INTERFACING MODBUS PLUS TO EPICS FOR KEKB ACCELERATOR CONTROL SYSTEM

J. Odagiri, T. Katoh, K. Kudo, S. Kurokawa, T.T. Nakamura, and N. Yamamoto
KEK, High Energy Accelerator Research Organization
Oho 1 - 1, Tsukuba, Ibaraki, JAPAN 305

Abstract

The KEKB Accelerator control system[1] is based on EPICS(Experimental Physics and Industrial Control System)[2] and uses many PLCs in the magnet protection systems and the radiation safety system. In order to monitor the interlock status, Modbus Plus[3] is adopted as the protocol between an IOC(Input/Output Controller) and PLCs.

For this purpose, a device support and a driver support for Modbus Plus have been developed. The device/driver support modules allow an IOC to communicate with PLC-s by asynchronous I/O transactions, in such a manner that the GPIB devices do. With the software modules, an IOC works always as a master device on the Modbus Plus network to read the status of controlled devices from PLC memory. While the main use of the software is to read the interlock status, it is also used to reset the interlock systems.

Details of the software structure are described. An application of this software in the KEKB accelerator control system is also presented.

1 INTRODUCTION

The commissioning of the KEKB accelerator was launched in December 1998 and is now under progress. The control system of the KEKB accelerator has been constructed based on the EPICS software tool kit. EPICS provides application developers with rich functionality at various layers, especially at the IOC layer[5]. However, site-specific hardware requires new driver software to interface with EPICS.

One of them in the KEKB control system is the PLC device. Several kinds of PLC devices are used in the KEKB control system. Among these, the PLCs which talk Modbus Plus protocol are used in the magnet protection systems and the radiation safety system. While these systems are managed by the PLCs locally and autonomously, the interlock status should be monitored on the console in the central control room.

A set of software has been developed in order to interface these PLCs to EPICS. It has the standard structure required for the device access layers of EPICS. It consists of a device support and a driver support. The device support deals with the processing specific to the Modbus Plus protocol. The device support is described in section 2 in detail. The driver support interfaces directly with the hardware and handles physical I/O requests. The driver support is described in section 3. This layered structure allows the software to be ported easily onto another system such as COACK-II(Component Oriented Accelerator Control Kernel-II)[4] in the future. Section 4 describes the application of this software in the KEKB accelerator control system. This paper focuses on the hardware access layer and does not refer to any user interfaces.

2 DEVICE SUPPORT

2.1 I/O transactions and record processing

The Modbus Plus protocol provides a set of standard data access commands to read the PLC memory or to write into it. An IOC sends these commands as a master device on the Modbus Plus network.

Since the communication with the PLC could cause a delay in the order of milli-seconds, the device support must handle the I/O in an asynchronous manner. The device support enters a request on its queue and notifies the driver that the new request must be processed. After the PLC completes the processing of the request, the IOC gets a response from the PLC and the transaction is completed. This is the case not only for reader access but also for writer access because PLCs return a response even to the Modbus commands for writer access. The device support inspects the response to confirm the validity of the transaction.

In this case, user applications can put a new value into EPICS records through Channel Access[6] even if an I/O transaction is in process. This is because the record is not in locked state while it is waiting for the asynchronous completion. However, that request can not be processed by the device support because the record is in busy state(PACT field of the record is being set to 1). This can cause incoherencies between the EPICS records and the remote PLC memory, if successive writer access requests are sent faster than the processing speed at the device access layer. The applications must be designed so that they can take care of this possible incoherency.

2.2 Record types and Modbus commands

The data accessible in PLC memory can be divided into two classes. One is a bit of data which is held on either coils or inputs. The other is a word of data(16 bits) which is held on holding registers or input registers. Bi/bo, mbbiDirect/mbboDirect, and waveform records are used for the former data class. Ai/ao and longin/longout records are used for the later data class[7]. The inputs are read-only for both classes, while the others are read and write.

In the Modbus Plus protocol, the type of data is denoted by the highest digit of the address. From the address together with the type of the record in question, the device support can decide which Modbus command is concerned. An appropriate Modbus command is selected for each record during the initialization of the device support. The users are requested to select an appropriate record type and to input the address information summarized below:

- Modbus Plus node address on the network
- address of the data in the PLC
- number of bits to be accessed(only for mbbiDirect/mbboDirect, and waveform records)
- routing path(described in the next subsection)

2.3 *Routing support*

Modbus Plus supports up to five stages of message routing through bridge nodes. The device support reads routing information together with the address information and passes them down to the driver support as a request. The driver support builds a data packet from the request and passes it down to the logical link control(LLC) layer which has been implemented on the interface board(described in the following section).

3 DRIVER SUPPORT

3.1 *Interface Board*

The model SV85 supplied by Modicon was adopted as the VME Modbus Plus interface. A processor on the board manages the processing required below the LLC layer. The driver support loads the data packet, which includes a Modbus command and routing information, and issues an interface command to have the board send the data packet. After the transaction is completed, the driver unloads the response from the board by issuing other interface commands.

Some of the features of the board are:

- functions as A24 or A16 VME slave
- no DMA capabilities
- transfer rate of approximately 500 ns/byte
- supports concurrent transactions on eight logical links to be in process at the same time.

That last feature had a significant impact on the design of the task assignment in the driver support.

3.2 *Task Assignment*

The driver support spawns the following tasks to support concurrent transactions on the logical links.

- tPutTask: has the board send the data packet when the task gets a request and an idle logical link, books which transaction is going on the logical link, and starts a watchdog timer for the transaction.
- tGetTask: inspects on which logical link a transaction has been completed when the task is awaken by an interrupt, cancels the watchdog timer, and unloads the response from the board, then releases the logical link.
- tAbortTask: checks if a logical link with the expired watchdog timer still holds a transaction, and if this is the case, it aborts the hanging transaction and releases the logical link.

These three cooperative tasks work together to get full transaction capability of the interface board.

Another task, named tKeepAlive, supplements the activities of the driver support. It just keeps issuing a interface command to keep the board alive even if there is nothing to be done.

3.3 *Interrupt handling*

The board generates two kinds of interrupts. One is the immediate reply to an issued interface command and the other one notifies the completion of a transaction on the logical links. An interrupt handler handles both of them.

4 APPLICATION

4.1 *Applied systems*

There are three systems which utilize the device/driver support to monitor the interlock status from the central control room. One is the radiation safety system and the others are the magnet protection systems for the beam transportation line and the KEKB main rings.

This subsection as well as following ones focus on the magnet protection system for the KEKB main rings, which is the largest among these systems. The PLC system is composed of fifteen sets of I/O modules distributed around the main rings. Only one of them has a CPU module and it collects all the data from the others through HDLC links. An IOC communicates with this representative unit through a Modbus Plus connection.

The KEKB main rings have about 1,600 water-cooled magnets. The status of the cooling water flow and the magnet temperature are monitored for each magnet. The status changes are latched on the associated holding coils provided in the PLC memory. The IOC monitors the latched status rather than raw inputs. This is needed because the input status is automatically recovered when the interlock condition is spontaneously restored. This is more likely to happen on the water flow status due to the fluctuation. The latched status is cleared by an operator through write channels.

In addition to the interlock status, the cumulative counts of communication errors, which occurred inside the PLC system, are also monitored by the IOC.

4.2 *Database design*

The IOC polls the PLC periodically in order to monitor the interlock status. This method is adopted because it does not require any additional programming for the PLC to achieve the monitoring, except for the latch logic.

To reduce the number of transactions required to scan the PLC memory, waveform records are used in this system. Unfortunately, the data to be transferred has to be divided into at least two groups, because the number of data points exceeds the maximum(two thousand bits) allowed for single Modbus Plus transaction. From a logical point of view, they are actually divided into eleven groups.

On the other hand, it is convenient for the higher level applications that the IOC holds each status bit as an independent bi record. It is possible to decompose the data bits in a waveform record into a set of bi records by using the database link mechanism only. However, it is simpler to create a special device support routine which directly maps the bi records onto a waveform record. This is purely a software device support in the sense that it does not cause any Modbus Plus transactions in itself. It just allows a bi record to read a bit of data from an appropriate position of the associated waveform record.

Consequently, the database has a double layer structure. The lower layer maps the PLC memory onto the waveform records on the IOC by executing Modbus Plus transactions. The higher layer maps the bi records onto the associated waveform record by using the software device support specific to this purpose. Applications running at the man-machine interface layer see only the bi records through Channel Access. Table 1 summarizes the ingredients of the database designed for the system.

Table 1: Records in the designed database

Rec. Type	# of Rec.	Use	HW access
bi	3340	int. status	no
waveform	11	int. status	read
longin	14	com. error	read
bo	43	latch clear, etc.	write

4.3 Resource consumption

A database scan processes more than three thousand records, and causes twenty-five Modbus Plus transactions. A scan rate of around 1 Hz is supposed to be sufficient for the accelerator operation. The measurement of the CPU load showed that the database scanning at the rate of 1 Hz caused about fifteen percent of the CPU load on the Force PowerCore-6750 CPU board(266MHz, 64MB RAM). Table 2 shows the CPU load arising from the relevant tasks. Most of the CPU load came from tGetTask, which had to copy the Modbus response from the interface board to the IOC memory. One of the EPICS general purpose task, named scan60, executed the processing of all records.

As for the memory consumption, the database required rather large amount of memory at the cost of holding a bit of status data as a record. The total memory consumption was about 4M bytes at run-time. This is in the acceptable range for the CPU board with 64M bytes RAM, which is now being used in the KEKB accelerator control system.

Table 2: CPU load(at 1Hz)

task	CPU load(%)
scan60	2.0
tPutTask	1.0
tGetTask	11.3
tAbortTask	0.0
tKeepAlive	0.2
interrupt	0.1
Total	14.6

5 CONCLUSION

A device support and a driver support for Modbus Plus have been developed in order to interface PLC devices to EPICS.

They are used to monitor the interlock status of the magnets and the radiation safety system, for the KEKB accelerator operation. The software has run stably for more than half a year on several IOCs since the commissioning of the KEKB accelerator was launched.

The database designed for the magnet protection system could monitor more than three thousand bits of status data at the scan rate of 1 Hz, consuming fifteen percent of CPU ability and about 4M bytes of memory.

6 REFERENCES

[1] T.Katoh, et al. "Present Status of the KEKB Control System", ICALEPCS'97, Beijin

[2] L.R.Dalesio et al., "EPICS Architecture", in Proceedings of ICALEPCS'91, KEK, Tsukuba, Japan, 1991, pp. 278-282.

[3] "Modicon Modbus Plus Network Planning and Installation Guide", Modicon, Inc., Industrial Automation Systems

[4] I.Abe, et al. "Project on Accelerator Control Kernel Development", in these proceedings.

[5] J.B.Anderson and M.R.Kraimer, "EPICS Input/Output Controller(IOC) Application Developer's Guide".

[6] J.O.Hill, "EPICS R3.12 Channel Access Reference Manual".

[7] J.B.Anderson and M.R.Kraimer, "EPICS Input/Output Controller(IOC) Record Reference Manual".