CUSTOMER NOTIFICATION

SUD-DT-03-0314-1-E
(SUD-DT-02-0087-1-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-V850E-MC (Control Code: A, B, C, D)
IE-V850E-MC-A (Control Code: B, C, D, E, F, G)

Operating Precautions

Be sure to read this document before using the product.

Notes on Using IE-V850E-MC-A, IE-V850E-MC

1. Product Version

Part number: IE-V850E-MC-A

Control Code	EVA Chip (for 3.3 V)	Usable Debugger
B	UPD703191AR DS2.0-3V	ID850 E2.00s (for NB85E) or later
C	UPD703191AR DS2.0-3V	ID850 E2.00s (for NB85E) or later
D	UPD703191AR DS3.0-3V	ID850 E2.20f or later
E	UPD703191AR DS4.1-3V	ID850 E2.20f or later
F	UPD703191AR ES4.3-3V UPD703191AR DS4.3-3V	ID850 E2.20f or later
G	UPD703191AR ES4.5-3V	ID850 E2.20f or later

Part number: IE-V850E-MC

Control Code	EVA Chip (for 5 V)	Usable Debugger
A	UPD703191R DS2.0-5V	ID850 E2.00s (for NB85E) or above
B	UPD703191R DS3.0-5V	ID850 E2.20f or later
C	UPD703191R DS4.1-5V	ID850 E2.20f or later
D	UPD703191R DS4.3-5V UPD703191R ES4.3-5V UPD703191R ES4.5-5V	ID850 E2.20f or later

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.

2. Product History

No.		Restrictions	Control Code									
			IE-V850E-MC				IE-V850E-MC-A					
			A	B	C	D	B	C	D	E	F	G
Restrictions dependent on CPU functions	a-1	Interrupt aborts LD instruction immediately before JMP	×	√	√	√	×	×	√	√	√	√
	a-2	Restrictions on IRAM read access after start of interrupt servicing	×	√	√	√	×	×	√	√	√	√
	a-3	Fetching is abnormal immediately after writing to SCRn register	×	√	√	√	×	×	√	√	√	√
	a-4	Single, line, or single-step transfer of 2-cycle DMA	×	√	√	√	×	×	√	√	√	√
	a-5	Port C is not set in control mode immediately after starting in ROMless mode.	×	√	√	√	×	×	√	√	√	√
	a-6	Restrictions on ports DH and DL	×	√	√	√	×	×	√	√	√	√
	a-7	HLDAK output illegal due to conflict of self-refresh cycle and HOLDRQ in STOP mode	×	√	√	√	×	×	√	√	√	√
	a-8	Fetch/data access fails if hardware STOP is executed after CBR refresh of DRAM/SDRAM	×	√	√	√	×	×	√	√	√	√
	a-9	Restrictions on data cache	Permanent restriction									
	a-10	PFCCM register cannot be read.	×	√	√	√	×	×	√	√	√	√
	a-11	VSB bus and memory controller (NB85E500/501/502) cannot be used together.	Permanent restriction									
	a-12	Restrictions on VSB bus signal	×	√	√	√	×	×	√	√	√	√
	a-13	Restrictions on NPB bus signal	Permanent restriction									
	a-14	Restrictions on memory controller (NB85E500) signal	×	√	√	√	×	×	√	√	√	√
	a-15	Restrictions on instruction cache	×	√	√	√	×	×	√	√	√	√
	a-16	Restriction on SDRAM access during bus hold	×	√	√	√	×	×	√	√	√	√
	a-17	Restriction on self-refresh cycle by SELFREF pin	×	√	√	√	×	×	√	√	√	√
	a-18	Restriction on flyby DMA transfer to EDO DRAM	×	√	√	√	×	×	√	√	√	√
	a-19	Restriction on EDO DRAM with idle state inserted	×	√	√	√	×	×	√	√	√	√
	a-20	Restriction on flyby DMA transfer	×	√	√	√	×	×	√	√	√	√
	a-21	Restriction on pin status in single-step mode 1 and ROMless modes 0 and 1	–	–	–	–	×	×	√	√	√	√
	a-22	Incorrect writeback with LD/SLD instruction when executing CALLT/SWITCH instruction	×	×	√	√	×	×	×	√	√	√
	a-23	Restriction on use of external bus when product is employed as an emulator for the V850E/IA1	×	√	√	√	–	–	–	–	–	–
	a-24	Restriction on output of the _DMAAK signal	×	×	√	√	×	×	×	√	√	√
	a-25	Restriction on starting DMA by built-in peripheral I/O interrupt	×	×	√	√	×	×	×	√	√	√
	a-26	Restriction on EDO DRAM bus collision	×	×	√	√	×	×	×	√	√	√
	a-27	Restriction on 2-way associative function of instruction cache	×	×	√	√	×	×	×	√	√	√
	a-28	Forced stop of external DMA transfer in DMA line transfer mode	×	×	√	√	×	×	×	√	√	√
	a-29	Restriction on reading DCHC register when DMA 2-cycle transfer is complete	×	×	√	√	×	×	×	√	√	√
	a-30	Restriction on conflict between SDRAM initialization and SELFREF input	×	×	√	√	×	×	×	√	√	√
	a-31	Restriction on halfword writing to BSC, BCC, DWC0, and DWC1 registers	×	×	√	√	×	×	×	√	√	√
	a-32	Restriction on SDRAM write operation	×	×	√	√	×	×	×	√	√	√

√: Already confirmed and solved

×: Corresponding restriction exists

–: Restriction does not apply

No.		Restrictions	Control Code									
			IE-V850E-MC				IE-V850E-MC-A					
			A	B	C	D	B	C	D	E	F	G
	a-33	Restriction on DRAM fetch immediately after block DMA transfer from DRAM to internal RAM	×	√	√	√	×	×	√	√	√	√
	a-34	Restriction on instruction cache (2)	Permanent restriction									
	a-35	Restriction on SLD instruction	×	×	×	√	×	×	×	×	√	√
	a-36	I/O that cannot be used when using a VSB bus	×	×	×	√	×	×	×	×	√	√
	a-37	Restriction on instruction cache (3)	×	×	×	√	×	×	×	×	√	√
	a-38	Restriction on DMAAK signal during DMA line transfer										
	a-39	Restriction caused by interrupt input during execution of bit manipulation instruction	×	×	×	√	×	×	×	×	√	√
	a-40	Restriction on hardware stop during bit manipulation instruction execution	×	×	×	√	×	×	×	×	√	√
	a-41	Restriction related to interruption of DMA transfer by external cause	×	×	×	√	×	×	×	×	√	√
	a-42	Restriction on SDCKE signal during bus hold	–	–	–	–	×	×	×	×	×	√
	a-43	Caution regarding SDRAM controller	Permanent restriction									
	a-44	Restriction on mul/mulu instruction	Permanent restriction									
	a-45	Restriction on page ROM access	Permanent restriction									
	a-46	Bug related to DMA transfer forcible termination	Permanent restriction									
	a-47	Bug that DMA transfer is forcibly suspended by NMI	Permanent restriction									
	a-48	Bug in program execution and DMA transfer in internal RAM	Permanent restriction									
	a-49	Bug related to DMA transfer whose transfer count is set to two (1)	Permanent restriction									
	a-50	Bug related to DMA transfer whose transfer count is set to two (2)	Permanent restriction									
	a-51	Bug that TCn bit of DMA is not cleared automatically	Permanent restriction									

√: Already confirmed and solved

×: Corresponding restriction exists

–: Restriction does not apply

No.		Restrictions		Control Code									
				IE-V850E-MC				IE-V850E-MC-A					
				A	B	C	D	B	C	D	E	F	G
Restrictions on debug functions	b-1	Restriction on operating frequency		×	×	×	×	×	×	×	√	√	√
	b-2	Restriction on break timing when guarded area is fetched		Permanent restriction									
	b-3	Restriction on trace in case of mis-alignment (during read access only)		×	√	√	√	×	×	√	√	√	√
	b-4	Restrictions on trace data on execution of HALT or STOP instruction		×	√	√	√	×	×	√	√	√	√
	b-5	Bit manipulation instruction (set1, clr1, not1, tst1) access data is illegally traced by tracer.		×	√	√	√	×	×	√	√	√	√
	b-6	Events including data conditions by access of bit manipulation instruction cannot be detected.		×	√	√	√	×	×	√	√	√	√
	b-7	Restriction on HOLD status		×	√	√	√	×	×	√	√	√	√
	b-8	ROM contents are rewritten if emulation ROM area is accessed for write.		Permanent restriction									
	b-9	Restriction on SFR illegal break		Permanent restriction									
	b-10	Restrictions on programmable I/O space	(a)	Supported by debugger									
			(b)	Permanent restriction									
	b-11	Break does not occur even if breakpoint is set.		Supported by debugger									
	b-12	Restriction related to access address during DMA trace		Permanent restriction									
	b-13	Restriction on DBPC and DBPSW access during a break		Permanent restriction									
	b-14	Restriction on DBTRAP instructions		Permanent restriction									
	b-15	Restriction on illegal guard break when IRAM size is 28KB		×	×	×	√	×	×	×	×	√	√
	b-16	Restriction on illegal trace when big endian is used		×	×	×	√	×	×	×	×	√	√
	b-17	Restriction on access data traced by DMA		Permanent restriction									
	b-18	Restriction on SFR read access during break		Supported by debugger and device file									
	b-19	Restriction that the debugger hangs up depending on the software break setting upon PSC register access		Permanent restriction									
	b-20	Restriction that the same branch instruction is traced twice		Permanent restriction									
	b-21	Restriction on 48-bit length mov instruction trace		Permanent restriction									
	b-22	Restriction on illegal trace of consecutive sld instruction		Permanent restriction									
Other	c-1	Modification through quality improvement		None				Provided					

√: Already confirmed and solved

×: Corresponding restriction exists

–: Restriction does not apply

3. Details of Bugs and Added Specifications

a-1. Interrupt aborts LD instruction immediately before JMP

[Description]

If an LD/SLD instruction immediately before a JR/JARL/Bcc instruction is aborted by an interrupt, the instruction may not be executed again after program execution has exited from the interrupt routine.

This phenomenon occurs when all the following conditions are satisfied (the branch destination of JR/JARL/Bcc is saved to EIPC/FEPC):

1) An LD/SLD instruction is executed immediately after a JR/JARL/Bcc instruction.
   - LD+JR if the instruction is fetched from IROM.
   - LD+ LD+ LD+JR if the instruction is fetched from external memory.
2) Two or more bus cycles of the preceding LD/SLD remain in the EX stage of the above LD/SLD instruction, and the EX stage of LD/SLD is held.
3) An interrupt occurs while the EX stage of the above LD/SLD is held.

- Mechanism -

If an interrupt occurs in the ID stage of a JR/JARL/Bcc instruction, the branch destination address is saved to EIPC/FEPC to increase the processing speed after execution has exited from the interrupt routine (because the branch instruction does not have to be executed again).

In the meantime, the LD/SLD/ST/SST instruction is aborted if a bus cycle has not yet been issued, after the EX stage has been started, in order to improve the interrupt response. If these functions conflict, i.e., if an interrupt occurs in the ID stage (= EX stage of LD/SLD) of JR/JARL/Bcc, processing of LD/SLD (saving its own address to EIPC/FEPC) must take precedence over the processing of JR/JARL/Bcc (saving the branch destination to EIPC/FEPC). However, processing of JR/JARL/Bcc takes precedence while the program is waiting for an external bus cycle and, consequently, the branch destination is saved to EIPC/FEPC.

Because the program waits for the ST/SST instruction in the ID stage, the above condition 2) is not satisfied, and therefore, the above phenomenon does not occur.

[Workaround]

Insert one or more instruction in between the LD/SLD and JR/JARL/Bcc instructions (because the above phenomenon does not occur if the EX stage of LD/SLD and ID stage of JR/JARL/Bcc do not overlap, any instruction (other than JR/JARL/Bcc/LD/SLD) may be inserted).

This bug has been corrected in the IE-V850E-MC with control code B.

This bug has been corrected in the IE-V850E-MC-A with control code D.

A tool that inserts a NOP instruction between the LD/SLD and JR/JARL/Bcc instruction is available as a patch for the compiler. This patch tool is not supplied with the current release of the compiler (from NEC Electronics and third parties) as standard. Consult the NEC Electronics Development Tool Support Center for details.

a-2. Restrictions on IRAM read access after start of interrupt servicing

[Description]

If the IRAM is read within 10 system CLK after interrupt servicing has been started (after execution has branched to the interrupt handler address), the IRAM cannot be correctly read and “0” is read.

The 10 system CLK is equivalent to the execution time of 20 steps of NOP instructions of assembler.

[Workaround]

To access IRAM for read after interrupt servicing has been started, insert a dummy read/write (to the external memory) instruction before the instruction that reads the IRAM.

This restriction has been corrected in the IE-V850E-MC with control code B.

This restriction has been corrected in the IE-V850E-MC-A with control code D.

a-3. Fetching is abnormal immediately after writing to SCRn register

[Description]

An abnormal fetch cycle occurs if the SCR register is written after the VSWC register is set to 11H so that SDRAM is used as the BCTn register.

This is because, if an access to external memory occurs immediately after the SCR register write cycle, the values of A[25:0] and CSZ[7:0] are not normal.

[Workaround]

Specify two or more wait cycles for VPSTB wait (set VSWL2, VSWL1, and VSWL0 of the VSWC register to 3’b010 or more).

This bug has been corrected in the IE-V850E-MC with control code B.

This bug has been corrected in the IE-V850E-MC-A with control code D.

a-4. Single, line, or single-step transfer of 2-cycle DMA

[Description]

If WAIT is inserted on the VSB bus for 3 clocks or more during single, line, or single-step transfer of 2-cycle DMA, fetch and data access cannot be executed until the DMA cycle is completed (until the TC signal is output).

[Workaround]

Insert a WAIT of 2 clocks or less. The emulator before correction executes a DMA operation more quickly than the actual chip (including the NB85E core).

This bug has been corrected in the IE-V850E-MC with control code B.

This bug has been corrected in the IE-V850E-MC-A with control code D.

a-5. Port C is not set in control mode immediately after starting in ROMless mode.

[Description]

Port C, which should be in the control mode immediately after the ROMless mode has been started, does not enter the control mode but enters the port mode. (The initial value of PMCCT is 00h.) DRAM cannot be accessed immediately after reset.

[Workaround]

Write ”1” to any bit of PMCCT, even after the ROMless mode has been started, to access DRAM.

Do not execute an application that accesses DRAM immediately after reset.

This bug has been corrected in the IE-V850E-MC with control code B.

This bug has been corrected in the IE-V850E-MC-A with control code D.

a-6. Restrictions on ports DH and DL

[Description]

Ports DH and DL enter the control mode (D31 to D0) immediately after they have been started (is not started in the port mode).

[Workaround]

The mode can be changed to the port mode by writing “0” to the PMCDH and PMCDL registers immediately after the emulator has been started.

This restriction has been corrected in the IE-V850E-MC with control code B.

This restriction has been corrected in the IE-V850E-MC-A with control code D.

a-7. HLDAK output illegal due to conflict of self-refresh cycle and HOLDRQ in STOP mode

[Description]

If self-refreshing of DRAM or SDRAM and HOLDRQ conflict in the STOP mode, the HOLDAK signal is illegally output even in the self-refresh cycle.

[Workaround]

Do not input HOLDRQ in the STOP mode.

This bug has been corrected in the IE-V850E-MC with control code B.

This bug has been corrected in the IE-V850E-MC-A with control code D.

a-8. Fetch/data access fails if hardware STOP is executed after CBR refresh of DRAM/SDRAM

[Description]

If hardware STOP is executed while the bus mastership is transferred from the device on the VSB bus to an internal unit of the NB85E after CBR refresh of DRAM or SDRAM, fetch/data access cannot be correctly executed because self refreshing caused by hardware STOP and fetch/data access conflict and the CS signal is disrupted (goes high).

[Workaround]

Do not use hardware STOP. Use software STOP.

This bug has been corrected in the IE-V850E-MC with control code B.

This bug has been corrected in the IE-V850E-MC-A with control code D.

a-9. Restrictions on data cache

[Description]

This emulator has no plan to support data cache.

a-10. PFCCM register cannot be read.

[Description]

Although the PFCCM register is an R/W register, it cannot be read.

[Workaround]

Use the PFCCM register as a write-only register.

This restriction has been corrected in the IE-V850E-MC with control code B.

This restriction has been corrected in the IE-V850E-MC-A with control code D.

a-11. VSB bus and memory controller (NB85E500/501/502) cannot be used together.

[Description]

The VSB bus signal pin is multiplexed with a memory controller pin. Therefore,

- The memory controller cannot be used while the VSB bus is being used.
- The VSB bus cannot be used while the memory controller is being used.

[Workaround]

There is no workaround. Please regard this as a permanent restriction.

a-12. Restrictions on VSB bus signal

[Description]

a) The master device on the VSB bus cannot access the slave source on the NPB bus.

b) The emulator does not output VBA [27:26].

c) The emulator does not output VBSEQ [2:0].

[Workaround]

a) There is no workaround.

b) Temporary workarounds are available for the IE-V850E-MC-EM1-A/B. For details, refer to the document describing the restrictions on the IE-V850E-MC-EM1-A/B.

c) There is no workaround.

These restrictions have been corrected in the IE-V850E-MC with control code B.

These restrictions have been corrected in the IE-V850E-MC-A with control code D.

a-13. Restrictions on NPB bus signal

[Description]

The emulator does not have a VPDACT signal pin and this signal is always fixed to the active level inside the emulator.

[Workaround]

There is no workaround. Please regard this as a permanent restriction.

a-14. Restrictions on memory controller (NB85E500) signal

[Description]

a) The emulator does not output the ASTBZ, DSTBZ, MPXCZ, RDCYZ, and BUSSTZ signals; therefore, the multiplex bus cannot be used.

b) The emulator does not output the BENZ3 to BENZ 0 and DC3 to DC0 signals.

c) The IORDZ and CSZ2, and IOWRZ and CSZ5 signals of the emulator are multiplexed.

d) The emulator does not have an MCE signal pin and this signal is always fixed to the active level inside the emulator. Therefore, the MEn bit (n = 0 to 7) of the BCTn register (n = 0 or 1) is always “1”, enabling operation.

e) Usually, the IORD and IOWR signals are asserted active in the normal read/write cycle and

during flyby transfer of DMA. But these signals, however, are asserted active only during flyby transfer of DMA in this memory controller.

[Workaround]

a) Do not use the multiplex bus. Use the separate bus.

Please regard this as a permanent restriction.

b) Design UDL that does not use BENZ3 to BENZ 0 and DC3 to DC0.

Please regard this as a permanent restriction.

c) Do not map external I/O to memory blocks 2 and 5.

The pin assignment of the IE-V850E-MC with control code B and IE-V850E-MC-A with control code D has been changed as follows.

Pin assignment of versions with the old control code:

_IORDZ and _CSZ2 are multiplexed.

_IOWRZ and _CSZ5 are multiplexed.

Pin assignment of the version with the new control code:

_IORDZ and _CSZ5 are multiplexed.

_IOWRZ and _CSZ2 are multiplexed.

d) There is no workaround. Please regard this as a permanent restriction.

e) There is no workaround.

This bug has been corrected in the IE-V850E-MC with control code B.

This bug has been corrected in the IE-V850E-MC-A with control code D.

a-15. Restrictions on instruction cache

[Description]

(a) If control is transferred from a cacheable area to an uncached area and then back to the cacheable area again after auto fill, the first one clock does not hit.

(b) If the cache clear bit is set while the instruction being cached in the instruction cache is being fetched, a burst transfer is performed immediately due to miss-caching.

(c) When clearing an instruction cache tag by setting the tag clear bit, the tag is cleared correctly but the LRU is not.

[Workaround]

There is no workaround.

These restrictions have been corrected in the IE-V850E-MC with control code B.

These restrictions have been corrected in the IE-V850E-MC-A with control code D.

a-16. Restriction on SDRAM access during bus hold

[Description]

If the external bus master accesses the SDRAM during bus hold, the bus master cannot correctly access the SDRAM because the page information and bank information immediately before the bus hold status is retained. The SDCLK signal goes into a high-impedance state in the bus hold status. The level of this signal before it goes into a high-impedance state is unstable.

[Workaround]

Do not access the SDRAM during bus hold.

The specification has been corrected in the IE-V850E-MC with control code B.

The specification has been corrected in the IE-V850E-MC-A with control code D.

The specification has been changed so that the SDCLK signal can be output during bus.

a-17. Restriction on self-refresh cycle by SELFREF pin

[Description]

If the self-refresh cycle is started by the SELFREF pin, the REFRQ(−) signal is not asserted.

[Workaround]

There is no workaround.

This restriction has been corrected in the IE-V850E-MC with control code B.

This restriction has been corrected in the IE-V850E-MC-A with control code D.

a-18. Restriction on flyby DMA transfer to EDO DRAM

[Description]

When the EDO DRAM is set so that no wait cycle is inserted during on-page access and that the RAS hold mode is enabled, a DMA cycle is stopped in mid-execution, no transfer is executed, and the EDO DRAM cannot be correctly accessed if flyby DMA transfer from the EDO DRAM to an external I/O takes place in the on-page status and during RAS hold.

[Workaround]

Insert at least one cycle of CAS precharge wait or data wait.

This restriction has been corrected in the IE-V850E-MC with control code B.

This restriction has been corrected in the IE-V850E-MC-A with control code D.

a-19. Restriction on EDO DRAM with idle state inserted

[Description]

If an EDO DRAM without a row address hold wait cycle is accessed immediately after an EDO DRAM with an idle state (1 to 3), the former EDO DRAM cannot be correctly read/written.

[Workaround]

Insert at least one cycle of row address hold wait for the EDO DRAM without a row address hold wait cycle.

This bug has been corrected in the IE-V850E-MC with control code B.

This bug has been corrected in the IE-V850E-MC-A with control code D.

a-20. Restriction on flyby DMA transfer

[Description]

If a certain DMA channel is set in a block transfer mode in which flyby DMA transfer is executed from an external memory to an external I/O and if another DMA channel is set in a single, single-step, or line mode in which flyby transfer from an external memory to an external I/O or from an external I/O to external memory is executed, and if these two DMA transfer operations conflict, DMA transfer is not correctly executed.

[Workaround]

If the external memory that executes DMA transfer in the block transfer mode is SRAM, make sure the total number of data wait states (including inserting of a wait state by using the external WAIT pin), address setup wait states, and idle insertion states does not exceed 1.

(Example: Data wait: 1, address setup wait: 0, idle state: 0, with no wait state inserted by external WAIT pin)

If the external memory of the channel that executes DMA transfer in the block transfer mode is EDO DRAM, use either of the channels for two-cycle transfer.

This restriction has been corrected in the IE-V850E-MC with control code B.

This restriction has been corrected in the IE-V850E-MC-A with control code D.

a-21. Restriction on pin status in single-step mode 1 and ROMless modes 0 and 1

[Description]

Some pins do not go into a high-impedance state, but rather output a specific level, when the device is reset in single-chip mode 1 or ROMless mode 0 or 1 (refer to the table below).

Operating Status / Pin	Current Status	After Change
	Reset (Single-Chip Mode 1, ROMless Mode 0,1)	Reset (Single-Chip Mode 1, ROMless Mode 0,1)
A0 to A15 (PAL0 to PAL15)	L	Hi-Z
A16 to A25 (PAH0 to PAH9)	L	Hi-Z
D0 to D15 (PDL0 to PDL15)	L	Hi-Z
$\overline{CS0}$ to $\overline{CS7}$ (PCS0 to PCS7)	H	Hi-Z
$\overline{RAS1}$, $\overline{RAS3}$, $\overline{RAS4}$, $\overline{RAS6}$ (PCS1, PCS3, PCS4, PCS6)	–	–
$\overline{IOWR}$ (PCS2)	–	–
$\overline{IORD}$ (PCS5)	–	–
$\overline{LWR}$, $\overline{UWR}$ (PCT0, PCT1)	H	Hi-Z
$\overline{LCAS}$, $\overline{UCAS}$ (PCT0, PCT1)	–	–
LDQM, UDQM (PCT0, PCT1)	–	–
$\overline{RD}$ (PCT4)	H	Hi-Z
$\overline{WE}$ (PCT5)	H	Hi-Z
$\overline{OE}$ (PCT6)	H	Hi-Z
$\overline{BCYST}$ (PCT7)	H	Hi-Z
$\overline{WAIT}$ (PCM0)	H	Hi-Z
CLKOUT (PCM1)	Operates	Operates
BUSCLK (PCM1)	–	–
$\overline{HLDAK}$ (PCM2)	H	Hi-Z
$\overline{HLDRQ}$ (PCM3)	–	Hi-Z
$\overline{REFRQ}$ (PCM4)	H	Hi-Z
SELFREF (PCM5)	–	Hi-Z
SDCKE (PCD0)	L	Hi-Z
SDCLK (PCD1)	Operates	Hi-Z
$\overline{SDCAS}$ (PCD2)	–	–
$\overline{LBE}$ (PCD2)	H	Hi-Z
$\overline{SDRAS}$ (PCD3)	–	–
$\overline{UBE}$ (PCD3)	H	Hi-Z
$\overline{DMAAK0}$ to $\overline{DMAAK3}$ (PBD0 to PBD3)	H	Hi-Z

Remark Hi-Z: High-impedance
H: High-level output
L: Low-level output
–: Input non-sampling

[Workaround]

There is no workaround. This restriction applies only to the V850E/MA1.

This restriction has been corrected in the IE-V850E-MC-A with control code D.

a-22. Incorrect writeback with LD/SLD instruction when executing CALLT/SWITCH instruction

[Description]

If the bits (bits 15 to 11) corresponding to the halfword WB field immediately after a CALLT/SWITCH instruction match the WB field of an LD/SLD instruction executed immediately before the CALLT/SWITCH instruction, and the CALLT/SWITCH instruction is executed before the bus cycle of the LD/SLD instruction is complete, then the result of LD/SLD is not written back to the register.

The conditions under which the phenomenon occurs are as follows.

1) An LD/SLD instruction is executed
2) Before the bus cycle of the above instruction is complete, a CALLT/SWITCH instruction is executed.
3) Bits 15 to 11 of the halfword after the CALLT/SWITCH instruction in 2) are the same as the writeback register field (bits 15 to 11) of the LD/SLD instruction in 1)

When all the above conditions occur, the results of the LD/SLD instruction in 1) are not written back. One more condition applies in the case of a fetch via VSB, i.e.:

4) There are three or fewer instructions between LD/SLD and CALLT/SWITCH (does not occur with ld-reti-callt/switch.)

[Workaround]

Either of the following two workarounds can be used:

1) Set bits 15 to 11 of the halfword immediately after CALLT/SWITCH to 00000B.
2) Insert "mov r31, r0" immediately before CALLT/SWITCH and at the end of interrupt servicing (before RETI.) (It must be inserted before RETI in case an interrupt occurs between the inserted mov and callt.)

It is not necessary to insert mov before RETI if operation is limited only to fetch via VSB.

This bug has been corrected in the IE-V850E-MC with control code C.

This bug has been corrected in the IE-V850E-MC-A with control code E.

A tool that inserts "mov r31, r0" before CALLT/SWITCH and at the end of interrupt servicing (before RETI) is available as a patch for the compiler. This patch tool is not supplied with the current release of the compiler (from NEC Electronics and third parties) as standard. Consult the NEC Electronics Development Tool Support Center for details.

a-23. Restriction on use of external bus when product is employed as an emulator for the V850E/IA1.

[Description]

The external bus cannot be used when used as emulator of the V850E/IA1.

[Workaround]

There is no workaround.

This restriction has been corrected in the IE-V850E-MC with control code B.

This restriction is not applicable to the IE-V850E-MC-A.

a-24. Restriction on output of the _DMAAK signal

[Description]

Although the _DMAAK signal is intended to remain active during 2-cycle DMA transfer, the _DMAAK(-) signal becomes inactive for one clock cycle between the read cycle and the write cycle. As a result, this product does not operate normally when the number of DMA transfers are counted externally or when external I/O is started with only the rising edge of the DMAAK(-) signal.

[Workaround]

There is no workaround.

This restriction has been corrected in the IE-V850E-MC with control code C.

This restriction has been corrected in the IE-V850E-MC-A with control code E.

a-25. Restriction on starting DMA by built-in peripheral I/O interrupt

[Description]

Depending on the timing, it may not be possible to start DMA by interrupt of the built-in peripheral I/O.

[Workaround]

There is no workaround.

This restriction has been corrected in the IE-V850E-MC with control code C.

This restriction has been corrected in the IE-V850E-MC-A with control code E.

a-26. Restriction on EDO DRAM bus collision

[Description]

There is a possibility of a bus collision in the EDO DRAM when this product is used in RAS hold mode because of an overlap of an EDO DRAM write cycle or the last state (TE state) of a read cycle of an EDO DRAM that does not have an idle state inserted and the first state of the next bus cycle.

[Workaround]

Do not use in RAS hold mold.

This restriction has been corrected in the IE-V850E-MC with control code C.

This restriction has been corrected in the IE-V850E-MC-A with control code E.

a-27. Restriction on the 2-way associative function of the instruction cache

[Description]

WAY control information will be incorrect and instructions corrupted as a result if any address A is a miss-hit, the following address B is a cache-hit, and the address A is a hit.

[Workaround]

There is no workaround.

Use 4K(2K)-Direct mode instead of the 8K(4K)-2way associative function.

This restriction has been corrected in the IE-V850E-MC with control code C.

This restriction has been corrected in the IE-V850E-MC-A with control code E.

a-28. Forced stop of external DMA transfer in DMA line transfer mode

[Description]

The DMA will occupy the VSB when the external DMA transfer stop signal (IDMASTP) is active in DMA line transfer mode. As a result, the CPU will no longer be able to access the VSB and the system may hangs up.

[Workaround]

Do not use IDMASTP in line transfer mode. In addition, use the IDMASTP signal with single-step transfer mode, not line transfer mode.

This bug has been corrected in the IE-V850E-MC with control code C.

This bug has been corrected in the IE-V850E-MC-A with control code E.

a-29. Restriction on reading DCHC register when DMA 2-cycle transfer is complete

[Description]

The TC bit may be cleared to 0 before it is read as 1 when the DCHC register is read upon completion of DMA transfer in 2-cycle transfer mode from one RAM connected to the VDB bus to another RAM connected to the VDB.

[Workaround]

Do not use RAM-to-RAM DMA transfer. Moreover, use three or fewer VSWL registers.

* **This applies only when developing a system LSI. This restriction does not apply when using the V850E/MA1, V850E/MA2, V850E/IA1, and V850E/IA2.**

This restriction has been corrected in the IE-V850E-MC with control code C.

This restriction has been corrected in the IE-V850E-MC-A with control code E.

a-30. Restriction on conflict between SDRAM initialization and SELFREF input

[Description]

If the system enters standby mode or SELFREF pin input generates a self-refresh cycle before initialization of the SDRAM is set (before writing to the SCR register), the following SDRAM cycle will be incorrect.

[Workaround]

Do not allow the system to enter standby mode or SELFREF pin input before SDRAM initialization setting is completed.

This restriction has been corrected in the IE-V850E-MC with control code C.

This restriction has been corrected in the IE-V850E-MC-A with control code E.

a-31. Restriction on halfword writing to BSC, BCC, DWC0, and DWC1 registers

[Description]

Data cannot be written correctly to the higher 8 bits (8 to 15) of the BSC, BCC, DWC0, and DWC1 register with halfword (16-bit) write.

The following bugs occur during emulation memory access as a result of this restriction.

(Not applicable to target memory access.)

- The IDLE-state set with the BCC register is not inserted in CS4 to CS7.
- The bus size for CS4 to CS7 does not change to the value set in the BSC register.
- DATA-wait set with the DWC0 register is not inserted in CS2 to CS3.

- DATA-wait set with the DWC1 register is not inserted in CS6 to CS7.

[Workaround]

Before performing halfword write to set the BSC, BCC, DWC0, and DWC1 registers, byte-write the data which is to be set to the higher 8 bits to the lower 8 bits and then perform halfword write.

Example: When writing 0x1234 to BSC (0xffff066)

Example of access without workaround (The bug occurs in this case.)

```
movhi    0xffff, r0, r10
ori      0xf0000, r10, r10
ori      0x1234, r0, r11
st.h     r11, 0x66[r10]
```

Example of access with workaround (The bug does not occur in this case.)

```
movhi    0xffff, r0, r10
ori      0xf0000, r10, r10
ori      0x12, r0, r11
st.b     r11, 0x66[r10]    ← Write higher 8 bits to lower 8 bits.
ori      0x1234, r0, r11
st.b     r11, 0x66[r10]    ← Halfword write 0x1234
```

This bug has been corrected in the IE-V850E-MC with control code C.

This bug has been corrected in the IE-V850E-MC-A with control code E.

a-32. Restriction on SDRAM write operation

[Description]

Data output from the second cycle in a sequence of SDRAM write cycles completes half a clock cycle earlier than originally intended, and, as a result it may not be possible to write to the SDRAM.

[Workaround]

Do not perform word data access when using the data bus with a 16-bit width. In the case of an 8-bit bus width, do not perform halfword data access or word data access.

This restriction has been corrected in the IE-V850E-MC with control code C.

This restriction has been corrected in the IE-V850E-MC-A with control code E.

a-33. Restriction on DRAM fetch immediately after block DMA transfer from DRAM to internal RAM

[Description]

The OE is not asserted at the first fetch cycle if DRAM fetch is performed immediately after block DMA transfer from DRAM to internal RAM. As a result, illegal data is fetched.

[Workaround]

There is no workaround.

This restriction has been corrected in the IE-V850E-MC with control code B.

This restriction has been corrected in the IE-V850E-MC-A with control code D.

a-34. Restriction on instruction cache (2)

[Description]

(1) Be sure to execute the instruction below before setting the cache configuration register (BHC) with the program settings immediately after a reset.

```
st.h  r0, 0x0fffff072[r0]
```

After executing this instruction, the cache can be enabled by setting the BHC register to "instruction cache enabled."

(2) The auto-fill function can only be used with WAY0. Be sure to write 0 to bit 5 of the ICC register.

(3) An instruction that sets the BHC register cannot be used to change the cache settings of the area in which the instruction itself exists. For example, A BHC setting instruction in the CS0 field cannot be used to change the CS0 field from a cache disabled area to a cache enabled area, or from a cache enabled area to a cache disabled area.
However, instructions in a memory area connected to VFB or VDB can be used to change cache settings for all CS fields.

[Workaround]

There is no workaround. Please regard this as a permanent restriction.

a-35. Restriction on SLD instruction

[Description]

The bug occurs in the following three types of instruction sequences (The asterisks below indicate any parameter).

In a situation where the value loaded by an instruction (this value is stored in register rX[1)]) is updated[2)] by another instruction, if an SLD instruction (sld *,rX) follows immediately after the instruction to update rX (yyy *,rX) , these two instructions are issued at the same time. At this time, if an instruction that follows the SLD instruction uses the rX register, the value that was loaded first (not updated) is used by the instruction[3)] even though the register is correctly updated.

Instruction sequence type 1

```
<1> xxx *
<2> sld * , rX                    ← 1) The value of rX register is loaded by the first
                                      instruction
<3> yyy * , rX                    ← 2) rX is updated by another instruction
<4> sld * , rY
<5> Instruction that uses value of rX  ← 3) The old rX value from <2> is used instead of the
                                      updated rX value from <3>.
```

Instruction sequence type 2

<1> ld/sld * , rX ← 1) The value of rX register is loaded by the first instruction

<2> xxx (One or more instructions)

<3> yyy * , rX ← 2) rX is updated by another instruction

<4> sld * , rY

<5> Instruction that uses value of rX ← 3) The old rX value from <1>is used instead of the updated rX value from <3>.

Instruction sequence type 3

<1> ld/sld * , rX ← 1) The value of rX register is loaded by the first instruction

<2> Any instruction sequence that does not perform a memory read.

<3> xxx (One or more instructions)

<4> yyy * , rX ← 2) rX is updated by another instruction

<5> sld * , rY

<6> Instruction that uses value of rX ← 3) The old rX value from <1> is used instead of the updated rX value from <4>.

The conditions under which this bug occurs differ depending on the combination of the CPU and emulator used. The conditions of occurrence for each CPU are explained below.

a) Details and conditions of occurrence on system LSI development
(Emulator IE-V850E-MC-A + IE-V850E-MC-EM1-A/B)
The bug occurs if any of the three types of instruction sequence shown below is executed with one of the following fetch conditions.

1) Fetch from emulation IROM or IRAM
2) Fetch from VSB bus when 32-bit access is 2clk or less
   2a) 32bit-0/1wait or 16bit-0wait on VSB bus
   2b) SRAM cycle with 32bit-0wait when using Nx85E500 memory controller
3) Fetch from VSB bus under all conditions when iCACHE enabled area is used

Instruction sequence type 1

<1> xxx * (Instruction in **Note 1**)

<2> sld * , rX (Emulator accesses IRAM)

<3> yyy * , rX (Instruction in **Note 2**)

<4> sld * , rY (Emulator accesses IRAM)

<5> Instruction that uses value of rX

Instruction sequence type 2

<1> ld/sld * , rX (Emulator accesses IRAM)
<2> xxx (One or more instructions) (Instruction in **Note 3**)
<3> yyy * , rX (Instruction in **Note 2**)
<4> sld * , rY (Emulator accesses IRAM)
<5> Instruction that uses value of rX

Instruction sequence type 3

<1> ld/sld * , rX (VSB bus access)
<2> Any instruction sequence that does not perform a memory read.
<3> xxx (One or more instructions) (Instruction in **Note 3**)
<4> yyy * , rX (Instruction in **Note 2**)
<5> sld * , rY (Emulator accesses IRAM)
<6> Instruction that uses value of rX

b) Details and conditions of occurrence with V850E/MA1
(Emulator IE-V850E-MC-A + IE-703107-MC-EM1)
The bug occurs if any of the three types of instruction sequence shown below is executed with a fetch from emulator IRAM or IROM.

Instruction sequence type 1

<1> xxx * (Instruction in **Note 1**)
<2> sld * , rX (Emulator accesses IRAM)
<3> yyy * , rX (Instruction in **Note 2**)
<4> sld * , rY (Emulator accesses IRAM)
<5> Instruction that uses value of rX

Instruction sequence type 2

<1> ld/sld * , rX (Emulator accesses IRAM)
<2> xxx (One or more instructions) (Instruction in **Note 3**)
<3> yyy * , rX (Instruction in **Note 2**)
<4> sld * , rY (Emulator accesses IRAM)
<5> Instruction that uses value of rX

Instruction sequence type 3

<1> ld/sld * , rX (See ♦1 below)
<2> Any instruction sequence that does not perform a memory read.
<3> xxx (One or more instructions) (Instruction in **Note 3**)
<4> yyy * , rX (Instruction in **Note 2**)
<5> sld * , rY (Emulator accesses IRAM)
<6> Instruction that uses value of rX

♦1: Only when an ld/sld instruction is fetched from EDO DRAM for which data access wait is 0, or SDRAM. Does not apply in the case of access to page ROM, SRAM or EDO DRAM for which data access wait is 1 or more.

c) Details and conditions of occurrence with V850E/IA1
(Emulator IE-V850E-MC + IE-703116-MC-EM1)
The bug occurs if any of the two types of instruction sequence shown below is executed with a fetch from emulator IRAM or IROM.

Instruction sequence type 1

<1> xxx *	(Instruction in **Note 1**)
<2> sld * , rX	(Emulator accesses IRAM)
<3> yyy * , rX	(Instruction in **Note 2**)
<4> sld * , rY	(Emulator accesses IRAM)
<5> Instruction that uses value of rX	

Instruction sequence type 2

<1> ld/sld * , rX	(Emulator accesses IRAM)
<2> xxx (One or more instructions)	(Instruction in **Note 3**)
<3> yyy * , rX	(Instruction in **Note 2**)
<4> sld * , rY	(Emulator accesses IRAM)
<5> Instruction that uses value of rX	

Instruction sequence type 3 is not applicable to the V850E/IA1.

Note 1: One of the following instructions using a register other than r0 or r30.
mov, not, satsubr, satsub, satadd, zxb, zxh, sxb, sxh, or, xor, and, subr, sub, add, shr, sar, shl

Note 2: One of the following instructions, writing to rX (rX is any register other than r0 and r30.)
mov, not, satsubr, satsub, satadd, zxb, zxh, sxb, sxh, or, xor, and, subr, sub, add, shr, sar, shl

Note 3: One of the following instructions, writing to any register other than r0 and rX, that does not use rX.
mov, not, satsubr, satsub, satadd, zxb, zxh, sxb, sxh, or, xor, and, subr, sub, add, shr, sar, shl, addi, movea, movhi, satsubi, ori, xori, andi, setf, ldsr, stsr, sasf, cmov, bsw, bsh, hsw

[Workaround]

NEC Electronics provides a check tool. Use this tool to check for the affected instruction sequence. Refer to the accompanying Readme file for details of how to use the check tool. Use one of the following remedies to avoid the problem if the check tool finds an affected instruction sequence.

a) Replace all SLD instructions with LD instructions.
This is possible using a compiler option with the GHS compiler only.

b) Insert a NOP instruction immediately after the affected SLD instruction.
The NEC Electronics compiler (CA850) provides an option to do this.

c) Change the instruction immediately before the affected SLD instruction. Assembly code can be modified if the SLD instruction is not a branch destination.
Use workaround a) or b) if you are using a C source compiler.

This restriction has been corrected in the IE-V850E-MC with control code D.
This restriction has been corrected in the IE-V850E-MC-A with control code F.

* Contact the Development Tool Support Center at the address or number below for details of the search tool and compiler updates.

a-36. I/O that cannot be used when using a VSB bus

[Description]

The following I/O addresses cannot be used when using a VSB bus (including image area).

64 MB mode: 3FFF480H to 3FFF4BEH
256 MB mode: FFFF480H to FFFF4BEH

[Workaround]

There is no workaround.
This restriction has been corrected in the IE-V850E-MC with control code D.
This restriction has been corrected in the IE-V850E-MC-A with control code F.

a-37. Restriction of instruction cache (3)

(1) A bug occurs depending on the timing of an interrupt issued to the CPU. If execution returns from an interrupt when the cache is not refilled with data, a hit occurs at an address that is not refilled. As a result, the CPU fetches illegal data and hangs up.
(2) If TAG is cleared by the ICC register, four lines of TAG (one line of cache data consists of 4 words), lines 0 through 3 (INDEX = 00h through 03h), may not be cleared, depending on the operation status of the cache before TAG is cleared.
(3) If the halt instruction is executed and then the halt mode is released, icache may return an illegal instruction to the CPU, resulting in a hang-up.

Conditions of occurrence

Bug (1) occurs under the following condition.
If a memory area including interrupt handler addresses (0x00000000h to 0x00000800h) is set as a cacheable area by the BHC register.

This bug does not occur under the following conditions.

1) If the memory area including interrupt handler addresses is set as an uncached area
2) If the memory area including interrupt handler addresses is connected to VFB and if bits 0, 4, 8, and 12 of the chip area select control register 0 (CSC0) are all cleared to 0 (so that VDCSZ does

not become active)

Bug (2) occurs in all the conditions where the instruction cache is used.

Bug (3) occurs under the following conditions.
If the halt instruction is in the icache visible area and if that halt instruction is not at the beginning of a 4-word boundary (If the halt instruction at an address other than address [3:0] = 4'b0000)

[Workaround]

Workaround for restriction (1)

Implement any of the following workarounds <1> to <3> below in software.

<1> Set the memory area including interrupt handler addresses (0x00000000h to 0x00000800h) as an uncached area using the BHC register.

<2> If it is necessary to set a CS (chip select) area including interrupt handler addresses as a cacheable area and if an uncached memory area exists on the other CS area.

Branch to the uncached area once and then branch to the original interrupt service routine after branching to the interrupt handler address. Add the shaded instructions below.

(Example)

```
                .offset 0x80 (Interrupt handler address)
                jr          INT0_UC

                .section "uncache"      -- Uncached area
INT_UC:
                ir          INT0

                .section "cache"        -- Cacheable area (original interrupt service routine)

INT0:
                …
                …
                …
                reti
```

<3> If it is necessary to set a CS (chip select) area including interrupt handler addresses as a cacheable area and if an uncached memory area does not exist in the other CS area

Immediately after branching to the interrupt handler address, branch to different TAG lines in the cacheable area more than four times, and then branch to the original interrupt service routine. Add the shaded instructions below.

(Example)

```
        .offset 0x80 (Interrupt handler address)
        jr      INT0_DMY

        .section “dmy”          -- Cacheable area
INT0_DMY:
        jr      line0
        nop; nop; nop; nop; nop; nop; nop;
line0:
        jr      line1
        nop; nop; nop; nop; nop; nop; nop;
line1:
        jr      line2
        nop; nop; nop; nop; nop; nop; nop;
line2:
        jr      line3
        nop; nop; nop; nop; nop; nop; nop;
line3:
        jr      INT0

          .section “cache”          -- Cacheable area (original interrupt service routine)
INT0:                                  (It is necessary that an instruction string of 1 line (4
                                       words) or more exist to completely update the
                                       contents of the TAG buffer.)
          …
          …
          …
          reti
```

This restriction has been corrected in the IE-V850E-MC with control code D.

This restriction has been corrected in the IE-V850E-MC-A with control code F.

Workaround for restriction (2)

To clear TAG in software, clear TAG two times (write to the ICC register and wait for completion).

(Example)

```
            mov     0x3,r2
    LOP0:
            ld.h    ICC[r0],r1
            cmp     r0,r1
            bnz     LOP0
            st.h    r2,ICC[r0]
    LOP1:                                   -- TAG cleared first time
            ld.h    ICC[r0],r1
```

```
                cmp     r0,r1
                bnz     LOP1
                st.h    r2,ICC[r0]
        LOP2:                                   -- TAG cleared second time
                ld.h    ICC[r0],r1
                cmp     r0,r1
                bnz     LOP2
```

Please regard this as a permanent restriction.

Workaround for restriction (3)

Allocate the halt instruction to the beginning of a 4-word boundary by software.

(Allocate the halt instruction to address [3:0] = 4'b0000.)

(Example in assembler)

```
jr  EXIT
        .section  "tmp2",  .text  >0x0402800        -- Clear the least significant bit to 0 in the hex mode.
    EXIT:
        halt
```

(Example in C)

```
asm("align 16");
asm("halt");
```

This restriction has been corrected in the IE-V850E-MC with control code D.

This restriction has been corrected in the IE-V850E-MC-A with control code F.

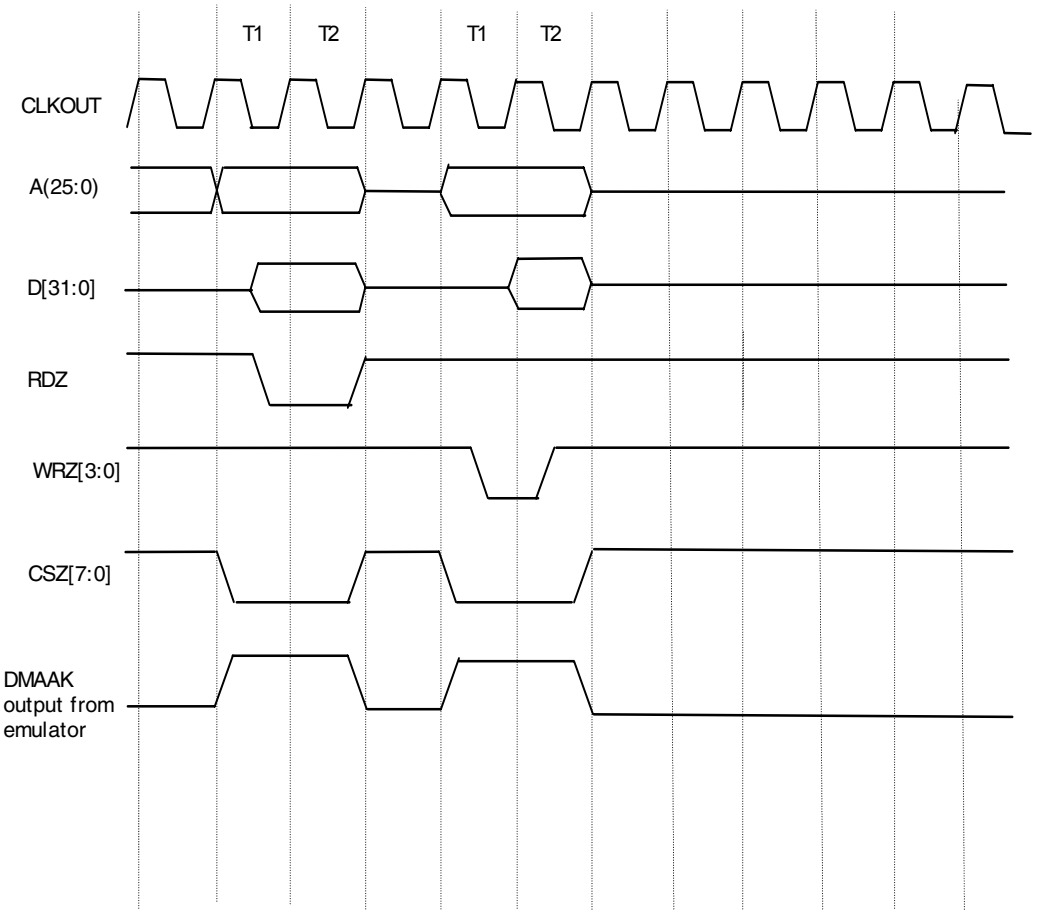
a-38. Restriction on DMAAK signal during DMA line transfer

[Description]

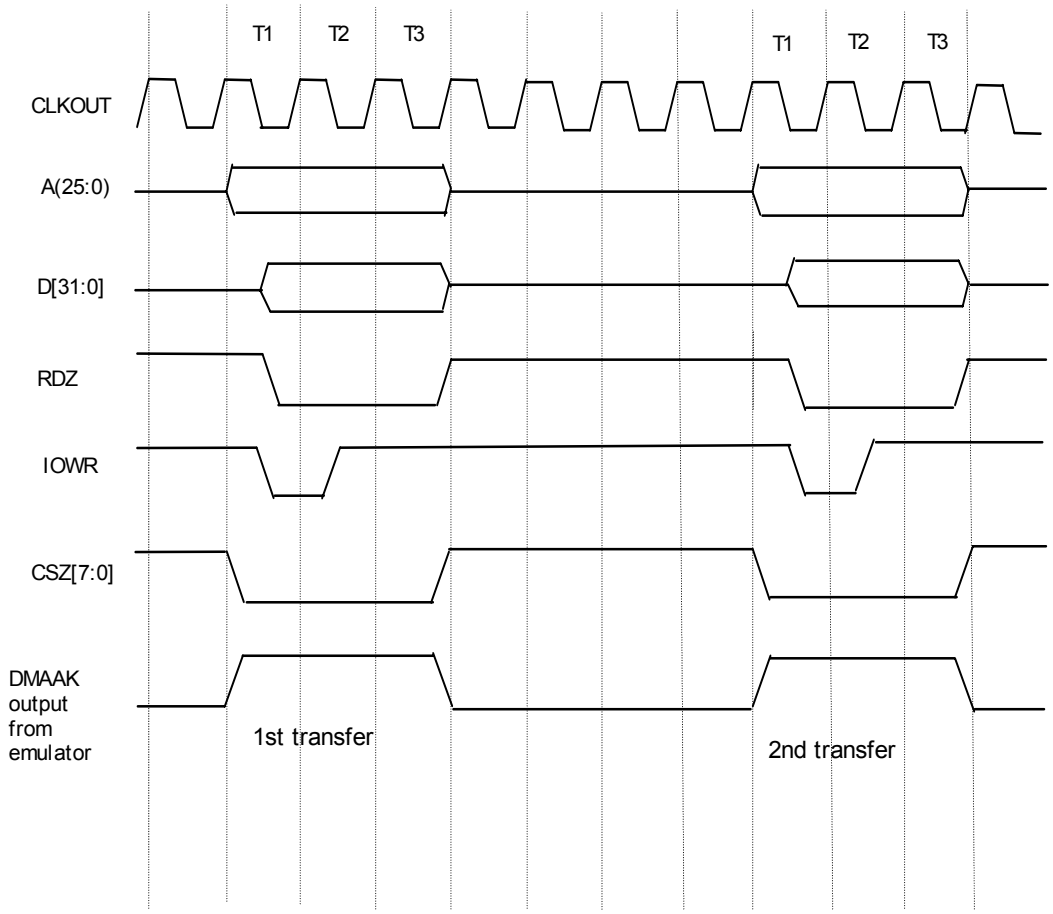
The DMAAK signal is output at the same timing as the NB85E core output signal during DMA line transfer (the output timing from an in-circuit emulator is below).

- When developing V850E/MA1, MA2:
  This restriction does not apply because these products do not support line transfer.
- When developing V850E/IA1, IA2:
  This restriction does not apply because these products do not support line transfer.
- When developing a system LSI:
  This restriction does not apply and the DMAAK signal is output correctly as a DMACTV signal.

2-cycle line transfer timing from in-circuit emulator (external SRAM → external SRAM)



Flyby line transfer timing from in-circuit emulator (external SRAM → external I/O)



a-39. Restriction caused by interrupt input during execution of bit manipulation instruction

[Description]

If a bit manipulation instruction access (set1, clr1, or not1) and an interrupt request to the following peripheral registers (in the NPB field) contend, two more addresses are illegally written to the interrupt return address saving registers (EIPC and FEPC) as interrupt return addresses, and execution branches to the illegal addresses when it has returned from the interrupt routine.

The contending interrupts are all maskable interrupts and the non-maskable interrupt (NMI).

- Peripheral registers –

- When developing V850E/MA1 or IA1
  - All registers mapped to 0xFFFF100 to 0xFFFF1FF
  - All registers mapped to 0xFFFF900 to 0xFFFF9FF
- When developing a system LSI (64 MB mode)
  - All registers mapped to 0x3FFF100 to 0x3FFF1FF
  - All registers mapped to 0x3FFF900 to 0x3FFF9FF
- When developing a system LSI (256 MB mode)
  - All registers mapped to 0xFFFF100 to 0xFFFF1FF
  - All registers mapped to 0xFFFF900 to 0xFFFF9FF

This bug occurs in the in-circuit emulator (IE-V850E-MC-A or IE-V850E-MC) and does not apply to the target device (V850E/MA1, IA1, or NB85E core).

Caution This bug does not occur if all the following conditions are satisfied:

- Interrupts are always disabled (DI status) when the peripheral registers are accessed.
- If NMI is not used or if NMI is never enabled when a peripheral register is accessed (when NMI is used to release the standby mode)

[Workaround]

(1) The above bug can be prevented in any of the following ways if NMI is not used or if NMI is never enabled when a peripheral register is accessed (when NMI is used to release the standby mode):

a) Always access the peripheral register (by using a bit manipulation instruction) in the DI status (interrupts are disabled).

Save PSW
→ di instruction
→ Access register with bit manipulation instruction.
→ Restore PSW

b) Disable interrupts (pending) by setting the interrupt mask registers (IMR0 to IMR3) when the peripheral register is accessed (by using a bit manipulation instruction).

Save values of IMR0 to IMR3.
→ Write values of IMR0 to IMR3 to 0xffff (disable interrupt servicing).
→ Access peripheral register (by using bit manipulation instruction).
→ Write saved values of IMR0 to IMR3.

c) Do not use a bit manipulation instruction to access the peripheral register. Use “ld instruction → bit operation with and/or instruction → st instruction” instead to manipulate bits.

- Refer to “NEC Electronics Compiler Coding Example” for coding with the NEC Electronics compiler.
- Refer to “GHS Compiler Coding Example” for coding with the GHS compiler.

(2) If there is a possibility that NMI is enabled while the peripheral register is accessed, do not use a bit manipulation to access the peripheral register. Use “ld instruction → bit operation with and/or instruction → st instruction” to execute a bit operation and to prevent the above bug.

- Refer to “NEC Electronics Compiler Coding Example” for coding with the NEC Electronics compiler.
- Refer to “GHS Compiler Coding Example” for coding with the GHS compiler.

This restriction has been corrected in the IE-V850E-MC with control code D.
This restriction has been corrected in the IE-V850E-MC-A with control code F.

[Other]

(1) If you use the real-time OS (RX850 or RX850 Pro)
This bug does not apply to task processing because the task processing of the real-time OS does not use bit manipulation instructions.

(2) The results of investigations into the middleware are as follows. Consult the Development Tool Support Center (toolsupport@lsi.jp.nec.com) for middleware other than shown below.

AP30100-B03	Middleware (JPEG)	Not applicable
AP30100-B12	Audio codec middleware (True Speech8.5)	Not applicable
AP30200-B03	Middleware (JPEG)	Not applicable
AP703000-B01	Middleware (MH/MR/MHR)	Not applicable
AP703000-B02	Middleware (JBIG)	Not applicable
AP703000-B03	Middleware (JPEG)	Not applicable
AP703000-B04	Middleware (ADPCM)	Not applicable
AP703000-B07	Middleware (speech recognition)	Applied to user-own coding block
AP703000-B08	Middleware (TTS)	Not applicable
AP703000-B09	Middleware hand-written character recognition	Not applicable

[NEC Electronics Compiler Coding Example]

a) To rewrite all the bit manipulation instructions accessing the peripheral register to “ld instruction → bit operation with and/or → st instruction” for coding in the assembler

- Avoiding set1 instruction

  Here is an example to set the third bit of IOR0 (peripheral register) to 1. The value of the bit that is set to 1 is ORed.

```
(Description before correction)    set1    3, IOR0
(Description after correction)     ld.b    IOR0, rX
                                   or      0x8, rX
                                   st.b    rX, IOR0
```

- Avoiding clr1 instruction

  Here is an example to clear the third bit of IOR0 to 0.
  The value of the bit that is cleared to 0 is ANDed.

```
(Description before correction)    clr1    3, IOR0
(Description after correction)     ld.b    IOR0, rX
                                   and     0xf7, rX
                                   st.b    rX, IOR0
```

- Avoiding not1 instruction

  Here is an example to negate the third bit of IOR0.

  The value of the bit that is negated (to 0) is ANDed.

```
(Description before correction)    not1    3, IOR0
(Description after correction)     ld.b    IOR0, rX
                                   andi    0x08, rX rZ     -- 0x08 = 0b00001000
                                   andi    0xf7, rX rY     -- 0xf7 = 0b11110111
                                   xor     0x08, rZ        -- 0x08 = 0b00001000
                                   or      rY, rZ
                                   st.b    rZ, IOR0
                                   jmp     [lp]
```

b) To avoid generating bit manipulation instruction code for coding of C source

The compiler mainly outputs a bit manipulation instruction when a bit field is used.

Therefore, rewrite the coding as follows so that a bit instruction is not output.

- Avoiding output of set1 instruction

  Here is an example of setting the third bit of IOR0 to 1.

  The value of only the bit that is set to 1 is ORed.

```
(Source in question)    IOR0.3 = 1;
(Corrected source)      IOR0 | = 0x08;                  /* 0x08 = 0b00001000 */
```

- Avoiding output of clr1 instruction

  Here is an example of clearing the third bit of IOR0 to 0.

  The value of only the bit that is cleared to 0 is ANDed.

  The value of only the bit that is set to 1 is ORed.

```
(Source in question)    IOR0.3 = 0;
(Corrected source)      IOR0 & = 0xf7;                  /* 0xf7 = 0b11110111 */
```

- Avoiding output of not1 instruction

  Here is an example of negating the third bit of IOR0.

  If the bit in question is 1, the value when that bit is cleared to 0 is ANDed.

  If the bit is 0, the value when that bit is set to 1 is ORed.

```
(Source in question)    IOR0.3 = ~IOR0.3;
(Corrected source)      if(IOR0.3){
                                IOR0 & = 0xf7;   /* 0xf7 = 0b11110111 */
                                }
                        else{
                                IOR0 | = 0x08;   /* 0x08 = 0b00001000 */
                            }
```

- Other exceptional patterns

  Here is an example of assigning the second bit of IOR0 to the first bit of P0.

  If the value to be assigned is 1, the value when that bit is set to 1 is ORed.

  If the value is 0, the value when that bit is cleared to 0 is ANDed.

```
(Source in question)    IOR0.1 = IOR0.2;
(Corrected source)      if(IOR0.2){
                                IOR0 | = 0x02;          /* 0x02 = 0b00000010 */
                                }
                        else{
                                IOR0 & = 0xfd;          /* 0xfd = 0b11111101 */
                            }
```

[GHS Compiler Coding Example]

The GHS compiler has a compile option that prevents output of a bit manipulation instruction.

Correct the source as shown in the following example.

It is assumed that the file configuration is as follows.

File configuration before correction

File Name	Function Name	Outline of Function
a.c	a_sub1()	Subroutine 1 of a.c *1
	a_sub2()	Subroutine 2 of a.c *1
	a_ior1()	Subroutine 3 of a.c *2
b.c	b_sub1()	Subroutine 1 of b.c *1
	b_ior1()	Subroutine 2 of b.c *2
c.c	c_sub1()	Subroutine 1 of c.c *1
	c_sub2()	Subroutine 2 of c.c *1
	c_ior1()	Subroutine 3 of c.c *2
	c_ior2()	Subroutine 4 of c.c *2

*1: The peripheral register is not accessed *2: The peripheral register is accessed

Of the above, only the access to the I/O register is extracted and assumed as d.c.

File configuration after correction

File Name	Function Name	Outline of Function
a.c	a_sub1()	Subroutine 1 of a.c
	a_sub2()	Subroutine 2 of a.c
b.c	b_sub1()	Subroutine 1 of b.c
c.c	c_sub1()	Subroutine 1 of c.c
	c_sub2()	Subroutine 2 of c.c
d.c	a_ior1()	Subroutine 3 of a.c
	b_ior1()	Subroutine 2 of b.c
	c_ior1()	Subroutine 3 of c.c
	c_ior2()	Subroutine 4 of c.c

When compiling the source, specify compile option “-Z982” for d.c only.

This prevents output of only the bit manipulation instruction that accesses IOR.

a-40. Restriction on hardware stop during bit manipulation instruction execution

[Description]

If the hardware stop signal is sampled between the read cycle and write cycle while a bit manipulation instruction is being executed, the STOP mode is set before the write cycle is executed.

The write cycle is executed after the hardware stop mode has been released.

* This applies only when developing a system LSI. This is a bug that occurs in the in-circuit emulator and does not occur in the target device (NB85E core).

[Workaround]

There is no workaround.

This restriction has been corrected in the IE-V850E-MC with control code D.

This restriction has been corrected in the IE-V850E-MC-A with control code F.

After correction, the STOP mode is set after execution of the write cycle.

a-41. Restriction related to interruption of DMA transfer by external cause

(1) When using the V850E/MA1 or V850E/IA1

[Description]

If a DMA single transfer triggered by internal peripheral I/O interrupt or DMARQn signal input (V850E/MA1 only) is aborted by an NMI input, the contents of the Enn bit are not saved in the DDIS register. Consequently, the aborted DMA transfer cannot be resumed by using the DRST register.

The bug does not occur if any of the following conditions are met.

- Single-step transfer mode or block transfer mode is used (this condition is only applicable to the channels using single transfer mode.)
- An NMI is not input during a DMA transfer.
- DMA is activated using a software trigger only.
- The DMA restart function by the DRST register is not used.

[Workaround]

Implement any of the following workarounds.

This restriction has been corrected in the IE-V850E-MC with control code D.

This restriction has been corrected in the IE-V850E-MC-A with control code F.

Workaround 1

Execute a dummy single transfer with one of DMA channels 0 to 3 before activating the first DMA. After that, do not end or abort transfer of that channel.

* If this workaround is used, one of the four channels will become unusable. Use this workaround

when there is a redundant DMA channel.

Set the channel that executes the dummy transfer as follows.

Step 1. Specify the same address for DSAX and DDAX.

Step 2. Set DBCX to a value of 0001 or greater (number of times of transfer: 2 min.).

Step 3. Set DADCX to 0000H.

Step 4. Set DCHCX to 03H.

(X: Of DMA channels 0 to 4, the DMA channel number that is not used)

Workaround 2

Be sure to execute a software-triggered dummy transfer once immediately before executing a single DMA transfer.

* All the DMA channels can be used by this method. Note, however, that the source and destination of the transfer are changed as a result of the dummy transfer and that the number of times of transfer must be set in advance.

Workaround 3

Be sure to use a software trigger to activate DMA, rather than an interrupt from the internal peripheral I/O or input to the DMARQn pin.

Workaround 4

When DMA is interrupted, save Enn (bit 0 of the DCHCX register) to the user space. When DMA is resumed, set the corresponding bit of the DRST register to 1 by viewing the saved value.

(2) When developing system LSI

[Description]

If a single DMA transfer that has been activated by DMARQn pin input is interrupted by inputting a signal of one VBCLK clock width to the IDMASTP pin, the contents of the Enn bit are not saved to the DDIS register. Consequently, the DMA transfer that has been interrupted by using the DRST register cannot be resumed.

This bug does not occur in any of the following cases:

- When an active level (high level) of 2 clocks is input to the IDMASTP pin at the rising edge of VBCLK
- When DMA is not executed in the single transfer mode
- If DMA is not interrupted
- If a software trigger is used to activate DMA
- If at least one of the DMA channels waits for a software trigger in the single transfer mode when an active level (high level) is input to the IDMASTP pin

[Workaround]

Implement any of the following workarounds.

This restriction has been corrected in the IE-V850E-MC with control code D.

This restriction has been corrected in the IE-V850E-MC-A with control code F.

Workaround 1

Input an active level (high level) of 2 clocks to the IDMASTP pin at the rising edge of VBCLK.

Workaround 2

Execute a dummy single transfer with one of DMA channels 0 to 3 before activating the first DMA. After that, do not end or abort transfer of that channel.

* If this workaround is used, one of the four channels will become unusable. Use this workaround when there is a redundant DMA channel.

Set the channel that executes the dummy transfer as follows.

Step 1. Specify the same address for DSAX and DDAX.

Step 2. Set DBCX to a value of 0001 or greater (number of times of transfer: 2 min.).

Step 3. Set DADCX to 0000H.

Step 4. Set DCHCX to 03H.

(X: Of DMA channels 0 to 4, the DMA channel number that is not used)

Workaround 3

Be sure to execute a software-triggered dummy transfer once immediately before executing a single DMA transfer.

* All the DMA channels can be used by this method. Note, however, that the source and destination of the transfer are changed as a result of the dummy transfer and that the number of times of transfer must be set in advance.

Workaround 4

Be sure to use a software trigger to activate DMA, rather than an interrupt from the internal peripheral I/O or input to the DMARQn pin.

Workaround 5

When DMA is interrupted, save Enn (bit 0 of the DCHCX register) to the user space. When DMA is resumed, set the corresponding bit of the DRST register to 1 by viewing the saved value.

a-42. Restriction on SDCKE signal during bus hold

[Description]

During the bus hold status, the SDCKE pin in the case of the V850E/MA1, MA2 and the CKE pin of the memory controller in the case of system LSI development should output a high level signal. However, the output of these pins becomes high impedance instead of high level.

This bug does not apply if any of the following conditions is satisfied.

1) The V850E/IA1, IA2 is used
2) SDRAM is not used
3) SDRAM is used, but the bus hold function is not used
4) Both SDRAM and the bus hold function are used, but a pull-up resistor is connected to the SDCKE pin
5) The external bus master does not access SDRAM during the bus hold status

[Workaround]

This bug can be avoided by connecting a pull-up resistor to the SDCKE or CKE pin.

This bug is not applicable to the IE-V850E-MC.

This restriction has been corrected in the IE-V850E-MC-A with control code G.

a-43. Caution regarding SDRAM controller

[Description]

A refresh cycle may be executed for the SDRAM immediately after the RFNn bit of the SDRAM refresh control register (RFSn) is set (1: refresh operation enabled). Note, however, that operations during or immediately after the refresh cycle generated at that time are not affected and that subsequent refresh cycles are executed normally at the set interval. (n = 1, 3, 4, 6)

[Workaround]

Operations during or immediately after the refresh cycle generated by the RFSn register's setting are not affected, and subsequent refresh cycles are executed normally at the set interval. However, in applications in which this bug causes problems, take workarounds by setting the RFSn register using the following procedure.

(1) Set the BTn1 and BTn0 bits of the BCTn register to 01 (page ROM connected) while the MEn bit is set (1). (n = 0 to 7)
(2) Set the RENn bit of the RFSn register (1) to enable refresh. (n = 1, 3, 4, 6)
(3) Set the BTn1 and BTn0 bits of the BCTn register to 11 (SDRAM connected) while the MEn bit is set (1). (n = 0 to 7)
(4) Set the SCRn register to initialize the SDRAM. (n = 1, 3, 4, 6)

Please regard this as a permanent restriction.

a-44. Restriction on mul/mulu instruction

[Description]

In the mul and mulu instructions, if an interrupt occurs during execution of an instruction that uses the same register for the 1st and 3rd operands, the operation result (register value of the 3rd operand) may be illegal. The instruction execution is terminated and the subsequent instruction is executed.

When the NEC Electronics C compiler is used, there is no problem as long as this restriction does not affect the description in the assembly language. The global search function, etc., in the Project Manager can be used to confirm the existence of such a description.

Refer to [Related products] below for information about GHS, Inc., Red Hat Inc. and Wind River Systems, Inc.

Example)

```
mul     reg1, reg2, reg1
```

And

```
mulu    reg1, reg2, reg1
```

; Registers reg1 and reg2 are not identical. reg1 ≠ r0 (zero register).

This restriction does not apply if the register used for the 1st an 3rd operands is not identical.

The NEC Electronics C compiler does not create the instruction format to which this restriction applies. In addition, the real-time OSs (RX850 and RX850 Pro), all middleware products do not use the instruction format to which this restriction applies.

[Workaround]

Describe the program as shown below.

```
mul     reg1, reg2, reg3
```

And

```
mulu    reg1, reg2, reg3
```

; **Registers reg1, reg2, and reg3 are not identical.** reg3 ≠ r0.

Or

```
mov     reg1, rtmp
mul     rtmp, reg2, reg1
```

And

```
mov     reg1, rtmp
mulu    rtmp, reg2, reg1
```

; **Registers reg1, reg2, and rtmp are not identical.** reg1 and rtmp ≠ r0.

[Related products]

- GHS products

In the C compiler up to Ver.1.8.9, the instruction format to which this restriction applies may be selected and created if the customer uses the embedded function __MULSH() or __MULUH(), and depending on the optimization mode setting.

Extract all the mul instructions by using "% gdump a.out | grep mul" by using the "gdump", which is a disassembler included in the GHS compiler, to check for the existence of the instruction to which this restriction applies.

In MULTI2000 Rel.3.5 and later, this instruction format will not be created in both C description and assembly language description, and the instruction will be detected as an error when assembling the program. MULTI2000 Rel.3.5 is scheduled for release in February 2002 in the

US and March 2002 in Japan. Contact Advanced Data Controls, Corp. for the detailed schedule. For the runtime library, the mul/mulu instructions are used but this instruction format is not.

- Products of Red Hat Inc. and Wind River Systems, Inc.
  The GNU compiler from these companies does not create the instruction format to which this restriction applies.

Please regard this as a permanent restriction.

a-45. Restriction on page ROM access

[Description]

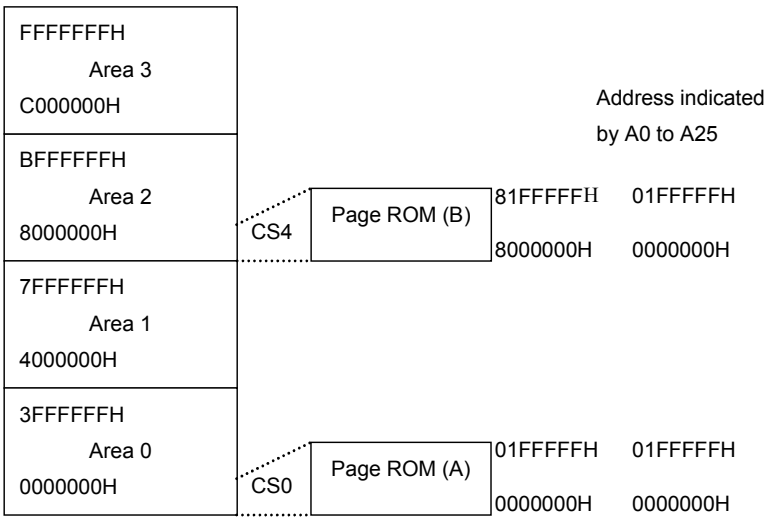
When a CPU data space is used in a system in which multiple page ROMs are connected to multiple CS areas, when a page ROM access and page ROM access to another CS area occur in succession, if the address value of the former address and that of the latter are within the same page length of the page ROM, it is judged that the same page ROM was accessed, even though the page ROMs with different CS areas were accessed. As a result, an on-page cycle will be issued for the latter access, but because the data access time for the latter access is insufficient, data cannot be read correctly.

Remark Page ROM includes memory that is capable of accessing continuous addresses on a page at high-speed, such as mask ROM with a page access function.

Example:

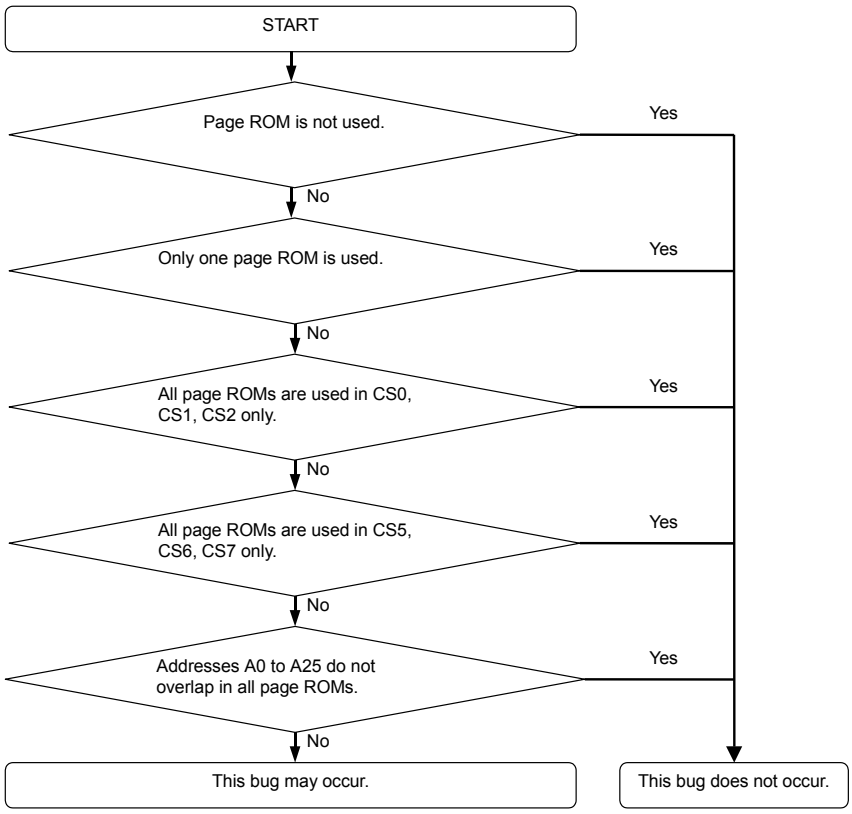
When accessing address 0xxxxx0H in the CS0 area is followed by accessing address 8xxxxx2H in the CS4 area, an on-page cycle will be issued for address 8xxxxx2H.

Example of configuration in which this bug can occur



Bug check procedure

The following flowchart can be used to check whether this bug occurs or not. Confirm if this bug applies your company's product using this flowchart.

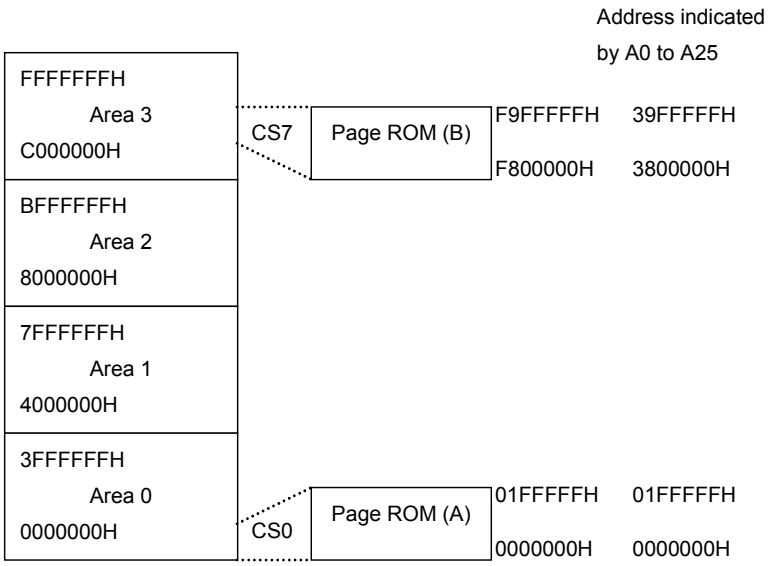


[Workaround]

When multiple page ROMs are used, allocate addresses 25 to 0 so that they do not overlap in each page ROM.

For example, when allocating two 2 MB page ROMs to different CS areas, allocate one page ROM within addresses 0000000H to 01FFFFFH, and the other within addresses F800000H to F9FFFFFH.

Example of configuration in which this bug does not occur



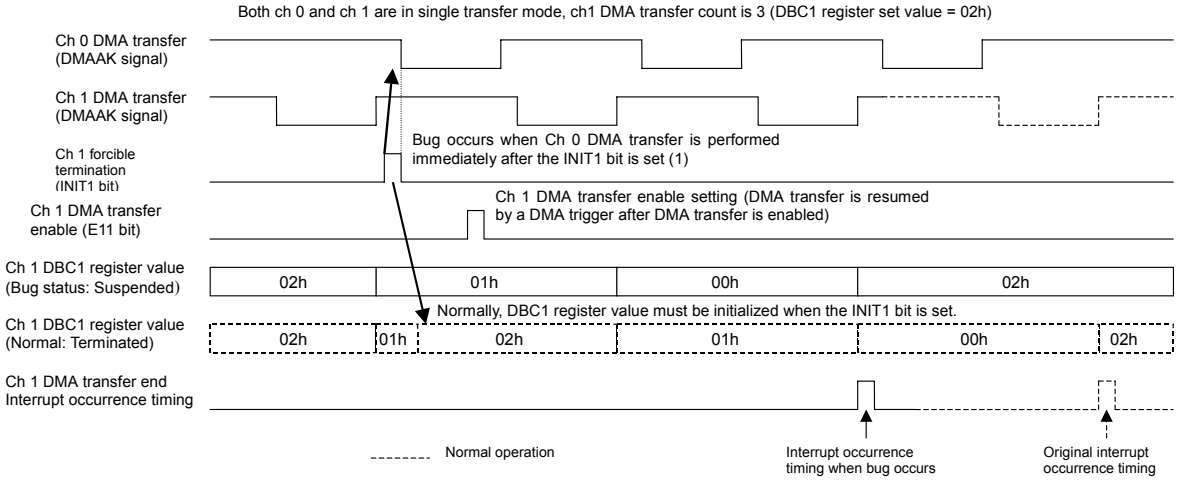
Please regard this as a permanent restriction.

a-46. 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.

a-47. Bug that DMA transfer is forcibly suspended by NMI

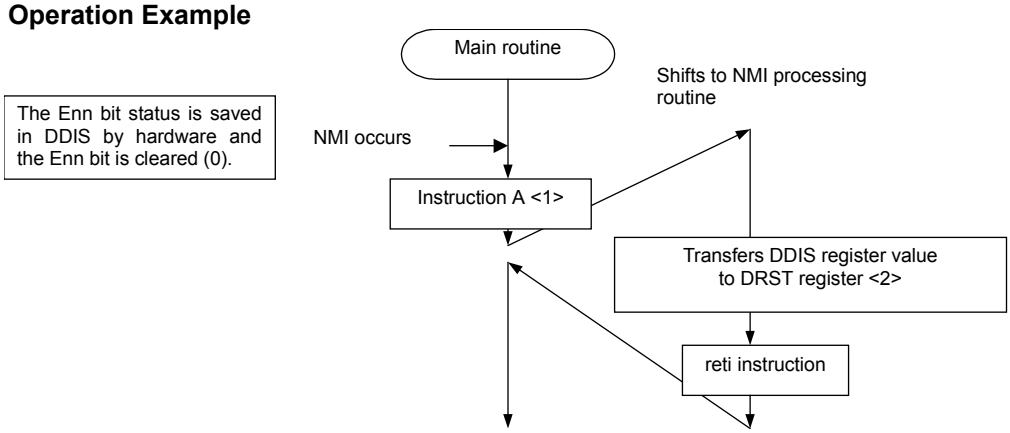
[Description]

DMA transfer is forcibly suspended by an NMI input during a DMA transfer. At this time, the DMA controller saves the status of the Enn bit (bit 0 of the DCHCn register) of all the DMA channels to the DDIS register, clears (0) the Enn bit, and disables the DMA transfer. In addition, when an NMI interrupt is acknowledged in the CPU pipeline operation, an instruction that has already been fetched is executed (1 instruction max.).

If this instruction is the one used to manipulate the Enn bit, the contents of the DDIS register are transferred to the DRST register in the NMI processing routine in order to resume the suspended DMA transfer. Therefore, the Enn bit setting immediately after the NMI input is not reflected.

As a result, the DMA transfer suspended by the NMI input cannot be restored as expected regardless of whether DMA transfer has been disabled or enabled.

Operation Example



When the Enn bit of the DCHCn register is set to disable or enable DMA using instruction A, the disable/enable status of each DMA channel is inconsistent in <1> and <2>. As a result, the normally suspended DMA transfer cannot be restored correctly.

Furthermore, when the Enn bit is set (1) using instruction A while the software trigger bit (STGn) of the DCHCn register is set (1), DMA transfer is performed even in the NMI processing routine.

There is no problem if an NMI is not used.

In addition, there is no problem if the system does not assume to resume the DMA transfer forcibly suspended by the NMI.

[Workaround]

Initialize the DMA transfer forcibly suspended by the NMI and then execute it again.

Please regard this as a permanent restriction.

a-48. 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.)

This bug does not occur if no instruction is executed in the internal RAM, or no DMA transfer is performed on the internal RAM.

[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.

a-49. Bug related to DMA transfer whose transfer count is set to two (1)

[Description]

A DMA transfer (not a resumed transfer but a new transfer) may be performed three times even though the transfer count is set to 2 (DBCn register value = 0001H).

In addition, when a DMA transfer channel that has been suspended by clearing the Enn bit of the DCHCn register is resumed by re-setting the Enn bit, a third DMA transfer may be performed only if two transfers remain (DBCn register value = 0001H).

In both cases, a DMA transfer (the third DMA transfer when the transfer count is set to 2) may be performed on the address one count ahead of the one at which the transfer should be terminated.

The setting or operation status of other DMA channels does not affect the occurrence of this bug.

This bug may occur when the following two conditions are satisfied at the same time.

- The transfer count is set to 2
- A flyby-type transfer, or a 2-cycle transfer whose transfer source is the internal RAM is used

The VSWC register set value (the number of waits for the internal peripheral register) is included in the bug occurrence condition. See the condition described on the following pages for details.

If the set value in the VSWC register is 12H, and the transfer count at which the transfer is suspended [Note 1] is 2, this bug may occur when an attempt is made to resume DMA transfer by setting (1) the Enn bit of the DCHCn register.

However, this bug does not occur if a new DMA transfer[Note 2] is enabled after the previous transfer is suspended with the count set to 2.

Notes 1. The suspended state means that the Enn bit of the DCHCn register or the ENn bit of the DRST register is cleared (0) for a DMA being processed.

2. The new DMA transfer must satisfy either of the following conditions.

- It is the first DMA transfer after a reset is released.
- The DMA transfer (the same DMA channel as the one to be activated) executed immediately before the channel in which the DMA is to be activated was terminated normally by a terminal count output.
- The DMA transfer (the same DMA channel as the one to be activated) executed immediately before the channel in which the DMA is to be activated was forcibly

terminated correctly by setting the INITn bit.

The “resumed DMA transfer” means a DMA transfer other than the above “new DMA transfer”. The “activation of a channel resumed after suspension other than the one targeted by INIT” is an example of this “resumed DMA transfer” in [Procedure to avoid bug] in **a-46. Bug related to DMA transfer forcible termination**.

[Workaround]

If this bug is applicable, implement the following workaround (1) to resume the suspended DMA. To activating a new DMA transfer, implement the following workaround (2).

(1) Resuming a suspended DMA transfer (resumed DMA transfer)

When resuming a DMA transfer that has been suspended by clearing (0) the Enn bit of the DCHCn register, set the ENn bit, not the Enn bit, of the DRST register.

Remark Be sure to disable DMA transfer in all the channels before enabling DMA using the DRST register; otherwise the following procedure must be implemented.

<1> Disable interrupts (DI state).

<2> Read the DRST register and transfer the ENn bit of each channel to a general-purpose register.

<3> Write 00H to the DRST register twice[Note].

<4> Among the bits stored in <2>, set (1) the relevant bit of the channel for which DMA transfer should be enabled.

<5> Write the value in <4> to the DRST register.

<6> Enable interrupts (EI state).

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

(2) Activating a DMA transfer by newly setting the transfer count to 2 (new DMA transfer)

Implement the following workaround when this bug is applicable due to the VSWC register set value.

- Implement the following workaround when two or more channels of DMA are activated simultaneously.
  - Perform a DMA transfer in which the transfer count is set to 1 (DBCn register set value = 0000H) twice.
- Implement the following workaround when only one channel is used.
  - Enable the DMA transfer by setting (1) the ENn bit of the DRST register, instead of setting the Enn bit of the DCHCn register.

Remark The above workaround can also be used when the DMA transfer count is variable, and if the value is set to 2 as a result of reading the DBCn register value.

Please regard this as a permanent restriction.

Bug related to DMA transfer when transfer count is set to 2 [Bug occurrence conditions]

This bug occurs only when the transfer count is set to 2. If the transfer count is 2, check the following occurrence conditions to confirm if this bug is applicable or not.

Check the internal signal (strobe signal width for peripheral I/O) according to the setting value of the system wait control register (VSWC).

VSWC Setting Value	Internal Signal (Strobe Signal Width for Peripheral I/O)	Remark
11H	2 clk	4 MHz ≤ fxx < 33 MHz
12H	3 clk	33 MHz ≤ fxx ≤ 50 MHz
13H	4 clk	
14H	5 clk	
77H	8 clk	Initial value

fxx: Operating frequency

Check the applicability of this bug in the following table using the internal signal (strobe signal width for peripheral I/O) obtained above.

Transfer Mode	Transfer Request	New/Resumed	Flyby Transfer External Memory ↔ External I/O	2-Cycle Transfer Internal RAM → Internal Peripheral Internal RAM → External I/O Internal RAM → External Memory
Single transfer	DMARQn internal peripheral I/O	New DMA transfer	Applicable with 5 clk or more	Applicable with 8 clk or more
		Resumed DMA transfer	Applicable with 3 clk or more	Applicable with 6 clk or more
	Software trigger	New DMA transfer	Not applicable	Not applicable
		Resumed DMA transfer	Not applicable	Not applicable
Single-step transfer	DMARQn internal peripheral I/O	New DMA transfer	Applicable with 5 clk or more	Applicable with 8 clk or more
		Resumed DMA transfer	Applicable with 3 clk or more	Applicable with 6 clk or more
	Software trigger	New DMA transfer	Applicable with 5 clk	Applicable with 8 clk
		Resumed DMA transfer	Applicable with 3 clk	Applicable with 6 clk
Block transfer	DMARQn internal peripheral I/O	New DMA transfer	Applicable with 5 clk or more	Applicable with 8 clk or more
		Resumed DMA transfer	–	–
	Software trigger	New DMA transfer	Applicable with 5 clk	Applicable with 8 clk
		Resumed DMA transfer	–	–

The 2-cycle transfers in the following cases are not applicable.

Transfer Source	Transfer Destination
Internal peripheral I/O	Internal peripheral I/O
Internal peripheral I/O	External I/O
Internal peripheral I/O	Internal RAM
Internal peripheral I/O	External memory

Transfer Source	Transfer Destination
External I/O	Internal peripheral I/O
External I/O	External I/O
External I/O	Internal RAM
External I/O	External memory

Transfer Source	Transfer Destination
External memory	Internal peripheral I/O
External memory	External I/O
External memory	Internal RAM
External memory	External memory

The following is an example when the set value of the VSWC register = 12H.

Transfer Mode	Transfer Request	New/Resumed	Flyby Transfer External Memory ↔ External I/O	2-Cycle Transfer Internal RAM → Internal Peripheral Internal RAM → External I/O Internal RAM → External Memory
Single transfer	DMARQn internal peripheral I/O	New DMA transfer	√	√
		Resumed DMA transfer	×	√
	Software trigger	New DMA transfer	√	√
		Resumed DMA transfer	√	√
Single-step transfer	DMARQn internal peripheral I/O	New DMA transfer	√	√
		Resumed DMA transfer	×	√
	Software trigger	New DMA transfer	√	√
		Resumed DMA transfer	×	√
Block transfer	DMARQn internal peripheral I/O	New DMA transfer	√	√
		Resumed DMA transfer	–	–
	Software trigger	New DMA transfer	√	√
		Resumed DMA transfer	–	–

√: Not applicable, ×: Applicable

a-50. Bug related to DMA transfer whose transfer count is set to two (2)

[Description]

With a DMA channel whose remaining transfer count (including the initial setting) is two (DBCn register value = 0001H), if the Enn bit of the DCHCn register is set (1) immediately before a DMA transfer cycle by flyby occurs, or if the Enn bit of the DCHCn register is cleared (0) immediately before the first flyby DMA transfer cycle is activated in block transfer mode in which the total transfer count is set to two, DMA transfer may be performed three times in total until the subsequent series of DMA transfers is complete. (n = 0 to 3)

In the bug operation, the transfer is performed at the address one cycle ahead of the address at which the transfer should have been complete (DMA transfer is performed three times whereas it is only set to two). The settings of the other DMA channels and their operating statuses do not affect the bug occurrence conditions.

Under the above bug occurrence conditions, setting the Enn bit of the DCHCn register (1) means rewriting the Enn bit that has already been set (unnecessary operation). In the same manner, a clear operation (0) means that an enabled DMA transfer in a block transfer mode in which the

total transfer count is set to two is disabled before it is activated (unnecessary operation). In addition, this bug does not affect manipulation of the Enn bit of the DRST register described in [Workaround] in **a-46. Bug related to DMA transfer forcible termination**.

[Workaround]

When the Enn bit of the DCHCn register is set (1), do not set or clear the Enn bit until the transfer count of the DMA transfer set in the DBCn register is complete, or until DMA transfer is forcibly terminated using the INITn bit of the DCHCn register.

Please regard this as a permanent restriction.

a-51. Bug that TCn bit of DMA is not cleared automatically

[Description]

The TCn bit of the DCHCn register should automatically be cleared when it is read. When two or more channels of DMA transfer to the internal RAM (transfer source or transfer destination) are used simultaneously, however, the TCn bit may not be cleared even if the DMA transfer is complete (n = 0 to 3)

This bug does not occur if any of the following conditions is satisfied.

- Only one channel is used for the DMA transfer.
- The internal RAM (transfer source or transfer destination) is not the target of the DMA transfer.

This bug occurs only when setting the TC bit (1) is held pending while the DCHCn register is being polled by the CPU program.

If all the following four conditions are satisfied, however, this bug may occur when the TC bit of the DCHCn register is read in the interrupt routine triggered by the DMA transfer end interrupt (INTDMAn).

(1) Multiple channels are used for the DMA transfer.
(2) The DMA transfer is performed from the internal RAM.
(3) The VSWC register setting value is 11H (operating frequency: 4 to less than 33 MHz).
(4) A load/store instruction is not performed on internal RAM, internal I/O area, or external memory before the DCHCn register is read in the DMA transfer end interrupt routine (the DCHCn register is the first register to be accessed).

[Workaround]

When reading the TCn bit of the DCHCn register corresponding to the DMA transfer channel that targets the internal RAM, read the TC bit that has already been set (1) and then perform two dummy reads on the DCHCn register in succession. These three successive reads will properly clear the TCn bit (0).

Please regard this as a permanent restriction.

b-1. Restriction on operating frequency

[Description]

The maximum operating frequency is 40 MHz.

[Workaround]

There is no workaround. Use a frequency of 40 MHz or lower.

IE-V850E-MC is schedule to be modified in the next version.

This restriction has been corrected in the IE-V850E-MC-A with control code E or later.

b-2. Restriction on break timing when guarded area is fetched

[Description]

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

[Workaround]

There is no workaround. Please regard this as a permanent restriction.

b-3. Restriction on trace in case of mis-alignment (during read access only)

[Description]

If a mis-align access occurs (the bus cycle occurs more than once), access data is displayed in the last access size, making access data display abnormal.

This occurs in a read cycle.

A sample program and an example of trace data are shown below.

The following data is written to memory in advance from address 0x2000:

[Memory data]

	0	1	2	3	4	5	6	7	8 …
0x2000	0x11	0x22	0x33	0x44	0x55	0x66	0x77	0x88	0x99 …
0X2010	…	…	…	…	…	…	…	…	…

[Sample program]

```
Address     Data      DisAsm
00001000    20560020  movea 0x2000, r0, r10
00001004    2aa70100  ld.w 0x0[r10], r20     ; Reads 0x44332211 from address 0x2000
00001008    0000      nop
0000100A    20560120  movea 0x2001, r0, r10
0000100E    2aaf0100  ld.w 0x0[r10], r21     ; Reads 0x55443322 from address 0x2001
00001012    0000      nop
00001014    20560220  movea 0x2002, r0, r10
00001018    2ab70100  ld.w 0x0[r10], r22     ; Reads 0x66554433 from address 0x2002
0000101C    0000      nop
0000101E    20560320  movea 0x2003, r0, r10
00001022    2abf0100  ld.w 0x0[r10], r23     ; Reads 0x77665544 from address 0x2003
00001026    0000      nop
00001028    20560020  movea 0x2000, r0, r10
0000102C    2ac70000  ld.h 0x0[r10], r24     ; Reads 0x2211 from address 0x2000
00001030    0000      nop
00001032    20560120  movea 0x2001, r0, r10
00001036    2acf0000  ld.h 0x0[r10], r25     ; Reads 0x3322 from address 0x2001
0000103A    0000      nop
0000103C    20560220  movea 0x2002, r0, r10
00001040    2ad70000  ld.h 0x0[r10], r26     ; Reads 0x4433 from address 0x2002
00001044    0000      nop
00001046    20560320  movea 0x2003, r0, r10
0000104A    2adf0000  ld.h 0x0[r10], r27     ; Reads 0x5544 from address 0x2004
0000104E    0000      nop
00001050    0000      nop
00001052    0000      nop
```

[Trace data of sample program] ( ) of Abnormal ( ): Expected value

Fram	Time	Address	Data	Status	Address	Data	Status	ExtProbe	DisAsm
00031	1	00001000	20560020	BRM1				00	movea 0x2000, r0, r10
00032	1	00001004	2AA70100	BRM1	00002000	44332211	R	00	ld.w 0x0[r10], r20
00033	1	00001008	00002056	M1		Normal		00	nop
00034	1	0000100A	20560120	M1				00	movea 0x2001, r0, r10
00035	1	0000100E	2AAF0100	M1				00	ld.w 0x0[r10], r21
00036	8				00002001	22	R	00	
00037	1	00001012	00002056	M1	<1> Abnormal (55443322)			00	nop
00038	1	00001014	20560220	M1				00	movea 0x2002, r0, r10
00039	1	00001018	2AB70100	M1				00	ld.w 0x0[r10], r22
00040	5				00002002	4433	R	00	
00041	1	0000101C	00002056	M1	<2> Abnormal (66554433)			00	nop
00042	1	0000101E	20560320	M1				00	movea 0x2003, r0, r10
00043	1	00001022	2ABF0100	M1				00	ld.w 0x0[r10], r23
00044	8				00002003	44	R	00	
00045	1	00001026	00002056	M1	<3> Abnormal (77665544)			00	nop
00046	1	00001028	20560020	M1				00	movea 0x2000, r0, r10
00047	2	0000102C	2AC70000	M1	00002000	2211	R	00	ld.h 0x0[r10], r24
00048	1	00001030	00002056	M1	Normal			00	nop
00049	1	00001032	20560120	M1				00	movea 0x2001, r0, r10
00050	1	00001036	2ACF0000	M1				00	ld.h 0x0[r10], r25
00051	5				00002001	22	R	00	
00052	1	0000103A	00002056	M1	<4> Abnormal (3322)			00	nop
00053	1	0000103C	20560220	M1				00	movea 0x2002, r0, r10
00054	2	00001040	2AD70000	M1	00002002	4433	R	00	ld.h 0x0[r10], r26
00055	1	00001044	00002056	M1	Normal			00	nop
00056	1	00001046	20560320	M1				00	movea 0x2003, r0, r10
00057	1	0000104A	2ADF0000	M1				00	ld.h 0x0[r10], r27
00058	5				00002003	44	R	00	
00059	1	0000104E	00000000	M1	<5> Abnormal (5544)			00	nop
00060	1	00001050	00000000	M1				00	nop
00061	1	00001052	00004000	M1				00	nop

<1> If address of address bit 1, 0 = 0, 1 is accessed in word units, data is displayed as byte access data by mistake.

<2> If address of address bit 1, 0 = 1, 0 is accessed in word units, data is displayed as halfword access data by mistake.

<3> If address of address bit 1, 0 = 1, 1 is accessed in word units, data is displayed as byte access data by mistake.

<4> If address of address bit 1, 0 = 0, 1 is accessed in halfword units, data is displayed as byte access data by mistake.

<5> If address of address bit 1, 0 = 1, 1 is accessed in halfword units, data is displayed as byte access data by mistake.

[Workaround]

There is no workaround.

Exercise care when referencing trace data if mis-align read access has been executed.

This bug has been corrected in the IE-V850E-MC with control code B.

This bug has been corrected in the IE-V850E-MC-A with control code D.

b-4. Restrictions on trace data on execution of HALT or STOP instruction

[Description]

a) Even if an instruction that accesses the PRCMD or PSC register is fetched to change the mode into the software STOP mode, the fetch information is not traced (access data to the register is traced).

After the PSC register has been accessed, execution of the last of several NOP instructions is not traced.

Example

```
mov 0x2,r2
movea base_address,r0,r20   ; base_address = FFFF0000H
st.h r11,PRCMD[R20]         ;PRCMD = 01FCH  <-------  Fetching this instruction is not traced.
st.h r11,PSC[R20]           ;PSC = 01FEH    <-------  Fetching this instruction is not traced.
nop
nop
nop
nop
nop
```

b) If the HALT instruction is executed, the fetch information of the HALT instruction is not traced, and one frame of excess trace data remains.

[Sample program]

```
00001000    0000        nop
00001002    0000        nop
00001004    0000        nop
00001006    0000        nop
00001008    e0072001    halt
0000100C    0000        nop
0000100E    0000        nop
00001010    0000        nop
00001012    0000        nop
00001014    0000        nop
```

[Trace data of sample program]

Fram	Time	Address	Data	Status	Address	Data	Status	ExtProbe	DisAsm
00001	1	00001000	00000000	BRM1			00		nop
00002	1	00001002	00000000	M1			00		nop
00003	1	00001004	00000000	M1			00		nop
00004	1	00001006	0000E007	M1			00		nop
00005	65535						00		Halt of 1008H should have been traced.
00006	2						00		← Excess frame
00007	3	00000010	00000000	M1			00		nop
00008	1	00000012	00000000	M1			00		nop
00009	1	00000014	00000000	M1			00		nop

[Workaround]

When referencing trace data, read the execution result from the fetch addresses of the instruction in question and the instructions before and after that instruction.

These restrictions have been corrected in the IE-V850E-MC with control code B.

These restrictions have been corrected in the IE-V850E-MC-A with control code D.

b-5. Bit manipulation instruction (set1, clr1, not1, tst1) access data is illegally traced by tracer.

[Description]

If bits of an address where the lower 2 bits are 01, 10, or 11, the access data of the trace data is traced as an illegal value (the instruction is normally executed).

[Sample program]

```
00001000    20560020    movea 0x2000, r0, r10
00001004    20a60500    movea 0x5, r0, r20
00001008    20ae0a00    movea 0xa, r0, r21
0000100C    4aaf0000    st.b r21, 0x0[r10]    ; Writes 0xa to address 0x2000
00001010    4aa70100    st.b r20, 0x1[r10]    ; Writes 0x5 to address 0x2000
00001014    4aaf0200    st.b r21, 0x2[r10]    ; Writes 0xa to address 0x2000
00001018    4aa70300    st.b r20, 0x3[r10]    ; Writes 0x5 to address 0x2000
0000101C    0000        nop
0000101E    ca070000    set1 0x0, 0x0[r10]    ; Sets bit 0 of address 0x2000
00001022    0000        nop
00001024    ca0f0100    set1 0x1, 0x1[r10]    ; Sets bit 1 of address 0x2001
00001028    0000        nop
0000102A    ca170200    set1 0x2, 0x2[r10]    ; Sets bit 2 of address 0x2002
0000102E    0000        nop
00001030    ca1f0300    set1 0x3, 0x3[r10]    ; Sets bit 3 of address 0x2003
00001034    0000        nop
```

[Trace data of sample program]

Fram	Time	Address	Data	Status	Address	Data	Status	ExtProbe	DisAsm
00016	17	00001004	20A60500	M1				00	movea 0x5, r0, r20
00017	17	00001008	20AE0A00	M1				00	movea 0xa, r0, r21
00018	17	0000100C	4AAF0000	M1	00002000	0A	W	00	st.b r21, 0x0[r10]
00019	1	00001010	4AA70100	M1	00002001	05	W	00	st.b r20, 0x1[r10]
00020	1	00001014	4AAF0200	M1				00	st.b r21, 0x2[r10]
00021	1	00001000	20560020	BRM1				00	movea 0x2000, r0, r10
00022	1	00001004	20A60500	BRM1				00	movea 0x5, r0, r20
00023	5	00001008	20AE0A00	M1				00	movea 0xa, r0, r21
00024	5	0000100C	4AAF0000	M1	00002000	0A	W	00	st.b r21, 0x0[r10]
00025	1	00001010	4AA70100	M1	00002001	05	W	00	st.b r20, 0x1[r10]
00026	1	00001014	4AAF0200	M1				00	st.b r21, 0x2[r10]
00027	16				00002002	0A	W	00	
00028	1	00001018	4AA70300	M1	00002003	05	W	00	st.b r20, 0x3[r10]
00029	1	0000101C	0000CA07	M1				00	nop
00030	1	0000101E	CA070000	M1				00	set1 0x0, 0x0[r10]; Trace with this instruction is normal.
00031	12				00002000	0A	R	00	
00032	1				00002000	0B	W	00	
00033	1	00001022	0000CA0F	M1				00	nop
00034	9	00001024	CA0F0100	M1	00002001	0A	R	00	set1 0x1, 0x1[r10]; Trace with this instruction is abnormal.
00035	6				00002001	0A	W	00	
00036	1	00001028	0000CA17	M1				00	nop
00037	10	0000102A	CA170200	M1	00002002	0A	R	00	set1 0x2, 0x2[r10]; Trace with this instruction is abnormal.
00038	1				00002002	0A	W	00	
00039	1	0000102E	0000CA1F	M1				00	nop
00040	9	00001030	CA1F0300	M1	00002003	0A	R	00	set1 0x3, 0x3[r10]; Trace with this instruction is abnormal.
00041	1				00002003	0A	W	00	

[Workaround]

There is no workaround.

Exercise care in referencing trace data when executing an instruction that manipulates the bits of the address in question.

This bug has been corrected in the IE-V850E-MC with control code B.

This bug has been corrected in the IE-V850E-MC-A with control code D.

b-6. Events including data conditions by access of bit manipulation instruction cannot be detected.

[Description]

If a bit manipulation instruction is executed on an address where the lower 2 bits are 01, 10, or 11, and access to this address is specified as an event condition, an event cannot be detected if a data condition is included.

[Workaround]

Specify an event condition by using an address, status, and bus size, and do not include data as a condition.

This bug has been corrected in the IE-V850E-MC with control code B.

This bug has been corrected in the IE-V850E-MC-A with control code D.

b-7. Restriction on HOLD status

[Description]

The “HOLD” status that should be displayed by the debugger when the HLDAK signal is valid (during external bus hold) is displayed when the HLDRQ signal is valid.

[Workaround]

Take “HOLD” status display as meaning that the HLDRQ signal is valid, instead of meaning that the HLDAK signal is valid (during external bus hold).

If the HLDRQ signal becomes valid even when the HLDRQ signal is masked, “HOLD” status is displayed.

This restriction has been corrected in the IE-V850E-MC with control code B.

This restriction has been corrected in the IE-V850E-MC-A with control code D.

b-8. 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. Please regard this as a permanent restriction.

b-9. Restriction on SFR illegal break

[Description]

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

[Workaround]

There is no workaround. Please regard this as a permanent restriction.

b-10. Restrictions on programmable I/O space

[Description]

a) If a programmable I/O space is mapped to the higher 32 MB area in the 64 MB mode, the programmable I/O space cannot be accessed during break.

b) If the programmable I/O space is accessed during program execution, an SFR illegal break occurs.

[Workaround]

a) These restrictions have been corrected in the following debuggers.

NEC Electronics debugger: ID850 V2.40 or later

GHS Multi: EX85032.DLL V5.60 or later

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

These restrictions have been corrected in the IE-V850E-MC with control code D.

These restrictions have been corrected in the IE-V850E-MC-A with control code F.

b-11. 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)
11  a += c;   (add r7,r10)
```

[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

1. If dst of move 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
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

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

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
0x1008    sld.b   0x8[ep], r11
```

```
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 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
0x1008 bn 0xf0
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
0x100e sld.b 0x10[ep], r13
```

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]

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

  NEC Electronics debugger: ID850 E2.20f or later versions

  GHS Multi: Use EX85032.DLL version 5.40 or later

- There is no workaround for hardware breaks. Please regard this as a permanent restriction.

b-12. 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. Please regard this as a permanent restriction.

b-13. Restriction on DBPC and DBPSW access during a break

[Description]

Although DBPC and DBPSW can be read during a break, they cannot be written to.

[Workaround]

There is no workaround. Please regard this as a permanent restriction.

b-14. 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. Please regard this as a permanent restriction.

b-15. Restriction on illegal guard break when IRAM size is 28 KB

[Description]

A guard break occurs in the IRAM area when a fetch is executed on the area 0x3ff8000 to 0x3ffc000 with an IRAM size of 28 KB.

[Workaround]

1) Set the size of IRAM to 60 KB.

2) Map the 1 MB region where IRAM area is mapped to the emulation memory or target memory.

This restriction has been corrected in the IE-V850E-MC with control code D.

This restriction has been corrected in the IE-V850E-MC-A with control code F.

b-16. Restriction on illegal trace when big endian is used

[Description]

When a bit manipulation instruction is executed on data in the external memory in big-endian mode, the access data is not traced correctly.

[Workaround]

There is no workaround.

This restriction has been corrected in the IE-V850E-MC with control code D.

This restriction has been corrected in the IE-V850E-MC-A with control code F.

b-17. 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. Please regard this as a permanent restriction.

b-18. Restriction on SFR read access during break

[Description]

The SFR bits that are normally cleared by a read access (e.g. TC bit of the DCHC register) are also cleared when displayed using SFR 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 SFR is not displayed by debugger.

In addition, this restriction can be avoided by using the device file and the following debuggers in combination.

- Device file

  DF703107: V1.20 or later

  DF703116: V1.21 or later

  DF703114: V1.00 or later

  DF703166: V1.00 or later

- Debugger

  NEC Electronics debugger: ID850 E2.20f or later

  GHS debugger (Multi) (Windows version): SV-V850-WIN32 Rel.4.0.5 or later
  Specify the -X2 option on starting

  Debugger by GHS (Multi) (Solaris version): SV-V850-Solaris Rel.4.0.5 or later
  Specify the -X2 option on starting

b-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.
```

Please regard this as a permanent restriction.

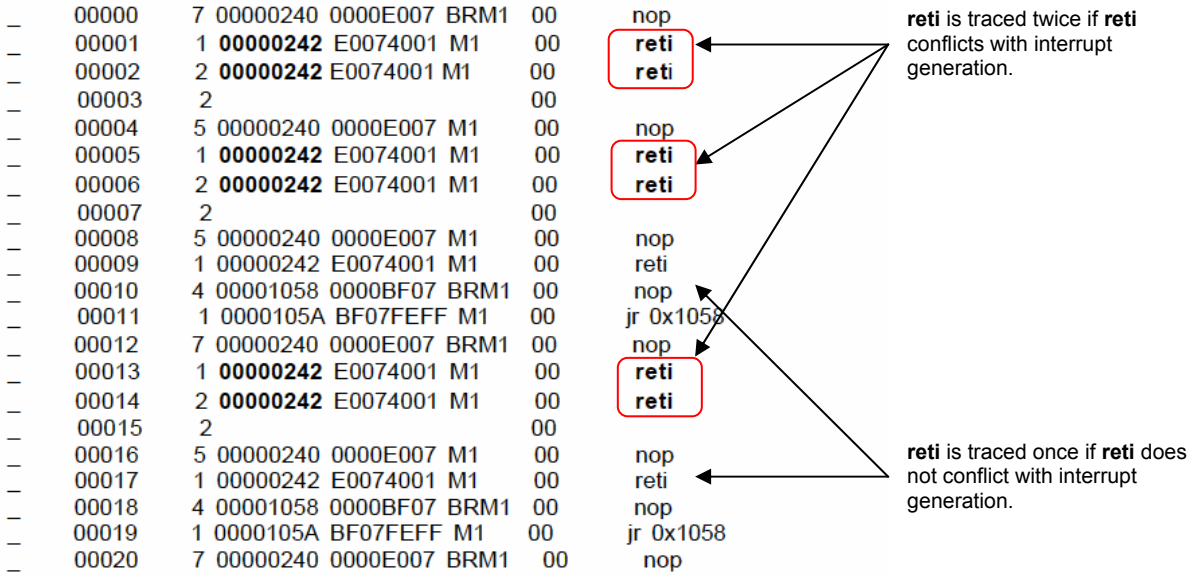
b-20. Restriction that the same branch instruction is traced twice

[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. Please regard this as a permanent restriction.

b-21. 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]

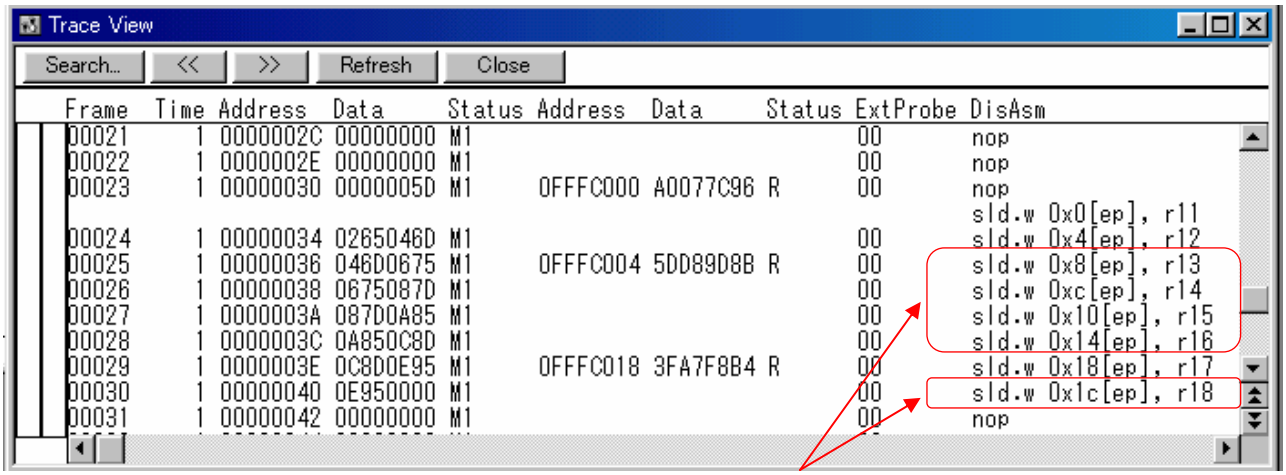
There is no workaround. Please regard this as a permanent restriction.

b-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. Please regard this as a permanent restriction.

C-1. Modification through quality improvement

[Description]

IE-V850E-MC-A control code C is being remodified for quality improvements, but this is the same as control code B in terms of actual operation.

4. Cautions

1) Notes on using emulation memory (emulation memory is mounted on option board)

- The number of waits to be inserted varies depending on the operating frequency of the emulator.

  4 MHz ≤ Operating frequency < 25 MHz: 0 waits

  25 MHz ≤ Operating frequency ≤ 40 MHz: 1 wait

  40 MHz < Operating frequency: 2 waits

- Set the bus width to 32 or 16 bits. The 8-bit bus cannot be used.
- The emulation memory cannot be mapped to addresses higher than 4000000H in the 64 MB mode.
- 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.

With ID850

The following three alternatives are available on the configuration screen.

		Emulation Memory Access	External Memory Access
WAIT MASK	Data wait	Fixed to 0 waits	Depends on the DWC register settings. WAIT signal is masked.
	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 DWC 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.
TARGET WAIT	Data wait	Depends on the DWC register setting. 1 wait when set to 0 waits.	Depends on the DWC 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.

With Multi

The wait pin can be masked or unmasked by the “pinmask” command.

		Emulation Memory Access	External Memory Access
WAIT: Mask EMWAIT: Mask	Data wait	Fixed to 0 waits	Depends on the DWC register settings. WAIT signal is masked.
	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 DWC 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 DWC register setting. 1 wait when set to 0 waits.	Depends on the DWC 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.

When setting the number of waits for the emulation memory to the same number as for the actual external memory for the purpose of performance testing, select “Target Wait” using the debugger.
At this time, if address waits are inserted in the targeted external memory, also add the number of waits inserted in the external memory to the emulation memory.

2) Pin handling

- This ICE uses the minimum pin handling, giving priority to compatibility with the device. Implement adequate workarounds for static electricity if the ICE is used without the target connected.
- Inside the ICE, pin handling is performed by the option board. For details, refer to the User’s Manual of the option board.

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
nop
nop
```

Insert five NOPs.

b) HALT instruction

```
00001008    e0072001    halt
0000100C    0000        nop
0000100E    0000        nop
00001010    0000        nop
00001012    0000        nop
00001014    0000        nop
```

Insert five NOPs.

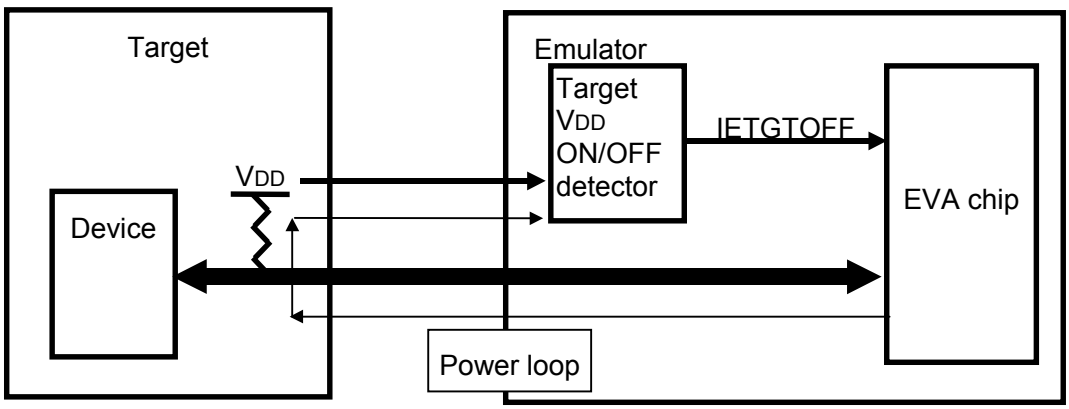
4) Pin control with target power off

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 circuit, and the emulator is automatically reset when target power is turned on or 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 circuit 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.



5) Notes on trace data

5)-1. 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)-2. 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.

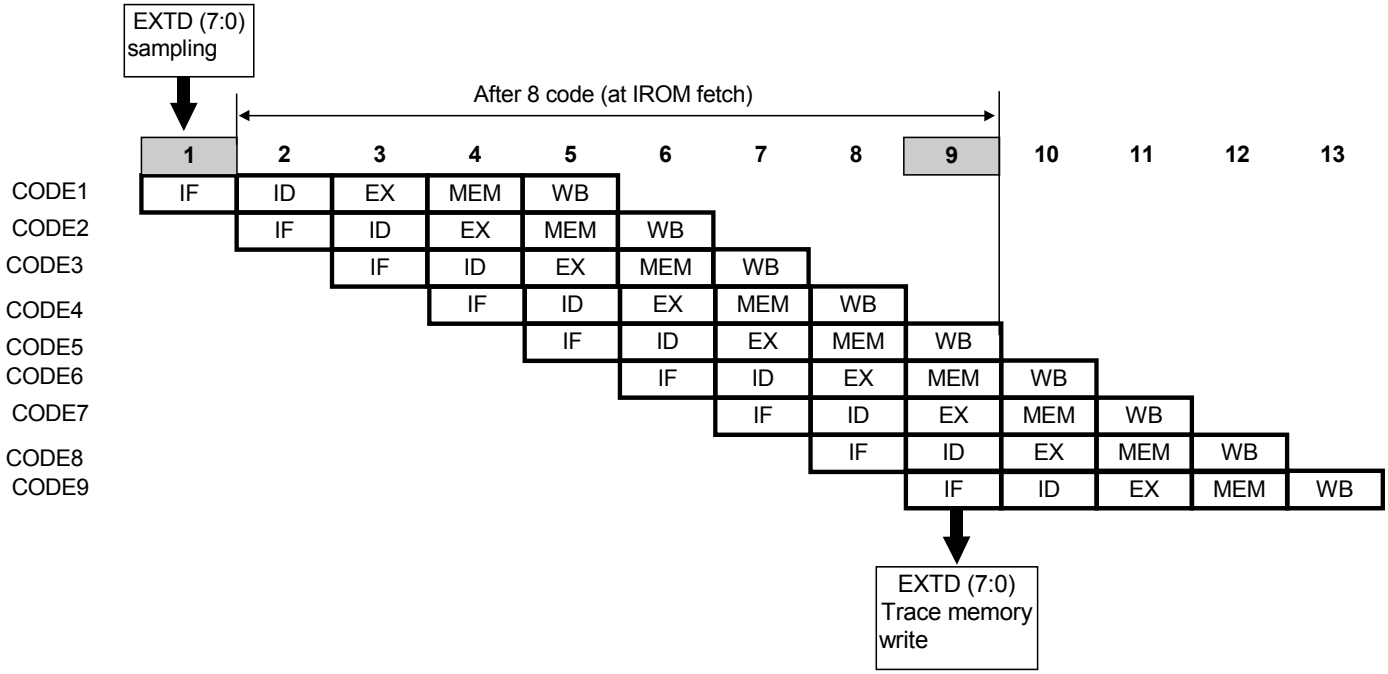
- Details -

The external logic data is sampled in synchronization with instruction execution and differs depending on whether a program is stored in IROM or external memory, and on the number of wait cycles.

When a program is placed in the external memory, it also differs depending on whether a read/write cycle is inserted in the external memory. The shortest time is 8 clocks when a program is placed in IROM.

The point to be noted is that the external logic data is not sampled every clock. Because it is not sampled unless instruction execution is performed, a signal that changes after execution of one instruction and before execution of another cannot be detected.

If there is a possibility that an instruction is executed every clock as is the case with IROM, therefore, the chance of missing the external logic data decreases. Conversely, if many wait cycles are inserted in the external memory, the chance of missing the external logic data increases.



6) Notes on Instruction Cache

The following notes of caution apply when using instruction cache:

(a) Emulator control code usable with instruction cache:

- IE-V850E-MC: Control code C or later
- IE-V850E-MC-A: Control code E or later

(b) Memory data rewriting during suspension of target program:

Even in the event of rewriting data in a space (space on which fetching is performed by instruction cache) locked due to autofiling while the target program is suspended, instruction cache data cannot be renewed. Where renewing instruction cache data, perform autofiling upon tag clearance.

(c) Setting software break:

Software breaks may not be specified for any instruction in a space (space on which fetching is performed by instruction cache) locked due to autofiling while the target program is suspended. In order to specify a software break, perform autofiling upon tag clearance, or use a hardware break.

7) Caution on fail-safe break in internal ROM/RAM space

The internal ROM/RAM 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

Internal RAM		
Debugger Setting	Internal RAM Space To Be Mapped	
	64 MB Mode	256 MB Mode
4096 bytes	3FFE000H to 3FFEFFFH	FFFE000H to FFFEFFFH
12288 bytes	3FFC000H to 3FFEFFFH	FFFC000H to FFFEFFFH
28672 bytes	3FF8000H to 3FFEFFFH	FFF8000H to FFFEFFFH
61440 bytes	3FF0000H to 3FFEFFFH	FFF0000H to FFFEFFFH
Other	Depends on the target device	

* Remark on internal ROM

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.

* Remark on internal RAM

A fail-safe break occurs when an access (instruction fetch or data access) to an address at which no memory exists is performed in accordance with the mapping settings.

When other settings are made, however, a space exists in which a fail-safe break cannot occur.

If the internal RAM space of the target device is mapped from 3FFC000H to 3FFE7FF (10 KB space), for example, the emulator maps the internal RAM space from 3FFC000H to 3FFEFFFH (12 KB space). Therefore, no fail-safe break occurs in the higher 2 KB space of the address (3FFE800H to 3FFEFFFH). To generate a fail-safe break upon an access to such an address, implement a measure such as setting an event break.