A Study on the Safety Evaluation of Real-Time Operating System in Nuclear Power Plants

Hyung Tae Kim, Choong Heui Jeong, Dail Il Kim
Instrument & Control Division, Korea Institute of Nuclear Safety, 19 Kusong-dong, Yuseong, Daejeon, Korea,
kshape@kins.re.kr, k148jch@kins.re.kr, dikim@kins.re.kr

1. Introduction

Along with the digitalization of the nuclear Instrumentation & Control (I&C) system, Real-Time Operating System (RTOS) is being widely used. The RTOS used in nuclear I&C system should satisfy strict performance requirements and resolve various technical issues under complicated conditions. In this regard a careful safety evaluation of RTOS is important for the safety of Nuclear Power Plants. The objective of this study is to provide a guideline for safety evaluation of RTOS appropriate to the nuclear I&C system. In this paper, we suggest evaluation approach for the RTOS.

2. Real-Time Operating System

An Operating System is said to be "real-time" if it provides some mechanisms to give predictability to task execution times. Usually, RTOS is classified as Hard real-time and Soft real-time. Hard RTOS provides facilities necessary to meet timing requirements under worst-case conditions. Soft RTOS meets timing requirements most of the time but are allowed to occasionally miss deadlines [1]. Because the nuclear power plants have to maintain a very high level of safety, the hard RTOS is usually used for safety-critical applications executing a required mission completely within its deadline.

3. Review Points of RTOS

A guideline for safety evaluation of RTOS includes real-time performance requirements and the characteristics of RTOS. The desirable features of an RTOS include the ability to schedule tasks and meet deadlines, ease of incorporating external hardware, error recovery, low task switching latency, small footprint and small overheads [2]. Also, because RTOS is tightly related with application task, both of them should be evaluated together.
The followings are the basic requirements of an RTOS [1,2,3,4].

- RTOS should have a real-time performance to ensure that real-time system meet deadlines with a deterministic behavior .
- Real-time components, such as scheduler or interrupt handling, should be evaluated.
- Known issues in RTOS, such as dead-lock or priority inversion, should be considered and their resolution method should be evaluated.
- Application software should be designed by a deterministic design method.
- Overall software performance should be deterministic and it should be verified to meet timing constraints under worst conditions or fault conditions.

4. Safety Evaluation Items of RTOS

We classified evaluation items into deterministic behavior, kernel function, task design and management, memory and resource management, error handling and communication. Table I shows safety evaluation items of RTOS.

Table I: Safety evaluation items of RTOS

•Deterministic Behavior	· Predictability · Timing Analysis · Timing Constraints
•Kernel Function	· Scheduler · Interrupt Handling · Synchronization Mechanisms · Avoidance of Priority Inversion
•Task Design and Management	· Task Design · Verification of Task Timing · Task scheduling · Task Synchronization · Reentrant Requirement
•Memory and Resource Management	· Access Time · Memory Allocation · Memory Management
•Error Handling	· Deadlock, Stack Overflow and Deadline Violation · Memory Protection · Diagnostics
•Communication	· Communication Scheduling

4.1 Deterministic Behavior

Behavior of overall software including RTOS and real-time applications should be deterministic.

- Predictability
  RTOS should have deterministic execution time bounds and respond to any external event in a predictable way to meet its specific timing constraints.
- Timing Analysis
  Timing Analysis should be performed to ensure that RTOS will meet its deadlines.
- Timing Constraints
  The following should be included its timing constraint.
  - Task switching latency
  - Interrupt latency
  - Interrupt dispatch latency
  - Interrupt Service Routine execution time
  - Execution Time of task

4.2 Kernel Function

Kernel is the core of an RTOS that provides task scheduling, inter-task communication, and task dispatching.

• Scheduler
RTOS should provide multi-tasking, preemtability and priority-based scheduling and use static scheduling.
• Interrupt Handling
Interrupt handling mechanism should be predictable.
All the existing interrupt sources must be evaluated appropriately and risk analysis of interrupt should be performed.
• Synchronization Mechanisms
RTOS should have the ability to synchronize resource access by multiple tasks and communicate between tasks and be predictable.
• Avoidance of Priority Inversion
RTOS should be designed to avoid priority inversion and proved that there is no priority inversion. If it isn't proved, RTOS should provide priority inversion control method.

4.3 Task Design and Management

Task should be designed and managed to ensure that system software will meet its deadlines.

• Task Design
Task should be designed by using deterministic design method such as static-priorities task, static task etc. Also, task stack should be sufficiently allocated according to the prediction of maximum usage of it with sufficient margin.
• Verification of Task timing
All tasks' timing constraint should be verified.
• Task scheduling.
Schedulability of tasks should be proved and schedulability analysis should be performed.
• Task synchronization
If synchronization method to share resource is used, risk analysis of synchronization should be performed in order to guarantee that there is no possibility to occur deadlock.
• Reentrant Requirement
A shared function executed multiple tasks should be reentrant functions to prevent any corrupt.

4.4 Memory and Resource Management

This section addresses issues of memory and resource management in an RTOS.

• Access Time
Memory and resources' access time should be bounded.
• Memory Allocation
Static Memory allocation should be used.
• Memory Management
Memory management should be predictable and use of virtual memory and demand-paging is prohibited.

4.5 Error Handling

This section addresses error handling requirements in an RTOS.

• Deadlock, Stack overflow and Deadline violation
RTOS should have an ability to detect and handle critical error such as deadlock, stack overflow and deadline violation.
• Memory protection
In general, operating system data and important memory such as task stack are protected against hardware faults and illegal writes. RTOS should have an ability to protect memory and detect memory corruption.
• Diagnostics
RTOS should have an ability to detect faults such as process stall, executive program error and application program error etc.

4.6 Communication

Even though a communication system is deterministic, overall system deadline performance may still fail.[1]

• Communication scheduling
Communication system usage should be included in scheduling and demonstrate that message arrival is timely as specified.

5. Conclusions

We suggested the evaluation items of RTOS for nuclear power plants. Both RTOS and real-time application (i.e., task) should be evaluated and verified to satisfy severe performance requirements even in worst case conditions or fault conditions. In the near future, we will develop more detailed evaluation guideline in order to review a various RTOS effectively.

REFERENCES

[1] NUREC/CR-6083, "Reviewing Real-Time Performance of Nuclear Reactor Safety Systems" August 1993.
[2] Baskiyar, S. and N. Meghanathan, "A survey of contemporary real-time operating systems." Informatica, 2005. 29: p. 233-240.
[3] KINS/GR-363, "Development of the Safety Regulation Technology for Digital I&C Systems".
[4] Lothar Thiele, Reinhard Wilhelm, "Design for Timing Predictability", Real-Time Systems, v.28 n.2-3, p.157-177, November-December 2004