ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-MOPHA165

AN EMBEDDED IOC FOR 100 MeV CYCLOTRON RF CONTROL

Z.G. Yin†, X.L. Fu, X.T. Lu, T.J. Zhang, China Institute of Atomic Energy, Beijing, China
X.E. Mu, North China University of Technology, Beijing, China

Abstract

An ARM9 based embedded controller for 100MeV cyclotron RF control has been successfully developed and tested with EPICS control software. The controller is implemented as a 3U VME long card, located in the first slot of the LLRF control crate, as a supervise module that continuously monitors the status of the RF system through a costume designed backplane and related ADCs located on other boards in the crate. For high components density and signal integrate considerations, the PCB layout adopts a 6 layer design. The Debian GNU/ Linux distribution for the ARM architecture has been selected as an operating system for both robustness and convenience. EPICS device support, as well as Linux driver routings, has been written and tested to interface database records to the onboard 12 multichannel 16bits ADCs and DACs. In the meantime, a chip selecting encoding-decoding strategy has been implemented from both software and hardware aspects to extend the SPI bus of the AT91SAM9g20 processor. The detailed software, as well as hardware designed, will be reported in this paper.

INTRODUCTION

CYCIAE-100 cyclotron is the proton-driven accelerator of Beijing Radioactive Ion Facility. It provides a continuously adjustable high-intensity proton beam with energy up to 100MeV and intensity up to 520uA [1]. The CYCIAE-100 cyclotron radiofrequency system consists of two sets of room temperature resonators, two 100kW power amplifiers and two sets of Low-Level RF systems [2-5]. The LLRF crate is located in the RF power amplifier room, typically with no operators nearby. So, it is required to supervise the RF system parameters such as the dee voltage amplitude, phase, incident power, and reflected power, etc. in the cyclotron control room, during daily operations. In the field of accelerator control, the EPICS software package is widely adopted to build control system and gain remote access over ethernet. Generally speaking, EPICS IOC control software can run on various kinds of hardware and operating systems. For example, it runs on x86 PC with windows and arm SOCs with Linux operating system. In order to reduce the volume and power consumption, the reported IOC was designed based on arm9 SOC chip with embedded Linux OS. Thus, it can be installed in the LLRF chassis as a long 3U VME card, as shown in Figure 1. The hardware, firmware and software design of this IOC module will be reviewed in the following section of this paper.

HARDWARE

ARM9 series processors combine the advantages of high computational power with a small footprint. It also provides advantages such as high reliability and low power consumption. These features make it ideal for embedded controller design. In recent years, IOC based on ARM9 processor has been widely adopted for the control of many large scientific devices around the world. For the hardware designs of the IOC described in this paper, an AT91SAM9g20 processor has been selected as the central processing unit. Besides, the hardware design also includes 64MB SDRAM as memory, selects NAND FLASH and SD card to store OS and data. Other related hardware resources are one USB bus, two serial ports, two SPI interfaces, one Ethernet, 40 GPIO, etc. The block diagram of the hardware design of the embedded IOC is shown in Figure 2.

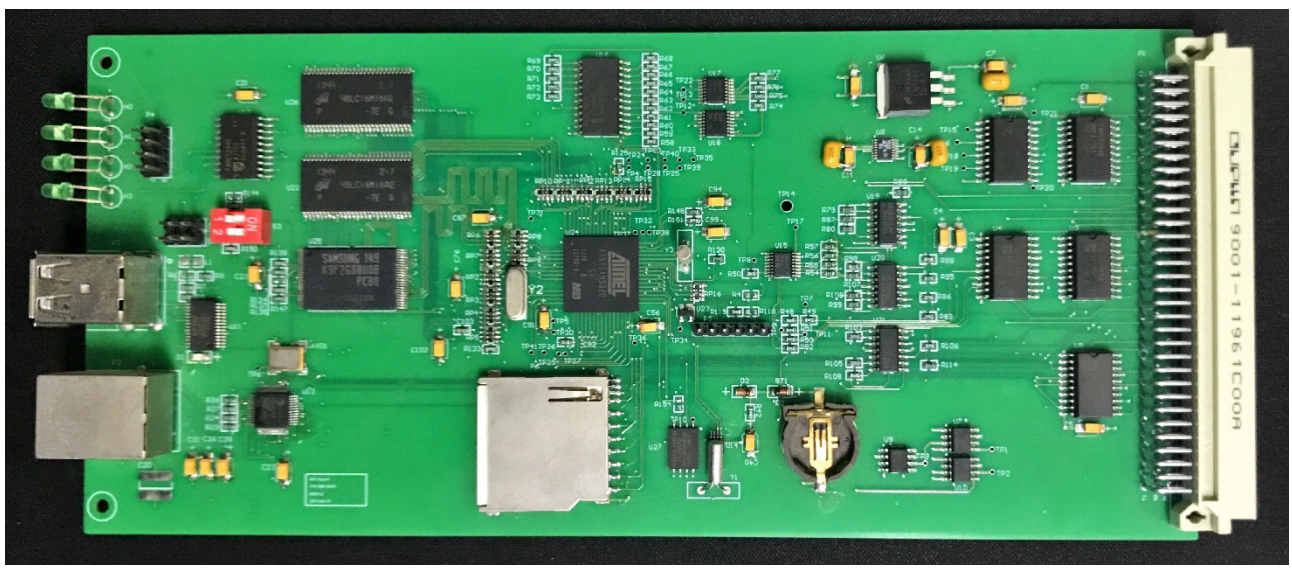

Figure 1: The embedded IOC as VME 3U module.

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-MOPHA165

The major function of the embedded IOC is to implement interlocks by using EPICS Process Variables of the main control system. This IOC monitors PV value of the main control system via channel access protocol, performs logical calculations and inhibit RF driven when necessary. The protections include the interlock of water-cooling of RF cavities, the main magnetic field, vacuum, and the RF amplifies. For example, if the final stage amplifier reflection power goes high, the PV value with the name "RFAmp:rev.Power" in the main control system changes accordingly. Periodically, the interlock routing in embedded IOC read this PV, if the value is higher than the preset threshold, the IOC will output low at certain GPIO pin to switch off the RF modulator. In this way, the RF drive to the RF power amplifier will be prohibited. Other related RF interlocks are implemented in a similar manner.

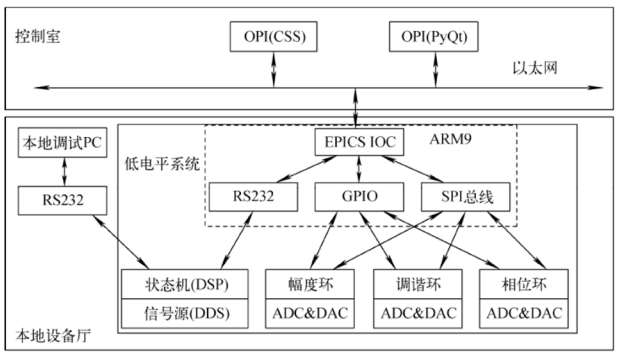

Figure 2: Hardware design of the embedded IOC.

In the reported low-level RF system, the low-level parameters such as the amplitude, phase of the open-loop driven are stored in the FLASH memory of DSP. Those high-level starting parameters, such as the power level of the pulse train, the relative phase of two cavities, etc. are managed by the IOC and stored in the file system of the embedded computer. At the operating time of CYCIAE-100 cyclotron, these high-level parameters can be adjusted online through the serial port, from IOC to DSP. Cyclotron operators can send commands/data to embedded IOC through CA protocol. After embedded IOC parsing the remote command and extracting parameters, it sends the command to DSP through the serial port to modify the relevant operating parameters of the RF system. In the meantime, the LLRF system can report status and abnormals to IOC through the serial port. Afterward, the embedded IOC can generate status information and alarm information, and forwarding to the remote control computer via CA Protocol.

In order to monitor and control the parameters of low-level RF system, the embedded IOC uses 4-channel ADC chips with 16-bit resolution (the ADS8341) to digitalize the key parameters of the LLRF system. It also utilizes 4-channel DAC chips with 16-bit resolution (the DAC8565) to set the work point of the low-level RF system. Typically, after DACs, the design includes several OPAMPs to obtain the required signal level. The ADC and DAC devices are connected to AT91SAM9g20 processor through SPI bus. This SPI bus is distributed to every module in the crate by the user-defined backplane. Since each low-level RF carte needs to detect and control up to 48 analogy signals, at least 12 SPI slave chip select signals are required, and the number of SPI bus slaves of the AT91SAM9g20 processor cannot meet the requirement. Therefore, we use 74LV154 decoder to expand the 4 bits chip select signals to 16 bits. Thus make embedded IOC hardware supports 64 analog input/output channels.

SOFTWARE

The software structure of the EPICS based control system typically consists of 3 layers, including the operator interface running at the operator end, IOC and channel access protocol running at the server-side. In the CYCIAE-100 cyclotron low-level RF system, the operator interface is developed using CSS and PyQt. It can run on multiple operating systems. A Debian GNU/Linux for the ARM architecture is selected for embedded IOC. A set of EPICS record database for the low-level system RF system has been developed, together with corresponding device support and EPICS device drivers. Since the hardware design adopts a 4 to16 extension of SPI chip select, a corresponding modified SPI Linux kernel driver has been developed. The Linux kernel of version 2.6.38 has been cross-compiled to integrate this special modification SPI driver. In the mean-time, the Board Support Package is also modified according to the hardware.

In brief, the embedded IOC software design work mainly includes the development of SPI device driver for Linux kernel; the development of EPCIS IOC database, EPCIS serial device, GPIO device, and SPI device driver support routing; OPI interface, etc. In the following section, the development of SPI device drivers, EPCIS serial device drivers, and OPI interfaces will be detailly reported.

SPI Device Driver

The hardware design of 100MeV IOC system modified the Chip Selection of SPI bus, the Linux kernel driver should be adjusted accordingly to enable system-level functionality. In Linux kernel 2.6, the SPI bus is treated as a character device. SPI devices have a limited userspace API, supporting basic half-duplex read() and write() access to SPI slave devices. Using ioctl() requests, full-duplex transfers and device I/O configuration are also available. The standard version implementation doesn't include the feature of chip select extending, therefore line by line read and modification of the code has to be done. The first step is to add an atmel_spi_data structure [6] to describe which GPIO they use as CS line and as well to enable or not the use of the CS decode feature. Secondly, the at91_add_device_spi() has been revamped, it is now used to add an SPI *controller* device only. The boards need to register their SPI devices with spi_register_board_info(). The third step is to modify the relevant code in the atmel_spi.c file under the Linux-2.6.38/drivers/spi/ folder, to add decoding support for the driver. Lastly, it is needed to modify the board support level definition file board-sam9g20ek.c under the

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-MOPHA165

Linux-2.6.38/arch/arm/match-at91/ folder, register the extended SPI device to the Linux device tree and initialize the 16 devices in the board level.

In practice, it is necessary to write a test program to verify the functionality of the SPI device driver. Several command-line testing codes have been developed to verifying read from ADC and write to DAC on a test board. These codes are cross-compiled to ARM microprocessor instruction set and executed in the target system console. Testing results show the modification of Linux kernel driver works as expected and the analog IO has been successfully extended to 64 channels.

In order to access the ADC and DAC devices using the IOC control software, the corresponding EPICS device support routing need to be developed. The development including coding for ADC and DAC device initialization functions, implementing EPICS record initialization functions, the record read/write functions, and registering related functions in the EPICS device structure. It is also necessary to provide the device support routing in the "Device Type" field in the device support file. All these routines are written and transferred to the file system that runs on AT91SAM9g20. Together with the IOC codes, EPICS driver support routings are compiled using GCC on the target system. A loop test by connecting the DAC output back to ADC has been performed. Multiple test results show that the relative error between the set values of the DAC and the reading values of the ADC are less than 0.4%. In such a way, we have a conclusion that the SPI driver, as well as the EPICS device driver, functions as expected.

Serial Communication with DSP

Stream Device is a wildly adopted EPICS serial communication driver in the past decade. It has been used in many EPICS based accelerator control system, to establish communication with intelligent power supply, vacuum gauge, etc. Stream Device uses a file to storage predefined protocol, thus it is suitable to communicate with the various kind of device. The protocol file is, generally speaking, both static and predefined. In most cases, it is very suitable for driving well defined serial devices. However, the DSP controller of CYCIAE-100 cyclotron low-level RF system takes advantage of a highly customized complex communication format to reduce dead-time of the real-time control. It is difficult to express the protocol into a static description. For example, the messages from the DSP controller are divided into three categories: command response, options menu list, and status information. The command response has a fixed format and length. The options menu and running status information don't, they depend on the context and time. In total the low-level RF system controller has up to 52 commands, including various error reporting and status information. It is possible to do the communication via Stream Device. Yet, in practices, it will be complicated and tricky. Therefore, the Python script is selected to implement the EPICS serial communication with the DSP controller.

Pyepics is the Python interface of the EPICS CA protocol developed by the University of Chicago. It uses the Python language to read and write EPICS PV via the CA protocol. The Pyepics operates through the EPICS base, performs operations such as reading, writing and monitoring EPICS PV by calling executable binaries including "caget", "caput", and "camonitor" in the EPICS base. In this case, it is necessary to compile the EPICS base in the target OS using AT91SAM9g20 SOC. In the meantime, Python, as well as the Pyserial extension runtime environment, should also be installed to the target embedded Debian system. After setting up these software modules, we can proceed with coding and verifying the Python device support routings.

The Python serial device support program is divided into three separate threads. They are the Pyepics thread, the reading serial port thread, and the writing serial port thread. In the device support, we set up two queues. One of them is to store the serial traffic. The other one is used to save EPICS message data. The read serial thread read messages from DSP, process it and translate it into EPICS PV access requests. The requests will then be processed by Pyepics thread. The Pyepics thread will also monitor the changes of certain PV value. If necessary, the change will be fetched to DSP by writing serial port thread.

Before putting into use, this EPICS serial driver is verified on the open-source hardware platform Raspberry Pi. After the preliminary test, the program is migrated to the embedded IOC. We found that, sometimes, a small amount of DSP responses are omitted by the driver. After trial and error, the cause of the problem is located. The response data generated by the DSP for some commands are bursty in nature. In limited time, a large frame of data can easily jam the PyEpics thread. That will eventually make one or two messages followed lost. To solve this problem, the Python regular expression is used to parse the DSP response message. In this way, effective parameter extraction is achieved. The difficulty of processing a sudden large amount of data transmission is conquered.

OPI Interface

The graphical user interface in the CYCIAE-100 cyclotron control room runs on multiple operating systems. It is required that the remote OPI of Low-Level RF should able to run on at least windows/ Ubuntu, and Macintosh OSX. In the early stage, especially in the system development phase, this OPI interface was developed using PyEpics and PyQt frameworks. It can be considered as a fast prototype for the development of IOC. Therefore, after commissioning of the cyclotron, in order to optimize and have better performance, we use CSS to rewrite the OPI.

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-MOPHA165

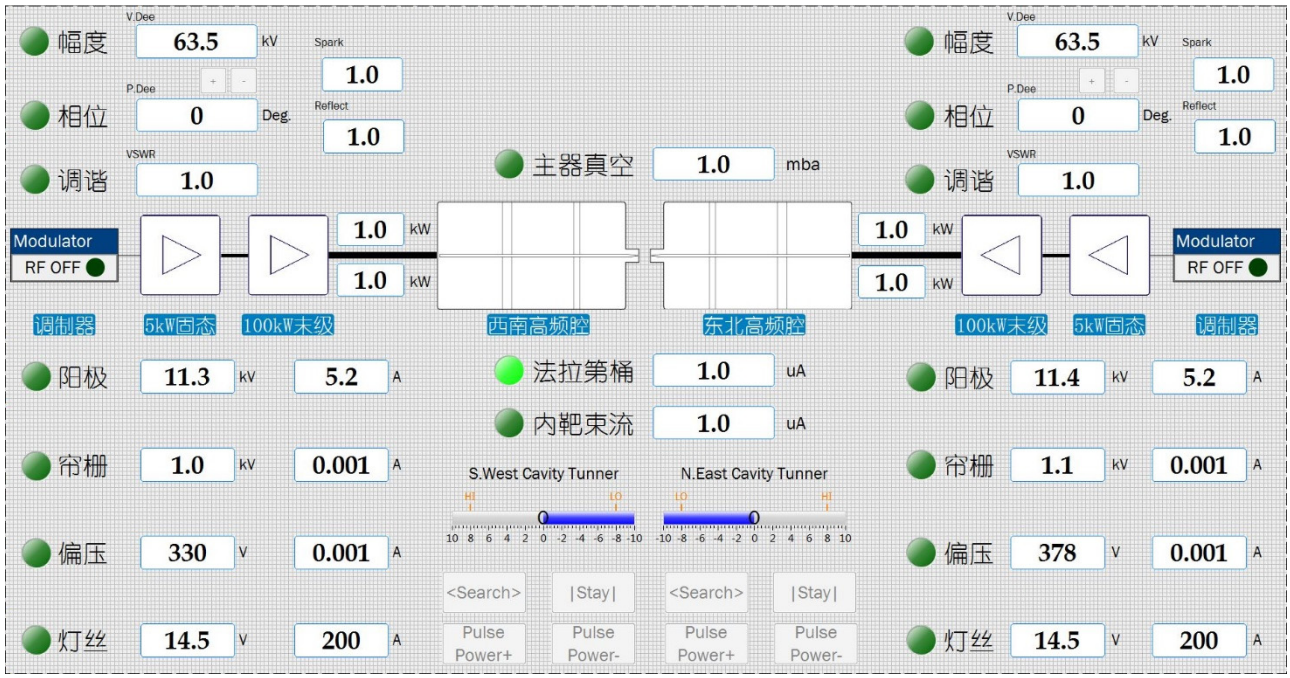

Figure 3: OPI interface for the LLRF IOC.

In the new OPI, we use the LED animation to display the switch status and use the text control to display the RF system information such as the Dee voltage, Driven amplitude, and phase, etc. Action button control as well as Jython, Javascript is used to accept user input and make change the PV value. For interlock process variables, the trigger PV is used in the script. Standardized animation and foreground/background color combinations are used to indicate the system interlock condition. Figure 3 shows OPI interface design.

CONCLUSION

This paper describes the technologies involves in the development of an embedded IOC based on ARM9 series processor, both from software and hardware aspects. The reported system achieved the goal of collecting data and implementing control using AT91SAM9g20 Soc. Special Linux kernel driver and EPICS device driver was developed to enhance the feature of extended Chip Select decoding. Python language is used to develop EPICS serial device support. The related EPICS database and operator interface are also carried out.

This embedded IOC is developed in late 2013 and goes online in early 2014. Together with the cyclotron LLRF control, it has been put into continuous operation for 24/7 after the commissioning of the cyclotron on May 4, 2014. Operational experience shows the design is stable and reliable. The embedded IOC was specially developed for the LLRF system of CYCIAE-100 cyclotron, yet the technology involved can be valuable for similar control systems.

REFERENCES

[1] Tianjue Zhang *et al.*, "mA beam acceleration efforts on 100MeV H− cyclotron at CIAE", *Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms*, vol. 406, part A, pp. 250-255, Sep. 2017. doi:10.1016/j.nimb.2016.11.024

[2] Xiulong Wang *et al.*, "The alternative of RF system design for the 100 MeV cyclotron at CIAE", *Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms*, vol. 261, nos. 1–2, pp. 70-74, Aug. 2007. doi:10.1016/j.nimb.2007.04.233

[3] Zhiguo Yin *et al.*, "RF control hardware design for CYCIAE-100 cyclotron", *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 801, pp. 104-107, Nov. 2015. doi:10.1016/j.nima.2015.08.057

[4] Zhiguo Yin *et al.*, "Digital control in LLRF system for CYCIAE-100 cyclotron", *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 819, pp. 33-36, May 2016. doi:10.1016/j.nima.2016.02.100

[5] Xiao-Liang Fu, Zhi-Guo Yin, Bin Ji, Zhen-Lu Zhao, Jun-Yi Wei, and Tian-Jue Zhang, "An automatic phase-matching technique of CYCIAE-100 cyclotron", *Nuclear Science and Techniques*, vol. 28, no. 9, p. 126, 2017. doi:10.1007/s41365-017-0277-9

[6] Christian Gagneraud, "[PATCH RFC 0/3] AT91: SPI: Add peripheral chip select decoding", Linux-arm-kernel mailing list, 31 May 2011.