Development of EPICS Input Output Controller and User Interface for the PEFP Low Level RF Control System

Young-Gi Song*, Han-Sung Kim, Kyung-Tae Seol, Hyeok-Jung Kwon, and Yong-Sub Cho
PEFP, KAERI, 150, Deokjin-dong, Yousung-gu, DaeJeon, 350-353, Korea
*Corresponding author: *ygsong@kaeri.re.kr*

1. Introduction

The Low-Level RF (LLRF) control system of the Proton Engineering Frontier Project (PEFP) [1] was developed for handling the driving frequency for Quadrupole (RFQ) and the Draft Tube Linac (DTL) cavities in 2006 [2]. The RF amplitude and phase of the accelerating field were controlled within 1% and 1 degree by stability requirements, respectively. Operators have been using the LLRF control system under the windows based text console mode as an operator interface. The LLRF control system could not be integrated with Experimental Physics Industrial Control System (EPICS) [3] Input Output Controllers (IOC) for each subsection of PEFP facility.

The main objective of this study is to supply operators of the LLRF control system with user friendly and convenient operating environment. The new LLRF control system is composed of a Verse Module Eurocard (VME) baseboard, a PCI Mezzanine Card (PMC), Board Support Package (BSP), EPICS software tool and a Real-Time Operating System (RTOS) VxWorks. A test with a dummy cavity of the new LLRF control system shows that operators can control and monitor operation parameters for a desired feedback action by using EPICS Channel Access (CA).

2. System Configuration

In order to supply operators with user friendly operating environment and control RF amplitude and phase of the accelerating field within 1% and 1 degree, it was decided to upgrade the digital RF feedback control system. The main focus is to improve latency, output phase locking and user interface. As a FPGA board, a Pentek 7142 PMC board was chosen. The Integrated Development Environment (IDE) is set up with Board Support Packages (BSP) of the MVME5100 baseboard and the Pentek 7142 by using Tornado IDE for vxWorks on SUN. The following is a process for the board test;

- Quadmode BSP compile and execution
- ADC BSP compile and execution
- DAC BSP compile and execution
- ADC signal input → DAC signal output
- ADC signal input → Control algorithm
  → DAC signal output

The configuration of hardware and software consist of two kinds of cross-compile environment for generating object codes for local test and a VME EPICS IOC server, respectively. Figure 1 shows a schematic block diagram of the network based IDE, target system and data storage system.

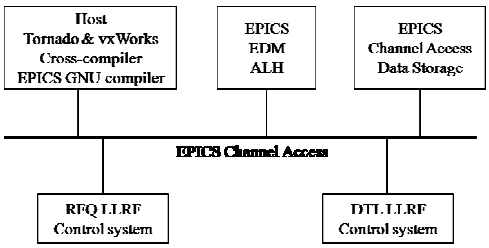

Figure 1. Integrated development environment and distributed control system

The host system is used to generate the vxWorks image and the BSP object code and compile EPICS software tool. The target system can download those from the host by using tftp, file transmission tool. The storage system is data management system that saves ADC signals transmitted from the PMC RAM to the baseboard RAM through PCI bus.

2.1 Target baseboard

The VME, PowerPC processor architecture, was adopted to accomplish compact hardware and flexibility of the feedback and feed-forward algorithm. The station of the target IOC that runs in a different environment where compiled is Motorola MVME5100 (Figure 2).

Figure 2. (a) Target baseboard and core PMC board

(b) VME crate and target board

Table 1. Test board and software

CPU Type	PowerPC 604
Clock Rate	400 MHz
RAM	512 MB
L1, L2 cache	32 KB, 2 MB
VxWorks Ver	5.5

2.2 Application implementation

First, the programs generated with BSP are executed on the target board. Next, the BSP for all the libraries is included in EPICS software architecture. Figure 3 shows EPICS software flowchart and pseudocode in Input Output Controller (IOC). The EPICS software consists of high level physics modeling; making use of live data stored in the site relational database and dedicated equipment controllers, which in turn interface to specific equipment. The vxWorks startup script is used to load EPICS software and database. The path for the startup script is specified in the boot parameters.

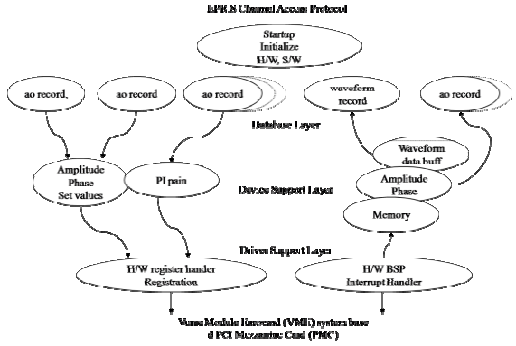

Figure 3. (a) Program flowchart of LLRF EPICS IOC

```
field (DTYP, "LLRF_AI")
field (DTYP, "LLRF_AO")
field (DTYP, "LLRF_BO")
field (DTYP, "LLRF_WF")

device(ai, VME_IO, devLLRFAi, "LLRF_AI")
device(ao, VME_IO, devLLRFOut, "LLRF_AO")
device(bo, VME_IO, devLLRFDigOut, "LLRF_BO")
device(waveform, VME_IO, devLLRFWf, "LLRF_WF")

epicsExportAddress(dset, devLLRFAi);
epicsExportAddress(dset, devLLRFOut);
epicsExportAddress(dset, devLLRFDigOut);
epicsExportAddress(dset, devLLRFWf);

LLRF devLLRFAi = {6,
                  NULL,
                  NULL,
                  init_ai_record,
                  NULL,
                  read_LLRF_record_ai,
                  NULL
};

LLRF devLLRFOut = {6,
                   NULL,
                   NULL,
                   init_ao_record,
                   NULL,
                   write_LLRF_record_ao,
                   NULL
};

LLRF devLLRFDigOut = {6,
                      NULL,
                      NULL,
                      init_bo_record,
                      NULL,
                      write_LLRF_record_bo,
                      NULL
};

LLRF devLLRFWf = {6,
                  NULL,
                  NULL,
                  init_wf_record,
                  NULL,
                  read_LLRF_record_wf,
                  NULL
};
```

(b) Pseudocode of LLRF EPICS IOC

3. Test Results

The EPICS object code was loaded and operated on PowerPC 604 CPU. The developed control algorithm core was also loaded in Xilinx FPGA. The ADC data uploaded from the PMC board which operates with an external gate signal was displayed on the operator interface. The running statuses of the CPU and memory of the baseboard are shown in Table 2, 3. Figure 4 shows operator interface using EPICS Extensible Display Manager (EDM).

Table 2. Task activity report using spy of a user interface subroutine library

NAME	PRI	total %	(ticks)	delta%	(ticks)
CAS-client	179	0%	(0)	0%	(0)
CAS-event	180	0%	(15)	0%	(2)
CAS-TCP	181	0%	(0)	0%	(0)
CAS-beacon	182	0%	(1)	0%	(0)
CAS-UDP	183	0%	(0)	0%	(0)
errlog	189	0%	(0)	0%	(0)
taskwd	189	0%	(0)	0%	(0)
KERNEL		0%	(1)	0%	(0)
INTERRUPT		0%	(3)	0%	(0)
IDLE		98%	(2969)	97%	(494)
TOTAL		98%	(3024)	98%	(505)

Table 3. Memory partition blocks and statistics

	status bytes	blocks	avg block	max block
current				
free	487513648	18	27084091	462447072
alloc	13117840	11520	1138	-
cumulative				
alloc	32675472	47853	682	-

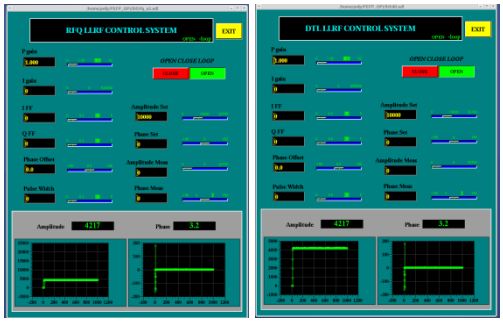

Figure 4. Operator interface for monitoring and control

4. Summary

The PEFP LLRF IOC was developed, and was able to be connected with other distributed control systems. In future, the IOC systems will be able to be integrated with a central control system to design sequential control and software interlock through channel access.

5. Acknowledgements

This work is supported by the Ministry of Education, Science and Technology of the Korean Government.

Reference

[1] Y. S. Cho, H. M. Choi, S. H. Jan, I. S. Hong, J. H. Jang, H. S. Kim, K. Y. Kim, Y. H. Kim, H. J. Kwon, K. T. Seol, and Y.G. Song, Test Result of the PEFP 20-MeV Proton Accelerator, EPAC 2006, p. 1609.

[2] Han-Sung Kim, Hyeok-Jung Kwon, Kyung-Tae Seol, Young-Gi Song, In-Seok Hong, and Yong-Sub Cho, Low Level RF Control System Development for the PEFP Proton Accelerator, J. Korean Phys. Soc. 50, 1431 (2007).

[3] Martin R. Kraimer: "EPICS IOC Application Developer's Guide, "APS/ANL, 1998