Real-Time Operating System (RTOS) Tasks for Hard Real-Time Systems: A Case Study over pCOS2 for POSAFE-Q NCPU-2Q Module

Hogeun Gimm [a*], Sungjae Hwang [a], Myeongkyun Lee [a], Jaewon Yun [a]
[a]*Soosan ENS Co., Ltd., Suseo HYUNDAI Venture Ville, 713 Suseo-Dong, Gangnam-Gu, Seoul, 06349, South Korea*
[*]*Corresponding author: nwoleader1@soosan.co.kr*

1. Introduction

As part of the development of the safety grade PLC (Programmable Logic Controller) called POSAFE-Q PLC, Soosan ENS implemented µC/OS-II-based pCOS2 as safety critical software for NCPU-2Q Module, processor module accepted by POSAFE-Q PLC. pCOS2 Tasks (System tasks and application tasks) play a central role in the operation of NCPU-2Q Module.

This paper describes pCOS2 tasks, focusing on their functions and state transition.

2. Real-Time System and Task Classifications

2.1 Real-Time System Classification

RTOSes have different timing constraints by which any particular job needs to be completed around a particular period of time. Unlike soft real-time systems where missing deadlines is tolerable to some extent, hard real-time systems should guarantee to meet deadlines. [1] That is, in hard-real time systems, run-to-completion of tasks within deadlines must be guaranteed. As a safety grade PLC, POSAFE-Q PLC is a hard real-time system.

2.2 pCOS2 Task Classifications

pCOS2 tasks consist of system tasks and application tasks. Application tasks are designed to do particular jobs by users in pSET-II installed on host PC. All pCOS2 tasks except application tasks are called system tasks.

pCOS2 tasks also include loop tasks running in infinite loop and carrying out the majority of the work needed to operate NCPU-2Q Module [2].

3. System Tasks and Application Tasks in pCOS2

In pCOS2, multiple different tasks do not have the same priority and preemptive priority-based scheduling is accepted [3].

As a unique RTC (Run-To-Completion) code, Startup Code[1] runs only once, aiming at pCOS2 initialization and startup after the POSAFFE-Q PLC is powered on [2]. Startup Code creates Display Task to run maximum 8 application tasks and most system tasks. Startup Code also activates Statistics and Idle Tasks.

System tasks activated by Display Task include two Dualization Tasks, two Serial Communication Tasks and Diagnosis Task.

System tasks and application tasks in pCOS2 are specified as follows.

- **Diagnosis Task**
  - **Classification**: System task & Loop task
  - **Functions**: Diagnosis Task detects POSAFE-Q PLC errors, writes details about PLC errors to DPRAM within NCPU-2Q Module and triggers fail-safe operation (ie, stops application tasks running and resets every PLC Module except NCPU-2Q and NSPS-2Q Modules). Diagnosis Task also performs I/O with Digital I/O (DI/DO) Modules, Analog I/O (AI/AO) Modules and Pulse Modules if any application task is not running in master NCPU-2Q Module.

- **Display Task**
  - **Classification**: System task & Loop task
  - **Functions**: Display Task activates two Dualization Tasks, two Serial Communication Tasks and Diagnosis Task. Application tasks are activated by Display Task only if NCPU-2Q Module is operated under Hot Boot mode[2]. Display Task also displays current POSAFE-Q PLC state via NCPU-2Q Module's external LEDs (RUN, ACS, ERR) and resets watchdog timer.

- **Serial Communication Task 1**
  - **Classification**: System task & Loop task
  - **Functions**: Serial Communication Task 1 stores a request message transmitted by a pSET-II in SRAM within master NCPU-2Q Module.

- **Serial Communication Task 2**
  - **Classification**: System task & Loop task
  - **Functions**: Serial Communication Task 2 checks over a request message from a pSET-II for CRC error and appropriately handles it. Serial Communication Task 2 then sends a response message with Result Code (Success/Failure) to the pSET-II.

[1] Startup Code is a function, not a task.

[2] Unlike Cold Boot mode, Hot Boot mode enables NCPU-2Q Module to start its operation with downloaded or running application tasks undeleted.

- **Dualization Task 1**
  - **Classification**: System task & Loop task
  - **Functions**: When main (master) NCPU-2Q Module has heartbeat errors, memory errors, user task running status errors or CPU reference clock errors, Dualization Task 1 enables redundant (slave) NCPU-2Q Module to become master NCPU-2Q Module. This work is called switchover.

- **Dualization Task 2**
  - **Classification**: System task & Loop task
  - **Functions**: At the request of slave NCPU-2Q Module during the initialization of slave NCPU-2Q Module, Dualization Task 2 running in master NCPU-2Q Module copies DPRAM contents (ie, current POSAFE-Q PLC state, and program unit and I/O variables of application tasks), flash memory contents (ie, compressed application task codes) and NvRAM contents (ie, PLC setting information) into slave CPU-2Q Module's DPRAM.

- **Statistics Task**
  - **Classification**: System task & Loop task
  - **Functions**: Statistics Task computes a proportion of Idle Task execution for a predefined period of time. This proportion is referred to as CPU utilization (unit: %). Statistics Task is one of basic tasks along with Idle Task in µC/OS-II.

- **Idle Task**
  - **Classification**: System task & Loop task
  - **Functions**: Idle Task runs in infinite loop until any task is running. Idle Task is used in order to compute CPU utilization in Statistics Task. Idle Task is a basic task in RTOSes.

- **Application Task**
  - **Classification**: Application task & Loop task
  - **Functions**: After maximum 8 application tasks are downloaded to master NCPU-2Q Module, then they run to do jobs intended by users. Users must program application tasks in pSET-II so that less scantime (also called execution period) an application task has, higher priority the task has.

4. pCOS2 Task State Transition

A task in running state might be running on CPU, while other tasks lie in ready state, blocked state (also called waiting state) or interrupted state (also called ISR running state) [3].[3]

If a currently running task is blocked by sleep or message receipt function call and the next highest priority task is in ready state, pCOS2 kernel saves the running task's context in its TCB (Task Control Block) and switches to the next highest priority task which will, in turn, have control of CPU. The previously running task lies in blocked state. A task blocked by sleep function invocation does not become ready until a period of time specified as an argument to sleep function passes. A task (ie, Serial Communication Task 1) blocked by message receipt function invocation gets ready when an incoming request message from a pSET-II leads to message transmission function invocation or a timeout specified as an argument to message receipt function expires. [3]

If interrupts are enabled[4] and a running task is interrupted by Timer or Serial Interrupt, CPU jumps to Timer or Serial Interrupt Handler, respectively. Both interrupt handlers service interrupt requests. The interrupted task lies in interrupted state. The interrupted task will be resumed or not, based on whether a higher priority task in ready state exists. In other words, if a higher priority task in ready state exists, when an interrupt handler completes, a context switch performed by pCOS2 kernel makes the higher priority readied task (not necessarily the interrupted task) start executing next. [3]

5. Conclusions

To describe system tasks and application tasks running in pCOS2 for NCPU-2Q Module consisting of POSAFE-Q PLC, this paper specifies their functions and state transition.

REFERENCES

[1] Anand Srinivasan and James H. Anderson, Efficient Scheduling of Soft Real-Time Applications on Multiprocessors, Journal of Embedded Computing, Vol.1, No.2, pp.285-302, 2005.
[2] Qing Li and Caroline Yao, Real-Time Concepts for Embedded Systems, CRC Press, pp.76, 2003.
[3] Jean J. Labrosse, MicroC/OS-II: The Real-Time Kernel, 2nd Edition, CMP Books, 2002.
[4] µC/OS-II Documentation, Kernel Structure, https://doc.micrium.com/display/osiidoc/Kernel+Structure

[3] Dormant state is also defined in µC/OS-II. Dormant state means that corresponding uncreated tasks reside in memory. [3]

[4] Interrupt disabling macro is used to enter critical section of code, and interrupt enabling macro is used to exit critical section of code. This macro set is basically provided by µC/OS-II, and useful for exclusive access to critical sections of code. [4]