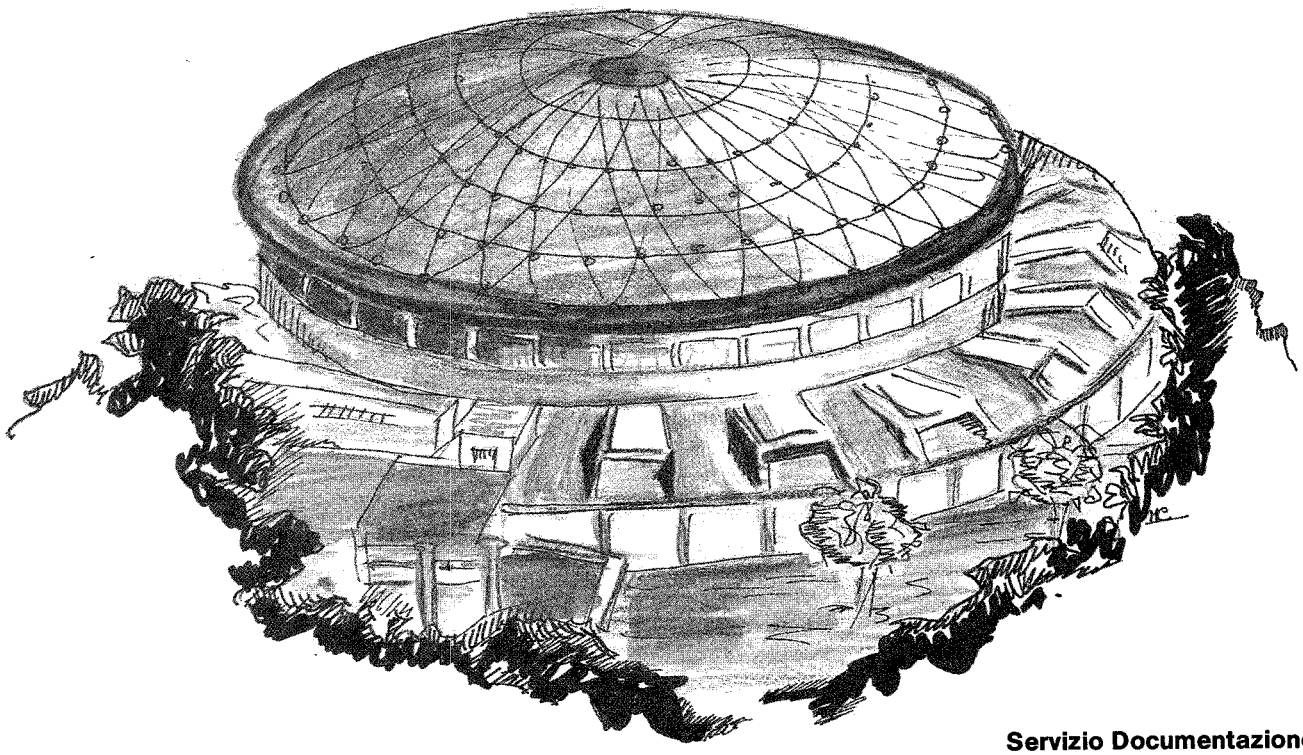
Laboratori Nazionali di Frascati

LNF-90/030(NT)
27 Aprile 1990

C. Milardi:

CONSOLE SOFTWARE DESIGN FOR THE LISA ACCELERATOR CONTROL SYSTEM



Servizio Documentazione
dei Laboratori Nazionali di Frascati
P.O. Box, 13 - 00044 Frascati (Italy)

INFN - Laboratori Nazionali di Frascati
Servizio Documentazione

LNF-90/030(NT)
27 Aprile 1990

CONSOLE SOFTWARE DESIGN FOR THE LISA ACCELERATOR CONTROL SYSTEM

C. Milardi
INFN - Laboratori Nazionali di Frascati, I-00044 Frascati

ABSTRACT

Particle accelerator dimensions and complexity tend constantly to increase. This has required suitable control systems, responding to the machine purposes and satisfying the request of being user friendly toward the operators.

At the same time the physics experiments sophistication has caused big improvements in the data acquisition systems.

Progresses in microprocessor technology, in fast local area networks and in workstation human interfaces, have been extremely helpful for progress in this fields.

LISA is a Linear Superconducting Accelerator under construction at the National Laboratories of the INFN in Frascati. Its control system has been planned keeping in mind all the developments from the aforementioned new trends.

Therefore LISA has a multiprocessor control system, based on the VME standard. It is organized on three main blocks performing three different operations: interface with machine devices, interface with the operators, supervision and connections between the previous two.

In this paper particular attention is reserved to the operator interface, all its hardware and software details are specified and the working philosophy is presented.

1.1 - CONTROL SYSTEMS OVERVIEW

Requirements in controls, as well as in data acquisition systems, have led large groups of people to begin with a systematic planning[1] and prototype developments. This fact has forced people to focus on the basic ideas[2][3], in order to change the old realization techniques. At the present, efforts are aimed at standard solutions, with a high degree of portability.

At the same time, big technology improvements in the hardware field, making available high performance and low price microprocessors[4] , have permitted a fruitful transition from centralized to distributed control systems.

As a direct consequence of that, one of the main features of a new generation control system is to be made up of different blocks, performing different tasks related in a hierarchical way. Usually these blocks are called levels.

The whole problem of controlling a machine can thus be split in many simpler parts, each of them controlling at least a single device. Transmission of information among different sections of the control system can be easily assured by high speed buses like VME, MacVEE or VMV, or by a variety of standard networks, available with different characteristics: from deterministic token bus and token ring, to non-deterministic, but reliable and commercially well supported Ethernet, or a combination of them.

This decentralization of intelligence has been for many years a basic characteristic of the control system projects. It is shared from machine started more then ten years ago, as the AGS machine[9] and the control system for the Stanford Photon Research Laboratory[10] ; up to recent projects as those for the machines LEP, ESRF[6] , ELETTRA[7] and the upgraded version of SLC[8] .

Obviously a multiprocessor environment is well suitable also for data acquisition and controls of large high energy experiments. This is confirmed by the UA1[11] experiment at CERN, or H1[12] on the HERA machine at Desy.

All this examples need short reaction times. Fast response is used, in the data acquisition systems, to collect a large amount of data and store them, after some filtering analysis based on a suitable set of criteria. On the other hand, in accelerator control systems, a high speed response is useful for a prompt feedback, in performing automatic operations and eventually in pointing out some anomaly in machine running. All the mentioned examples are built with a modular structure, very useful in developing control as well as data acquisition systems and mandatory for their maintenance and upgrading.

However, improvements in the hardware field are meaningful only if followed by a fundamental change in the operator-control system interaction philosophy. So the progress in workstation capabilities has started a new way of accessing the accelerators, avoiding the use of the old touch-panels and the split between devices for graphics display and devices for information input-output [5] . This fact has led to the development of sophisticated, but easy to use, operator interfaces[13] .

The importance of the operator interface on the whole control system performances can be emphasized by some examples. Real time characteristics could be impaired by an interaction protocol that doesn't provide either a speedy visualization of the machine status, or permit to perform operations in a short time. Modularity could be made useless by a monolithic control system software.

So the operator interface has to be constituted by building blocks performing the basic operations on the machine; it must be capable of providing the user whatever combination of these basic operations. Moreover it must give an immediate understanding of the action it refers to, using symbolic tools. This subject will be treated extensively later in this paper. The new generation of workstations available today, with their powerful processors and high resolution monitors, are what is necessary for this purposes. They make possible the use of windows displaying information graphically instead of using character strings. At the same time new pointing devices like the mouse, permit a direct, fast and easy manipulation of operations, reducing the possibility of mistakes too.

The big progress represented by the use of windows is soon clear if we think how large is the amount of information that can be presented at a glance. This can be also hidden in menus, if not immediately necessary. Many windows can be viewed together, so that the operator can set an element of the machine and look for the effects on many other machine parts. It is quite obvious that this means as well, to avoid the bottleneck usually inherent to the displaying of input and output data.

2.1 - A LISA CONTROL SYSTEM OVERVIEW

The project for the Lisa[14] control system has been planned keeping in mind all of these features, and exploiting the big advantage that this machine is new. So there's no link to preexistent hardware, or software, but the most advanced techniques that are nowadays available are employed.

The final design is a three level multiprocessor control system, based on the VME standard[15] [16] .

The first level enables the operator to monitor all the equipment from the control room; it consists of three Macintosh IIx consoles connected to the VME crate of the second level.

The second level is constituted by a CPU, on a VME crate, that reads commands and data coming from the consoles, sends them to the third level and stores all information about the machine element status in a RAM on its same crate. Each console can read this central memory independently looking at it as an extension of its own.

The third level is constituted by several VME crates, each containing at least one CPU. This CPU executes the commands it receives from the second level, acting on the second level elements through the necessary hardware interfaces. The last ones are also used to monitor every machine status change, that eventually is transmitted to the second level.

The choice of the Macintosh as console was determined by the fact that it was the first computer realized thinking that the user has to be in control. This means that the Macintosh is able to accept instructions about the next action to perform, in whatever unpredictable order. In

practice, this is made possible giving to the user a hand held pointing device: the mouse, a high resolution bit mapped display and a large software library. The mouse is fundamental because it allows to interact with the computer in an easy, speedy and direct way, previously limited to the keyboard use. The high resolution bit mapped display, with its full RGB colour system, permits to realize a sophisticate graphic display on a little screen. The library [17] is essential for developing graphical objects, for processing and responding to the inputs via the mouse and the keyboard. It is stored in an internal ROM and it is written in assembly language, although it is also accessible by high level languages, such as Pascal, C, Fortran. The choice of the Macintosh was also supported by the coincidence between the processors on the computer and the processors used by the chosen VME CPU, the MVME 133 A. All these microprocessors belong to the Motorola 68K family. This coincidence assures consistency in data organization and a straightforward interaction between the first and the second level.

Both the part of the control program running on the Macintosh consoles, and the one running on the VME are being written using a development environment designed at CERN, called MacSyS[18] . It provides access to the VME system using a RTF Fortran compiler[19] , a M68K Assembler and a Linker[20] . The last one allows to link programs on a Macintosh computer, but also to load and run them on a remote CPU[21] connected to the Macintosh.

RTF is a real time Fortran, it has ideal characteristics for programming in an interrupt driven control environment, including pointer based variables, allocation of memory at run time and extended equivalence instructions. Using RTF it is also possible to call the Macintosh library routines by simple Fortran statements. Moreover RTF permits: direct access to the hardware registers, to absolute memory addresses, generation of position independent code, flexibility of embedding assembly instructions. All these performances are necessary in a VME developing environment.

The decision of writing the control program in Fortran is related to the fact that Fortran is the most popular programming language among physicists, so we hope to provide an understandable control code for the people working on specific machine devices.

2.2 - INTERACTION PRINCIPLES BETWEEN THE FIRST LEVEL AND THE SECOND LEVEL

Each console can send commands to the machine, as well as read information about the machine status, by direct access to the second level RAM[22] [23] . Direct access means that each console looks at the second level memory like an extension of its own; so in order to read an information, it is only necessary to know the address where the information was stored. This operation is made easy by a systematic organization of the data in the RAM locations[24] . Different care is necessary, if the access to the second level produces some element setting. In fact the following criteria have been chosen: each console is able to access all element

information at the same time, but only one is allowed to set the parameters of one of them. So it is necessary to built something like a queue to rule the action of the three consoles on each machine device. This queue has been realized building on the second level three mailboxes connecting the first and the second level. Each console that wants to send a command downstairs, checks that no other console is operating on the same device; if this condition is true, it writes the message and signals that there's a command. The second level CPU checks cyclically the mailboxes, looking for commands, eventually reads it, resets the initial mailbox status and finally runs procedures to send the command to the third level. This strategy ensures that only one console operates on any element at each time and that a command is read only after it has been completely written.

The use of a high speed bus, together with the limited machine dimensions, avoids the use of a network; we only use an Apple-Talk, for connecting the consoles to same facilities, as printing or data storage devices.

2.3 - INTERACTION BETWEEN THE FIRST LEVEL AND THE OPERATOR

Interaction between the first level and the operator is the most critical aspect of the first level structure, in fact it connects an analogic multitasking 'device': the operator and a digital device: the Macintosh. The operator that seats at the console, would like to interact without limitation with the machine. This means that he wants the possibility of changing all machine parameters, to check their influences on each other, to display the physical variables graphically, to save particular machine status, and to restore it if necessary. All this would have to be done by few simple handles, able to access whatever machine device at whatever information degree. Important is that all this handles share the same general aspects, so that it would be possible to recognize at a glance the kind of operation the handle refers to. A different information degree is essential whether to the operators that have to manage the machine at run time, or to the specialist in checking and testing the single devices.

Moreover some tools have to be built for summarizing the machine general status, during specific operations.

An interface with such properties must be constituted by basic routines and must be able of reconfiguring its structure according to the operator 's requests, since it is not possible to foresee all his actions.

These are the general guidelines which a new generation operator interface has to be built on; obviously doesn't exist a suitable solution available commercially, because this general aspects have to be adapted to the machine's peculiarities. Therefore what usually happens, is that physicists realize by themselves the operator interface, building the necessary handles looking at the requests of the machine operations, as well as to the underlying philosophy of the operating system running on the console.

3. - THE BASIC TOOLS OF THE CONTROL SOFTWARE

3.1 - THE WINDOW

A window is a variable part of the screen selected for a particular purpose, concerning the user communication. All software packages running on modern workstations make use of windows, so their deeper definition depends on the computer they are developed on.

On the Macintosh computer, each window[17] represents an individual plane having its own drawing environment: the graphport. The last one contains all the information specifying where and how the graphical object has to be drawn. For example, some of the meaningful data stored in the graphport are: the pen mode, the pen pattern, the background pattern, the text font, the text face, the address of the pixel map, the background colour, the foreground colour.

On the same screen it is possible to open many graphports at the same time, with the same graphical possibilities. It is extremely easy to access a graphport because it is identified by a pointer, i.e. the address of the memory location were it is stored. The user can work only in one graphport at each time, that corresponding to the frontmost window, called active window. It is also possible to bring a window in front of another one and to move it around on the screen.

The Macintosh toolbox library provides six different kinds of windows and the user can define many others by himself, but all the corresponding graphports have the same structure.

Each different window part, as well as everything inside it, can be shown in whatever of the 256 colours available on the monitor, this is due to the correspondence between a point in the graphport and the triplet of pixels on the RGB screen's matrix.

In the control system software simple windows are reserved for a few purposes. All alarm messages about devices, which the operator is not acting on, can be read in a window that is opened by the operator as a consequence of an alarm signal: this is usually an icon on the screen that becomes highlighted. Furthermore a general view of the machine status with all the data updated will be displayed on a window. Finally also the graphs of the relevant physical variables will be shown in windows. It is so clear that the simple windows are reserved for the flux of data from the control system toward the operator who cannot do anything inside them.

3.2 - DIALOG BOX

A dialog box[17] is a window of fixed dimensions that encloses some particular objects as knobs and strings of alpha-numeric characters; so that it looks like the frontal panel of an instrument.

There are two kind of standard Macintosh dialog boxes: modal and modeless. The first kind restricts the user 's actions to the same dialog box until it doesn't disappear. On the contrary, a modeless dialog box imposes no limitations, this means that many of them can be opened at the same time and the operator can switch from one to another.

The objects enclosed in a dialog box are called items, there are the following standard Macintosh items :

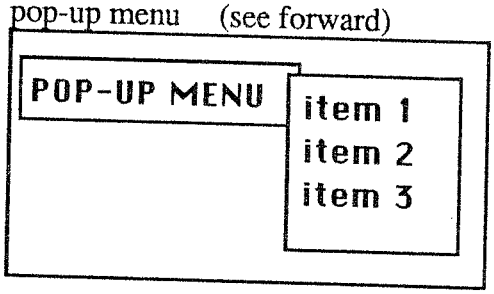
user items (rectangle of whatever dimensions where the operator can do everything he likes: write, draw and so on)

editable text (rectangle of variable dimensions where the operator can write directly a character string)

static text (fixed string of characters)

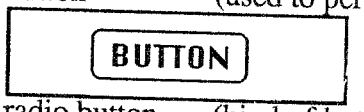
icon

pop-up menu (see forward)

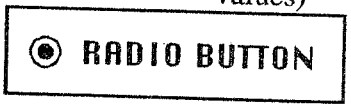


and the following standard Macintosh control items:

button (used to perform some action by clicking with the mouse on it)



radio button (kind of button reserved for variables that can switch between two or more values)



scroll bar (control that permits to select a value among a large set)

In the LISA operator interface the dialog boxes are used to get in input all the data about a specific operation and for conveying data from the machine device to the operator. Therefore a dialog box is an object where either the user, or the control system can operate.

3.3 - MENU

A menu [17] is a group of options, summarized by a title that is displayed in a rectangle at the top of the screen. A menu is pulled down showing options only under operator request.

In the menu bar of the LISA control system, will be enclosed general operations like the opening of the dialog boxes inherent to the section of machine which the operator is working on, the transfer to another machine section, the display of graphs and finally the exit from the control program. One menu will be reserved to the titles list of all windows and dialog boxes opened on the screen. The title list is arranged in the windows order of appearance on the screen and is updated according to any change, in the window's order itself.

3.4 - THE POP-UP MENU

A pop-up menu[17] is a menu that can appear everywhere on the screen. It is used only inside the dialog boxes, for setting some variables that are changed seldom as the units in displaying data or the response sensitivity.

3.5 - CONTROL PROGRAM STRATEGY

Although the general aspects of the tools are important, the ability of a control program is not in opening windows or dialog boxes, but in arranging them in a proper way. At the moment a first version of the control software, phase '0', inspired to simple logical design criteria, is being built.

All windows are titled with the name of the element or the operation they refer to, so that their field of action is immediately clear. Windows and dialog boxes that refer to elements of the same kind have the same size, colours, and contain the same controls arranged in the same way. These features permit to limit the number of windows that have to be built, help the user to read them and what is most important allow the management of all windows of a particular set by the same Fortran subroutines.

Each dialog box, whatever its task, has some items of general use: a button "CANCEL", a button "UNDO", a button "ABORT", a button "SET", a user item titled "MESSAGES", and a radio button switching between "READ/WRITE STATUS".

The button "CANCEL" deletes the dialog box from the screen, the user item "MESSAGES" shows the information of interest for the operator including faults and warnings.

When a dialog box is opened it is always in the read status, it can only read and consequently show the machine data. The values presented in the dialog box regard the physical variables it deals with, currently updated. When the operator wishes to change these variables, he switches the dialog box status from read to write; this operation will be successful only if no other console is operating on the same element, otherwise the console number and the dialog box name, using that element, are written in the user item "MESSAGES". The "SET" button allows, if the dialog box is in the write status, to send the new operator's value to the machine element. The button " UNDO" deletes the last action performed by the operator, unless it is a click on the "SET" button; while the "ABORT" button interrupts whatever command is in execution and waits for the operator decision.

Every dialog box include also two user items and one control for each machine device it can operate upon. The first user item shows the current element value, while on the second one, the wished value can be set up by the keyboard or moving the control by the mouse. So the operator can modify a device value using only the mouse, he can always change the selected value before a "SET" and, what is more important, he can check the result of its action on the current status user item.

In building the tools necessary for the control program, large use was done of resources. In Macintosh philosophy a resource is a part of standalone code containing some specific information. It is completely independent from the code that includes it, and can be separately edited. In a resource not only the data but also the application's code itself can be stored. Resources are identified by types; there are some standard Toolbox resource types[17] , but the user can define his own. In the phase '0' of the control software a large use of the standard resources has been done, to built all the previously introduced tools. All the windows are made using "WIND" and "wctb" resources; the first contains all the information about dimensions, title and shape of the window, while the second says in which colour the frame, the content region, the title, if there is one, and the title highlighted have to appear. The dialog boxes are built using the "DLOG", "DITL", "dctb" and "ictb" resources. The "DLOG" provides all the information about the window contained in the dialog box, it is analogue to the "WIND" resource, as well as the "dctb" contains the list of the colours of the window like the "wctb". The "DITL" lists all the items inclosed in the dialog box, their position, dimensions, kind and the associated text. The "ictb" allows to specify the colours of the body, of the frame, of the associated text, of the items itself. Moreover it permits to specify the font and the text size for the text item. The menu are drawn using the "MENU" and the "mctb" resources that specify in the order the menu items with their eventual equivalent keys, and the colours of the menu different parts.

As an example of the utility of the resource method, all the dialog boxes necessary for the control system program are opened by the same subroutine using different templates contained in the resources. There is no need for the programmer to provide, embedded in his code, details like dimensions, title, colour and so on. All descriptive information can be stored in a resource; this permits to gain a lot of time and to avoid a lot of mistakes. Moreover any editing and/or compiling of large code, do not involve any changes in resources; the last one is surely the main features of the resource philosophy. Also an eventual translation of the control code to another language will be absolutely simple, because resources must not be modified.

4. - USER INTERFACE PROGRAM STRUCTURE

The central body of the user interface is a loop on the events. In the Macintosh philosophy an event is every action the operator does, like to press, to release, to drag the mouse, or to press a keyboard key. There are also events that are consequent to another one. For example an operator click on the "CANCEL" button causes a dialog box to disappear and makes it necessary to draw again whatever was displayed behind the dialog box; in Macintosh terms an update event is generated by a mouse down event [17] .

In the LISA control system philosophy, every change in the second level RAM data, regarding the windows opened on the screen, is an event too and it is necessary continually to check and detect it.

In fig.(1) a flowchart of the central control loop is shown.

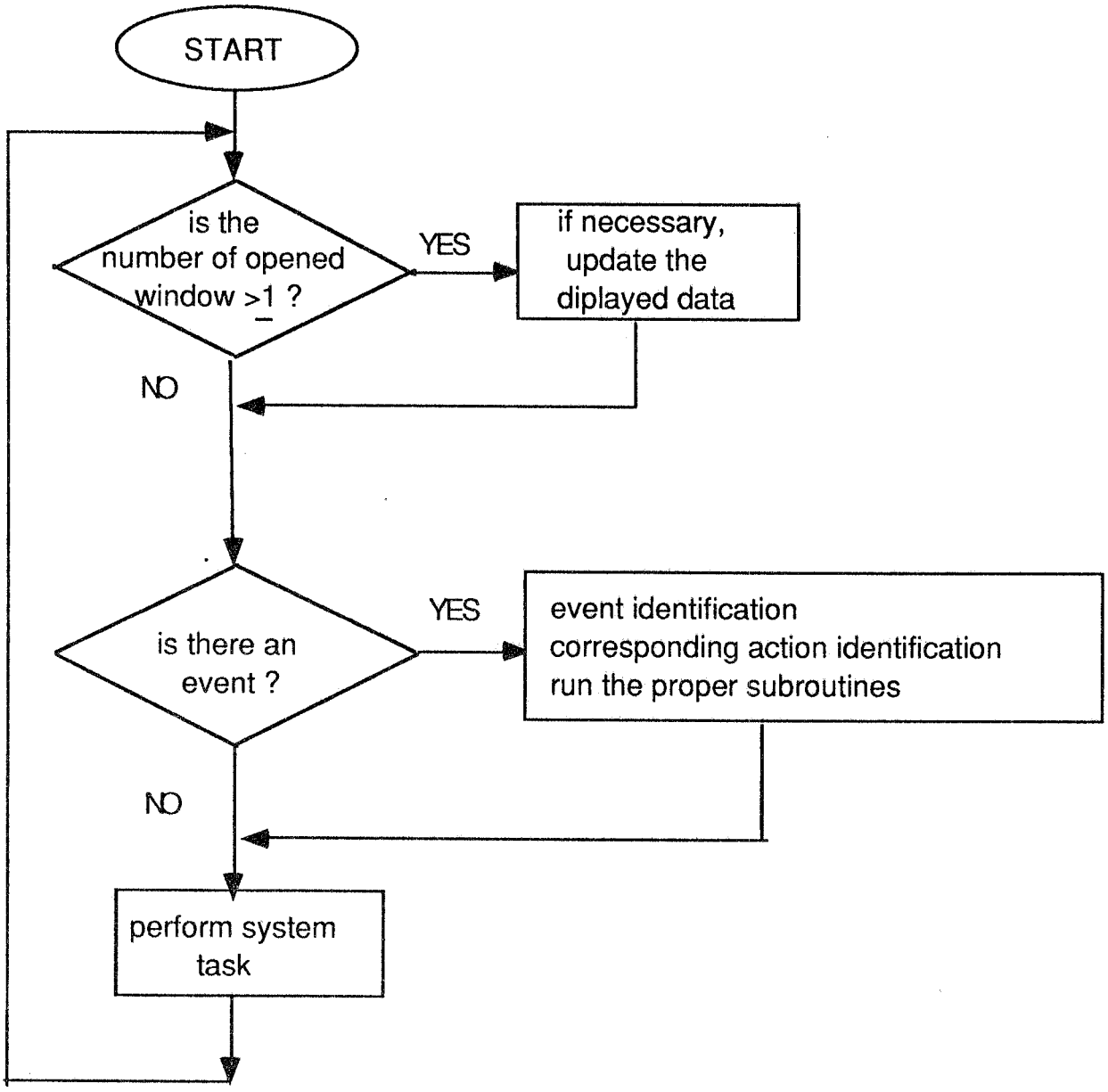


FIG. 1

The event identification is a laborious task. It is necessary to say that according to the underlying interface philosophy, the only relevant event is a mouse down, no interest there is for key down event, unless it involves a keystroke equivalent to some menu item. The computer operating system stores all the Macintosh events in a buffer called, in Macintosh terms, event queue; the user interface scans this queue until it finds a mouse down event. The program distinguishes if a mouse down is in a dialog box or not; if not it checks the other two possibilities: a mouse down in the menu bar or inside a simple window. After a mouse down in

the menu bar the program asks for the numbers identifying the menu and the item selected, then the consequent actions are performed; the same process is generated for a keystroke event, if the menu item has some equivalent key. More attention is required for a mouse down in a window: it can be in the content region of the frontmost window, in which case no action is performed, in the content region of a not frontmost window, in which case the window has to be redrawn in the first plane (selected), or in the window title bar. This requires the control program to be able to move the window around on the screen after have selected it, if necessary.

A mouse down in a dialog box, if it is not the frontmost window, is recorded like an event in a simple window and can cause only the window to be selected. On the other side when the dialog box is on the first plane, the user interface checks if the mouse down was in a control and after an affirmative answer in what control. Each control, when the dialog box is opened, is identified by a pointer so it is particularly easy to select the subsequent action to perform. At last if the mouse down event is not in a control, the program returns the item number and the corresponding window pointer to identify the consequent procedures.

5. - CURRENT STATUS

At the moment a first version of the user interface is running; it performs a phase '0', extremely simple, but including all the basic routines. In fact they are available, with complete reliability, the routines for the central loop, for creating whatever: window, dialog box, menu, pop-up menu and to manage a general scroll bar. It is so possible to read data and messages from the control system second level, to display them in the proper window, to send commands down to the third level, to respond to menu and pop-up menu selection and to manage the appearance of, at least, twenty windows contemporarily on the screen.

The actual first implementation of the LISA user interface program starts showing a picture of the machine. Clicking on a particular machine section it is possible to visualize that section in detail and using the menu items, it is possible to open dialog boxes to perform operations on devices in that section or to open windows for graphs. Another menu allows to return to the machine general view in order togive attention on another section, or to exit from the program.

In fig.(2) there is a picture of what appears on the screen, when the operator is setting the solenoid elements in the LISA injection section.

This first user interface prototype is being tested realizing the upgraded control system version, for the LINAC steering magnets on the Adone machine. This is a project that has very small dimensions and no problem respect to the LISA control system. however the work will be useful to test the user interface with the operators at run time, so to check the criteria's validity and to understand developing guidelines.

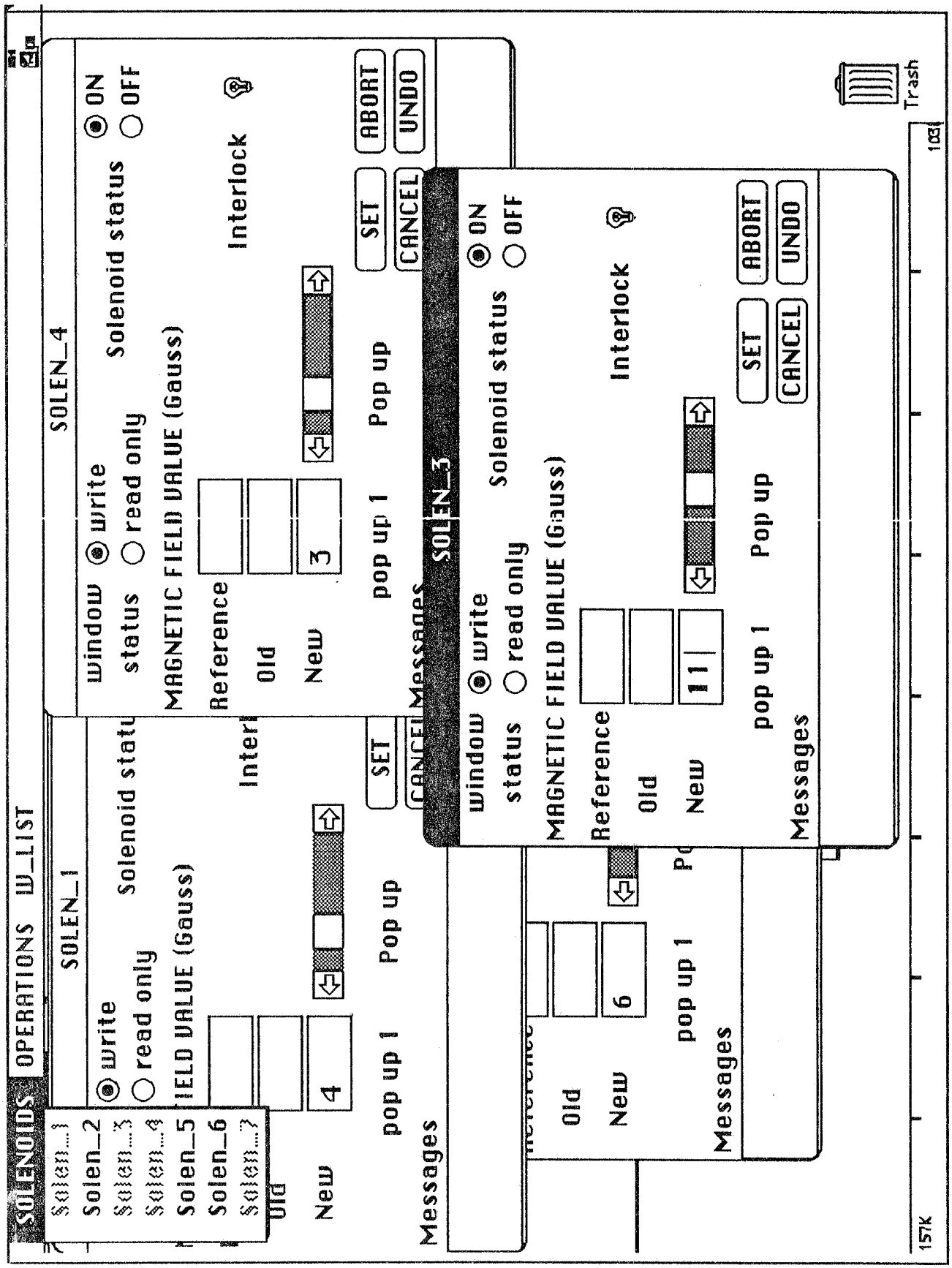

SOLENOIDS
OPERATIONS
W_LIST
SOLEN_1
SOLEN_4
SOLEN_3
Solen_2
Solen_5
Solen_6
window
status
write
read only
Solenoid status
ON
OFF
MAGNETIC FIELD VALUE (Gauss)
Reference
Old
New
Interlock
pop up 1
Pop up
SET
CANCEL
ABORT
UNDO
Messages
Trash
157K

FIG. 2

6. - FUTURE PERSPECTIVE

Further developments of the phase '0', are toward the inclusion in the control program of graphical utilities, this is in progress looking to many standard packages developed in the last years for the Macintosh computer. A next version will provide also dialog boxes for acting not on a single device, but on many devices involved in the realization of a complex operation as the beam focusing.

Constant attention is reserved to the developments in the field of the application running on Macintosh looking for ever more powerful tools and more versatile development environment.

Thanks are due to Michele Castellano, Nicola Cavallo, Giampiero Di Pirro and Luciano Trasatti for many interesting discussions.

REFERENCES

[1] Stuart C. Schaller and David E. Schultz: 'On Designing a Control System for a New Generation of Accelerator'; Presented at the 1987 Europhysics conference on control systems for experimental physics. Villars sur Ollon (VD), Switzerland, Switzerland, Sept 28-Oct 2 , 1987.

[2] M.Lee, S.Clearwater, E.Theil, V.Paxon, 'Modern Approches to Accelerator Simulation and On Line Control' presented to the Particle Accelerator Conference Washington, March 1987

[3] J.Altaber, F.Beck, M.C.Crowley-Milling, R.Raush: 'Suggested principles for the Control of Future Accelerator', IEEE Transaction on Nuclear Science, Vol. NS-26, No. 3, June 1979.

[4] E.Theil, V.Jacobson, V.Paxon, 'The impact of new Computer tecnology on accelerator control ' presented to the IEEE Conference, 1987

[5] V.Paxon, V.Jacobson, E.Theil, M.Lee, S.Clearwater, ' Workstation Operator Interface for Accellerator Control', presented to the IEEE Conference, 1987

[6] W.D.Klotz: 'Present Status of the ESRF Control System', ESRF-MAC-05-07 (1987).

[7] D.Bulfone, M.Migliacco: 'the Conceptual Design of the ELETTRA Control System', SINCROTRONE TRIESTE Area di Ricerca Padriciano 99, february 1989.

[8] J.Bogart, A.Hunter, M.Sullenberger, S.Kleban, N.Phinney, N.Spencer: 'Workstation Console for SLC', SLC-PUB-4212, February 1987 (A).

[9] A.Abola, R.Casella, T.Clifford, L.Hoff, R.Katz, S.Kennel, S.Mandell, E.Mac Breen, D.P.Weygand: ' Experiences in Using Workstation as Host in an Accelerator Control Enviroment', Presented to the IEEE Conference, 1987.

[10] C.Cork, J.Fox, R.Melen: ' A Real Time Control System for the Stanford Photon Research Laboratory', Presented to the IEEE Conference, 1987.

[11] S.Cittolin, M.Demoulin, W.J.Haynes, W.Yank, E.Pietarinen, P.Rossi: ' the UA1 VME Based Data Readout and Multiprocessor System', Proceedings of the Int. Conf. of Recent Developments in Computing Processors, and Software Research for High-Energy-Physics, Guanajuato Mexico, May 1984.

[12] W.J.Heynes: 'The H1 VME-Based Data Acquisition System', Paper submitted for the Preliminary Proceedings of the ESONE VMEbus in Research Conference, Zurich, Switzerland, 11-13 October, 1988.

[13] B.Shneiderman, Designing The User Interface, Addison Wesley, 1987.

[14] A.Aragona, C.Biscari, R.Boni, M.Castellano, A.Cattoni, V.Chimenti, S.De Simone, G.Di Pirro, S.Faini, U.Gambardella, A.Ghigo, S.Guiducci, S.Kulinskj, L.Maritato, G.Modestino, P.Patteri, M.Preger, C.Sanelli, M.Serio, B.Spataro, S.Tazzari, F.Tazzioli, L.Trasatti, M.Vescovi, N.Cavallo, F.Cevenini, 'The Linear Superconducting Accelerator Project Lisa ', presented to the European Particle Accelerator Conference, Roma, 1988.

[15] M.Castellano, S. De Simone, G.Di Pirro, S.Guiducci, P.Patteri, M.Serio, L.Trasatti, N.Cavallo, F.Cevenini, ' Diagnostic and Control System for the Lisa Project ', presented to the Europian Particle Accelerator Conference, Roma, 1988.

[16] M. Castellano, L. Catani, N. Cavallo, F. Cevenini, G. Di Pirro, S.Guiducci,C. Milardi, P. Patteri, M. Serio, L. Trasatti: "A Distributed VME Control System for the Lisa Superconducting Linac"; Presented to the Williamsburg RT 89.

[17] Apple: ' Inside Macintosh Volumes: I, II, III, IV, V', Addison Wesley 1988.

[18] 'MacUA1, the M68K-VME UA1 Macintosh based development system', UA1 Online M68K, CERN, 1985.

[19] H. von der Schmitt, "RTF/68K, Real-Time FORTRAN-77 for 68K Processors"; Physikalisches Institut Universitat Heidelberg and CERN UA1, March 9, 1988.

[20] S.Cittolin: ' MacUA1 MACASM1/MACLINK1' CERN UA1 technical notes 31-3-88.

[21] M. Demoulin: "CPUA1MON user's manual", CERN UA1 technical note 85/36, May 27, 1985.

[22] B.G. Taylor, "The MICRON User Manual", Rev.1.0, CERN EP Division, 1988.

[23] B.G. Taylor, "The MacVEE Hardware User Manual"; Rev.4.1, CERN EP Division, 1987.

[24] L.Catani, G.Di Pirro, F.Giacco, F.Serlini, 'Primi Sviluppi del Secondo e Terzo Livello del Sistema di Controllo di Lisa', memorandum interno, INFN LNF Frascati, Lis-47, 19/7/89.