Design of Deterministic OS for SPLC

Choul-Woong Son [a*], Dong-Hoon Kim [a], Gwang-Seop Son [a]
[a]*Korea Atomic Energy Research Institute, 1045 Daedeokdaero, Yuseong, Daejeon, 305-353*
[*]*Corresponding author: scw@kaeri.re.kr*

1. Introduction

Existing safety PLCs for using in nuclear power plants operates based on priority-based scheduling, in which the highest priority task runs first. This type of scheduling scheme determines processing priorities when multiple requests for processing or when there is a lack of resources available for processing, guaranteeing execution of higher-priority tasks [1][2]. This type of scheduling is prone to exhaustion of resources and continuous preemptions by devices with high priorities, and therefore there is uncertainty every period in terms of smooth running of the overall system. Hence, it is difficult to apply this type of scheme to where deterministic operation is required, such as in nuclear power plant.

Also, existing PLCs either have no output logic with regard to devices' redundant selection or it was set in a fixed way, and as a result it was extremely inefficient to use them for redundant systems such as that of a nuclear power plant and their use was limited.

Therefore, functional modules that can manage and control all devices need to be developed by improving on the way priorities are assigned among the devices, making it more flexible. A management module should be able to schedule all devices of the system, manage resources, analyze states of the devices, and give warnings in case of abnormal situations, such as device fail or resource scarcity and decide on how to handle it. Also, the management module should have output logic for device redundancy, as well as deterministic processing capabilities, such as with regard to device interrupt events [3].

2. Deterministic OS Design

The designed OS has the following structure and features.

2.1 Structure of the OS

The scope and components of the designed OS is shown in Fig. 1. The OS is composed of MicroC/OS-II Kernel, IHS (Interrupt Handling System), I/O tasks (input task, output task), communication task (SC_Read_Comm, SS_Read_Comm, SC_Write_Comm, SS_Write_Comm), diagnostic task, interface task, application task, and supervisory task, which manages all tasks.[4] Table. 1. shows the function of each type of task.

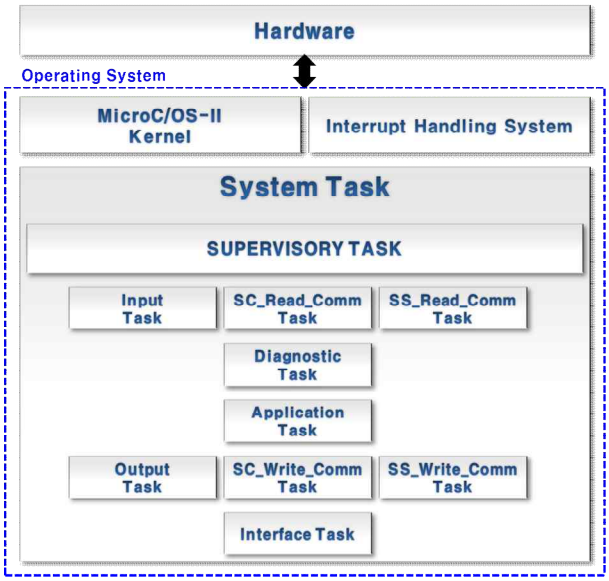

Fig. 1. OS structure

Table. I. Task Functions

Task Name	Function
Input	Input processing of I/O modules
SC_Read_Comm	Data read from safety critical communication modules
SS_Read_Comm	Data read from safety status communication modules
Diagnostic	Diagnosis of external hardware modules
Application	Executes an application that the user has programmed
Output	Output handling of I/O modules
SC_Write_Comm	Data write to safety critical communication modules
SS_Write_Comm	Data write to safety status communication modules
Interface	Interfacing with external modules
Supervisory	- Periodic Scan Time Control - Task Sequence & Time Control - System-Level (Critical Function) Diagnostic, - Security Control

The OS is subdivided according to different functions. The kernel, system task and application program task, under the management of the supervisory task and according to program operation, perform I/O with the hardware.

2.2 Supervisory's management function

A supervisory module was added, which manages and controls all components. It controls operation of all devices. It includes processing period management of the entire system wherein devices are included, system resources management, time management for each device, diagnosis and analysis of devices and the system, and scheduling function. It allocates resources in a way so that devices are linked in order according to their correlations with one another, and makes sure, other than interrupt, other devices can't preempt.[4] Fig. 2. shows the supervisory's management functions.

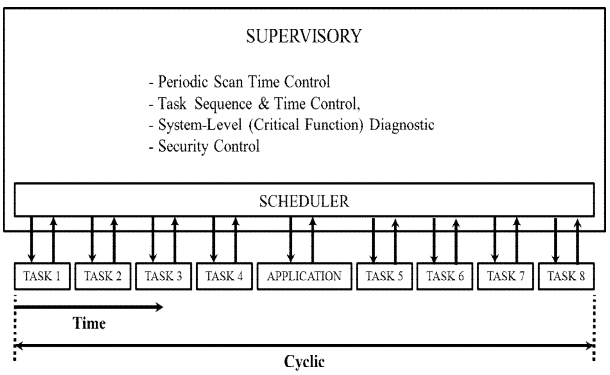

Fig. 2. Supervisory's management functions

The supervisory task manages all tasks, schedules tasks in the order of input-processing-output, and prevents resource competition.

2.3 Redundancy handling

The OS includes a feature for handling processing logic with regard to redundant structural changes from single to triple for input task, output task, and application task, among the software modules [3]. In this OS, at boot time, the supervisory receives from the user configuration information of hardware modules. A software module is then selected which can perform hardware module redundancy according to the configuration information, and scheduling is done.

3. Conclusions

We designed a deterministic OS for SPLCs. In the OS, the supervisory manages all functions, eliminating any risk that may be caused by resource scarcity, and timing issues were resolved as software modules, subdivided according to the function, are managed by a successive scheduling scheme.

Also, a feature was included for handling redundancy structure of SPLCs, for flexible handling of hardware module redundancy. The designed OS has a higher level of determinism, such SPLC enhance safety than existing PLC

REFERENCES

[1] JEAN J. LABROSSE, "MicroC/OS-II The Real Time Kernel"
[2] JEAN J. LABROSSE, "MicroC/OS-III The Real Time Kernel"
[3] D. H .Yun, "Design Report for SPLC (ANICS-SPLC-DR101)", Posco ICT, 2011
[4] M. G. Lee, "Detailed Design Report for SPLC (ANICS-SPLC-RR101)", Posco ICT, 2012

ACKNOWLEDGMENT

This work was supported by the Nuclear Technology Development Program of the Korea Institute of Energy Technology Evaluation and Planning (KETEP) grant funded by the Korea government Ministry of Knowledge Economy (No. 2010161010001G)