Introduction

The SPIRIT1 is a very low power RF transceiver, intended for RF wireless applications in the sub-1 GHz band. It is designed to operate both in the license-free ISM and SRD frequency bands at 169, 315, 433, 868, 915 and 920 MHz. This application note describes how to use the low duty cycle (LDC) mode in order to further optimize power consumption during reception and transmission operation.
## Contents

1 Overview of low duty cycle mode ................................................................. 3  
   1.1 LDC reception mode ........................................................................... 3  
       1.1.1 RX timeout using sync detection .................................................. 4  
       1.1.2 RX timeout using RSSI ................................................................. 5  
       1.1.3 Current consumption figures ....................................................... 6  
2 Programming tips ..................................................................................... 9  
3 Reference ................................................................................................. 10  
4 Revision history ..................................................................................... 11
1 Overview of low duty cycle mode

The SPIRIT1 includes hardware support to manage low duty cycle operation that can be used in two ways:
1. To reduce average power consumption during receive operations
2. To build a synchronized star network where both transmitter and receiver can sleep periodically to reduce average power consumption.

LDC mode is essentially controlled by two timers:
1. The LDC timer which defines the window where the duty cycle operation take place
2. The RX_TIMEOUT timer which defines the amount of time that the receiver is active.

1.1 LDC reception mode

In this mode, the SPIRIT1 can periodically switch on the receiver to check if a packet is being transmitted, and then go back to sleep in order to save power. This mode allows the reduction of average reception power consumption, while allowing the device to continue receiving packets under certain conditions.

Figure 1: LDC reception mode

In this mode, the user defines an operation cycle ($t_{\text{LDC\_cycle}}$ in Figure 1: “LDC reception mode”) whose max length is about 2 seconds with a step of 29 μs. In the $t_{\text{LDC\_cycle}}$ period, a period is also defined where the receiver is automatically turned on for packet reception ($t_{\text{RX}}$ in Figure 1: “LDC reception mode”). In this mode, users can set the duration of the $t_{\text{LDC\_cycle}}$ and the $t_{\text{RX}}$, in order to reduce average power consumption based on the target application. The average current consumption, in mA for the SPIRIT1, is calculated using:

$$\text{Average Current Consumption} = \frac{t_{\text{RX}}}{t_{\text{LDC\_cycle}}} \times \text{Current during RX}$$
Overview of low duty cycle mode

Equation 1

\[
 RX_{\text{consumption\_average}} = \frac{t_{\text{RX}}}{t_{\text{LDC\_cycle}}} \cdot 9 + \frac{t_{\text{LDC\_cycle}} - t_{\text{RX}}}{t_{\text{LDC\_cycle}}} \cdot 0.00085 \text{ mA}
\]

The following example shows how to set up the SPIRIT1 with a specific duty cycle. Assuming that the application is using a \(t_{\text{LDC\_cycle}}\) of 100 ms and a \(t_{\text{RX}}\) of 10 ms, the average RX current consumption will be:

Equation 2

\[
 \frac{10}{100} \cdot 9 + \frac{100 - 10}{100} \cdot 0.00085 = 0.9 \text{ mA}
\]

The registers to configure the SPIRIT1 timers are as described in Table 1: "LDCR and RX_TIMEOUT prescaler and counter settings".

<table>
<thead>
<tr>
<th>Register name</th>
<th>Register address</th>
<th>Register value</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDCR_PRESCALER</td>
<td>0x55</td>
<td>0x26</td>
<td>Assuming RCO clock of 34.7 kHz</td>
</tr>
<tr>
<td>LDCR_COUNTER</td>
<td>0x56</td>
<td>0x58</td>
<td></td>
</tr>
<tr>
<td>RX_TIMEOUT_PRESCALER</td>
<td>0x53</td>
<td>0x2A</td>
<td>Assuming 26 MHz digital domain clock</td>
</tr>
<tr>
<td>RX_TIMEOUT_COUNTER</td>
<td>0x54</td>
<td>0x04</td>
<td></td>
</tr>
</tbody>
</table>

1.1.1 RX timeout using sync detection

In this mode, the RX timeout is stopped when a valid SYNC word, which marks the beginning of the packet, is received and the receiver is expected to receive a complete packet. When the receiver is working in LDC mode, the transmitter should take this into account, retransmitting the packet for a period of time at least equal to \(t_{\text{LDC\_cycle}}\), so that it is certain that the receiver has the opportunity to catch the SYNC word. In addition, the \(t_{\text{RX}}\) needs to be selected according to the length of the packet to be received in order to ensure that a valid sync word is detected during the RX window. The \(t_{\text{RX}}\) should be equal to the packet period plus the length of the preamble. The packet period can be defined as follows:

Equation 3

\[
 t_{\text{packet\_period}} = (t_{\text{preamble}} + t_{\text{sync}} - t_{\text{packet}}) \cdot \text{datarate} + t_{\text{interpacket}}
\]

where datarate is the bit rate at which the information is sent and the other equation parameters are according to Table 2.
Finally, the parameter tRX can be set based on the following equation:

Equation 4

\[ t_{RX} \geq t_{packet\_period} + (t_{preamble} + t_{sync}) \cdot datarate \]

This ensures that a node doing RX in LDC mode is able to receive a packet sent by a transmitter, which repeats the message for a certain period of time. Figure 2 shows the minimum RX ON time to ensure a valid sync is captured.

The transmitter needs to repeat the packet to be transmitted for at least tLDC_cycle. From this we can see how the consumption of the receiver is reduced at the expenses of increasing the consumption on the transmitter side and increasing the latency of response. This may be perfectly acceptable and desirable, for example, in applications where the receiver is battery-operated and the transmitter is powered by mains.

### 1.1.2 RX timeout using RSSI

In this mode, the receiver timeout is stopped upon detection of signal energy above a certain user defined threshold. This mode will further reduce average power consumption by decreasing the time when receiver must be on. On the other hand this mode will add some further burden on the micro which needs to check that a valid message is received within a user defined timeout. In this mode, the SPIRIT1 cycles between RX active and sleep as described in Figure 1; the only difference is that the minimum time tRX will be much smaller than that described in Section 1.1.1. The time to measure RSSI of the incoming signal will vary according to the RX filter bandwidth shown in Table 3: "RSSI detection time".

### Table 2: Packet period key parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lpreamble</td>
<td>Length of the preamble in bits</td>
</tr>
<tr>
<td>lsync</td>
<td>Length of the sync in bits</td>
</tr>
<tr>
<td>lpacket</td>
<td>Length of the packet in bits</td>
</tr>
<tr>
<td>tinterpacket</td>
<td>Delay between two consecutive transmissions; it is the cumulative time of interrupt response time, reload of the TX fifo, send TX command, ready to TX transition.</td>
</tr>
</tbody>
</table>

Figure 2: RX on minimum time for RX timeout on sync

The transmitter needs to repeat the packet to be transmitted for at least tLDC_cycle. From this we can see how the consumption of the receiver is reduced at the expenses of increasing the consumption on the transmitter side and increasing the latency of response. This may be perfectly acceptable and desirable, for example, in applications where the receiver is battery-operated and the transmitter is powered by mains.

### 1.1.2 RX timeout using RSSI

In this mode, the receiver timeout is stopped upon detection of signal energy above a certain user defined threshold. This mode will further reduce average power consumption by decreasing the time when receiver must be on. On the other hand this mode will add some further burden on the micro which needs to check that a valid message is received within a user defined timeout. In this mode, the SPIRIT1 cycles between RX active and sleep as described in Figure 1; the only difference is that the minimum time tRX will be much smaller than that described in Section 1.1.1. The time to measure RSSI of the incoming signal will vary according to the RX filter bandwidth shown in Table 3: "RSSI detection time".
Since in this mode the RX active time can be very short, it may be worthwhile to also take into account the time to move from the SLEEP state to the RX state, which is in the order of 120 µs. An equation that approximates the average current consumption in this mode is as follows:

**Equation 5**

\[
RX_{\text{consumption\_average}} = \frac{120^{-6} + t_{RX}}{t_{LDC\_cycle}} \cdot 9 + \frac{t_{LDC\_cycle} - t_{RX} - 120^{-6}}{t_{LDC\_cycle}} \cdot 0.00085 \text{ mA}
\]

The equation assumes that there is an average current consumption of 9 mA during the transition from SLEEP to RX. It is important to understand that this mode also requires some cooperation from the microcontroller driving the SPIRIT1. In particular, since the RX timeout is stopped upon detection of an RSSI value above a certain threshold, two cases can occur:

1. A valid packet is detected and the reception is completed and signaled to the micro.
2. No packet is detected and the SPIRIT1 can stay indefinitely in the RX state.

In order to avoid case 2, it is suggested that upon detection of the RSSI signal, the microcontroller starts a timer and then aborts reception if a valid packet is not received within a user-defined timeout.

### 1.1.3 Current consumption figures

A set of measurements have been taken using an LDC cycle of 100 ms and a variable duty cycle. The values have been taken using a Keysight N6782A source and measure unit with ‘Seamless Current Ranging’ and are plotted in Figure 3. The actual values with comparison to predicted values are shown in Table 4.
Overview of low duty cycle mode

Figure 3: Average RX current consumption vs. duty cycle

![Graph showing average RX current consumption vs. duty cycle](image)

Table 4: Current consumption data with different duty cycles

<table>
<thead>
<tr>
<th>Duty cycle (%)</th>
<th>Current (µA)</th>
<th>Predicted value (µA)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1%</td>
<td>80.0372</td>
<td>90.84</td>
</tr>
<tr>
<td>2%</td>
<td>162.9491</td>
<td>171.83</td>
</tr>
<tr>
<td>3%</td>
<td>249.9657</td>
<td>257.32</td>
</tr>
<tr>
<td>4%</td>
<td>332.9219</td>
<td>342.82</td>
</tr>
<tr>
<td>5%</td>
<td>419.8115</td>
<td>428.31</td>
</tr>
<tr>
<td>10%</td>
<td>841.9774</td>
<td>855.77</td>
</tr>
<tr>
<td>20%</td>
<td>1690.3268</td>
<td>1710.68</td>
</tr>
<tr>
<td>30%</td>
<td>2541.7257</td>
<td>2565.60</td>
</tr>
<tr>
<td>40%</td>
<td>3410.6563</td>
<td>3420.51</td>
</tr>
<tr>
<td>50%</td>
<td>4267.8439</td>
<td>4275.43</td>
</tr>
<tr>
<td>60%</td>
<td>5120.7135</td>
<td>5130.34</td>
</tr>
</tbody>
</table>

Figure 4: "Current consumption during wakeup" shows the dynamic power consumption during the transition from sleep to RX.
Figure 4: Current consumption during wakeup
2 Programming tips

This section provides some indication of how to program the SPIRIT1 to perform RX in LDC mode.

Assuming that the transceiver has been properly initialized, the following steps are required to activate the LDC mode in reception using the SPIRIT C driver:

1. `SpiritTimerSetWakeUpTimerMs (LDC_CYCLE);` // Specify the LDC cycle time in ms
2. `SpiritTimerSetRxTimeoutMs (RX_TIMEOUT);` // Specify RX timeout in ms
3. `SpiritTimerLdcrMode(S_ENABLE);` // Enable LDC mode
4. `SpiritCmdStrobeRx(SPIRIT_CMD_RX);` // Start RX

The same commands can be performed as a set of registers access as follows (assuming an LDC cycle of 100 ms and a RX timeout of 1 ms and a crystal of 50 MHz):

1. `WriteRegister(0x55,0x26), WriteRegister(0x56,0x58)` # Set LDC cycle time
2. `WriteRegister(0x53,0x09), WriteRegister(0x54,0x02)` # Set Rx timeout
3. `WriteRegister(0x50, ReadRegister(0x50) | 1)` # Set bit 0 of register 50 to 1 to activate LDC mode
4. `SendStrobe (0x61)`
3 Reference

Reference

1. SPIRIT1 datasheet
4 Revision history

Table 5: Document revision history

<table>
<thead>
<tr>
<th>Date</th>
<th>Revision</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>07-Jan-2013</td>
<td>1</td>
<td>Initial release.</td>
</tr>
<tr>
<td>27-Feb-2015</td>
<td>2</td>
<td>Updated: Section 1.1.3</td>
</tr>
</tbody>
</table>
IMPORTANT NOTICE – PLEASE READ CAREFULLY

STMicroelectronics NV and its subsidiaries (“ST”) reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice. Purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST’s terms and conditions of sale in place at the time of order acknowledgement.

Purchasers are solely responsible for the choice, selection, and use of ST products and ST assumes no liability for application assistance or the design of Purchasers’ products.

No license, express or implied, to any intellectual property right is granted by ST herein.

Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.

ST and the ST logo are trademarks of ST. All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.

© 2015 STMicroelectronics – All rights reserved