REAL TIME CAPABILITY TEST ON THE RASPBERRY Pi BASED EPICS CONTROLLER

D. Wang[1,2*], K. Furukawa[2], S. Sasaki[2], M. Satoh[2], T. Obina[2], Y. Enomoto[2]
[1]SOKENDAI, Kanagawa, 240-0193, Japan
[2]KEK, Ibaraki, 305-0801, Japan

Abstract

Raspberry Pi is a series of small single-board computers designed and published by the Raspberry Pi Foundation. It is also a low cost and popular solution in building the EPICS Input/output Controller (IOC) owing to its high flexibility, expansion capability and affordability. EPICS is also adopted in the electron-positron collider, SuperKEKB accelerator. However, the requirement of the real time capability of the control system is critical since the specific demand of the injection system. The injector linac needs to switch lots of hardware parameters every 20 millisecond to perform the injection into several different storage rings. To assess the possibility and stability of Raspberry Pi based EPICS IOC within the 50 Hz beam repetition rate, we perform a latency test on the Raspberry Pi and compare the standard kernel and real time kernel.

INTRODUCTION

The injector linac provides beams to the SuperKEKB electron ring, positron ring as well as two synchrotron light source rings. EPICS (Experimental Physics and Industrial Control System) is chosen as the standard framework for the control system at the SuperKEKB facility. Since one injector linac serves for four rings, the control parameters and signals needs to change every beam repetition period. The IOC response delay time is of great significance to the normal operation of the accelerator.

At the early stage of the EPICS, the modular crate electronics, like CAMAC and VME chassis, are mainly used as the hardware of the EPICS IOC. The real time operation systems, like VxWorks and RTEMS, are usually utilized to meet the real time requirement. Now a variety of hardware can operation well as an IOC like Yokogawa's FA-M3 PLC based EPICS IOC [1], which is currently used at KEK in Japan. Some single-board computers based EPICS IOC are also used in the aceelerator facility , like Banana Pi used at TPS in Taiwan [2] and Raspberry Pi at SPES in Italy [3].

The single-board computer usually runs Linux and the default kernel is non-preemptive. One of the most famous one is the Raspberry Pi (RPi), The RPi Foundations also provides the open source real time kernel. In this current work, the real time performance of RPi based EPICS IOC is studied and presented by comparing the standard kernel and real time kernel.

* sdcswd@post.kek.jp

RASPBERRY PI

Several generations of Raspberry Pis have been released. The newest one, Raspberry Pi 4 Model B was first released in June 2019 [4]. The hardware specification of RPi is shown at Table 1.

Hardware

The new version of Raspberry Pi has a much powerful processor, full gigabit Ethernet, four USB ports, and 40 GPIO pins. The GPIO pins can be used with a variety of alternative functions, like PWM, SPI, I2C and Serial port.

Table 1: Hardware Specification of the Raspberry Pi

	Raspberry Pi 4B
CPU	Broadcom BCM2711 quad-core Cortex-A72 64-bit SoC @ 1.5GHz
Memory	2GB, 4GB or 8GB LPDDR4
Network	IEEE 802.11b/g/n/ac wireless Bluetooth 5.0 Gigabit Ethernet
I/O	USB, 40-pin GPIO header
OS	Debian, Raspberry Pi OS
Power	5V DC via USB-C connector

RT-Kernel

The latency means the time after a task is invoked and before executed. The Interrupt Latency and context switch latency measurements of EPICS IOC core was studied by Shifu Xu based on Linux kernel 2.6.13 which was built both preemptive and non-preemptive on 2005 but the results shows that no big difference had been found between two kernels [5]. On RPi 4B, the default Linux kernel is 4.19 and can be upgraded to 5.4 and the latest RT-kernel 4.19 could be compiled and installed by the user [6].

Test Conditions

To achieve a comprehensive understanding of the real time capability of the Raspberry Pi, several test conditions which includes the CPU frequency, the program running time, the CPU stress method and different kernel are considered.

The frequency governor determines the policy of CPU frequency. The default governor policy is "ondemand" and default frequency is 600 MHz. The available frequency steps are 600 MHz, 750 MHz, 1000 MHz, 1.50 GHz on kernel

5.4 while only 600 MHz and 1.5 GHz are available on RT kernel.

Since the execution of the program from idle state is always quite fast even on non-RT system, it is necessary to add real-time stress conditions to evaluate the performance. Three stress methods are utilized to simulate the high CPU load situation.

Test Tools

The RT-tests is an open source suite which contains several programs that test various real time features [7]. The cyclictest program insides the toolbox can be used to measure the latency of kernel. The idea of the cyclictest program is quite straightforward. The pthread library is used to create one thread per available processor. Every thread reads the system clock and then sleeps for a fixed interval. This thread reads the system clock again when it is invoked by the kernel and the difference between the real interval of designated interval is regarded as the latency.

Another program called hackbench inside the RT-tests toolbox is also used to stress the CPU load since the results is meaningless between standard kernel and real time kernel when the CPU is idle. The hackbench is a stress test for the Linux kernel scheduler which creates a specified number of pairs of threads to communicate via sockets by sending data back and forth to increase the CPU load. Like hackbench, the pi-stress program also provides the ability to stress the CPU.

Results

By comparing Fig. 1 and Fig. 2, under the 600 MHz CPU frequency, the real time capability is significantly improved on the RT kernel. The maximal latency under the hackbench stress on RT kernel is about 80 us while the latency is 200 us on standard kernel.

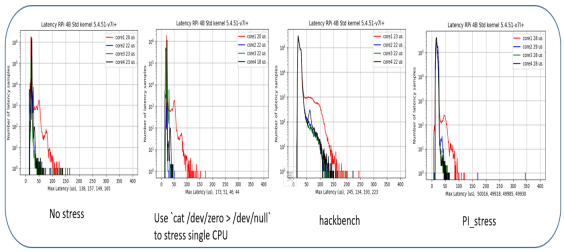

Figure 1: The CPU stress method comparison of the Std kernel.

For Fig. 3 and Fig. 4, the average latency and maximal latency under 1.5 GHz decreases to about half of which under 600 MHz both on standard kernel and RT kernel. The best real time performance, whose average latency is 8 us and maximal latency is 50 us, appears on the RT kernel under 1.5 GHz. The results from cyclictest shows that the latency decreases a lot and can be more stable on the RT kernel compared with the standard kernel.

Apart from the stress method and CPU frequency, the program running time is also tested and the result shows

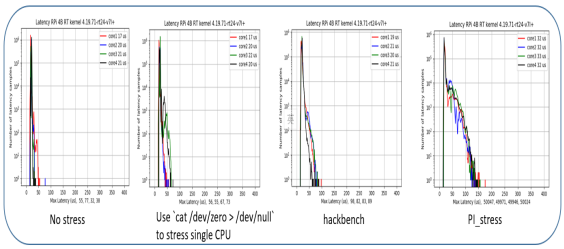

Figure 2: The CPU stress method comparison of the RT kernel.

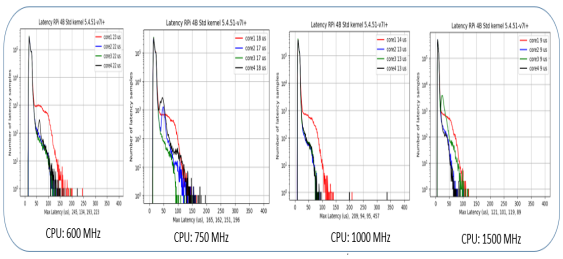

Figure 3: CPU frequency comparison of the Std kernel.

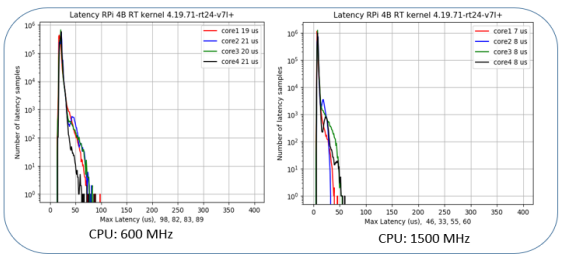

Figure 4: CPU frequency comparison of the RT kernel.

that there is no huge difference between 10 minutes and 50 minutes test. The result can be seen at Fig. 5.

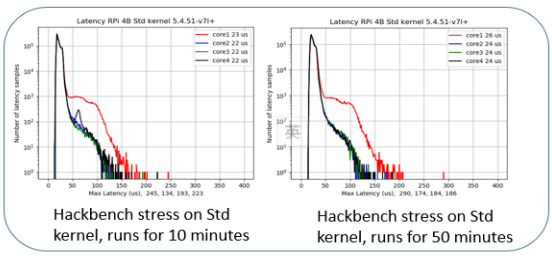

Figure 5: Program running time comparison of the Std kernel.

LATENCY OF EPICS IOC

Since the EPICS IOC does need some time to process the device support and record. It is necessary to verify the real delay of RPi with EPICS IOC.

Architecture of the Test Bench

As the Fig. 6 shows, a 25 Hz signal generator generates trigger signal and distributes them to RPi IOC through GPIO as well as the oscilloscope. The EPICS IOC simply read

the GPIO pin and then write it to another GPIO pin which connects with the oscilloscope. The time difference is calculated and recorded in the oscilloscope. EPICS version is 3.14.12.8 and only two EPICS records are used in this IOC. To achieve a better performance, the CPU frequency is set as 1.5 GHz on both kernels and the priority of IOC program is set as relatively high. The hackbench program is selected to stress the CPU.

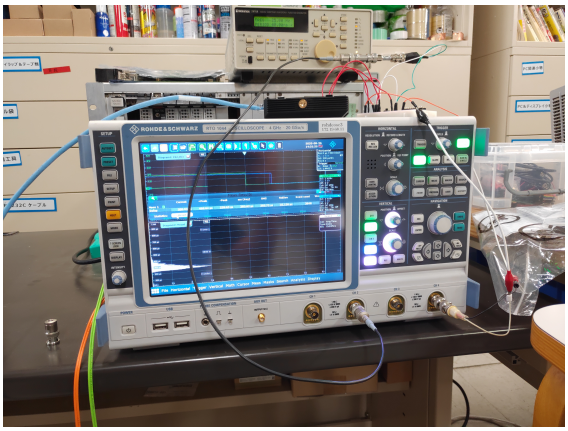

Figure 6: Architecture of the test bench.

Delay of EPICS IOC

The result of the EPICS IOC delay measurements are shown at the Fig. 7 and Fig. 8. The average delay value is around 300 to 400 us and the standard deviation is 34 us while on RT kernel the average delay becomes slightly longer but the stability is better.

The satisfactory performance of the standard kernel is quite unexpected. One possible explanation is that too much interrupt and context switch take up much time on the real time kernel.

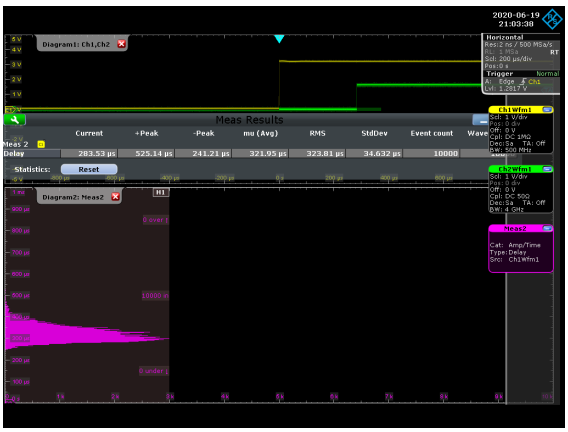

Figure 7: Latency with CPU stress of the Std kernel.

SUMMARY

The real time performance test is conducted on the Raspberry PI with standard kernel and real time kernel. The result shows that the real time kernel has a better real time capability and the latency is smaller with the increment of the CPU frequency. Another concern which is not talked above is the temperature of the CPU and GPU. Although the CPU temperature is acceptable during the 1.5 GHz experiment, it is highly recommended that the cooling plan should be taken into consideration for the long term operation.

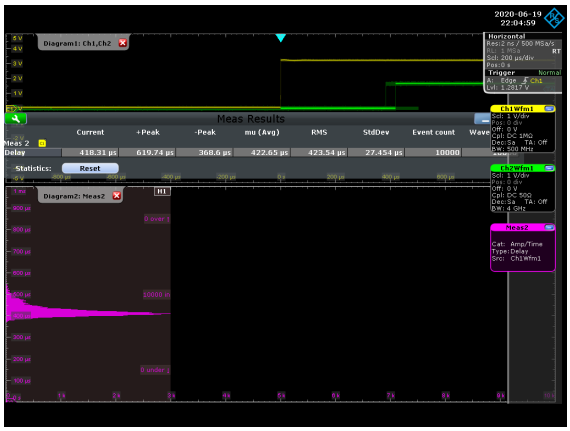

Figure 8: Latency with CPU stress of the RT kernel.

Then the delay of EPICS IOC processing is measured. The result is of considerable referential importance for those who wants use the Raspberry PI as an EPICS IOC. More experiments should be done to find out the reason of the poor performance of EPICS IOC on real time kernel compared with the standard kernel.

REFERENCES

[1] J. Odagiri *et al.*, “Integration of PLC with EPICS IOC for SuperKEKB Control System”, in *Proc. ICALEPCS’13*, San Francisco, USA, Oct. 2013, paper pp.31-34, MOCOBAB02.

[2] Y.-S. Cheng, K. T. Hsu, K. H. Hu, C. H. Huang, D. Lee, and C. Y. Liao, “Design and Implementation of Embedded Applications with EPICS Support for Accelerator Controls”, in *Proc. IPAC’16*, Busan, Korea, May 2016, pp. 4122–4124. doi:10.18429/JACoW-IPAC2016-THPOY017

[3] J. A. Vasquez, A. Andrighetto, G. P. Prete, and M. Bertocco, “New Control System for the SPES Off-line Laboratory at LNL-INFN using EPICS IOCs based on the Raspberry Pi”, in *Proc. ICALEPCS’13*, San Francisco, CA, USA, Oct. 2013, paper TUPPC053, pp. 687–690.

[4] Raspberry PI, http://www.raspberrypi.org/

[5] S. Xu and M. R. Kraimer, “Real Time Performance Measurements of EPICS IOCcore”, in *Proc. ICALEPCS’05*, Geneva, Switzerland, Oct. 2005, paper P3-075.

[6] RT kernel for raspberry pi, http://github.com/raspberrypi/linux/tree/rpi-4.19.y-rt/tools

[7] Suite of real-time tests, http://git.kernel.org/pub/scm/utils/rt-tests/rt-tests.git