CUSTOMER NOTIFICATION

SUD-T-4204-6-E

January 26, 2001

Yoichi Hirasawa, Expert
Microcomputer Engineering Dept.
Solution Engineering Div.
NEC Electron Devices
NEC Corporation

CP (K), O

# IE-703102-MC (Control Code: A, B, C, D)
# IE-703102-MC-EM1 (Control Code: A, B)
# IE-703102-MC-EM1-A (Control Code: A, B)

# RESTRICTIONS

Be sure to read this document before using the product.

**This document is common to 5 V emulation and 3.3 V emulation.**

- Configuration of 5 V emulation: IE-703102-MC + IE-703102-MC-EM1
- Configuration of 3.3 V emulation: IE-703102-MC + IE-703102-MC-EM1-A

# 1. PRODUCT VERSION

Part number: IE-703102-MC

| Control Code | EVA Chip | Usable Debugger | Remark |
|---|---|---|---|
| A | UPD703190R DS3.1 | ID850 V1.32 | |
| B | UPD703190R DS5.0 | ID850 E2.00r or above | |
| C | UPD703190R DS5.1 | ID850 E2.00r or above | |
| D | UPD703190R ES5.1 | ID850 E2.21f or above | |

**Note** Debugger versions that can be used with the IE-703102-MC differ depending on the control code.

**If you have any questions concerning the product, consult the engineers in charge of the V850 Series in-circuit emulator or debugger at the Development Tool Support Center.**

**TEL (044) 435-9454 (TELNET: 8-22-24085)**

**E-mail: toolsupport@saed.tmg.nec.co.jp**

Part number: IE-703102-MC-EM1

| Control Code | Peripheral Chip | Remark |
|---|---|---|
| A | uPD70F3102GJ (5 V: ES5.0, 5.1, 6.1) | |
| B | uPD70F3102GJ (5 V: MI specification) | |

Part number: IE-703102-MC-EM1-A

| Control Code | Peripheral Chip | Remark |
|---|---|---|
| A | uPD70F3102AGJ (3.3 V: ES2.0, 2.1) | |
| B | uPD70F3102AGJ (3.3 V: MI specification) | |

**Note** The control code is the second digit from the left of a 10-digit control code that starts from E.

## 2. LIST OF RESTRICTIONS

| No. | Bugs | Affected Products and Corresponding Control Code | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | IE-703102-MC | | | | IE-703102-MC-EM1 | | IE-703102-MC-EM1-A | |
| | | A | B | C | D | A | B | A | B |
| 1 | The maximum operating frequency is 33 MHz. | × | × | × | √ | × | √ | × | √ |
| 2 | Forced shut-down during DMA transfer | × | √ | √ | √ | – | – | – | – |
| 3 | Restriction on Repeated Execution of sld Instruction | × | × | × | × | – | – | – | – |
| 4 | Bus lock bug | × | √ | √ | √ | – | – | – | – |
| 5 | Invalid output of DMAAK signal | × | √ | √ | √ | – | – | – | – |
| 6 | Problem related to multiplication instructions | × | √ | √ | √ | – | – | – | – |
| 7 | A non-map break occurs when an internal I/O or an internal RAM is accessed with a ROMless product or during operation in ROMless mode | × | √ | √ | √ | – | – | – | – |
| 8 | An illegal I/O break occurs when an internal I/O space is accessed by DMA | × | √ | √ | √ | – | – | – | – |
| 9 | Illegal write protect break in ROMless mode | × | √ | √ | √ | – | – | – | – |
| 10 | Illegal internal RAM fetch when branching from external memory to the internal ROM | × | √ | √ | √ | – | – | – | – |
| 11 | Coverage bug | × | × | × | × | – | – | – | – |
| 12 | Bug on DMAAK output timing | × | √ | √ | √ | – | – | – | – |
| 13 | Bug on CLKOUT relative specification | × | √ | √ | √ | – | – | – | – |
| 14 | Bug on port register read at output port | × | × | × | × | – | – | – | – |
| 15 | Bug on trace immediately after standby release | × | × | × | × | – | – | – | – |
| 16 | Illegal write protect break at write access is performed to the emulation RAM | × | × | × | × | – | – | – | – |
| 17 | A write protect break does not occur even when write access to the emulation ROM is performed using a program stored in IRAM | √ | × | √ | √ | – | – | – | – |
| 18 | Bug on flyby DMA when IDLE insertion, DMA request, and refresh conflict | √ | × | √ | √ | – | – | – | – |
| 19 | Restriction on STOP mode release | – | – | – | – | × | √ | × | √ |
| 20 | Restriction on temperature range | × | × | × | × | × | × | × | × |

| No. | Bugs | Affected Products and Corresponding Control Code | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | IE-703102-MC | | | | IE-703102-MC-EM1 | | IE-703102-MC-EM1-A | |
| | | A | B | C | D | A | B | A | B |
| 21 | Breaks and events may not be detected at branch instruction | × | × | × | × | – | – | – | – |
| 22 | Restriction on PC interface board | × | × | × | √ | – | – | – | – |
| 23 | Restriction on operation at 40 MHz | × | × | × | × | × | × | × | × |
| 24 | Connection to the IE-703100-MC | – | – | – | – | – | – | – | **Note** |

√: Already confirmed and solved

×: Corresponding restriction exists

–: Bugs do not apply

**Note**: Refer to 3. DETAILS OF RESTRICTIONS

## 3. DETAILS OF RESTRICTIONS

(1) The maximum operating frequency is 33 MHz.

[Description] When operating frequency exceeds 33 MHz, the internal ROM is illegally accessed and the peripheral I/O does not operate correctly.

[Countermeasures] This bug is corrected in the IE-703102-MC with control code D.

(2) Forced shut-down during DMA transfer

[Description]

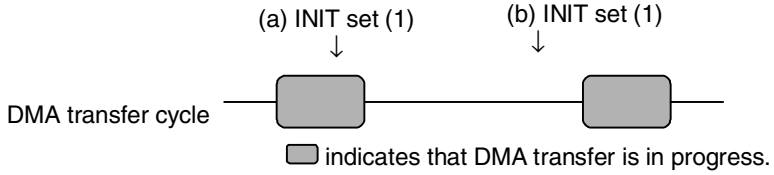
Normal operation:

If the following registers are re-written during DMA transfer, the new information is validated by issuing a terminal count or setting the DCHCn register INIT bit to 1.

Abnormal operation:

It is impossible to re-set the following registers, even if the INIT bit is set to 1, only in a case where single transfer mode is used. This problem invariably occurs during transfer from internal I/O to internal RAM. This problem does not occur if the INIT bit set (1) operation overlaps with the DMA transfer cycle when transfer is performed between two external memory areas (or external I/O). (See figure.)

**Transfer between two external memory areas (or external I/O)**




Problem does not occur when INIT bit is set to 1 with timing (a).

Problem occurs when INIT bit is set to 1 with timing (b).

Applicable registers:

DMA source address register (DSAnH, DSAnL)

DMA destination address register (DDAnH, DDAnL)

DMA byte count register (DBCn) (n = 0-3)

[Countermeasures]

Because this problem only occurs in single transfer mode, desired operation is possible by re-initializing once you have switched to single step transfer mode.

After forced DMA shut-down by setting the INIT bit, set an address that does not negatively influence the system, even if DMA transfer to the applicable address is executed, in the DMA related peripheral I/O register and start single step transfer mode DMA transfer to that address with the soft-trigger. With the instruction that immediately follows (executed when the CPU uses the bus interchangeable with DMA), reset the INIT

bit. When this is done, normal initialization can be performed.

Specific instruction sequence:

```
DMA_INIT:  mov     0x04, r28
           st.b    r28, DCHCn          --Sets INIT bit and prohibits DMA transfer.

           movea   0xFFFFFFFF, r0, r28
           movea   0xFFFFFFFC, r0, r29
           st.h    r28, DSAnH          -- Sets the source and destination addresses
                                          to the reserved area of the internal
                                          peripheral I/O.
           st.h    r29, DSAnL
           st.h    r28, DDAnH
           st.h    r29, DDAnL

           movea   0x00A4, r0, r28
           st.h    r28, DADCn          -- Changes to single step transfer
           mov     0x03, r28
           st.b    r28, DCHCn          -- Reatarts dummy DMA transfer with soft-
                                          trigger.

           mov     0x04, r28
           st.b    r28, DCHCn          -- Reset INIT bit and prohibits DMA transfer.
```

(3) Restriction on repeated execution of sld Instruction

[Description]

The data that should be loaded may not be transferred correctly to the register when sld instructions accessing external memory are repeatedly executed, and interrupt processing is executed.

Details of Malfunction:

This malfunction occurs when the following conditions are met:

The instruction sequence <1> to <4> is executed, and at <4> the sld instruction is executed repeatedly. The malfunction occurs when an interrupt occurs during the execution of the second sld instruction.

When the malfunction occurs, the data loaded by the sld instruction immediately before the interrupt is incorrectly written to the register.

<1> ld instruction or sld instruction

<2> One or more of any instruction other than the ld or mul instruction

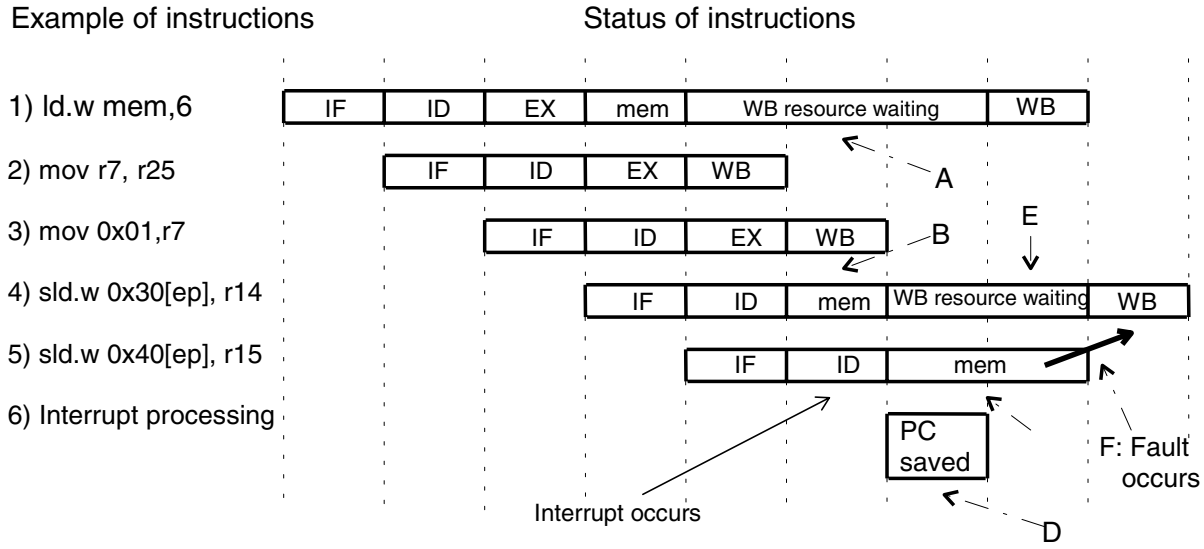
<3> One or more instructions that write to a register (applicable instructions are shown in table 1)

<4> sld instructions (repeated execution of sld instructions where data is loaded from external memory.)
sld instruction ← Interrupt occurs.
sld instruction
:

This malfunction does not occur unless the sequence <1> to <4> is executed. Cases where this malfunction does not occur are shown below:

- The sld instructions in <4> load from internal memory
- The instructions in <4> are load instructions other than sld
- The repeated sld instructions are separated by a branch instruction (such as a jr instruction or a reti instruction)
- An ep (element pointer) setting occurs immediately before the repeated sld instructions
- A nop instruction is executed immediately before the repeated sld instructions
- Interrupts are disabled before executing the sequence <1> to <4>.

This error occurs when the following pipeline status occurs.




<Explanation of the above pipeline operation>

A. To improve performance with the V850E/MS1, the pipelining of other instructions goes ahead regardless of the dst waiting for resources in instruction 1, and consequently the WB of instructions 2 and 3 are executed before the WB of instruction 1.

↓

B. The EX stage of instruction 4 is omitted and mem access is executed because address calculation resources are available in the ID stage.

↓

C. Execution of mem access cannot be stopped even if an interrupt is received in the ID stage of instruction 5, so a dummy access is performed.

↓

D. The PC is saved, and so on, due to the interrupt processing. Since this uses WB resources, the WB of instruction 1 is delayed even more.

↓

E. The WB of instruction 4 is made to wait one clock cycle because otherwise it would conflict with the WB of instruction 1.

↓

F. The data written in the WB stage of instruction 4 is the result of the dummy cycle of the mem stage of instruction 5, because reading cannot be masked due to the dummy cycle in instruction 5, and an incorrect write to r14 is performed, causing the malfunction.

**Table 1. Instructions that write to a register immediately before the sld instructions**

| Mnemonic | Operand | Mnemonic | Operand |
|---|---|---|---|
| mov | R, r | movea | imm16, R, r |
| not | R, r | movhi | imm16, R, r |
| divh | R, r | satsubi | imm16, R, r |
| satsubr | R, r | mov | Imm32, R |
| satsub | R, r | ori | imm16, R, r |
| satadd | R, r | xori | imm16, R, r |
| zxb | R | andi | imm16, R, r |
| sxb | R | setf | cccc, r |
| zxh | R | ldsr | R, sr |
| sxh | R | stsr | SR, r |
| or | R, r | shr | R, r |
| xor | R, r | sar | R, r |
| and | R, r | shl | R, r |
| subr | R, r | sasf | cccc, r |
| sub | R, r | divh | R, r, w |
| add | R, r | divhu | R, r, w |
| mov | imm5, r | div | R, r, w |
| satadd | imm5, r | divu | R, r, w |
| add | imm5, r | cmov | imm5, r, m |
| shr | imm5, r | cmov | R, r, w |
| sar | imm5, r | bsw | r, w |
| shl | imm5, r | bsh | r, w |
| addi | imm16, R, r | hsw | r, w |

[Countermeasures]

<Assembler>

This malfunction can be avoided by using one of the following countermeasures **a** through **c** below:

a. Change the first sld instruction to an ld instruction where repeated sld instructions are used. This malfunction will not occur if all the sld instructions are changed to ld instructions.
b. Set ep immediately before the first sld instruction when repeated sld instructions are used.
c. Insert a nop instruction immediately before the first sld instruction when repeated sld instructions are used.

<NEC Compiler>

We have confirmed that the NEC compiler does not output the instruction sequence that causes this malfunction. Therefore, it is not necessary to take any countermeasures.

<GHS Compiler>

This malfunction can be avoided using one of the following countermeasures so that the compiler does not output repeated sld instructions.

1) Specify the following option at compilation:
   -Z1412
2) Or, avoid using a TDA (Tiny Data Area) function pragma.

<OS (RX850, RX850PRO)>

We have confirmed that these operating systems are not affected by this malfunction. Therefore, it is not necessary to take any countermeasures.

<Middleware (MH, MR, MMR, JBIG, JPEG, Handwriting Recognition, Text To Speech, US File, CF Driver)>

We have confirmed that this middleware is not affected by this malfunction. Therefore, it is not necessary to take any countermeasures.

(4) Bus lock bug

[Description]

If an instruction fetch request from external memory and DMA request using the external bus conflict during misalign-access (refer to Attachment 1) to external memory, or when SET1, NOT1, and CLR1 instructions are executed, the external bus cycle will lock and CPU instruction execution will stop. However, internal peripheral I/O itself will continue operating.

Non-applicable conditions:

Any of the following conditions will render this problem non-applicable whether the DMA function is used or not.

- Instruction fetch is not performed to external memory (the instruction code exists in the internal memory only).
- DMA transfer is performed between internal RAM and internal I/O only.

- There is no misalign-access and SET1, NOT1, and CLR1 instructions are not used.

Details of problem:

As indicated by Attachment 1, a number of bus cycles occur when misalign-access is performed to external memory. Instruction fetch cycles may occur between these bus cycles as a result of pre-fetch queue status. Similarly, an instruction fetch occurs between the read cycle and write cycle because a bit operation instruction includes a read modify write operation.

The external bus locks due to a conflict between the fetch request for this instruction fetch and the DMA request that uses the external bus.

Moreover, the bus cycle executed between each bus cycle generated by a misalign-access or bit operation instruction is limited to an instruction fetch. The timing in which instruction fetch cycles occur during misalign-access is shown in Attachment 1.

Currently, with this problem the instruction fetch request that causes the lock is cleared if an interrupt request occurs while the external bus is locked and the bus conflict status is eliminated.

[Countermeasures]

Do not use SET1, NOT1, and CLR1 instructions or instructions accompanying misalign-access to the external bus when a program is stored in external memory and DMA transfer is performed using external memory.

**Note** A bit operation instruction will not be generated if optimization is performed with an NEC compiler (CA850 Ver.2.02) without using a bit field.

(5) Invalid output of DMAAK signal

[Description]

If an internal refresh request or external bus hold request occurs when a cycle operates under condition (B), as stated below, following a cycle operating under condition(A), the DMAAK signal responding to the DMA that is attempting to operate in an idle cycle (a one cycle period) which begins immediately before will be output abnormally (Figure 2). The cycle following the idle cycle that is input after a cycle in condition (A), however, becomes a refresh cycle or a external bus hold cycle because a refresh request or external bus hold request is given priority ever a DMA cycle in condition (B). A DMA cycle is not generated while the abnormally output DMAAK cycle is active (DMA transfer is not performed).

Condition (A):

- A read cycle (including an instruction fetch) to an external memory area for which an idle state insert is set with software.
- A fly-by transfer mode DMA cycle to external I/O from an external memory area for which an idle state insert is set with software.

Condition (B):

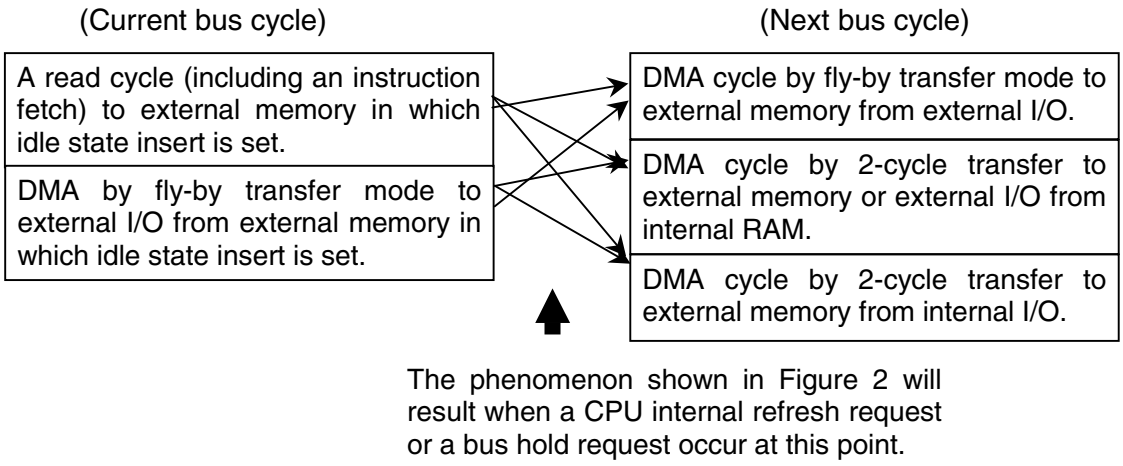
- A fly-by transfer mode DMA cycle from external I/O to external memory.
- A 2-cycle transfer mode DMA cycle from internal peripheral I/O to external memory.
- A 2-cycle transfer mode DMA cycle from internal RAM to external memory (including memory mapped I/O).

Non-applicable condition:

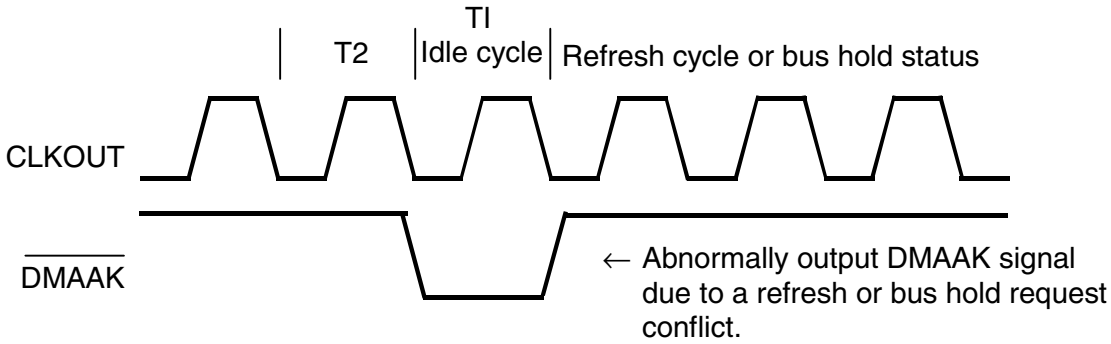
Any of the following conditions will render this problem non-applicable whether the DMA function is used or not.

- An idle state is not inserted with software.
- The internal refresh and external bus hold functions are not used.
- DMA fly-by transfer is not performed to external I/O.

**Note** This problem may affect applications in which handshaking is performed with the DMARQ and DMAAK signals and DMA transfer frequency is counted on the external I/O.




**<Figure 1>**




**<Figure 2>**

[Countermeasures]

It is possible to avoid this problem with any of the following methods.

- Do not insert an idle state with software.
- Do not use the internal refresh and external bus hold functions.
- Use the AND signal of the DMAAK output and IORD output that are generated by external circuitry instead of DMAAK output when abnormal DMAAK output due to this phenomenon becomes a problem with application systems or when the applicable DMA channel is used as a fixed channel from external I/O to external memory in fly-by transfer mode.
  (For applications for handshaking to an external DMA request.)
- Use DMA in combination with different modes those of the occurrence conditions.

(6) Problem related to multiplication instructions

[Description]

After a multiplication command is executed, an problem occurs when several commands that contain a write-back instruction (writing back to a register) follow. Before the results of the first multiplication command are stored in the register, the next command is fetched. If the execution of this command is cancelled by an interrupt then the results of first multiplication will be stored in the second multiplication command's destination register. This applies to programs that repeatedly execute multiplication instructions (mul, mulh, mulhi, mulu). The results of the multiplication process are not written back to the first multiplication command's destination register, and the previous data remains in the register.

This problem does not occur if instructions not accompanying a write-back operation are placed between multiplication instructions, even for a single instruction. Moreover, this problem will not occur even for instructions accompanying write-back operations if the following instructions are enclosed[Note 2]. Nor does this problem occur when the destination register of instructions accompanying a write-back operation is r0.

- Instructions accompanying write-back operations that result in problem conditions.
  Load instructions, arithmetic operation instructions, logic operation instructions, saturation operation instructions, LDSR, STSR, JARL
  (Refer to the chapter on pipelining in the V850E/MS1 user's manual (architecture edition) for details on the applicability of each instruction type and write-back operation.)

**Notes 1.** This problem exerts no actual influence in the cases given below.

- When the destination registers of the preceding and following multiplication instructions are the same.
- When the result of the preceding multiplication instruction is not used after the following multiplication instruction is executed.

**2.** Although the instructions below accompany write-back operations, they do not

result in problem conditions.
Division instructions (DIVH, DIV, DIVU), MOV imm32, TRAP, PREPARE, DISPOSE

An example of problem occurrence:

```
mulh    r0,    r2
mov     r0,    r5
mov     r0,    r6
mulh    r0,    r3   ← This instruction is canceled by an interrupt.
```

[Countermeasures]
In principal, this problem can be avoided by inserting a NOP instruction between two multiplication instructions.

NEC is preparing an automated patch tool for compiler-generated intermediate code (assembler source code) to support compiler-assisted software development. Please contact the development tool support center below for information on GHS manufactured compiler (CC850) patch tools. This problem will be dealt with in the NEC manufactured compiler (CA850) with an officially released version (Ver.2.02, scheduled for release at the end of September). In addition, there are a number of cases where this problem is applicable to the NEC manufactured real-time OS (RX850) and middleware. For the RX850, the applicable instruction pattern exists only during an interrupt prohibited period, and, therefore, only NMI is applicable.
However, this is not considered to be a problem because techniques such as reset in the NMI processing routine when the OS is in use are quite common. For middleware, please contact an NEC sales representative or distributor for individual consultations.

(7) A non-map break occurs when an internal I/O or an internal RAM is accessed with a ROMless product or during operation in ROMless mode
[Description]
A non-map break occurs when an internal I/O or an internal RAM is accessed during operation in ROMless mode.
[Countermeasures]
This bug can temporarily be avoided by mapping addresses 3F00000h to 3FFFFFFh to the “emulation RAM” or “target” using the memory mapping function of the in-circuit emulator.
However, the maximum emulation memory that can freely be set is 1 MB (contiguous space) after taking this countermeasure.

[Caution] The debugger (ID850) can forcibly set the non-map break to be invalid in the memory space 3F00000h to 3FFFFFFh. In this case, it is not necessary to take the above countermeasure.

Refer to the cautions on using the debugger for details.

(8) An illegal I/O break occurs when an internal I/O space is accessed by DMA

[Description]

An illegal I/O break may occur when an internal I/O space (3FFF000h to 3FFFFFFh) is accessed by DMA.

[Countermeasures]

The debugger (ID850) can forcibly set the non-map break to be invalid in the memory space 3FFF000h to 3FFFFFFh.

(9) Illegal write protect break in ROMless mode

[Description]

An illegal write protect break occurs when mapping an emulation memory to ROM and RAM and successively performing RAM write and ROM read (including fetch) in ROMless mode.

This bug does not occur in single-chip mode.

[Countermeasures]

Do not perform ROM mapping but perform RAM mapping in ROMless mode.

(10) Illegal internal RAM fetch when branching from external memory to the internal ROM

[Description]

When a branch from external memory to the internal RAM occurs, the fetch from the internal RAM is illegal.

[Countermeasures]

To branch from the external memory to the internal RAM, put an unconditional branch instruction to branch to the next instruction after the branch destination in the internal RAM.

(11) Coverage bug

[Description]

Coverage measurement is performed each time the program is executed (it is not possible to specify no coverage measurement).

[Countermeasures]

There are no countermeasures.

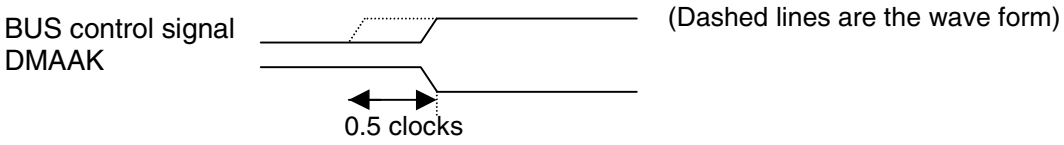
Please regard this as permanent restriction.

(12) Bug on DMAAK output timing

[Description]

Normally, a 0.5-clock logical width from bus control signals ($\overline{RD}$, $\overline{IORD}$, $\overline{UWR}$/$\overline{UCAS}$, $\overline{LWR}$/$\overline{LCAS}$, and $\overline{IOWR}$) ↑ to $\overline{DMMAK}$↓ is assured by the specification. However, the bus control signal output and the $\overline{DMAAK}$ output seem to change at almost the same timing because output delay time of the bus control signal is long.

In the same manner, the bus control signal and TC0 to TC3 seem to change at almost the same timing.




[Countermeasures]

The IE-703102-MC with control code D has been modified.

(13) Bug on CLKOUT relative specification

[Description]

Because CLKOUT is output relatively earlier, the timing relative to CLKOUT is out of specification.

(The signal delay time to CLKOUT↑ or ↓ seems to be too long)

Example $t_{DKA}$: Address and _CSn output delay time (to CLKOUT↓)

This bug applies to the relative timing between CLKOUT and signals (e.g. Address bus, data bus, RD/WR strobe), and does not apply to the relative timing between signals.

[Countermeasures]

The IE-703102-MC with control code B and later has been modified.

(14) Bug on port register read at output port

[Description]

If a port register is read during setting the output mode, the pin status is read instead of the port register value.

Target ports: P4, 5, 6, A, B

[Countermeasures]

There are no countermeasures.

Please regard this as permanent restriction.

(15) Bug on trace immediately after standby release

[Description]

The instruction to access PSC register is not traced if standby mode has been released by NMI input.

[Countermeasures]

There are no countermeasures.

Please regard this as permanent restriction.

(16) Illegal write protect break at write access is performed to the emulation RAM

[Description]

An illegal write protect break occurs when mapping a DRAM-specified space to the

emulation memory and performing write access using an I/O register.

[Countermeasures]

Do not perform emulation RAM mapping but perform target mapping for the DRAM-specified space using an I/O register.

(17) A write protect break does not occur even when write access to the emulation ROM is performed using a program stored in IRAM

[Description]

A write protect break does not occur even when write access to the emulation ROM is performed using a program stored in IRAM.

[Countermeasures]

The IE-703102-MC with control code C and later has been modified.

(18) Bug on flyby DMA when IDLE insertion, DMA request, and refresh conflict

[Description]

If a DMA request and refresh request in flyby transfer from external I/O to external memory conflict during a read cycle (including instruction fetch) reading an external memory space where an idle state is set to be inserted, data transfer by DMA is not performed, but the DMA byte count register, transfer source, and the destination address are updated.

At this time, the waveform temporarily looks as if the DMA cycle and refresh cycle are generated simultaneously.

[Countermeasures]

- Do not insert an idle state using software setting.
- Do not use the internal refresh function.
- Do not use flyby transfer mode for transfer from the external I/O to the external memory.

(this does not cause problem in flyby transfer from the external memory to the external I/O or two-cycle transfer)

The product with control code C and later has been modified.

(19) Restriction on STOP mode release

[Description]

If the emulator is used in PLL mode and the CESEL flag of the PSC register is 0, the program does not branch to the NMI handler address even if STOP mode is released by NMI input, but the PC is updated. (STOP mode can be released by NMI input.)

[Countermeasures]

a) Before releasing STOP mode by NMI input, use the emulator in PLL mode and write “1” to CESEL flag.

b) To perform debugging while writing “0” to CESEL flag, use IDLE mode (substitute for STOP mode), or use direct mode (PLL mode prohibited).

The IE-703102-MC-EM1 and IE-703102-MC-EM1-A with control code B and later have been modified.

(20) Restriction on temperature range

[Description]

Use the emulator at room temperature.

(21) Breaks and events may not be detected at branch instruction

[Description]

When branch instructions are successively performed, a break or an event may not be detected at the first branch instruction.

Example: Bcond $LOOP ← A break does not occur at this instruction.

JR $LABEL

[Countermeasures]

There are no countermeasures.

Please regard this as permanent restriction.

(22) Restriction on PC interface board

[Description]

The IE-70000-MC-SV3, IE-70000-CD-IF, and IE-70000-CD-IF-A cannot be used.

[Countermeasures]

The IE-70000-MC-SV3 can be used with the product with control code B and later, but the IE-70000-CD-IF, and IE-70000-CD-IF-A cannot be used with any product. This restriction is permanent.

(23) Restriction on operation at 40 MHz

[Description]

The following restrictions apply to emulation at 40 MHz.

- Insert two or more waits to access the target memory or emulation memory.
- When using the SC-144 (target extension probe), the target memory and emulation memory cannot be accessed even if two or more waits are inserted.

[Countermeasures]

There are no countermeasures.

(24) Connection to the IE-703100-MC

[Description]

When the IE-703100-MC main unit is used connected to the IE-703102-MC-EM1-A, emulation @ HVDD = 3.3 V is possible.

Use IE-703100-MC with control code E.

**To customers:**

If the IE-703100-MC requires repair when using the product at 3 V emulation, be sure to send the IE-703100-MC and IE-703102-MC-EM1-A connected unit to the repairing company.

# 4. CAUTIONS

(1) Target device product line-up.

- This in-circuit emulator is for use only with the V850E/MS1.

(2) Cautions on using emulation memory

- Be sure to insert a wait when operating at the following frequencies:
  - 25 MHz ≤ Operating frequency ≤ 33 MHz: Insert one or more waits.
  - 33 MHz < Operating frequency ≤ 40 MHz: Insert two or more waits.
- Bus size is fixed to 16 bits. It cannot be used with 8-bit bus.
- Unlike the external memories that can be used with the V850E/MS1 (e.g. SRAM, page, DRAM), the emulation memory can be accessed only by setting the MM register. Consequently, the settings for PMC8, PMC9, PMCX, that are required for the external memory expansion, cannot be debugged with the emulation memory.
- The emulation memory cannot be specified as the source or destination of DMA transfer regardless of the ROM/RAM attribute.

(3) NMI, HLDRQ, WAIT pins mask options

- Masking of the NMI, HLDRQ, and WAIT pins functions as a mask of the port pins even when these pins are used as port pins.

(4) Pin processing

- This in-circuit emulator uses the minimum pin processing, giving priority to compatibility with the device. Take care to avoid damage to the emulator by electrostatic discharge when not using the target.
- Pins processed inside the in-circuit emulator are listed below. Other pins are basically connected to the emulation device directly.

| | |
|---|---|
| CKSEL: | Pulled up by the internal HVDD with 100 Ω/pulled down with 33 kΩ (selectable by SW1) |
| X1: | Pulled up by the internal VDD with 5.1 kΩ |
| X2: | Open |
| MODE0: | Pulled down by GND with 33 kΩ |
| MODE1: | Pulled up by the internal HVDD with 5.1 kΩ |
| MODE2 to 3: | Open |
| RESET: | Pulled up by the internal HVDD with 5.1 kΩ |
| CVDD: | Open |
| CVSS: | GND |
| HVDD1 to 2: | Connected to the internal HVDD via relay |
| VDD1 to 2: | Connected to the internal VDD via relay |
| VSS1 to 4: | GND |

(5) Peripheral functions during break

If the operation of the CPU peripheral functions (e.g. timer, serial interface, A/D converter) provided for the emulator are enabled during program execution, they continue operation if the program is stopped with their operation enabled.

Consequently, the following phenomena may occur while a program is stopped.

<1> Program is stopped while the peripheral functions are enabled.

↓

<2> Because the peripheral functions are operating, interrupt request flags from the peripheral functions are reset while program is stopped.

↓

<3> The interrupt request is held until the program starts again.

↓

<4> The interrupt request that has been held occurs immediately after re-starting the program.

(6) Leak current from the emulator to the target

a. Pin control when the target power supply OFF and emulator power supply ON

A leak current may flow from the emulator to the target.

b. Pin control when the target power supply OFF and emulator power supply OFF

A small leak current may flow from the emulator to the target.

Note that a small leak current flows from the host machine to the target when PC interface cable is connected to the emulator and the power supply of the host machine is ON.

## Attachment 1

Please regard this as permanent restriction.

<Misalign-access>

Misalign-access occurs when the word border is not aligned (the two low order bits of the address are 0) in the case of word data, and when the half-word border is not aligned (the one low order bit of the address is 0) in the case of half-bit data.

<Timing in which instruction fetch cycle occurs during misalign-access>

- With external 16-bit bus width
  - Half-word access to address 2n+1.

    Two bus cycles occur: a byte access to address 2n+1 and a byte access to address 2n+2. An instruction fetch cycle occurs between these two bus cycles.
  - Word access to address 2n+1.

    Three bus cycles occur: a byte access to address 2n+1, a half-word access to address 2n+2, and a byte access to address 2n+3. In this case, an instruction fetch cycle occurs between bus cycles generated by the half-word access to address 2n+2 and the byte access to address 2n+3.

    **Note** An instruction fetch cycle does not occur between bus cycles generated by the byte access to address 2n+1 and the half-word access to address 2n+2.

  - Word access to address 2n+2

    Two bus cycles occur: a half-word access to address 2n+2 and a half-word access to address 2n+4. An instruction fetch cycle occurs between these two bus cycles.

- With external 8-bit bus width
  - Half-word access to address 2n+1

    Two bus cycles occur: a byte access to address 2n+1 and a byte access to address 2n+2. An instruction fetch cycle occurs between these two bus cycles.
    - Word access to address 2n+1.

    Four bus cycles occur: a byte access to address 2n+1, a byte access to address 2n+2, a byte access to address 2n+3, and a byte access to address 2n+4. An instruction fetch cycle occurs between bus cycles generated by the byte access to address 2n+3 and the byte access to address 2n+4.

    **Note** An instruction fetch cycle does not occur between bus cycles generated by the byte access to address 2n+1 and the byte access to address 2n+2 or between the bus cycles generated by the byte access to address 2n+2 and the byte access to address 2n+3.