Report on Texas Instruments' LM3S6965 Silicon Bug in UART

Report by Garry Anderson
for Robust Firmware Ltd
28th October 2014

I have just found an error in TI's LM3S6965 Silicon for the UART1 Peripheral (and it may exist on UART0 and UART2 as well), but since TI has abandoned support for this part, they will do nothing to fix the silicon. So, each user must be aware of the bug and implement the appropriate work-around.

This **Report** explains what the error is, and when it causes a problem.

I discovered the error when my application occasionally failed to start after Watchdog Timeout (WDT) Reset. The error occurs infrequently, like about 1 in 10 times, and its frequency is not repeatable making it hard to capture. After many attempts, I was able to capture exactly what was going wrong, and verify my findings by implementing my work-around.

The error manifests itself as a BUSY bit in the Flag Register when the UART is not busy, like just after WDT Reset. After WDT Reset, the BUSY bit should be zero, which it is most of the time, however, occasionally the BUSY bit is set to ONE (1) after WDT Reset. The busy bit does not clear itself if the UART's clocks are not running. Of course, this is an error since the Flag Register specification states that BUSY is Zero (0) on Reset (UARTFR, Register 3, page 435, DataSheet DS-LM3S6965-7787), and the infrequent occurrence of the BUSY bit indicates that it has something to do with a Restart sequence error. Perhaps the "Reset" bits in the Data Sheet only applies to Power-On Reset and not WDT Reset, although I don't think that is the case. WDT Reset should behave the same as Power-On Reset:

Register 3: UART Flag (UARTFR), offset 0x018

The **UARTFR** register is the flag register. After reset, the TXFF, RXFF, and BUSY bits are 0, and TXFE and RXFE bits are 1.

UART Flag (UARTFR)
UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
Offset 0x018
Type RO, reset 0x0000.0090

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								TXFE	RXFF	TXFF	RXFE	BUSY	reserved		
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0

The error first manifested itself when the UARTConfigSetExpClk() function did not return, thereby stalling my application from starting. This function is supposed to set the

Report on Texas Instruments' LM3S6965 Silicon Bug in UART

UART clocks, but before setting the clocks, this function calls the UARTDisable function. Further investigation showed that the UARTDisable function was stalling while waiting for not BUSY in the Flag Register. Of course, if BUSY is set, it will not clear if the clocks are not running, thus the UARTConfigSetExpClk() function SHOULD NOT call UARTDisable before the UART's clocks are running with this bug in the silicon.

One work-around solution is to edit the UART.C file in the DRIVERLIB library folder, and comment out the code which calls UARTDisable from the **UARTConfigSetExpClk()** function (around line 276), as follows:

```
void UARTConfigSetExpClk(unsigned long ulBase, unsigned long ulUARTClk,
                         unsigned long ulBaud, unsigned long ulConfig)
{
    unsigned long ulDiv;

    //
    // Check the arguments.
    //
    ASSERT(UARTBaseValid(ulBase));
    ASSERT(ulBaud != 0);
    ASSERT(ulUARTClk >= (ulBaud * UART_CLK_DIVIDER));

    //
    // Stop the UART.
    //
    // BUG FIX - UARTDisable may not return if clocks are not running:
    // UARTDisable(ulBase);


    //
    // Is the required baud rate greater than the maximum rate supported
    // without the use of high speed mode?
    //
    if((ulBaud * 16) > ulUARTClk)
    {
        //
```

Another work-around solution is to edit the UART.C file in the DRIVERLIB library folder, and comment-out the code which waits for BUSY to be clear in the **UARTDisable()** Function (around line 451):

```
void UARTDisable(unsigned long ulBase)
{
    //
    // Check the arguments.
    //
    ASSERT(UARTBaseValid(ulBase));

    //
    // Wait for end of TX.
```

```
    //
// OCCASIONAL HANG-UP AFTER WDOG POR SEQUENCE
//    while(HWREG(ulBase + UART_O_FR) & UART_FR_BUSY)
//    {
//    }

    //
    // Disable the FIFO.
    //
    HWREG(ulBase + UART_O_LCRH) &= ~(UART_LCRH_FEN);

    //
    // Disable the UART.
    //
    HWREG(ulBase + UART_O_CTL) &= ~(UART_CTL_UARTEN | UART_CTL_TXE |
                                    UART_CTL_RXE);
}
```

The UARTDisable function works correctly if the hardware works correctly and the UART's clocks are running, and BUSY is clear as it should be, but, when BUSY is stuck on, then this function will never return.

Once an application has made a decision to call UARTDisable, there is no reason to even wait for the TX to become not busy, because any loss of such characters in the transmitter is irrelevant.

I hope this Report helps other developers who have noticed that their application occasionally stalls during a re-start sequence, like from WDT.

Hopefully, if Texas Instruments is responsible, it will issue an ERRATA addressing this bug and presenting the work-around for other developers in the community.