IEEE Transactions on Nuclear Science, Vol. NS-32, No. 5, October 1985

SMACC

W. Heinze, R. Cailliau, B. Carpenter, G. Cuisinier,
A. Gagnaire, F. Perriollat, W. Remmer

CERN, PS Division, CH-1211 Geneva 23, Switzerland

Abstract

The SMACC is a powerful, MC68000 based Auxiliary Crate Controller for CAMAC. It will be used as principal processing element for the control of the LEP preinjector (LPI) [1], and hence will be part of the control system of the PS accelerator complex. Its utilization in LPI controls is multiple. First, the SMACC will support intricate beam instrumentation devices including buffering of fast data bursts, preprocessing of the data and calibration of the device. Second, the SMACC allows stand-alone control of entire subsystems like klystron/ modulator groups in the LPI. Third, it does cycle-to-cycle parameter refreshing (the so-called pulse-to-pulse modulation, PPM).

The SMACC is now produced in industry and available off the shelf. It is delivered with the real-time operating system RMS68K and a package for communications with the Front End Computer (FEC), burnt in EPROM. The languages supported under RMS68K are NODAL, an interpretative language used for accelerator control, and P+, a compiled block-oriented language used for writing the applications software modules. However, other languages like PASCAL can easily be made to run under RMS68K too.

Utilization of the SMACC for LPI control

Contrarily to the previously used ACCs which control the PS accelerator complex, the SMACC is not a pure auxiliary controller but acts in many aspects like an autonomous processor. In fact, the "New Applications Programming Structure" (NAPS) [1] distributes many of the processing tasks which were previously executed by the FECs to the SMACCs, thus making them the principal computing elements. One of the reasons is to allow control of LPI subsystems from a SMACC close to the process equipment as well as from the central Main Control Room (MCR), from where the whole PS complex is controlled. Thus, together with the FEC, the SMACCs form of a computer network. They communicate with each other over Serial CAMAC: for LPI, the FEC controls Serial CAMAC loops containing some 20 SMACCs. So, the physical layer of the computer communication consists of a medium which is not symmetrical (only the FEC can access the Serial CAMAC loop).

This basic hardware structure was chosen because most of the hardware interface modules exist in CAMAC. The restriction imposed by this choice is that all communications between SMACCs must pass via the FEC making them less efficient. However, SMACC to SMACC communication, being exceptional, will be catered for in a later stage. By far the most frequent case is reading of SMACCs by the FEC.

The first class of applications in LPI control consists of supporting beam measurement systems. In general, one or a few similar systems are interfaced with one CAMAC crate which is controlled by one SMACC. E.g., the magnetic beam position monitors are interfaced together with the wall current monitors. Another crate interfaces 3 SEMgrids and also contains the control for one magnetic dipole to form a system to measure the energy of the beam. Another crate is interfacing wire scan monitors for measuring the beam profile but also contains the control for the mechanical displacement of the wires. However, to allow beam emittance measurement locally, the SMACC of this crate must also be able to change the current in a quadrupole which is interfaced by another crate: this is one of the cases where SMACC-to-SMACC communication would be needed.

These 3 examples are representative for LPI beam measurement systems. As it is desirable that these systems can be operated locally even if the FEC is down, the crates contain - in addition to the interface modules for the specific beam monitors - a video display controller together with its memory. This provides the possibility to generate beam profiles, histograms etc. on local displays. Local access allows watching of the specific equipment during computer access: after connecting a terminal locally to the SMACC, the equipment can be accessed by writing small NODAL programs directly or by calling resident procedures (like equipment modules), also via NODAL calls. More important NODAL programs needing a secondary storage medium for storing and loading can be developed by connecting a MacIntosh [2] via RS-232C.

The next class of applications is similar to the one described above: the control of entire subsystems. A typical example is the control of 6 klystron-modulator ensembles each including high-voltage supply, de-Q-ing circuitry, delay lines, thyratron switch tube and klystron tank. One SMACC controls one klystron-modulator. This permits local control of this autonomously working subsystem for testing and tuning if the rest of the machine is down including the control system.

In the third class of applications, the SMACC acts as a controller simply changing the parameters of the LPI on a cycle-to-cycle basis, as required for accelerating or storing either electrons or positrons. The necessary information is derived from two sources: tables in the SMACC contain the parameter values, and the PLS telegram [1] sent to every SMACC via a specialized CAMAC module tells which set of parameters is selected for the next cycle.

To satisfy the above mentioned classes of applications, the complete SMACC package consisting of hardware design, operating system and programming languages has to be considered together.

Hardware description

The SMACC is a 2-board CAMAC module built around the microprocessor MC68000-L8. It contains 2 serial communication channels, a real-time clock, a calendar, LAM and interrupt handling, CAMAC access to the memory, CAMAC cycle generation and 2 communication registers with semaphore properties. The module can accept 48 standard dual-in-line memory circuits of 28 pins, i.e. either static RAMs of 8kbytes, EPROMs of 8k or 16kbytes or - limited to one column which corresponds to 8 circuits - EPROMs of 32kbytes. Thus, the total memory capacity varies between 384kbytes (all RAM) and 832kbytes (all EPROM except one column in RAM). Part of the RAM area, intended to contain the systems data, can be protected from write access in user mode. The standard memory configuration of the SMACCs in LPI controls consists of 128kbytes EPROM and 192kbytes RAM.

0018-9499/85/1000-2041$01.00© 1985 IEEE

The whole memory is accessible by the microprocessor and - via the CAMAC dataway - by the crate controller and hence the FEC. For the user, the memory looks like a dual port memory. For the MC68000, a CAMAC memory access is a DMA using the microprocessor's internal bus allocation protocol.

The CAMAC cycle generation of the SMACC is memory mapped, i.e. a CAMAC function is generated by accessing the corresponding CAMAC address space making special CAMAC call functions unnecessary. The data width is limited to 16 bits. The XQ response of the generated CAMAC access is stored in an internal CAMAC status register. Control functions have to be executed as read functions: the XQ response is already contained in the word or byte read.

Three different mechanisms are provided for the allocation of the CAMAC dataway between SMACC and main crate controller: via the ACL signal for L2 Serial Crate Controllers (the one used for LPI controls), via the ACB signals for A-2 Crate Controllers and via 2 handshake signals REQUEST/ GRANT AFC for ND-10/ ND-100 Dedicated Crate Controllers (ND-10s and ND-100s are the minicomputers used in the PS control system).

The communication registers consist of an input register which is written by a CAMAC function (hence by the FEC), but read and cleared by the SMACC, and an output register which is written by the SMACC but read and cleared by CAMAC functions. Reading and clearing are indivisible operations. Writing to the communication output register causes an interrupt to the SMACC, writing to the communication input register generates a LAM to the FEC. The communication registers are used to implement a proper communications protocol.

Other interrupt sources for the SMACC are all LAMs from the other modules in the CAMAC crate, 4 external (front-panel) interrupts, the real-time clock, the 2 serial communication channels, a CAMAC abort function and power fail. The interrupt priority levels for the MC68000 are burnt in PAL and can be altered if a special application should need it.

The SMACC performs a simple LAM grading for up to 3 LAMs, selected by jumpers: one from the SMACC station itself, one from another station, and the power fail LAM L24. The LAM grading consists in generating a SGLE pattern for the Serial Crate Controller.

The two serial communication channels are independently programmable. One channel contains the electrical signals necessary for RS-232C and 20 mA current loop interfaces, the other for RS-232C and RS-422 interfaces. The respective electrical standard is selected by the connection cable. The interface chip Z8530 allows programming of a large variety of serial communications protocols. The baud rates of both channels are programmable and independent.

It should be mentioned that the mechanisms provided for the dataway allocation are sufficient to permit several SMACCs to work together in one crate. However, this possibility will probably not be used in LPI controls.

The complete hardware description of the SMACC is found in [3].

Operating system

All SMACCs in LPI controls run with the real-time operating system RMS68K from Motorola [4]. Together with the I/O drivers it occupies roughly 16kbytes of EPROM and 32kbytes of protected RAM. The RAM represents the dynamic data space. Its basic concept, like in all executives of this kind, consists of breaking down the application system into several tasks and executing them concurrently by moving the tasks through various task states. Task states are READY (ready to run), RUN, SUSPEND, WAIT FOR EVENT, WAIT FOR COMMAND, WAIT ON SEMAPHORE etc.. A task makes a state transition when any of the following actions take place:

a) A task control directive is issued by the task itself or by another task while it is running. RMS68K provides some 100 directives; they are all accessible via NODAL.

b) An exception occurs in the running task.

c) An event is placed in the task's asynchronous service queue.

d) Special function handling initiated by RMS68K such as timeouts (used in the datagram service), and semaphore signaling (possibly used to synchronize display tasks with data acquisition tasks).

For LPI control, an important service offered by RMS68K is the possibility to run tasks concurrently in a proper way. A typical case is, e.g., when a program in the FEC reads a data table in a SMACC at the "same time" as a NODAL program in the SMACC, and on top of that, another program in the FEC (running independently from the data acquisition program) changes some control parameters in the SMACC. Although synchronization between these various activities could also be done specifically for every application, the use of standardized services of an operating system simplifies writing and improves maintainability of the applications programs.

If one does not choose to synchronize tasks by attaching them to timing pulses arriving in correct order, RMS68K provides 3 types of semaphores to do the job:

a) type 1 semaphores are used when more than one task requires exclusive access to one resource,

b) type 2 semaphores are used to control the execution sequence of tasks,

c) type 3 semaphores are used when one task controls a resource which other tasks wish to use.

Using these and other services of RMS68K is done via directives and allows the implementation of complicated, concurrently executing tasks. Directives can be issued directly from Assembler and Nodal programs. They are hidden from the programmer in the high-level language P+, which has its own mechanisms for program synchronization. (Internally, of course, the P+ run-time library routines make use of the RMS68K directives).

Utilization of RMS68K directives is very similar in Assembler and NODAL programs. In Assembler programs, the requesting task loads the directive number into register D0, loads the required parameter, if needed, into address register A0, and then executes a TRAP #1 instruction. E.g., the directive SGSEM ("signal release of a semaphore controlled resource"), named R1SEM, is written as

```
MOVEQ   #SGSEM, D0      MOVE DIRECTIVE NR. INTO D0
LEA     R1SEM, A0       MOVE PARAM. ADDRESS INTO A0
TRAP    #1              PERFORM DIRECTIVE CALL
```

In NODAL, the same directive is issued by

```
>OSIGNAL-SEMAPHORE "R1SEM", key
```

A no less important role of the SMACC is its duty to act rapidly on demands arising from the equipment interfaced by modules in the SMACC's CAMAC crate. In general, it asks for a data acquisition which has typically to be started within some 100µs and finished in about 1ms. To take into account this auxiliary crate controller's role of the SMACC, the RMS68K's Asynchronous Service Queue (ASR) mechanism is too slow. Instead, the Interrupt Service Routine (ISR) mechanism is used. ISRs are directly connected to interrupts generated by the process equipment calling for a rapid action. These interrupts arrive at the SMACC as LAMs or pulses at the SMACC's front panel. Consequently, interrupts of this kind have next to the power fail and the ABORT function the highest priority in the SMACC. ISRs are started only ca. 70µs after arrival of the interrupt, but cannot use any RMS68K directive except the one to quit the ISR's state. Of course, programming of ISRs has to be done very carefully because infinite loops would block the system.

After termination, an ISR can also provoke execution of an Asynchronous Service Routine (ASR): the time critical part would then be done by the ISR, access to RMS68K services by the ASR. E.g., an I/O driver could be written in that way: reading of the characters could be done by the ISR, but at the end of a line the ASR would store the whole character string in a buffer.

Communication SMACC - FEC

The lowest communications layer next to the physical layer, called the data link layer, provides simple, unprotected transfers of blocks of data. The associated routines are GETBL (read a block of data from the SMACC memory) and PUTBL (write a block of data to the SMACC memory). In fact, these simple but efficient transfer routines are directly used to transfer the bulk of data acquired by the SMACC to the FEC, before they are processed and/ or displayed on the consoles in the Main Control Room. In this case, the proper synchronization between acquisition and reading is performed by external timing pulses representing the cyclic behaviour of the LPI machine, i.e. data acquisition takes place during beam cycles and data reading between beam cycles (or at other moments during which no data acquisition takes place).

Using this low level block transfer, a datagram service is implemented which offers a more protected but slower form of communications. The additional features needed by the datagram service are the Communications Input Register (CIR) and the Communications Output Register (COR) in the SMACC. They provide a handshake mechanism which makes sure that data, transmitted via the datagram service, are really transmitted and transmitted only once. The factor limiting the efficiency is not so much the speed of Serial CAMAC but the asymmetry between FEC and SMACC accessing each other via CAMAC, and the LAM handling in the FEC.

The datagram routines are used to construct Remote Procedure Calls, RPCs, called from P+, and the IMEX/ EXEC functions, used by NODAL. E.g., a RPC in P+ looks like

```
computer_name & function_name(parameters)
```

where the function resides in the remote computer. In NODAL, typing the result of an equipment module access which resides in a remote computer looks like

```
>IMEX(computer) SET CC=0; TY! TRAFO(parameters,CC)
```

The datagram service can be associated with layer 3 (network layer), the RPC, IMEX, EXEC calls with higher layers of the ISO model. Remote procedures and IMEX/ EXEC calls will mainly be used by control programs running on the FEC and accessing equipment via the software in the SMACCs.

Language support

Languages supported are Assembler, P+ and NODAL. P+ is a block-oriented language with PASCAL flavour; it supports programming in "modules", by allowing separate compilation of these modules [5]. Protection of non-reentrant procedures is done with the help of "critical routines" which rely on the use of RMS68K semaphores.

The P+ compiler does not run on a SMACC but on a ND-500, a 32 bit computer, on which the complete program development is done. The equivalent is valid for programming in Assembler. The language support routines being part of the operating system are written in Assembler. An interactive symbolic debugger (MoniCa) [7], to be installed soon, will allow such operations as tracing, setting breakpoints, inspection/ modification of variables and single stepping/ single procedure calls.

NODAL is an interpretative language executing line by line [6]. The advantage of NODAL is that even non-programmers can easily write and test little programs at the cost of a much lower execution speed. The complete NODAL system is resident in the SMACC, it includes the complete set of NODAL functions and the commands of the RMS68K directives. NODAL as well as P+ use the same terminal I/O driver supported by RMS68K. Remote file access is available from the MacIntrotte [4]. Planned is also the possibility to remotely access files from the FEC's mass storage via Serial CAMAC.

References

[1] D. Kuiper, "Controls for the LEP preinjector", this conference.

[2] F. DiMaio, F. Perriollat, "MacIntrotte", this conference.

[3] W. Heinze, "User's manual of the SMACC", CERN PS/ CO/Note 84-24.

[4] Motorola, "M68000-family real time multitasking software user's manual", 1984.

[5] R. Cailliau, B.Carpenter, "P+, a real time control language user's manual", CERN PS/CO/Note 82-21.

[6] M.C. Crowley-Milling, G.C. Shering, "The NODAL system for the SPS", CERN 78-07.

[7] H. von Eicken, "MoniCa, a symbolic debugging monitor for the M68000 family, user's guide", CERN/ DD, in preparation.