EUROPEAN ORGANIZATION FOR NUCLEAR RESEARCH

The Readout Controller of the Aleph Time Projection Chamber

S.R.Amendolia[1], F.Fidecaro[1], S.Galeotti[1], B.Löfstedt[2], A.Lusiani[1], P.S.Marrocchesi[1] and D.Passuello[1]

Geneva, 26 September 1985

Abstract: The readout system of the Aleph TPC is described. Each of the 72 processors has to perform data formatting and reduction, chamber monitoring and calibration, as essential tasks to ensure the quality of the data coming from the detector. The implementation of the processor as an intelligent Fastbus module based on the Motorola 68020 microprocessor is described. Solutions for Fastbus interfacing and for data processing are discussed. Operational problems raised by the complexity of the system are also briefly discussed.

1. Introduction

The Aleph Time Projection Chamber is an imaging detector providing 3-dimensional position measurements of particle tracks. The information is obtained by letting the ionization electrons drift towards the chamber endplates, where these are detected by a system of 6'000 proportional sense wires and 44'000 cathode pads. The r-ϕ measurement is obtained by analizing the relative pulse height on neighbouring pads, while the position along the drift direction is obtained from the time of the pulse.

Each analog channel is connected to its own circuit of fast sampling electronics, so that the 44 μs long signal is digitized into about 500 8-bit samples. This result in 25 Mbytes of raw data being stored in the read-out system at each event. The suppression of non-significant data is achieved in the digitizing module, by considering as valid data only those samples belonging to a succession of samples above a given threshold. The useful data are then formatted and processed by a dedicated intelligent Fastbus module, the Time Projection Processor, to be then assembled as part of a full Aleph event. This paper describes the functions this module has to perform, not only to read out the data but also to analyze these in order to monitor the performance of the detector and of its electronics.

2. Structure of the TPC readout

Each of the two endplates of the Aleph TPC is subdivided into 18 sectors (see fig. 1), to optimize the tracking efficiency still having an acceptable dead space. The sensitive part of each sector is made out of typically about 180 wires and 1500 pads. While the preamplifiers reside on the sector, the rest of the sector's electronics is housed in the Time Projection Digitizers (TPD), residing in 3 Fastbus crates and coping each one with 64 channels. For each sector there are 3 TPD for the wire channels and from 18 to 24 TPD for the pad channels, residing in 3 Fastbus crates (see fig. 2 for the structure of the TPC readout from the Fastbus point of view). These modules, together with the calibration system, are controlled in each sector by 2 Time Projection Processors, one for the wires, the other for the pads. The TPP are connected by cable segments to the TPC Event Builder, where the data from all sectors are assembled to build the TPC event. The TPP are Fastbus masters on the Fastbus crate segment,

[1] INFN, Dipartimento di Fisica, Scuola Normale Superiore, Pisa, Italy.
[2] CERN, EP Division, Geneva, Switzerland

the latter being extended to a set of 3 crates by means of Crate Clusterizing Cards [1]. The TPD are slaves on the same segment. On the other hand, the TPP are slaves on the Fastbus cable segment that comunicates with the Event Builder.

When an event is accepted by the first level of the trigger logic, the sampling electronics records the pulses from the channels. This is foreseen to happen at a rate of at most 500 Hz. The second level trigger (at a rate of at most 10 Hz) enables then the readout of these data. As a first step, a simple zero-suppression procedure is performed inside each TPD. The output of this procedure is a collection of pulses, characterized by a channel number, a beginning time and a pulse length, together with the relevant samples. The data are collected by the TPP, where they receive appropriate treatment, as described in the next chapter. Data from the TPP are then collected by the TPC event builder.

The typical data volume from the TPC for a Z^0 event amounts to 60 kbytes, 80 % of these coming from the wires and only 20 % from the pads. These data are not uniformely distributed over all sectors: on average a sector contributes with 1.8 kbytes, but it can have up to 20'000 good samples. These fluctuations are dealt with by a scheme that includes 4 front end buffers in the TPD and 2 output buffer in the TPP. The effectiveness of this setup to reduce dead time has been studied by simulating the whole behaviour of the whole Aleph read out system [2].

3. Operations to be performed by the TPP

3.1 Operations during read-out

The minimal set of operations to be performed under TPP control during data acquisition are :

- data read-out from TPD modules.
- data formatting

The first operation consists in performing Fastbus transactions with the concerned TPD, then to store the data inside the TPP memory in order to avoid unnecessary memory-to-memory copies. Block identifiers and wordcounts have to be added, to build the required data structure and to allow subsequently an efficient processing. In addition, channel numbers become changed to the detector numbering scheme, the latter being closely related to the geometrical position inside the chamber.

3.2 Beyond read-out

An online data reduction can be performed safely on the wire data, which contribute to most of the data volume, but are carrying information less critical than the one coming from the pads. The reduction consists in replacing the samples by a pulse height and time computed with a simple algorithm. The expected volume of data from the TPC after reduction is about 60 % of the original one. This reduction has to be performed at a speed of 4 μs/sample, requiring therefore specialized hardware.

If allowed by the trigger rate, the installed computing power can be used to perform parallel tasks that do not require comunication between processors. An example of this is the recognition of pulses that are at the same time on neighbouring pads and the subsequent computation of the position. Another possibility is the recovery of samples near a good pulse that were lost by the zero-suppression procedure. If performed, these tasks would ease the amount of offline computing needed to reconstruct a TPC event.

3.3 Other TPP tasks

A number of tasks have to be performed asynchronously and sometimes concurrently to the data acquisition. These include the management of the system itself, the calibration of the analog chain and monitoring the TPC.

The purpose of the electronics calibration procedure is to adjust all channel's gains within 1 % before data taking, so that the data do not require further offline corrections. This calibration of the TPC is expected to take only a few minutes, using the parallelism achieved by the TPP.

To monitor the TPC and its readout system, several programs will run inside the TPP. As a basic requirement, checks on the read-out functionality will be provided, flagging time-outs and errors and ensuring system recovery in case of faults. Statistical information with and without distinction of channel can be collected to monitor the performance of the detector with respect to noise, discharges, etc. These distributions on a channel by channel basis, together with hit frequencies will then help detecting faulty channels. These tasks will be performed during TPP idle time, keeping an event in memory until it is fully analyzed. Results will then be sent to the host computer, to be analyzed and archived by an appropriate software.

4. TPP hardware

The functional block structure of the TPP is shown in fig. 3. It consists of a CPU and a bus which sees memory and peripherals, the most important among them being the Fastbus coprocessor, the slave port on the cable segment and the local area network interface.

4.1 CPU and bus

The CPU is a Motorola 68020. The reasons for the choice of the 68000 family are mainly the flexibility achieved by the powerful set of instructions and the availability of software both inside and outside CERN. In addition it is the supported family of microprocessors inside CERN. The choice of the 68020 is then due to the fact that it supports external 32-bit architecture, the one of our data acquisition and offline computers, that it has a high instruction execution speed, adequate to our data volume, and also because it features the concept of co-processor, that allows an easy interface to special hardware (like FB or signal processing unit) or to a floating point unit.

4.2 Memories

The memory accessible from the CPU is subdivided as follows : there is an EPROM to contain start-up and some system code, the main RAM where programs reside, the output buffers that can also be accessed from the slave port and a section of the main RAM which allows duplication at writing time of an output buffer.

The ROM is 64 kbytes, and it contains the resident monitor. The RAM is 512 kbytes, with an access time of 55 ns, requiring thus no wait states.

The two output buffers have a size of 32 kbytes each. However only one of these buffers is accessible from the CPU, the other appearing instead as Fastbus data space on the cable segment. The buffers are swapped by a switch that can be set by writing in a special location of the 68020 address space. The monitoring buffer, also with a size of 32 kbyte, can contain a copy of an event, to be analyzed during between the arrival of events. To avoid active copying by the CPU, the monitoring buffer is written at each time the bus makes a write cycle addressing the output buffer. This is enabled by a switch at some place in the 68000 address space.

4.3 The slave port

The slave port allows access and control of the TPP via FB through the CSR space and allows the transfer of the TPC data that are present in the Data space. In addition, it must be able to communicate with the event builder, by asserting a Service Request and by responding to FB operations, either while being addressed and during Broadcasts operations.

4.4 Fastbus master port

The cluster of crates containing the TPD is not connected via an SI to the Aleph Fastbus configuration. Since there are only 2 TPP that can act as master on the extended backplane, and they don't need to comunicate, the TPP has only a master port on the backplane, without slave. Thus, the master port on the backplane is in normal operation the only way to access the TPD's, for read-out, calibration, testing, debugging. An analysis of the operations involved in these segments shows that there is no need of implementing "elementary primitive" Fastbus commands: only complete operations are at disposal of the user (e.g. no primary address cycle alone, without secondary address and data cycles). Implementing separately, for higher flexibility, the various components of a FB instruction would require a more complex machine instruction and a greater number of hardware components. On the other hand, the TPD transmit part of its data in 8-bit format, without packing these in 32-bit words. Thus the TPP needs to implement a block transfer capability for 8-bit or 32-bit data read/write. The 8-bit data will be packed into 32-bit longwords in the TPP before writing into memory.

The following FB operations are available at the master FB port:

i. random read/write - 32 bit
   options:

- keep bus at end of transaction
- release bus

ii. block transfer read/write options:

- DMA or NON DMA mode
- 8 or 32 bit field

During block transfer read the following options are available:

- START : starts block xfer read on memory longword boundary
- CONTINUE : resumes block xfer read from previous address (inside a longword if necessary)

For random r/w, the primary address (PA), the secondary address (SA) and the datum (D) can be given in the instruction stream, or can be found in a block in memory addressed by the istruction; for block transfers, a block is always created in memory.

In case of a random r/w FB command issued by the VAX and to be executed on a TPD (as a partial software emulation of an SI), the following steps are followed:

- a block with PA,SA,D is written to TPP's CSR space
- the 68020 is appropriately interrupted
- a command is issued from the 68020 to the FB coprocessor to execute the "list" of PA,SA,D described in the block.

An alternative way, since the TPP slave port can access the whole address space, is that any master, emulating the 68020 coprocessor protocol (e.g. the VAX), can access directly the master port in the TPP.

To implement the communication between the 68020 and the FB coprocessor, only General Type instructions are used [3]. One 16-bit Command Word is enough to accomodate all the options (see fig. 4). This Command Word is fetched by the coprocessor immediately after the General Type (GT) instruction. The 68020 waits for the end of the Fastbus transaction before executing the next instruction. There are therefore less problems in case of errors in a multitasking environment.

The events that cause the coprocessor to generate a trap are the following:

- invalid command word
- parity error
- timeout on AR, AS, DS
- SS not 0 during random R/W
- SS not 0 or 2 during block transfer

A 32-bit register is available, reporting the state of the Fastbus and of the interface at the moment of the error.

The implementation is made with a set of 3 finite state machines, which transitions are caused by the various signals of the FB and 68020 coprocessor protocols. All signals are sampled and latched by the 68020 clock. With a 60 ns clock for the CPU, the coprocessor can transfer a 32-bit word in block mode and DMA every 90 ns, if the slave responds to DS in less than 50 ns, and every 120 ns if the slave responds in 50 − 80 ns.

4.5 Auxiliary Arithmetic Unit

This other 68020 coprocessor will perform the wire data reduction. It will consists of a dedicated signal processing unit, to perform the reduction algorithms at the required speed. The design of this unit will start at the end of this year.

4.6 Interface to the Local Area Neork

The foreseen LAN is, at the moment, Cheapernet. An interface, possibly run by a 6809 microprocessor, is required to transfer data from the network to the main memory. This interface should be able to recognize specific messages, like "reset TPP", to be used in case the FB hangs up. The design of this interface, possibly using existings projects, willlso start at the end of this year.

5. Software considerations

5.1 TPP general software

There are many programs expected to run concurrently in the TPP. Together with a high priority data acquisition and formatting process, there will be one or more monitoring programs, that will run during idle time. All these programs require to comunicate with the TPC VAX, for error logging and to store the accumulated statistics for further processing. Dialogue with the host is also required for downloading and to specify what the TPP should do. We expect that these programs will run under supervision from a real time kernel (RMS68K is for the moment our candidate) to exploit priority schemes and resource allocations provided by such a software.

5.2 TPP Fastbus software

To allow migration of programs, the standard CERN Fastbus Fortran calls are implemented, as an FBPACK, written in assembly language. These routines fill the extension word of the FB coprocessor instruction according to the values of their parameters and then execute it. The exception handling routine returns an hardware and software status stored in word 2 of the STATUS I*4 array, and an "F – code" status stored in word 9, to give access to the state of the finite machines and of the Fastbus signals at the moment of the error.

In addition to the standard calls there is:

- a KEEP BUS option in single listener routines
- a BLOCK TRANSFER CONTINUE option
- a RELEASE BUS option when secondary address is transferred

5.3 VAX software

The TPC VAX will manage the whole read-out system, by down-loading the programs that run in the TPP, triggering checks and calibrations, collecting results and preparing the system for data acquisition. There will be various running modes of the system (TPC normal data acquisition in experiment of stand-alone, self-triggering, calibrations, ...). In particular, there will be on the VAX a management program that can access all TPPs, down-loading the appropriate code for each running condition and ensuring to the operator access to the single TPP in the debugging phase.

This requires a careful design of the dialogue procedures between the host computer and the TPP and also of the way the system is seen by the user. The latter point will determine the manageability of the TPC read-out in absence of the experts.

The collection and analysis of the calibration constants and of the results of monitoring will give the ultimate checks of the performances of the system. Due to the large number of channels, at least semi-automatic analysis should be provided in order to detect faulty channels. This includes availability of calibration constants in an interactive database management system, statistical analysis to pick up channels out of tolerances, histogram comparison facilities to evidentiate breaking or smooth deterioration of elements, etc.

5.4 Production of the TPP software

The programs will be adapted from running programs on the TPC VAX where we expect to have a very similar environment (for monitoring tasks) to the one in the TPP. These programs will then migrate most of the calibration and test software will migrate from the TPC VAX to the TPP as we expect not to switch from FORTRAN 77 to another language (Pascal or 68020 Assembler). We expect to write only the data acquisition part of the software (including the drivers for the peripherals) in 68000 assembler, relying on the goodness of the code produced by the available compilers for the remaining software. This software is available (real time KERNEL, VAX cross compilers, linkers and pushers) and is being adapted to CERN needs by current microprocessor users. We expect this software to be sufficiently performant at the time the first TPP is tested.

The Auxiliary Arithmetic Unit, if programmable, requires an ad-hoc assembler that can generate the microcode. The design of the microcode and of the assembler goes in parallel with the design of the unit. The task of producing a compiler is then a well mastered exercise.

References

[1] The Crate Clustering Card S.R. Amendolia et al., these Proceedings.
[2] Aleph DAQ system - Hardware Functional Specifications (in preparation)
[3] 68020 design specification manual, p. 126, Motorola.

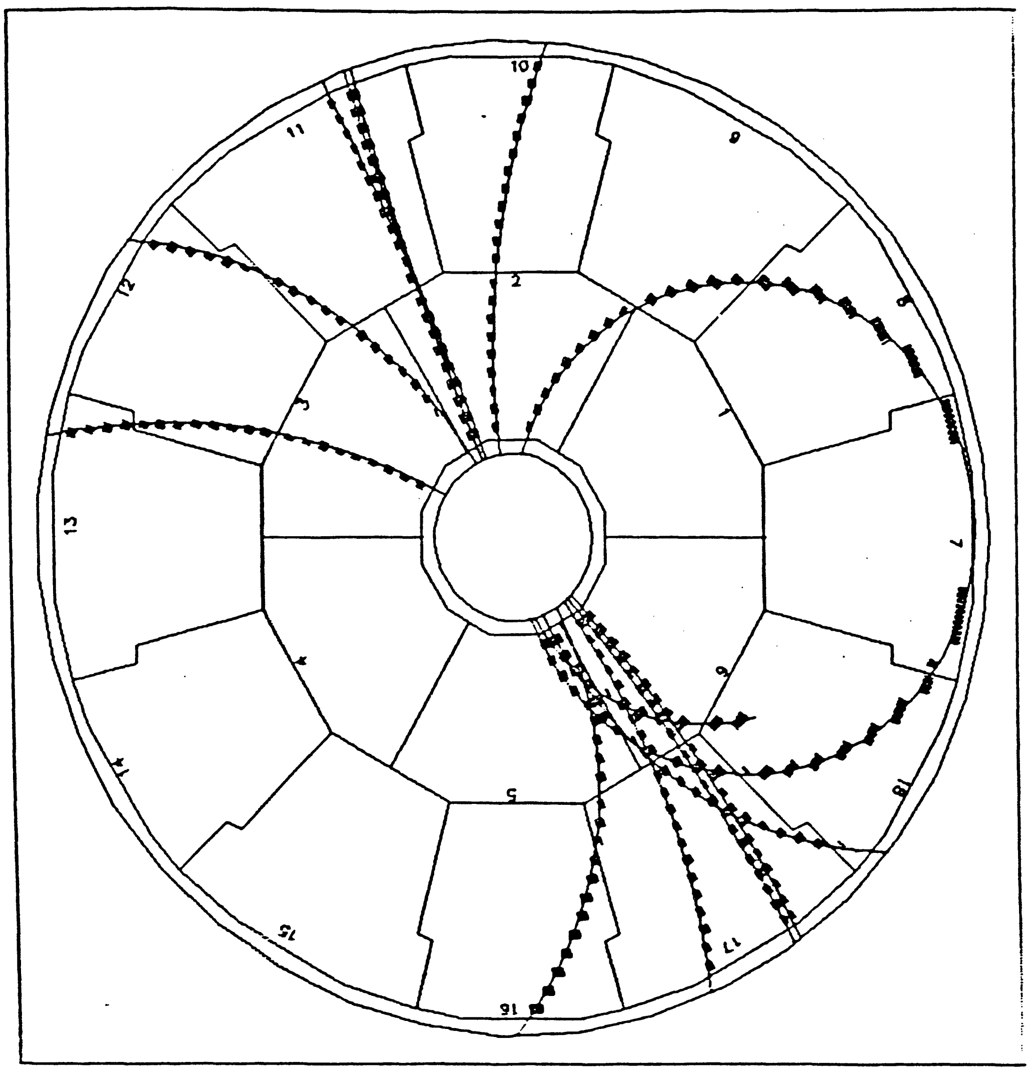

Fig.1 Aleph TPC endplate

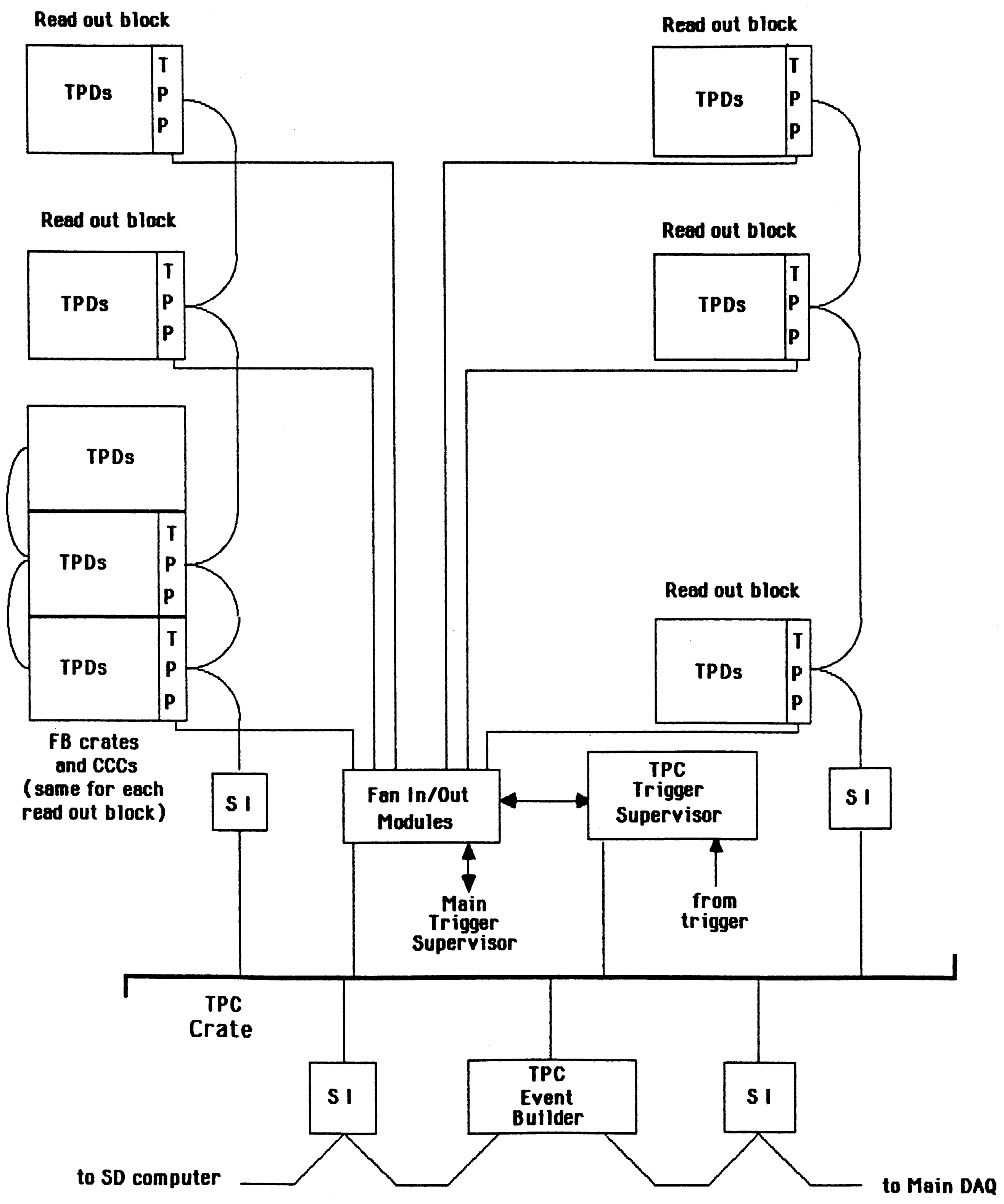

Fig. 2 : TPC read-out scheme

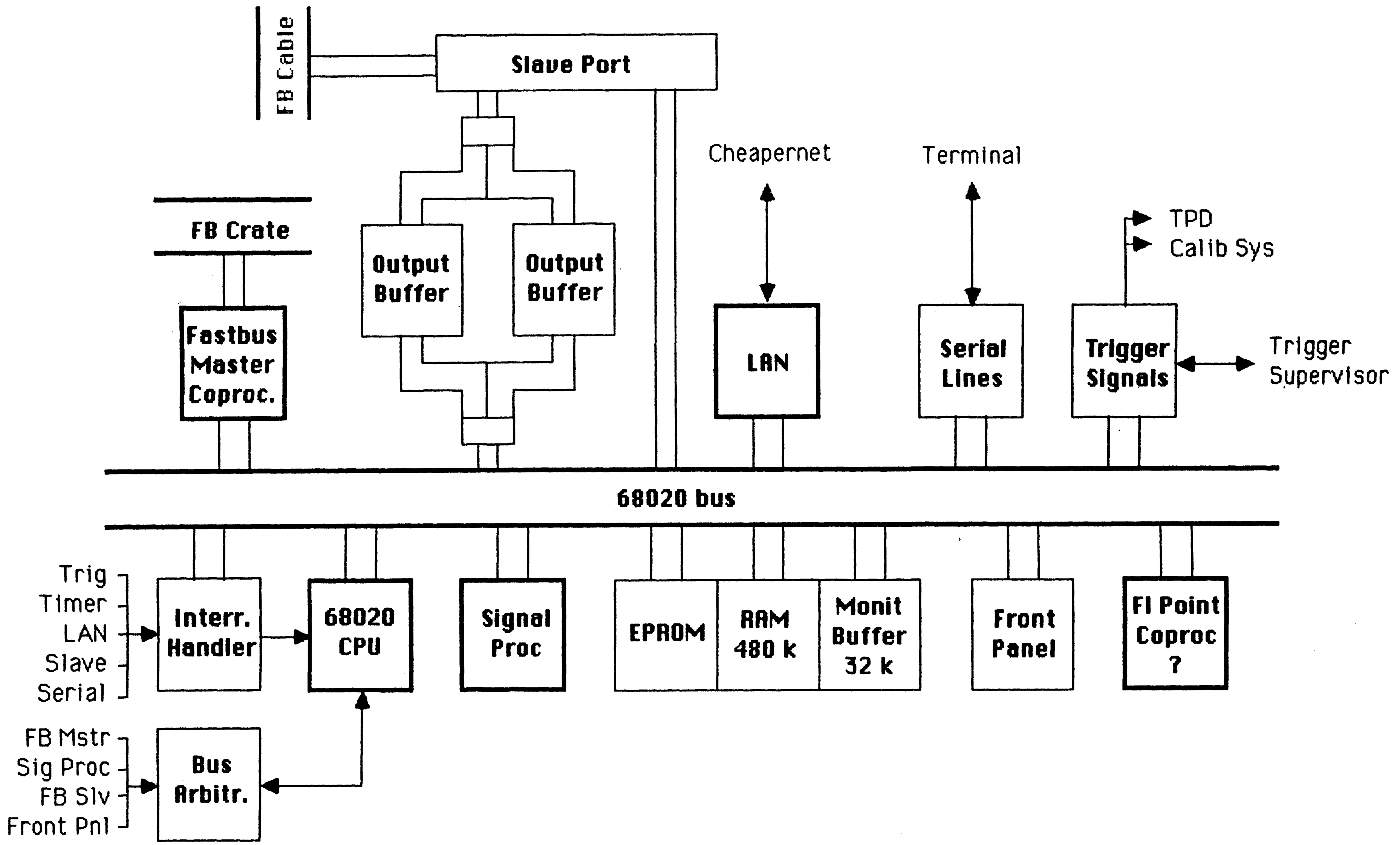

Fig. 3 : TPP functional structure

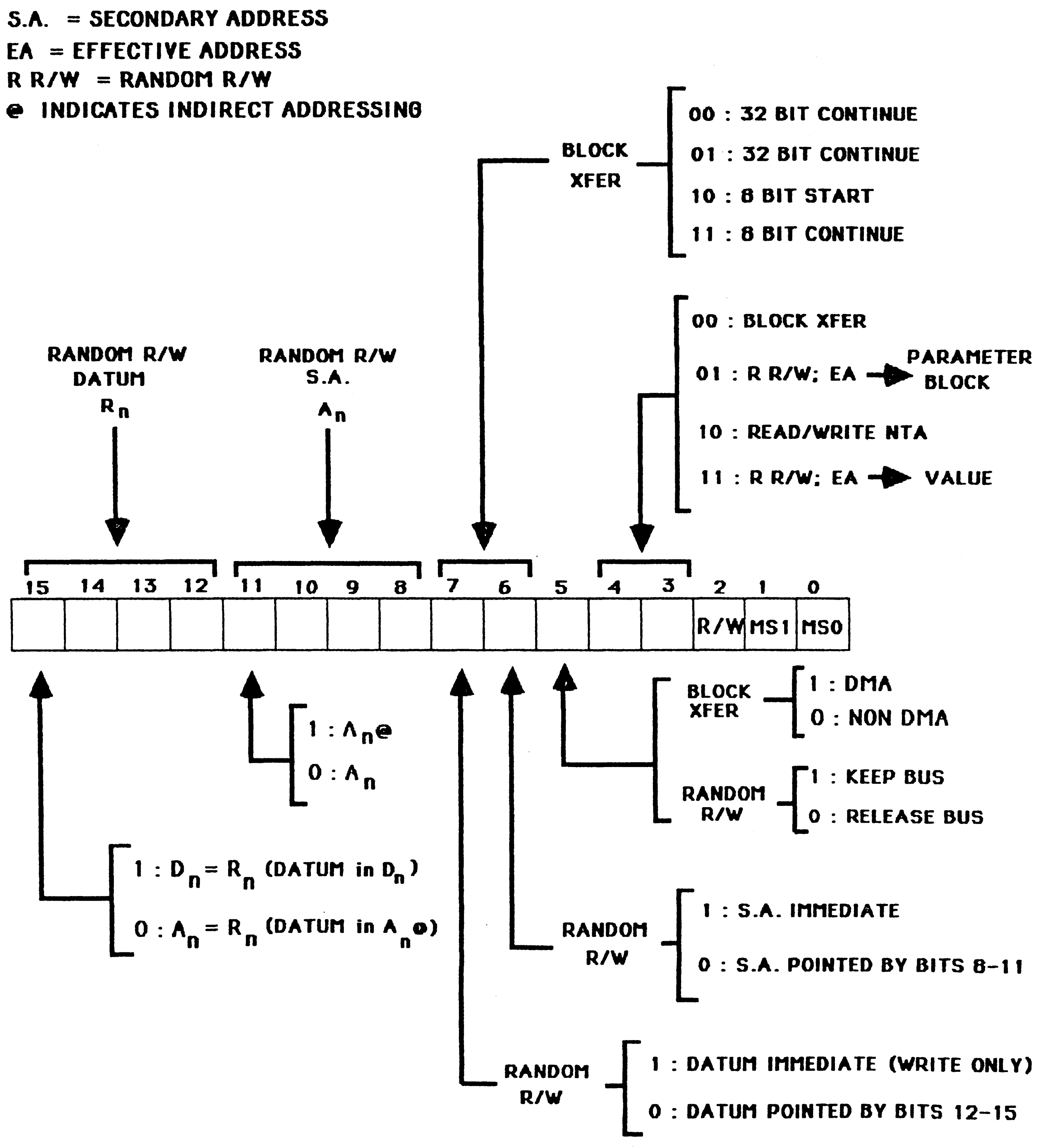

Fig. 4 : Extension word to the General Type Coprocessor Instruction in the TPP Fastbus Master Port