DEVELOPMENT AND UTILIZATION
OF A
SIGNAL AVERAGING SYSTEM

DEVELOPMENT AND UTILIZATION OF A SIGNAL AVERAGING SYSTEM

by

T. B. REMPLE, B. Eng.

A Project Report
Submitted to the School of Graduate Studies
in Partial Fulfilment of the Requirements
for the Degree
Master of Engineering

McMaster University
November 1979

DESCRIPTIVE NOTE

The software for an existing signal averaging system was developed, and the system was then used to measure pulse dispersion in graded index fibres.

ABSTRACT

A sampling oscilloscope was interfaced to a PDP11/10 minicomputer via one analog-to-digital convertor (ADC) and one digital-to-analog convertor (DAC), with the minicomputer being interfaced to a real time oscilloscope or plotter via two more DAC's. A software program that employed both assembler and Fortran was developed. This program controlled the signal averaging, did some data analysis, could output the data in normalized form onto a real time scope or a plotter, and also input and output the signal using punched tape.

The usefulness of the system was then demonstrated by measuring the rise time of a short laser pulse through various lengths of graded index fibre, and the fibre response was extracted as a function of fibre length.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF FIGURES (cont.)

LIST OF FIGURES (cont.)

1. INTRODUCTION

An experimentalist is often faced with the problem of measuring signals that are obscured by a large amount of noise. If meaningful measurements are to be made on the noisy signal, something must be done to reduce the amount of noise. In such cases, a process called signal averaging can often be used to enhance the signal-to-noise ratio.

Signal averaging can be described in the following way. Generally, a signal can be viewed on an oscilloscope, and a trace of signal amplitude versus time is then obtained, as in figure 1.1. This trace can be imagined to be the sum of two traces, one of the "pure, noiseless" signal, (figure 1.2), and one of the noise, (figure 1.3). The sum of the traces in figures 1.2 and 1.3 would result in the trace of figure 1.1.

If a second trace of the pure signal were made, and added to the trace in figure 1.2, the result would be a trace with exactly twice its original amplitude. However, if a second trace of the noise were added to the trace of figure 1.3, the result would not generally have twice its original amplitude, because the "peaks" and "valleys" would not occur in the same place.

SINGLE SCAN OF NOISE PLUS SIGNAL

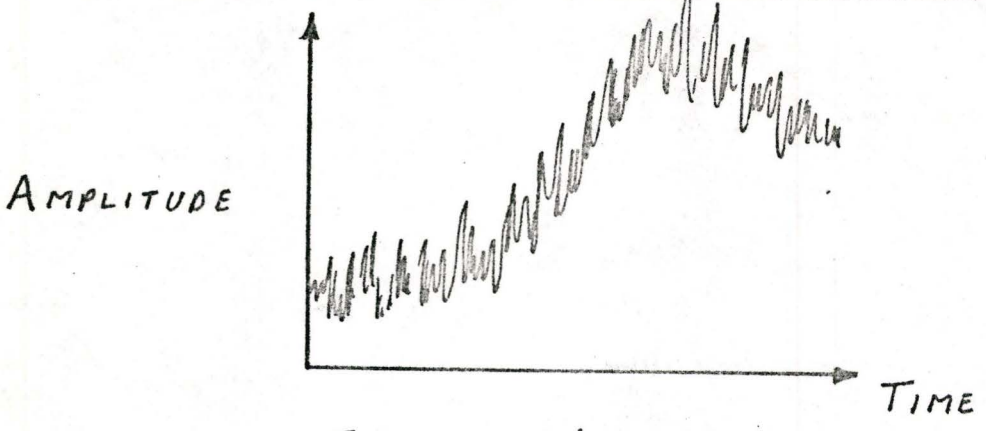

FIGURE 1.1

SINGLE SCAN OF SIGNAL

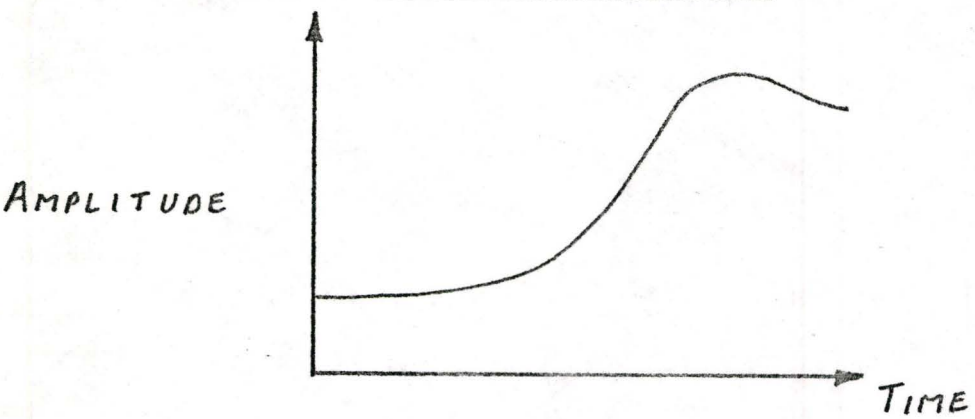

FIGURE 1.2

SINGLE SCAN OF NOISE

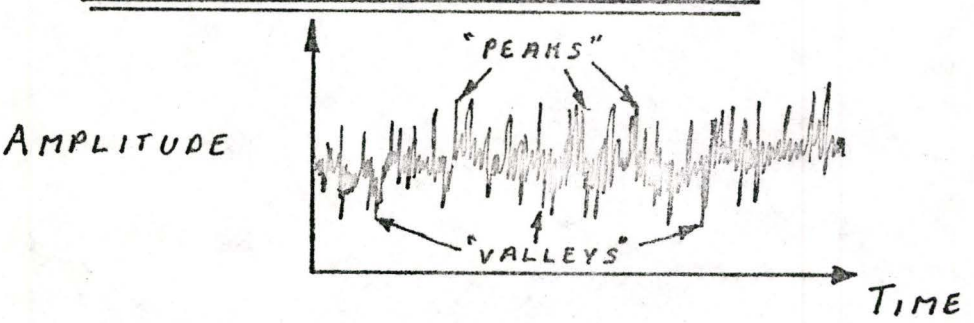

FIGURE 1.3

If we kept taking traces of signal plus noise, and added them to the trace of figure 1.1, the signal would grow in direct proportion to the number of traces added, but the noise would grow more slowly. Thus the signal to noise retio could be raised to any desired value, in principle, using this technique.

2. A DESCRIPTION OF THE SYSTEM

The practical implementation of the process described in section 1 was done with a sampling scope, a minicomputer, an analog-to-digital convertor (ADC) and several digital-to-analog convertors (DAC's).

The sampling scope works on the following principle. Suppose that the input signal is repetitive, as in figure 2.1, and that there is a trigger pulse which occurs a small fixed time before the signal. The sampling scope will start timing itself after it receives the trigger pulse "a", sample the pulse at position "A" on the horizontal time scale, and display this amplitude at position "A" on the sampling scope screen, (see figure 2.2). When the trigger pulse "b" occurs, the sampling scope will wait a slightly longer time, sample the signal at time "B", and display this amplitude at position "B" on the sampling scope screen. This continues with positions "C", "D", etc. until a complete scan has been made, at which time the sampling scope will start over again and sample the signal at time "A".

The sampling scope is readily interfaced to a computer, and the system that was used is shown schematically

Trigger Pulse and Input Signal versus Time

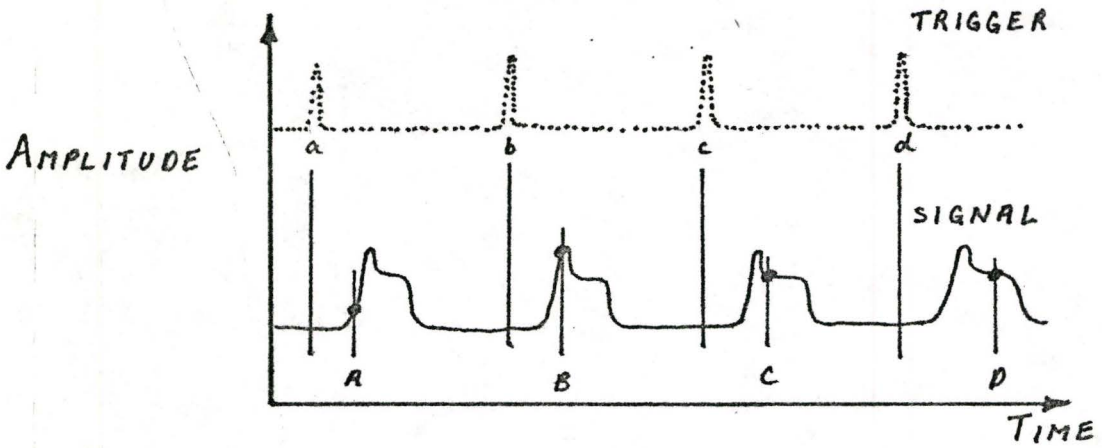

Figure 2.1

Input Signal versus Time as Displayed on Sampling Scope

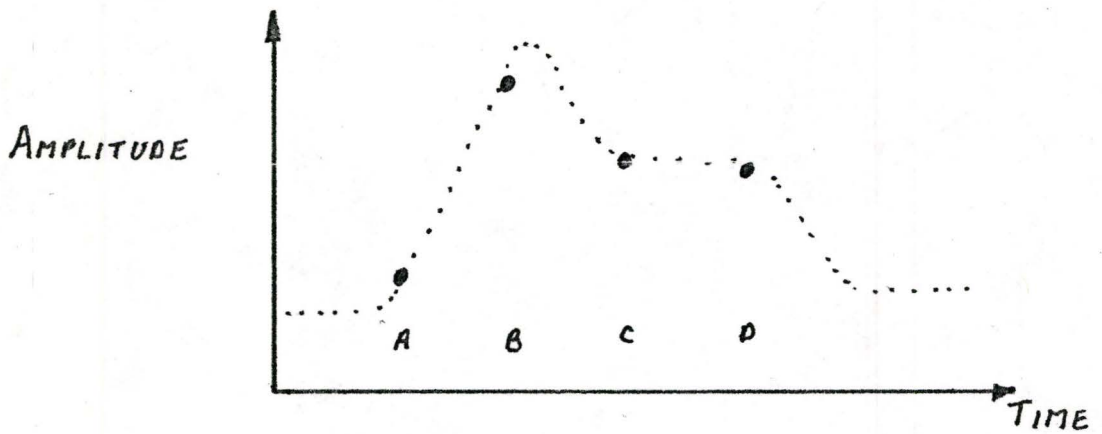

Figure 2.2

in figure 2.3. The minicomputer first sets up and zeros an array with n elements, inside computer memory. Then it sends out a number, i, to DAC2, which controls the horizontal time position at which the signal will be sampled. A command is sent to the ADC, and the amplitude of the signal, a_i, is converted to a number by the ADC. The minicomputer adds this number to element S_i of the array S, and moves on to position i+1. After completing the n^{th} position, the computer starts over again at position 1, and so on, until the desired number of scans have been taken.

Once the sampling is complete, the computer normalizes the array S, so that its minimum value is 0, and its maximum value is 511. (ie. 511 is the maximum value that can be sent to DAC1.)

After normalization, the computer sends the value i to DAC0, which controls the horizontal position of the display scope, and it sends the normalized value S_i to DAC1, which controls the vertical position, (see figure 2.3). Immediately after that, it outputs the values i+1 and S_{i+1}. In this way, the averaged result can be viewed and measured on the display scope.

The signal averaged result can also be output onto a plotter, in which case the user can specify the plotting

Schematic of Signal Averaging System

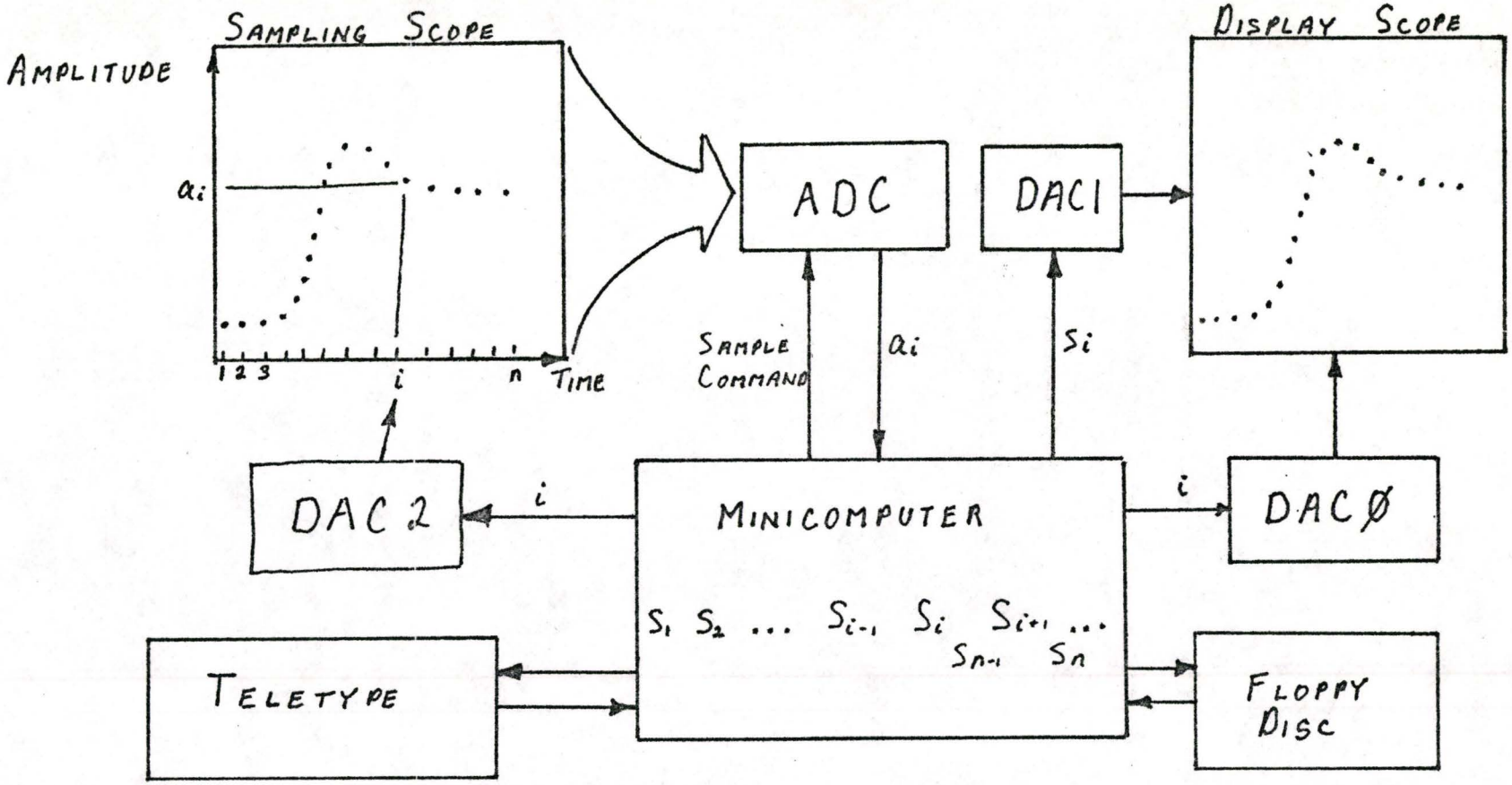

Figure 2.3

speed, and whether or not axes should be drawn. As well, the signal can be stored on punched tape, read back at a later time, and then be read out onto the display scope or plotter.

In the course of developing the control program, it was tested with the setup described in detail in the next section. Briefly though, a light pulse was launched into an optical fibre, and detected at the other end as a function of intensity versus time. The dispersion was then measured by observing how much the pulse spread out in time. The more a pulse spread out, the lower would be its intensity, because energy, which is a function of intensity times time, must be conserved for every pulse, in the absence of significant loss.

The first signal that was obtained was unusually weak, because the testing was done with step index fibre. (ie. Step index fibre is highly dispersive.) After signal averaging, the weak signal was indeed more visible, but now it was obscured by the emergence of coherent, or repeatable, noise almost as large as the signal itself. This coherent noise was thought to be the result of stray coupling between components of the sampling scop. If this was the case, then each time the sampling scope fired, a ringing would occur between some components of the scope, couple

itself to the signal input circuitry, and appear on the scope screen as an additional weak signal.

To remove the influence of such coherent noise signals, the program was modified so that it could first sample the signal, then while the operator blocked the signal, the program would sample the coherent noise, and finally, the program would calculate and display the signal minus the noise.

This process of averaging signal plus noise, then the noise, and subtracting to obtain signal, worked very well for the weak signal from the step index fibre. However, because the signal from the graded index fibre was so much stronger, the noise subtracting feature was not required in subsequent measurements reported here.

The control program operated so as to divide the sampling scope scan into 461 points, and was able to make about 15 to 20 scans per second if the trigger mode of the sampling scope was set on free run. The experiment described in section three had a signal frequency of approximately 2000 Hz, giving about $2000 \frac{\text{signals}}{\text{second}} / 461 \frac{\text{signals}}{\text{scan}} \doteq 4.3 \frac{\text{scans}}{\text{second}}$, (see figure 2.1). This number of points, 461, and this sampling speed, $4.3 \frac{\text{scans}}{\text{second}}$, turned out to be very concenient in terms of obtaining a well resolved signal in a short amount of time. A listing of this program

and a discription of it are given in appendices A and B.

However, if this program were used with a different experiment where the signal frequency was much slower, say 5 Hz, one would only have a sampling speed of 5 $\frac{\text{signals}}{\text{second}}$ / 461 $\frac{\text{signals}}{\text{scan}} \doteq \frac{1 \text{ scan}}{92 \text{ seconds}}$. In order to obtain reasonable signal averaging, it is desirable to have at least 50 scans, and this would take approximately 1¼ hours.

Besides the slow sampling speed, this program had the further disadvantage of destroying all the original data, whenever the program was interupted to display the signal already obtained. Hence, once a signal was displayed, it could not be enhanced further by performing additional signal averaging on it.

For these and other reasons, another version of the same program was written, named SIG1, which had the following additional features. The array S, which stored the sum of the amplitudes of the signal for each point being sampled, would only be cleared if the user so specified. In this way, one could signal average, display the result, and continue signal averaging without having to start over again. Also, the number of sampling points per scan could be specified by the user, as well as their position on the screen. A listing of SIG1 is given in appendix C.

3. FIBRE DISPERSION MEASUREMENT WITH THE SYSTEM

The next step after having completed the signal averaging system, was to demonstrate its usefulness under experimental conditions. The experiment chosen for this end is described below.

An important parameter of any optical fibre is the amount of pulse dispersion it will exhibit for a given length of fibre. If a 100 psec full-width-at-half-maximum (FWHM) light pulse is launched into a typical 1 km length of graded index fibre, one would expect the output pulse to undergo an increase in FWHM and a lowering of peak intensity. (See figure 3.1.)

This phenomenon is known as pulse dispersion, and is the result of three separate effects. Each effect causes some of the light to travel at a different speed than the rest of the light in the pulse. These differences in speed give rise to the spread in arrival times at the end of the fibre. These effects are summarized below.

1) Suppose all of the light were of exactly one frequency, or color. Then, because the outside fibre is made

A Conceptual Look at Dispersion

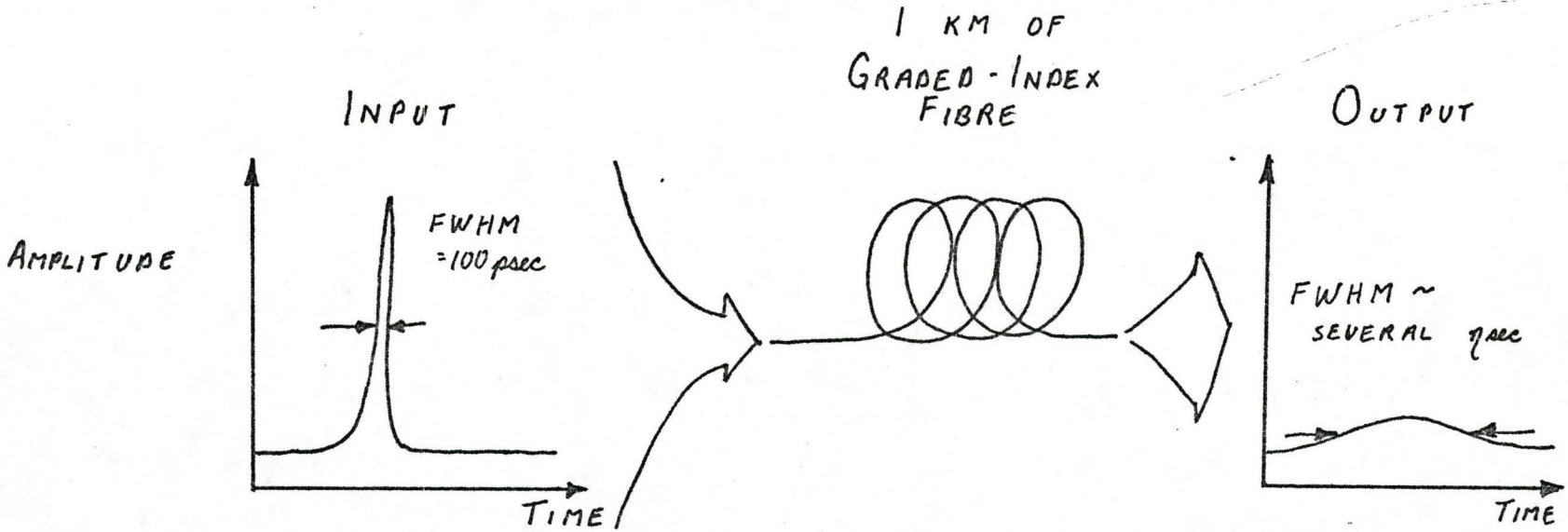

FIGURE 3.1

of glass with a lower refractive index than the glass at the centre of the fibre, light near the outside of the fibre travels faster. In other words, different modes travel at different speeds.

2) However, any input light pulse always contains a range of frequencies, and conditions in the fibre design cause different frequencies to travel at different speeds, over and above that in 3) below.

3) In addition to the fibre design causing dispersion, the glass itself causes different frequencies to travel at different speeds, because the refractive index of glass depends upon frequency.

The purpose, then, of this experiment, was to couple a pulsed laser into various lengths of fibre, and try to determine the relationship between fibre length and pulse spreading, (or more precisely, between fibre length and fibre response).

The experimental setup is shown schematically in figure 3.2. A constant voltage of 113 volts was applied to the laser, and a 10 volt pulse with a repitition rate of 2000 Hz was used to trigger it. As soon as the laser fired, a pulse of current travelled through the laser,

SCHEMATIC OF EXPERIMENTAL SETUP

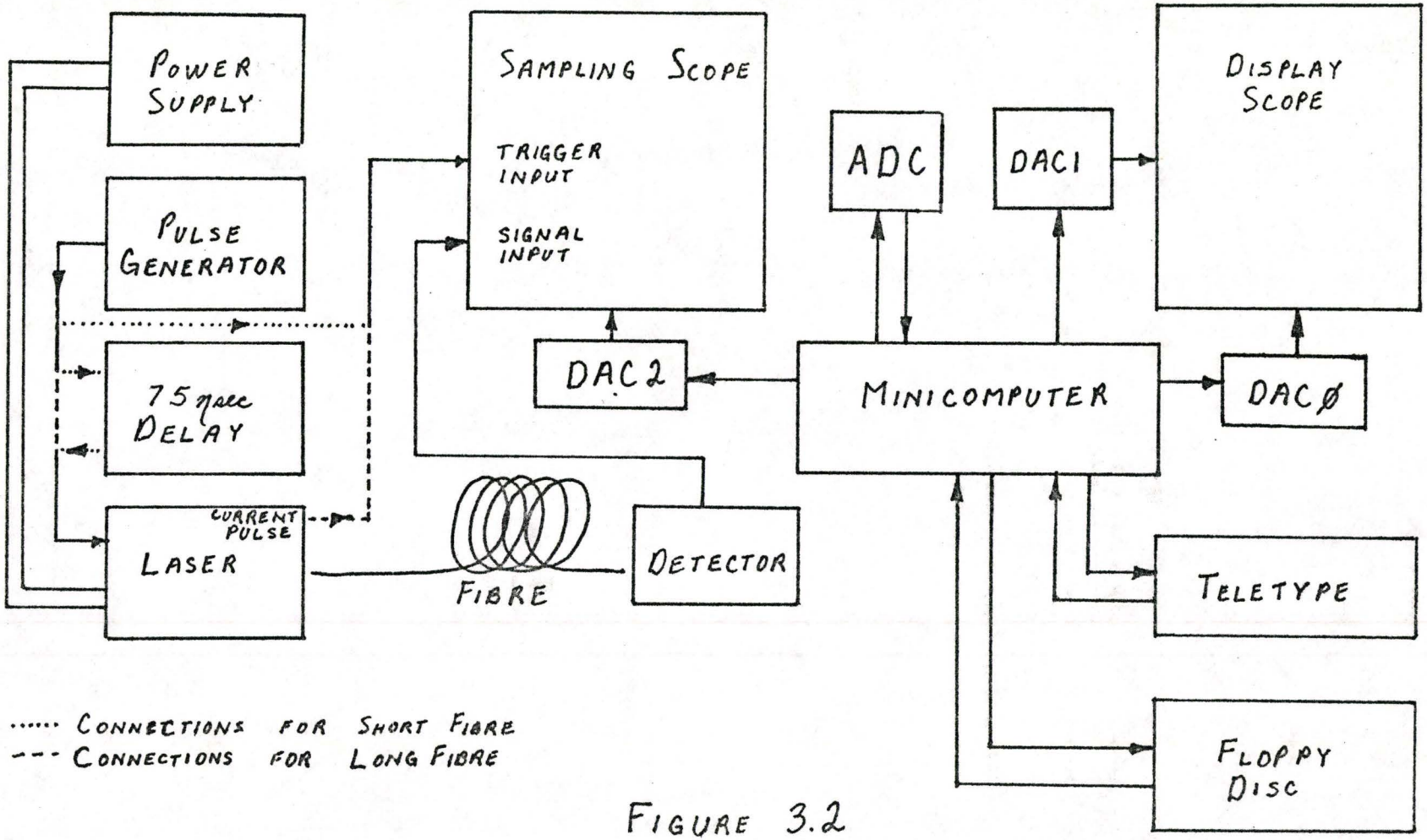

FIGURE 3.2

and this was used to trigger the sampling scope. Notice that although the current pulse occurred at the same time as the light pulse, it was still suitable for use as a trigger because the light pulse would be delayed by as much as several hundred nanoseconds in travelling through the longer lengths of fibre. The sampling scope would first see the current pulse, trigger itself, and still have time to wait for the light pulse.

For short lengths of fibre, the pulse generator output was fed into a passive delay device. This device would send a trigger pulse to the sampling scope, and wait 75 nanoseconds before sending a pulse to fire the laser.

Tests were made to determine whether the response of the detector was dependent on the signal intensity. Such an effect would have a significant influence on the measurements because the signals through the longer lengths of fibre were expected to be weaker than those through the shorter lengths.

To answer this question, a fibre was butted up to the detector and the 10 to 90% output rise time of a 120 mv signal was measured to be 140 ± 2 psec. The fibre was then moved laterally against the face of the detector, so that only part of the light reached the detector.

The rise time was then measured for output signal levels of 90 mv, 60 mv, 40 mv and 20 mv. In all cases, the 10 to 90% rise time was within the range of 140 ± 2 psec. All of the measurements throughout the rest of the experiment had signal levels between 20 to 120 mv, so the detector response was assumed not to have contributed to any spread in the rise times.

A second procedure employed, that might have affected dispersion, was that of using two short fibres to make one long one by carefully aligning the ends of the fibre and applying some index-matching liquid to the joint. The possibility exists that only some of the modes in the first fibre will be coupled to the second one. Should all of the cladding modes be stripped at such a joint, then a measurement of the light being emitted from the second fibre would display less dispersion than if an equivalent single length of fibre were used. It would also be possible for low order modes in one fibre to be coupled to high order modes in the second, further affecting the light pulse.

To check this procedure and its effect on dispersion, the rise time of a pulse through 2.7 m of fibre was measured, and then compared with the rise time of a pulse through the 2.7 m fibre coupled to a .2 m fibre.

There was no measurable change in the rise time, so throughout the rest of the experiment, fibre coupling was assumed not to affect dispersion.

Figure 3.3 shows how the signal appeared on the sampling scope through 2.7 m of fibre, if no signal averaging was performed. As can be seen, the signal is of such a quality that the experiment might have been attempted without signal averaging. But the errors incurred in measuring the 10 to 90% rise time would have ± 20 psec, instead of the ± 2 psec for the averaged signal. This increase in absolute error would have almost negated any conclusions made from the results, so in fact, the signal averaging system was instrumental in the experiment's success.

Careful examination of figure 3.3 will reveal a slight discontinuity in the signal near 300 psec on the horizontal time scale. This was due to a cursor in the sampling scope that could be used to position the signal on the screen. In most other cases, this cursor was placed to the far right of the screen and didn't interfere with the signal.

Figure 3.4 shows the same signal as that in figure 3.3, except that this one has been averaged 128 times,

SINGLE SCAN OF ACTUAL SIGNAL

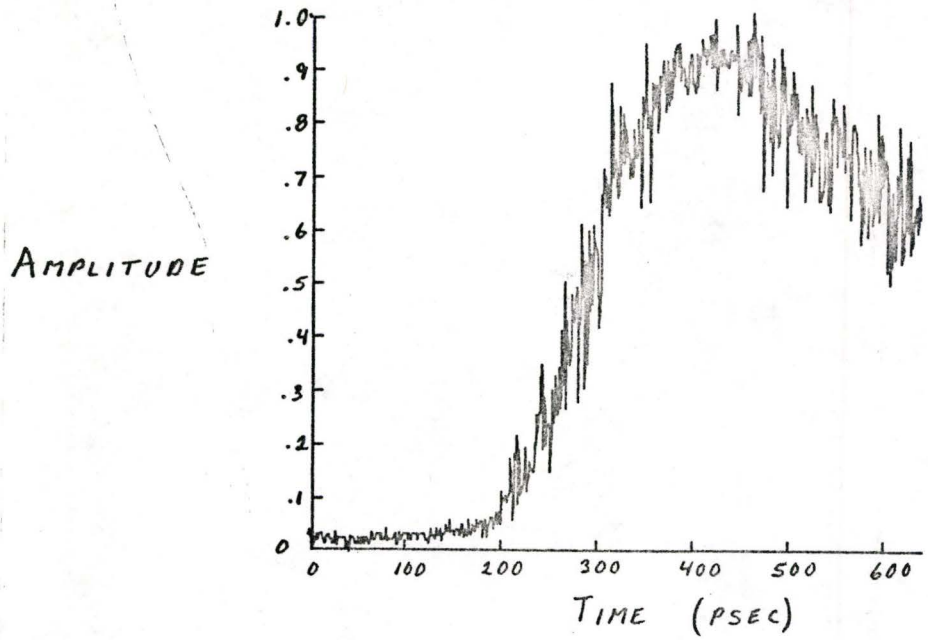

FIGURE 3.3

ACTUAL SIGNAL AVERAGED ON 500 PSEC/DIV

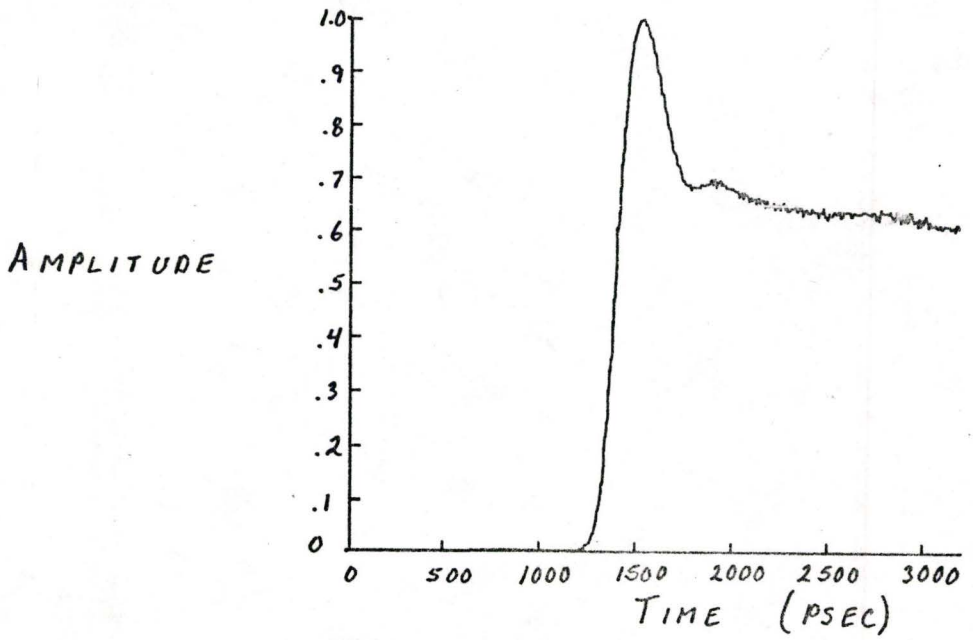

FIGURE 3.4

and is set on a scale of 500 psec/division. This trace proved useful in the analysis, as will be explained later.

Figure 3.5 is again the same signal as that in figures 3.3 and 3.4, (ie. going through 2.7 m of fibre), but this time it has been averaged 128 times, and has been set to a time scale of 100 psec/division. Then, in figures 3.6 to 3.10, one can see how the signal changes as the length of the fibre increases. All traces in figures 3.5 to 3.10 have been signal averaged about 128 times and are set to a scale of 100 psec/division. The length of fibre used, as well as the 10 to 90% rise time as measured from a display oscilloscope with accurate calibrations, are given in each figure.

In addition to these five measurements of rise time for a given fibre length, about 100 other measurements were made over a period of roughly 10 days, on these and other fibres. Although considerable effort was put into consistently reproducing the same experimental conditions, there were significant differences between some measurements made on the same length of fibre but at different days. (ie. In figure 3.12 are the combined results of ten days of measurements, showing the spread in rise time measurements made on different lengths of fibre.)

SIGNAL AFTER 2.7 m OF FIBRE

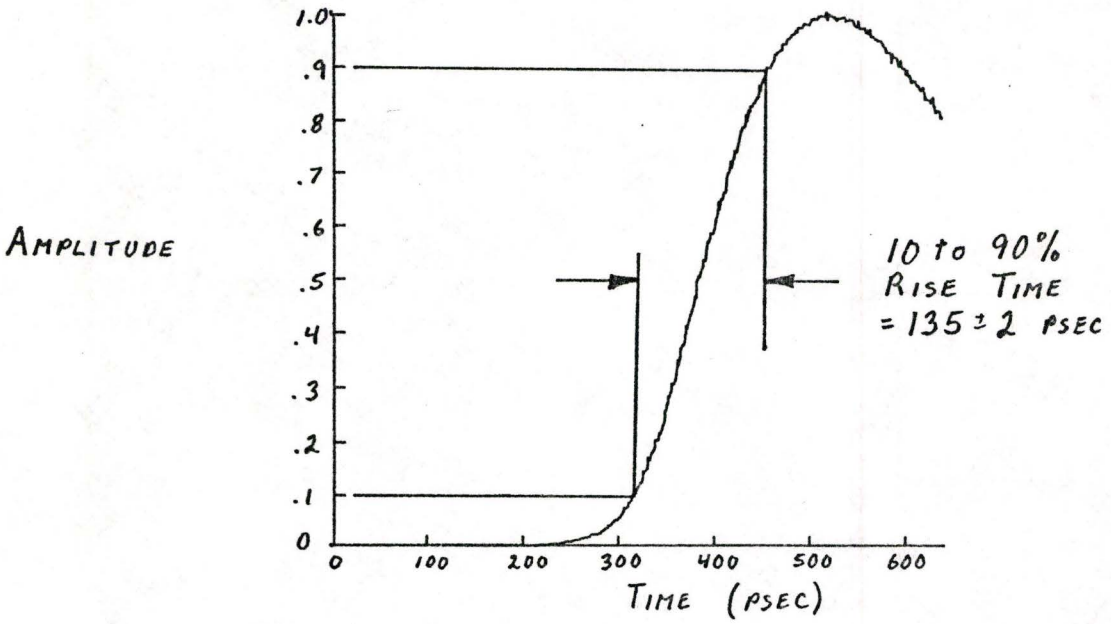

FIGURE 3.5

SIGNAL AFTER 11.4 m OF FIBRE

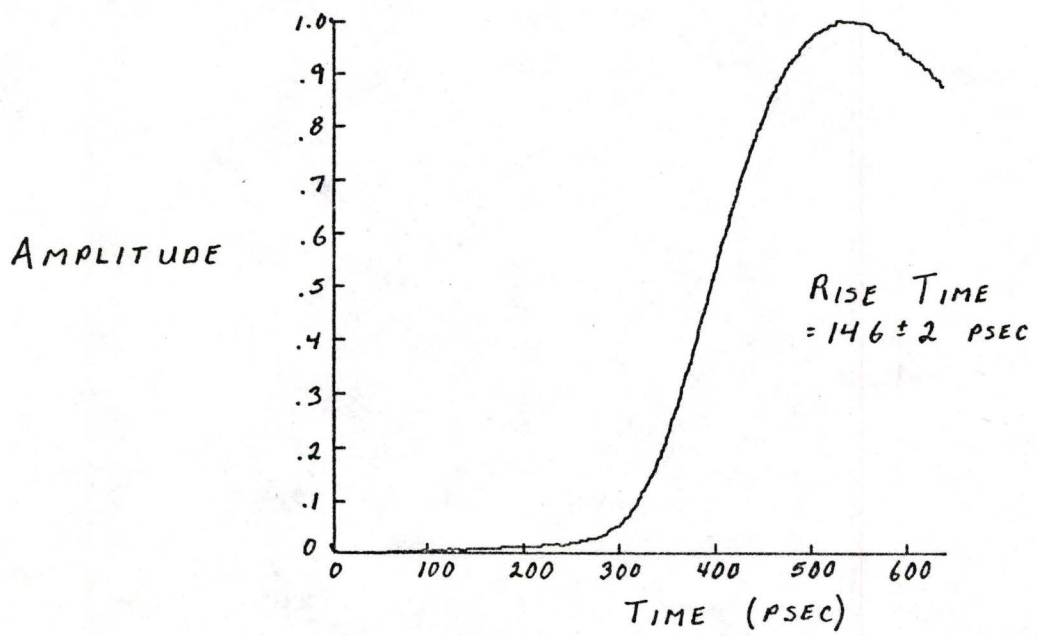

FIGURE 3.6

SIGNAL AFTER 20.7 m OF FIBRE

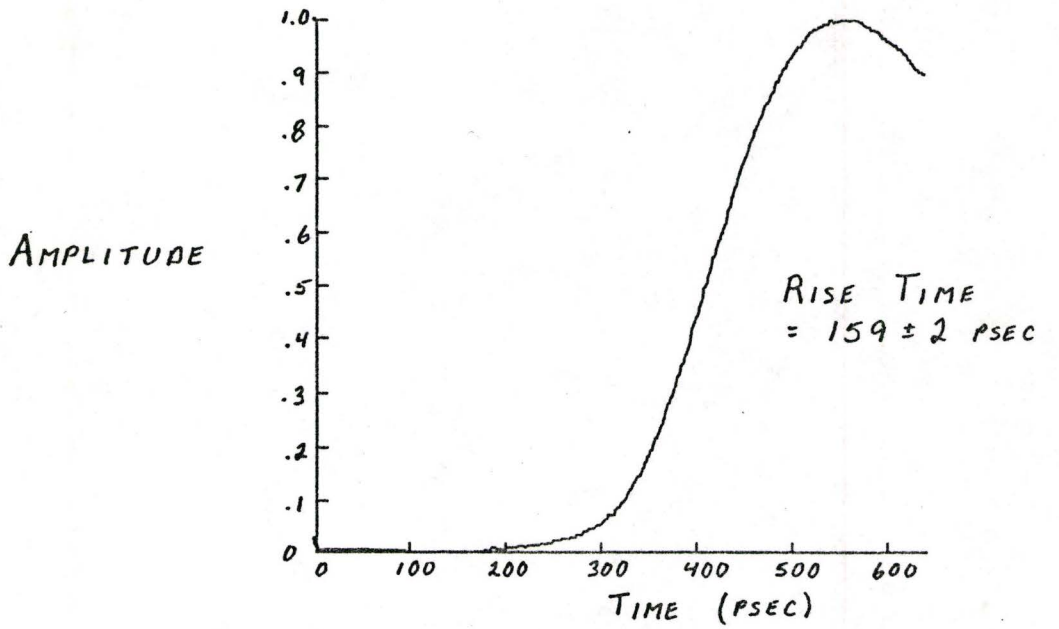

FIGURE 3.7

SIGNAL AFTER 37.0 m OF FIBRE

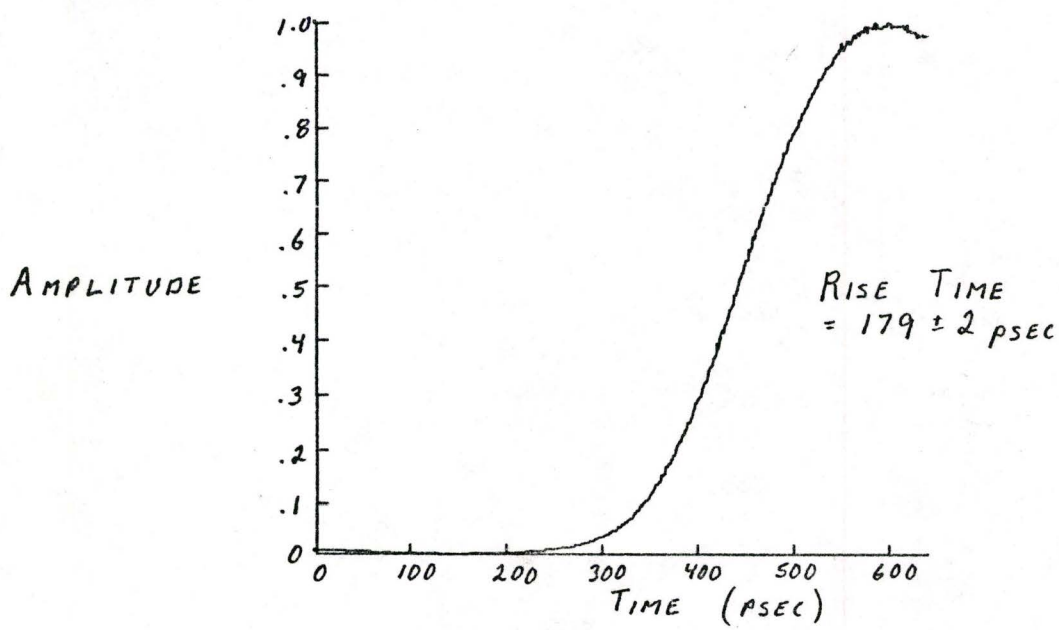

FIGURE 3.8

SIGNAL AFTER 50.9 m OF FIBRE

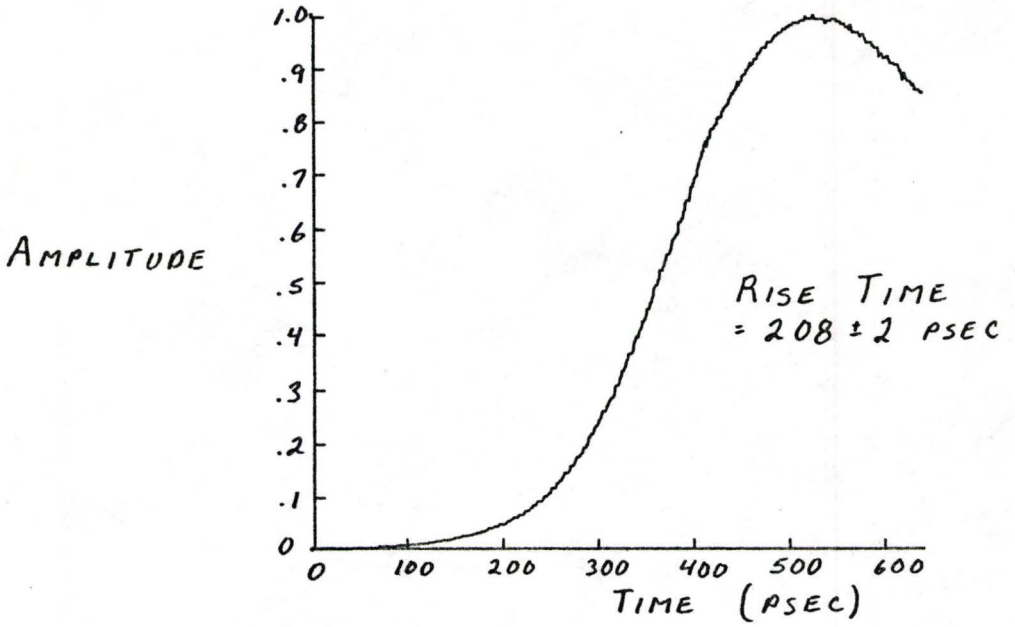

FIGURE 3.9

SIGNAL AFTER 87.9 m OF FIBRE

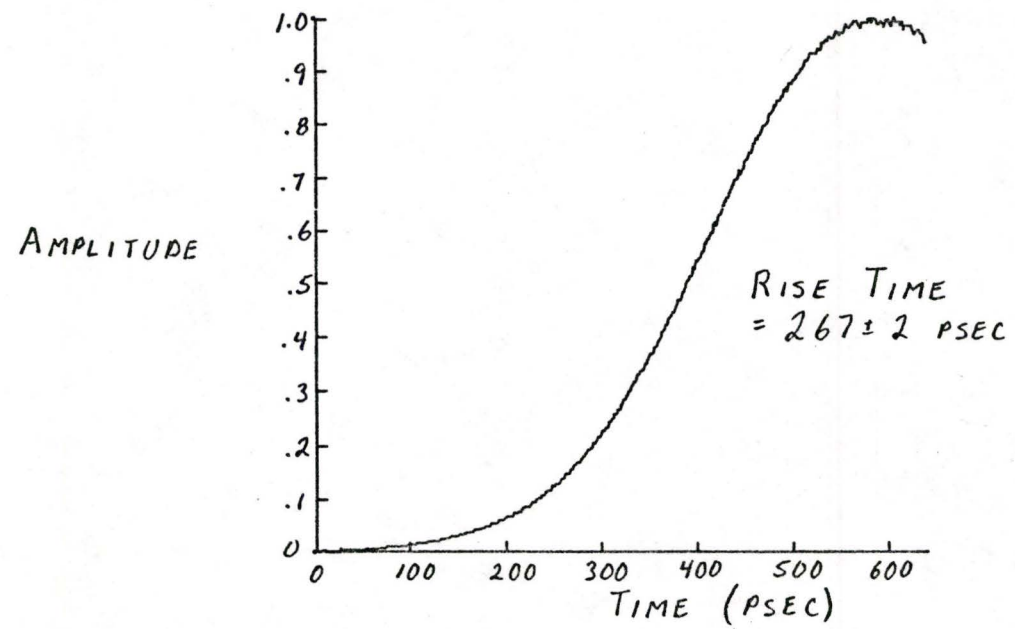

FIGURE 3.10

Undoubtably, some of these errors arose simply because the equipment was unfamiliar, and it did take time to develop satisfactory experimental techniques. Had the fibre been available for a longer period of time, these techniques could have been used to make more accurate measurements. The following points were noted to be important for obtaining consistent results.

1) A 10 to 12 hour period of time in the lab should have been arranged, during which all measurements should be made. This would help ensure that the operating conditions of the laser remained unchanged.

2) Instead of making measurements on a certain fibre, removing it from the laser, and then realigning another one, there should be one short length of fibre, (ie. about 2 m), permanently aligned with the laser. When a longer length had to be used, it could simply be coupled to the short length. The reasoning behind this procedure is that each time a fibre is cleaved, the end will have a different contour. (ie. a cleaved fibre does not have a flat end, but also has irregularities such as cracks, chips, etc.) These irregularities, plus the fact that the emitting region of the laser is only a few microns wide compared with a fibre

core diameter of 60 microns, makes it very difficult to launch the light consistently into each mode. This "random mode coupling" would affect the measured dispersion, because each mode travels at a different speed.

However, an earlier attempt to minimize this "random mode coupling" was made by trying to remove the cladding modes from both the beginning and end of the fibre. The fibre was stripped of its protective plastic coating, and about 8 cm of the fibre at each end was immersed in glycerin. This procedure produced no measurable change from the case where the fibre ends were not immersed in glycerin. Hence, either the launching of the light into the fibre is not critical, or else it is critical because of large differences in coupling between the lower order modes. (ie. the cladding modes appear to be highly attenuated.)

3) After measuring the rise time through a long length of fibre, it would be useful to remeasure the rise time through the short length that was permanently aligned with the laser. This would serve as an additional check on whether or not the laser or detector had changed.

4) Better results could also be obtained if one could start with a single long length of fibre and cut it into smaller pieces as needed. In this experiment, several different lengths were used, and although they supposedly all had the same specifications, two of the seven fibre pieces available had responses so different, that they were thought to have in fact been different kinds of fibres. This cast a shadow of uncertainty upon the uniformity of the remaining pieces.

Although time did not permit the testing of these techniques, it is hoped they will prove useful for other users of this and similar systems.

During one day, all of the necessary measurements were made, and they displayed very nicely the expected behavior, (see figure 3.11 for a graph of 10 to 90% rise times versus fibre length). However, the combined results of all ten days were not as well-behaved, as can be seen in figure 3.12.

Because the longest length of fibre available barely doubled the rise time of the input pulse, it was difficult to extract the fibre response from the graph of rise time versus fibre length. Ideally, one would

RISE TIME VERSUS FIBRE LENGTH FOR A SINGLE DAY

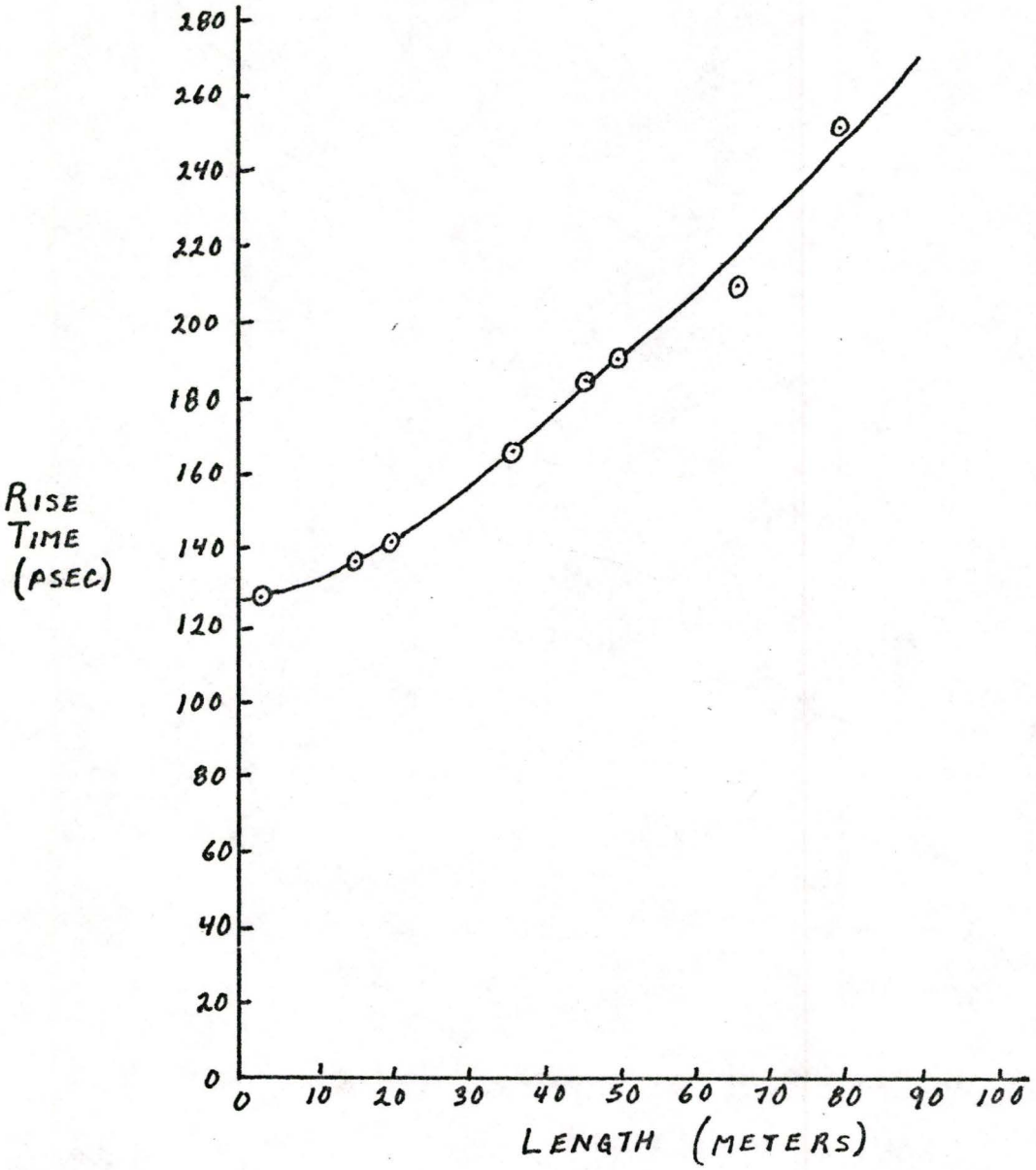

FIGURE 3.11

RISE TIME VERSUS FIBRE LENGTH FOR TEN DAYS

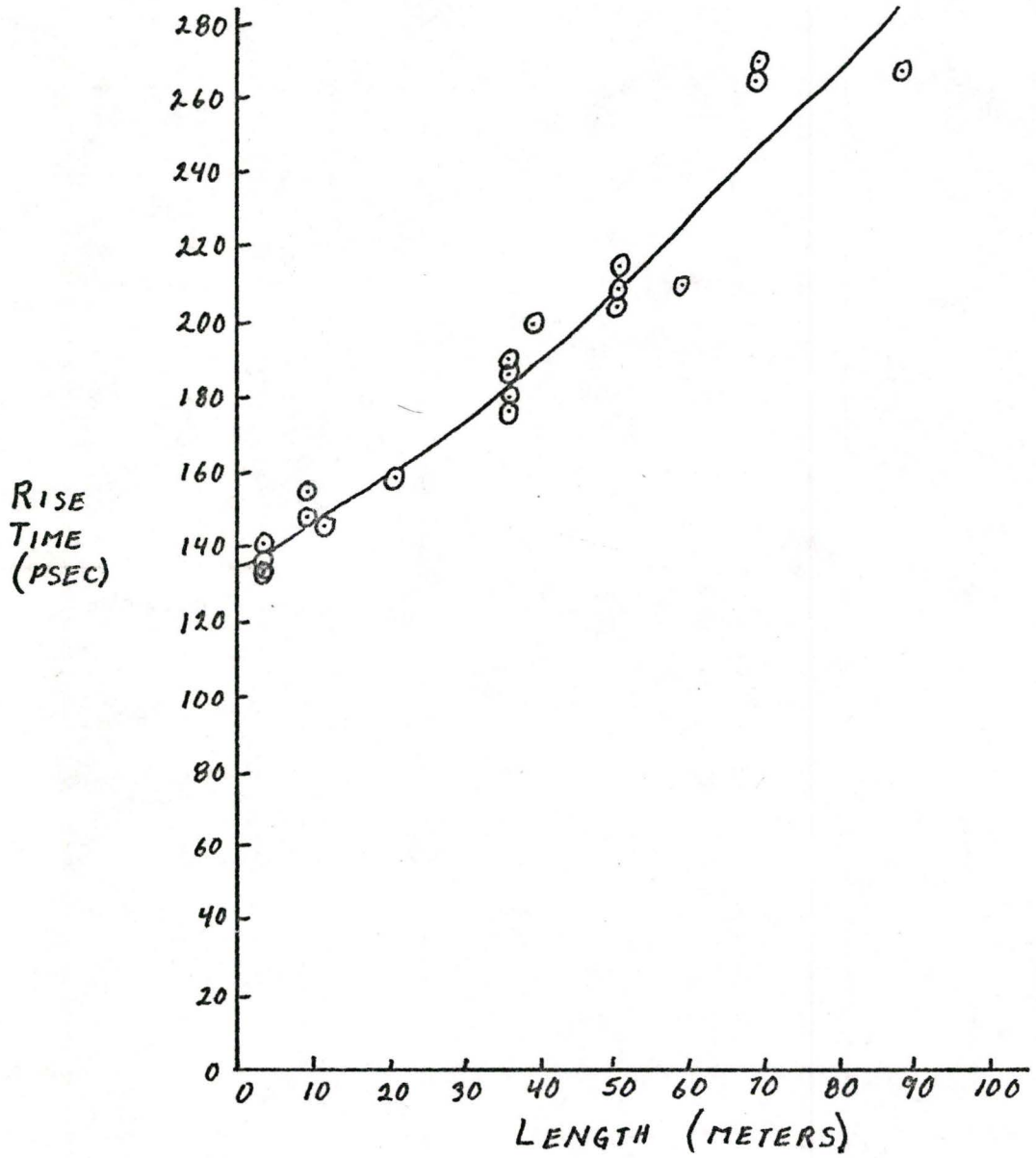

FIGURE 3.12

like to have a length of fibre that is long enough to increase the rise time of the input pulse by a factor of 3 or more. Roughly speaking, if

t_i is the rise time of the input pulse to the detector system in psec

R_f is the response of the fibre in psec/meter

L is the length of the fibre in meters

t_o is the rise time of the output pulse in psec

then

$$t_o = \sqrt{t_i^2 + (R_f L)^2} \ .$$

Notice that if $R_f L$ is much greater than t_i, then t_i can be ignored and the response of the fibre, R_f, is simply the slope of the graph of t_o versus L, for large L.

However, since long lengths of fibre were not available, the next best thing would have been to deconvolute the output pulse from a long length of fibre with that of a short length of fibre, thus obtaining the response function for the length equal to the difference of the two lengths. But exact deconvolution is a nontrivial task, and the experiment did not warrant such a sophisticated analysis. Instead, since the input function was available, (ie. figure 3.5), an approximate technique was employed in which a response function was guessed at, and the two were convoluted. If the convoluted result was similar to the measured results, (ie. figures 3.6

to 3.10), then the guessed response function must have been close to the actual response of the fibre.

Because nature seems to be partial towards Gaussians and Lorentzians, these two functions were used as the guessed response functions, to which the input function would be convoluted. The limiting behavior of each function could then be observed, and compared with actual results. The forms of these two equations were:

Gaussian $$A = \exp(-\tfrac{1}{2}(\frac{t-t_m}{\Delta t})^2) \qquad 3.1$$

Lorentzian $$A = \frac{(\Delta t/2)^2}{(t-t_m)^2 + (\Delta t/2)^2} \qquad 3.2$$

where A = amplitude

t = time

t_m = time at the middle of the trace (ie. 320 psec)

Δt = parameter that determines the width of the response function.

In figure 3.13 is a graph showing the Gaussian of equation 3.1, for Δt = 20 psec, and in figure 3.14 is a graph showing the convolution of this Gaussian with the input pulse in figure 3.5. Figures 3.15 to 3.18 are similar, except that Δt has been changed to 60 psec and

GAUSSIAN WITH Δt = 20 PSEC

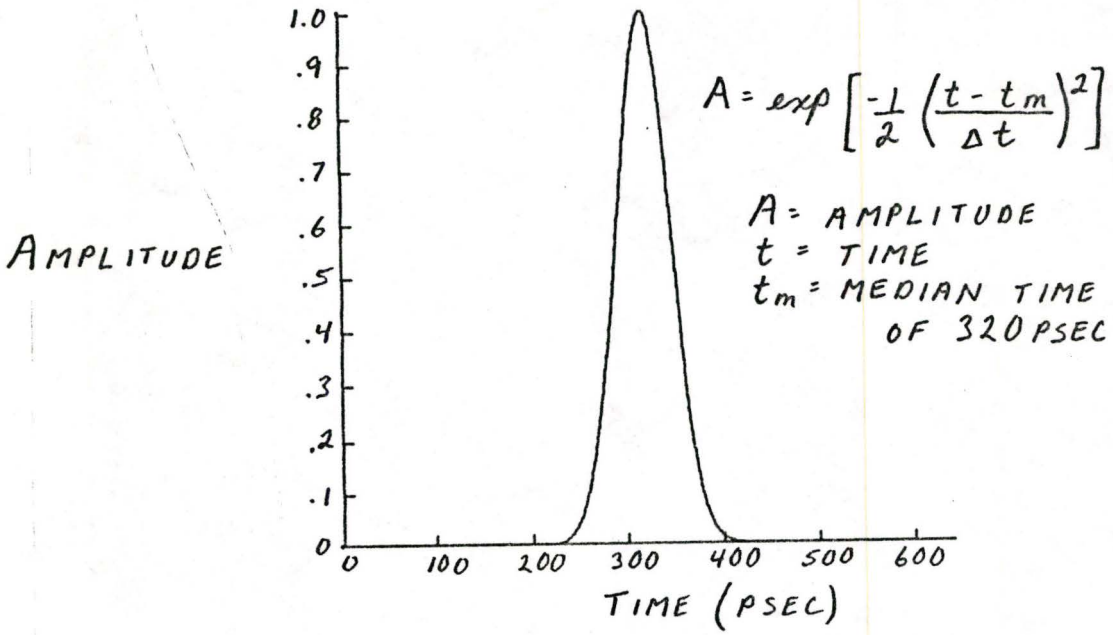

FIGURE 3.13

CONVOLUTED OUTPUT FOR GAUSSIAN FIBRE RESPONSE WITH Δt = 20 PSEC

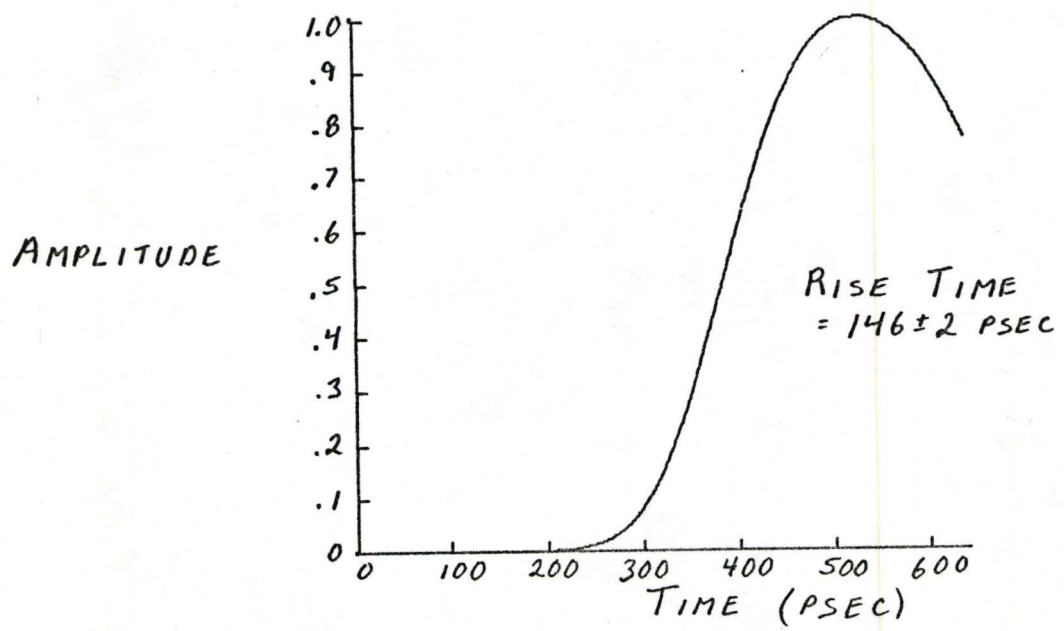

FIGURE 3.14

GAUSSIAN WITH $\Delta t = 60$ PSEC

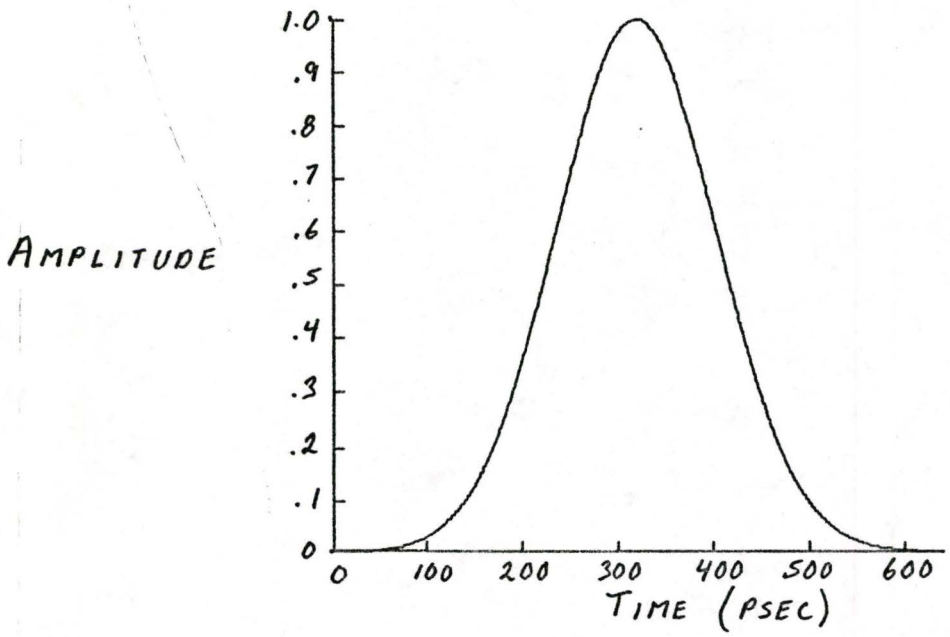

FIGURE 3.15

CONVOLUTED OUTPUT FOR GAUSSIAN FIBRE RESPONSE WITH $\Delta t = 60$ PSEC

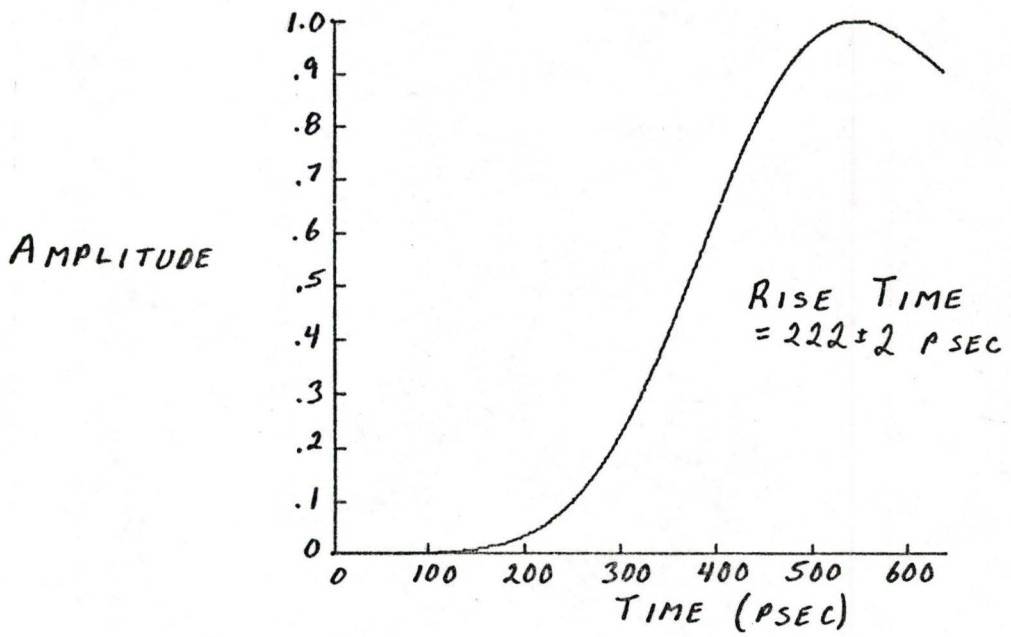

FIGURE 3.16

GAUSSIAN WITH Δt = 100 PSEC

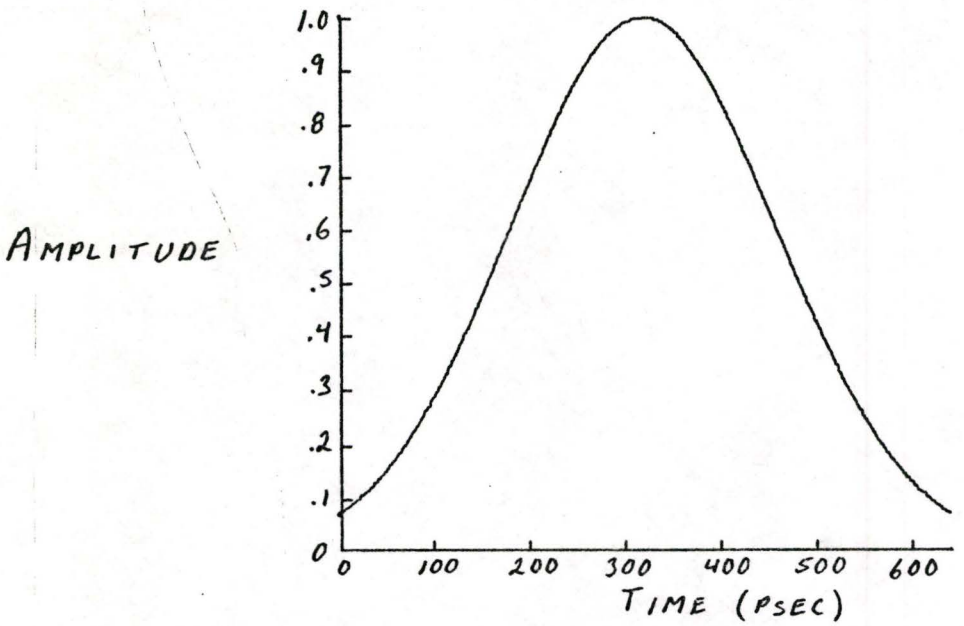

FIGURE 3.17

CONVOLUTED OUTPUT FOR GAUSSIAN FIBRE RESPONSE WITH Δt = 100 PSEC

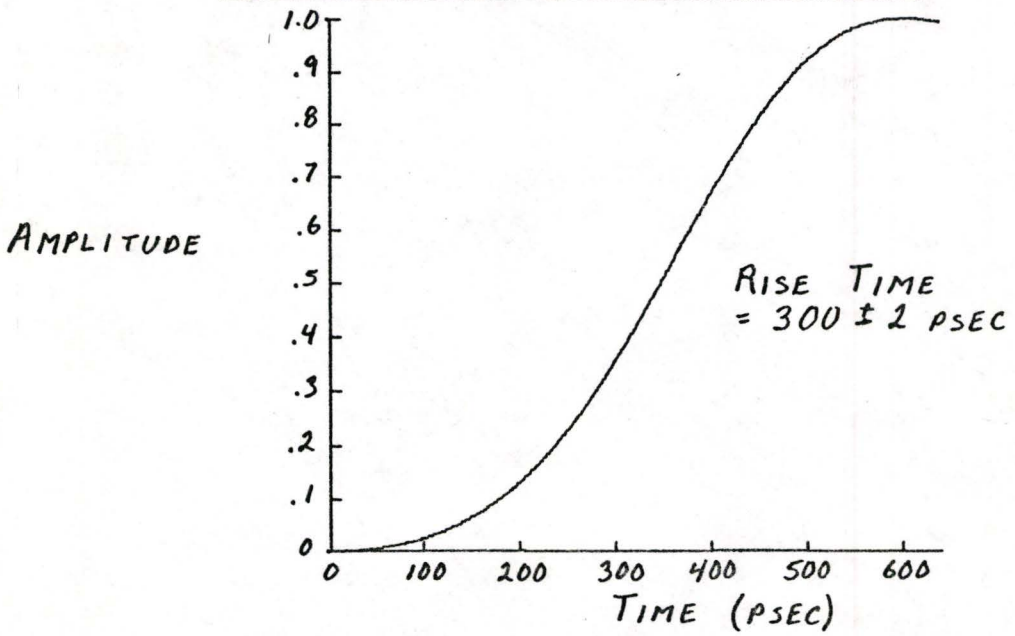

FIGURE 3.18

100 psec respectively. Figures 3.19 and 3.20 show composite graphs of the Gaussians and the convoluted outputs for several Δt. As expected, and as indicated by the figures, the rise time of the convoluted outputs increases as Δt is increased.

A similar set of graphs is shown in figures 3.21 to 3.28, this time for the Lorentzian fibre response.

The method used to determine which function, the Gaussian or the Lorentzian, more closely approximated the actual fibre response, was to superimpose the convoluted outputs of figures 3.13 to 3.28 with the measured outputs of figures 3.6 to 3.10. This was done for as many of the figures as possible, and figures 3.29 and 3.30 are samples, characteristic of the results observed.

When figure 3.7 of the actual output was superimposed with the Gaussian convoluted outputs, it was found to be almost identical to the one where Δt = 40 psec. In figure 3.29, the dotted line represents the actual output, and the solid line represents the convoluted output. As can be seen, the two lines are virtually on top of each other.

When figure 3.7 was compared with the Lorentzian

SUPERIMPOSED GAUSSIANS OF DIFFERENT Δt

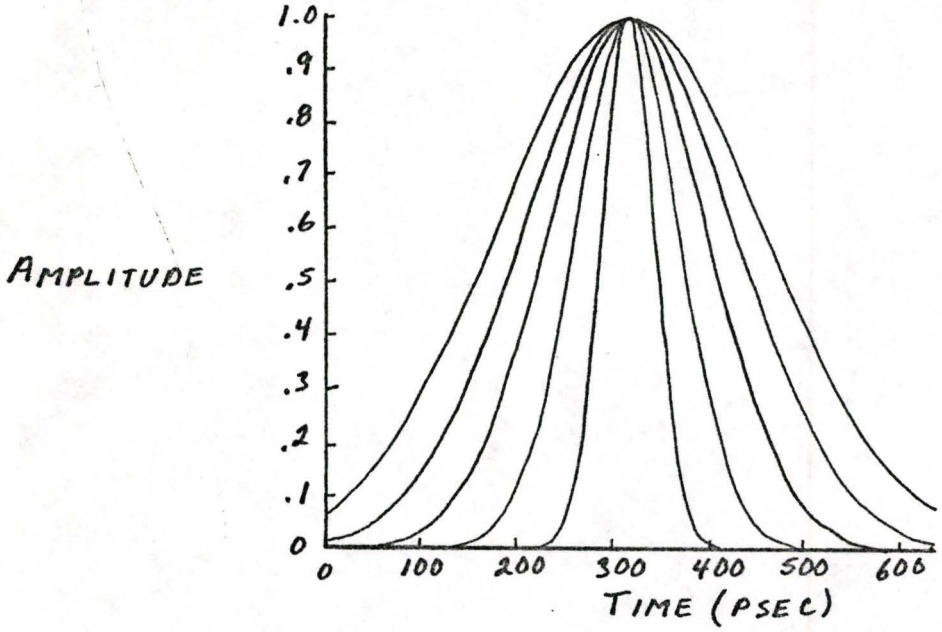

FIGURE 3.19

SUPERIMPOSED CONVOLUTED OUTPUTS FOR GAUSSIAN RESPONSES

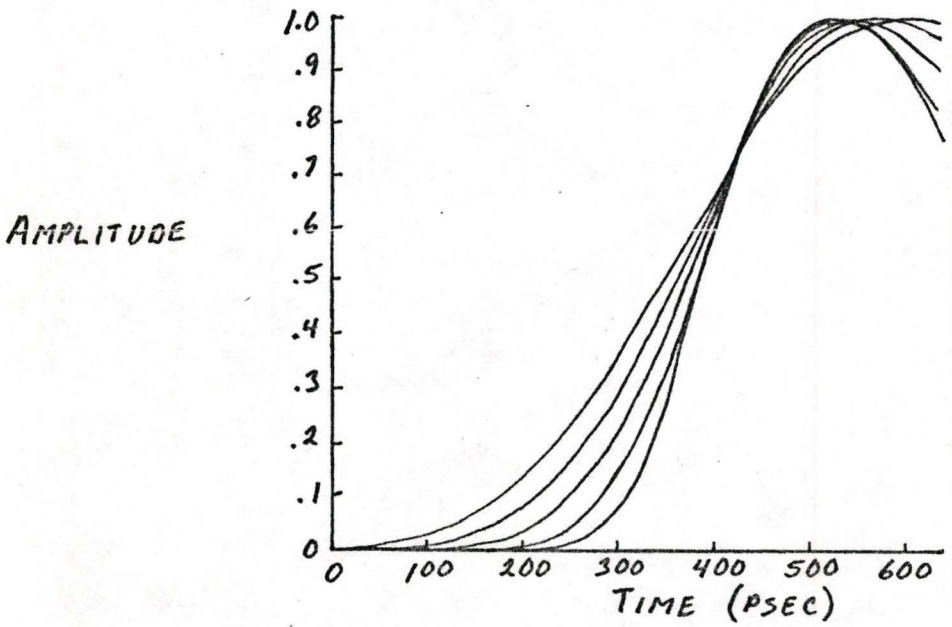

FIGURE 3.20

LORENTZIAN WITH Δt = 20 PSEC

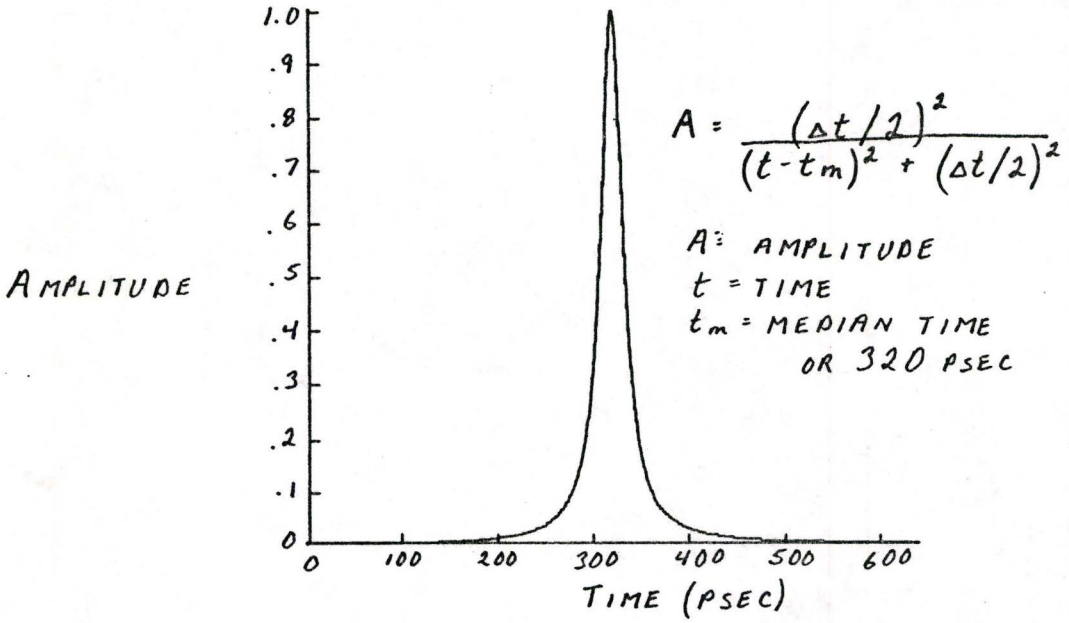

FIGURE 3.21

CONVOLUTED OUTPUT FOR LORENTZIAN FIBRE RESPONSE WITH Δt = 20 PSEC

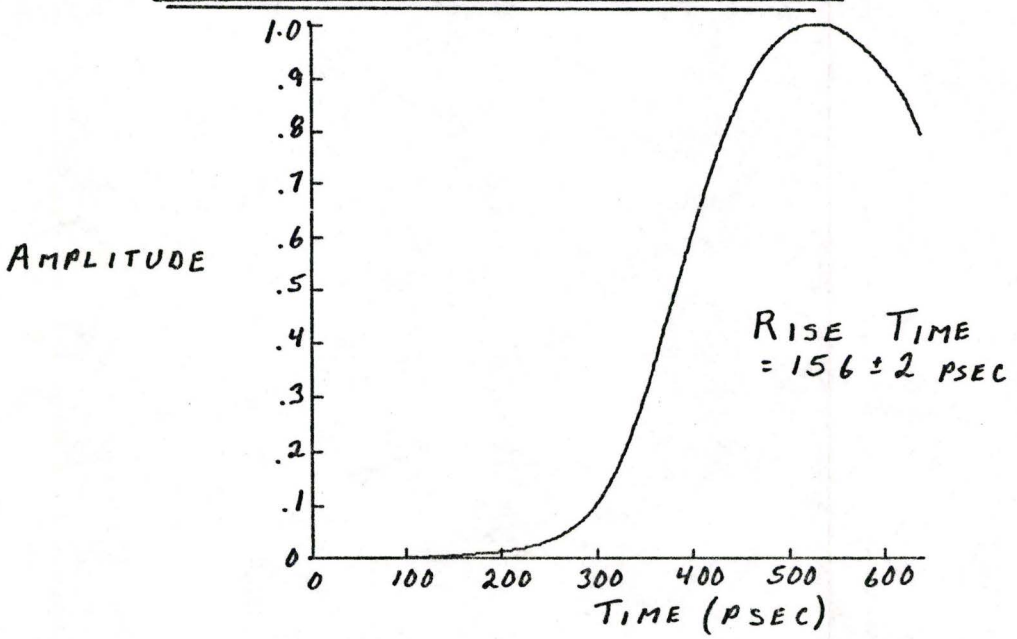

FIGURE 3.22

LORENTZIAN WITH Δt = 60 PSEC

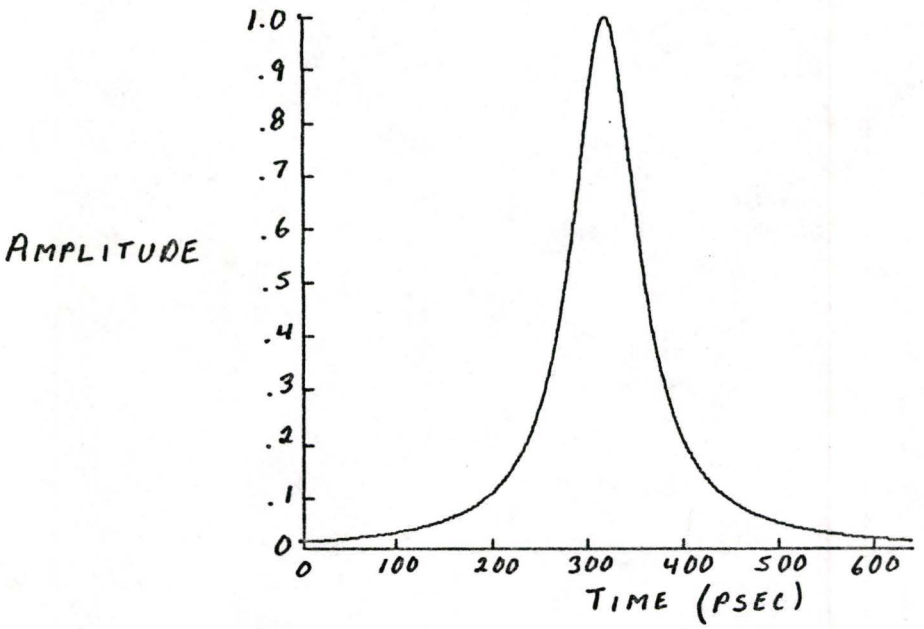

FIGURE 3.23

CONVOLUTED OUTPUT FOR LORENTZIAN FIBRE RESPONSE WITH Δt = 60 PSEC

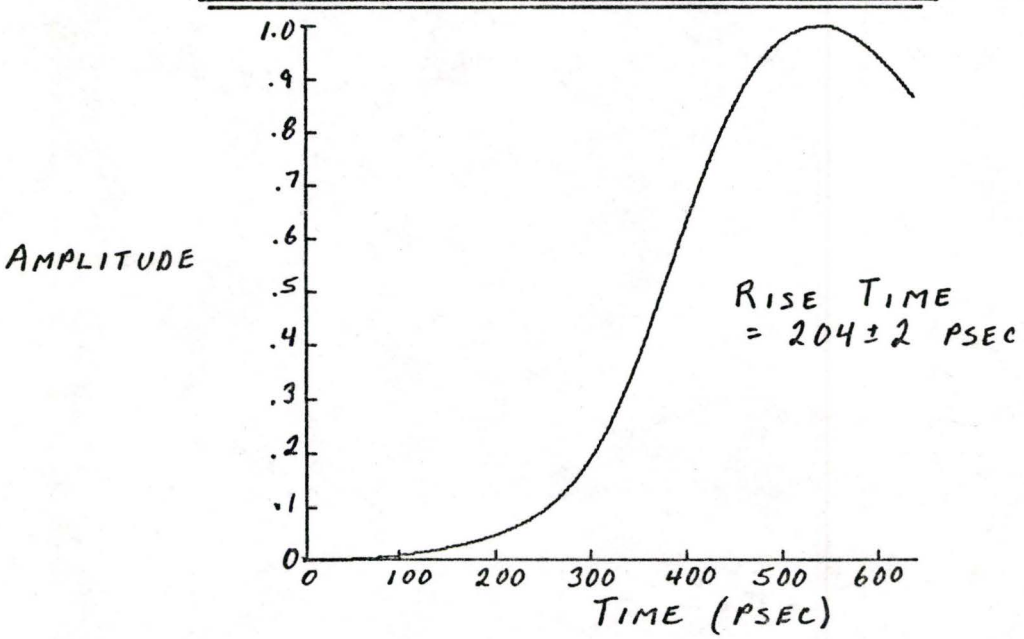

FIGURE 3.24

LORENTZIAN WITH $\Delta t = 100$ PSEC

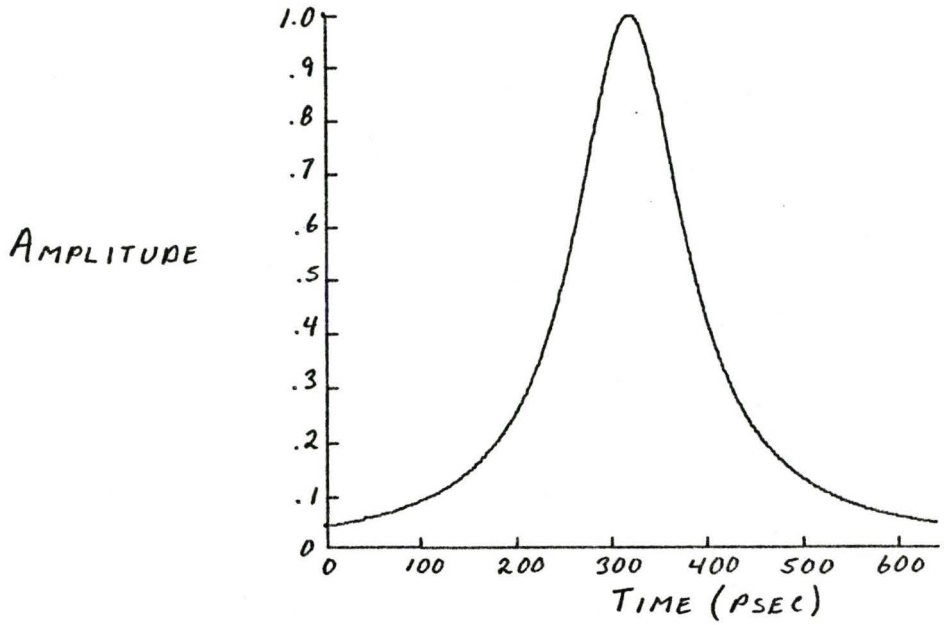

FIGURE 3.25

CONVOLUTED OUTPUT FOR LORENTZIAN FIBRE RESPONSE WITH $\Delta t = 100$ PSEC

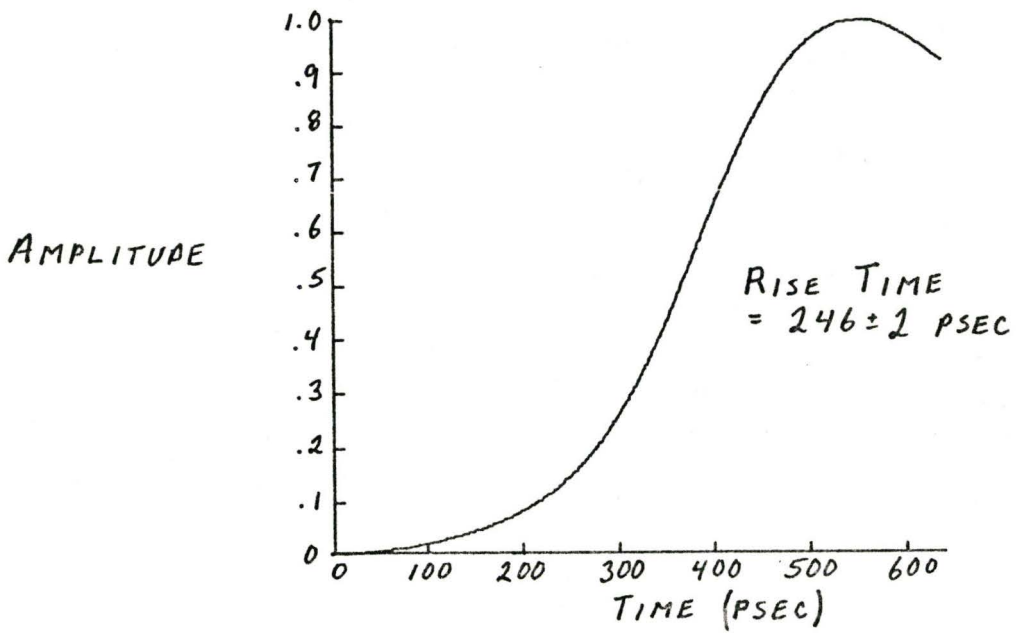

FIGURE 3.26

SUPERIMPOSED LORENTZIAN OF DIFFERENT Δt

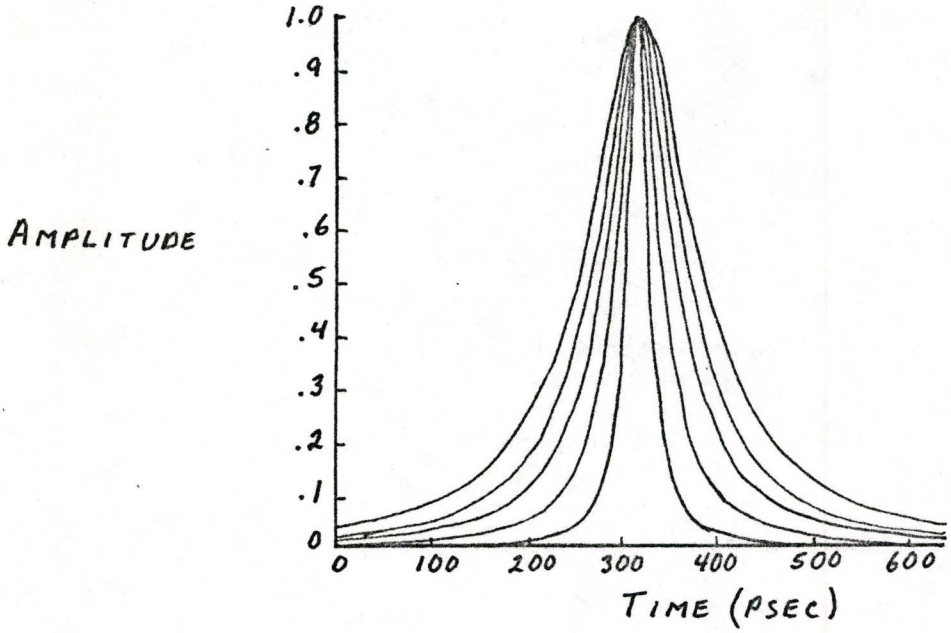

FIGURE 3.27

SUPERIMPOSED CONVOLUTED OUTPUTS FOR LORENTZIAN RESPONSES

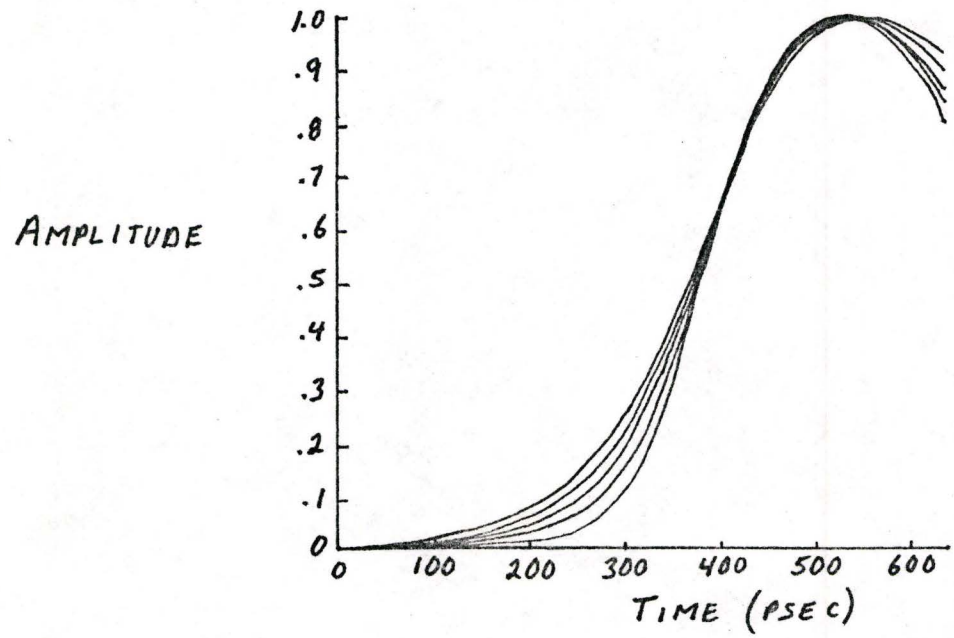

FIGURE 3.28

MEASURED OUTPUT COMPARED WITH
GAUSSIAN CONVOLUTED OUTPUT

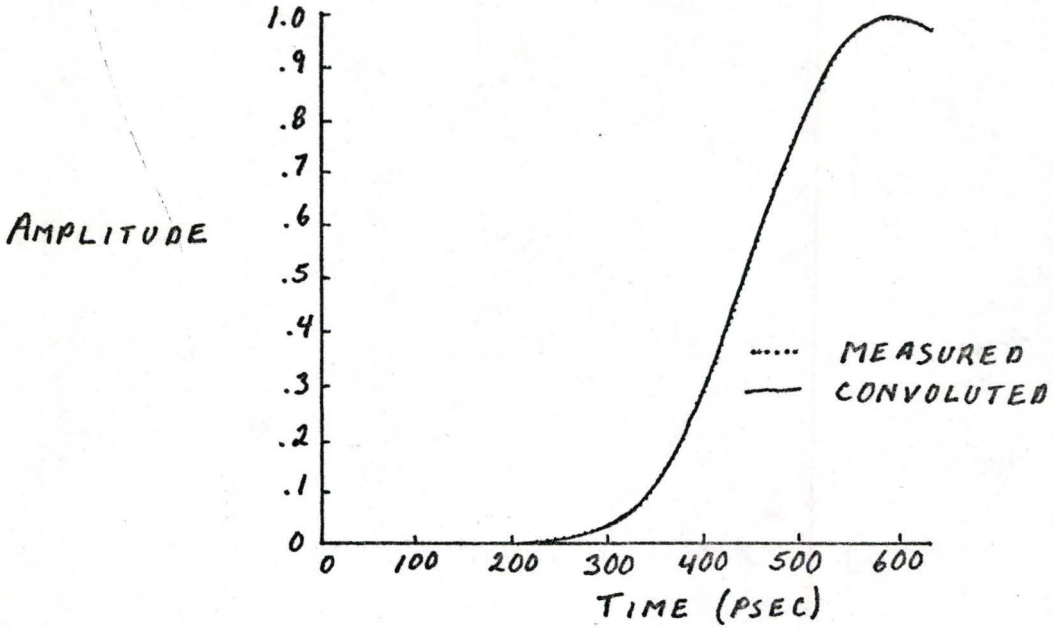

FIGURE 3.29

MEASURED OUTPUT COMPARED WITH
LORENTZIAN CONVOLUTED OUTPUT

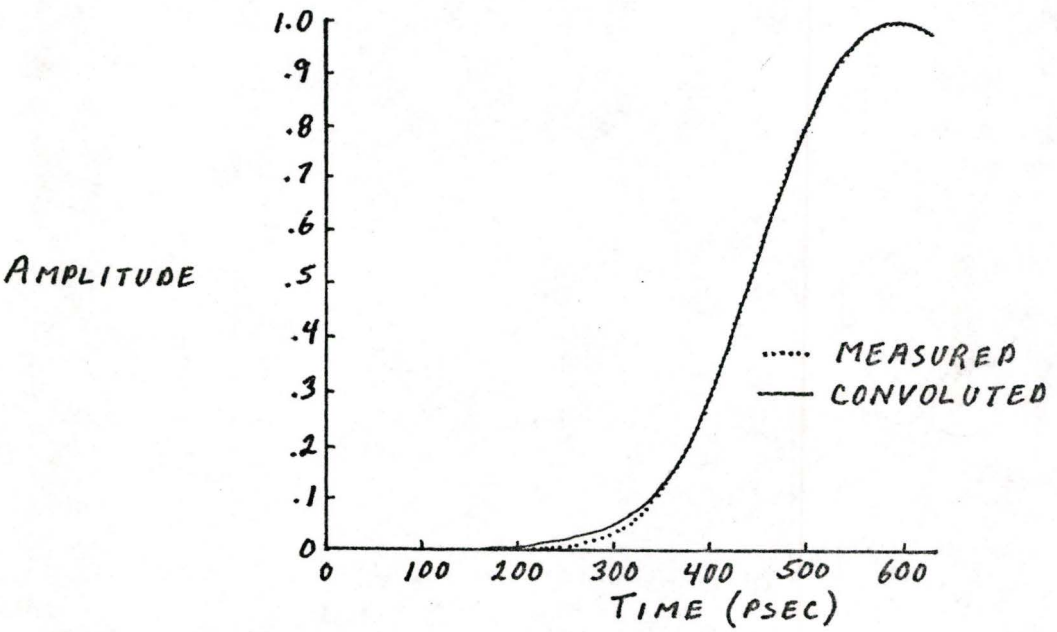

FIGURE 3.30

convoluted outputs, it was found to most closely match the one where Δt was also 40 psec. But the Lorentzian convoluted output exhibited a longer rising edge than the actual output, as shown in figure 3.30. This behavior was fairly characteristic of the Lorentzian convoluted outputs, so the fibre response was assumed to approximate a Gaussian.

In continuing with the analysis, a graph was drawn of the rise times of the Gaussian convoluted outputs versus Δt, as shown in figure 3.31. Thus, knowing the actual rise time of a pulse after coming through a certain fibre, this graph could be used to find the Δt corresponding to the equivilent Gaussian response of that fibre. This was done for both plots of rise time versus fibre length shown in figures 3.11 and 3.12, and plots of Gaussian Δt versus fibre length were constructed, as in figures 3.32 and 3.33.

From these plots it is very easy to extract the fibre response as a function of length. Both of these plots are approximately linear, although figure 3.33 shows more scatter; probably as a result of combining all the measurements from a number of different days. Using a linear regression to determine the best straight line fit, one sees that this fibre exhibits a Gaussian response

RISE TIME OF GAUSSIAN CONVOLUTED OUTPUTS VERSUS Δt

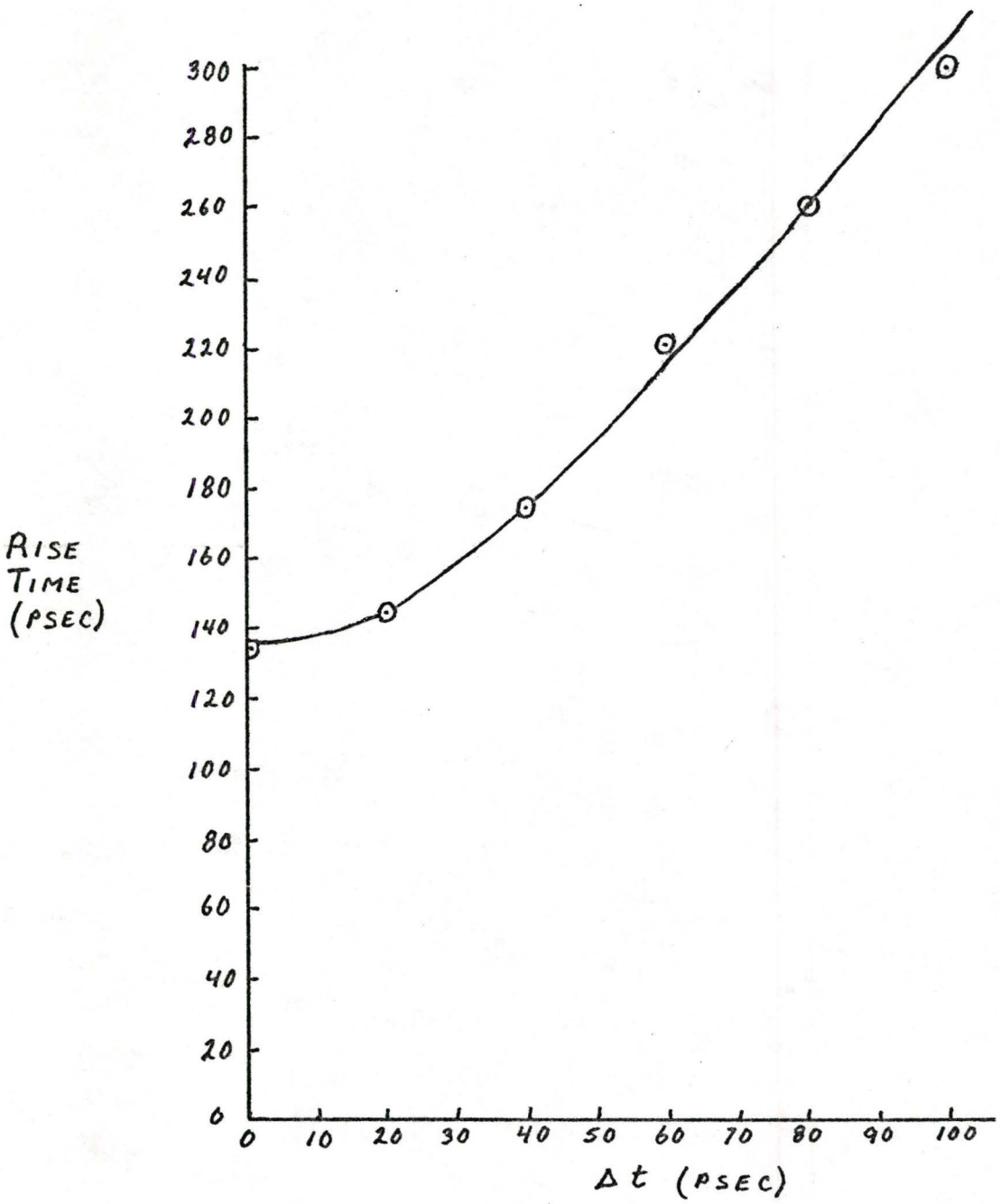

FIGURE 3.31

GAUSSIAN Δt VERSUS FIBRE LENGTH
FOR SINGLE DAY

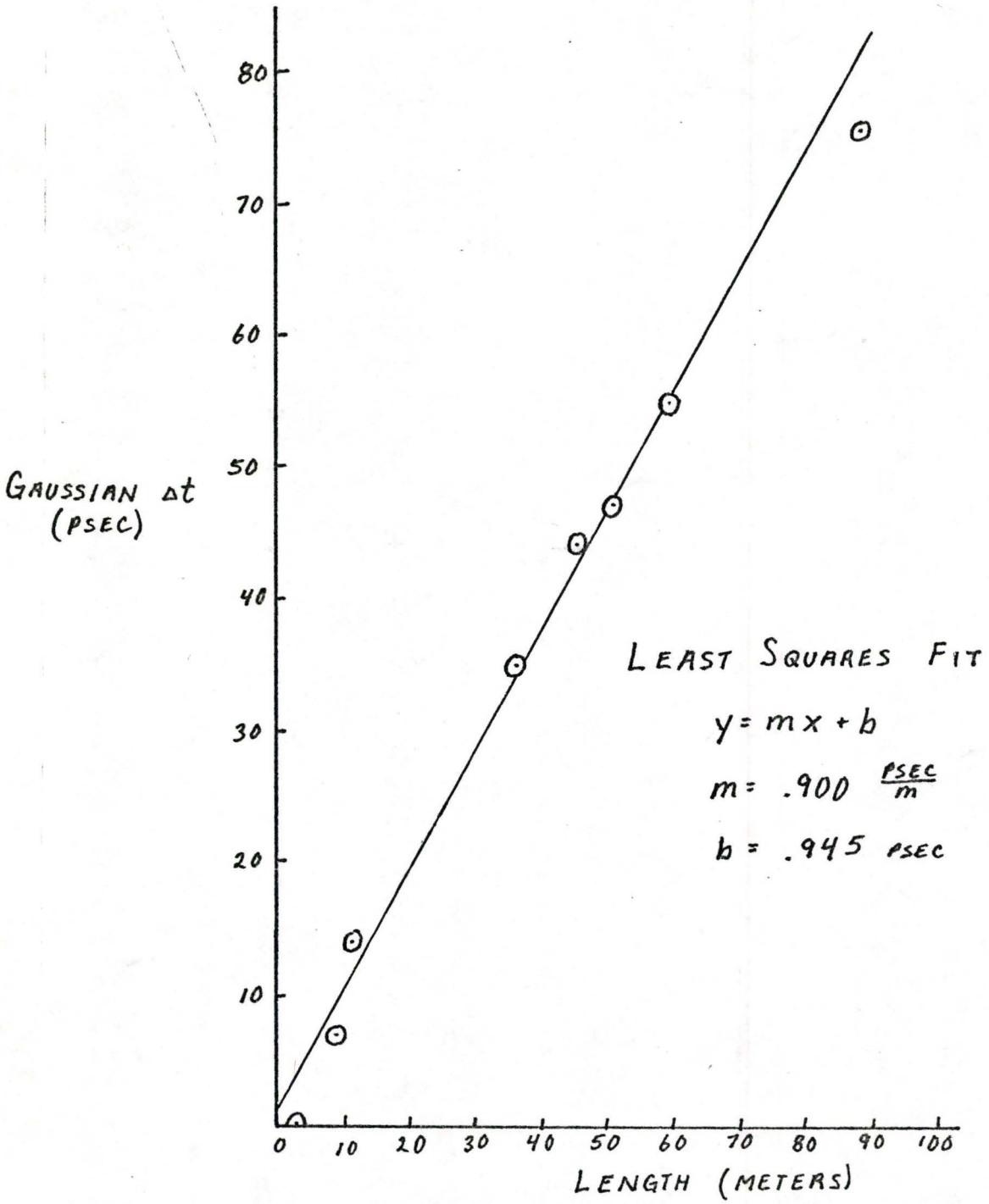

FIGURE 3.32

GAUSSIAN Δt VERSUS FIBRE LENGTH
FOR TEN DAYS

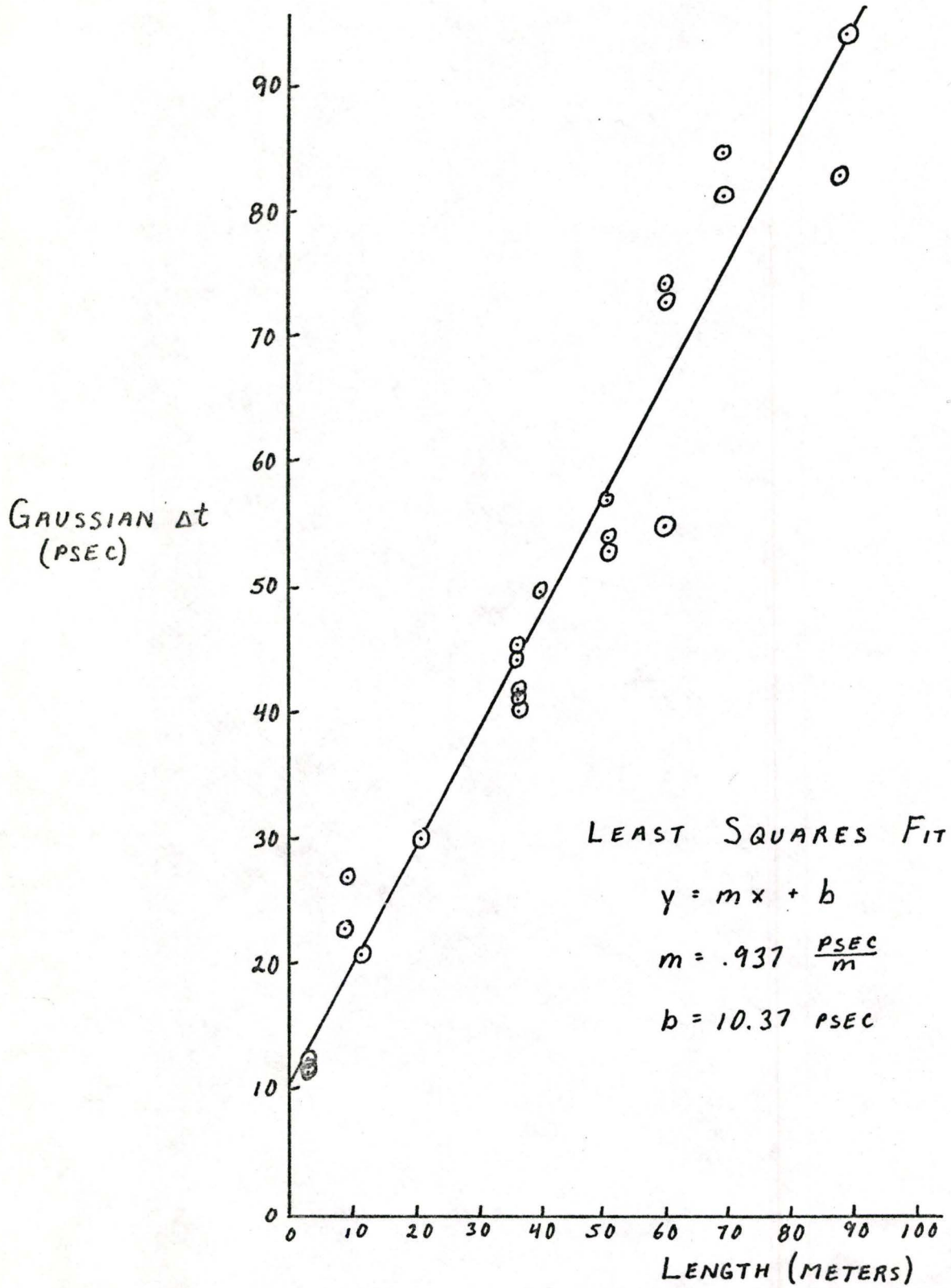

FIGURE 3.33

with a Δt of .900 psec/m for figure 3.32 and .937 psec/m for figure 3.33.

Using the parameters of figure 3.32, one would predict that a 1 km length of this fibre would have a Gaussian response function with a Δt of 900 psec. If the pulse of figure 3.5 were launched into this 1 km length, it would be almost a unit step function, because the response of the fibre is so much slower. Then the leading edge of the output pulse would simply be the leading edge of the Gaussian of equation 3.1. Using equation 3.1,

$$A = \exp(-\tfrac{1}{2}(\frac{t-tm}{\Delta t})^2) \qquad 3.1$$

and a Δt of 900 psec, we can now solve for the 10 to 90% rise time of the output pulse. ie. Assume t_m = 0 psec. Then

$$t = \pm \sqrt{-2 \ln(A)}\ \Delta t \qquad 3.3$$

$$t_{10\%} = -\sqrt{-2 \ln(.1)} * 900 \text{ psec}$$
$$= -1930 \text{ psec}$$

$$t_{90\%} = -\sqrt{-2 \ln(.9)} * 900 \text{ psec}$$
$$= -190 \text{ psec}$$

Hence, the rise time of the pulse in figure 3.5 after going

through 1 km of this fibre would be

$$t_{90\%} - t_{10\%} = (-190 \text{ psec}) - (-1930 \text{ psec})$$
$$= 1740 \text{ psec}$$
$$= 1.74 \text{ nsec} .$$

4. CONCLUSIONS

The object of this project was to develop the software for an existing signal averaging system. The usefulness of the system was then to be demonstrated by measuring the rise time of short laser pulse after transmittal through various lengths of fibre, and hence extract the fibre response as a function of length.

In developing the software, four generations of the controlling program evolved, each one containing additional features which would allow it to be used in a wider range of applications. This experiment was done completely with the third generation program called SIG, because the fourth one contained options not needed in this situation. Both programs are listed and described in the appendices.

Initially it was hoped to determine the fibre response as a function of length directly from a graph of rise time versus fibre length, where the rise time refers to the 10 to 90% rise time of a 140 psec laser pulse, after passing through a given length of fibre. However, the longest length of fibre was only able to double the

rise time of the input pulse, whereas it would be necessary for it to increase by a factor of at least three or more to use the graph of rise time versus fibre length directly to determine fibre response.

Nevertheless, one can still determine fibre response by deconvoluting the output of a fibre with it's input. Because of the difficulty of such an operation, it was instead dicided to convolute reasonable fibre response functions with the known input, and compare the result with the output. The two functions chosen were a Gaussian and a Lorentzian. The Gaussian convoluted output proved to be extremely close to the measured output, while the Lorentzian convoluted output displayed a longer rising edge than the measured output.

A graph was made of Δt, (from the equation $A = \exp(-\frac{1}{2}(\frac{t-tm}{\Delta t})^2)$), versus the rise time of the Gaussian convoluted output. Thus, knowing the rise time of the measured output for a given fibre length, a graph was constructed of Δt versus fibre length. This graph was linear and the fibre response for any length of fibre could be determined from the slope of this line.

APPENDIX A - COMPUTER LISTING OF SIG

```
      INTEGER*2 CODE,COL,IRAY(1000,3,3),MODE,NSCNH,NSCNL,PLTFQ,
     +   PLTFQR,SCNMOD,SNSCH(3,2),SNSCL(3,2),SPEED,SUMR,TYPE
      DOUBLE PRECISION DPRAY(465,2),MAX,MIN,SCL,SMAX(3,3)
      REAL*4 TIME(3),WIDTH
      DATA SMAX/9*0.D0/SNSCH,SNSCL/12*0/
      CALL SETUP(NSCNL,NSCNH,SCNMOD,SPEED)
C
C     SETUP IS AN ASSEMBLER ROUTINE THAT STORES ADDRESSES
C     OF ABOVE VARIABLES FOR LATER USE BY ASSEMBLER.
C
      TYPE 10
10    FORMAT ('$CODE? ')
      READ (5,20) CODE
20    FORMAT (5I6)
      IF ( CODE .EQ. 0 )  GOTO 40
      TYPE 30
30    FORMAT (' 0 = NO.'/' 1 = YES.'/
     +    ' COL1,COL2,COL3 = AVAILABLE COLUMNS.'/
     +    ' DPH = DIFFERENCE PULSE HEIGHT.'/
     +    ' LP = LOWER PEN ON PLOTTER.'/
     +    ' MODE1 = SIGNAL AVERAGE.'/
     +    ' MODE2 = DISPLAY ON OSCILLOSCOPE OR PLOTTER.'/
     +    ' MODE3 = OUTPUT DATA ONTO PUNCHED TAPE.'/
     +    ' MODE4 = INPUT DATA FROM PUNCHED TAPE.'/
     +    ' MODE5 = QUIT.'/
     +    ' NSCNH = HIGH WORD SPECIFYING # OF SCANS TO BE SAMPLED.'/
     +    ' NSCNL = LOW WORD SPECIFYING # OF SCANS TO BE SAMPLED.'/
     +    ' PLTFQ = FREQUENCY OF POINTS USED IN PLOTTING BORDER.'/
     +    ' PLTFQR = REDUCED PLOTTING FREQUENCY FOR PLOTTING SCALES.'/
     +    ' RP = RAISE PEN ON PLOTTER.'/
     +    ' SCNMOD0 = SINGLE SWEEP FOR PLOTTER.'/
     +    ' SCNMOD1 = CONTINUOUS SWEEP FOR OSCILLOSCOPE.'/
     +    ' SUMR0 = DO NOT PRINT SUMMARY OF RESULTS.'/
     +    ' SUMR1 = PRINT SUMMARY OF RESULTS.'/
     +    ' TYPE1 = SIGNAL.'/
     +    ' TYPE2 = NOISE.'/
     +    ' TYPE3 = SIGNAL MINUS NOISE.'/
     +    ' WIDTH = WIDTH OF SCALE MARKINGS ON BORDER OF PLOT.'/
     +    ' 0 = CHARACTER USED TO STOP SAMPLING OR DISPLAYING.'/)
40    TYPE 50
50    FORMAT ('$MODE? ')
      READ (5,20) MODE
      GOTO (60,210,340,370,500)  MODE
C
C
C     MODE1 - SIGNAL AVERAGE.
C     ***********************
C
60    TYPE 70
70    FORMAT ('$COL,TYPE,NSCNH,NSCNL? ')
      READ (5,20) COL,TYPE,NSCNH,NSCNL
      CALL SIGAV(IRAY(1,COL,TYPE))
C
C     SIGAV IS AN ASSEMBLER ROUTINE THAT TAKES AN ADC
C     READING FROM SCOPE, ADDS THIS VALUE TO IRAY(1,COL,
C     TYPE), AND ADDS THE CARRY BIT TO IRAY(2,COL,TYPE),
C     ETC., TILL 461 READINGS HAVE BEEN TAKEN.  THEN IT
C     DOES ANOTHER SCAN, ETC.
C
```

```
      SNSCH(COL,TYPE) = NSCNH
      SNSCL(COL,TYPE) = NSCNL
      MAX = 0D0
      MIN = 1D38
      DO 110 J=9,929,2
         IF (IRAY(J,COL,TYPE).LT.0) GOTO 90
         DPRAY((J+1)/2,TYPE) = IRAY(J+1,COL,TYPE)*65536D0
     +      +IRAY(J,COL,TYPE)
         GOTO 100
90       DPRAY((J+1)/2,TYPE) = (IRAY(J+1,COL,TYPE)+1)*65536D0
     +      +IRAY(J,COL,TYPE)
100      MAX = DMAX1(MAX,DPRAY((J+1)/2,TYPE))
110      MIN = DMIN1(MIN,DPRAY((J+1)/2,TYPE))
C
C     ABOVE LOOP TRANSFERRED SIGNAL IN IRAY (AN INTEGER
C     MATRIX WHERE EACH DATA POINT NEEDED A LOW AND HIGH
C     ORDER NUMBER) TO DPRAY (A DOUBLE PRECISION MATRIX,
C     WHERE EACH DATA POINT NEEDS ONE NUMBER).  LOOP ALSO
C     FOUND MAX AND MIN FOR LATER USE.
C
      SCL = NSCNH*65536D0+NSCNL
      DO 120 J=5,465,1
120      DPRAY(J,TYPE) = (DPRAY(J,TYPE)-MIN)/SCL
C
C     ABOVE LOOP REMOVED DC OFFSET FROM SIGNAL AND DIVIDED
C     IT BY THE NUMBER OF SCANS.
C
      SMAX(COL,TYPE) = (MAX-MIN)/SCL
      SCL = 511D0/SMAX(COL,TYPE)
      DO 130 J=5,465,1
         IRAY(J,COL,TYPE) = DPRAY(J,TYPE)*SCL
130      IRAY(J,COL,TYPE) = IRAY(J,COL,TYPE)+"142000
C
C     ABOVE LOOP SCALED SIGNAL IN DPRAY TO FIT 10 BIT DAC,
C     PUT IT INTO IRAY AND ADDED VERTICAL DAC ADDRESS.
C
      TYPE 140,NSCNH,NSCNL,SMAX(COL,TYPE)
140   FORMAT (' NSCN=',2I8,5X,'PH=',F8.2)
      IF ( TYPE .EQ. 2 )  GOTO 170
      TYPE 150
150   FORMAT ('$TIME/CM? ')
      READ (5,160) TIME(COL)
160   FORMAT (F10.5)
      GOTO 40
C
C     BOTH SIGNAL AND NOISE SCANS HAVE BEEN TAKEN IF CONTROL
C     GOES TO 170.  PROGRAM NOW CALCULATES SIGNAL-NOISE.
C
170   MAX = -1D38
      MIN = 1D38
      DO 180 J=5,465,1
         DPRAY(J,1) = DPRAY(J,1)-DPRAY(J,2)
         MAX = DMAX1(MAX,DPRAY(J,1))
180      MIN = DMIN1(MIN,DPRAY(J,1))
C
C     DPRAY(J,1) NOW CONTAINS SIGNAL-NOISE.
C
```

```
      SMAX(COL,3) = MAX-MIN
      SCL = 511D0/SMAX(COL,3)
      DO 190 J=5,465,1
         IRAY(J,COL,3) = (DPRAY(J,1)-MIN)*SCL
190      IRAY(J,COL,3) = IRAY(J,COL,3)+"142000
C
C     IRAY(J,COL,3) NOW CONTAINS SIG-NOISE SCALED TO 10 BITS
C     WITH VERTICAL DAC ADDRESS ADDED.
C
      TYPE 200,SMAX(COL,3)
200   FORMAT (' DPH=',F8.2)
      GOTO 40
C
C
C     MODE2 - DISPLAY.
C     ****************
C
210   TYPE 220
220   FORMAT ('$COL,TYPE,SCNMOD,SPEED,SUMR?')
      READ (5,20) COL,TYPE,SCNMOD,SPEED,SUMR
      IF ( SUMR .EQ. 0 )  GOTO 240
      TYPE 230,SNSCH(COL,1),SNSCL(COL,1),SNSCH(COL,2),
     +   SNSCL(COL,2),SMAX(COL,1),SMAX(COL,2),SMAX(COL,3),TIME(COL)
230   FORMAT (20X,'TYPE1',14X,'TYPE2',14X,'TYPE3'/' NSCNH,NSCNL=',
     +   I6,2X,I6,5X,I6,2X,I6/' PH=',F21.2,F19.2,F19.2/
     +   ' TIME/CM=',E9.2/)
240   IF ( SCNMOD .EQ. 1 )  GOTO 330
      TYPE 250
250   FORMAT ('$PLTFQ,PLTFQR,WIDTH?')
      READ (5,260) PLTFQ,PLTFQR,WIDTH
260   FORMAT (2I10,F10.2)
      CALL PLOT(50,511)
      PAUSE 'LP'
      WS1=0.
      DO 290 J=1,PLTFQ
         IWS1=511.-511.*J/PLTFQ
         CALL PLOT(50,IWS1)
         IF ( FLOAT(J)/PLTFQ .LT. WS1 )  GOTO 290
         WS1=WS1+.1
         DO 270 I=1,PLTFQR
            IWS2=50.+WIDTH*I/PLTFQR
            CALL PLOT(IWS2,IWS1)
270         CONTINUE
         DO 280 I=1,PLTFQR
            IWS2=50.+WIDTH-WIDTH*I/PLTFQR
            CALL PLOT(IWS2,IWS1)
280         CONTINUE
290      CONTINUE
      WS1=.15625
      DO 320 J=1,PLTFQ
         IWS1=50.+461.*J/PLTFQ
         CALL PLOT(IWS1,0)
         IF ( FLOAT(J)/PLTFQ .LT. WS1 )  GOTO 320
         WS1=WS1+.15625
         DO 300 I=1,PLTFQR
            IWS2=WIDTH*I/PLTFQR
            CALL PLOT(IWS1,IWS2)
300         CONTINUE
```

```
         DO 310 I=1,PLTFQR
            IWS2=WIDTH-WIDTH*I/PLTFQR
            CALL PLOT(IWS1,IWS2)
310         CONTINUE
320      CONTINUE
      PAUSE 'RP'
      CALL PLOT(50,IRAY(5,COL,TYPE))
      PAUSE 'LP'
330   CALL DISP(IRAY(5,COL,TYPE))
C
C     DISP SIMPLY MOVES INTEGERS STORED IN IRAY TO
C     VERTICAL DAC.
C
      GOTO 40
C
C
C     MODE3 - PUNCH OUTPUT.
C     *********************
C
340   TYPE 350
350   FORMAT ('$COL,TYPE?')
      READ (5,20) COL,TYPE
      DO 355 J=1,25
         CALL PUNCHO(0)
355      CONTINUE
      CALL PUNCHO("177777)
      DO 360 J=5,465
         CALL PUNCHO(IRAY(J,COL,TYPE))
360      CONTINUE
      GOTO 40
C
C
C     MODE4 - PUNCH INPUT.
C     ********************
C
370   TYPE 380
380   FORMAT ('$COL,TYPE?')
      READ (5,20) COL,TYPE
      DO 390 J=1,25
         CALL PUNCHI(IWS)
         IF ( IWS .EQ. "177777)  GOTO 400
390      CONTINUE
400   DO 410 J=5,465
         CALL PUNCHI(IRAY(J,COL,TYPE))
410      CONTINUE
      GOTO 40
C
C
C     MODE5 - QUIT.
C     *************
C
500   END
```

```
        .GLOBL SETUP,SIGAV,DISP,PLOT,PUNCHO,PUNCHI
ANSCNL: 1                       ;ADDRESS OF NSCNL.
ANSCNH: 1                       ;ADDRESS OF NSCNH.
TNSCNL: 1                       ;TEMPORARY STORAGE FOR
TNSCNH: 1                       ;   NSCN.
ASCNMOD:1                       ;ADDRESS OF SCNMOD.
ASPEED: 1                       ;ADDRESS OF SPEED.
STORE:  1                       ;TEMP STORAGE FOR SIGAV.
TTINT1: 1                       ;ORIGINAL TTY INTERUPT VECTOR
TTINT2: 1                       ;   WILL BE STORED HERE.
CNT:    1                       ;USED IN DELAY LOOP.
SETUP:  MOV     #5037,@#132626  ;STOP CLOCK INTERUPT.
        MOV     #177546,@#132630;
        MOV     #2,@#132632     ;
        ADD     #2,%5           ;%5 CONTAINS ADDRESS NSCNL.
        MOV     (%5)+,ANSCNL    ;ANSCNL CONTAINS ADDRESS OF NSCNL.
        MOV     (%5)+,ANSCNH    ;ANSCNH CONTAINS ADDRESS OF NSCNH.
        MOV     (%5)+,ASCNMOD   ;ASCNMOD CONTAINS ADDRESS OF SCNMOD.
        MOV     (%5)+,ASPEED    ;ASPEED CONTAINS ADDRESS OF SPEED.
        MOV     @#60,TTINT1     ;SAVE OLD TTY INTERUPT
        MOV     @#62,TTINT2     ;   VECTORS.
        RTS     %7
SIGAV:  MOV     #TTYINT,@#60    ;WHEN TTY INTERUPTS, GOTO
        MOV     #340,@#62       ;   TTYINT WITH PRIORITY 7.
        MOV     #200,@#177776   ;RUN SUBROUTINE AT PRIORITY 4.
        ADD     #2,%5           ;%5 POINTS AT ADDRESS OF IRAY(1,COL).
        MOV     (%5),%0         ;%0 CONTAINS ADDR OF IRAY(1,COL).
        MOV     (%5),%1         ;%1 CONTAINS ADDR OF IRAY(1,COL).
        ADD     #3560,%0        ;%0 CONTAINS ADDR OF IRAY(922,COL).
L0:     MOV     #40000,(%1)+    ;CLEAR COLUMN OF IRAY.
        CMP     %0,%1           ;
        BNE     L0              ;
        MOV     #L1,@#300       ;INITIALIZE A/D BY LETTING
        MOV     #100,@#167770   ;   IT INTERUPT.
        BR      L2              ;
L1:     RTI                     ;
L2:     MOV     #SERVE,@#300    ;WHEN A/D INTERUPTS, GOTO SERVE
        MOV     #340,@#302      ;   WITH PRIORITY 7.
        MOV     @ANSCNL,TNSCNL  ;STORE NSCN IN TNSCN.
        MOV     @ANSCNH,TNSCNH  ;
        ADD     #1,TNSCNL       ;INITIALIZE TNSCN TO COUNT
        ADC     TNSCNH          ;   # OF SCANS LEFT.
L3:     MOV     #144014,@#167772 ;POSITION HORIZ DAC.(62=1 V OFFSET)
        MOV     #0,CNT          ;DELAY LOOP TO ALLOW
L3A:    INC     CNT             ;   VERT A/D TO STABILIZE.
        CMP     CNT,#200        ;
        BNE     L3A             ;
        SUB     #1,TNSCNL       ;DECREMENT # OF SCANS REMAINING.
        BNE     L4              ;IF TNSCNL#0, GOTO L4.
        TST     TNSCNH          ;TEST TNSCNH.
        BNE     L4              ;IF TNSCNH#0, GOTO L4.
        MOV     TTINT1,@#60     ;TNSCN=0. RESTORE TTY
        MOV     TTINT2,@#62     ;   INTERUPT VECTOR AND
        RTS     %7              ;   RETURN TO MAIN PROGRAM.
L4:     SBC     TNSCNH          ;SUBTRACT CARRY FROM TNSCNH.
        MOV     #0,@#177776     ;ALLOW TTY TO INTERUPT.
        MOV     #200,@#177776   ;DISALLOW TTY TO INTERUPT.
```

```
        MOV     #144014,%0      ;%0 CONTROLS HORIZ DAC.(62=1 V OFFSET)
        MOV     (%5),%1         ;%1 CONTAINS ADDR OF IRAY(1,COL).
L5:     ADD     #1,%0           ;INCR HORIZ DAC POSITION.
        CMP     %0,#145000      ;IS SCAN COMPLETE?
        BEQ     L3              ;    IF YES, START NEW SCAN.
        MOV     @#167774,STORE  ;STORE DATA AND ENABLE INTERUPT.
        WAIT                    ;WAIT FOR INTERUPT.
        BR      L5              ;INTERUPT COMPLETE. GOTO L5.
SERVE:  MOV     %0,@#167772     ;OUTPUT HORIZ DAC POSITION.
        MOV     STORE,%2        ;RETRIEVE DATA FROM LAST HOR DAC POS.
        BPL     L6              ;IF POS, GOTO L6.
        NEG     %2              ;ADD DATA TO IRAY.
        SUB     %2,(%1)+        ;
        SBC     (%1)+           ;
        RTI                     ;RETURN FROM INTERUPT.
L6:     ADD     %2,(%1)+        ;ADD DATA TO IRAY.
        ADC     (%1)+           ;
        RTI                     ;RETURN FROM INTERUPT.
TTYINT: CMPB    #300,@#177562   ;IS TTY INTERUPTING WITH "@"?
        BEQ     L7              ;    IF YES, PROCESS AT L7.
        RTI                     ;    IF NO, RETURN FROM INTERUPT.
L7:     MOV     TTINT1,@#60     ;RESTORE ORIGINAL TTY INTERUPT
        MOV     TTINT2,@#62     ;    VECTOR.
        SUB     TNSCNL,@ANSCNL  ;IF INTERUPTED FROM SIGAV,
        SBC     @ANSCNH         ;    THEN NSCN CONTAINS
        SUB     TNSCNH,@ANSCNH  ;    # OF SCANS COMPLETED.
        MOV     #766,%6         ;SET STACK POINTER TO RETURN
        RTS     %7              ;    TO MAIN PROGRAM.
DISP:   MOV     #0,@#177776     ;RUN SUBROUTINE AT PRIORITY 0.
        MOV     #TTYINT,@#60    ;WHEN TTY INTERUPTS, GOTO
        MOV     #340,@#62       ;    TTYINT WITH PRIORITY 7.
        ADD     #2,%5           ;%5 POINTS AT ADDRESS OF FRAY(1,COL).
L8:     MOV     (%5),%1         ;%1 CONTAINS ADDRESS OF FRAY(1,COL).
        MOV     #140062,%4      ;%4 CONTROLS HORIZONTAL DAC.
        SUB     #1,%4           ;INITIALIZE %4.
L9:     ADD     #1,%4           ;INCREMENT HORIZ DAC.
        MOV     %4,@#167772     ;OUTPUT HORIZONTAL AND VERTICAL
        MOV     (%1)+,@#167772  ;    DAC POSITIONS.
        MOV     #0,CNT          ;INITIALIZE DELAY VARIABLE.
L10:    INC     CNT             ;SLOW DOWN SCAN WITH DELAY
        CMP     CNT,@ASPEED     ;    LOOP.
        BNE     L10             ;
        CMP     %4,#140776      ;IS SCAN COMPLETE?
        BNE     L9              ;    IF NO, CONTINUE SCAN.
        CMP     #0,@ASCNMOD     ;    IF YES, NEED MORE THAN 1 SCAN?
        BNE     L8              ;        YES. KEEP SCANNING.
        MOV     TTINT1,@#60     ;        NO. RESTORE TTY INTERUPT
        MOV     TTINT2,@#62     ;            VECTORS AND QUIT
        RTS     %7              ;            DISPLAYING.
PLOT:   MOV     #0,@#177776     ;RUN SUBROUTINE AT PRIORITY 0.
        MOV     #TTYINT,@#60    ;WHEN TTY INTERUPTS, GOTO
        MOV     #340,@#62       ;    TTYINT WITH PRIORITY 7.
        MOV     (%5)+,%3        ;%3 CONTAINS 2*# OF POINTS.
L11:    BIC     #2000,@(%5)     ;CLEAR POSSIBLE VERT DAC ADDRESS.
        BIS     #140000,@(%5)   ;APPEND HORIZ DAC ADDRESS.
        MOV     @(%5)+,@#167772 ;OUTPUT HORIZ DAC POSITION.
        BIS     #142000,@(%5)   ;APPEND VERT DAC ADDRESS.
```

```
        MOV     @(%5)+,@#167772 ;OUTPUT VERT DAC POSITION.
        MOV     #0,CNT          ;DELAY LOOP.
L12:    INC     CNT             ;
        CMP     CNT,@ASPEED     ;
        BNE     L12             ;
        SUB     #2,%3           ;DECR # OF PNTS TO BE PLOTTED.
        BNE     L11             ;IF PNTS NOT ALL PLOTTED, CONT.
        MOV     TTINT1,@#60     ;RESTORE ORIGINAL TTY
        MOV     TTINT2,@#62     ;   INTERUPT VECTORS.
        RTS     %7              ;PLOTTING COMPLETE.
PUNCHO: ADD     #2,%5           ;%5 PNTS. TO ADD. OF IRAY(J,COL,TYPE).
        MOV     @(%5),%4        ;%4 CONTAINS IRAY(J,COL,TYPE).
        MOV     @#177564,%3     ;STORE PUNCH STATUS REGISTER.
        BIC     #104,@#177564   ;CLEAR INTERUPT AND MAINTENANCE.
PO1:    BIT     #200,@#177564   ;TEST IF PUNCH IS FREE.
        BEQ     PO1             ;IF PUNCH NOT FREE, GOTO PO1.
        MOV     %4,@#177566     ;OUTPUT SECOND BYTE OF IRAY(J,COL,TYPE).
        SWAB    %4              ;SWAP BYTES OF %4.
PO2:    BIT     #200,@#177564   ;TEST IF PUNCH IS FREE.
        BEQ     PO2             ;IF PUNCH NOT FREE, GOTO PO2.
        MOV     %4,@#177566     ;OUTPUT FIRST BYTE OF IRAY(J,COL,TYPE).
        MOV     %3,@#177564     ;RESTORE PUNCH STATUS REGISTER.
        RTS     %7              ;
PUNCHI: ADD     #2,%5           ;%5 PNTS. TO ADD. OF IRAY(J,COL,TYPE).
        MOV     @#177560,%3     ;STORE READER STATUS REGISTER.
        BIC     #100,@#177560   ;CLEAR INTERUPT.
        BIS     #1,@#177560     ;ENABLE READER.
PI1:    BIT     #200,@#177560   ;IS BUFFER READY?
        BEQ     PI1             ;IF NOT, GOTO PI1.
        MOV     @#177562,@(%5)  ;MOVE SECOND BYTE TO IRAY(J,COL,TYPE).
        SWAB    @(%5)           ;SWAP BYTES OF IRAY(J,COL,TYPE).
        BIS     #1,@#177560     ;ENABLE READER.
PI2:    BIT     #200,@#177560   ;IS BUFFER READY?
        BEQ     PI2             ;IF NOT, GOTO PI2.
        MOVB    @#177562,@(%5)  ;MOVE FIRST BYTE TO IRAY(J,COL,TYPE).
        SWAB    @(%5)           ;SWAP BYTES OF IRAY(J,COL,TYPE).
        MOV     %3,@#177560     ;RESTORE READER STATUS REGISTER.
        RTS     %7              ;
        .END
```

APPENDIX B - DESCRIPTION OF SIG

This program is capable of averaging a signal, averaging the background noise, (the user must block the signal to the detector), subtracting the two to obtain signal minus coherent background noise, displaying any one of these on an oscilloscope or plotter at various speeds, with or without axis drawn, and outputting or inputting any of these from punched tape.

Upon running the program, the user will be asked what he wants done, and how. The user must simply refer to the code printed within the program to determine what the program is asking for.

A typical use of the program might look something like this. (ie. COMP refers to the computer and USER refers to the person using the program.)

```
COMP:  .                        ;the computer is in the mon-
                                   itor waiting for a command.

USER:  RUN DX1:SIG.SAV          ;user wants to run SIG.
```

```
COMP:  CODE?                    ;computer is now running
                                   SIG, and is asking if the
                                   code should be printed.

USER:  0                        ;from the code, 0 means no.
                                   The code won't be printed.

COMP:  MODE?                    ;referring to code, there are
                                   five possible modes.  Com-
                                   puter is asking which one.

USER:  1                        ;MODE1 is signal average.

COMP:  COL,TYPE,NSCNH,NSCNL?    ;computer now needs to know
                                   into which column the sig-
                                   nal should be placed, 1,
                                   2, or 3, what type of av-
                                   eraging is being done,
                                   (signal - TYPE1 or noise -
                                   TYPE2), and how many scans
                                   should be made, (# of scans
                                   = 16384*NSCNH + NSCNL).

USER:  1,1,1,1                  ;user responds with column one,
                                   signal (TYPE1) and 16385
                                   scans.
```

;the user may now observe the sampling scope screen and see the signal averaging taking place. The user may wait till 16385 scans have been taken, or he may stop averaging by typing "@".

USER: @ (typed but not printed)

;user decides to interupt signal averaging. Computer now normalizes the averaged signal.

COMP: NSCN= 0 381 PH= 64.52

;computer prints out that 0*16384 + 381*1 = 381 scans were taken. The average height of the signal on the screen was 64.52 in uncalibrated units.

COMP: TIME/CM? ;computer asks for the time per cm of the signal averaged trace, and will output this # in the summary of results.

USER: 1 ;user choses to remember the scaling factor himself, and inputs a dummy number.

COMP: MODE? ;computer is now ready to execute another mode. ie. it could average the noise and calculate signal minus noise for column one or average another signal into columns two or three, or display any column and type, or output any column and type onto punched tape, or input, or quit. In short, any mode is valid.

USER: 2 ;referring to code, user decides to display a signal.

COMP: COL,TYPE,SCNMOD,SPEED,SUMR?
;computer needs to know which column and type to display, whether on oscilloscope, or plotter, how fast, and if a summary of results is to be printed.

```
USER:  1,1,0,4000,0          ;user specifies column one,
                               type one, scan mode zero,
                               (for plotter), speed of
                               4000, (the smaller the fast-
                               er), and no summary of re-
                               sults.

COMP:  PLTFQ,PLTFQR,WIDTH?   ;because signal is being out-
                               put onto plotter, computer
                               must know how the scales
                               are to be drawn.  PLTFQ is
                               the number of points to be
                               plotted on the horizontal
                               and vertical axis, PLTFQR is
                               the number of points to be
                               plotted on the calibration
                               markings and WIDTH is the
                               width of the calibration
                               markings.  Making PLTFQ and
                               PLTFQR to small means the
                               plotter will have to travel
                               to fast, and the scales will
                               not be uniform.  Making
                               them too large will waste
                               time.
```

```
USER:  400,10,10.          ;user opts for values that
                             were used for all plots in
                             this report.

COMP:  PAUSE 'LP'          ;computer tells user to lower
                             plotter pen.

USER:  (type return key)   ;user lowers plotter pen and
                             types return key.

COMP:  PAUSE 'RP'          ;computer tells user to raise
                             plotter pen.

USER:  (type return key)   ;

COMP:  PAUSE 'LP'

USER:  (type return key)   ;the computer plots graph.

COMP:  MODE?               ;computer is now ready to ex-
                             ecute any mode.

USER:  5                   ;user decides to quit using
                             program.
```

```
COMP:  - STOP -                ;program ends, and period
       .                          indicates the computer is
                                  now running the monitor.

USER:  GET HALT                ;user loads bootstrap loader
                                  to start up computer next
                                  day.  Computer can now be
                                  turned off.
```

The only other information needed concerns the inputting of daya from punched tape. The tapes will have a few inches of leader on them, and for input, the first nonzero byte must be placed directly over the punch reader. This byte will always be all holes, followed by a second byte of all holes.

APPENDIX C - COMPUTER LISTING OF SIG1

```
      INTEGER*2 APMAX,APMIN,BPNT,CODE,COL,DELAY,HDI,IRAYD(512,2,3),
     +   IRAYS(1024,2,3),MODE,NSCNH,NSCNL,PLTFQ,SCALE,SMAX(2,3),
     +   SNSCNH(2,2),SNSCNL(2,2),SPEED,SPMAX,SPMIN,SUMR,TYPE
      REAL DPRAY(512,2),MAX,MIN,SCL
      DATA SMAX,SNSCNH,SNSCNL/14*0/
      CALL SETUP(APMIN,BPNT,DELAY,HDI,MODE,NSCNH,NSCNL,SPEED,SPMIN)
      TYPE 10
10    FORMAT ('$CODE?')
      READ (5,20) CODE
20    FORMAT (7I10)
      IF ( CODE .EQ. 0 )  GOTO 50
      TYPE 30
30    FORMAT (' 0 = NO'/' 1 = YES'/
     +    ' APMAX = MAXIMUM POINT TO BE ANALYZED. DEFAULT=512.'/
     +    ' APMIN = MINIMUM POINT TO BE ANALYZED. DEFAULT=62.'/
     +    ' BPNT = BASELINE POINT, TO SUBTRACT BASELINE DRIFT. DFLT=31.'/
     +    ' CODE0 = DO NOT PRINT THIS CODE TO USING PROGRAM.'/
     +    ' CODE1 = DO PRINT CODE.'/
     +    ' COL1,COL2 = AVAILABLE COLUMNS.'/
     +    ' DELAY = # OF TIMES DELAY LOOP DONE AT END OF SCAN. DFLT=1.'/
     +    ' HDI = HORIZONTAL DAC INCREMENT. MAX=512. DEFAULT=1.'
     +    ' MODE1 = CLEAR COLUMN, THEN SIGNAL AVERAGE NORMALLY.'/
     +    ' MODE2 = DO NOT CLEAR COLUMN, AVERAGE NORMALLY.'/
     +    ' MODE3 = CLEAR COLUMN, AVERAGE SUBTRACTING BASELINE.'/
     +    ' MODE4 = DO NOT CLEAR COLUMN, AVERAGE SUBTRACTING BASELINE.'/
     +    ' MODE5 = OUTPUT DISPLAY FOR OSCILLOSCOPE.'/
     +    ' MODE6 = OUTPUT DISPLAY FOR PLOTTER.'/
     +    ' MODE7 = OUTPUT SIGNAL ON PUNCHED TAPE.'/
     +    ' MODE8 = INPUT SIGNAL FROM PUNCHED TAPE.'/
     +    ' MODE9 = QUIT.'
     +    ' NSCNH = HIGH ORDER WORD FOR # OF SCANS.'/
     +    ' NSCNL = LOW ORDER WORD FOR # OF SCANS.'/
     +    ' PLTFQ = FREQUENCY OF POINTS TO DRAW BORDER OF PLOT.'/
     +    ' SCALE0 = ACCEPT DEFAULT CONDITIONS FOR SIGNAL SCALING.'/
     +    ' SCALE1 = SPECIFY OWN SCALING PARAMETERS.'/
     +    ' SPEED = # OF TIMES DELAY LOOP IS EXECUTED PER DISPLAY PNT.'/
     +    ' SPMAX = MAXIMUM POINT TO BE SIGNAL AVERAGED. DEFAULT=511.'/
     +    ' SPMIN = MINIMUM POINT TO BE SIGNAL AVERAGED. DEFAULT=31.'/
     +    ' SUMR0 = DO NOT OUTPUT SUMMARY OF RESULTS.'/
     +    ' SUMR1 = OUTPUT SUMMARY OF RESULTS.'/
     +    ' TYPE1 = SIGNAL.'/
     +    ' TYPE2 = NOISE.'/
     +    ' TYPE3 = SIGNAL-NOISE.'/
     +    ' @ = CHARACTER USED TO STOP SAMPLING OR DISPLAYING.')
50    APHDI=62
      APMAX=512
      APMIN=62
      BPNT=31
      DELAY=1
      HDI=1
      SPMAX=511
      SPMIN=31
60    TYPE 70
70    FORMAT ('$MODE?')
      READ (5,20) MODE
      GOTO (80,80,300,300,500,500,600,700,800)  MODE
```

```
C
C
C     MODE1 AND MODE2 - SIGAV.
C     ************************
C
80    TYPE 90
90    FORMAT ('$COL,NSCNH,NSCNL,SCALE,TYPE? ')
      READ (5,20) COL,NSCNH,NSCNL,SCALE,TYPE
      IF ( SCALE .EQ. 0 )  GOTO 105
      TYPE 100
100   FORMAT ('$APMAX,APMIN,BPNT,DELAY,HDI,SPMAX,SPMIN? ')
      READ (5,20) APMAX,APMIN,BPNT,DELAY,HDI,SPMAX,SPMIN
105   IF ( MODE .EQ. 1 )  CALL CLEAR(IRAYS(1,COL,TYPE))
      CALL SIGAV(IRAYS(SPMAX*2+1,COL,TYPE),IRAYS(SPMIN*2+1,COL,TYPE))
C
C     SIGAV IS AN ASSEMBLER ROUTINE THAT TAKES SUCCESSIVE ADC
C     READINGS FROM SCOPE, ADDS THEM TO IRAYS (STARTING AT
C     IRAYS(SMIN,COL,TYPE)), ADDS CARRY TO NEXT ROW ELEMENT OF
C     IRAYS, CONTINUING TO IRAYS(SPMAX,COL,TYPE).
C
      IF ( MODE .EQ. 1 ) GOTO 110
      SNSCNH(COL,TYPE)=SNSCNH(COL,TYPE)+NSCNH
      SNSCNL(COL,TYPE)=SNSCNL(COL,TYPE)+NSCNL
      GOTO 115
110   SNSCNH(COL,TYPE)=NSCNH
      SNSCNL(COL,TYPE)=NSCNL
115   MAX=-1.E38
      MIN=1.E38
      DO 120 J=APMIN,APMAX
         DPRAY(J,TYPE)=IRAYS(J*2-1,COL,TYPE)+
     +       65536.*(IRAYS(J*2,COL,TYPE)-IRAYS(APMIN*2,COL,TYPE))
         IF ( IRAYS(J*2-1,COL,TYPE) .LT. 0 )
     +       DPRAY(J,TYPE)=DPRAY(J,TYPE)+65536.
         MAX=AMAX1(MAX,DPRAY(J,TYPE))
120      MIN=AMIN1(MIN,DPRAY(J,TYPE))
C
C     ABOVE LOOP TRANSFERRED SIGNAL IN IRAYS ( AN INTEGER ARRAY
C     FOR SIGNAL AVERAGING WHERE EACH DATA POINT NEEDED A LOW AND
C     HIGH ORDER NUMBER ) TO DPRAY ( A DOUBLE PRECISION ARRAY,
C     WHERE EACH POINT ONLY NEEDS ONE NUMBER ). LOOP ALSO FOUND
C     MAX AND MIN FOR LATER USE.
C
      SCL=SNSCNH(COL,TYPE)*65536.+SNSCNL(COL,TYPE)
      DO 130 J=APMIN,APMAX
130      DPRAY(J,TYPE)=(DPRAY(J,TYPE)-MIN)/SCL
C
C     ABOVE LOOP REMOVED DC OFFSET FROM SIGNAL AND DIVIDED SIGNAL
C     BY NUMBER OF SCANS.
C
      WS1=(MAX-MIN)/SCL
      SMAX(COL,TYPE)=100*WS1
      SCL=511./WS1
      DO 140 J=APMIN,APMAX
         IRAYD(J,COL,TYPE)=DPRAY(J,TYPE)*SCL
140      IRAYD(J,COL,TYPE)=IRAYD(J,COL,TYPE)+"142000
C
C     ABOVE LOOP SCALED SIGNAL IN DPRAY TO FIT 10 BIT DAC, PUT IT
C     INTO IRAYD, AND ADDED VERTICAL DAC ADDRESS.
C
```

```
      TYPE 150,SNSCNH(COL,TYPE),SNSCNL(COL,TYPE),SMAX(COL,TYPE)
150   FORMAT (' NSCN=',2I8,5X,' PH=',I8)
      IF ( TYPE .EQ. 1 )  GOTO 60
C
C     BOTH SIGNAL AND NOISE SCANS HAVE BEEN TAKEN IF TYPE=2.
C     PROGRAM NOW CALCULATES SIGNAL-NOISE.
C
      MAX=-1.E38
      MIN=1.E38
      DO 160 J=APMIN,APMAX
         DPRAY(J,1)=DPRAY(J,1)-DPRAY(J,2)
         MAX=AMAX1(MAX,DPRAY(J,1))
160      MIN=AMIN1(MIN,DPRAY(J,1))
C
C     DPRAY(J,1) NOW CONTAINS SIGNAL-NOISE.
C
      SMAX(COL,3)=MAX-MIN
      SCL=511.D0/SMAX(COL,3)
      DO 170 J=APMIN,APMAX,1
         IRAYD(J,COL,3)=(DPRAY(J,1)-MIN)*SCL
170      IRAYD(J,COL,3)=IRAYD(J,COL,3)+"142000
C
C     IRAYD(J,COL,3) NOW CONTAINS SIGNAL-NOISE SCALED TO 10 BITS
C     WITH VERTICAL DAC ADDRESS ADDED.
C
      TYPE 180,SMAX(COL,3)
180   FORMAT (' DPH=',F8.2)
      GOTO 60
C
C
C     MODE3 AND MODE4 - SIGMBL.
C     *************************
C
300   TYPE 100
      READ (5,20) COL,NSCNH,NSCNL,TYPE
      IF ( MODE .EQ. 3 )  CALL CLEAR(IRAYS(1,COL,TYPE))
      CALL SGMBL(IRAYS(SPMAX*2+1,COL,TYPE),IRAYS(SPMIN*2+1,COL,TYPE))
C
C     SIGMBL IS AN ASSEMBLER ROUTINE LIKE SIGAV, BUT BETWEEN EACH
C     SIGNAL POINT, IT SAMPLES BPNT (BASELINE POINT) AND SUBTRACTS
C     IT FROM SIGNAL TO REMOVE BASELINE DRIFT.
C
      GOTO 110
C
C
C     MODE5 AND MODE6 - DISPLAY.
C     **************************
C
500   TYPE 510
510   FORMAT ('$COL,SPEED,SUMR,TYPE?')
      READ (5,20) COL,SPEED,SUMR,TYPE
      IF ( SUMR .EQ. 0 )  GOTO 530
      TYPE 520,SNSCNH(COL,1),SNSCNL(COL,1),SNSCNH(COL,2),
     +   SNSCNL(COL,2),SMAX(COL,1),SMAX(COL,2),SMAX(COL,3)
520   FORMAT (20X,'TYPE1',14X,'TYPE2',14X,'TYPE3'/' NSCNH,NSCNL=',
     +   I6,2X,I6,5X,I6,2X,I6/' PH=',I21,I19,I19)
530   IF ( MODE .EQ. 5 )  GOTO 560
      CALL PLOT(SPMIN,511)
```

```
      TYPE 535
535   FORMAT ('$PLTFQ?')
      READ (5,20) PLTFQ
      PAUSE 'LP'
      DO 540 J=1,PLTFQ,1
         IWS=511.-511.*J/PLTFQ
         CALL PLOT(SPMIN,IWS)
540      CONTINUE
      DO 550 J=1,PLTFQ,1
         IWS=SPMIN+1.*J*(SPMAX-SPMIN)*HDI/PLTFQ
         CALL PLOT(IWS,0)
550      CONTINUE
      PAUSE 'RP'
      CALL PLOT(APMIN-1,IRAYD(APMIN,COL,TYPE))
      PAUSE 'LP'
560   CALL DSPL(IRAYD(APMAX,COL,TYPE),IRAYD(APMIN,COL,TYPE))
      GOTO 60
C
C
C     MODE7 - PNCHOU.
C     ***************
C
600   TYPE 610
610   FORMAT ('$COL,TYPE?')
      READ (5,20) COL,TYPE
      DO 615 J=1,25
         CALL PNCHOU(0)
615      CONTINUE
      CALL PNCHOU("177777)
      CALL PNCHOU(APMAX)
      CALL PNCHOU(APMIN)
      CALL PNCHOU(BPNT)
      CALL PNCHOU(DELAY)
      CALL PNCHOU(HDI)
      CALL PNCHOU(SNSCNH(COL,TYPE))
      CALL PNCHOU(SNSCNL(COL,TYPE))
      CALL PNCHOU(SMAX(COL,TYPE))
      CALL PNCHOU(SPMAX)
      CALL PNCHOU(SPMIN)
      DO 620 J=APMIN,APMAX
         CALL PNCHOU(IRAYD(J,COL,TYPE))
620      CONTINUE
      PAUSE 'PT'
630   TYPE 640,APMAX,APMIN,BPNT,DELAY,HDI,SNSCNH(COL,TYPE),
     +    SNSCNL(COL,TYPE),SMAX(COL,TYPE),SPMAX,SPMIN
640   FORMAT (' APMAX =',I7/' APMIN =',I7/' BPNT  =',I7/' DELAY =',
     +    I7/' HDI   =',I7/' NSCNH =',I7/' NSCNL =',I7/' PH    =',
     +    I7/' SPMAX =',I7/' SPMIN =',I7)
      GOTO 60
C
C
C     MODE8 - PNCHIN
C     **************
C
700   TYPE 610
      READ (5,20) COL,TYPE
```

```
      DO 710 J=1,25
         CALL PNCHIN(IWS)
         IF ( IWS .EQ. "177777)  GOTO 720
710      CONTINUE
720   CALL PNCHIN(APMAX)
      CALL PNCHIN(APMIN)
      CALL PNCHIN(BPNT)
      CALL PNCHIN(DELAY)
      CALL PNCHIN(HDI)
      CALL PNCHIN(SNSCNH(COL,TYPE))
      CALL PNCHIN(SNSCNL(COL,TYPE))
      CALL PNCHIN(SMAX(COL,TYPE))
      CALL PNCHIN(SPMAX)
      CALL PNCHIN(SPMIN)
      DO 730 J=APMIN,APMAX
         CALL PNCHIN(IRAYD(J,COL,TYPE))
730      CONTINUE
      GOTO 630
C
C
C     MODE9 - QUIT.
C     *************
C
800   END
```

```
        .GLOBL CLEAR,DSPL,PLOT,PNCHOU,PNCHIN,SETUP,SGMBL,SIGAV
AAPMIN: 1                       ;ADDRESS OF APMIN. ETC.
ABPNT:  1                       ;
ADELAY: 1                       ;
AHDI:   1                       ;
AMODE:  1                       ;
ANSCNH: 1                       ;
ANSCNL: 1                       ;
ASPEED: 1                       ;
ASPMIN: 1                       ;
CNT:    1                       ;USED AS A COUNTER.
STORE:  1                       ;STORES SIGNAL DURING ADC INTERUPT.
TNSCNH: 1                       ;TEMPORARY STORAGE FOR NSCN.
TNSCNL: 1                       ;
TTINT1: 1                       ;ORIGINAL TTY INTERUPT VECTOR
TTINT2: 1                       ;   IS STORED HERE.
                                ;
CLEAR:  ADD     #2,%5           ;%5 PNTS. @ ADD. OF IRAYS(1,COL,TYPE).
        MOV     (%5),%0         ;%0 CNTNS. ADD. OF IRAYS(1,COL,TYPE).
        MOV     (%5),%1         ;%1 CNTNS. ADD. OF IRAYS(1,COL,TYPE
        ADD     #3776,%1        ;%1 CNTNS. ADD. OF IRAYS(1024,COL,TYPE).
CL1:    MOV     #40000,(%0)+    ;CLEAR COLUMN OF IRAYS.
        CMP     %0,%1           ;
        BLE     CL1             ;
        RTS     %7              ;RETURN TO MAIN PROGRAM.
                                ;
DSPL:   MOV     #0,@#177776     ;RUN SUBROUTINE AT PRIORITY 0.
        MOV     #TTYINT,@#60    ;WHEN TTY INTERUPTS, GO TO
        MOV     #340,@#62       ;   TTYINT WITH PRIORITY 7.
        MOV     @AAPMIN,%0      ;PUT APMIN IN %0.
        ADD     #137777,%0      ;ADD HORIZ. DAC ADD.  (APMIN>=1).

        SUB     @AHDI,%0        ;SUBTRACT HDI.
        ADD     #2,%5           ;%5 PNTS. @ ADD. OF IRAYD(APMAX,COL,TYPE
        MOV     (%5)+,%1        ;%1 CNTNS. ADD. OF IRAYD(APMAX,COL,TYPE)
DS1:    MOV     (%5),%2         ;%2 CNTNS. ADD. OF IRAYD(APMIN,COL,TYPE)
        MOV     %0,%3           ;PUT FIRST HORIZ. DAC POS. IN %3.
DS2:    ADD     @AHDI,%3        ;INCREMENT HORIZ. DAC.
        MOV     %3,@#167772     ;OUTPUT HORIZ. DAC. POSITION.
        MOV     (%2)+,@#167772  ;OUTPUT VERTICAL DAC. POSITION.
        MOV     #0,CNT          ;INITIALIZE DELAY VARIABLE.
DS3:    INC     CNT             ;SLOW DOWN SCAN WITH
        CMP     CNT,@ASPEED     ;   DELAY LOOP.
        BLE     DS3             ;
        CMP     %2,%1           ;IS SCAN COMPLETE?
        BLE     DS2             ;   IF NO, CONTINUE SCAN.
        CMP     @AMODE,#6       ;   IF YES, NEED MORE THAN 1 SCAN?
        BNE     DS1             ;      YES. KEEP SCANNING.
        MOV     TTINT1,@#60     ;      NO. RESTORE TTY INTERUPT
        MOV     TTINT2,@#62     ;         VECTORS AND QUIT
        RTS     %7              ;         DISPLAYING.
                                ;
PLOT:   ADD     #2,%5           ;%5 PNTS. @ ADD. OF X POSITION.
        MOV     @(%5)+,%4       ;MOVE X TO %4.
        BIS     #140000,%4      ;SET HORIZ. DAC ADD.
        MOV     %4,@#167772     ;OUTPUT X.
        MOV     @(%5),%4        ;MOVE Y TO %4.
        BIS     #142000,%4      ;SET VERT. DAC ADD.
        MOV     %4,@#167772     ;OUTPUT Y.
        RTS     %7              ;RETURN TO MAIN PROGRAM.
                                ;
```

```
PNCHOU: ADD     #2,%5           ;%5 PNTS. @ ADD. OF WORD TO BE PUNCHED.
        MOV     @(%5),%4        ;%4 CONTAINS WORD TO BE PUNCHED.
        MOV     @#177564,%3     ;STORE PUNCH STATUS REGISTER.
        BIC     #104,@#177564   ;CLEAR INTERUPT AND MAINTENANCE.
PO1:    BIT     #200,@#177564   ;TEST IF PUNCH IS FREE.
        BEQ     PO1             ;IF PUNCH NOT FREE, GOTO PO1.
        MOV     %4,@#177566     ;OUTPUT SECOND BYTE OF WORD.
        SWAB    %4              ;SWAP BYTES OF %4.
PO2:    BIT     #200,@#177564   ;TEST IF PUNCH IS FREE.
        BEQ     PO2             ;IF PUNCH NOT FREE, GOTO PO2.
        MOV     %4,@#177566     ;OUTPUT FIRST BYTE OF WORD.
        MOV     %3,@#177564     ;RESTORE PUNCH STATUS REGISTER.
        RTS     %7              ;
                                ;
PNCHIN: ADD     #2,%5           ;%5 PNTS @ ADD. OF WORD REQUIRING INPUT.
        MOV     @#177560,%3     ;STORE READER STATUS REGISTER.
        BIC     #100,@#177560   ;CLEAR INTERUPT.
        BIS     #1,@#177560     ;ENABLE READER.
PI1:    BIT     #200,@#177560   ;IS READER DONE?
        BEQ     PI1             ;IF NOT, GOTO PI1.
        MOV     @#177562,@(%5)  ;MOVE 2ND BYTE TO WORD REQUIRING INPUT.
        SWAB    @(%5)           ;SWAP BYTES OF WORD REQUIRING INPUT.
        BIS     #1,@#177560     ;ENABLE READER.
PI2:    BIT     #200,@#177560   ;IS READER DONE?
        BEQ     PI2             ;IF NOT, GOTO PI1.
        MOVB    @#177562,@(%5)  ;MOVE 1ST BYTE TO WORD REQUIRING INPUT.
        SWAB    @(%5)           ;SWAP BYTES OF WORD REQUIRING INPUT.
        MOV     %3,@#177560     ;RESTORE READER STATUS REGISTER.
        RTS     %7              ;
                                ;
SETUP:  MOV     #5037,@#132626  ;STOP CLOCK INTERUPT.
        MOV     #177546,@#132630;
        MOV     #2,@#132632     ;
        ADD     #2,%5           ;%5 PNTS. @ ADD. OF APMIN.
        MOV     (%5)+,AAPMIN    ;AAPMIN CNTNS. ADD. OF APMIN. ETC.
        MOV     (%5)+,ABPNT     ;
        MOV     (%5)+,ADELAY    ;
        MOV     (%5)+,AHDI      ;
        MOV     (%5)+,AMODE     ;
        MOV     (%5)+,ANSCNH    ;
        MOV     (%5)+,ANSCNL    ;
        MOV     (%5)+,ASPEED    ;
        MOV     (%5)+,ASPMIN    ;
        MOV     @#60,TTINT1     ;SAVE FORTRAN TTY INTERUPT
        MOV     @#62,TTINT2     ;   VECTOR.
        RTS     %7              ;
                                ;
SGMBL:  RTS     %7              ;INSERT SUBROUTINE HERE.
                                ;
SIGAV:  MOV     #200,@#177776   ;RUN SUBROUTINE @ PRIORITY 7.
        MOV     #SI1,@#300      ;INITIALIZE ADC BY LETTING IT INTERUPT.
        MOV     #100,@#167770   ;
        BR      SI2             ;
SI1:    RTI                     ;RETURN FROM ADC INTERUPT.
SI2:    MOV     #TTYINT,@#60    ;WHEN TTY INTERUPTS, GOTO
        MOV     #340,@#62       ;   TTYINT WITH PRIORITY 7.
        MOV     #SI9,@#300      ;WHEN ADC INTERUPTS, GOTO
        MOV     #340,@#302      ;   SI9 WITH PRIORITY 7.
```

```
        MOV     @ANSCNH,TNSCNH  ;STORE NSCN IN TNSCN.
        MOV     @ANSCNL,TNSCNL  ;
        ADD     #1,TNSCNL       ;INITIALIZE TNSCN TO COUNT
        ADC     TNSCNH          ;    # OF SCANS LEFT.
        MOV     @ASPMIN,%4      ;%4 CNTNS. SPMIN.
        ADD     #144000,%4      ;APPEND HORIZ. S. SCOPE DAC ADDRESS.
        ADD     #2,%5           ;%5 PNTS. @ ADD. OF IRAYS(SPMAX,COL,TYPE
        MOV     (%5)+,%3        ;%3 CNTNS. ADD. OF IRAYS(SPMAX,COL,TYPE)
        BR      SI5             ;
SI3:    MOV     STORE,%2        ;ANALYZE LAST POINT AS PER SI9.
        BPL     SI4             ;
        NEG     %2              ;
        SUB     %2,(%1)+        ;
        SBC     (%1)            ;
        BR      SI5             ;
SI4:    ADD     %2,(%1)+        ;
        ADC     (%1)            ;
SI5:    MOV     %4,%0           ;%0 CNTNS. SPMIN + HORIZ DAC ADD.
        MOV     %0,@#167772     ;POSITION HORIZ. DAC.
        MOV     (%5),%1         ;%1 CNTNS. ADD. OF IRAYS(SPMIN,COL,TYPE)
        MOV     #0,CNT          ;DELAY LOOP TO ALLOW VERTICAL
SI6:    INC     CNT             ;    ADC TO STABILIZE.
        CMP     CNT,@ADELAY     ;
        BNE     SI6             ;
        SUB     #1,TNSCNL       ;DECREMENT # OF SCANS REMAINING.
        BNE     SI7             ;IF TNSCNL#0, GOTO SI7.
        TST     TNSCNH          ;TEST TNSCNH.
        BNE     SI7             ;IF TNSCNH#0, GOTO SI7.
        MOV     TTINT1,@#60     ;TNSCN=0. RESTORE FORTRAN
        MOV     TTINT2,@#62     ;    TTY INTERUPT VECTOR.
        RTS     %7              ;
SI7:    SBC     TNSCNH          ;SUBTRACT CARRY FROM TNSCNH.
        MOV     #3,@#177776     ;ALLOW TTY TO INTERUPT.
        MOV     #200,@#177776   ;DISALLOW TTY TO INTERUPT.
SI8:    CMP     %1,%3           ;IS SCAN COMPLETE?
        BEQ     SI3             ;IF YES, FINISH OLD AND START NEW SCAN.
        MOV     @#167774,STORE  ;STORE DATA AND ENABLE INTERUPT.
        WAIT                    ;WAIT FOR INTERUPT.
        BR      SI8             ;INTERUPT COMPLETE. SAMPLE NEXT POINT.
SI9:    ADD     @AHDI,%0        ;INCREMENT HORIZ. POSITION.
        MOV     %0,@#167772     ;OUTPUT NEW HORIZ. POSITION.
        MOV     STORE,%2        ;RETRIEVE DATA FROM OLD HORIZ. POSITION.
        BPL     SI10            ;IF DATA IS POSITIVE, GOTO SI10.
        NEG     %2              ;ADD NEGATIVE DATA TO IRAYS.
        SUB     %2,(%1)+        ;
        SBC     (%1)+           ;
        RTI                     ;RETURN FROM ADC INTERUPT.
SI10:   ADD     %2,(%1)+        ;ADD POSITIVE DATA TO ARRAY.
        ADC     (%1)+           ;
        RTI                     ;RETURN FROM ADC INTERUPT.
TTYINT: CMPB    #300,@#177562   ;IS TTY INTERUPTING WITH "@"?
        BEQ     TTY1            ;    IF YES, PROCESS AT TTY1.
        RTI                     ;    IF NO, RETURN FROM INTERUPT.
TTY1:   MOV     TTINT1,@#60     ;RESTORE ORIGINAL TTY INTERUPT
        MOV     TTINT2,@#62     ;    VECTOR.
        SUB     TNSCNL,@ANSCNL  ;IF INTERUPTED FROM SIGAV,
        SBC     @ANSCNH         ;    THEN NSCN CONTAINS
        SUB     TNSCNH,@ANSCNH  ;    # OF SCANS COMPLETED.
        MOV     #764,%6         ;SET STACK POINTER TO RETURN
        RTS     %7              ;    TO MAIN PROGRAM.
        .END
```

```
      INTEGER*2 SGNL(961),OUT(461,3),RES,MODE,TYPE,SCNMOD,
     +   SPEED,PLTFQ,PLTFQR
      REAL*4 RSPNS(501),CONV(461),MAX,MIN,WS1,SD,WIDTH
      CALL SETUP(1,1,SCNMOD,SPEED)
      DO 10 J=251,711
         CALL PUNCHI(OUT(J-250,1))
10       SGNL(J)=OUT(J-250,1)-"142000
      DO 20 J=1,250
20       SGNL(J)=0
      DO 30 J=712,961
30       SGNL(J)=332
      GOTO 40
C
C
33    TYPE 35
35    FORMAT ('$MODE,TYPE?')
      READ (5,37) MODE,TYPE
37    FORMAT (2I10,F10.2)
      GOTO (40,150,160,180) MODE
C
C
C     MODE1 - CALCULATE RESPONSE AND CONVOLUTION.
C     *******************************************
C
40    TYPE 50
50    FORMAT ('$RES,SD?')
      READ (5,60) RES,SD
60    FORMAT (I10,F10.2)
      IF ( RES .EQ. 2 )  GOTO 80
      DO 70 J=1,501
70       RSPNS(J)=EXP(-.5*((J-251.)/SD)**2.)
      GOTO 100
80    WS1=(SD/2.)**2.
      DO 90 J=1,501
90       RSPNS(J)=1./((J-251.)**2.+WS1)
100   DO 110 J=1,461
         OUT(J,2)=511.*RSPNS(J+20)/RSPNS(251)
110      OUT(J,2)=OUT(J,2)+"142000
      MAX=-1.E38
      MIN=1.E38
      DO 130 J=1,461
         WS1=0.
         DO 120 I=1,501
120         WS1=WS1+RSPNS(I)*SGNL(I+J-1)
         CONV(J)=WS1
         MAX=AMAX1(MAX,WS1)
130      MIN=AMIN1(MIN,WS1)
      DO 140 J=1,461
         OUT(J,3)=511.*(CONV(J)-MIN)/(MAX-MIN)
140      OUT(J,3)=OUT(J,3)+"142000
      GOTO 33
C
C
C     MODE2 - DISPLAY.
C     ****************
C
150   TYPE 220
220   FORMAT ('$SCNMOD,SPEED?')
      READ (5,37) SCNMOD,SPEED
```

```
      IF ( SCNMOD .EQ. 1 )  GOTO 240
      TYPE 230
230   FORMAT ('$PLTFQ,PLTFQR,WIDTH?')
      READ (5,37) PLTFQ,PLTFQR,WIDTH
      CALL PLOT(50,511)
      PAUSE 'LP'
      WS1=0.
      DO 520 J=1,PLTFQ
         IWS1=511.-511.*J/PLTFQ
         CALL PLOT(50,IWS1)
         IF ( FLOAT(J)/PLTFQ .LT. WS1 )  GOTO 520
         WS1=WS1+.1
         DO 500 I=1,PLTFQR
            IWS2=50.+WIDTH*I/PLTFQR
            CALL PLOT(IWS2,IWS1)
500         CONTINUE
         DO 510 I=1,PLTFQR
            IWS2=50.+WIDTH-WIDTH*I/PLTFQR
            CALL PLOT(IWS2,IWS1)
510         CONTINUE
520      CONTINUE
      WS1=.15625
      DO 550 J=1,PLTFQ
         IWS1=50.+461.*J/PLTFQ
         CALL PLOT(IWS1,0)
         IF ( FLOAT(J)/PLTFQ .LT. WS1 )  GOTO 550
         WS1=WS1+.15625
         DO 530 I=1,PLTFQR
            IWS2=WIDTH*I/PLTFQR
            CALL PLOT(IWS1,IWS2)
530         CONTINUE
         DO 540 I=1,PLTFQR
         IWS2=WIDTH-WIDTH*I/PLTFQR
         CALL PLOT(IWS1,IWS2)
540         CONTINUE
550      CONTINUE
      PAUSE 'RP'
      CALL PLOT(50,OUT(1,TYPE))
      PAUSE 'LP'
240   CALL DISP(OUT(1,TYPE))
      GOTO 33
C
C
C     MODE3 - PUNCH.
C     **************
C
160   DO 170 J=1,461
170      CALL PUNCHO(OUT(J,TYPE))
      GOTO 33
C
C
C     MODE4 - QUIT.
C     *************
C
180   END
```

APPENDIX D - DESCRIPTION OF SIG1

SIG1 is in most ways very similar to SIG. The differences are noted below.

Because of the increased versatility of the program, there was only enough room in core to accomodate two columns instead of three.

In SIG, as soon as the user stopped signal averaging, he would have to start all over again if he wanted to do more averaging of the same signal. By the appropriate choice of modes in SIG1, (see code), the user can continue signal averaging onto a signal that he just finished displaying, outputting or inputting.

Finally, by specifying that new scaling parameters are to be used, the user can control the number and position of sampling points. There are 512 different horizontal positions on the sampling scope that can be sampled. SPMIN is the first position that will be sampled. HDI is the horizontal DAC increment, or the number of positions between points to be sampled. SPMAX - SPMIN is the number of points to be sampled. Thus, suppose a

person wanted to sample at positions 250, 260, 270 and 280. Then SPMIN is 250, HDI is 10 and SPMAX is 253.

Now, sometimes a person will want to sample a number of positions, (in the above case 250, 260, 270 and 280), but only analyze a certain region of them, (say positions 260 and 270). This can be done by setting APMIN = 251 + 1 = 252 and APMAX = 252 + 1 = 253.

Although this is a somewhat awkward system for the user, it was convenient in terms of positioning DAC's and indexing arrays. Anyone making extensive use of this program might want to change this form of the I/O.