# A CAMAC EMBEDDED REAL TIME EXECUTIVE WITH HIGH LEVEL LANGUAGE FOR USE AT GANIL

M. Ulrich, E. Lecorché, and the Ganil Control Group
J.M. Loyant (Operation Group)
Ganil B.P. 5027 - 14021 Caen Cedex

**Abstract** : Since the first ion beam of Ganil, the Real Time processes of the accelerator have been under the control of distributed Intel 8080 controlled Camac crates. Application tasks are therefore mainly written in assembly language.

These devices were recently replaced by 68000 based crate controllers named DIVA. And the opportunity to write processes tasks in LTR, the same high level language as for the rest of our control software has existed since then.

This upgrading has been completed by designing and realizing a small Real Time Executive.

This paper emphasizes the main features of the RTE : tasks, delay, resources management, etc... Functional links to the control computer and integration of the 68000 microprocessor in the Ganil Control System are discussed. An application example is then presented (beam switching control in the experimental area).

In conclusion, merits and limitations of our Real Time Executive are presented and possible improvements for the next years to come are suggested.

## 1. INTRODUCTION

GANIL is a heavy ion accelerator consisting of three cyclotrons in cascade which provides physicists with various kinds of beam, about 2000 equipments have to be driven by the control system.

The GANIL Control System widely described elsewhere (1,2) is based on a minicomputer Mitra 625 linked to the equipments through the CAMAC standard. Controlling the accelerator is also achieved by using distributed intelligence mainly consisting of microprocessors embedded in autonomous CAMAC controllers. These controllers are used either to drive the main console and four movable consoles or to perform local processes (handling RF cavities, measuring the beam phase, collecting data from wire chambers...).

Most of these microprocessors are Intel 8080 housed in JCAM10 autonomous controllers, a new generation named "DIVA" has been designed at GANIL and built around the 68000 Motorola microprocessor. Hardware and software environment of this DIVA controller are now briefly described in order to show our needs and requirements.

### 1.1. Hardware :

The DIVA autonomous controller consists of two boards (3). One card is the CPU board including up to 512 kbytes of DRAM memory ; the input/output devices are three RS232 lines on 6850 ACIAs (console, printer) and a 68230 parallel interface timer (PIT) used to handle a SCSI floppy disk interface. Time management is achieved using this PIT and a real time clock with battery back-up ; for specific needs a 6840 timer has also been integrated on the card.

The other card is mainly devoted to CAMAC addressing purposes ; CAMAC is seen as a specific map of memory among the whole memory addressing space.

"Look-at-me" (LAM) CAMAC interrupts are encoded into graded-lams so that a vector number of the 68000 is attached to a dedicated graded-lam.

The most significant interrupt levels are shown below in descending order :

- level 7 : abort signal of the 68000 and local/remote switch
- level 6 : no CAMAC response (known as "no X")
- level 5 : timer interrupts from the PIT
- level 4 : CAMAC interrupts (LAM)
- level 3 : console interrupts coming through the ACIA port
- level 2 : not specified
- level 1 : user thumbwheel switch on the front panel of the DIVA.

### 1.2. Software :

As our minicomputers are programmed in LTR which is a real time high level language, we decided to adopt this same language for our new controllers ; applications are therefore cross-compiled on the development computer.

The requirement was to allow real time applications to run on the 68000 while remaining fully in line with the philosophy of the control system.

As a first step, a small real time kernel was written to provide the basic facilities. Quite quickly, it was made clear that much more capabilities must be added for user programs.

The present executive we have designed is named "MTR-D". It meets all the major requirements of a real time monitor and still remains simple and quite small (less than 7 kbytes for the basic kernel).

## 2. THE "MTR-D" MONITOR

### 2.1. Basic items

The "MTR-D" monitor allows tasks to be runned under a real time environment. There are two kinds of tasks : "delayed" tasks which are performed at interrupt level 0 of the 68000 and are scheduled by the executive according to their priority ; "immediate" tasks which are attached to a dedicated vector number of the microprocessor.

"MTR-D" provides the following system services:

- up to 32 delayed tasks can be polled by the scheduling task
- each delayed task has its own priority ranging from 1 to 32 (static assignment)

- up to 10 delayed tasks at the same time can be waiting for a delay
- tasks can create or kill delayed tasks
- delayed tasks can be suspended waiting for ressource allocation
- the RTE integrates all necessary interface to the LTR language which calls the monitor by the "TRAP 2" 68000 statement
- LTR tasks use A4 to A7 address registers ; whatever for delayed or for immediate tasks these registers are loaded approprialy at the same time as the program counter and status register.

Besides the Real Time kernel, "MTR-D" includes some system tasks :

- processing of CAMAC interrupts such as "LAM" and "no X"
- handling the local/remote switch so that application tasks can be executed from a local console or be synchronized with other user tasks performed on the control computer via dedicated messages
- when in local mode, a console server allow to create and to delete tasks in a friendly way by simulating messages coming from the control computer
- other services have been added such as a debugger, a loader and an initialization routine

As not needed, the RTE neither includes memory allocation, mailboxes nor a file server.

## 2.2. Scheduling of the tasks

"MTR-D" is using a round-robin method but it does not provide a time-slicing of the tasks. If needed, such a service can be implemented using the 6840 timer on the CPU board. The scheduler is called in the following cases :

- a delayed task directly calls the scheduler or ends
- a delayed tasks suspends itself waiting for a delay
- a delayed task reserves, waits or frees a resource
- a delayed tasks creates or kills another one
- the shortest delay which was running has been elapsed

All these transitions are shown in figure 1 with indication of the potential states of delayed tasks. These states are defined as followed :

- dormant : there is no requirement to run the task
- required : the task has been "created" so that the scheduler could run it according to its priority
- running : the task is running, scheduled by the "MTR-D" kernel from the delayed task control block list
- suspended : the task is suspended pending a delay or a resource ; it moves to the ready state when the delay is reached or when the required resource is released
- ready : according to its priority, the task can be elected by the scheduler to be continued.

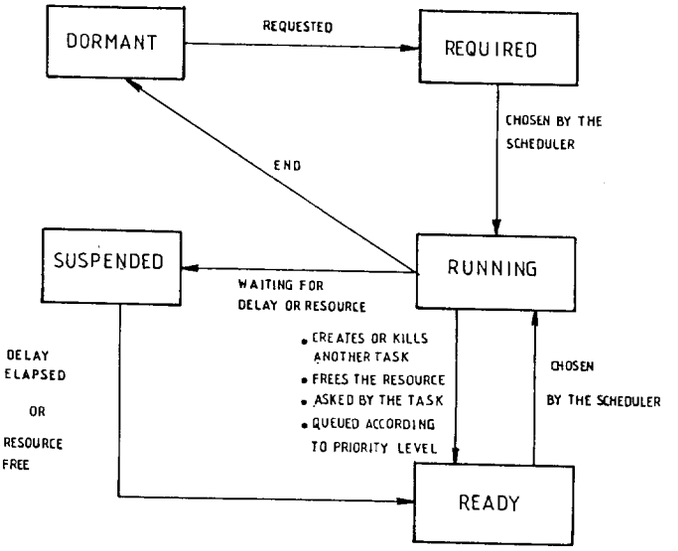


**Figure 1 : States of a delayed task**

## 2.3. The control block lists

The executive also contains an initialization routine loading the application (shared data and routines + MTR-D system + user tasks) from a disk ram or floppy. It also completes a static task control block list to map the whole application. A dynamic task control block list is therefore built for delayed tasks integrating additional informations such as the identifier number of the task, its priority, a task flag indicator ; it also includes space for context switching.

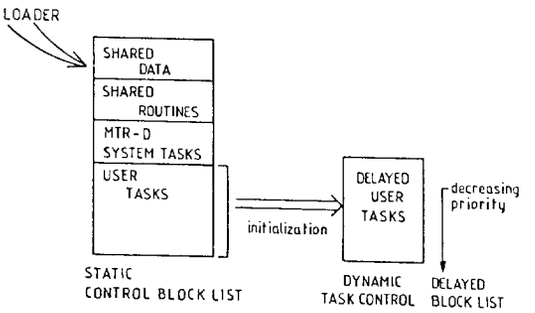


**Figure 2 : The control block list**

The delayed task to be elected by the scheduler is always the first task found in the delayed task control block list being in the ready state.

For delayed tasks, the scheduler loads address and starting point registers from the initial registers settings of the task ("required" state i.e starting the task) or from the context switching area ("ready state i.e resuming the task).

On interrupts, context switching uses directly the system stack. When first running an immediate task, address and program counter registers are loaded from the static task control block list.

2.4. Delays management

The PIT is used to achieve delays management ; the timer is always programmed with the shortest delay to be waited by a delayed task. Obviously, immediate tasks must not call for such a facility. Up to ten tasks can be waiting for different delays being therefore in the "suspended" state ; when the delay is over, they will be commuted into the "ready" state.

A waiting task list allows this management as shown in figure 3.

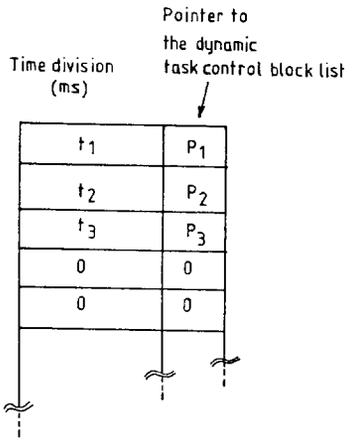


**Figure 3 : The waiting task list**

task "$P_1$" is waiting for $t_1$ ms
task "$P_2$" is waiting for $t_1 + t_2$ ms
task "$P_3$" is waiting for $t_1 + t_2 + t_3$ ms

2.5. Other features

A delayed task can create or kill another delayed task. Delayed tasks can be killed only if a dedicated bit in their flag indicator has been set. Killing a task is seen by the monitor as re-entering the task with initial registers settings and a dedicated bit in the task flag indicator. If needed, user tasks can integrate their own specific kill sequence (for instance, to reset some pieces of equipments or shared data ..).

Tasks can also ask for a resource. If the resource is free they automatically reserve it an go on running. If not, they will wait until the resource is free being therefore in the "suspended" state.

Two different bits in the task flag indicator allow to distinguish between waiting for a delay or for a resource. A killed task goes immediatly to the "ready" state even if it was waiting for a delay or a resource. The resource reserved by a killed task is automatically released.

2.6. Local/remote mode

Some system tasks have been added to the kernel to provide local/remote working mode. First, an immediate task is attached to a switch connected to the interrupt level 7 of the microprocessor, multiplexed with the ABORT signal of the 68000. Using this facility is quite efficient when testing programs and one can always change the working mode by acting on the switch. The task alternately enables and disables interrupts of the console port and of the CAMAC data link module with the control computer.

An other system task gives people a user friendly interface when performing tasks in a local way ; it can also simulate the remote working mode.

Finally any user task can be created or killed from the control computer. This is achieved via specific messages coming through the data link module handled by a dedicated system task.

## 3. AN EXAMPLE OF THE MTR-D USE : THE BEAM SWITCHING PROCESS

The beam switching process has been the first application running on the DIVA controller two years ago (5) ; as new needs had appeared, it has been re-designed this year taking benefit of the new facilities of the "MTR-D" monitor (6).

3.1. First aim of the process

The GANIL experimental area consists of nine experiment rooms. Two of the rooms can share the ion beam on a time distribution pattern required by the physicists. The process switches the ion beam alternately to one of the two experiment rooms and handles in that purpose two pulsed power supplies. It continuously surveys the power supplies currents and status. As shown in figure 4 the two rooms are named "upstream" and "downstream" rooms ; the upstream power supply is pulsed to achieve the time pulsation while the downstream one is obviously always delivering a continuous current. When the upstream power supply is pulsed to provide beam switching, the process handles at each room transition a beam cutter during 500 ms to let the current go to the appropriate value.

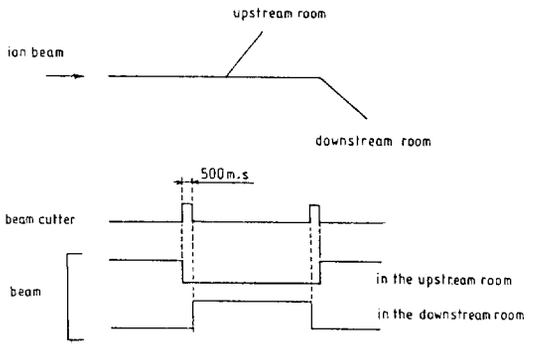


**Figure 4 : The beam switching process**

While switching the beam, the process also drives an intensity modulator to adjust the beam intensity for each of the two rooms.

3.2. Management of the experiment rooms

An other important job performed by the process is to manage the couple of experiment rooms when one of them starts or stops working, for instance when physicists want to enter the room to change targets or calibrate detectors. To do so, the process also drives a beam pulser and a beam stopper as shown in figure 5.

Configuration 1 is the standard working mode : both of the rooms are sharing the ion beam, the beam pulsation being given by the upstream power supply which is pulsed.

When one of the two rooms wants to stop receiving the beam, the other one may go on with a continuous beam or a maintained pulsed beam, switching is therefore kept by the beam pulser which simulates the pulsation of the pulsed power supply now delivering a continuous current. This last feature is mainly used when physicists synchronize their detectors on the beam switching signal.

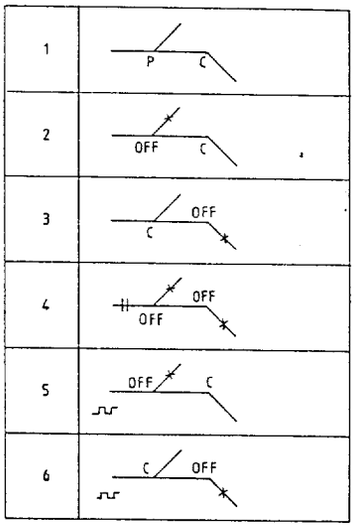


**Figure 5 : Configuration modes of the process**

P = pulsed power supply "on"
C = pulsed power supply "on" delivering a continuous current
OFF = pulsed power supply "off"
—— = the room is receiving the ion beam
-✶- = no beam is delivered into the room
-‖- = the beam stopper is set and stops the beam at the experiment switchyard entrance
⎍ = the beam pulser is set and provides the beam pulsation

So, when the upstream room stops, configuration 1 is turned into configuration 2 (continuous beam in the downstream room) or 5 where the beam pulser maintains the beam switching in the downstream room. An analog situation occurs for configurations 3 and 6 coming from configuration 1 when the downstream romm stops running, as already said the upstream pulsed power supply is here used in continuous.

Finally, configuration 4 corresponds to the case where both of the rooms have stopped accepting the ions beam ; the power supplies are turned off and the beam stopper is set at the experimental switchyard entrance.

### 3.3. Software aspects

The application process consists of 7 user tasks requiring about 400 Kbytes of memory.

One immediate task is in charge of surveying the power supplies status, currents and the stops of the beam intensity modulator and beam pulser.

The first delayed task of the process is an initialization task loading the names of the experiment rooms, parameters such as the starting current values and beam timing, use of the beam pulser... Other delayed tasks allow to modify the beam switching times, to change the power supplies currents values or to handle the modulator.

The more sophisticated delayed tasks of the process are a task in charge of the management of the two experiment rooms as described in 3.2 and a task devoted to the beam switching pulsation between the two rooms.

Finally, a task reads informations from the process such as power supplies currents values, status and sends them to the control computer.

Eight other user tasks can be performed on the control computer to interface the operator to the process, one of them collects and displays the parameters of the process on a graphic color terminal as represented on fig. 6.

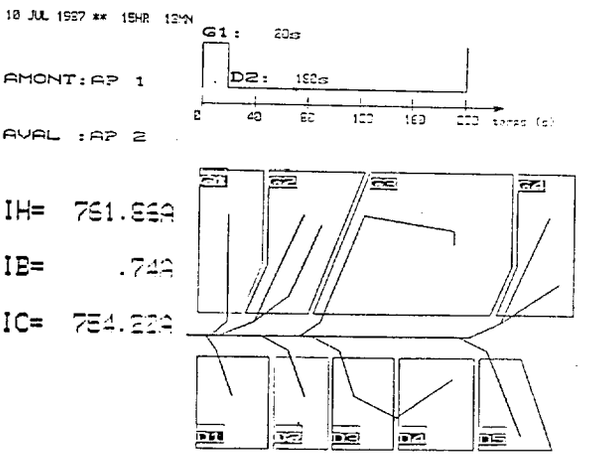


**Figure 6 : schematic display of the process**

"G1" and "D2" rooms are sharing the ion beam (20/180 seconds). The beam has been delivered to "D2" for about 50 seconds as shown on the display.

## 4. CONCLUSION

Despite the fact that the MTR-D executive is quite small and very simple, it gives us most of usual features required by a real time monitor. Having designed and written our own monitor, it's quite very easy to maintain. Some complementary facilities could be introduced if needed ; for greater convenience it would be useful to add some software utility such a spying facility to dump task lists and queues under an easy to interprete way.

The next application of "MTR-D" will be the handling of magnetic probes inside the cyclotrons, the corresponding programm is currently under way.

## REFERENCES

(1) E. LECORCHE and the GANIL Control Group
New developments of the Ganil Control System
2 International Workshop on Accelerator Control Systems - Los Alamos - U.S.A (October 1985)

(2) T.T. LUONG, L DAVID, E. LECORCHE and the Ganil Control Group - Present Status and recent improvements of the Ganil Control System .
This Conference.

(3) M. PROME and M. VASSSENT
Le dispositif Intelligent à Vocation Autonome, DIVA 68C - Ganil Internal Report 83R/035/CC/14

(4) E. LECORCHE and M. ULRICH
Le processus alimentations pulsées
Ganil Internal Report 85R/016/CC/05

(5) M. ULRICH and E. LECORCHE
"MTR-D" Moniteur temps réel pour DIVA
Ganil Internal Report 87R/24R/CC/17

(6) J.M. LOYANT
Evolutions du processus commutations de salles
Ganil Internal Report