Concerned Products:	**Customer Notification**	Date: May 28, 01
		NEC Electronics (Europe) GmbH LSI & Discrete Business Unit Technical Product Support
IE-V850E-MC ***IE-V850E-MC-A***	**Bug and Limitation Report**	Source Doc: SBG-T-1983 SUD-4503-7, SUD-4503-8-E IMR-3CC-00xxx
		Author: T. Hoeveken

Date of initial issue:	Jun 21, 00	Doc No.: TPS-HE-B-2754
Date of last update:	Dec 21, 00	

(A) PRODUCT VERSION

1. Product Code: IE-V850E-MC-A

Control Code[Note]	Evachip (3.3 V)	Usable Exec Version
B	D703191R DS2.0-3V	EX85032.DLL version 5.21 or higher
C	D703191R DS2.0-3V	EX85032.DLL version 5.21 or higher
E	D703191AR DS4.1-3V	EX85032.DLL version 5.41 or higher

2. Product Code: IE-V850E-MC

Control Code[Note]	Evachip (5 V)	Usable Exec Version
A	D703191R DS2.0-5V	EX85032.DLL version 5.21 or higher
B	D703191R DS3.0-5V	EX85032.DLL version 5.41 or higher
D	D703119R DS4.3-5V	EX85032.DLL version 5.41 or higher

Note The control code is the digit from the left od a 10-digit control code that starts from E, e.g. EA9040009L means control code A.

Caution In conjunction with the usable exec version a qualified device file (Dxxxxx.800) is additionally necessary for the corresponding device, which has to be emulated. Make sure that you use the appropriated version of the device file. Please refer to the documentation of the dedicated EM1-board.

(B) LIST OF RESTRICTIONS

Bug No.		Restrictions	Control code					
			IE-V850-MC			IE-V850E-MC-A		
			A	B	D	B	C	E
Restricions dependent on CPU functions	a-1	Interrupt aborts LD instruction immediately before JMP	✋	✓	✓	✋	✋	✓
	a-2	Restriction 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 in not set in control mode immediately after starting in ROM-less mode	✋	✓	✓	✋	✋	✓
	a-6	Restriction on port DH/DL	✋	✓	✓	✋	✋	✓
	a-7	$\overline{\text{HLDAK}}$ output illegal due to conflict of self-refresh cycle and $\overline{\text{HLDRQ}}$ in STOP mode	✋	✓	✓	✋	✋	✓
	a-8	Fetch/data access fails if hardware STOP is executed after CBR refresh of DRAM/ SDRAM	✋	✓	✓	✋	✋	✓
	a-9	Restriction on data cache	💣	💣	💣	💣	💣	💣
	a-10	PFCCM register cannot be read	✋	✓	✓	✋	✋	✓
	a-11	VSB bus and memory controller (NB85E500/501/502) cannot be used together	💣	💣	💣	💣	💣	💣
	a-12	Restrictions on VSB bus signal	✋	✓	✓	✋	✋	✓
	a-13	Restrictions on NPB bus signal	💣	💣	💣	💣	💣	💣
	a-14	Restrictions on memory controller (NB85E500/501/502) signal	✋	✓	✓	✋	✋	✓
	a-15	Restrictions on instruction cache (1)	✋	✓	✓	✋	✋	✓
	a-16	Restriction related to SDRAM access during bus hold	✋	✓	✓	✋	✋	✓
	a-17	Restriction of self-refresh cycle by SELFREF pin	✋	✓	✓	✋	✋	✓
	a-18	Restrictions related to flyby DMA transfer to EDO DRAM	✋	✓	✓	✋	✋	✓

	Bug No.	Restrictions	Control code					
			IE-V850-MC			IE-V850E-MC-A		
			A	B	D	B	C	E
Restricions dependent on CPU functions	a-19	Restrictions of EDO DRAM with idle state inserted	✋	✓	✓	✋	✋	✓
	a-20	Restrictions related to flyby DMA transfer	✋	✓	✓	✋	✋	✓
	a-21	Restrictions of pin status in single-step mode 1 an ROMLESS modes 0 and 1	✋	✓	✓	✋	✋	✓
	a-22	Incorrect write-back 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	✋	✓	✓	Not applicable		
	a-24	Restriction related to the output of the DMAAK signal	✋	✋	✓	✋	✋	✓
	a-25	Restriction related to starting DMA by built-in peripheral I/O interrupt	✋	✋	✓	✋	✋	✓
	a-26	Restriction related to EDO DRAM bus collision	✋	✋		✋	✋	✓
	a-27	Restriction on the 2-way associative function of the instruction cache	✋	✋	✓	✋	✋	✓
	a-28	Forced stop of external DMA transfer in DMA line transfer mode	✋	✋	✓	✋	✋	✓
	a-29	Restriction on reading the DCHC register when DMA 2-cycle transfer is completed	✋	✋	✓	✋	✋	✓
	a-30	Restriction related to conflict between SDRAM initialization and SELFREF input	✋	✋	✓	✋	✋	✓
	a-31	Restriction on half-word writing to BSC, BCC, DWC0 and DWC1 registers	✋	✋	✓	✋	✋	✓
	a-32	Restriction related to SDRAM write operation	✋	✋	✓	✋	✋	✓
	a-33	Restriction on DRAM fetch immediately after block DMA transfer from DRAM to internal RAM	✋	✋	✓	✋	✋	✓
	a-34	Restriction on instruction cache (2)	✋	✋	✋	✋	✋	✋
	a-35	Restriction on SLD instruction	✋	✋	✓	✋	✋	✋

	Bug No.	Restrictions	Control code					
			IE-V850-MC			IE-V850E-MC-A		
			A	B	D	B	C	E
Restricions dependent on CPU functions	a-36	Unusable I/O when VSB bus is used	✋	✋	✋	✋	✋	✋
	a-37	Restriction on instruction cache (3)	✋	✋	✋	✋	✋	✋
	a-38	Restriction of $\overline{\text{DMAAK}}$ signal during DMA line transfer	💣	💣	💣	💣	💣	💣
	a-39	Restriction caused by interrupt input during bit manipulation instruction	✋	✋	✓	✋	✋	✋
	a-40	Restriction on hardware stop during bit manipulation instruction	✋	✋	✓	✋	✋	✋
	a-41	Restriction related to interruption of DMA transfer by external cause	✋	✋	✓	✋	✋	✋
Restricions on debug functions	b-1	Restriction on operating frequency	✋	✓	✓	✋	✋	✓
	b-2	Restrictions on break timing when guard area is fetched	💣	💣	💣	💣	💣	💣
	b-3	Restrictions 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	Restrictions on HOLD status	✋	✓	✓	✋	✋	✓
	b-8	ROM contents are rewritten if emulation ROM area is accessed for write	💣	💣	💣	💣	💣	💣
	b-9	Restriction of SFR illegal break	💣	💣	💣	💣	💣	💣
	b-10	Restriction on programmable I/O space	💣	💣	💣	💣	💣	💣
	b-11	Break does not occur even if breakpoint is set	Supported by debugger					
	b-12	Restriction related to access address during DMA trace	💣	💣	💣	💣	💣	💣
	b-13	Restriction on DBPC and DBPSW access during a break	💣	💣	💣	💣	💣	💣

	Bug No.	Restrictions	Control code					
			IE-V850-MC			IE-V850E-MC-A		
			A	B	D	B	C	E
Restricions on debug functions	b-14	Restriction on DBTRAP instruction	💣	💣	💣	💣	💣	💣
	b-15	Restriction on illegal guard break when IRAM size is 28 KB	✋	✋	✓	✋	✋	✋
	b-16	Restriction on illegal trace when big endian is used	✋	✋	✓	✋	✋	✋
	b-17	Restriction on access data	💣	💣	💣	💣	💣	💣
	b-18	Restriction on SFR access during break	Supported by debugger and device file					

✓: No problem, or corrected by version upgrade

✋: Bug (will be corrected by version upgrade)

💣: Bug (restriction, not corrected by version upgrade)

(C) DESCRIPTION OF RESTRICTIONS

1. Restrictions on CPU functions

No.	Outline	Description
a-1	Interrupt aborts LD instruction immediately before JMP	*Details* 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 before 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 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 the LD/SLD and the ID stage of JR/JARL/Bcc do not overlap, any instrucion other than JR/JARL/Bcc/LD/SLD) may be inserted. *To customers* A tool that insert 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 and third parties) as standard. Consult NEC's Development Tool Support Center for details.

No.	Outline	Description
a-2	Restrictions on IRAM read access after start of interrupt servicing	*Details* 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 NOP instructions. *Workaround* To access IRAM for read after interrupt servicing has been started, insert a dummy RD/WR (external memory) instruction before the instruction that reads the IRAM.
a-3	Fetching is abnormal immediately after writing to SCRn register	*Details* An abnormal fetch cycle occurs if the SCR register is written after the VSWC register is set to 11H and SDRAM is specified by the BCTn register. This is because, if an access to external memory occurs immediately after the SCR register write cycle, the values of A25 to A0 and CSZ7 to CSZ0 are not normal. *Workaround* Specify two or more wait cycles for VPSTB wait (set bits VSWL2 to VSWL0 of the VSWC register to 010B or more).
a-4	Single, line, or single-step transfer of 2-cycle DMA	*Details* 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 NB85E core).
a-5	Port C is not set on control mode immediately after starting in ROM-less mode	*Details* Port C, which should be in the control mode immediately after the ROM-less 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 ROM-less mode has been started, to access DRAM. Do not execute an application that accesses DRAM immediately after reset.
a-6	Restrictions on Port DH/DL	*Details* Port DH/DL enters the control mode (D31 to D0) immediately after it has been started, and 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 ICE has been started.

No.	Outline	Description
a-7	$\overline{\text{HLDAK}}$ output illegal due to conflict of self-refresh cycle and $\overline{\text{HLDRQ}}$ in STOP mode	*Details* If self refreshing of DRAM/SDRAM and $\overline{\text{HLDRQ}}$ conflict in the STOP mode, the $\overline{\text{HLDAK}}$ signal is illegally output even in the self-refresh cycle. *Workaround* Do not input $\overline{\text{HLDRQ}}$ in the STOP mode.
a-8	Fetch/data access fails if hardware STOP is executed after CBR refresh of DRAM/ SDRAM	*Details* 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/SDRAM, fetch/data access cannot be correctly executed because self refreshing causes 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.
a-9	Restriction on data cache	*Details* This emulator has no plan to support data cache. *Workaround* None.
a-10	PFCCM register cannot be read	*Details* Although the PFCCM register is a R/W register, it cannot be read. *Workaround* Use the PFCCM register as a write-only register.
a-11	VSB bus and memory controller (NB85E500/ 501/502) cannot be used together	*Details* 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* None. Please regard this as a permanent restriction.

No.	Outline	Description
a-12	Restriction on VSB bus signal	*Details* a) The master device on the VSB bus cannot access the slave source on the NPB bus. b) The emulator does not output VBA27, VBA26. c) The emulator does not output VBSEQ2 to VBSEQ0. *Workaround* a) There are no preventive measures. This restriction is planned to be corrected in the next version. b) Temporary measures 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 are no preventive measures.
a-13	Restrictions on NPB bus signal	*Details* The emulator does not have a VPDACT signal pin and this signal is always fixed to the active level inside the emulator. *Workaround* None.

<table>
<tr><th>No.</th><th>Outline</th><th>Description</th></tr>
<tr><td>a-14</td><td>Restrictions on memory controller (NB85E500/ 501/502) signal</td><td>
<u>Details</u>
a) The emulator does not output the ASTBZ, DSTBZ, MPXCZ, RDCYZ, and BUSSTZ signals; therefor the multiplexed bus cannot be used.

b) The emulator does not output the BENZ3 to BENZ0 and DC3 to DC0 signals.

c) The IORDZ and CSZ2, and IOWRZ and CSZ5 signals of the emulator are multiplexed incorrectly. The revision will take place by next version as follows:
<table>
<tr><th>Current Version</th><th>Revised Version</th></tr>
<tr><td>IORDZ and CSZ2 are multiplexed.</td><td>IORDZ and CSZ5 are multiplexed.</td></tr>
<tr><td>IOWRZ and CSZ5 are multiplexed.</td><td>IOWRZ and CSZ2 are multiplexed.</td></tr>
</table>
d) The emulator does not have an MCE signal pin and this signal is always fixed to the active level inside the emulator. Therefor, the MEn bit (n= 0 to 7) of the BCTn register (n = 0 to 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 in this memory controller.

<u>Workaround</u>
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 BENZ0 and DC3 to DC0.
Please regard this as a permanent restriction.

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

d) There are no preventive measures.
Please regard this as a permanent restriction.

e) There are no preventive measures.
</td></tr>
<tr><td>a-15</td><td>Restrictions on instruction cache (1)</td><td>
<u>Details</u>
If control is transferred from a cacheable area to an uncacheable area and then back to the cacheable area again after auto fill, the first clock does not hit.

<u>Workaround</u>
None.
</td></tr>
</table>

No.	Outline	Description
a-16	Restrictions related to SDRAM access during bus hold	*Details* 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.
a-17	Restriction of self-refresh cycle by SELFREF pin	*Details* If the self-refresh cycle is started by the SELFREF pin, the REFRQ(-) signal is not asserted active. *Workaround* None.
a-18	Restriction related to flyby DMA transfer to EDO DRAM	*Details* 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 es 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.
a-19	Restrictions of EDO DRAM with idle state inserted	*Details* 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.

No.	Outline	Description
a-20	Restrictions related to flyby DMA transfer	*Details* 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.

<table>
<tr><th>No.</th><th>Outline</th><th>Description</th></tr>
<tr><td>a-21</td><td>Restrictions of pin status in single-step mode 1 and ROM-less modes 0 and 1</td><td>
Details
A few pins do not go into a high-impedance state, but rather output a specific level, when the device is reset in the single-chip mode 1 and ROM-less modes 0 and 1 (see below).
<table>
<tr><th>Operating Status / Pin</th><th>Current</th><th>Revised</th></tr>
<tr><td></td><td>Reset (Single-Chip Mode 1, ROM-less Mode 0,1)</td><td>Reset (Single-Chip Mode 1, ROM-less Mode 0,1)</td></tr>
<tr><td>A0 to A15 (PAL0 to PAL15)</td><td>L</td><td>Hi-Z</td></tr>
<tr><td>A16 to A25 (PAL0 to PAL15)</td><td>L</td><td>Hi-Z</td></tr>
<tr><td>D0 to D15 (PDL0 to PDL15)</td><td>L</td><td>Hi-Z</td></tr>
<tr><td>$\overline{CS0}$ to $\overline{CS7}$ (PCS0 to PCS7)</td><td>H</td><td>Hi-Z</td></tr>
<tr><td>$\overline{LWR}$, $\overline{UWR}$ (PCT0, PCT1)</td><td>H</td><td>Hi-Z</td></tr>
<tr><td>$\overline{RD}$ (PCT4)</td><td>H</td><td>Hi-Z</td></tr>
<tr><td>$\overline{WE}$ (PCT5)</td><td>H</td><td>Hi-Z</td></tr>
<tr><td>$\overline{OE}$ (PCT6)</td><td>H</td><td>Hi-Z</td></tr>
<tr><td>$\overline{BCYST}$ (PCT7)</td><td>H</td><td>Hi-Z</td></tr>
<tr><td>$\overline{WAIT}$ (PCM0)</td><td>H</td><td>Hi-Z</td></tr>
<tr><td>CLKOUT (PCM1)</td><td>Operates</td><td>Operates</td></tr>
<tr><td>$\overline{HLDAK}$ (PCM2)</td><td>H</td><td>Hi-Z</td></tr>
<tr><td>$\overline{HLDRQ}$ (PCM3)</td><td>–</td><td>Hi-Z</td></tr>
<tr><td>$\overline{REFRQ}$ (PCM4)</td><td>H</td><td>Hi-Z</td></tr>
<tr><td>SELFREF (PCM5)</td><td>–</td><td>Hi-Z</td></tr>
<tr><td>SDCKE (PCD0)</td><td>L</td><td>Hi-Z</td></tr>
<tr><td>SDCLK (PCD1)</td><td>Operates</td><td>Hi-Z</td></tr>
<tr><td>$\overline{LBE}$, $\overline{UBE}$ (PCD2, PCD3)</td><td>H</td><td>Hi-Z</td></tr>
<tr><td>$\overline{DMAAK0}$ to $\overline{DMAAK3}$ (PBD0 to PBD3)</td><td>H</td><td>Hi-Z</td></tr>
</table>
Workaround
None. This restriction applies only to V850E/MA1.
</td></tr>
</table>

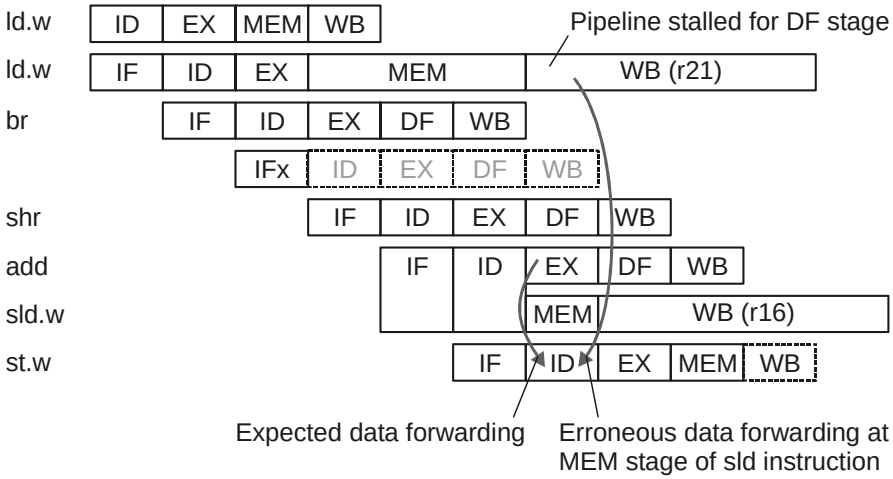
No.	Outline	Description
a-22	Incorrect write-back with LD/SLD instruction when executing CALLT/SWITCH instruction	*Details* An LD/SLD instruction does not write correctly to the register in the following case: • A SWITCH/CALLT instruction is executed before the bus cycle of the LD/SLD instruction is complete, and • bits 15 to 11 of the instruction op-code immediately after the SWITCH/CALLT instruction are the same as bits 15 to 11 of op-code of the previous LD/SLD instruction. *Workaround* Either of the following two measures can be used: 1) Set bits 15 to 11 of the half-word immediately after CALLT/SWITCH to 00000B. 2) Insert the instruction “MOV reg2, r0” just before the SWITCH/CALLT instruction and at the end of an interrupt routine (just before the RETI instruction). “reg2” must be the same register as the second operand of LD/SLD instruction (reg2). The reason why the MOV instruction has to be inserted in the interrupt processing routine is that the same phenomenon can occur if the LD/SLD instruction is executed in the interrupt processing routine. It is not necessary to insert a MOV before RETI if operation is limited to fetch via VSB.
a-23	Restriction on use of external bus when product is employed as an emulator for the V850E/IA1	*Details* The external bus cannot be used when used as emulator of V850E/IA1. *Workaround* None.
a-24	Restriction related to the output of the DMAAK signal	*Details* Although the $\overline{\text{DMAAK}}$ signal is intended to remain active during 2-cycle DMA transfer, the $\overline{\text{DMAAK}}$ signal becomes inactive for one clock cycle between the read cycle an 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 $\overline{\text{DMAAK}}$ signal. *Workaround* None.
a-25	Restriction related to starting DMA by built-in peripheral I/O interrupt	*Details* Depending on the timing, it might be possible that starting the DMA by interrupt of the built-in peripheral I/O fails. *Workaround* None.

No.	Outline	Description
a-26	Restriction related to EDO DRAM bus collision	*Details* 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 mode.
a-27	Restriction on the 2-way associative function of the instruction cache	*Details* 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* None. Use 4 KB (2 KB)-Direct mode instead of 8 KB (4 KB)-2way associative function.
a-28	Forced stop of external DMA transfer in DMA line transfer mode	*Details* The DMA will monopolize 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 hang-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.
a-29	Restriction on reading the DCHC register when DMA 2-cycle transfer is completed	*Details* 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 internal RAM connected to the VDB bus to another internal RAM connected to the VDB. *Workaround* None. Do not use RAM-to-RAM DMA transfer.
a-30	Restriction related to conflict between SDRAM initialization and SELFREF input	*Details* 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 is completed.

No.	Outline	Description
a-31	Restriction on half-word writing to BSC, BCC, DWC0, and DWC1 registers	*Details* Data cannot be written correctly to the high-order 8 bits (8 to 15) of a BSC, BCC, DWC0, and DWC1 register with half-word (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 $\overline{CS4}$ to $\overline{CS7}$. • The bus size for $\overline{CS4}$ to $\overline{CS7}$ does not changed to the value set with the BSC register. • DATA-wait set with the DWC0 register is not inserted in $\overline{CS4}$ to $\overline{CS7}$. • DATA-wait set with the DWC1 register is not inserted in $\overline{CS6}$ to $\overline{CS7}$. *Workaround* Before performing half-word write to set the BSC, BCC, DWC0, and DWC1 registers, byte-write the data which is to be set to the high-order 8 bits to the low-order 8 bits and then perform half-word write. *Example:* When writing 0x1234 to BSC (0xFFFFF066) *Example of access without preventive measure:* movhi 0xFFFF,r0,r10 ori 0xF0000,r10,r10 ori 0x1234,r0,r11 st.h r11,0x66[r10] *Example of access with preventive measure* movhi 0xFFFF,r0,r10 ori 0xF0000,r10,r10 ori 0x12,r0,r11 st.b r11,0x66[r10] ori r0x1234,r0,r11 st.b r11,0x66[r10]
a-32	Restriction related to SDRAM write operation	*Details* Data output from the second cycles 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 half-word data access or word data access.
a-33	Restriction on DRAM fetch immediately after block DMA transfer from DRAM to internal RAM	*Details* The OE is not asserted active 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* None.

No.	Outline	Description
a-34	Restriction on instruction cache (2)	*Details* 1) A certain instruction order is necessary before enabling the instruction cache by the cache configuration register (BHC). 2) 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 area cannot be used to change the instruction cache setting of the CS0 area. However, instructions in a memory area connected to VFB (IROM) or VDB (IRAM) can be used to change the cache settings for all CS areas. *Workaround* 1) Be sure to execute the following instruction before setting the BHC register with the program settings immediately after reset: st.h r0, 0xFFFFF072[r0] 2) Set FILL0 bit (bit 5) of the instruction cache controll register (ICC) to 0. The auto fill function can only be used with way 0.

No.	Outline	Description
a-35	Restriction on SLD instruction	*Details* Register read data does not reflect the latest value after dual-issued pipe-line operation of SLD instruction with the following instruction sequences: *Sequence type 1:* (1) xxx[1] *, * -- 1 clk 2 byte instruction with WB to any GP register (2) sld *, rX -- data load from iRAM to rX (old data) (3) xxx[2] *, rX -- 1 clk 2 byte instruction with WB to rX ⎫ dual (4) sld *, rY -- data load from iRAM to rY ⎭ issued (5) xxx[3] (instruction which reads from rX) -- old data is used for rX incorrectly *Sequence type 2:* (1) ld/sld *[r*], rX -- data load from iRAM to rX (old data) (2) xxx[4] (one or more instructions) -- 1 clk instruction with WB, but no write to rX (3) xxx[2] *, rX -- 1 clk 2 byte instruction with WB to rX ⎫ dual (4) sld *, rY -- data load from iRAM to rY ⎭ issued (5) xxx[3] (instruction which reads from rX) -- old data is used for rX incorrectly *Sequence type 3:* (1) ld/sld *[r*], rX -- data load from external memory to rX (old data) (2) xxx[5] (any instruction, which is not a load, and without WB to rX) (3) xxx[4] (one or more instructions) -- 1 clk instruction with WB, but no write to rX (4) xxx[2] *, rX -- 1 clk 2 byte instruction with WB to rX ⎫ dual (5) sld *, rY -- data load from iRAM to rY ⎭ issued (6) xxx[3] (instruction which reads from rX) -- old data is used for rX incorrectly **Notes 1.** Instructions, which write to GP register other than r0, r30 (ep): add, and, mov, not, or, sar, satadd, satsub, satsubr, shl, shr, sub, subr, sxb, sxh, xor, zxb, zxh **2.** Instructions, which write to rX register other than r0, r30 (ep): add, and, mov, not, or, sar, satadd, satsub, satsubr, shl, shr, sub, subr, sxb, sxh, xor, zxb, zxh **3.** Instructions, which read from rX register: add, addi, and, andi, bsh, bsw, clr1, cmov, cmp, div, divh, divhu, divu, hsw, jmp, ld.b, ld.h, ld.w, ld.bu, ld.hu, ldsr, mov, movea, movhi, mul, mulh, mulhi, mulu, not, not1, or, ori, sar, sasf, satadd, satsub, satsubi, satsubr, set1, shl, shr, sld.b, sld.h, sld.w, sld.bu, sld.hu, sst.b, sst.h, sst.w, st.b, st.h, st.w, sub, subr, switch, sxb, sxh, tst, tst1, xor, xori, zxb, zxh **4.** Instructions, which write to GP register other than r0, rX and do not use rX: add, addi, and, andi, bsh, bsw, cmov, hsw, ldsr, mov, movea, movhi, not, or, ori, sar, sasf, satadd, satsub, satsubi, satsubr, setf, shl, shr, stsr, sub, subr, sxb, sxh, xor, xori, zxb, zx **5.** Any instruction, which is neither load instruction, nor does WB to rX: All instructions **except** ld.b, ld.h, ld.w, ld.bu, ld.hu, sld.b, sld.h, sld.w, sld.bu, sld.hu, dispose, set1, not1, clr1, tst1, callt, switch

No.	Outline	Description
a-35	(cont.)	*Example (Sequence type 3)* rX = ***r21*** Var0, Var1 = labels of variables located in external memory Var2 = label of variable located in internal RAM tgt = label of branch address ld.w Var0[r0], r20 ***ld***.w Var1[r0], ***r21*** (1) ... external memory access br tgt (2) ... instruction, which is not a load, and without WB to rX : tgt: shr 1, r20 (3) ... 1 clk instruction with WB add r20, ***r21*** (4) ... 1 clk 2 byte instruction with WB to rX ***sld***.w Var2[ep], r16 (5) ... iRAM access st.w ***r21***, Var1[r0] (6) ... instruction which reads from rX  Legend IF : Instruction Fetch. Instruction is fetched and fetch pointer is incremented. ID : Instruction Decode. Instrcution is decoded, immediate data is generated, and register is read. EX : EXecution of ALU, multiplier, and barrel shifter. The decoded instruction is executed. MEM: MEMory access. The memory at specified address is accessed. WB : Write Back. The result of execution is written to register. DF : Data Fetch. Execution data is transferred to the WB stage.
a-36	Unusable I/O when VSB bus is used	*Details* The following I/O addresses cannot be used when using the VSB bus (including image area). 64 MB mode: 3FFF480H to 3FFF4BEH 256 MB mode: FFFF480H to FFFF4BEH *Workaround* None.

No.	Outline	Description
a-37	Restriction on instruction cache (3)	*Details* 1) 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 an illegal instruction and hangs up. This bug occurs, *if a memory area including interrupt handler addresses from 00000000H to 00000800H is set as a cachable area by the BHC register*. This bug does not occur, when • The memory area including interrupt handler addresses is set as an uncacheable area. • The memory area including interrupt handler addresses is located in the internal ROM area , and if the bits 0, 4, 8, and 12 of the chip area selection control register 0 (CSC0) are all cleared to 0. 2) If TAG is cleared by the ICC register, four lines of TAG (one line of cache consists of 4 words), lines 0 to 3 (INDEX = 00H to 03H), may not be cleared, depending on the operation status of the cache before TAG is cleared. This bug occurs *under all conditions where the instruction cache is used*. 3) If the HALT instruction is executed and then the halt mode is released, the instruction cache may return an illegal instruction to the CPU, resulting in a hang-up. This bug occurs, *if the HALT instruction is not aligned with a 4-word boundary*. *Workaround* 1) To avoid the first restriction take any of the following measures a) to c) below in software. a) Set the memory area including interrupt handler addresses from 00000000 to 00000800H as an uncacheable area using the BHC register. b) If it is necessary to set a chip select area including interrupt handler addresses as a cacheable area, and if an uncacheable memory area exists in another chip select area, ... branch to the uncacheable area first and from there branch to the original interrupt processing routine. Example: `.offset 0x80 -- interrupt handler address` `jr INT0_UC` `.section “nocache” -- uncacheable area` `INT0_UC:` `jr INT0` `.section “cache” -- cacheable area` `INT0: -- (original interrupt processing routine)` `...` `...` `...` `reti`

No.	Outline	Description
a-37	(cont.)	c) If it is necessary to set a chip select area including interrupt handler addresses as a cacheable area, and if an uncacheable memory area does not exist in another chip select area, ... branch to different TAG lines in the cacheable area more than four times, immediately after branching to the interrupt handler address, and the branch to the original interrupt processing routine. Example: .offset 0x80 -- interrupt handler address jr INT0_DMY .section “dmy” -- uncacheable area INT0_DMY: jr line0 .rept 7 nop .endr line0: jr line1 .rept 7 nop .endr line1: jr line2 .rept 7 nop .endr line2: jr line3 .rept 7 nop .endr line3: jr INT0 .section “cache” -- cacheable area INT0: -- (original interrupt processing routine) ... ... reti

No.	Outline	Description
a-37	(cont.)	2) To clear the TAG in software, clear the TAG two times (by writing to the ICC register and waiting for completion). Example: `mov 0x03, r2` `LOP0:` `ld.h ICC[r0], r1` `cmp r0, r1` `bnz LOP0` `st.h r2, ICC[r0] -- TAG cleared first time` `LOP1:` `ld.h ICC[r0], r1` `cmp r0, r1` `bnz LOP1` `st.h r2, ICC[r0] -- TAG cleared second time` `LOP2:` `ld.h ICC[r0], r1` `cmp r0, r1` `bnz LOP2` **Please regard this as a permanent restriction.** 3) To avoid the third restriction allocate the HALT instruction to the beginning of a 4-word boundary, or 16-byte boundary respectively. Example (assembler): `jr EXIT` `.section “tmp2”, .text` `.align 16` `EXIT:` `halt`
a-38	Restriction of $\overline{\text{DMAAK}}$ signal during DMA line transfer	*Details* The $\overline{\text{DMAAK}}$ signal is output at the same timing as the NB85E core output signal during DMA line transfer. *Workaround* None. **Please regard this as a permanent restriction.**

No.	Outline	Description
a-39	Restriction caused by interrupt input during execution of bit manipulation instruction	*Details* If a bit manipulation instruction access (SET1, CLR1, or NOT1) and an interrupt request to the following registers contend, two more addresses are illegal written into the interrupt return address saving registers (EIPC and FEPC) as interrupt addresses, and execution branches to the illegal addresses when it has returned from the interrupt routine. All maskable interrupts as well as the non-maskable interrupt (NMI) are concerned, when bit manipulation instruction are executed for the following peripheral registers: • for V850E/CA1 or system LSI (64 MB mode) all registers mapped from 3FFF100 to 3FFF1FF all registers mapped from 3FFF900 to 3FFF9FF • for V850E/MA1, V850E/IA1 or system LSI (256 MB mode) all registers mapped from FFFF100 to FFFF1FF all registers mapped from FFFF900 to FFFF9FF **Note** This bug occurs in the in-circuit emulator (IE-V850E-MC or IE-V850E-MC-A) only, and does not affect the target device (V850E/MA1, V850E/IA1, V850E/CA1, or NB85E core). The bug does not occur, if all of the following conditions are satisfied: • Interrupts are always disabled (DI status) when the peripheral registers are accessed. • NMI is not used, or NMI is never enabled when a peripheral register is accessed (NMI is used to release the standby mode only). *Workaround* 1) The malfunction by execution of bit manipulation instructions can be avoided in any of the following ways. a) If the NMI is not used, or if NMI is never enabled when a peripheral register is accessed: Always access the peripheral register (by using a bit manipulation instruction) when the interrupts are disabled (DI status). Example: stsr PSW, rx di set1/clr1/not1 instruction ldsr rx, PSW b) Disable interrupts 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 value 0xffff into IMR0 to IMR3 (disable interrupt processing) access peripheral register (by using a bit manipulation instruction) restore the saved values of IMR0 to IMR3

No.	Outline	Description
a-39	(cont.)	c) Do not use a bit manipulation instruction to access the concerned peripheral registers. Instead of a bit manipulation instruction use a read-modify-write sequence. Example (assembler) for bit clearing ld sfr[r0], rx andi not MASK, r0, rx st rx, sfr[r0] Example (assembler) for bit setting ld sfr[r0], rx ori MASK, r0, rx st rx, sfr[r0] Example (C compiler) for bit clearing: sfr &= ~MASK; Example (C compiler) for bit setting: sfr |= MASK; 2) If there is a NMI occurrence possible while a concerned peripheral register is accessed, do not use a bit manipulation instruction. Instead of a bit manipulation instruction use a read-modify-write sequence (see above).
a-40	Restriction on hardware stop during bit manipulation instruction execution	*Details* 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. **Note** This applies only when debugging a system LSI. This a bug that occurs in the in-circuit emulator (IE-V850E-MC or IE-V850E-MC-A) and does not occur in the target device (NB85E core). *Workaround* None.

No.	Outline	Description
a-41	Restriction related to interruption of DMA transfer by external cause	*Details* 1) Device development (V850E/MA1, V850E/IA1, V850E/CA1) If an NMI is input during a DMA transfer, the transfer operation is usually interrupted and the contents of the Enn bits of the DCHCn registers are retained in the DDID register. If , however, DMA is started by the DMARQn signal or internal peripheral I/O interrupt and transfer is performed in single transfer mode, the contents of the Enn bits of the DCHCn registers are not retained in the DDIS register even if an NMI is input. Consequently, the interrupted DMA transfer cannot be resumed by using the DRST register (n = 0 to 3). The bug is not effective if any of the following conditions are met. • Single step transfer mode, or block transfer mode is used. • An NMI is not input during DMA transfer. • DMA is activated using a software trigger only. • The DMA restart function by the DRST register is not used. 2) System LSI development If a single DMA transfer that has been activated by DMARQn pin input is interrupted by inputting a signal of one VBCLK width to the IDMASTP pin, the contents of the Enn bits of the DCHCn registers are not saved to the DDIS register even. Consequently, the interrupted DMA transfer cannot be resumed by using the DRST register (n = 0 to 3). The bug is not effective if any of the following conditions are met. • When active level (high level) of 2 clocks is input to the IDMASTP pin at the rising edge of VBCLK. • When DMA transfer is not executed in single transfer mode. • If DMA transfer is not interrupted. • DMA is activated using a software trigger only. • 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* Take one of the following measures to avoid the bug occurence. 1) Device development (V850E/MA1, V850E/IA1, V850E/CA1) a) Execute a dummy single transfer with one of the DMA channels 0 to 3 before activating the first DMA. After that, do not end or abort transfer of that 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 0001H or greater (number of transfer times: 2 min.). Step 3: Set DADCx to 0000H. Step 4: Set DCHCx to 03H x = number of DMA channel that is not used = 0 to 3 **Note** If this measure is used, the DMA channel that is used for dummy transfer becomes unusable. Use this measure, when there is a redundant DMA channel.

No.	Outline	Description
a-41	(cont.)	b) Execute a software-triggered dummy transfer once immediately before executing a single DMA transfer. c) Use a software trigger to activate DMA, rather than an interrupt from the internal peripheral I/O or input to the DMARQn pin. d) When DMA transfer is interrupted, save Enn (bit 0 of the DCHCx register) into user space by the NMI processing routine. When DMA transfer is resumed by leaving the NMI processing routine, set the corresponding bit of the DRST register to 1 by viewing the saved value. 2) System LSI development a) Input an active level (high level) of 2 clocks to the IDMASTP pin at the rising edge of VBCLK. b) Execute a dummy single transfer with one of the DMA channels 0 to 3 before activating the first DMA. After that, do not end or abort transfer of that 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 0001H or greater (number of transfer times: 2 min.). Step 3: Set DADCx to 0000H. Step 4: Set DCHCx to 03H x = number of DMA channel that is not used = 0 to 3 **Note** If this measure takes place, the DMA channel that is used for dummy transfer becomes unusable. Use this measure, when there is a redundant DMA channel. c) Execute a software-triggered dummy transfer once immediately before executing a single DMA transfer. d) Use a software trigger to activate DMA, rather than an interrupt from the internal peripheral I/O or input to the DMARQn pin. e) When DMA transfer is interrupted, save Enn (bit 0 of the DCHCx register) into user space by the NMI processing routine. When DMA transfer is resumed by leaving the NMI processing routine, set the corresponding bit of the DRST register to 1 by viewing the saved value.

2. Restrictions on debug functions

No.	Outline	Description
b-1	Restrictions on operating frequency	*Details* The maximum operating frequency is 40 MHz. *Workaround* None. Use a frequency of 40 MHz or lower.
b-2	Restrictions on break timing when guard area is fetched	*Details* When program execution enters the guard area, one instruction in the guard area is executed and then a break occurs. *Workaround* None. Please regard this as a permanent restriction.
b-3	Restrictions on trace in case of mis-alignment (during read access only)	*Details* 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. *Workaround* None. Exercise care when referencing trace data if mis-align read access has been executed.
b-4	Restriction on trace data on execution of HALT or STOP instruction	*Details* 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 in not traced (access data to the register is traced). After the PSC register has been accessed, execution of the last of several NOP instructions in not traced. 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. *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.
b-5	Bit manipulation instruction (set1, clr1, not1, tst1) access data is illegally traced by tracer	*Details* If bits of an address where the low-order 2 bits are not 00B, the access data of the trace data has an illegal value. *Workaround* None. Exercise care when referencing trace data of an instruction that manipulates the bits of an address in question.

No.	Outline	Description
b-6	Events including data conditions by access of bit manipulation instruction cannot be detected	*Details* If a bit manipulation instruction is executed on an address where the low-order 2 bits are not 00B, 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.
b-7	Restrictions on HOLD status	*Details* 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.
b-8	ROM contents are rewritten if emulation ROM area is accessed for write	*Details* An illegal break occurs if the emulation ROM area is accessed for write, and the data of ROM is rewritten. *Workaround* None. Please regard this as a permanent restriction.
b-9	Restriction on SFR illegal break	*Details* SFR illegal break is detected by “address condition + R/W attribute”. However, a break occurs if an 8-bit SFR is written in half-word units (break does not occur if an 8-bit SFR is read in half-word units). *Workaround* None. Please regards this as a permanent restriction.
b-10	Restriction on programmable I/O space	*Details* a) If a programmable I/O space is mapped to the high-order 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 access during program execution, an SFR illegal break occurs. *Workaround* a) Map the programmable I/O space to the low-order 32 MB area b) Preventive measures will be announced in a separate document. This restriction is planned to be corrected in the next version.

No.	Outline	Description
b-11	Break does not occur even if breakpoint is set	*Details* • (Dis)assembly level: If breakpoints are set for two instructions in a row and a break occurs at the first instruction, a break may not occur at the second instruction in response to the subsequent request for resumption of execution. • Source level: If breakpoints are set for two execution statements in a row (of which is es 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. *Workaround* • For software breakpoints preventive measures are implemented to the following debugger versions: NEC debugger: ID850 E2.20f and later versions. GHS Multi: Use EX85032.DLL version 5.40 or later. • There is no preventive measure for hardware breaks.
b-12	Restrictions related to access address during DMA trace	*Details* If DMA is started while 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* None. Please regard this as a permanent restriction.
b-13	Restriction on DBPC and DBPSW access during a break	*Details* Although DBPC and DBPSW can be read during a break, they cannot be written to. *Workaround* None. Please regard this as a permanent restriction.
b-14	Restriction on DBTRAP instruction	*Details* If a break occurs in the interrupt processing 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* None. Please regard this as a permanent restriction.
b-15	Restriction on illegal guard break when IRAM size is 28 KB	*Details* A guard break occurs in the IRAM area when a fetch is executed on the area 3FF8000H to 3FFC000H 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.

No.	Outline	Description
b-16	Restriction on illegal trace when big endian is used	*Details* 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* None.
b-17	Restriction on access data traced by DMA	*Details* When data in IRAM is read by DMA, the read data value is not traced correctly. *Workaround* None.
b-18	Restriction on SFR read access during break	*Details* The SFR bits that are normally cleared by a read access (e.g. TC bit of the DCHC register) are also cleared when displayed during a break by using the SFR display function in the debugger. (Even though these SFR bits are not read by the program they are cleared by displaying them.) *Workaround* None. However, the bit is not reset if the relevant SFR is not displayed by the debugger.

(D) Cautions

This chapter is intended to give additional information for situations that could affect the system operation and performance.

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

- Insert 1 wait cycle if the operating clock is 25 MHz or higher.
- Set the bus width to 16 or 32 bits. The 8-bit bus cannot be used.
- The emulation memory cannot be mapped to addresses higher that 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 the setting of the debugger or the setting of the wait control register.

With Multi the wait pin can be masked or unmasked by the “pinmask” command:
a) wait mask (default) -> Accessed with 1 wait.
b) non mask -> Accessed with the number of wait cycles set by the DWC0/1 register. However, the number of wait cycles is 1 regardless of whether 0 or 1 wait cycle is specified.

2. Pin processing

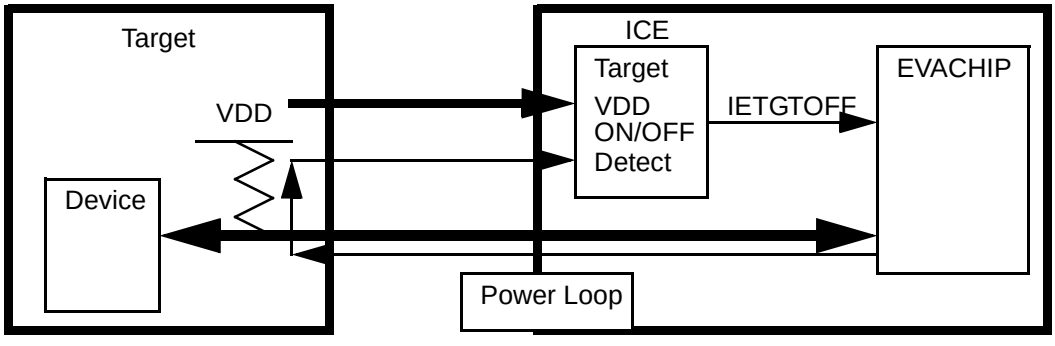
- This ICE uses minimum pin processing, giving priority to transparency with the device. Take adequate countermeasures against static electricity if the ICE is used without the target connected.
- Inside the ICE, pin processing is performed by the option board. For details, refer to the User’s Manual 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).

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 the target power is turned ON or OFF. In this reset status, the external bus signal go inter 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.



(E) Notes on Trace Data

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.

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.

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.

The external logic data is sampled in synchronization with instruction execution and differs depending 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 on 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, therefor 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.