Real-Time Operating System
(Day 3 Lab)

Jong-Chan Kim

Graduate School of Automotive Engineering

KMU 국민대학교
KOOKMIN UNIVERSITY

17-1. Loss (공유 자원 data loss 확인)

- Task1: Low priority, AUTOSTART
- Task2: High priority, 1 ms 주기 실행
- 공유 전역 변수 (`volatile unsigned long shared`)
  - Task1의 루프에서 `shared++`
  - Task2 주기적으로 반복 실행하며 `shared ++`
- 양 쪽에서 더한 숫자가 모두 유지되는지 확인
- Task와 ISR 사이에서도 같은 문제가 발생하는지 확인

17-1. Loss

bsw.c

```
int main(void)
{
    osEE_tc_stm_set_clockpersec();
    osEE_tc_stm_set_sr0(1000U, 1U);
```

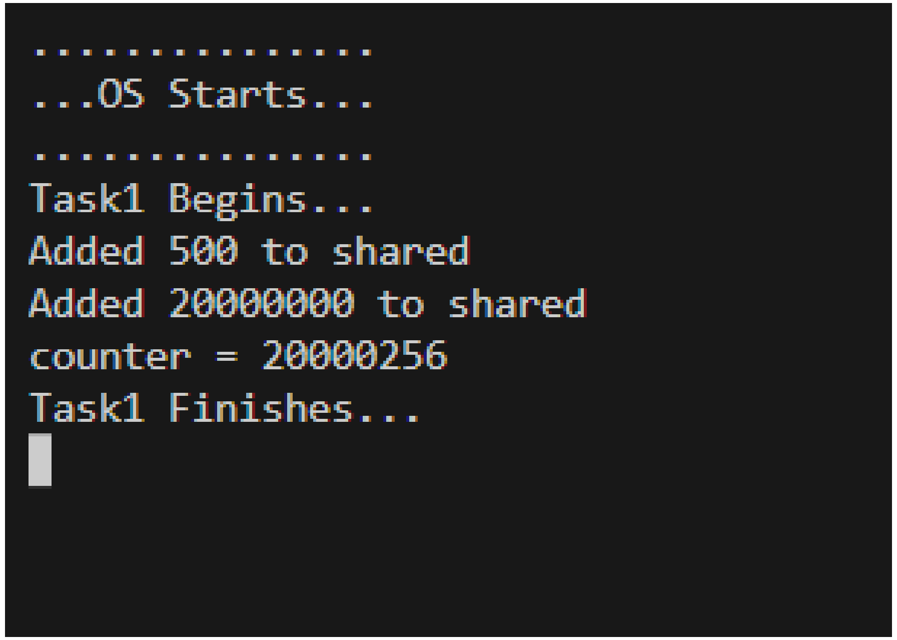
```
#include "bsw.h"
volatile unsigned long shared = 0;
```

```
TASK(Task1)
{
    unsigned long i;
    printfSerial("Task1 Begins...\n");
    for (i = 0; i < 20000000; i++) {
        shared++;
    }
    printfSerial("Added 20000000 to shared\n");
    printfSerial("counter = %lu\n", shared);
    printfSerial("Task1 Finishes...\n");
    TerminateTask();
}
```

```
TASK(Task2)
{
    static unsigned long i = 0;
    if (i < 500) {
        shared++;
    } else if (i == 500) {
        printfSerial("Added 500 to shared\n");
    }
    i++;
    TerminateTask();
}
```

17-1. Loss

- Resource를 이용해서 Integrity Loss 문제 해결 필요



```
...............
...OS Starts...
...............
Task1 Begins...
Added 500 to shared
Added 20000000 to shared
counter = 20000256
Task1 Finishes...
```

20000000 + 500 = 20000256(!!!)

17-2. No Loss

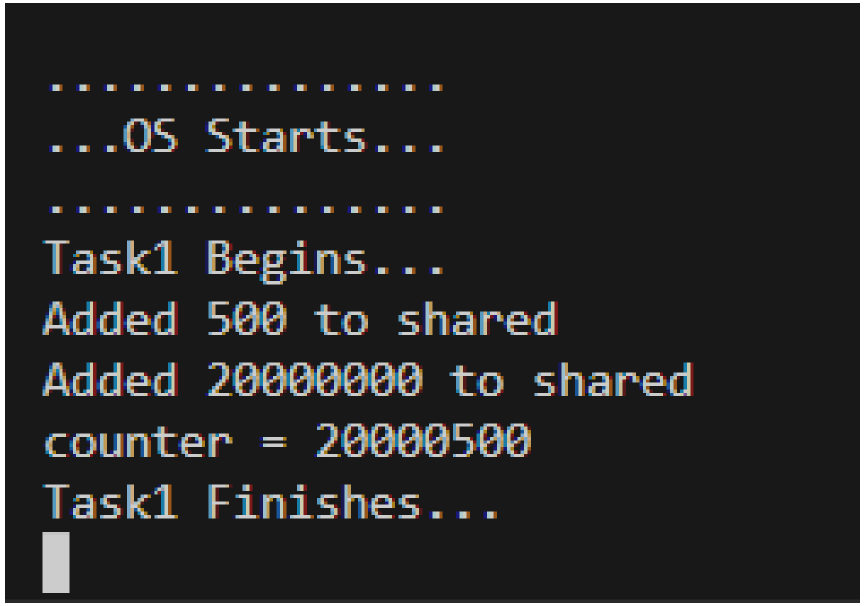
- OSEK의 RESOURCE 기능을 이용하여 Data Loss 문제 해결

```
GetResource(S1);
shared++;
ReleaseResource(S1);
```

```
RESOURCE S1 {
    RESOURCEPROPERTY = STANDARD;
};

…

TASK Task1 {
    PRIORITY = 1;
    STACK = SHARED;
    SCHEDULE = FULL;
    AUTOSTART = TRUE;
    ACTIVATION = 1;
    RESOURCE = S1;
};
```

17-2. No Loss

- Data Integrity 문제 해결



```
................
...OS Starts...
................
Task1 Begins...
Added 500 to shared
Added 20000000 to shared
counter = 20000500
Task1 Finishes...
```

20000000 + 500 = 20000500(!!!)

18. Mutex

- mutex.c

Waiting/Wakeup 을 위해 Event 지정 필요

```c
#include "ee.h"
#include "bsw.h"
#include "mutex.h"

void InitMutex(MutexType *mutex, EventMaskType event)
{
    mutex->flag = UNLOCKED;
    mutex->waiting_task = 0;
    mutex->event = event;
}

void GetMutex(MutexType *mutex)
{
    if (mutex->flag == LOCKED) {
        printfSerial("  -->  BLock");
        GetTaskID(&(mutex->waiting_task));
        WaitEvent(mutex->event);
    }
    mutex->flag = LOCKED;
}
```

```c
void ReleaseMutex(MutexType *mutex)
{
    if (mutex->flag == LOCKED) {
        mutex->flag = UNLOCKED;
        if (mutex->waiting_task != 0) {
            SetEvent(mutex->waiting_task, mutex->event);
        }
    }
}
```

PCP 없이 Mutex 사용할 경우 문제점을 확인하기 위한 Dummy 구현

18. Mutex

- mutex.h

```
#ifndef MUTEX_H_
#define MUTEX_H_

#define LOCKED   1
#define UNLOCKED  0

typedef struct _MutexType {
    int flag;
    EventMaskType event;
    TaskType waiting_task;
} MutexType;

void InitMutex(MutexType *mutex, EventMaskType event);

void GetMutex(MutexType *mutex);

void ReleaseMutex(MutexType *mutex);


#endif /* MUTEX_H_ */
```

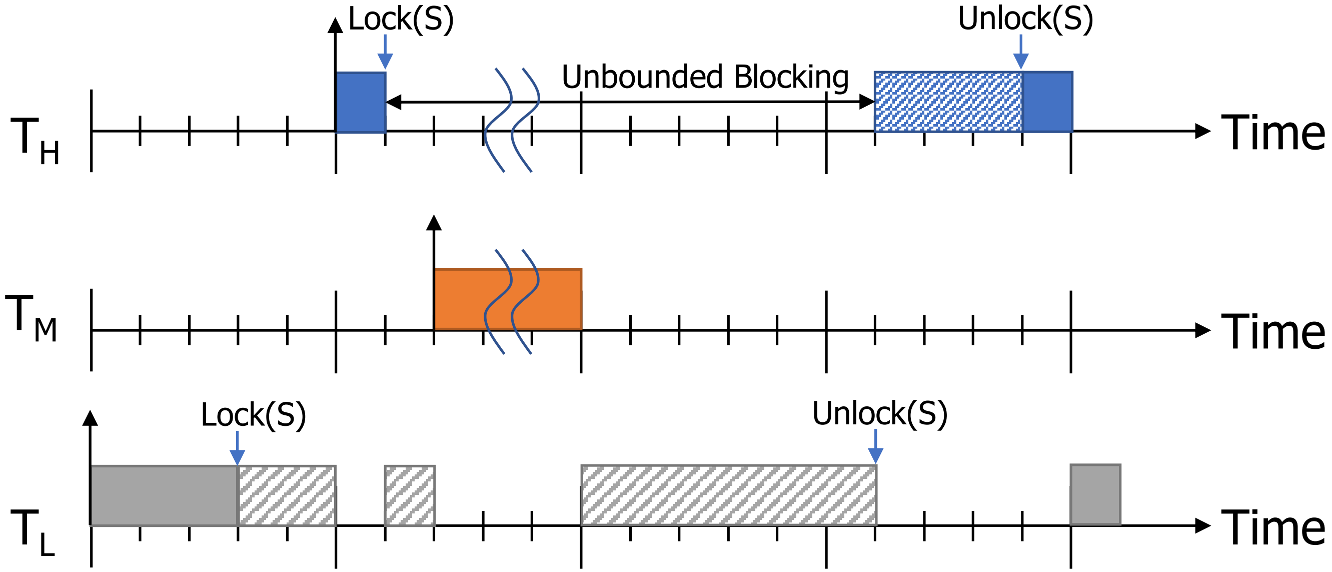
18. Mutex

- Mutex 선언
- Timer ISR 이용
  - Mutex 초기화
  - Task Activation
- Mutex의 동작 확인

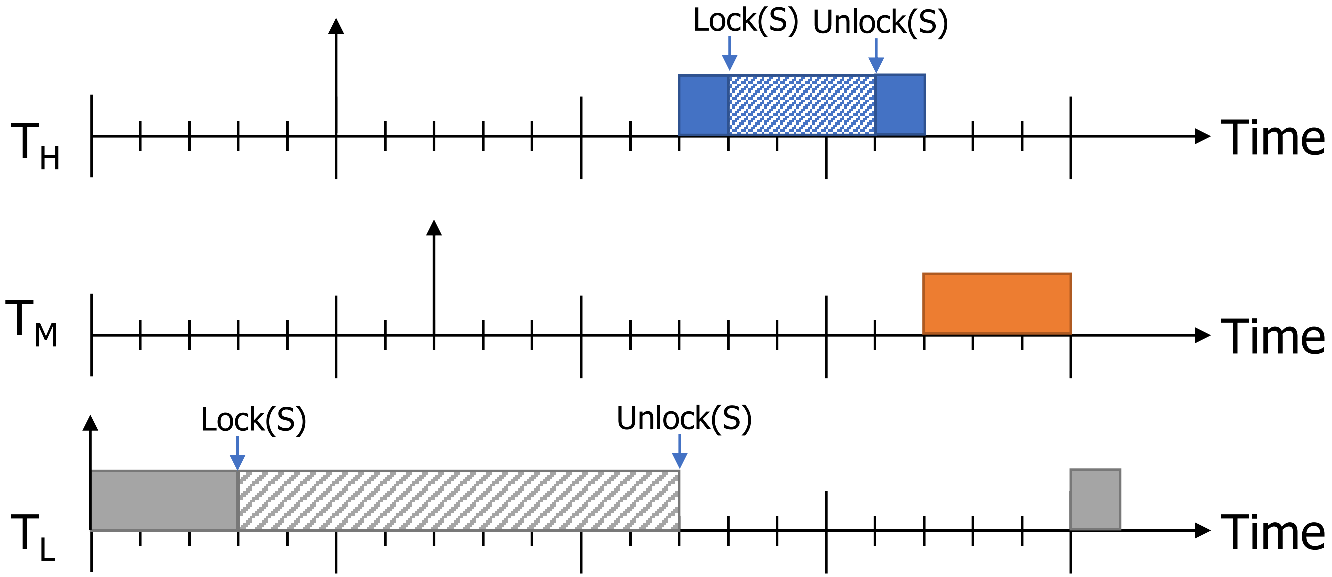
```
MutexType s1;


ISR2(TimerISR)
{
    osEE_tc_stm_set_sr0_next_match(1000000U);
    static long c = -5;
    printfSerial("\n%4ld: ", ++c);
    if(c == -4) {
        InitMutex(&s1, Event1);
    } else if (c == 0) {
        ActivateTask(TaskL);
    } else if (c == 5) {
        ActivateTask(TaskH);
    }
}
```

19-1. Priority Inversion

- 아래 스케쥴 재현하기
- PCP가 적용된 RESOURCE를 이용하여 스케쥴 변화 확인

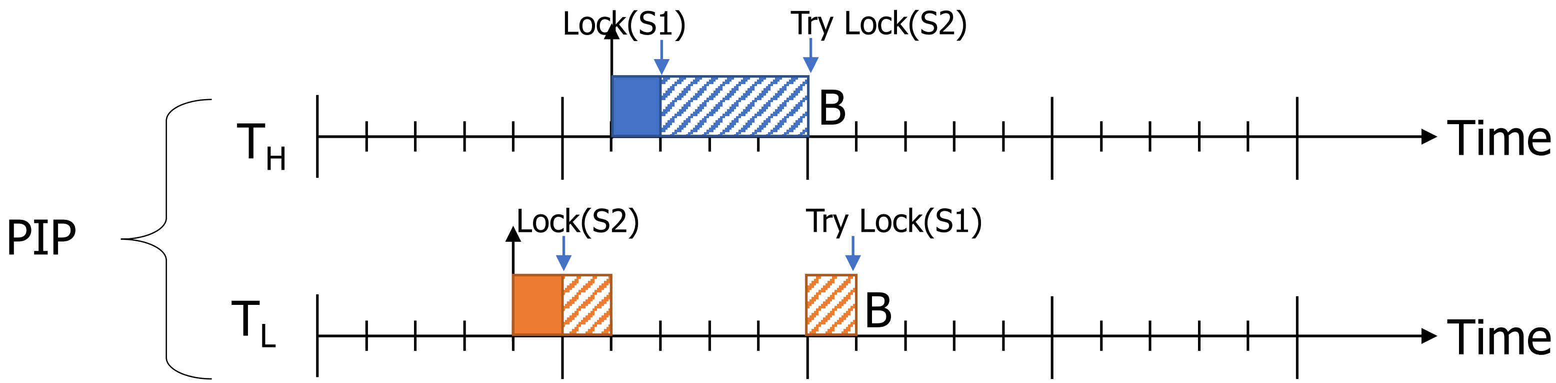


19-2. No Priority Inversion

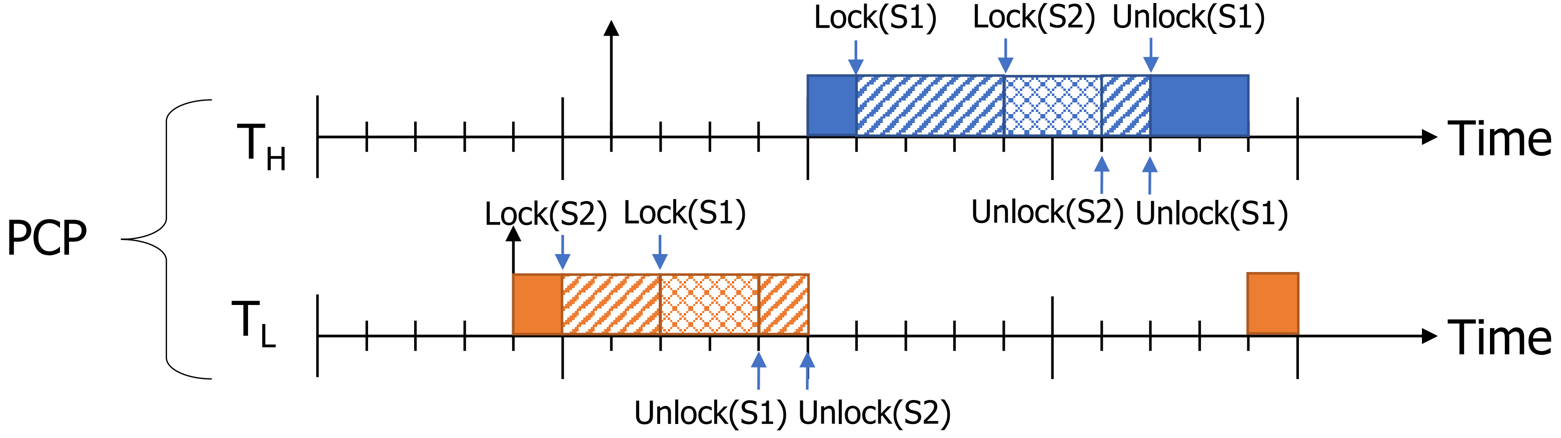


20-1. Deadlock

- Deadlock 재현하기
- PCP가 적용된 RESOURCE를 이용하여 Deadlock 해결

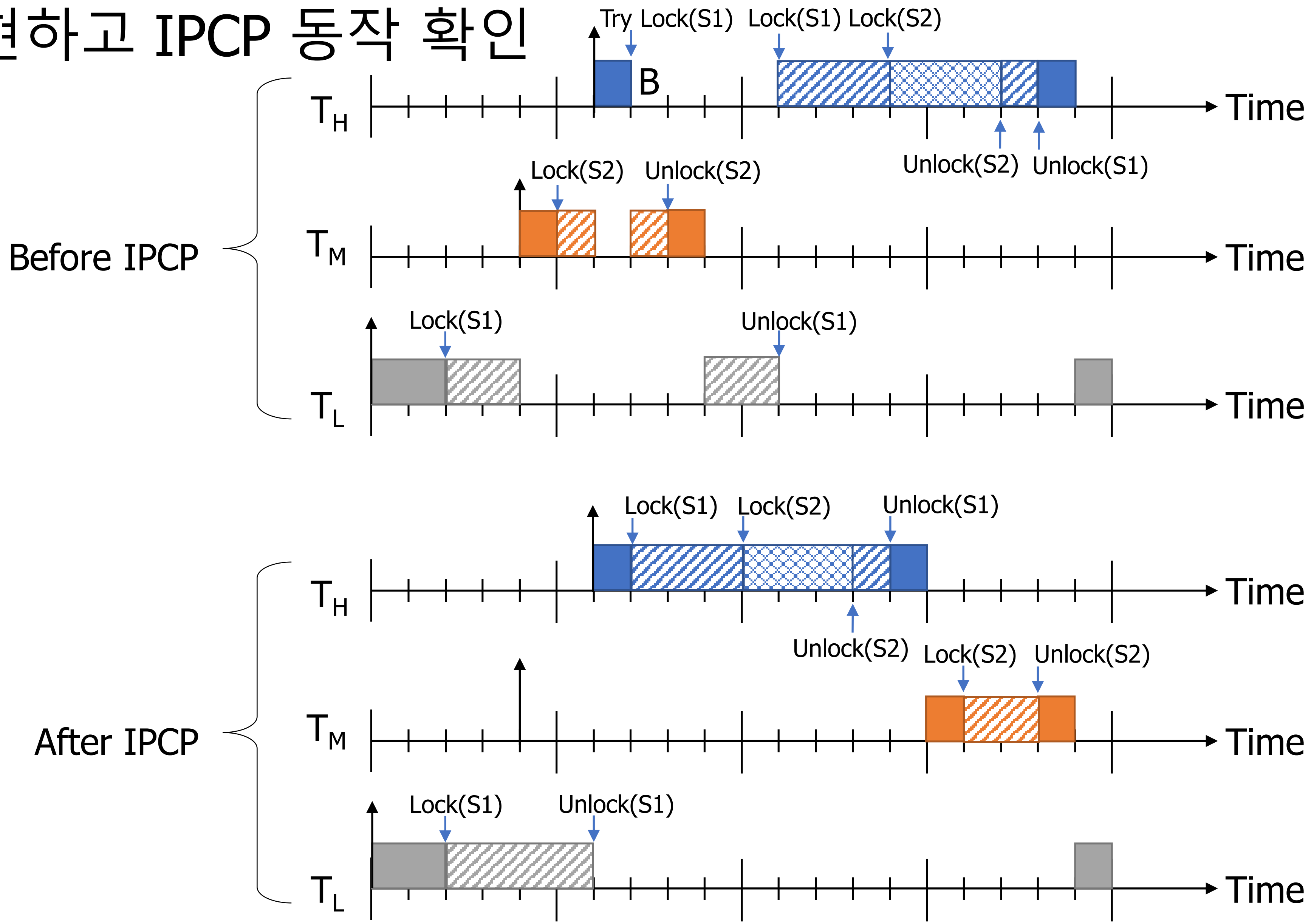


20-2. No Deadlock



21. Before/After IPCP

- 아래 스케쥴 재현하고 IPCP 동작 확인



Questions