Application Report
SLAA459A–May 2010–Revised July 2012

CC430 Wake-On-Radio Functionality

Dung Dang — *MSP430 Applications*

ABSTRACT

The Wake-On-Radio (WOR) feature of the CC430 is a recommended method for conserving power in wireless systems in which the radio periodically wakes up from SLEEP mode and listens for incoming packets. This application note describes the theory, RF1A registers, and timing of the WOR feature. It also describes the special use case of the CC430F61xx and CC430F51xx devices. This application note concludes with providing an example application to demonstrate the use of WOR on the CC430F61xx and CC430F51xx devices.

Related code examples can be downloaded from http://www.ti.com/lit/zip/slaa459.

Contents

List of Figures

List of Tables

Glossary of Terms

EVENT0	Sleep timer period or cycle (radio)
EVENT1	Time to account for radio transition from IDLE mode to RX mode and oscillator stabilization
HW	Hardware
IRQ	Interrupt Request
ISR	Interrupt Service Routine
LNA_PD	Low Noise Amplifier Power Down (radio) (RX has been turned off)
LPMx	Low-Power Mode for MCU
RF	Radio Frequency
RF_RDY	RF ready (indicates XOSC stable)
RX_EOP	RX End of Packet
SIDLE	Strobe SIDLE (command to change radio into IDLE state)
SRX	Strobe RX (command to change radio into RX state)
STX	Strobe TX (command to change radio into TX state)
SW	Software
SWOR	Strobe SWOR (command to change radio into SWOR mode, starting with SLEEP state)
WOR	Wake On Radio

1 Wake-On-Radio (WOR) Introduction

The Wake-On-Radio (WOR) functionality enables the radio to periodically wake up from SLEEP mode and listen for incoming packets with minimal CPU interaction. This application report assumes prior knowledge on WOR, which can be found in the *CC430 Family User's Guide* (SLAU259), *CC1101 Low-Power Sub-1 GHz RF Transceiver* (SWRS061), and *AN047 -- CC1100/CC2500 Wake on Radio* (SWRA126).

The WOR feature on the CC430 family devices provides two events, Event 0 and Event 1, which can be leveraged to wake up and stabilize the radio core oscillator, and to change the radio to RX mode. Another programmable parameter that the WOR feature uses is the RX timeout, which determines the period during which the radio stays in RX mode. If a packet is received before the period reaches the RX timeout value, the CC430 can process the received packet and return to SLEEP mode. On the other hand, if no packet is received during the RX active period, the radio resumes the SLEEP state after the RX timeout.

Figure 1 shows the relationship between the WOR events and the different radio states. The application has the flexibility to specify the wake up interval (t_{Event0}) as well as the RX active period ($t_{RX\ timeout}$) within each interval. RF systems that require power optimization could lengthen the wake up interval or decrease the RX active duty cycle. Vice versa, more responsive systems with higher packet reception rates could be obtained by decreasing t_{Event0} or increasing $t_{RX\ timeout}$ values. To determine the optimal duration values for t_{Event0}, t_{Event1}, and $t_{RX\ timeout}$, see the *CC430 Family User's Guide* (SLAU259), *AN047 -- CC1100/CC2500 Wake on Radio* (SWRA126), and *DN111 - Current Consumption for a Polling Receiver* (SWRA207).

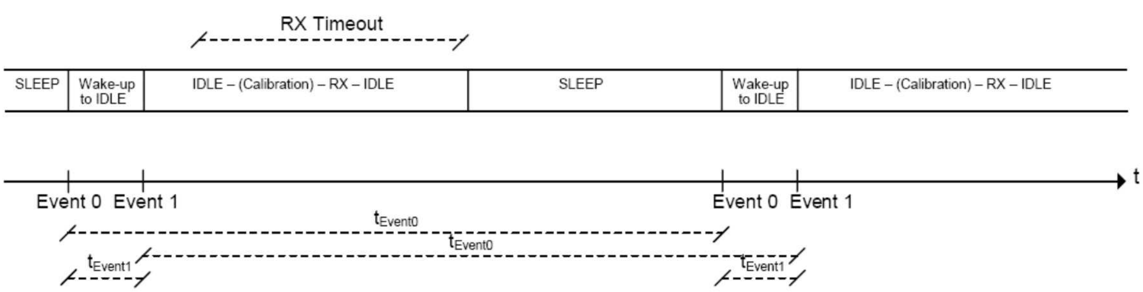

Figure 1. WOR – Event 0 and Event 1

2 CC1101 WOR vs CC430 WOR

The WOR Timer feature on the CC430 is implemented slightly different from the previous implementation in the CC1101. The WOR functionality in CC1101 allows the chip to transition between states (WOR, SLEEP, RX, receive packet) without any CPU intervention. In the CC430 family devices, the WOR Timer feature provides the similar vehicle for the radio to replicate that RF operation flow, but it requires the CPU to manage the transitions between the radio states directly. Specifically, after the RF1A is configured in WOR mode, Event 0 can be configured as an interrupt for the CPU core to initiate the RF1A state change from SLEEP to RX mode. Additionally, the Low Noise Amplifier Power Down (LNA_PD) or RX End of Packet (RX_EOP) events can be used to identify either an RX timeout or the reception of an incoming packet, respectively. Following the LNA_PD or RX_EOP interrupt, the radio can be returned to the SLEEP state.

3 WOR Implementation on the CC430 Family

3.1 Program Flow

Figure 2 shows the WOR program flow suggested by Chapter 19.3.9 of the *CC430 Family User's Guide* (SLAU259).

1. After setting up the RF1A core for standard RF operation (core registers + power amplifier table), the main program configures the WOR control registers and enables WOR_EVENT0 interrupt before going into low power mode (LPMx).
2. Upon the EVENT0 interrupt, the program enters the interrupt handler RF1A_ISR where an SRX command strobe is issued to transition the RF1A core from SLEEP state to IDLE and subsequently RX state.

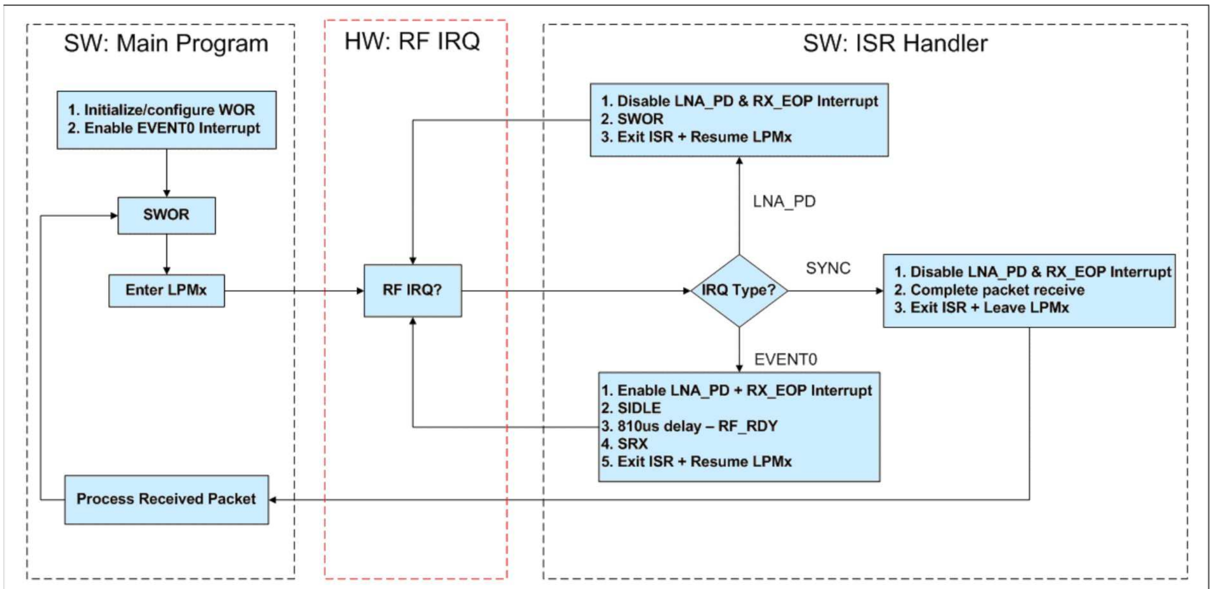

Figure 2. Code Flow for CC430 WOR

3. The *CC430 Family User's Guide* (SLAU259) (see the "Crystal Control" section of the *CC1101-Based Radio Module (RF1A)* chapter) describes the unreliability of RF1A registers during the RF1A core state transition from sleep to active, while the radio oscillator is stabilizing before being switched to source the RF1A core. This stabilization might take up to 810 µs, during which any access to the RF1A register might fail. Therefore, after a wake-up strobe to the RF1A core from, the CPU must stay active for 810 µs to ensure the RF oscillator is stabilized before accessing the radio registers.
4. Once RF1A is in RX mode, the Low Noise Amplifier Power Down (LNA_PD) and RX End of Packet (RX_EOP) interrupts are enabled. CPU returns to sleep mode and waits for one of these two events.
5. In case of a packet reception, CPU enters RF1A_ISR to disable the interrupts, returns to active mode to process the packet, and strobes RF1A core to enter the SLEEP state again with the SWOR command.
6. If no packet is received, after the programmed time, RX will time out and signal the LNA_PD interrupt. In the LNA_PD ISR, the MSP430 strobes SWOR over the RF1A interface to enter the SLEEP state and wait for the next EVENT0.

3.2 Application Impact

The manual handling of the RF1A radio core requires the CPU to stay in active mode for certain periods of time. Careful structure/usage/method/flow should be considered to minimize this active time and maintain the low power consumption feature of the device. Two main focuses where careful handling should take place are at Event 0 and during RF1A oscillator stabilization.

3.2.1 Event 0

In both CC1101 and CC430, the Event 0 is used to wake up the radio and start the radio code state transition process. However, the key usage difference between these two devices lies at the automation of this trigger. The CC430 must wake up the CPU manually with a strobe command. Therefore, to minimize power consumption and to keep the CC430 in LPMx for as much as possible, Event 0 can be set up as an interrupt to trigger the CC1101 RF1A Interrupt Handler. The interrupt handler should strobe the RF1A core, currently in SLEEP mode, to IDLE or RX mode and quickly returns the CPU to low-power mode (LPM) afterwards.

3.2.2 810 µs RF1A Oscillator Stabilization

As mentioned in Section 3.1 and in the *CC430 Family User's Guide* (SLAU259) (see the "Crystal Control" section of the *CC1101-Based Radio Module (RF1A)* chapter), the 810-µs delay is required to prevent access to the RF1A registers during the stabilization of the RF oscillator when the RF1A core transitions from SLEEP state to IDLE or RX mode. Without using additional hardware peripherals such as timers, this delay must be done in software, causing the CPU to stay active for the entire period.

4 Application and Software Example

4.1 Basic Operation

The demo application is developed on the EM430F6137RF900 evaluation board. The board provides button S1 as user input to make the RF mode selection, and two LEDs to indicate the RF state.

The demo application has three RF operation modes: RX mode, TX mode, and RX in Wake-on-Radio mode. Once a mode is made via user selection, the CC430 device will stay permanently in that mode.

Upon startup, the program prompts for RF mode selection via button S1. Short button presses can be used to cycle through three RF modes. The selected RF mode can be indicated by the status of the 2 LEDs based on Table 1.

Table 1. RF Mode Selection Based On LED Indication

	Green LED On	Green LED Off
Red LED On	WOR Mode	TX Mode
Red LED Off	RX Mode	No Selection

A long press (hold down S1 for approximately 0.5 seconds) can be used to lock in the RF state. The program will then start the associated RF operation and stay in that mode for its program lifetime.

Detailed description of the three RF modes:

- RX Mode

  Device stays in RX mode and CPU stays in LPM3. RF1A_ISR is configured to handle RF packet reception. Green LED toggles to indicate packet reception.
- TX Mode

  Device stays in TX mode and periodically transmits an RF packet every 100 ms. Red LED toggles to indicate packet transmission.
- WOR Mode

  Device enters WOR mode and CPU goes to LPM3, proceeds through the program flow described in Section 3.1 in each cycle with the period t_{EVENT0} = 1.06 s, $t_{RX\ Timeout}$ = 1.96 ms. If a packet is received while the RF1A is in RX mode, both LEDs are toggled.

4.2 Verify RF Operation Modes

Two EM430F6137RF900 evaluation boards can be set up to verify all three RF modes. The first board can be selected to transmit and permanently run in TX mode. The red LED toggling indicates the frequency of packet transmissions. The second board can be used to receive packets and be selected in either RX mode or RX WOR mode. The green LED in RX mode and both LEDs in WOR mode toggle after each RF packet reception.

4.3 Code Organization

The Wake-on-Radio project is developed in IAR and contains the following files:

- 5xx library code examples

  5xx libraries that configure the device's system peripherals including the power management module (PMM), the universal clock system (UCS).

- RfRegSettings.c

  Contains the RF1A radio core register settings using the output format from the Texas Instruments SmartRF Studio. The structure (struct RF_SETTINGS) is defined in RF1A.h.

- RF1A.h and RF1A.c

  Header and c files for the RF1A radio core that contain basic function calls, including functions shown below.

```
unsigned char Strobe(unsigned char strobe);
void ResetRadioCore (void);
void WriteRfSettings(RF_SETTINGS *pRfSettings);
void WriteSingleReg(unsigned char addr, unsigned char value);
void WriteBurstReg(unsigned char addr, unsigned char *buffer, unsigned char count);
unsigned char ReadSingleReg(unsigned char addr);
void ReadBurstReg(unsigned char addr, unsigned char *buffer, unsigned char count);
```

- TX_RX_WOR.c

  Contains the main program, initializes and configures CC430 peripherals including RF1A core, and processes RF operations. Some of the key functions are shown below.

```
void GetRfMode(void); // Allows user to select RF mode via push button manipulation
void RfModeRX(void); //Sets up the RF1A in RX mode and puts the CPU core in LPM3 mode.
void RfModeTX(void); //Sets up the RF1A in TX mode, sends a packet every 10ms.
void RfModeWOR(void) //Sets up the RF1A in WOR mode and puts the CPU core in LPM3 mode.
{
  WriteSingleReg(WOREVT1, 0x87);
  WriteSingleReg(WOREVT0, 0x6B);
  WriteSingleReg(MCSM2, 0x00);
  WriteSingleReg(WORCTRL, (7 << 4) | 0);
  WriteSingleReg(IOCFG1, 0x29);             // GDO1 = RF_RDY
  while (1)
  {
    Strobe(RF_SWOR);
    wor = 1;
    RF1AIE |= ( (BIT6) << 8 );              // BIT14 = BIT6 << 8
    __bis_SR_register(LPM3_bits + GIE);
  }
}
```

- RF1A CC1101 Interrupt Handler example

  A simplified example of how to handle RX interrupt, TX interrupt, and interrupts for different stages of WOR is shown below.

```
#pragma vector=CC1101_VECTOR
__interrupt void CC1101_ISR(void)
{
  switch(__even_in_range(RF1AIV,32))        // Prioritizing Radio Core Interrupt
  {
    case  4:                                // GDO1 = LNA_PD signal
      RF1AIE &= ~(BIT1+BIT9);
      Strobe(RF_SWOR);                      // Go back to sleep
      break;
    case 20:                                // RFIFG9
      if (!transmitting)                    // RX end of packet
      {
        RF1AIE &= ~(BIT1+BIT9);
        // Read the length byte from the FIFO
        RxBufferLength = ReadSingleReg( RXBYTES );
        ReadBurstReg(RF_RXFIFORD, RxBuffer, RxBufferLength);
      }
      else                                  // TX end of packet
        RF1AIE &= ~BIT9;                    // Disable TX end-of-packet interrupt
      __bic_SR_register_on_exit(LPM3_bits);
      break;
    case 30:                                // WOR_EVENT0
      RF1AIE |= BIT9 + BIT1;
      RF1AIFG &= ~(BIT9 + BIT1);
      RF1AIES |= BIT9;                      // Falling edge of RFIFG9
      Strobe( RF_SRX );                     // Built-in 810us delay in Strobe()
      break;
    default:  break;
  }
}
```

5 References

1. *CC430 Family User's Guide* (SLAU259)

2. *CC1101 Low-Power Sub-1 GHz RF Transceiver* (SWRS061)

3. *AN047 -- CC1100/CC2500 Wake on Radio* (SWRA126)

4. *DN111 - Current Consumption for a Polling Receiver* (SWRA207)