Transactions of the Korean Nuclear Society Autumn Meeting
Gyeongju, Korea, October 29-30, 2015

An Applicability Evaluation on Real-time Kernel Prototype for Nuclear Safety-Grade Controller

Kwang Il Jeong [a*], Joon Ku Lee [a], Geun Ok Park [a], Je Youn Park [a], In Soo Koo [a], Byeong Seok Yoo [b]
[a]*I&C/HF Division, Korea Atomic Energy Research Institute, 111, Daedeok-daero 989Beon-gil, Yuseong-gu, Daejeon*
b Next Technology R&D Center, MDS Technology Co., Ltd., 3,4FL. Hancom Tower, 49, Daewangpangyo-ro 644 Beon-gil Bundang-gu, Seongnam-si, Gyeonggi-do, 13493, Republic of Korea

1. Introduction

The safety critical system such as reactor protection system in instrumentation and control (I&C) system of nuclear power plants shall be designed and operate to perform the safety function in any circumstances. A safety-grade controller performing the safety critical function of nuclear power plants uses a real-time operating system (RTOS) to run application programs for monitoring inputs from plants and transferring the processed outputs or control signals to plants. Generally, the safety critical systems of the domestic nuclear power plants have been applied mainly a commercial RTOS as a commercial grade item (CGI), where a commercial RTOS does not developed in compliance with nuclear requirements and qualification assurance. In this paper, we evaluate the applicability of commercial RTOS in order to use commercial RTOS on nuclear safety-grade controller. This paper describes the analyzed demands of domestic commercial RTOS, the developed real-time kernel prototype and the evaluation results of applicability of the developed real-time kernel prototype through various tests and analysis.

2. Real-time Kernel Prototype Development

We considered the work scope and procedures of real-time kernel prototype development. The established work scope and procedures are as followings:

1) Requirement analysis of RTOS in nuclear safety-grade controller
   • Detail analysis of scheduler function in nuclear safety-grade controller
   • Kernel software requirement specification development based on commercial real-time kernel
2) Commercial real-time kernel transplant to nuclear safety-grade controller
   • Modification of commercial real-time kernel according to developed kernel software requirement specification
   • Hardware support layer development on reference target board embedded with a MPC8359 processor
3) Development of Hardware abstraction and I/O function for nuclear safety-grade controller
   • Input/output device driver
   • Timer device driver for real-time kernel
4) Real-time kernel prototype function test on reference target board
   • Real-time kernel prototype function test according to test procedure
   • Performance validation tests of real-time kernel prototype

2.1 Requirement analysis of RTOS in nuclear safety-grade controller

We analyzed that RTOS shall meet the following major function requirements and guarantee its function and performance through an independent verification and validation(V&V) in accordance with IEEE 7-4.3.2.

- Deterministic performance: The overall performance of RTOS and real-time application shall be deterministic. Therefore a real-time kernel is required to support an application program to have a deterministic
- Predicable performance: RTOS must have a deterministic run limit. It must also respond to external time in a predictable way of which meet the defined timing requirement.
- Timing Requirement: RTOS must have predictable response in all load conditions and under the strict timing requirements possible.
- Scheduler: A scheduler of RTOS has to ensure that application programs shall complete within each cycle.
- Interrupt handling: RTOS should provide the processing time for a high priority interrupt. Further, the interrupt handling techniques must be capable of forecast.
- Thread management: Real-time operating systems should be designed and manage to be ensure that an operating thread meet a deadline.

2.2 Development environment

The development environment using following compiler, assembler, linker and debugger is configured in host PC and used to build the developed source codes of real-time kernel prototype and run the execution files of real-time kernel prototype as shown in Fig. 1.

- Compiler: GNU C Compiler Ver. 4.6.4
- Assembly Compiler: GNU Assembly Ver. 2.22
- Linker: GNU ld Ver. 2.22
- Builder: GNU make Ver. 3.81
- Archiver: GNU ar Ver. 2.22

- JTAG USB emulator: Trace32 debugger

3. Real-time Kernel Prototype Test for evaluation

Unit tests and integration tests on the reference target board were carried out to evaluate the functions and performance requirements for the nuclear safety-grade controller

3.1 Unit test

The functions of real-time kernel prototype for nuclear safety-grade controller were test whether the functions were completely implemented with needed functions in accordance with requirement specification. And the functions were tested through white box test whether there were any errors or logically faulty contexts under implementation. The test environments are shown in Fig. 2. Unit tests were carried out with priority given to thread management, scheduler and the semaphore functions for synchronization and 8 unit test results out of 80 unit test items are given in Table I.

3.2 Integration Test

The integration tests were performed to verify the software integration of real-time kernel prototype through black box tests of the function oriented and operation test of user APIs. Integration tests were carried out with priority given to user APIs and 6 integration test results out of 80 integration test items are given in Table II.

3.3 Performance Validation Test

3.3.1 Timing Analysis

The timing analysis is performed to measure the non-deterministic performance of hardware and verify whether the real-time system misses a deadline. We performed the deterministic performance analysis through the timing analysis about the following performance variables in the real-time kernel prototype.

3.3.2 Thread Switching Latency

The thread switching latency shown in Fig. 3 is the time needed by the operating system to switch the CPU to store the context information and to run another thread. To verify the thread switching latency, we created two threads (Thread A and Thread B) and measured the switching time from thread A to thread B.

The measurement values of thread switching latency were measured in at least 74 clocks and up to 95 clocks as shown in Fig. 4 when the measurement clock was 66.666MHz. That is, it was confirmed that the thread switching latency was required in at least 1.1*us* and up to 1.424*us*. Therefore it was confirmed that the real-time kernel prototype can be designed to have deterministic timing characteristics for its small variation of thread switching latency.

3.3.3 Interrupt Latency

The interrupt latency shown in Fig. 5 is the time is the time that elapses from when an interrupt is generated to when the source of the interrupt is serviced. The decrementer timer capable of generating the interrupt every 1 second was used to measure interrupt latency and we measured the interrupt latency 100,000 times. The measurement values of interrupt latency were measured in 14 clocks steadily as shown in Fig. 6 when the measurement clock was 66.666MHz. That is, it was confirmed that the interrupt latency was required in at least 210.0021 *ns*. Therefore it was confirmed that the real-time kernel prototype can be designed to have deterministic timing characteristics for its no variation of interrupt latency.

4. Conclusions

The real-time kernel prototype was tested the functions and performance requirements for the nuclear safety-grade controller through unit tests and integration tests on the reference target board. It showed the possibilities that the real-time kernel prototype can be used as an operating system for a nuclear safety-grade controller. Also the work is in progress to transplant the developed real-time kernel prototype to the developing hardware for nuclear safety-grade controller and it will be tested its functions and performance. After this procedure, we will make final conclusion.

ACKNOWLEDGMENT

This work has been supported by the National Research Foundation of Korea (NRF) granted financial resource from the Ministry of Science, ICT and Future Planning (MSIP), and Republic of Korea (No. 2015M2B9A1024473).

REFERENCES

[1] IEEE Std. 7-4.3.2, IEEE Standard Criteria for Digital Computers in Safety Systems of Nuclear Power Generating Stations, IEEE, 2010.
[3] KINS/HR-719, Evaluation of Real Time Operating System, KINS, 2006.
[4] Y.D. Kang, K.T Chong, Safety Evaluation on Real Time Operating Systems for Safety-Critical Systems, Journal of The Korea Academia-Industrial cooperation Society, Vol. 11, No. 10 p. 3885-3892, 2010.

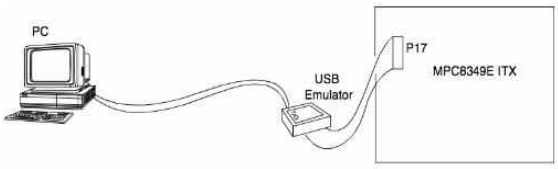

Fig. 1. Real-time kernel prototype development environment

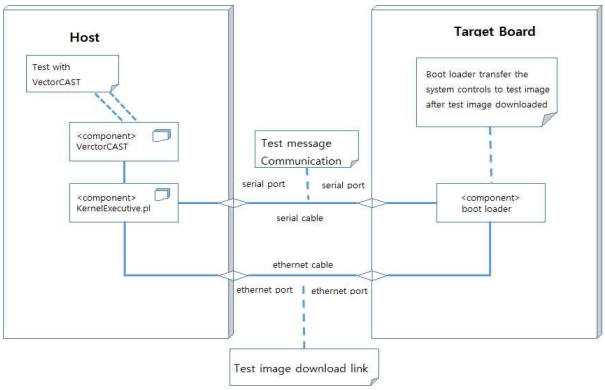

Fig. 2. Test Environment with host and reference target board

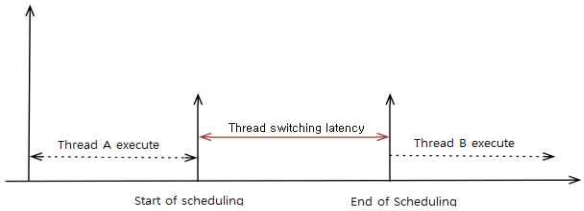

Fig. 3. Measurement range of thread switching latency

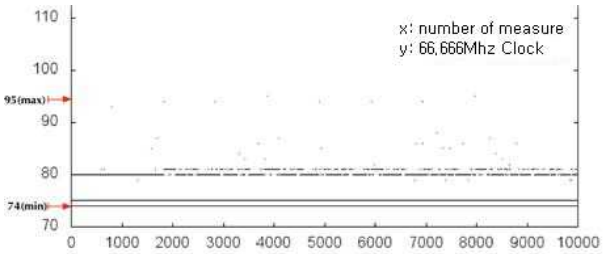

Fig. 4. Measurement result of thread switching latency

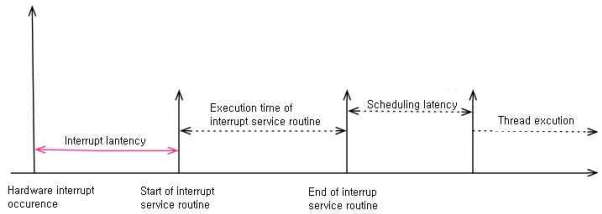

Fig. 5. Measurement range of interrupt latency

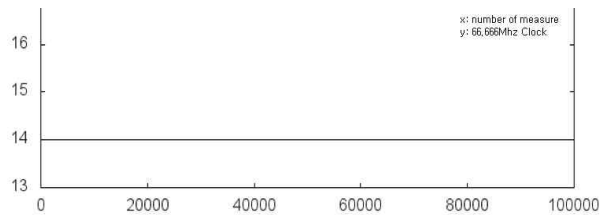

Fig. 6. Measurement result of interrupt latency

Table I: Unit Test Items and Results

Test Item (8 items out of 80 items)	Expected Return Result	Result
Thread Initiation test in Interrupt service function using ThreadInit()	E_ISR_NOT_CALLABLE	E_ISR_NOT_CALLABLE
Periodic thread initiation test in interrupt service function using ThreadPeriodicInit()	E_ISR_NOT_CALLABLE	E_ISR_NOT_CALLABLE
Deadline type input error of ThreadPeriodicInit()	E_INVALID_DEADLINE	E_INVALID_DEADLINE
'0' deadline time test of ThreadPeriodicInit()	E__INVALID_TIME_CAPACITY	E__INVALID_TIME_CAPACITY
'0' periodic time test of ThreadPeriodicInit()	E_INVALID_TIME_PERIOD	E_INVALID_TIME_PERIOD
Period waiting function of ThreadWaitNextPeriod() in interrupt service function	E_ISR_NOT_CALLABLE	E_ISR_NOT_CALLABLE
Function call test of ThreadReplenishDeadline() in interrupt service function	E_ISR_NOT_CALLABLE	E_ISR_NOT_CALLABLE
Error processing test when update time of ThreadReplenishDeadline() is infinite	E_INVALID_TICKS	E_INVALID_TICKS

Table II: Integration Test Items and Results

Test Item (6 items out of 25 items)	Expected Result	Result
Thread information function of real-time kernel	Thread Information Providing Functions shall Provide Information without Failure	Provide all Thread Information
Halt and restart function of real-time kernel	Threads shall be halt and restarted	Threads were halt and restarted
Automatic thread stop function of real- time kernel	Threads shall be stopped automatically	Threads were stopped automatically
Priority allocation and return function of real-time kernel	Thread priority shall be allocated and returned	Thread priority was allocated and returned
Scheduler lock and unlock function of real-time kernel	Scheduler shall be locked and unlocked	Scheduler was locked and unlocked
Thread deletion prohibit function of real- time kernel	Thread shall not be deleted when thread deletion function was initiated	Thread was not deleted when thread deletion function was initiated