Development of Scheduler for Executing Safety-Critical Software Based On Single CPU Board

Gi Ho Cho [a*], Hee Seok Park [a], Je Yun Park [a], Chang Hwan Cho [b], Young Rok Sim [b]
[a] *Div. of SMART MMIS, KAERI, 150 Dukjin-dong, Yuseong-gu, Daejon, Korea, 305-353*
[b] *Control Tech. Research Inst., SEC Ltd., 974-1 Goyeon-ri Woongchon-myon, Ulju-gun, Ulsan, Korea, 689-871*
[*]Corresponding author: cgh@kaeri.re.kr

1. Introduction

Safety I&C Systems are composed of various hardware resources and software modules that are executed periodically. These systems require the efficient use of hardware resources

The OS (Operating System) of an industrial system is selected to meet the requirements of each system such as managing software modules and hardware resources efficiently. The OS of safety-critical system shall satisfy the requirements of V&V (Verification and Validation) and reliability analysis, etc. The problem is that the current OS is too complicated to allow the analysis and V&V of systems compatibility anomalies. In addition, the cost is expensive.

This paper describes the development of a scheduler. The scheduler is able to manage a single CPU board that does not have an OS. The scheduler shall periodically execute a variety of software modules and monitor their state.

2. Methods and Results

As a real-time digital I&C system must respond within time constraints required by plant process system, the severe performance requirements are imposed upon the design of the real-time systems. According to the reference [1], requirements for the scheduler are as follows;

1) The safety system of I&C should be predictable and deterministic
2) Tasks should have deadlines
3) Tasks should have the start time and end time
4) Hard real-time systems should adhere to the deadlines of tasks
5) Periodic tasks should be executed every cycle or interval time

The scheduling method is composed of static methods and dynamic methods. A dynamic scheduling method is excluded because the task is unpredictable and the schedulability is very difficult to prove. A static scheduling method is recommended for use on the safety system. The static scheduling method is consist of static table-based scheduling and static priority-based preemptive scheduling. We use the static table-based scheduling method and cyclic executive scheduling method. The following are the attributes of the cyclic executive scheduling method.

1) All tasks are performed according to a predetermined sequence.
2) The input data is stored in the global buffer and tasks that are executed sequentially access the data by the polling method.
3) Tasks get the information from the global buffer(or memory) and the calculated results are stored in the global buffer(or memory)

2.1 Implementation

The scheduler implementation environment is as follows;

1) Single CPU Board : TMS320C40(60Mhz) DSP board
2) NIC : Optical Network Interface Card
3) Tools : CCS(Code Composer Studio)
4) Language : Assembly

The scheduler functions are as follows;

1) The task of 25ms cycle should be performed every 25ms
2) The task of 50ms cycle should be performed every 50ms
3) The task of 100ms cycle should be performed every 100ms
4) The task of 1s cycle should be performed every 1s
5) The task of 2s cycle should be performed every 2s
6) The scheduler should generate a watch dog timer reset signal every 25ms.
7) Each task should be given priority and should be executed according to priority
8) Each task is performed at a certain time should be monitored

2.1.1 Scheduler configuration

The scheduler is organized as follows;

1) Register initiation Module
2) Timer0_ISR(timer0 Interrupt Service Routine)
3) Timer1_ISR(timer1 Interrupt Service Routine)
4) Sel_Process module
5) 25ms_Process module
6) 50ms_Process module
7) 1s_process module
8) 2s_process module
9) Common_util module

The scheduler use Timer0 and Timer1. Timer0 is call Sel_Process module. Sel_Process module count tic signal and execute each process periodically. The Sel_Process functional diagram appears in Figure 1.

A process executes tasks in order of priority. A task sets the Timer1 and performs the calculation program.

The end of calculation program generates the timer1 reset signal.

Timer1 monitor task which is performed within a period of time. The task, monitoring functional diagram appears in Figure 2.

Common_util module has the function of delay, division, multiplication, etc. The scheduler schematic diagram appears in Figure 3.

2.2 Example

Using the scheduler, SMART Core Protection system was implemented. As the first step in our analysis, tasks are to identify the types, characteristics and period. Each task shall be prioritized. At the SMART core protection system, the order of performing the tasks is in Table 1.

The scheduler executes the tasks as follows

1) 25ms : WDT_Reset
2) 50ms : WDT_Reset ⇒ NIC_Input ⇒ COOLANT ⇒ NIC_Output
3) 75ms : WDT_Reset
4) 100ms : WDT_Reset ⇒ NIC_Input ⇒ COOLANT ⇒ CRPOS ⇒ CHECK ⇒ NIC_Output
5) 125ms : WDT_Reset
6) 150ms : WDT_Reset ⇒ NIC_Input ⇒ COOLANT ⇒ NIC_Output
7) 1s : WDT_Reset ⇒ NIC_Input ⇒ COOLANT ⇒ CRPOS ⇒ CHECK ⇒ POWER ⇒ NIC_Output
8) 2s : WDT_Reset ⇒ NIC_Input ⇒ COOLANT ⇒ CRPOS ⇒ CHECK ⇒ POWER ⇒ THERM ⇒ NIC_Output

2.3 Result

From 25ms to 2.025ms, the program ran with step by step and checked registers, running order, access memory, etc. Running time of each task was measured after the exercise and was compared with the estimated values. The scheduler performed periodically and executed the tasks was appropriately in order. The mission time of worst conditions were within the time of 10ms.

3. Conclusions

The scheduler we developed has only minimal functionality. There are a number of functions that remain to be developed. It is believed that further experimentation with the methods outlined is worthwhile.

REFERENCES

[1] Development of technologies for Evaluating Real-Time Performance of Digital I&C Systems, KINS/RR-103, 2000. 4
[2] SMART Core Protection System Design Requirement
[3] Functional Design Requirement for SMART Core Protection System
[4] TMS320C40 User's Guide
[5] TMS320C40 Floating-Point DSP Assembly Language Tools User's Guide

Table 1 Periodic task priority

Period	Task	Priority
25 ms	WDT_Reset	0
50 ms	NIC_Input	1
	COOLANT	2
	NIC_Output	7
100 ms	CRPOS	3
	CHECK	4
1 sec	POWER	5
2 sec	THERM	6

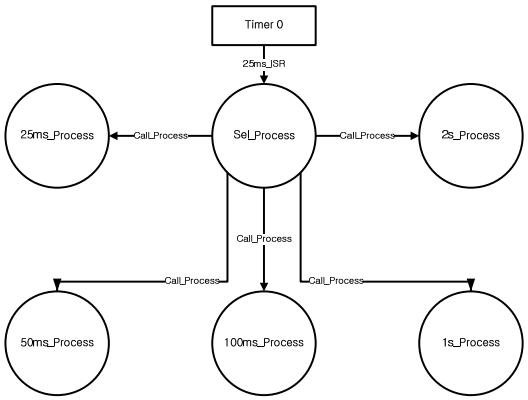

Fig. 1 Sel_Process functional diagram

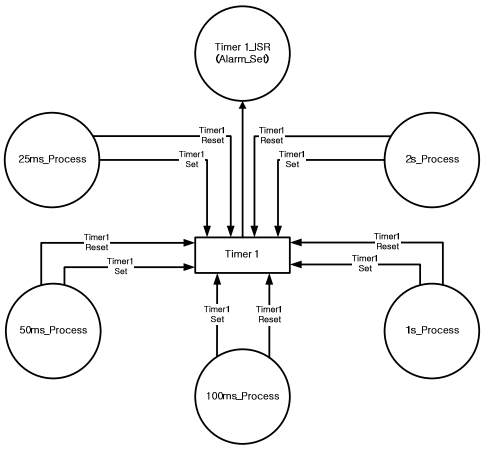

Fig. 2 The task monitoring functional diagram

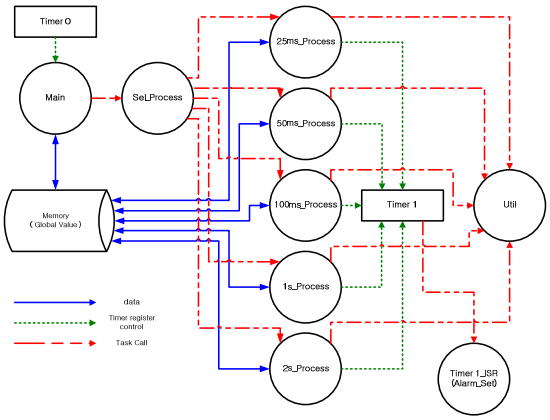

Fig. 3 The scheduler schematic diagram