The Prototype of a VME-Based Front-End System for the T-C Factory Control System

F. Zhang J. Zhao
Institute of High Energy Physics,Chinese Academy of Sciences
P.O.Box 918(10) Beijing 100039,China

Abstract

The VME-based prototype is aimed at researching some basic technical matters in the front-end system of the T-C factory control system. Hardware and software aspects of the prototype system will be described in this paper. In particular, we will discuss the embedded system generator: *vmexgen*, network download, multiprocessor operation, VMEbus address space, interrupt service routines(ISR), I/O device driver and VMEexec Network Environment (VNE). Finally, a VMEbus-based parallel processing system will be discussed.

1 Introduction

The Beijing Tau-Charm Factory(BTCF) is a new electron-positron collider project which has a luminosity which is a hundred times higher than that of BEPC. Its beam energy is 1.5 to 3 GeV. The BTCF will serve high energy physical experiments. Its front-end system consists of FECs(Front-End Computers), I/O devices and real-time applications running on the FECs. The FECs connected with Ethernet or VMEbus access the I/O devices via the VMEbus or VMEbus-field bus interfaces. The basic technical matters in the front-end system of the T-C factory control system will be described in the following paragraphs.

2 Structure of the prototype system

2.1 Hardware

In the prototype system shown in figure 1, a MVME187 host, two MVME162 targets and some I/O boards are assembled in one VME chassis. Both, the MVME187 and the MVME162 have on-board Ethernet interfaces. Because they can be assigned to different logical chassis in the VMEexec software package even though they locate in the same chassis, the prototype system can support both, VMEbus-based and network-based configurations.

2.2 Software

The software of the prototype consists of UNIX SYSTEM V and the Motorola VMEexec software package which is a real-time system development environment based upon a host running on a VMEmodule-driven SYSTEM V operating system. It provides a comprehensive set of board support and an I/O driver package for Motorola`s line of VME boards. The Application Program Interface (API) of VMEexec is shown in figure 2.

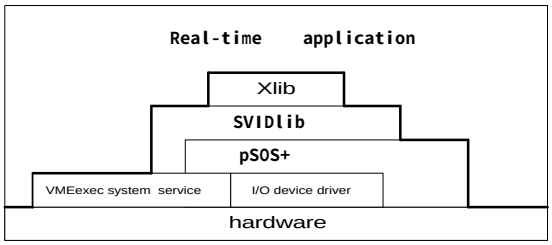

Figure 2. The application program interface of VMEexec

3 The basic technical matters

3.1 vmexgen

Vmexgen is the process used to select and make modifications to VMEexec configurations, define the structure of the system, create VMEexec target kernels based on the selected configurations. The kernels generated by *vmexgen* will be the run-time environment of the corresponding real-time application. Each of the following technical aspects has its own *vmexgen* configuration.

3.2 Network download

Network download refers to VMEexec kernels and real-time applications downloading from host to target over Ethernet. The network downloads usually fall into two general categories: active and passive.

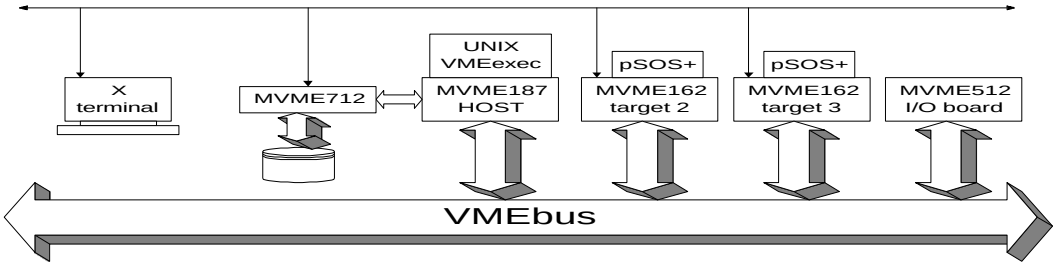

Figure 1. Structure of the prototype system

In the case of active downloads, the targets initiate the network download process after the targets' reset or power-up. With the network download information which is predefined in the targets' NVRAM, the on-board Ethernet interface and the tftp function of the on-board ROM, the targets download their real-time kernels and applications, then start the real-time applications.

In the case of passive downloads, the targets boot from the local ROM and start a network download assistance process. With this process, the host command *gload* can be used to download the corresponding real-time kernel and application to the targets.

The network download assistance process can be downloaded through the active way, or be written in EPROM and inserted in the socket of the target.

If there is no Ethernet interface on a target, its kernel and application can be indirectly downloaded with the assistant of another target which stays in the same VME chassis and has an Ethernet interface. It must be noted that only the passive style can be used in this condition, because the network download assistance process can provide the indirect download service, but the tftp function can`t.

3.3 Multiprocessor operations

"Multiprocessor operations" means that tasks that make up an application can reside on several targets and still exchange data and synchronize execution as if they were running on a single processor. The pSOS+ defines a series of global objects which can be reached or referenced from any target in the system. The global object classes include task, event, memory partition, message queue and semaphore. Using global objects, pSOS+ provides a mechanism of parallel processing based on multiprocessors. For example:

- Multitask scheduling based on multiprocessors was realized using a global task.
- Multitask synchronizing based on multiprocessors was realized using a global event and/or semaphore.
- Exchanging data between tasks runing on different processors was realized using a global message queue.
- Dual-ported memory was realized in VMEbus-based systems using global memory partitions which is only meaningful in tightly(i.e. , memory) coupled systems and global message queues.

Because VMEexec provides a software layer hiding the communication media features, the implementation of multiprocessor operations based on VME and Ethernet have the same synopsis.

3.4 VMEbus address space, Interrupt Service Routines (ISR) and I/O device driver

Because the VMEbus has address lines A00-A31, it's address space ranges is from $0 to $FFFFFFFF. In VMEbus-based multiprocessing systems, each board, as a VMEbus slave, occupies a section of the VMEbus address space. The board with the master functional module can access the VMEbus address space[5]. The VMEbus address space of the prototype system is shown in figure 3. The host`s VMEbus address space access is forbidden. Because each MVME162 can be the VMEbus master and/or slave, their 4MB shared DRAMs are accessible for each other, which is the hardware environment of global memory partition operations. MVME512 is a VMEbus slave which occupies 1KB section of the VMEbus address space. Targets can control actions of the MVME512 and read the A/D data by accessing the 1KB VMEbus address space.

Interrupt Service Routines(ISR) are critical to any real-time system. Two interrupts are generated and handled in the prototype system. One is the VMEchip2 Tick Time 1 interrupt which is generated and handled by the MVME162. The other is the VMEchip2 VMEbus IRQ3 interrupt which is generated by the MVME512 and handled by the MVME162 through the VMEbus IRQ3 line. We use the VMEexec system service *claimvct* to declare the entry address and another parameters of the ISRs in both ISR implementations.

A device driver handles requests made by the pSOS+ kernel with regard to a particular type of device. The pSOS+ furnishes a device-independent, standard methord for integrating drivers into the system and for the user`s application to call these drivers. We have realized two drivers, one for the A/D board MVME512, another for the digital board MVME510, and integrated them into the the system. By isolating device-specific code in a device driver and by having a consistent interface to a kernel and application, adding a new device and performing I/O operations is easier. The I/O device driver is the base of the control software in large scale real-time systems.

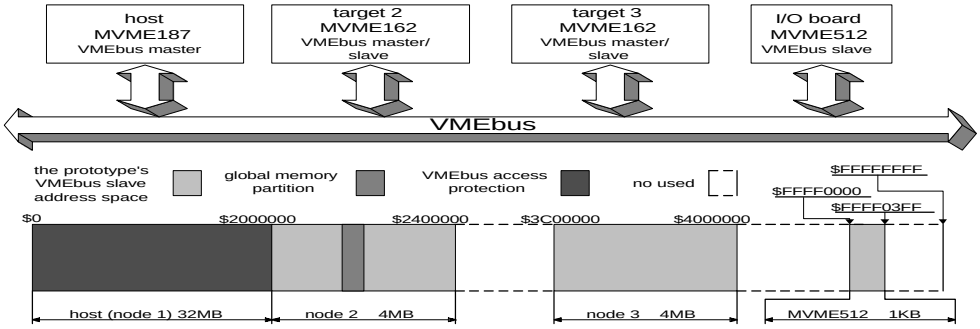

Figure 3. The VMEbus address space of the prototype system

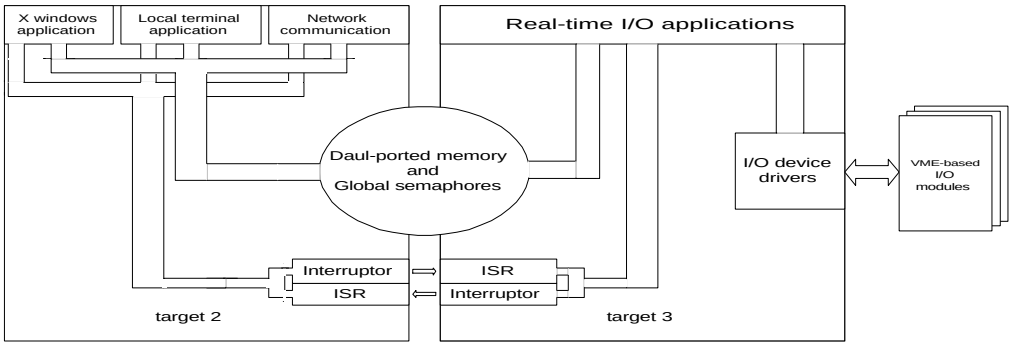

Figure 4. The software architecture of the VMEbus-based parallel processing system

3.6 VMEexec network environment(VNE)

The VNE server task provides a socket interface for the communication between VMEexec targets and computers which run different operating systems with the TCP/IP protocol. We ran a pair of sending and recieving applications in two conditions. In both conditions, the UNIX socket interface is used by the host`s receiving application.

- In the first condition, the sending application and the VNE server task just reside in target 2.
- In the second condition, the VNE server task resides on target 2 and the sending application resides on target 3. The VNE socket interface used by target 3 transparently used a Remote Procedure Call (RPC) mechanism to communicate its requests to the VNE server residing on target 2 via the VMEbus.

4 The VMEbus-based parallel processing system

The VMEbus-based parallel processing system includes two MVME162 modules and several high performance I/O modules. All these modules are assembled in one VME chassis. The tasks of the system include the X Windows man-machine interface, local character terminal applications, network communications, and real-time I/O operations.

Because the system includes several high performance I/O modules, such as the MVME512 with a A/D conversion time of 33μs, and because the system is required to fully use the performance of the I/O modules, the first MVME162 runing the real-time I/O operations bears a heavy load. The second MVME162 must be added to share other tasks. The software architecture of the system is shown in figure 4.

Target 2 runs X Windows, the local terminal and network communication applications. Target 3 performs real-time I/O operations through I/O device drivers. Two targets exchange data and sychronize their operations through daul-ported memories and global semaphores. The interruptors and the ISRs reside on both targets to accomplish some critical operations related with the two targets. In this VMEbus-based parallel processing system, the performance of the real-time embedded applications is scaled beyond the capabilities of a single CPU.

5 Current status

With the prototype system, the network download, multiprocessor operations, ISRs, I/O device driver and network communications through the VNE socket interface have been realized. In the R&D stage of the BTCF, a SUN workstation, the VxWorks operating system and the EPICS software tool kits will be installed in our system to develop the control system of BTCF.

References

[1] VMEexec User`s Guide," by MOTOROLA Computer Group.
[2] pSOS+/68K Real-time Executive User`s Guide," by MOTOROLA Computer Group.
[3] VMEexec System Services User`s Guild," by MOTOROLA Computer Group.
[4] VMEexec Networking Environment User`s Guild," by MOTOROLA Computer Group.
[5] THE VMEbus SPECIFICATION, by VMEbus International Trade Association.
[6] MVME162 Embedded Controller Programmer`s Reference Guild," by MOTOROLA Computer Group.
[7] MVME512-003 and MVME512-004 User`s Manual, by MOTOROLA Computer Group.