Microcomputer Technical Information

CP(K), O

IE-V850ES-G1 In-Circuit Emulator for V850ES Usage Restrictions		Document No.	ZBG-CD-04-0058		1/2
		Date issued	September 10, 2004		
		Issued by	Development Tool Group Multipurpose Microcomputer Systems Division 3rd Systems Operations Unit NEC Electronics Corporation		
Related documents	User's manual: U16313EJ1V0UM00	Notification classification	√	Usage restriction	
				Upgrade	
				Document modification	
				Other notification	

1. Affected product

   IE-V850ES-G1 Control code[Note]: A, B, C, D, E, F

2. Details of restrictions

   There are no additional restrictions. Control code F information has been added.

   See the attachment for details.

3. Workarounds

   See the attachment for details.

4. Modification schedule

   Products in which bug No. 2 is modified are scheduled for release as follows.

   Newly shipped products: From the shipment of September 2004 (control code: F)

   Upgrade for already shipped products: Available from September, 2004

   * Note that this schedule is subject to change without notice. For the detailed release schedule of the modified products, contact an NEC Electronics sales representative.

5. List of restrictions

   See the attachment for details.

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 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.

6. Document history

IE-V850ES-G1 In-Circuit Emulator for V850ES Usage Restrictions

Document Number	Issued on	Description
SBG-TT-0127-E	June 12, 2002	Newly created
SBG-DT-03-0216-E	July 31, 2003	Addition of new bugs (No .15 to No. 23)
ZBG-CD-04-0058	September 10, 2004	Correction of bug (No. 2)

Notes on Using IE-V850ES-G1

This document describes restrictions applicable only to the emulator and restrictions that are planned for correction in the emulator.

Refer to the following documents for the restrictions in the target device.

- User's manual of target device
- Restrictions notification document for target device

Also refer to the user's manual of the emulator for cautions on using the emulator.

1. Product Version

Part number: IE-V850ES-G1

Control Code[Note]	EVA Chip	Usable Debugger
A	UPD703191AR DS4.5-3 V or higher	ID850 E2.31c or later
B	UPD703191AR DS4.5-3 V or higher	ID850 E2.31c or later
C	UPD703191AR DS4.5-3 V or higher	ID850 E2.31c or later
D	UPD703191AR DS4.5-3 V or higher	ID850 E2.31c or later
E	UPD703191AR DS4.5-3 V or higher	ID850 E2.31c or later
F	UPD703191AR DS4.5-3 V or higher	ID850 E2.31c or later

Use the latest debugger.

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 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.

2. Product History

No.	Bugs and Changes/Additions to Specifications	Control Code[Note]				
		A	B	C	D/E	F
1	The current product case is provisional	×	√	√	√	√
2	Restriction on operating frequency	×	×	×	×	√
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 peripheral I/O register illegal break	Permanent restriction				
6	Restrictions on programmable I/O space	Permanent restriction				
7	Hardware break does not occur even if breakpoint is set.	Permanent restriction				
8	Restriction related to access address during DMA trace	Permanent restriction				
9	Restriction on DBPC and DBPSW access during a break	Permanent restriction				
10	Restriction on DBTRAP instructions	Permanent restriction				
11	Restriction on access data traced by DMA	Permanent restriction				
12	Restriction on peripheral I/O register read access during break	Supported by debugger				
13	Restriction on target operating voltage	×	√	√	√	√
14	Improvement of power supply block safety	×	×	√	√	√
15	Bug related to DMA transfer forcible termination	Permanent restriction				
16	Bug in program execution and DMA transfer in internal RAM	Permanent restriction				
17	Restriction on accessing emulation ROM area and emulation RAM area	Permanent restriction				
18	Change of product attachment	×	×	×	√	√
19	Restriction on 48-bit length mov instruction	Permanent restriction				
20	trace Restriction that the same branch instruction is traced twice	Permanent restriction				
21	Restriction on illegal trace of consecutive sld instruction	Permanent restriction				
22	Restriction on pin mask function	Permanent restriction				
23	Restriction that the same branch instruction is traced twice	Permanent restriction				

√: Already corrected

×: Corresponding restriction exists

–: Restriction does not apply

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 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 The current product case is provisional.

[Description]

The current product case of the IE-V850ES-G1 is provisional.

[Workaround]

There is no workaround. This bug has been corrected in control code B and later products.

No. 2 Restriction on operating frequency

[Description]

The maximum operating frequency is 20 MHz.

[Workaround]

There is no workaround. Use a frequency of 20 MHz or lower.

This restriction has been corrected in control code F and later products. The maximum operating frequency after correction is as follows.

- V850ES/SG2, V850ES/SJ2: 32 MHz
- V850ES/ST2: 34 MHz

The maximum operating frequency also varies depending on the control code of the emulation board, so also refer to the restriction document of the emulation board.

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 peripheral I/O register illegal break

[Description]

Peripheral 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 written in halfword units (break does not occur if an 8-bit I/O register is read in halfword units).

[Workaround]

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

No. 6 Restrictions on programmable I/O space

[Description]

a) If a programmable I/O space is mapped to the higher 32 MB area, the programmable I/O space cannot be accessed during break.

b) If the programmable I/O space is accessed during program execution, a peripheral I/O register illegal break occurs.

[Workaround]

a) Map the programmable I/O space to the lower 32 MB area.
This restriction has been removed in ID850 V2.40 or later.

b) Map the area where the programmable I/O space exists to the emulation memory or target memory.

Regard this as a permanent restriction.

No. 7 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 non-word align

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 (write back) 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 Architecture User’s Manual of the processor.**

[Workaround]

- The workaround used when setting software breaks can be applied to the following debugger versions.

  NEC Electronics debugger: ID850 E2.20f and later versions
  GHS Multi: Use EX85032.DLL version 5.40 or later

- There is no workaround for hardware breaks.
  Regard this as permanent restriction.

  * Use the latest debugger.

No. 8 Restriction related to access address during DMA trace

[Description]

If DMA is started while the internal RAM is accessed or the program in the internal RAM is executed, either the source address or destination address of DMA will become 3FFExxxh, indicating an internal RAM address for either of the above or for trace data.

[Workaround]

There is no workaround. Regard this as permanent restriction.

No. 9 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. 10 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.11 Restriction on access data traced by DMA

[Description]

When data in internal RAM is read by DMA, the read data value is not traced correctly.

* Read address, write data, and write address are traced correctly.

[Workaround]

There is no workaround. Regard this as permanent restriction.

No. 12 Restriction on peripheral I/O register read access during break

[Description]

The peripheral I/O register bits that are normally cleared by a read access (e.g. the TC bit of the DCHC register) are also cleared when displayed using peripheral I/O register display in the debugger during a break. (They are cleared when displayed by the debugger even though they are not read by the program.)

[Workaround]

The bit is not reset if the relevant peripheral I/O register is not displayed by debugger.

This restriction can be avoided by using the following debuggers.

Debugger by NEC Electronics:	ID850 E2.20f or later
Debugger by GHS (Multi):	SV-V850-WIN32 Rel.4.0.5 or later
(Windows version)	Specify the -X2 option on starting
Debugger by GHS (Multi):	SV-V850-Solaris Rel.4.0.5 or later
(Solaris version)	Specify the -X2 option on starting

* Use the latest debugger.

No. 13 Restriction on target operating voltage

[Description]

Emulation cannot be performed when the target operating voltage is 3.5 V or higher.

[Workaround]

There is no workaround. This bug has been corrected in control code B and later products.

No. 14 Improvement of power supply block safety

[Description]

The safety of the power supply block has been improved.

(There is no safety problem even if this improvement has not been performed.)

This improvement has been implemented in control code C and later products.

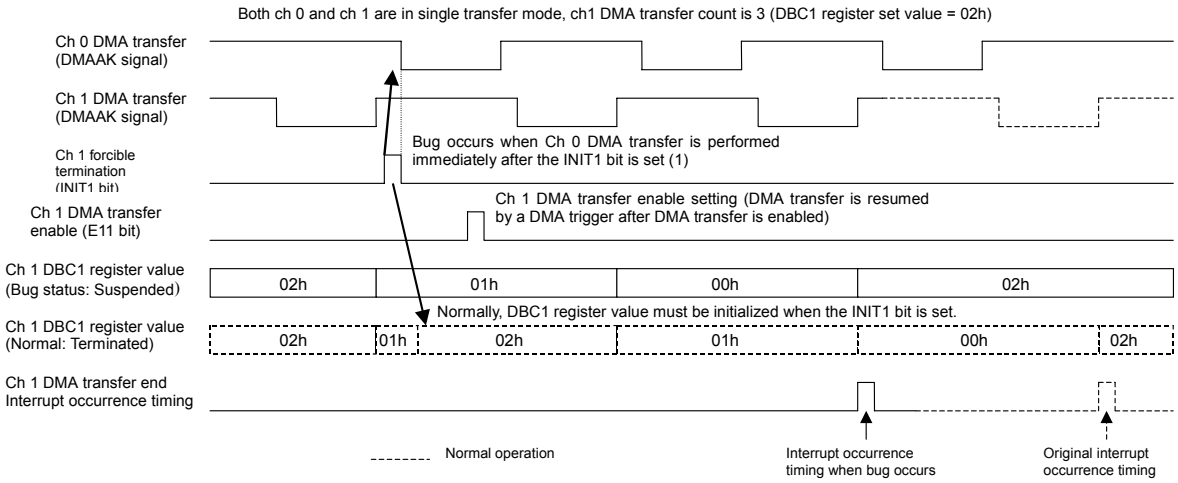
No. 15 Bug related to DMA transfer forcible termination

[Description]

When terminating a DMA transfer by setting the INITn bit of the DCHCn register, the transfer may not be terminated, but just suspended, even though the INITn bit is set (1). As a result, when the DMA transfer of a channel that should have been terminated is resumed, the DMA transfer will

terminate after an unexpected number of transfers are completed and a DMA transfer completion interrupt may occur (n = 0 to 3). This bug occurs if a DMA transfer is executed immediately after a forcible termination is set (by setting the INITn bit) (see the figure below).

This bug does not depend on the number of transfer channels, transfer type (2-cycle or flyby), transfer target (between memory and memory, memory and I/O; including internal resources), transfer mode (single, single-step, or block), or trigger (external request, interrupt from internal peripheral I/O, or software), and can occur with any combination of the above elements that can be set under the specifications. In addition, another channel may affect the occurrence of this bug.



The following registers are buffer registers with a 2-stage FIFO configuration of master and slave. If these registers are overwritten during a DMA transfer or in the DMA-suspended status, the value is written to the master register, and reflected in the slave register when the DMA transfer of the overwritten channel is terminated.

The “initialization” in the above figure means that the contents of the master register are reflected in the slave register.

2-stage FIFO configuration registers:

- DMA source address register (DSAnH, DSAnL)
- DMA destination address register (DDAnH, DDAnL)
- DMA transfer count register (DBCn)

[Workaround]

This bug can be avoided by implementing any of the following procedures using the software.

(1) Stop all the transfers from DMA channels temporarily

The following measure is effective if the following condition is satisfied.

- Except for the following workaround processing, the program does not assume that the TCn bit of the DCHCn register is 1. (Since the TCn bit of the DCHCn register is cleared (0) when it is read, execution of the following procedure (b) under <5> clears this bit.)

[Procedure to avoid bug]

<1> Disable interrupts (DI state).

<2> Read the DMA restart register (DRST) and transfer the ENn bit of each channel to a general-purpose register (value A).

<3> Write 00H to the DMA restart register (DRST) twice[Note].
By executing twice[Note], the DMA transfer is definitely stopped before proceeding to <4>.

<4> Set (1) the INITn bit of the DCHCn register of the channel that should be terminated forcibly.

<5> Perform the following operations for value A read in <2>. (Value B)
(a) Clear (0) the bit of the channel that should be terminated forcibly
(b) If the TCn and ENn bits of the channel that is not terminated forcibly are 1 (AND makes 1), clear (0) the bit of the channel.

<6> Write value B in <5> to the DRST register.

<7> Enable interrupts (EI state).

Remarks 1. Be sure to execute <5> to prevent the ENn bit from being set illegally for channels that are terminated normally during the period of <2> and<3>.
2. n = 0 to 3

Note Execute three times if the transfer target (transfer source or transfer destination) is the internal RAM.

(2) Repeat setting the INITn bit until the forcible DMA transfer termination is correctly performed (n = 0 to 3)

[Procedure to avoid bug]

<1> Copy the initial transfer count of the channel that should be terminated forcibly to a general-purpose register.

<2> Set (1) the INITn bit of the DCHCn register of the channel that should be terminated forcibly.

<3> Read the value of the DMA transfer count register (DBCn) of the channel that should be terminated forcibly and compare the value with the one copied in <1>. If the values do not match, repeat <2> and <3>.

Remarks 1. When the DBCn register is read in procedure <3>, the remaining transfer count will be read if the DMA is stopped due to this bug. If the forcible DMA termination is performed correctly, the initial transfer count will be read.
2. Note that it may take some time for forcible termination to take effect if this workaround is implemented in an application in which DMA transfers other than for channels subject to forcible termination are frequently performed.

Please regard this as a permanent restriction.

No. 16 Bug in program execution and DMA transfer in internal RAM

[Description]

When a DMA transfer for the internal RAM and a bit manipulation instruction (SET1, CLR1, or NOT1) allocated in the internal RAM or a data access instruction for a misaligned address are executed simultaneously, the CPU may deadlock due to conflict between the internal bus

operations. At this time, only a reset can be acknowledged. (An NMI or interrupt cannot be acknowledged.)

[Workaround]

Implement any of the following workarounds.

- Do not perform a DMA transfer for the internal RAM when an instruction allocated in the internal RAM is being executed.
- Do not execute an instruction allocated in the internal RAM when a DMA transfer for the internal RAM is being performed.

Please regard this as a permanent restriction.

No. 17 Restriction on accessing emulation ROM area and emulation RAM area

[Description]

The emulation ROM area and emulation RAM area cannot be accessed via the 8-bit bus.

[Workaround]

Use the 16-bit bus for accessing the emulation ROM area or emulation RAM area.

Regard this as a permanent restriction.

No. 18 Change of product attachment

[Description]

The items included with the IE-V850ES-G1 have been changed from control code E or later.

- Change of PC interface cable (for PCI bus, PCMCIA)
  The cable length has been changed from 1 m to 2 m.
- Addition of power supply cable
  A power supply cable for types C and BF has been added.

If customers who have IE-V850ES-G1 control code A to D require a 2 m PC interface cable or power supply cable for types C and BF, contact an NEC Electronics sales representative or distributor.

No. 19 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. 20 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.

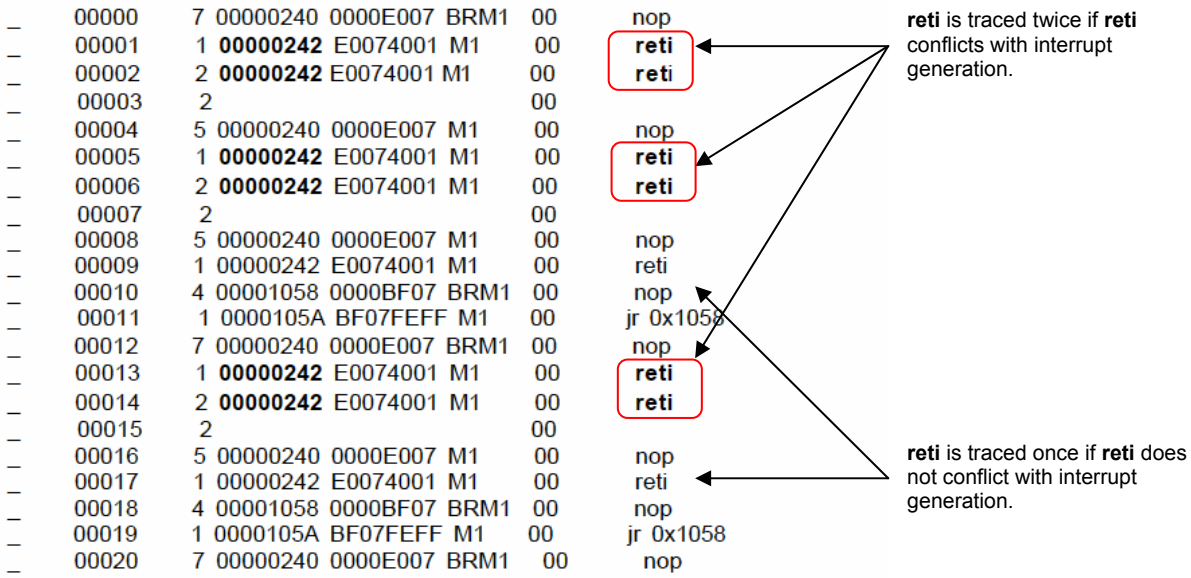
No. 21 Restriction that the same branch instruction is traced twice

[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]

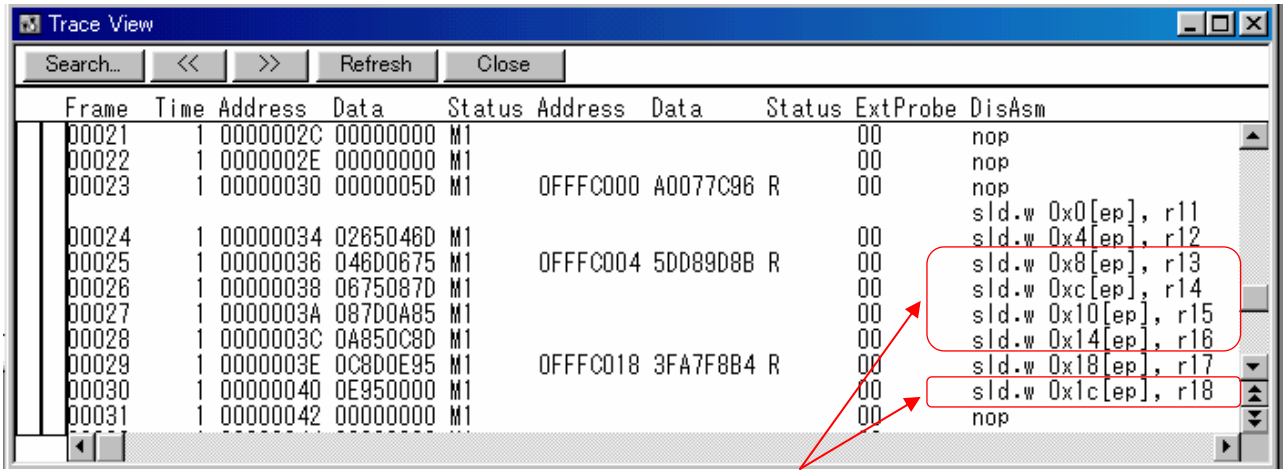
There is no workaround. Regard this as a permanent restriction.

No. 22 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. 23 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

4.1 Notes on using emulation memory

- The emulation memory cannot be mapped to addresses higher than 0x4000000.
- The number of wait cycles for the emulation memory is not affected by the _WAIT signal but is determined by setting of the debugger or setting of the wait control register (see the table below).

(1) When using ID850

Select one of the following three settings on the configuration screen.

Setting	Wait Type	Emulation Memory Access	External Memory Access
0 WAIT ACCESS	Data wait	Fixed to 0 waits	Depends on the DWC0 register settings and WAIT signal status.
	Address wait	Fixed to 0 waits	Depends on the AWC register settings.
	Idle state	Fixed to 0 cycles	Depends on the BCC register settings.
1 WAIT ACCESS	Data wait	Fixed to 1 wait	Depends on the DWC0 register settings and WAIT signal status.
	Address wait	Fixed to 0 waits	Depends on the AWC register settings.
	Idle state	Fixed to 0 cycles	Depends on the BCC register settings.
DWC0, DWC1, BCC	Data wait	Depends on the DWC0 register setting. 1 wait when set to 0 waits.	Depends on the DWC0 register settings and WAIT signal status.
	Address wait	Fixed to 0 waits	Depends on the AWC register settings.
	Idle state	Depends on the BCC register setting.	Depends on the BCC register settings.

(2) When using Multi

Select masking/unmasking of the WAIT and EMWAIT pins using the “pinmask” command.

Setting	Wait Type	Emulation Memory Access	External Memory Access
WAIT: Mask EMWAIT: Mask	Data wait	Fixed to 0 waits	Depends on the DWC0 register settings and WAIT signal status.
	Address wait	Fixed to 0 waits	Depends on the AWC register settings.
	Idle state	Fixed to 0 cycles	Depends on the BCC register settings.
WAIT: Unmask EMWAIT: Mask	Data wait	Fixed to 1 wait	Depends on the DWC0 register settings and WAIT signal status.
	Address wait	Fixed to 0 waits	Depends on the AWC register settings.
	Idle state	Fixed to 0 cycles	Depends on the BCC register settings.
WAIT: Unmask EMWAIT: Unmask	Data wait	Depends on the DWC0 register setting. 1 wait when set to 0 waits.	Depends on the DWC0 register settings and WAIT signal status.
	Address wait	Fixed to 0 waits	Depends on the AWC register settings.
	Idle state	Depends on the BCC register setting.	Depends on the BCC register settings.

* Remark on address wait

Address waits cannot be inserted in the emulation memory.

If it is necessary to insert an address wait, set as follows.

Number of data waits for CS space in emulation memory	=	Number of address waits for external memory or external I/O	+	Number of data waits for external memory or external I/O

This setting is effective to make the speed of access to the emulation memory equal to that to the external memory or external I/O when measuring the performance, etc.

Refer to the table on the previous page for how to insert waits in the emulation memory.

* Remark on EXIMC register

The setting of the EXIMC register is not valid for the emulation memory; it is only valid for the memory and I/O that are mapped to the target. The emulation memory operates via the separate bus regardless of the EXIMC settings.

Since the difference between the cycle lengths of the multiplexed bus and separate bus is 1 clock, set the multiplexed bus as follows when it is used.

This setting is effective to make the speed of access to the emulation memory equal to that to the external memory or external I/O when measuring the performance, etc.

Number of data waits for CS space in emulation memory	=	1	+	Number of data waits for external memory or external I/O

4.2 Pin connection

- This emulator uses the minimum pin connection, giving priority to compatibility with the device. Take adequate workarounds for static electricity if the emulator is used without the target connected.
- Inside the emulator, pin connection is performed by the emulation board. Refer to the user's manual of the emulation board for details.

4.3 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  }
```

4.4 Notes on trace data

a) 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]

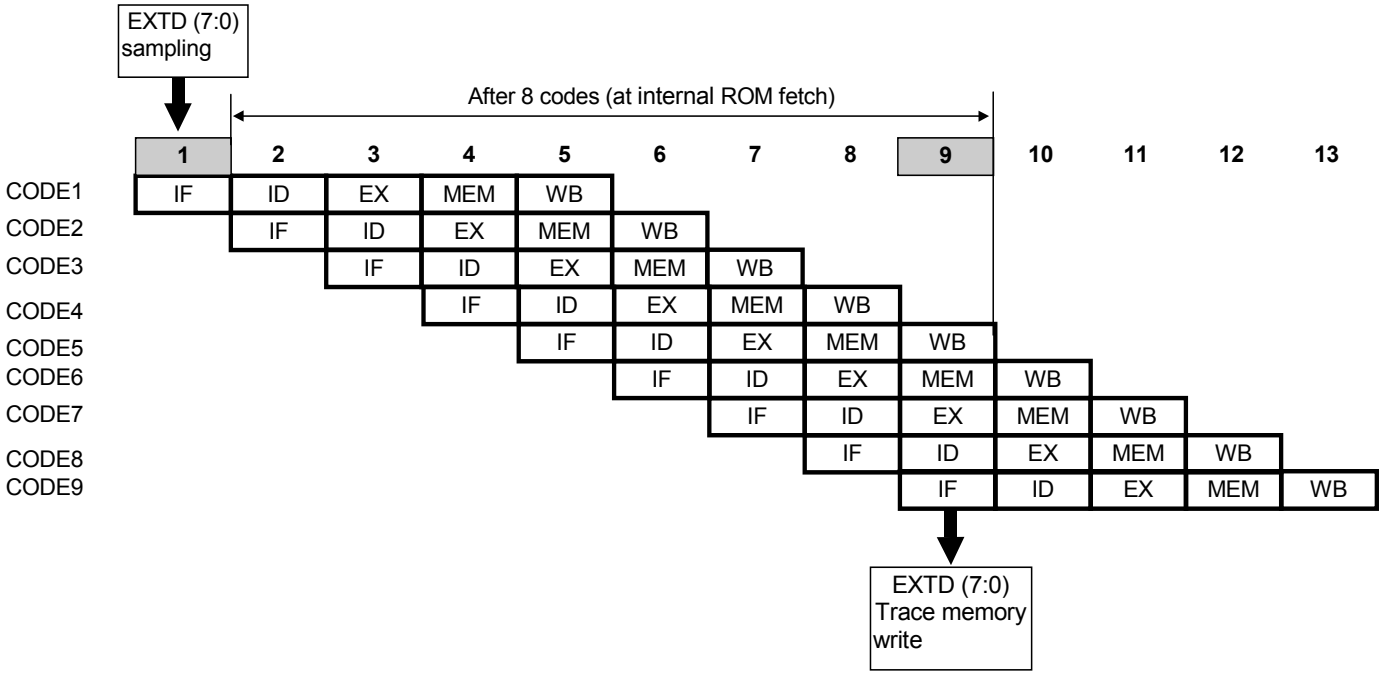
```
*    00000700 init      0000          nop
*    00000702 bpc       4056ff03      movhi 0x3ff, r0, r10
*    00000706           8a5e64f0      ori 0xf064, r10, r11
*    0000070A           2d06bb8f0000  mov 0x8fbb, r13
*    00000710           6b6f0000      st.h r13, 0x0[r11]      ; Writes 0x8FBB to address 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
```

b) Trace timing of external logic data

- It takes the external logic data **the number of clocks required for fetching 8 x 1 times** to be output to the tracer. The figure below illustrates the case when an instruction is executed in the internal ROM.
- The external logic data is traced in synchronization with instruction execution. Therefore, when the instruction is fetched from a memory that can be accessed only at a low speed, such as the external memory, the interval at which the external logic data remains in the trace buffer becomes longer.



4.5 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 (When Mapped from 0H Address)
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.

In addition, no fail-safe break occurs when an access (instruction fetch or data read access) to 00100000H to 001FFFFFH is performed if internal ROM exists from address 00100000H.

4.6 General cautions on handling this product

a) Cases in which NEC Electronics warranty does not apply

- When the product is disassembled, reconstructed, or modified by the user
- When the product receives a heavy shock such as being dropped or falling down
- When the product is used with excessive voltage or is stored outside the guaranteed
- When power is applied while the AC adapter, PC interface cable, or target system is not connected securely
- When the AC adapter cable, PC interface cable, or emulation probe is excessively twisted or stretched
- When an AC cable other than the one supplied with the IE-V850ES-G1 is used
- When water is spilled on the product
- When the product and target system are connected in a system in which the voltage potential between the GND of the product and the target system GND differ
- When the connector or cable is connected or disconnected while the power is being applied to the product
- When an excessive load is applied to the connector or socket

b) Cautions on safe use

- The product heats up (to approx. 50 to 60°C) when it operates for a long time. Take care not to receive injuries such as burns from a rise in the temperature.
- Be very careful to avoid electric shocks. There is risk of electric shock if the product is used as described in the item above.