# Product Update

## Errata to the eZ80L92 MPU

## Issues Related to eZ80L92 Microprocessor

This document highlights the issues and workarounds (if available) related to the eZ80L92 MPU. These errata are listed by date codes in Table 1 and Table 2.

## eZ80L92 MPU with Date Codes 0220 and Later

The errata listed in Table 1 highlights the issues and workarounds (if available) with package date codes 0220 and later. These issues are assembled in week 20 in the year 2002 and later.

**Table 1. Errata to eZ80L92 Device with Date Codes 0220 and Later**

| No. | Issue | Detailed Description |
|---|---|---|
| 1 | During DEBUG mode, the eZ80® CPU may accept external bus requests even when the BUSREQ/BUSACK feature is disabled. | When either On-Chip Instrumentation (OCI) or ZiLOG Debug Interface (ZDI) is enabled for debugging and BUSREQ/BUSACK is disabled, it is possible for an external device to gain access to the bus. This situation occurs when the debugger requests the eZ80® CPU to execute code. During this time period, the eZ80® CPU may accept an external BUSREQ. As a result, changes to the Program Counter can occur that makes it difficult for the debugger to return the eZ80L92 MPU to the correct program state.<br>**Workaround**<br>Do not allow external devices to assert BUSREQ while debugging.<br>BUSREQ can be tied High during debugging. |
| 2 | Configuring CS0 register settings prior to CS1 affects Intel bus mode Address Latch Enable (ALE) signal. | In Intel® bus mode, the control register settings for the chip select range (CS0) is accessed prior to accessing the chip select configured for Intel bus mode (CS1), affects the operation of the Intel bus mode ALE signal. For example, if CS0BMC is set to `02h` (the reset condition) and CS1BMC is set to `84h`, the pattern of the first ALE signal in a sequence is one cycle long and occurs in the bus mode transition cycle (also known as the extra cycle). If access continues in Intel bus mode, subsequent ALE cycles are generated accurately. The ALE cycles are also generated accurately when CS0BMC is set to `x0h` (or `!=x2h`). |
| 3 | Incorrect bus mode selection, Intel and Motorola bus modes. | In Intel and Motorola® bus modes, for certain devices that detect the bus cycle used, the chip select gets active and can cause the device to select the incorrect bus mode. |
| 4 | Motorola mode signal de-assertion. | In Motorola mode, the R/W signal de-asserts at the end of S6 instead of the end of S7. |
| 5 | The infrared encoder/ decoder (endec) receiver misses bits when configured for low-data rates and the incoming signals are only 1.6 µs. | The infrared endec samples the incoming IR pulses using the baud rate clock divided by 16. This sampling rate can be insufficient to capture the incoming pulses when they use a short-pulse format and low data rates. This short pulse, 1.6 µs, is within IrDA specifications. However, not all transmitters use this particular signalling format. When the external transmitter is sending $^{3}/_{16}$ IR pulses, the endec on the eZ80L92 MPU receives the data properly. |

**Table 1. Errata to eZ80L92 Device with Date Codes 0220 and Later (Continued)**

| No. | Issue | Detailed Description |
|---|---|---|
| 6 | The real time clock (RTC) consumes excess current when the eZ80L92 is not in SLEEP mode. | When the eZ80L92 MPU is not in SLEEP mode, the system clock drives the real time clock's control and data registers. As a result, excess current is consumed through the RTC_$V_{DD}$ pin, that supplies power to the portion of the clock tree which drives the RTC registers. This current consumption is a function of operating frequency. Typical values are 500 µA at 20 MHz and 1 mA at 50 MHz.<br>**Workaround**<br>To prevent excess current consumption, design with a battery-charging circuit for the RTC_$V_{DD}$ pin. As a result, the charge will remain on your RTC battery when power is supplied to the chip. |
| 7 | A pulse on the SCL line when the $I^2C$ bus is idle and the SDA line is held High causes the $I^2C$ to lock. | A pulse on the SCL line prior to a START condition or after a STOP condition causes the $I^2C$ bus to lock. This situation occurs, regardless of the $I^2C$ control register ENAB settings (I2C_CTL). If this situation occurs, an $I^2C$ software reset does not unlock the $I^2C$ bus.<br>**Workarounds**<br>(1) To prevent a lock from occurring, it is possible to completely disable the $I^2C$ block prior to any bus activity using the clock peripheral Power-Down Register 1 (CLK_PPD1). Disable the $I^2C$ block before setting ENAB in the $I^2C$ control register.<br>(2) If a lock occurs, after the SCL line is released, another device on the $I^2C$ bus can issue a STOP by pulsing the SDA line. As a result, the $I^2C$ should unlock. Another option is to initiate an overall SYSTEM RESET of the eZ80F91 device to reset the $I^2C$ block. |
| 8 | RTC count errors of seconds, minutes, and hours can occur during eZ80L92 MPU power-up. | The eZ80L92 MPU contains a private test mode register powered by $V_{DD}$. The test mode enables fast RTC counting (for production test) and is synchronously reset by the system clock. The test register is not tied down during reset. If this register power-up in a state that enables test mode, the RTC counters will start incrementing in fast mode until the register is reset by the first toggles of the system clock.<br>**Workaround**<br>During eZ80L92 MPU power-up, gate off the RTC clock source and hold external RESET active (for at least three system clock periods) after $V_{DD}$ ramps up to 3.3 V and the system clock source is stable. |
| 9 | GPIO edge trigger interrupt mapping error. | For edge triggered interrupts (Mode 6 and Mode 9), erroneous logic dependencies for the interrupt clearing logic exist on all port pins within each of the specific ports. To achieve proper interrupt clearing behavior for a particular port pin its *mirror pin* must be programmed in a similar manner. This affects how the designer utilizes GPIO alternate function pins with GPIO interrupt modalities of those port pins.<br>The definition *mirrored pin* refers to any PORT where for Port X, pin 0 is mirrored to pin 7, pin 1 is mirrored to pin 6, pin 2 is mirrored to pin 5, pin 3 is mirrored to pin 4.<br>**Note:** X is defined as Port B, C, or D.<br>(continued) |

**Table 1. Errata to eZ80L92 Device with Date Codes 0220 and Later (Continued)**

| No. | Issue | Detailed Description |
|---|---|---|
| | | (continued from previous page)<br>For example, if PB0 is programmed as an edge triggered interrupt, the logic dependency to clear the interrupt by writing to PB0_DR and protecting the actual PB0_DR register value from change comes from the *mirror pin* PB7 logic. This is an errata problem which causes erratic behavior problems.<br>In the above example, the problem is that PB0_DR itself can be altered and might change the mode of operation for the port pin PB0. To correctly set up the logic dependencies, the *mirrored pin* must be placed in the same mode as its counterpart. As the functionally of these port pins need to be *mirrored* in order to correct the logic dependency, the alternate function assignments of these ports would not work correctly.<br><br>To use the SPI alternate function modality (for example, SPI alternate function pins PB2, PB3, PB6, and PB7) you will not be able to use the *mirror* port pins PB5, PB4, PB1, and PB0 for Mode 6 and Mode 9 interrupt and vice versa.<br>Any Port pin configured with Mode 6 or Mode 9 (an edge triggered interrupt) exhibits this behavior and affects the alternate function modality. The mirror mapping affects all Ports, specifically within the respective port pin pairs 0 and 7, 1 and 6, 2 and 5, and 3 and 4.<br>**Note:** This design flaw in no way affects the IVECT address for the GPIO interrupts.<br>**Workaround**<br>Below is an example setup:<br>PB0 Input, falling edge interrupt<br>PB1 Input, dual edge interrupt<br>PB2 Input, falling edge interrupt<br>PB3 Input, falling or rising edge interrupt, depending on hardware configuration<br>PB4 Input (not used)<br>PB5 Input, falling edge interrupt<br>PB6 Input, dual edge interrupt<br>PB7 Input, falling edge interrupt – This could be Rising, Dual, or even left floating (OPEN connection)<br>Using the above example, if PB0 is Input, falling edge interrupt, then PB7 must be a separate input, EDGE MODE interrupt. In this example, PB7 is setup as a falling edge interrupt the same way as PB0.<br>(continued) |

**Table 1. Errata to eZ80L92 Device with Date Codes 0220 and Later (Continued)**

| No. | Issue | Detailed Description |
|---|---|---|
| | | (continued from previous page)<br>This additional separate input interrupt signal connected to PB7 could be rising, falling, dual, or you can leave it unconnected (OPEN) as well. You must not use PB7 for GPIO I/O or LEVEL sensitive interrupts. This same thought process is applicable to all Port bits pairs, specifically bits 0 and 7, 1 and 6, 2 and 5, and 3 and 4. |
| 10 | The UART is continually interrupting; the user cannot clear the interrupt. | The root cause of this issue has been duplicated with certain signal conditions after which a software instruction was executed to clear the receive FIFO. The signals involved were from bit 0 of the ISR, and the trigger counter with RXFIFO enabled and RXINTERRUPT disabled.<br>**Workarounds**<br>To prevent this error condition from occurring, you can perform one of the following two actions:<br>(1)Do not enable the transmit or receive FIFO if it is not required. The Receive FIFO was the initial problem; however, both the TX and RX FIFOs are affected. Set bit 0 (FIFOEN) of the UART0_FCTL (0x0C2h) or UART1_FCTL (0x0D2h) registers to a value of zero.<br>(2)If you are using either the transmit or receive FIFOs, mask off the following two bit locations to avoid changing the default bit value of zero. If bit 0 (FIFOEN) of the UART0_FCTL (0x0C2h) or UART1_FCTL (0x0D2h) registers is set to 1, then mask off bit 1 (CLRRxF) and bit 2 (CLRTxF) so that any Write accesses to the UART0_FCTL (0x0C2h) or UART1_FCTL (0x0D2h) registers will not alter this default zero value.<br>(3)To correct this error condition when the UART Rx interrupt occurs but there is no Rx data detected, the user can clear the condition in software by putting the UART in loopback mode and then transmitting a single character. The following is an example of this workaround: |

```
//====================workaround====================
/*
 * Check for 'stuck' Fifo
 */
if( (Iir == SD_IIR_RX_INT) && ((Lsr & LSR_DR) == ) )
{
    UINT32 Mcr;

    /*
     * To clear this condition, put the UART in loopback
     * mode and send a character
     */
    Mcr = BSP_RD32 ( pUART->Base|UART_REG_MCTL );
    BSP_WR32( pUart->Base|UART_REG_MCTL, Mcr | MCTL_LOOP);
    BSP_WR32( pUart->Base|UART_REG_THR, 'Z' );
    /*
     * Wait for the character to hit the Rx fifo
     */
```

(continued)

**Table 1. Errata to eZ80L92 Device with Date Codes 0220 and Later (Continued)**

| No. | Issue | Detailed Description |
|---|---|---|
| | | (continued from previous page) |

```
        Lsr = BSP_RD32( pUart->Base | UART_REG_LSR );
        while( !(Lsr & LSR_DR) )
        {
          Lsr = BSP_RD32( pUart->Base | UART_REG_LSR );
        }
        /*
         * Ignore any data trapped in the Rx fifo
         */
        while( Lsr & LSR_DR )
        {
          Lsr = BSP_RD32( pUart->Base | UART_REG_RBR );
          Lsr = BSP_RD32( pUart->Base | UART_REG_LSR );
        }

        /*
         * Return to normal operation
         */
        BSP_WR32( pUart->Base|UART_REG_MCTL, Mcr | MCTL_LOOP);

        /*
         * Record this event
         */
        StuckRxCount++;
    }

//==================normalhandling====================
```

## eZ80L92 MPU with Date Codes 0140–0220

The errata listed in Table 2 highlights the issues and workarounds (if available) eZ80L92 with package date codes 0140–0220. The issues are assembled after week 40 in the year 2001 and during the first 20 weeks of the year 2002.

**Table 2. Errata to eZ80L92 Device with Date Codes 0140–0220**

| No. | Issue | Detailed Description |
|---|---|---|
| 1 | While using OCI in DEBUG mode, false software break points can occur. | When OCI is enabled, false software break points can occur. The debugger breaks whenever a 7Fh value appears on the data bus—even if it is not an opcode fetch. This error only affects operation using OCI during DEBUG mode. Normal operation and debug via the ZDI are not affected. For more information on OCI operation, refer to the *First Silicon Solutions, Inc*. |
| 2 | During DEBUG mode using OCI, execution of a JR instruction can generate incorrect trace frames when running from Program Memory with WAIT states. | During DEBUG mode using OCI, execution of an unconditional Jump Relative (JR) instruction can generate incorrect trace frames when running from Program Memory with WAIT states. Normal operation and debug via the ZDI interface are not affected. For more information on OCI operation, refer to the *First Silicon Solutions, Inc*. |
| 3 | Z80 BUS MODE $\overline{RD}$ signal assertion states. | Z80 BUS MODE $\overline{RD}$ signal assertion occurs during state T1 rather than state T2, as indicated in the *eZ80L92 specification (PS0130)*. |
| 4 | Data bus contention is possible during successive READ and WRITE memory operations in either eZ80® or Z80® bus modes. | While operating in either eZ80® or Z80 bus modes, contention on the data bus is possible when a WRITE operation immediately follows a READ operation. The interval between the de-assertion of the $\overline{RD}$ signal and the eZ80L92 driving the data bus may not be sufficient for the external memory device to release the data bus. The Intel® and Motorola® bus modes can be used to avoid this bus contention issue. |
| 5 | During DEBUG mode, the eZ80® CPU may accept external bus requests even when the BUSREQ/BUSACK feature is disabled. | When either OCI or ZDI is enabled for debugging and BUSREQ/BUSACK is disabled, it is possible for an external device to gain access to the bus. This situation occurs when the debugger requests the eZ80® CPU to execute code. During this time period, the eZ80® CPU may accept an external BUSREQ. As a result, changes to the Program Counter can occur that make it difficult for the debugger to return the eZ80L92 to the correct program state. |
| 6 | Configuring CS0 register settings prior to CS1 affects Intel bus mode ALE signal. | In Intel bus mode, the control register settings for the chip select range (CS0) that is accessed previous to accessing the chip select configured for Intel bus mode (CS1) affects the operation of the Intel bus mode ALE signal. For example, if CS0BMC is set to 02h (reset condition) and CS1BMC is set to 84h, the pattern of the first ALE signal in a sequence is one cycle long and occurs in the bus mode transition cycle (also known as the extra cycle). If access continues in Intel bus mode, subsequent ALE cycles are generated correctly. If CS0BMC is set to x0h (or != x2h), the ALE is generated correctly. |

**Table 2. Errata to eZ80L92 Device with Date Codes 0140–0220 (Continued)**

| No. | Issue | Detailed Description |
|---|---|---|
| 7 | Incorrect bus mode selection, Intel® and Motorola® bus modes. | In Intel® and Motorola® bus modes, for certain devices that detect the bus cycle that is used, the chip select can cause the device to select the incorrect bus mode. |
| 8 | Motorola® mode signal de-assertion. | In Motorola® mode, the R/W signal de-asserts at the end of S6 instead of at the end of S7. |
| 9 | A pulse on the SCL line while the $I^2C$ bus is idle and the SDA line is held High causes the $I^2C$ to lock. | A pulse on the SCL line prior to a START condition or after a STOP condition causes the $I^2C$ bus to lock. This situation occurs regardless of the $I^2C$ control register ENAB settings (I2C_CTL). If this situation occurs, an $I^2C$ Software Reset does not unlock the $I^2C$ bus.<br>**Workarounds**<br>(1) To prevent a lock from occurring, completely disable the $I^2C$ block prior to any bus activity using the Clock Peripheral Power-Down Register 1 (CLK_PPD1). Disable the $I^2C$ block before setting ENAB in the $I^2C$ Control register.<br>(2) If a lock occurs, after the SCL line is released, another device on the $I^2C$ bus can issue a STOP by pulsing the SDA line. As a result, the $I^2C$ should unlock. Another option is to initiate an overall SYSTEM RESET of the eZ80F91 device to reset the $I^2C$ block. |
| 10 | GPIO edge trigger interrupt mapping error. | For edge triggered interrupts (Mode 6 and Mode 9), erroneous logic dependencies for the interrupt clearing logic exist on all port pins within each of the specific ports. To achieve proper interrupt clearing behavior for a particular port pin its *mirror pin* must be programmed in a similar manner. This affects how the designer utilizes GPIO alternate function pins with GPIO interrupt modalities of those port pins.<br>The definition *mirrored pin* refers to any PORT where for Port X, pin 0 is mirrored to pin 7, pin 1 is mirrored to pin 6, pin 2 is mirrored to pin 5, pin 3 is mirrored to pin 4.<br>**Note**: X is defined as Port B, C, or D.<br>For example, if PB0 is programmed as an edge triggered interrupt, the logic dependency to clear the interrupt by writing to PB0_DR and protecting the actual PB0_DR register value from change comes from the *mirror pin* PB7 logic. This is an errata problem which causes erratic behavior problems.<br>In the above example, the problem is that PB0_DR itself can be altered and might change the mode of operation for the port pin PB0. To correctly set up the logic dependencies, the *mirrored pin* must be placed in the same mode as its counterpart. As the functionally of these port pins need to be *mirrored* in order to correct the logic dependency, the alternate function assignments of these ports would not work correctly.<br>To use the SPI alternate function modality (for example, SPI alternate function pins PB2, PB3, PB6, and PB7) you will not be able to use the mirror port pins PB5, PB4, PB1, and PB0 for Mode 6 and Mode 9 interrupt and vice versa.<br>(continued) |

**Table 2. Errata to eZ80L92 Device with Date Codes 0140–0220 (Continued)**

| No. | Issue | Detailed Description |
|---|---|---|
| | | (continued from previous page)<br>Any Port pin configured with Mode 6 or Mode 9 (an edge triggered interrupt) exhibits this behavior and affects the alternate function modality. The mirror mapping affects all Ports, specifically within the respective port pin pairs 0 and 7, 1 and 6, 2 and 5, and 3 and 4.<br>**Note**: This design flaw in no way affects the IVECT address for the GPIO interrupts.<br>**Workaround**<br>Below is an example setup:<br>PB0 Input, falling edge interrupt<br>PB1 Input, dual edge interrupt<br>PB2 Input, falling edge interrupt<br>PB3 Input, falling or rising edge interrupt, depending on hardware configuration<br>PB4 Input (not used)<br>PB5 Input, falling edge interrupt<br>PB6 Input, dual edge interrupt<br>PB7 Input, falling edge interrupt – This could be Rising, Dual, or even left floating (OPEN connection)<br>Using the above example, if PB0 is Input, falling edge interrupt, then PB7 must be a separate input, EDGE MODE interrupt. In this example, PB7 is setup as a falling edge interrupt the same way as PB0.<br>This additional separate input interrupt signal connected to PB7 could be rising, falling, dual, or you can leave it unconnected (OPEN) as well. You must not use PB7 for GPIO I/O or LEVEL sensitive interrupts. This same thought process is applicable to all Port bits pairs, specifically bits 0 and 7, 1 and 6, 2 and 5, and 3 and 4. |

**Table 2. Errata to eZ80L92 Device with Date Codes 0140–0220 (Continued)**

| No. | Issue | Detailed Description |
|---|---|---|
| 11 | The UART is continually interrupting; the user cannot clear the interrupt. | The root cause of this issue has been duplicated with certain signal conditions after which a software instruction was executed to clear the receive FIFO. The signals involved were from bit 0 of the ISR, and the trigger counter with RXFIFO enabled and RXINTERRUPT disabled.<br><br>**Workarounds**<br><br>To prevent this error condition from occurring, you can perform one of the following two actions:<br><br>(1) Do not enable the transmit or receive FIFO if it is not required. The Receive FIFO was the initial problem; however, both the TX and RX FIFOs are affected. Set bit 0 (FIFOEN) of the UART0_FCTL (0x0C2h) or UART1_FCTL (0x0D2h) registers to a value of zero.<br><br>(2) If you are using either the transmit or receive FIFOs, mask off the following two bit locations to avoid changing the default bit value of zero. If bit 0 (FIFOEN) of the UART0_FCTL (0x0C2h) or UART1_FCTL (0x0D2h) registers is set to 1, then mask off bit 1 (CLRRxF) and bit 2 (CLRTxF) so that any Write accesses to the UART0_FCTL (0x0C2h) or UART1_FCTL (0x0D2h) registers will not alter this default zero value.<br><br>(3) To correct this error condition when the UART Rx interrupt occurs but there is no Rx data detected, the user can clear the condition in software by putting the UART in loopback mode and then transmitting a single character. The following is an example of this workaround:<br><br>`//=====================workaround=====================`<br>`/*`<br>`* Check for 'stuck' Fifo`<br>`*/`<br>`if( (Iir == SD_IIR_RX_INT) && ((Lsr & LSR_DR) == ) )`<br>`{`<br>`UINT32 Mcr;`<br><br>`/*`<br>`* To clear this condition, put the Uart in loopback`<br>`* mode and send a character`<br>`*/`<br>`Mcr = BSP_RD32 ( pUART->Base\|UART_REG_MCTL );`<br>`BSP_WR32( pUart->Base\|UART_REG_MCTL, Mcr \| MCTL_LOOP);`<br>`BSP_WR32( pUart->Base\|UART_REG_THR, 'Z' );`<br><br>`/*`<br>`* Wait for the character to hit the Rx fifo`<br>`*/`<br>`Lsr = BSP_RD32( pUart->Base \| UART_REG_LSR );`<br>`while( !(Lsr & LSR_DR) )`<br>`{`<br>`Lsr = BSP_RD32( pUart->Base \| UART_REG_LSR );`<br>`}`<br>`/*`<br>`* Ignore any data trapped in the Rx fifo`<br>`*/`<br><br>(continued) |

**Table 2. Errata to eZ80L92 Device with Date Codes 0140–0220 (Continued)**

| No. | Issue | Detailed Description |
|---|---|---|
| | | (continued from previous page) |

```
        while( Lsr & LSR_DR )
        {
          Lsr = BSP_RD32( pUart->Base | UART_REG_RBR );
          Lsr = BSP_RD32( pUart->Base | UART_REG_LSR );
        }
        /*
         * Return to normal operation
         */

        BSP_WR32( pUart->Base|UART_REG_MCTL, Mcr | MCTL_LOOP);

        /*
         * Record this event
         */
        StuckRxCount++;
    }

    //==================normalhandling==================
```

**Warning:** DO NOT USE IN LIFE SUPPORT

### LIFE SUPPORT POLICY

ZiLOG'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS PRIOR WRITTEN APPROVAL OF THE PRESIDENT AND GENERAL COUNSEL OF ZiLOG CORPORATION.

### As used herein

Life support devices or systems are devices which (a) are intended for surgical implant into the body, or (b) support or sustain life and whose failure to perform when properly used in accordance with instructions for use provided in the labeling can be reasonably expected to result in a significant injury to the user. A critical component is any component in a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system or to affect its safety or effectiveness.

### Document Disclaimer