFail,CurrShift,XRealAdj,
                                  XRealTime,CurrError);
                 DisplayScreen;
                 DisplayText(x,y,CurrLine);
               end;

        #75:   begin  {left arrow}
                 CurrShift:= CurrShift - MoveInc;
                 OverlayErrorCalc(PassOrFail,CurrShift,XRealAdj,
                                  XRealTime,CurrError);
                 DisplayScreen;
                 DisplayText(x,y,CurrLine);
               end;

        #77:   begin  {right arrow}
                 CurrShift:= CurrShift + MoveInc;
```

```
                OverlayErrorCalc(PassOrFail,CurrShift,XRealAdj,XRealTime;
                                 CurrError);
                DisplayScreen;
                DisplayText(x,y,CurrLine);
              end;

        #72:  begin  {up arrow}
                CurrScale:= CurrScale + ScaleInc;
                ScalePulse(CurrScale,XReal,XRealAdj);
                OverlayErrorCalc(PassOrFail,CurrShift,XRealAdj,XRealTime,
                                 CurrError);
                DisplayScreen;
                DisplayText(x,y,CurrLine);
              end;

        #80:  begin  {down arrow}
                CurrScale:= CurrScale - ScaleInc;
                ScalePulse(CurrScale,XReal,XRealAdj);
                OverlayErrorCalc(PassOrFail,CurrShift,XRealAdj,XRealTime,
                                 CurrError);
                DisplayScreen;
                DisplayText(x,y,CurrLine);
              end;
        end; {case}
      end; {if}
    end; {if}
  until (ch= #27) and not(keypressed);
  ClearText(x,y,CurrLine);

end;  { MoveAndScale }


procedure PlotScreen;

{--------------------------------------------------------------------}
{-                                                                  -}
{-   Plots the screen with the current scale and shift settings     -}
{-                                                                  -}
{--------------------------------------------------------------------}

const
  listparm: array[0..4] of integer = (1,128,1,0,0);
                             { parameters for lopen command }
  segment: integer = 0;      { memory segment }


var
  listfile: string[15];            { output file name }
  listbuff: array[1..128] of char; { used in lopen command }
  color: integer;                  { text color }
  fill: integer;                   { text fill }
  height,aspect: real;             { for stroke text }
  mode,path: integer;              { stroke and bit text parms. }
  FontName: string[15];            { current font name }
  ht,wd: integer;                  { bit text height and width }


begin    { PlotScreen }
  { open display list }
  listfile:= 'list.out';
  lopen(addr(listfile),addr(segment),addr(listbuff),addr(listparm));

  { load stroke text font }
  BtextOn:= false;  { use stroke text }
  FontName:= 'halo106.fnt';
  setfont(addr(FontName));
  color:= 7; fill:= 0;
  setstclr(addr(color),addr(fill));
  height:= 0.09; aspect:= 1.0; path:= 0;
  setstext(addr(height),addr(aspect),addr(path));
  DisplayScreen;
  lclose;
  { reset the text to bit text }
  BtextOn:= true;
  FontName:= 'halo011.fnt';
  setfont(addr(FontName));
  fill:= 0;
  settextclr(addr(color),addr(fill));
  ht:= 1; wd:= 1; path:= 0; mode:=1;
  settext(addr(ht),addr(wd),addr(path),addr(mode));
  DisplayScreen;
end;  { PlotScreen }


procedure InteractiveState;

{-----------------------------------------------------------------------}
{-                                                                     -}
{-   This block handles the interaction with the user to manually      -}
{-   optimize the pulse scale and mask position.  The user is presented -}
{-   with three choices: 1) Move Mask and Scale Pulse, 2) Plot Screen-  -}
{-   and 3) Quit.  Depending on which number is entered, the program   -}
{-   calls the appropriate procedures.                                 -}
{-                                                                     -}
{-----------------------------------------------------------------------}

const
  line = '1) Move Mask and Scale Pulse   2) Plot Screen   3) Quit';
  prompt = 'Enter Selection:';

var
  x,y : real;       { x,y coordinates of text cursor }
  ch: char;         { input character }
  CurrLine: StringVar;
  x1,y1,x2,y2: real;


begin  { InteractiveState }
  repeat
    { print options at bottom of screen }
    x:=0; y:= -1.8;
    CurrLine:= line;
    DisplayText(x,y,CurrLine);
    y:= -2; CurrLine:= prompt;
    DisplayText(x,y,CurrLine);
    { read in the user's choice }
    repeat
      read(kbd,ch);
    until ch in ['1'..'5'];
    y:= -1.8;
    CurrLine:= line;
    ClearText(x,y,CurrLine);
    y:=-2; CurrLine:= prompt;
    ClearText(x,y,CurrLine);
    BtextOn:= true;
    case ch of
      '1': MoveAndScale;
      '2': PlotScreen;
      '5': begin
             y1:=-2;y2:=2.25;x1:=200;x2:=8000;
             setworld(addr(x1),addr(y1),addr(x2),addr(y2));
             DisplayScreen;
             y1:=minY;y2:=maxY;x1:=minX;x2:=maxX;
             setworld(addr(x1),addr(y1),addr(x2),addr(y2));
           end;

    end; {case}
  until ch = '3';  { quit }
end;  { InteractiveState }


procedure LPF;
```

```
{
Multiplies the FFT of the DS-1 pulse by a 3rd order Butterworth
LPF transfer function with 3 dB frewuency of 1.544 MHz.
}

var
  R,I : real;
  n: integer;
  Ratio : real;

begin  { LPF }
  for n:=0 to NumPoints div 2 do
    begin
    Ratio:= (193E3 * n) / 5E6;
    R:= 1 - 2 * sqr(Ratio);
    I:= Ratio * ( 2 - sqr(Ratio));
    if n < NumPoints div 2 then
      ComplexDivide(XReal^[n],XImag^[n],R,I);
    ComplexDivide(XReal^[NumPoints - n], XImag^[NumPoints - n],R,-I);
  end;  {for}

end;  { LPF }


{***********************************************************************}
{********************                           ***************************}
{********************      MAIN PROGRAM         ***************************}
{********************                           ***************************}
{***********************************************************************}

begin { PulseAnalysis }
  DeltaT:= (1/(DS1freq*NSampPerBit)) * 1E9;
  Initialize;
  GetData;
  Inverse:= false;
  RealFFT( NumPoints, Inverse, XReal, XImag, Error);

  LPF;
  { Pass frequency domain pulse thru transfer function }
  {
  ZinCalc;
  }
  TransferFunction;
  { Perform inverse FFT }
  Inverse:= true;
  ComplexFFT(NumPoints, Inverse, XReal,XImag, Error);
  { proceed into graphics mode }

  Yshift:= 0;
  ShiftPulse;
  InitHalo;
  BtextOn:= true;
  Analyze(CurrShift, CurrScale, CurrError, PassOrFail);
  InteractiveState;
  closegraphics;
end.
```

C.2.2 Short Line Pulse Shape

Line	Value	Line	Value	Line	Value
1	0	51	3.02	101	0
2	0	52	3.02	102	0
3	0	53	-0.93	103	0
4	0	54	-0.93	104	0
5	0	55	-0.93	105	0
6	0	56	-0.93	106	0
7	0	57	-0.7	107	0
8	0	58	-0.7	108	0
9	0	59	-0.7	109	0
10	0	60	-0.7	110	0
11	0	61	-0.47	111	0
12	0	62	-0.47	112	0
13	0	63	-0.47	113	0
14	0	64	-0.47	114	0
15	0	65	0	115	0
16	0	66	0	116	0
17	0	67	0	117	0
18	0	68	0	118	0
19	0	69	0	119	0
20	0	70	0	120	0
21	0	71	0	121	0
22	0	72	0	122	0
23	0	73	0	123	0
24	0	74	0	124	0
25	0	75	0	125	0
26	0	76	0	126	0
27	0	77	0	127	0
28	0	78	0	128	0
29	0	79	0	129	0
30	0	80	0	130	0
31	0	81	0	131	0
32	0	82	0	132	0
33	0	83	0	133	0
34	0	84	0	134	0
35	0	85	0	135	0
36	0	86	0	136	0
37	3.25	87	0	137	0
38	3.25	88	0	138	0
39	3.25	89	0	139	0
40	3.25	90	0	140	0
41	3.02	91	0	141	0
42	3.02	92	0	142	0
43	3.02	93	0	143	0
44	3.02	94	0	144	0
45	3.02	95	0	145	0
46	3.02	96	0	146	0
47	3.02	97	0	147	0
48	3.02	98	0	148	0
49	3.02	99	0	149	0
50	3.02	100	0	150	0

Line	Value
151	0
152	0
153	0
154	0
155	0
156	0
157	0
158	0
159	0
160	0
161	0
162	0
163	0
164	-3.25
165	-3.25
166	-3.25
167	-3.25
168	-3.02
169	-3.02
170	-3.02
171	-3.02
172	-3.02
173	-3.02
174	-3.02
175	-3.02
176	-3.02
177	-3.02
178	-3.02
179	-3.02
180	0.93
181	0.93
182	0.93
183	0.93
184	0.7
185	0.7
186	0.7
187	0.7
188	0.47
189	0.47
190	0.47
191	0.47
192	0
193	0
194	0
195	0
196	0
197	0
198	0
199	0
200	0
201	0
202	0
203	0
204	0
205	0
206	0
207	0
208	0
209	0
210	0
211	0
212	0
213	0
214	0
215	0
216	0
217	0
218	0
219	0
220	0
221	0
222	0
223	0
224	0
225	0
226	0
227	0
228	0
229	0
230	0
231	0
232	0
233	0
234	0
235	0
236	0
237	0
238	0
239	0
240	0
241	0
242	0
243	0
244	0
245	0
246	0
247	0
248	0
249	0
250	0
251	0
252	0
253	0
254	0
255	0
256	0

C.2.3 Long Line Pulse Shape

Line	Value	Line	Value	Line	Value
1	0	51	2.79	101	0
2	0	52	2.79	102	0
3	0	53	-1.63	103	0
4	0	54	-1.63	104	0
5	0	55	-1.63	105	0
6	0	56	-1.63	106	0
7	0	57	-0.93	107	0
8	0	58	-0.93	108	0
9	0	59	-0.93	109	0
10	0	60	-0.93	110	0
11	0	61	-0.47	111	0
12	0	62	-0.47	112	0
13	0	63	-0.47	113	0
14	0	64	-0.47	114	0
15	0	65	0	115	0
16	0	66	0	116	0
17	0	67	0	117	0
18	0	68	0	118	0
19	0	69	0	119	0
20	0	70	0	120	0
21	0	71	0	121	0
22	0	72	0	122	0
23	0	73	0	123	0
24	0	74	0	124	0
25	0	75	0	125	0
26	0	76	0	126	0
27	0	77	0	127	0
28	0	78	0	128	0
29	0	79	0	129	0
30	0	80	0	130	0
31	0	81	0	131	0
32	0	82	0	132	0
33	0	83	0	133	0
34	0	84	0	134	0
35	0	85	0	135	0
36	0	86	0	136	0
37	3.48	87	0	137	0
38	3.48	88	0	138	0
39	3.48	89	0	139	0
40	3.48	90	0	140	0
41	2.79	91	0	141	0
42	2.79	92	0	142	0
43	2.79	93	0	143	0
44	2.79	94	0	144	0
45	2.79	95	0	145	0
46	2.79	96	0	146	0
47	2.79	97	0	147	0
48	2.79	98	0	148	0
49	2.79	99	0	149	0
50	2.79	100	0	150	0

Line	Value
151	0
152	0
153	0
154	0
155	0
156	0
157	0
158	0
159	0
160	0
161	0
162	0
163	0
164	-3.48
165	-3.48
166	-3.48
167	-3.48
168	-2.79
169	-2.79
170	-2.79
171	-2.79
172	-2.79
173	-2.79
174	-2.79
175	-2.79
176	-2.79
177	-2.79
178	-2.79
179	-2.79
180	1.63
181	1.63
182	1.63
183	1.63
184	0.93
185	0.93
186	0.93
187	0.93
188	0.47
189	0.47
190	0.47
191	0.47
192	0
193	0
194	0
195	0
196	0
197	0
198	0
199	0
200	0
201	0
202	0
203	0
204	0
205	0
206	0
207	0
208	0
209	0
210	0
211	0
212	0
213	0
214	0
215	0
216	0
217	0
218	0
219	0
220	0
221	0
222	0
223	0
224	0
225	0
226	0
227	0
228	0
229	0
230	0
231	0
232	0
233	0
234	0
235	0
236	0
237	0
238	0
239	0
240	0
241	0
242	0
243	0
244	0
245	0
246	0
247	0
248	0
249	0
250	0
251	0
252	0
253	0
254	0
255	0
256	0

C.2.4 Pulse Mask

X Coordinate	Y Coordinate
0	0.05
250	0.05
325	0.8
325	1.15
425	1.15
500	1.05
675	1.05
725	-0.07
1100	0.05
1250	0.05
1250	-0.05
1100	-0.05
925	-0.2
800	-0.45
650	-0.45
650	0.5
600	0.9
500	0.95
400	0.95
350	0.5
350	-0.05
0	-0.05

Figure B1:
LBO CIRCUIT SCHEMATIC
Designed by Myron Borys
Sheet 1 of 1
Dated January 22, 1988
Project # 140101
(c) Alberta Telecommunications Research Centre
(ATRC)