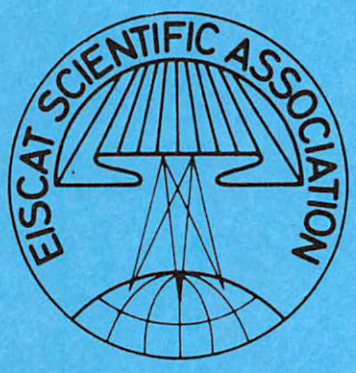
83/37



EISCAT
TECHNICAL
NOTE

THE LAG PROFILE ROUTINE and THE UNIVERSAL PROGRAM for THE EISCAT DIGITAL CORRELATORS

by
Terrance Ho, Tauno Turunen
Johan Silén, Markku Lehtinen

KIRUNA
Sweden

THE LAG PROFILE ROUTINE AND THE UNIVERSAL PROGRAM FOR THE EISCAT DIGITAL CORRELATORS

by

TERRANCE HO*, TAUNO TURUNEN**,
JOHAN SILEN** and MARKKU LEHTINEN**

*MAX PLANCK INSTITUTE,
3411 KATLENBURG-LINDAU,
W. GERMANY

**EISCAT,
GEOPHYSICAL OBSERVATORY,
SF-99600 SODANKYLA,
FINLAND.

EISCAT Technical Note 83/37
Printed in Sweden
EISCAT Scientific Association
Kiruna, February 1983
ISSN 0349-2710

TABLE OF CONTENTS

		PAGE
I.	INTRODUCTION	1
II.	THE PRINCIPLE OF OPERATION	2
III.	BASIC DEFINITIONS	3
IV.	BASIC FORMULAE	4
V.	COMPARISONS WITH THE LAG PROFILE ROUTINE	9
VI.	DESCRIPTION OF THE LAG PROFILE SUBROUTINE	13
VII.	DESCRIPTION OF THE UNIVERSAL PROGRAM	16
VIII.	HOW TO USE THE UNIVERSAL PROGRAM	21
IX.	AN EXAMPLE	25
X.	CONCLUSIONS	32
XI.	REFERENCES	34
APPENDIX A.	CORRELATOR DESCRIPTION AND LISTINGS OF THE LAG PROFILE SUBROUTINE AND THE UNIVERSAL PROGRAM	34
APPENDIX B.	PULSE TO PULSE CORRELATION	53

I. INTRODUCTION

In this report two new correlator algorithms are presented for the EISCAT radar system. The first one is a subroutine called the Lag Profile Routine and the second is a general program, which uses a slightly modified form of the Lag Profile Routine, called the Universal Program. This name was selected because it can be shown that this program can handle virtually any practical incoherent scatter modulation scheme which is possible in the EISCAT radar system.

The Universal Program was selected from three possible candidates on the basis that one could find a flexible correlator input data register definition that would suit all test modulation patterns which were tried. Altogether some tens of possible modulation patterns were tested and in some cases real monostatic receiving simulation was carried out in the EISCAT system. A typical test pattern consisted of six frequency channels having different modulations, such as a long pulse channel, 2-4 pulse coded channels and 1-3 power profile channels using a short pulse. Pulse codes having 3-6 pulses in the modulation pattern have been used. The channels may or may not be Barker coded. The Universal Program can also handle a combination of long pulse, pulse codes and short pulses with each pulse scheme individually Barker coded or not. However, it is not possible to have this mixture of Barker coding and no Barker coding in the EISCAT system because of hardware limitations.

The Universal Program performs well in the multi channel pulse to pulse correlation experiments appearing, for example in mesospeheric measurements, and it can also be used to calculate all the elements needed in cross-spectral and coherence analysis for a limited amount of data.

In the following the Lag Profile Routine and the Universal Program are handled in detail and some comparisons are made with the earlier well established algorithms used in EISCAT and other incoherent scatter stations. It is shown that the Universal Program can do all the tasks done by several other programs. Finally, a relatively simple example is handled in detail, in order to show how the Universal Program can be

used in a multi channel problem having three different modulation patterns.

In the EISCAT computer system the Universal Program has the file name: UNI-PROG:DATA.

II. THE PRINCIPLE OF OPERATION

The Lag Profile Routine and the Universal Program do not calculate the autocorrelation functions. They are used to calculate the expectation values of the complex cross products which are needed in the formation of the autocorrelation function. The parameters loaded into the correlator define the cross products which have to be calculated for a given modulation but they do not define anything which could be called a "rangecell" and they do not contain any information on the exact pulse pattern of the pulse code. The parameters only define the maximum delay used in the cross product calculations and depending on the type of modulation definehow the cross products are formed.

In the post processing of the recorded data one has to define the "rangecells" and one is totally free to select the spatial weighting factors which are possible for a given modulation. In this process one also has to decode the pulse code, ie to take into account the modulation pattern in order to form an autocorrelation function estimate for a given spatial volume with either the maximum resolution allowed by the modulation or for some larger volume depending on the demands or the noise level of the data.

It can be shown that by using the Universal Program all the information which is in the data for a given modulation is used. This is not necessarily done in the classical algorithms, for example the traditional algorithms for the monostatic long pulse do not usually calculate all the cross products which contain information. On the other hand a few percent of the data points produced by the Universal Program are not relevant. This price has to be paid otherwise one has to define the exact modulation pattern instead of the modulation type and one then rapidly looses the possibility to handle several modulations simultaneously. Two other restrictions are that if the pulse code has the so called

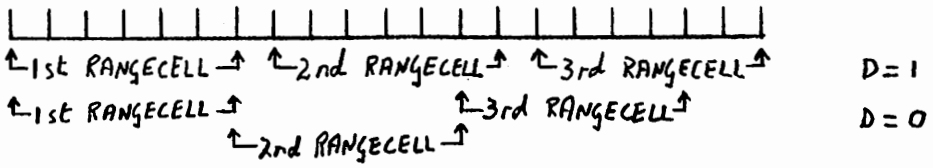
missing lags there is no way to avoid its computation and the program does not allow computation of certain cross products used for system balance checking as done in some other algorithms.

A relatively simple way to understand the operation of the Universal Program is to imagine a single data vector which is formed from all the available data being measured at several channels and arranged such that all the channels having a similar type of modulation are placed in succession. One can then define a matrix of dimension N×N, where N is the total number of data points, which contains all the possible cross products as its elements. The parameters of the Universal Program define which elements in the matrix are going to be calculated for a given set of modulations and for a given arrangement of the data vector. If the number of different types of modulations is limited to four and if one of them is a short pulse, which is used for the power profile estimation, then the Universal Program can do the job effectively with only a very limited amount of obsolete data points appearing in the result data vector, independent of the number of channels used in the experiment.

III. BASIC DEFINITIONS

In this report the following terminology is used:
For the Traditional Routines:

1) The No. of Samples in a Rangecell means the No. of Samples taken from a particular sacttering volume.
2) The No. of Rangecells for Time Average means the number of scattering volumes (ranges) used for a particular time average.
3) The No. of Lags in a Rangecell means that the number of Lags can be less than or equal to the number of Samples.
4) The Rangecell Increment (D) means that overlapping of Rangecells can be obtained. If D=1 there is no overlapping. If $D \leq 0$ overlapping is obtained (see diagram).



For the Lag Profile Routine:

1) The No. of Samples for Time Average means the total No. of Samples for a particular time average that forms one block of data containing one or more channels, which can be grouped together because of similar modulation properties, from which all meaningful cross products are to be computed.
2) The No. of Lag Profiles means the number of Lag Profiles to be calculated for the block of data as defined in 1).
3) The Lag Increment is the increment between samples which is used to form the Lag Profile(1).

IV. BASIC FORMULAE

Here the formulae for the Lag Profile, Power Profile, Single Pulse and Multi Pulse algorithms as used in the correlator are given. Only the Lag Profile algorithm will be explained in detail as a complete description of the other algorithms are given in detail in another report (see REF 1).

LAG PROFILE

Assume that we have a set of N complex samples $Z_0, Z_1, .., Z_{N-1}$ where $Z_0 = X_0 + jY_0$ etc, (ie starting at location 0 in the buffer memory) covering the total scattering volume (ie N is the total number of complex samples which cover all rangecells) for a particular time average.

Then:

$$LP(L) = Z_i Z^*_{i+(L\times LI)}$$

$$= (X_i X_{i+(L\times LI)} + Y_i Y_{i+(L\times LI)}) + j(X_{i+(L\times LI)} Y_i - X_i Y_{i+(L\times LI)})$$

where Z^* denotes the complex conjugate, $i=0,1,..,N-1-(L\times LI)$, $L=0,1,..,P$, $P \leq N-1$ and LI=Lag Increment.

If we consider $Z_0, Z_1, .., Z_{N-1}$ to be the set of points in the data vector then we can construct the following N×N matrix:

	Z_0	Z_1	Z_2	Z_3	Z_4	Z_5	Z_6	Z_{N-1}
Z_0	Z_0Z_0	Z_0Z_1	Z_0Z_2	Z_0Z_3	Z_0Z_4	Z_0Z_5	Z_0Z_6	Z_0Z_{N-1}
Z_1		Z_1Z_1	Z_1Z_2	Z_1Z_3	Z_1Z_4	Z_1Z_5	Z_1Z_6	Z_1Z_{N-1}
Z_2			Z_2Z_2	Z_2Z_3	Z_2Z_4	Z_2Z_5	Z_2Z_6	Z_2Z_{N-1}
Z_3				Z_3Z_3	Z_3Z_4	Z_3Z_5	Z_3Z_6	Z_3Z_{N-1}
Z_4					Z_4Z_4	Z_4Z_5	Z_4Z_6	Z_4Z_{N-1}
Z_5						Z_5Z_5	Z_5Z_6	Z_5Z_{N-1}
Z_6							Z_6Z_6	Z_6Z_{N-1}
Z_{N-1}								$Z_{N-1}Z_{N-1}$

In this case only the upper triangle of the matrix need be considered. This is because we are only considering the case of autocorrelation. However, the following discussion would also apply to the case of cross correlation, whereby the bottom half of the matrix would have to be taken into account. This half would represent the negative lags of the cross correlation function. For autocorrelation the negative lags can be formed from the positive lags as it is only necessary to negate the sign of the imaginary part. It should be also noted that the cross products in the matrix are written as Z_0Z_0 instead of $Z_0Z_0^*$ where * denotes the complex conjugate. However, for the purposes of explanation it is only necessary to consider which Z sample is being used.

In the simplest case the diagonals are computed in the following order and then stored sequentially in the result memory. First the Z_0Z_0 diagonal, then the Z_0Z_1 diagonal, then the Z_0Z_2 diagonal, etc. Each diagonal can be thought of as representing its corresponding Lag Profile, therefore, it is very easy to see which cross products belong to which Lag Profile. This can be verified by expanding the formula; by setting i=0,1,..,N-1-L, L=0,1,..,N-1, LI=1 we have:

$LP(0)=Z_0Z_0, Z_1Z_1, Z_2Z_2, Z_3Z_3, ..., Z_{N-1}Z_{N-1}$ i=0,1,..,N-1, L=0

$LP(1)=Z_0Z_1, Z_1Z_2, Z_2Z_3, \ldots, Z_{N-2}Z_{N-1}$ i=0,1,..,N-2, L=1

$LP(2)=Z_0Z_2, Z_1Z_3, Z_2Z_4, \ldots, Z_{N-3}Z_{N-1}$ i=0,1,..,N-3, L=2

.
.
.

$LP(N-1)=Z_0Z_{N-1}$ i=0, L=N-1

In almost all cases it is not practical to define P=N-1. The parameter P would depend on the type of pulse scheme used (ie Long Pulse or Pulse Code). For example, in a 5 pulse multi pulse scheme the Lag Profiles would go from 0-11. The so called missing lag (10) would also be calculated. This profile, of course, has no meaning and would have to be discarded later. It would probably be better to consider if Lag Profile(11) is worth calculating and define P=9 instead of 11. A general rule is that if missing lags are included in the range of P then they will also be calculated as Profiles.

There are many cases where we do not wish to compute all the diagonals in the matrix. For example, in the case of pulse codes and Barker coding. In these cases the Lag Increment, LI, plays an important role. For Barker coding with a 13 baud code we would set LI=13. The algorithm would then compute the following diagonals: for LP(0) the Z_0Z_0 diagonal, LP(1) the Z_0Z_{13} diagonal, LP(2) the Z_0Z_{26} diagonal, etc. A general rule to find which diagonal corresponds to its appropriate lag profile is: LP(L) has its corresponding diagonal starting with $Z_0Z_{L\times LI}$. This also applies for multi pulse schemes.

For multi pulse schemes there is one more point to consider and that is for a given Lag Profile, where in the diagonal is the first correlating cross product? In order to illustrate this, consider a simple 3 pulse code where Z_0, Z_2, Z_6 correspond to the first sample of each pulse and LI=2. Then LP(0) has the the Z_0Z_0 diagonal, LP(1) has the Z_0Z_2 diagonal, LP(2) has the Z_0Z_4 diagonal and LP(3) has the Z_0Z_6 diagonal. The LP(1) and LP(3) profiles will have their first correlating element starting at Z_0Z_2 and Z_0Z_6 respectively, however, LP(2) will have its first element starting at Z_2Z_6. The first two elements Z_0Z_4 and Z_1Z_5 are non correlating elements and can be considered

as rubbish. A general rule to find where the first correlating element in a diagonal lies is always to look for a product which is a combination of the first sample positions of each pulse. All the following elements of the diagonal are correlating elements for that profile and correspond to successive volumes which are to be analysed.

The Lag Profile algorithm is therefore able to handle power profiles, long pulses, multi pulses and Barker coded versions of each of them.

POWER PROFILE

Assume that we have a set of N complex samples $Z_0, Z_1, .., Z_{N-1}$ in a rangecell, where $Z_0 = X_0 + jY_0$, etc and that there are r=1,..,M rangecells. Then the zero lag estimate for the rth rangecell is defined as:

$$K_r = \sum_{i=0}^{N-1} Z_{i+(N+D-1)(r-1)} Z^*_{i+(N+D-1)(r-1)}$$

$$K_r = \sum_{i=0}^{N-1} X^2_{i+(N+D-1)(r-1)} + Y^2_{i+(N+D-1)(r-1)}$$

where * denotes the complex conjugate and D is the overlap factor between rangecells, such that for D=1 there is no overlapping and for $D \leq 0$ there is overlapping of rangecells.

SINGLE PULSE

Assume that we have a set of N complex samples $Z_0, Z_1, .., Z_{N-1}$ in a rangecell, where $Z_0 = X_0 + jY_0$, etc and that there are r=1,..,M rangecells.

Then:

$$K_{L,r} = \sum_{i=0}^{N-L-1} Z_{i+(N+D-1)(r-1)} Z^*_{i+L+(N+D-1)(r-1)}$$

$$= \sum_{i=0}^{N-L-1} X_{i+(N+D-1)(r-1)} X_{i+L+(N+D-1)(r-1)} +$$

$$Y_{i+(N+D-1)(r-1)} Y_{i+L+(N+D-1)(r-1)}$$

$$+ j \sum_{i=0}^{N-L-1} X_{i+L+(N+D-1)(r-1)} Y_{i+(N+D-1)(r-1)} -$$

$$X_{i+(N+D-1)(r-1)} Y_{i+L+(N+D-1)(r-1)}$$

where * denotes the complex conjugate and D is the overlap factor between rangecells, such that for D=1 there is no overlapping and for D≤0 there is overlapping of rangecells.

MULTI PULSE

Assume that we have a set of P complex samples $Z_0, Z_1, .., Z_{P-1}$ where $Z_0 = X_0 + jY_0$, etc, N pulses in the pulse group and r=1,..,M rangecells. The various lag products from the same height can be expressed as:

$$K_{L,r} = Z_{S+r-1} Z^*_{S+L+r-1}$$

$$= (X_{S+r-1} X_{S+L+r-1} + Y_{S+r-1} Y_{S+L+r-1}) +$$

$$j(X_{S+L+r-1} Y_{S+r-1} - X_{S+r-1} Y_{S+L+r-1})$$

where * denotes the complex conjugate.

Let J_0 = Position of 1st Pulse

J_1 = Sample difference between 1st and 2nd pulse

J_2 = Sample difference between 1st and 3rd pulse

.

.

J_{N-1} = Sample difference between 1st and Nth pulse

Then $S = J_0, J_1, J_2, \ldots, J_{N-2}$

And $L = J_1 - S, \ldots, J_{N-1} - S$ with the restriction L>0

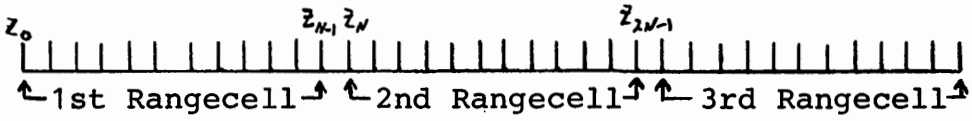
In this scheme the order of the lags computed are scrambled and would have to be unscrambled in the post processing program.

V. COMPARISONS WITH THE LAG PROFILE ROUTINE

Here a comparison between each traditional routine and the Lag Profile routine will be given. It will be shown that the Lag Profile routine produces the same results as each individual traditional routine.

POWER PROFILE

Looking at the schemes pictorially we have:



CONSIDER AS SAMPLES IN THE BUFFER MEMORY

Then for the Power Profile routine:

$$K_1 = Z_O^2+Z_1^2+Z_2^2+.....+Z_{N-1}^2$$

$$K_2 = Z_N^2+Z_{N+1}^2+.....+Z_{2N-1}^2$$

etc

Here the zero lag is accumulated for each rangecell and then written to the result memroy. Thus K_1 is written first to the result memory, K_2 second, etc.

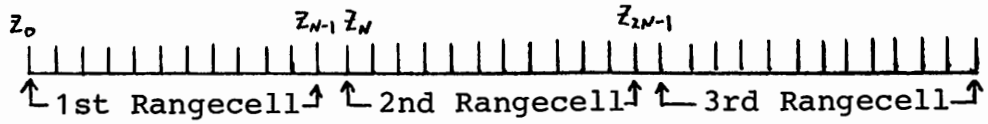
For the Lag Profile routine by setting:
i=O,1,...,M-1 where M=Total No. of Samples for all scattering volumes, L=O and LI=Anything, we have:

$$LP(O) = Z_O^2, Z_1^2, \ldots, Z_{N-1}^2, Z_N^2, Z_{N+1}^2, \ldots, Z_{M-2}^2, Z_{M-1}^2$$

Here the zero lag products are not accumulated but written separately in the result memory. The products can later be split into rangecells and accumulated accordingly.

SINGLE PULSE

Looking at the schemes pictorially we have:



CONSIDER AS SAMPLES IN THE BUFFER MEMORY

Then for the Single Pulse routine:

$$K_{O,1} = Z_O^2+Z_1^2+Z_2^2+.....+Z_{N-1}^2$$

$$K_{1,1} = Z_O Z_1+Z_1 Z_2+.....$$

.
.
.

$$K_{N-1,1} = Z_O Z_{N-1}$$

$$K_{O,2} = Z_O^2+Z_1^2+Z_2^2+.....+Z_{2N-1}^2$$

$$K_{1,2} = Z_N Z_{N+1}+Z_{N+1} Z_{N+2}+..$$

.
.
.

$$KN-1,2 = Z_N Z_{2N-1}$$

etc

Here the different lags are accumulated for each rangecell and then written sequentially in the result memory.

For the Lag Profile routine by setting:
i=O,1,..,M-1-L where M=Total No. of Samples for all scattering volumes, L=O,1,...,N-1 and LI=1, we have

$$LP(O) = Z_O^2, Z_1^2, ..., Z_{N-1}^2, Z_N^2, Z_{N+1}^2, ..., Z_{M-2}^2, Z_{M-1}^2$$

$$LP(1) = Z_O Z_1, Z_1 Z_2, .., Z_{N-1} Z_N, Z_N Z_{N+1}, ..., Z_{M-2} Z_{M-1}$$

$$LP(2) = Z_O Z_2, Z_1 Z_3, .., Z_{N-1} Z_{N+1}, Z_N Z_{N+2}, .., Z_{M-3} Z_{M-1}$$

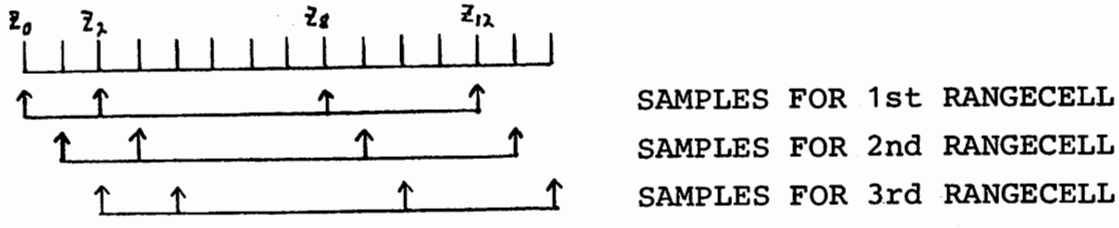
.
.

$$LP(N-1) = Z_O Z_{N-1}, Z_1 Z_N, Z_2 Z_{N+1}, ..., Z_{M-N} Z_{M-1}$$

Here each product is written sequentially in the result memory with the LP(0) products first, LP(1) cross products second, etc. Thus it is possible to split the data into range-cells and accumulate the products to form the appropriate lags with the necessary weighting (ie triangular or block). It can be seen that some of the data in the result memory will not be used, depending on which weighting function is used. In the case of triangular weighting, such products as $Z_{N-1}Z_N, Z_{2N-1}Z_N$, etc would be unused points and in the case of block weighting $Z^2_{M-N}, Z^2_{M-N+1}, Z^2_{M-N+2}, \ldots, Z^2_{M-1}$, etc would now be considered as unused points.

MULTI PULSE

Looking at the schemes pictorially we have:



Assume that we have four pulses with Z_0, Z_2, Z_8, Z_{12} corresponding to the first sample in each pulse and we are sampling for three rangecells.
Then for the Multi Pulse routine:

$K_{2,1} = Z_0 Z_2$

$K_{8,1} = Z_0 Z_8$

$K_{12,1} = Z_0 Z_{12}$

$K_{6,1} = Z_2 Z_8$

$K_{10,1} = Z_2 Z_{12}$

$K_{4,1} = Z_8 Z_{12}$

$K_{2,2} = Z_1 Z_3$

$K_{8,2} = Z_1 Z_9$

$K_{12,2} = Z_1 Z_{13}$

$K_{6,2} = Z_3 Z_9$

$K_{10,2} = Z_3 Z_{13}$

$K_{4,2} = Z_9 Z_{13}$

$K_{2,3} = Z_2 Z_4$

$K_{8,3} = Z_2 Z_{10}$

$K_{12,3} = Z_2 Z_{13}$

$K_{6,3} = Z_4 Z_{10}$

$K_{10,3} = Z_4 Z_{14}$

$K_{4,3} = Z_{10} Z_{14}$

Here it can be seen that the lags are scrambled and would have to be unscrambled in the post processing program. Note that $K_{2,1}$, $K_{8,1}$, $K_{12,1}$, $K_{6,1}$, $K_{10,1}$, $K_{4,1}$ are actually equal to lags $K_{1,1}$, $K_{4,1}$, $K_{6,1}$, $K_{3,1}$, $K_{5,1}$, $K_{2,1}$.

For the Lag Profile routine by setting:
i=0,1,...,14-(2×L), L=0,1,...,6 and LI=2, we have:

LP(0) = $Z_0^2, Z_1^2, Z_2^2, Z_3^2, Z_4^2, \ldots\ldots\ldots\ldots\ldots\ldots, Z_{14}^2$

LP(1) = $Z_0 Z_2, Z_1 Z_3, Z_2 Z_4, \ldots\ldots\ldots\ldots\ldots, Z_{12} Z_{14}$

LP(2) = $Z_0 Z_4, \ldots\ldots\ldots\ldots\ldots, Z_8 Z_{12}, Z_9 Z_{13}, Z_{10} Z_{14}$

LP(3) = $Z_0 Z_6, .., Z_2 Z_8, Z_3 Z_9, Z_4 Z_{10}, \ldots\ldots\ldots\ldots, Z_8 Z_{14}$

LP(4) = $Z_0 Z_8, Z_1 Z_9, Z_2 Z_{10}, \ldots\ldots\ldots\ldots\ldots, Z_6 Z_{14}$

LP(5) = $Z_0 Z_{10}, .., Z_2 Z_{12}, Z_3 Z_{13}, Z_4 Z_{14}$

LP(6) = $Z_0 Z_{12}, Z_1 Z_{13}, Z_2 Z_{14}$

LP(1) has its first correlating element in the $Z_0 Z_2$ diagonal at $Z_0 Z_2$, LP(2) has its first correlating element in the $Z_0 Z_4$ diagonal at $Z_8 Z_{12}$, LP(3) has its first correlating element in the $Z_0 Z_6$ diagonal at $Z_2 Z_6$, LP(4) has its first correlating element in the $Z_0 Z_8$ diagonal at $Z_0 Z_8$, LP(5) has its first correlating element in the $Z_0 Z_{10}$ diagonal at $Z_2 Z_{12}$ and LP(6) has its first correlating element in the $Z_0 Z_{12}$ diagonal at $Z_0 Z_{12}$. These first correlating elements would then define the lags of the ACF for the first rangecell, the next element in each diagonal would then define the lags of the ACF for the second rangecell, etc.

VI. DESCRIPTION OF THE LAG PROFILE SUBROUTINE

The Lag Profile subroutine was written to conform to the basic programming philosophy of the EISCAT correlator (REF 1). This subroutine exists in the EISCAT computer system and any user may use it to create a special program to suit a particular experiment. The program listing is given in Appendix A.

APB REGISTER STACK

RS(15) = No. of Samples-1 for Time Average
RS(14) = No. of Lag Profiles-1
RS(13) = Increment (=1)
RS(12) = Lag Increment
RS(11) = Temporary Storage
RS(10) = Temporary Storage
RS(9) = Start Address of Data

PC= 0	Do Nothing	
PC= 1	RS(11)=F	F→Q-RS(13)
PC= 2	RS(11)=F	F→RS(13)+RS(11)
PC= 3	RS(11)=F	F→O
PC= 4	RS(11)=F	F→RS(12)+RS(11)
PC= 5		F→RS(15)-RS(11)
PC= 6	Do Nothing	
PC= 7	RS(10)=F	F→Q-RS(13)
PC= 8	RS(10)=F	F→RS(13)+RS(10)
PC= 9		F→RS(11)+RS(10)
PC=10	Do Nothing	

Note that the Q register must be defined with the Start Address of the data (ie RS(9)), LCR1 must be RELOADED with RS(15) and LCR2 must be reloaded with RS(14) in the main program.

APM REGISTER STACK

RS(15) = Increment (=1)

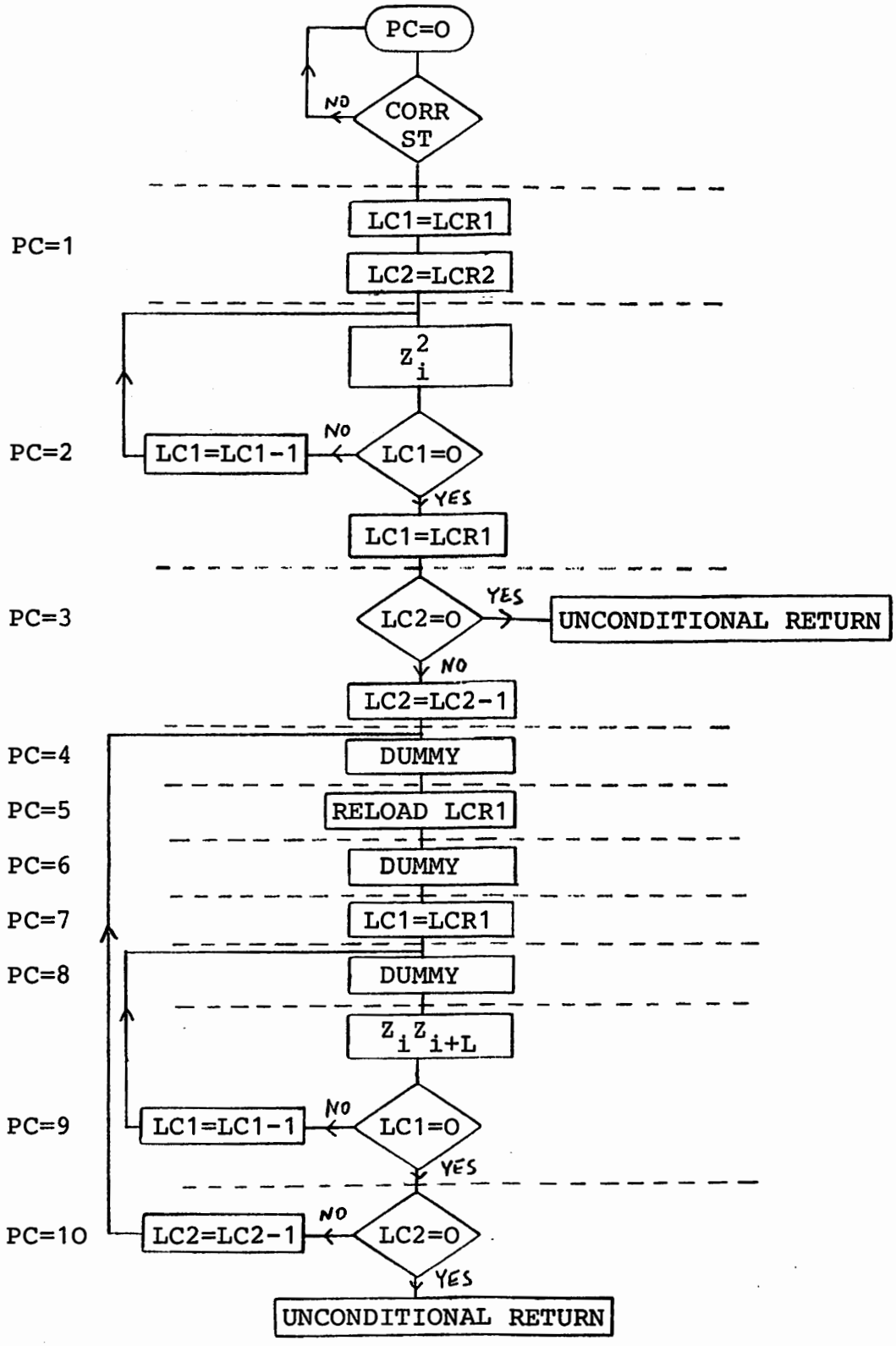
```
PC= 0    Do Nothing
PC= 1              Q=F    F→Q-RS(15)
PC= 2              Q=F    F→Q+RS(15)
PC= 3    Do Nothing
PC= 4    Do Nothing
PC= 5    Do Nothing
PC= 6    Do nothing
PC= 7    Do Nothing
PC= 8    Do Nothing
PC= 9              Q=F    F→Q+RS(15)
PC=10    Do Nothing
```

Note that the Q register must be defined with the Start Address (normally =0) for the result memory in the main program. The Q register contains the last address of the result memory that the data was written to when a RETURN to the main program is made.

The Total No. of Locations used in the Result Memory =

$$N(P+1)-LI(P(P+1)/2)$$

where N=Total No. of Samples for the block of data
P=No. of Lag Profiles-1
LI=Lag Increment



FLOW CHART FOR THE LAG PROFILE SUBROUTINE

VII. DESCRIPTION OF THE UNIVERSAL PROGRAM

The Universal Program exists on the EISCAT computer system and may be used by any experimenter, for a particular experiment. For details on how to use the program refer to the next chapter. The program listing is given in Appendix A.

Full details of how the main program is constructed is given here. There is one modification to the Lag Profile subroutine in order to allow this program to have the capability of skipping program slices. This modification is inserted after PC=1 of the the Lag Profile subroutine and the statement is IF(LC1.EQ.0) THEN RETURN ELSE CONTINUE. Thus the Lag Profile subroutine used in this program has eleven instead of ten statements.

APB REGISTER STACK

RS(15) = No. of Samples-1 for Time Average	⎫	
RS(14) = No. of Lag Profiles-1	⎪	
RS(13) = Increment (=1)	⎪	
RS(12) = Lag Increment	⎪	1st Program Slice
RS(11) = Temporary Storage	⎪	
RS(10) = Temporary Storage	⎪	
RS(9) = Start Address of Data Computed 1st	⎭	
RS(8) = No. of Samples-1 for Time Average (Same for 3rd Program Slice)	⎫	
RS(7) = No. of Lag Profiles-1	⎪	2nd Program Slice
RS(6) = Lag Increment (Same for 3rd Program Slice)	⎪	
RS(5) = Start Address of Data Computed 2nd	⎭	
RS(4) = Flag for 3rd Program Slice (=0 Then Jump to Power Program Slice)	⎫	
RS(3) = No. of Lag Profiles-1	⎪	3rd Program Slice
RS(2) = Start Address of Data Computed 3rd	⎭	
RS(1) = No. of Samples-1 for Time Average	⎫	Power Program
RS(0) = Start Address of Data Computed 4th	⎭	

For the Main Program, we have:

```
PC= 0    Do Nothing
PC= 1                        F→RS(15)
PC= 2    Do Nothing
PC= 3                        F→RS(14)
PC= 4              Q=F       F→RS( 9)
PC= 5                        F→RS( 8)
PC= 6    Do nothing
PC= 7                        F→RS( 7)
PC= 8              Q=F       F→RS( 5)
PC= 9                        F→RS( 4)
PC=10    Do Nothing
PC=11    Do Nothing
PC=12    Do Nothing
PC=13                        F→RS( 8)
PC=14    Do Nothing
PC=15                        F→RS( 3)
PC=16              Q=F       F→RS( 2)
PC=17                        F→RS( 1)
PC=18    Do Nothing
PC=20              Q=F       F→RS( 0)
PC=21    Do Nothing
PC=22    Do Nothing
```

APM REGISTER STACK

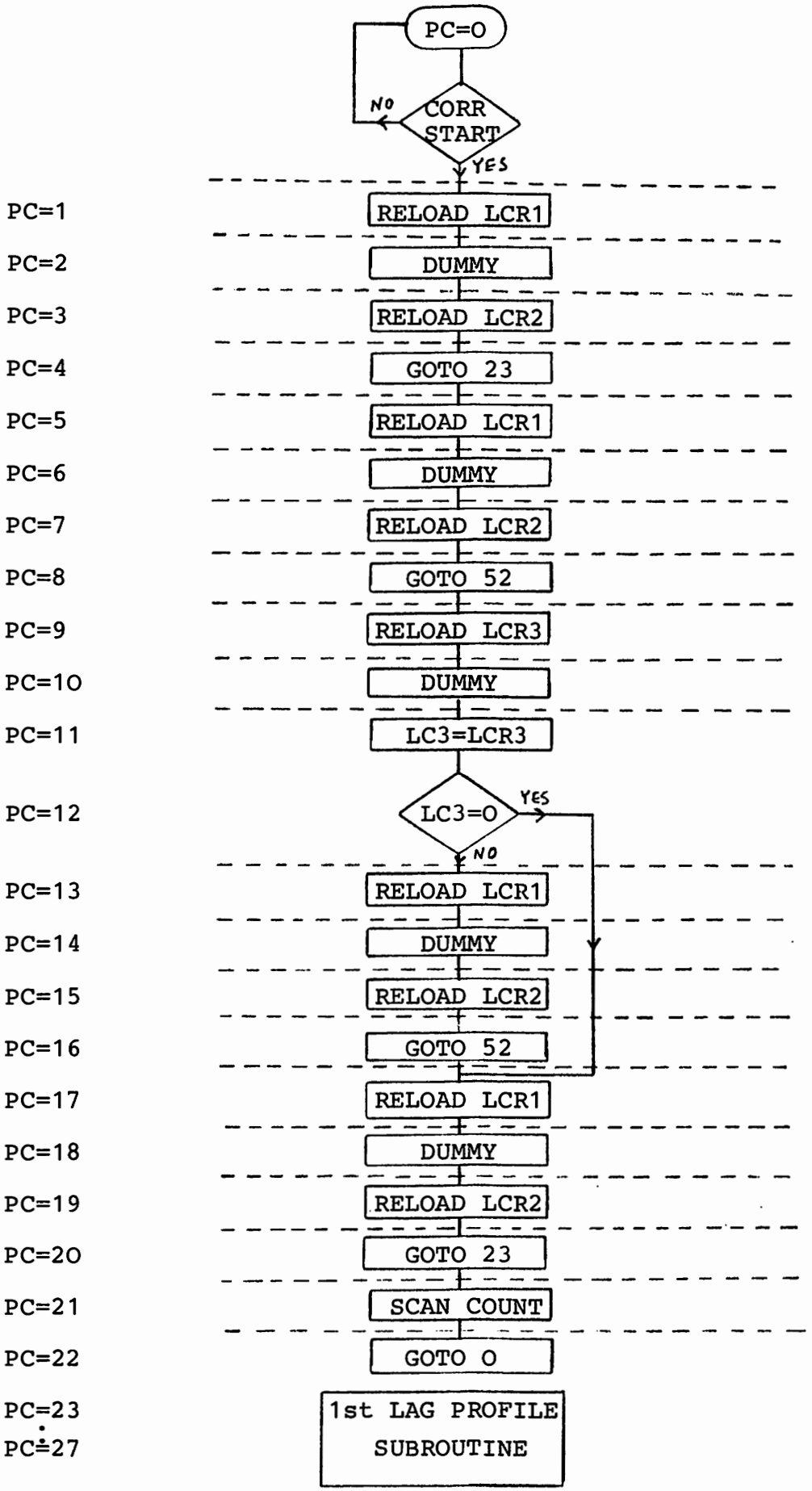
```
RS(15) = Increment (=1)
RS( 0) = Increment for Transfer Program (=1)

PC= 0    Do Nothing
PC= 1    Do Nothing
PC= 2    Do Nothing
PC= 3    Do Nothing
PC= 4              Q=F       F→0
PC= 5    Do Nothing
PC= 6    Do Nothing
PC= 7    Do Nothing
PC= 8              Q=F       F→Q+RS(15)
PC= 9    Do Nothing
```

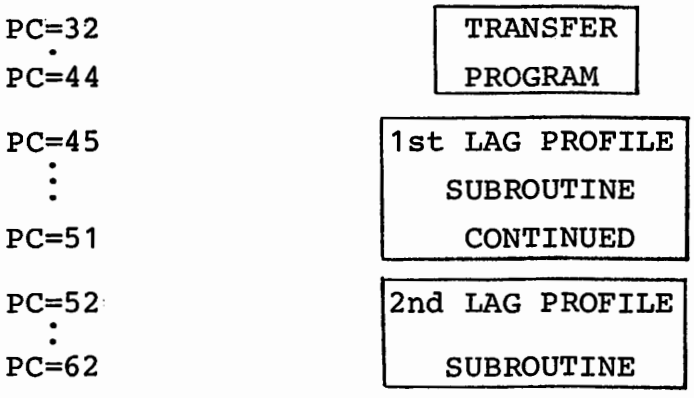
```
PC=10     Do Nothing
PC=11     Do Nothing
PC=12     Do Nothing
PC=13     Do Nothing
PC=14     Do Nothing
PC=15     Do Nothing
PC=16               Q=F     F→Q+RS(15)
PC=17     Do Nothing
PC=18     Do Nothing
PC=19     Do Nothing
PC=20               Q=F     F→Q+RS(15)
PC=21               Q=F     F→Q+RS(15)
PC=22     Do Nothing

SAR=1
DATAI REG. = No. of 64 BIT Words for DMA transfer
```

The first program slice is computed using the 1st Lag Profile subroutine. The second and third program slices are computed using the 2nd Lag Profile subroutine. This is why RS(8) and RS(6) have to be the same for the third program slice as these registers are used internally in the subroutine itself, so their values have to be fixed for both program slices. As a consequence of this, if the second program slice is skipped then the third program slice will also be skipped. However, this is not a serious restriction when designing an experiment. The Power Program slice is computed using the 1st Lag Profile subroutine. It could have also been done using the 2nd Lag Profile subroutine as inly LP(0) is computed.

The SCAN COUNT (number of Start Computes counted) is placed after the last sample in the result memory and must be included when defining the DATAI register.

Note that the correlator STATUS and CONTROL words are transferred out before the data from the result memory, therefore the maximum value for the DATAi register is 2047 and not 2048. The reason for this is that an area equivalent to the correlator's 2K (ie 8k NORD 10 computer words) memory is reserved in the NORD 10 and the correlator STATUS and CONTROL words are also dumped into this area.


PC=0
NO
CORR START
YES
PC=1
RELOAD LCR1
PC=2
DUMMY
PC=3
RELOAD LCR2
PC=4
GOTO 23
PC=5
RELOAD LCR1
PC=6
DUMMY
PC=7
RELOAD LCR2
PC=8
GOTO 52
PC=9
RELOAD LCR3
PC=10
DUMMY
PC=11
LC3=LCR3
PC=12
LC3=0
YES
NO
PC=13
RELOAD LCR1
PC=14
DUMMY
PC=15
RELOAD LCR2
PC=16
GOTO 52
PC=17
RELOAD LCR1
PC=18
DUMMY
PC=19
RELOAD LCR2
PC=20
GOTO 23
PC=21
SCAN COUNT
PC=22
GOTO 0
PC=23
PC≒27
1st LAG PROFILE SUBROUTINE



FLOW CHART FOR THE UNIVERSAL PROGRAM

VIII. HOW TO USE THE UNIVERSAL PROGRAM

The Universal Program consists of four program slices. The first two slices are general with the third slice sharing some parameters from the second slice and the fourth slice (Power Program) computes LP(O) only. By implication one would assume that one could receive on a maximum of four different frequencies simultaneously (a big increase over the traditional programs!). However, this is not the case, this program can in fact actually cope with eight different frequency channels simultaneously (maximum for EISCAT). Whether this is actually feasable will have to be seen in the future as there are possibly two limitations. The first is that the program can only handle four different types of modulations simultaneously; one of which must necessarily be a power profile group. Different types of modulation do not refer to the modulation pattern itself but to the parameters which define the pattern, ie the Number of Lag Profiles (P) and the Lag Increment (LI). If these two parameters are the same for a set modulation patterns then they are said to be similar and are of the same type. The second limitation is that when using eight different frequency channels simultaneously, it may be possible that the correlator result memory might turn out to be too small.

A six channel program, designed by Tauno Turunen, which was mainly used for testing purposes, will be given as an example.

The correlator data field is defined as:

SAR (Start Address Register) = 1
DATAI REGISTER = No. of 64 Bit Words for DMA Transfer

APB REGISTER STACK

RS(15) = No. of Samples-1 for Time Average
RS(14) = No. of Lag Profiles-1
RS(13) = Increment (=1)
RS(12) = Lag Increment
RS(11) = Temporary Storage
RS(10) = Temporary Storage
RS(9) = Start Address of Data Computed 1st

} 1st Program Slice

RS(8) = No. of Samples-1 for Time Average (Same for 3rd Program Slice)
RS(7) = No. of Lag Profiles-1
RS(6) = Lag Increment (Same for 3rd Program Slice)
RS(5) = Start Address of Data Computed 2nd
} 2nd Program Slice

RS(4) = Flag for 3rd Program Slice (=0 Then Jump to Power Program Slice)
RS(3) = No. of Lag Profiles-1
RS(2) = Start Address of Data Computed 3rd
} 3rd Program Slice

RS(1) = No. of Samples-1 for Time Average
RS(0) = Start Address of Data Computed 4th
} Power Program

APM REGISTER STACK

RS(15) = Increment (=1)
RS(0) = Increment for Transfer Program (=1)

The DATAI register must also contain the location for the SCAN COUNT (No. of Start Computes counted).
The registers RS(15) - RS(9) defines the parameters for the 1st program slice.
The registers RS(8) - RS(5) defines the parameters for the 2nd program slice.
The registers RS(4) - RS(2) defines the parameters for the 3rd program slice.
The registers RS(1) - RS(0) defines the parameters for the Power program slice.

The program gives the user the ability to skip program slices as not every experiment will need four program slices. In order to achieve this define:

RS(15)=0 to skip the 1st program slice.
RS(8)=0 to skip the 2nd program slice.
RS(4)=0 to skip the 3rd program slice.
RS(1)=0 to skip the Power program slice.

It must be noted that if you skip the 2nd program slice you must also skip the 3rd program slice. However, if you do not skip the 2nd program slice you can skip the 3rd program slice.

It is also possible to obtain the Power program by using any of the other 3 program slices by defining:

RS(14)=0 for the 1st program slice.
RS(7)=0 for the 2nd program slice.
RS(3)=0 for the 3rd program slice.

It can be seen that the correlator always handles the data from the buffer memory in the same way regardless of the pulse scheme transmitted. Therefore, it is possible to consider the samples from different frequency channels as one block of data for a program slice if the modulations give the same number of Lag Profiles (or defining the same number of Lag Profiles even if a modulation could give more) and the Lag Increment is the same. Thus a block of data for a program slice can be constructed if the modulations are similar or made to be similar. However, this should be done constructively because it is so that a long pulse, which normally needs a Lag Increment of 1, could be grouped together with data analysed with a Lag Increment of 2 which would give every other Lag Profile. This could be compensated by oversampling, if necessary, but this would be a totally ineffective use of the program and is not needed in practice. Of course, this will make the post processing program somewhat complex as an exact result memory map will be needed so as to know which locations contain data and which contain non correlating data. But on the other hand very complex experiments can be designed which utilize the full potential of the EISCAT radar system.

Here general guidelines are given on how to go about deciding what the parameters should be and tips on how to make a buffer and result memory map. Having decided on what pulse schemes should be transmitted for the experiment, the next question is, how many samples should be taken on each channel which will cover the range that I want and will also fit into the result memory? Unfortunately, there is no cut and dried method which gives an answer immediately. It is essentially only through trial and error, compromise between range and result memory that a solution will be found. A way to go about this procedure systematically is to decide which channels

should be grouped together, whereby the data, noise and calibration samples of one channel form a sub-group of the block of data. For the block of data for a given program slice the number of result memory locations can be calculated by using the formula:

No. of Result Memory Locations = $N(P+1)-LI(P(P+1)/2)$

where N = Total No. of Samples for the block of data.
P = No. of Lag Profiles-1
LI = Lag Increment

It should be noted here that the maximum amount of result memory locations = 2047. Therefore, the total amount of result memory locations used for the data should not exceed 2046 as one location is needed for the SCAN COUNT. In this way the number of samples used in each block of data can be adjusted until an acceptable solution can be found.

A buffer memory map is a very good way to see pictorially how the data should be placed in the buffer memory and thus reduces the risk of defining wrong start addresses for each channel in the ELAN program. Extra care must be taken when using Barker coding, as for each set of samples here (ie data, noise and calibration now considered separately) in the sub-group there are always 12 rubbish samples before and after each set. If this will always be so will depend on how the matched filter operations are finally arranged in EISCAT. It is possible to have a solution which virtually eliminates the need to analyse samples which are not fully decoded. For each program slice the data must be placed contiguously in the buffer memory to form one block of data, with all samples from one channel (ie signal, noise and calibration) forming a sub-group. It is not absolutely necessary to place the blocks of data for different program slices together so that they form a contiguous set of data in the buffer memory but it's good practice to do so.

The parameters for the Universal Program can now be set as everything has been defined for running an experiment.

Finally, a result memory map will be necessary when looking at the correlator output in real-time. As mentioned earlier there will be rubbish data in the output so it would be good

to know in advance which locations in the result memory contain potentially good data (ie where the lag profiles lie in the result memory). A handy formula to use when calculating the number of locations used for a given lag profile from a particular set of data is:

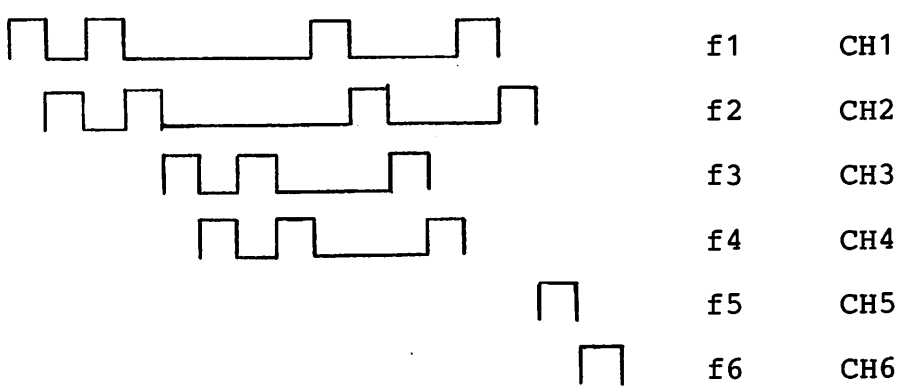
No. of Locations used = N-(L×LI)

where N = No. of Samples in the data set.
L = Lag Profile Number (ie 0,1,2,..,P).
LI = Lag Increment for the data set.

The term potentially good data is used because for a given lag profile some of the cross products do not have any meaning, depending on the pulse scheme transmitted. In the case of the Long Pulse scheme (ie for autocorrelation) it depends on what weighting scheme is used (triangular or block). For the Multi Pulse scheme it depends on how the pulses are spaced.

IX. AN EXAMPLE

This example is given as it covers most of the principles given in the previous chapter. Consider the following scheme:



Each element pulse is 15 usec. The total transmission time is 240 usec. The receiver bandwidth is set to ±50 KHz and the sampling rate is 15 usec.

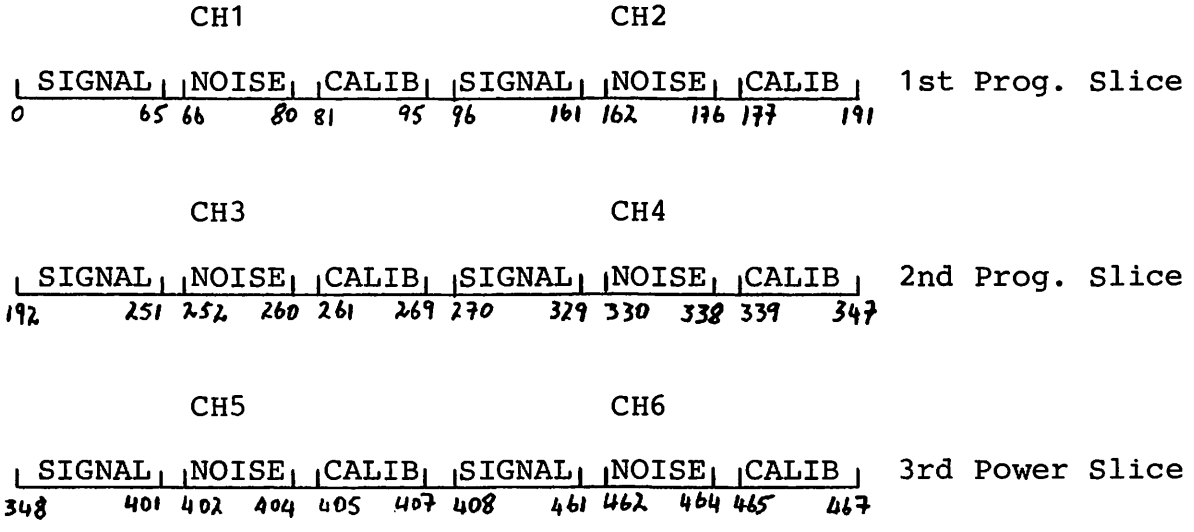
CH1 and CH2 are grouped together to form the 1st block of data, CH3 and CH4 are grouped together to form the 2nd block of data and CH5 and CH6 are grouped together to form the 3rd block of data. The 1st block of data will use the 1st program slice, the 2nd block of data will use the 2nd program slice and the 3rd block of data will use the Power slice.

Through trial and error we have a scheme that looks like this:

Range 92.5 - 211.75 Km.

	SIGNAL	NOISE	CALIB		RES. MEM.
CH1	66	15	15	192 Samples	1302
CH2	66	15	15		
CH3	60	9	9	156 Samples	612
CH4	60	9	9		
CH5	54	3	3	120 Samples	120
CH6	54	3	3		
				468 Samples	2034+1 (SCAN COUNT)

BUFFER MEMORY MAP



All the necessary information is now available and for each program slice we have:

Start Address = 0
No. of Samples = 192
No. of Lag Profiles = 7
Lag Increment = 2
} 1st Program Slice

Start Address = 192
No. of Samples = 156
No. of Lag Profiles = 4
Lag Increment = 2
} 2nd Program Slice

Start Address = 348
No. of Samples = 120
} Power Program Slice

For the Universal Program the parameters would be:

SAR = 1
DATAI REG = 2035

APB REGISTER STACK

RS(15) = 191
RS(14) = 6
RS(13) = 1
RS(12) = 2
RS(11) = Temporary Storage
RS(10) = Temporary Storage
RS(9) = 0
RS(8) = 155
RS(7) = 3
RS(6) = 2
RS(5) = 192
RS(4) = 0
RS(3) = 0
RS(2) = 0
RS(1) = 119
RS(0) = 348

APM REGISTER STACK

RS(15) = 1
RS(0) = 1

The Result Memory Map on the following page gives where the potentially good data for each lag profile in the result memory lie. Cross over points are omitted as they have no meaning for the formation of the autocorrelation functions.

RESULT MEMORY MAP

CH1	SIGNAL	NOISE	CALIB
LAG PROFILE(0)	0-65	66-80	81-95
LAG PROFILE(1)	192-255		
LAG PROFILE(2)	382-443		
LAG PROFILE(3)	570-629		
LAG PROFILE(4)	756-813		
LAG PROFILE(5)	940-995		
LAG PROFILE(6)	1122-1175		

CH2	SIGNAL	NOISE	CALIB
LAG PROFILE(0)	96-161	162-176	177-191
LAG PROFILE(1)	288-351		
LAG PROFILE(2)	478-539		
LAG PROFILE(3)	666-725		
LAG PROFILE(4)	852-909		
LAG PROFILE(5)	1036-1091		
LAG PROFILE(6)	1218-1271		

CH3	SIGNAL	NOISE	CALIB
LAG PROFILE(0)	1302-1361	1362-1370	1371-1379
LAG PROFILE(1)	1458-1515		
LAG PROFILE(2)	1612-1667		
LAG PROFILE(3)	1764-1817		

CH4	SIGNAL	NOISE	CALIB
LAG PROFILE(0)	1380-1439	1440-1448	1449-1457
LAG PROFILE(1)	1536-1593		
LAG PROFILE(2)	1690-1745		
LAG PROFILE(3)	1842-1895		

CH5	SIGNAL	NOISE	CALIB
LAG PROFILE(0)	1914-1967	1968-1970	1971-1973

CH6	SIGNAL	NOISE	CALIB
LAG PROFILE(0)	1974-2027	2028-2030	2031-2033

The result memory map only gives where the potentially good data is for each lag profile. The actual useful data is as follows.

CH1	USEFUL DATA	NOISE	CAILB
LAG PROFILE(0)	0-53	66-80	81-95
LAG PROFILE(1)	192-245		
LAG PROFILE(2)	390-443		
LAG PROFILE(3)	572-625		
LAG PROFILE(4)	756-809		
LAG PROFILE(5)	942-995		
LAG PROFILE(6)	1122-1175		

CH2	USEFUL DATA	NOISE	CALIB
LAG PROFILE(0)	96-149	162-176	177-191
LAG PROFILE(1)	288-341		
LAG PROFILE(2)	486-539		
LAG PROFILE(3)	668-721		
LAG PROFILE(4)	852-905		
LAG PROFILE(5)	1038-1091		
LAG PROFILE(6)	1218-1271		

CH3	USEFUL DATA	NOISE	CALIB
LAG PROFILE(0)	1302-1355	1362-1370	1371-1379
LAG PROFILE(1)	1458-1511		
LAG PROFILE(2)	1614-1667		
LAG PROFILE(3)	1764-1817		

CH4	USEFUL DATA	NOISE	CALIB
LAG PROFILE(0)	1380-1433	1440-1448	1449-1457
LAG PROFILE(1)	1536-1589		
LAG PROFILE(2)	1692-1745		
LAG PROFILE(3)	1842-1895		

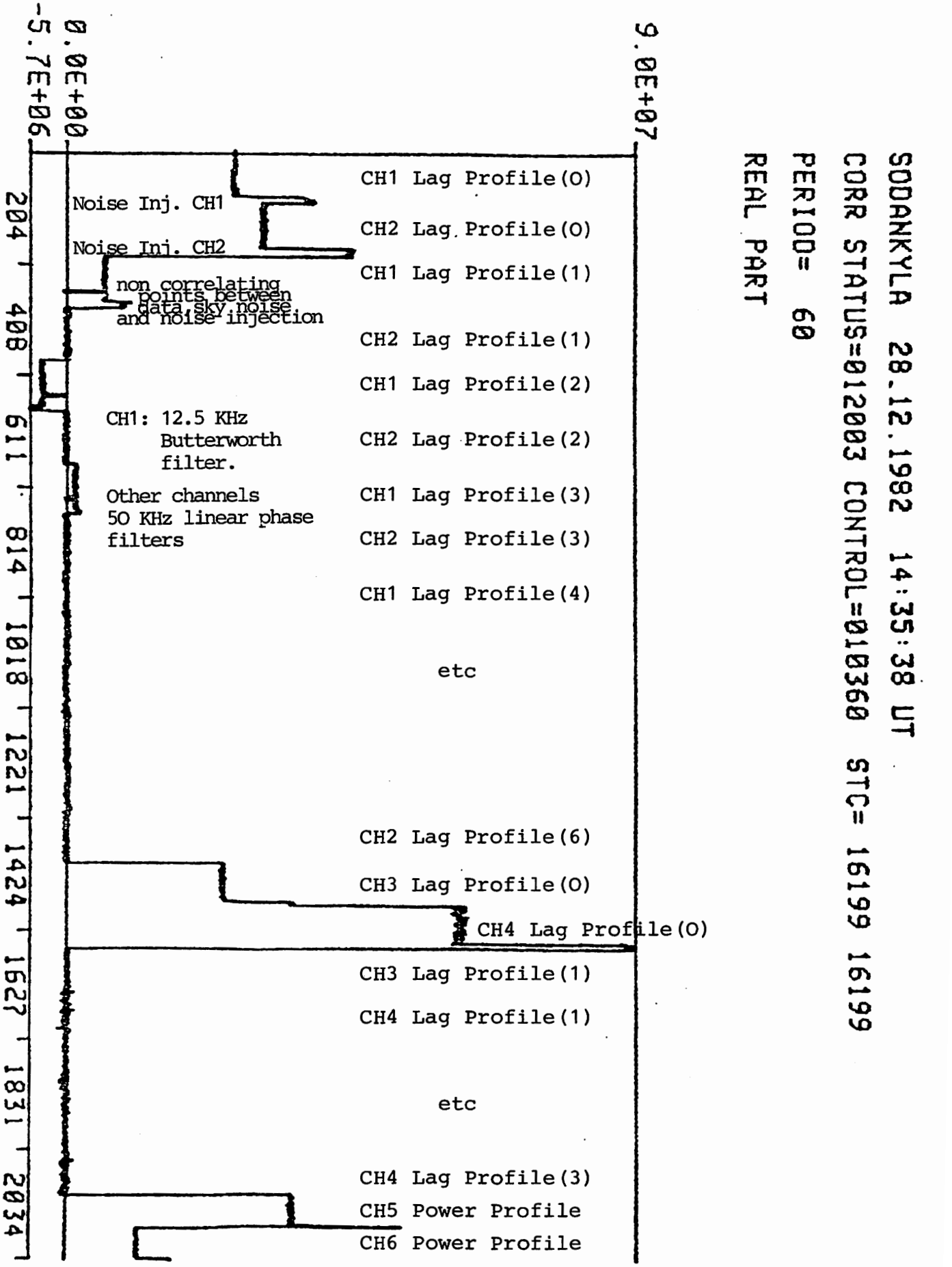
CH5	USEFUL DATA	NOISE	CALIB
LAG PROFILE(0)	1914-1967	1968-1970	1971-1973

CH6	USEFUL DATA	NOISE	CALIB
LAG PROFILE(0)	1974-2027	2028-2030	2031-2033

It can be seen that in the case of the Power Program the terms potentially good data and useful data are synonymous.

The following figure shows the correlator output of this program which was run at the Sodankylä site. In this run only sky noise was received. It can be seen that the noise injection for calibration is clearly distinguishable. This can be recognized by the "spikes" in the Lag Profile (0) for each channel. The start of the non correlating points are distinguishable in CHANNEL 1 in the Lag Profile (1) - Lag Profile (3) by the sudden drop towards zero. This feature is also present in CHANNEL 2 - CHANNEL 4, however it cannot be seen in this plot because of the scaling. The attenuation level of each channel is clearly distinguishable and was so set in order to make the differentiation between channels clearer.

In an experiment this kind of display in real-time gives the experimenter a very effective means of seeing as to how the experiment is progressing. It enables the experimenter, at a glance, to see such features as: where the zero crossings of the ACF occur, an estimation of the depth of the first minimum, if the attenuators are set correctly, the sky noise level and the calibration levels for all channels. If these features can be seen on the plot then the experimenter can be fairly sure that everything is working correctly.


SODANKYLA 28.12.1982 14:35:38 UT
CORR STATUS=012003 CONTROL=010360 STC= 16199 16199
PERIOD= 60
REAL PART
-5.7E+06
0.0E+00
9.0E+07
204
408
611
814
1018
1221
1424
1627
1831
2034
Noise Inj. CH1
Noise Inj. CH2
non correlating points between data, sky noise and noise injection
CH1: 12.5 KHz Butterworth filter.
Other channels 50 KHz linear phase filters
CH1 Lag Profile(0)
CH2 Lag Profile(0)
CH1 Lag Profile(1)
CH2 Lag Profile(1)
CH1 Lag Profile(2)
CH2 Lag Profile(2)
CH1 Lag Profile(3)
CH2 Lag Profile(3)
CH1 Lag Profile(4)
etc
CH2 Lag Profile(6)
CH3 Lag Profile(0)
CH4 Lag Profile(0)
CH3 Lag Profile(1)
CH4 Lag Profile(1)
etc
CH4 Lag Profile(3)
CH5 Power Profile
CH6 Power Profile

X. CONCLUSIONS

The Universal Program has many advantages over the traditional programs. It virtually eliminates the need to write new correlator programs. Whereas in the traditional case, special programs have to be written to fit a particular combination of pulse schemes (two at the most without complication) for an experiment, the Universal Program makes no distinction between schemes and thus it is possible to receive on all 8 frequency channels, provided that number of types of different modulation patterns do not exceed 4, in the EISCAT radar system. The Universal Program can also handle the pulse to pulse correlation experiments appearing, for example, in mesopheric experiments. A short discussion on how the Universal Program can be applied to pulse to pulse correlation is given in Appendix B . In fact a multi channel Barker coded pulse to pulse correlation problem is relatively easy but it is not handled in this report. It could also, in principle, handle simultaneously Barker coded and non Barker pulse codes at several channels together with a couple of long pulses but the EISCAT receiving system cannot do it.

It relieves the experimenter of having to worry about if there is a suitable correlator program on the system and in the worst case situation, having to develop a new program. The full potential of the EISCAT radar system can be utilized with the use of the Universal Program, whereas with the traditional programs this can never be the case. It gives the experimenter full control in manipulating the data in the post processing process as all the basic information is there. It is also effective in helping to test various parts of the hardware system.

The most obvious disadvantage with the Universal Program is that it uses more result memory locations. This is necessarily so because it makes use of all information which exists in the data. In the multi pulse case, if there are missing lags which are included in the range of the lag profiles then this missing lag will also be calculated as a lag profile which will have to be discarded later. The experimenter must also know exactly which locations contain potentially

good or useful data. Another drawback which is not in the Universal Program but in the correlator is that the result memory can in some cases turn out to be too small for some experiments (for example, using all 8 frequencies). With only 2 K memory size this will be a limiting factor when designing experiments.

ACKNOWLEDGEMENT

The co-author Terrance Ho would like to thank the Geophysical Observatory in Sodankylä for sponsoring this work, the EISCAT Scientific Association for the use of their facilities and the EISCAT staff in Finland for their help.

XI. REFERENCES

1. TERRANCE HO and HANS-JØRGEN ALKER: "SCIENTIFIC PROGRAMMING OF THE EISCAT DIGITAL CORRELATOR (REVISED)". EISCAT TECHNICAL NOTE. 81/24 1981

APPENDIX A

Here listings of the Lag Profile subroutine and the Universal Program are given.

MICRO-PROGRAM FOR DIGITAL CORRELATOR

AUTOR: TERRANCE HO DATE: 1/10/82

PROGRAM NAME: LAG PROFILE SUBROUTINE

FILE-NAME (NORD 10): PROG16:DATA

PROGRAM DESCRIPTION:

DATA CHANNEL 1

$$\mathrm{Re}\{\text{Lag Profile}(L)\} = X_i X_{i+L\times LI} + Y_i Y_{i+L\times LI}$$

DATA CHANNEL 2

$$\mathrm{Im}\{\text{Lag Profile}(L)\} = X_{i+L\times LI} Y_i + X_i Y_{i+L\times LI}$$

WHERE i=0,1,2,...N-1-(L×LI)

N=TOTAL NO. OF SAMPLES

L=0,1,2,...,P

P=NO. OF LAG PROFILES-1

LI=LAG INCREMENT

RESTRICTIONS

MINIMUM NO. OF SAMPLES IN RANGEDATA: 1

NOTES

1. The formulae above are given with respect to the way in which the X, Y samples are read from the buffer memory.
2. The Q registers of the APB, APM processors must be defined with the start addresses of the buffer, result memories in the main program.
3. The LCR1 register must be reloaded with the APBRS(15) register, the LCR2 register must be reloaded with the APBRS(14) register in the main program.

START ADDRESS FOR PROGRAM: 1

PROGRAM-MEMORY LOCATIONS USED: 1 - 10

MICRO-PROGRAM FOR DIGITAL CORRELATOR

AUTOR: TERRANCE HO DATE: 1/10/82

PROGRAM NAME: LAG PROFILE SUBROUTINE
FILE-NAME (NORD 10): PROG16:DATA
PROGRAM DESCRIPTION:

NOTES (CONTINUED)

4. The APBRS(9) register contains the start address of the data in the buffer memory.
5. The Q register of the APM processor, on RETURN from the subroutine, contains the address of the last location which was written to in the result memory.

START ADDRESS FOR PROGRAM:
PROGRAM-MEMORY LOCATIONS USED:

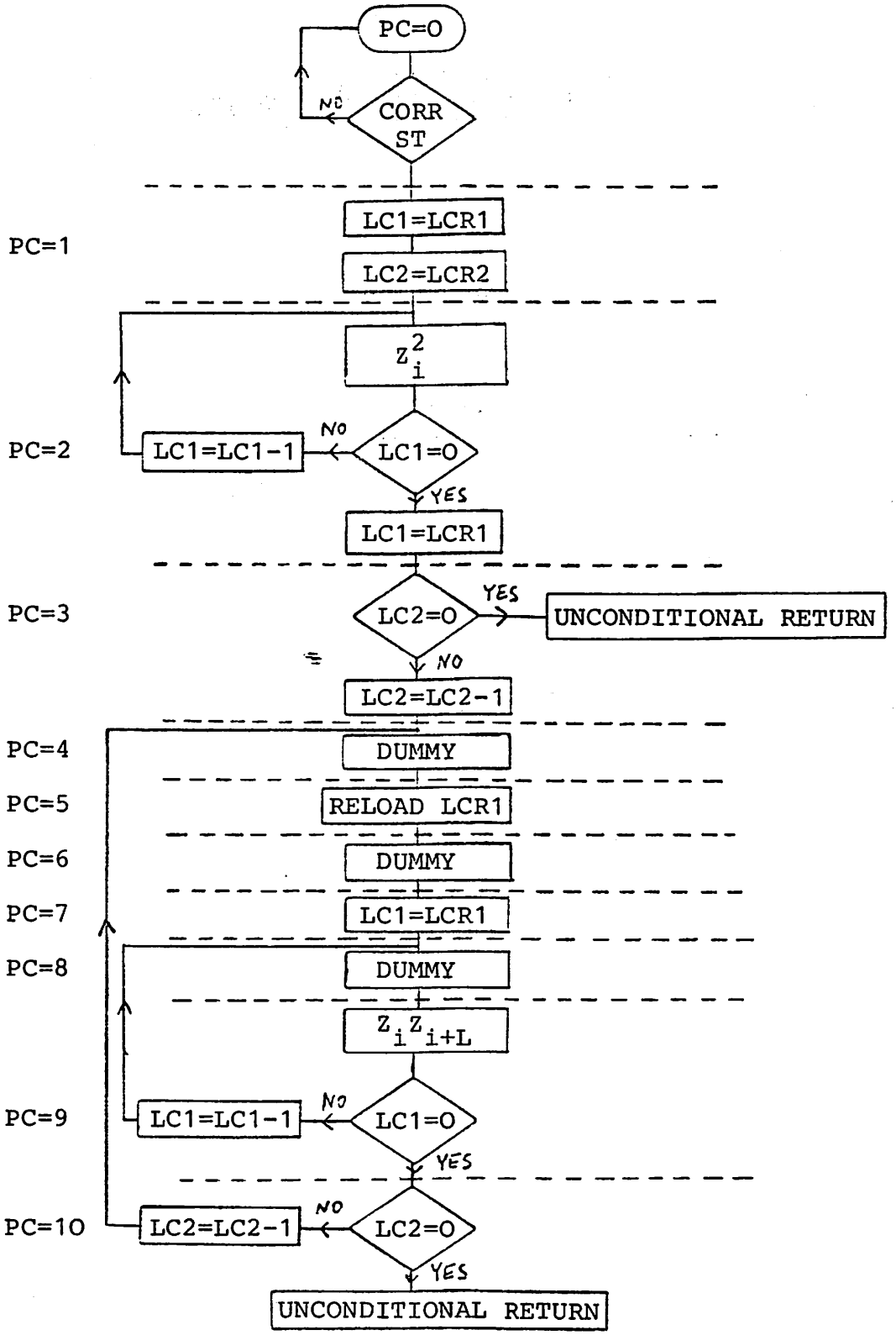
MICRO-PROGRAM FOR DIGITAL CORRELATOR

AUTOR: TERRANCE HO

DATE: 1/10/82

PROGRAM NAME: LAG PROFILE SUBROUTINE
FILE-NAME (NORD 10): PROG16:DATA

REGISTER NAME	REGISTER ADDRESS	PARAMETER
SAR	4	START ADDRESS OF SUBROUTINE
APB RS(15)	16,15	TOTAL NO. OF SAMPLES-1
APB RS(14)	16,14	NO. OF LAG PROFILES-1
APB RS(13)	16,13	INCREMENT (=1)
APB RS(12)	16,12	LAG INCREMENT
APB RS(11)	16,11	TEMPORARY STORAGE
APB RS(10)	16,10	TEMPORARY STORAGE
APB RS(9)	16,9	START ADDRESS OF DATA
APM RS(15)	17,15	INCREMENT (=1)



FLOW CHART FOR THE LAG PROFILE SUBROUTINE

MICRO-PROGRAM FOR DIGITAL CORRELATOR

AUTOR: TERRANCE HO DATE: 1/10/82

PROGRAM NAME: LAG PROFILE SUBROUTINE
FILE-NAME (NORD 10): PROG16:DATA
PROGRAM DESCRIPTION:

	APB PROCESSOR		APM PROCESSOR	
PC=1	RS(11)=F	F→Q-RS(13)	Q=F	F→Q-RS(15)
PC=2	RS(11)=F	F→RS(13)+RS(11)	Q=F	F→Q+RS(15)
PC=3	RS(11)=F	F→Ø	DO NOTHING	
PC=4	RS(11)=F	F→RS(12)+RS(11)	DO NOTHING	
PC=5		F→RS(15)-RS(11)	DO NOTHING	
PC=6	DO NOTHING		DO NOTHING	
PC=7	RS(10)=F	F→Q-RS(13)	DO NOTHING	
PC=8	RS(10)=F	F→RS(13)+RS(10)	DO NOTHING	
PC=9		F→RS(11)+RS(10)	Q=F	F→Q+RS(15)
PC=10	DO NOTHING		DO NOTHING	

START ADDRESS FOR PROGRAM:
PROGRAM-MEMORY LOCATIONS USED:

1 *** CORRELATOR SIMULATOR ***

PROGRAM NAME : LAG PROFILE SUBROUTINE

CODING (INTEGER) OF SEPARATE CORRELATOR FUNCTIONS

PROGRAM-INSTRUCTIONS DEFINED

MEM-LOC.	COND.CODE	CODE-B	CODE-A	ADDR.	LC1	LCR1A	LC2	LC3	RELOAD	R-ADDR.
0	56	0	4	0	0	0	0	0	0	0
1	56	0	4	0	2	0	3	0	0	0
2	57	4	6	2	6	0	0	0	0	0
3	58	1	4	0	0	0	1	0	0	0
4	56	0	4	0	0	0	0	0	0	0
5	56	0	4	0	0	0	0	0	1	18
6	56	0	4	0	0	0	0	0	0	0
7	56	0	4	0	2	0	0	0	0	0
8	56	0	4	0	0	0	0	0	0	0
9	57	4	6	8	1	0	0	0	0	0
10	58	1	6	4	0	0	1	0	0	0

APB-INSTRUCTIONS DEFINED

MEM-LOC.	ALU-SOURCE	ALU-FUNCTION	ALU-DESTIN.	A-ADDR.	B-ADDR.	SELECT
0	7	5	1	0	0	0
1	0	1	3	13	11	0
2	1	0	3	13	11	0
3	7	5	3	0	11	0
4	1	0	3	12	11	0
5	1	2	1	15	11	0
6	7	5	1	0	0	0
7	0	1	3	13	10	0
8	1	0	3	13	10	0
9	1	0	1	11	10	0
10	7	5	1	0	0	0

APM-INSTRUCTIONS DEFINED

MEM-LOC.	ALU-SOURCE	ALU-FUNCTION	ALU-DESTIN.	A-ADDR.	B-ADDR.
0	7	5	1	0	0
1	0	1	0	15	0
2	0	0	0	15	0
3	7	5	1	0	0
4	7	5	1	0	0
5	7	5	1	0	0
6	7	5	1	0	0
7	7	5	1	0	0
8	7	5	1	0	0
9	0	0	0	15	0
10	7	5	1	0	0

ARITHMETICAL INSTRUCTIONS DEFINED

MEM-LOC.	M1A	M1B	M2A	M2B	M3A	M3B	M4A	M4B	SM1	SM2	SM3	SM4	M12	M34
0	0	0	0	0	0	0	0	0	0	0	0	0	15	15
1	0	0	0	0	0	0	0	0	0	0	0	0	15	15
2	0	0	1	1	1	0	0	1	3	3	3	3	9	6
3	0	0	0	0	0	0	0	0	0	0	0	0	15	15
4	0	0	0	0	0	0	0	0	0	0	0	0	15	15
5	0	0	0	0	0	0	0	0	0	0	0	0	15	15
6	0	0	0	0	0	0	0	0	0	0	0	0	15	15
7	0	0	0	0	0	0	0	0	0	0	0	0	15	15
8	0	0	1	1	1	0	0	1	1	1	1	1	9	6
9	0	0	1	1	1	0	0	1	2	2	2	2	9	6
10	0	0	0	0	0	0	0	0	0	0	0	0	15	15

ACCUMULATOR INSTRUCTIONS DEFINED

MEM-LOC.	STROBE I/O	WRITE	READ	CLEAR1	SET1	CLEAR2	SET2
0	0	0	0	0	0	0	0
1	1	0	0	1	0	0	0
2	1	1	1	0	0	0	0
3	1	0	0	0	0	0	0
4	1	0	0	0	0	0	0
5	1	0	0	0	0	0	0
6	1	0	0	0	0	0	0
7	1	0	0	0	0	0	0
8	1	0	0	0	0	0	0
9	1	1	1	0	0	0	0
10	1	0	0	0	0	0	0

I/O INSTRUCTIONS DEFINED

MEM-LOC.	SETF	CLEARF	SELECT	BUF.ADDR.	STROBE	I-REG.	ENABLE EDB	ENABLE EAB
0	0	0	0		0	0	0	
1	0	0	0		0	0	0	
2	0	0	0		0	0	0	
3	0	0	0		0	0	0	
4	0	0	0		0	0	0	
5	0	0	0		0	0	0	
6	0	0	0		0	0	0	
7	0	0	0		0	0	0	
8	0	0	0		0	0	0	
9	0	0	0		0	0	0	
10	0	0	0		0	0	0	

OUTPUT-TRANSFER INSTRUCTIONS DEFINED

MEM-LOC.	TRANSFER	INHIBIT	CLOCK	DATA-READY	TRANSFER-CODE	SOURCE
0	0	0	0	0	0	0
1	0	0	0	0	0	0
2	0	0	0	0	0	0
3	0	0	0	0	0	0
4	0	0	0	0	0	0
5	0	0	0	0	0	0
6	0	0	0	0	0	0
7	0	0	0	0	0	0
8	0	0	0	0	0	0
9	0	0	0	0	0	0
10	0	0	0	0	0	0

MICRO-PROGRAM FOR DIGITAL CORRELATOR

AUTOR: TERRANCE HO DATE: 1/10/82

PROGRAM NAME: UNIVERSAL PROGRAM
FILE-NAME (NORD 10): UNI-PROG:DATA
PROGRAM DESCRIPTION:

DATA CHANNEL 1

$$Re\{\text{Lag Profile}(L)\} = X_i X_{i+L\times LI} + Y_i Y_{i+L\times LI}$$

DATA CHANNEL 2

$$Im\{\text{Lag Profile}(L)\} = X_{i+L\times LI} Y_i + X_i Y_{i+L\times LI}$$

WHERE i=0,1,2,...,N-1-(L×LI)
N=TOTAL NO. OF SAMPLES FOR BLOCK OF DATA
L=0,1,2,...,P
P=NO. OF LAG PROFILES-1
LI=LAG INCREMENT

RESTRICTIONS

MINIMUM NO. OF SAMPLES IN RANGEDATA: 2

NOTES

1. This program assumes that the start address of the result memory is zero. The start address of the buffer memory must be defined in the APB register stack.
2. The number of start computes counted is placed after the last data sample in the result memory. Therefore the real and imaginary parts of this location will contain the number of start computes counted as a negative number.

START ADDRESS FOR PROGRAM: 1
PROGRAM-MEMORY LOCATIONS USED: 1 - 27, 32 - 62

MICRO-PROGRAM FOR DIGITAL CORRELATOR

AUTOR: TERRANCE HO DATE: 1/10/82

PROGRAM NAME: UNIVERSAL PROGRAM
FILE-NAME (NORD 10): UNI-PROG:DATA
PROGRAM DESCRIPTION:

NOTES (CONTINUED)

3. The location for the number of start computes counted must be included in the number of 64 BIT words for the DMA transfer, ie the total number of samples computed+1 (for number of start computes counted).
4. The Status and Control words are transferred out before the data to the computer.
5. All X, Y samples for a particular program slice must be placed contiguously in the buffer memory. All samples are written contiguously in the result memory.
6. If more than one channel is being used in a program slice, then the data, noise and calibration samples should be placed together contiguously to form a sub-group for that particular channel in the buffer memory.

HOW TO USE THE PROGRAM

RS(15) - RS(9) defines the parameters for the 1st Program Slice
RS(8) - RS(5) defines the parameters for the 2nd Program Slice
RS(4) - RS(2) defines the parameters for the 3rd Program Slice
RS(1) - RS(0) defines the parameters for the Power Program Slice

To skip the 1st Program Slice define RS(15)=0
To skip the 2nd Program Slice define RS(8)=0
To skip the 3rd Program Slice define RS(4)=0
To skip the Power Program Slice define RS(1)=0
Note that if you skip the 2nd Program Slice you must also skip the 3rd Program Slice. However, if you do not skip the 2nd Program Slice you can

START ADDRESS FOR PROGRAM:
PROGRAM-MEMORY LOCATIONS USED:

MICRO-PROGRAM FOR DIGITAL CORRELATOR

AUTOR: TERRANCE HO DATE: 1/10/82

PROGRAM NAME: UNIVERSAL PROGRAM
FILE-NAME (NORD 10): UNI-PROG:DATA
PROGRAM DESCRIPTION:

HOW TO USE THE PROGRAM (CONTINUED)

skip the 3rd Program slice.

It is also possible to obtain the Power Program using any of the other 3 Program Slices by:

For the 1st Program Slice define RS(14)=0

For the 2nd Program Slice define RS(7)=0

For the 3rd Program Slice define RS(3)=0

EXECUTION TIME OF THE PROGRAM

	PROGRAM USED	SKIPPED
1st Program Slice S1 =	8+N+2NP+5P-(LI×P(P+1))	6
2nd Program Slice S2 =	7+N+2NP+5P-(LI×P(P+1))	6
3rd Program Slice S3 =	11+N+2NP+5P-(LI×P(P+1))	4
Power Program Slice S4 =	7+N	6

TOTAL EXECUTION TIME = (S1+S2+S3+S4+8)×0.1666667 usec

Note that the factor 0.1666667 is based on the correlator cycle time of 167 nanosec. It is very likely that that the actual cycle time is 200 nanosec and so the factor should be changed to 0.2.

START ADDRESS FOR PROGRAM:
PROGRAM-MEMORY LOCATIONS USED:

MICRO-PROGRAM FOR DIGITAL CORRELATOR

AUTOR: TERRANCE HO DATE: 1/10/82

PROGRAM NAME: UNIVERSAL PROGRAM
FILE-NAME (NORD 10): UNI-PROG:DATA

REGISTER NAME	REGISTER ADDRESS	PARAMETER
SAR	4	START ADDRESS OF PROGRAM
DATAI REGISTER	6	NO. OF 64 BIT WORDS FOR DMA TRANSFER
APB RS(15)	16,15	NO. OF SAMPLES-1
APB RS(14)	16,14	NO. OF LAG PROFILES-1
APB RS(13)	16,13	INCREMENT (=1)
APB RS(12)	16,12	LAG INCREMENT
APB RS(11)	16,11	TEMPORARY STORAGE
APB RS(10)	16,10	TEMPORARY STORAGE
APB RS(9)	16,9	START ADDRESS OF DATA (COMPUTED 1st)
APB RS(8)	16,8	NO. OF SAMPLES-1 (SAME FOR 3rd PROGRAM SLICE)
APB RS(7)	16,7	NO. OF LAG PROFILES-1
APB RS(6)	16,6	LAG INCREMENT (SAME FOR 3rd PROGRAM SLICE)
APB RS(5)	16,5	START ADDRESS OF DATA (COMPUTED 2nd)
APB RS(4)	16,4	FLAG FOR 3rd PROGRAM SLICE (=0 THEN JUMP TO 4th PROGRAM SLICE)
APB RS(3)	16,3	NO. OF LAG PROFILES-1
APB RS(2)	16,2	START ADDRESS OF DATA (COMPUTED 3rd)
APB RS(1)	16,1	NO. OF SAMPLES-1
APB RS(0)	16,0	START ADDRESS OF DATA (COMPUTED 4th)
APM RS(15)	17,15	INCREMENT (=1)
APM RS(0)	17,0	INCREMENT FOR TRANSFER PROGRAM (=1)

*** CORRELATOR SIMULATOR ***

PROGRAM NAME : UNIVERSAL PROGRAM

CODING (INTEGER) OF SEPARATE CORRELATOR FUNCTIONS

PROGRAM-INSTRUCTIONS DEFINED

MEM-LOC.	COND.CODE	CODE-B	CODE-A	ADDR.	LC1	LCR1A	LC2	LC3	RELOAD	R-ADDR.
0	56	0	4	0	0	0	0	0	0	0
1	56	0	4	0	0	0	0	0	1	18
2	56	0	4	0	0	0	0	0	0	0
3	56	0	4	0	0	0	0	0	1	19
4	56	0	10	23	0	0	0	0	0	0
5	56	0	4	0	0	0	0	0	1	18
6	56	0	4	0	0	0	0	0	0	0
7	56	0	4	0	0	0	0	0	1	19
8	56	0	10	52	0	0	0	0	0	0
9	56	0	4	0	0	0	0	0	1	20
10	56	0	4	0	0	0	0	0	0	0
11	56	0	4	0	0	0	0	2	0	0
12	60	6	4	17	0	0	0	0	0	0
13	56	0	4	0	0	0	0	0	1	18
14	56	0	4	0	0	0	0	0	0	0
15	56	0	4	0	0	0	0	0	1	19
16	56	0	10	52	0	0	0	0	0	0
17	56	0	4	0	0	0	0	0	1	18
18	56	0	4	0	0	0	0	0	0	0
19	56	0	4	0	0	0	0	0	1	19
20	56	0	10	23	0	0	0	0	0	0
21	56	0	4	0	0	0	0	0	0	0
22	56	0	6	0	0	0	0	0	0	0
23	56	0	4	0	2	0	3	0	0	0
24	57	1	4	0	0	0	0	0	0	0
25	57	4	6	25	6	0	0	0	0	0
26	58	1	4	0	0	0	1	0	0	0
27	56	0	6	45	0	0	0	0	0	0
32	56	0	4	0	0	0	0	0	1	20
33	56	0	4	0	0	0	0	0	0	0
34	56	0	4	0	0	0	0	0	0	0
35	56	0	4	0	0	0	0	0	0	0
36	56	0	8	0	0	0	0	2	0	0
37	56	0	4	0	0	0	0	1	0	0
38	56	0	4	0	0	0	0	0	0	0
39	56	0	4	0	0	0	0	0	0	0
40	60	0	5	0	0	0	0	0	0	0
41	56	0	4	0	0	0	0	0	0	0
42	56	0	4	0	0	0	0	0	0	0
43	56	0	4	0	0	0	0	0	0	0
44	56	0	6	0	0	0	0	0	0	0
45	56	0	4	0	0	0	0	0	0	0
46	56	0	4	0	0	0	0	0	1	18
47	56	0	4	0	0	0	0	0	0	0
48	56	0	4	0	2	0	0	0	0	0
49	56	0	4	0	0	0	0	0	0	0
50	57	4	6	49	1	0	0	0	0	0
51	58	1	6	45	0	0	1	0	0	0
52	56	0	4	0	2	0	3	0	0	0
53	57	1	4	0	0	0	0	0	0	0
54	57	4	6	54	6	0	0	0	0	0
55	58	1	4	0	0	0	1	0	0	0
56	56	0	4	0	0	0	0	0	0	0
57	56	0	4	0	0	0	0	0	1	18
58	56	0	4	0	0	0	0	0	0	0

60	56	0	4	0	0	0	0	0	0	0
61	57	4	6	60	1	0	0	0	0	0
62	58	1	6	56	0	0	1	0	0	0

APB-INSTRUCTIONS DEFINED

MEM-LOC.	ALU-SOURCE	ALU-FUNCTION	ALU-DESTIN.	A-ADDR.	B-ADDR.	SELECT
0	7	5	1	0	0	0
1	4	0	1	15	0	0
2	7	5	1	0	0	0
3	4	0	1	14	0	0
4	4	0	0	9	0	0
5	4	0	1	8	0	0
6	7	5	1	0	0	0
7	4	0	1	7	0	0
8	4	0	0	5	0	0
9	4	0	1	4	0	0
10	7	5	1	0	0	0
11	7	5	1	0	0	0
12	7	5	1	0	0	0
13	4	0	1	8	0	0
14	7	5	1	0	0	0
15	4	0	1	3	0	0
16	4	0	0	2	0	0
17	4	0	1	1	0	0
18	7	5	1	0	0	0
19	7	5	1	0	0	0
20	4	0	0	0	0	0
21	7	5	1	0	0	0
22	7	5	1	0	0	0
23	0	1	3	13	11	0
24	7	5	1	0	0	0
25	1	0	3	13	11	0
26	7	5	3	0	11	0
27	7	5	1	0	0	0
32	7	0	1	0	0	0
33	7	5	1	0	0	0
34	7	5	1	0	0	0
35	7	5	1	0	0	0
36	7	5	1	0	0	0
37	7	5	1	0	0	0
38	7	5	1	0	0	0
39	7	5	1	0	0	0
40	7	5	1	0	0	0
41	7	5	1	0	0	0
42	7	5	1	0	0	0
43	7	5	1	0	0	0
44	7	5	1	0	0	0
45	1	0	3	12	11	0
46	1	2	1	15	11	0
47	7	5	1	0	0	0
48	0	1	3	13	10	0
49	1	0	3	13	10	0
50	1	0	1	11	10	0
51	7	5	1	0	0	0
52	0	1	3	13	11	0
53	7	5	1	0	0	0
54	1	0	3	13	11	0
55	7	5	3	0	11	0
56	1	0	3	6	11	0
57	1	2	1	8	11	0
58	7	5	1	0	0	0
59	0	1	3	13	10	0
60	1	0	3	13	10	0
61	1	0	1	11	10	0
62	7	5	1	0	0	0

APP-INSTRUCTIONS DEFINED

MEM-LOC.	ALU-SOURCE	ALU-FUNCTION	ALU-DESTIN.	A-ADDR.	B-ADDR.
0	7	5	1	0	0
1	7	5	1	0	0
2	7	5	1	0	0
3	7	5	1	0	0
4	7	5	0	0	0
5	7	5	1	0	0
6	7	5	1	0	0
7	7	5	1	0	0
8	0	0	0	15	0
9	7	5	1	0	0
10	7	5	1	0	0
11	7	5	1	0	0
12	7	5	1	0	0
13	7	5	1	0	0
14	7	5	1	0	0
15	7	5	1	0	0
16	0	0	0	15	0
17	7	5	1	0	0
18	7	5	1	0	0
19	7	5	1	0	0
20	0	0	0	15	0
21	0	0	0	15	0
22	7	5	1	0	0
23	0	1	0	15	0
24	7	5	1	0	0
25	0	0	0	15	0
26	7	5	1	0	0
27	7	5	1	0	0
32	7	5	1	0	0
33	7	5	1	0	0
34	7	5	1	0	0
35	7	5	0	0	0
36	0	1	0	0	0
37	0	0	0	0	0
38	2	0	1	0	0
39	2	0	1	0	0
40	2	0	1	0	0
41	7	5	1	0	0
42	7	5	1	0	0
43	7	5	1	0	0
44	7	5	1	0	0
45	7	5	1	0	0
46	7	5	1	0	0
47	7	5	1	0	0
48	7	5	1	0	0
49	7	5	1	0	0
50	0	0	0	15	0
51	7	5	1	0	0
52	0	1	0	15	0
53	7	5	1	0	0
54	0	0	0	15	0
55	7	5	1	0	0
56	7	5	1	0	0
57	7	5	1	0	0
58	7	5	1	0	0
59	7	5	1	0	0
60	7	5	1	0	0
61	0	0	0	15	0
62	7	5	1	0	0

ARITHMETICAL INSTRUCTIONS DEFINED

MEM-LOC.	M1A	M1B	M2A	M2B	M3A	M3B	M4A	M4B	SM1	SM2	SM3	SM4	M12	M34
0	0	0	0	0	0	0	0	0	0	0	0	0	15	15
1	0	0	0	0	0	0	0	0	0	0	0	0	15	15
2	0	0	0	0	0	0	0	0	0	0	0	0	15	15
3	0	0	0	0	0	0	0	0	0	0	0	0	15	15
4	0	0	0	0	0	0	0	0	0	0	0	0	15	15
5	0	0	0	0	0	0	0	0	0	0	0	0	15	15
6	0	0	0	0	0	0	0	0	0	0	0	0	15	15
7	0	0	0	0	0	0	0	0	0	0	0	0	15	15
8	0	0	0	0	0	0	0	0	0	0	0	0	15	15
9	0	0	0	0	0	0	0	0	0	0	0	0	15	15
10	0	0	0	0	0	0	0	0	0	0	0	0	15	15
11	0	0	0	0	0	0	0	0	0	0	0	0	15	15
12	0	0	0	0	0	0	0	0	0	0	0	0	15	15
13	0	0	0	0	0	0	0	0	0	0	0	0	15	15
14	0	0	0	0	0	0	0	0	0	0	0	0	15	15
15	0	0	0	0	0	0	0	0	0	0	0	0	15	15
16	0	0	0	0	0	0	0	0	0	0	0	0	15	15
17	0	0	0	0	0	0	0	0	0	0	0	0	15	15
18	0	0	0	0	0	0	0	0	0	0	0	0	15	15
19	0	0	0	0	0	0	0	0	0	0	0	0	5	15
20	0	0	0	0	0	0	0	0	0	0	0	0	15	15
21	0	0	0	0	0	0	0	0	0	0	0	0	12	12
22	0	0	0	0	0	0	0	0	0	0	0	0	15	15
23	0	0	0	0	0	0	0	0	0	0	0	0	15	15
24	0	0	0	0	0	0	0	0	0	0	0	0	15	15
25	0	0	1	1	1	0	0	1	3	3	3	3	9	6
26	0	0	0	0	0	0	0	0	0	0	0	0	15	15
27	0	0	0	0	0	0	0	0	0	0	0	0	15	15
32	0	0	0	0	0	0	0	0	0	0	0	0	5	15
33	0	0	0	0	0	0	0	0	0	0	0	0	15	15
34	0	0	0	0	0	0	0	0	0	0	0	0	15	15
35	0	0	0	0	0	0	0	0	0	0	0	0	15	15
36	0	0	0	0	0	0	0	0	0	0	0	0	15	15
37	0	0	0	0	0	0	0	0	0	0	0	0	15	15
38	0	0	0	0	0	0	0	0	0	0	0	0	15	15
39	0	0	0	0	0	0	0	0	0	0	0	0	15	15
40	0	0	0	0	0	0	0	0	0	0	0	0	15	15
41	0	0	0	0	0	0	0	0	0	0	0	0	15	15
42	0	0	0	0	0	0	0	0	0	0	0	0	15	15
43	0	0	0	0	0	0	0	0	0	0	0	0	15	15
44	0	0	0	0	0	0	0	0	0	0	0	0	15	15
45	0	0	0	0	0	0	0	0	0	0	0	0	15	15
46	0	0	0	0	0	0	0	0	0	0	0	0	15	15
47	0	0	0	0	0	0	0	0	0	0	0	0	15	15
48	0	0	0	0	0	0	0	0	0	0	0	0	15	15
49	0	0	1	1	1	0	0	1	1	1	1	1	9	6
50	0	0	1	1	1	0	0	1	2	2	2	2	9	6
51	0	0	0	0	0	0	0	0	0	0	0	0	15	15
52	0	0	0	0	0	0	0	0	0	0	0	0	15	15
53	0	0	0	0	0	0	0	0	0	0	0	0	15	15
54	0	0	1	1	1	0	0	1	3	3	3	3	9	6
55	0	0	0	0	0	0	0	0	0	0	0	0	15	15
56	0	0	0	0	0	0	0	0	0	0	0	0	15	15
57	0	0	0	0	0	0	0	0	0	0	0	0	15	15
58	0	0	0	0	0	0	0	0	0	0	0	0	15	15
59	0	0	0	0	0	0	0	0	0	0	0	0	15	15
60	0	0	1	1	1	0	0	1	1	1	1	1	9	6
61	0	0	1	1	1	0	0	1	2	2	2	2	9	6
62	0	0	0	0	0	0	0	0	0	0	0	0	15	15

ACCUMULATOR INSTRUCTIONS DEFINED

MEM-LOC.	STROBE I/O	WRITE	READ	CLEAR1	SET1	CLEAR2	SET2
0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0
2	1	0	0	0	0	0	0
3	1	0	0	0	0	0	0
4	1	0	0	0	0	0	0
5	1	0	0	0	0	0	0
6	1	0	0	0	0	0	0
7	1	0	0	0	0	0	0
8	1	0	0	0	0	0	0
9	1	0	0	0	0	0	0
10	1	0	0	0	0	0	0
11	1	0	0	0	0	0	0
12	1	0	0	0	0	0	0
13	1	0	0	0	0	0	0
14	1	0	0	0	0	0	0
15	1	0	0	0	0	0	0
16	1	0	0	0	0	0	0
17	1	0	0	0	0	0	0
18	1	0	0	0	0	0	0
19	1	0	0	0	0	0	0
20	1	0	0	0	0	0	0
21	1	1	1	1	0	0	0
22	1	0	0	0	0	0	1
23	1	0	0	1	0	0	0
24	1	0	0	0	0	0	0
25	1	1	1	0	0	0	0
26	1	0	0	0	0	0	0
27	1	0	0	0	0	0	0
32	0	0	0	0	0	1	0
33	0	0	0	0	0	0	0
34	0	0	0	0	0	0	0
35	0	0	0	0	0	0	0
36	0	0	0	0	0	0	0
37	0	0	0	0	0	0	0
38	0	0	0	0	0	0	0
39	0	0	0	0	0	0	0
40	0	0	0	0	0	0	0
41	0	0	0	0	0	0	0
42	0	0	0	0	0	0	0
43	0	0	0	0	0	0	0
44	0	0	0	0	0	0	0
45	1	0	0	0	0	0	0
46	1	0	0	0	0	0	0
47	1	0	0	0	0	0	0
48	1	0	0	0	0	0	0
49	1	0	0	0	0	0	0
50	1	1	1	0	0	0	0
51	1	0	0	0	0	0	0
52	1	0	0	1	0	0	0
53	1	0	0	0	0	0	0
54	1	1	1	0	0	0	0
55	1	0	0	0	0	0	0
56	1	0	0	0	0	0	0
57	1	0	0	0	0	0	0
58	1	0	0	0	0	0	0
59	1	0	0	0	0	0	0
60	1	0	0	0	0	0	0
61	1	1	1	0	0	0	0
62	1	0	0	0	0	0	0

I/O INSTRUCTIONS DEFINED

MEM-LOC.	SETF	CLEARF	SELECT	BUF.ADDR.	STROBE	I-REG. ENABLE	EOB ENABLE	EAB
0	0	0	0	0	0	0	0	
1	0	0	0	0	0	0	0	
2	0	0	0	0	0	0	0	
3	0	0	0	0	0	0	0	
4	0	0	0	0	0	0	0	
5	0	0	0	0	0	0	0	
6	0	0	0	0	0	0	0	
7	0	0	0	0	0	0	0	
8	0	0	0	0	0	0	0	
9	0	0	0	0	0	0	0	
10	0	0	0	0	0	0	0	
11	0	0	0	0	0	0	0	
12	0	0	0	0	0	0	0	
13	0	0	0	0	0	0	0	
14	0	0	0	0	0	0	0	
15	0	0	0	0	0	0	0	
16	0	0	0	0	0	0	0	
17	0	0	0	0	0	0	0	
18	0	0	0	0	0	0	0	
19	0	0	0	0	0	0	0	
20	0	0	0	0	0	0	0	
21	0	0	0	0	0	0	0	
22	0	0	0	0	0	0	0	
23	0	0	0	0	0	0	0	
24	0	0	0	0	0	0	0	
25	0	0	0	0	0	0	0	
26	0	0	0	0	0	0	0	
27	0	0	0	0	0	0	0	
32	0	0	0	0	0	0	0	
33	0	0	0	0	0	0	0	
34	0	0	0	0	0	0	0	
35	0	0	0	0	0	0	0	
36	0	0	0	0	0	0	0	
37	0	0	0	0	0	0	0	
38	0	0	0	0	0	0	0	
39	0	0	0	0	0	0	0	
40	0	0	0	0	0	0	0	
41	0	0	0	0	0	0	0	
42	0	0	0	0	0	0	0	
43	0	0	0	0	0	0	0	
44	0	0	0	0	0	0	0	
45	0	0	0	0	0	0	0	
46	0	0	0	0	0	0	0	
47	0	0	0	0	0	0	0	
48	0	0	0	0	0	0	0	
49	0	0	0	0	0	0	0	
50	0	0	0	0	0	0	0	
51	0	0	0	0	0	0	0	
52	0	0	0	0	0	0	0	
53	0	0	0	0	0	0	0	
54	0	0	0	0	0	0	0	
55	0	0	0	0	0	0	0	
56	0	0	0	0	0	0	0	
57	0	0	0	0	0	0	0	
58	0	0	0	0	0	0	0	
59	0	0	0	0	0	0	0	
60	0	0	0	0	0	0	0	
61	0	0	0	0	0	0	0	
62	0	0	0	0	0	0	0	

OUTPUT-TRANSFER INSTRUCTIONS DEFINED

MEM-LOC.	TRANSFER	INHIBIT	CLOCK	DATA-READY	TRANSFER-CODE	SOURCE
0	0	0	0	0	0	0
1	0	0	0	0	0	0
2	0	0	0	0	0	0
3	0	0	0	0	0	0
4	0	0	0	0	0	0
5	0	0	0	0	0	0
6	0	0	0	0	0	0
7	0	0	0	0	0	0
8	0	0	0	0	0	0
9	0	0	0	0	0	0
10	0	0	0	0	0	0
11	0	0	0	0	0	0
12	0	0	0	0	0	0
13	0	0	0	0	0	0
14	0	0	0	0	0	0
15	0	0	0	0	0	0
16	0	0	0	0	0	0
17	0	0	0	0	0	0
18	0	0	0	0	0	0
19	0	0	0	0	0	0
20	0	0	0	0	0	0
21	0	0	0	0	0	0
22	0	0	0	0	0	0
23	0	0	0	0	0	0
24	0	0	0	0	0	0
25	0	0	0	0	0	0
26	0	0	0	0	0	0
27	0	0	0	0	0	0
32	0	0	0	0	0	0
33	0	0	0	0	0	0
34	1	0	0	0	0	0
35	1	1	1	0	0	0
36	1	1	1	1	0	0
37	1	1	1	3	0	0
38	1	1	1	2	0	0
39	1	1	1	5	0	0
40	1	1	1	4	0	0
41	1	0	0	0	0	0
42	1	0	0	0	0	0
43	1	0	0	0	0	0
44	1	0	0	0	0	0
45	0	0	0	0	0	0
46	0	0	0	0	0	0
47	0	0	0	0	0	0
48	0	0	0	0	0	0
49	0	0	0	0	0	0
50	0	0	0	0	0	0
51	0	0	0	0	0	0
52	0	0	0	0	0	0
53	0	0	0	0	0	0
54	0	0	0	0	0	0
55	0	0	0	0	0	0
56	0	0	0	0	0	0
57	0	0	0	0	0	0
58	0	0	0	0	0	0
59	0	0	0	0	0	0
60	0	0	0	0	0	0
61	0	0	0	0	0	0
62	0	0	0	0	0	0

APPENDIX B. PULSE TO PULSE CORRELATION

Proposals for mesospheric experiments using pulse to pulse correlation techniques have been made. For example, the EISCAT technical Note: No. 82/35, A Mesospheric Experiment: Measurement Scheme and Program Implementation by W Kofman. This section describes how the Universal Program can be applied to this measuring technique.

The transmitter sends the first pulse at time t=0 and successive complex samples $Z_{0,A1}, Z_{0,A2}, Z_{0,A3}, \ldots, Z_{0,AN}$, where the first subscript refers to the first pulse and the second subscript Ai refers to the ith altitude. The second transmission occurs after a few milliseconds and the receiver is opened to take the new data $Z_{1,A1}, Z_{1,A1}, Z_{1,A2}, \ldots, Z_{1,AN}$. After G pulses are transmitted the START COMPUTE signal to the correlator is given. The desired autocorrelation function for each altitude is given by:

$$K_{Ai}(L) = \sum_{j=0}^{G-L-1} Z_{j,Ai} Z^{*}_{j+L,Ai} \qquad 0 \leq L \leq G-1$$

where * denotes the complex conjugate.

For the Universal Program the necessary parameters are:

No. of Samples-1 = Number of Altitudes×No. of Pulses-1
$(N \times G-1)$

No of Lag Profiles-1 = L $\quad 0 \leq L \leq G-1$

Lag Increment = Number of Altitudes (N)

If we were to consider the samples written into the buffer memory as $Z_0, Z_1, Z_2, \ldots, Z_{N-1}, Z_N, Z_{N+1}, \ldots, Z_{2N-1}$, etc where N denotes the number of samples taken from each pulse transmission, ie Nth altitude. Then from the basic cross product matrix the Universal Program will compute the following diagonals as lag profiles: Lag Profile(0)=Z_0^2 diagonal, Lag Profile(1)=$Z_0 Z_N$ diagonal, Lag Profile(2)=$Z_0 Z_{2N}$ diagonal, etc. The cross products corresponding to a particular altitude in each diagonal will have an index separation of N. Thus, for Lag Profile(0) the cross products for the 1st altitude will be Z_0^2, Z_N^2, Z_{2N}^2, etc. For Lag Profile(1) the cross products for the 1st altitude will be $Z_0 Z_N, Z_N Z_{2N}, Z_{2N} Z_{3N}$, etc.

Thus it can be seen that the Universal Program is very easy to apply to the pulse to pulse correlation technique. The decoding of the relevent cross products for each altitude is very straightfoward as they just have an index increment of N.

It should be noted that the formula given for the desired autocorrelation assumes triangular weighting. A better use of all the information given by the Universal Program would be to construct an autocorrelation function which has block weighting. This is given by the formula:

$$K_{Ai}(L) = \sum_{j=O}^{P} Z_{j,Ai} Z^{*}_{j+L,Ai}$$

where * denotes the complex conjugate.

L=O,1,2,...,N-1

N=No. of Lags

G=No. of Pulses Transmitted

P=G-N