Discovery of the cause

Task_smp.c

```
/*↓
 *  ======== Task_sleep ========↓
 */↓
Void Task_sleep(UInt32 timeout)↓
{↓
    Task_PendElem elem;↓
    UInt hwiKey, tskKey;↓
    Clock_Struct clockStruct;↓
↓
    if (timeout == BIOS_NO_WAIT) {↓
        return;↓
    }↓
↓
    Assert_isTrue((timeout != BIOS_WAIT_FOREVER),↓
                  Task_A_badTimeout);↓
↓
    /*↓
     * BIOS_clockEnabled check is here to eliminate Clock module↓
     * references in the custom library↓
     */↓
    if (BIOS_clockEnabled) {↓
        /* add Clock event */↓
        Clock_addI(Clock_handle(&clockStruct), (Clock_FuncPtr)Task_sleepTimeout, timeout, (UArg)&elem);↓
        elem.clockHandle = Clock_handle(&clockStruct);↓
    }↓
↓
    /* MISRA.CAST.FUNC_PTR.2012 MISRA.ETYPE.INAPPR.OPERAND.BINOP.2012 */↓
    Log_write3(Task_LM_sleep, (UArg)Task_self(), (UArg)(Task_self()->fxn),↓
               (UArg)timeout);↓
↓
    hwiKey = Hwi_disable();↓
↓
```

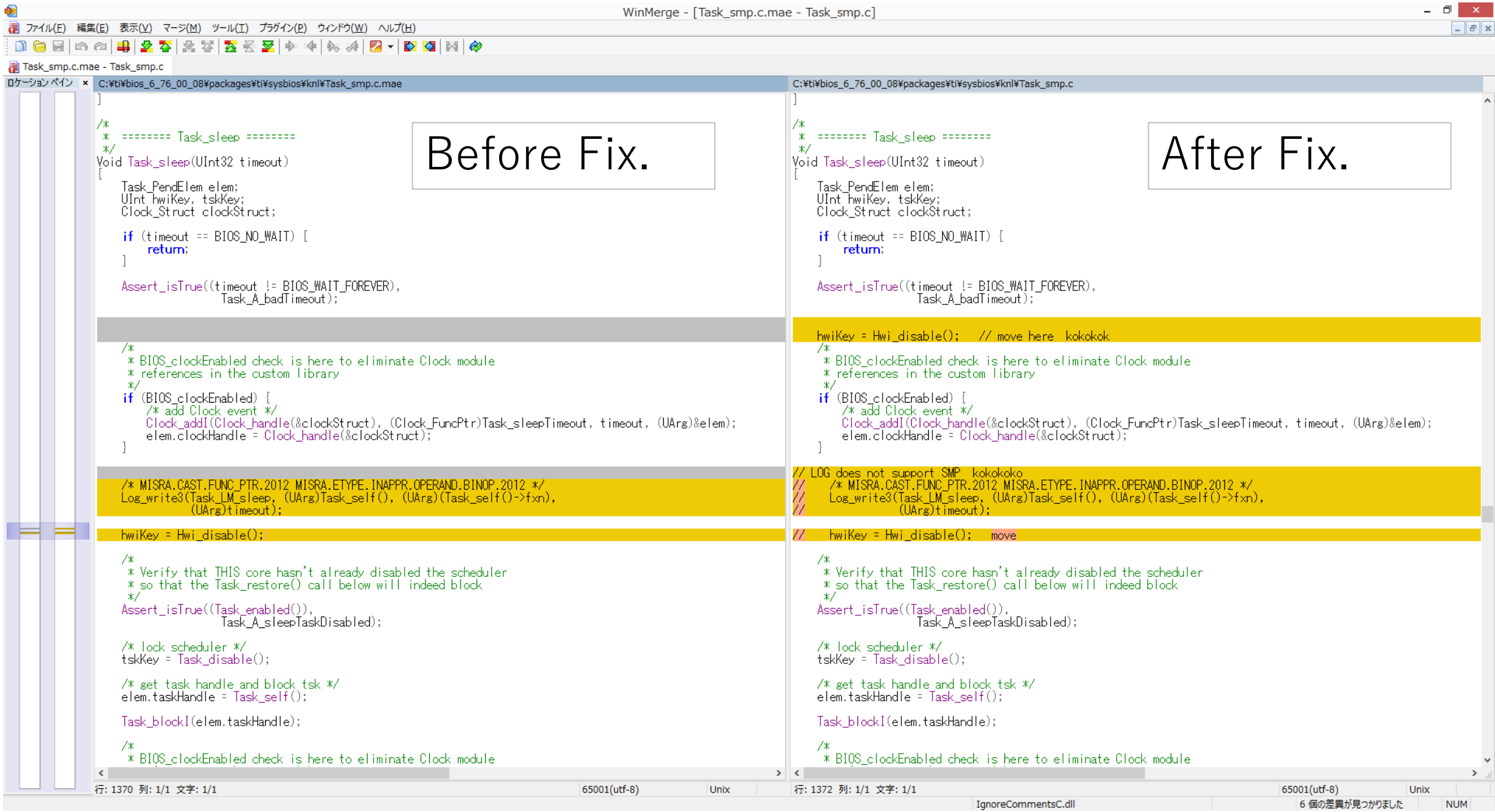
If task dispatch occured and Task_destruct ->construct are executed in this section, elem is filled with 0xBE.

Workaround

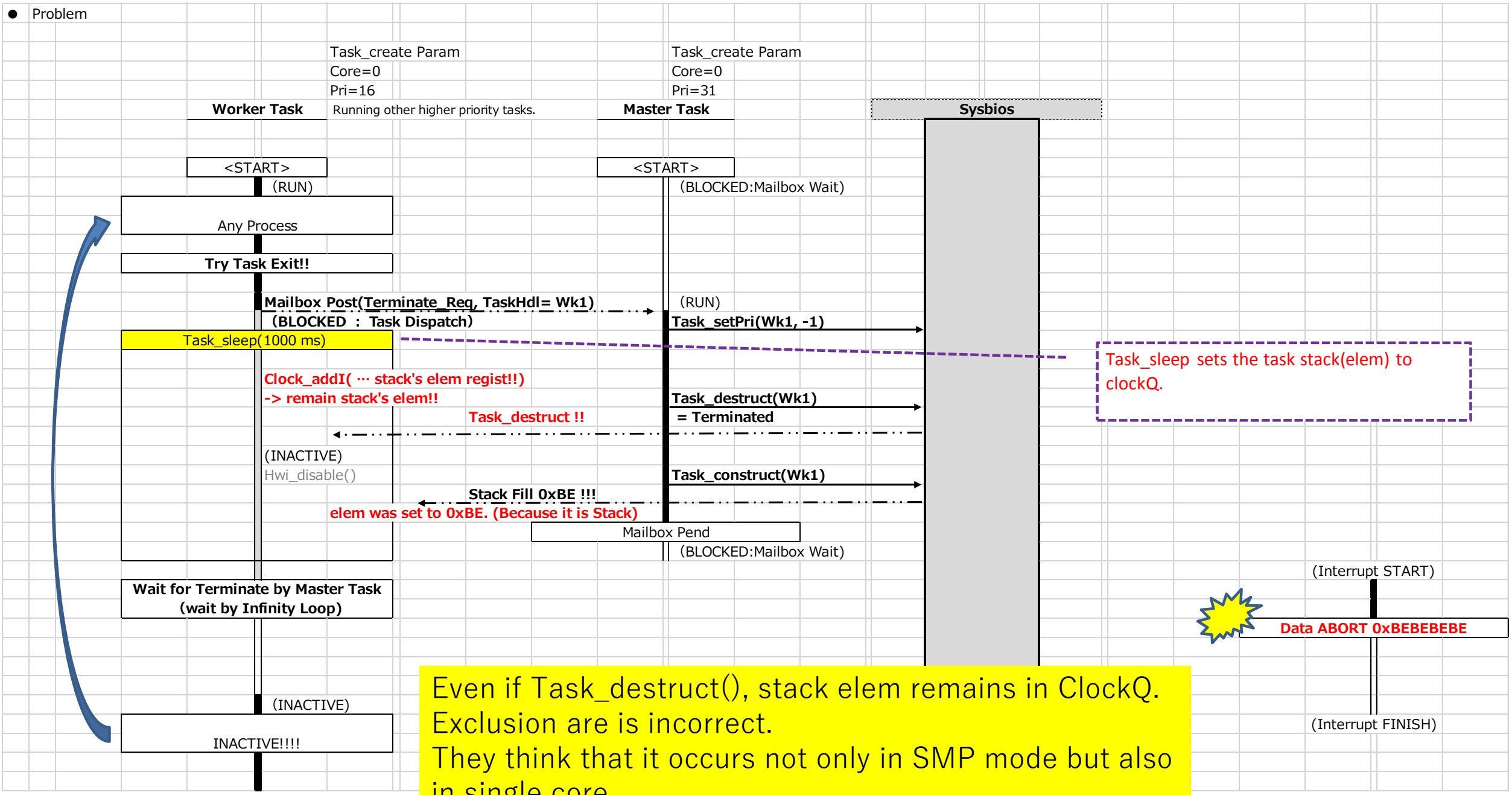
Task_smp.c

```
/*↓
 *  ======== Task_sleep ========↓
 */↓
Void Task_sleep(UInt32 timeout)↓
{↓
    Task_PendElem elem;↓
    UInt hwiKey, tskKey;↓
    Clock_Struct clockStruct;↓
↓
    if (timeout == BIOS_NO_WAIT) {↓
        return;↓
    }↓
↓
    Assert_isTrue((timeout != BIOS_WAIT_FOREVER),↓
                  Task_A_badTimeout);↓
↓
    hwiKey = Hwi_disable();   // move here  kokokok↓
    /*↓
     * BIOS_clockEnabled check is here to eliminate Clock module↓
     * references in the custom library↓
     */↓
    if (BIOS_clockEnabled) {↓
        /* add Clock event */↓
        Clock_addI(Clock_handle(&clockStruct), (Clock_FuncPtr)Task_sleepTimeout, timeout, (UArg)&elem);↓
        elem.clockHandle = Clock_handle(&clockStruct);↓
    }↓
↓
// LOG does not support SMP  kokokoko↓
//    /* MISRA.CAST.FUNC_PTR.2012 MISRA.ETYPE.INAPPR.OPERAND.BINOP.2012 */↓
//    Log_write3(Task_LM_sleep, (UArg)Task_self(), (UArg)(Task_self()->fxn),↓
//               (UArg)timeout);↓
↓
//    hwiKey = Hwi_disable();   move↓
↓
```

They moved Hwi_disable() to forward


WinMerge - [Task_smp.c.mae - Task_smp.c]
ファイル(F) 編集(E) 表示(V) マージ(M) ツール(T) プラグイン(P) ウィンドウ(W) ヘルプ(H)
Task_smp.c.mae - Task_smp.c
ロケーション ペイン
C:¥ti¥bios_6_76_00_08¥packages¥ti¥sysbios¥knl¥Task_smp.c.mae
C:¥ti¥bios_6_76_00_08¥packages¥ti¥sysbios¥knl¥Task_smp.c
Before Fix.
After Fix.
}
/*
 * ======== Task_sleep ========
 */
Void Task_sleep(UInt32 timeout)
{
Task_PendElem elem;
UInt hwiKey, tskKey;
Clock_Struct clockStruct;
if (timeout == BIOS_NO_WAIT) {
return;
}
Assert_isTrue((timeout != BIOS_WAIT_FOREVER),
Task_A_badTimeout);
/*
* BIOS_clockEnabled check is here to eliminate Clock module
* references in the custom library
*/
if (BIOS_clockEnabled) {
/* add Clock event */
Clock_addI(Clock_handle(&clockStruct), (Clock_FuncPtr)Task_sleepTimeout, timeout, (UArg)&elem);
elem.clockHandle = Clock_handle(&clockStruct);
}
/* MISRA.CAST.FUNC_PTR.2012 MISRA.ETYPE.INAPPR.OPERAND.BINOP.2012 */
Log_write3(Task_LM_sleep, (UArg)Task_self(), (UArg)(Task_self()->fxn),
(UArg)timeout);
hwiKey = Hwi_disable();
/*
* Verify that THIS core hasn't already disabled the scheduler
* so that the Task_restore() call below will indeed block
*/
Assert_isTrue((Task_enabled()),
Task_A_sleepTaskDisabled);
/* lock scheduler */
tskKey = Task_disable();
/* get task handle and block tsk */
elem.taskHandle = Task_self();
Task_blockI(elem.taskHandle);
/*
* BIOS_clockEnabled check is here to eliminate Clock module
hwiKey = Hwi_disable(); // move here kokokok
// LOG does not support SMP kokokoko
// /* MISRA.CAST.FUNC_PTR.2012 MISRA.ETYPE.INAPPR.OPERAND.BINOP.2012 */
// Log_write3(Task_LM_sleep, (UArg)Task_self(), (UArg)(Task_self()->fxn),
// (UArg)timeout);
// hwiKey = Hwi_disable(); move
行: 1370 列: 1/1 文字: 1/1
65001(utf-8)
Unix
行: 1372 列: 1/1 文字: 1/1
65001(utf-8)
Unix
IgnoreCommentsC.dll
6 個の差異が見つかりました
NUM

● Problem



Even if Task_destruct(), stack elem remains in ClockQ.
Exclusion are is incorrect.
They think that it occurs not only in SMP mode but also in single core.