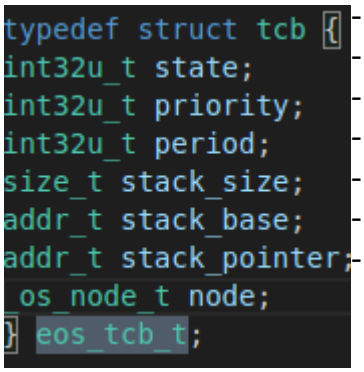
OS_Project 3

2014-1995 송원재

1. eos_tcb_t 구조체



- state : task 의 state (READY , RUNNING, WAITING)
- priority : task 의 priority
- period : task 의 period
- stack_size : task 에 할당된 stack_size
- stack_base : task 에 할당된 stack 의 base 주소
- stack_pointer : task restore 할 때 돌아갈 stack 주소
- node : task 의 node

2. eos_schedule() 함수

-현재 task 가 NULL 이 아닐경우 save_context() call 하고 해당 task 가 RUNNING 상태였다면 ready queue 와 ready_table 에 다시 넣어줌

- os_get_highest_priority 함수를 call 하여 수행할 다음 task 를 확인.
- 해당 task 를 ready_queue 에서 제거하고 해당 task 의 priority ready_queue 가 비었다면 unset_ready call
- 해당 task 의 state 를 RUNNING 으로 바꾸고 restore_task() call

3. eos_create_task() 함수

- 기존 기느에서 state 를 READY 로 바꾸고 ready_queue 에 해당 task 를 넣는 과정추가

4. eos_set_period() 함수

- task->period 를 주어진 period 로 설정

5. eos_sleep() 함수

- 새로운 eos_alarm_t 구조체 선언 (새롭게 설정할 alarm 을 위한 구조체)
- system_timer counter 에 task 의 period 에 맞게 eos_set_alarm() call
- 현재 task 의 state 를 WAITING 으로 변경
- eos_shedule() call

6. os_wakeup_sleeping_task() 함수

- 해당 task 의 state 를 READY 로 바꿈.
- 해당 task 를 ready_queue 에 넣고 ready_table 에 표기

7. eos_set_alarm() 함수

- 해당 alarm 을 counter 의 alarm_queue 에서 제거
- alarm 구조체의 member variable 에 조건 값 넣기(timeout(call back 되는 time), handler, entry, alarm_node.ptr_data(alarm 구조체의 pointer), alarm_node 의 priority(남은시간 → timeout 을 넣으면 같은 효과를 볼 수 있다.)
- counter 의 alarm_queue 에 priority 순으로 add_node

8. eos_trigger_counter() 함수

- counter 의 tick++
- counter 의 alarm_queue 에서 울려야 하는 모든 alarm 의 콜백함수를 호출

(이 때 alarm_queue 가 남은시간 순서로 정렬되어 있기때문에 앞에서부터 timeout 되는 alarm 만 call back 함수 호출하고 아직 시간이 남은 alarm 나오면 break 해서 loop 탈출)

- int32u_t alarm_run 변수를 통해 울리는 alarm 이 있는지 확인한 후에 alarm 이 울린 게 있다면 eos_shedule() call (항상 eos_schedule() 함수 call 하면 overhead 예상)

10. 테스트 프로그램 수행 결과