

32-BIT MICROCONTROLLER **MB91460 SERIES**

FUNCTIONAL LIMITATION IRQ CLEARING OF LIN-USART AND RTC

2009-06-23



Revision History

Date	Issue
2009-06-23	V1.0, Initial Version

This document contains 9 pages.

Abbreviations:

FME	Fujitsu Microelectronics Europe GmbH
MCU	Microcontroller

Contents

REVISION HISTORY	2
CONTENTS	3
1 DESCRIPTION.....	5
2 CONDITIONS.....	5
3 AFFECTED DEVICES	5
4 AFFECTED MODULES	5
5 DETAILED EXPLANATION.....	6
5.1 Interrupt bits with additional output delay.....	6
5.2 Unaffected RTC ISR	7
5.3 Affected RTC ISR.....	7
5.3.1 No additional read.....	7
5.3.2 Single additional read.....	7
6 SOFTWARE ADAPTION OF ISR HANDLING	9
6.1 Software adaption to avoid the occurrence of the double ISR execution.....	9
6.1.1 Redesigned RTC ISR.....	9

Fujitsu does not bear any warranty in the case this handling note is not fully observed.

1 Description

An issue was discovered in the interrupt clear timing of the LIN-USART and RTC (real time clock) modules. Under certain conditions it can not be guaranteed that the signalled interrupt is already inactive when leaving the ISR (hence causing a double ISR execution).

The effect is comparable to the double ISR execution which is caused by the R-bus and D-bus write buffers (see application note **mcu-an-300025-e-v12-fr60_isr_double_execution**).

2 Conditions

The double ISR execution may occur if the following condition is met:

LIN-USART

- returning from the ISR before 1xCLKP cycle time has elapsed after clearing the interrupt cause bit

RTC (real time clock)

- returning from the ISR before 2xCLKP cycle time has elapsed after clearing the interrupt cause bit

3 Affected Devices

The following devices are affected:

- All MB91460 series devices (all date codes)

4 Affected Modules

This problem is affecting the interrupt generation consisting of:

- LIN-USART
- RTC (real time clock)

5 Detailed Explanation

5.1 Interrupt bits with additional output delay

Unlike other resources on the microcontroller the LIN-USART and RTC module output the interrupt request signal to CPU delayed by additional clock cycles (CLKP) on clearing the interrupt flags.

The following table shows the related interrupt bits:

Resource	Register	Bit	Delay [CLKP cycle]	Comment
LIN-USART	SSRx	RDRF	1	RDRF is cleared by reading RDRx.
	SCRx	CRE	1	Bit that clears error flag (PE,ORE,FRE)
	SMRx	UPCL or SRST	1	Reset bit of LIN-USART that clears all flags (TDRE,RDRF,LBD,PE,ORE,FRE)
RTC	WTCER	INT4	2	0.5 second interrupt cause bit
	WTCR	INT3	2	1day interrupt cause bit
	WTCR	INT2	2	1hour interrupt cause bit
	WTCR	INT1	2	1minute interrupt cause bit
	WTCR	INT0	2	1second interrupt cause bit

Due to this behaviour it can not be guaranteed that the signalled interrupt request is already inactive when leaving the ISR (i.e. additional clock cycles have elapsed when leaving the ISR).

The effect is comparable to the double ISR execution which is caused by the R-bus and D-bus write buffers (see application note **mcu-an-300025-e-v12-fr60_isr_double_execution**).

- **LIN-USART:** Double IRQ execution doesn't occur if the description of using single RB_SYNC is followed (the additional 1xCLKP cycle is covered by the method of preventing double IRQ execution caused by the write buffer).
- **RTC:** Double IRQ execution may occur even if the description of using single RB_SYNC is followed (the additional 2xCLKP cycles are not covered by the method of preventing double IRQ execution caused by the write buffer). I.e. for the RTC ISR a particular handling is necessary and explained in following sections.

Remark: the method of using RB_SYNC is described in the microcontroller application note **mcu-an-300025-e-v12-fr60_isr_double_execution**.

5.2 Unaffected RTC ISR

RTC ISR with two additional read accesses to the respective memory space after clearing the interrupt flag are not affected in this double executed interrupt service routine issue.

```

__interrupt void IRQHandler (void) {
    /* clear ext. interrupt flag */
    WTCR_INT0 = 0;

    /* interrupt service code */
    /* with read access to res. */
    /* memory */
    a = WTCR_INT0;
    b = WTBR0;
}

```

Unaffected RTC interrupt service routine

5.3 Affected RTC ISR

There are two cases which affects the RTC ISR into the double IRQ issue.

5.3.1 No additional read

The affected interrupt service routine as it can be found in common applications is shown in the figure below.

```

__interrupt void IRQHandler (void) {
    /* clear ext. interrupt flags */
    WTCR_INT0 = 0;

    /* no add. resource read */
    /* access within interrupt */
    /* service code */
    ...
}

```

Affected RTC interrupt service routine (no read)

5.3.2 Single additional read

The affected interrupt service routine as it can be found in common applications is shown in the figure below.

```

__interrupt void IRQHandler (void) {
    /* clear ext. interrupt flags */
    WTCR_INT0 = 0;

    /* interrupt service code */
    /* with single read access to */
    /* resource memory */
    a = WTCR_INT0;
}

```

Affected RTC interrupt service routine (single read)

6 Software adaption of ISR handling

6.1 Software adaption to avoid the occurrence of the double ISR execution

The following software adaption of the RTC ISR will fully resolve the issue of double IRQ execution and should be implemented wherever applicable.

6.1.1 Redesigned RTC ISR

The RTC interrupt service routine shown in the figure below inhibits the double execution by adding the macro call RB_SYNC:

ONCE to the end of the interrupt service routine if there is already a single read:

```

__interrupt void IRQHandler (void) {
    /* clear RTC interrupt flag */
    WTCR_INT0 = 0;

    /* interrupt service code */
    /* with single read access to */
    /* resource memory */
    a = WTCR_INT0;

    /* Synchronization with R-Bus ONCE*/
    RB_SYNC;
}

```

Redesigned RTC interrupt service routine (single sync)

TWICE to the end of the interrupt service routine if there is no read:

```

__interrupt void IRQHandler (void) {
    /* clear RTC interrupt flag */
    WTCR_INT0 = 0;

    /* interrupt service code */
    ...

    /* Synchronization with R-Bus TWICE*/
    RB_SYNC;
    RB_SYNC;
}

```

Redesigned RTC interrupt service routine (double sync)

That forces additional read accesses to the Resource memory space which waits until the interrupt clear instruction on RTC WTCR/WTCER registers is completed, the write buffer is empty and the additional clock cycles have elapsed.