Development of Device Drivers on Real-time UNIX for the SPring-8 Storage Ring Control System

T. Masuda, S. Fujiwara, T. Fukui, A. Taketani,
R. Tanaka, T. Wada, W. Xu, and A. Yamashita

SPring-8, Kamigori, Ako-gun, Hyogo 678-12, Japan

The HP-RT device drivers and API functions have been developed for VME boards such as DI, DO, TTL DI/O, AI, PTG, GP-IB interface, and RIO (Remote I/O). The drivers access the devices through the SWSM (System Wide Shared Memory) mechanism. Design concepts of the APIs were to provide a simple interface, uniform types of function and arguments and method of handling. The APIs are the only way to access and control the devices from application programs.

1. Introduction

A distributed computing system has been adopted for the SPring-8 storage ring control system [1]. We use several workstations for operator consoles and 28 VMEbus computer systems with HP-RT around the storage ring as the front-end controllers. Each VME system is connected by an optical fiber to a 100Mbps FDDI backbone LAN through a FDDI-Ethernet switching HUB. Hence, each VME system can use the full bandwidth of Ethernet (10Mbps) for communication with upper-level computers. The optical-linked RIO system is mainly used as the field-bus for the control of magnet power supplies, vacuum system, and beam position monitoring system [2].

2. VME system

To consider expandability and maintainability, we have adopted industry-standard hardware as far as possible. Thus we have adopted a VMEbus computer system as the front-end controller. Because of the large number of manufacturers making equipment to this standard, we can take advantage of the various single board computers, I/O boards and other bus/network interface boards available.

The equipment is directly controlled by I/O modules in the VME system or through the RIO or the GPIB field bus as shown in Fig.1.

2.1 CPU board

The CPU board is a HP9000/743rt/64 which has a 32-bit PA-RISC 7100LC with a 64MHz clock. This CPU board is very powerful and performance is quite satisfactory (77.7MIPS) as a front-end controller. It has an AUI interface port, two RS-232C ports, an HP-parallel interface port and a single-ended SCSI II interface on its front panel, and a PCMCIA interface on the mother board. A maximum of 128MB main memory can be fitted on a single-slot board. In our case, 16MB of on-board memory and a 20MB PCMCIA flash ROM card are attached. The flash ROM card is used as a boot device for the operating system (OS).

2.2 I/O boards

We use several types of I/O boards, such as a 64-bit digital input (DI) board, a 64-bit digital output (DO) board, a 96-bit TTL level digital input/output (TTL I/O) board, a 32-channel single-ended/16-channel differential analog input (AI) board and a 5-axis control pulse-train generator (PTG) board, mainly for RF equipment control.

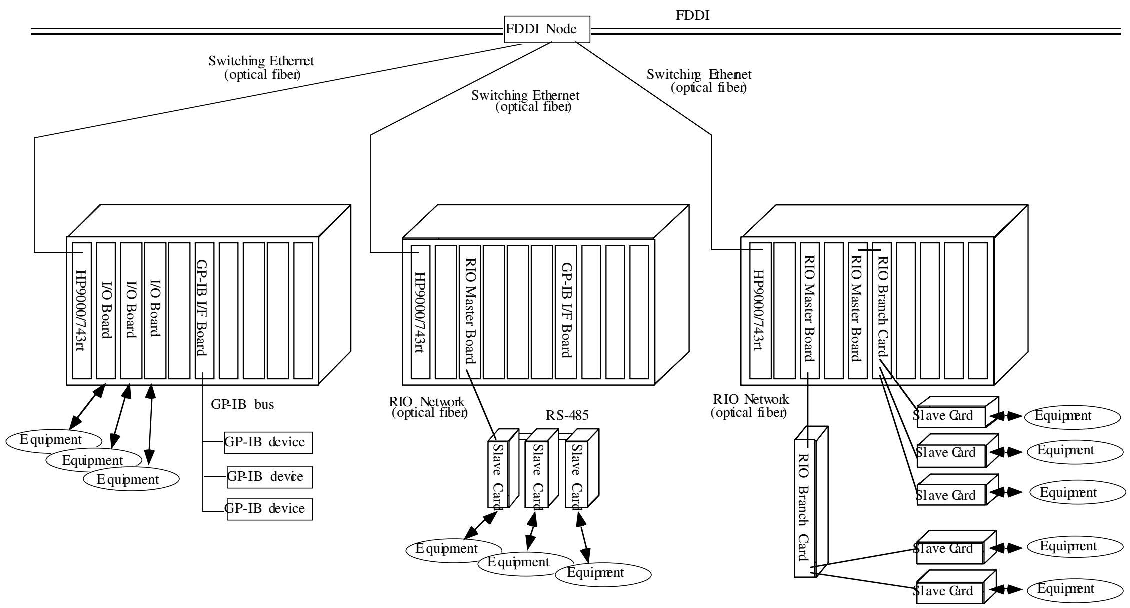

Fig.1. Schematic diagram of VME system and field buses.

2.3 Field buses

A fiber distributed RIO system was originally developed for the control of the magnet power supplies. The purpose of the development was electric isolation, noise rejection and economy.

The RIO system consists of the master boards, several types of slave cards and 8-port branch cards if needed. Optical fiber cables are used for connection in a star topology between the master board and the slave cards. The master is a VME board with a 4KB dual port RAM. Currently, six types of slave cards are available. These are Euro cards but not VME boards. The specifications of each slave card are as follows:

- Type-A: 1-channel AI with 16-bit resolution (integration type, conversion time < 128msec), 1-channel analog output (AO) with 16-bit resolution, 8-bit DI and 8-bit DO,
- Type-B: 32-bit DI and 32-bit DO,
- Type-D: 40-bit DI,
- Type-E: 1-channel AI with 16-bit resolution (integration type, conversion time < 32msec), 8-bit DI and 8-bit DO,
- Type-F: 4-channel AI with 12-bit resolution (successively conversion type, conversion time < 1msec) with 32KB local buffer and 1-bit external synchronization signal input,
- Type-G: 16-channel DI and 64-bit DO.

The features of the RIO system are as follows:

- optical fiber star connection between master board and slave cards,
- serial-link communication,
- maximum of 62 slave cards can be connected per one master board,
- 1Mbps transmission rate,
- cyclic transmission needs 0.2msec for type-A, B, D, E card, 0.5msec for type-G card and 27msec for type-F card for communication,
- up to 1 km transmission distance,
- HDLC protocol,
- can be connected by twisted pair cable (RS-485).

GPIB is also used as the field bus and largly used for the control of intelligent devices such as DVMs.

3. Operating system

3.1 Selection criteria

The selection criteria of the OS for the front-end controller were mainly real-time features and open system features, and that it should be compliant with POSIX which is a standard OS proposed by IEEE. Among such OSs, we chose LynxOS based HP-RT. The LynxOS supports many platforms such as i80386/80486/Pentium, M68030/40, SPARC2/microSPARC, R3000 and PowerPC 601/603/604. The HP-RT is a migrated OS of LynxOS to PA-RISC. Availability is restricted because HP-RT was originally only running on particular CPU boards such as HP9000/742rt or 743rt. However we expect better support from the company. It is a crucial point in reducing development cost and time for the new OS.

3.2 Features of HP-RT

HP-RT is just UNIX with real-time features, so it is not difficult to understand and manage. Actually we can get the same development environment and use the same commands as UNIX. LynxOS provides both self- and cross-development environments. On the other hand HP-RT provides only a cross-development environment and a host system is required. HP9000/700 or 800 series workstations are available as the host computer of HP-RT. Application programs and the kernels for the target system are built on the host system. HP-RT provides two types of kernel, one is RAM-based and the other is disk-based. In the case of RAM-based kernel, it is booted from the host system through the network with BOOTP (BOOTstrap Protocol), so the host must be the BOOTP server for many target systems with different kernels. At the same time, the host system has to provide the building environment for each kernel which has a different configuration. In other words, the host system has to provide both the download service for the specified kernel and the development environment for a different kernel. When the first installation of the specified kernel has been madeto the local mass storage device and problems arise, we must consider the same problem for the disk-based kernel This setup is a little complex. From the point of view of the management of the target system, the host-target system is more preferable.

4. Device driver

4.1 Feature of the HP-RT device driver

Device drivers are embedded in the OS kernel and are the glue between the kernel and the devices in the VME system. They convert the commands from the kernel to the devices and vice versa. By separating the drivers from the kernel itself, hardware independence of the original kernel is kept.

The HP-RT device driver is a collection of the functions called "entry points". The kernel can access the driver only through these entry points. The HP-RT device driver has nine entry points as shown in Table 1.

Table 1
Name of entry points of HP-RT device driver and its purpose

Entry Point	Purpose
install	Called to install a major device
uninstall	Called to uninstall a major device
open	Called when device is opened
close	Called when device is closed
read	Called to read data
write	Called to write data
ioctl	Called to control a device
select	support select system call
strategy	called to perform block read and write operation (only for block device)

The HP-RT device drivers can be written in C. Many special library functions with a C interface are provided because almost all ordinary C library functions are not available in the drivers. They are called "driver service calls". They provide the functions such as printf for kernel debugging, dynamic memory allocation, hardware interrupt configuration, kernel thread management, 10msec and 1ms resolution timer interrupt settings, system semaphore control, address conversion and so on. The driver accesses the devices in the VME system through the SWSM (System Wide Shared Memory) mechanism by which any VME address within the specified VME space is mapped into the virtual address of the kernel. By means of the SWSM mechanism and some particular service calls, we don't have to take care of the mapping of the virtual address corresponding to the VME address.

Three types of hardware interrupt handler are supported, SIGNAL_ONLY, ASM_CALL and C_CALL. The handler of ASM_CALL (written in PA-RISC assembly language) and C_CALL (written in C) are called by the interrupt dispatcher directly, while the handler of SIGNAL_ONLY is the kernel thread to wait for the system semaphore from the dispatcher.

4.2 Design concepts

Design concepts of our device driver are generality, maintainability and reliability. We should realize the primitive functions and eliminate any sequence depending on an application program, because the driver has to be modified when this sequence is changed. Such particular sequences should be realized as libraries outside of the drivers.

With regard to RIO, we should not implement the control sequences of each slave card into the device driver. When an application programmer wants to control the slave card directly, he does not want to know how to control the master. We realize such particular functions by creating libraries. These libraries are a part of the device drivers in a wide sense.

4.3 Status

The early versions of the device drivers for I/O boards have been developed except for the TTL DI/O, the RIO master board and the GPIB interface board. Hardware interrupts are supported in the AI, DI, PTG and GPIB interface boards by the SIGNAL_ONLY type.

We start the test and the modification of the RIO device driver because the test of our control software scheme, especially the test of the *EM (Equipment Manager)*, is planned to handle the steering magnet power supplies [3]. They are controlled through the RIO and interfaced to type-A slave cards.

Through the test of APIs of the RIO(see 5.2), the RIO device driver was confirmed to work well. This RIO device driver was installed in the real control system which has 7 RIO master boards. We checked that all master boards are controlled through the driver.

5. API for device handling

5.1 Design concepts

We have designed the application program interfaces (APIs) for the application programs such as *EM* to handle the devices in the VME system. Any application program should be written by using the APIs and never by touching the drivers directly by using system calls or libraries.

The design concepts of APIs are the following:

- provide flat and primitive APIs,
- provide uniform type of function and arguments and manner of handling.

5.2 Status

The first version of the APIs for open/close the device, RIO master board control and type-A and type-B slave card control have been developed. The master board is identified by the file descriptor which is a return value from the open function of the device. The slave card is identified by the slave number. Thirteen APIs have been developed for master control which involve the initialization of the master, control of the transmission, execution of the self-check and getting the status. Additionally 6 APIs for type-A slave and 3 APIs for type-B slave are provided for data taking and reading.

The primary system was found to work well. Actually these APIs were linked with the *EM* running on HP-RT for the control of the steering magnet. Two steering magnet power supplies were controlled. We could successfully set the power supplies to on/off, reset the error status, set and read current value and read the status with the RIO APIs and the *EM*.

References

[1] R.Tanaka et al., these proceedings (ICALEPCS'95, Chicago, USA, 1995)
[2] H.Takebe et al., Proc. of the 4th European Particle Accelerator Conf., London, June 1994, Page 1827-1829
[3] A.Taketani et al., these proceedings (ICALEPCS'95, Chicago, USA, 1995)