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

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Introduction</td>
<td>1</td>
</tr>
<tr>
<td>2. A Description of the System</td>
<td>4</td>
</tr>
<tr>
<td>3. Fibre Dispersion Measurement With the System</td>
<td>11</td>
</tr>
<tr>
<td>4. Conclusions</td>
<td>46</td>
</tr>
<tr>
<td>Appendix A - Computer Listing of SIG</td>
<td>48</td>
</tr>
<tr>
<td>Appendix B - Description of SIG</td>
<td>55</td>
</tr>
<tr>
<td>Appendix C - Computer Listing of SIG1</td>
<td>62</td>
</tr>
<tr>
<td>Appendix D - Description of SIG1</td>
<td>72</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

Fig. 1.1: Single Scan of Noise plus Signal
Fig. 1.2: Single Scan of Signal
Fig. 1.3: Single Scan of Noise
Fig. 2.1: Trigger Pulse and Input Signal versus Time
Fig. 2.2: Input Signal versus Time as Displayed on Sampling Scope
Fig. 2.3: Schematic of Signal Averaging System
Fig. 3.1: A Conceptual Look at Fibre Dispersion
Fig. 3.2: Schematic of Experimental Setup
Fig. 3.3: Single Scan of Actual Signal
Fig. 3.4: Actual Signal Averaged on 500 psec/div
Fig. 3.5: Signal After 2.7 m of Fibre
Fig. 3.6: Signal After 11.4 m of Fibre
Fig. 3.7: Signal After 20.7 m of Fibre
Fig. 3.8: Signal After 37.0 m of Fibre
Fig. 3.9: Signal After 50.9 m of Fibre
Fig. 3.10: Signal After 87.9 m of Fibre
Fig. 3.11: Rise Time versus Fibre Length for a Single Day
Fig. 3.12: Rise Time versus Fibre Length for Ten Days
Fig. 3.13: Gaussian With $\Delta t = 20$ psec
Fig. 3.14: Convoluted Output for Gaussian Fibre Response

With $\Delta t = 20$ psec
LIST OF FIGURES (cont.)

Fig. 3.15: Gaussian With $\Delta t = 60\,\text{psec}$

Fig. 3.16: Convoluted Output for Gaussian Fibre Response
With $\Delta t = 60\,\text{psec}$

Fig. 3.17: Gaussian With $\Delta t = 100\,\text{psec}$

Fig. 3.18: Convoluted Output for Gaussian Fibre Response
With $\Delta t = 100\,\text{psec}$

Fig. 3.19: Superimposed Gaussians of Different $\Delta t$

Fig. 3.20: Superimposed Convoluted Outputs for Gaussian Responses

Fig. 3.21: Lorentzian With $\Delta t = 20\,\text{psec}$

Fig. 3.22: Convoluted Output for Lorentzian Fibre Response
With $\Delta t = 20\,\text{psec}$

Fig. 3.23: Lorentzian With $\Delta t = 60\,\text{psec}$

Fig. 3.24: Convoluted Output for Lorentzian Fibre Response
With $\Delta t = 60\,\text{psec}$

Fig. 3.25: Lorentzian With $\Delta t = 100\,\text{psec}$

Fig. 3.26: Convoluted Output for Lorentzian Fibre Response
With $\Delta t = 100\,\text{psec}$

Fig. 3.27: Superimposed Lorentzians of Different $\Delta t$

Fig. 3.28: Superimposed Convoluted Outputs for Lorentzian Responses

Fig. 3.29: Measured Output Compared With Gaussian Convoluted Output
LIST OF FIGURES (cont.)

Fig. 3.30: Measured Output Compared With Lorentzian Convoluted Output

Fig. 3.31: Rise Time of Gaussian Convoluted Outputs versus $\Delta t$

Fig. 3.32: Gaussian $\Delta t$ versus Fibre Length for Single Day

Fig. 3.33: Gaussian $\Delta t$ versus Fibre Length for Ten Days
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**

**Amplitude**

**Figure 1.1**

**Single Scan of Signal**

**Amplitude**

**Figure 1.2**

**Single Scan of Noise**

**Amplitude**

"Peaks" and "Valleys"

**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 ratio 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

Amplitude

TRIGGER

SIGNAL

Figure 2.1

INPUT SIGNAL VERSUS TIME AS DISPLAYED ON SAMPLING SCOPE

Amplitude

Time

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 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 \( \frac{2000 \text{ signals}}{\text{second}} / 461 \frac{\text{signals}}{\text{scan}} = 4.3 \frac{\text{scans}}{\text{second}} \), (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 convenient in terms of obtaining a well resolved signal in a short amount of time. A listing of this program
and a description 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 $\frac{5 \text{ signals}}{\text{second}}$. Therefore, the sampling rate would be $\frac{461 \text{ signals}}{\text{scan}} \div \frac{92 \text{ seconds}}{\text{scan}} = 5.0 \text{ signals/second}$. In order to obtain reasonable signal averaging, it is desirable to have at least 50 scans, and this would take approximately $1\frac{3}{4}$ hours.

Besides the slow sampling speed, this program had the further disadvantage of destroying all the original data, whenever the program was interrupted 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

1 km of Graded-Index Fibre

FWHM ~ Several ps

Input

FWHM = 100 ps

Amplitude

Time

Output
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 repetition 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**

---

**Power Supply**

**Pulse Generator**

**75 msec Delay**

**Laser**

**Fibre**

**Detector**

**Minicomputer**

**DAC2**

---

**Connections for Short Fibre**

**Connections for Long Fibre**

**ADC**

**DAC1**

**Display Scope**

**DAC0**

**Teletype**

**Floppy Disc**

---

**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.)
**Figure 3.5**

Signal After 2.7 m of Fibre

Amplitude

10 to 90% Rise Time = 135 ± 2 psec

**Figure 3.6**

Signal After 11.4 m of Fibre

Amplitude

Rise Time = 146 ± 2 psec
Signal After 20.7 m of Fibre

Amplitude

Rise Time = 159 ± 2 psec

Figure 3.7

Signal After 37.0 m of Fibre

Amplitude

Rise Time = 179 ± 2 psec

Figure 3.8
**Signal After 50.9 m of Fibre**

![Graph](image)

**Amplitude**

**Rise Time**

\[208 \pm 2 \text{ psec}\]

**Figure 3.9**

**Signal After 87.9 m of Fibre**

![Graph](image)

**Amplitude**

**Rise Time**

\[267 \pm 2 \text{ psec}\]

**Figure 3.10**
Undoubtedly, 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

\[ A = \exp\left(-\frac{1}{2}\left(\frac{t-t_m}{\Delta t}\right)^2\right) \]  

Lorentzian

\[ A = \frac{(\Delta t/2)^2}{(t-t_m)^2 + (\Delta t/2)^2} \]

where

- \( A \) = amplitude
- \( t \) = time
- \( t_m \) = time at the middle of the trace
  
  (ie. 320 psec)

\( \Delta t \) = parameter that determines the width of the response function.

In figure 3.13 is a graph showing the Gaussian of equation 3.1, for \( \Delta t = 20 \) 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 \( \Delta t \) has been changed to 60 psec and
**Gaussian with \( \Delta t = 20 \text{ psec} \)**

\[
A = \exp \left[ -\frac{1}{2} \left( \frac{t-t_m}{\Delta t} \right)^2 \right]
\]

- \( A \) = Amplitude
- \( t \) = Time
- \( t_m \) = Median Time of 320 psec

**Figure 3.13**

**Convoluted Output for Gaussian Fibre Response with \( \Delta t = 20 \text{ psec} \)**

**Figure 3.14**
**Gaussian With \( \Delta t = 60 \text{ psec} \)**

![Graph of Gaussian with \( \Delta t = 60 \text{ psec} \)]

**Figure 3.15**

**Convoluted Output for Gaussian Fibre Response With \( \Delta t = 60 \text{ psec} \)**

![Graph of convoluted output for Gaussian fibre response with \( \Delta t = 60 \text{ psec} \)]

**Figure 3.16**

- Rise Time = 222 ± 2 psec
**Gaussian With $\Delta t = 100$ psec**

![Graph of Gaussian Pulse]

**Figure 3.17**

**Convolved Output For Gaussian Fibre Response With $\Delta t = 100$ psec**

![Graph of Convolved Output]

**Figure 3.18**

- **Rise Time**: $300 \pm 2$ psec
100 psec respectively. Figures 3.19 and 3.20 show composite graphs of the Gaussians and the convoluted outputs for several $\Delta t$. As expected, and as indicated by the figures, the rise time of the convoluted outputs increases as $\Delta t$ 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 $\Delta t = 40$ 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**

![Graph showing Superimposed Gaussians of Different Amplitude](image)

**Figure 3.19**

**SUPERIMPOSED CONVOLUTED OUTPUTS FOR GAUSSIAN RESPONSES**

![Graph showing Superimposed Convoluted Outputs for Gaussian Responses](image)

**Figure 3.20**
Lorentzian with $\Delta t = 20$ psec

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

$A$: amplitude
$t$: time
$t_m$: median time or 320 psec

Figure 3.21

Convoluted output for Lorentzian fibre response with $\Delta t = 20$ psec

Rise time $= 156 \pm 2$ psec

Figure 3.22
**LORENTZIAN WITH $\Delta t = 60$ psec**

![Graph of Lorentzian Function](image1)

**Figure 3.23**

**CONVOLUTED OUTPUT FOR LORENTZIAN FIBRE RESPONSE WITH $\Delta t = 60$ psec**

![Graph of Convoluted Output](image2)

**Figure 3.24**
Lorentzian with $\Delta t = 100$ psec

![Graph of Lorentzian function with amplitude and time (psec) on the axes.]

**Figure 3.25**

Convulated output for Lorentzian Fibre Response with $\Delta t = 100$ psec

![Graph showing increasing amplitude over time (psec) with rise time labeled as $246 \pm 2$ psec.]

**Figure 3.26**
Superimposed Lorentzian of Different $\alpha$

**Figure 3.27**

Superimposed Convoluted Outputs for Lorentzian Responses

**Figure 3.28**
**MEASURED OUTPUT COMPARED WITH GAUSSIAN CONVOLuted OUTPUT**

![Diagram](image1)

**Figure 3.29**

**MEASURED OUTPUT COMPARED WITH LORENTZIAN CONVOLuted OUTPUT**

![Diagram](image2)

**Figure 3.30**
convoluted outputs, it was found to most closely match the one where $\Delta t$ 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 $\Delta t$, 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 $\Delta t$ corresponding to the equivalent 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 $\Delta t$ 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 $\Delta t$

Figure 3.31
GAUSSIAN $\Delta t$ VERSUS FIBRE LENGTH FOR SINGLE DAY

LEAST SQUARES FIT

\[ y = mx + b \]

\[ m = 0.900 \text{ psec/m} \]

\[ b = 0.945 \text{ psec} \]

Figure 3.32
Gaussian $\Delta t$ versus fibre length for ten days

Least squares fit

$y = mx + b$

$m = 0.937 \frac{\text{psec}}{\text{m}}$

$b = 10.37 \text{ psec}$

Figure 3.33
with a $\Delta t$ 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 $\Delta t$ 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,

$$A = \exp\left(-\frac{1}{2}\left(\frac{t-t_m}{\Delta t}\right)^2\right)$$

and a $\Delta t$ of 900 psec, we can now solve for the 10 to 90% rise time of the output pulse. i.e. Assume $t_m = 0$ psec. Then

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

$$t_{10\%} = - \sqrt{-2 \ln(0.1)} \times 900 \text{ psec}$$
$$= -1930 \text{ psec}$$

$$t_{90\%} = - \sqrt{-2 \ln(0.9)} \times 900 \text{ psec}$$
$$= -190 \text{ psec}$$

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 its input. Because of the difficulty of such an operation, it was instead decided 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 $\Delta t$, (from the equation $A = \exp\left(-\frac{1}{2}\left(\frac{t-t_m}{\Delta t}\right)^2\right)$), 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 $\Delta t$ 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, IRAYC(100, 3, 3), MODE, NSCNH, NSCNL, PLTFQ,
+ PLTFOR, SCNMOD, SNSCH(3, 2), SNSCL(3, 2), SPEED, SUMR, TYPE
DOUBLE PRECISION CPRAYC(465, 2), MAX, MIN, SCL, SMAX(3, 3)
REAL*4 TIME(3), WIDTH
DATA SMAX/99.0D0/SNSCH, SNSCL/120/
CALL SETUP(NSCNH, NSCNL, 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
GOTO 30

20 FORMAT (515)
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 OSCILLOGRAPH 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 OSCILLOGRAPH.*/
+ ' 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.*/
+ ' @ = CHARACTER USED TO STOP SAMPLING OR DISPLAYING.*/
)

TYPE 50
50 FORMAT ('$MODE?')
READ (5,20) MODE
GOTO (60, 210, 340, 370, 500) MODE

C
C MODEL - SIGNAL AVERAGE.
******************************

C
C TYPE 70
70 FORMAT ('$COL, TYPE, NSCNH, NSCNL?')
READ (5,20) COL, TYPE, NSCNH, NSCNL
CALL SIGAV(CRAYC(1, COL, TYPE))

C
C SIGAV IS AN ASSEMBLER ROUTINE THAT TAKES AN ADC
C READING FROM SCOPE, ADDS THIS VALUE TO IRAYC(1, COL,
C TYPE), AND ADDS THE CARRY BIT TO IRAYC(2, COL, TYPE).
C ETC. TILL 461 READINGS HAVE BEEN TAKEN. THEN IT
C DOES ANOTHER SCAN, ETC.
C
48
SNSCH(COL,TYPE) = NSCNH
SNSCL(COL,TYPE) = NSCNL
MAX = #D0
MIN = #D38
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 = DMAXI(MAX,DPRAY((J+1)/2,TYPE))
110   MIN = DMINI(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
   DPRAY(J,TYPE) = (DPRAY(J,TYPE)-MIN)/SCL
   
120   
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
       +IRAY(J,COL,TYPE)*"142000
130   
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=",218.5X,"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 = #D38
   MIN = #D38
   DO 180 J=5,465,1
      DPRAY(J,1) = DPRAY(J,1)-DPRAY(J,2)
      MAX = DMAXI(MAX,DPRAY(J,1))
   180   MIN = DMINI(MIN,DPRAY(J,1))
   
C
C DPRAY(J,1) NOW CONTAINS SIGNAL-NOISE.
SMAX(COL, 3) = MAX-MIN
SCL = 511/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)*142880
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 NODE2 - DISPLAY.
C
***************
C
210 TYPE 220
220 FORMAT ('SCOL,TYPE,SCNNOD,SPEED,SUMR')
READ (5,220) COL,TYPE,SCNMOD,SPEED,SUMR
IF ( SUMR .EQ. 0 ) GOTO 240
TYPE 230,SNH,SNH,COL,1),SNH(COL,2),
+ SNH(COL,2),SMAX(COL,1),SMAX(COL,2),SMAX(COL,3),TIME(COL)
230 FORMAT (28X,'TYPE1',14X,'TYPE2',14X,'TYPE3'/ NSCNH,NSCNL=','
+ TIME/CM=',E9.2/
240 IF ( SCNMOD .EQ. 1 ) GOTO 330
TYPE 250
250 FORMAT ('SPTFQ,PLTFQR,WIDTH')
READ (5,260) PLTFQ,PLTFQR,WIDTH
260 FORMAT (2110,F10.2)
CALL PLOT(50,511)
PAUSE 'LP'
VS1=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)
      CONTINUE
   DO 280 I=1,PLTFQR
      IWS2=50.+WIDTH-WIDTH*I/PLTFQR
      CALL PLOT(IWS2,IWS1)
      CONTINUE
290 CONTINUE
WS1=.15625
DO 320 J=1,PLTFQ
   IWS1=50.+461.*J/PLTFQ
   CALL PLOT(IWS1,3)
   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
  II$2=WIDTH-WIDTH*I/PLTFQR
  CALL PLOT(IWS1,IWS2)
310 CONTINUE
320 CONTINUE
PAUSE 'RP'
CALL PLOT(59,IRAY(5,COl,TYWE))
PAUSE 'LP'
330 CALL DISP(IRAY(5,COl,TYWE))
C
C DISP SIMPLY MOVES INTEGERS STORED IN IRAY TO
C VERTICAL DAC.
C
GOTO 40
C
C MODE3 - PUNCH OUTPUT.
C **********************
340 TYPE 350
350 FORMAT ('5COL,TYWE' )
READ (5,23) COl,TYWE
DO 355 J=1,25
  CALL PUNCHI(J)
355 CONTINUE
CALL PUNCHO('177777')
DO 360 J=5,465
  CALL PUNCHI(IRAY(J,COl,TYWE))
360 CONTINUE
GOTO 40
C
C MODE4 - PUNCH INPUT.
C ********************
370 TYPE 380
380 FORMAT ('5COL,TYWE' )
READ (5,20) COl,TYWE
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,TYWE))
410 CONTINUE
GOTO 40
C
C MODES - QUIT.
C ****************
C
500 END
GLOBL SETUP, SIGAV, DISP, PLOT, PUNCHO, PUNCHI

ANSCNL: I ;ADDRESS OF NSCNL.
ANSCNH: I ;ADDRESS OF NSCNH.
TNSCNL: I ;TEMPORARY STORAGE FOR
TNSCNH: I ; NSCN.
ASCNMOD: I ;ADDRESS OF SCNMOD.
ASPEED: I ;ADDRESS OF SPEED.
STORE: I ;TEMP STORAGE FOR SIGAV.
TTINT1: I ;ORIGINAL TTY INTERRUPT VECTOR
TTINT2: I ; WILL BE STORED HERE.

CNT: I ;USED IN DELAY LOOP.

SETUP: MOV #5337, @#132626 ;STOP CLOCK INTERRUPT.
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 INTERRUPT
MOV @#62, TTINT2 ; VECTORS.
RTS %7

SIGAV: MOV #TTYINT, @#60 ;WHEN TTY INTERRUPTS, 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 #3569, %2 ;%2 CONTAINS ADDR OF IRAY(922, COL).

L0: MOV #43300, (1)+ ;CLEAR COLUMN OF IRAY.
CMP %0, %1 ;
BNE L0 ;
MOV @L1, @#330 ;INITIALIZE A/D BY LETTING
MOV @103, @#167770 ; IT INTERRUPT.
BR L2 ;

L1: RTI ;

L2: MOV #SERVE, @#300 ;WHEN A/D INTERRUPTS, GOTO SERVE
MOV #340, @#302 ; WITH PRIORITY 7.
MOV @ANSCLN, TNSCNL ;STORE NSCN IN TNSCN.
MOV @ANSCLN, TNSCNH ;
ADD %1, TNSCNL ; INITIALIZE TNSCN TO COUNT
ADC TNSCNH ; # OF SCANS LEFT.

L3: MOV #144314, @#167772 ;POSITION HORIZ DAC-(621 V OFFSET)
MOV %0, CNT ; DELAY LOOP TO ALLOW
MOV @L3A, CNT ; VERT A/D TO STABILIZE.
CMP CNT, @203 ;
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 ; INTERRUPT VECTOR AND
RTS %7 ; RETURN TO MAIN PROGRAM.

L4: SVC TNSCNH ;SUBTRACT CARRY FROM TNSCNH.
MOV @#0, TNSCNL ;ALLOW TTY TO INTERRUPT.
MOV @#203, TNSCNL ;DISALLOW TTY TO INTERRUPT.
MOV #140014, X0 ;$0 CONTROLS HORIZ DAC, \(62^2\) V OFFSET
MOV (Z5), X1 ;$1 CONTAINS ADDR OF IRAY(1, COL).

L5: ADD #1, X3 ;INCR HORIZ DAC POSITION.
CMP %0, #145000 ;IS SCAN COMPLETE?
BEQ L3 ;IF YES, START NEW SCAN.
MOV @#167774, STORE ;STORE DATA AND ENABLE INTERRUPT.

WAIT ;WAIT FOR INTERRUPT.
BR L5 ;INTERRUPT COMPLETE, GOTO L5.

SERVE: MOV %0, @#167772 ;OUTPUT HORIZ DAC POSITION.
MOV STORE, %2 ;RETRIEVE DATA FROM LAST HORIZ DAC POS.
BPL L6 ;IF POS, GOTO L6.
NEG %2 ;ADD DATA TO IRAY.
SUB %2, (X1)+ ;
SGC (X1)+ ;
RTI ;RETURN FROM INTERRUPT.

L6: ADD %2, (X1)+ ;ADD DATA TO IRAY.
ADC (X1)+ ;
RTI ;RETURN FROM INTERRUPT.

TTYINT: CMPB #300, @#177562 ;IS TTY INTERRUPTING WITH "@"?
BEQ L7 ;IF YES, PROCESS AT L7.
RTI ;IF NO, RETURN FROM INTERRUPT.

L7: MOV TINT1, #60 ;RESTORE ORIGINAL TTY INTERRUPT
MOV TINT2, #62 ;VECTOR.
SUB TNSCNL, @ANSCNL ;IF INTERRUPTED FROM SIGAV.
SBC @ANSCNH ;THEN NSCN CONTAINS # OF SCANs COMPLETED.
MOV #765, #16 ;SET STACK POINTER TO RETURN
RTS ;TO MAIN PROGRAM.

DISP: MOV #0, @#17776 ;RUN SUBROUTINE AT PRIORITY 0.
MOV @TTYINT, #60 ;WHEN TTY INTERRUPTS, GOTO
MOV #340, #62 ;TTYINT WITH PRIORITY 7.
ADD #2, #5 ;%5 POINTS AT ADDRESS OF FrAY(1, COL).
L8: MOV (Z5), %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 #14, @#167772 ;OUTPUT HORIZONTAL AND VERTICAL
MOV (X1)+, @#167772 ;DAC POSITIONS.
MOV #0, CNT ;INITIALIZE DELAY VARIABLE.

L10: INC CNT ;SLOW DOWN SCAN WITH DELAY
CMP CNT, @ASPEED ;LOOP.
BNE L18 ;
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 TINT1, #60 ;NO, RESTORE TTY INTERRUPT
MOV TINT2, #62 ;VECTORS AND QUIT
RTS ;DISPLAYING.

PLOT: MOV #0, @#177776 ;RUN SUBROUTINE AT PRIORITY 0.
MOV @TTYINT, #60 ;WHEN TTY INTERRUPTS, GOTO
MOV #340, #62 ;TTYINT WITH PRIORITY 7.
MOV (Z5)+, #3 ;%3 CONTAINS # OF POINTS.
L11: BIS #2000, (X5) ;CLEAR POSSIBLE VERT DAC ADDRESS.
BIS #140000, (X5) ;APPEND HORIZ DAC ADDRESS.
MOV @ (X5)+, @#167772 ;OUTPUT HORIZ DAC POSITION.
BIS #142000, (X5) ;APPEND VERT DAC ADDRESS.
MOV @(5),#@167772  ;OUTPUT VERT DAC POSITION.
MOV @6, CNT      ;DELAY LOOP.
L12: INC CNT
    CMP CNT, ASPEED
    BNE L12
    SUB #2, X3
    BNE L11
    MOV TTINT1, #68  ;IF PNTS NOT ALL PLOTTED, CONT.
    MOV TTINT2, #62  ;INTERUPT VECTORS.
    RTS #7
    ;PLOTTING COMPLETE.
PUNCHO: ADD #2, X5
    MOV @(5), #4
    MOV #177564, #3
    BIC #104, #177564
    RTS #7
    ;%5 PNTS. TO ADD. OF IRAY(J, COL, TYPE).
    ;%4 CONTAINS IRAY(J, COL, TYPE).
    ;STORE PUNCH STATUS REGISTER.
    ;CLEAR INTERRUPT AND MAINTENANCE.
    ;TEST IF PUNCH IS FREE.
    ;IF PUNCH NOT FREE, GOTO P01.
    MOV %4, #177566
    SWAB %4
    ;OUTPUT SECOND BYTE OF IRAY(J, COL, TYPE).
    ;SWAP BYTES OF %4.
P01: BIT #200, #177564
    BEQ P01
    MOV %4, #177566
    SWAB %4
    ;OUTPUT FIRST BYTE OF IRAY(J, COL, TYPE).
    ;RESTORE PUNCH STATUS REGISTER.
    ;TEST IF PUNCH IS FREE.
    ;IF PUNCH NOT FREE, GOTO P02.
P02: BIT #200, #177564
    BEQ P02
    MOV %4, #177566
    SWAB %4
    ;RESTORE PUNCH STATUS REGISTER.
    ;TEST IF PUNCH IS FREE.
    ;IF PUNCH NOT FREE, GOTO P02.
    MOV %4, #177566
    SWAB %4
    ;OUTPUT FIRST BYTE OF IRAY(J, COL, TYPE).
    ;RESTORE PUNCH STATUS REGISTER.
    ;TEST IF PUNCH IS FREE.
    ;IF PUNCH NOT FREE, GOTO P02.
PUNCHI: ADD #2, X5
    MOV #177560, #3
    BIC #100, #177560
    RTS #7
    ;%5 PNTS. TO ADD. OF IRAY(J, COL, TYPE).
    ;STORE READER STATUS REGISTER.
    ;CLEAR INTERRUPT.
    ;ENABLE READER.
    ;TEST IF PUNCH IS FREE.
    ;IF NOT, GOTO P11.
P11: BIT #200, #177560
    BEQ P11
    MOV #177562, @(5)
    ;MOVE SECOND BYTE TO IRAY(J, COL, TYPE).
    ;SWAP BYTES OF IRAY(J, COL, TYPE).
    ;ENABLE READER.
P12: BIT #200, #177560
    BEQ P12
    MOV #177562, @(5)
    ;MOVE FIRST BYTE TO IRAY(J, COL, TYPE).
    ;SWAP BYTES OF IRAY(J, COL, TYPE).
    ;RESTORE READER STATUS REGISTER.
    ;DISABLE READER.
    ;TEST IF PUNCH IS FREE.
    ;IF NOT, GOTO P12.
    MOV #177562, @(5)
    ;MOVE FIRST BYTE TO IRAY(J, COL, TYPE).
    ;SWAP BYTES OF IRAY(J, COL, TYPE).
    ;RESTORE READER STATUS REGISTER.
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 monitor 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. Computer is asking which one.

USER: 1 ; MODE1 is signal average.

COMP: COL, TYPE, NSCNH, NSCNL? ; computer now needs to know into which column the signal should be placed, 1, 2, or 3, what type of averaging 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 interrupt 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 un-calibrated 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.
1

COMP: MODE?

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

COMP: COL,TYPE,SCNMOD,SPEED,SUMR?

;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.

2

;referring to code, user decides to display a signal.

;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 execute 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 APMAX,APMIN,BPNT,CODE,COL,DELAY,HDI,IRAY(512,2,3),
+ IRAYS(1024,2,3),MODE,NSCNH,NSCNL,PLTFO,SCALE,SMAX(2,3),
+ SNSCN(2,2),SNSCNL(2,2),SPEED,SPMAX,SPMIN,SUMR,TYPE
REAL DPRAY(512,2),MAX,MIN,5CL
DATA SMAX,NSCNH,NSCNL/14*0/
CALL SETUP(APMIN,APMAX,BPNT,DELAY,HDI,MODE,NSCNH,NSCNL,SPEED,SPMIN)

TYPE 10
10 FORMAT('#CODE?')
READ (5,20) CODE
20 FORMAT(7I0)
IF ( CODE .EQ. 0 ) GOTO 50

TYPE 30
30 FORMAT('# = NO'/! = 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.'/
+ CODE8 = DO NOT PRINT THIS CODE TO USING PROGRAM.'/
+ CODE9 = 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.'/
+ PLTFO = 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=51.'/
+ SPMIN = MINIMUM POINT TO BE SIGNAL AVERAGED. DEFAULT=31.'/
+ SUMR = DO NOT OUTPUT SUMMARY OF RESULTS.'/
+ SUMR = OUTPUT SUMMARY OF RESULTS.'/
+ TYPE1 = SIGNAL.'/
+ TYPE2 = NOISE.'/
+ TYPE3 = SIGNAL-NOISE.'/
+ @ = CHARACTER USED TO STOP SAMPLING OR DISPLAYING.')

APHI=62
APMAX=512
APMIN=62
BPNT=31
DELAY=1
HDI=1
SPMAX=511
SPMIN=31

TYPE 70
70 FORMAT('#$MODE?')
READ (5,20) MODE
GOTO (80,80,300,300,500,500,600,700,800) MODE
**MODE1 AND MODE2 - SIGAV**

***************

**TYPE 90**

90 FORMAT ('$SCOL,NSCHH,NSCNL,SCALE,TYPE?')
READ (5,20) COL,NSCHH,NSCNL,SCALE,TYPE
IF ( SCALE .EQ. 0 ) GOTO 105
**TYPE 100**

100 FORMAT ('$APMAX,APMIN,BPNT,DELAY,HD1,SPMAX,SPMIN?')
READ (5,20) APMAX,APMIN,BPNT,DELAY,HD1,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**
**SIGAV IS AN ASSEMBLER ROUTINE THAT TAKES SUCCESSIVE ADC**
**READINGS FROM SCOPE, ADDS THEM TO IRAYS (STARTING AT**
**IRAYS(SMIN,COL,TYPE)), ADDS CARRY TO NEXT ROW ELEMENT OF**
**IRAYS, CONTINUING TO IRAYS(SPMAX,COL,TYPE).**

**C**
**IF**
**MODE**
**.EQ. 1**
**GOTO 110**

SNSCHH(COL,TYPE)=SNSCHH(COL,TYPE)+NSCHH
SNSCNL(COL,TYPE)=SNSCNL(COL,TYPE)+NSCNL
GOTO 115

110 SNSCHH(COL,TYPE)=NSCHH
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=ANAXI(MAX,DPRAY(J,TYPE))
MIN=ANAXI(MIN,DPRAY(J,TYPE))

120 MAX=ANAXI(MAX,DPRAY(J,TYPE))
MIN=ANAXI(MIN,DPRAY(J,TYPE))

**ABOVE LOOP TRANSFERRED SIGNAL IN IRAYS (AN INTEGER ARRAY**
FOR SIGNAL AVERAGING WHERE EACH DATA POINT NEEDED A LOW AND
HIGH ORDER NUMBER ) TO DPRAY (A DOUBLE PRECISION ARRAY,
WHERE EACH POINT ONLY NEEDS ONE NUMBER ). LOOP ALSO FOUND
MAX AND MIN FOR LATER USE.

SCL=SNSCHH(COL,TYPE)*65536.*SNSCNL(COL,TYPE)
DO 130 J=APMIN,APMAX
DPRAY(J,TYPE)=(DPRAY(J,TYPE)-MIN)/SCL

130 SCL=SNSCHH(COL,TYPE)*65536.*SNSCNL(COL,TYPE)
DO 130 J=APMIN,APMAX
DPRAY(J,TYPE)=(DPRAY(J,TYPE)-MIN)/SCL

**ABOVE LOOP REMOVED DC OFFSET FROM SIGNAL AND DIVIDED SIGNAL**
**BY NUMBER OF SCANS.**

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
IRAYD(J,COL,TYPE)=IRAYD(J,COL,TYPE)*142000

140 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
IRAYD(J,COL,TYPE)=IRAYD(J,COL,TYPE)*142000

**ABOVE LOOP SCALED SIGNAL IN DPRAY TO FIT 10 BIT DAC, PUT IT**
**INTO IRAYD, AND ADDED VERTICAL DAC ADDRESS.**
TYPE 150, SNSCNH(COL, TYPE), SNSCNL(COL, TYPE), SMAX(COL, TYPE)
FORMAT (' NSCN=',2I8,' PH=',I8)
IF ( TYPE .EQ. 1 ) GOTO 60
C
BOTH SIGNAL AND NOISE SCANS HAVE BEEN TAKEN IF TYPE=2.
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=MAX(MAX,DPRAY(J,1))
   MIN=MIN(MIN,DPRAY(J,1))
160
DPRAY(J,1) NOW CONTAINS SIGNAL-NOISE.
C
SMAX(COL,3)=MAX-MIN
SCL=511.E8/SMAX(COL,3)
DO 170 J=APMIN,APMAX
   IRAYD(J,COL,3)=(DPRAY(J,1)-MIN)*SCL
   IRAYD(J,COL,3)=IRAYD(J,COL,3)+142300
170
IRAYD(J,COL,3) NOW CONTAINS SIGNAL-NOISE SCALED TO 10 BITS
WITH VERTICAL CAC ADDRESS ADDED.
C
TYPE 153, SMAX(COL,3)
FORMAT (' DPHE=',FS,2)
GOTO 60
C
C
MODE3 AND MODE4 - SIGMEL.
C
***********************************************************
C
TYPE 100
READ (5,20) COL,NSCNH,NSCNL,TYPE
IF ( MODE .EQ. 3 ) CALL CLEAR(IRAYS(1,COL,TYPE))
CALL SIGMEL(IRAYS(SPMAX+2+I,COL,TYPE),IRAYS(SPMIN+2+I,COL,TYPE))
C
SIGMEL IS AN ASSEMBLER ROUTINE LIKE SIGAV, BUT BETWEEN EACH
SIGNAL POINT, IT SAMPLES BPNT (BASELINE POINT) AND SUBTRACTS
IT FROM SIGNAL TO REMOVE BASELINE DRIFT.
C
GOTO 110
C
C
MODE5 AND MODE6 - DISPLAY.
C
***********************************************************
C
TYPE 510
510 FORMAT ('$COL,SPEED,SUMR>Type?')
READ (5,20) COL,SPEED,SUMR,TYPE
IF ( SUMR .EQ. 0 ) GOTO 530
C
520 FORMAT (28X,'TYPE1=',1X,'TYPE2=',1X,'TYPE3=',1X,'TYPE4=',1X,'TYPE5=',1X,'TYPE6=',1X)
+ 16X,16,S9X,16'PH=',I2,4X,1I9,1I9)
IF ( MODE .EQ. 5 ) GOTO 560
CALL PLOT(SPMIN,511)
TYPE 535

FORMAT ('$PLTFQ?')
READ (5,20) PLTFQ
PAUSE 'LP'
DO 540 J=1,PLTFQ
   IVS=511,-511,*J/PLTFQ
   CALL PLOT(SPMIN,IVS)
CONTINUE
DO 550 J=1,PLTFQ
   IVS=SPMIN+1.*J*(SPMAX-SPMIN)*HD1/PLTFQ
   CALL PLOT(IVS,0)
CONTINUE
PAUSE 'RP'
CALL PLOT(APMIN-1,IRAYD(APMIN,COL,TYPE))
PAUSE 'LP'
CALL DSPL(IRAYD(APMAX,COL,TYPE),IRAYD(APMIN,COL,TYPE))
GOTO 60
C C
C MODE7 - PNCHOU.
C ***************
C 610 TYPE 610

610 FORMAT ('$COL,TYPE?')
READ (5,20) COL,TYPE
DO 615 J=1,25
   CALL PNCHOU(I)
CONTINUE
CALL PNCHOU("17777")
CALL PNCHOU(APMAX)
CALL PNCHOU(APMIN)
CALL PNCHOU(BPNT)
CALL PNCHOU(DELAY)
CALL PNCHOU(HDI)
CALL PNCHOU(SNSCNH(COL,TYPE))
CALL PNCHOU(SNSCNL(COL,TYPE))
CALL PNCHOU(SMAX(COL,TYPE))
CALL PNCHOU(SPMAX)
CALL PNCHOU(SPMIN)
DO 620 J=APMIN,APMAX
   CALL PNCHOU(IRAYD(J,COL,TYPE))
CONTINUE
PAUSE 'PT'
630 TYPE 640,APMAX,APMIN,BPNT,DELAY,HDI,SNSCNH(COL,TYPE),
   + SNSCNL(COL,TYPE),SMAX(COL,TYPE),SPMAX,SPMIN
640 FORMAT ('$AMAX =',I7,' $APMIN =',I7,' $BPNT =',I7,' $DELAY =',I7,' $HDI =',I7,
   + ' $SNSCNH =',I7,' $SNSCNL =',I7,' $SMAX =',I7,' $SPMAX =',I7,
   + ' $SPMIN =',I7)
GOTO 60
C C
C MODES - 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
    CONTINUE
710 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  MODE9 - QUIT.
C
C  *************
C
803 END
*GLOBL CLEAR, DSPL, PLOT, PNCCHU, PNCIN, SETUP, SGMB, SIGAV

AAPMIN: #ADDRESS OF APMIN, ETC.

ABPNT: #

ADELAY: #

ANDI: #

AMODE: #

ANSCNH: #

ANSCNL: #

ASPEED: #

ASPMIN: #

CNT: #

STORE: #STORES SIGNAL DURING ADC INTERRUPT.

TNSCNH: #TEMPORARY STORAGE FOR NSCN.

TNSCNL: #

TTINT1: #ORIGINAL TTY INTERRUPT VECTOR IS STORED HERE.

TTINT2: #

CLEAR: ADD %2, %5 #IS PNTS. ADD OF IRAYS(1, COL, TYPE).

MOV (5), %20 #%20 CNTNS. ADD OF IRAYS(1, COL, TYPE).

MOV (5), %26 #%26 CNTNS. ADD OF IRAYS(1, COL, TYPE).

ADD #3776, %1 #%1 CNTNS. ADD OF IRAYS(1024, COL, TYPE).

CLI: MOV #00000, %0 #CLEAR COLUMN OF IRAYS.

CMP %9, %1 #

BLE CLI #

RTS %7 #RETURN TO MAIN PROGRAM.

DSPL: MOV #0, #177776 #RUN SUBROUTINE AT PRIORITY 3.

MOV #TTYINT, %6 #WHEN TTY INTERRUPTS, GO TO

MOV #340, 62 #TTYINT WITH PRIORITY 7.

MOV #AAPMIN, %2 #PUT APMIN IN %2.

ADD #137777, %3 #ADD HORIZ. DAC ADD. (APMIN=1).

SUB #AHD1, %3 #SUBTRACT HDI.

ADD %2, %5 #IS PNTS. ADD OF IRAYD(APMAX, COL, TYPE)

MOV (5), %26 #%26 CNTNS. ADD OF IRAYD(APMAX, COL, TYPE)

DS1: MOV (5), %2 #%2 CNTNS. ADD OF IRAYE(APMIN, COL, TYPE)

MOV %3, %3 #PUT FIRST HORIZ. DAC POS. IN %3.

DS2: ADD #AHD1, %3 #INCREMENT HORIZ. DAC.

MOV %2, #167772 #OUTPUT HORIZ. DAC POSITION.

MOV (5), #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?

CJN TPINT1, %6 #IF NO. CONTINUE SCAN.

BNE DS1 #IF YES. NEED MORE THAN 1 SCAN?

MOV TTINT1, %60 #YES. KEEP SCANNING.

MOV TTINT2, %62 #NO. RESTORE TTY INTERRUPT

MOV %7, %7 #VECTORS AND QUIT

RTS %7 #DISPLAYING.

PLOT: ADD %2, %5 #IS PNTS. ADD OF X POSITION.

MOV @(#15)+, %4 #MOVE X TO %4.

BIS #140000, %4 #SET HORIZ. DAC ADD.

MOV %4, #167772 #OUTPUT X.

MOV @(#15), %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 #100,%#177564  ;CLEAR INTERRUPT 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 %3,#177566  ;OUTPUT FIRST BYTE OF WORD.
RTS %7  ;

PNCHIN: ADD #2,%5  ;#5 PNTS # ADD. OF WORD REQUIRING INPUT.
MOV @#177568,%3  ;STORE READER STATUS REGISTER.
BIC #100,%#177568  ;CLEAR INTERRUPT.
BIS #1,%#177568  ;ENABLE READER.
PI1: BIT #200,%#177568  ;IS READER DONE?
BEQ PI1  ;IF NOT, GOTO PI1.
MOV @#177564,%(15)  ;MOVE 2ND BYTE TO WORD REQUIRING INPUT.
SWAB @(%5)  ;SWAP BYTES OF WORD REQUIRING INPUT.
BIS #1,%#177568  ;ENABLE READER.
PI2: BIT #200,%#177568  ;IS READER DONE?
BEQ PI2  ;IF NOT, GOTO PI2.
MOV @#177562,%(15)  ;MOVE 1ST BYTE TO WORD REQUIRING INPUT.
SWAB @(%5)  ;SWAP BYTES OF WORD REQUIRING INPUT.
MOV %3,#177568  ;RESTORE READER STATUS REGISTER.
RTS %7  ;

SETUP: MOV #5037,#132626  ;STOP CLOCK INTERRUPT.
MOV #177564,#132630  ;
MOV #2,#132632  ;
ADD #2,%5  ;#5 PNTS. # ADD. OF APMIN.
MOV (15)+,AAPMIN  ;AAPMIN CNTNS. ADD. OF APMIN. ETC.
MOV (15)+,ABPNT  ;
MOV (15)+,ADELAY  ;
MOV (15)+,AHD1  ;
MOV (15)+,ANGE  ;
MOV (15)+,ANSCNH  ;
MOV (15)+,ANSCNL  ;
MOV (15)+,ASPEED  ;
MOV (15)+,ASPMIN  ;
MOV @#60,TTINT:  ;SAVE FORTRAN TTY INTERRUPT
MOV @#62,TTINT2  ;VECTOR.
RTS %7  ;

SGMBL: RTS %7  ;INSERT SUBROUTINE HERE.

SIGAV: MOV #200,#177776  ;RUN SUBROUTINE # PRIORITY 7.
MOV #11,#300  ;INITIALIZE ADC BY LETTING IT INTERUPT.
MOV #100,#167770  ;
BR S12  ;

S11: RTI  ;RETURN FROM ADC INTERRUPT.
S12: MOV #TTYINT,#60  ;WHEN TTY INTERRUPTS, GOTO
MOV #348,#62  ;TTYINT WITH PRIORITY 7.
MOV #519,#300  ;WHEN ADC INTERRUPTS, GOTO
MOV #348,#302  ;S19 WITH PRIORITY 7.
MOV @ANSCH1, TNSCNH, STORE NSCN IN TNSCN.
MOV @ANSCHL, TNSCNL.
ADD #1, TNSCNL, INITIALIZE TNSCN TO COUNT.
ADC TNSCNH, #0 OF SCANS LEFT.
MOV @ASPMIN, #4, ADD SPMIN SPMIN.
ADD #140000, #4, APPEND HORIZ. S. SCOPE DAC ADDRESS.
ADD #2, #5, #5 PNTS. # ADD. IF IRAYSPMAX.COL.TYP.
MOV #5, #3, #3 CNTNS. # ADD. IF IRAYSPMAX.COL.TYP.
BR S15.

BPL S14.
NEG #2.
SUB #2, (R1),
SBC (R1),
BR S15.

S14: ADD #2, (R1),
ADC (R1),

S15: MOV R13, #4, ADD SPMIN SPMIN + HORIZ. DAC ADD.
MOV R13, #9, #167772, POSITION HORIZ. DAC.
MOV (R5), #1, #1 CNTNS. ADD. OF IRAYSPMAX.COL.TYP.
MOV #3, CNT.
DELAY LOOP TO ALLOW VERTICAL.

S16: INC CNT, #1, SEC TO STABILIZE.
CMP CNT, #0, #0, DENE #167776.
DNE S16.
SUB #1, TNSCNL, DECREMENT # OF SCANS REMAINING.
BNE S17.
TST TNSCNH, TEST TNSCNH.
BNE S17.
JIF TNSCNH, #0, GOTO S17.
MOV TTYINT1, #60, #60, TNSCN=0. RESTORE FORTRAN.
MOV TTYINT2, #62, #62, TTY INTERRUPT VECTOR.
RTS S7.

S17: MOV TNSCNH, #1, TNSCNH, SUBTRACT CARRY FROM TNSCNH.
MOV #3, #167776, SUBTRACT CARRY FROM TNSCNH.
MOV #3, #167776, JALLOW TTY TO INTERRUPT.

S18: CMP R1, #3, #3, JIS SCAN COMPLETE?
BEQ S13.
JIF YES, FINISH OLD AND START NEW SCAN.
MOV #167776, STORE #167776, STORE DATA AND ENABLE INTERRUPT.
WAIT JWAIT FOR INTERRUPT.

S19: ADD @AHD1, #8, INCREMENT HORIZ. POSITION.
MOV R13, #9, #167772, OUTPUT NEW HORIZ. POSITION.
MOV STORE, #2, RETRIEVE DATA FROM OLD HORIZ. POSITION.
BPL S110.
JIF DATA IS POSITIVE, GOTO S110.
NEG #2.
ADD #2, (R1),
SBC (R1),
RTI.

S110: ADD #2, (R1),
ADC (R1),
RTI.
RETURN FROM ADC INTERRUPT.

TTYINT: CMP R13, #3000, #3000, #3000, #3000, #3000, #3000.
BEQ TTY1.
JIS TTY INTERRUPTING WITH "8"?
RTI.
JIF YES, PROCESS AT TTY1.

TTY1: MOV TTYINT1, #60, #60, TTYINTERUPT VECTOR.
MOV TTYINT2, #62, #62, TTYINT2.
SUB TNSCNL, @ANSCHL, IF INTERRUPTED FROM SIGAV.
SBC @ANSCHH, @ANSCHH, THEN NSCN CONTAINS
SUB TNSCNH, @ANSCHH, #0 OF SCANS COMPLETED.
MOV #764, #26, SET STACK POINTER TO RETURN.
RTS S7.
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
C
TYPE 35
35 FORMAT ('$MODE,TYPE?')
READ (5,37) MODE,TYPE
37 FORMAT (2I8,F8.1)
GOTO (40,150,160,180) MODE
C C
C
MODE1 - CALCULATE RESPONSE AND CONVOLUTION.
************************************************************
C C
C
TYPE 50
50 FORMAT ('$RES,SD?')
READ (5,60) RES,SD
60 FORMAT (110,F10.2)
IF ( RES .EQ. 2 ) GOTO 80
DO 70 J=1,501
70   RSPNS(J)=EXP(-5*((J-251.)*SD)**2.)
GOTO 130
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
   WSI=0.
   DO 120 I=1,501
   WSI=WSI+RSPNS(I)*SGNL(I+J-1)
   CONV(J)=WSI
   MAX=AMAX1(MAX,WSI)
130   MIN=AMAX1(MIN,WSI))
   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
C
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'
WSI=0.
DO 520 J=1,PLTFQ
    IWS1=511.-511.*J/PLTFQ
    CALL PLOT(IWS1,WSI)
    IF ( FLOAT(J)/PLTFQ .LT. WSI ) GOTO 520
    WSI=WSI+.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
    CONTINUE
WSI=.15625
DO 550 J=1,PLTFQ
    IWS1=511.-461.*J/PLTFQ
    CALL PLOT(IWS1,0)
    IF ( FLOAT(J)/PLTFQ .LT. WSI ) GOTO 550
    WSI=WSI+.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
    CONTINUE
PAUSE 'RP'
CALL PLOT(50,OUT(I,TYPE))
PAUSE 'LP'
240 CALL DISP(OUT(I,TYPE))
GOTO 33

C
C MODE3 - PUNCH.
C ***************
C
160 DO 170 J=1,461
170 CALL PUNCH0(OUT(J,TYPE))
GOTO 33

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 accommodate 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.