[Material for Limited Distribution and Use]

CUSTOMER NOTIFICATION

SUD-T-4528-4

October 18, 2000

Ikutaro Okuda, Project Manager
Microcomputer Engineering Dept.
Solution Engineering Div.
NEC Electron Devices
NEC Corporation

CP(K), O

V850E/MA1 μPD70F3107 Flash Memory Model (Ver. 2, Ver.3) Restrictions

This document details usage restrictions confirmed to date related to versions 2 and 3 of the V850E/MA1 flash memory model (μPD70F3107).

1. Overview of Restrictions

Item	Description of Bugs	Ver.2.0/2.1	Ver.3.0/3.1	Ver.3.2
(1)	Bug related to EDO DRAM access	X	√	√
(2)	Bug in which DMA is started with an interrupt by the integrated peripheral I/O.	X	√	√
(3)	Bug related to the SLD instruction	X	√	√
(4)	Bug related to the DMA function	X	X	√

X: Not modified (Applicable), √: Modified

* The modifications from Ver.2.0 to 2.1 and Ver.3.0 and 3.1 are for improving the operation margin only; there are no functional differences.
* All the above bugs will be modified in Ver.3.2. A Ver. 3.2 sample is scheduled to be available by the end of December 2000.

2. Description of Restrictions

(1) Bug related to EDO DRAM access

EDO DRAM cannot be accessed.

[Countermeasure] There is no countermeasure.

(2) Bug in which DMA is started with an interrupt by the integrated peripheral I/O.

A DMA request is retained if an interrupt set to trigger the start of DMA transfer is generated while DMA transfer is prohibited (including abortion by NMI or forced termination by software) when DMA transfer is started by an interrupt from the integrated peripheral I/O. The retained request, however, cannot be cleared. As a result, there is a possibility that an interrupt set to trigger the start of DMA transfer will be generated while DMA transfer is prohibited. If you do not want to start DMA transfer with this interrupt at the stage where DMA transfer is enabled, execute the instructions on the Attachment. In particular, please bear this in mind for cases where one DMA channel is used with multiple applications (start factor is changed while in progress, etc.), DMA transfer is terminated in progress, or forcibly terminated.

[Countermeasure] Take the following procedure.

<1> Set the DMA trigger factor register (DTFRn) to 00H to prohibit a DMA request from the integrated peripheral I/O.

<2> To perform dummy DMA transfer, set the DMA source address register (DSAn) and the DMA destination address register (DDAn) to a region that does not influence the system.

<3> Set the DMA transfer count register (DBCn) to 0000H.

<4> Set the Enn bit and the STGn bit of the DMA channel control register (DCHCn) to 1, and perform dummy DMA transfer with the software trigger. When doing this, make

sure that the MLEn bit of the same register is cleared to 0. This will ensure that dummy DMA transfer is generated along with a DMA transfer completion interrupt (INTDMAn), and the Enn bit will automatically clear to 0.

Note 1: If you do not want to generate a DMA transfer completion interrupt through dummy DMA transfer, set the DMANKn bit of the interrupt control register (DMAICn) to 1 before generating dummy DMA transfer to mask the interrupt.

Note 2: If DMA transfer is reset by a DMA request from the integrated peripheral I/O after a DMA request is cleared, first set the DSAn, DDAn, and DBCn registers, then set the DTFRn register and the Enn bit of the DCHCn register.

(3) Bug related to the SLD instruction

In the following instruction sequences, the value loaded by the first LD or SLD instruction (marked <1> in the sequences below), is modified by the next instruction <2>. After executing the SLD instruction, marked <3>, the register used by the first LD/SLD instruction <1> is used again by the instruction marked <4>. The value used should be the modified value resulting from instruction <2>, but instead this bug causes the value initially loaded by instruction <1> to be used.

Instruction sequence type 1

(1) xxx (Instruction in Note 1)
(2) sld * , rX (Accesses internal RAM) …<1>
(3) yyy * , rX (Instruction in Note 2) …<2>
(4) sld * , rY (Accesses internal RAM) …<3>
(5) Instruction that uses value rX …<4>

Instruction sequence type 2

(1) ld/sld * , rX (Accesses internal RAM) …<1>
(2) xxx (One or more instructions) (Instruction in Note 3)
(3) yyy * , rX (Instruction in Note 2) …<2>
(4) sld * , rY (Accesses internal RAM) …<3>
(5) Instruction that uses value rX …<4>

Instruction sequence type 3

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

♦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.

♦2 : The asterisks above indicate any parameter

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

[Unaffected cases]

The bug does not occur if any of the following conditions are met. Even if none of these conditions are met, the bug will not occur if the search tool mentioned below does not find an instruction sequence that causes this bug (excluding sequence 3 above.)

- If SLD instructions accessing internal RAM are not used.
- If there are no instruction fetches from EDO DRAM for which the data access wait is 0, internal ROM, internal RAM, or SDRAM.

[Countermeasure]

Use one of the following countermeasures

- It is possible to prevent the problem at the instruction level by replacing SLD instructions with LD instructions.
  - The bug does not affect sequence 3 above if there are no instruction fetches from EDO DRAM for which the data access wait is 0, or SDRAM. A tool that searches for instruction sequences 1 and 2 is available for such cases.

- Options will be added to the GHS and NEC compilers to avoid this bug. The NEC compiler is expected to be available from April 12th. The GHS compiler is expected to be ready from April 7th.

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

Development Tool Support Center
Tel.: 044-548-7963
E-mail: toolsupport@saed.tmg.nec.co.jp

(4) Bug related to DMA function

If an NMI is input during a DMA transfer, the transfer operation is usually interrupted and the contents of the Enn bit of the DCHCn register are retained in the DDIS 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 bit of the DCHCn register 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).

[Unaffected cases]

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.)
- DMA is started using a software trigger only (setting the STGn bit of the DCHCn register to 1).
- The DMA restart function by the DRST register is not used.
- NMI input is not used, or an NMI is not input during a DMA transfer.

[Countermeasure]

This bug does not occur if an NMI is input after a DMA transfer that was started by a software trigger. Therefore, avoid the problem by taking one of the following countermeasures.

- When using single transfer mode, be sure to first perform a one-time dummy DMA transfer started by a software trigger (this should also be performed when restarting DMA after the occurrence of a terminal count). To start DMA by using an interrupt from the internal peripheral I/O, for example, start the first DMA using a software trigger (setting the STGn bit of the DCHC register to 1), and then immediately enable the interrupt that is set as the start trigger. From the second time onward, DMA is started by the start trigger interrupt.
  To start DMA by using the DMARQn signal, use the software trigger to start DMA transfer and activate the DMARQn signal.
  When using this countermeasure, bear in mind that the transfer source address and transfer destination address will be changed by the dummy transfer and the transfer count setting.

- If there is a DMA channel not used, perform a dummy transfer by a software trigger using that channel. In this case, set the transfer count as twice or more, and perform a DMA transfer by software trigger only once before performing a DMA transfer for the channel via which a single transfer will be performed. After the dummy transfer, do

not write any data to the DCHCn register of the channel that performs the dummy transfer. It causes no problem even if the Enn bit of DCHCn, which is performing the dummy transfer, is cleared by the NMI input (n = 0 to 3).