I/O Module Variants in the DAQ-Unit of DAQ/EF Prototype -1

Authors : H. Beck, D. Francis, M. Joos and J. Petersen
Keywords : ROC, SFC, DAQ-Unit, I/O Modules, Tasks, Message Passing, Scheduler

Abstract

The models of the Read-Out Crate, the Sub-Farm Crate and the DAQ-Unit in the DAQ/EF prototype -1 are reviewed. Implementations of the Read-Out Crate and the Sub-Farm Crate on different configurations of Single Board Computers (SBC) are discussed and the software analysed with emphasis on the task structure of the I/O Module instances. For each implementation, measurements of the event rate in the Read-Out Crate and the Sub-Farm Crate were performed with event fragments generated either locally (by the SBC) or via ROB-IN's (Read-Out Crate only). The hardware consisted of PowerPC based VMEbus SBCs (CES RIO8062) equipped with up to four ROB-IN's (CES MFCC8441).

NoteNumber :
Version : 1.3
Date : 25-02-00
Reference :

Document Change Record

Table 1. Document Change Record.

1. Document Title:			
2. Document Reference Number			**DAQ Technical note number xxx**
4. Issue	**5. Revision**	**6. Date**	**6. Reason for change**
1	*0*	*08 Nov. '99*	*Birth.*
1	*1*	*02 Feb '00*	*Enlarge scope from ROC-only to cover entire DAQ-Unit*
1	*2*	*15 Feb '00*	*Added SFC measurements and conclusions*
1	*3*	*25 Feb '00*	*remove L2IF, review conclusions*

1 The Read-Out Crate, the Sub-Farm Crate, the DAQ-Unit and the I/O Module.

1.1 The Read-Out Crate Model

An analysis of the main data flow in the Read-Out Crate (ROC) has allowed to identify three major functional modules: the TRG, the EBIF and the ROB [1]. Figure 1 shows the relation between them and, in addition, their interfaces to the trigger, detector front-end and event builder systems. A brief description of these modules is given below, for more details, see [1].

The TRG receives and buffers data control messages from the Level 1 and/or the Level 2 trigger systems and sends data control messages to the ROBs and the EBIF. The Level 2 reject messages (L2R) are sent to the ROBs and inform those to reject one or more ROB fragments. Level 2 Region Of Interest messages (ROI) are sent to the ROBs to inform those to provide data for the Level 2 trigger system. Level 2 Accept messages (L2A) are sent to the EBIF which, upon reception, performs the process of Data Collection (DC) i.e. collects the event data fragments from the ROBs, builds the crate fragment and forwards it to the event building system. On completion of the data collection, a discard message is sent to the ROBs which remove the event which has been collected. The ROB receives and buffers input from the detector front-end system via the Read-Out Links (ROL). One or more ROLs may be associated with one ROB (e.g. in the case of ROB-INs).

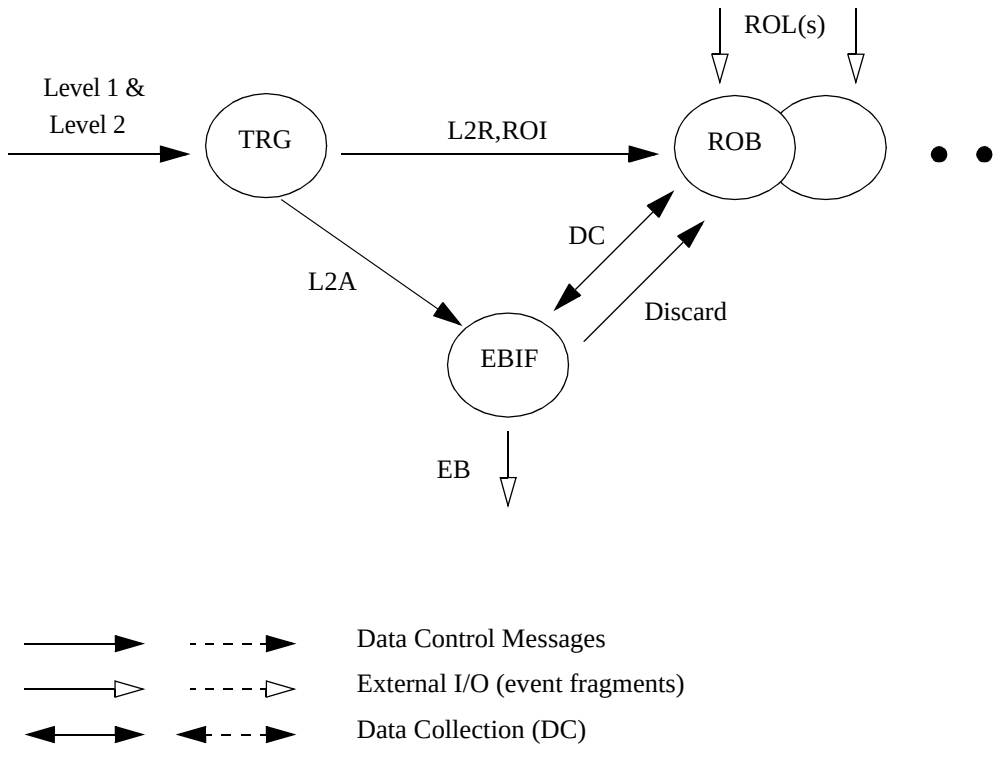

Figure 1. Logical model of the data flow in the ROC.

1.2 The Sub-Farm Crate Model

An analysis of the main data flow in the Sub-Farm Crate (SFC) has allowed to identify two major functional modules: the SFI, and the SFO [1]. Figure 2 shows the relation between them and, in addition, their interfaces to the Event Builder, Event Handler and Mass Storage systems. A brief description of these modules is given below, for more details, see [1].

The SFI receives, buffers and sends data control messages and receives, buffers and merges ROC fragments from the Event Builder system. It subsequently sends the completed events to the Event Handler. The SFO receives complete events, possibly reconstructed and filtered, from the Event Handler which are then sent to the Mass Storage system for recording. The role of the Event Handler is to separate the processing elements, responsible for event reconstruction and filtering, from the Sub-Farm Crate.

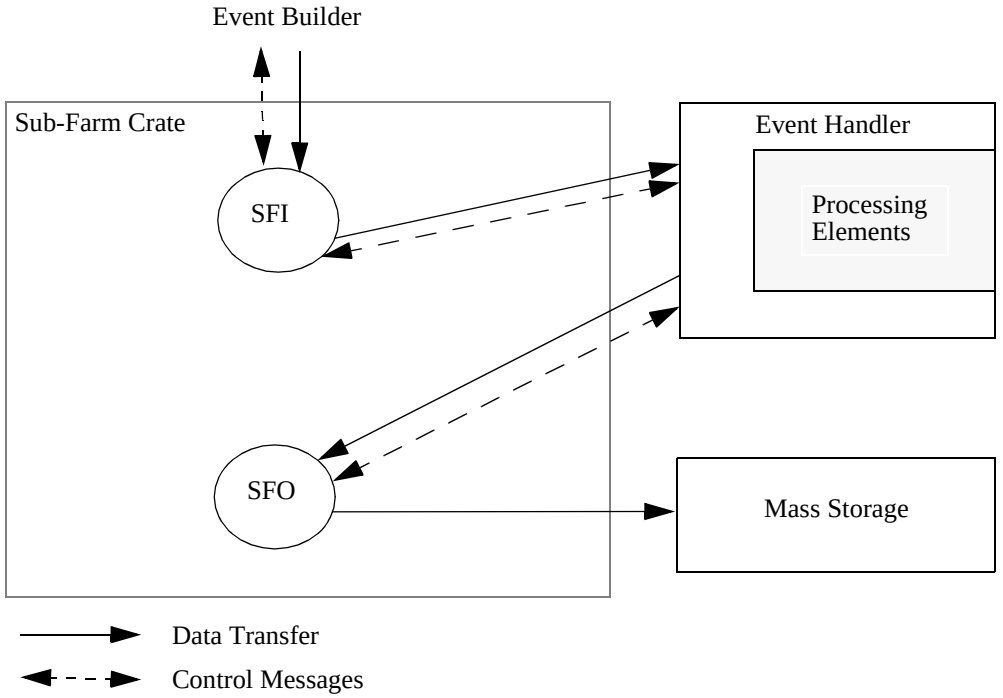

Figure 2. Logical model of the data flow in and out of the SFC.

1.3 The DAQ-Unit and the IOM model

The DAQ-Unit is responsible for the flow of event data in the ROC and SFC. The main component of the DAQ-Unit is the I/O Module (IOM). It receives data on one or more input channels, buffers the data and forwards data to one or more output channels. The data may be event data (e.g. ROB or crate fragments) or data control messages (e.g. messages from the Level 2 trigger system). An IOM is driven by “events” usually associated with an I/O channel. The software functionality associated to an I/O channel is called a *Task* and an IOM is, therefore, structured as a set of Tasks.

This task framework and other common software infrastructure is provided by the Generic IOM [2]. The main software components from the generic IOM used by the IOM instances are: the task scheduler[3], the event manager[4] and the message passing[5].

1.4 ROC and IOM configurations

The logical model shown in Figure 1 may be implemented on different configurations of Single Board Computers (SBC) in the ROC, as shown in Figure 3. The main parameters of each configuration are: number of SBCs, the intra-crate traffic and the number of (network) interfaces to the external systems. The intra-crate communication between the SBCs may be via buses and/or links; below this is referred to as the intra-crate "bus".

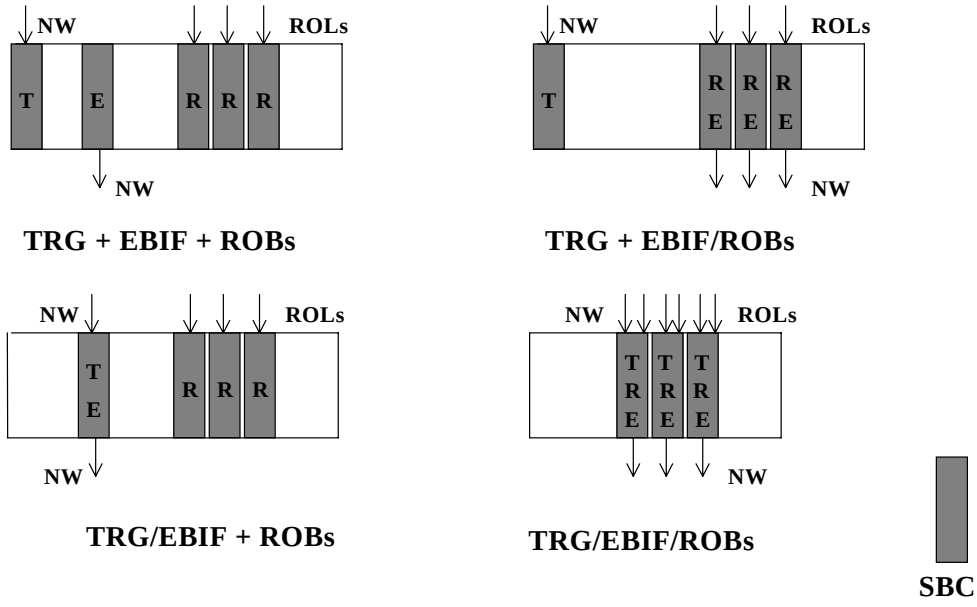

Figure 3. ROC IOM Single Board Computer configurations.

- the TRG, EBIF & ROBs on different SBCs: the "TRG + EBIF + ROBs" configuration.

  In this configuration the TRG receives all the trigger messages and the ROC has only one interface to the trigger systems. Data control messages are dispatched to the EBIF and the ROBs via the intra-crate communication. The EBIF performs the data collection over the intra-crate bus and provides the interface from the ROC to the EB. The ROBs have no direct interfaces to external trigger and event builder systems.

- the TRG and EBIF on one SBC and the ROBs on separate SBCs: the "TRG/EBIF + ROBs" configuration.

  The merged TRG/EBIF receives all trigger messages and, in addition, interfaces to the event builder or a Level 2 trigger system. In the latter case, all interaction with the Level 2 system is via the interface on the TRG/EBIF SBC and this configuration is, therefore, also referred to as the "LVL2 interface" configuration. The dispatching of the trigger messages and the data collection is via the intra-crate bus and, as in the previous case, the ROBs do not have any direct interfaces to the external trigger and event builder systems.

- the TRG on one SBC and the EBIF/ROBs on separate SBCs: the "TRG + EBIF/ROBs" configuration.

In this configuration, the EBIF is merged with the ROBs and implemented on a one SBC. Each EBIF/ROB SBC has an interface to the event builder system. As a consequence, the data collection traffic has been removed from the intra-crate bus. As in the first configuration, above, the TRG receives all the trigger messages and the ROC has only one interface to the trigger systems. Data control messages are dispatched to the EBIF/ROBs via the intra-crate communication path.

- the TRG, EBIF and ROB functions on one SBC: the “TRG/EBIF/ROBs” configuration.

  In this configuration, the TRG, EBIF and ROB functions are implemented on a single SBC which has interfaces to the trigger and event builder systems. Trigger messages are received and dispatched to the ROB and EBIF via buses on-board the SBC. Similarly, the data collection process takes place via on-board buses. In this configuration, the intra-crate bus is not involved in the main data flow (but could be used for control and monitoring).

1.5 SFC and IOM configurations

The logical model shown in Figure 2 may be implemented on different configurations of SBC in the SFC. In the absence of an Event Filter Sub-Farm, the Event Handler may be implemented as a Task inside the SFO, thus emulating a *real external Event Handler*. The SFI and SFO can run on different SBCs in the SFC or they can be implemented as one IOM on one SBC only.

- the SFI & SFO on different SBCs: the “SFI + SFO” configuration.

  In this configuration the SFI receives all the ROC fragment data and control messages from the Event Builder (optionally local event generation is possible) and sends the full event data to an external Event Handler running on a workstation outside the SFC crate. The SFO receives events from the Event Handler and sends them to the Mass Storage system for recording. The communication with the Event Handler uses the Event Handler API [6] which was implemented using ILU/CORBA [8].

- the SFI and SFO on one SBC: the “SFI/SFO” configuration.

  The merged SFI/SFO receives all the ROC fragment data and control messages from the Event Builder (optionally local event generation is possible) and sends the complete event to the SFO task within the same IOM. The communication uses the Event Handler API, which was implemented using system memory and memcpy calls, via the software fifo package[9]. It is obvious that in this configuration no processing of the event is possible.

2 Measurements

The performance measurements presented in this note are based on VMEbus SBCs, more specifically the CES RIO8062 [11]. Input from external trigger systems and output to the EB has been emulated. For input to the ROBs, events were generated either locally, by the ROB host CPU, or remotely via ROB-INs. The latter were based on CES MFCC PMC modules [12]. In both cases the event data was emulated (i.e. no SLINK input). The measurements are presented in the form of number of events “flowing” through the ROC per second. This depends on a number of parameters which have been fixed to the following values: event fragment size = 1kbyte, #L2Rs per message = 10 (grouping) and the ratios between data control messages L2R:ROI:L2A = 100:10:1. The page size was 1 kByte, the number of pages

equal to 200 and the number of classes 16 [4]. Communication between the SBCs is via VMEbus, only (PVIC was not used).

3 The Read-Out Crate Implementations

3.1 The baseline configuration: TRG + EBIF + ROBs.

3.1.1 Task Structure

The implementation of the baseline ROC configuration is shown schematically in Figure 4. A high level description of the tasks in the IOMs is given in [1]. Below, a description of the tasks specific to the baseline configuration is presented.

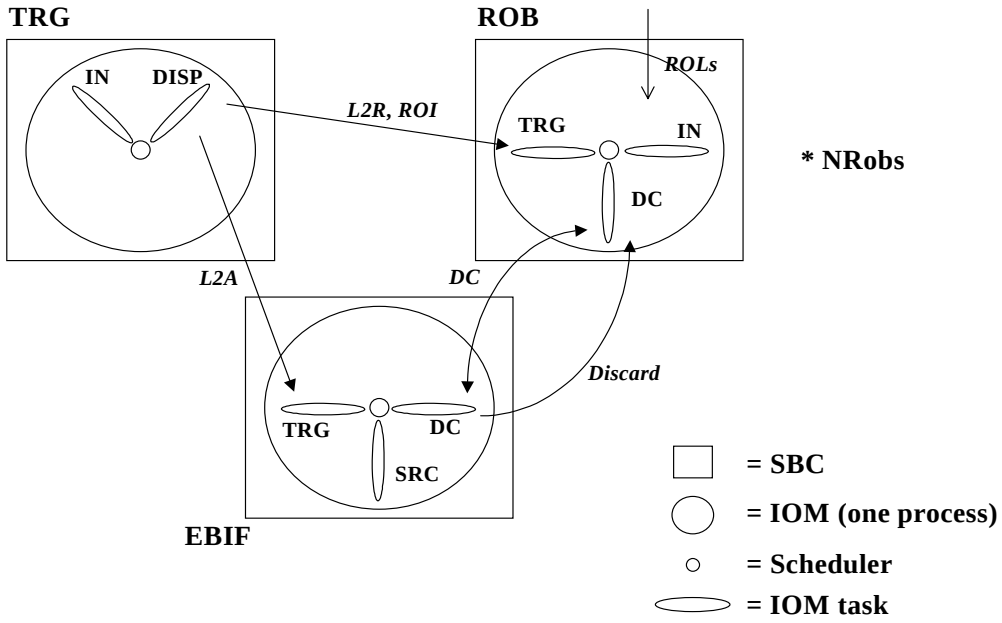

Figure 4. ROC implementation in TRG + EBIF + ROBs configuration.

The TRG functionality is implemented as an IOM on a SBC and has the following task structure:

- Input (IN): This task inputs and buffers the data control messages from the trigger systems. In emulation mode it generates the trigger messages internally. When the message buffer is full, a request for processing is sent to the DISP task, via a Software FIFO [9].
- Dispatcher (DISP): Upon reception of a processing request, this task processes the data control messages placed in the trigger message buffer by the IN task. L2A and ROI messages are sent directly to the EBIF and ROBs, resp. while the L2R messages are stacked and sent as one message when the stack is full. Messages are sent over VMEbus or PVIC using the message passing library.

The EBIF has the following task structure:

- Trigger task (TRG): This task receives L2A messages from the TRG IOM. The event identifier is extracted from the message and sent to the DC task via a software FIFO.
- Data collection task (DC): This task performs the data collection using a message based protocol. When an event ID is received from the TRG task, contiguous memory space for a crate fragment is allocated using the Event Manager [4]. A message is sent to all ROBs requesting those to retrieve information about the event fragments for the given event ID. Each ROB returns pointers and fragment lengths to the EBIF via a reply message. When all ROBs have replied, the EBIF collects the fragments via a chained block transfer across VMEbus into the contiguous memory area. The crate fragment header is constructed and a crate fragment is created, using the Event Manager. A discard message is then sent to the ROBs to inform those to remove the event fragments just being collected (actually, the discard message is piggy-backed onto the next request message in order to save a message transfer). Finally, the ID of the collected event is sent to the SRC task via a software FIFO.
- The Source task (SRC): upon reception of an event ID from the DC task, the SRC outputs the crate fragment to the event builder system. In the case of output emulation, the SRC task just deletes the event. For a more complete description of the SRC task structure, see [10].

The ROB has the following task structure:

- Input (IN): This task receives and buffers the ROD fragments from a ROL. Memory space for an event fragment is allocated, data transferred into that space under control of the task and an event created [4]. In emulation mode, the data is generated internally.
- Trigger task (TRG): This task receives and processes the L2R and ROI messages from the TRG. For L2R messages, the events defined in the received message (grouping) are removed from the ROB buffer. The ROI messages are discarded: this implementation of the ROB has no support for interfacing to the Level 2 trigger.
- Data Collection task (DC): This task implements the ROB part of the Data Collection process in the ROC as already described for the EBIF, above. When a data collection request is received, information about the event defined in the message is extracted: VMEbus addresses of the event fragments and their lengths. The ROB header is built and the fragment information returned to the EBIF. In addition, this task removes events which have been collected by the EBIF. The discard message is received together with the request message for the following event to be collected.

The description of the IN task of the ROB applies to the case where event fragments are read from the ROL under control of the ROB CPU. However, the ROB model also allows for the possibility of ‘remote’ event sources where the event fragments are input under control of other processors i.e. *ROB-IN*’s. Whether the event sources are ‘local’ or ‘remote’ is largely hidden to the ROB application which sees the event sources via the event manager [4]. Switching between the local and the remote event manager is performed by linking different libraries (with the same API) into the application.

3.1.2 Results

The results for the baseline configuration are shown in Table 2 and plotted in Figure 5 (which, in addition, contains results from measurements in other configurations). The event rate with local event generation is determined mainly by the VMEbus traffic which explains the large difference between the results for one and two ROBs (no broadcast on VMEbus). For remote event generation, and with one ROB, the rate is about 115 kHz for one and two ROB-IN’s. In

this case the limiting factor is the software in the ROB-IN where the event overhead is about 8 μs [14]. With three and four MFCCs the rate drops to about 90 kHz which is due to the increased load on PCI bus.

Event Source	Event Rate (kHz)
One ROB, local event generation	*168*
Two ROBs, local event generation	*98*
One ROB, one ROB-IN's	*116*
One ROB, two ROB-IN's	*114*
One ROB, three ROB-IN's	*102*
One ROB, four ROB-IN's	*89*

Table 2. Event rates in the configuration TRG + EBIF + ROBs.

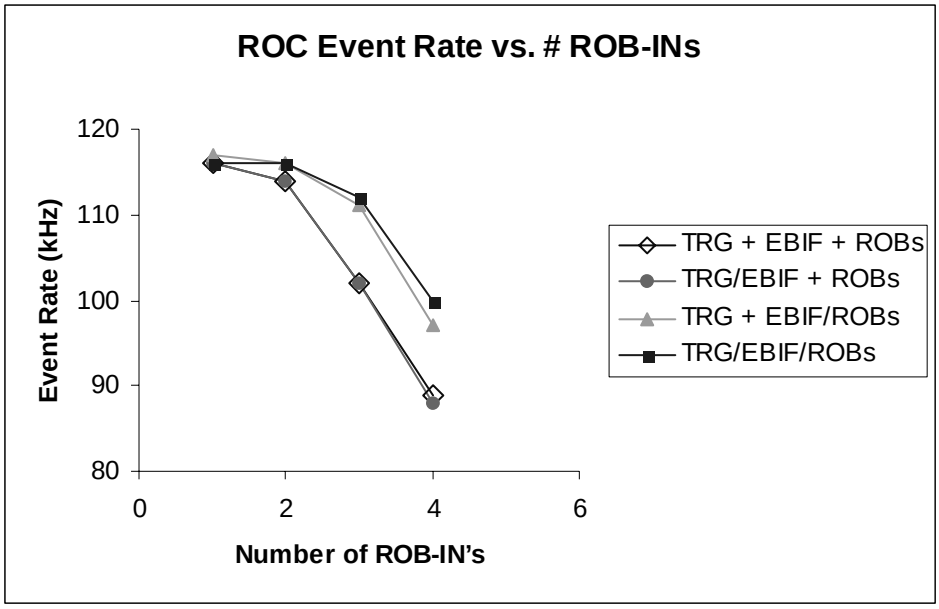

Figure 5. Event rate in the ROC in SBC configurations including one ROB with up to four MFCCs.

3.2 TRG/EBIF + ROBs

3.2.1 Task Structure

This section describes the ROC implementation of the TRG/EBIF + ROBs IOM configuration, see Section 1.4, in which all interaction between the ROC and the trigger and event builder systems are via the TRG/EBIF. Compared to the baseline configuration only minor changes to the tasks have been performed, see Figure 6.

The IN and the DISP tasks have been integrated in the EBIF. The L2A is sent from the DISP task to the TRG task using internal message passing [5], minimising the code changes in the DISP and TRG tasks. The data collection is unchanged compared to the baseline configuration.

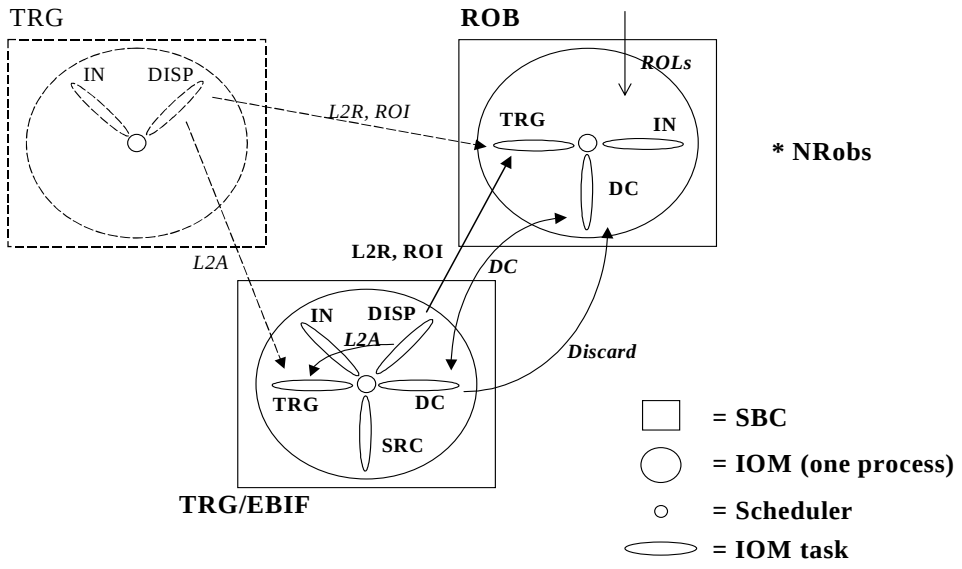

Figure 6. ROC implementation in TRG/EBIF + ROBs configuration.

3.2.2 Results

A set of measurements were performed on the TRG/EBIF & ROBs configuration similar to those described in Section 3.1.2. The results are shown in Table 3 and Figure 5. In the case of local event generation, the event rates are lower by about 10% compared to the numbers in Table 2. This is expected since the TRG and EBIF functions are now merged on a single SBC while the traffic on VMEbus is almost the same as in the baseline configuration (only the L2A messages are internal). For remote event generation, the rates are the same as those in the baseline configuration.

Event Source	Event Rate (kHz)
One ROB, local event generation	*153*
Two ROBs, local event generation	*88*
One ROB, one ROB-IN's	*116*
One ROB, two ROB-IN's	*114*
One ROB, three ROB-IN's	*102*
One ROB, four ROB-IN's	*88*

Table 3. Event rates in the configuration TRG/EBIF + ROBs.

3.3 TRG + EBIF/ROB

3.3.1 Task Structure

This section describes the ROC implementation of the TRG + EBIF/ROB IOM configuration see Section 1.4. In this configuration where there is one interface to the event builder per ROB, the data collection process is local to the ROB and therefore has to be modified. In the EBIF/ROB, when a L2A message is received, information about the accepted event is retrieved and made available to the event builder interface task allowing this to transmit the event fragment to the EB. If the network interface software is capable of directly accessing memory areas in user space, the information consists of event addresses and lengths of the event fragments in the ROB memory space. If the network interface software requires the event data to be in a contiguous memory area, the event fragments must be copied before being transmitted to the EB.

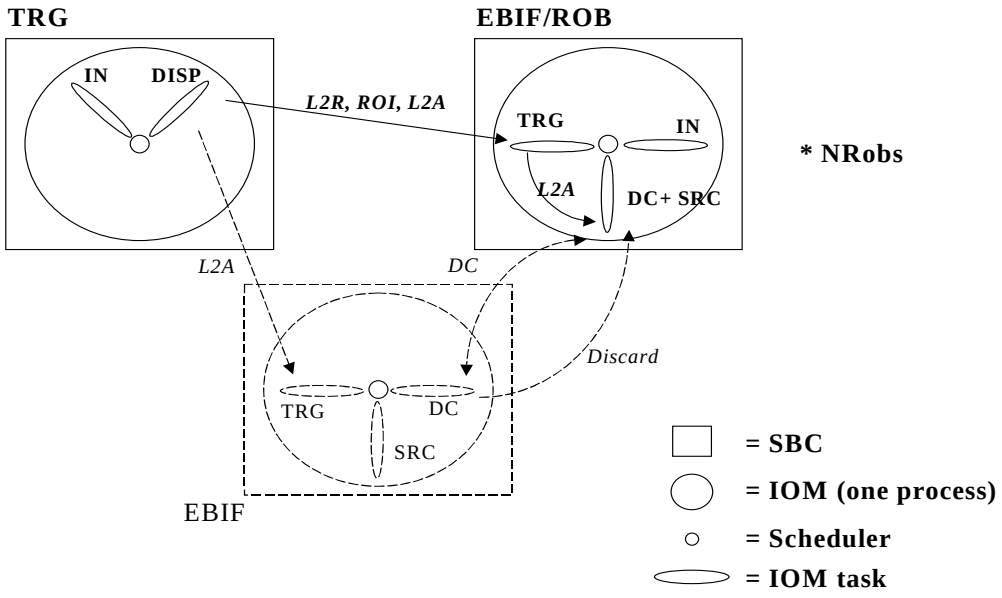

Figure 7. ROC implementation in TRG + EBIF/ROBs configuration.

Compared to the baseline configuration, the following changes to the tasks have been made, see Figure 7:

- the TRG task from the EBIF has been merged with the TRG task in the ROB. The L2A messages from the DISP task in the TRG are sent to the TRG task in the EBIF/ROB on the same channel as the L2R and the ROI messages. The L2A messages are received, the event ID extracted and forwarded to the DC+SRC task via a software FIFO.
- the DC and SRC tasks have been merged. When the L2A event ID is read from the SW FIFO, the corresponding event fragments are copied, using the event manager, into a contiguous memory area. Since the EB output is emulated, the event is then deleted.

3.3.2 Results

The measurements on the TRG & EBIF/ROBs configuration are similar to those described in Section 3.1.2 except that there is only one ROB. The results are shown in Table 4. In the case of local event generation, the event rates are a little higher (about 5%) compared to the numbers in Table 2: the data collection process and the associated data flow is now internal to the EBIF/ROB. For remote event generation, the rates for one and two MFCCs are as before and determined by software at the level of the ROB-IN's. For three and four MFCCs the decrease in event rate is less than in the baseline and TRG/EBIF + ROBs configurations. The PCI bus is in this case less loaded since the ROB fragments are not sent over VMEbus to the EBIF.

Event Source	Event Rate (kHz)
One ROB, local event generation	*179*
One ROB, one ROB-IN's	*116*
One ROB, two ROB-IN's	*114*
One ROB, three ROB-IN's	*111*
One ROB, four ROB-IN's	*97*

Table 4. Event rates in the configuration TRG & EBIF/ROBs.

3.4 TRG/EBIF/ROB

3.4.1 Task Structure

This section describes the ROC implementation of the TRG/EBIF/ROB IOM configuration see Section 1.4. The TRG, EBIF and ROB functions are collapsed on a single SBC which has interfaces to the detector, trigger and event builder systems. Compared with the TRG + EBIF/ROB configuration in the previous chapter, the following changes to the task structure have been made, see Figure 8:

- the IN and DISP tasks of the TRG have been integrated into the EBIF/ROB. The data control messages are dispatched to the TRG and DC/SRC task via internal message passing.

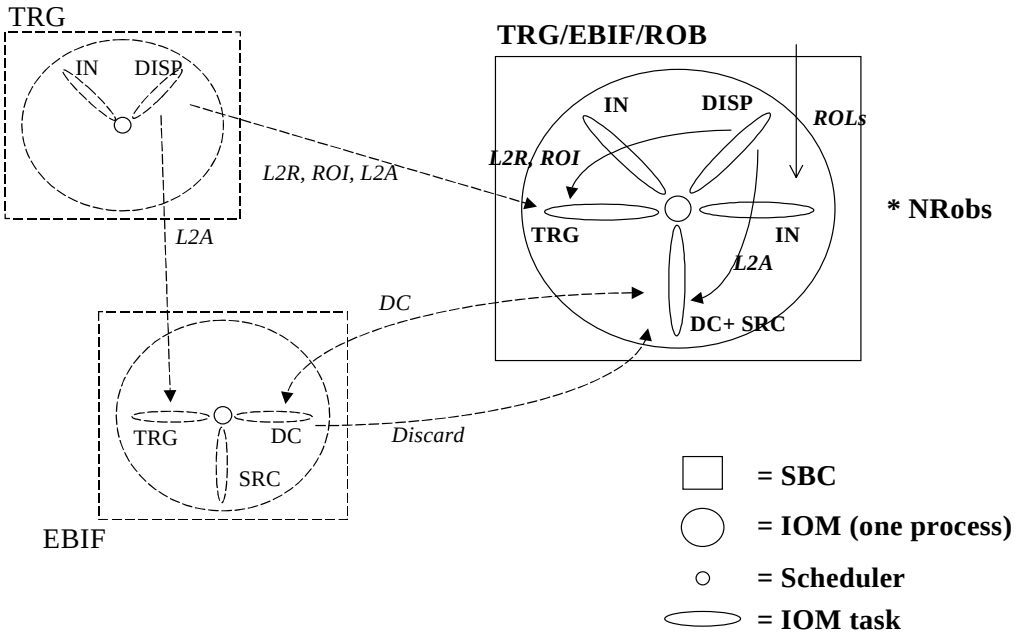

Figure 8. ROC implementation in TRG/EBIF/ROBs configuration.

3.4.2 Results

The measurements on the TRG/EBIF/ROBs configuration have been performed with parameters similar to those in Section 3.3.2. The results are shown in Table 5. In the case of local event generation, the event rates are lower (about 10%) compared to the numbers in Table 2. The trigger message and event fragment traffic is now eliminated from VMEbus but on the other hand all the TRG, EBIF and ROB functions are merged on one SBC. For remote event generation, the rates are as in the TRG + EBIF/ROBs configuration, see Section 3.3.2.

Event Source	Event Rate (kHz)
Local event generation	*153*
One ROB-IN	*116*
Two ROB-IN's	*116*
Three ROB-IN's	*112*
Four ROB-IN's	*100*

Table 5. Event rates in the configuration TRG/EBIF/ROBs.

4 The Implementation of the Sub-Farm Crate

4.1 SFI + SFO.

4.1.1 Task Structure

The implementation of the SFI + SFO configuration of the SFC is shown schematically in Figure 9. A high level description of the tasks in the IOMs is given in [1]. Below, a description of the tasks specific to the baseline configuration is presented.

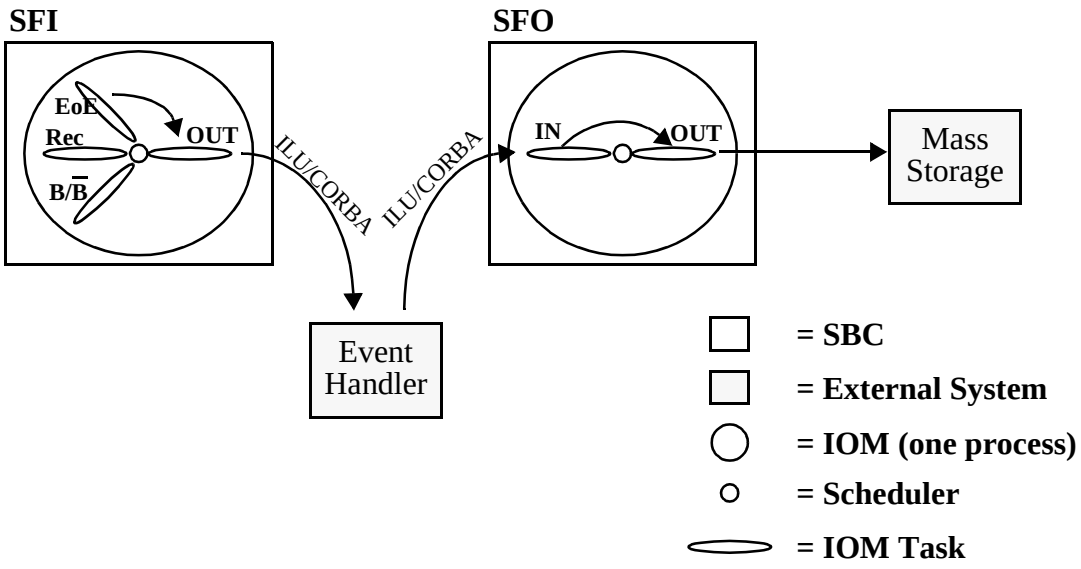

Figure 9. SFC implementation in SFI + SFO configuration.

The SFI functionality is implemented as an IOM on a SBC and has the following task structure:

- Input from the EB system (Rec; EoE; B/$\overline{B}$): These Tasks interface with the Event Building system. They receive and buffer ROC fragments (Rec Task), manage the event completion (EoE Task) and handle the busy/non-busy mechanism (B/$\overline{B}$ Task). The Event Manager is used to allocate the memory space needed for the buffering of the event data. In emulation mode ROC fragments are generated internally. When an event is completed, a request for processing is sent to the Out Task, via a software fifo [9].
- Output (OUT: Upon reception of a processing request, this Task sends complete events to the Event Handler. Buffer space associated with the sent event is deleted immediately after a successful send. The current implementation of this Task needs to copy the complete event into contiguous memory before the event can be sent out.

The SFO has the following task structure:

- Input (IN): This Task receives complete events, possibly reconstructed and filtered, from the Event Handler. In emulation mode it generates complete events internally. When a complete event is received, a request for processing is sent to the OUT Task via a software fifo.

- Output (OUT): Upon reception of a processing request, this Task sends complete events to Mass Storage. Memory space associated with the sent event is deleted immediately afterwards.

4.1.2 Results

Input from the EB has been emulated locally by the SFI; two ROC fragments, from different subdetectors, were generated before the complete event was built. The complete event was then sent to the Event Handler running on a remote workstation. The transport media was standard ETHERNET used non-exclusively and the protocol used to implement the Event Handler API was ILU/CORBA. The output to Mass Storage was emulated which means that the output Task upon the reception of a process request just released the buffer space associated with the current event. The aim of these tests were to prove the feasibility and robustness of the proposed scheme and less to achieve a high performance.

The results for the SFI + SFO configuration with four distinct and geographically dispersed Event Handlers are shown in Table 6:.

Event Handler	Event Rate	Throughput
Solaris Workstation on local area network at CERN	*125 Hz*	*65 kByte/s*
Marseille Event Handler in Marseille (France)	*15 Hz*	*4.6 kByte/s*
Pavia Event Handler in Pavia (Italy)	*11 Hz*	*3.3 kByte/s*
THOR Event Handler in Alberta (Canada)	*2 Hz*	*200 Byte/s*

Table 6: Event rates in the configuration SFI + SFO.

The event rates achieved are of course biased by the Wide Area Network traffic.

4.2 SFI/SFO.

4.2.1 Task Structure

The implementation of the SFI/SFO configuration of the SFC is shown schematically in Figure 10.

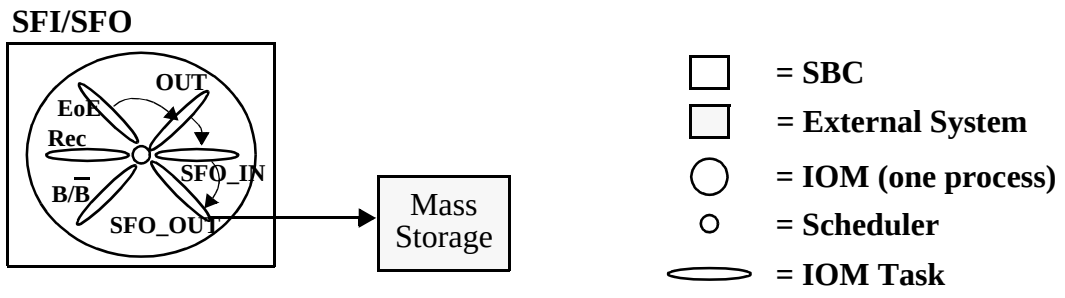

Figure 10. SFC implementation in SFI/SFO configuration.

Both the SFI functionality and the SFO functionality are implemented as an IOM on a SBC and have the following task structure:

- Input from the EB system (Rec; EoE; B/$\overline{B}$): These Tasks interface with the Event Building system. They receive and buffer ROC fragments (Rec Task), manage the event completion (EoE Task) and handle the busy non busy mechanism (B/$\overline{B}$ Task). The Event Manager is used to allocate the, for the buffering of the event data, needed memory space. In emulation mode ROC fragments are generated internally. When an event is completed, a request for processing is sent to the Out Task, via a Software FIFO.
- SFI output (OUT): Upon reception of a processing request, this Task sends complete events to the Event Handler. Buffer space associated with the sent event is deleted immediately after a successful send. The current implementation of this Task needs to copy the complete event into contiguous memory before the event can be sent out. In this configuration, sending to the Event Handler is implemented as another copy of the event into a software fifo. The internals of the software fifo used here are completely hidden from this Task via the use of a special implementation of the Event Handler API.
- SFO input (SFO_IN): This Task retrieves and buffers the event from the fore mentioned software fifo again via the use of the Event Handler API. A request for processing is sent to the SFO output Task via a software fifo.
- SFO output (SFO_OUT): Upon request for processing, this task sends the event data to the Mass Storage system for recording. Memory space associated with the sent event is then freed immediately afterwards.

4.2.2 Results

Input from the EB has been emulated locally by the SFI/SFO; two ROC fragments from different subdetectors, were generated before the complete event was built. The complete event was sent to the Event Handler which means nothing else than a copy into a software fifo from which it had to be copied out for becoming accessible in the SFO_IN Task. The SFO_OUT Task, responsible for storing the event was not activated for these measurements and therefore, no output to a Mass Storage system was done.

In this configuration, an event rate within the SFC crate of 1.5 kHz was measured.

5 Summary and Conclusions

The models of the Read-Out Crate, the Sub-Farm Crate and the DAQ-Unit were briefly reviewed. The main components of the DAQ-Unit are Tasks and I/O Modules. Common IOM software functionality is provided by the Generic IOM, mainly Message Passing, Event Management and Task Scheduling. Implementations of the ROC and the SFC model on different configurations of VMEbus SBCs were described with emphasis on the task structure of the IOM instances. For each configuration, a set of measurements were performed with event fragments generated locally (ROC and SFC) and via MFCC based ROB-IN's (ROC). For the ROC the trigger messages were generated in the ratio L2A:ROI:L2R = 1:10:100 i.e. a Level 2 accept proportion of 1%. External I/O to trigger and event builder systems were emulated. In particular it should be noted that no network interfaces were implemented. The SFC didn't need a trigger and generated ROC fragments as fast as possible.

The conclusions are the following:

- based on the DAQ-Unit model and the Generic IOM, the Read-Out Crate and the Sub-Farm Crate have been implemented on a number of different SBC configurations. The I/O Module variants are built by combining a set of basic tasks in different ways.
- the baseline configuration with IOMs on different SBCs was readily ported to other, “collapsed”, configurations. Starting from a configuration where the functions TRG, EBIF, ROB, SFI and SFO were implemented on different SBCs helped in defining the basic tasks.
- the Message Passing allows tasks to communicate via channels which may external or internal (system memory) to an SBC. This has enabled to move the basic tasks between different SBCs with minimal changes to the software.
- examples of measurements with local event generation illustrate the trade-off between CPU power and inter-processor communication. In the baseline configuration TRG+EBIF+ROBs the performance is determined by the intra-crate traffic over VMEbus while the fully “collapsed” configuration TRG/EBIF/ROBs is limited by the performance of the (single) CPU. Large differences in performance between the different configurations were not observed. The relative performances will obviously change with other characteristics of the intra-crate bus and the CPUs and also with other trigger message parameters.
- examples of measurements with remote event generation show that, with one or two ROB-INs’s and a L2A/L2R ratio of 1%, the ROC event rate is mainly determined by the software overhead of the MFCC based ROB-IN. With more ROB-IN’s the PCI bus becomes the limiting factor. For more details on multi ROB-IN measurements, see [14].

References

[1] The Main Data Flow in the Read-Out Crate of ATLAS DAQ prototype -1. http://atd-doc.cern.ch/Atlas/Notes/047/Note047-1.html

[2] The generic I/O module in ATLAS DAQ prototype -1. http://atd-doc.cern.ch/Atlas/Notes/041/Note041-1.html

[3] Event Scheduling in the I/O Module, http://atd-doc.cern.ch/Atlas/Notes/018/Note018-1.html

[4] The Event Manager API in ATLAS DAQ/EF prototype -1, http://atd-doc.cern.ch/Atlas/Notes/071/Note071-1.html

[5] The DAQ-Unit Message Passing API, http://atd-doc.cern.ch/Atlas/Notes/091/Note091-1.html

[6] The Event Handler API, http://atddoc.cern.ch/Atlas/Notes/098/Note098-1.html

[7] ILU is manufactured by PARC Xerox Co., ftp://ftp.parc.xerox.com/pub/ilu/ilu.html

[8] OMG, CORBA, http://www.omg.org/corba/

[9] Software FIFOs for the IOM modules in ATLAS DAQ/EF prototype -1, http://atd-doc.cern.ch/Atlas/Notes/077/Note077-1.html

[10] A high-level design for the Event Building Source and Destination in the ATLAS DAQ Prototype, http://atddoc.cern.ch/Atlas/Notes/072/Note072-1.html

[11] RIO8062 User's Manual, http://www.ces.ch/MainProducts.html

[12] MFCC User's Manual, http://www.ces.ch/MainProducts.html

[13] PVIC User's Manual, http://www.ces.ch/MainProducts.html

[14] ROB Summary Milestone