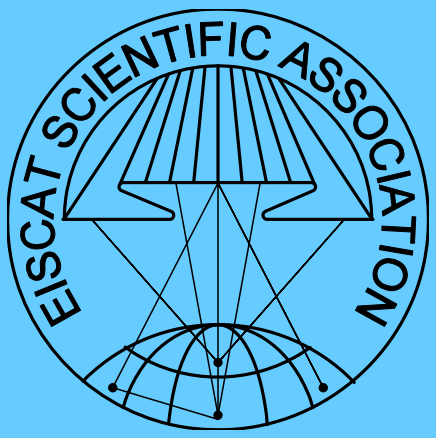
79/9




**EISCAT**
**TECHNICAL**
**NOTES**

# PROGRAM CORRSIM:
# SYSTEM FOR PROGRAM DEVELOPMENT AND SOFTWARE SIMULATION OF EISCAT DIGITAL CORRELATOR

USER'S MANUAL

Hans-Jørgen Alker

**KIRUNA**
**Sweden**

EISCAT Technical Note 79/9

# PROGRAM CORRSIM:
# SYSTEM FOR PROGRAM DEVELOPMENT AND SOFTWARE SIMULATION
# OF
# EISCAT DIGITAL CORRELATOR

## USER'S MANUAL

by

Hans-Jørgen Alker

EISCAT Scientific Association
S-981 01 Kiruna, Sweden
March 1979

This report has also been published by the Institute of Mathematical and Physical Sciences, University of Tromsö, October 1978.

CONTENTS

## 1. INTRODUCTION

The program CORRSIM is a multi-function software package for development and testing of user-defined correlator programs. CORRSIM also includes software-drivers for program-loading of correlator and direct-memory-access (DMA) handling of data from the external correlator source.

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

The EISCAT digital correlator is a multi-processor system, and for real-time control and interfacing of separate internal functions, the programming is based on micro-instruction definitions in absolute form (octal coded). It is assumed that the user is familiar with the basic instruction-set of the correlator before working with the CORRSIM. This information can be found in the report: "Instruction Manual for the EISCAT Digital Correlator". This present report gives an introduction to the use of CORRSIM. For further program details, the report: "Program CORRSIM: Program Documentation" is available.

## 2. 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 includes the function-selection program SIMUL and the subroutine EDITOR which handles the correlator program setup and modifications. The program blocks SYMTAB, PROTEST, ARITEST and CORRIO are constructed with subroutines for:

- decoding of user-defined micro-instructions,
- real-time simulation of correlator program execution,
- simulation of correlator arithmetic for fixed programs,
- interactive communication with correlator hardware.

The system includes a set of data-files storing programs, data from simulation and DMA-transfer from correlator. The CORRSIM files PROG 1-PROG 9 (read access only) realizes a library of fixed programs which will be identical to the fixed programs in the correlator. The CORRSIM files STATUS and DMADUMP are loaded from the RT-program DMAREAD, automatically started by DMA-end interrupt. These files have access: read/write. The CORRSIM files X-DATA and Y-DATA stores the test-data values, which are also fixed in the correlator. These data are the basis for the simulation of correlator arithmetic. User access is read. The user-defined file PROG0 can be used as "scratch-file" storing program/data defined by user. This file together with the simulation data-file DATA will automatically be created in user-file directory if they are referenced without being created. The user must by himself be sure that space is available in user-directory Typical size for DATA is 8 pages and for PROG0 2 pages. After CORRSIM is started a program COMMON area is created storing an "image" of the correlator data-field (programmable data-registers) and program-field (program-memory), together with the test-data sequence. This COMMON area is accessed by all program blocks.

The main operational mode when running the CORRSIM is by interactive communication with the program, and it is recommended to use a display terminal. Print-commonds are included in CORRSIM for program and simulation documentation.

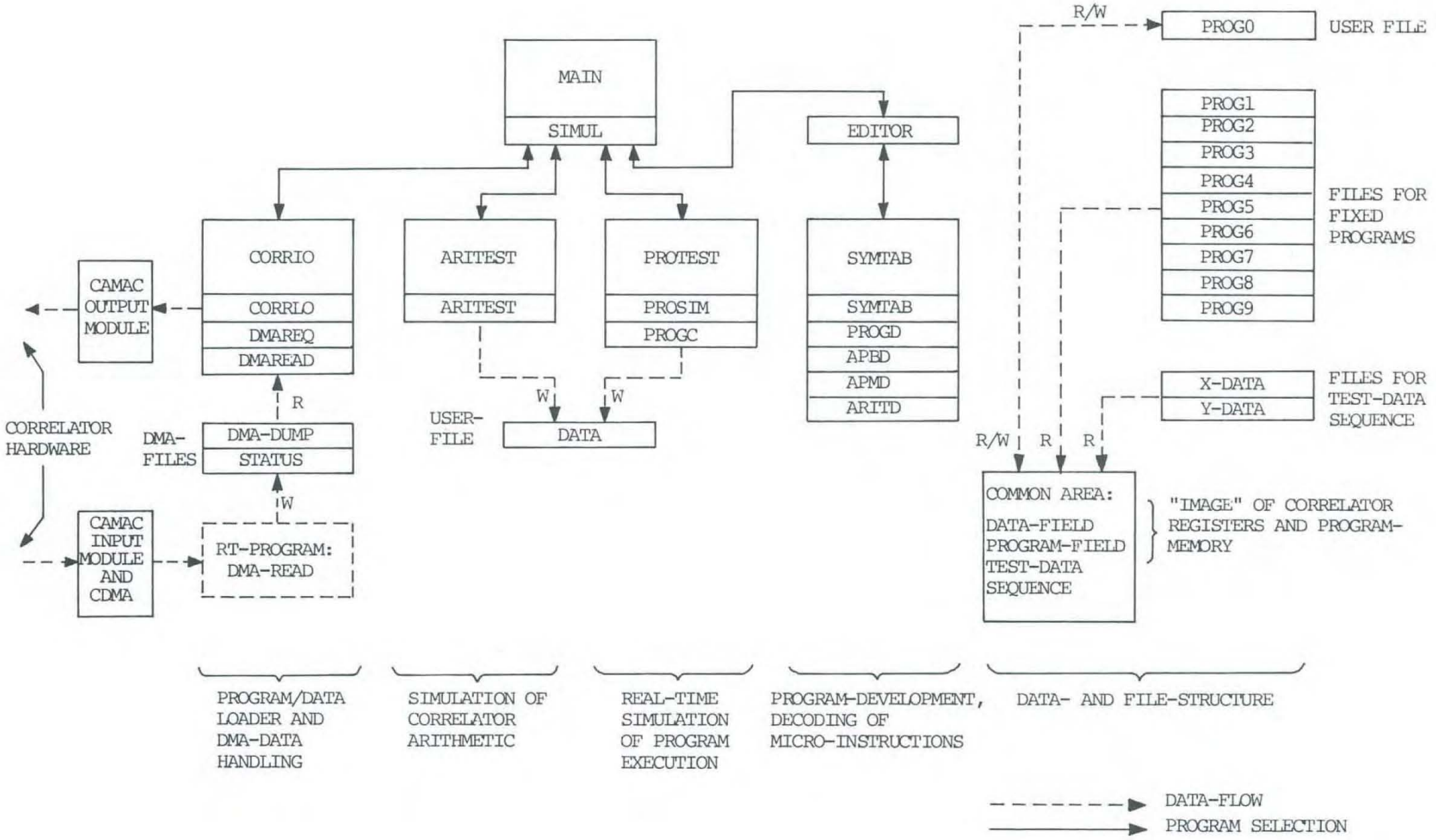



Figure 1. CORRSIM STRUCTURE

NOTE THAT ALL NUMMERIC INPUT/OUTPUT IN CORRSIM ARE OCTAL, THE ONLY EXCEPTIONS ARE FILE-NUMBERS 8 AND 9 REFERENCING CORRSIM PROGRAM-FILES PROG8 AND PROG9.

## 3. START OF CORRSIM

The program is stored on the user-file (CORR)SIM and is started with the SINTRAN-command:

```
RECOVER (CORR)SIM
```

or simplified:

```
(CORR)SIM.
```

The response from the program is

```
**** PROGRAM (CORR)SIM IS ACTIVE ****

THE SIMULATOR CONSISTS OF 4 MAIN SUBROUTINES:
SUBROUTINE EDITOR FOR PROGRAM SET-UP AND MODIFICATION.
SUBROUTINE CORRLO FOR I/O-COMMUNICATION WITH THE CORRELATOR.
SUBROUTINE PROTEST FOR SIMULATION OF PROGRAM-EXECUTION.
SUBROUTINE ARITEST FOR SIMULATION OF FIXED CORRELATOR PROGRAMS.

THE SIMULATOR IS TERMINATED BY TYPING FIN.
```

The program then asks for function-selection:

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

(_:denotes neccessary character input).

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

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

Whenever a non-legal command is typed on the terminal, the response is:

ILLEGAL COMMAND.

All nummeric input/output of CORRSIM is octal, and the format for input-values are 6 octal digits, which requires that all nummeric input with less than 6 octal digits should be terminated by ",".

When the program is started with RECOVER (CORR)SIM or (CORR)SIM, the initial phase is resetting all main internal program parameters. This means that after a FIN-command has been generated, all earlier set parameters are lost. This problem can be avoided by stopping the CORRSIM with user-break and SINTRAN-commands are available. The program can be restarted from break-point with earlier defined values with the SINTRAN-command

CONTINUE.

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

ERROR-MESSAGES FORM CORRSIM IN THE START PHASE:

| | | |
|---|---|---|
| SAMPLE-FILE IS NOT SUCCESSFULLY OPENED | : | CORRSIM X-DATA OR Y-DATA file can not be opened. The program is terminated. |
| ERROR IN READING FROM DATA-FILE | : | system error in reading from X,Y-DATA files, the program is terminated. |
| ILLEGAL COMMAND | : | An illegal command has beer entered from the terminal. |

## 4. DEVELOPMENT OF CORRELATOR PROGRAMS, USE OF FUNCTION EDITOR

When the EDITOR-command is given, the editor starts with a listing of all available editor-commands. The editor is ready for accepting a command when

C*

is typed on the terminal.

The basic principle for developments of correlator programs is a creation of an "image" of the correlator programmable registers and program-memory. This "image" is octal 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 16 bits. One correlator micro-instruction is 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 octal ADDRESS-parameter, individual register-stack and memory-module locations are identified by an octal SUB-ADDRESS-parameter. The data- and program-field structures are given in Figure 2 together with octal addresses/subaddresses and value-ranges. By the program structure given, 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 created in editor is kept in common and are used by all other function programs in the CORRSIM. Together with the "image" in program-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:

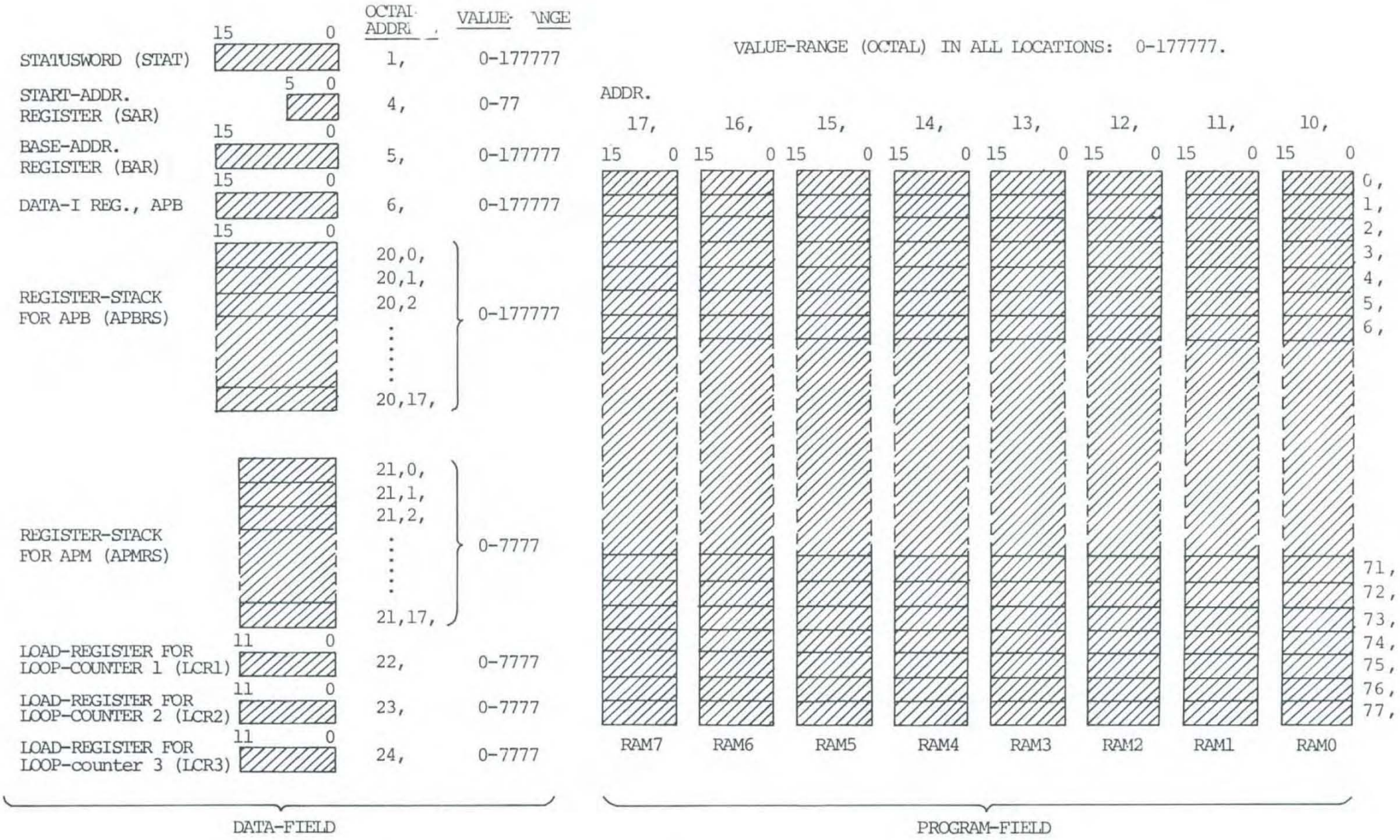



Figure 2. COMPUTER "IMAGE" OF CORRELATOR DATA/PROGRAM-LOCATIONS

SET-VALUE COMMANDS: for loading and definition of 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.

READ DATAFILE : input from user-defined file or fixed program setup from CORRSIM files.

WRITE DATAFILE : output of defined values to user-file.

## 4.1. SET-VALUE COMMANDS

There are two ways of entering data to the different registers/ program-memory locations of the correlator

Command SDP (set data - or program-field):

In this mode the registers of the data-field can be defined, and each memory-module RAM0, RAM1,.... is regarded as groups of registers, each group contains $77_8$ 16-bit registers. Each register is addressed by the sub-address parameter.
The CORRSIM response to the SDP-command is:

GIVE ADDRESS, SUBADDRESS, (TYPE O FOR END):

The register octal address (and subaddress) is then entered with nummeric field-termination with ",". After cheeking the address and subaddress values, the response will be (if address/subaddress is valid):

GIVE DATA:

The octal value can then be entered. After checking the value, CORRSIM is recycling asking for new address-input. The loading sequence is terminated by entering "0", and the response is:

nn REGISTERS/MEMORY-LOCATIONS ARE DEFINED

*C

where nn is the octal number of register/memory-locations defined in the data/program-field.

Note that whenever subaddress is not required (the single registers in the data-field), it is not neccessary to type "0". Whenever subsequent locations in register-stacks or memory-modules should be loaded with the same value, it is possible to use the address-forma

address, lower subaddress, upper subaddress,

which loads all referenced subaddresses.

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 arithmetic. |
| ACC | instruction for accumulators. |
| I/O | instruction for input/output bus handling. |
| OUT | DMA output and display instruction. |

The different FUNCTION-names corespond to the correlator main internal operations processed in parallel. Because the different instructions are mixed in the 128-bit instruction word, it is recommended to use the SPS-command for program construction.

The program response to the command is:

GIVE MEMORY-LOCATION, (TYPE 100 FOR END):

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

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

where the parameters-names follow the structure given in the instruction-manual, ref. [1]. After the octal values of the parameters is entered (must be on the same line), parameter-values are tested and a display of the new instruction word follows with earlier defined instruction words at lower memory-locations (at most 15 words are displayed). The loading sequence is repeated intil termination address 100 is given. The SPS loading-sequence is terminated with the number of present function instructions defined. The repeat loading format lower, upper memory-location can be used when address is entered.

Note that the SPS-command only affects the bits in the present instruction word. If one memory-location is shared with different function instructions, bits corresponding to other functions are preserved. If these are not defined, they will be set to zero.

## 4.2. DELETE COMMANDS

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

Command D: Delete all defined parameters.

Command DD (delete from data-field):

The response from the program is:

GIVE ADDR., SUBADDR., FOR DELETE FROM DATAFIELD (TYPE O FOR END)

Repetetive subaddress-field is allowed.

Command DP (delete from program-field):

uses the same structure as for command DD.

Note that loading zero into a register/mem. location do not result in deleting the address-reference from the table of defined parameters. This table is used when parameters are transferred to the correlator in the subprogram CORRLO.

## 4.3. LIST COMMANDS

The following commands are available

Command LD (list data-field):

gives display of all defined registers in the data-field.

Command LP (list program-field):

The program response is:

GIVE LOWER, UPPER, VALUE IN MEMORY FOR DISPLAY:

When only one address-parameter is given, the memory-location in RAM0, RAM1, ..... is referenced, and all values are displayed if one of the location in the modules are defined, otherwise the program displays:

NO LOCATIONS IN PROGRAM-FIELD DEFINED

indicating that the referenced location is not defined. When lower, upper values are given, all defined locations within the address-range are displayed. If no defined locations are found in the address-range, the program-response above is displayed. Default address values (0,0) gives display of all defined locations in the program-memory.

Command LPS: FUNCTION (list program-function instructions separate):

where the legal function-names are given under SPS-command.
The address format is as for LP-command.

Command LPD: FUNCTION (list program-function instructions decoded):

The function-names is referenced above, with the difference that ARI and ACC function names, give the same decoding. The function name I/O is not included.

The LPD option gives decoding of a function instruction and display in high-level "assembly"-language the internal hardware correlator function. Test of legal instruction word value is performed.

## 4.4. PRINT COMMANDS:

The response to a print-command is:

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

and a text-string can be entered for text-heading on the print.

Command PDP (print data- and program-field):

gives print of all defined locations in data/program-field.

Command PPS (print program-function instructions separate):

gives print of all defined instructions for all functions.

Command PPD (print program-function instructions separate):

gives print of all defined instructions decoded for all functions.

Print on the line-printer starts after exit from editor.

## 4.5. CORRSIM FILE-HANDLING COMMANDS:

Command W or W 0:

0 denotes file no. 0, and can store defined values in the data-field and, if specified by the user, location area of the program-memory. After write-command has been entered, the program asks for:

FILE IDENTIFICATION NAME:

where a text-string of max. 80 characters can be loaded as a heading in the file. The file format will be for defined locations:

```
(TEXT-STRING)
(addr.), (subaddr.),          both parameters:  2 octal digits
(data-value)                  parameter:  6 octal digits
(addr.), (subaddr.),
(data-value)
    :
    :
    :
(addr.), (subaddr.),
(data-value)
0,
EOF
```

Note that a "0" is loaded at the end for indication of end. The file 0 is user-defined and must have the following name in the file-directory:

PROG0:DATA.

If the file is not created by the user before the W-command is issued, the file will be automatic created in the user file-directory.

Command R (I) (I: value from 0 to 9):

I denotes the file number, and the data-format on file must have the structure as given above. I=0 denotes user-defined file PROG0:DATA and must be created in user file-directory either by CORRSIM W-command or ordinary SINTRAN-command before starting the CORRSIM.

After the R (I) command is given, the program displays:

GIVE ENTRY-POINT IN CORRELATOR MEMORY:

Default value (0) corresponds to the following:
All data-values referencing program-memory will be loaded into locations given by the subaddress-parameter.
Entry-point (positive or negative) different from zero will result in the following loading to program-memory:
All data-valurs referencing program-memory will be loaded into locations given by the subaddress-parameter plus the entry-point value
Program-values referencing location 0 on file will be loaded to location 0.
Note the following: Program-instructions with JUMP-ADDRESS value equal to one of the locations which have been loaded into a modified address-location, will also be modified with the entry-point value.

By use of the entry-point parameter all referenced locations on file different from zero will effectively be address-relative to user-defined entry-point value, and moving program-sections from one memory area to another are possible.

File numbers 1 to 9 are defined by CORRSIM and will store fixed programs (write-access by user is inhibited).

## 4.6. ERROR-MESSAGES:

### ERROR-MESSAGES FROM EDITOR:

| | | |
|---|---|---|
| ILLEGAL COMMAND | : | Illegal command after C*. |
| ADDRESS-VALUE OUT OF BOUNDS | : | Address-value referencing non-existing register/mem.-location entered. |
| DATA-VALUE OUT OF BOUNDS | : | Register with less than 16 bits has been entered with a too large value, or a illegal octal instruc tion-value has been entered after SPS-command. |
| NOT SUCCESSFULLY OPENING OF L-P | : | A print-command has been entered but line-printer can not be opene |

DATA-FILE NOT SUCCESSFULLY OPENED : A read or write command to file has been entered but the referenced file can not be opened.

ERROR IN ADDRESS-VALUE FROM FILE : An illegal address-value is entere from the file .

ERROR IN SUBADDRESS-VALUE FROM FILE: Illegal subaddress-value from fil

ERROR IN READING FROM DATA-FILE : Error has been detected on transfer from file.

ERROR IN DATA-VALUE FROM FILE : To large data-value from file.

ERROR IN WRITING TO DATA-FILE : Error has been detected on transfer to file.

ERROR-MESSAGES FROM SPD-COMMAND:

*WARNING: START-ADDRESS OF PROGRAM IS NOT DEFINED* : SAR-register is not defined.

xxxx ERROR: PROGRAM LOCATION O CAN NOT BE USED AS START REFERENCE xxxx: SAR-register is defined with the illegal value O.

xxxx ERROR: THIS REGISTER CAN NOT BE REDEFINED xxxx : Register-reference can not be reloaded by program.

xxxx ERROR: REGISTER IS REDEFINED IN IDLE STATUS xxxx : Registers can not be reloaded by program in location O.

xxxx ERROR: ILLEGAL STATEMENT xxxx: Illegal octal instruction value.

xxxx ERROR: INSTRUCTION NOT PROPERLY DEFINED xxxx : Missing defined RAM-locations in instruction word.

xxxx ERROR: SYSTEM-CLOCK IS INHIBITED IN IDLE STATUS xxxx : Not allowed instruction for location O.

xxxx ERROR: DATA-READY IS GENERATED TO THE COMPUTER IN IDLE STATUS xxxx: Output transfer is generated in O and is an illegal instruction.

xxxx ERROR: SYSTEM-CLOCK IS INHIBITED
BUT DATA-READY IS NOT TRANSFERRED TO THE COMPUTER xxxx : Illegal matching of transfer-signals.

xxxx ERROR: OUTPUT-TRANSFER IS INITIATED BUT TRANSFER-MODE IS NOT SELECTED xxxx : Illegal matching of transfer-signals.

## 5. 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. The main control unit in the correlator is the hardware for controlling the μ-program execution. A programming model of this system is sketched in Figure 3. The program-instruction word read from the memory contains sub-instructions 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 realized 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 data-field. Also a temporary storage-register, LCR1A, can be used for storing values from LC1. A four-level register-stack with LIFO (last-in-first-out) structure for storing address-values (back-loop-branching). The next PC-value can be taken from one of four sources:

From the PC-incrementer (realizes a continue statement).
From the register-stack (return statement).
From address in the instruction word (go to or jump statment).
From SAR-register (start reference 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. The available instruction-set for the program control function can be found in ref. [1]. All locations in the program-memory can be used for programming but two locations are used for special purposes:

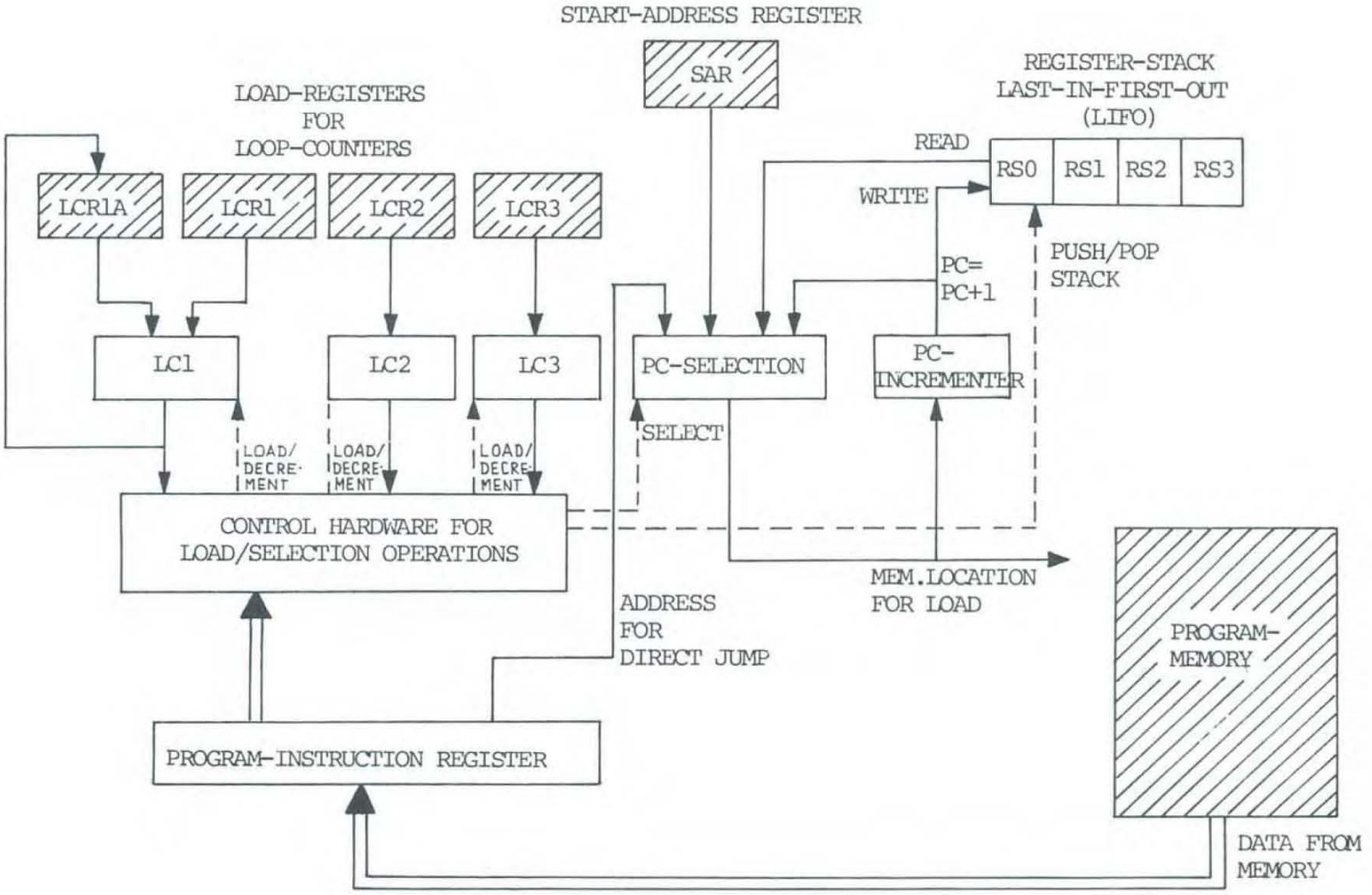



Figure 3. CONTROL LOGIC FOR INTERNAL μ-PROGRAM EXECUTION.

Program-location 0:

This location is used for idle status (correlator inactive) and should be programmed with a unconditionally CONTINUE-statment. 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 the function program CORRLO of the CORRSIM, the PC-value automatically is set from the SAR-register and the program branches to the start-point of the program. Be sure that the SAR-register has a proper definition when the correlator is started. In program location 0 it is not permitted to have DMA-output transfer or reloading of registers in the data-field controlled by program.

Program-location $77_8$:

In this location the PC-incrementer is also inhibited. This location is to be used with the real-time signal INTERRUPT PROGRAM, see ref.[1]

Note that the program control system is a syncronous system operated at the end of the clock-interval. This means that when a instruction word is loaded in the instr. register at the end of one cycle, corresponding action of 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) in the mid-part of the cycle.

The CORRSIM function PROTEST is a software simulation of the program execution from the user-defined values in the data- and program-field. The program simulates cycle-by-cycle the internal operations of the program control hardware and gives as an output a mapping of the program-counter together with present values of the loop-counters and the register LCR1A for every clock-cycle of the system. Together with the program-map, test os also performed on all reference registers in the data-field for assignments of values and for endless looping on the same instruction. At the end of the simulation, a calculation is made of required CPU-time when executed in the physical system.

After entering the function PROTEST the program displays a text-string giving information of general commands available, and then asks if simulation results should be printed on the line-printer. The program then generates the following text:

```
GIVE TRIGGER CONDITION FOR DISPLAY (PRINT)
TRIGGERPOINTS AVAILABLE: T CLOCK-INTERVAL
                         P PROGRAM-COUNTER
                         1 LOOP-COUNTER 1
                         2 LOOP-COUNTER 2
                         3 LOOP-COUNTER 3
S GIVES NEW PROGRAM SIMULATION
GIVE TRIGGER CONDITION:
```

The trigger-point given (T,P,1,2 or 3) by the user, is the function which triggers the output to display and print. After trigger function is entered information about "trigger-window" is required. The following concept is used:

| | | |
|---|---|---|
| For T | : | trigger-window is given by lower, upper octal value of the clock-cycle number. |
| For P,1,2 and 3 | : | trigger-window is given by trigger-value of the selected function, and number of clock-cycles to be displayed after the trigger-value. |

Default-values (0,0) when entering trigger-values gives display of all clock intervals. After trigger-conditions have entered the program asks if simulation-data also should be transferred to a user-defined data-file. This data-file name should be:

```
DATA:DATA.
```

If the user has not created the file before starting CORRSIM, the program automaticly creates it.

The display/print format during execution is:

```
TIME  PROGRAM-   RS0  RS1  RS2  RS3  LC1  LCR1A  LC2  LC3  NEXT
      LOCATION                                              PC
```

where:

| | |
|---|---|
| TIME | : no. of clock interval |
| PROGRAM-LOCATION | : present instruction in instr. register |
| RS0, ..., RS3 | : present register-stack values (RS0 is the first reg. in the LIF0) |
| LC1, ..., LC3 | : present values in the loop-counters and LCR1A |
| NEXT PC | : next progr. location to be read at the end of the clock interval. |

The data stored on file is only TIME and PROGRAM-LOCATION.
All displayed values are octal. When the program loops on the same PC-value the text-strong generated is:

LOOP-COUNTER (I) IS DECREMENTED

and the display starts again when a new PC-value is generated.
Note that a complete simulation from start to end is performed independently of trigger conditions. This is used only for display.
When the simulation is successful, the program generates:

NO PROGRAM ERRORS WERE DETECTED
CPU-TIME IN CORRELATOR IN nn µSEC.

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

## 5.1. ERROR-MESSAGES FROM PROTEST:

| | |
|---|---|
| ILLEGAL COMMAND | : Illegal input generated by the user. |
| NOT SUCCESSFULLY OPENING OF L-P | : Same as for editor. |
| NOT SUCCESSFULLY OPENING OF DATA-FILE | : File DATA can not be opened. |
| ERROR IN WRITING TO DATA-FILE | : Error occurring in data,transfer to file. |

| | | |
|---|---|---|
| PROG.LOC. (nn) HAS MISSING DEFINITION | : | RAM-modules storing program-instructions are not defined for location nn. |
| ERROR IN PROGRAM-LOCATION 00, CONDITIONAL TEST IN IDLE-STATUS | : | Illegal condition-code in loc. 00 |
| ERROR IN PROGRAM-LOCATION 00, LOOP-COUNTER IS DECREMENTED IN IDLE-STATUS | : | Decrement instruction in loc. 00. |
| ERROR:SAR=0 IS NOT A START-ADDRESS | : | SAR-value illegal. |
| ERROR IN PROGRAM-LOCATION nn, COUNTER (I) IS NOT DEFINED | : | Loop-counter (I) is referenced at loc. nn but is not assigned a value. |
| ERROR IN PROGRAM-LOCATION nn, COUNTER-REGISTER (I) IS NOT DE-FINED | : | LCR (I) is not defined, referenced at location nn. |
| ERROR IN PROGRAM-LOCATION nn, COUNTER-REGISTER LCR1A IS NOT DEFINED | : | LCR1A is referenced at loc. nn but is not assigned a value. |
| ERROR IN PROGRAM-LOCATION nn, ILLEGAL STATEMENT IN CONDITIONAL TESTING | : | Cond.code illegal at loc. nn. |
| WARNING: IN PROGR. LOC. nn, REGISTER-STACK VALUE LOST | : | 4 levels of return addresses are exceeded at loc. nn. |
| ERROR IN PROGRAM-LOCATION nn, REGISTER-STACK VALUE NOT DEFINED | : | Return address referenced at loc. nn, but no values in register-stack. |
| FATAL ERROR: NO TEST ON LOOP-COUNTER (I) IN PROGRAM LOC.nn | : | Program looping in location nn and LC(I) is decremented, but no test on this counter, causing endless loop. |

FATAL ERROR: PROGRAM STOP AT LOC.nn : Endless program loop at loc. nn.

All simulation errors except WARNING terminate the program simulation.

## 6. INTERACTIVE COMMUNICATION WITH THE CORRELATOR, USE OF FUNCTION CORRIO.

The CORRIO-function is a software system which connects the CORRSIM program to the correlator hardware. The input/output (I/O) transfer is established by use of standarized CAMAC interface modules, and the CORRIO-function uses ordinary CAMAC-calls when transferring/ receiving data. The physical interface structure is shown in Figure 4. For loading the correlator-system the CAMAC module 9043 (dual output register) is used. 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 be defined before a loading is started. The ident-code is a select-command and is located in the address transferred. The software system checks that response is given by the selected correlator module. The A-register bus is used both for address and data and a 16-bit data-word is transferred in two cycles: 1.cycle isthe transfer of ident-code together with the internal register-address, the 2. cycle is the data-transfer. The CAMAC B-register bus transmits neccessary control-commands including computer start-command of correlators and MASTER-RESET signal. This control bus is common for all correlator modules. Input transfer from the correlator system is trough the CAMAC module 9041 (dual input register) where only the A-register is used. The CAMAC input module is under control of the CDMA module for direct-memory access (DMA) of correlator data to the computer memory. Note that it is possible to take test-data from the correlator-module during micro-program execution and this data can be used for debugging of micro-programs with test-data from the physical system. Output transfer of data from correlators are controlled by the actual loading micro-programs.

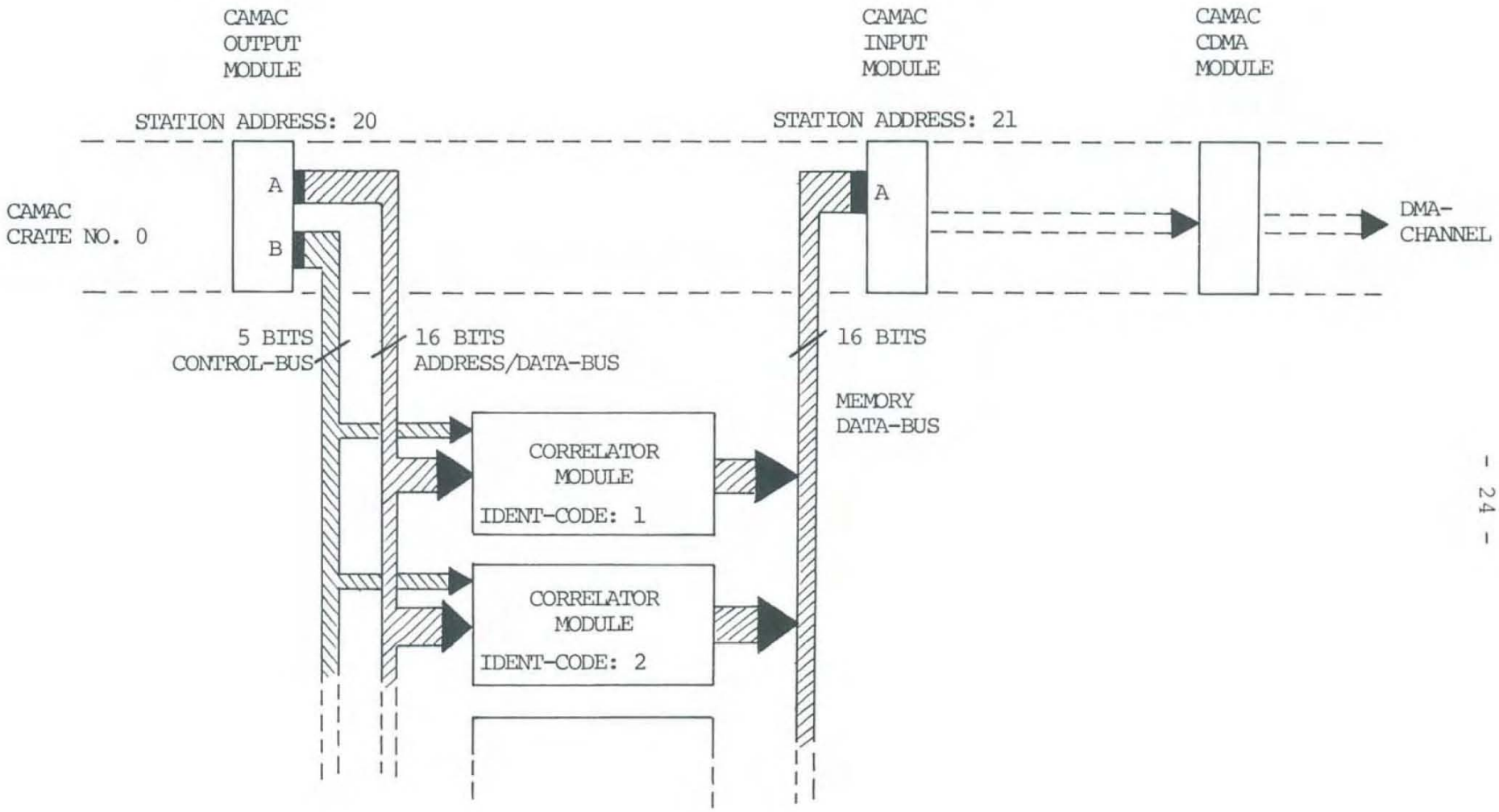



Figure 4. INTERFACE BETWEEN CORRELATOR SYSTEM AND COMPUTER.

The CORRSIM subprogram is structurized as a command-processor which, on input from the user, starts either RT-programs, CAMMAC-calls or ordinary "back-ground" program execution. A flow-diagram describing the system is shown in Figure 5. After the function CORRIO is called, the response is a display of the available commands. A command can be entered when CORRIO is displaying

```
L*GIVE COMMAND:
```

indicating that the command-processor is active.

An error-situation may arise when the CORRIO function is selected and power is not turned on the CAMAC create and/or the correlator system. When the CAMMAC create is inactive, an error message is printed on the console:

```
(time-code) ERROR 37;  IOX ERROR
ADDRESS: 46236; LEVEL (DEC.): 1
(time-code) ERROR 37;  IOX ERROR
ADDRESS: 46203; LEVEL (DEC.): 1
```

This error-message will be printed out each time a CORRIO call is made. The last error-message will also be printed each time an exit is made from the CORRIO. When the power is not turned on the correlator, the user get the message:

```
NO COMMUNICATION WITH CORRELATOR
```

This situation terminates the CORRIO function and a return to CORRSIM is made.

If the correlator is active when the CORRIO is selected, the message:

```
CORRELATOR IS RUNNING
```

is displayed. This is not an error situation and CORRIO commands can be executed. In the following subsections the commands are described.

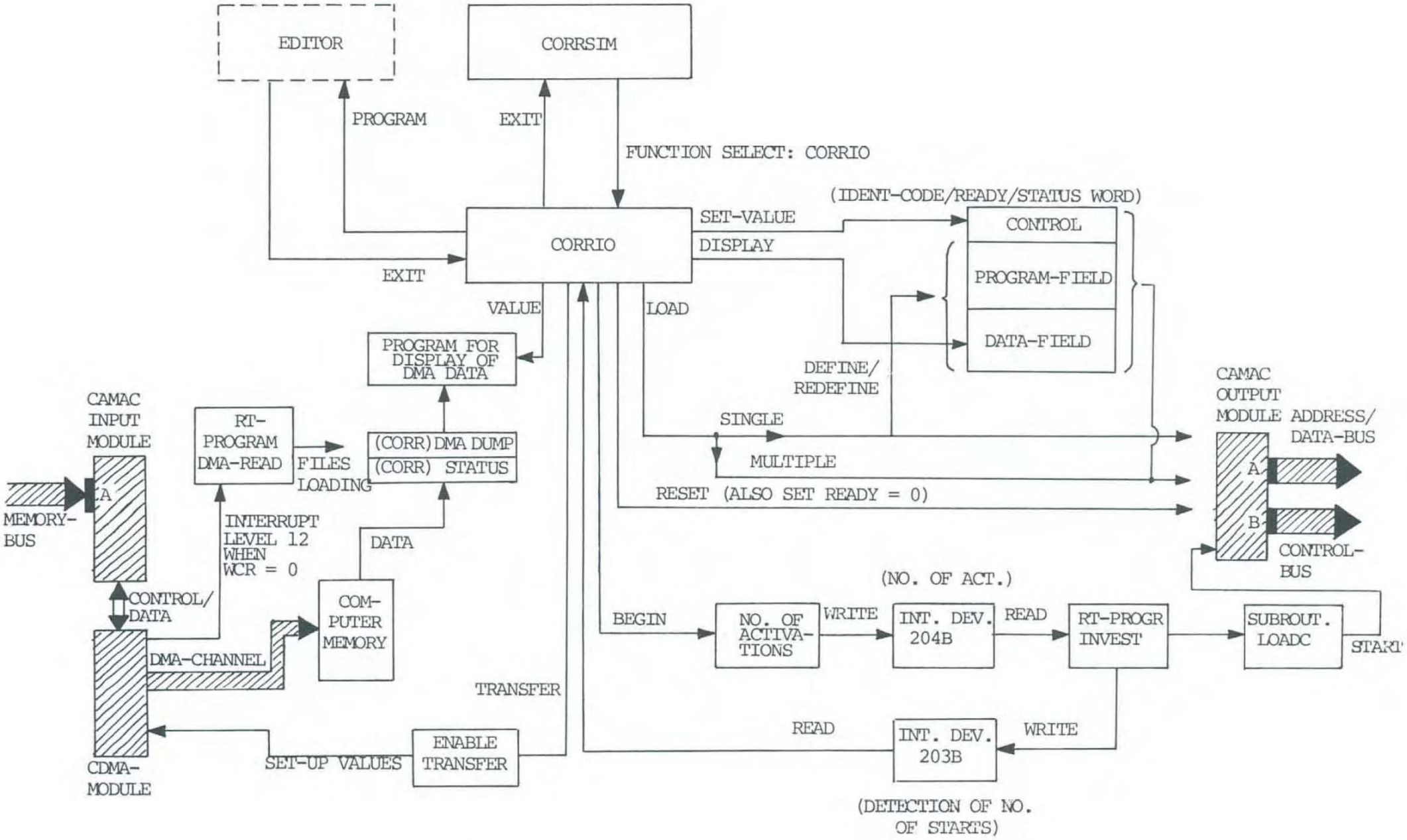



Figure 5. FUNCTION PROGRAM CORRIO

## 6.1. SET-VALUE COMMANDS

When SET-VALUE is typed (S is enough), the response is

IDENT ADDR., READY OR STATUSWORD:

When the answer is IDENT ADDR. a correlator ident-code can be entered. This parameter must be defined before any load can be made to the correlator. Once an ident-code is entered this value is active until termination of the CORRSIM or a new value is entered. When a load-command is given from the user, a transfer between the CORRIO and the addressed correlator is started. Note that the ident-code displayed on the front-panel is decimal, the parameter value entered to CORRSIM is octal.
When the answer to the response of the SET-VALUE is STATUSWORD, the statusword-register in the correlator can be loaded. If the statusword is already defined in the EDITOR program, the register in the datafield will be redefined. Note that also a transfer to the correlator is made so the correlator must be ready for loading. This implies that the correlator must not be running and control mode at the correlator front-panel must be in REMOTE. If the correlator is running this information is displayed on the terminal. If the correlator is in MANUEL control mode, the error-message:

ERROR IN COMMUNICATION WITH THE CORRELATOR

is displayed. Note that the status-word must be defined before a start of the correlator is made.

When the answer to the SET-VALUE response is 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. The register will be set until a RESET-command is generated.

## 6.2. RESET-COMMAND

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

## 6.3. DISPLAY-COMMAND

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

## 6.4. PROGRAM-COMMAND

When the PROGRAM-command is given, a direct call to the EDITOR is made, and a later EXIT from the EDITOR will give control back to the CORRIO processor. This is a possible way for quickly changes in the program-field.

## 6.5. LOAD-COMMANDS

When LOAD-command is given the response is:

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

The answer MULTIPLE gives direct load of all previous defined parameters after check on definition of ident-code and inactive state of the correlator. During loading the communication with the correlator is checked, and if no errors occur the loading is terminated by:

```
MULTIPLE LOAD ENDED, nn 16-BITS VALUES TRANSFERRED
```

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

Single load is possible by giving SINGLE. When this command is used,

the value of the register must be given and the corresponding location in program- or data-field will be either defined or re-defined during transfer to the correlator.

## 6.6. BEGIN-COMMAND

Starting a user-defined program in the correlator is activated by giving BEGIN. Note that the right conditions must be set in advance, these include:

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

The response from CORRIO, when "begin" is given, is:

GIVE NUMBER OF ACTIVATIONS (OCTALLY):

and the correlator will start (and restart) the number of activations given. After the number of starts is entered, an RT-program is started for the real-time communication with the correlator; see fig. 5.

Real-time communication between "Background"-program CORRIO and RT-program INVEST is as follows:

Number of wanted activations is written into interval device 204B. INVEST is always awaiting a number to be written into this device. When this happens, INVEST starts and calls a subroutine LOADC appropriately many times.

When INVEST is done, it writes the number of successfully activations into the interval device 203B.

CORRIO is, during the execution-time of INVEST in a waiting state awaiting INVEST to supply a number into 203B.

The RT-program INVEST should always be started prior to any BEGIN

command; else some initial starting problems could occur. (It is however possible to start INVEST after a BEGIN command provided that only one request has been placed in 204B. IF more requests have been placed there 204B will be cleared and CORRIO will be waiting forever).

LOADC is a subroutine-driver routine for the correlator written in MAC.

LOADING OF INVEST:

```
@ RT-LOADER
* NEW-SEGM
  SEG. NO:  <Seg. no.>
  RING   :  2
  :
* NREE    INVEST,
* NREE    LOADC,
* END-LO
* EX-LO
@ RT      INVEST
@ LOG
  :
```

During execution, the programs check for endless looping in the correlator (programs with no end), and if this situation happens, the message from CORRSIM will be:

```
ERROR:  ENDLESS LOOP IN CORRELATOR
nn ACTIVATION(S) WERE ASKED FOR,
kk ACTIVATION(S) WERE COMPLETED.
```

where nn is the number entered, and kk the number of successfully starts. If no error were detected, CORRSIM displays:

```
nn ACTIVATION(S) (ALL) WERE COMPLETED.
```

## 6.7 DMA-COMMANDS

Two commands are available:

TRANSFER: which is a initiatization routine for enabling DMA transfer.

VALUE : which is a routine for displaying transferred data on the terminal.

The real-time control of the DMA-communication is from the internal program in the correlator. With reference to Figure 5 a DMA-transfer is done in the following way:

After the actual micro-program is entered into the correlator, the DMA-transfer is enabled by the command TRANSFER. The response from the program is:

```
ARE THE COST/STATUS REGISTERS BEING TRANSFERRED (YES OR NO):
```

where the COST/STATUS-registers are the control registers in the correlator. Then the type of transfer-data is asked for:

```
MEMORY WORDS OR TESTWORDS TO BE TRANSFERRED
WRITE - MEMORY- FOR MEMORY WORDS (64-BIT WORDS)
WRITE - TEST - FOR TEST WORDS (16-BIT WORDS)
```

If the answer is MEMORY; the program generates

```
SPECIFY NUMBER OF 64-BITS WORDS TO BE TRANSFERRED (OCTAL)
MUST NOT BE GREATER THAN 1777 (OCTAL)
```

and the actual number can be entered.

If the answer is TEST, the program is displaying:

```
NO. OF 16-BIT TESTWORDS TO BE TRANSFERRED (OCTAL),
MUST BE LESS THAN 1777 (OCTAL),
NO. OF WORDS (-1 ⇒ RETURN):
```

The CDMA-module in the CAMMAC create will now be enabled, and the

word-count-register, WCR, will be loaded with the resulting number of 16-bit words. The start phase is ended with

READY FOR DMA-TRANSFER

and control is given back to CORRIO. After the correlator has been started with the BEGIN signal, data is automatically loaded into the computer memory through memory-bus from the correlator, CAMMAC input module and CDMA-channel transfer. When the WCR equals zero, the CDMA-module generates an interrupt on level 12 and the internal action in the computer is that RT-program DMA READ is activated. Note that before any DMA-operation is made, the RT-program must be started under user-name RT, with the command:

@ RT DMA READ

This is a real-time program for loading the DMA-data on the file DMA DUMP (under user CORR) and loads DMA-control words onto the file STATUS (user CORR). Both files has READ/WRITE access and can be used by the present user of CORRSIM.

After the DMA-transfer is finished, data can be displayed by the VALUE. If the transfer is not completed the program displays:

**** TRANSFER NOT COMPLETED ****

and the program tests this situation for 10 sec, and then if not completed, generates:

ERROR IN DMA-TRANSFER

If the transfer were successful, the program generates

MEMORY ADDRESS REGISTER: <VAL.>
WORD-COUNT-REGISTER OF CDMA: <VAL.>
COST REGISTER OF CDMA: <VAL.>

where the registers are located in the CDMA-module. (For detailed information about register contents, see instr. manual for CDMA).

If memory-transfer was selected during DMA setup, the program displays:

NO. OF 64-BIT WORDS TRANSFERRED: nn

If test-transfer was selected:

NO. OF 16-BIT TESTWORDS TRANSFERRED (STATUS/COST REGISTERS INCLUDED IF ANY): nn

WAS THE COST/STATUS REGISTERS TRANSFERRED BEFORE OF AFTER THE DATA WORDS:

The last lines is a question for identifying the COST/STATUS REGISTERS if this option was selected in advance.

The program then follows with:

PRINTOUT TO LINE-PRINTER (YES OR NO):

and the program has reached the state of possible display of data on the terminal:

TYPE <CONT> TO DISPLAY CONTROL - AND STATUS-REGISTER
<DATA> TO DISPLAY WHOLE OR PARTS OF MEMORY-AREA
<PC> TO DISPLAY PROG.-COUNTER IN CASE OF TESTWORDS
<EXIT> TO EXIT FROM THIS SUBROUTINE

Several stop instructions (terminates CORRSIM) have been placed in the DMA-routines, and if termination has occurred, the program will generate a STOP-code at the terminal:

STOP 1: UNSUCCESSFULLY OPENING OF THE FILE (CORR)DMA DUMP: DATA

STOP 2: UNSUCCESSFULLY SETTING OF BLOCK-SIZE OF THE FILE (CORR)DMA DUMP: DATA

STOP 3: ERROR IN WRITING TO (CORR)DMA DUMP: DATA

STOP 4: HAVE TRIED 100 TIMES TO OPEN THE FILE (CORR)STATUS: DATA WITHOUT SUCCESS (I.E. FILE ALREADY OPENED BY ANOTHER USER

STOP 5: UNSUCCESSFULLY OPENING OF THE FILE (CORR)STATUS: DATA

STOP 6: ERROR IN WRITING TO STATUS: DATA

STOP 7: UNSUCCESSFULLY CLOSING OF THE FILE (CORR)STATUS: DATA

Information concerning the displaying of data on a terminal and the printing of data on the line-printer:

IF the option of x-ferring the STATUS/CONTROL registers is chosen, CORRIO takes for granted that STATUS/CONTROL registers are always transferred either before or after the "main-data" in the given order.

Thus a transferred block of data is always considered like one of the following:

1) {<DATAWORDS>}

2) {<STATUS-REG>, <CONTROL-REG>,<DATAWORDS>}

3) {<DATA-WORDS>,<STATUS-REG>, <CONTROL-REG>}

4) {<STATUS-REG>, <CONTROL-REG>}

<DATAWORDS> = <TESTWORDS> OR <MEMORY WORDS>

However these constraints can easily be completely avoided like this: Specify to CORRIO that you only want datawords transferred (case 1 above). When you specify number of words, you include register transfers, too. Thereby you can have any mixture of data and registers - you only have to remember which were which.

## 7. SIMULATION OF CORRELATOR ARITHMETIC, USE OF FUNCTION ARITEST

The program simulates the arithmetic hardware section of the correlator for fixed programs. Input data used during simulation is the test-sequence which can, by statusword selection, be used for loading the correlator. The output from the simulation is the

content (octal numbers) of the correlator result-memory.

At present only 5 programs have been entered into the system, and can be loaded by use of the READ-command in the editor:

FILE 2: PROGRAM FOR TRANSFERRING DATA THROUGH THE CORRELATOR WITHOUT CALCULATIONS

FILE 3: PROGRAM FOR AUTOCORRELATION

FILE 4: PROGRAM FOR "POWER-PROFILE" MEASUREMENTS

FILE 5: PROGRAM FOR HANDLING DMA TRANSFER FROM CORRELATOR MEMORY

These programs are only preliminary versions of the fixed and are used for hardware check-out of the correlator prototype. The programs will not be described here, because a separate report on the fixed programs will be made in the future.

## 8. INTERFACING USER-DEFINED PROGRAMS WITH CORRSIM

The absolute version of the CORRSIM program is so large that it completely fills the available user area (64 K memory size) when activated. In order to make possible interface with other user-defined programs, the CORRSIM program is segmented so only parts of it are loaded during execution. This creates free memory-locations for other programs. The exact used area for CORRSIM can be found in the program documentation report.

## 9. REFERENCES

[1]: Hans-Jørgen Alker: "INSTRUCTION MANUAL FOR EISCAT DIGITAL CORRELATOR".
Report from the Auroral Observatory, Tromsø, Norway (Sept. 1978)

[2]: Hans-Jørgen Alker: "A PROGRAMMABLE CORRELATOR MODULE FOR THE EISCAT RADAR SYSTEM".
Report no. 51-78 from the Auroral Observatory, Tromsø, Norway (Febr. 1978)

[3]: Hans-Jørgen Alker: "PROGRAM CORRSIM: PROGRAM DOCUMENTATION".
Report from the Auroral Observatory (Sept. 1978).

**EISCAT SCIENTIFIC ASSOCIATION**
S-98101 KIRUNA 1. SWEDEN
TELEPHONE 0980/187 40
TELEX 8754 GEOFYSK S