To the Graduate Council:

I am submitting herewith a thesis written by Sharon Neufeldt Carter entitled "Data Acquisition and Reduction Program for Measuring the Phase Shift and Period of Two-Photodetector Signals for Estimating the Size and Velocity of Jet Fuel Particles." I have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Electrical Engineering.

Bruce W. Bomar

Dr. Bruce W. Bomar, Major Professor

We have read this thesis
and recommend its acceptance:

Roy S Joseph

L Montgomery Smith

Accepted for the Council:

Associate Vice Chancellor
and Dean of The Graduate School

Data Acquisition and Reduction Program for Estimating the Size and Velocity of Jet Fuel Particles

A Thesis

Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

Sharon Neufeldt Carter

May 1995

Acknowledgments

The author would like to thank the people who helped in the preparation of this thesis including Dr. Bruce Bomar who acted as principle thesis advisor. She would also like to thank the members of the committee, Dr. Roy Joseph and Dr. Montgomery Smith.

The author would also like to thank James Hornkohl, Newton Wright, Fred Schwartz, and Anthony Alford who were instrumental in the completion of this thesis.

Finally, the author would like to thank her husband, Steve, and her parents, Harvey and Anne Neufeldt who through their support and encouragement made the completion of such a task possible.

Abstract

The Center for Laser Applications at The University of Space Institute required a computer program to calculate the period and phase shift between two digitized photodetector signals. The phase shift and period can be used to determine the size and the velocity of jet fuel particles falling through a laser probe volume.

Particles are dropped in the probe volume where two laser beams intersect. A particle scatters light as it passes through the probe volume and the scattered light is detected by two photodetectors. The photodetector signals are digitized by 8-bit analog-to-digital converters and read into the RAM of two digital signal processing cards which can hold the data from approximately 1000 particles.

The data placed in RAM is used to determine the phase shift and period using the crosscorrelation function. The limitations on using the crosscorrelation for this purpose are discussed as well as methods to reduce the error introduced during the data reduction process.

The performance of the data acquisition system was verified by checking that the DSP cards were triggering simultaneously so that no erroneous phase shift is introduced during data acquisition. This was accomplished by introducing simulated signals into the DSP cards via different length cables. The time lag was calculated and it was determined that the cards were functioning correctly.

Program checkout consisted of verifying the code on simulated data and determining under what conditions the program would give valid results. Simulated signals

similar to the signals produced by the experiment were input into the data reduction portion of the program. The signal visibility and background light power were varied to produce signals with different levels of noise. These signals were input to the program to test the conditions under which the program would fail. The program yielded accurate results if the signal visibility is greater than 75% and the background light power is less than 25%.

It was determined that an accurate phase and period measurement results if the number of periods used in the crosscorrelation calculation is between five and twenty. If the signals do not contain five periods, there are not enough periods to calculate an accurate crosscorrelation. If there are more than twenty periods, the determination of the peaks is difficult because they may fall between two points on the crosscorrelation curve. The peaks of the crosscorrelation curve are crucial in the determination of phase and period.

Table of Contents

CHAPTER 1 INTRODUCTION **1**

CHAPTER 2 EXPERIMENT THEORY AND CONFIGURATION **4**

CHAPTER 3 PROGRAM ORGANIZATION **10**

PROGRAM CONFIGURATION 10

MIXED LANGUAGE PROGRAMMING 11

CHAPTER 4 DATA ACQUISITION **15**

THEORY 15

IMPLEMENTATION 16

CHAPTER 5 DATA REDUCTION **19**

PHASE AND PERIOD DETERMINATION 19

DATA INTERPOLATION 24

ERROR MINIMIZATION 25

CHAPTER 6 DATA SYSTEM CHECKOUT **36**

CHAPTER 7 PROGRAM CHECKOUT **39**

PROGRAM VALIDITY 39

PROGRAM LIMITS 42

CHAPTER 8 CONCLUSIONS **49**

REFERENCES **51**

APPENDICES **53**

APPENDIX A.....54

REDUCE3.FOR.....55

RED2INTT.FOR.....67

APPENDIX B.....80

Search.c.....81

Setseg.c.....82

Getdat.c.....83

Setup.c.....84

VITA.....86

FIGURES

Figure		Page
2.1	Laser Configuration	4
2.2	Receiving Apparatus	6
2.3	Signal Generated by Particle in the Probe Volume	7
2.4	Scattering Light from a Particle	8
2.5	Phase vs. Particle Size for Water Droplets Calculated by Durst and Naqwi	9
3.1	Flowchart for Mixed Language Programming	13
5.1	Signals X and Y	20
5.2	Crosscorrelation of signals x and y	21
5.3	Crosscorrelation Sequence Calculated in the Frequency Domain	23
5.4	Flowchart for REDUCE3.FOR	28
5.5	Flowchart for RED2INTT.FOR	32
6.1	Frequency Generator Output with Same Length Cable, Clock Rate 12.5 MHz	37
6.2	Frequency Generator Output with Cable Length Difference of 85 Feet, Clock Rate 50 MHz	38
7.1	150 Samples of the Crosscorrelation	40
7.2	100 Samples of X and Y	41
7.3	Interpolation of Input Data	43
7.4	Generated Signals with Noise	47

Chapter 1

Introduction

The Center for Laser Applications (CLA) located at The University of Tennessee Space Institute is developing a laser-based system to determine the optimal size of jet fuel particles for combustion. The speed and size of the particles can be measured using laser techniques. Therefore, the CLA required a program to determine the period of two signals and the phase shift between them; the size of jet fuel particles can be determined from this phase shift and the velocity of the particle can be determined from the period of two photodetector signals [1-3]. The premise for the correlation between phase shift and size and period and velocity will be discussed in Chapter 2. The objective of this thesis was to develop the program necessary to acquire the data from the photodetectors and perform this phase shift and period calculation. The program has two major parts: data acquisition and data reduction. Two versions of this program exist: one interpolates the data before data reduction takes place and the other does not. Interpolation reduces error if the particles are moving at high velocity.

The data acquisition portion digitizes the signals received from the photodetectors. The data reduction portion then calculates the crosscorrelation function to determine the phase and period of two signals. The practical limitations of this method are discussed in Chapter 5.

Crosscorrelation is a mathematical operation on two signals with the objective of measuring the degree to which the two signals are similar. If the input signals have the

same period, their average product will be the greatest when the lag between the signals is zero [12]. Therefore, the first peak of the crosscorrelation will occur at the sample number of the lag between two signals and the crosscorrelation function will be periodic with a period that is the same as that of the input signals [7,12].

The measurement of this lag or time delay is used in several applications. In radar or sonar, the crosscorrelation is computed between the transmitted and received signals. The position of a target can be determined by obtaining the time delay from the crosscorrelation [7]. Another application is determination of the time required for a signal to pass through a given system by calculating the crosscorrelation between the input and output signals [12].

The crosscorrelation does more than measure a time delay. It can act as a filter because the crosscorrelation function removes most of the noise contained in the input sequences. If a signal is buried in noise and this signal is known, a crosscorrelation calculated between the noisy signal and the known signal will extract the data from the noise [12].

The method used in this thesis is similar to other methods that determine time delay from the crosscorrelation function. The inputs to the crosscorrelation operation will be the signals from the two photodetectors. The lag will determine the phase shift and thus the size of the particles. However, the method will also calculate the period from the crosscorrelation function. The method used in this thesis will also take advantage of the fact that the crosscorrelation filters most of the noise from the input sequences before phase shift and period determination. This will be discussed in Chapter 5.

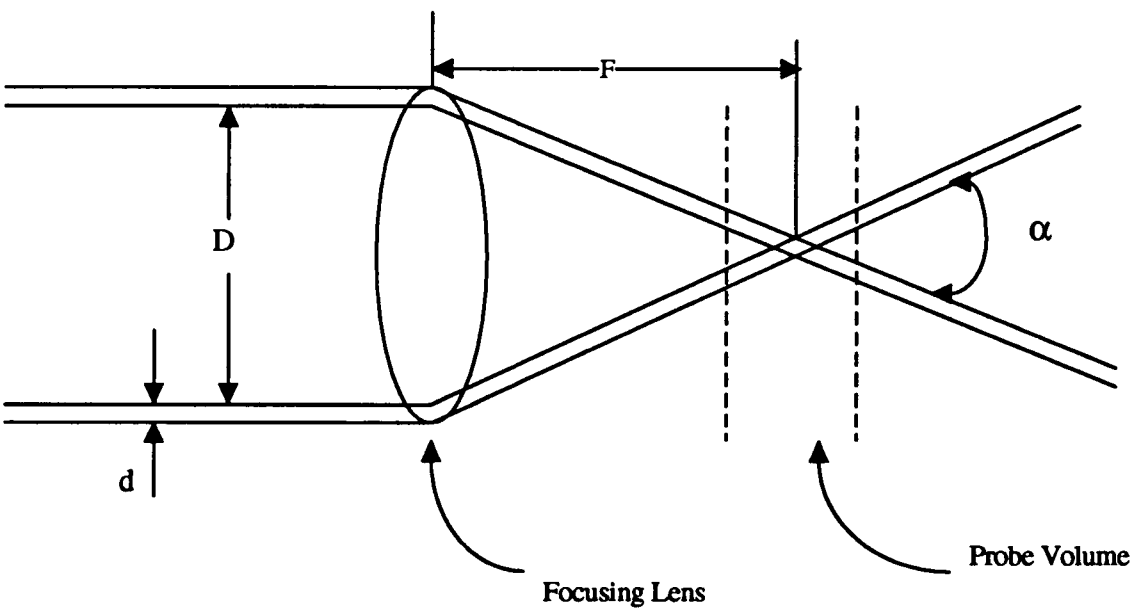
This thesis will discuss the methodology of the data acquisition and data storage and some general theory of the experimental apparatus. The theory of the experiment is discussed in Chapter 2 along with the configuration of the experimental apparatus. Chapter 3 discusses program organization which includes program configuration and a means for mixed language programming using FORTRAN and C. The majority of the programming was written in FORTRAN but C subroutines are called because the DSP cards to acquire the data use C. Chapter 4 will discuss data acquisition and data storage accomplished by the two DSP cards. Chapter 5 will discuss the data reduction method and the limitations of the methods used. Data reduction takes place on the host computer. Chapter 6 verifies that the DSP cards are triggering simultaneously since the time lag calculation between the two input signals is crucial in the particle size measurement. Chapter 7 verifies that the programs are written correctly and tests the limits at which the program will fail. Chapter 8 presents the results of the data acquisition and data reduction and makes some recommendations for improvement of these processes.

Chapter 2

Experiment Theory and Configuration

This chapter presents a general explanation of the experiment and its configuration. The photodetector signals generated by the apparatus are related to the size and velocity of the particles passing through a probe volume [1-3].

In the experimental configuration, an argon laser is split into two coherent beams of equal intensity. The beams are focused to intersect at an angle, α, using a focusing lens. The three dimensional field of this intersection is called the probe volume. The beams incident on the lens must be parallel or the probe volume will not be focused on the point of interest. This configuration is shown in Figure 2.1.



Laser Configuration
Figure 2.1

When two beams of light intersect, the light waves create an interference pattern containing alternating patterns of light called fringes. These fringes are critical for the measurement of the size and velocity of particles traveling through the probe volume. The following equations will determine the velocity of the particles through the probe volume [3, 6]. The diameter of the probe volume is represented by $2b_0$ and is determined by

$$2b_0 = \frac{4\lambda F}{\pi d} \tag{2.1}$$

where

λ = wavelength of the laser beam

F = Focal length from focusing lens to probe volume

d = laser beam diameter

D = distance between the laser beams

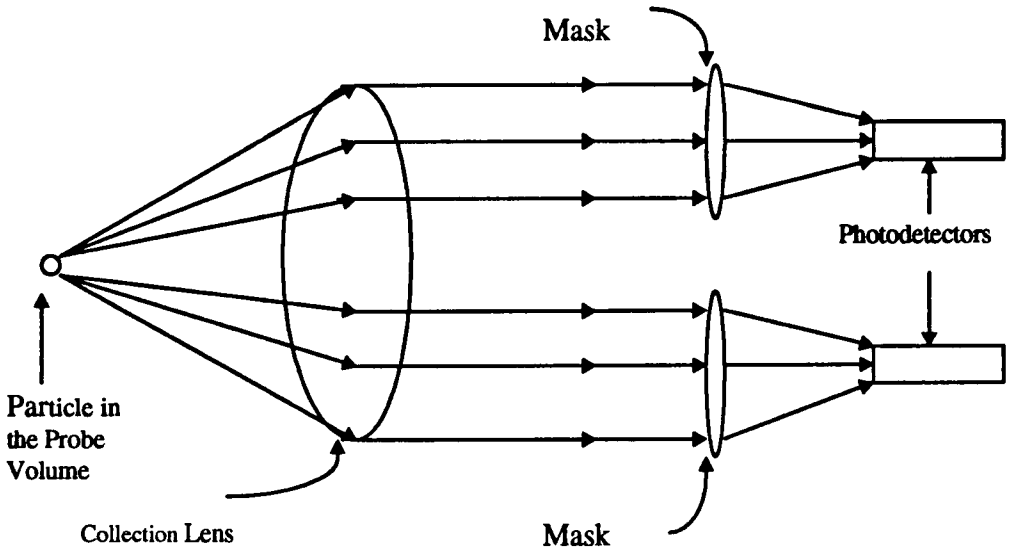
The spatial fringe period or the distance between the fringes is represented by δ where

$$\delta = \frac{\lambda}{2\sin(\alpha/2)} \approx \frac{\lambda F}{D} \tag{2.2}$$

The number of fringes within the probe volume can be estimated from the following equation:

$$N = \frac{2b_0}{\delta} = \frac{4\lambda F}{\pi d \delta} = \frac{4D}{\pi d} \approx \frac{D}{d} \tag{2.3}$$

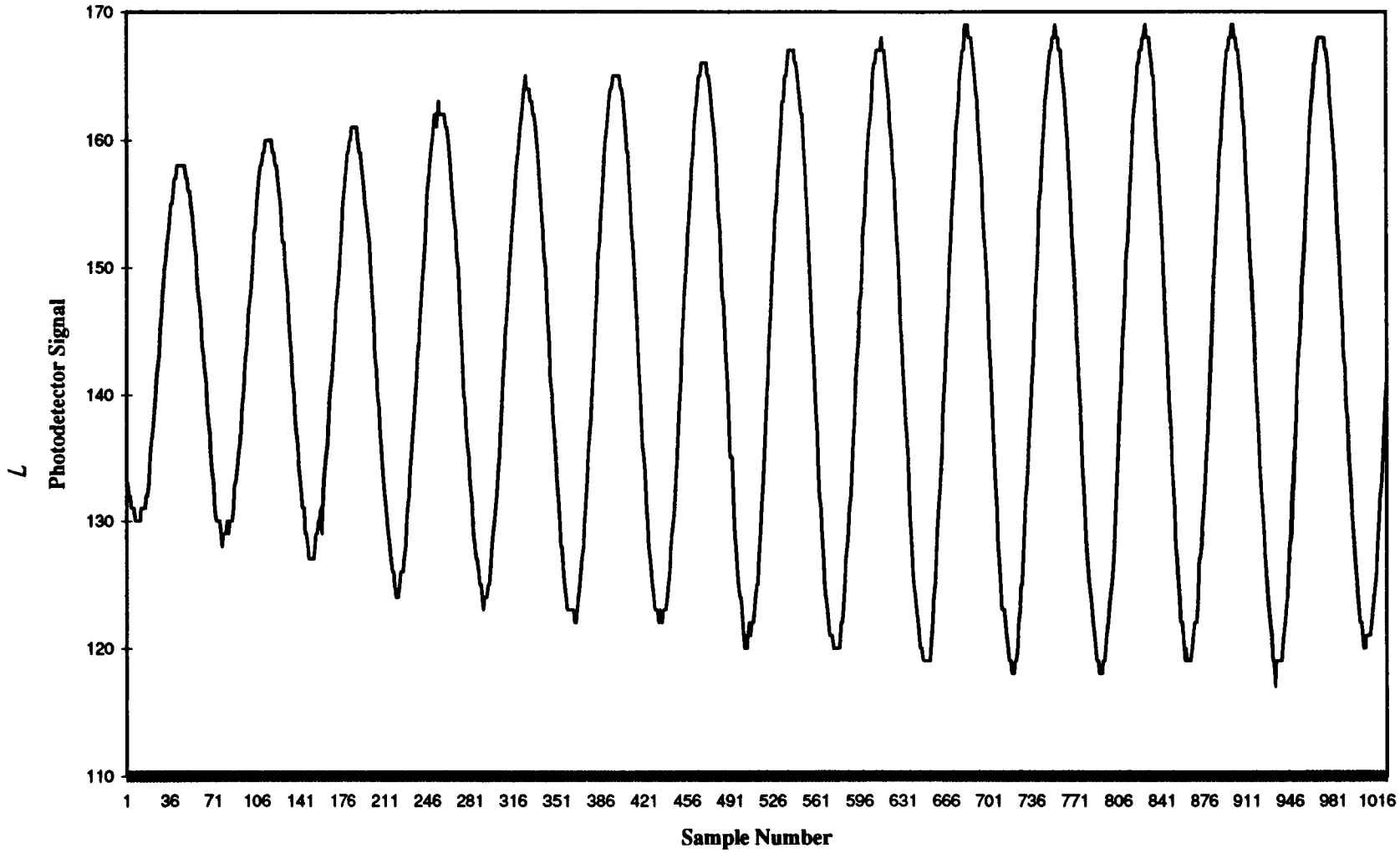
Particles fall into the probe volume scattering light in every direction. A collection lens collimates the scattered light. Two masks are then used to direct the light toward two photodetectors as shown in Figure 2.2.



Receiving Apparatus
Figure 2.2

The signal detected by each photodetector is input to a separate DSP card and to an oscilloscope. The oscilloscope trigger is connected to the trigger input of card one. Card two is slaved to card one so that both cards are triggered simultaneously when a particle falls into the probe volume.

To determine the velocity of the particle as it falls through the probe volume, only one of these photodetectors is needed. The particle is subjected to alternating regions of light and dark as it passes through the fringes. The signal detected by the photodetector is periodic with a period determined by the velocity of the particle through the probe volume. An example of one of these signals is shown in Figure 2.3. One period of the signal, τ, is the time it takes the particle to transverse one fringe. Since fringe spacing is



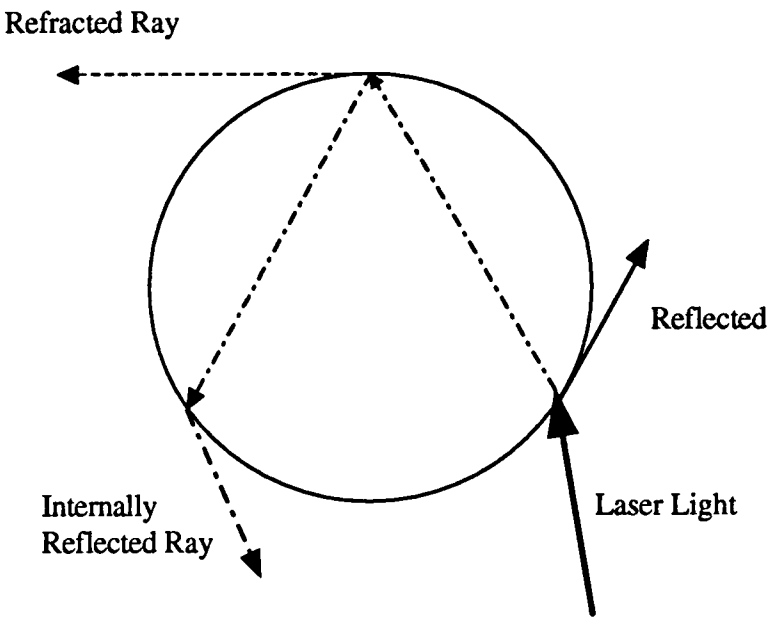
Signal Generated by Particle in the Probe Volume

Figure 2.3

known from equation 2.2, the velocity, v, of the particle through the probe volume can be determined from

$$v = \frac{\delta}{\tau} = \frac{\lambda F}{D\tau} \tag{2.4}$$

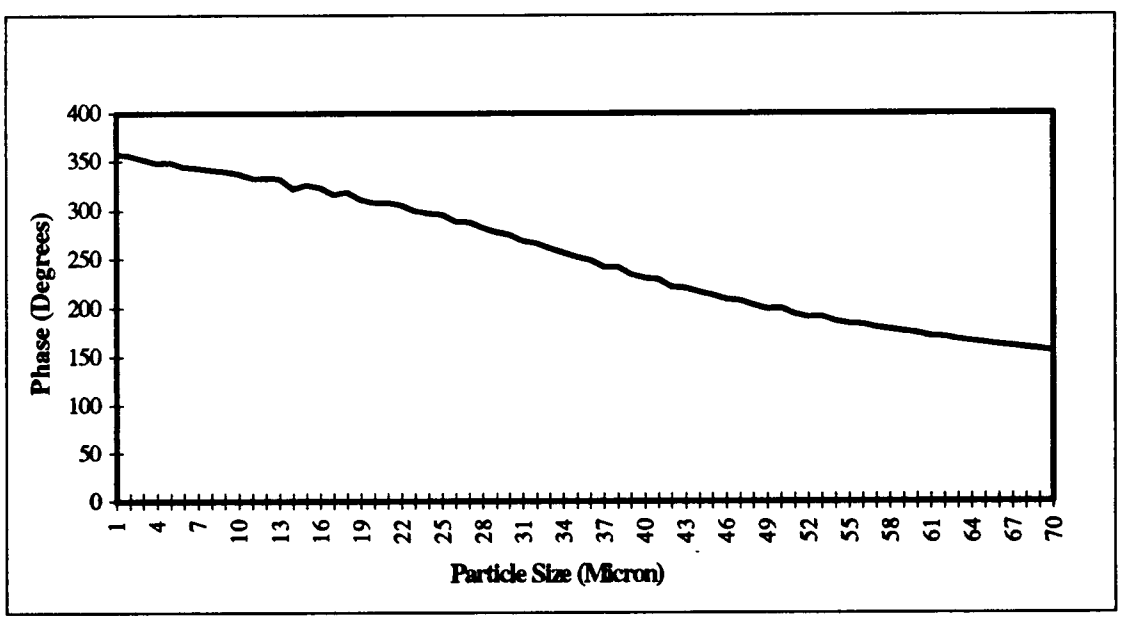
The determination of particle size is more subtle than the determination of velocity. When the particle passes through the probe volume, one of three things will happen to the light as it is scattered. The light will either be reflected, refracted, or reflected internal to the particle. This is shown in Figure 2.4.



Scattering Light from a Particle
Figure 2.4

Both photodetectors are required for the determination of size. Each photodetector reads a part of the light that is reflected or refracted but not the same part. Also, the path the light takes to reach each detector will not be the same. The optical path

to each detector will depend on the distance of the detector from the particle and the size of the particle. The time skew in the plane of the collection lens is the result of the different optical paths of the light through the particle. The signals observed by the detectors will have the same period but will be out of phase due to this time skew and a large particle will have a greater phase shift than a small particle. Durst and Naqwi developed this theory using a configuration similar to that of the CLA [1, 2]. The graph in Figure 2.5 is used to find the size of water droplets from the phase shift measured using the crosscorrelation function. This relationship depends on the placement of the photodetectors with respect to each other and the probe volume, the wavelength of the laser beam, λ, and the index of refraction of the particle which for the graph below is water.



Phase vs. Particle Size for Water Droplets Calculated by Durst and Naqwi [1, 2]
Figure 2.5

Chapter 3

Program Organization

Two digital signal processing(DSP) cards, both model DASP100 by Signatec, were used for the data acquisition and storage of the two signals. The software required to address the cards is written in C and is called Cbuild [5]. The data acquisition and data reduction programs are called REDUCE3.FOR and RED2INTT.FOR and are listed in Appendix A. These programs are written in FORTRAN and call C programs that are listed in Appendix B as subroutines needed to control data acquisition. Data is transferred from the DSP cards and reduced using a 486DX 33Mhz personal computer.

Program Configuration

The programs REDUCE3.FOR and RED2INTT.FOR consist of three major sections as follows:

1. DSP Board Configuration
2. Data Acquisition
3. Data Reduction

Each time one of the programs is run, data from several particles will be acquired and reduced. The phase and period from each particle will put be in separate data files. The data file names are specified by the user.

The DSP cards must be configured before data acquisition can begin. A list of the configuration parameters and their options are listed in Table 3.1 [4]. The amplifier type used is linear within the voltage range of the typical input signals. Burst mode data acquisition is used to ensure that the signal is only digitized when a particle is present. The cards are set to trigger when an external start signal having a positive slope occurs from the oscilloscope. The oscilloscope is triggered when a particle falls into the probe volume. The active memory option specifies how many data points will be acquired. The clock divider specifies the clock frequency. This is set to fit the estimated velocity of the particle through the probe volume. These options can be changed as required by the user with selections on the main program menu prior to data acquisition.

The data acquisition is identical for both programs. The differences in the programs are in the data reduction and are discussed in the Chapter 5.

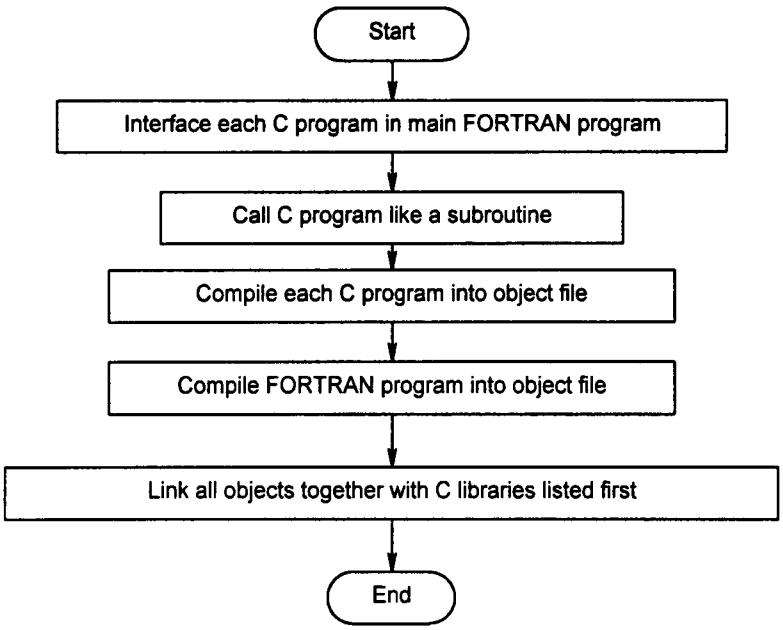
Mixed Language Programming

Cbuild is a library of functions written in C for data acquisition and data reduction required for the DASP100 DSP cards. The majority of the programming for this thesis is written in FORTRAN but the data acquisition portion was written in C with the FORTRAN calling C subroutines. The C subroutines are written using functions resident in the Cbuild library [5]. Examples of C calls from FORTRAN are in the programs listed in Appendix A. The C programs are listed in Appendix B.

A flowchart showing the steps required for this mixed language programming is shown in Figure 3.1. Each C subroutine must have an interface statement that precedes

Table 3.1 - Configuration Parameters

Value	Parameter	Value	Parameter
Amplifier Type		**Acquisition Mode**	
0	Linear	0	Burst
1	Compression	1	Pretrigger
		2	Continuous
		3	Segmented
Voltage Range		**ADC Offset**	
0	60 mV	0 through 255	
1	200 mV		
2	600 mV		
3	2 V		
Threshold		**Timeout Cycles**	
0 through 255		0 through 31	
Trigger Source		**Trigger Coupling**	
0	External	0	ADC
1	Internal	1	DC
Active Memory		**Clock Divider**	
0	16 kbytes	0	100 MHz
1	32 kbytes	1	50 MHz
2	64 kbytes	2	25 MHz
3	128 kbytes	3	12.5 MHz
4	256 kbytes	4	6.25 MHz
5	512 kbytes	5	3.125 MHz
6	1 Mbytes	6	1.56 MHz
7	2 Mbytes	7	781 kHz
Trigger Slope		**Number of Data Points**	
0	Negative	Any number	Number of Data Samples pulled out of RAM at a time
1	Positive		



Flowchart for Mixed Language Programming
Figure 3.1

the main program and is called in the main program in a similar manner as FORTRAN subroutines.

After the programming is complete, the main program and each C program must be complied into separate object files. The FORTRAN compiler and the C compiler must be compatible so the programs can be linked together. Four C compilers were tried before finding one that would work with the version of FORTRAN that was being used. The compilers found to work together were Microsoft FORTRAN 5.1 and Microsoft C 6.0.

After all the programs are compiled, their objects are linked together using the Cbuild , C, and FORTRAN libraries. The C libraries have to be listed before the FORTRAN library on the command line when linking.

Chapter 4

Data Acquisition

This chapter will explain the data acquisition portion of the programs. The data acquisition is identical for both programs.

Theory

The data acquisition mode parameter is set to burst which means data is taken when the trigger for the DSP cards is satisfied and is terminated when the trigger is not satisfied for more than the number of timeout cycles specified. The trigger is satisfied by the presence of a particle in the probe volume. Whenever data acquisition is suspended, between one and four markers with values of 255 are placed in RAM.

Each time a particle falls into the probe volume, the cards are triggered commencing data acquisition. Data acquisition continues until the particle leaves the probe volume. The program will continue the process of waiting for triggers, taking data, and inserting 255 count markers until the amount of RAM is filled which is specified by G, the active memory configuration parameter. After data acquisition is complete, the RAM contains these data sets separated by 255 count markers. These markers are crucial in determining the location of the data for each particle.

To determine the location of the data sets in RAM, the address for each marker must be known. The RAM on each DSP card has sixty-four 32kbyte segments. The active memory configuration parameter, G, determines how much of the RAM will be

used to store acquired data. G can take on values of 0-7 where the amount of memory equals $16{,}000*2^{G}$ for a maximum of 2Mbytes with segments numbered 0-63. The number of the last segment of data, maxseg, equals $2^{(G-1)}-1$. For example if G=7, maxseg equals 63. Each segment has 32768 addresses ranging from 0-32767. The maximum segment address is referred to as iseg and is equal to 32767 unless G=0 where it is 16383. In the program, segment is 'IS' and address is 'IA'. For example, if the marker was in the second address of the forty-first segment, the location of the marker IA=1 and IS=40.

Implementation

Before data acquisition can begin, the program is configured by the user as described in Chapter 3. Then, the arrays are initialized prior to data acquisition. SETUP.C is a C program listed in Appendix B that is called to initialize the DSP cards by downloading the Signatec initialization program and the configuration parameters onto the DSP cards. After the cards have been initialized, data is digitized into the RAM of the DSP cards in the manner discussed earlier in the chapter.

After the data have been acquired, the program calls another C program named GETDAT.C listed in Appendix B. This program retrieves the data from the memory of the DSP cards at the address of the segment of RAM specified. The number of samples retrieved is specified by the number of data points configuration parameter, in this case 1024. At this point, GETDAT.C is retrieving the first set of data.

After GETDAT.C retrieves the data from RAM, the data are put in two arrays called data_hold1 and data_hold2. These two arrays are formatted unsigned char in C

meaning unsigned counts 0 to 255. FORTRAN has no equivalent to this so the data is formatted integer*1. The data acquired is in the form of two's complement in FORTRAN and unsigned binary in C. The data that ranges from 0 to 127 is equivalent in C and FORTRAN but 256 is effectively subtracted from each value greater than 127 due to the two's complement, so 256 must be added to any negative value in the FORTRAN program.

After the data is retrieved, the program makes sure there are no 255 count spikes present in the data set. The spikes can occur for several reasons. If the particle is traveling at a high velocity, it might not be in the probe volume long enough to satisfy the trigger for the length of time to digitize 1024 samples. The cards have also been known to be erratic and insert spikes in the center of the data for reasons unknown. Also, the cards insert one to four spikes between data sets so the program might pull a data set with a spike at the beginning. These spikes are important for locating the particles in RAM but if they occur in data, the program outputs will not be correct. The program searches for any 255 counts spikes in the data set. If any are found, the data reduction portion of the program is skipped. If the data set contains no spikes, data reduction will take place. This will be discussed in the next chapter.

After data reduction is complete, the program calls the C program SEARH.C listed in Appendix B. This program locates the next set of 255 count levels in the current segment of data and returns this address. Another C program, SETSEG.C is called to change the address to the one found in SEARCH.C. GETDAT.C is called to retrieve the data set at this address. The FORTRAN program checks to make sure the address is

greater than iseg. If it is, the segment is incremented and the address is set to one unless it is equal to maxseg in which case data reduction is complete for all particles. If it is not the last particle, the program returns to reduce the data of the next particle. When all the data has been reduced, the phase and period of each data set is written to their respective files. The mean of all the measured phases is also written to the screen.

Chapter 5

Data Reduction

Phase and Period Determination

The data signals input to the DSP cards are labeled x and y in the programs. The crosscorrelation is a function which determines the relationship between two signals at different periods of time. The crosscorrelation of x(n) and y(n) is defined as the following:

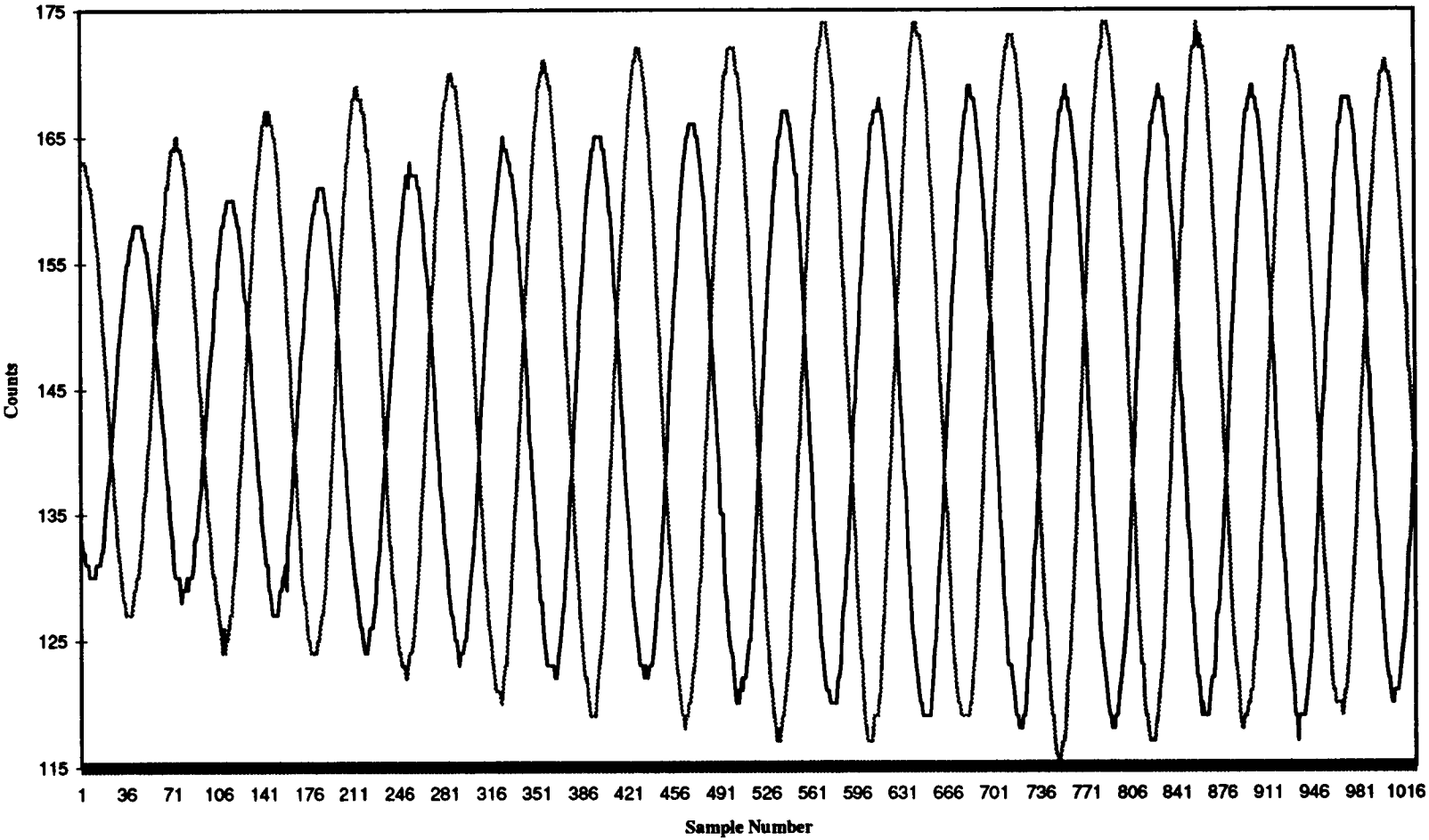
$$r_{xy}(k) = \sum_{n=0}^{N-1} x(n)y(n-k), \quad k = \pm 0, \pm 1, \pm 2, \ldots \qquad (5.1)$$

or

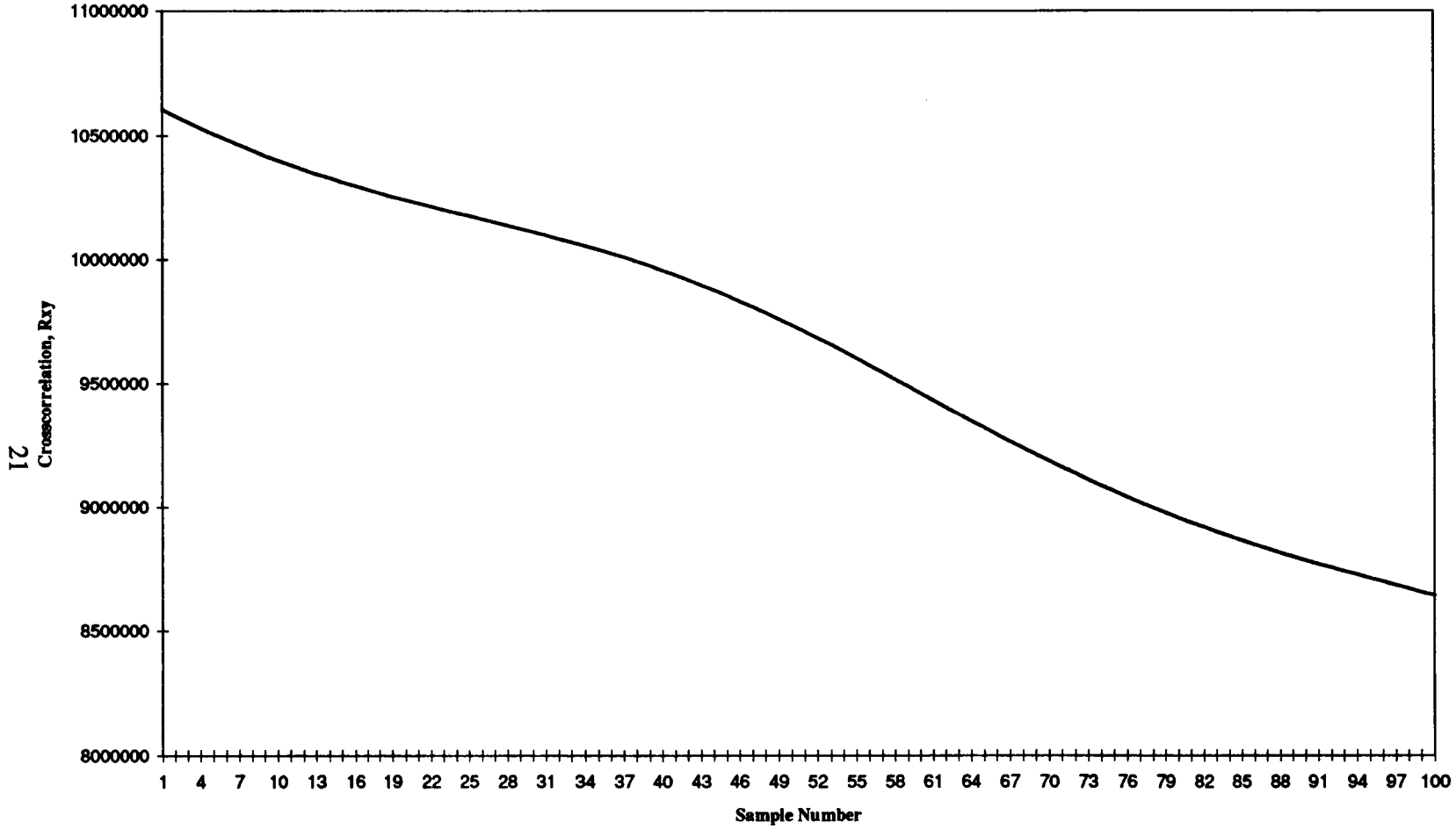
$$r_{yx}(k) = \sum_{n=0}^{N-1} y(n)x(n-k), \quad k = \pm 0, \pm 1, \pm 2, \ldots \qquad (5.2)$$

For the purposes of this work, it makes no difference whether r_{xy} or r_{yx} is calculated. If the signals have the same period, the crosscorrelation will have the same period as the signals [7]. The first peak of the crosscorrelation will occur at the sample number of the lag between the signals [12].

Figure 5.1 is a graph of two typical signals x and y. These signals are essentially the same except out of phase. Figure 5.2 is the crosscorrelation of these two signals. The peaks are hard to discern. This is not useful since the peaks are needed in the calculation



Signals X and Y
Figure 5.1



Crosscorrelation of signals x and y
Figure 5.2

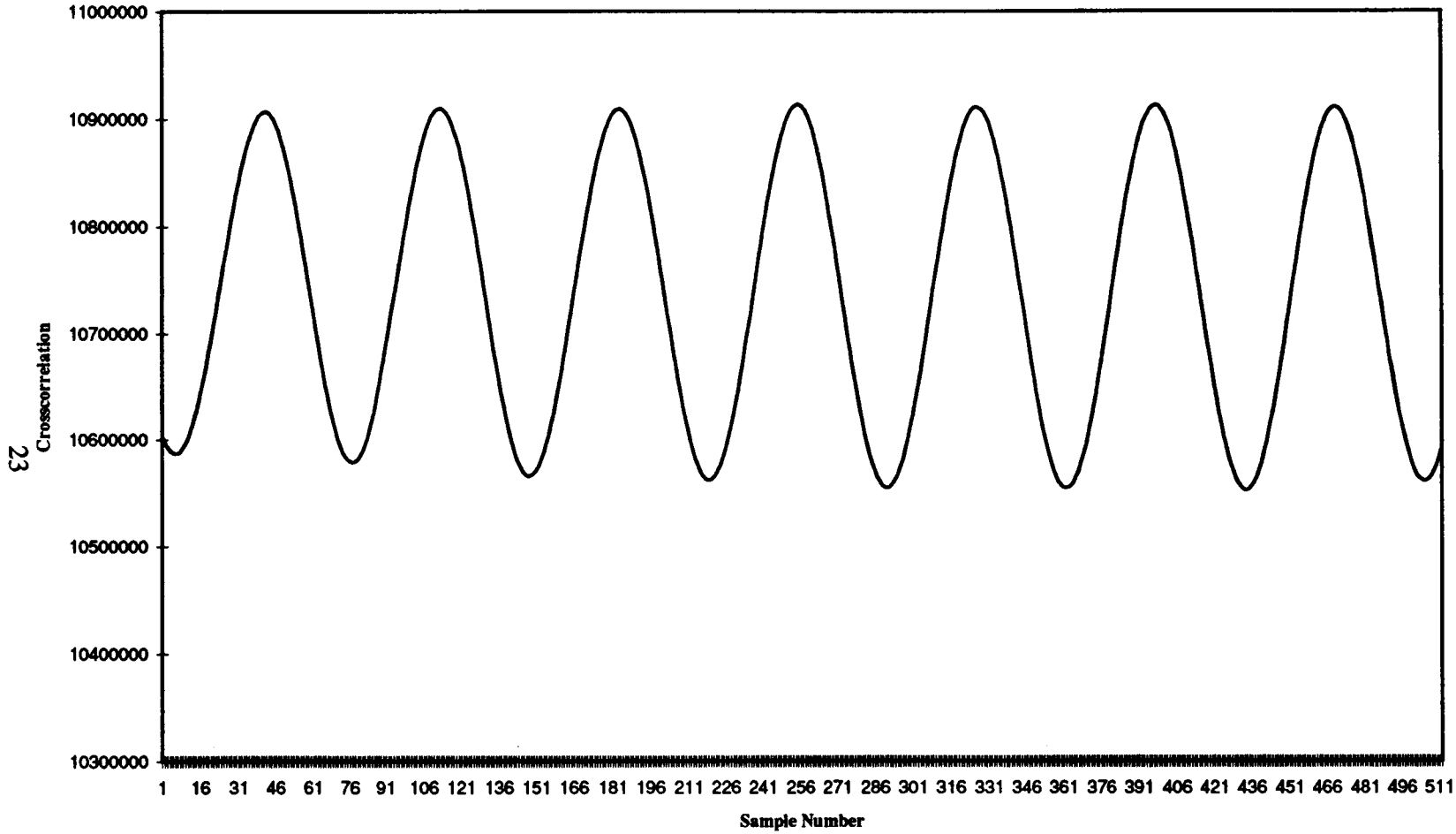
of the phase and the period of the signals. Since the signals contain 1024 points, the first point on the crosscorrelation curve is effectively the sum of 1024 products (even if it is actually computed via the FFT). For the second crosscorrelation point the signals are shifted so only 1023 products are summed. The last crosscorrelation sample will have only one product between x and y and 1023 products between y and zero because x does not have samples beyond 1024. This is referred to as an end effect which will skew the data [11]. It could move the peak a sample or two which would give a false answer for the phase shift. In Figure 5.2, each of the peaks should be the near the same magnitude since the peaks of the original signals are approximately the same magnitude. However, they are not because of the end effect.

To eliminate this problem, each crosscorrelation calculation should have the same number of products for the summation of x and y. One solution to this problem is to make the nonzero values of one of the sequences at least twice as long as the other [11]. The crosscorrelation computed with half of y zeroed is shown in Figure 5.3. The peaks are easily discernible.

It is apparent from equations 5.1 and 5.2 that if there is a large number of data samples, the crosscorrelation function requires many multiplies. If the crosscorrelation is computed in the frequency domain using a radix-2 1024 point fast Fourier transform(FFT), there will be fewer multiplies [7].

If $$x(n) \overset{FFT}{\leftrightarrow} X(\omega), \tag{5.3}$$

then $$y(n) \overset{FFT}{\leftrightarrow} Y(\omega), \tag{5.4}$$



Crosscorrelation Sequence Calculated in the Frequency Domain
Figure 5.3

$$r_{xy}(k) = x(n) * y(-n), \tag{5.5}$$

and

$$FFT\{r_{xy}(k)\} = X(\omega)\, Y(-\omega). \tag{5.6}$$

If x(n) and y(n) are real, $Y(-\omega) = Y^*(\omega)$ so

$$r_{xy}(k) = IFFT\{X(\omega)\, Y^*(\omega)\} \tag{5.7}$$

The crosscorrelation is calculated in both programs using equation 5.7.

After the crosscorrelation is computed, the first and second peaks are located. The number of samples in the period is taken as the difference between the second and first peak of the crosscorrelation. The sample rate on the DSP card is

$$\text{Clock Frequency} = \frac{10E8}{2**F} \tag{5.8}$$

where F = Clock Divider Configuration Parameter [4,5].

The period, τ, is calculated in microseconds as

$$\tau = \frac{(\text{Second Peak Sample\#} - \text{First Peak Sample\#}) \cdot 10E6}{\text{Clock Frequency}} \tag{5.9}$$

The phase, ϕ, is calculated in degrees using

$$\phi = 360 \cdot \frac{\text{First Peak Sample\#}}{\text{Second Peak Sample\#} - \text{First Peak Sample\#}} \tag{5.10}$$

Data Interpolation

If the particles are traveling at a high velocity, there may not be enough samples per signal period for accurate phase and period measurement. Interpolation can be performed on the raw data before the crosscorrelation is performed to effectively increase

the sample rate. The program RED2INTT.FOR interpolates the data prior to performing the crosscorrelation.

Interpolation is accomplished by first inserting zeros between the data samples [7]. If the data rate is doubled, one zero is inserted between every two samples. If the data rate is to be tripled, two zeros are inserted, and so forth. After this is completed, the FFT is taken. The FFT will be the FFT of the original sequence and images of the original FFT. Each FFT is lowpass filtered to remove the images. The inverse FFT is then performed to obtain the interpolated sequence. The first few samples must be thrown away depending on the length of the lowpass filter. Since a 65-point lowpass filter is used, the first 64 points contain distorted information due to circular convolution. Since the same linear phase finite-impulse-response (FIR) filter is applied to both data sets, the relative phase between the two is unaltered. The program RED2INTT.FOR can interpolate the data two, four, or eight times the original data rate. The user decides what rate of interpolation is required.

Error Minimization

The first way to minimize error was discussed earlier in the chapter. Zeroing at least half of one signal will eliminate the end effect.

An error will also be introduced in the phase shift measurement if an integral number of periods is not used to calculate the crosscorrelation. The peak is shifted slightly depending on whether larger or smaller values are used from the fraction of a period that is involved. The more periods used to calculate the crosscorrelation the smaller this error

will become. It was found that at least five periods were needed to reduce this error below ± 2 samples.

This error can be eliminated if an integral number of periods is used in the calculation of the crosscorrelation. Therefore, crosscorrelation is performed a first time only to calculate the period of the signals. It is then calculated again with the length of y having two constraints. It must be less than half of the length of x to eliminate the end effect and have an integral number of periods to eliminate the shifting of the peak.

The crosscorrelation is performed only once in RED2INTT.FOR with half of y zeroed. It is not performed twice because too much time is required with interpolation. The interpolation of the data takes almost two seconds per particle and the crosscorrelation takes approximately one second. If the crosscorrelation is performed twice, the program would take almost four seconds per particle on the 486DX personal computer being used. The CLA deemed this too lengthy.

Another problem arises from the quantization of the continuous signal. The true peak of the crosscorrelation could fall in between two samples. This can introduce an error of up to 1 sample.

The number of samples in the period is the difference between the second and first peak of the crosscorrelation. Referring to equation 5.10 if the period is a small number and the first peak is off by one sample, the error for the phase could be large. If the period is 40 samples and the peak has a one sample error, the error in the crosscorrelation will be less than 2.5%. Therefore, the program requires a period of greater than 40 samples to make sure this error remains less than 2.5%.

Periods greater than 200 are discarded for having too few signal periods and less than 40 for having too few samples per period. If the majority of the particles fall outside these limits, the clock frequency can be changed to take more or less data samples per particle or RED2INTT.FOR can be used to interpolate the data.

Error is also introduced if the data is noisy. The crosscorrelation is usually very smooth, but if the data is extremely noisy the crosscorrelation reflects this error. The phase shift and period is determined by locating the peaks of the crosscorrelation. If there is noise in the crosscorrelation, the programs can mistake the noise for peaks. The phase and period determined would be incorrectly calculated.

There are two ways to reduce the severity of this noise problem. The first way is to discard any data from a particle where the period is less than the phase or the phase is greater than 360 degrees. The second way to avoid this problem is to avoid the false peaks. The programs look at where the data starts increasing or decreasing to locate the peaks in the crosscorrelation. The false peaks usually occur near the true peaks. Therefore, after the program has located the first peak, the program skips the next five samples before starting to find the next peak. This method should ignore most of the false peaks. Both these methods are used in the programs. Flow diagrams showing the flow of REDUCE3.FOR and RED2INTT.FOR are shown in Figures 5.4 and 5.5 respectively.

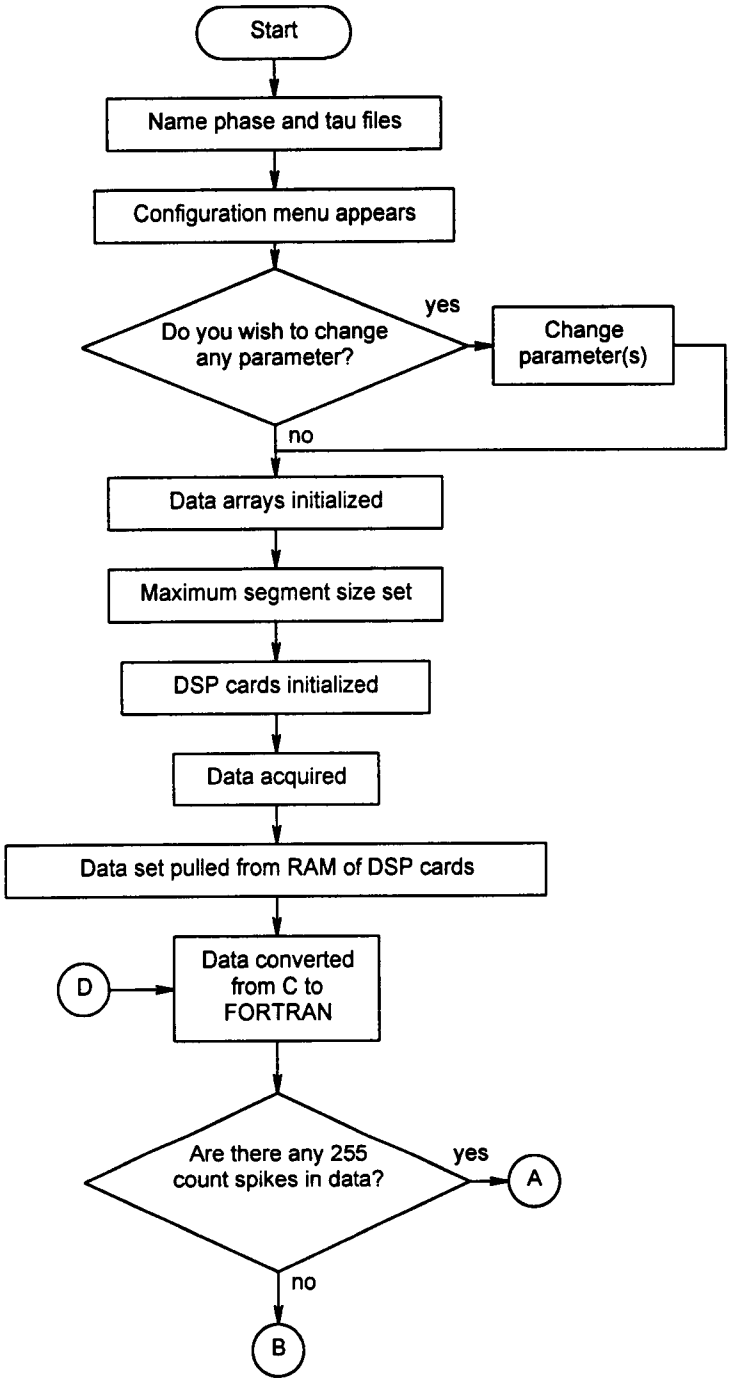


Figure 5.4- Flowchart for REDUCE3.FOR

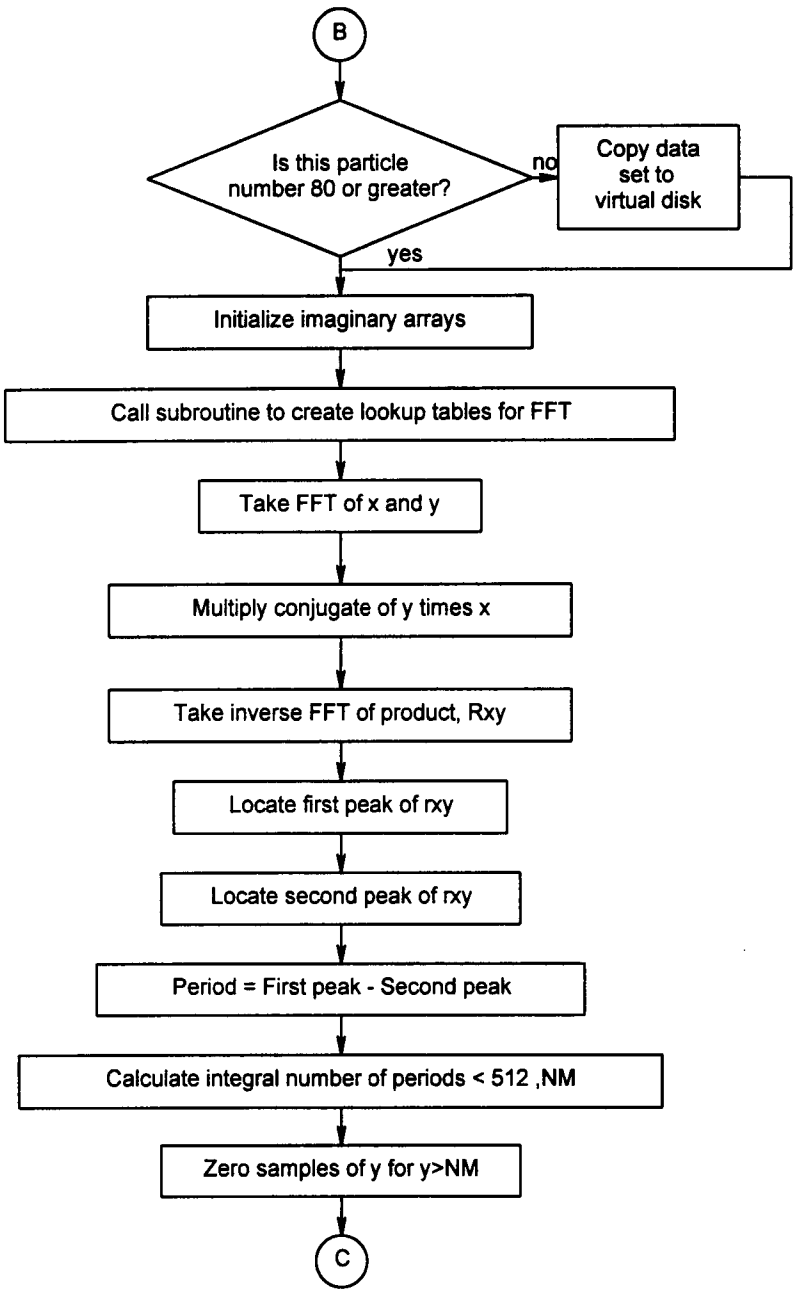


Figure 5.4 (continued) - Flowchart for REDUCE3.FOR

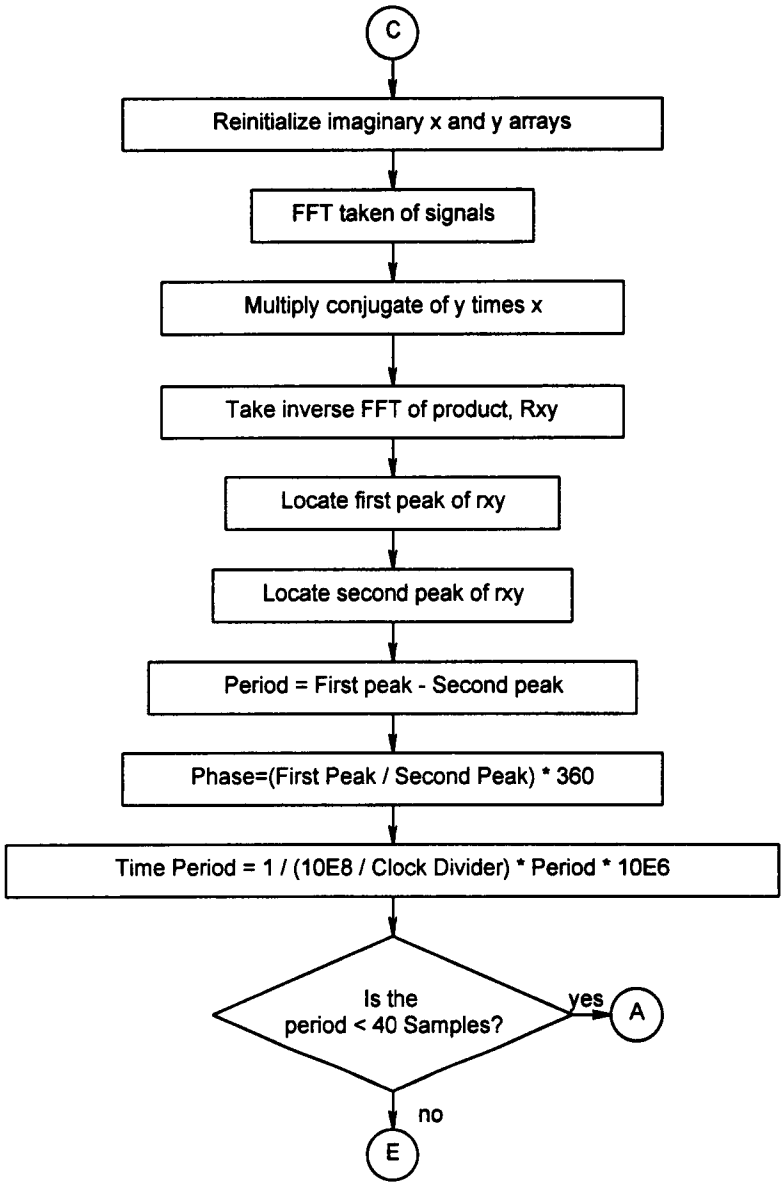


Figure 5.4 (continued) - Flowchart for REDUCE3.FOR

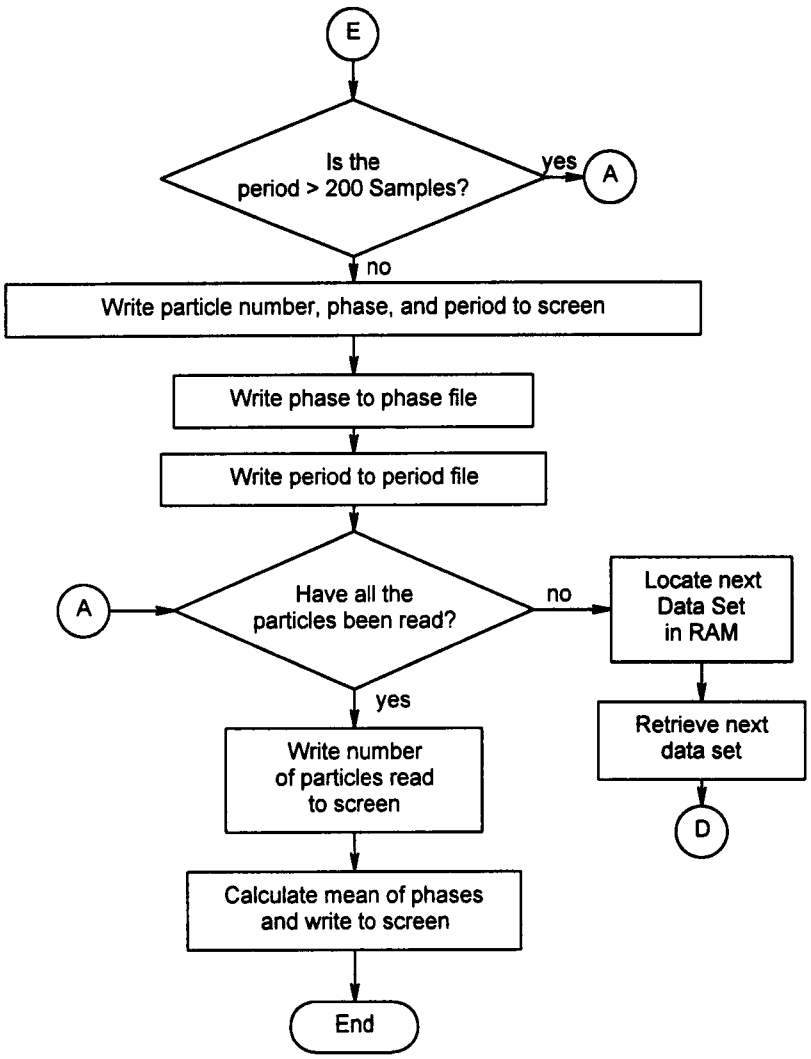


Figure 5.4 (continued) - Flowchart for REDUCE3.FOR

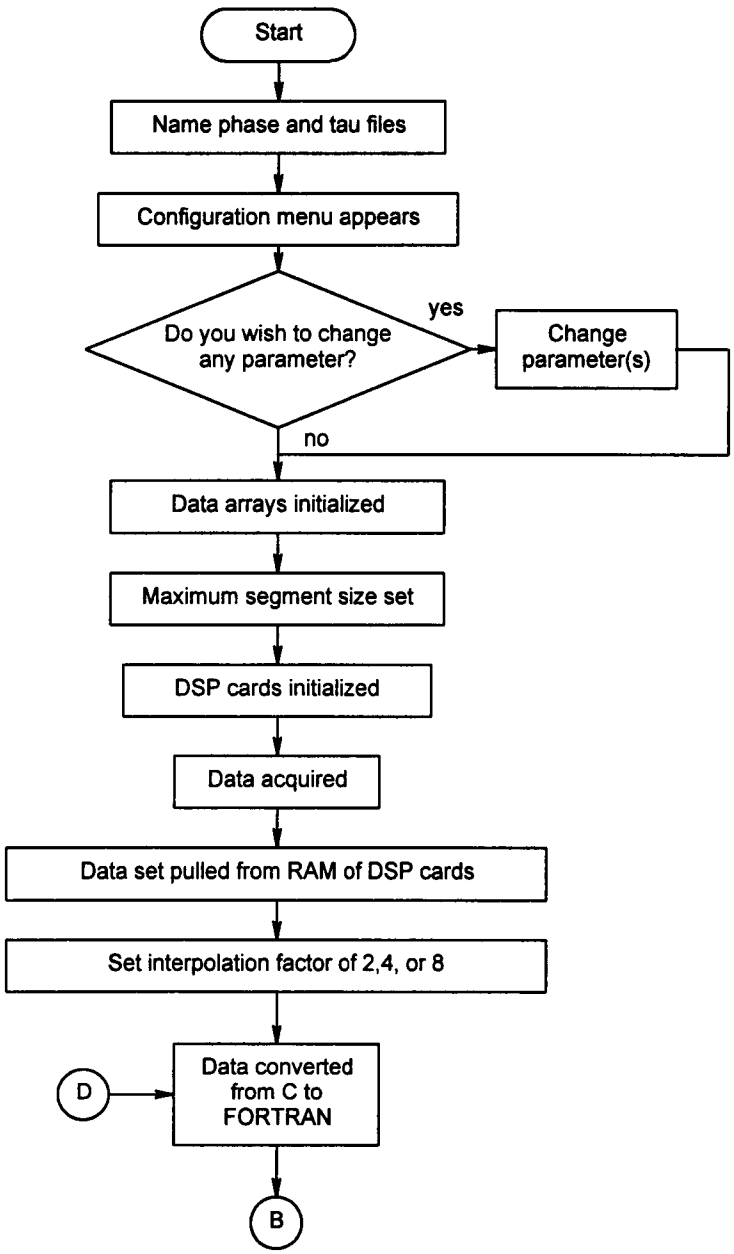


Figure 5.5 - Flowchart for RED2INTT.FOR

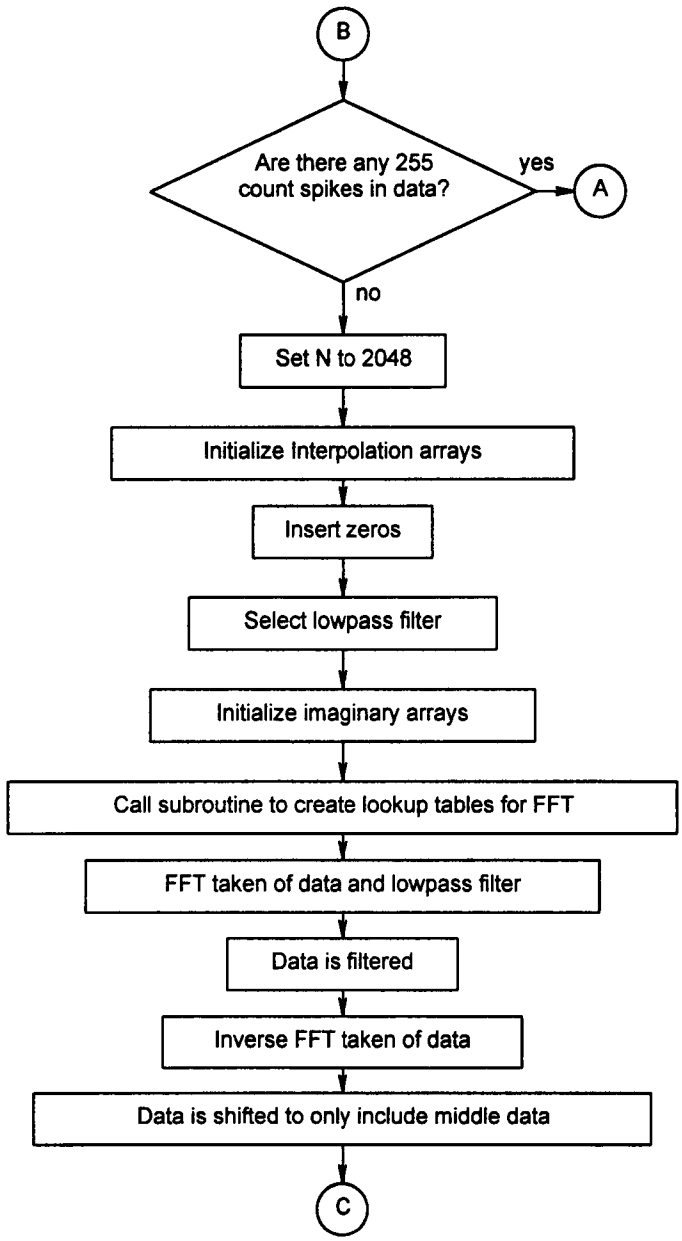


Figure 5.5 (continued) - Flowchart for RED2INTT.FOR

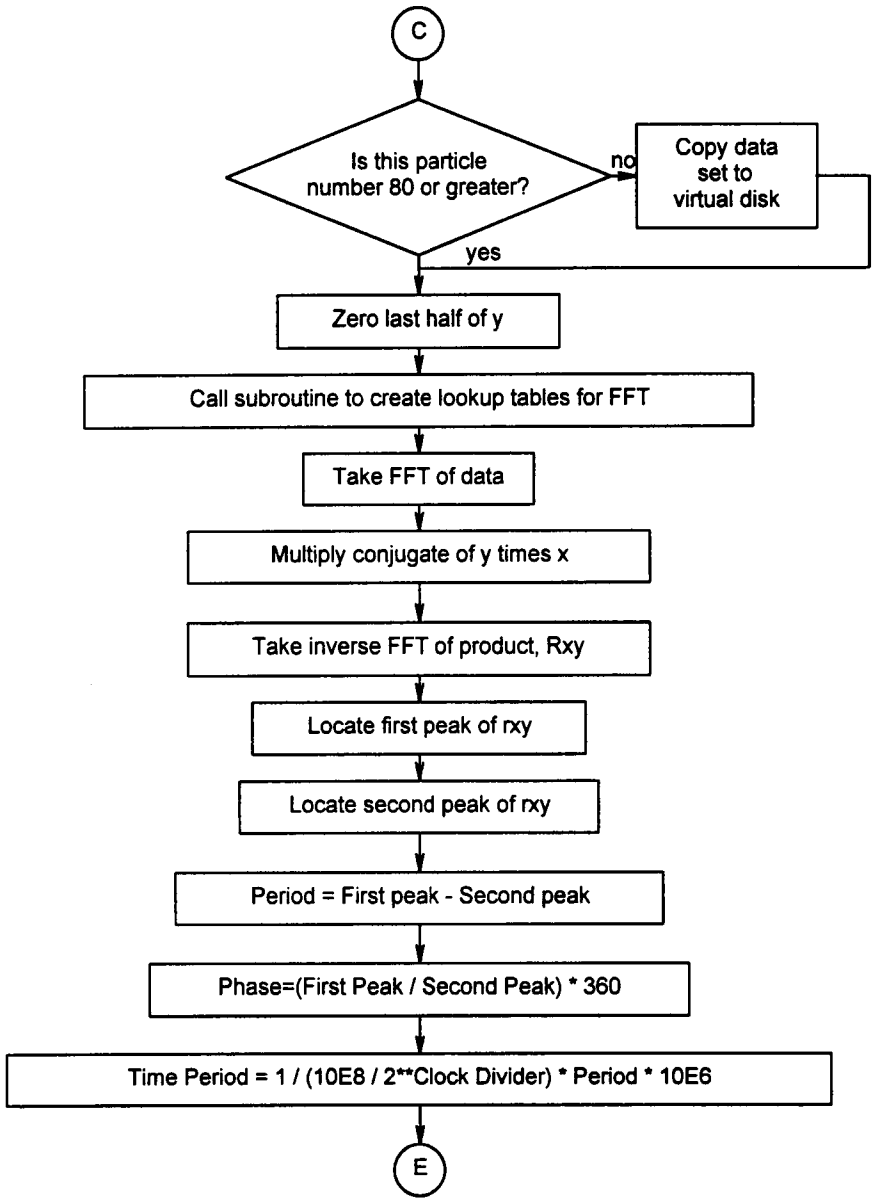


Figure 5.5 (continued) - Flowchart for RED2INTT.FOR

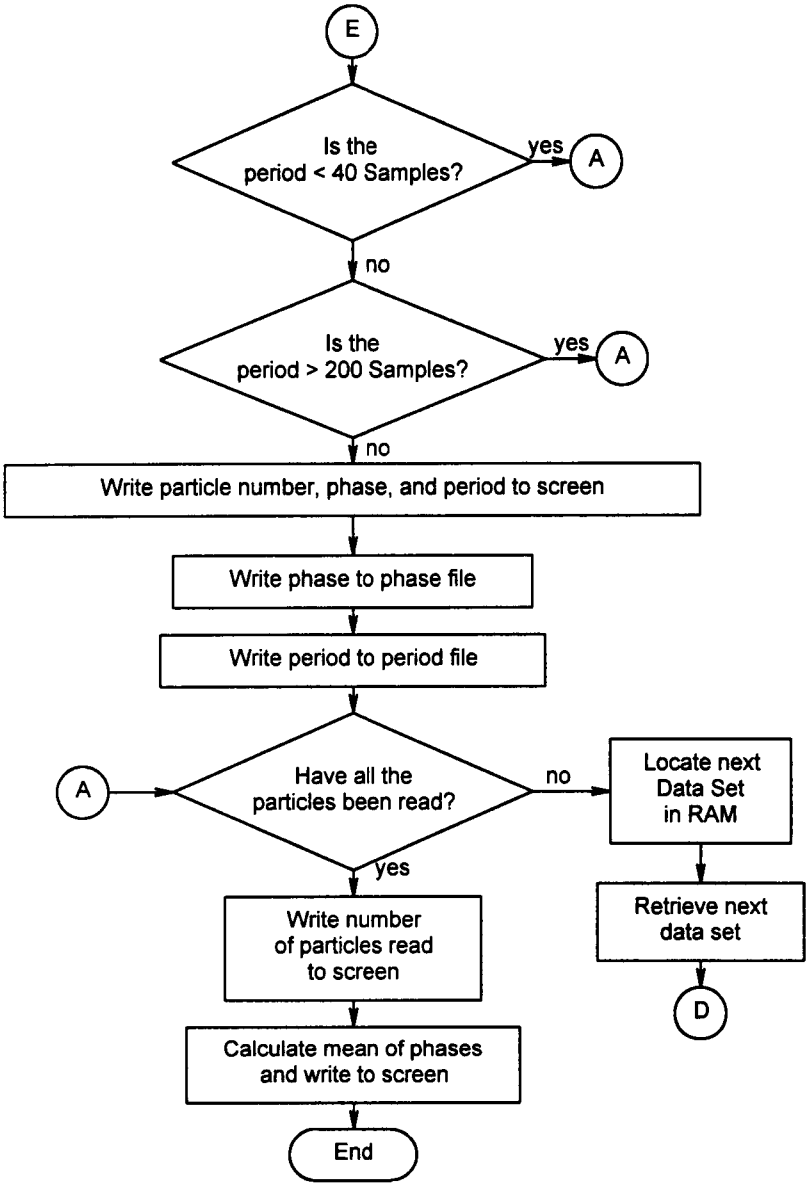


Figure 5.5 (continued) - Flowchart for RED2INTT.FOR

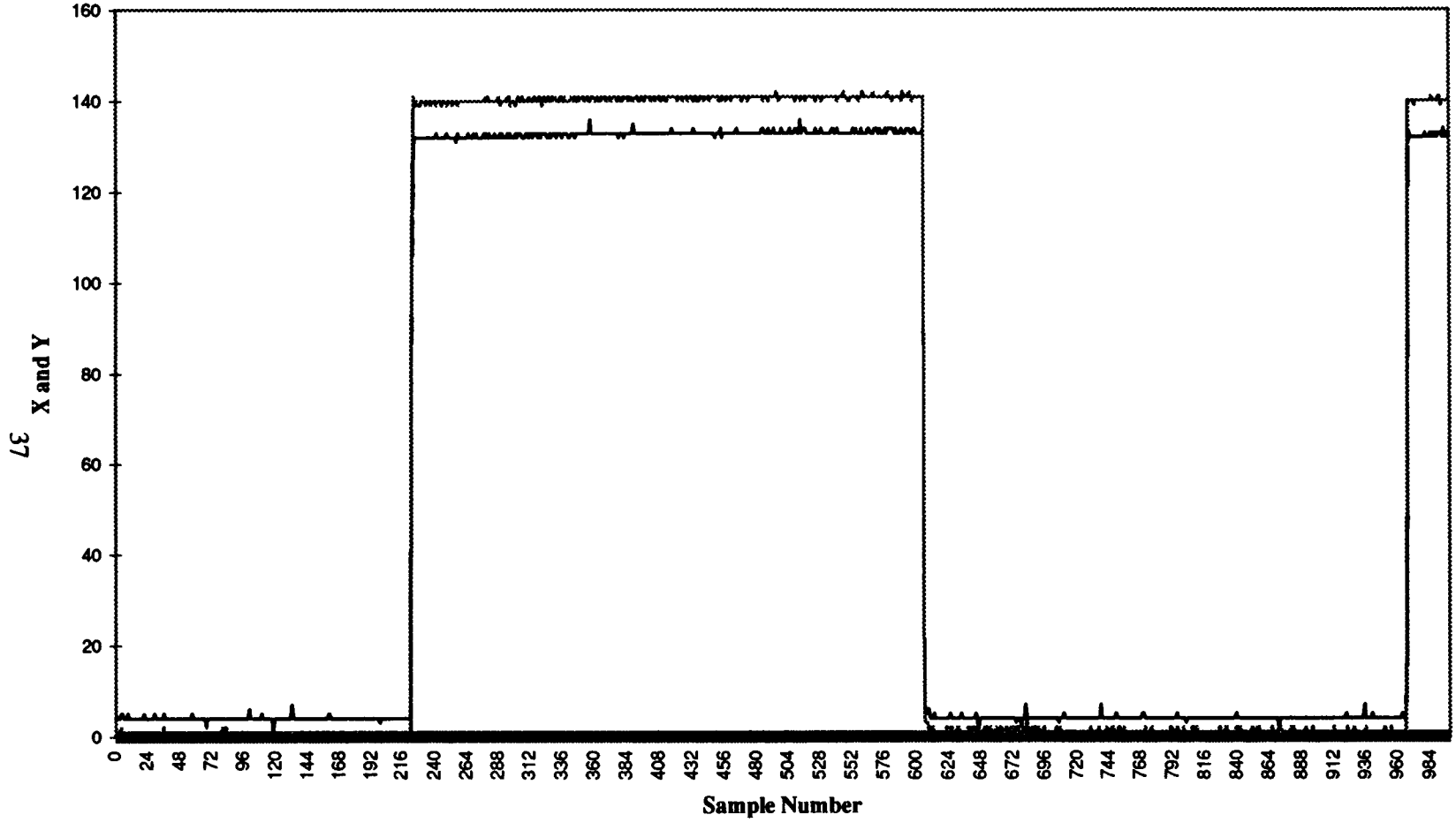
Chapter 6

Data System Checkout

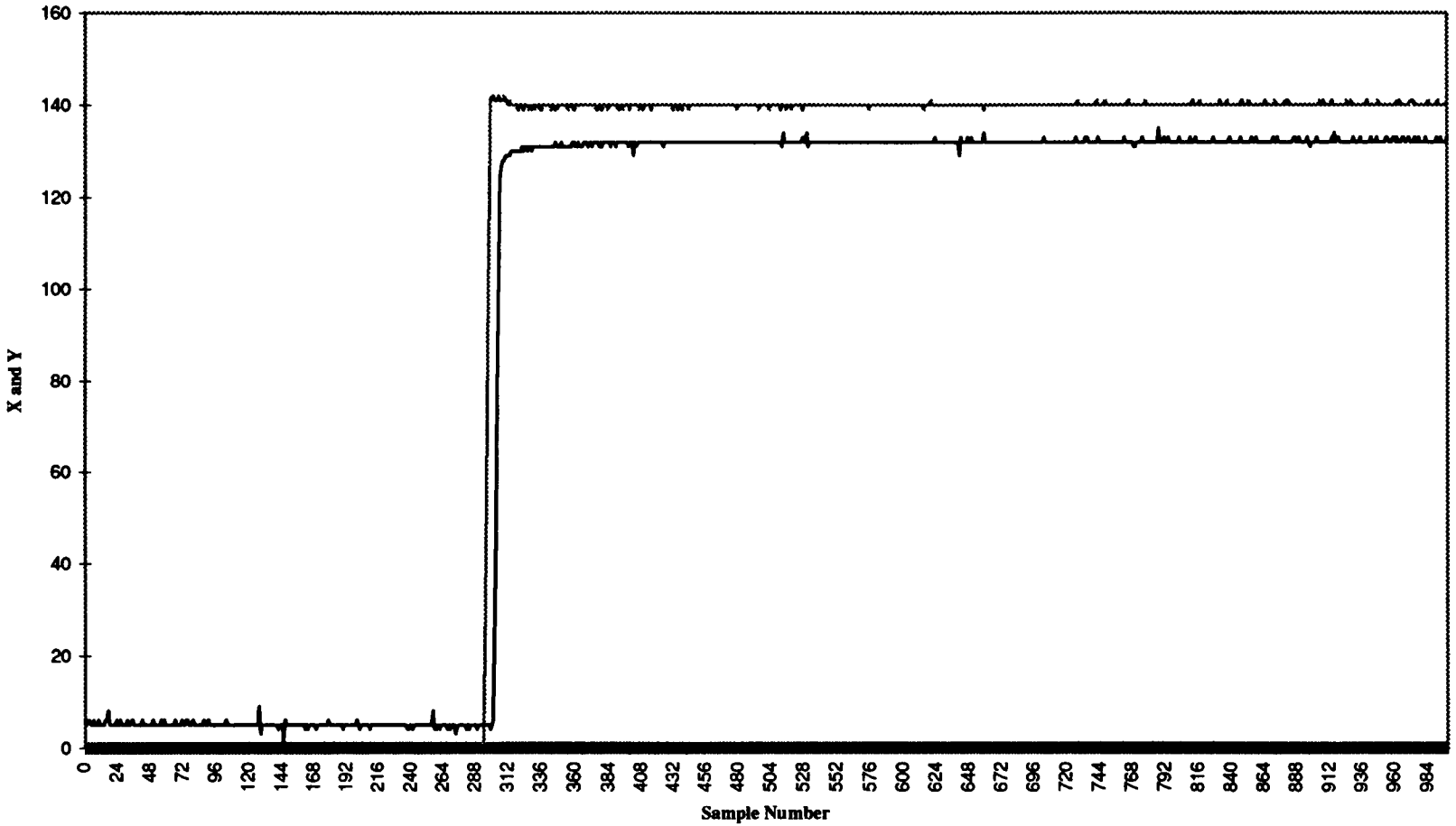
A major objective of the computer program is to calculate the phase shift between two signals. To ensure a valid measurement, the DSP cards are slaved together and must trigger at the same time. Verification of the simultaneous triggering of the two DSP cards is discussed in this chapter.

A frequency generator was used to check the data system. A square wave was input to both cards using cables of the same length. Data was taken and is shown in Figure 6.1. Both signals appear to be triggering at the same time since there is no delay measured.

Data was then acquired after one cable was lengthened by 85 feet. This is shown in Figure 6.2. These signals show a time delay. The type of coaxial cable used is Belden RG058. This is specified to MIL-C-17 as having a velocity of propagation of 66% of the speed of light. The propagation delay is computed as 130.94 nanoseconds. Referring to figure 6.2, the delay between the signals is seven data points at a clock rate of 50Mhz. The measured time delay between the signals was thus 140 nanoseconds. Therefore, the error was approximately 9 nanoseconds. Since error was less than one clock period, it appeared that the DSP cards were properly triggering.



Frequency Generator Output with Same Length Cable
Clock Rate 12.5 MHz
Figure 6.1



Frequency Generator Output with Cable Length Difference of 85 Feet
Clock Rate 50MHz
Figure 6.2

Chapter 7

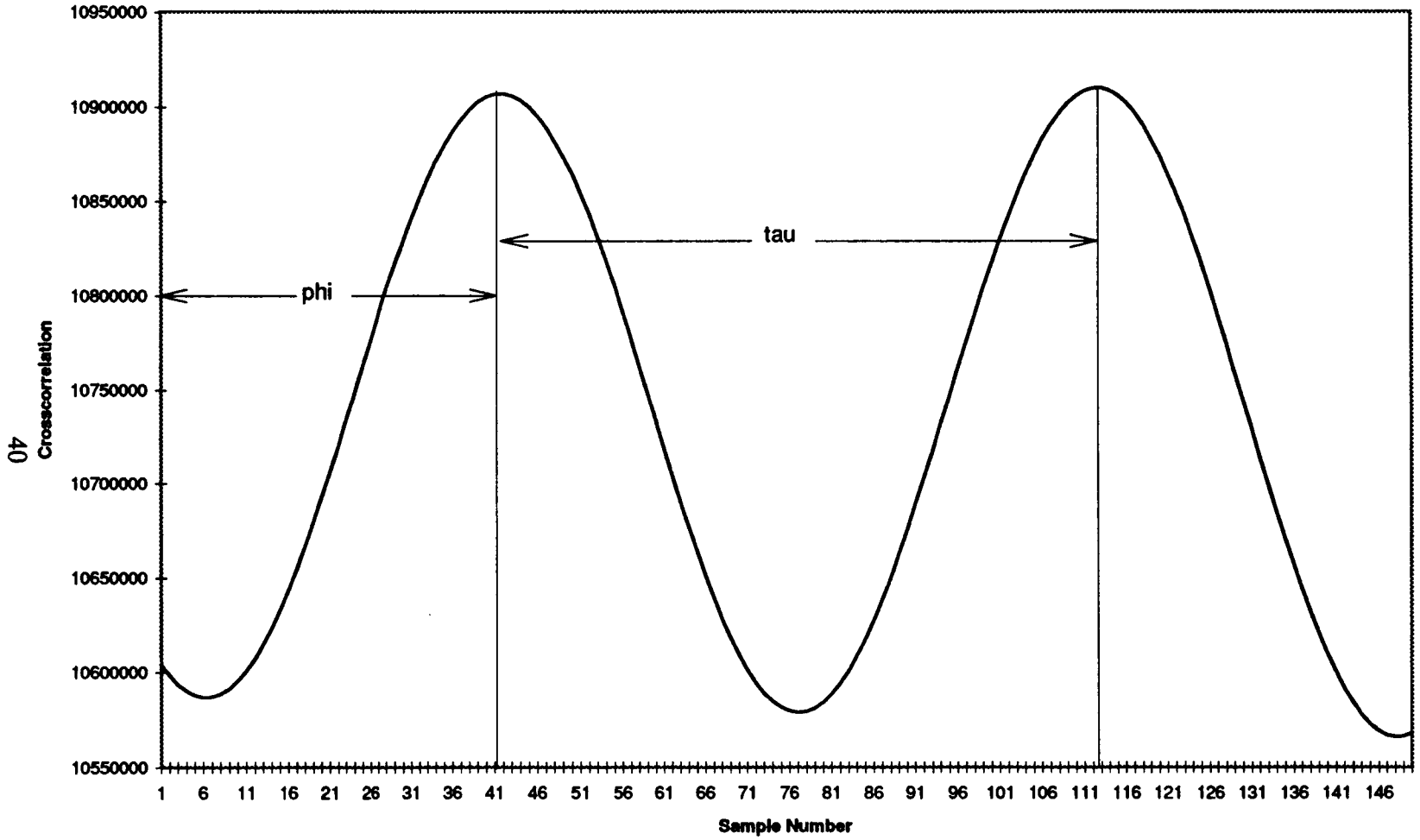
Program Checkout

Program checkout consisted of two parts: testing basic program validity and testing for the limits at which the program fails in terms of signal noise. This chapter will discuss these two aspects of program checkout.

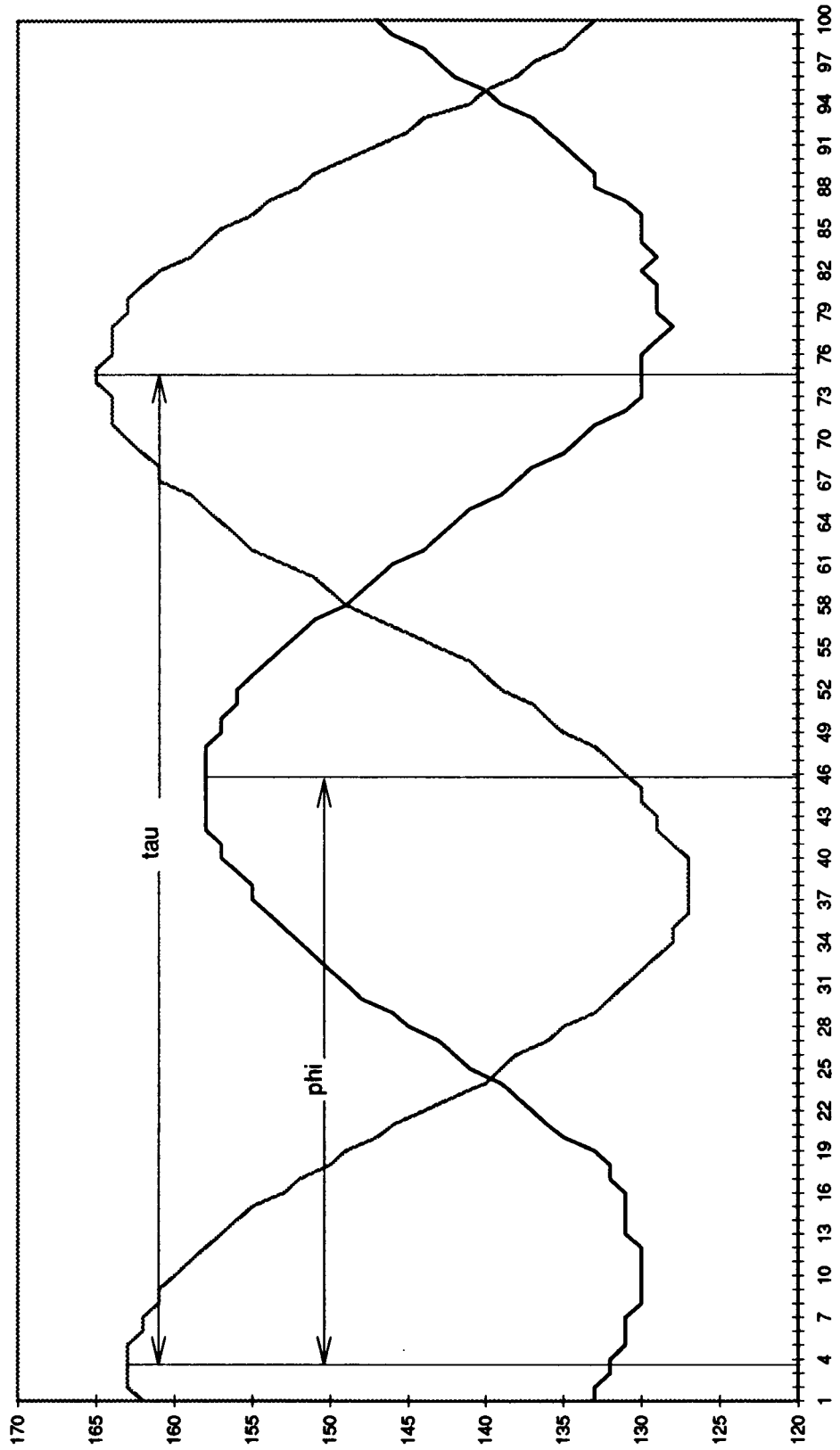
Program Validity

The programming for data reduction was tested for functionality. The crosscorrelation can be calculated in the time domain (eqs. 5.1 and 5.2) and in the frequency domain (eq. 5.7). If the crosscorrelation of the signals calculated in the time domain is compared to the calculation in the frequency domain, they should be equal. If so it is highly likely that both methods are programmed correctly. The crosscorrelation was computed for the signals in Figure 5.1 in both domains. They were identical, so this verified that the crosscorrelation was programmed correctly.

In theory, the crosscorrelation should peak at the sample number of the phase shift. The period of the input, τ, should be the difference between two consecutive peaks. Figure 7.1 is the first 150 samples from the crosscorrelation calculated in Figure 5.3. The first peak is at the forty-first sample and the difference between two peaks is approximately 70 samples. Looking at Figure 7.2 which is the first 100 samples of Figure 5.1, the period of the signals is approximately 40 samples, and the phase shift is approximately 71 samples.



150 Samples of the Crosscorrelation
Figure 7.1



100 Samples of X and Y
Figure 7.2

Notice that this information is more easily extracted from the clean crosscorrelation of Figure 7.1.

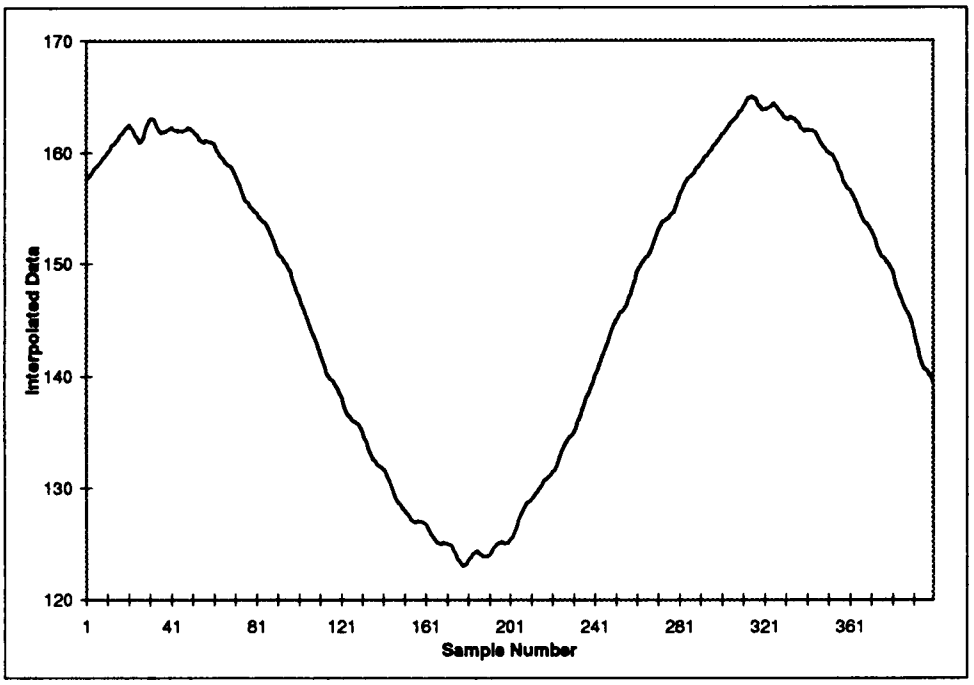
A valid interpolated sequence should be identical to the original sequence except there are more samples. In Figure 7.3 a data sequence is shown with the corresponding interpolated sequence. In general, the waveforms are identical, but the interpolated sequence appears smoother due to the additional plotting points.

Program Limits

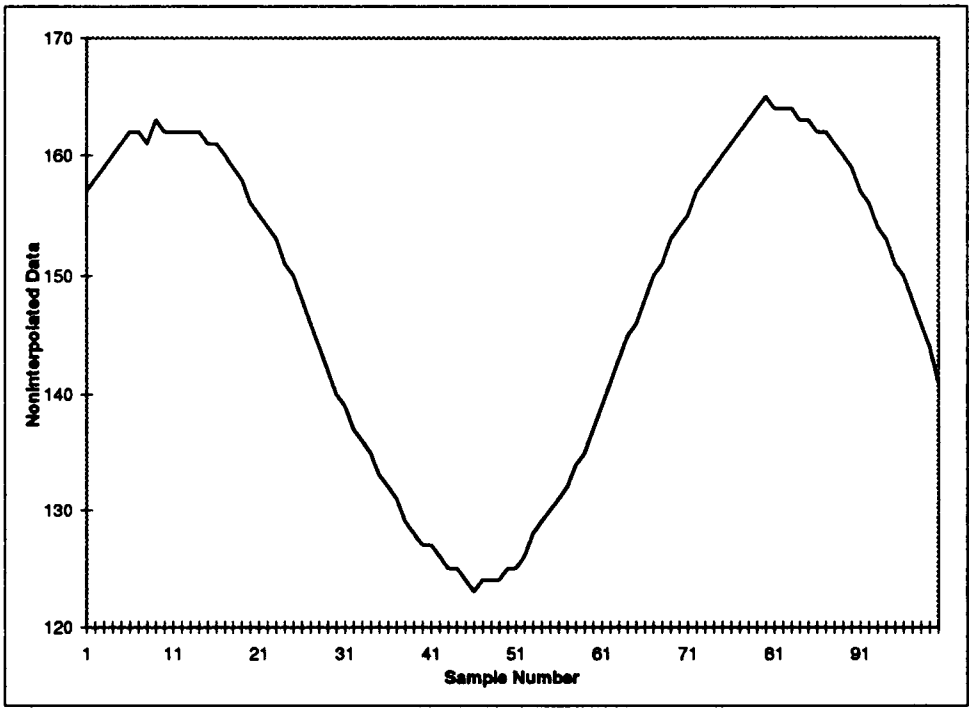
If the input signals have a great deal of noise, the crosscorrelation will likely contain noise. The phase and period determination can be incorrect if excessive noise is present with the input signal.

The amount of noise in a photodetector signal is influenced by the signal visibility and the background light power. The signal visibility is the ratio of the magnitude of the "ac" component of the signal to the "pedestal" component or mean. The background light power refers to the non-signal photons that the photodetector sees [10]. These photons increase the shot noise of the detector making the signal noisy. This shot noise is proportional to the square root of the photocurrent produced by these non-signal photons.

To test the limits of the program, a way was needed to generate known signals with noise. A program was used to produce signals similar to the signals produced by the experiment [8]. Noise could be superimposed on these signals. The signal visibility, VIS,



(a)



(b)

Figure 7.3 - Interpolation of Input Data: (a) Interpolated Data, (b) Original Data

and the background light power, BGRND, can be varied from 0 to 100 percent. The signals are produced using a random number generator. For the purposes of this checkout, the seed of the random number generator, iseed, was varied from -10 to +10, with each case generating a different signal in terms of its random noise components.

To test the limits of the calculation of the crosscorrelation, signal visibility was varied while the background light power was held constant and vice versa. The results of this checkout are shown in Tables 7.1 and 7.2. The signal, x, is always produced to have zero degree phase shift and iseed=-1. The y signal is produced to be ninety degrees phase shifted, but iseed is varied. For each row of the tables, y is produced 10 times with iseeds of -10, -8, -4, -2,-1, +1, +2, +4, +8, and +10. The mean and standard deviation is calculated for each row. The maximum and minimum errors are also listed.

From Table 7.1 the mean of the measured phase shifts has very little error as long as the signal visibility is 75% or greater. Below 75% the crosscorrelation produces phase shift measurements results that are unacceptable.

From Table 7.2 the mean of the measured phase shifts is good as long as the background light power is less than 25%. It is also good if x is 25% and y is 50%. If the background light power is above 50%, the data reduction results are unacceptable.

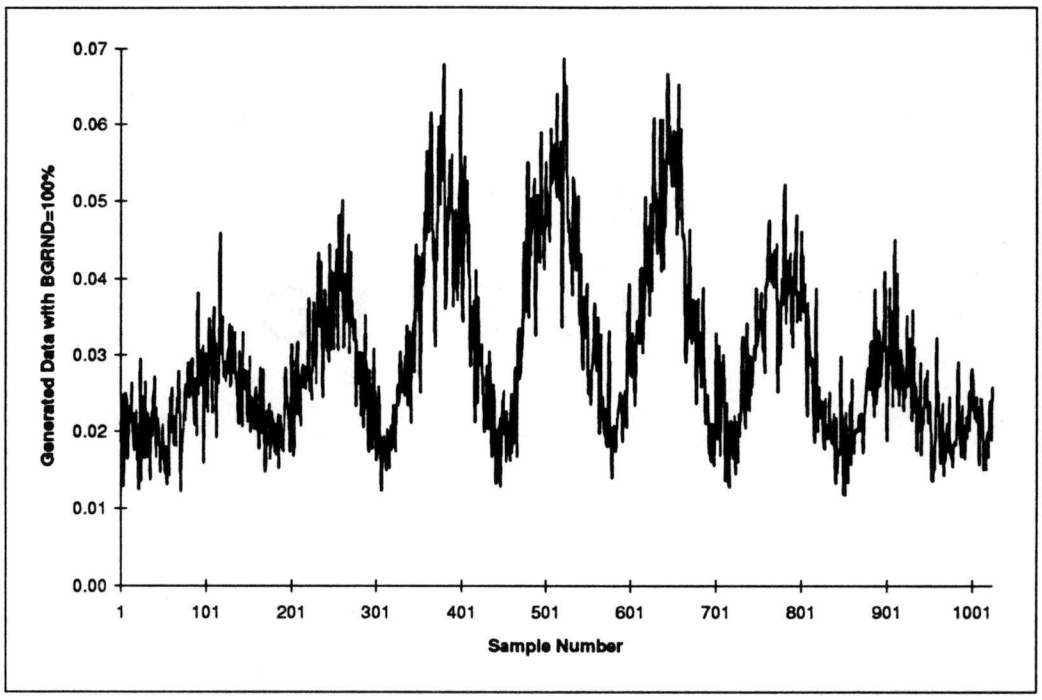
The data from some of these signals would be discarded in the programs because their measured phase shifts are greater than 360 degrees. This was discussed in Chapter 5. Examples of signals with vis=25% and bgrnd=100% are shown in Figure 7.4. These signals are extremely noisy. As long as the experiment generating the signals is constrained within the minimum visibility and the maximum background light power, the

Table 7.1 - Mean and Standard Deviation for X and Y with varied Signal Visibility

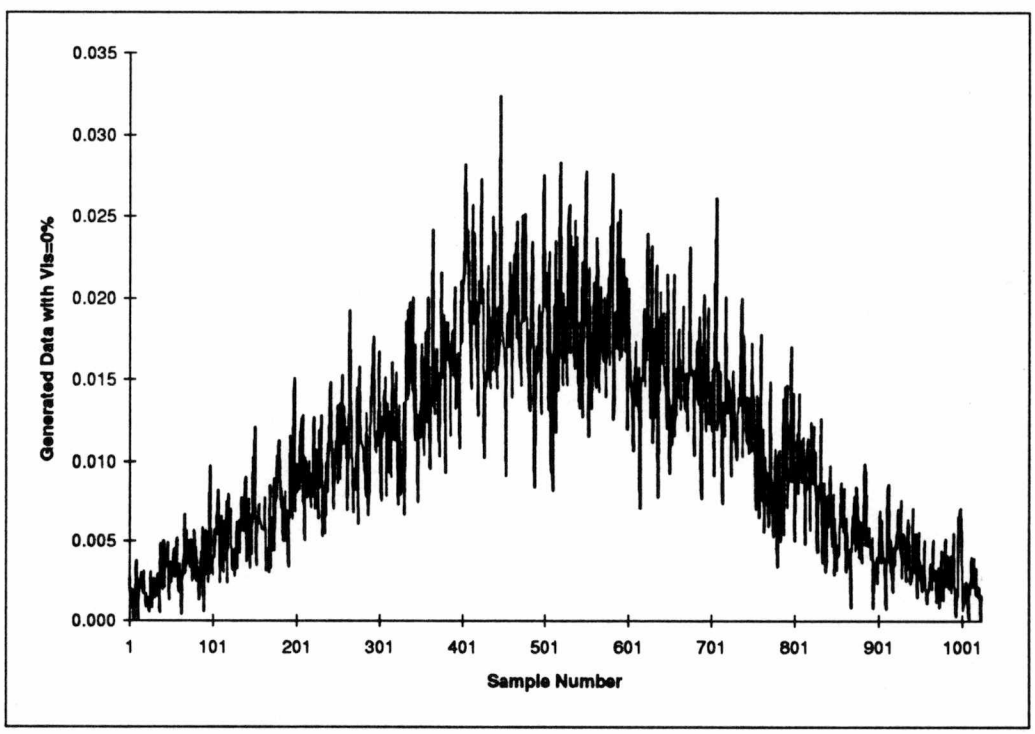
X	Y	ERROR (degrees)		PHASE (degrees)	
VIS	VIS	MAX	MIN	MEAN	STDDEV
1.00	1.00	3.44	0.00	90.55	2.01
1.00	0.75	7.33	0.00	89.08	3.36
0.75	0.75	6.92	0.00	91.47	4.27
1.00	0.50	92.95	0.00	101.29	30.55
0.75	0.50	145.64	2.73	132.21	57.67
0.50	0.50	1098.00	2.81	245.24	338.38
1.00	0.25	153.87	1.38	143.22	64.39
0.75	0.25	2224.29	7.33	381.17	682.25
0.50	0.25	2550.00	41.43	1218.08	1050.81
0.25	0.25	8550.00	303.49	4280.92	3047.26

Table 7.2 - Mean and Standard Deviation for X and Y with varied Background Light Power

X	Y	ERROR (degrees)		PHASE (degrees)	
BGRND	BGRND	MAX	MIN	MEAN	STDDEV
0.00	0.00	3.44	0.00	90.55	2.01
0.00	0.25	4.15	0.00	90.84	2.72
0.25	0.25	7.67	0.00	91.33	3.15
0.00	0.50	6.18	0.00	91.86	3.16
0.25	0.50	6.92	0.00	91.31	3.12
0.50	0.50	105.25	0.00	101.97	32.85
0.00	0.75	56.88	0.00	100.49	20.74
0.25	0.75	1770.00	0.00	282.21	556.31
0.50	0.75	213.16	0.00	123.58	73.21
0.75	0.75	6.92	0.68	92.01	3.62
0.00	1.00	132.55	0.00	131.65	57.41
0.25	1.00	59.76	0.00	98.13	18.86
0.50	1.00	9.69	0.00	93.84	2.49
0.75	1.00	150.00	0.00	108.76	46.21
1.00	1.00	298.24	0.00	135.90	96.99



(a)



(b)

Figure 7.4 - Generated Signals with Noise:
(a) Signal with Background Light Power = 100%, (b) Signal with Signal Visibility = 0%

results will be acceptable. The measured input signals of the experiment are well within the bounds for acceptable program results.

Chapter 8

Conclusions

Programs were written that can acquire data from two signals generated by photodetectors that detect scattered light from a particle passing through a probe volume. Two DSP cards were programmed to store data for approximately 1000 particles. The programs use data from RAM on these cards one particle at a time and calculate the period of the signals and the phase shift between them.

The crosscorrelation function was used to calculate the phase shift and period of the two signals. This method functions well, but it has its limitations. The crosscorrelation function is sensitive to the sample rate. If the sample rate is too high, not enough periods of the signal are available to calculate an accurate crosscorrelation. If the sample rate is too low, the measurement of the peak of the crosscorrelation function can be hard to determine accurately because it can fall between the samples. Provisions were added to the programs that will discard measurements when the number of periods is not between five and twenty.

The data can be digitally interpolated with acceptable results. This is necessary if the particles are traveling at a high velocity through the probe volume and increasing the clock frequency can not decrease the number of periods to fewer than twenty.

It was shown that the data system adds no significant magnitude or phase errors. This was done using input signals with known characteristics. Also, it was shown that the program will produce usable results provided the signal visibility is 75% or greater and

the background light power is 25% or lower. An area for future work would be to improve the program so it will calculate accurate data for noisier signals than the above limitations currently allow. One area that is being looked at is bandpass filtering the data before the crosscorrelation is computed. This would hopefully remove much of the noise present in the data.

The program accomplishes the objectives set forth by the CLA, but it requires a large amount of time to complete the data reduction. It takes approximately two seconds to reduce the data for one particle. For 1000 particles, this is approximately 33 minutes.

The DSP cards used for this experiment are a few years old, use fixed-point arithmetic, and do not have the capabilities available in some of the newer units. Another area for future improvement would be to purchase two new DSP cards that use floating-point digital signal processors. The data reduction could then be accomplished on the DSP cards themselves. This would greatly increase the execution speed.

Overall, the program achieved the objectives set forth by the CLA. The program inputs two signals and accurately calculates the period and the phase shift between them. The CLA is currently using the program to estimate the size and velocity of jet fuel particles.

REFERENCES

REFERENCES

1. A. Naqwi and F. Durst, "Light Scattering applied to LDA and PDA Measurements, Part 2: Computational Results and their Discussion," *Particle and Particle Systems Characterization,* vol.9, pp. 66-80, 1992.

2. A. Naqwi and F. Durst, "Light Scattering applied to LDA and PDA Measurements, Part 1: Theory and Numerical Treatments," *Particle and Particle Systems Characterization,* vol. 8, pp. 245-258, 1991.

3. W.D. Bachelo and M.J. Houser, "Phase/Doppler spray analyzer for simultaneous measurements of drop size and velocity distributions," *Optical Engineering,* vol. 23, pp. 583-590, 1984.

4. Signetec Corporation, *Signatec DASP100 Operators Manual,* Corona, CA, 1989.

5. Signatec Corporation, *Cbuild Reference Manual Version 3.5,* Corona, CA, 1989-1991.

6. F. Durst, A. Melling, and J.H. Whitelaw: *Principles and Practice of Laser Doppler Anemometry.* New York: Academic Press, 1976.

7. J.G. Proakis and D.G. Manolakis: *Digital Signal Processing, Principles, Algorithms, and Applications*, 2nd ed. New York: Macmillan Publishing Company, 1992.

8. B.W. Bomar, "Improved Algorithms and an Accuracy Analysis for the Discrete Fourier Transform Laser Velocimeter Signal Processor," UTSI Report No. UTSI 86/06, Tullahoma, TN, 1986.

9. M. Ross, *Laser Receivers, Devices, Techniques, Systems.* New York: John Wiley & Sons, Inc., 1966.

10. D.W. Roberds, C.W. Brasier and B.W. Bomar, "Use of a particle interferometer to study water droplet size distribution," *Optical Engineering,* vol. 18 no. 3, pp. 236-242, May/June 1979.

11. E. C. Ifeachor and B.W. Jervis, *Digital Signal Processing.* England: Addison-Wesley, 1993.

12. J. S. Bendat and A.G. Piersol, *Random Data: Analysis and Measurement Procedures,* New York: John Wiley & Sons, Inc. 1971.

APPENDICES

APPENDIX A

PROGRAM REDUCE3.FOR

```
      INTERFACE TO SUBROUTINE SETSEG [C] (IS)
      INTEGER*2 IS [FAR,REFERENCE]
      END

      INTERFACE TO SUBROUTINE SEARCH [C] (IS,IA,MAXSEG,ISEG,IERR)
      INTEGER*2 IS [FAR,REFERENCE]
      INTEGER*2 IA [FAR,REFERENCE]
      INTEGER*2 MAXSEG [FAR,REFERENCE]
      INTEGER*2 ISEG [FAR,REFERENCE]
      INTEGER*2 IERR [FAR,REFERENCE]
      END

      INTERFACE TO SUBROUTINE GETDAT [C] (IA,DATA_HOLD1,DATA_HOLD2)
      INTEGER*1 DATA_HOLD1 [FAR,REFERENCE] (1024)
      INTEGER*1 DATA_HOLD2 [FAR,REFERENCE] (1024)
      INTEGER*2 IA [FAR,REFERENCE]
      END

      INTERFACE TO SUBROUTINE SETUP [C] (CONFIG)
      CHARACTER CONFIG [FAR,REFERENCE] (256)
      END

      PROGRAM REDUCE3
      PARAMETER (MX=1024)
      DIMENSION RXYR(MX),XI(MX),YI(MX),WR(MX),WI(MX),PHSE(2048)
      CHARACTER*30 STRINGX,STRINGY
      CHARACTER*3 EXTEN
      COMPLEX XF(MX),YF(MX),RXYF(MX),RXY(MX)
      DIMENSION RXYI(MX),X(MX),Y(MX),YR(MX),XR(MX)
      INTEGER*1 DATA_HOLD1( MX),DATA_HOLD2( MX)
      INTEGER A,B,C,D,E,F,G,H,I,J,K,L
      CHARACTER*15 AT,BT,CT,FT,GT,IT,JT,KT
      CHARACTER P
      INTEGER*2 IS,ISEG,MAXSEG
      INTEGER*2 IA,IERR
      CHARACTER*30 TAU,FPHASE
      CHARACTER CONFIG(256)
      WRITE (*,'( "   FILE NAME FOR PHASE", )')
      READ(*,'(A)')FPHASE
      OPEN(UNIT=3,FILE=FPHASE)
      WRITE (*,'( "   FILE NAME FOR TAU", )')
      READ(*,'(A)')TAU
      OPEN(UNIT=4,FILE=TAU)
      OPEN(UNIT=1,FILE='DASP_SET.CFG')
      READ(1,210)A,B,C,D,E,F,G,H,I,J,K,L
210   FORMAT(I3,/,I3,/,I3,/,I5,/,I4,/,I3,/,I3,/,I3,/,I3,/,I3,/,I3,/,
     $I6)
      WRITE(*,220)
220   FORMAT(/,' THE FOLLOWING ARE THE DEFAULT PARAMETERS FOR THE',/
     $' DATA ACQUISITION PROGRAM.  FOR MORE INFORMATION ABOUT THESE',/
```

PROGRAM REDUCE3.FOR

```
     $' PARAMETERS REFER TO README.SET.')
      IF (A.EQ.0) AT = ' LINEAR    '
      IF (A.EQ.1) AT = ' COMPRESSION'
      IF (B.EQ.0) BT = ' BURST     '
      IF (B.EQ.1) BT = ' PRETRIGGER '
      IF (B.EQ.2) BT = ' CONTINUOUS '
      IF (B.EQ.3) BT = ' SEGMENTED  '
      IF (C.EQ.0) CT = ' 60MV      '
      IF (C.EQ.1) CT = ' 200MV     '
      IF (C.EQ.2) CT = ' 600MV     '
      IF (C.EQ.3) CT = ' 2V        '
      IF (F.EQ.0) FT = ' 100MHZ    '
      IF (F.EQ.1) FT = ' 50MHZ     '
      IF (F.EQ.2) FT = ' 25MHZ     '
      IF (F.EQ.3) FT = ' 12.5MHZ   '
      IF (F.EQ.4) FT = ' 6.25MHZ   '
      IF (F.EQ.5) FT = ' 3.125MHZ  '
      IF (F.EQ.6) FT = ' 1.56MHZ   '
      IF (F.EQ.7) FT = ' 781KHZ    '
      IF (G.EQ.0) GT = ' 16K       '
      IF (G.EQ.1) GT = ' 32K       '
      IF (G.EQ.2) GT = ' 64K       '
      IF (G.EQ.3) GT = ' 128K      '
      IF (G.EQ.4) GT = ' 256K      '
      IF (G.EQ.5) GT = ' 512K      '
      IF (G.EQ.6) GT = ' 1M        '
      IF (G.EQ.7) GT = ' 2M        '
      IF (I.EQ.0) IT = ' EXTERNAL  '
      IF (I.EQ.1) IT = ' INTERNAL  '
      IF (J.EQ.0) JT = ' AC        '
      IF (J.EQ.1) JT = ' DC        '
      IF (K.EQ.0) KT = ' NEGATIVE  '
      IF (K.EQ.1) KT = ' POSITIVE  '
      WRITE(*,230)AT,BT,CT,D,E,FT,GT,H,IT,JT,KT,L
  230 FORMAT(/,'  AMPLIFIER TYPE   - ',A11,/
     1' ACQUISITION MODE - ',A11,/
     2' VOLTAGE RANGE    - ',A11,/
     3' ADC OFFSET       - ',I4,/
     4' THRESHOLD        - ',I4,/
     5' CLOCK DIVIDER    - ',A11,/
     6' ACTIVE MEMORY    - ',A11,/
     7' TIMEOUT CYCLES   - ',I3,/
     8' TRIGGER SOURCE   - ',A11,/
     9' TRIGER COUPLING  - ',A11,/
     1' TRIGGER SLOPE    - ',A11,/
     2' #DATA POINTS     - ',I6,/)
      WRITE(*,*)'DO YOU WISH TO CHANGE ANY OF THE PARAMETERS? Y OR N'
      READ(*,240)P
  240 FORMAT(A1)
      IF(P.EQ.'Y '.OR. P.EQ.'Y ')CALL M(A,B,C,D,E,F,G,H,I,J,K,L)
```

PROGRAM REDUCE3.FOR

```
570  FORMAT(I6)
     CONFIG(1) = A
     CONFIG(2) = B
     CONFIG(3) = C
     CONFIG(4) = D
     CONFIG(5) = E
     CONFIG(6) = F
     CONFIG(7) = G
     CONFIG(8) = H
     CONFIG(9) = I
     CONFIG(10) = J
     CONFIG(11) = K
     NDATA = L
     N = L
     REWIND 1
     WRITE(1,250)A,B,C,D,E,F,G,H,I,J,K,L
250  FORMAT(I1,/,I1,/,I1,/,I3,/,I3,/,I1,/,I1,/,I1,/,I1,/,I1,/,I1,/,
    $I4)
     CLOSE(UNIT=1)
     IF (A.EQ.0) AT = ' LINEAR    '
     IF (A.EQ.1) AT = ' COMPRESSION'
     IF (B.EQ.0) BT = ' BURST     '
     IF (B.EQ.1) BT = ' PRETRIGGER '
     IF (B.EQ.2) BT = ' CONTINUOUS '
     IF (B.EQ.3) BT = ' SEGMENTED  '
     IF (C.EQ.0) CT = ' 60MV      '
     IF (C.EQ.1) CT = ' 200MV     '
     IF (C.EQ.2) CT = ' 600MV     '
     IF (C.EQ.3) CT = ' 2V        '
     IF (F.EQ.0) FT = ' 100MHZ    '
     IF (F.EQ.1) FT = ' 50MHZ     '
     IF (F.EQ.2) FT = ' 25MHZ     '
     IF (F.EQ.3) FT = ' 12.5MHZ   '
     IF (F.EQ.4) FT = ' 6.25MHZ   '
     IF (F.EQ.5) FT = ' 3.125MHZ  '
     IF (F.EQ.6) FT = ' 1.56MHZ   '
     IF (F.EQ.7) FT = ' 781KHZ    '
     IF (G.EQ.0) GT = ' 16K       '
     IF (G.EQ.1) GT = ' 32K       '
     IF (G.EQ.2) GT = ' 64K       '
     IF (G.EQ.3) GT = ' 128K      '
     IF (G.EQ.4) GT = ' 256K      '
     IF (G.EQ.5) GT = ' 512K      '
     IF (G.EQ.6) GT = ' 1M        '
     IF (G.EQ.7) GT = ' 2M        '
     IF (I.EQ.0) IT = ' EXTERNAL  '
     IF (I.EQ.1) IT = ' INTERNAL  '
     IF (J.EQ.0) JT = ' AC        '
     IF (J.EQ.1) JT = ' DC        '
     IF (K.EQ.0) KT = ' NEGATIVE  '
```

PROGRAM REDUCE3.FOR

```
      IF (K.EQ.1) KT = ' POSITIVE  '
      WRITE(*,230)AT,BT,CT,D,E,FT,GT,H,IT,JT,KT,L
C **** INITIALIZE DATA ARRAYS ****
      DO I=1,NDATA
           DATA_HOLD1(I)=0
           DATA_HOLD2(I)=0
      ENDDO
      IA=10
      IS=0
      IF ((G.EQ.0).OR.(G.EQ.1)) THEN
           MAXSEG=0
      ELSE
           MAXSEG = 2**(G-1)-1
      ENDIF
      IF (G.GT.0) THEN
           ISEG = 32767-L
      ELSE
           ISEG = 16384-L
      ENDIF
C **** INITIALIZE A/D CARDS *****
      CALL SETUP(CONFIG)
C ****   ACQUIRE DATA  ******
      WRITE(*,13)
13    FORMAT('    DATA ACQUISITION FINISHED')
C **** CARD MEMORY IS FULL, READ INDIVIDUAL PARTICLES  ****
      IPARTICLES=1
      CALL GETDAT(IA,DATA_HOLD1,DATA_HOLD2)
C **** CONVERT C DATA TYPES UNSIGNED CHAR TO FORTRAN INTEGER ****
800   DO I=1,NDATA
           IF(DATA_HOLD1(I).LT.0)THEN
              X(I)=DATA_HOLD1(I)+256.0
           ELSE
              X(I)=DATA_HOLD1(I)
           ENDIF
           IF(DATA_HOLD2(I).LT.0)THEN
              Y(I)=DATA_HOLD2(I)+256.0
           ELSE
              Y(I)=DATA_HOLD2(I)
           ENDIF
           XR(I)=X(I)
           YR(I)=Y(I)
C **** LOCATING DATA WITH NO 255 COUNT SPIKES ****
      IF ((X(I).EQ.255.0).OR.(Y(I).EQ.255.0))THEN
           IPARTICLES = IPARTICLES-1
           GOTO 330
      ENDIF
      ENDDO
C **** COPYING THE FIRST 80 X AND Y FILES TO THE VIRTUAL DISK ****
      IF(IPARTICLES.GT.80)GOTO 815
           STRINGX(1:4)='E:\X'
```

```
        STRINGY(1:4)='E:\Y'
        STRINGX(5:5)='.'
        STRINGY(5:5)='.'
        WRITE(EXTEN,1000)IPARTICLES
        STRINGX(6:8)=EXTEN(1:3)
        STRINGY(6:8)=EXTEN(1:3)
        OPEN(UNIT=6,FILE=STRINGX(1:8),FORM='UNFORMATTED')
        WRITE(6)INT2(NDATA),(FLOAT(I),X(I),I=1,NDATA)
        CLOSE(UNIT=6)
        OPEN(UNIT=6,FILE=STRINGY(1:8),FORM='UNFORMATTED')
        WRITE(6)INT2(NDATA),(FLOAT(I),Y(I),I=1,NDATA)
        CLOSE(UNIT=6)
815  CONTINUE
1000    FORMAT(I3.3)
         DO I=1,NDATA
           XI(I)=0.0
         ENDDO
         DO I=1,NDATA
           YI(I)=0.0
         ENDDO
C **** THE FFT IS TAKEN OF BOTH SIGNALS ****
         N=NDATA
         CALL INITAB(N,WR,WI)
         CALL FFTSUB(N,X,XI,WR,WI)
         CALL FFTSUB(N,Y,YI,WR,WI)
C **** CONJUGATE OF Y IS MULTIPLED BY X ****
         DO I=1,1024
           XF(I)=CMPLX(X(I),XI(I))
           YF(I)=CMPLX(Y(I),YI(I))
           RXYF(I)=CONJG(YF(I))*(XF(I))
           RXYR(I)=REAL(RXYF(I))
           RXYI(I)=IMAG(RXYF(I))
         ENDDO
         CALL FFTSUB(N,RXYR,RXYI,WI,WR)
C **** INVERSE FFT OF THE CROSSCORRELATION IS TAKEN ****
         DO I=1,N
           RXY(I)=CMPLX(RXYR(I),RXYI(I))/N
         ENDDO
C **** THE FIRST PEAK OF THE CROSS CORRELATION FUNCTION
C **** IS THE NUMBER OF SAMPLES LAG BETWEEN THE TWO SIGNALS. THE NUMBER
C **** OF SAMPLES BETWEEN TWO PEAKS IS THE PERIOD OF THE SIGNAL.  THE
C **** TIME PERIOD IS CALCULATED #SAMPLES/CLOCK FREQUENCY. THE PHASE IN
C **** DEGREES IS CALCULTED 360*#SAMPLES LAG/#SAMPLES BETWEEN PEAKS.
         J=1
         DO I=1,1023
           N=I+1
           IF(CABS(RXY(N)).GT.CABS(RXY(N-1)))GOTO 280
           J=J+1
         ENDDO
280     DO I=1,1024
```

```
          K=J+I
          IF(CABS(RXY(K)).LT.CABS(RXY(K-1)))GOTO 430
          JSAMPLE=K-1
        ENDDO
430     DO I=1,1024-JSAMPLE
          J=JSAMPLE+5+I
          IF(CABS(RXY(J)).GT.CABS(RXY(J-1)))GOTO 440
          KPERIODD=J-1
        ENDDO
440     DO I=1,1024-KPERIODD
          J=KPERIODD+1+I
          IF(CABS(RXY(J)).LT.CABS(RXY(J-1)))GOTO 270
          KSAMPLE=J
        ENDDO
270     KPERIOD=KSAMPLE-(JSAMPLE+1)
C **** THE CROSSCORRELATION IS RECOMPUTED WITH AN INTEGRAL NUMBER OF
PERIODS.
        NP=512/KPERIOD
        WRITE(*,*)NP
        NM=NP*KPERIOD
        WRITE(*,*)NM
        DO I=(NM+1),NDATA
          YR(I)=0.0
        ENDDO
        DO I=1,1024
          XI(I)=0.0
          YI(I)=0.0
        ENDDO
        CALL FFTSUB(N,XR,XI,WR,WI)
        CALL FFTSUB(N,YR,YI,WR,WI)
        DO I=1,1024
          XF(I)=CMPLX(XR(I),XI(I))
          YF(I)=CMPLX(YR(I),YI(I))
          RXYF(I)=CONJG(YF(I))*(XF(I))
          RXYR(I)=REAL(RXYF(I))
          RXYI(I)=IMAG(RXYF(I))
        ENDDO
        CALL FFTSUB(N,RXYR,RXYI,WI,WR)
        DO I=1,N
          RXY(I)=CMPLX(RXYR(I),RXYI(I))/N
        ENDDO
170     FORMAT(I5,3X,F20.4)
          J=1
          KPERIOD=1
          JSAMPLE=0
        DO I=1,N-1
          ML=I+1
          IF(CABS(RXY(ML)).GT.CABS(RXY(ML-1)))GOTO 1280
          J=J+1
        ENDDO
```

PROGRAM REDUCE3.FOR

```
1280    DO I=1,N
          K=J+I
          IF(CABS(RXY(K)).LT.CABS(RXY(K-1)))GOTO 1230
          JSAMPLE=K-1
        ENDDO
1230    DO I=1,N-JSAMPLE
          J=JSAMPLE+5+I
          IF(CABS(RXY(J)).GT.CABS(RXY(J-1)))GOTO 1240
          KPERIODD=J-1
        ENDDO
1240    DO I=1,N-KPERIODD
          J=KPERIODD+1+I
          IF(CABS(RXY(J)).LT.CABS(RXY(J-1)))GOTO 1270
          KSAMPLE=J
        ENDDO
1270    KPERIOD=KSAMPLE-(JSAMPLE+1)
        PHASE=360.0*JSAMPLE/KPERIOD
        WRITE(*,*)JSAMPLE
        TPERIOD=1.0/(100000000.0/2.0**F)*KPERIOD*1000000.0
C
C **** REJECT PARTICLES OUT OF MEASUREMENT RANGE ****
        IF ((KPERIOD.LT.(40.0)).OR.
   &    (ABS(PHASE).GT.360.0).OR.(KPERIOD.GT.200))THEN
          IPARTICLES = IPARTICLES - 1
          GOTO 330
        ENDIF

        WRITE(*,311)IPARTICLES
        WRITE(*,300)PHASE
        WRITE(*,310)TPERIOD
300  FORMAT(' X LAGS Y BY ',1PG10.3,1X,'DEGREES. THE PERIOD OF THE ')
310     FORMAT(' SIGNAL IS ',F8.2,1X,'MICROSECONDS.')
311  FORMAT(5X,' PARTICLE # ',I5)
C **** THE PHASE AND TIME PERIOD OF THE SIGNAL ARE WRITTEN TO FILES
C **** SPECIFIED AT THE START OF THE PROGRAM.
        WRITE(3,*)IPARTICLES,PHASE
        WRITE(4,*)IPARTICLES,TPERIOD
320     FORMAT(5X,F8.2)


C **** CHECK IF ALL PARTICLES HAVE BEEN READ ****
330  IF ((IS.GT.MAXSEG).OR.
   &  ((IS.EQ.MAXSEG).AND.(IA.GT.ISEG))) GOTO 820
   IF (IA.GT.ISEG) THEN
        IA = 1
        IS = IS+1
   ELSE
        IA = IA+1
   ENDIF
```

PROGRAM REDUCE3.FOR

```
340  CALL SEARCH(IS,IA,MAXSEG,ISEG,IERR)
     CALL SETSEG(IS)
     IF((IS.EQ.MAXSEG).AND.(IA.GT.ISEG)) GOTO 820
     IF(IERR.NE.0)GOTO 820
     CALL GETDAT(IA,DATA_HOLD1,DATA_HOLD2)

     DO WHILE((DATA_HOLD1(1).EQ.-1).AND.(IA.LE.ISEG))
          IA=IA+1
          CALL GETDAT(IA,DATA_HOLD1,DATA_HOLD2)
     ENDDO
     IF ((IS.LT.MAXSEG).AND.(IA.GT.ISEG)) THEN
          IS = IS+1
          IA = 1
          GOTO 340
     ENDIF
     IF ((IS.EQ.MAXSEG).AND.(IA.GT.ISEG)) GOTO 820

     IPARTICLES = IPARTICLES+1
     GOTO 800
C **** ALL PARTICLES DONE ****
820  WRITE (*,821) IPARTICLES
821  FORMAT (10X,I5,' PARTICLES ACQUIRED')
     OPEN(UNIT=9,FILE='CORREL.DAT')
          DO I=1,1024
            WRITE(9,*)I,CABS(RXY(I))
          ENDDO
     PMEAN=0.0
     REWIND 3
     REWIND 7
     DO I=1,IPARTICLES
          READ(3,*)J,PHSE(I)
          PMEAN=PMEAN+PHSE(I)
     ENDDO
          WRITE(*,850)PMEAN/IPARTICLES
850     FORMAT(' THE MEAN OF THE PHASES IS',F8.2)
     CLOSE(3)
     CLOSE (4)
     CLOSE (9)
     END

     SUBROUTINE M(A,B,C,D,E,F,G,H,I,J,K,L)
     INTEGER A,B,C,D,E,F,G,H,I,J,K,L
     CHARACTER S


1    WRITE(*,10)
10   FORMAT(/,' A - AMPLIFIER TYPE',/
    1' B - ACQUISITION MODE',/
    2' C - VOLTAGE RANGE',/
    3' D - ADC OFFSET',/
```

PROGRAM REDUCE3.FOR

```
   4' E - THRESHOLD',/
   5' F - CLOCK DIVIDER',/
   6' G - ACTIVE MEMORY',/
   7' H - TIMEOUT CYCLES',/
   8' I - TRIGGER SOURCE',/
   9' J - TRIGGER COUPLING',/
   1' K - TRIGGER SLOPE',/
   2' L - #DATA POINTS',/
   3' X - EXIT           - ',$)
20   READ(*,30)S
30   FORMAT(A1)
     IF (S.EQ.'A' .OR. S.EQ.'A') THEN
           WRITE (*,101)
101     FORMAT(' 0 = LINEAR, 1 = COMPRESSED')
           GOTO 100
     ELSE IF (S.EQ.'B' .OR. S.EQ.'B') THEN
           WRITE(*,201)
201     FORMAT(' 0 = BURST, 1= PRETRIGGER, 2 = CONTINUOUS,'
    +' 3 = SEGMENTED')
           GOTO 200
     ELSE IF (S.EQ.'C' .OR. S.EQ.'C') THEN
           WRITE(*,301)
301     FORMAT(' 0 = 60MV, 1 = 200MV, 2 = 600MV, 3 = 2V')
           GOTO 300
     ELSE IF (S.EQ.'D'.OR. S.EQ.'D') THEN
           WRITE(*,401)
401     FORMAT(' 0 TO 255')
           GOTO 400
     ELSE IF (S.EQ.'E' .OR. S.EQ.'E') THEN
           WRITE(*,501)
501     FORMAT(' 0 TO 255')
           GOTO 500
     ELSE IF (S.EQ.'F' .OR. S.EQ.'F') THEN
           WRITE(*,601)
601     FORMAT(' 0 = 100M, 1 = 50M, 2 = 25M, 3 = 12.5M, 4 = 6.25M,'
    +' 5 = 3.125M, 6 = 1.5625M,     7 = 781K')
           GOTO 600
     ELSE IF (S.EQ.'G' .OR. S.EQ.'G') THEN
           WRITE(*,701)
701     FORMAT(' 0 = 16K, 1 = 32K, 2 = 64K, 3 = 128K, 4 = 256K,'
    +' 5 = 512K, 6 = 1M, 7 = 2M')
           GOTO 700
     ELSE IF (S.EQ.'H' .OR. S.EQ.'H') THEN
           WRITE(*,801)
801     FORMAT(' 0 TO 31 ODD VALUES ONLY')
           GOTO 800
     ELSE IF (S.EQ.'I' .OR. S.EQ.'I') THEN
           WRITE(*,901)
901     FORMAT(' 0 = EXTERNAL, 1 = INTERNAL')
           GOTO 900
```

PROGRAM REDUCE3.FOR

```
   ELSE IF (S.EQ.'J'.OR. S.EQ.'J') THEN
         WRITE(*,1001)
1001     FORMAT(' 0 = AC, 1 = DC')
         GOTO 1000
   ELSE IF (S.EQ.'K' .OR. S.EQ.'K') THEN
         WRITE(*,1101)
1101     FORMAT(' 0 = NEGATIVE, 1 = POSITIVE')
         GOTO 1100
   ELSE IF (S.EQ.'L' .OR. S.EQ.'L') THEN
         GOTO 1200
   ELSE IF (S.EQ.'X' .OR. S.EQ.'X')THEN
         GOTO 2000
   ELSE
         GOTO 1
   ENDIF
100  WRITE(*,*)' A'
   CALL EDIT(A)
   GOTO 1
200  WRITE(*,*)' B'
   CALL EDIT(B)
   GOTO 1
300  WRITE(*,*)' C'
   CALL EDIT(C)
   GOTO 1
400  WRITE(*,*)' D'
   CALL EDIT(D)
   GOTO 1
500  WRITE(*,*)' E'
   CALL EDIT(E)
   GOTO 1
600  WRITE(*,*)' F'
   CALL EDIT(F)
   GOTO 1
700  WRITE(*,*)' G'
   CALL EDIT(G)
   GOTO 1
800  WRITE(*,*)' H'
   CALL EDIT(H)
   GOTO 1
900  WRITE(*,*)' I'
   CALL EDIT(I)
   GOTO 1
1000 WRITE(*,*)' J'
   CALL EDIT(J)
   GOTO 1
1100 WRITE(*,*)' K'
   CALL EDIT(K)
   GOTO 1
1200 WRITE(*,*)' L'
   CALL EDIT(L)
```

PROGRAM REDUCE3.FOR

```
   GOTO 1
2000 RETURN
   END

   SUBROUTINE EDIT(Y)
   INTEGER Y
   WRITE(*,*)'ENTER NEW VALUE'
   READ(*,*)Y
   WRITE(*,*)Y
   RETURN
   END

C COOLY-TUKEY RADIX-2, DIF FFT PROGRAM.
C COMPLEX INPUT DATA IN ARRAYS XR AND XI.
C TABLE LOOK-UP OF W VALUES.
   SUBROUTINE FFTSUB (NPTS,XR,XI,WR,WI)
   DIMENSION XR(NPTS),XI(NPTS),WR(NPTS),WI(NPTS)
   M = NINT( ALOG(FLOAT(NPTS)) / ALOG(2.0) )
   N2 = NPTS
   DO 10 K=1, M
        N1 = N2
        N2 = N2 / 2
        IE = NPTS / N1
        IA = 1
        DO 20 J=1, N2
          C = WR(IA)
          S = WI(IA)
          IA = IA + IE
          DO 30 I=J,NPTS, N1
            L = I + N2
            XRT  = XR(I) - XR(L)
            XR(I) = XR(I) + XR(L)
            XIT  = XI(I) - XI(L)
            XI(I) = XI(I) + XI(L)
            XR(L) = C*XRT + S*XIT
            XI(L) = C*XIT - S*XRT
 30     CONTINUE
 20   CONTINUE
 10 CONTINUE
   CALL REVERS(NPTS,XR,XI)
   RETURN
   END

   SUBROUTINE REVERS (NPTS,XR,XI)
   DIMENSION XR(NPTS),XI(NPTS)
   J = 1
   N1 = NPTS - 1
   DO 70 I=1, N1
        IF (I.GE.J) GO TO 40
        XRT  = XR(J)
```

```
        XR(J) = XR(I)
        XR(I) = XRT
        XIT   = XI(J)
        XI(J) = XI(I)
        XI(I) = XIT
40    K = NPTS / 2
50    IF (K.GE.J) GO TO 60
        J = J - K
        K = K / 2
        GO TO 50
60    J = J + K
70  CONTINUE
    RETURN
    END

C INITIALIZE SINE AND COSINE TABLES FOR FFT'S WITH TABLE LOOKUP.

    SUBROUTINE INITAB (NPTS,WR,WI)
    DIMENSION WR(NPTS),WI(NPTS)
    P = 2.0 * ACOS(-1.0) / NPTS
    DO 10 N=1, NPTS
        A = (N-1) * P
        WR(N) =  COS(A)
        WI(N) = -SIN(A)
  10 CONTINUE
    RETURN
    END
```

PROGRAM RED2INTT.FOR

```
      INTERFACE TO SUBROUTINE SETSEG [C] (IS)
      INTEGER*2 IS [FAR,REFERENCE]
      END

      INTERFACE TO SUBROUTINE SEARCH [C] (IS,IA,MAXSEG,ISEG,IERR)
      INTEGER*2 IS [FAR,REFERENCE]
      INTEGER*2 IA [FAR,REFERENCE]
      INTEGER*2 MAXSEG [FAR,REFERENCE]
      INTEGER*2 ISEG [FAR,REFERENCE]
      INTEGER*2 IERR [FAR,REFERENCE]
      END

      INTERFACE TO SUBROUTINE GETDAT [C] (IA,DATA_HOLD1,DATA_HOLD2)
      INTEGER*1 DATA_HOLD1 [FAR,REFERENCE] (1024)
      INTEGER*1 DATA_HOLD2 [FAR,REFERENCE] (1024)
      INTEGER*2 IA [FAR,REFERENCE]
      END

      INTERFACE TO SUBROUTINE SETUP [C] (CONFIG)
      CHARACTER CONFIG [FAR,REFERENCE] (256)
      END

      PROGRAM RED2INTT
      PARAMETER (MX=2049)
      DIMENSION RXYR(MX),WR(MX),WI(MX),PHSE(2048)
      CHARACTER*30 STRINGX,STRINGY
      CHARACTER*3 EXTEN
      COMPLEX XF(MX),YF(MX),RXYF(MX),RXY(MX),XINC(MX),YINC(MX)
      DIMENSION RXYI(MX),X(MX),Y(MX),XIN(MX),YIN(MX)
      DIMENSION YINI(MX),XINI(MX),HR(MX),HI(MX),HC(MX)
      INTEGER*1 DATA_HOLD1( MX),DATA_HOLD2( MX)
      INTEGER A,B,C,D,E,F,G,H,I,J,K,L
      CHARACTER*15 AT,BT,CT,FT,GT,IT,JT,KT
      CHARACTER P
      INTEGER*2 IS,ISEG,MAXSEG
      INTEGER*2 IA,IERR
      CHARACTER*30 TAU,FPHASE
      CHARACTER CONFIG(256)
      WRITE (*,'(" FILE NAME FOR PHASE", )')
      READ(*,'(A)')FPHASE
      OPEN(UNIT=3,FILE=FPHASE)
      WRITE (*,'(" FILE NAME FOR TAU", )')
      READ(*,'(A)')TAU
      OPEN(UNIT=4,FILE=TAU)
      OPEN(UNIT=1,FILE='D:\SHARON\DASP_SET.CFG')
      OPEN(UNIT=9,FILE='D:\SHARON\CORREL.DAT')
      READ(1,210)A,B,C,D,E,F,G,H,I,J,K,L
210   FORMAT(I3,/,I3,/,I3,/,I5,/,I4,/,I3,/,I3,/,I3,/,I3,/,I3,/,
     $I6)
      WRITE(*,220)
```

```
220  FORMAT(/,' THE FOLLOWING ARE THE DEFAULT PARAMETERS FOR THE',/
     $' DATA ACQUISITION PROGRAM.  FOR MORE INFORMATION ABOUT THESE',/
     $' PARAMETERS REFER TO README.SET.')
     IF (A.EQ.0) AT = ' LINEAR    '
     IF (A.EQ.1) AT = ' COMPRESSION'
     IF (B.EQ.0) BT = ' BURST     '
     IF (B.EQ.1) BT = ' PRETRIGGER '
     IF (B.EQ.2) BT = ' CONTINUOUS '
     IF (B.EQ.3) BT = ' SEGMENTED  '
     IF (C.EQ.0) CT = ' 60MV      '
     IF (C.EQ.1) CT = ' 200MV     '
     IF (C.EQ.2) CT = ' 600MV     '
     IF (C.EQ.3) CT = ' 2V        '
     IF (F.EQ.0) FT = ' 100MHZ    '
     IF (F.EQ.1) FT = ' 50MHZ     '
     IF (F.EQ.2) FT = ' 25MHZ     '
     IF (F.EQ.3) FT = ' 12.5MHZ   '
     IF (F.EQ.4) FT = ' 6.25MHZ   '
     IF (F.EQ.5) FT = ' 3.125MHZ  '
     IF (F.EQ.6) FT = ' 1.56MHZ   '
     IF (F.EQ.7) FT = ' 781KHZ    '
     IF (G.EQ.0) GT = ' 16K       '
     IF (G.EQ.1) GT = ' 32K       '
     IF (G.EQ.2) GT = ' 64K       '
     IF (G.EQ.3) GT = ' 128K      '
     IF (G.EQ.4) GT = ' 256K      '
     IF (G.EQ.5) GT = ' 512K      '
     IF (G.EQ.6) GT = ' 1M        '
     IF (G.EQ.7) GT = ' 2M        '
     IF (I.EQ.0) IT = ' EXTERNAL  '
     IF (I.EQ.1) IT = ' INTERNAL  '
     IF (J.EQ.0) JT = ' AC        '
     IF (J.EQ.1) JT = ' DC        '
     IF (K.EQ.0) KT = ' NEGATIVE  '
     IF (K.EQ.1) KT = ' POSITIVE  '
     WRITE(*,230)AT,BT,CT,D,E,FT,GT,H,IT,JT,KT,L
230  FORMAT(/,'  AMPLIFIER TYPE    - ',A11,/
    1' ACQUISITION MODE  - ',A11,/
    2' VOLTAGE RANGE     - ',A11,/
    3' ADC OFFSET        - ',I4,/
    4' THRESHOLD         - ',I4,/
    5' CLOCK DIVIDER     - ',A11,/
    6' ACTIVE MEMORY     - ',A11,/
    7' TIMEOUT CYCLES    - ',I3,/
    8' TRIGGER SOURCE    - ',A11,/
    9' TRIGER COUPLING   - ',A11,/
    1' TRIGGER SLOPE     - ',A11,/
    2' #DATA POINTS      - ',I6,/)
     WRITE(*,*)'DO YOU WISH TO CHANGE ANY OF THE PARAMETERS? Y OR N'
     READ(*,240)P
```

PROGRAM RED2INTT.FOR

```
240  FORMAT(A1)
     IF(P.EQ.'Y '.OR. P.EQ.'Y ')CALL M(A,B,C,D,E,F,G,H,I,J,K,L)
570  FORMAT(I6)
     CONFIG(1) = A
     CONFIG(2) = B
     CONFIG(3) = C
     CONFIG(4) = D
     CONFIG(5) = E
     CONFIG(6) = F
     CONFIG(7) = G
     CONFIG(8) = H
     CONFIG(9) = I
     CONFIG(10) = J
     CONFIG(11) = K
     NDATA = L
     N = L
     REWIND 1
     WRITE(1,250)A,B,C,D,E,F,G,H,I,J,K,L
250  FORMAT(I1,/,I1,/,I1,/,I3,/,I3,/,I1,/,I1,/,I1,/,I1,/,I1,/,I1,/,
    $I4)
     CLOSE(UNIT=1)
     IF (A.EQ.0) AT = 'LINEAR    '
     IF (A.EQ.1) AT = 'COMPRESSION'
     IF (B.EQ.0) BT = 'BURST     '
     IF (B.EQ.1) BT = 'PRETRIGGER'
     IF (B.EQ.2) BT = 'CONTINUOUS'
     IF (B.EQ.3) BT = 'SEGMENTED '
     IF (C.EQ.0) CT = '60MV      '
     IF (C.EQ.1) CT = '200MV     '
     IF (C.EQ.2) CT = '600MV     '
     IF (C.EQ.3) CT = '2V        '
     IF (F.EQ.0) FT = '100MHZ    '
     IF (F.EQ.1) FT = '50MHZ     '
     IF (F.EQ.2) FT = '25MHZ     '
     IF (F.EQ.3) FT = '12.5MHZ   '
     IF (F.EQ.4) FT = '6.25MHZ   '
     IF (F.EQ.5) FT = '3.125MHZ  '
     IF (F.EQ.6) FT = '1.56MHZ   '
     IF (F.EQ.7) FT = '781KHZ    '
     IF (G.EQ.0) GT = '16K       '
     IF (G.EQ.1) GT = '32K       '
     IF (G.EQ.2) GT = '64K       '
     IF (G.EQ.3) GT = '128K      '
     IF (G.EQ.4) GT = '256K      '
     IF (G.EQ.5) GT = '512K      '
     IF (G.EQ.6) GT = '1M        '
     IF (G.EQ.7) GT = '2M        '
     IF (I.EQ.0) IT = 'EXTERNAL  '
     IF (I.EQ.1) IT = 'INTERNAL  '
     IF (J.EQ.0) JT = 'AC        '
```

```
      IF (J.EQ.1) JT = ' DC       '
      IF (K.EQ.0) KT = ' NEGATIVE  '
      IF (K.EQ.1) KT = ' POSITIVE  '
      WRITE(*,230)AT,BT,CT,D,E,FT,GT,H,IT,JT,KT,L
C **** INITIALIZE DATA ARRAYS *****
      DO I=1,NDATA
           DATA_HOLD1(I)=0
           DATA_HOLD2(I)=0
      ENDDO
      IA=10
      IS=0
      IF ((G.EQ.0).OR.(G.EQ.1)) THEN
        MAXSEG=0
      ELSE
        MAXSEG = 2**(G-1)-1
      ENDIF
      IF (G.GT.0) THEN
        ISEG = 32767-L
      ELSE
        ISEG = 16384-L
      ENDIF
C **** INITIALIZE A/D CARDS *****
      CALL SETUP(CONFIG)
C ****   ACQUIRE DATA ******
      WRITE(*,13)
13    FORMAT('  DATA ACQUISITION FINISHED')
C **** CARD MEMORY IS FULL, READ INDIVIDUAL PARTICLES  ****
      IPARTICLES=1
      CALL GETDAT(IA,DATA_HOLD1,DATA_HOLD2)
C *** SET THE INTERPOLATION FACTOR OF 2,4 OR 8 ****
      WRITE(*,'("       IS THE INTERPOLATION FACTOR 2,4, OR 8",)')
      READ(*,'(I2)')LI
C **** LOCATING DATA WITH NO 255 COUNT SPIKES *****
800   DO 900 I=1,NDATA
           IF(DATA_HOLD1(I).LT.0)THEN
              X(I)=DATA_HOLD1(I)+256.0
           ELSE
              X(I)=DATA_HOLD1(I)
           ENDIF
           IF(DATA_HOLD2(I).LT.0)THEN
              Y(I)=DATA_HOLD2(I)+256.0
           ELSE
              Y(I)=DATA_HOLD2(I)
       ENDIF
      IF ((X(I).EQ.255.0).OR.(Y(I).EQ.255.0))THEN
        IPARTICLES = IPARTICLES-1
        GOTO 330
      ENDIF
900   CONTINUE
C
```

PROGRAM RED2INTT.FOR

```
C   THE FOLLOWING DO LOOP ZEROS HALF OF ONE SIGNAL.  THE PURPOSE OF
C   THIS IS SO THE CROSS CORRELATION FUNCTION WILL HAVE NO END EFFECTS.
C   IN THE TIME DOMAIN, THIS WOULD BE THE SAME AS SLIDING ONLY HALF
C   OF THE Y SIGNAL ACROSS THE X SIGNAL THEREBY EVERY CORRELATION
C   CALCULATION IS CALCULATED USING THE SAME NUMBER OF POINTS.
C
815  CONTINUE
1000     FORMAT(I3.3)
C **** BEGINNING OF THE INTERPOLATION PROCESS ****
     N=2048
     DO I=1,N
          XIN(I)=0.0
          YIN(I)=0.0
     ENDDO
     XIN(1)=X(1)
     YIN(1)=Y(1)
C **** INSERT THE INTERPOLATION FACTOR OF ZEROS BETWEEN DATA POINTS ****
     DO I=2,N-1,LI
          J=I/LI
          XIN(I+1)=X(J)
          YIN(I+1)=Y(J)
     ENDDO
C **** LOWPASS FILTER CHOSEN CORRESPONDING TO THE INTERPOLATION FACTOR ***
     SELECT CASE(LI)
     CASE(2)
          OPEN(UNIT=6,FILE='B642.DAT')
     CASE(4)
          OPEN(UNIT=6,FILE='B64EX.DAT')
     CASE(8)
          OPEN(UNIT=6,FILE='B648.DAT')
     END SELECT
     DO I=1,N
       HR(I)=0.0
     ENDDO
     DO I=1,65
          READ(6,*)HR(I)
     ENDDO
     DO I=1,N
          HI(I)=0.0
          YINI(I)=0.0
          XINI(I)=0.0
     ENDDO
C **** FFT TAKEN OF DATA AND FILTER ****
     CALL INITAB(N,WR,WI)
     CALL FFTSUB(N,XIN,XINI,WR,WI)
     CALL FFTSUB(N,YIN,YINI,WR,WI)
     CALL FFTSUB(N,HR,HI,WR,WI)
C **** FILTER IS MULTIPLIED BY THE DATA SIGNALS ****
     DO I=1,N
          XINC(I)=CMPLX(XIN(I),XINI(I))
```

```
          YINC(I)=CMPLX(YIN(I),YINI(I))
          HC(I)=CMPLX(HR(I),HI(I))
          XINC(I)=HC(I)*XINC(I)
          YINC(I)=HC(I)*YINC(I)
          XINI(I)=IMAG(XINC(I))
          XIN(I)=REAL(XINC(I))
          YINI(I)=IMAG(YINC(I))
          YIN(I)=REAL(YINC(I))
      ENDDO
C **** INVERSE FFT TAKEN OF DATA SIGNALS ****
      CALL FFTSUB(N,XIN,XINI,WI,WR)
      CALL FFTSUB(N,YIN,YINI,WI,WR)
      DO I=1,N
          XINC(I)=CMPLX(XIN(I),XINI(I))
          YINC(I)=CMPLX(YIN(I),YINI(I))
      ENDDO
C **** DATA IS SHIFTED ****
      DO I=1,(N/2)
          J=I+N/4
          XINI(I)=IMAG(XINC(J))
          XIN(I)=REAL(XINC(J))
          YINI(I)=IMAG(YINC(J))
          YIN(I)=REAL(YINC(J))
          XINC(I)=CMPLX(XIN(I),XINI(I))
          YINC(I)=CMPLX(YIN(I),YINI(I))
      ENDDO
C     DO I=((N)/4),((N)/2)
C        YIN(I)=0.0
C        YINI(I)=0.0
C     ENDDO
      DO I=1,((N)/2)
          YINC(I)=CMPLX(YIN(I),YINI(I))
      ENDDO
      DO I=1,N
          XINI(I)=0.0
          YINI(I)=0.0
       XIN(I)=CABS(XINC(I))/N
       YIN(I)=CABS(YINC(I))/N
      ENDDO
C **** COPYING THE FIRST 80 A AND Y FILES TO THE VIRTUAL DISK *****
      IF(IPARTICLES.GT.80)GOTO 1015
       STRINGX(1:4)='E:\X'
       STRINGY(1:4)='E:\Y'
       STRINGX(5:5)='.'
       STRINGY(5:5)='.'
       WRITE(EXTEN,1000)IPARTICLES
       STRINGX(6:8)=EXTEN(1:3)
       STRINGY(6:8)=EXTEN(1:3)
       OPEN(UNIT=6,FILE=STRINGX(1:8),FORM='UNFORMATTED')
       WRITE(6)INT2(NDATA),(FLOAT(I),XIN(I),I=1,NDATA)
```

```
      CLOSE(UNIT=6)
      OPEN(UNIT=6,FILE=STRINGY(1:8),FORM='UNFORMATTED')
      WRITE(6)INT2(NDATA),(FLOAT(I),YIN(I),I=1,NDATA)
      CLOSE(UNIT=6)

1015     IMAX = 1024
       DO 140 I=1,IMAX
         J=I+IMAX
         YIN(J)=0.0
140    CONTINUE
C **** THE FFT IS TAKEN OF BOTH SIGNALS ****
     N=1024
     CALL INITAB(N,WR,WI)
     CALL FFTSUB(N,XIN,XINI,WR,WI)
     CALL FFTSUB(N,YIN,YINI,WR,WI)
C **** CONJUGATE OF Y IS MULTILPLIED BY X *****
     DO 150 I=1,1024
          XF(I)=CMPLX(XIN(I),XINI(I))
          YF(I)=CMPLX(YIN(I),YINI(I))
          RXYF(I)=CONJG(YF(I))*(XF(I))
          RXYR(I)=REAL(RXYF(I))
          RXYI(I)=IMAG(RXYF(I))
150  CONTINUE
C **** INVERSE FFT OF THE CROSSCORRELATION IS TAKEN ****
     CALL FFTSUB(N,RXYR,RXYI,WI,WR)
     DO 145 I=1,1024
          RXY(I)=CMPLX(RXYR(I),RXYI(I))/N
145  CONTINUE
C **** THE FIRST PEAK OF THE CROSS CORRELATION FUNCTION
C **** IS THE NUMBER OF SAMPLES LAG BETWEEN THE TWO SIGNALS. THE NUMBER
C **** OF SAMPLES BETWEEN TWO PEAKS IS THE PERIOD OF THE SIGNAL.  THE
C **** TIME PERIOD IS CALCULATED #SAMPLES/CLOCK FREQUENCY. THE PHASE IN
C **** DEGREES IS CALCULTED 360*#SAMPLES LAG/#SAMPLES BETWEEN PEAKS.
     J=1
     DO 1290 I=1,N-1
          ML=I+1
          IF(CABS(RXY(ML)).GT.CABS(RXY(ML-1)))GOTO 280
          J=J+1
1290  CONTINUE
280   DO 1200 I=1,N-J
          K=J+I
          IF(CABS(RXY(K)).LT.CABS(RXY(K-1)))GOTO 231
          JSAMPLE=K-1
1200  CONTINUE
231   DO 1210 I=1,N-JSAMPLE
          J=JSAMPLE+5+I
          IF(CABS(RXY(J)).GT.CABS(RXY(J-1)))GOTO 241
          KPERIODD=J-1
1210  CONTINUE
241   DO 1260 I=1,N-KPERIODD
```

PROGRAM RED2INTT.FOR

```
          J=KPERIODD+1+I
          IF(CABS(RXY(J)).LT.CABS(RXY(J-1)))GOTO 270
          KSAMPLE=J
1260  CONTINUE
270   KPERIOD=KSAMPLE-(JSAMPLE+1)
      IF(KPERIOD.EQ.0)GOTO 31
      PHASE=360.0*JSAMPLE/KPERIOD
       TPERIOD=(1.0/(100000000.0/2.0**F)*KPERIOD*1000000.0)/LI

C  **** REJECT PARTICLES OUT OF MEASUREMENT RANGE ****
31        IF ((KPERIOD.LT.(40.0)).OR.
     &     (ABS(PHASE).GT.360.0).OR.(KPERIOD.GT.200))THEN
          IPARTICLES = IPARTICLES - 1
          GOTO 330
        ENDIF

        WRITE(*,311)IPARTICLES
        WRITE(*,300)PHASE
        WRITE(*,310)TPERIOD
300   FORMAT(' X LAGS Y BY ',1PG10.3,1X,'DEGREES. THE PERIOD OF THE ')
310     FORMAT(' SIGNAL IS ',F8.2,1X,'MICROSECONDS.')
311   FORMAT(5X,' PARTICLE # ',I5)
C
C     THE PHASE AND TIME PERIOD OF THE SIGNAL ARE WRITTEN TO FILE
C     D:\SHARON\PHASE.DAT.
C
        WRITE(3,*)IPARTICLES,PHASE
        WRITE(4,*)IPARTICLES,TPERIOD
320     FORMAT(5X,F8.2)


C **** CHECK IF ALL PARTICLES HAVE BEEN READ ****
330   IF ((IS.GT.MAXSEG).OR.
     &   ((IS.EQ.MAXSEG).AND.(IA.GT.ISEG))) GOTO 820
      IF (IA.GT.ISEG) THEN
        IA = 1
        IS = IS+1
      ELSE
        IA = IA+1
      ENDIF

340   CALL SEARCH(IS,IA,MAXSEG,ISEG,IERR)
      CALL SETSEG(IS)
      IF((IS.EQ.MAXSEG).AND.(IA.GT.ISEG)) GOTO 820
      IF(IERR.NE.0)GOTO 820
      CALL GETDAT(IA,DATA_HOLD1,DATA_HOLD2)

      DO WHILE((DATA_HOLD1(1).EQ.-1).AND.(IA.LE.ISEG))
        IA=IA+1
        CALL GETDAT(IA,DATA_HOLD1,DATA_HOLD2)
```

PROGRAM RED2INTT.FOR

```
      ENDDO
      IF ((IS.LT.MAXSEG).AND.(IA.GT.ISEG)) THEN
        IS = IS+1
        IA = 1
        GOTO 340
      ENDIF
      IF ((IS.EQ.MAXSEG).AND.(IA.GT.ISEG)) GOTO 820

      IPARTICLES = IPARTICLES+1
      GOTO 800
C **** ALL PARTICLES DONE ****
820   WRITE (*,821) IPARTICLES
821   FORMAT (10X,I5,' PARTICLES ACQUIRED')
        DO  I=1,1024
          WRITE(9,*)I,CABS(RXY(I))
        ENDDO

      PMEAN=0.0
      REWIND 3
      REWIND 7
      DO I=1,IPARTICLES
        READ(3,*)J,PHSE(I)
        PMEAN=PMEAN+PHSE(I)
      ENDDO
        WRITE(*,850)PMEAN/IPARTICLES
850     FORMAT(' THE MEAN OF THE PHASES IS',F8.2)
      CLOSE(3)
      CLOSE (4)
      CLOSE (9)
      END

      SUBROUTINE M(A,B,C,D,E,F,G,H,I,J,K,L)
      INTEGER A,B,C,D,E,F,G,H,I,J,K,L
      CHARACTER S


1     WRITE(*,10)
10    FORMAT(/,' A - AMPLIFIER TYPE',/
     1' B - ACQUISITION MODE',/
     2' C - VOLTAGE RANGE',/
     3' D - ADC OFFSET',/
     4' E - THRESHOLD',/
     5' F - CLOCK DIVIDER',/
     6' G - ACTIVE MEMORY',/
     7' H - TIMEOUT CYCLES',/
     8' I - TRIGGER SOURCE',/
     9' J - TRIGGER COUPLING',/
     1' K - TRIGGER SLOPE',/
     2' L - #DATA POINTS',/
     3' X - EXIT           - ',$)
```

PROGRAM RED2INTT.FOR

```
20   READ(*,30)S
30   FORMAT(A1)
     IF (S.EQ.'A' .OR. S.EQ.'A') THEN
       WRITE (*,101)
101    FORMAT(' 0 = LINEAR, 1 = COMPRESSED')
       GOTO 100
     ELSE IF (S.EQ.'B' .OR. S.EQ.'B') THEN
       WRITE(*,201)
201    FORMAT(' 0 = BURST, 1= PRETRIGGER, 2 = CONTINUOUS,'
    +' 3 = SEGMENTED')
       GOTO 200
     ELSE IF (S.EQ.'C' .OR. S.EQ.'C') THEN
       WRITE(*,301)
301    FORMAT(' 0 = 60MV, 1 = 200MV, 2 = 600MV, 3 = 2V')
       GOTO 300
     ELSE IF (S.EQ.'D'.OR. S.EQ.'D') THEN
       WRITE(*,401)
401    FORMAT(' 0 TO 255')
       GOTO 400
     ELSE IF (S.EQ.'E' .OR. S.EQ.'E') THEN
       WRITE(*,501)
501    FORMAT(' 0 TO 255')
       GOTO 500
     ELSE IF (S.EQ.'F' .OR. S.EQ.'F') THEN
       WRITE(*,601)
601    FORMAT(' 0 = 100M, 1 = 50M, 2 = 25M, 3 = 12.5M, 4 = 6.25M,'
    +' 5 = 3.125M, 6 = 1.5625M,    7 = 781K')
       GOTO 600
     ELSE IF (S.EQ.'G' .OR. S.EQ.'G') THEN
       WRITE(*,701)
701    FORMAT(' 0 = 16K, 1 = 32K, 2 = 64K, 3 = 128K, 4 = 256K,'
    +' 5 = 512K, 6 = 1M, 7 = 2M')
       GOTO 700
     ELSE IF (S.EQ.'H' .OR. S.EQ.'H') THEN
       WRITE(*,801)
801    FORMAT(' 0 TO 31 ODD VALUES ONLY')
       GOTO 800
     ELSE IF (S.EQ.'T' .OR. S.EQ.'T') THEN
       WRITE(*,901)
901    FORMAT(' 0 = EXTERNAL, 1 = INTERNAL')
       GOTO 900
     ELSE IF (S.EQ.'J'.OR. S.EQ.'J') THEN
       WRITE(*,1001)
1001   FORMAT(' 0 = AC, 1 = DC')
       GOTO 1000
     ELSE IF (S.EQ.'K' .OR. S.EQ.'K') THEN
       WRITE(*,1101)
1101   FORMAT(' 0 = NEGATIVE, 1 = POSITIVE')
       GOTO 1100
     ELSE IF (S.EQ.'L' .OR. S.EQ.'L') THEN
```

PROGRAM RED2INTT.FOR

```
      GOTO 1200
    ELSE IF (S.EQ.'X' .OR. S.EQ.'X')THEN
      GOTO 2000
    ELSE
      GOTO 1
    ENDIF
100 WRITE(*,*)' A'
    CALL EDIT(A)
    GOTO 1
200 WRITE(*,*)' B'
    CALL EDIT(B)
    GOTO 1
300 WRITE(*,*)' C'
    CALL EDIT(C)
    GOTO 1
400 WRITE(*,*)' D'
    CALL EDIT(D)
    GOTO 1
500 WRITE(*,*)' E'
    CALL EDIT(E)
    GOTO 1
600 WRITE(*,*)' F'
    CALL EDIT(F)
    GOTO 1
700 WRITE(*,*)' G'
    CALL EDIT(G)
    GOTO 1
800 WRITE(*,*)' H'
    CALL EDIT(H)
    GOTO 1
900 WRITE(*,*)' I'
    CALL EDIT(I)
    GOTO 1
1000 WRITE(*,*)' J'
    CALL EDIT(J)
    GOTO 1
1100 WRITE(*,*)' K'
    CALL EDIT(K)
    GOTO 1
1200 WRITE(*,*)' L'
    CALL EDIT(L)
    GOTO 1
2000 RETURN
    END

    SUBROUTINE EDIT(Y)
    INTEGER Y
    WRITE(*,*)'ENTER NEW VALUE'
    READ(*,*)Y
    WRITE(*,*)Y
```

```
      RETURN
      END

C COOLY-TUKEY RADIX-2, DIF FFT PROGRAM.
C COMPLEX INPUT DATA IN ARRAYS XR AND XI.
C TABLE LOOK-UP OF W VALUES.
      SUBROUTINE FFTSUB (NPTS,XR,XI,WR,WI)
      DIMENSION XR(NPTS),XI(NPTS),WR(NPTS),WI(NPTS)
      M = NINT( ALOG(FLOAT(NPTS)) / ALOG(2.0) )
      N2 = NPTS
      DO 10 K=1, M
        N1 = N2
        N2 = N2 / 2
        IE = NPTS / N1
        IA = 1
        DO 20 J=1, N2
          C = WR(IA)
          S = WI(IA)
          IA = IA + IE
          DO 30 I=J,NPTS, N1
            L = I + N2
            XRT  = XR(I) - XR(L)
            XR(I) = XR(I) + XR(L)
            XIT  = XI(I) - XI(L)
            XI(I) = XI(I) + XI(L)
            XR(L) = C*XRT + S*XIT
            XI(L) = C*XIT - S*XRT
   30     CONTINUE
   20   CONTINUE
   10 CONTINUE
      CALL REVERS(NPTS,XR,XI)
      RETURN
      END

      SUBROUTINE REVERS (NPTS,XR,XI)
      DIMENSION XR(NPTS),XI(NPTS)
      J = 1
      N1 = NPTS - 1
      DO 70 I=1, N1
        IF (I.GE.J) GO TO 40
        XRT  = XR(J)
        XR(J) = XR(I)
        XR(I) = XRT
        XIT  = XI(J)
        XI(J) = XI(I)
        XI(I) = XIT
   40   K = NPTS / 2
   50   IF (K.GE.J) GO TO 60
        J = J - K
        K = K / 2
```

```
      GO TO 50
60    J = J + K
70  CONTINUE
    RETURN
    END

C INITIALIZE SINE AND COSINE TABLES FOR FFT'S WITH TABLE LOOKUP.

    SUBROUTINE INITAB (NPTS,WR,WI)
    DIMENSION WR(NPTS),WI(NPTS)
    P = 2.0 * ACOS(-1.0) / NPTS
    DO 10 N=1, NPTS
      A = (N-1) * P
      WR(N) =  COS(A)
      WI(N) = -SIN(A)
  10 CONTINUE
    RETURN
    END
```

APPENDIX B

```
                /* NOTE:  This file must be linked with and cbuild.lib.  Set project
  file to del4.prj to accomplish this. */

#include <d:\c600\include\stdlib.h>
#include <d:\c600\include\stdio.h>
#include <d:\c600\include\conio.h>
#include <d:\c600\include\cbuild.h>
#include <d:\c600\include\dos.h>


void search(int far *is,int far *ia,int far *maxseg,
            int far *iseg,int far *ierr)

{

 /* Find beginning of next data point */
         *ierr=findmark(Brd1,is,ia,*maxseg,*iseg);
/*       printf("is = %d ia = %d",*is,*ia); */

}
```

```
/* NOTE: This file must be linked with and cbuild.lib. Set project
  file to del4.prj to accomplish this. */

#include <d:\c600\include\stdlib.h>
#include <d:\c600\include\stdio.h>
#include <d:\c600\include\conio.h>
#include <d:\c600\include\cbuild.h>
#include <d:\c600\include\dos.h>


void setseg(int far *is)

{
     unsigned char s;

    s=(char)*is;

 /* set correct segment address on board */
        c_block(Brd1,s);
        c_block(Brd2,s);
}
```

```
/* NOTE: This file must be linked with and cbuild.lib.  Set project
  file to del4.prj to accomplish this. */

#include <d:\c600\include\stdlib.h>
#include <d:\c600\include\stdio.h>
#include <d:\c600\include\conio.h>
#include <d:\c600\include\cbuild.h>
#include <d:\c600\include\dos.h>


void getdat(int far *ia,unsigned char far *data_hold1,
        unsigned char far *data_hold2)

{
     int n;
     n = 1024;

    getsram(Brd1,*ia,data_hold1,n);
    getsram(Brd2,*ia,data_hold2,n);
/*   printf("ia = %d",*ia);          */

}
```

```
/* NOTE: This file must be linked with and cbuild.lib. Set project
  file to del4.prj to accomplish this. */

#include <d:\c600\include\stdlib.h>
#include <d:\c600\include\stdio.h>
#include <d:\c600\include\conio.h>
#include <d:\c600\include\cbuild.h>
#include <d:\c600\include\dos.h>

void setup(unsigned char far *config)

{
        int j, n, ierr;
        unsigned char i;
      n = 1024;
/* First download DASPOS by calling download with NULL filename */
        ierr = download(Brd1,"");
        ierr = download(Brd2,"");


/* set up DASP boards */
        wrtboard(Brd1,config);
      wrtboard(Brd2,config);

/* prepare board to acquire data: */
/* This code is cribbed from cbuild manual, page 15 */

      c_mode(Brd2,AprMode);
      poke(dasp_seg,t_command,1);
      c_mode(Brd2,OffMode);

      c_mode(Brd1,AprMode);
      poke(dasp_seg,t_command,1);
      c_mode(Brd1,OffMode);

/* acquire data: */

      c_mode(Brd2,AcqMode);
      c_mode(Brd1,AcqMode);

     printf("\nAcquiring...\n\n");
```

```
/* Take data until signal ram is full */
        do { } while((c_flag(Brd1)&0x20)==0);

      c_mode(Brd1,AprMode);
      poke(dasp_seg,t_command,0);
      c_mode(Brd1,SpMode);
         c_mode(Brd1,OffMode);

      c_mode(Brd2,AprMode);
      poke(dasp_seg,t_command,0);
      c_mode(Brd2,SpMode);
         c_mode(Brd2,OffMode);

/* select block 0 */
         c_block(Brd1,0);
      c_block(Brd2,0);
 }
```

Vita

Sharon Neufeldt Carter was born in Chatham, Ontario, Canada in 1964. She moved to East Lansing, Michigan shortly thereafter. Her family moved to Cookeville, Tennessee after her father completed his Ph.D. at Michigan State University in 1970. She received her early education in Cookeville and graduated from Cookeville High School in 1982. She entered Tennessee Technological University in the fall of 1982 and received her citizenship while she was there. She graduated with a Bachelor of Science degree in Electrical Engineering in 1986.

After graduation, she accepted a position with Calspan, Inc, AEDC group where she is currently employed. She is now an employee of Microcraft Technology because they bought the AEDC division of Calspan in 1994 where she is an instrumentation engineer in the High Temperature Laboratory in the ISF Facility. She was put on a part-time status after the birth of her second child. She entered the GRA program at UTSI in the fall of 1994 and received her Master of Science in Electrical Engineering in May 1995.

She is married to Joseph Steven Carter. They have two children, Kelly and Ryan and are living in Tullahoma, Tennessee.