CC1100/CC2500 - Wake-On-Radio

By Siri Namtvedt

Keywords

- *CC1100*
- *CC1101*
- *CC1100E*
- *CC2500*
- *WOR*
- *Event 0*
- *Event 1*
- *RX timeout*
- *RSSI Threshold*

1 Introduction

The CC1100, CC1101, CC1100E, and CC2500 all have Wake on Radio (WOR) functionality, which enables the radio to periodically wake up from SLEEP mode and listen for incoming packets without MCU interaction. After a programmable time in RX, the chip goes back to the SLEEP unless a packet has been received. The purpose of this application note is to explain the theory of operation and the different registers involved when using Wake on Radio, as well as highlighting important aspects when using WOR mode. Figure 1 shows the relationship between the WOR events and the different radio states.

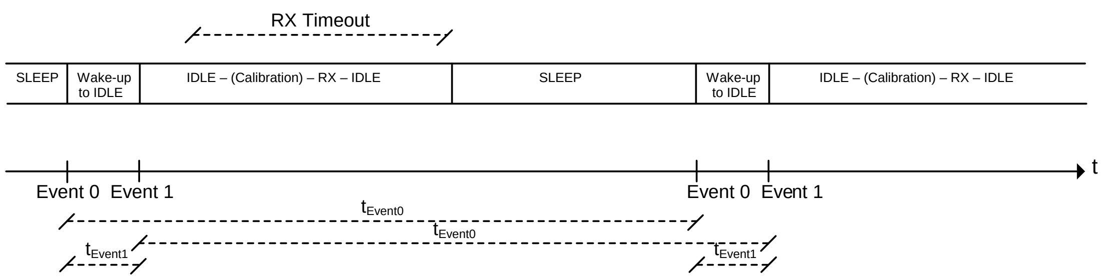

Figure 1. WOR Events and Radio States

Table of Contents

2 Abbreviations

CS	Carrier Sense
MCU	Micro Controller Unit
NA	Not Applicable
PQT	Preamble Quality Threshold
RSSI	Received Signal Strength Indicator
WOR	Wake on Radio
XOSC	Crystal Oscillator

3 Registers

This section covers the theory of operation, the equations governing WOR operation, as well as configuration of the different registers relevant for WOR mode. For more details on the registers described in this section, please see [1], [2], [3], and [4].

3.1 WOREVT1 and WOREVT0

In SLEEP mode with WOR enabled, reaching Event 0 will turn on the digital regulator and start the crystal oscillator. The time between two consecutive Event 0s is programmed with a mantissa value given by `WOREVT1.EVENT0` and `WOREVT0.EVENT0`, and an exponent value set by `WORCTRL.WOR_RES`. See Equation 1.

$$t_{Event0} = \frac{750}{f_{XOSC}} \cdot EVENT0 \cdot 2^{5 \cdot WOR\_RES}$$

Equation 1. t_{Event0}

Event 0 can be monitored on one of the GDOx pins by setting `IOCFGx.GDOx_CFG` = 0x24. See Figure 2.

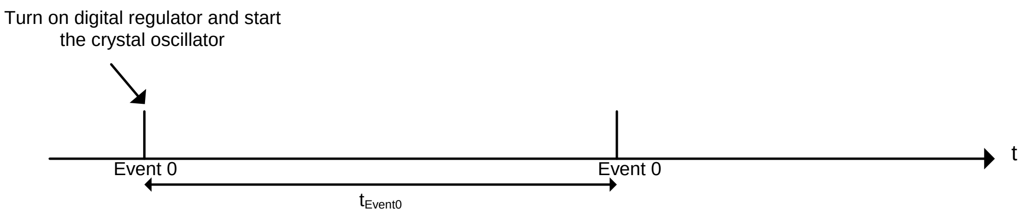

Figure 2. Event 0

Due to a design error related to the WOR timer module, see [5], [6], [7], and [8], the time from the radio enters SLEEP mode until the next Event 0 is programmed to appear (t_{SLEEP}) should not be less than 11.08 ms when using a 26 MHz crystal and 10.67 ms when a 27 MHz crystal is used. If $t_{SLEEP} < t_{SLEEP_{min}}$, there is a chance that the consecutive Event 0 will occur (750·128) / f_{XOSC} s too early.

$t_{SLEEP_{min}}$ can be calculated as showed in Equation 2:

$$t_{SLEEP_{\min}} = \frac{750}{f_{xosc}} \cdot 384$$

Equation 2. $t_{SLEEP_{min}}$

The minimum time between two Event 0s ($t_{Event0_{min}}$) depends on t_{Event1} (see 3.2.2), if the PLL is being calibrated or not, and the RX timeout. Example 1 will illustrate these dependencies. By looking at a plot of current consumption vs. time when the radio is configured for WOR mode (see Figure 3), an equation for $t_{Event0_{min}}$ can be found (see Equation 3).

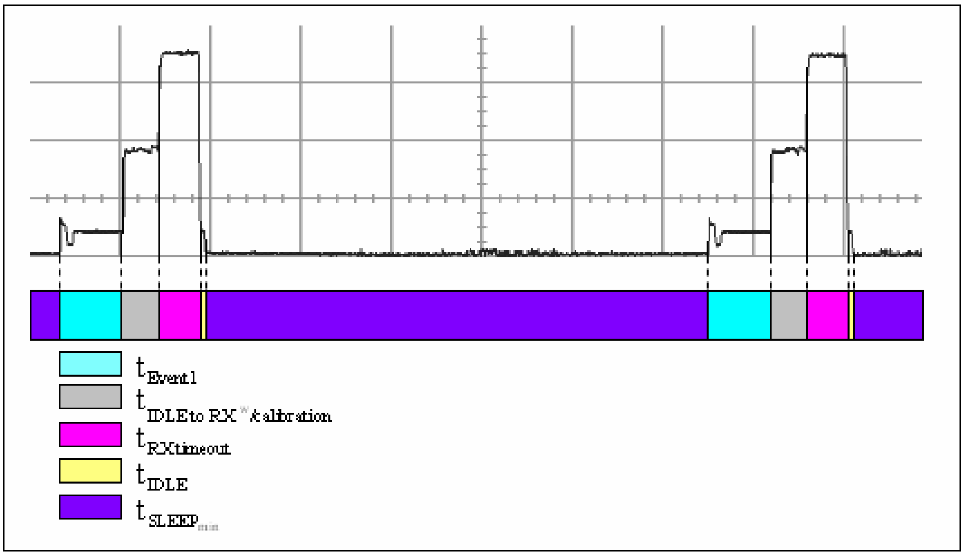

Figure 3. Current Consumption vs. Time

$$t_{Event0_{min}} = t_{Event1} + t_{IDLE\ to\ RX\ ^{w}/Calibration} + t_{RX\ timeout} + t_{IDLE} + t_{SLEEP_{min}}$$

Equation 3. $t_{Event0_{min}}$

$$t_{Event1} = x \cdot \frac{750}{f_{XOSC}}$$

, where x is given by Table 1

Equation 4. t_{Event1}

x	WORCTRL.EVENT1
4	0
6	1
8	2
12	3
16	4
24	5
32	6
48	7

Table 1. *x* Values to use in Equation 4

See section 3.2.2 for considerations that need to be taken into account when programming t_{Event1}.

<u>Example 1:</u>

Assume using the register settings listed in Table 2 (f_{xosc} = 26 MHz).

Register	Value	Comment
MCSM0	0x18	Calibrate when going from IDLE to RX
WORCTRL	0x78	EVENT1 = 7 and WOR_RES = 0
MCSM2	0x01	RX_TIME = 1

Table 2. Register Settings for Example 1

$$t_{Event1} = 48 \cdot \frac{750}{26 \cdot 10^{6}} = 1.385 \cdot 10^{-3} = 1.385[ms]$$ (Equation 4 and Table 1)

It takes 798.8 us to go from IDLE to RX mode with calibration when using a 26 MHz crystal (see [1], [2], [3], and [4]).

⇒ $t_{\text{IDLE to RX}}\ ^{\text{w}}/_{\text{Calibration}}$ = 798.8 us

`MSCM2.RX_TIME` = 1 and `WORCTRL.WOR_RES` = 0 ⇒ Duty cycle = 6.25 % (see [1], [2], [3], and [4]). The duty cycle is used to calculate $t_{\text{RX timeout}}$ (Equation 7).

$t_{IDLE} = 150us$ (see 3.2.3)

$$t_{SLEEP_{\min}} = \frac{750}{26 \cdot 10^{6}} \cdot 384 = 11.08 \cdot 10^{-3} = 11.08\left[ms\right] \qquad \text{(Equation 2)}$$

$$t_{Event0_{\min}} = 1.385 \cdot 10^{-3} + 798.8 \cdot 10^{-6} + (t_{Event0_{\min}} \cdot 6.25\%) + 150 \cdot 10^{-6} + 11.08 \cdot 10^{-3}$$
$$\Rightarrow t_{Event0_{\min}} = 14.31 \cdot 10^{-3} = 14.31\left[ms\right] \qquad \text{(Equation 3)}$$

$$t_{Event0} = \frac{750}{26 \cdot 10^{6}} \cdot EVENT0 \cdot 2^{5 \cdot 0} \geq 14.31 \cdot 10^{-3} \Rightarrow EVENT0 \geq 497 \qquad \text{(Equation 1)}$$

`EVENT0` = 497 ⇒ `WOREVT1.EVENT0` = 0x01 and `WOREVT0.EVENT0` = 0xF1.

$$t_{Event0_{\min}} = \frac{750}{26 \cdot 10^{6}} \cdot 497 \cdot 2^{5 \cdot 0} = 14.34 \cdot 10^{-3} = 14.34\left[ms\right] \qquad \text{(Equation 1)}$$

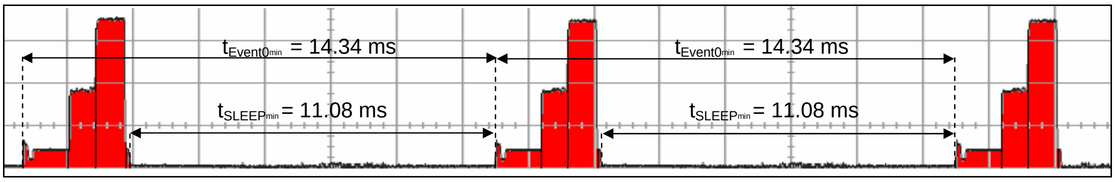

Figure 4. Current Consumption vs. Time (No Packets Received)

Figure 4 shows how the radio will wake up every 14.34 ms when no packets are being received. If a packet is received, the packet will typically be processed by the MCU before the radio is being put back into WOR mode by issuing an SWOR strobe command (see Figure 5).

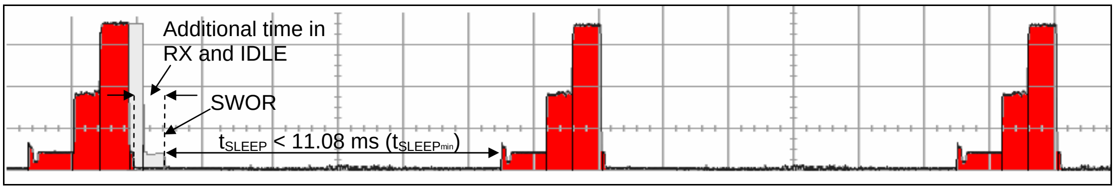

Figure 5. Current Consumption vs. Time (One Packet Received)

When a packet has been received and an SWOR strobe has been issued, t_{SLEEP} becomes less than $t_{SLEEP_{\min}}$ and there is a chance that the consecutive Event 0 will occur (750·128) / f_{XOSC} = 3.69·10^{-3} = 3.69 ms too early (see Figure 6). t_{Event0} will in this case be 14.34 ms – 3.69 ms = 10.65 ms.

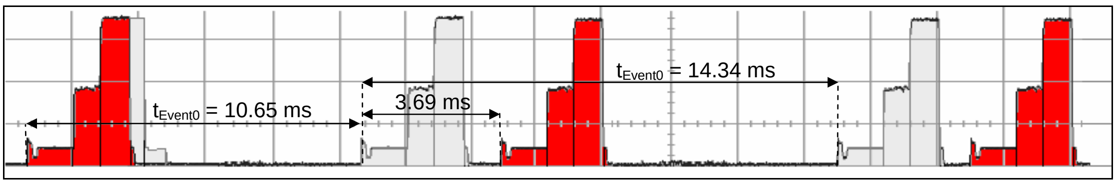

Figure 6. Current Consumption vs. Time (t_{Event0} = 10.65 ms)

After this, t_{Event0} = 14.34 ms, as it is supposed to, but there will be a permanent time shift of -3.69 ms compared to the case where no packets were received (Figure 4).

If it is important for the application that there are no time shifts, a solution is to stay in IDLE until the next Event 0 occurs (can be monitored on a GDOx pin) and then strobe SWOR to make sure that the time from strobing SWOR until the next Event 0 is greater than $t_{SLEEPmin}$. The application will then miss one RX period, but the next Event0 will occur when it is supposed to (see Figure 7).

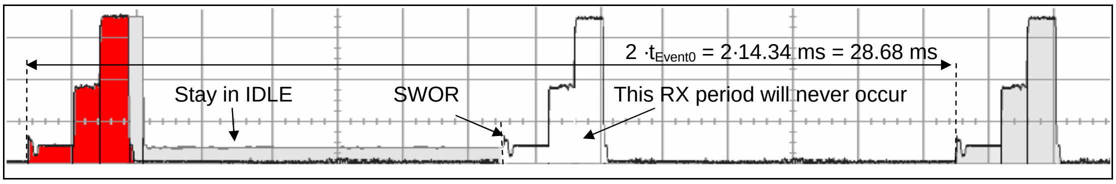

Figure 7. Current Consumption vs. Time (Avoiding Time Shift)

Example 2:

Assume another scenario using the registers settings listed in Table 3 (f_{xosc} = 26 MHz).

Register	Value	Comment
`MCSM0`	0x18	Calibrate when going from IDLE to RX
`WORCTRL`	0x38	`EVENT1` = 3 and `WOR_RES` = 0
`MCSM2`	0x00	`RX_TIME` = 0
`WOREVT1`	0x28	`EVENT0` = 10400
`WOREVT0`	0xA0	

Table 3. Register Settings for Example 2

$$t_{Event0} = \frac{750}{26 \cdot 10^6} \cdot 10400 \cdot 2^{5 \cdot 0} = 300 \cdot 10^{-3} = 300\,[ms]$$ (Equation 1)

$$t_{Event1} = 12 \cdot \frac{750}{26 \cdot 10^6} = 346.15 \cdot 10^{-6} = 346.15\,[us]$$ (Equation 4 and Table 1)

`MSCM2.RX_TIME` = 0 and `WOR_RES` = 0 ⇒ Duty cycle = 12.5 %

$$t_{RX\ timeout} = 0.3 \cdot 12.5\% = 37.5 \cdot 10^{-3} = 37.5\,[ms]$$ (Equation 7)

From Equation 3 we have that:

$$t_{SLEEP} = t_{Event0} - t_{Event1} - t_{IDLE\ to\ RX\ ^{w}/Calibration} - t_{RX\ timeout} - t_{IDLE}$$

$$t_{SLEEP} = 0.3 - 346.15 \cdot 10^{-6} - 798.8 \cdot 10^{-6} - 37.5 \cdot 10^{-3} - 150 \cdot 10^{-6} = 261.21 \cdot 10^{-3} = 261.21\,[ms]$$

Even if $t_{SLEEP} > t_{SLEEPmin}$ when no packets are being received, one has to make sure that an SWOR strobe is not issued too close to the following Event 0. This can be done by reading the WOR timer value from the `WORTIME1` and `WORTIME0` registers.

`WORCTRL.WOR_RES` = 0 means that the Event 0 resolution is given by 1·(750/f_{xosc}) (see [1], [2], and [4]).

$$x \cdot \left[1 \cdot \frac{750}{26 \cdot 10^6}\right] = t_{Event0} - t_{SLEEP_{min}} = 300 \cdot 10^{-3} - 11.08 \cdot 10^{-3} \Rightarrow x = 10016$$

To make sure that there is more than 11.08 ms until the next Event 0 occurs, `WORTIME1:WORTIME0` must be less than 10016.

Assume the same t_{Event0}, but using `WORCTRL.WOR_RES` = 1 instead.

`WORCTRL.WOR_RES` = 1 means that the Event 0 resolution is given by $2^5 \cdot (750/f_{xosc})$ (see [1], [2], [3], and [4]).

$$x \cdot \left[2^5 \cdot \frac{750}{26 \cdot 10^6} \right] = t_{Event0} - t_{SLEEP_{min}} = 300 \cdot 10^{-3} - 11.08 \cdot 10^{-3} \Rightarrow x = 313$$

This means that to make sure that there is more than 11.08 ms to the next Event 0, the timer value read from `WORTIME1` and `WORTIME0` must be less than 313 if it should be safe to issue an SWOR command.

Due to a design error affecting the synchronization mechanism between the SPI clock domain and the internal clock domain special care must be taken when reading the `WORTIME1` and `WORTIME0` registers. Please see [5], [6], [7], and [8] for more details. In addition to the workaround described in [5], [6], [7], and [8] (reading the registers several times) another workaround can be used since these registers are updated on the falling edge of the RC oscillator output. This workaround is to make sure that the registers are read on the rising edge of the RC oscillator output and that the reading is completed before the consecutive falling edge. The RC oscillator output can be monitored on a GDOx pin by settings `IOCFGx.GDOx_CFG` = 0x27.

3.2 WORCTRL

3.2.1 RC_PD

The `RC_PD` bit is the power down signal for the RC oscillator. This bit must be cleared for the RC oscillator to run.

3.2.2 EVENT1[2:0]

Event 1 follows Event 0 after a programmed timeout (see Equation 4). The radio will wake up on Event 0 and issue an RX strobe on Event 1. If calibration of the PLL is to be performed when going from IDLE to RX (`MCSM0.FS_AUTOCAL` = 1) the radio will enter RX mode 798.8 us after Event 1 occurred given that a 26 MHz crystal is being used. If calibration is disabled (`MCSM0.FS_AUTOCAL = 0`), RX mode is entered 75.1 us after Event 1 (26 MHz crystal). Event1 can be monitored on one of the GDOx pins by setting `IOCFGx.GDOx_CFG` = 0x25. t_{Event1} must be long enough for the `CHP_RDYn` signal to be asserted before reaching Event 1.

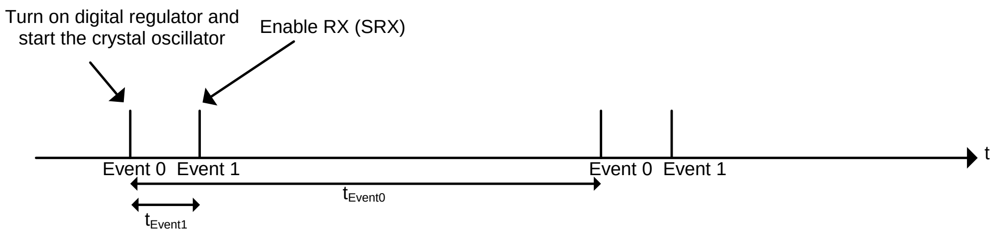

Figure 8. Event 1

It is possible to keep the crystal on during SLEEP mode (`MCSM0.XOSC_FORCE_ON` = 1) to be able to reduce t_{Event1}, but this will cause the current consumption in SLEEP mode to increase significantly (see [1], [2], [3], and [4]). Figure 9 shows the current consumption when the crystal is turned off in SLEEP mode (A) and when the crystal is kept on (B). It is not possible to observe the difference in SLEEP current due to the resolution of the y-axis, but in the case where the crystal is running, one can see that the chip is ready and in IDLE mode about 110 us earlier than what is the case if the crystal oscillator has been turned off. In the case where the crystal oscillator is running in SLEEP mode, it can be seen that the radio returns to SLEEP earlier than what is the case when XOSC is turned off. This is due to the fact that the calibration of the RC oscillator (see 3.2.3) starts earlier and thus finishes of sooner.

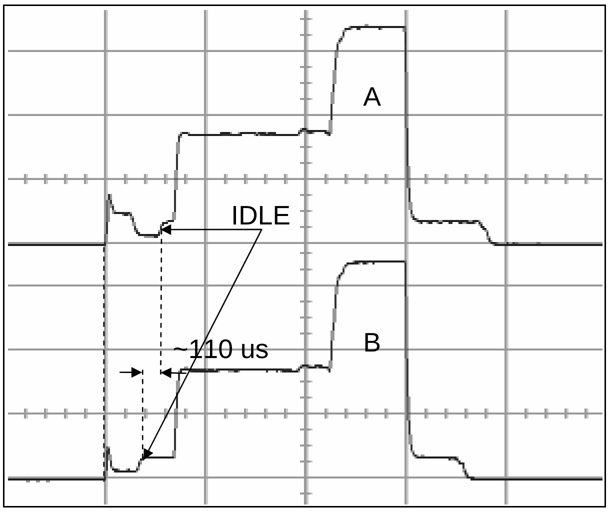

Figure 9. `MCSM0.XOSC_FORCE_ON` = 1

3.2.3 RC_CAL

The `RC_CAL` bit enables calibration of the RC oscillator when set to one. The frequency of the low-power RC oscillator varies with temperature and supply voltage, and in order to keep the frequency as accurate as possible, the RC oscillator will be calibrated continuously whenever the XOSC is running and the radio is not in the SLEEP mode. When the radio enters SLEEP mode, the RC oscillator will use the last valid calibration result. It typically takes 2 ms to calibrate the RC oscillator. If the crystal is stable for ~7 ms every time the radio wakes up from SLEEP, the RC oscillator will have time to complete 3 calibrations and be in the middle of calibration number 4 when the radio goes back to SLEEP mode via IDLE (see Figure 10). It will in this case use calibration result number 3, which is the last valid calibration. However, if the crystal is running for less than 2 ms, the radio will enter IDLE mode after the RX timeout, and then stay in IDLE until one calibration of the RC oscillator is performed, before returning to SLEEP mode (see Figure 11). This is important to have in mind when calculating current consumption in WOR mode. Table 4 shows register setting for WOR mode where the crystal will be on and stable for about 7 ms.

Register	Value	Comment
MCSM0	0x18	Calibrate the PLL when going from IDLE to RX or TX (or FSTXON)
WORCTRL	0x38	EVENT1 = 3 ⇒ t_{Event1} = 346.15 us WOR_RES = 0
WOREVT1	0x06	EVENT0 = 1733
WOREVT0	0xC5	
MCSM2	0x00	RX_TIME = 0 ⇒ Duty cycle = 12.5 %

Table 4. Register Values for RX Timeout > RC Oscillator Calibration Time

Using the settings in Table 4 gives the following values for t_{Event0} and the RX timeout:

$$t_{Event0} = \frac{750}{26 \cdot 10^6} \cdot 1733 \cdot 2^{5 \cdot 0} = 49.99 \cdot 10^{-3} = 49.99\,[ms] \qquad \text{(Equation 1)}$$

$$t_{RX\ timeout} = 49.99 \cdot 10^{-3} \cdot 12.5\% = 6.25 \cdot 10^{-3} = 6.25\,[ms] \qquad \text{(Equation 7)}$$

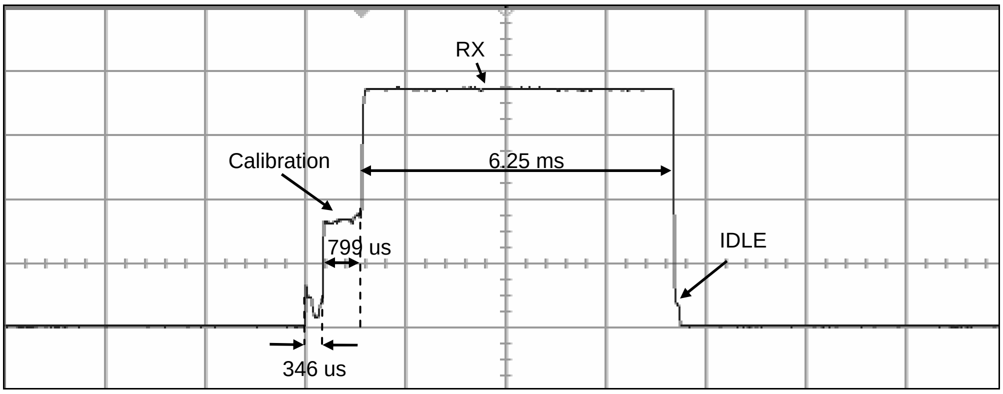

Figure 10. Current Consumption (EVENT1 = 3 and RX Timeout = 6.25 ms)

In this case, the RC oscillator has sufficient time to calibrate when the crystal is running. When the RX timeout is reached, the radio will go back to SLEEP mode, via IDLE, even if it is in the middle of a calibration. In IDLE mode the RCOSC will be set to use the last valid result and after a few 34.667 kHz periods (assuming a 26 MHz crystal), the radio will go back to SLEEP. For calculations one can use 150 us (~5 periods).

Changing `MCSM2` to 0x05 (`MCSM2.RX_TIME` = 5) will reduce the duty cycle to 0.391 %.

$$t_{RX\ timeout} = 49.99 \cdot 10^{-3} \cdot 0.391\% = 195.46 \cdot 10^{-6} = 195.46\,[us] \qquad \text{(Equation 7)}$$

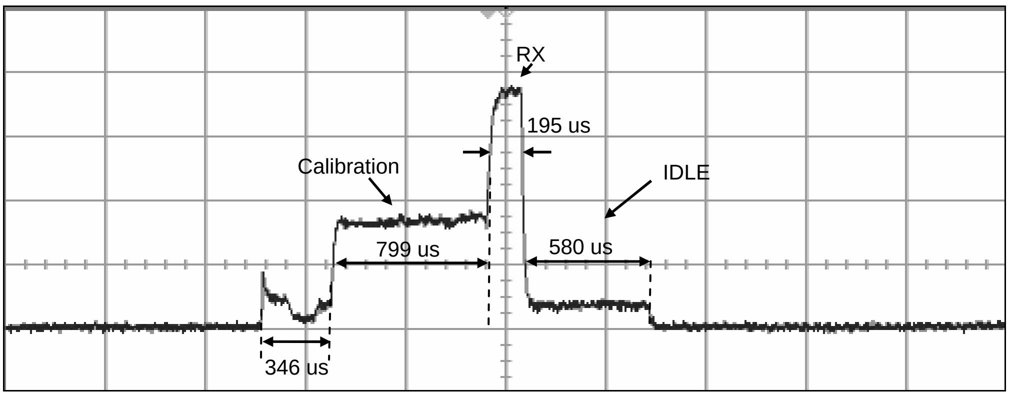

Figure 11. Current Consumption (EVENT1 = 3 and RX Timeout = 195.49 us)

The calibration of the RC oscillator will not have completed when the RX timeout is reached and the radio will stay in IDLE mode until the calibration is finished.

In applications where the radio wakes up very often, typically several times every second, it is possible to turn off the calibration of the RC oscillator to reduce the current consumption. How this should be done is shown in the following code example:

```
//------------------------------------------------------------------------------
halSpiWriteReg(CCxxx0_MCSM2,    0x03);  // RX_TIME = 3
                                        // (duty cycle = 1.563% when WOR_RES = 0)
halSpiWriteReg(CCxxx0_WOREVT1,  0x03);  // EVENT0 = 800
halSpiWriteReg(CCxxx0_WOREVT0,  0x20);
halSpiWriteReg(CCxxx0_WORCTRL,  0x38);  // EVENT1 = 3
                                        // RC_CAL = 1
                                        // WOR_RES = 0
//------------------------------------------------------------------------------
halWait(3000);                          // Wait for RCOSC calibration
halSpiWriteReg(CCxxx0_WORCTRL,  0x30);  // EVENT1 = 3
                                        // RC_CAL = 0
                                        // WOR_RES = 0

calib1 = halSpiReadStatus(CCxxx0_RCCTRL1_STATUS);
calib0 = halSpiReadStatus(CCxxx0_RCCTRL0_STATUS);

halSpiWriteReg(CCxxx0_RCCTRL1, calib1);
halSpiWriteReg(CCxxx0_RCCTRL0, calib0);
halSpiStrobe(CCxxx0_SWORRST);
halSpiStrobe(CCxxx0_SWOR);

while (TRUE);
//------------------------------------------------------------------------------
```

If the calibration of the RC oscillator is turned off it will have to manually be turned on again at regular intervals. How often this must be done depends on the environment in which the application operates (voltage and temperature changes).

3.2.4 WOR_RES[1:0]

The WOR_RES bits control the resolution of t_{Event0}, and hence the maximum timeout of the WOR module. For WOR applications, WORCTRL.WOR_RES should be set to either 0 or 1. WORCTRL.WOR_RES = 0 gives a resolution equal to $750/f_{XOSC}$ while WORCTRL.WOR_RES = 1 gives a resolution equal to $2^5 \cdot 750/f_{XOSC}$. For WORCTRL.WOR_RES equal to 2 or 3, the RX window will be so small compared to the accuracy of Event 0, that it will not make sense to use it. In applications where WOR is not used, other values of WORCTRL.WOR_RES can be used to achieve a certain RX timeout. For an example, see 3.3.3. It is also possible to use the WOR timer as a SLEEP timer and wake up the radio on external interrupts on every Event 0. In this case, it can be useful to use WORCTRL.WOR_RES = 2 or WORCTRL.WOR_RES = 3

3.3 MCSM2

3.3.1 RX_TIME_RSSI

When the `RX_TIME_RSSI` bit is set to one, the RSSI level is compared to a programmable threshold a given time after RX mode has been entered. If the RSSI level is below the programmed threshold at this time, the radio will exit RX mode regardless of the RX timeout. If it is above the same threshold (carrier sense is asserted), it will stay in RX until a packet has been received or until it returns to SLEEP due to an RX timeout. The time it takes from entering RX mode until the RSSI level is valid is estimated using Equation 5.

$$\boxed{RSSI\ response\ time = \frac{8}{BW_{channel}} \cdot \left(\frac{WAIT\_TIME}{2} + 1 + 2^{FILTER\_LENGTH} \right) + MAX\left[\frac{20}{BW_{channel}}, \frac{10}{(8 \cdot DataRate)} \right]}$$

Equation 5. RSSI Response Time

Where

$$\boxed{BW_{channel} = \frac{f_{xosc}}{8 \cdot (4 + CHANBW\_M) \cdot 2^{CHANBW\_E}}}$$

Equation 6. $BW_{Channel}$

For further details regarding the RSSI response time, please see [9].

<u>Example 3:</u>

This example uses the register settings from Table 4 but `MCSM2` is changed from 0x00 to 0x10 (`MCSM2.RX_TIME_RSSI = 1`). For all the other registers, the register settings recommended by SmartRF® Studio for 10 kbps (***CC2500***) is to be used. This means that `AGCCTRL0` = 0x91 and `MDMCFG4` = 0x78.

`AGCCTRL0` = 0x91 ⇒ `AGCCTRL0.WAIT_TIME` = 1 and
`AGCCTRL0.FILTER_LENGTH` = 1

`MDMCFG4` = 0x78⇒ `MDMCFG4.CHANBW_E` = 1 and
`MDMCFG4.CHANBW_M` = 3

$$BW_{channel} = \frac{26 \cdot 10^6}{8 \cdot (4+3) \cdot 2^1} = 232.143 \cdot 10^3 = 232.143 \left[kHz\right] \qquad \text{(Equation 6)}$$

$$RSSI\ response\ time = \frac{8}{232143} \cdot \left(\frac{1}{2} + 1 + 2^1 \right) + MAX\left[\frac{20}{232143}, \frac{10}{(8 \cdot 10000)} \right] \qquad \text{(Equation 5)}$$
$$= 245.6 \cdot 10^{-6} = 245.6 \left[us\right]$$

Figure 12 shows how the radio exits RX mode after only 245 us due to the lack of CS when `MCSM2.RX_TIME_RSSI = 1`. In the case where `MCSM2.RX_TIME_RSSI = 0` or `MCSM2.RX_TIME_RSSI` = 1 and RSSI is above the programmed threshold, the radio will stay in RX searching for a sync word for 6.25 ms.

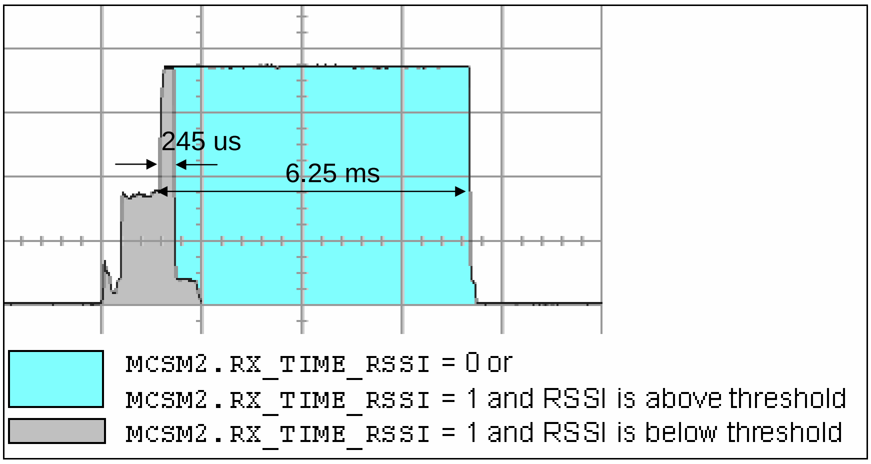

Figure 12. RX Termination based on CS

For details on how to program the RSSI threshold, please see [9].

3.3.2 RX_TIME_QUAL

When RX_TIME_QUAL = 0, the radio will stay in RX when the RX_TIME timer expires if a sync word is found. When RX_TIME_QUAL = 1, the radio stays in RX mode if a sync word is found or the preamble quality threshold is reached (PKTSTATUS.PQT_REACHED = 1). This threshold can be programmed using PKTCTRL1.PQT[2:0] and the higher the threshold, the smaller is the chance of receiving a false packet. An internal counter is being increased by one each time a bit is received that is different from the previous bit and decreased by 8 each time a bit is received that is the same as the last bit. A threshold of 4·PQT is used to gate sync word detection. PKTSTATUS.PQT_REACHED will be held high as long as the preamble quality exceeds the threshold and one sync word length (2 or 4 bytes depending on the MDMCFG2.SYNC_MODE setting) after it goes below. When PKTSTATUS.PQT_REACHED goes low, the radio goes back to SLEEP given that a sync word has not been detected. Be aware that using this feature might increase the time the radio is in RX mode even if no packets are being received. Due to this, it is possible that t_{SLEEP} becomes less than $t_{SLEEP_{min}}$ (see 3.1)

3.3.3 RX_TIME[2:0]

The RX_TIME bits set the timeout for sync word search in RX for both WOR mode and ordinary RX operation. When using WOR mode, RX_TIME can have a value between 0 and 6 when WORCTRL.WOR_RES = 0, and between 0 and 3 when WORCTRL.WOR_RES = 1. RX_TIME = 7 means that the radio will stay in RX mode until a packet is received, regardless of the WORCTRL.WOR_RES setting. The timeout is relative to t_{Event0}.

$$\boxed{t_{RX\ timeout} = t_{Event0} \cdot DutyCycle}$$

Equation 7. $t_{RX\ Timeout}$

RX_TIME[2:0]	WOR_RES = 0	WOR_RES = 1
0 (000)	12.5 %	1.95 %
1 (001)	6.250 %	9765 ppm
2 (010)	3.125 %	4883 ppm
3 (011)	1.563 %	2441 ppm
4 (100)	0.781 %	NA
5 (101)	0.391 %	NA
6 (110)	0.195 %	NA
7 (111)	Until end of packet	

Table 5. Duty Cycle Approximation

The longest RX timeout possible when using WOR is 1.18 s. This is achieved by settings `WORCTRL.WOR_RES[1:0]` = 1 and `MSCM2.RX_TIME[2:0]` = 0.

$$t_{Event0} = \frac{750}{26 \cdot 10^6} \cdot 65536 \cdot 2^{5 \cdot 1} = 60.495[s]$$ (Equation 1)

$$t_{RX\ timeout} = 60.495 \cdot 1.95\% = 1.18[s]$$ (Equation 7)

When running ordinary RX mode, longer timeouts can be achieved by selecting `WORCTRL.WOR_RES` = 2 and `MCSM2.RX_TIME` = 0, or by settings `WORCTRL.WOR_RES` = 3 and `MCSM2.RX_TIME` = 0 or 1. RX timeout (in us) is given by EVENT0 · C · 26/f_{xosx} (in MHz), where C is given by Table 6.

RX_TIME[2:0]	WOR_RES = 0	WOR_RES = 1	WOR_RES = 2	WOR_RES = 3
0 (000)	3.6058	18.0288	32.4519	46.8750
1 (001)	1.8029	9.0144	16.2260	23.4375
2 (010)	0.9014	4.5072	8.1130	11.7188
3 (011)	0.4507	2.2536	4.0565	5.8594
4 (100)	0.2254	1.1268	2.0282	2.9297
5 (101)	0.1127	0.5634	1.0141	1.4648
6 (110)	0.0563	0.2817	0.5071	0.7324
7 (111)	Until end of packet			

Table 6. Constants Used to Calculate RX Timeout

Max achievable RX timeout = $EVENT0_{Max} \cdot C_{Max} = 2^{16} \cdot 46.8750$ = 3072000 [us] = 3.07 s

4 Strobe Commands

4.1 SWOR

Issuing a SWOR strobe command will put the radio in WOR mode when CSn is released given that the radio is in IDLE mode when the strobe command is being issued and the RC oscillator has been enabled by setting `WORCTRL.RC_PD` = 0.

4.2 SWORRST

Issuing this strobe command will reset the WOR timer to the programmed Event 1. Assume that t_{Event0} = 14.34 ms and t_{Event1} = 1.385 ms. The time from an SWORRST command strobe has been issued until the next Event 0 occurs is 14.34 ms – 1.385 ms = 12.96 ms.

5 Waking the Radio from WOR Mode

To exit WOR mode, an SIDLE strobe must be issued.

6 References

[1] CC1100 Low-Cost Low-Power Sub- 1 GHz RF Transceiver (cc1100.pdf)

[2] CC1101 Low-Cost Low-Power Sub-1GHz RF Transceiver (Enhanced CC1100) (cc1101.pdf)

[3] CC1100E Data Sheet (cc1100E.pdf)

[4] CC2500 Single-Chip Low Cost Low Power RF-Transceiver, Data sheet (cc2500.pdf)

[5] CC1100 Errata Notes (swrz012.pdf)

[6] CC1101 Errata Notes (swrz020.pdf)

[7] CC1100E Errata Notes (swrz028.pdf)

[8] CC2500 Errata Notes (swrz002.pdf)

[9] DN505 RSSI Interpretation and Timing (swra114.pdf)

7 General Information

7.1 Document History

Revision	Date	Description/Changes
SWRA126B	2009.03.15	Added references to CC1100E. The time it takes to go from IDLE to RX mode with calibration when using a 26 MHz crystal is changed from 809 us to 798.8 us. The time it takes to go from IDLE to RX mode without calibration when using a 26 MHz crystal is changed from 88.4 us to 75.1 us. In Section 3.3.2, the description of the PQT threshold is changed in accordance with the data sheets [1], [2], [3], and [4]
SWRA126A	2008.13.03	Added CC1101, removed logo from header. Removed references to `PKTCTRL1.AUTO_SYNC`.
SWRA126	2007.01.12	Initial release.