81/25

EISCAT
TECHNICAL
NOTE

Programs CORRSIM, CORRTEST: System for Program Development and Software Simulation of EISCAT Digital Correlator

User's manual

by
Terrance Ho

KIRUNA
Sweden

PROGRAMS CORRSIM, CORRTEST:
SYSTEM FOR PROGRAM DEVELOPMENT AND SOFTWARE SIMULATION
OF
EISCAT DIGITAL CORRELATOR.

<u>USER'S MANUAL</u>

by

TERRANCE HO
MAX PLANCK INSTITUT
POSTFACH 20
D-3411 KATLENBURG-LINDAU 3
W. GERMANY

EISCAT Technical Note 81/25
Printed in Sweden
EISCAT Scientific Association
Kiruna, March 1981
ISSN 0349-2710

TABLE OF CONTENTS

PREFACE

The program CORRSIM as described in the report "PROGRAM CORRSIM: SYSTEM FOR DEVELOPMENT AND SOFTWARE SIMULATION OF EISCAT DIGITAL CORRELATOR. USER'S MANUAL" by Hans-Jørgen Alker, printed October 1978, has been taken off the NORD 10 computer system and has been replaced by two new programs CORRSIM and CORRTEST. These two programs have been written as there was a need to refine and to further develop the original CORRSIM program. Therefore all references made to CORRSIM refer to the new CORRSIM program and should not be confused with the old CORRSIM program.

Terrance Ho

Max Planck Institut für Aeronomie

D-3411 Katlenburg-Lindau 3

I. INTRODUCTION

The programs CORRSIM and CORRTEST are multi-function software packages for development and testing of user defined correlator programs. CORRSIM also includes software-drivers for program-loading of the correlator and direct-memory-access (DMA) handling of data from the external correlator source to the NORD 10 computer.

CORRSIM offers a systematic way of real-time correlator programming and with the additional options for interactive communication with the correlator creates an effective system for external hardware testing and control.

CORRTEST offers a systematic way of testing correlator programs off-line by simulating the correlator hardware and producing outputs of each micro-instruction as it loops through the micro-program. Furthermore the data results can be stored on a file which can be compared directly with the data results from the correlator when using CORRSIM.

The EISCAT digital correlator is a multi-processor system for real-time control and interfacing of separate internal functions. The programming is based on micro-instruction definitions in decimal form (integer code) or can also be programmed using the "higher level language" PREPASS system which produces a data file that can be read by CORRSIM. Further information can be found in the report: "A description of the assembly language for the EISCAT Digital Correlator" by Baard Törustad, printed August 1979. To be able to use CORRSIM and CORRTEST it is assumed that the user is familiar with the basic instruction set of the correlator. This information can be found in the report: "Instruction Manual for EISCAT Digital Correlator (Revised)".

II. 1 CORRSIM PROGRAM STRUCTURE

The program is modular constructed and split into blocks of program-routines corresponding to separate program functions. The program separation is sketched in Figure 1. The block termed MAIN has the function selection program SIMUL for either the subroutine EDITOR or CORRLO. The various program blocks can be broken down in the following way:

EDITOR	- Correlator program development and modification
SYMTAB	
CODEAB	
LC1	
LC2	
LC3	- Decoding of user-defined micro-instructions.
APBM	
ARI	
SACC	
OUTPTR	
CORRLO	- Interactive communication with correlator hardware
WCORR	- Load correlator data registers / program memory.
DMAEMABLE	- Enable CAMAC registers for DMA transfer from correlator to Nord 10.
BEGIN	- Start correlator in several different ways.
INSPECTRTC	- Look at data in RT-COMMON
VALUE	- Display of data transferred via DMA from correlator result memory.
PRINT	- Print data on line printer.
GRAPHIC	- Graphic display of data

The files PROG0, PROG1, PROG2, etc are a library of fixed subroutines and programs (read access only). The user file can be a file of any name. Thus any number of fixed subroutines can be read into the common area and a main program constructed to link the subroutines and the resulting program written to a file of the users choice.

The main operational mode when running CORRSIM is by interactive communication with the program and it is recommended to use a display terminal. However, if the user wishes to use the GRAPHIC option he/she must use a Tektronix terminal.

All numeric input/output in CORRSIM can be given in INTEGER or OCTAL format, depending on which mode is chosen. The only exception to this is when defining the data/program fields. These must be given in INTEGER format regardless of which mode that has been selected.

II.2 CORRTEST PROGRAM STRUCTURE

The program is modular constructed and split into blocks of program routines corresponding to separate program functions. The program separation is sketched in Figure 2. The block termed MAIN has the function selection program CORRTEST for either the subroutines EDITOR, PROTST, ARITST. The various program blocks can be broken down in the following way:

EDITOR	- Modification of data field only.
PROTST, DAPBM	- Real-time simulation of correlator hardware
ARITST	- Simulation of correlator arithmetic for fixed programs.

The CORRTEST program assumes that complete correlator programs are read into the common area. Only modification to the data field is allowed. The CORRTEST files X-DATA1 and Y-DATA1 stores the test data values which are also fixed in the correlator. This data forms the basis for the simulation of correlator arithmetic. The CORRTEST file DMADUMP is a file to which the data of the simulated correlator result memory can be written to. Thus data from the ARITST or PROTST routines can be compared with the actual values of the correlator result memory when using CORRSIM.

The main operational mode when running CORRTEST is by interactive communication with the program and it is recommended to use a display terminal.

Note that all numeric input in CORRTEST are in INTEGER format. Numeric output can be given in either INTEGER or OCTAL format, depending on which option that has been chosen.

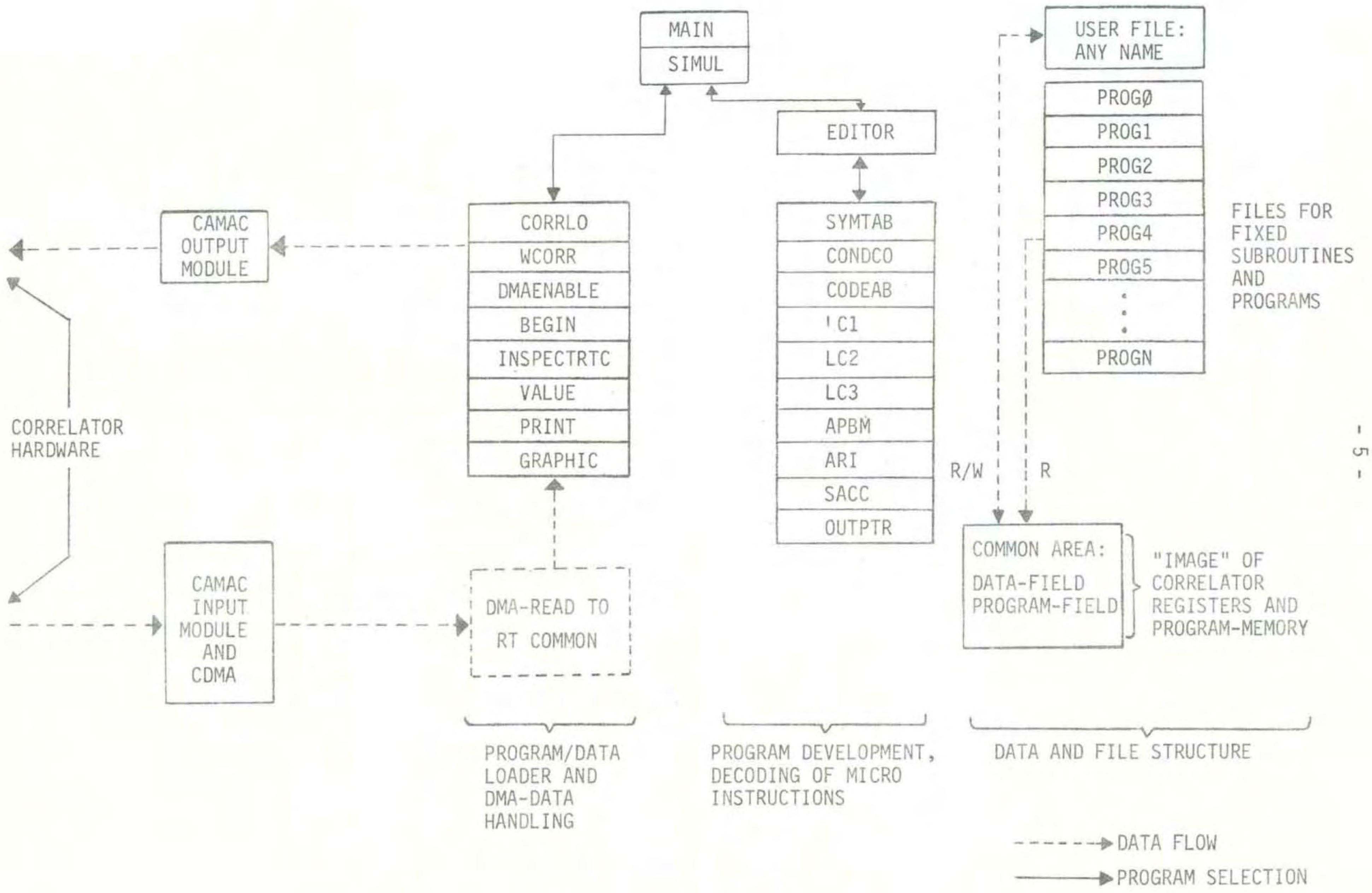
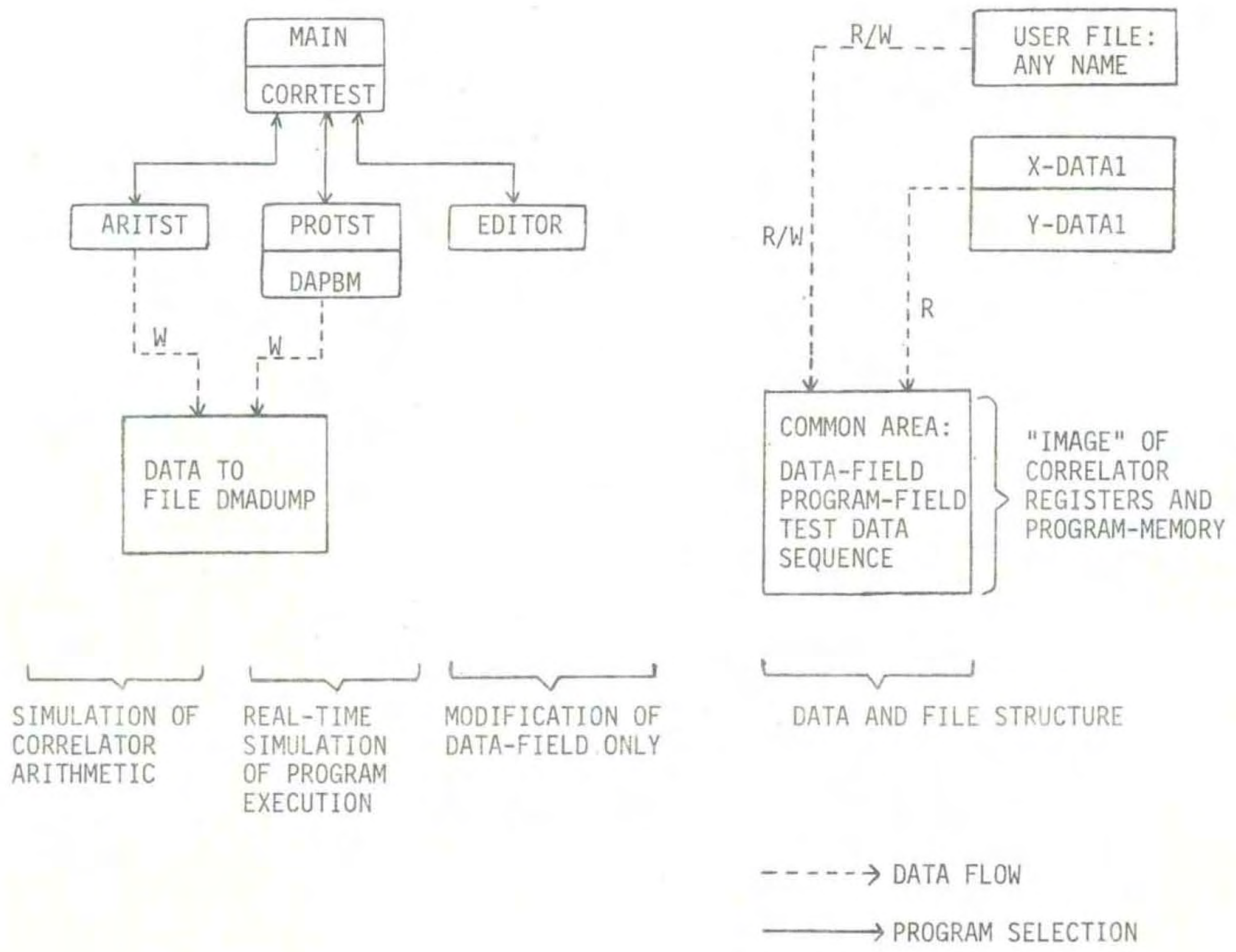


FIGURE 1. CORRSIM STRUCTURE



MAIN
CORRTEST
ARITST
PROTST
DAPBM
EDITOR
W
W
DATA TO FILE DMADUMP
R/W
USER FILE: ANY NAME
X-DATA1
Y-DATA1
R/W
R
COMMON AREA: DATA-FIELD PROGRAM-FIELD TEST DATA SEQUENCE
"IMAGE" OF CORRELATOR REGISTERS AND PROGRAM-MEMORY
SIMULATION OF CORRELATOR ARITHMETIC
REAL-TIME SIMULATION OF PROGRAM EXECUTION
MODIFICATION OF DATA-FIELD ONLY
DATA AND FILE STRUCTURE
DATA FLOW
PROGRAM SELECTION

FIGURE 2. CORRTEST STRUCTURE

III. START OF CORRSIM

The program is stored on the user-file (CORR-TEST)CORRSIM and is started with the SINTRAN command: (CORR-TEST)CORRSIM
or simplified: (CO-TE)CORRSIM

The response from the program is

```
    PROGRAM CORRSIM IS ACTIVE


  THE SIMULATOR CONSTITS OF 2 MAIN SUBROUTINES:
  SUBROUTINE EDITOR FOR PROGRAM SET-UP AND MODIFICATION
  SUBROUTINE CORRLO FOR I/O-COMMUNICATION WITH THE CORRELATOR
  THE SIMULATOR IS TERMINATED BY TYPING FIN
  NOTE: FOR SIMULATION OF PROGRAM-EXECUTION USE PROGRAM CORRTEST
```

The program then asks for function-selection:

```
  GIVE COMMAND (EDITOR, CORRLO, OF FIN):
```

(- denotes the necessary character input)

General commands available in all function-subroutines, whenever the program asks for a non-numeric input, are:

EXIT	(termination of function-subroutine)
FIN	(termination of CORRSIM)

whenever an illegal command is typed on the terminal the response is:

```
  ILLEGAL COMMAND
```

All numeric input/output of CORRSIM is either INTEGER or OCTAL, with the exception when defining the data/program fields which are always in INTEGER format. All numeric input must be terminated by "," regardless of which mode is being used.

When the program is started with (CO-TE)CORRSIM all main internal program parameters are reset. This means that after a FIN command has been generated all earlier set parameters are lost. This problem can be avoided by stopping CORRSIM with a user-break, SINTRAN commands will then be available. The program can be restarted from break-point with earlier defined values with the SINTRAN command:

```
GOTO USER
```

Alternatively the program with its particular set of parameters can be saved by writing it onto a file before generating a FIN command.

Together with program generated error messages the run time error diagnostics in SINTRAN are active, mainly giving error messages when input format specifications are not fulfilled.

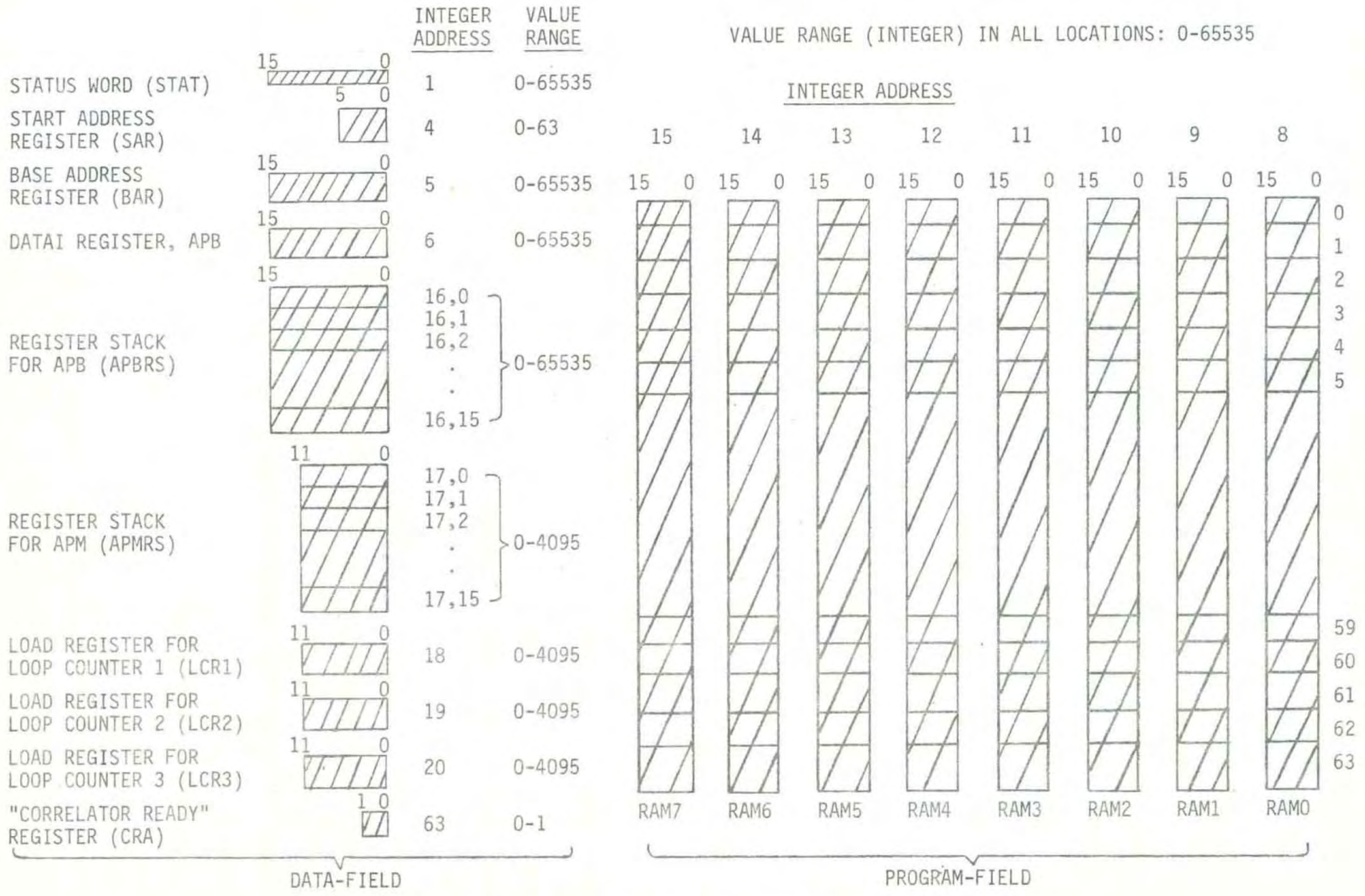

INTEGER ADDRESS
VALUE RANGE
STATUS WORD (STAT)
1
0-65535
START ADDRESS REGISTER (SAR)
4
0-63
BASE ADDRESS REGISTER (BAR)
5
0-65535
DATAI REGISTER, APB
6
0-65535
REGISTER STACK FOR APB (APBRS)
16,0
16,1
16,2
16,15
0-65535
REGISTER STACK FOR APM (APMRS)
17,0
17,1
17,2
17,15
0-4095
LOAD REGISTER FOR LOOP COUNTER 1 (LCR1)
18
0-4095
LOAD REGISTER FOR LOOP COUNTER 2 (LCR2)
19
0-4095
LOAD REGISTER FOR LOOP COUNTER 3 (LCR3)
20
0-4095
"CORRELATOR READY" REGISTER (CRA)
63
0-1
DATA-FIELD
VALUE RANGE (INTEGER) IN ALL LOCATIONS: 0-65535
INTEGER ADDRESS
15
14
13
12
11
10
9
8
RAM7
RAM6
RAM5
RAM4
RAM3
RAM2
RAM1
RAM0
0
1
2
3
4
5
59
60
61
62
63
PROGRAM-FIELD

FIGURE 3. COMPUTER "IMAGE" OF CORRELATOR DATA/PROGRAM LOCATIONS

IV DEVELOPMENT OF CORRELATOR PROGRAMS, USE OF FUNCTION EDITOR (CORRSIM)

When the EDITOR COMMAND is given, the editor will respond by typing

```
C*
```

on the terminal. The editor is now ready for accepting commands.

The basic principle for the development of correlator programs is a creation of an "image" of the correlator programmable registers and program memory, this image is integer coded and is split into two parts:

DATA-FIELD where all correlator data registers are located.

PROGRAM-FIELD where the correlator program memory is located.

The program memory is split into 8 separate modules (given by hardware construction) termed RAM0, RAM1..., RAM8. Each module has a word length of 16 BITS and 64 memory locations. One correlator micro-instruction contains 128 bits and allocates one 16 BIT word location in each module. The correlator data-registers, register-stacks and memory modules are identified by an integer ADDRESS parameter, individual register stack and memory module locations are identified by an integer SUB-ADDRESS parameter. The data and program field structures are given in Figure 3 together with the integer addresses/subaddresses and value range. With this program structure a micro-instruction location is identified by the SUB-ADDRESS parameter (same location in all modules). For further information about micro-instruction separation into main internal correlator functions and available instruction set see Ref. 1.

The data/program field in the editor is kept in common and are used by all other function programs in CORRSIM. Together with the "image" in common is also stored a library of flag registers indicating which of the register/program locations are defined by the user.

The editor commands are divided into sub-groups according to different functions. These are:

SET VALUE COMMANDS:	for loading and defining locations in data/ program field.
DELETE COMMANDS:	Reset-function
LIST COMMANDS:	for display output of defined values
PRINT COMMANDS:	for line printer output of defined values.
SHIFT COMMAND:	to shift a set of defined memory locations from one part of the memory to another.
READ DATA FILE:	input from user defined file or fixed program from CORRSIM files.
WRITE DATA FILE:	output of defined values to user file.

IV.1 SET VALUE COMMANDS

There are two commands available in order to enter data in the different registers and program memory locations of the correlator.

COMMAND SDF (Set data field)

The registers of the data field can now be defined. The CORRSIM response to the SDF command is:

```
GIVE ADDRESS, SUBADDRESS. (TYPE 0 for END):
```

The register integer address and subaddress is then entered with the numeric field termination ",". After checking the address and subaddress values the response will be (if address/subaddress values are valid):

```
GIVE DATA:
```

The integer value can then be entered. After checking the value (and is valid) a new address input will be asked for. The loading sequence is terminated by entering "0" for the address or simply by giving a "return". The response will then be:

```
nn REGISTER LOCATIONS ARE DEFINED
C*
```

Note that whenever a subaddress is not required (the single registers in the data-field) it is not necessary to enter "0".

COMMAND SPS:FUNCTION (set program function instructions separate)

The function has the following legal names:

PRO	program instruction
APB	instruction for buffer address processor.
APM	instruction for result memory address processor.
ARI	instruction for correlator arithmethic
ACC	instruction for the accumulators
I/O	instruction for input/output bus handling.
OUT	DMA output and display function.

The different FUNCTION names correspond to the correlator main internal operations which are processed in parallel.

The program response to the command is:

```
GIVE MEMORY-LOCATION. (TYPE 64 FOR END):
```

where the location corresponds to the subaddress parameter in the program-memory. If the location is already defined the program displays the integer values of the different sub-instructions of the function and then asks for:

```
GIVE INSTR.: (STRING OF SUB-INSTRUCTION PARAMETERS)
```

where the parameter names follow the structure given in the instruction-mannual (Ref.1) After the integer values of the parameters have been entered (all values on the same line) they are then tested for validity and if they are valid a display of the new instruction word follows with earlier defined instruction words of lower memory locations (at most 15 locations are displayed). The loading sequence is then repeated until the termination address 64 is given. The loading sequence is then terminated and the number of defined function instructions are given. When consecutive function locations should have the same values then the repeat loading format can be used by entering the lower, upper memory location when the address is given.

When loading an undefined memory location the SPS:PRO instruction must be used first.

IV. 2 DELETE COMMANDS

There are three commands for deleting defined values in the data and program fields.

COMMAND D

All defined parameters in the data and program fields are deleted.

COMMAND DD (delete data)

The program response to the command is:

```
GIVE ADDR., SUBADDR. FOR DATAFIELD (TYPE 0 FOR END):
```

COMMAND DP (delete program)

The memory location of all the RAM modules (ie. 128 BITS) will be deleted. The program response to the command is:

```
GIVE MEMORY-LOCATION (TYPE 64 FOR END):
```

The repeat delete format can be used for deleting concecutive memory location by entering the lower, upper memory location.

IV.3 LIST COMMANDS

The following commands are available.

COMMAND LD (list data)

All defined data registers are displayed.

COMMAND LP (list program field)

The program response is:

```
GIVE LOWER, UPPER, VALUE IN MEMORY FOR DISPLAY:
```

The defined memory locations of all the RAM modules in octal format will be displayed. If none of the referenced memory locations are defined the program will display:

```
NO LOCATIONS IN PROGRAM-FIELD DEFINED
```

If only one memory location should be displayed then the LOWER parameter should be defined. Default address values (0,0,) or just a "return" will give a display of all defined locations of the memory.

COMMAND LPS:FUNCTION (list program function instructions separate)

The legal function names are given under the SPS command.

The address format is as for the LP command.

COMMAND LPD (list program decoded)

The address format is as for the LP command.

The LPD command gives a decoding of the complete micro instruction in a "high level" language. All values in the micro instruction are tested for legality.

IV.4 COMMAND SH (shift)

The program response is:

```
GIVE LOWER, UPPER, SHIFT PAR.:
```

The LOWER to UPPER memory locations will be shifted by an amount defined by the SHIFT PAR. (ie. LOWER+SHIFT PAR. to UPPER+SHIFT PAR.) in the memory. If a direct jump address is defined it will be automatically adjusted. The memory locations from LOWER to UPPER will also remain defined.

IV.5 PRINT COMMANDS

The response to the first print command is:

```
CURRENT IDENTIFICATION NAME:
```

(TEXT STRING GIVING PROGRAM NAME)

```
DO YOU WISH NEW IDENTIFICATION NAME? TYPE Y OR N
```

If Y is typed the next response is:

```
GIVE PROGRAM-IDENTIFICATION NAME (MAX. 80 CHA.) FOR PRINT:
```

A text string can now be entered. The old text will then be replaced by the new one.

COMMAND PDP (print data and program field)

All defined registers in the data field will be printed and an "image" of all the defined locations in the RAM modules will be printed in octal.

COMMAND PPS (print program function instructions separate)

All defined memory locations will be printed for all functions.

COMMAND PPD (print program decoded)

All defined memory locations will be printed in the decoded form.

COMMAND PEND (print end)

The print-out on the line-printer will be obtained immediately.

IV.6 COMMAND R (read file)

This command allows any user-defined file to be read into CORRSIM.

The program response is:

```
GIVE FILE NAME:
```

The user-defined file name can now be entered.

The next response is:

```
GIVE ENTRY-POINT IN CORRELATOR MEMORY:
```

This parameter enables the user to place the program being read anywhere in the memory in locations relative to those defined by the subaddresses in the program. If default value (0) is used then all data values will be loaded into the locations defined by the subaddress parameter. If the entry-point is other then zero (positive or negative) then all data values will be loaded into the memory locations defined by the subaddress parameter plus the entry-point value. Memory location 0 is not affected by the entry-point value and if a jump address has been defined this will be automatically adjusted.

IV.7 COMMAND W (Write file)

The command allows the user to write the program (data/program field) onto any file.

The program response is:

```
GIVE FILE NAME:
```

The user can enter any file name.

The next response is:

```
    CURRENT IDENTIFICATION NAME:
(TEXT STRING GIVING PROGRAM NAME)
    DO YOU WISH NEW IDENTIFICATION NAME? TYPE Y OR N
```

If Y is typed the next response is:

```
    GIVE FILE IDENTIFICATION NAME (MAX. 80 CHA.):
```

A text string can now be entered. The old text will then be replaced by the new one. The next response is:

```
    GIVE LOWER? UPPER MEM-LOCATION FOR FILE-DUMP:
```

With default values (0,0) all defined data registers and program locations will be written onto the file. With any other values all defined data registers and only those defined memory locations between LOWER, UPPER will be written onto the file.

The file format for defined locations is:

```
    (TEXT STRING)      FORMAT (80A1)
    (Addr.),(Subaddr.) (data value)  FORMAT (2I3,I6)


    (Addr.),(Subaddr.) (data value)
    0,0,0
```

The 0's loaded at the end is a indication of end of data.

V SUMMARY OF ALL COMMANDS FOR THE EDITOR (CORRSIM)

A	Address assignments of data/program fields of the correlator
D	Delete data and program fields
DD	Delete data field
DP	Delete program field.
E,EX,EXIT	Exit from EDITOR
F,FIN	terminate CORRSIM
H,HELP	Lists valid commands for EDITOR
LD	All defined data registers are displayed
LP	All defined program locations are displayed in Octal as RAM modules.
LPD	All defined program locations are decoded
LPS:PRO	All defined program instructions are displayed
LPS:APB	All defined instruction for buffer address processor are displayed
LPS:APM	All defined instructions for result memory address processor are displayed
LPS:ARI	All defined instructions for correlator arithmetic are displayed
LPS:ACC	All defined instructions for accumulators are displayed.
LPS:I/O	All defined instructions for input/output bus handling are displayed
LPS:OUT	All defined instructions for DMA output are displayed.
PDP	All defined data registers and program locations in octal as RAM modules are printed
PPS	All defined program functions are printed separately.
PPD	All defined program locations are printed decoded.
PEND	To obtain output on line-printer

R	Read from file
SDF	Data registers are defined
SPS:PRO	Program instructions are defined
SPS:APB	Buffer address processor instructions are defined
SPS:APM	Result memory address processor instructions are defined
SPS:ARI	Correlator arithmetic instructions are defined
SPS:ACC	Accumulator instructions are defined
SPS:I/O	Input/output was instructions are defined
SPS:OUT	DMA output instructions are defined.
W	Write to file.

VI INTERACTIVE COMMUNICATION WITH THE CORRELATOR, USE OF FUNCTION CORRLO

The CORRLO function is a software system which connects the CORRSIM program to the correlator hardware. The input/output transfer of data is established by use of stardardized CAMAC interface modules and the CORRLO function uses ordinary CAMAC calls to achieve this data transfer. The physical interface structure is shown in Figure 4.

The CAMAC module 9043 (dual output register) is used for loading the correlator system. The A register output is the driver for the correlator system address/data bus. This bus connects all correlator modules to the computer. Each correlator module is identified by the CORRELATOR IDENT CODE. This parameter must always be defined in the status word (see Ref. 1 for further details) of the correlator program before entering CORRLO. The ident-code is a select command and is located in the address transferred. The software system checks that a response is given by the selected correlator module. The A register bus is used for both the address and data transfer in which a 16 BIT data word is transferred in two cycles: the 1st cycle is for the transfer of the ident-code together with the internal register address, the 2nd cycle is for the data transfer. The B register bus transmits the necessary control commands, including the computer start command of the correlators and the MASTER RESET signal. This control bus is common for all correlator modules. The input transfer from the correlator system is through the CAMAC module 9041 (dual input register) where only the A register is used. The CAMAC input module is under the control of the CDMA module for direct memory access (DMA) of the correlator data to the computer memory.

Note that it is possible to use test data (identical to the data on the X-DATA1, Y-DATA1 files. See Fig. 2) in the correlator module during microprogram execution and the results can then be used for debugging purposes by comparison with the results from the simulator system.

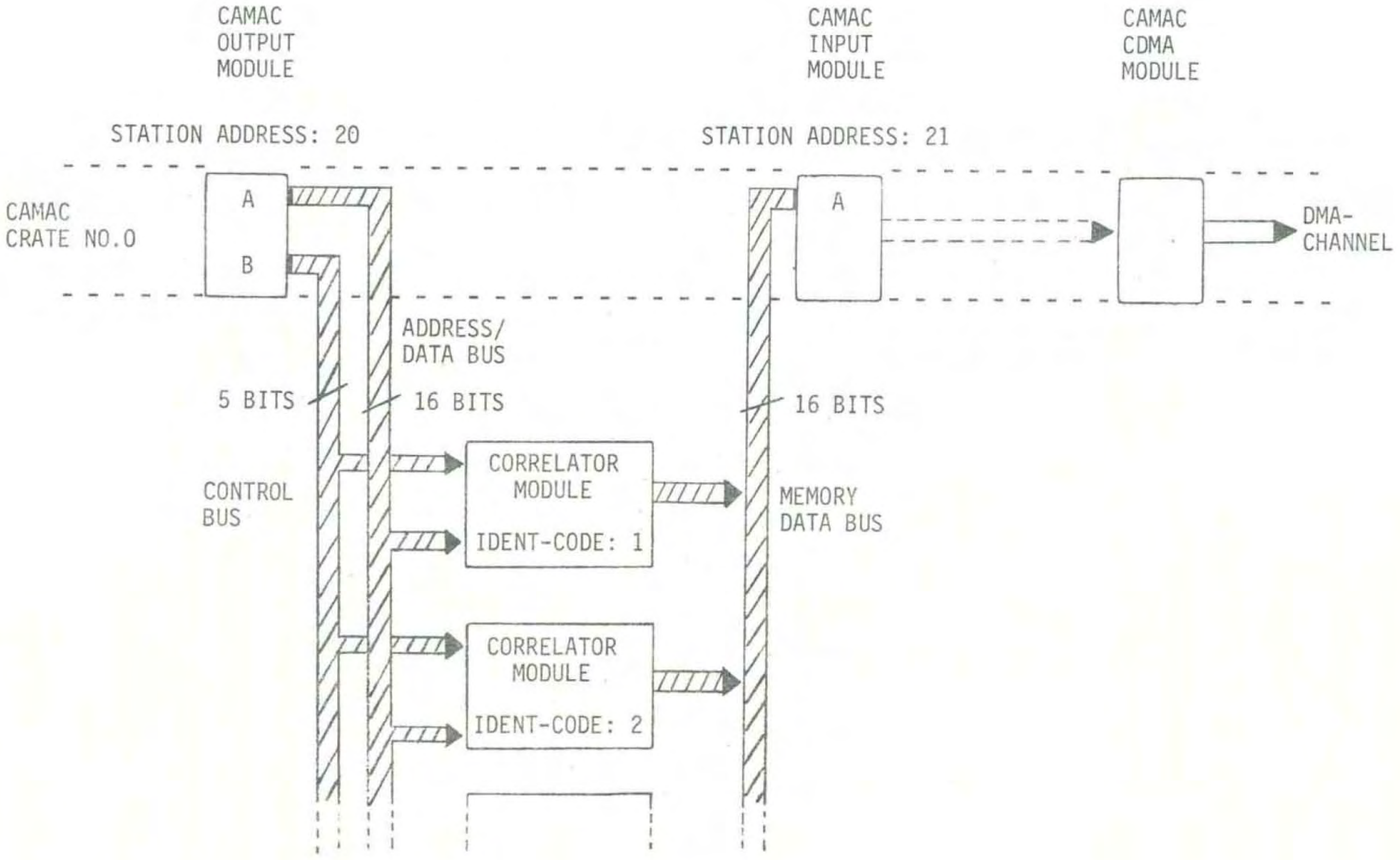


FIGURE 4. INTERFACE BETWEEN CORRELATOR SYSTEM AND COMPUTER

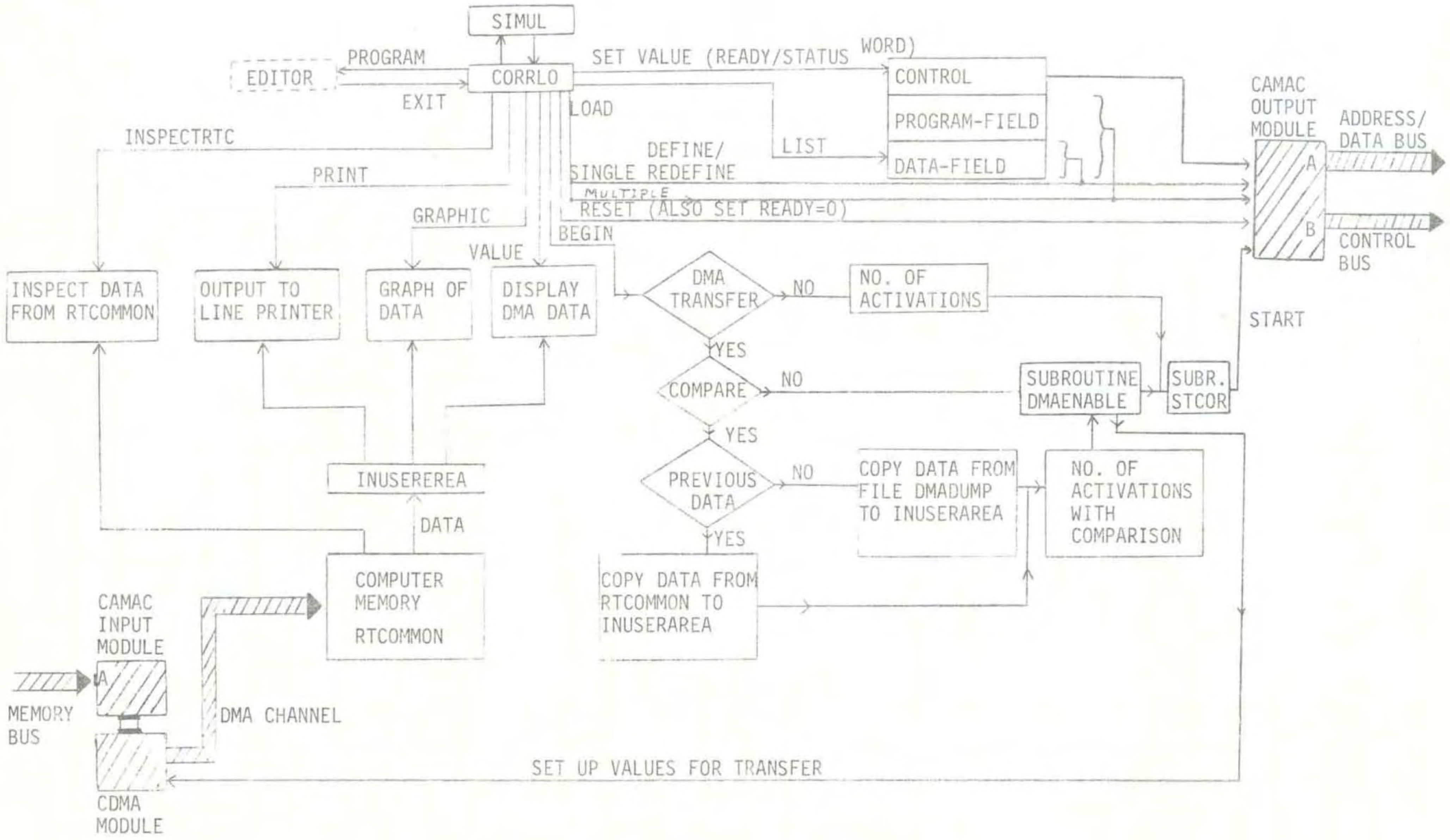


FIGURE 5. STRUCTURE OF CORRLO SYSTEM

The output transfer of data from the correlator is under micro-program control.

The CORRLO subroutine is structurized as a command processor. A flow diagram describing the system is shown in Figure 5. After the subroutine CORRLO is called the response is:

L*

indicating that the command processor is active.

Several error situations may arise when CORRLO is selected. These are:

1. If the statusword is not defined. The response will be :

   WARNING: STATUSWORD NOT DEFINED.

2. If the statusword is defined but the correlator IDENT-CODE is not set. The response will be:

   WARNING: CORRELATOR-MODULE NO. MUST BE $\neq$ 0.

   In the case of 1 and 2 the user should redefine the statusword.

3. If the power is not turned on to the CAMAC crate and/or the correlator module. The response will be:

   NO COMMUNICATION WITH CORRELATOR

   and a return to SIMUL will be made.

4. If the correlator is in a data load sequence. This can occur if another user is using the correlator. The response will be:

   CORRELATOR IN DATA-LOAD SEQUENCE

   L*

5. If the correlator is running. This can occur if another user is using the correlator. The response will be:

   CORRELATOR IS RUNNING

   L*

   4 and 5 are essentially not errors but the user will not be able to communicate with the correlator until it is inactive again. The best policy here would be to check with who is using the correlator so as to avoid any misunderstanding!

VI.1 COMMAND S (set value)

The programm response is:

READY OR STATUSWORD:

If the response is R (READY), the correlator READY register will be set. This register must always be set before a START command is generated either from the computer or the radar controller. If the CRA register (address = 63) in the data field is not set, it will automatically be set. By implication the correlator READY register can be set if the CRA register is defined and a LOAD command is used. The register will be set until a RESET command is generated.

If the response is S (STATUSWORD), the status word register in the correlator can be redefined. The STAT register (address = 1) will automatically be redefined. The correlator status word register can also be redefined by use of the LOAD SINGLE command. Note that the correlator IDENT CODE must always be given when redefining the statusword.

VI.2 COMMAND R (RESET)

The R command will terminate internal program execution in the correlator and reset the READY register. All other register/program locations will not be affected.

VI.3 COMMAND D (Display data field)

The D command will display all defined registers in the data field.

VI.4 COMMAND P (Program)

The P command will execute a jump to the EDITOR. A later EXIT in the EDITOR will return control back to the command processor. This is a means whereby quick changes can be made to the program field.

VI.5 COMMAND M (Mode)

The M command defines what kind of data will be transferred from the correlator. By default when entering CORRLO the following assumptions are made:

1. 64 BIT result memory words are transferred
2. The status and control words are not transferred
3. The DATAI register in the data field is not used for the transfer parameter (i.e. no. of words being transferred from the correlator)
   The various options are:

1. 64 BIT result memory words; 16 BIT testwords of kind 1, 16 BIT testwords of kind 2.
2. The status and control words are transferred either before or after the data.
3. The DATAI register can be used for the transfer parameter. It is recommended to use this register as all final programs for experiment must use this register for the transfer parameter.

VI.6 COMMAND LM (List Mode)

An output of the mode status will be given.

VI.7 COMMAND INT (Integer)

The INT command sets the processor in integer mode. All numeric input to the processor must be given in integer format. All numeric output will be given in integer format.

VI.8 COMMAND OCT (Octal)

The OCT command sets the processor in octal mode. All numeric input to the processor must be given in octal format. The only exception to this is when using the LOAD SINGLE command which accepts only integer format. All numeric output will be given in octal format except when using the LOAD command.

VI.9 COMMAND C (CAMAC)

The C command gives the present status of the word-count and memory-address registers in octal format.

VI.10 COMMAND L (LOAD)

The program response is:

```
MULTIPLE LOAD GIVES TRANSFER OF ALL DEFINED VALUES
SINGLE OR MULTIPLE TRANSFER?:
```

If the response is M (Multiple) a direct load of all previous defined parameters will be made. During loading the communication with the correlator is checked and if no errors occur the loading is terminated by

```
MULTIPLE LOAD ENDED. nn 16-BIT VALUES TRANSFERRED.
```

where nn is the integer number of registers/RAM locations defined.

If the response is S (single) the program will respond with:

```
GIVE ADDRESS, SUBADDRESS. (TYPE 0 FOR END):
```

If the address, subaddress parameters are valid the next response is

```
GIVE DATA:
```

Note that only the data-field can be loaded in this way and these registers will be defined/redefined during the transfer to the correlator.

VI.11 COMMAND B (Begin)

The B command enables the user to start the correlator from the computer. Before using this command the following conditions must exist:

1. The correlator must not be busy.
2. The correlator READY must be set.
3. The bit in the correlator status word must be set for enabling the computer to start the correlator.

The program response is:

    DMA (Y OR RETURN/N):

If the response is N the program will respond with:

    NR. OF STARTS (xxx)/FOREVER:

where xxx will either be INT or OCT depending on which mode is chosen. When all starts have been completed the program will respond with:

```
ALL (nn) ACTIVATIONS COMPLETED
L*
```

where nn is the no. of starts.

If the response is Y the program will respond with:

    COMPARISON (Y/N OR RETURN):

If the response is N and the DATAI register is not being used as the transfer parameter as defined in MODE the program will then ask for either the no. of 64 BIT result memory words or 16 BIT testwords as defined in MODE. The CDMA module in the CAMAC crate will now be enabled and the word count register, WCR, will be loaded with the number of 16 BIT words and the correlator started. If the DATAI register has been defined in MODE the program will use this register to extract the relevant no. of words to be transferred. When successfull activations and DMA transfer has been executed the response is then:

```
CORRELATOR DONE
L*
```

If the response is Y the program response is:

    YOU ARE COMPARING WITH WHAT (PREVIOUS DUMP: PREV/FILE):

If the response is PREV then the data in RTCOMMON will be copied over to INUSERAREA for comparison.

If the response is FILE then the data from file DMADUMP will be copied over to INUSERAREA for comparison.

The next response is:

    NO. OF STARTS WITH COMPARISON (xxx. NO./ETERNAL):

where xxx is either INT or OCT depending on which mode is chosen. The CDMA module will now be enabled, the correlator started and a comparison made between the data in INUSERAREA and RTCOMMON. If there is no mismatch then after every successful activation the program response is:

```
ACT. nn OK
```

where nn is the no. of times the correlator has started.

If there is a mismatch then the program response is:

```
MISMATCH:ADDR NO.: nn: SYMB  MASTER: YY   CORRELATOR: ZZ
```

where nn is the location of the 64 BIT word in the result memory, SYMB is whether it is the most/least significant part in CHANNEL 1 or 2, YY is the value from INUSERAREA and zz is the value from the correlator. The values are in either integer or octal depending on which mode has been set.

If there are more than 14 mismatches the program will stop.

VI.12 COMMAND V (value of data)

The V command displays the real and imaginary parts (32 BITS for each part) of the data in either integer or octal.

VI.13 COMMAND I (Inspect RTCOMMON)

The I command displays the data as transferred from the correlator i.e. 16 BIT values in either integer or octal.

VI.14 COMMAND PRINT

The PRINT command outputs the real and imaginary parts of the data in either integer or octal on the line printer.

VI.15 COMMAND G (Graphic)

The G command displays a graph of the real and imaginary parts of the data.

VI.16 COMMAND NODMA

The NODMA Command enables the user to initialize CORRSIM to "grab" data from RT-COMMON when the correlator is under control from another source, e.g. from another user or under EROS.

VI.17 COMMAND ADC (A/D converter)

The ADC command enables the user to set the channel number and the sampling rate of the A/D converter.

VI.18 COMMAND BUFF (Buffer)

The BUFF command enables the user to set the channel number (program counter) and start address of the buffer memory.

VII. SUMMARY OF ALL COMMANDS FOR CORRLO

ADC	Set A/D converter
B	Start the correlator
BUFF	Set buffer memory
C	Status of CAMAC word-count and memory-address registers are displayed
D	All defined registers in the data field are displayed
E	Exit from CORRLO
F	Terminate CORRSIM
G	Real and Imaginary parts of data are displayed.
H or HELP	Lists valid commands for CORRLO
I	Data is displayed as transferred from correlator.
INT	Sets processor in integer mode
L	Loads the correlator
LM	Outputs the MODE status
M	Sets mode for data transfer from correlator
NODMA	Initialize CORRSIM to "grab" data from RT-COMMON
OCT	Sets processor in octal mode.
P	Direct jump to EDITOR
PRINT	Outputs data on line printer
R	Reset the correlator
S	Set value, either correlator READY or STATUS WORD
V	Displays data on the terminal.

VIII START OF CORRTEST

The program is stored on the user-file (CORR-TEST)CORRTEST and is started with the SINTRAN Command: (CORR-TEST)CORRTEST
or simplified: (CO-TE)CORRTEST

The response from the program is

```
                    PROGRAM CORRTEST IS ACTIVE




     THE SIMULATOR CONSISTS OF 3 MAIN SUBROUTINES:
     SUBROUTINE EDITOR FOR PARAMETER SET-UP AND MODIFICATION
     SUBROUTINE PROTEST FOR SIMULATION OF PROGRAM AND ARITHMETIC EXECUTION
     SUBROUTINE ARITEST FOR SIMULATION OF FIXED CORRELATOR PROGRAMS
     THE SIMULATOR IS TERMINATED BY TYPING FIN


Note: FOR PROGRAM SET-UP AND MODIFICATION AND DIRECT COMMUNICATION
     WITH CORRELATOR USE PROGRAM CORRSIM
```

The program then asks for function-selection:

```
     GIVE COMMAND (EDITOR, PROTEST, ARITEST OR FIN):
```

(_ denotes the necessary character input)

IX USE OF FUNCTION EDITOR (CORRTEST)

This EDITOR is a limited version of the EDITOR as described for program CORRSIM. The main difference is that the program field cannot be modified or decoded. The reason for this is that it is already assumed that complete correlator programs exist and that the user wishes to test them off line.

The commands that exist are used in exactly the same way as those in the EDITOR for CORRSIM therefore only a summary of the commands will be given (see chapter X). For a description of the commands see chapter IV.)

X. SUMMARY OF ALL COMMANDS FOR EDITOR (CORRTEST)

A	Address assignments of the data field of the correlator
D	Delete data and program fields
DD	Delete data field
E,EX,EXIT	Exit from EDITOR
F,FIN	Terminate CORRTEST
H.HELP	Lists all valid commands for EDITOR
LD	All defined data registers are displayed
R	Read from file
SDF	Data registers are defined
W	Write to file.

NOTE: With the R command an Entry-Point in Correlator Memory is not asked for.

XI. REAL-TIME CORRELATOR PROGRAM SIMULATION, USE OF FUNCTION PROTEST

Program development with the EISCAT correlator requires "real-time" programming, caused by the neccessary interfacing between the different internal processors operated in parallel. There are two main control units in the correlator; the CPU (Central Processing Unit) and the DPU (Data Processing Unit). The CPU is for controlling the micro program execution and generation of the read and write addresses for the buffer and result memories, the DPU is for data processing operations on the X,Y samples. Programming models of these two units are shown in Figures 6 and 7.

The program instruction word read from the memory contains subinstructions which controls the generation of the next program location to be read in the next clock-interval. This location is given by the program-counter (PC). Conditional branching in the program is realised by three separate loop counters (LC1, LC2 and LC3) which are operated in parallel with control from different subinstructions of the program instruction word. The loop-counters can be loaded from separate load-registers (LCR1, LCR2 and LCR3) located in the datafield. Also a temporary storage-register, LCRIA, can be used for storing values from LC1. A four-level register-stuck with LIFO (Last-in first-out) structure can be used for storing address-values (back loop branching). The next PC value can be taken from one of four sources:

From the PC-incrementer (continue statement)

From the register-stack (return statement)

From address in the instruction (go to or jump statement)

From SAR-register (start address for the program)

Conditional branching or break-points of the program-counter is dependent on the present instruction in the instruction register and the values of the loop counters. See Ref. 1 for the available instruction set. All locations in the program memory can be used for programming. However two locations are used for special purposes:

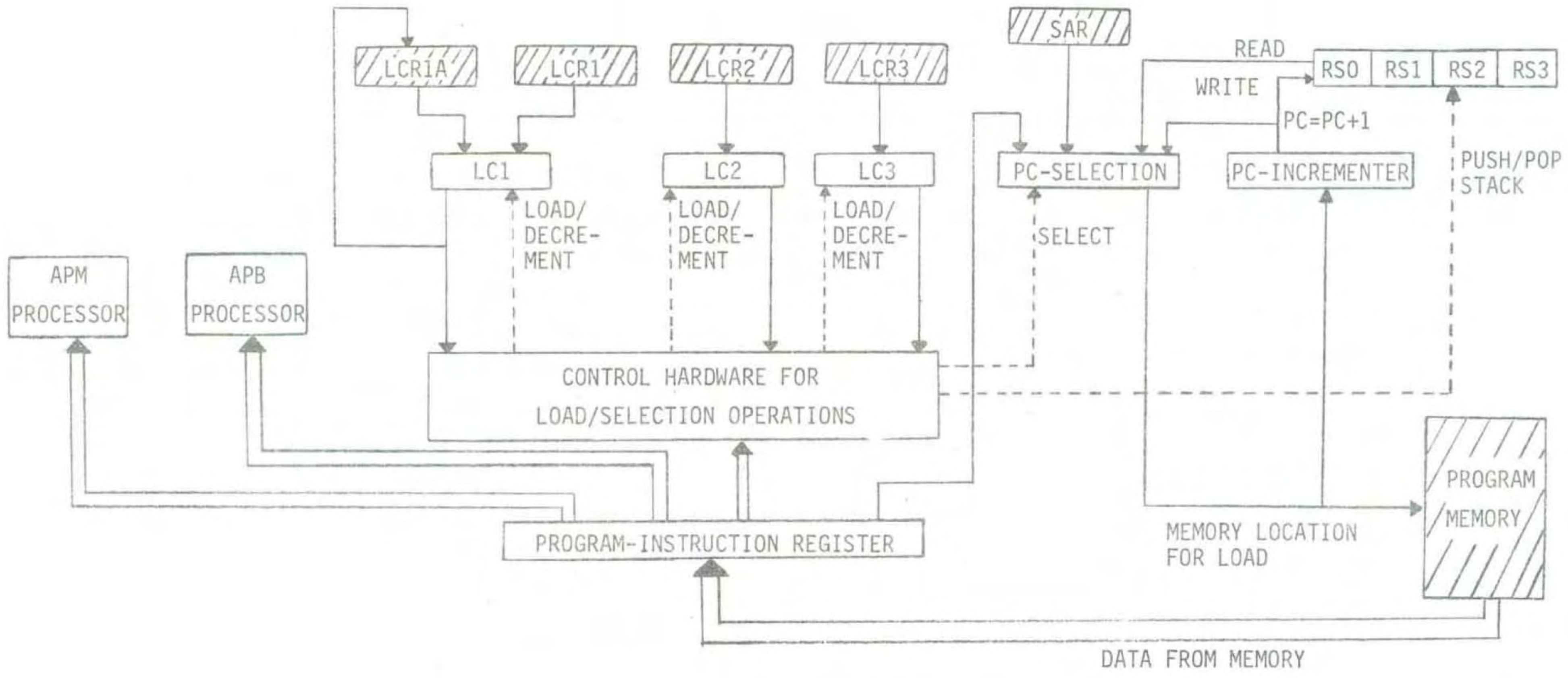


FIGURE 6. CONTROL LOGIC OF CENTRAL PROCESSING UNIT

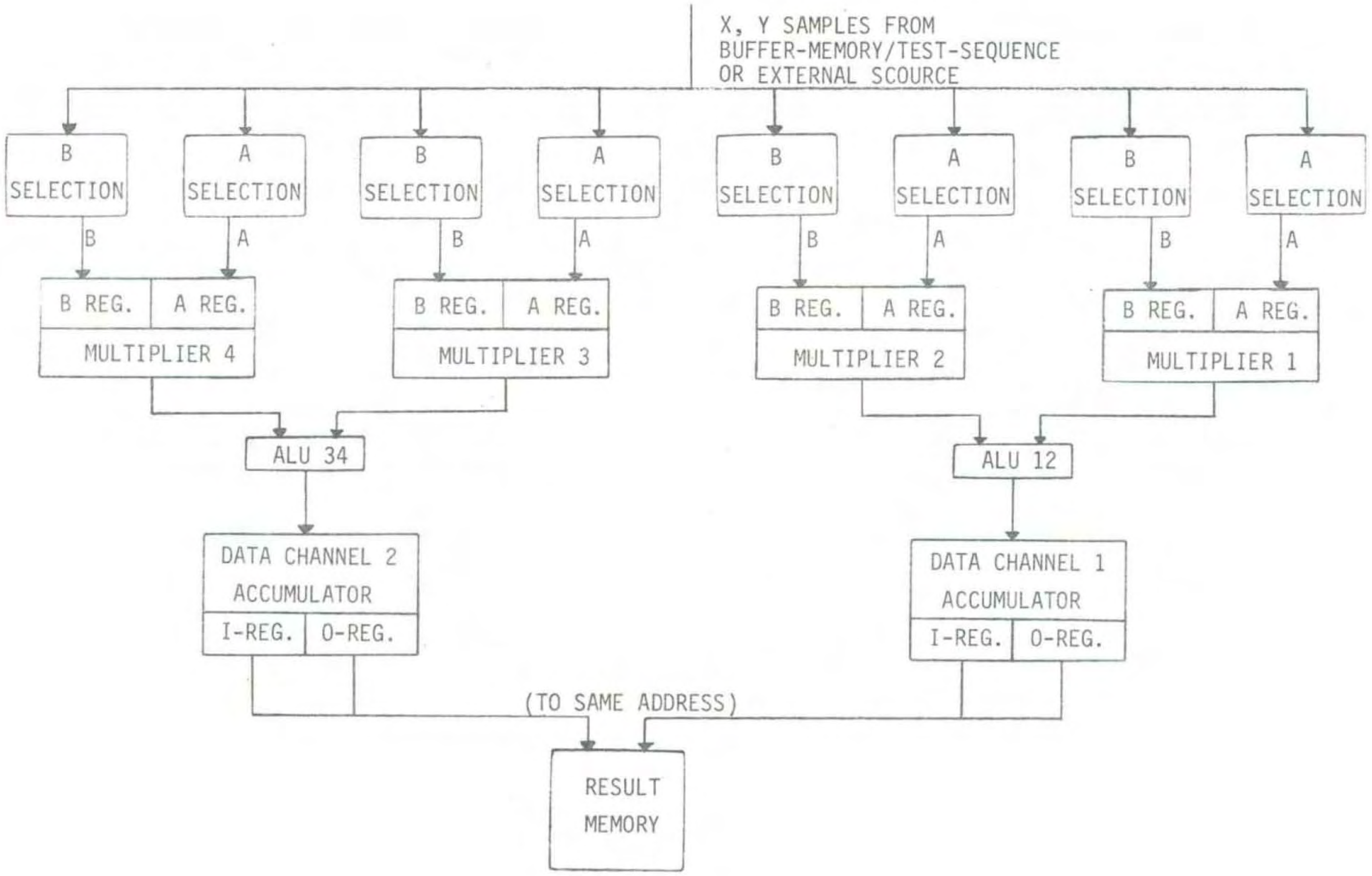


FIGURE 7. CONTROL LOGIC OF DATA PROCESSING UNIT

Program Location 0: This location is used for the idle status (correlator inactive) and should be programmed with an unconditional CONTINUE statement. In this location the PC incrementer is inhibited so the correlator is "idle" looping in location 0. When a START command is entered from the radar controller or from the function CORRLO of CORRSIM the PC value is automatically set from the SAR register and the program branches to the start point of the program. Therefore it is absolutely essential that the user has defined the SAR register with the correct start point before the correlator has been started. In location 0 it is not permitted to have a DMA-output transfer or reloading of registers in the data field controlled by the program.

Program Location 63: In this location the PC incrementer is also inhibited. This location is used with the real-time signal INTERUPT PROGRAM. See REF. 1. Note that when an instruction word is loaded into the instruction register at the end of one cycle the corresponding action on the loop counters are performed at the end of the next clock cycle. The only register not following this concept is the LCR1A which is operated (loaded) on in the mid-part of the cycle.

The X, Y samples are strobed into the A and B registers of the four multipliers and then multiplied together. The ALU12 and ALU34 (Arithmetical Logical Unit for multipliers 1 and 2, 3 and 4) then performs an operation on the multipliers and the results are then strobed into the Accumulators of Data Channel 1 and 2 which are then added together with the values in the I-registers respectively. The results are strobed into the O-registers and then written into the correlator result memory as a 64 BIT word. Note that although the strobing of the same address from/to the result memory to the I-registers and from the O-registers is defined in the same program instruction word the operation does not actually take place in the same clock cycle. Because of the pipelining structure of the correlator the values in the O-registers are actually strobed to the result memory in the next clock cycle therefore the same address in the result memory should never be read from/to in the next instruction word.

The CORRTEST function PROTEST is a software simulation of the correlator hardware. The program will execute cycle by cycle all defined micro program instructions except those of the OUT instructions and tests are performed for validity of those instructions. An output mapping of the program counter, the LIFO register stacks, the loop counters, the LCRIA register, output values of the APB and APM processors, the values of the multipliers, the values of the ALU's, the values of the data channels and an indication of whether a read/write was made from/to the result memory. At the end of the simulation a calculation is made of the required CPU time when executed in the physical system.

After entering PROTEST the program will then respond with:

```
START ON SUBROUTINE FOR SIMULATION OF PROGRAM AND ARITHMETICAL
EXECUTION


GENERAL COMMANDS AVAILABLE:
   EXIT TERMINATE THE SUBROUTINE
   FIN TERMINATES THE PROGRAM
   LIST GIVES LISTING OF DEFINED DATA-FIELD



GIVE COMMAND (FIN, EXIT, LIST, CONTINUE):
```

(_ denotes the necessary character input)

After the C has been given the next response is:

```
PRINT OF RESULTS? (TYPE Y OR N):
```

If the input is Y the next response is:

```
DO YOU WISH INTEGER OR OCTAL VALUES FOR ARITHMETICAL PART
OF SIMULATION PRINTED? (TYPE INT OR OCT):
```

If OCT is chosen only the output mapping as described above will be printed in octal. All other outputs will be in integer format. The next response will be

```
WHICH OPTION FOR DISPLAY? (PROGRAM, ARITHMETIC OR NEITHER. TYPE P,A,
OR N):
```

If the input is P then only values of the program counter, LIFO register stacks, the loop counters, the LCRIA register and output values of the APB and APM processors will be displayed in integer format.

If the input is A then only values of the program counter, multipliers, ALU's the two data channels and an indication whether a read and/or write was made from/to the result memory. The user now has the possibility of having an intėger or octal output as the next response will be:

```
DO YOU WISH INTEGER OR OCTAL VALUES DISPLAYED? (Type INT OR OCT):
```

If the input is N no output will be given when the microprogram execution is started.

The next response will be:

```
DUMP RES. MEM ON FILE CORRDATA? (TYPE Y OR N):
```

Here the user has the possibility of storing the data from the simulated result memory on file which can be used for comparison with data from the correlator or CORRTEST function ARITEST. This is particularly useful when developing new programs or for making correlator hardware tests. At the end of a successful simulation the program then generates:

```
NO PROGRAM ERRORS WERE DETECTED
CPU-TIME IN CORRELATOR IS nn USEC
```

Where nn is the real time CPU time consumption in the physical correlator.

The last response from the program is

```
CONTINUE OF SIMULATION OF NEXT TIME AVERAGE? (TYPE Y OR N):
```

Thus the user has the possibility of simulating any number of time averages.

XII SIMULATION OF CORRELATOR ARITHMETIC, USE OF FUNCTION ARITEST

The function ARITEST simulates the arithmetical hardware section of the correlator for fixed programs and uses the same registers as defined for the basic subroutines, see appendix A.

After intering ARITEST the program will respond with:

```
START ON SUBROUTINE FOR SIMULATION OF DATA PROCESSING
PRESENT PROGRAM VERSION CONSISTS OF FOLLOWING PROGRAMS:
PROG  1: POWER PROFILE PROGRAM (VERSION 1)
PROG  2: POWER PROFILE PROGRAM (VERSION 2)
PROG  3: SINGLE PULSE (NO. OF LAGS .EQ. NO. OF SAMPLES)
PROG  4: SINGLE PULSE (NO. OF LAGS .LE. NO. OF SAMPLES)
PROG  5: MULTI PULSE PROGRAM
PROG  6: CROSS CORRELATION (NO. OF LAGS. .EQ. NO. OF SAMPLES)
PROG  7: CROSS CORRELATION (NO. OF LAGS. .LE. NO. OF SAMPLES)
```

```
PROGRAM NO.?:
```

The program no. can now be entered (i.e. 1 to 7). The next response is:

```
GIVE COMMAND (FIN, EXIT, LIST, CONTINUE)
```

(_ denotes the necessary character input). The input parameters have the same meaning as described in PROTEST.

After C has been given the next response is:

```
PRINT OF RESULTS? (TYPE Y OR N):
```

If the input is Y the next response is:

```
DO YOU WISH INTEGER OR OCTAL VALUES PRINTED? (TYPE INT OR OCT):
```

If OCT is chosen only the computed results will be printed in octal. All other outputs will be given in integer format.

The next response will be:

```
DISPLAY OF RESULTS? (TYPE Y OR N):
```

If the input is Y the program will respond with:

```
        DO YOU WISH INTEGER OR OCTAL VALUES DISPLAYED? (TYPE INT OR OCT):
```

The next response is:

```
        DUMP DATA ON FILE CORRDATA? (TYPE Y OR N)
```

If the input is N then the program will respond with:

```
        COMPARISON WITH DATA ON FILE CORRDATA? (TYPE Y OR N):
```

Here a comparison with the results from PROTEST can be made. After the simulation has been made the last response will be:

```
        CONTINUE OF SIMULATION OF NEXT TIME AVERAGE? (TYPE Y OR N):
```

Thus the user has the possibility of simulating any number of time averages.

ACKNOWLEDGEMENT

I would like to thank the EISCAT Scientific Association for sponsering this work and to all of the staff in Tromsö for their help.

XIII REFERENCES

1: TERRANCE HO:	"INSTRUCTION MANUAL FOR EISCAT DIGITAL CORRELATOR" (REVISED) EISCAT Technical Notes No. 81/26, 1981

Further Literature:

HANS-JÖRGEN ALKER:	"A PROGRAMMABLE CORRELATOR MODULE FOR THE EISCAT RADAR SYSTEM". EISCAT Technical Notes No. 79/11, 1979
TERRANCE HO HANS-JÖRGEN ALKER:	"SCIENTIFIC PROGRAMMING OF THE EISCAT DIGITAL CORRELATOR. (REVISED) EISCAT Technical Notes No. 81/24, 1981
TERRANCE HO:	"POCKET MANUAL FOR PROGRAMMING THE EISCAT DIGITAL CORRELATOR. EISCAT Technical Notes No. 81/28, 1981
TERRANCE HO:	"STANDARD SUBROUTINES AND PROGRAMS FOR EISCAT DIGITAL CORRELATOR. EISCAT Technical Notes No. 81/27, 1981

APPENDIX A

The parameters which have to be defined for the basic subroutines are given here. Note that when using ARITEST the statusword must also be defined. A dummy correlator IDENT-CODE can be given in order to define the statusword.

MICRO-PROGRAM FOR DIGITAL CORRELATOR

AUIOR: TERRANCE HO DATE: 6/8/80

PROGRAM NAME: POWER PROFILE SUBROUTINE (VERSION 1)
FILE-NAME (NORD 10): PROGØ:DATA
PROGRAM DESCRIPTION:

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

DATA CHANNEL 2 MEAN VALUE ESTIMATION $M_r = \sum_{i=0}^{N-1} (X_{i+(N+D-1)(r-1)} + Y_{i+(N+D-1)(r-1)})$

WHERE N=NO. OF SAMPLES IN RANGECELL
D=OVERLAP FACTOR (=1 FOR NO OVERLAPPING AND$\leqslant$0 FOR OVERLAPPING)
r=1,2,...,M RANGECELLS FOR TIME AVERAGE

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 and 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 - 6

MICRO-PROGRAM FOR DIGITAL CORRELATOR

AUTOR: TERRANCE HO DATE: 6/8/80

PROGRAM NAME: POWER PROFILE SUBROUTINE (VERSION 1)
FILE-NAME (NORD 10): PROGØ:DATA

REGISTER NAME	REGISTER ADDRESS	PARAMETER
SAR	4	START ADDRESS OF SUBROUTINE
APB RS(15)	16,15	NO. OF SAMPLES-1 IN RANGECELL
APB RS(14)	16,14	NO. OF RANGECELLS-1 FOR TIME AVERAGE
APB RS(13)	16,13	RANGECELL INCREMENT (=1 FOR NO OVER-LAPPING OF RANGECELLS)
APB RS(12)	16,12	SAMPLE INCREMENT (NORMALLY=1)
APM RS(15)	17,15	INCREMENT (=1)

MICRO-PROGRAM FOR DIGITAL CORRELATOR

AUTOR: TERRANCE HO DATE: 6/8/80

PROGRAM NAME: POWER PROFILE SUBROUTINE (VERSION 2)

FILE-NAME (NORD 10): PROG1:DATA

PROGRAM DESCRIPTION:

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

DATA CHANNEL 2 MEAN VALUE ESTIMATION $M_r = \sum_{i=0}^{N-1} (X_{i+(N+D-1)(r-1)} + Y_{i+(N+D-1)(r-1)})$

DATA CHANNEL 1 MEAN VALUE X ESTIMATION $M_{X,r} = \sum_{i=0}^{N-1} X_{i+(N+D-1)(r-1)}$

DATA CHANNEL 2 MEAN VALUE Y ESTIMATION $M_{Y,r} = \sum_{i=0}^{N-1} Y_{i+(N+D-1)(r-1)}$

WHERE N=NO. OF SAMPLES IN RANGECELL

D=OVERLAP FACTOR (=1 FOR NO OVERLAPPING AND ≤ 0 FOR OVERLAPPING)

r=1,2,....,M RANGECELLS FOR TIME AVERAGE

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 and the LCR2 register must be reloaded with the APBRS(14) register in the main program.
4. In a particular rangecell the zero lag estimation K_r and M_r are computed first and the mean value estimation $M_{X,r}$ and $M_{Y,r}$ second.

START ADDRESS FOR PROGRAM: 1

PROGRAM-MEMORY LOCATIONS USED: 1 - 6

MICRO-PROGRAM FOR DIGITAL CORRELATOR

AUTOR: TERRANCE HO DATE: 6/8/80

PROGRAM NAME: POWER PROFILE SUBROUTINE (VERSION 2)
FILE-NAME (NORD 10): PROG1:DATA

REGISTER NAME	REGISTER ADDRESS	PARAMETER
SAR	4	START ADDRESS OF SUBROUTINE
APB RS(15)	16,15	NO. OF SAMPLES-1 IN RANGECELL
APB RS(14)	16,14	NO. OF RANGECELLS-1 FOR TIME AVERAGE
APB RS(13)	16,13	RANGECELL INCREMENT (=1 FOR NO OVER-LAPPING OF RANGECELLS)
APB RS(12)	16,12	SAMPLE INCREMENT (NORMALLY=1)
APM RS(15)	17,15	INCREMENT (=1)

MICRO-PROGRAM FOR DIGITAL CORRELATOR

AUTOR: TERRANCE HO DATE: 6/8/80

PROGRAM NAME: SINGLE PULSE SUBROUTINE (NO. OF LAGS .EQ. NO. OF SAMPLES)
FILE-NAME (NORD 10): PROG2:DATA
PROGRAM DESCRIPTION:

DATA CHANNEL 1

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

DATA CHANNEL 2

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

WHERE N=NO. OF SAMPLES IN RANGECELL

L=0,1,2,...,N-1

D=OVERLAP FACTOR (=1 FOR NO OVERLAPPING AND ≤ 0 FOR OVERLAPPING)

r=1,2,...,M RANGECELLS FOR TIME AVERAGE

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 and 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 - 6

MICRO-PROGRAM FOR DIGITAL CORRELATOR

AUTOR: TERRANCE HO DATE: 6/8/80

PROGRAM NAME: SINGLE PULSE SUBROUTINE (NO. OF LAGS .EQ. NO. OF SAMPLES)
FILE-NAME (NORD 10): PROG2:DATA

REGISTER NAME	REGISTER ADDRESS	PARAMETER
SAR	4	START ADDRESS OF SUBROUTINE
APB RS(15)	16,15	NO. OF SAMPLES-1 IN RANGECELL
APB RS(14)	16,14	NO. OF RANGECELL-1 FOR TIME AVERAGE
APB RS(13)	16,13	RANGECELL INCREMENT (=1 FOR NO OVER-LAPPING OF RANGECELLS)
APB RS(12)	16,12	SAMPLE INCREMENT (NORMALLY=1)
APB RS(11)	16,11	TEMPORARY STORAGE
APM RS(15)	17,15	RANGECELL INCREMENT (=NO. OF LAGS COMPUTED)
APM RS(14)	17,14	INCREMENT (=1)
APM RS(13)	17,13	TEMPORARY STORAGE

MICRO-PROGRAM FOR DIGITAL CORRELATOR

AUTOR: TERRANCE HO   DATE: 6/8/80

PROGRAM NAME: SINGLE PULSE SUBROUTINE (NO. OF LAGS .LE. NO. OF SAMPLES)
FILE-NAME (NORD 10): PROG3:DATA
PROGRAM DESCRIPTION:

DATA CHANNEL 1

$$\mathrm{Re}\left\{K_{L,r}\right\}=\sum_{i=0}^{N-L-1}\left(X_{i+(N+D-1)(r-1)}X_{i+L+(N+D-1)(r-1)}+Y_{i+(N+D-1)(r-1)}Y_{i+L+(N+D-1)(r-1)}\right)$$

DATA CHANNEL 2

$$\mathrm{Im}\left\{K_{L,r}\right\}=\sum_{i=0}^{N-L-1}\left(X_{i+L+(N+D-1)(r-1)}Y_{i+(N+D-1)(r-1)}-X_{i+(N+D-1)(r-1)}Y_{i+L+(N+D-1)(r-1)}\right)$$

WHERE N=NO. OF SAMPLES IN RANGECELL

$L=0,1,2,\ldots,P \quad P\leq N-1$

D=OVERLAP FACTOR (=1 FOR NO OVERLAPPING AND ≤ 0 FOR OVERLAPPING)

$r=1,2,\ldots,M$ RANGECELLS FOR TIME AVERAGE

RESTRICTIONS

MINIMUM NO. OF SAMPLES IN RANGEDATA: 1

MINIMUM NO. OF LAGS IN RANGEDATA: 2

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 and the LCR3 register must be reloaded with the the APBRS(13) register in the main program.

START ADDRESS FOR PROGRAM: 1
PROGRAM-MEMORY LOCATIONS USED: 1 - 7

MICRO-PROGRAM FOR DIGITAL CORRELATOR

AUTOR: TERRANCE HO DATE: 6/8/80

PROGRAM NAME: SINGLE PULSE SUBROUTINE (NO. OF LAGS .LE. NO. OF SAMPLES)
FILE-NAME (NORD 10): PROG3:DATA

REGISTER NAME	REGISTER ADDRESS	PARAMETER
SAR	4	START ADDRESS OF SUBROUTINE
APB RS(15)	16,15	NO. OF SAMPLES-1 IN RANGECELL
APB RS(14)	16,14	NO. OF RANGECELLS-1 FOR TIME AVERAGE
APB RS(13)	16,13	NO. OF LAGS-1 IN RANGECELL
APB RS(12)	16,12	RANGECELL INCREMENT (=1 FOR NO OVER-LAPPING OF RANGECELLS)
APB RS(11)	16,11	SAMPLE INCREMENT (NORMALLY=1)
APB RS(10)	16,10	TEMPORARY STORAGE
APM RS(15)	17,15	RANGECELL INCREMENT (=NO. OF LAGS COMPUTED)
APM RS(14)	17,14	INCREMENT (=1)
APM RS(13)	17,13	TEMPORARY STORAGE

MICRO-PROGRAM FOR DIGITAL CORRELATOR

AUTOR: TERRANCE HO DATE: 6/8/80

PROGRAM NAME: MULTI PULSE SUBROUTINE
FILE-NAME (NORD 10): PROG4:DATA
PROGRAM DESCRIPTION:

DATA CHANNEL 1 $\mathrm{Re}\left\{K_{L,r}\right\} = X_{S+r-1}X_{S+L+r-1} + Y_{S+r-1}Y_{S+L+r-1}$

DATA CHANNEL 2 $\mathrm{Im}\left\{K_{L,r}\right\} = X_{S+L+r-1}Y_{S+r-1} - X_{S+r-1}Y_{S+L+r-1}$

LET J_0=POSITION OF 1st PULSE
J_1=SAMPLE DIFFERENCE BETWEEN 1st AND 2nd PULSE
:
J_{N-1}=SAMPLE DIFFERENCE BETWEEN 1st AND Nth PULSE

WHERE $S=J_0,J_1,J_2,\dots,J_{N-2}$
$L=J_1-S,J_2-S,\dots,J_{N-1}-S$ WITH THE RESTRICTION $L>0$
$r=1,2,\dots,M$ RANGECELLS FOR TIME AVERAGE

RESTRICTIONS

MINIMUM NO. OF ELEMENT PULSES IN PULSE GROUP: 2
MAXIMUM NO. OF ELEMENT PULSES IN PULSE GROUP: 13

NOTES

1. The formulae above are given witn 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 and the LCR2 register must be reloaded with the APBRS(14) register in the main program.
4. The zero lag is not computed in this algorithm, therefore to calculate the number of lags computed use the formula: $N(N-1)/2$ where N is the number of element pulses in the pulse group.

START ADDRESS FOR PROGRAM: 1
PROGRAM-MEMORY LOCATIONS USED: 1 - 6

MICRO-PROGRAM FOR DIGITAL CORRELATOR

AUTOR: TERRANCE HO DATE: 6/8/80

PROGRAM NAME: MULTI PULSE SUBROUTINE
FILE-NAME (NORD 10): PROG4:DATA

REGISTER NAME	REGISTER ADDRESS	PARAMETER
SAR	4	START ADDRESS OF SUBROUTINE
APB RS(15)	16,15	NO. OF PULSES-1 IN RANGECELL
APB RS(14)	16,14	NO. OF RANGECELLS-1 FOR TIME AVERAGE
APB RS(13)	16,13	RANGECELL INCREMENT (NORMALLY=1)
APB RS(12)	16,12	TEMPORARY STORAGE
⋮	⋮	⋮
APB RS(1)	16,1	SAMPLE DISTANCE BETWEEN 2nd LAST AND 1st PULSE
APB RS(0)	16,0	SAMPLE DISTANCE BETWEEN LAST AND 1st PULSE
APM RS(15)	17,15	INCREMENT (=1)

MICRO-PROGRAM FOR DIGITAL CORRELATOR

AUTOR: TERRANCE HO DATE: 6/8/80

PROGRAM NAME: CROSS CORRELATION SUBROUTINE (NO. OF LAGS .EQ. NO. OF SAMPLES)

FILE-NAME (NORD 10): PROG5:DATA

PROGRAM DESCRIPTION:

DATA CHANNEL 1

$$\mathrm{Re}\left\{K_{L,r}\right\} = \sum_{i=0}^{N-L-1} \left(X_{i+(N+D-1)(r-1)}X_{i+S+L+(N+D-1)(r-1)} + Y_{i+(N+D-1)(r-1)}Y_{i+S+L+(N+D-1)(r-1)}\right)$$

DATA CHANNEL 2

$$\mathrm{Im}\left\{K_{L,r}\right\} = \sum_{i=0}^{N-L-1} \left(X_{i+S+L+(N+D-1)(r-1)}Y_{i+(N+D-1)(r-1)} - X_{i+(N+D-1)(r-1)}Y_{i+S+L+(N+D-1)(r-1)}\right)$$

WHERE N=NO. OF SAMPLES IN RANGECELL

L=0,1,2,...,N-1

S=SAMPLE DIFFERENCE BETWEEN THE TWO SETS OF DATA

D=OVERLAP FACTOR (=1 FOR NO OVERLAPPING AND ≤ 0 FOR OVERLAPPING)

r=1,2,...,M RANGECELLS FOR TIME AVERAGE

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 and the LCR2 register must be reloaded with the APBRS(14) register in the main program.
4. Only half of the correlation function can be obtained with this scheme.
5. The Single Pulse autocorrelation scheme (see PROG2:DATA) can be obtained by setting APBRS(12)=0.

START ADDRESS FOR PROGRAM: 1

PROGRAM-MEMORY LOCATIONS USED: 1 - 8

MICRO-PROGRAM FOR DIGITAL CORRELATOR

AUTOR: TERRANCE HO DATE: 6/8/80

PROGRAM NAME: CROSS CORRELATION SUBROUTINE (NO. OF LAGS .EQ. NO. OF SAMPLES)
FILE-NAME (NORD 10): PROG5:DATA

REGISTER NAME	REGISTER ADDRESS	PARAMETER
SAR	4	START ADDRESS OF SUBROUTINE
APB RS(15)	16,15	NO. OF SAMPLES-1 IN RANGECELL
APB RS(14)	16,14	NO. OF RANGECELLS-1 FOR TIME AVERAGE
APB RS(13)	16,13	RANGECELL INCREMENT (=1 FOR NO OVER-LAPPING OF RANGECELLS)
APB RS(12)	16,12	START ADDRESS OF 2nd FIELD-START ADDRESS OF 1st FIELD
APB RS(11)	16,11	SAMPLE INCREMENT (NORMALLY=1)
APB RS(10)	16,10	TEMPORARY STORAGE
APM RS(15)	17,15	RANGECELL INCREMENT (=NO. OF LAGS COMPUTED)
APM RS(14)	17,14	INCREMENT (=1)
APM RS(13)	17,13	TEMPORARY STORAGE

MICRO-PROGRAM FOR DIGITAL CORRELATOR

AUTOR: TERRANCE HO DATE: 6/8/80

PROGRAM NAME: CROSS CORRELATION SUBROUTINE (NO. OF LAGS .LE. NO. OF SAMPLES)

FILE-NAME (NORD 10): PROG6:DATA

PROGRAM DESCRIPTION:

DATA CHANNEL 1

$$\mathrm{Re}\left\{K_{L,r}\right\} = \sum_{i=0}^{N-L-1} \left(X_{i+(N+D-1)(r-1)}X_{i+S+L+(N+D-1)(r-1)} + Y_{i+(N+D-1)(r-1)}Y_{i+S+L+(N+D-1)(r-1)}\right)$$

DATA CHANNEL 2

$$\mathrm{Im}\left\{K_{L,r}\right\} = \sum_{i=0}^{N-L-1} \left(X_{i+S+L+(N+D-1)(r-1)}Y_{i+(N+D-1)(r-1)} - X_{i+(N+D-1)(r-1)}Y_{i+S+L+(N+D-1)(r-1)}\right)$$

WHERE N=NO. OF SAMPLES IN RANGECELL

$L=0,1,2,\ldots,P$ $P \leqslant N-1$

S=SAMPLE DIFFERENCE BETWEEN THE TWO SETS OF DATA

D=OVERLAP FACTOR (=1 FOR NO OVERLAPPING AND $\leqslant 0$ FOR OVERLAPPING)

$r=1,2,\ldots,M$ RANGECELLS FOR TIME AVERAGE

RESTRICTIONS

MINIMUM NO. OF SAMPLES IN RANGEDATA: 1

MINIMUM NO. OF LAGS IN RANGEDATA: 2

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 and the LCR3 register must be reloaded with the APBRS(13) register in the main program.
4. Only half of the correlation function can be obtained with this scheme.
5. The Single Pulse autocorrelation scheme (see PROG3:DATA) can be obtained by setting APBRS(11)=0.

START ADDRESS FOR PROGRAM: 1

PROGRAM-MEMORY LOCATIONS USED: 1 - 8

MICRO-PROGRAM FOR DIGITAL CORRELATOR

AUTOR: TERRANCE HO DATE: 6/8/80

PROGRAM NAME: CROSS CORRELATION SUBROUTINE (NO. OF LAGS .LE. NO. OF SAMPLES)
FILE-NAME (NORD 10): PROG6:DATA

REGISTER NAME	REGISTER ADDRESS	PARAMETER
SAR	4	START ADDRESS OF SUBROUTINE
APB RS(15)	16,15	NO. OF SAMPLES-1 IN RANGECELL
APB RS(14)	16,14	NO. OF RANGECELLS-1 FOR TIME AVERAGE
APB RS(13)	16,13	NO. OF LAGS-1 IN RANGECELL
APB RS(12)	16,12	RANGECELL INCREMENT (=1 FOR NO OVER-LAPPING OF RANGECELLS)
APB RS(11)	16,11	START ADDRESS OF 2nd FIELD-START ADDRESS OF 1st FIELD
APB RS(10)	16,10	SAMPLE INCREMENT (NORMALLY=1)
APB RS(9)	16,9	TEMPORARY STORAGE
APM RS(15)	17,15	RANGECELL INCREMENT (=NO. OF LAGS COMPUTED)
APM RS(14)	17,14	INCREMENT (=1)
APM RS(13)	17,13	TEMPORARY STORAGE

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 Correlator (revised)
EISCAT Technical Note 81/24