Concerned Products:	**Customer Notification**	Date: 26. Aug. 98
		NEC-Electronics (Europe) GmbH EAD -Technical Product Support
µPD30111 **(V_R-4111)**	**Bug Report**	Source Doc: SBB-T-9449, SBB-T-9449-1, SBB-T-9449-2; SBG-T-0627
		Author: Guido Hilker
Date of initial issue: 17. Feb. 98	Doc. No.: TPS-HE-B-6003	

Remarks:
In this document, products are classified in terms of their revision numbers (x.x), which are marked on the package to enable identification of samples of products being developed as shown below.
There are no revision number markings for mass-production products, but you can distinguish the revision by the manufacturing standard in the product control number. Currently, only Rev. 1.1 product is mass-produced and shipped and the manufacturing standard is either “E” or “K.” You cannot check the revision number using the manufacturing standard for samples.

NEC JAPAN	
D30111S1	<- Product name
VR4111 ES1.1	<- Nickname & revision No.
9746I2901	

(A) BUG LIST

Bug No.		**Outline**	**μPD30111S1**			
			R1.0	R1.1	R2.0	R3.0*
1		Exceptional priority level bug	💣	✋	✓	✓
2		MIPS16 mode transition bug (1)	💣	✋	✓	✓
3		MIPS16 mode transition bug (2)	💣	✋	✓	✓
4		Jump instruction bug in MIPS16 mode	💣	✋	✓	✓
5		18.432MHz oscillation circuit bug	✋	✓	✓	✓
6		GPIO output enable bug	💣	✋	✓	✓
7		MIR bug	💣	✋	✓	✓
8		GPIO pull-up/pull-down enable bug	💣	✋	✓	✓
9		Change in RTCRST# low duration after power-on	(1)	(1)	(1)	(1)
10		Key auto sop restriction	💣	💣	(1)	(1)
11		MIPS16 load/jump instruction sequence bug	💣	✋	✓	✓
12		MIPS16 branch instruction bug	💣	✋	✓	✓

✓:	No problem
✋:	Bug (will be corrected by next version upgrade)
💣:	Bug (restriction, not corrected by version upgrade)
*:	Scheduled
(1):	Note This item has been (or will be) incorporated into the specifications

Bug No.		Outline	μPD30111S1			
			R1.0	R1.1	R2.0	R3.0*
13		SIU parity setting change	(1)	(1)	(1)	(1)
14		GPIO[3:0] interrupt detection bug	💣	✋	✓	✓
15		Exceptions and ERET instruction contention bug	💣	💣	✋	✓
16		Index_Store_Tag_D instruction bug	💣	💣	✋	✓
17		Multiple exception bug	💣	💣	✋	✓
18		MIPS16 branch instruction bug	💣	💣	✋	✓
19		MIPS16 mode transition bug (3)	💣	💣	✋	✓
20		MIPS16 SHIFT instruction specification change	💣	💣	✋	✓
21		MIPS16 link register bug	💣	💣	✋	✓
22		MIPS16 mode transition bug (4)	💣	💣	✋	✓
23		MIPS16 Extend instruction bug	💣	💣	✋	✓

✓:	No problem
✋:	Bug (will be corrected by next version upgrade)
💣:	Bug (restriction, not corrected by version upgrade)
*:	Scheduled
(1):	Note This item has been (or will be) incorporated into the specifications

(B) BUG DESCRIPTION

Bug No.	Outline	Description
1	The priority level of the exception is erroneous when contention occurs between the CP0 interlock, an exception and an interrupt.	Description In the event that the CP0 interlock competes with the syscall exception, the break exception, the reserved instruction exception or the coprocessor disable exception and furthermore when an interrupt exception competes with the aforementioned exceptions, the interrupt exception is processed ahead of the others, thus causing the contents of the cause register to be in error. There are two factors that trigger the CP0 interlock: mtc0 rx, Status and mtc0 rx, Config. Example: mtc0 rx, Status # cp0 interlock factor syscall # cp0 interlock + interrupt Countermeasure until correction Immediately after the mtc0 instruction, do not place an instruction code that triggers the syscall instruction, the break instruction or the reserved instruction exception, nor any instruction that triggers the coprocessor disable exception. If the mtc0 instruction is in the branch delay slot, do not place an instruction code that triggers the syscall instruction, the break instruction and the reserved instruction exception on this branch destination instruction, nor an instruction that triggers the coprocessor disable exception.
2	Does not change to MIPS16 mode with the JALX instruction immediately after the CP0 interlock.	Description An attempt to change from the MIPSIII mode the MIPS16 mode with the jalx instruction immediately after an instruction that triggers the CP0 interlock results in causing the sequence of instructions from the branch destination to be executed as MIPS instructions without changing to the MIPS16 mode. There are two factors that trigger the CP0 interlock: mtc0 rx, Status and mtc0 rx. Config. Example: mtc0 rx, Status # cp0 interlock factor jalx m16 Jump to # MTPS16 mode any Inst. Note: This bug does not occur with the jr and jalr instructions. Countermeasure until correction Do not place the jalx instruction immediately after the mtc0 instruction. If the mtc0 instruction is in the branch delay slot, do not place the jalx instruction in the relevant branch destination instruction.

3	The instructions in the delay slot of the JALX instruction continue to be executed.	Description In the event that an instruction for generating the CP0 interlock in the delay slot of the jalx, jr and jalr instructions for changing to the MIPS16 mode, the instructions of the delay slot continue to be executed during execution of the MIPS16 instruction after the MIPS16 mode change. There are two factors that trigger the CP0 interlock: mtc0 rx, Status and mtc0 rx. Config. Example: jalx m16 Jump to # MTPS16 mode mtc0 rx, Status # cp0 interlock factor m16: any Inst. # mtc0 rx, Status instruction is executed at the same time any Inst. # Until exiting the MIPS16 mode, the mtc0 rx, Status instructions are also executed at the same time. Countermeasure until correction Do not place an instruction that generates the CP0 interlock in the delay slot of the jalx, jr and jalr instructions.
4	The instructions in the delay slot of the jump instruction are not executed in MIPS16 mode.	Description When an instruction cache error occurs in the delay slot of jr, jalr and jalx instructions during execution in MIPS16 mode, the instructions in the branch delay slot are not executed. Even when the instructions in the delay slot of the above-mentioned jump instruction are the uncached area, the instructions in the branch destination slot are not executed. Example: jr # MIPS16 mode any Inst. # Cache Miss or Uncache Not executed Countermeasure until correction Place nop in the delay slot of the jr, jalr and jalx instructions during execution in MIPS16 mode.
5	The output of the 18.432MHz oscillation circuit is not transmitted internally and SIU, A/D and PIU don't operate.	Description Because the output of the 18.432MHz oscillation circuit is not performed, the units (SIU, A/D, PIU) which use the 18.432MHz clock directly do not function at all. Because the clock input to PLL is not performed due to the same reason, the PLL runs automatically and the pipeline clock operates only at that frequency (around 40MHz). Countermeasure until correction No countermeasure is available.

6	When in 16bit bus mode, the output value of GPIO[29:26] cannot be set.	Description If IOS [29:26] of the GIUIOSELH register (0x0b00 0102) is set to 1 in 16bit bus mode, the corresponding GPIO (GPIO [29:26]) is set but the GPIO [29:26] output enable cannot be set for each bit of IOS [29:26] of the GIUIOSELH register (0x0b00 0102). All the IOS [29:26] settings must be of the same value and the IOS [31] setting must be adjusted as well. If different values are set, only a Low level will be output regardless of the PIOD [29:26] settings of GPIO [29:26] to GIUPIODH registers (0x0b00 0106). Cause The selection condition of GPIO[29:26] output value does not use the GIUIOSELH register's IOS[29:26] but mistakenly uses IOS[31](the selection condition bit of GPIO[31] output value). Countermeasure until correction When using GPIO[29:26] as an output pin, use all of GPIO[29:26] and GPIO[31] as the output pins. Specifically, always set the IOS[31] and IOS[29:26] bits of the GIUIOSELH register to the same value.
7	MIR mode (1.152Mbps and 0.576Mbps) cannot be used.	Description Cannot be used on the current VR4111's IrDA communication function, because the data transmission speed in MIR (1.152Mbps and 0.576Mbps) mode departs substantially from the standard. Countermeasure until correction No countermeasure is available. Only communications in FIR (4Mbps) mode can be used in FIR unit.
8	In hibernate mode, the internal pull-up/pull-down function of the GPIO[14:0] pin is disabled.	Description For the GPIO[14:0] pin, the corresponding bit of the GIUIOSELL register (0x0b00 0100) is set to 0 to set that pin as an input pin, and the corresponding bit of the GIUUSEUPDN register (0x0b00 02e0) is set to 1. This enables the internal pull-up or pull-down function of the GPIO pin. (The pull-up or pull-down function is enabled using the GIUTERMUPDN register (0x0b00 02e2).) According to the specifications, the above setting must be retained in hibernate mode. However, when the system enters hibernate mode, the contents of the GIUUSEUPDN register are initialized (set to 0), thus disabling the pull-up/pull-down function. Even when the system returns to full-speed mode, the pull-up/pull-down function remains disabled. Countermeasure until correction Set the GPIO[14:0] pin as an input pin. If the pull-up/pull-down function is required in hibernate mode, connect an external pull-up/pull-down resistor. If the pull-up/pull-down function is not required in hibernate mode, but is required in other operation modes, reset the pull-up/pull-down function when the system returns to full-speed mode from hibernate mode.

9	For an RTC reset immediately after power-on, a low input duration of at least two seconds is required for the RTCRST# signal.	Description According to the specifications, for an RTC reset immediately after power-on, a low input duration of at least 600 milliseconds is provided for the RTCRST# signal. During this period, a stable oscillation of 32.768 kHz should be provided for RTC. However, it has been shown that, for some products, the oscillation fails to stabilize during this period. Therefore, a low input duration of at least two seconds is required for the RTCRST# signal pin immediately after power-on. Countermeasure until correction This item has been incorporated into the specifications.
10	If the number of stops/repeats is set to 0, operation stops with on-time scanning only irrespective of the key input status.	Description According to the specification, when the ATSTP bit of the KIUSCANREP register (0x0b00 0190) is set to 1 (bit[1]), if the STPREP bit (bit[9:4]) of the same register is set to 0 and no key is pressed for 64 times consecutively, key scanning is supposed to stop. However, it has been found that in this product, operation stops after only a single scan whether a key is pressed or not. That is, the system judges that operation is completed even if no key is pressed 0 times. Countermeasure When using auto stop, do not set ALL0 in the STPREP bit of the KIUSCANREP register. This restriction shall be regarded as part of the specification. If you have any problem with this restriction, contact us as soon as possible.
11	In MIPS16 instruction mode, if a load instruction is followed by a jump instruction, the jump instruction cannot be executed normally	Description In MIPS16 instruction mode, when a load instruction (all load instructions such as lw and lwpc) or a transfer instruction (mflo, mfhi) is followed by a jump instruction (jal, jaix), if any of the following conditions is established, the jump instruction is not executed normally. Condition 1: When the value of bit[10:8] of the instruction code of a load or transfer instruction matches the value of bit[10:8] of the instruction code of the immediately following jump instruction. Condition 2: When the value of bit[10:8] of the instruction code of a load or transfer instruction matches the value of bit[7:5] of the instruction code of the immediately following jump instruction. Condition 3: When the value of bit[7:5] of the instruction code of a load or transfer instruction matches the value of bit[10:8] of the instruction code of the immediately following jump instruction. Condition 4: When the value of bit[7:5] of the instruction code of a load or transfer instruction matches the value of bit[7:5] of the instruction code of the immediately following jump instruction. Example: lwpc a0, 0x10(pc) 0xB404 [10110100 00000100] #bits 8 to 10 match. jalx 0x1C00 [00011100 00000000] □ Cannot be executed normally. 0x5585 Countermeasure until correction Do not insert a load instruction or transfer instruction (mflo, mfhi) immediately before a jump instruction.

12	In MIPS16 instruction mode, if an attempt is made to branch to a specific address by a branch instruction, the branch cannot be made normally	Description When a branch instruction (b / beqz / bnez / bteqz / btnez) of MIPS16 is executed, if the lower bit at the branch destination address (target address) is 0xE[0x1110], a branch is made to the target address + address 16. Example: 0x00 b LABEL Brach to #LABEL : : LABEL: 0x0E any inst. # Should branch here originally. : : 0x1E any inst. # Branches here. Countermeasure until correction Map the branch destination address (label) in MIPS16 mode onto the word boundary.
13	SIU's odd/even parities are set in reverse order.	Description: The STU parities are set using bit LCR4 of the STULC register (0x0C00 0003), but the bit setting described in the user's manual (1st ed.) is incorrect. Bit 4 LCR4 parity setting Incorrect: 1 for odd parity 0 for even parity Correct: 1 for even parity 0 for odd parity Countermeasure: Change the setting of bit LCR4 of the STULC register. This description will be included in the specifications

14	GPIO[3:0] interrupt detection and hold are affected by GPIO[12:9] interrupt type.	Description: When INTT[12:9] and INTT[3:0] of the GIUINTTYPL register (0x0b000110) that sets the interrupt request detection trigger and INTH[3:0] of the GIUINTHTSELL register (0x0b000118) that specify the interrupt request hold is set in one of the following combinations, an interrupt from the GPIO[3:0] terminal cannot be detected and held correctly. **GIUINTTYPL** — **GIUNTHTSELL** — **Operational flaws** INTT [12:9] — INTT[3:0] — INTH[3:0] 0 (Level) — 1 (Edge) — 1 (Hold) — When a GPIO[3:0] interrupt is detected, interrupt of both level and edge are detected and held. 1 (Edge) — 0 (Level) — 1 (Hold) — When a GPIO[3:0] interrupt is detected, the interrupt request cannot be held resulting in the same operation as when Through is set. The corresponding bits are as follows: INTT[12] -> GPIO[3] interrupt INTT[11] -> GPIO[2] interrupt INTT[10] -> GPIO[1] interrupt INTT[9] -> GPIO[0] interrupt Countermeasure until correction: When using GPIO[3:0] as an interrupt, set either of the following. 1. Set the GIUINYTYPL registers INTT[12] and INTT[3], INTT[11] and INTT[2], INTT[10] and INTT[1], and INTT[9] and INTT[0] respectively to the same interrupt detection method. 2. Set the relevant bit of the GIUINTTYPL register INTT[3:0] to 0 (Level) and the relevant bit of the GIUINTHTSELL register INTH[3:0] to 0 (Through).
15	Occurrence of an exception to either of the two instructions immediately before the ERET instruction clears the internal signal of the EXT_ERL bit, leading to faulty operation.	[Description] In case that an exception occurs at the DC stage of instruction two instructions before the ERET instruction or at the EX stage of the instruction immediately preceding the ERET instruction, the EXL or ERL bit of the status register will be set after occurrence of the exception. However, this will clear the EXL or ERL signal, which are internal signals of the CPU. Therefore, the status of the CPU cannot be controlled normally and subsequent operation cannot be guaranteed. Example 1: any inst. # DC stage exception occurs any inst. ERET Example 2: any inst. # EX stage exception occurs ERET Countermeasure until correction: Insert two nop instructions immediately before an ERET instruction. Nop nop ERET

16	The W bit cannot be set with the Index_Store_Tag_D instruction.	Description: The W bit of the data cache cannot be set to 1 by the Index_Store_Tag_D instruction. Countermeasure until correction: If the value of the W bit of the TagLo register used by the Index_Store_Tag_D instruction is 1, perform Read/Write in the same line after executing the Cache instruction or set the W bit after executing the Cache_Create_Dirty_Exclusive instruction.
17	Occurrence of an exception at the start address of the exception handler leads the contents of the cause register and the BadVAdder register to be illegal.	Description: In case that an exception occurs at the IF stage or RF stage of any instruction, and this exception causes another exception to occur at the RF stage of the first instruction of the exception handler that has been jumped to. The code of the latter exception cannot be written to the cause register (so the code of the former exception remains set). At this time, the contents of the BadVAddr register are destroyed. Exceptions causing this bug include the SYSCALL exception, the BREAK exception, the co-processor disable exception, the reserved instruction exception and the instruction bus error exception. Countermeasure: Except when a path error has occurred at the instruction fetch for the instruction before the exception handler, the bug can be avoided by applying the following restrictions. 1. By not locating the SYSCALL instruction and the NREAK instruction as the first instructions of the exception handler. 2. By not locating codes that cause the co-processor disable exception and the reserved instruction exception in the first instruction of the exception handler. * We are considering the possibility of applying the above as permanent restrictions. We apologise for any inconvenience this may cause, and if you have any problem in connection with the above restrictions, please contact our staff in charge of the products.
18	In the MIPS16 instruction mode, an Extend instruction immediately after a branch instruction results in a bug.	Description: The MIPS16 branch instruction does not support the branch delay slot. However, in case that the instruction immediately after the branch instruction is the EXTEND instruction and the branch condition matches, the EXTEND instruction is not killed but is executed in the same manner as the delay slot instruction, thus resulting in expanding the immediate value of the branch destination instruction. Countermeasure until correction: Insert a NOP immediately after an MIPS16 branch instruction. Alternately, do not use the EXTEND instruction.

19	If a stall competes with the transition to the MIPS16 instruction mode, a non-conformity occurs in the internal state and operations become unstable.	Description When changing from the 32 bit instruction (MIPS-I, II, III ISA) mode to the 16 bit instruction mode with the JALX, JALR or JR instructions and there is a LOAD, STORE or CACHE instruction just before the jump instruction, non-conformity occurs in the internal state of the CPU and the operation becomes unstable. Countermeasure until correction: Do not place a LOAD, STORE or CACHE instruction just before the JALX, JALR or JR instructions to change to the 16 bit instruction mode. If the JALX, JALR or JR instructions directly branch to change to the 16 bit instruction mode, do not place a LOAD, STORE or CACHE instruction at the delay slot of the origin of the branch of these instructions.
20	Target register and destination register are incorrect with the SHIFT SLL/SRA/SRL instructions from the MIPS16 instruction.	Description: If the MIPS16 SLL, SRA or SRL instructions from Rev. 2.0 or earlier are executed, the contents of the target register and destination register are reversed giving incorrect results. Correct register access is performed with Rev. 3.0 instructions. Note: This problem is caused by a specification change in the latest MIPS16, and shall be regarded as part of the specification. Countermeasure until correction: There are no countermeasures for using the device revision 2.0. We are developing some compiler products that will enable switching between both shift instruction specifications.
21	When an operating instruction which is the source of the content of the link register is executed at the delay slot of the JAL/JALR MIPS16 instruction, the results of the operations are incorrect.	Description: When an operating instruction that uses the content of the link register as the source is executed at the delay slot of the JAL/JALR MIPS16 instruction, the results of the operations are incorrect. Countermeasure until correction: Do not place an instruction that uses the contents of the link register at the delay slot of the JAL/JALR MIPS16 instruction.
22	A bug occurs when an operation is executed that could cause an overflow exception at the delay slot when changing to the MIPS16 instruction mode with the JALX/JALR/JR instructions.	Description: If an operation instruction (such as add or sub) is performed that could cause an overflow exception at the delay slot when changing to the MIPS16 instruction mode using JALX, JALR or JR instructions, and an operation is performed which will create overflow conditions by the MIPS16 instruction operations (such as when the operation results cause a digit overflow) after the change, an overflow exception will occur. With the normal specifications, an overflow exception does not occur with a MIPS16 operation instruction. Countermeasure until correction: Do not place instructions that can cause overflow exceptions in the JALX, JALR or JR instruction delay slots to change to the 16 bit instruction mode.

23	Reserved instruction exception does not occur when the next instruction code after the MIPS16 Extend instruction code is an undefined instruction.	Description: If the next instruction code is an undefined instruction code (ones where Extend is not possible) in the Extend instruction in the MIPS16 mode, the Extend instruction is handled as an NOP without the reserved instruction exception occurring. Countermeasure until correction: Do not place undefined instruction codes (ones where Extend is not possible) in the next instruction code for the Extend instruction in the MIPS16 mode.