CUSTOMER NOTIFICATION

SUD-DT-04-0193 (1/11)
April 23, 2004

Koji Nishibayashi, Senior System Integrator
Microcomputer Group
2nd Solutions Division
Solutions Operations Unit
NEC Electronics Corporation

CP(K), O

QB-V850ESSX2

(Control Code: A, B, C)

Operating Precautions

Be sure to read this document before using the product.

Notes on Using QB-V850ESSX2

This document describes restrictions applicable only to the emulator and restrictions that are planned for correction in the emulator.

Refer to the following documents for the restrictions in the target device.

- User's manual of target device
- Restrictions notification document for target device

Also refer to the user's manual of the emulator for cautions on using the emulator.

1. Product Version

Control Code[Note]	Remark
A	–
B	Version in which bug No.15 in control code A is corrected
C	Change of parts board and JP2 specifications (see **4.1 Changes in control code C**)

2. Product History

No.	Bugs and Changes/Additions to Specifications	Control Code[Note]		
		A	B	C
1	Bug in watchdog timer during break	Permanent restriction		
2	Bug in 16-bit timer M during break	Permanent restriction		
3	Bug in accessing UAnRX register during break	Permanent restriction		
4	Bug in accessing DR register during break	Permanent restriction		
5	Bug in accessing CBnRX register during	Permanent restriction		
6	Bug in accessing C0RGPT register during break	Permanent restriction		
7	Bug in accessing C0TGPT register during break	Permanent restriction		
8	Bug in accessing C0GNCTRL register during break	Permanent restriction		
9	Restriction on operating frequency	×	×	×
10	Bug related to DMA transfer forcible termination	×	×	×
11	Bug in program execution and DMA transfer in internal RAM	×	×	×
12	Restriction that emulator hangs up upon internal reset	×	×	×
13	Restriction that emulator hangs up while downloading data or setting software break	Permanent restriction		
14	Restriction on CLKOUT pin status in standby mode	×	×	×
15	Restriction on internal ROM misfetch	×	√	√
16	Restriction that data loss occurs when external RAM is connected	×	×	×

×: Applicable, √: Not applicable or already corrected

Note The "control code" is the second digit from the left in the 10-digit serial number in the warranty supplied with the product you purchased (if it has not been upgraded). If the product has been upgraded, a label indicating the new version is attached to the product and the x in V-UP LEVEL x on this label indicates the control code.

3. Details of Bugs and Added Specifications

No.1 Bug in watchdog timer during break

[Description]

When both the following conditions (a) and (b) are satisfied and a break occurs, the watchdog timer does not stop and a reset or a non-maskable interrupt occurs. If a reset occurs, the debugger hangs up.

(a) The main clock or subclock is selected as the source clock for the watchdog timer

(b) The Ring clock is stopped (RSTOP flag =1)

[Workaround]

Implement (a) or (b) below.

(a) Use the Ring clock as the source clock.

(b) Do not stop Ring clock oscillation.

Please regard this item as a permanent restriction.

No.2 Bug in 16-bit timer M during break

[Description]

When a break occurs while the following both (a) and (b) are satisfied, timer M does not stop even if the Peripheral Break function has been set to “Break”.

(a) INTWT, Ring clock (f_R/8), or subclock is selected as the source clock of timer M.

(b) The main clock is stopped by setting the MCK flag.

[Workaround]

Implement (a) or (b) below to stop timer M during a break using the Peripheral Break function.

(a) Use the main clock (fxx, fxx/2, fxx/4, fxx/64, or fxx/512) as the source clock.

(b) Do not stop main clock oscillation.

Please regard this item as a permanent restriction.

No.3 Bug in accessing UAnRX register during break

[Description]

An overrun error occurs under the following conditions (a) to (c).

(a) If a break occurs after reading the UART receive buffer register (UAnRX) and the UAnRX register is displayed in the I/O register window of the debugger, an overrun error occurs when UART reception is performed next time.

(b) If a software break occurs immediately after reading the UART receive buffer register (UAnRX), an overrun error occurs when UART reception is performed next time regardless of whether or not the UAnRX register is displayed in the I/O register window.

(c) If a DMA transfer from the UART receive buffer register (UAnRX) is performed during a break[Note], an overrun error occurs when UART reception is performed next time.

Note Including breaks by the RAM monitor function or DMM function. However, the real-time RAM monitor function does not cause this bug because it does not set breaks.

Remark An overrun error also occurs when UART receives data multiple times during a break. This is an emulator specification.

[Workaround]

(a) Do not display the UAnRX register in the I/O register window.

(b) Set a hardware break when setting a break immediately after reading the UAnRX register.

(c) There is no workaround.

Please regard items (a), (b), and (c) as permanent restrictions.

No.4 Bug in accessing DR register during break

[Description]

An overrun error occurs under the following conditions (a) and (b).

(a) If a software break occurs immediately after reading the IEBus data register (DR), an overrun error occurs when IEBus reception is performed next time regardless of whether or not the UAnRX register is displayed in the I/O register window.

(b) If a DMA transfer from the IEBus data register (DR) is performed during a break[Note], an overrun error occurs when IEBus reception is performed next time.

Note Including breaks by the RAM monitor function or DMM function. However, the real-time RAM monitor function does not cause this bug because it does not set breaks.

Remark An overrun error also occurs when UART receives data multiple times during a break. This is an emulator specification.

[Workaround]

(a) Set a hardware break when setting a break immediately after reading the DR register.

(b) There is no workaround.

Please regard items (a) and (b) as permanent restrictions.

No.5 Bug in accessing CBnRX register during break

[Description]

When the CSIBn receive data register (CBnRX) is read, it usually starts the next reception operation. Under the following conditions (a) and (b), however, the next reception operation is not started even if CBnRX is read.

(a) If a software break occurs immediately after reading the CSIBn receive data register (CBnRX).

(b) If a DMA transfer from the CSIBn receive data register (CBnRX) is performed during a break[Note].

As a result, communication stops or the DMA controller stops.

Note Including breaks by the RAM monitor function or DMM function. However, the real-time RAM monitor function does not cause this bug because it does not set breaks.

[Workaround]

(a) Set a hardware break when setting a break immediately after reading the CBnRX register.

(b) There is no workaround.

Please regard items (a) and (b) as permanent restrictions.

No.6 Bug in accessing C0RGPT register during break

[Description]

Under the following conditions (a) and (b), the read pointer (RGPT) that should be incremented is not incremented, and the same data as previously read is read.

(a) If a software break occurs immediately after reading the CAN0 module receive history list register (C0RGPT).

(b) If a DMA transfer from the CAN0 module receive history list register (C0RGPT) is performed during a break[Note].

Note Including breaks by the RAM monitor function or DMM function. However, the real-time RAM monitor function does not cause this bug because it does not set breaks.

[Workaround]

(a) Set a hardware break when setting a break immediately after reading the C0RGPT register.

(b) There is no workaround.

Please regard items (a) and (b) as permanent restrictions.

No.7 Bug in accessing C0TGPT register during break

[Description]

Under the following conditions (a) and (b), the read pointer (TGPT) that should be incremented is not incremented, and the same data as previously transmitted is transmitted.

(a) If a software break occurs immediately after reading the CAN0 module transmit history list register (C0TGPT).

(b) If a DMA transfer from the CAN0 module transmit history list register (C0TGPT) is performed during a break[Note].

Note Including breaks by the RAM monitor function or DMM function. However, the real-time RAM monitor function does not cause this bug because it does not set breaks.

[Workaround]

(a) Set a hardware break when setting a break immediately after reading the C0TGPT register.

(b) There is no workaround.

Please regard items (a) and (b) as permanent restrictions.

No.8 Bug in accessing C0GNCTRL register during break

[Description]

When a register access is performed in the following sequence, a forcible shutdown that should not take place normally may occur after the sequence is complete.

[Sequence for bug occurrence]

(1) The EFSD bit of the CAN0 module control register (C0GMCTRL) is set.

(2) The I/O register[Note] is accessed.

(3) The GOM bit of the CAN0 module control register (C0GMCTRL) is cleared.

Note I/O register access except for clearing the GOM bit of the C0GMCTRL register

Conditions under which a forcible shutdown takes place are shown below.

(a) If a break occurs immediately after the I/O register access in (2) occurs

(b) If a break by the RAM monitor function or DMM function occurs immediately after the I/O register access in (2) occurs

(c) Stepwise execution is performed for the I/O register access in (2)

[Workaround]

Be sure to set the EFSD bit and clear the GOM bit successively when executing a forcible shutdown.

Do not perform register access in the above sequence when not performing a forcible shutdown.

Please regard this item as a permanent restriction.

No.9 Restriction on operating frequency

[Description]

The maximum operating frequency is 20 MHz.

[Workaround]

There is no workaround. Use the emulator at a frequency of 20 MHz or lower.

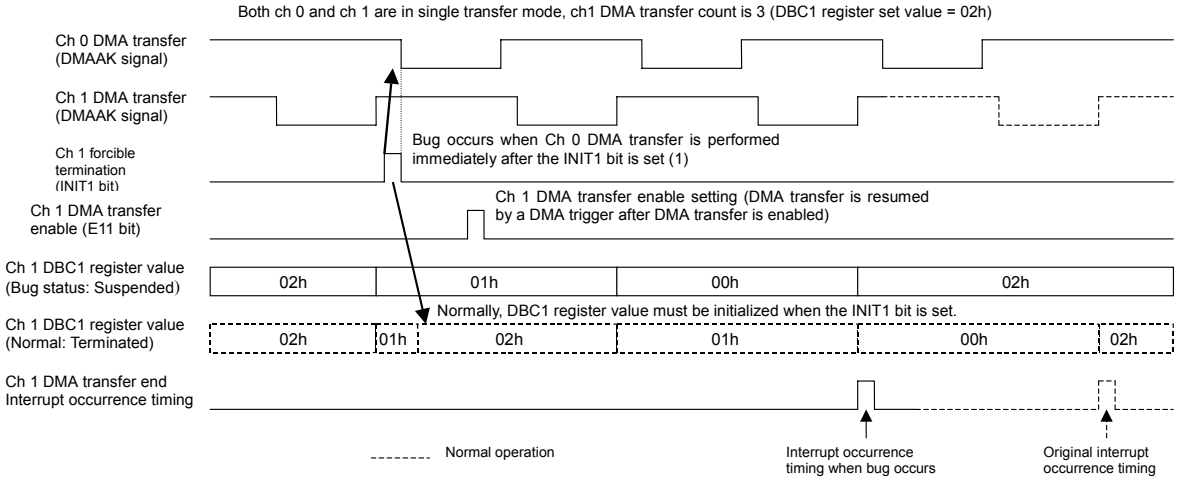
This restriction will be corrected in the next version.

No.10 Bug related to DMA transfer forcible termination

[Description]

When terminating a DMA transfer by setting the INITn bit of the DCHCn register, the transfer may not be terminated, but just suspended, even though the INITn bit is set (1). As a result, when the DMA transfer of a channel that should have been terminated is resumed, the DMA transfer will terminate after an unexpected number of transfers are completed and a DMA transfer completion interrupt may occur (n = 0 to 3). This bug occurs if a DMA transfer is executed immediately after a forcible termination is set (by setting the INITn bit) (see the figure below).

This bug does not depend on the number of transfer channels, transfer type (2-cycle or flyby), transfer target (between memory and memory, memory and I/O; including internal resources), transfer mode (single, single-step, or block), or trigger (external request, interrupt from internal peripheral I/O, or software), and can occur with any combination of the above elements that can be set under the specifications. In addition, another channel may affect the occurrence of this bug.



The following registers are buffer registers with a 2-stage FIFO configuration of master and slave. If these registers are overwritten during a DMA transfer or in the DMA-suspended status, the value is written to the master register, and reflected in the slave register when the DMA transfer of the overwritten channel is terminated.

The “initialization” in the above figure means that the contents of the master register are reflected in the slave register.

2-stage FIFO configuration registers (n = 0 to 3):

- DMA source address register (DSAnH, DSAnL)
- DMA destination address register (DDAnH, DDAnL)
- DMA transfer count register (DBCn)

[Workaround]

This bug can be avoided by implementing any of the following procedures using the software.

(1) Stop all the transfers from DMA channels temporarily

The following measure is effective if the following condition is satisfied.

- Except for the following workaround processing, the program does not assume that the TCn bit of the DCHCn register is 1. (Since the TCn bit of the DCHCn register is cleared (0) when it is read, execution of the following procedure (b) under <5> clears this bit.)

[Procedure to avoid bug]

<1> Disable interrupts (DI state).

<2> Read the DMA restart register (DRST) and transfer the ENn bit of each channel to a general-purpose register (value A).

<3> Write 00H to the DMA restart register (DRST) twice[Note].
By executing twice[Note], the DMA transfer is definitely stopped before proceeding to <4>.

<4> Set (1) the INITn bit of the DCHCn register of the channel that should be terminated forcibly.

<5> Perform the following operations for value A read in <2>. (Value B)

(a) Clear (0) the bit of the channel that should be terminated forcibly

(b) If the TCn and ENn bits of the channel that is not terminated forcibly are 1 (AND makes 1), clear (0) the bit of the channel.

<6> Write value B in <5> to the DRST register.

<7> Enable interrupts (EI state).

Remarks 1. Be sure to execute <5> to prevent the ENn bit from being set illegally for channels that are terminated normally during the period of <2> and<3>.

2. n = 0 to 3

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

(2) Repeat setting the INITn bit until the forcible DMA transfer termination is correctly performed (n = 0 to 3)

[Procedure to avoid bug]

<1> Copy the initial transfer count of the channel that should be terminated forcibly to a general-purpose register.

<2> Set (1) the INITn bit of the DCHCn register of the channel that should be terminated forcibly.

<3> Read the value of the DMA transfer count register (DBCn) of the channel that should be terminated forcibly and compare the value with the one copied in <1>. If the values do not match, repeat <2> and <3>.

Remarks 1. When the DBCn register is read in procedure <3>, the remaining transfer count will be read if the DMA is stopped due to this bug. If the forcible DMA termination is performed correctly, the initial transfer count will be read.

2. Note that it may take some time for forcible termination to take effect if this workaround is implemented in an application in which DMA transfers other than for channels subject to forcible termination are frequently performed.

This bug will be corrected in the next version.

No.11 Bug in program execution and DMA transfer in internal RAM

[Description]

When a DMA transfer for the internal RAM and a bit manipulation instruction (SET1, CLR1, or NOT1) allocated in the internal RAM or a data access instruction for a misaligned address are executed simultaneously, the CPU may deadlock due to conflict between the internal bus operations. At this time, only a reset can be acknowledged. (An NMI or interrupt cannot be acknowledged.)

[Workaround]

Implement any of the following workarounds.

- Do not perform a DMA transfer for the internal RAM when an instruction allocated in the internal RAM is being executed.
- Do not execute an instruction allocated in the internal RAM when a DMA transfer for the internal RAM is being performed.

This bug will be corrected in the next version.

No.12 Restriction that emulator hangs up upon internal reset

[Description]

The emulator may hang up when a reset is generated by watchdog timer 2 or the low-voltage detector (LVI).

[Workaround]

Stop watchdog timer 2 after reset. Do not emulate the low-voltage detector (LVI).

This restriction will be corrected in the next version.

No.13 Restriction that emulator hangs up while downloading data or setting software break

[Description]

The emulator may hang up if the WAIT pin or HLDRQ pin is at the active level while data is being downloaded to the internal ROM area or a software break is set to the internal ROM area.

[Workaround]

If the WAIT and HLDRQ pins are not used, mask the WAIT and HLDRQ pins using the pin mask function of the debugger.

If the WAIT and HLDRQ pins are used, do not make these pins active while data is being downloaded to the internal ROM area or a software break is set to the internal ROM area.

Please regard this item as a permanent restriction.

No.14 Restriction on CLKOUT pin status in standby mode

[Description]

Normally a low level is output in standby mode (software STOP or IDLE), but the operating clock is output as is if the mode is shifted from CLKOUT output mode to standby mode. This restriction is not applicable to the operation in HALT mode.

[Workaround]

If outputting CLKOUT in standby mode may cause problems, be sure to set the CLKOUT pin so that it outputs a low level before the mode is shifted from CLKOUT output mode to standby mode using the following procedure.

(1) Clear bit 1 of the PMCCM register to 0 (switches to I/O port)

(2) Clear bit 1 of the PMCM register to 0 (sets output port)

(3) Clear bit 1 of the PCM register to 0 (low-level output)

When this procedure is implemented, set bit 1 of the PMCCM register to 1 immediately after standby release (switching to CLKOUT output mode) to output CLKOUT.

This restriction will be corrected in the next version.

No.15 Restriction on internal ROM misfetch

[Description]

A misfetch may occur during program execution in the internal ROM (an unexpected instruction is executed).

[Workaround]

There is no workaround.

This restriction has been corrected in control code B or later.

No.16 Restriction that data loss occurs when external RAM is connected

[Description]

A write cycle to the external bus is generated when downloading data to the internal ROM is executed or a software break is set to the internal ROM. Therefore, if RAM is connected to the target system, data in the RAM may be lost.

[Workaround]

There is no workaround if the bug occurs as a result of downloading data to the internal ROM. However, it does not cause any problem if the internal RAM values are initialized by program execution after download (all the values in the RAM are overwritten), because the lost data is overwritten by normal values.

If the bug occurs as a result of setting a software break to the internal ROM, do not use a software break for the internal ROM space; use a hardware break instead.

This restriction will be corrected in the next version of the debugger.

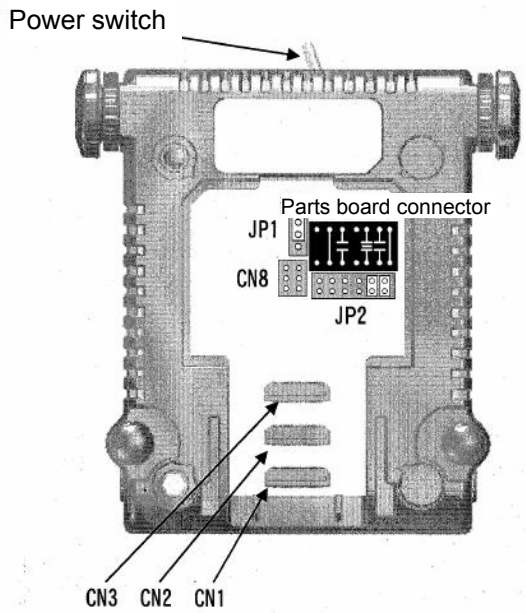
4. Cautions

4.1 Changes in control code C

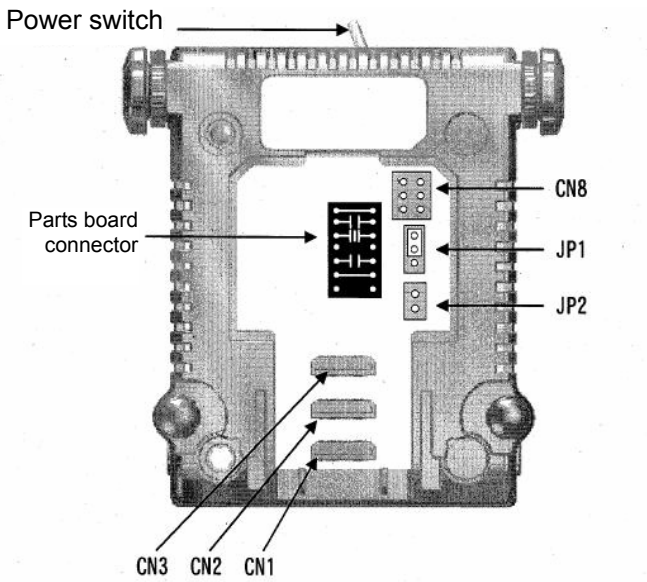
4.1.1 Change of parts layout

The parts layout has been modified in control code C or later as shown below.
This modification facilitates connection and disconnection of the parts board and jumpers.

Parts layout in control codes A and B



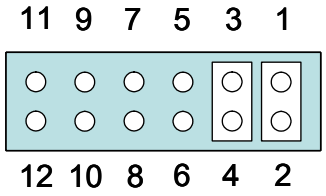
Parts layout in control code C



4.1.2 Modification of JP2 specification

The JP2 specification has been changed as shown below.
Use the default settings in control codes A, B, and C.

JP2 in control codes A and B



JP2 in control code C