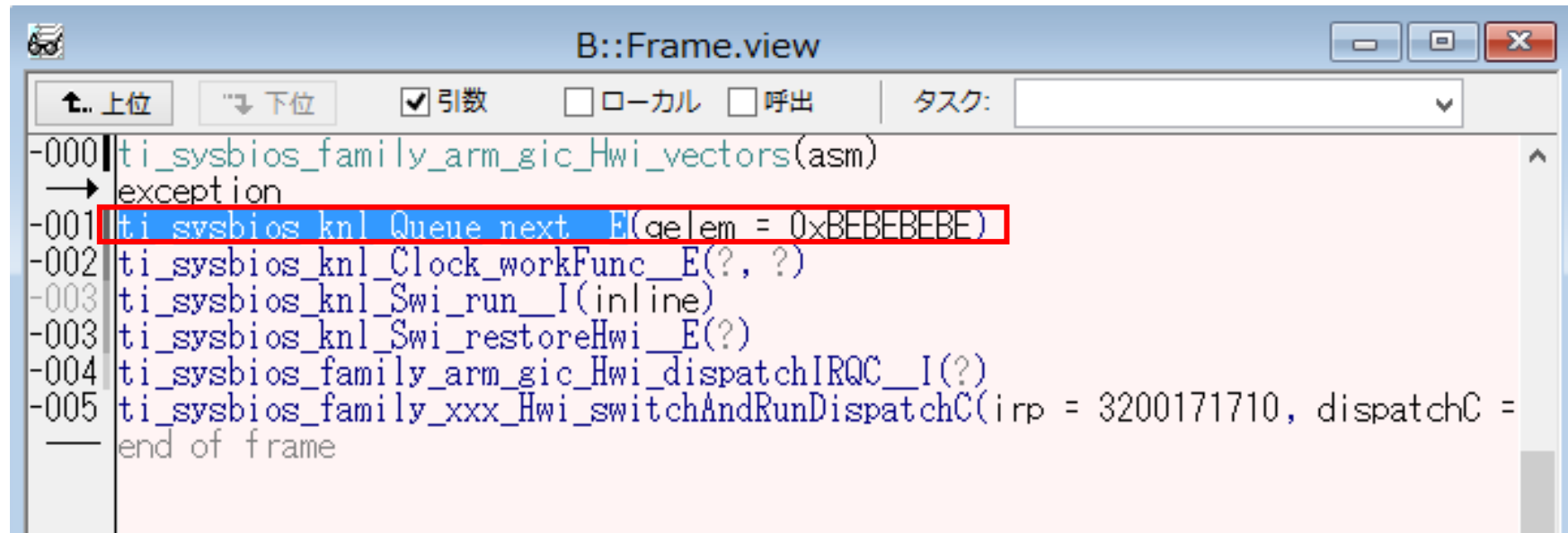
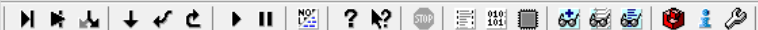


“Data abort” have occurred

Clock object's qelem = 0xBEBEBEBE



```
B::Frame.view
┌.. 上位  ──> 下位   引数   ローカル   呼出  タスク:
-000 | ti_sysbios_family_arm_gic_Hwi_vectors(asm)
      | ──> exception
-001 | ti_sysbios_knl_Queue_next_E(qelem = 0xBEBEBEBE)
-002 | ti_sysbios_knl_Clock_workFunc_E(?, ?)
-003 | ti_sysbios_knl_Swi_run_I(inline)
-003 | ti_sysbios_knl_Swi_restoreHwi_E(?)
-004 | ti_sysbios_family_arm_gic_Hwi_dispatchIRQC_I(?)
-005 | ti_sysbios_family_xxx_Hwi_switchAndRunDispatchC(irp = 3200171710, dispatchC =
      | ─── end of frame
```



[B::List.auto]

C:\ti\bios_6_52_00_12\packages\ti\sysbios\knl

Clock.c

B::Fr...

 上位
 下位

```

/*
 * Here count can be zero. When Clock_tick() runs it increments
 * swiCount and posts the Clock_workFunc. In Clock_workFunc we
 * get the value of swiCount atomically. Before we read swiCount, an
 * interrupt could occur, Clock_tick() will post the swi again.
 * That post is unnecessary as we are getting ready to process that
 * tick. The next time this swi runs the count will be zero.
 */

```

```

342 while (count) {
344     compare = compare + 1U;
345     count = count - 1U;

    /* Traverse clock queue */

349     clockQ = Clock_Module_State_clockQ();
350     elem = Queue_head(clockQ);
351     clockQElem = Queue_prev(elem);

```

```

353 while (elem != clockQElem) {
355     // kokokokok
356     if (elem == 0xBEBEBEBE) {
        svBreak++;
    }

```

Their debug code (maybe)

```

587 obj->timeout = (UInt32)timeout;
588 obj->period = params->period;
589 obj->fxn = func;
570 obj->arg = params->arg;
571 obj->active = FALSE;

```

```

/*
 * Clock object is always placed on Clock w

```

B::Var.Watch svHwiTrace svTskTrace_Switch

```

clockQ = 0x8005981C → (
  elem = (
    next = 0x800597D4,
    prev = 0x81D9241C → (
      next = 0x8005981C,
      prev = 0x81E3DDC4 → (
        next = 0x81D9241C,
        prev = 0x81E5D404 → (
          next = 0x81E3DDC4,
          prev = 0x81E3A4FC → (
            next = 0x81E5D404,
            prev = 0x81D2BCA4 → (
              next = 0x81E3A4FC,
              prev = 0x81DDC6AC → (
                next = 0x81D2BCA4,
                prev = 0x81D5A92C → (
                  next = 0x81DDC6AC,
                  prev = 0x81E6CB04 → (
                    next = 0x81D5A92C,
                    prev = 0x81D2212C → (
                      next = 0x81E6CB04,
                      prev = 0x81DE8AF4 → (
                        next = 0xBEBEBEBE,
                        prev = 0xBEBEBEBE))))))))))

```

Qelem = 0xBEBEBEBE

```

elem = (
  next = 0xBEBEBEBE,
  prev = 0xBEBEBEBE,
  timeout = 3200171710,
  currTimeout = 3200171710,
  period = 3200171710,
  active = 48830,
  fxn = 0xBEBEBEBE,
  arg = 3200171710)

```

```

svHwiTrace = ((mNextNum = 126789, mLastNum = 0, mSave = 0, mLastTime = 76369217, mCell = ((mGapTop = 0, mEvent = 0x20000001, mArg = 0x0
svTskTrace_Switch = ((mNextNum = 8602, mLastNum = 409, mSave = 1, mLastTime = 76368880, mCell = ((mGapTop = 0, mEvent = 0x0, mArg = 0x0

```

80089800	-1.	ready	810998EC	TaskApiDefault
80089888	23.	ready	81099D54	TaskApsBase

B::

81D2212C: __stack_Task_xxx

This task use Event_pend(…100ms)
⇒ Polling Ethernet

81DE8AF4: __stack_Task_yyy

This task use Task_sleep(1000ms)
⇒ After processing is done,
it will be Task_destruct().

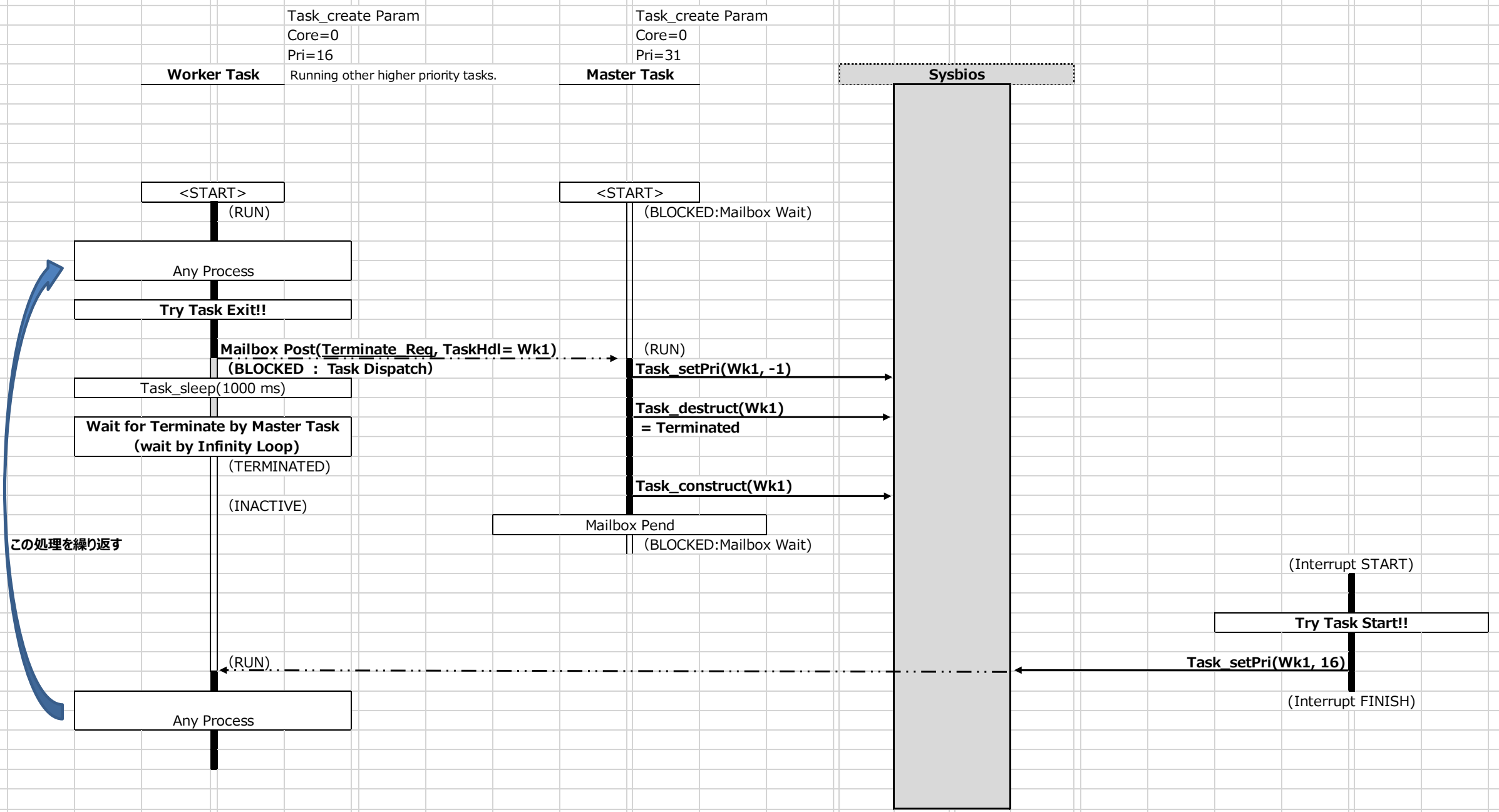
The screenshot shows a debugger window titled "B::Var.Watch svHwiTrace svTskTrace_Switch". The main area displays a tree view of a linked list structure. The root node is "clockQ = 0x8005981C", which points to a list of "elem" nodes. Each "elem" node contains "next" and "prev" pointers to other nodes in the list. A red rectangular box highlights a specific node in the list with the following details:

- next = 0x81D5A92C, → (
- prev = 0x81D2212C → (
- next = 0x81E6CB04, → (
- prev = 0x81DE8AF4 → (
- next = 0xBEBEBEBE,
- prev = 0xBEBEBEBE)))))))))

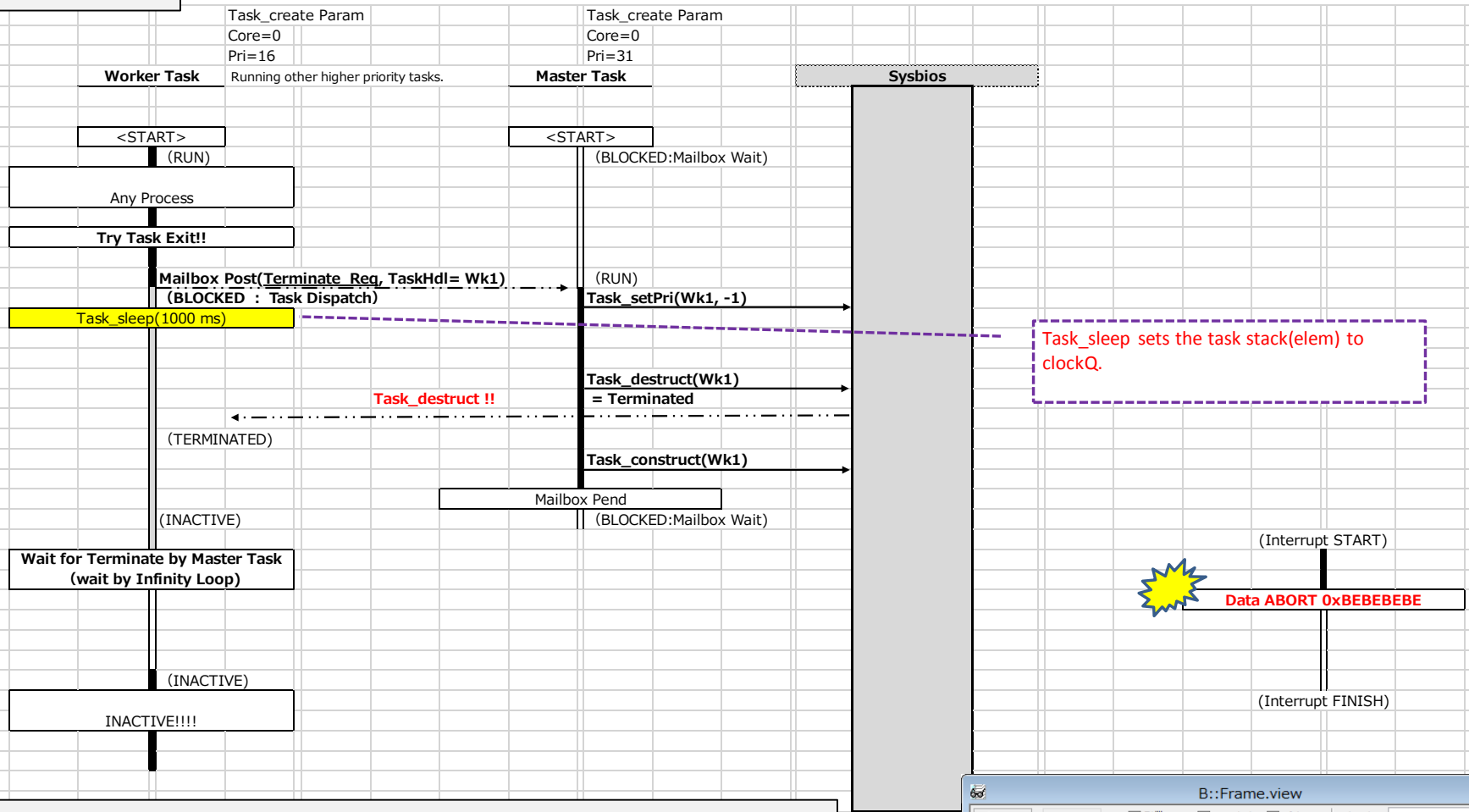
Below the list, other variables are shown, including "obj = 0x81DE8AF4" and "svHwiTrace" and "svTskTrace_Switch". At the bottom, a small table shows task status:

Address	State	Priority	Task Name
80089800	ready	-1	TaskApIDefault
810998EC	ready	-1	TaskApIDefault

Expected operation



Actual operation



Assumptions:
 In Data_WorFunc,
 "Data abort" occurs because elem's address is 0xBEBEBE.
 Task_destruct () does not change the stack address.

```

B::Frame.view
-000|ti_sysbios_family_arm_gic_Hwi_vectors(asm)
    → exception
-001|ti_sysbios_knl_Queue_next_E(qelem = 0xBEBEBEBE)
-002|ti_sysbios_knl_Clock_workFunc_E(?, ?)
-003|ti_sysbios_knl_Swi_run_l(swi = 0x8005947C)
-004|ti_sysbios_knl_Swi_restoreHwi_E(?)
-005|ti_sysbios_family_arm_gic_Hwi_dispatchIRQC_I(?)
-006|ti_sysbios_family_xxx_Hwi_switchAndRunDispatchC(irp = 3200171710, dispatchC
    end of frame
    
```