CUSTOMER NOTIFICATION

SUD-DT-03-0316-1-E

(SUD-TT-0225-1-E revision)

July 31, 2003

Koji Nishibayashi, Senior System Integrator
Microcomputer Group
2nd Solutions Division
Solutions Operations Unit
NEC Electronics Corporation

CP(K), O

IE-V850ESK1-ET (Control Code: A)

Operating Precautions

Be sure to read this document before using the product.

Notes on Using IE-V850ESK1-ET

1. Product Version

Part number: IE-V850ESK1-ET

Control Code	Board Version	EVA Chip
A	1.00	IC19: uPD703191AR DS 4.5 or later IC24: uPD703217GJ ES 1.4 or later

Use the debugger supplied with the IE-V850ESK1-ET.

2. Product History

No.	Bugs and Changes/Additions to Specifications	Control Code [Note]
		A
1	ROM correction function cannot be emulated	Permanent restriction
2	Restriction on use-prohibited area	Permanent restriction
3	Restriction on break timing when guard area is fetched	Permanent restriction
4	ROM contents are rewritten if emulation ROM area is accessed for write.	Permanent restriction
5	Restriction on I/O register illegal break	Permanent restriction
6	Hardware break does not occur even if breakpoint is set.	Permanent restriction
7	Restriction on DBPC and DBPSW access during a break	Permanent restriction
8	Restriction on DBTRAP instructions	Permanent restriction
9	Restriction that the debugger hangs up depending on the software break setting upon PSC register access	Permanent restriction
10	Restriction that the same branch instruction is traced twice	Permanent restriction
11	Restriction on 48-bit length mov instruction trace	Permanent restriction
12	Restriction on illegal trace of consecutive sld instruction	Permanent restriction
13	Restriction on pin mask function	Permanent restriction

Note The “control code” is the second digit from the left in the 10-digit serial number in the warranty supplied with the product you purchased (if it has not been upgraded). If the product has been upgraded, a label indicating the new version is attached to the product and the x in V-UP LEVEL x on this label indicates the control code.

3. Details of Bugs and Additions to Specifications

No.1 ROM correction function cannot be emulated

[Description]

The ROM correction function cannot be emulated.

[Workaround]

There is no workaround. Regard this as a permanent restriction.

No.2 Restriction on use-prohibited area

[Description]

A fail-safe break does not occur even if the program is executed or an access occurs in 0x3FFC000 to 0x3FFDFFF (device with 6 KB internal RAM: 0x3FFC000 to 0x3FFD7FF) in the use-prohibited area of the target device.

[Workaround]

Regard this as a permanent restriction.

A break can be generated when the program is executed or an access occurs by setting a break on the debugger under the following conditions.

♦ Device with 4 KB internal RAM

- Detects program execution at 0x3FFC000 to 0x3FFDFFF
  - Event status: Execution
  - Address: 0x3ffc000 to 0x3ffdfff

  (Two execution events are used by setting the above conditions.)
- Detects access for 0x3FFC000 to 0x3FFDFFF
  - Event status: R/W
  - Access size: No Condition
  - Address: 0x3ffc000 to 0x3ffdfff

  (Two access events are used by setting the above conditions.)

♦ Device with 6 KB internal RAM

- Detects program execution at 0x3FFC000 to 0x3FFD7FF
  - Event status: Execution
  - Address: 0x3ffc000 to 0x3ffd7ff

  (Two execution events are used by setting the above conditions.)
- Detects access for 0x3FFC000 to 0x3FFD7FF
  - Event status: R/W
  - Access size: No Condition
  - Address: 0x3ffc000 to 0x3ffd7ff

  (Two access events are used by setting the above conditions.)

No.3 Restriction on break timing when guard area is fetched

[Description]

When program execution enters the guard area, one instruction in the guard area is executed and then a break occurs.

[Workaround]

There is no workaround. Regard this as a permanent restriction.

No.4 ROM contents are rewritten if emulation ROM area is accessed for write.

[Description]

An illegal break occurs if the emulation ROM area is accessed for write, and the data of ROM is rewritten.

[Workaround]

There is no workaround. Regard this as a permanent restriction.

No.5 Restriction on I/O register illegal break

[Description]

An I/O register illegal break is detected by “address condition + R/W attribute”. However, a break occurs if an 8-bit I/O register is read in halfword units (break does not occur if an 8-bit I/O register is written in halfword units).

[Workaround]

There is no workaround. Regard this as a permanent restriction.

No.6 Hardware break does not occur even if breakpoint is set.

[Description]

- (Dis)assembly level -

If breakpoints are set for two instructions in a row and a break occurs at the first instruction, a break does not occur at the second instruction in response to the subsequent request for resumption of execution.

```
0x80049c  mov   r9, r10          ← Setting of breakpoint
0x80049e  add   r7, r10          ← Setting of breakpoint
0x8004a0  addi  1, r10, r17
```

- Source level -

If breakpoints are set for two execution statements in a row (of which each is expanded for a single instruction) and a break occurs at the first statement, a break may not occur at the second statement in response to the subsequent request for resumption of execution.

```
10  a = b;    (mov r9,r10)       ← Setting of breakpoint
11  a += c;   (add r7,r10)       ← Setting of breakpoint
```

[Cause]

If resumption of execution is requested at the position where program execution is stopped at a breakpoint, the instruction at the breakpoint is internally executed in one instruction step, and execution is resumed.

Some CPUs execute two instructions at a time, depending on the combination of instructions. In the above (dis)assembly level example, execution is resumed from address 08004a0. Therefore, the breakpoint set at address 0x80049e is not hit.

[Combination of instructions for which two instructions are simultaneously executed]

- Conditions in which “mov + operation instruction” are executed as one instruction

If dst of mov and dst of the operation instruction are the same register, except when that register is r0, in the combination “mov src, dst” and the following instruction:

Format I	satsubr/satsub/satadd/mulh
	or/xor/and
	subr/sub/add
Format II	shr/sar/shl/mulh

This combination of instructions is executed as one instruction only when the mov instruction is at the first position.

- Condition for parallel execution of instructions

(1) Combination of following instructions and br

Format I	nop/mov/not/sld
	satsubr/satsub/satadd/mulh
	or/xor/and/tst
	subr/sub/add/cmp
Format II	mov/satadd/add/cmp
	shr/sar/shl/mulh
Format IV	sld.b/sst.b/sld.h/sst.h/sld.w/sst.w

(2) Combination of following instructions (instructions in 1 excluding those that update flags) and bcc instruction except br

Format I	nop/mov/sld*
	mulh/sxb/sxh/zxb/zxh
Format II	mov/mulh
Format IV	sld.b/sst.b/sld.h/sst.h/sld.w/sst.w

(3) Combination of following instructions and sld

Format I	nop/mov/not
	satsubr/satsub/satadd/mulh
	or/xor/and/tst
	subr/sub/add/cmp
Format II	mov/satadd/add/cmp
	shr/sar/shl/mulh

In (1) to (3) above, parallel execution is performed only when br/bcc/sld instruction is at the second position of the above combination of instructions.

In the following cases, parallel execution is not performed even in the above combination.

a) If the first instruction is the first instruction after branch to unaligned word
b) If the second instruction is sld and if writing to the register of ep is not completed (short path is not performed)

<<Example>>
Two instructions are not executed at the same time if instructions are programmed as follows.

```
0x1000      nop
0x1002      nop
0x1004      nop
0x1006      mov r10, ep          ← Setting of breakpoint
0x1008      sld.b 0x8[ep], r11   ← Setting of breakpoint
0x100c      nop
0x100e      nop
```

In this case, the mov instruction at address 0x1006 writes the value of r10 to the ep register. When the sld.b instruction at address 0x1008 is executed, however, WB (writeback) of the mov instruction is not completed and therefore, the two instructions are not executed at the same time.

c) If the second instruction is bcc (conditional branch instruction) and flag hazard occurs
(if there is a possibility that the flag is updated immediate before or by the preceding instruction)

<<Example>>
Two instructions are not executed at the same time if the instructions are programmed as follows.

```
0x1000      nop
0x1002      nop
0x1004      nop
0x1006      cmp r0, r10          ← Setting of breakpoint
0x1008      bn 0xf0              ← Setting of breakpoint
0x100a      nop
0x100c      nop
```

Because the S flag is changed by the cmp instruction at address 0x1006, the bn instruction that references the S flag and branches must wait for execution of the cmp instruction. Consequently, a flag hazard occurs when the bn instruction is executed, and two instructions are not executed at the same time.

d) If sld is used and the load buffer of both the instructions are in the WB wait status

<<Example>>
Suppose the following instructions are located in memory

```
0x1000      nop
0x1002      nop
0x1004      ld.w 0x3000[r10], r11
0x1008      ld.w 0x3004[r10], r12
0x100c      mov r8, r9           ← Setting of breakpoint
0x100e      sld.b 0x10[ep], r13  ← Setting of breakpoint
```

At this time, several wait state clocks are inserted if ld.w at addresses 0x1004 and 0x1008 accesses the external memory. When address 0x100e is executed, therefore, WB of the ld.w instruction at addresses 0x1004 and 0x1008 is not completed and “WB wait” status begins. Consequently, the two instructions at addresses 0x100c and 0x100e are not executed at the same time.

*** Formats I, II, and IV are the instruction formats shown in the User’s Manual (Architecture) of the processor.**

[Workaround]

There is no workaround for hardware breaks. Regard this as permanent restriction.
(This restriction does not apply when setting software breaks.)

No.7 Restriction on DBPC and DBPSW access during a break

[Description]

Write access to DBPC and DBPSW cannot be performed during a break.
(Read access is possible.)

[Workaround]

There is no workaround. Regard this as permanent restriction.

No.8 Restriction on DBTRAP instructions

[Description]

If a break occurs in the interrupt servicing of a DBTRAP instruction that is executed while a user program is running, the DBPC and DBPSW will be read incorrectly by subsequent RUN instructions.

[Workaround]

There is no workaround. Regard this as permanent restriction.

No.9 Restriction that the debugger hangs up depending on the software break setting upon PSC register access

[Description]

The debugger hangs up if the STB bit of the PSC register is set (1) and a software break is set for the subsequent instruction.

[Workaround]

Do not set a software break for the subsequent instruction. The following shows an example.

```
mov 0x2,r1
st.b r1,prcmd
st.b r1,psc
nop     ← The debugger hangs up if a software break is set here.
nop     ← Setting a software break hereafter causes no problem.
```

Regard this as a permanent restriction.

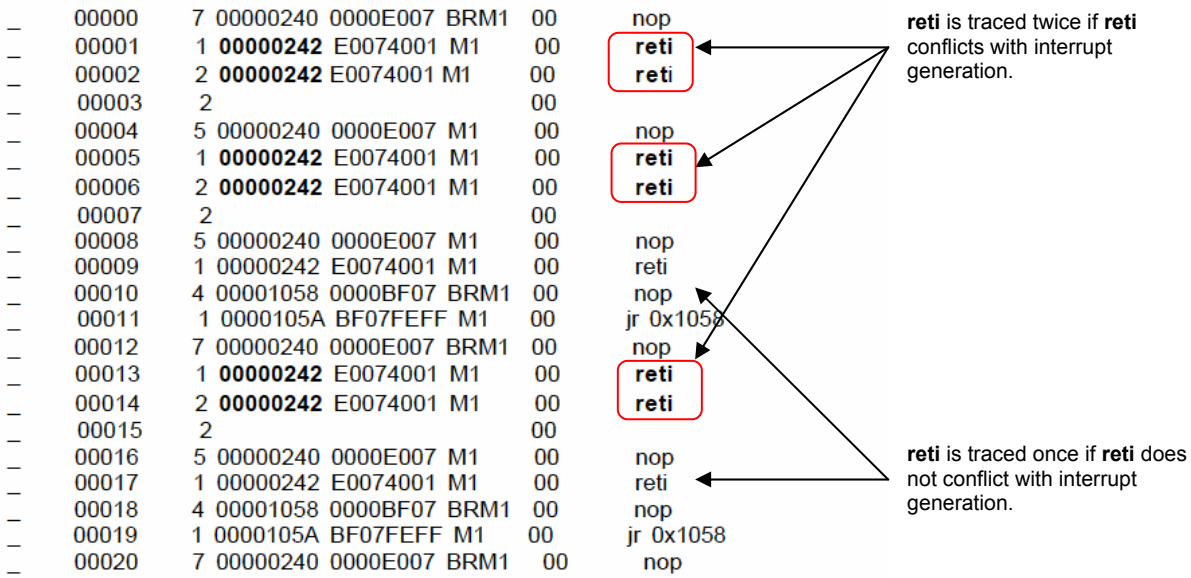
No.10 Restriction that the same branch instruction is traced twice

[Description]

If interrupt generation and execution of a branch instruction (such as JMP, BR, CALLT, or CTRET) conflict, the branch instruction is traced twice.

This restriction affects the trace display only; the actual instruction is executed only once.

[Bug example]



[Workaround]

There is no workaround. Regard this as a permanent restriction.

No.11 Restriction on 48-bit length mov instruction trace

[Description]

If an interrupt occurs at the same time as a 48-bit length mov instruction is executed, the trace result is illegal. At this time, the trace result of the subsequent instruction may also be illegal.

This restriction affects the trace display only; the actual instruction is executed normally.

[Trace result when instruction and interrupt do not conflict (this restriction does not apply)]

```
110  1  000010BE 2F061851   M1                0  mov 0x205118, r15
111  2  000010BE 2F062000                     0
112  1  000010C4 6F070000   M1  205118 0  W   0  st.h r0, 0x0[r15]
113  1  000010C8 BF07EAFF   M1                0  jr 0x10b2
```

[Trace result when instruction and interrupt conflict (this restriction applies)]

```
110  1  000010BE 2F061851   M1                0  ****
111  1  000010BE 2F062000   M1                0  mov 0x20f146, r15
                                                  not r0, r0
112  2                                         0
```

[Workaround]

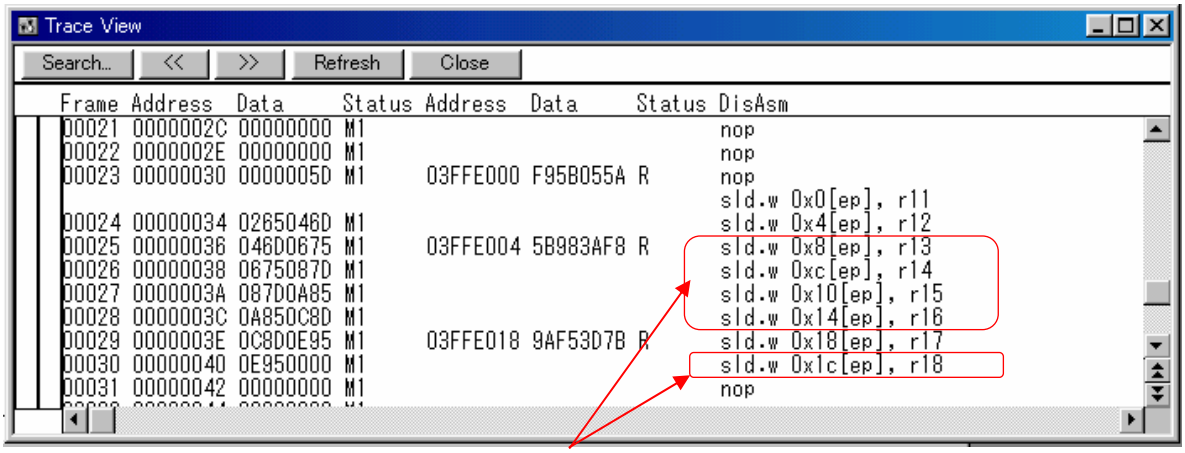
There is no workaround. Regard this as a permanent restriction.

No.12 Restriction on illegal trace of consecutive sld instruction

[Description]

When the sld instruction is executed successively, the access data or access address in the trace data may not be displayed. The disassemble data is displayed normally.

This restriction affects the trace display only; the actual instruction is executed normally.



Instruction for which access address/access data is not displayed

[Workaround]

There is no workaround. Regard this as a permanent restriction.

No.13 Restriction on pin mask function

[Description]

When using the ID850, the WAIT and HLDRQ pins are not masked even if pin masking is set in the MASK setting area in the Configuration dialog box.

When using MULTI, the WAIT and HLDRQ pins are not masked even if pin masking is set using the PINMASK command.

The RESET, STOP, and NMI pins can be masked normally.

[Workaround]

There is no workaround. Regard this as a permanent restriction.

4. Cautions

1) Pin handling

This emulator uses the minimum pin handling, giving priority to compatibility with the device. Take adequate workarounds for static electricity if the emulator is used without the target connected. For details, refer to the user's manual.

2) Power saving

To save power, be sure to insert five NOP instructions after executing the HALT instruction and an instruction that sets the STP bit (PSC register).

a) STP bit (PSC register) setting instruction

```
mov 0x2,r2
movea base_address,r0,r20     ; base_address = FFFF0000H
st.h r11,PRCMD[R20]           ; PRCMD = 01FCH
st.h r11,PSC[R20]             ; PSC = 01FEH
nop  ┐
nop  │
nop  } Insert five NOPs.
nop  │
nop  ┘
```

b) HALT instruction

```
halt
nop  ┐
nop  │
nop  } Insert five NOPs.
nop  │
nop  ┘
```

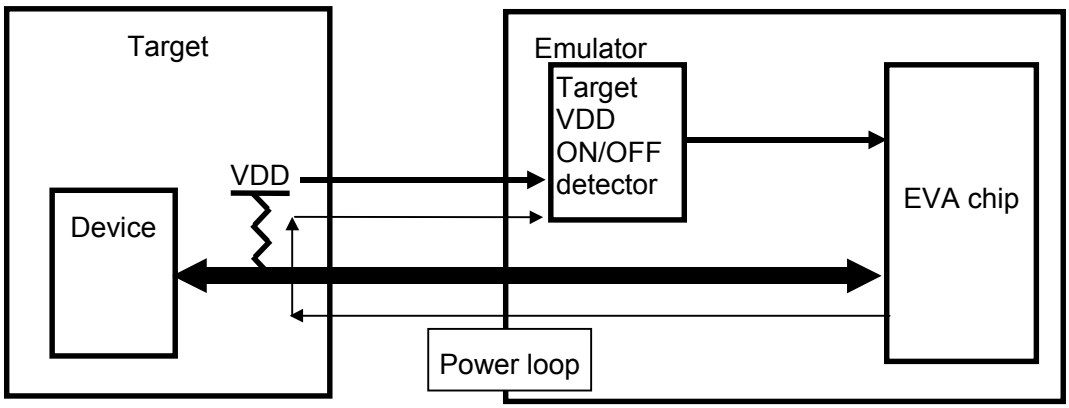
3) Pin control with target power off

When power to the emulator is on and that to the target is off, leakage current may flow from the emulator to the target.

When the target is connected, the emulator always senses the target supply voltage by using a target supply voltage detector, and the emulator is automatically reset when target power is turned off. In this reset status, the external bus signals go into a high-impedance state.

Some external bus signals, however, drive a high level, and a current may leak into VDD of the target via the pull-up resistor of the target.

The target supply voltage detector of the emulator detects this VDD, and the emulator assumes that power is applied to the target. Consequently, reset is cleared and the external bus signals are driven. As a result, a leakage current flows.



4) Trace sequence of access data of LD and ST instructions

If the LD and ST instructions are executed in that order, access of the ST instruction and access of the LD instruction are traced in this order for trace data.

If the LD instruction is the shortest (IRAM access), the read data is written to the same frame as the LD instruction. If the bus cycle is extended because of external memory access, the read data is written to trace after the ST instruction.

For instruction execution (fetch), the instruction that comes first is traced, and relation can be established based on the information on the access validity flag and direction of read/write.

- Details -

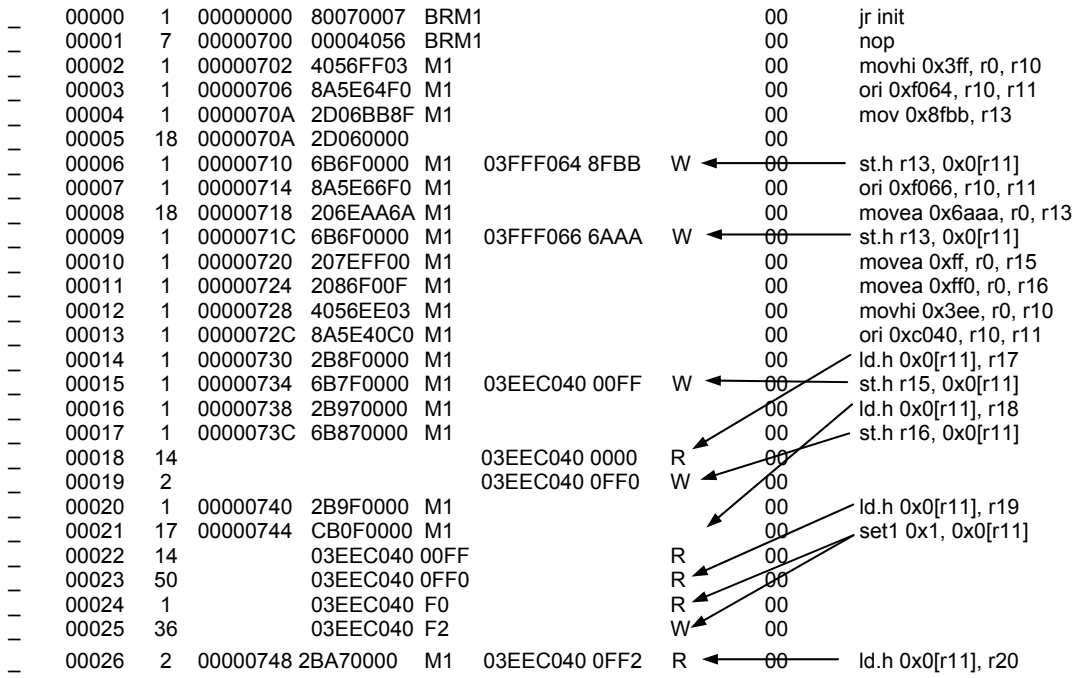
Basically, because the data is known in the write cycle, preparation for writing the data to the tracer is made when the ST instruction is executed. In the read cycle, the data is written to the tracer when the read cycle is completed and the data is loaded. If the LD and ST instructions are arranged, therefore, the sequence of only the access data of the trace data may be reversed, like the data of the ST instruction → data of the LD instruction.

Here is a sample program and its trace data.

[Sample program]

```
*    00000700 init      0000            nop
*    00000702 bpc       4056ff03        movhi 0x3ff, r0, r10
*    00000706           8a5e64f0        ori 0xf064, r10, r11
*    0000070A           2d06bb8f0000    mov 0x8fbb, r13
*    00000710           6b6f0000        st.h r13, 0x0[r11]          ; Writes 0x8FBB to address 0x3FFF064
*    00000714 bsc       8a5e66f0        ori 0xf066, r10, r11
*    00000718           206eaa6a        movea 0x6aaa, r0, r13
*    0000071C           6b6f0000        st.h r13, 0x0[r11]          ; Writes 0x6AAA to address 0x3FFF066
*    00000720           207eff00        movea 0xff, r0, r15
*    00000724           2086f00f        movea 0xff0, r0, r16
*    00000728 start     4056ee03        movhi 0x3ee, r0, r10
*    0000072C npb       8a5e40c0        ori 0xc040, r10, r11
*    00000730           2b8f0000        ld.h 0x0[r11], r17          ; Reads 0x0000 from address 0x3EEC040
*    00000734           6b7f0000        st.h r15, 0x0[r11]          ; Writes 0x00FF to address 0x3EEC040
*    00000738           2b970000        ld.h 0x0[r11], r18          ; Reads 0x00FF from address 0x3EEC040
*    0000073C           6b870000        st.h r16, 0x0[r11]          ; Writes 0x0FF0 to address 0x3EEC040
*    00000740           2b9f0000        ld.h 0x0[r11], r19          ; Reads 0x0ff0 from address 0x3EEC040
*    00000744           cb0f0000        set1 0x1, 0x0[r11]          ; SET instruction (RMW) to address 0x3EEC040
*    00000748           2ba70000        ld.h 0x0[r11], r20          ; Reads 0x0FF2 from address 0x3EEC040
*    0000074C           0000            nop
```

[Trace display example of sample program]



_	00000	1	00000000	80070007	BRM1			00	jr init
_	00001	7	00000700	00004056	BRM1			00	nop
_	00002	1	00000702	4056FF03	M1			00	movhi 0x3ff, r0, r10
_	00003	1	00000706	8A5E64F0	M1			00	ori 0xf064, r10, r11
_	00004	1	0000070A	2D06BB8F	M1			00	mov 0x8fbb, r13
_	00005	18	0000070A	2D060000				00	
_	00006	1	00000710	6B6F0000	M1	03FFF064	8FBB W	00	st.h r13, 0x0[r11]
_	00007	1	00000714	8A5E66F0	M1			00	ori 0xf066, r10, r11
_	00008	18	00000718	206EAA6A	M1			00	movea 0x6aaa, r0, r13
_	00009	1	0000071C	6B6F0000	M1	03FFF066	6AAA W	00	st.h r13, 0x0[r11]
_	00010	1	00000720	207EFF00	M1			00	movea 0xff, r0, r15
_	00011	1	00000724	2086F00F	M1			00	movea 0xff0, r0, r16
_	00012	1	00000728	4056EE03	M1			00	movhi 0x3ee, r0, r10
_	00013	1	0000072C	8A5E40C0	M1			00	ori 0xc040, r10, r11
_	00014	1	00000730	2B8F0000	M1			00	ld.h 0x0[r11], r17
_	00015	1	00000734	6B7F0000	M1	03EEC040	00FF W	00	st.h r15, 0x0[r11]
_	00016	1	00000738	2B970000	M1			00	ld.h 0x0[r11], r18
_	00017	1	0000073C	6B870000	M1			00	st.h r16, 0x0[r11]
_	00018	14				03EEC040	0000 R	00	
_	00019	2				03EEC040	0FF0 W	00	
_	00020	1	00000740	2B9F0000	M1			00	ld.h 0x0[r11], r19
_	00021	17	00000744	CB0F0000	M1			00	set1 0x1, 0x0[r11]
_	00022	14		03EEC040	00FF		R	00	
_	00023	50		03EEC040	0FF0		R	00	
_	00024	1		03EEC040	F0		R	00	
_	00025	36		03EEC040	F2		W	00	
_	00026	2	00000748	2BA70000	M1	03EEC040	0FF2 R	00	ld.h 0x0[r11], r20

5) Caution on fail-safe break in internal ROM space

The internal ROM of the emulator is set as follows depending on the debugger setting.

Internal ROM	
Debugger Setting	Internal ROM Space to Be Mapped
0 KB	None
32 KB	00000000H to 00007FFFH
64 KB	00000000H to 0000FFFFH
128 KB	00000000H to 0001FFFFH
256 KB	00000000H to 0003FFFFH
512 KB	00000000H to 0007FFFFH
1024 KB	00000000H to 000FFFFFH
Other	Depends on the target device

No fail-safe break occurs in any mapping case when an access (instruction fetch or data read access) to 00000000H to 000FFFFFH is performed. A write-protect break occurs when a write access is performed.

To disable accessing to the space from 00080000H to 000FFFFFH, for example when 512 KB is set, implement a measure such as setting an event break.