April 17, 2012

Si4x3x Errata (rev. B)

Errata Status Summary

Errata #	Title	Impact	Status
1	Register modification required for frequency operation from 240–320 MHz and 480–640 MHz.	Minor	Work around is available.
2	Incorrect nIRQ signal operation after shutdown (SDN) or initial power up.	Minor	Work around is available
3	Current draw may be higher than expected when in SLEEP mode when the Low Battery Detection (LBD) function is ENABLED.	Major	Multiple software workarounds are available.

Impact Definition: Each erratum is marked with an impact, as defined below:

- Minor: Workaround exists.
- Major: Errata that do not conform to the data sheet or standard.
- Information: The device behavior is acceptable the data sheet will be changed to match the device behavior.

Errata Details

1. **Description**: Register modifications are required for operation in the frequency bands between 240–320 MHz and 480–640 MHz with a temperature above 60 °C.

   Impact: In extremely rare cases, the synthesizer will not lock in these bands when the device is operated at a temperature above 60 °C. The software workaround is only required for these frequency and temperature ranges.

   Workaround: Write 03h to register 59h and 02h to register 5Ah.

   Resolution: Software workaround with register modification.

2. **Description**: Incorrect nIRQ signal operation after shutdown (SDN) or initial power up.

   Impacts: In a small percentage of cases after Shutdown (SDN) or initial power up the nIRQ signal can be low during the Power-on-Reset (POR) period. Typically the nIRQ signal will be high during this period and will exhibit a high to low transition at the expiration of the POR period. The interrupt status registers in 03h and 04h will report the correct status of the POR and the nIRQ function is normal after the initial POR period.

   Workaround: The nIRQ line should not be monitored for POR after SDN or initial power up. The POR signal is available by default on GPIO0 and GPIO1 and should be monitored as an alternative to nIRQ for POR. A second potential workaround is also available by running a timer on the microcontroller after SDN for 26 ms and then reading the interrupt status registers in 03h and 04h to check for POR and chip ready (XTAL start-up/ready). After the initial interrupt is cleared the operation of the nIRQ pin will be normal.

   Resolution: Will be fixed in the next revision.

3. **Description:** This issue **ONLY** affects SLEEP mode when the low battery detection (LBD) function is ENABLED (enlbd = 1 in Register 07h). Note that the LBD function is DISABLED by default.

   If LBD is ENABLED, approximately 5% of devices will enter into a state that draws more current than expected in SLEEP mode. Such devices will draw an average current of approximately 350 µA in SLEEP mode versus the expected 1 µA. These devices will also report an inaccurate battery voltage level in SLEEP mode.

   The cause of the issue has been verified to be the improper reset of the LBD circuitry when in SLEEP mode.

   We strongly recommend all customers affected by this issue to implement one of the software workarounds described below.

   Impact: Affected devices will draw more current than expected in SLEEP mode which will reduce the battery life of battery-backed products.

Workaround: If the low battery detection (LBD) function is not required during SLEEP mode, this issue can be resolved by ensuring that LBD is DISABLED (enlbd = 0 in Register 07h).

Alternatively, we have verified two software workarounds that allow the battery level to be checked during SLEEP mode with low current consumption. Complete details and reference code are provided in the attached Appendix. A summary of each workaround is included below.

Workaround A: This workaround utilizes the Low Duty Cycle Mode function of the device to periodically check the battery level. The wake-up timer (WUT) period must be set in conjunction with the LDC mode on time to 0.5 s for this workaround to function properly. It does not require any intervention by an external MCU. The average current when using this workaround is 2.5 µA.

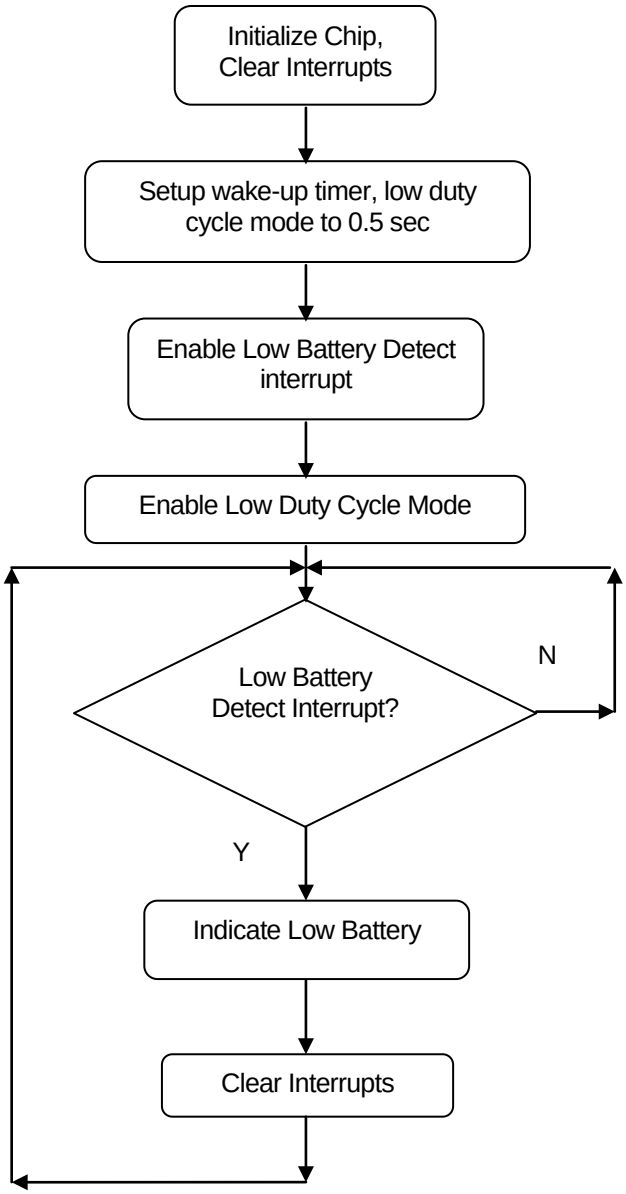
Workaround B: This workaround requires an external MCU to wake up the device and check battery levels periodically. The average current when using this workaround is dependent on the duty cycle of the wake up interval. If the interval is set to 500 s, the average current in this mode will be 1 µA.

Resolution: Implementation of the workarounds described in this document will enable the battery level to be checked with low current consumption.

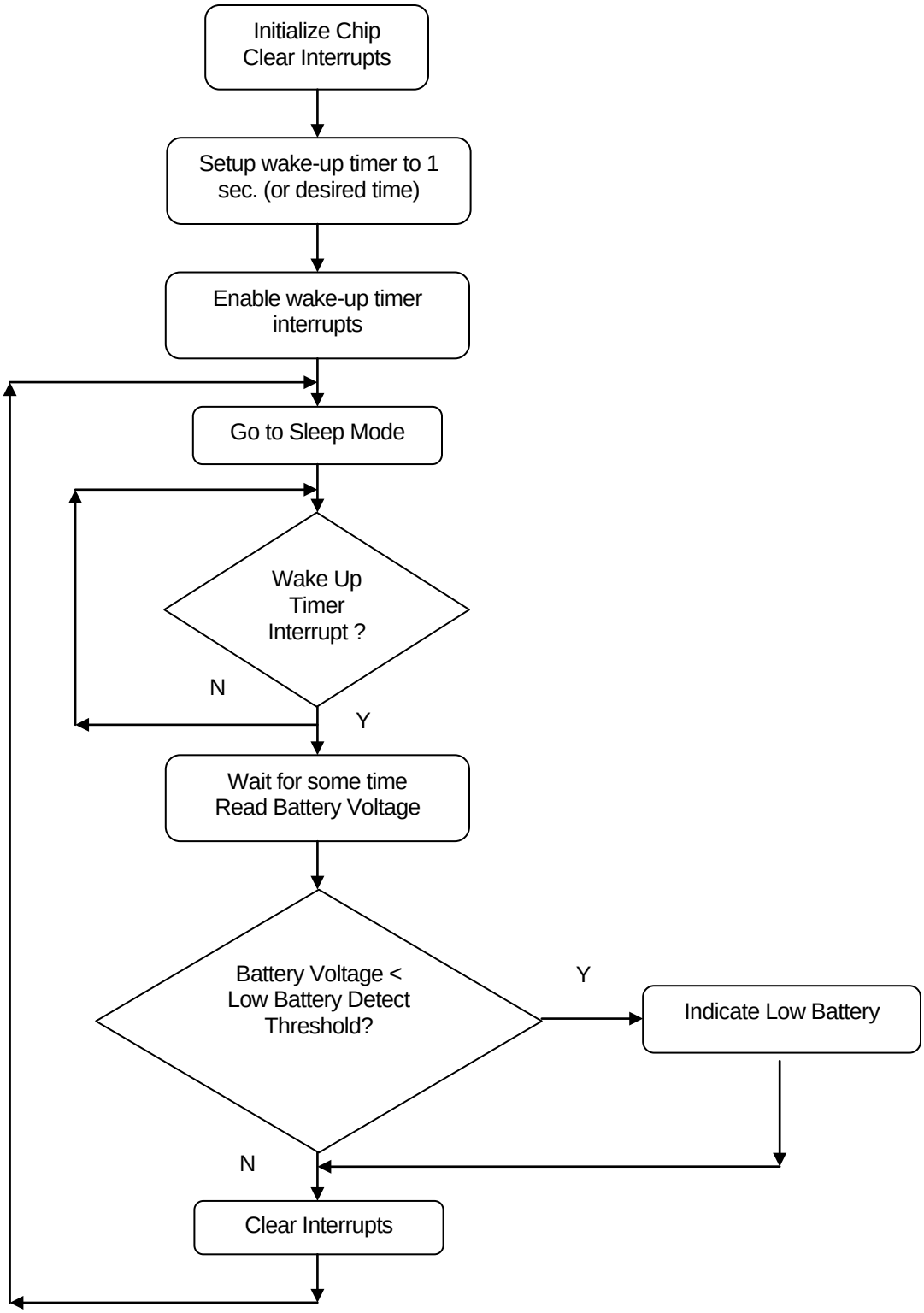
The low battery detection (LBD) function is no longer recommended for use in SLEEP mode and will be removed from the next revision of the Si4x3x data sheet.

Appendix

Workaround A: This Low Battery Detection using Low Duty Cycle Mode (automatic LBD workaround).



Workaround B: Low Battery Detection Using Wake-Up Timer.



Sample Code

Workaround A: Low Battery Detection using Low Duty Cycle Mode

```
//Function PROTOTYPES
//MCU initialization
void MCU_Init(void);
//SPI functions
void SpiWriteRegister (U8, U8);
U8 SpiReadRegister (U8);
//                                          GLOBAL VARIABLE
xdata U16 wWaitCnt;
void main(void)
{
U8 ItStatus1,ItStatus2,Battery_Voltage_Level,LBDT;
U16 delay;
U8 temp8;
//Initialize the MCU:
//                          - set IO ports for the Software Development board
//                          - set MCU clock source
//                          - initialize the SPI port
//                          - turn off LEDs
MCU_Init();
//Initialize the Si4x3x ISM chip
//read interrupt status registers to clear the interrupt flags and release NIRQ pin
ItStatus1 = SpiReadRegister(0x03);          //read the Interrupt Status1 register
ItStatus2 = SpiReadRegister(0x04);          //read the Interrupt Status2 register
for (temp8=0;temp8<15;temp8++)
{
```

```
for(delay=0;delay<10000;delay++);
}
//SW reset
SpiWriteRegister(0x07, 0x80);       //write 0x80 to the Operating & Function Control1 register
//wait for POR interrupt from the radio (while the nIRQ pin is high)
while ( NIRQ == 1);
//read interrupt status registers to clear the interrupt flags and release NIRQ pin
ItStatus1 = SpiReadRegister(0x03);          //read the Interrupt Status1 register
ItStatus2 = SpiReadRegister(0x04);          //read the Interrupt Status2 register
//wait for chip ready interrupt from the radio (while the nIRQ pin is high)
while ( NIRQ == 1);
//read interrupt status registers to clear the interrupt flags and release NIRQ pin
ItStatus1 = SpiReadRegister(0x03);          //read the Interrupt Status1 register
ItStatus2 = SpiReadRegister(0x04);          //read the Interrupt Status2 register
// Setup Wake Up Timer to wake up every 1/2 second
SpiWriteRegister(0x14, 0x00);
SpiWriteRegister(0x15, 0x10);
SpiWriteRegister(0x16, 0x00);
// Setup Low Duty Cycle ON Period
SpiWriteRegister(0x19, 0x01);
// Set LBDT (Low Battery Detect) Threshold
SpiWriteRegister(0x1A, 0x14);
// Store LBDT for comparison with Battery Voltage Register
LBDT = SpiReadRegister(0x1A);
//Write Interrupt Enable Registers,
SpiWriteRegister(0x05, 0x00);               //write 0x00 to the Interrupt Enable 1 register
SpiWriteRegister(0x06, 0x04);               //write 0x04 to the Interrupt Enable 2 register
// Delay
```

```
for(delay=0;delay<10000;delay++);
// Read Battery Voltage
Battery_Voltage_Level = SpiReadRegister(0x1B);
// Set Test Bus
#ifdef DEBUG
// GPIO0=LBDComp, GPIO1=wake-up, GPIO2 = Active_State
SpiWriteRegister(0x0B, 0x0D);
SpiWriteRegister(0x0C, 0x0B);
SpiWriteRegister(0x0D, 0x15);
SpiWriteRegister(0x50, 0x1C);
SpiWriteRegister(0x51, 0x01);
#else
// GPIOs set to GND
SpiWriteRegister(0x0B, 0x1F);
SpiWriteRegister(0x0C, 0x1F);
SpiWriteRegister(0x0D, 0x1F);
#endif
#ifdef ANTENNA_DIVERSITY
 SpiWriteRegister(0x0C, 0x17);
//write 0x17 to the GPIO1 Configuration(set the Antenna 1 Switch used for antenna diversity )
SpiWriteRegister(0x0D, 0x18);
//write 0x18 to the GPIO2 Configuration(set the Antenna 2 Switch used for antenna diversity )
#endif
#ifdef ONE_SMA_WITH_RF_SWITCH
SpiWriteRegister(0x0C, 0x12);      //write 0x12 to the GPIO1 Configuration(set the TX state)
SpiWriteRegister(0x0D, 0x15);      //write 0x15 to the GPIO2 Configuration(set the RX state)
#endif
// Enable Low Duty Cycle Mode
```

```
SpiWriteRegister(0x08, 0x04);
// Enable Wake Up Timer and Low Batter Detect
SpiWriteRegister(0x07, 0x60);
//MAIN Loop//
while(1)
{
while ( NIRQ == 1);
//read interrupt status registers to clear the interrupt flags and release NIRQ pin
ItStatus1 = SpiReadRegister(0x03);          //read the Interrupt Status1 register
ItStatus2 = SpiReadRegister(0x04);          //read the Interrupt Status2 register
if ((ItStatus2 & 0x04) == 0x04 )            // Low Battery Detect Interrupt Occured
{
LED2 = 1;
Battery_Voltage_Level = SpiReadRegister(0x1B);
Battery_Voltage_Level = 0x00;
// Disable Low Battery Detect Measurement
SpiWriteRegister(0x07, 0x20);
// Delay
for (temp8=0;temp8<255;temp8++)
{
for(delay=0;delay<10000;delay++);
}
// Enable Low Battery Detect Measurement
SpiWriteRegister(0x07, 0x60);
LED2 = 0;
}
}
}
```

Workaround B: Low Battery Detection Using Wake-Up Timer

```
//**Function PROTOTYPES**
//MCU initialization
void MCU_Init(void);
//SPI functions
void SpiWriteRegister (U8, U8);
U8 SpiReadRegister (U8);
//GLOBAL VARIABLE
xdata U16 wWaitCnt;
void main(void)
{
U8 ItStatus1,ItStatus2,Battery_Voltage_Level,LBDT;
U16 delay;
U8 temp8;
//Initialize the MCU:
//                    - set IO ports for the Software Development board
//                    - set MCU clock source
//                    - initialize the SPI port
//                    - turn off LEDs
MCU_Init();

//                              Initialize the Si4x3x ISM chip
//read interrupt status registers to clear the interrupt flags and release NIRQ pin
ItStatus1 = SpiReadRegister(0x03);          //read the Interrupt Status1 register
ItStatus2 = SpiReadRegister(0x04);          //read the Interrupt Status2 register
for (temp8=0;temp8<15;temp8++)
{
                        for(delay=0;delay<10000;delay++);
```

```
}
//SW reset
 SpiWriteRegister(0x07, 0x80);      //write 0x80 to the Operating & Function Control1 register
//wait for POR interrupt from the radio (while the nIRQ pin is high)
while ( NIRQ == 1);
//read interrupt status registers to clear the interrupt flags and release NIRQ pin
ItStatus1 = SpiReadRegister(0x03);          //read the Interrupt Status1 register
ItStatus2 = SpiReadRegister(0x04);          //read the Interrupt Status2 register
//wait for chip ready interrupt from the radio (while the nIRQ pin is high)
while ( NIRQ == 1);
//read interrupt status registers to clear the interrupt flags and release NIRQ pin
ItStatus1 = SpiReadRegister(0x03);          //read the Interrupt Status1 register
ItStatus2 = SpiReadRegister(0x04);          //read the Interrupt Status2 register
// Setup Wake Up Timer       : This should be set based on application requirements and battery
// characteristics.Ideally, wake up should be done as minimal as possible.
SpiWriteRegister(0x14, 0x00);
SpiWriteRegister(0x15, 0x20);
SpiWriteRegister(0x16, 0x00);
// Set LBDT (Low Battery Detect) Threshold
SpiWriteRegister(0x1A, 0x14);
// Store LBDT for comparison with Battery Voltage Register
LBDT = SpiReadRegister(0x1A);
// Set Test Bus
#ifdef DEBUG
//GPIO0 = 32KHz Clock, GPIO1=wake-up event, GPIO2=Active State
SpiWriteRegister(0x0B, 0x0B);
SpiWriteRegister(0x0C, 0x0B);
SpiWriteRegister(0x0D, 0x15);
```

```
SpiWriteRegister(0x51, 0x01);
#else
SpiWriteRegister(0x0B, 0x1F);
SpiWriteRegister(0x0C, 0x1F);
SpiWriteRegister(0x0D, 0x1F);
SpiWriteRegister(0x51, 0x01);
#endif
#ifdef ANTENNA_DIVERSITY
SpiWriteRegister(0x0C, 0x17);      //write 0x17 to the GPIO1 Configuration(set the Antenna 1 Switch used // for
antenna diversity )
SpiWriteRegister(0x0D, 0x18);
//write 0x18 to the GPIO2 Configuration(set the Antenna 2 Switch used for antenna diversity )
#endif
#ifdef ONE_SMA_WITH_RF_SWITCH
SpiWriteRegister(0x0C, 0x12);              //write 0x12 to the GPIO1 Configuration(set the TX state)
SpiWriteRegister(0x0D, 0x15);              //write 0x15 to the GPIO2 Configuration(set the RX state)
#endif
SpiWriteRegister(0x05, 0x00);              //write 0x00 to the Interrupt Enable 1 register
SpiWriteRegister(0x06, 0x08);              //write 0x08 to the Interrupt Enable 2 register
//Radio in sleep mode, with wake up timer
SpiWriteRegister(0x07, 0x20);
//MAIN Loop//
while(1)
{
while( NIRQ == 1 );
// enable active state
SpiWriteRegister(0x07, 0x67);
//read interrupt status registers
```

```
ItStatus1 = SpiReadRegister(0x03);                    //read the Interrupt Status1 register
ItStatus2 = SpiReadRegister(0x04);                    //read the Interrupt Status2 register
// Wake Up Timer Interrupt Occured
if ((ItStatus2 & 0x08) == 0x08 )
{
// Delay
for(delay=0;delay<5000;delay++);
Battery_Voltage_Level = SpiReadRegister(0x1B);
// Error in Battery Voltage Reading
if (Battery_Voltage_Level== 0x00)
{
LED1 = 1;
for (temp8=0;temp8<255;temp8++)
{
for(delay=0;delay<10000;delay++);
}
}
// Low Battery Detected
else if (LBDT > Battery_Voltage_Level)
{
LED2 = 1;
LED1 = 0;
for (temp8=0;temp8<255;temp8++)
{
for(delay=0;delay<10000;delay++);
}
}
// No Low Battery
```

```
else
{
LED2 = 0;
LED1 = 0;
}
Battery_Voltage_Level = 0x00;
//Radio in sleep mode, with wake up timer
SpiWriteRegister(0x07, 0x20);
}
}
}
```

Current Consumption:

Workaround A: Low Battery Detection using Low Duty Cycle Mode

No.	State	Time	Current	Average Current
1	Sleep Mode	0.5 s	1 µA	2.5 µA
2	Active State	0.1221 ms	18.5 mA	

Table 1 : Timing & Current Consumption for Low Battery Detection using Low Duty Cycle Mode

Workaround B: Low Battery Detection Using Wake Up Timer

No.	State	Time	Current	Average Current
1	Sleep Mode	User Programmable	1 µA	1 µA (500 s interval)
2	Active State	~4.9 ms	18.5 mA	

Table 2 : Timing & Current Consumption for Low Battery Detection using Wake-up Timer