82/34

EISCAT
TECHNICAL
NOTE

THE EISCAT CORRELATOR

by

Régis Gras

KIRUNA
Sweden

THE EISCAT CORRELATOR

Régis Gras

EISCAT Scientific Association
S-981 27 Kiruna, Sweden
March 1982

EISCAT Technical Note 82/34
Printed in Sweden
ISSN 0349-2710

This note is an edited translation of "Programmation du Correlateur d'EISCAT" by Régis Gras published by CEPHAG, Grenoble.

CONTENTS

Note : This technical note was first published in French in June 1980 "to make EISCAT users familiar with the programming and operation of the EISCAT correlator." It succeeded so well in this aim that I thought a translation would be valuable. In preparing this translation I received considerable help from Kristen Folkestad, Hans-Jorgen Alker and Régis Gras. The original contained a section on the development tools available for the writing and implementation of correlator microprograms. The early version of CORRSIM has been replaced by an entirely new version which, together with the program CORRTEST, is described in EISCAT Technical Note 81/25. The section on program development has therefore been omitted from this translation. Phil Williams (Ed.)

THE EISCAT CORRELATOR

The EISCAT correlator is a semi-specialised hardware device.

- specialised because it has been constructed to calculate expressions of the form

$$\sum_{i=0}^{N} x_i \, x_{i-j} \qquad (x_i \text{ and } x_{i-j} \text{ are complex samples})$$

- semi-specialised because it is not set in a hard-wired configuration, but can be microprogrammed so that a certain versitility is available in the type of calculation and the way in which it is carried out.

1. OUTLINE DESCRIPTION

This correlator includes a buffer memory, where sampled data are stored before processing, and a result memory where the results are stored.

The correlator is not isolated but forms part of a system which also includes:

- the radar controller

- a Nord-10 computer from Norsk Data.

The correlator is a multiprocessor. For different reasons, especially the need for speed, the correlator is a multiprocessor in which each micro-instruction can carry out 7 operations in parallel.

The correlator is microprogrammable and possesses a program memory of 64 words, each 128 bits, and different fields of data.

1.1 The Buffer Memory and Result Memory

1.1.1 The Buffer Memory.

The Buffer memory is divided into two symmetric parts which can simultaneously be read by the correlator and written in by the A/D-converter. Each part of this memory holds 16-bit words with a maximum capacity of 64 kilowords (for the time being the capacity is 4 kilowords). It is in this memory that the sampled measurements are stored; each sample is stored as 2 8-bit numbers, X being sampled from the in-phase channel and Y from the quadrature channel.

As soon as the correlator has finished its calculations on the part of the memory it has just read, it switches to the other part of the memory which has just received data. The part which previously was being read now receives new data. This permutation is conducted by the radar controller.

1.1.2 The Result Memory

The result memory holds 2048 64-bit words, and is designed with the possibility for a doubling of the present capacity. It is here that the correlator stores the correlation functions it has calculated. Each word in this memory is made up of:

- 32 bits for the real part of the result and
- 32 bits for the imaginary part.

The real and imaginary parts are stored together at the same address.

1.2 Links with other equipment.

1.2.1 Links with the Nord-10

The correlator is linked to Nord-10 via Direct Memory Access: DMA, and via a NON-DMA I/O-port.

The links make use of CAMAC. (CAMAC is an international standard for electrical connection between different pieces of equipment).

These links serve:

- to transfer data from the result memory to the computer memory via the high-speed DMA channel
- to load the programmable memories of the correlator from the Nord-10 (cf. 1.4.1 and 1.4.2) via the low-speed CAMAC output port.

1.2.2 Links with the Radar Controller

As it has been stated previously, the radar controller controls the buffer memory of the correlator. The radar controller gives the following commands:

- start sampling
- switch buffer memory
- start computation.

1.3 The Correlator as a Multiprocessor

It is capable of carrying out 7 operations in parallel. These 7 simultaneous operations make up one step in the basic computation.

In general, each basic computation can be broken down into:

- a choice of the operand in the buffer memory
- an arithmetic calculation on the operand selected
- storing of the result, with or without accumulation, in the result memory.

A complete calculation, carried out by the correlator, is made up of a certain number of basic operations. The control of these basic computations is carried out by the loop counters.

The 5 parallel functions of the correlator are:

- calculation of the buffer memory address (APB Processor)
- control of the arithmetic unit
- calculation of the result memory address (APM Processor)
- accumulation of the results in the result memory
- control of the loop counters (i.e. control of the running microprogram, 4.2.1).

To these must be added two other functions:

- control of the DMA transfer to the computer
- control of the communications between correlators (assuming a multicorrelator system which does not yet exist).

1.4 The Different Parts of the Correlator Memory.

To be executed, every program needs

- instructions
- data.

In what we normally call a program, written in a high-level or an assembly language, instructions and data can be stored in the same memory. For the correlator, however, the instructions and the parameters are stored physically in different hardware memories, serving as program-memory and parameter-memory. The last is actually a set of 2 register stacks.

1.4.1 The Program Memory

This can contain 64 instructions, each occupying 128 bits. A single instruction is divided into 7 fields, each field being a single processor instruction. The 7 fields are executed simultaneously.

Restriction:

Of the 64 words in the program memory, only 62 can be used

- the word 0 in the memory is used as an idle loop at the start of the calculation
- the words 63 is used in the control of external interrupts.

1.4.2 The Register Stacks

It is in these registers that the operands for the different processors are stored.

There are two fields for the principle parameters:

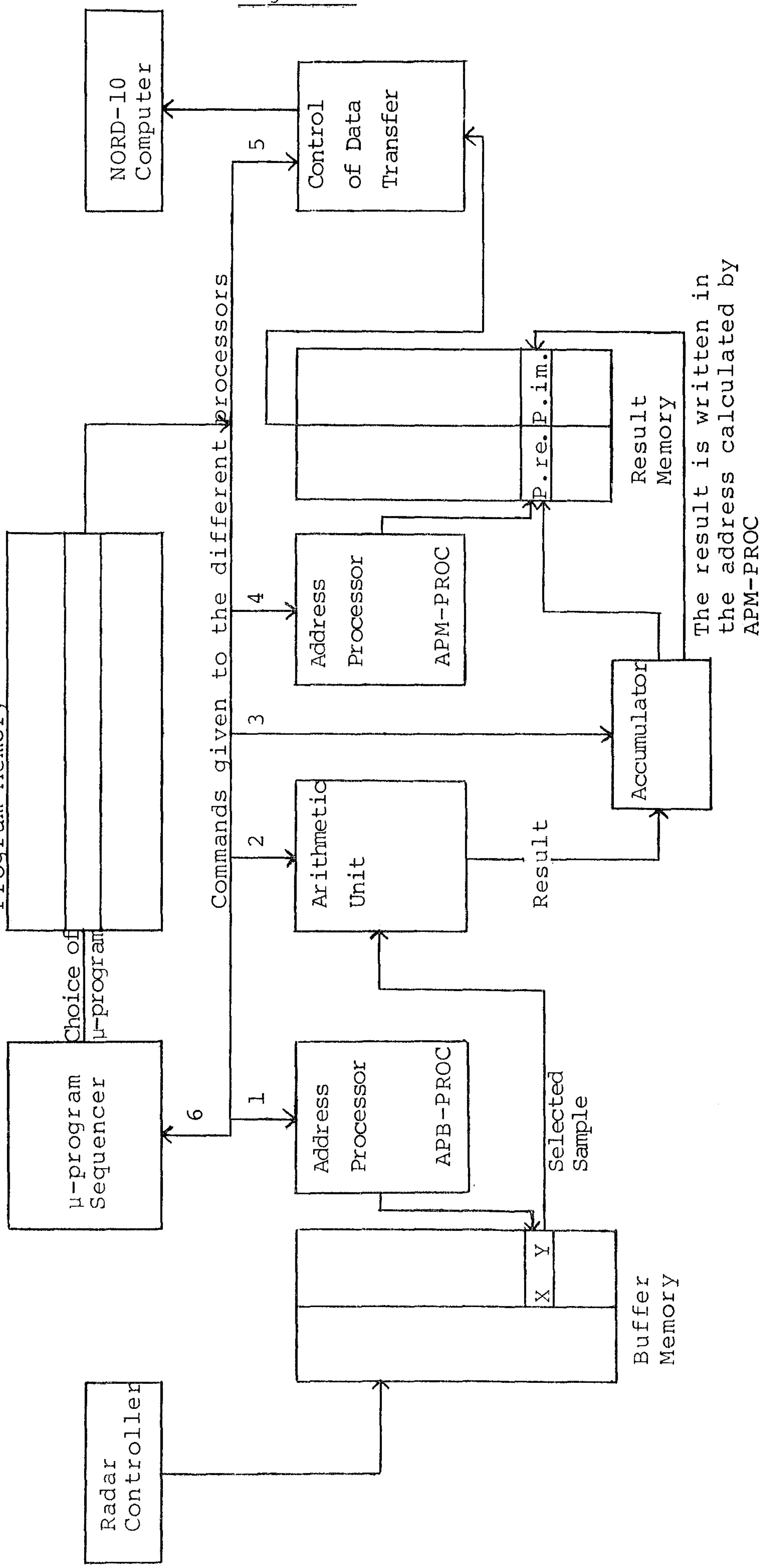
- the APB stack: a stack of 16 registers serving as a memory for the APB-processor (the processor which controls the addresses in the buffer memory; cf 3) and optionally reloads the loop-counters.

- the APM-stack: a stack of 16 registers identical to the previous stack and serving as a memory for the APM-processor (the processor which controls the addresses in the result memory; cf 3)

Note : there are also three registers to initialise
- the status register
- the SAR register (start address of the program)
- the DATA-I-register (containing the number of words to be transferred to the computer).

This introductory description is summarised in Fig.1.

Figure 1


NORD-10 Computer
Control of Data Transfer
5
Commands given to the different processors
Program Memory
Choice of μ-program
μ-program Sequencer
6
1
2
3
4
Address Processor APB-PROC
Arithmetic Unit
Address Processor APM-PROC
Accumulator
Result
Selected Sample
P.re.
P.im.
Result Memory
The result is written in the address calculated by APM-PROC
X
Y
Buffer Memory
Radar Controller

2. OPERATION AND PROGRAMMING

The different processors in the correlator are now described from two viewpoints:

- their method of operation
- their programming.

Each microinstruction of 128 bits is divided into 7 fields which each controls a particular function. During the execution of a microinstruction, the seven functions are carried out in parallel.

The internal division of a single microinstruction is given in Fig. 2.

6 bits	8 bits	7 bits	36 bits	17 bits	18 bits	34 bits
I/Ø	ØUT	ACC	ARI	APM	APB	PRØ

Figure 2.

PRØ: controls the execution of the program in the correlator and permits the loading of the programmable registers

APB: controls the addressing of data in the buffer memory

APM: controls the addressing of data in the result memory

ARI: controls the arithmetic operations carried out on the data

ACC: controls the accumulation of the results calculated at a given instant with those already stored in the result memory

ØUT: carries out the transfer of data from the result memory to the computer by DMA

I/Ø: would control communication in a multicorrelator system (not in use).

<u>Figure 3.1</u>

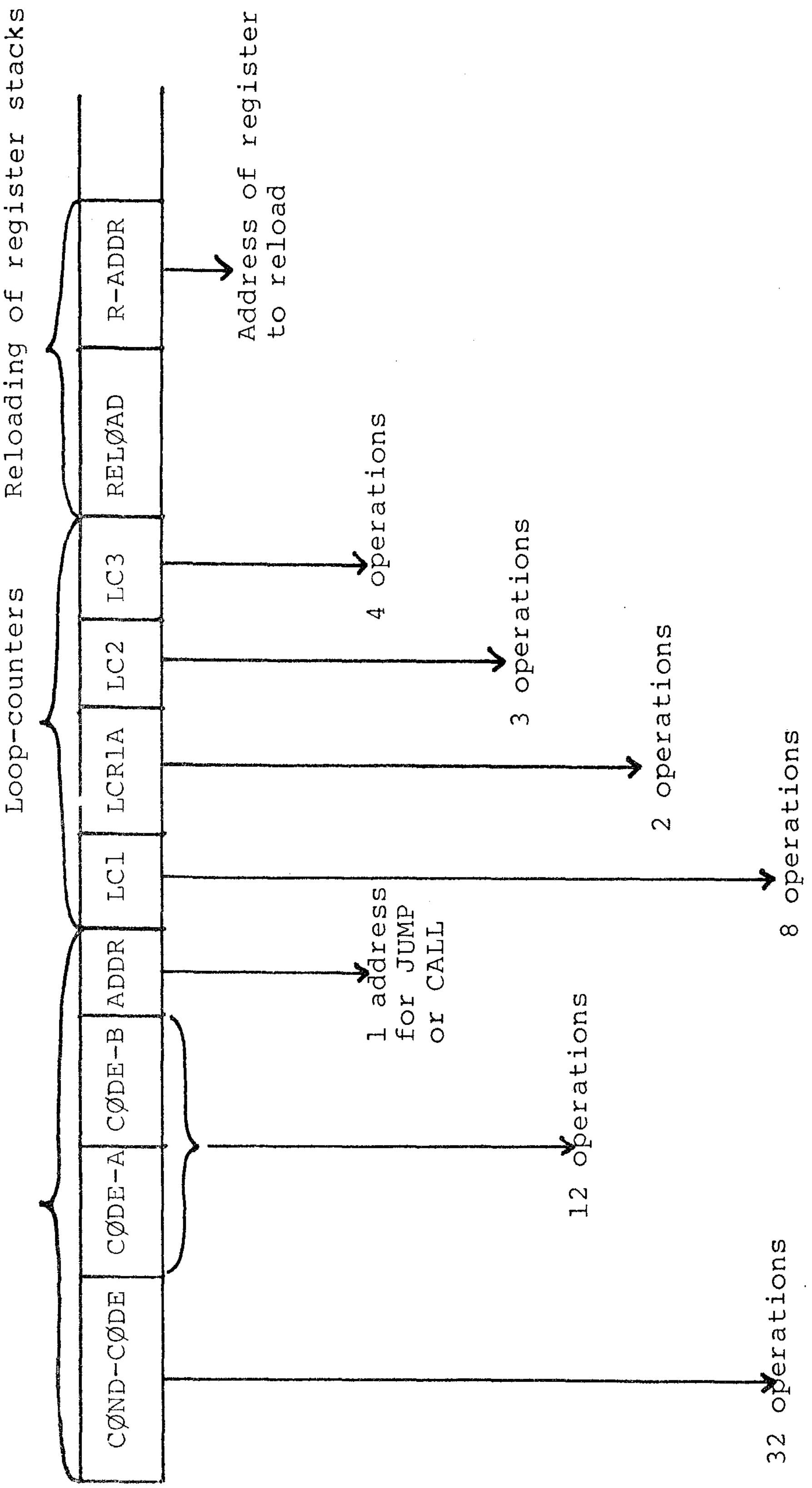

Reloading of register stacks
Loop-counters
R-ADDR
RELØAD
LC3
LC2
LCR1A
LC1
ADDR
CØDE-B
CØDE-A
CØND-CØDE
Address of register to reload
4 operations
3 operations
2 operations
8 operations
1 address for JUMP or CALL
12 operations
32 operations

Figure 3,2

Control of the Program Counter

3	(IF LC1≠0 THEN B ELSEIF LC2≠0 THEN A OTHERWISE CONT)
6	(IF LC3≠0 THEN B ELSEIF LC2≠0 THEN A OTHERWISE CONT)
7	(IF LC1≠0 OR LC3≠0 THEN B ELSEIF LC2≠0 THEN A OTHERWISE CONT)
13	(IF LC1≠0 THEN B ELSEIF LC2=0 THEN A OTHERWISE CONT)
15	(IF LC1≠0 OR LC3≠0 THEN B OTHERWISE CONT)
16	(IF LC3≠0 THEN B ELSEIF LC2=0 THEN A OTHERWISE CONT)
17	(IF LC1≠0 OR LC3≠0 THEN B ELSEIF LC2=0 THEN A OTHERWISE CONT)
23	(IF LC1=0 THEN B ELSEIF LC2≠0 THEN A OTHERWISE CONT)
26	(IF LC3=0 THEN B ELSEIF LC2≠0 THEN A OTHERWISE CONT)
27	(IF LC1=0 OR LC3=0 THEN B ELSEIF LC2≠0 THEN A OTHERWISE CONT)
33	(IF LC1=0 THEN B ELSEIF LC2=0 THEN A OTHERWISE CONT)
35	(IF LC1=0 OR LC3=0 THEN B OTHERWISE CONT)
36	(IF LC3=0 THEN B ELSEIF LC2=0 THEN A OTHERWISE CONT)
37	(IF LC1=0 OR LC3=0 THEN B ELSEIF LC2=0 THEN A OTHERWISE CONT)
40	(USE-A)
46	(IF LC2≠0 OR LC3≠0 THEN B ELSE A)
47	(IF LC1=0 OR LC2≠0 OF LC3≠0 THEN B ELSE A)
54	(IF LC3≠0 THEN B ELSE A)
55	(IF LC1=0 OR LC3≠0 THEN B ELSE A)
56	(IF LC2=0 OR LC3≠0 THEN B ELSE A)
57	(IF LC1=0 OR LC2=0 OR LC3≠0 THEN B ELSE A)
62	(IF LC2≠0 THEN B ELSE A)
63	(IF LC1=0 OR LC2≠0 THEN B ELSE A)
66	(IF LC2≠0 OR LC3=0 THEN B ELSE A)
67	(IF LC1=0 OR LC2≠0 OR LC3=0 THEN B ELSE A)
71	(IF LC1=0 THEN B ELSE A)
72	(IF LC2=0 THEN B ELSE A)
73	(IF LC1=0 OR LC2=0 THEN B ELSE A)
74	(IF LC3=0 THEN B ELSE A)
75	(IF LC1=0 OR LC3=0 THEN B ELSE A)
76	(IF LC2=0 OR LC3=0 THEN B ELSE A)
77	(IF LC1=0 OR LC2=0 OR LC3=0 THEN B ELSE A)

Different possible tests

0	PC=PC+1, POP STACK
1	PC=RETURN ADDR., POP STACK
2	PC= ADDR., POP STACK
3	PC= SAR, POP STACK
4,14	PC=PC+1
5,15	PC=RETURN ADDR.
6,16	PC=ADDR.
7,17	PC=SAR
10	PC= PC+1, PUSH STACK
11	PC= RETURN ADDR., PUSH STACK
12	PC= ADDR., PUSH STACK
13	PC= SAR, PUSH STACK

Figure 3,3

Operations on the Loop Counters

0 NOOP
1 LC1=LC1-1
2 LC1=LCR1
3 LC1=LCR1A
4 IF LCL=0: LC1=LCR1,LC2=
LC2-1, ELSE: LC1=LC1-1
5 IF LC1=0 and LC3=0:LC1=
LCR1A, ELSE: LC1=LC1-1
6 IF LC1=0: LC1=LCR1,
ELSE: LC1=LC1-1
7 IF LC1=0: LC1=LCR1A,
ELSE: LC1=LC1-1

LC1

0 NOOP
1 LCR1A=LC1

LCR1A

0 NOOP
1 LC2=LC2-1
3 LC2=LCR2

LC2

0 NOOP
1 LC3=LC3-1
2 LC3=LCR3
3 IF LC3=0: LC3=LCR3,
ELSE: LC3=LC3-1

LC3

Reloading of a register

RELOAD: 0 NOOP
1 REGISTER-RELOAD

R-ADDR.: 4 RELOAD SAR
5 " BAR, APB
22 " LCR1
23 " LCR2
24 " LCR3

The reloading is only implemented if the bit RELØAD equals 1

2.1 The PRØ-instructions (Fig. 3)

This field controls:

- the loop-counters
- the loading of the programmable registers from the APB
- the program counter.

2.1.1 The Loop-Counters (see 2.1.2)

To carry out a calculation, the correlator has 3 12 bits loop-counters. These are: LC1, LC2, LC3 and a temporary register LCR 1A.

The 3 loop-counters are not identical; in particular, only the first, LC1, can be re-loaded from LCR 1A. In the field of the PRØ-instruction, 4 sub-fields indicate the operation to be effected on LC1, ... LC3, LCR 1A.

2.1.2 Method of loading the loop-counters

To control the number of loops in a program, the loop-counters are tested at 0. Now the only operation that can be performed on a loop-counter is to decrease it, so it is necessary to load the loop-counters with their initial values.

These initial values are contained in three "load registers for loop-counters" LCR1, LCR2, LCR3. These three registers can themselves be loaded from the values stored in the APB-Stack memory (cf. 1.2.2). Once these registers are loaded, their contents can be transferred at will to the loop-counters. The mechanism is represented in Fig. 4.

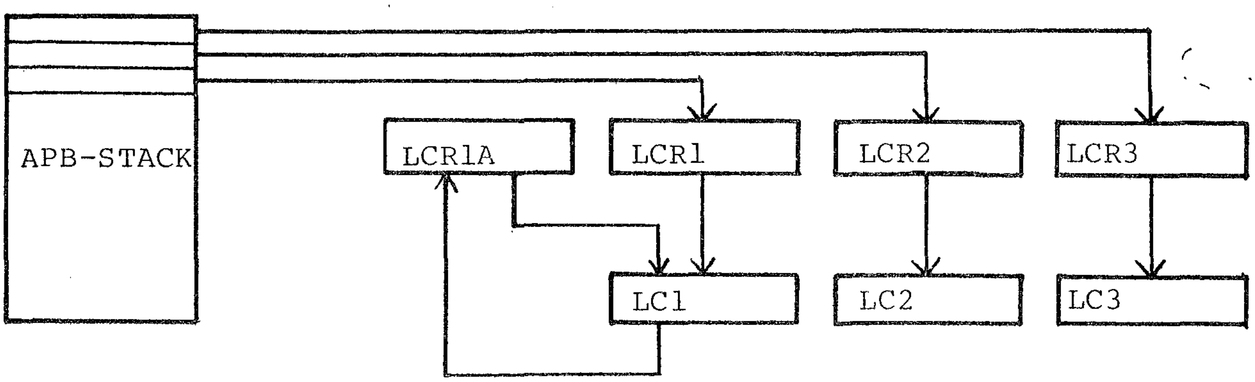


Figure 4.

Note: Although the 3 loop-counters can be loaded from the LCRs by a single microinstruction, this does not apply to the loading of the LCRs themselves. The loading of an LCR with a value stored in the APB-Stack requires 3 microinstructions:

- one to program the reloading of an LCR
- two NOOP (for the correlator needs two clock cycles after a reload).

2.1.3 Control of the Program Counter

During the execution of a program, the program counter (PC) takes different values corresponding to different instructions.

The value at any instant t is a function of:

- its value at the instant t-1
- the result of the instruction executed at t-1.

Generally, from one step in a program to the next the program counter is increased by 1 (PC = PC + 1), except when the instruction being executed is a branch instruction (conditional or not).

Example:

```
      I = 1                  0
10    IF (I.EQ.2) GO TO 20   1
      I = I + 1              2
      GO TO 10               3
20    CONTINUE               4
```

Given a machine where one instruction only occupies one line and they are stored at addresses 0, 1, 2, 3 and 4, the values of PC would be:

t = 0	PC = 0
t = 1	PC = 1
t = 2	PC = 2
t = 3	PC = 3
t = 4	PC = 1
t = 5	PC = 4

In a program language at the lowest level (assembler), 2 types of instruction can be distinguished

- those which increase the program counter (after their execution the computer passes to the next instruction)
- those which load the program counter with an other address (conditionally branching or not); it is this latter type of instruction which allows loops in a program.

The mechanism of conditional branching in the correlator programs is a little different in the sense that the tests and the operations on the counter are separate.

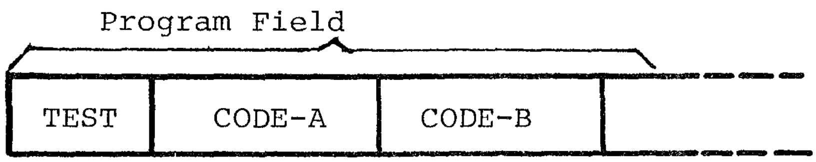
In the instruction field of the correlator there are two sub-fields called CODE-A and CODE-B (cf. Fig. 3.1).

Each one receives the code of an operation possible on the program counter.

A third sub-field contains the code of one of the 32 possible tests on the loop-counters (listed in Fig. 3.2). It is the result of this test which determines the choice between code A and code B.

The mechanism is summarised in Fig. 5.

Instruction directed by PC at the instant C

t: PC 

Code instruction = Code A or Code B } as a function of the test

t + 1: New PC = f (code instruction)

Figure 5.

Programming of the Counter

To facilitate the programming of the loops and to allow microprograms to have the structure of sub-routines, the correlator makes use of a stack.

This stack consists of 4 memories and it is carried out according to the LIFO principle (last in, first out).

At a given instant, only the last value loaded in the stack can be referred to. The set of instructions that can be programmed

- controls this stack

  a) safe guarding in the stack the value of PC + 1 (the return address when calling a subroutine)

  b) destroying the last return address in the stack.

  c) branching to the last return address in the stack

The complete list of instructions is given in Fig. 3.2.

- permits conditional branching or continuing in sequence.

Programming Test

As has already been mentioned, the 32 possible tests (cf. Fig. 3.2) are tests on the value of the loop-counters. No particular value can be tested - only if it is zero or not.

There are two types of test; simple and double, which allow a choice between CODE-A and CODE-B.

For simple tests, the choice is made as follows: if the test is true, use CODE-B; otherwise use CODE-A.

Example:

IF LC1 = 0 THEN B ELSE A (code test 71)
IF LC1 = 0 OR LC2 ≠ 0 THEN B ELSE A (code test 63).

For double tests, if test 1 is true, then use CODE-B; otherwise, if test 2 is true, then use CODE-A, otherwise continue in sequence:

Example:

Test 1	Test 2
IF LC3 ≠ 0 THEN B	ELSE IF LC2 ≠ 0 THEN A

OTHERWISE CONTINUE.

Restriction:

The instruction field (cf. Fig. 3) only allows a single address for branching, so that only one of the two codes A or B can be a branching instruction.

Note:

In the program field, operations on the loop-counters and tests on their value are carried out. Each operation occurs in two stages:

- first the test on the value of the counter
- afterwards the operation on the counter.

2.2 The APB - APM Instructions

2.2.1 General

These are the instructions which control the addressing of the buffer memory and the result memory. The instructions are identical, but they concern two different processors, operating in parallel.

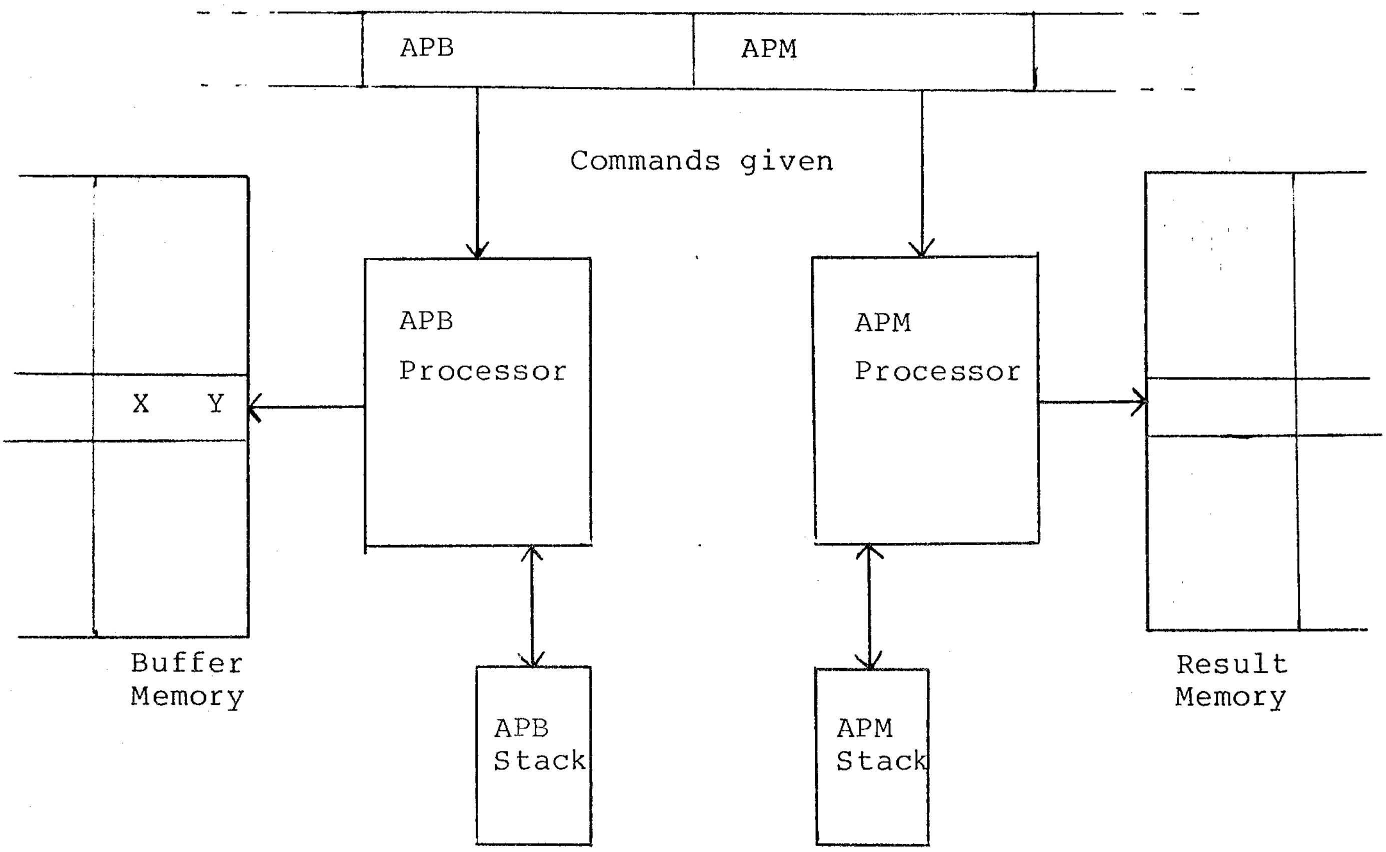


Figure 6.

Each of the two processors has its own parameter memory containing 16 registers, the APB-Stacks and the APM-Stacks.

Note

- the addressing capacity of the APM is 4 K of 64 bits words (only 2 K words are implemented at present with 32 + 32 = real and maginary) and that of the APB 64 K 16-bits words (4 K words implemented at present, 8 bits for each sample)
- an address in the buffer memory concerns the two samples X and Y

- an address in the result memory concerns the real and imaginary part of the result
- each element in the parameter memory APB-STACK and APM-STACK, is accessed like a table according to its index or address in the table.

  The instruction fields APM and APB only contain two addresses of operands in the parameter memory. At a given instant, only two elements in the APB-STACK or APM-STACK memories can be addressed in the same micro-instruction. In addition to the 16 registers of the APB/APM processors, each processor has an extra programable register for temporary storage.

Terminology

The terminology used in EISCAT Technical Notes is as follows:

- R and S are operands (selectable) in each of the parameter memories
- A and B are the addresses of these operands
- the parameter memory is called RS, so that an operand is: R = RS (A) or S = RS (B)
- the programmable register is the Q-REGISTER.

2.2.2 Use of APB and APM Processors

The calculation of an address by a processor can be broken down into three stages:

a) choice of an operand (ALU-SOURCE). This operand can be RS (A), RS (B), Q, O, DATA-I-REGISTER

b) calculation on this operand (ALU-FUNCTION) For the two chosen operands, we can program addition, subtraction or operations of less obvious interest such as OR, AND, exclusive OR etc.

b) use of the result

The result of the calculation effected on the operands serves to address the buffer memory APB processor) or the result memory (APM processor). In the EISCAT literature this address is called OUT.

However, we should distinguish:

OUT which is the address in the memory

F which is the result of the calculation itself.

In general, this result:

- serves as an address in the memory: OUT = F

and

- can be stored:

  - . in the parameter memory : RS (B) = F

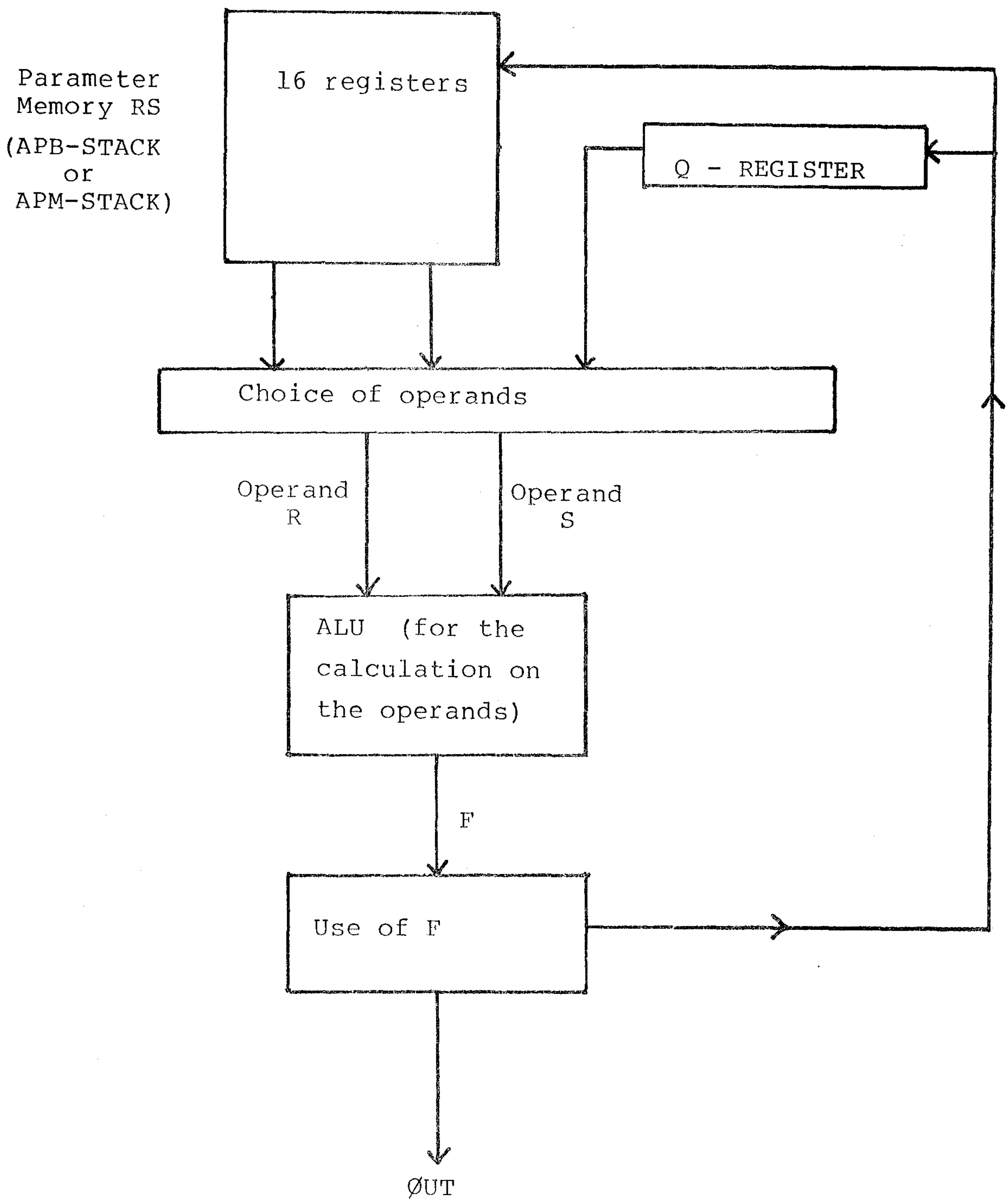
    or

  - . in the Q-Register : Q = F

The possibility of storing temporary results allows registers to be used as base-addresses for calculating the next address in a vector-addressing scheme.

The mechanism is illustrated in Fig. 7

Figure 7


Parameter Memory RS
(APB-STACK or APM-STACK)
16 registers
Q - REGISTER
Choice of operands
Operand R
Operand S
ALU (for the calculation on the operands)
F
Use of F
ØUT

2.3 The Arithmetic Instructions

In the buffer memory, each 16-bit sample is considered as a complex sample with 8-bits representing the real part X and the imaginary part Y.

The arithmetic unit of the correlator is designed to carry out the complex product:

$$(a_1 + ib_1) \times (a_2 + ib_2) = (a_1a_2 - b_1b_2) + i(a_1b_2 + a_2b_1)$$

To calculate this product as quickly as possible, the arithmetic unit carries out 4 multiplications in parallel (cf. Fig. 8).

- each multiplier has two inputs, called A & B
- the multipliers can be supplied either with an internal sample (from the buffer memory) or with an external value (for correlator test programs).

Each arithmetic operation occurs in two stages:

a) choice of operand for each input to each multiplier; given the multiplier 1, for example, we can choose for the input A:

The value $X_{internal}$ or $Y_{internal}$ (the pair X, Y being at an address calculated by the APB processor)

The value $X_{external}$ or $Y_{external}$ (test).

This choice must be made for the inputs A and B to each multiplier.

Note: the value 1 can be chosen for the input A.

b) use of the result from the multipliers.

Each multiplier produces the product A x B.

For each pair of multipliers; the following operations can be carried out:

$M_1 + M_2$

$M_1 - M_2$

M_1

Result: ignoring the result of one of the multipliers

M_2

$- 1$

What emerges from this last operation is the result from the arithmetic unit. This result (part real and part imaginary) is stored in the result memory at the address calculated by the APM processor. (Obviously this only applies if storage is required c.f. ACC instruction).

Note :

If, as seen in a) above, we can decide each one of the inputs A and B for the 4 multipliers, it is impossible to address more than one complex sample at a time. As a result, X and Y are the same for all multipliers, and so an operation between two complex samples requires two steps.

2.4. The Process of Accumulation

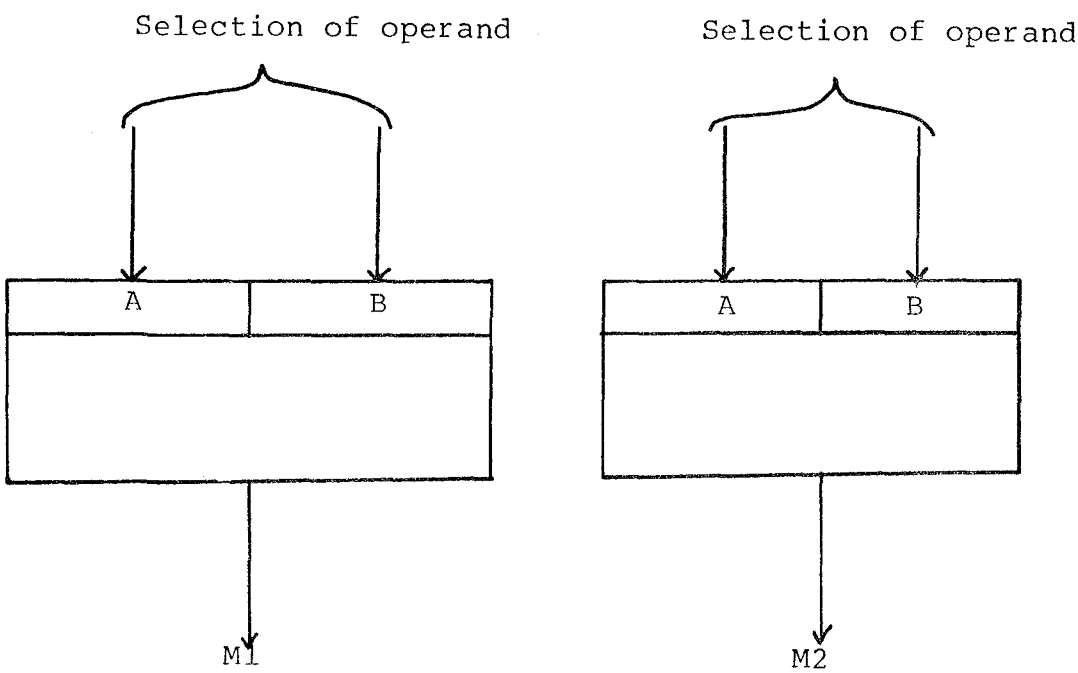
2.4.1 The Method in Use

2.4.1.1 Method 1

Let us calculate the expression $S = \sum_{0}^{N} X_i X_{i+1}$

It can be calculated in N + 1 operations.

Figure 8



Selection of operand : X or Y or 1 (for A entry only)

Let X_0 X_1 be the first term in this sum. It will be stored in the result memory at a given address. If the second term X_1 X_2 is stored at the same address, it will erase the preceeding term.

To calculate S correctly, it must be added to the preceding term. That is the method of accumulation.

The result of a calculation by the arithmetic unit can be:

- stored in the result memory (erasing what was there before)
- be added to that which was in the result memory and "re-stored" at the same address.

2.4.1.2 Method 2

Let an experiment proceed in several stages in line, and suppose we want to accumulate the result of each stage in the result memory.

e.g.

At stage 0, the correlator calculates $S_0 = \sum_{i=0}^{N} X_{oi} \; X_{o\ i-1}$

" " 1, " " " S_1

" " M, " " " $S_M = \sum_{i=0}^{N} X_{Mi} \; X_{M.\ i-1}$

The final result is $S = \sum_{j=0}^{M} S_j$

Method 1 does not work as at each stage the result erases the previous stage.

To resolve this problem, the correlator possesses a second method of accumulation, with higher priority, allowing us to inhibit the first method and to make the accumulation in the result memory even if the first method indicates that it cannot be done.

This corresponds to 2 modes of operation of the correlator

- an experiment commenses and the contents of the result memory are erased
- an experiment continues and the results are accumulated in the result memory.

The mode in which the correlator operates is determined by the internal microprogram (c.f. 2.4.2.3) which:

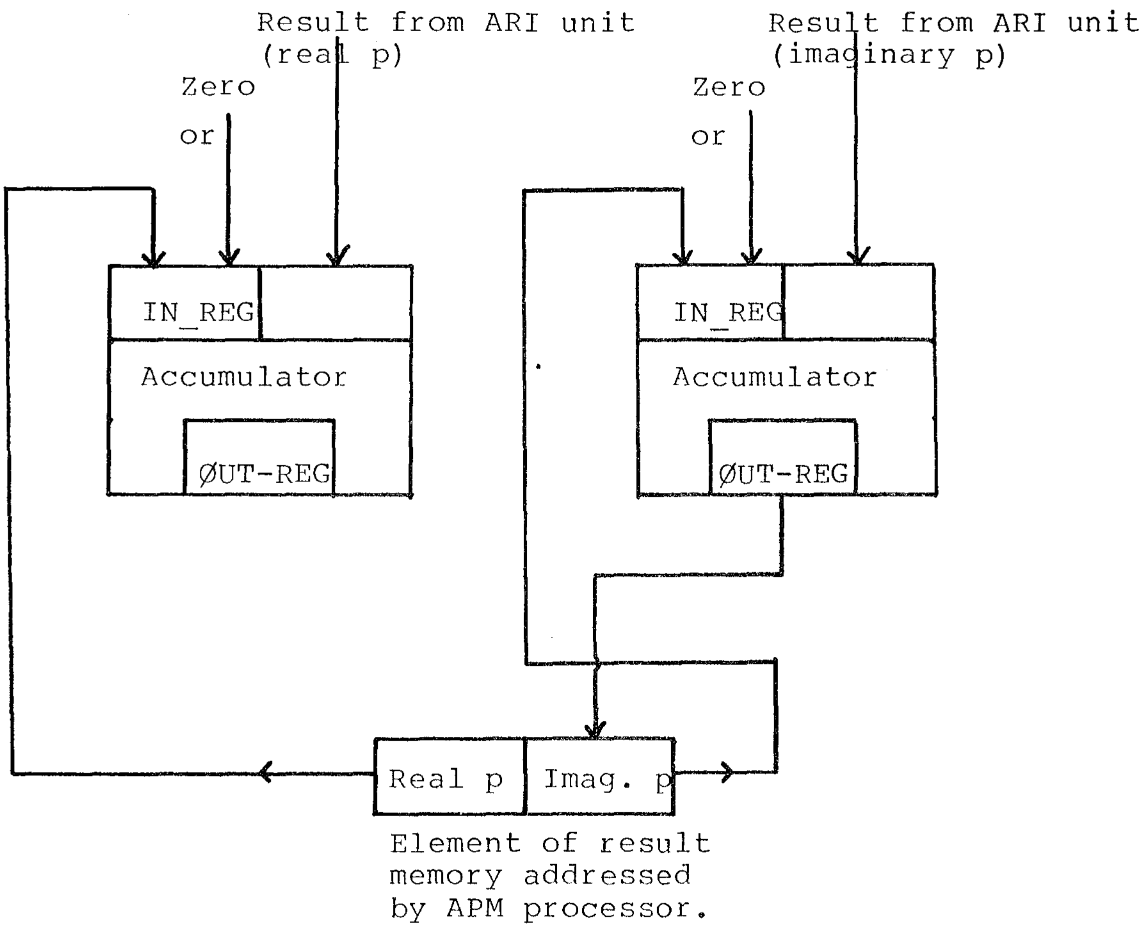
- launches an experiment with the order START - EXPERIMENT
- continues an experiment with the order CONTINUE - EXPERIMENT

2.4.2 Description of the Methods of Accumulation

2.4.2.1 Description

The accumulation of the multiplier-results, for real and imaginary values is achieved by two accumulators. This mechanism is described in Fig. 9.

Figure 9


Result from ARI unit
(real p)
Zero
or
Result from ARI unit
(imaginary p)
Zero
or
IN_REG
Accumulator
ØUT-REG
IN_REG
Accumulator
ØUT-REG
Real p
Imag. p
Element of result
memory addressed
by APM processor.

Each accumulator has two inputs. One of these is always the result of calculation by the arithmetic unit (AU). Accumulation is carried out, or not, according to the value in the other input of the accumulator which already contains the result memory, or which has the value zero.

2.4.2.2 Terminology

Each accumulator has two inputs and one output. The output is the OUT - REGISTER, which can be written in the result memory at the address calculated by the APM processor.

The two inputs are: - the output of the ALU
- the IN - REGISTER

The IN - REGISTER is loaded:

- either with the contents of the results memory (accumulation)
  or with zero (non-accumulation).

- the first method of accumulation is called FF1

- the second is called FF2.

2.4.2.3 Function

Whichever method is used, the accumulation instructions contain 3 bits indicating whether or not:

- the IN/OUT REGISTER are loaded into the accumulators

- the OUT - REGISTER is written into the result memory

- The IN - REGISTER is loaded with the contents of the result memory (this loading, if it is requested, will be, or will not be implemented according to whether the method FF1 or FF2 is used).

A necessary (but not sufficient) condition for accumulation is that these 3 bits are all 1.

In both methods FF1 AND FF2 is controlled by 2 bits

- one SET which sets up the method (bit = 1);
- one CLEAR which deactivates if it was in use (bit = 1);

  (the value 0 correspond to NO - OPERATION).

One set up, FF1 ØR FF2 remain in force as long as a CLEAR operation is not programmed.

- if FF2 is inhibited, the reading of the result memory in the IN - REGISTER is controlled by FF1
- if FF2 is in force, the control of FF is inhibited.

If FF2 is inhibited:

- if FF1 is in force, the IN - REGISTER is loaded with the contents of the result memory (in agreement with the 3 bits mentioned earlier)
- if not, the IN - REGISTER is loaded with 0.

If FF2 is in force, the result memory is always read into the IN - REGISTER. (In agreement with the 3 bits mentioned earlier.

EISCAT publications

F. du Castel, O. Holt, B. Hultqvist, H. Kohl and M. Tiuri:
A European Incoherent Scatter Facility in the Auroral Zone (EISCAT).
A Feasibility Study ("The Green Report") June 1971. (Out of print).

O. Bratteng and A. Haug:
Model Ionosphere at High Latitude, EISCAT Feasibility Study, Report No. 9.
The Auroral Observatory, Tromsö July 1971. (Out of print).

A European Incoherent Scatter Facility in the Auroral Zone, UHF System and Organization ("The Yellow Report"), June 1974.

EISCAT Annual Report 1976.(Out of print).

P.S. Kildal and T. Hagfors:
Balance between investment in reflector and feed in the VHF cylindrical antenna.
EISCAT Technical Notes No. 77/1, 1977.

T. Hagfors:
Least mean square fitting of data to physical models.
EISCAT Technical Notes No. 78/2, 1978.

T. Hagfors:
The effect of ice on an antenna reflector.
EISCAT Technical Notes No. 78/3, 1978.

T. Hagfors:
The bandwidth of a linear phased array with stepped delay corrections.
EISCAT Technical Notes No. 78/4, 1978.

Data Group meeting in Kiruna, Sweden, 18-20 Jan. 1978
EISCAT Meetings No. 78/1, 1978

EISCAT Annual Report 1977

H-J. Alker:
Measurement principles in the EISCAT system
EISCAT Technical Notes No. 78/5, 1978

EISCAT Data Group meeting in Tromsö, Norway 30-31 May, 1978
EISCAT Meetings No. 78/2, 1978.

P-S. Kildal:
Discrete phase steering by permuting precut phase cables.
EISCAT Technical Notes No. 78/6, 1978

EISCAT UHF antenna acceptance test.
EISCAT Technical Notes No. 78/7, 1978.

P-S. Kildal:
Feeder elements for the EISCAT VHF parabolic cylinder antenna.
EISCAT Technical Notes No. 78/8, 1978.

H-J. Alker:
Program CORRSIM: System for program development and software simulation of EISCAT digital correlator, User's Manual.
EISCAT Technical Notes No. 79/9, 1979.

H-J. Alker:
Instruction manual for EISCAT digital correlator.
EISCAT Technical Notes No. 79/10, 1979

H-J. Alker:
A programmable correlator module for the EISCAT radar system.
EISCAT Technical Notes No. 79/11, 1979.

T. Ho and H-J. Alker:
Scientific programming of the EISCAT digital correlator.
EISCAT Technical Notes No. 79/12, 1979.

S. Westerlund (editor):
Proceedings EISCAT Annual Review Meeting 1969. Part I and II,
Abisko, Sweden, 12-16 March 1979.
EISCAT Meetings No. 79/3, 1979.

J. Murdin:
EISCAT UHF Geometry.
EISCAT Technical Notes No. 79/13, 1979.

T. Hagfors:
Transmitter Polarization Control in the EISCAT UHF System.
EISCAT Technical Notes No. 79/14, 1979.

B. Törustad:
A description of the assembly language for the EISCAT digital correlator.
EISCAT Technical Notes No. 79/15, 1979.

J. Murdin:
Errors in incoherent scatter radar measurements.
EISCAT Technical Notes No. 79/16, 1979.

EISCAT Digital Correlator. TEST MANUAL.
EISCAT Technical Notes No. 79/17, 1979.

G. Lejeune:
A program library for incoherent scatter calculation.
EISCAT Technical Notes No. 79/18, 1979.

K. Folkestad:
Lectures for EISCAT Personnel, Volume I
EISCAT Technical Notes No. 79/19, 1979.

Svein A. Kvalvik:
Correlator Buffer-Memory for the EISCAT Radar system
EISCAT Technical Notes. No. 80/20.

P-S. Kildal:
EISCAT VHF Antenna Tests
EISCAT Technical Notes No. 80/21

J. Armstrong:
EISCAT Experiment Preparation Manual
EISCAT Technical Notes No. 80/22

A. Farmer:
EISCAT Data Gathering and Dissemination
EISCAT Technical Note 80/23

Terrance Ho and Hans-Jørgen Alker:
Scientific Programming of the EISCAT Digital Correlator (Revised)
EISCAT Technical Note 81/24

Terrance Ho:
Programs Corrsim, Corrtest: System for Program Development and Software Simulation of EISCAT Digital Correlator. User´s manual.
EISCAT Technical Note 81/25

Terrance Ho:
Instruction Manual for EISCAT Digital Correlator (Revised).
EISCAT Technical Note 81/26

Terrance Ho:
Standard Subroutines and Programs for EISCAT Digital Correlator.
EISCAT Technical Note 81/27

Terrance Ho:
Pocket Manual for Programming the EISCAT Digital Correlator.
EISCAT Technical Note 81/28

K. Folkestad:
Lectures for EISCAT Personnel, Volume II.
EISCAT Technical Note 81/29

M. Lehtinen och Anna-Liisa Turunen:
EISCAT UHF antenna direction calibration
EISCAT Technical Note 81/30

K. Folkestad:
Use of the EISCAT Radar as a supplement to rocket measurements.
EISCAT Technical Note 81/31

T. Turunen, T Mustonen and P J S Williams:
EISCAT UHF RECEIVERS: Report and Recommendations
EISCAT Technical Note 81/32

Phil Williams:
Polarisers in the EISCAT System
EISCAT Technical Note 81/33