Application Note

CA-8210 Low Power Modes

Release, Rev. 1.0, Jan 2017

Revision Control

Date	Contributors	Revision	Comments
13/01/17	W. Bruchner	1.0	Created

Table of Contents

Application Note

CA-8210 Low Power Modes

Release, Rev. 1.0, Jan 2017

1 Introduction

This application note gives an overview of the CA-8210 IEEE 802.15.4 transceiver modem low power modes, how they can be programmed and used in combination with a host microcontroller (MCU). It explains the different levels of low power modes, the wake-up conditions and the required steps to re-initialise the CA-8210 for normal operation. Code examples (in C) are given independently of the host MCU.

Several different low power modes have been implemented to allow different levels of savings in power consumption, coupled with different levels of wake-up conditions and data retention.

2 Accessing the CA-8210 Low Power Modes via the SPI interface API

As low power modes and their handling are not part of the IEEE 802.15.4 specification, they are implemented in the device-specific part of the API, namely the CA-8210 Hardware Management Entity (HWME). All low power modes can be programmed by using the HWME-SET request command with the attribute POWERCON (00H). As HWME-SET request is a synchronous command, a HWME-GET confirm is immediately returned and gives information if the command was successful (i.e all parameters were valid). After issuing the HWME-SET confirm, and if there are no outstanding scheduled tasks, the CA-8210 enters the programmed low power mode until a wake-up condition is fulfilled. Note that the POWERCON attribute is a 'write-only' attribute (it can be set by a HWME-SET set request, but cannot be read by a HWME-GET request), as (except for active mode) the chip would not be able to respond after entering the specified low power mode until the wake-up condition is triggered.

When waking up from a low power mode the CA-8210 asserts the NIRQ interrupt and sends a HWME-WAKEUP indication to signal to the MCU that it is ready for normal operation again.

The POWERCON attribute has 5 bytes of parameters: Byte 1 is the low power mode in combination with the corresponding wake-up condition(s). Bytes 2 to 5 are the 32-bit value of the on-chip sleep timer wake-up time in [ms] if this is used as a wake-up condition.

Attribute	Byte	Value	Description
POWERCON	1	00H	Active
		10H	Active – Use Sleep Timer
		04H	Standby – Wake-Up by System Reset only
		14H	Standby – Wake-Up by Sleep Timer
		24H	Standby – Wake-Up by GPIO Activity
		34H	Standby – Wake-Up by Sleep Timer or GPIO Activity
		0AH	Power-Off Mode 0 – Wake-Up by System Reset only
		1AH	Power-Off Mode 0 – Wake-Up by Sleep Timer
		2AH	Power-Off Mode 0 – Wake-Up by GPIO Activity
		3AH	Power-Off Mode 0 – Wake-Up by Sleep Timer or GPIO Activity
		1CH	Power-Off Mode 1 – Wake-Up by Sleep Timer
	2-5	00000000H - 07FFFFFFH	Sleep Timer Wake-Up time in [ms]

Except when wake-up by sleep timer is used as wake-up condition, bytes 2 to 5 have no effect and can be set to 0.

3 Low Power Mode Levels

This section explains the different low power modes which are implemented on the CA-8210.

Mode	On-Chip Supplies	Data Retention	16 MHz Clock Status	Wake-Up Conditions	Wake-Up Latency [ms]	Supply Current [uA]
Active	On	Yes	CPU Idle modes internally controlled	All Interrupts	-	200
Standby	Standby	Yes	Crystal Oscillator off, all clocks halted	Sleep Timer, GPIO Activity	0.7	10
Power-Off 0	Off except I/O	No	Crystal Oscillator off, all clocks halted	Sleep Timer, GPIO Activity	4.3	2
Power-Off 1	Off	No	Crystal Oscillator off, all clocks halted	Sleep Timer	4.6	0.2

The supply current figures are approximate values for comparison only. For exact values refer to the CA-8210 data sheet, section 2. The value for active mode is with radio disabled (transmitter and receiver off) and the co-processor idle.

The figures for wake-up latency are approximate values measured from the wake-up condition occurring to the insertion of the interrupt (NIRQ) for the HWME-WAKEUP indication.

3.1 Mode 0: Active

This is the default mode and strictly speaking this is not a low power mode, as all enabled peripherals and clocks can be active. The code on the on-chip co-processor manages the power consumption. The radio transmitter or receiver can be on or off, controlled by the higher layer protocol levels over the IEEE 802.15.4 MLME and MCPS MAC functions in the API. If the internal co-processor (CPU) has no tasks queued up, the CA-8210 goes into idle mode (co-processor core and memory access are switched off) until it is woken up by an interrupt such as an incoming SPI packet from the external MCU (which is the SPI master), or a valid received 802.15.4 radio packet. The 16MHz crystal oscillator is active. All data is retained. This default mode is always entered after waking up from a low power mode, and does not have to be re-programmed.

A facility exists to use the sleep timer externally when programming the parameter byte 1 of the POWERCON attribute to 10H (Active – Use Sleep Timer) and waiting for a wake-up indication which is issued on timeout. However, this should only be accessed when there are no transactions scheduled (no radio packets are queued to be transmitted).

3.2 Mode 1: Standby

Standby mode is the first level of low power modes. In this mode the 16MHz Oscillator is switched off, and all on-chip supplies are put into a low power standby mode. Radio packets cannot be transmitted or received. The wake-up condition can be programmed to be either a system reset, sleep-timer timeout, activity on all digital GPIOs (such as the SPI SSB pin), or both sleep-timer and GPIO activity. All data is retained in Standby mode.

3.3 Mode 2: Power-Off 0

Power-Off 0 mode is the second level of low power modes. In this mode the 16MHz Oscillator is switched off, and all on-chip supplies except the I/O supply are switched off. Radio packets cannot be transmitted or received. The wake-up condition can be programmed to be either a system reset, sleep-timer timeout, activity on all digital GPIOs (such as the SPI SSB pin), or both sleep-timer and GPIO activity. Data is not retained, and must be re-initialised after wake-up.

3.4 Mode 3: Power-Off 1

Power-Off 1 mode is the third level of low power modes and the mode with the lowest power consumption. In this mode the 16MHz Oscillator is switched off and all on-chip supplies are switched off. Radio packets cannot be transmitted or received. The chip can only be woken up by the internal sleep-timer timeout. Data is not retained, and must be re-initialised after wake-up.

Note that in most modes the NIRQ pin is driven by the CA-8210. However, in Power-Off 1 mode the I/O output drivers as well as pull-ups are disabled. In order to prevent incorrect triggering on NIRQ interrupt the corresponding MCU GPIO pin should be using a pull-up when using this mode. If this is not possible, an external pull-up (100k) should be added to the board.

4 Low Power Modes and Higher-level Protocol Stacks

Radio packet scheduling (when to transmit and receive packets over air) is usually performed by the higher-level software stacks running above the IEEE802.15.4 MAC layer. Although when packets are to be transmitted is usually obvious, the scheduling for the reception of packets depends very much on which higher-level stack is used. Some stacks give an indication when the receiver can be turned off (i.e. by setting the MAC PIB attribute macRxOnWhenIdle to 0 or using the MLME-RX-ENABLE request command). The time when no packets are to be transmitted and the receiver is switched off can be used for putting the CA-8210 into low power modes and therefore saving energy. Note that some wake-up latency has to be expected, and non-default PIB entries have to be re-initialised after using a low power mode with no data retention.

5 Code Examples for Low Power Modes

This section provides code examples in C to demonstrate the implementation of low power modes. Only functions handling the CA-8210 low power modes are described here. Other functions, such as the MCU and platform specific BSP_* functions are omitted. Their functionality is explained in the associated comments.

The function MCU_PowerDown() is an example of how to put the system (MCU and CA-8210) into low power mode. It also implements the re-initialisation after wake-up. 3 different low power modes are implemented:

- PDM_STANDBY: The CA-8210 is put into Standby mode (mode 1) and the MCU is powered down. A MCU timer serves as wake-up interrupt, and the MCU wakes up the CA-8210. CA-8210 data is retained.
- PDM_POWERDOWN: The CA-8210 is put into Power-Off 0 mode (mode 2) and the MCU is powered down. A MCU timer serves as wake-up interrupt, and the MCU wakes up the CA-8210. CA-8210 data is not retained and has to be re-initialised.
- PDM_POWEROFF: The CA-8210 is put into Power-Off 1 mode (mode 3) and the MCU is powered down. The CA-8210 sleep timer serves as wake-up condition, and the CA-8210 wakes up the MCU. CA-8210 data is not retained and has to be re-initialised.

The function first checks if the CA-8210 receiver is permanently enabled (the MAC PIB Attribute macRxOnWhenIdle has been set), and disables it during power-down if necessary. Then CA-8210 and the MCU are powered down. After wake-up and code continuation it is checked that the CA-8210 has woken up correctly, system parameters are re-initialised, and the CA-8210 MAC PIB is re-initialised if necessary.

Note that all synchronous API functions such as MLME_SET_request_sync() are returning the status value of the corresponding confirm and can be checked for successful completion. However, these checks have been omitted in most places in the example code in order to improve readability.

```
/*****************************************************************************/
/*****************************************************************************/
/****** MCU_PowerDown()                                                  ******/
/*****************************************************************************/
/****** Brief: Sends both CA8210 and MCU into Low Power Mode             ******/
/******        Uses timer time-out as Wake-Up Condition                  ******/
/*****************************************************************************/
/****** Power-Down Modes (mode Enumerations):                            ******/
/******                                                                  ******/
/****** Enumeration          CA8210 LP Mode       Wakeup Condition       ******/
/******                                                                  ******/
/****** 0: PDM_ACTIVE        Active               None                   ******/
/****** 1: PDM_STANDBY       Standby              MCU Timer              ******/
/****** 2: PDM_POWERDOWN     Power-Down 0         MCU Timer              ******/
/****** 3: PDM_POWEROFF      Power-Down 1         CA8210 Sleep Timer     ******/
/******                                                                  ******/
/*****************************************************************************/
/****** Param:  mode: low power mode                                     ******/
/******         sleeptime_sec: sleep timer timeout in [seconds]          ******/
/*****************************************************************************/
/****** Return: -                                                        ******/
/*****************************************************************************/
/*****************************************************************************/
void MCU_PowerDown(u8_t mode, u16_t sleeptime_sec)
 {
 u8_t length;
 u8_t attr;
 u8_t rxon;

 // check parameters
 if((mode == PDM_ACTIVE) || (mode > PDM_POWEROFF) || (sleeptime_sec == 0))
  return;

 // get MAC PIB macRxOnWhenIdle attribute to check if CA8210 receiver is on
 MLME_GET_request_sync(macRxOnWhenIdle, 0, &length, &rxon);
 // disable receiver if it is on (macRxOnWhenIdle is set)
 if(rxon)
  {
  attr = 0;
  MLME_SET_request_sync(macRxOnWhenIdle, 0, 1, &attr);
  }

 // power down CA8210
 CA8210_PowerDown(mode, sleeptime_sec);

 // set MCU timer
 if(mode != PDM_POWEROFF)
  BSP_SetWakeUpTimer(sleeptime_sec);

 // power down MCU
 // this function also has to make sure that all GPIOs are configured
```

```
// correctly and that the wake-up interrupt on the GPIO used for NIRQ
// is enabled
BSP_PowerDown();
// continuation after wake-up from here

// wake up CA8210
CA8210_Wakeup(mode);

// re-initialise MCU setup
BSP_SystemInit();

// re-initialise CA8210 MAC PIB if necessary
if((mode == PDM_POWEROFF) || (mode == PDM_POWERDOWN))
 CA8210_InitPIB();

// re-enable CA8210 receiver if MAC PIB macRxOnWhenIdle was set
if(rxon)
 MLME_SET_request_sync(macRxOnWhenIdle, 0, 1, &rxon);

} // End of MCU_PowerDown()
```

The following function, CA8210_PowerDown(), implements the HMWE-SET request for the POWERCON attribute which puts the CA-8210 into the desired low power mode.

```
/*****************************************************************************/
/*****************************************************************************/
/****** CA8210_PowerDown()                                                ******/
/*****************************************************************************/
/****** Brief:  Program CA8210 Low Power Mode                             ******/
/*****************************************************************************/
/****** Param:  mode: low power mode                                      ******/
/******         sleeptime_sec: sleep timer timeout in [seconds]           ******/
/*****************************************************************************/
/****** Return: -                                                         ******/
/*****************************************************************************/
/*****************************************************************************/
void CA8210_PowerDown(u8_t mode, u16_t sleeptime_sec)
 {
 u8_t pdparam[5];
 u32_t sleeptime_ms;

 // power down mode (POWERCON first byte)
 if(     mode == PDM_POWEROFF)   pdparam[0] = 0x1C;  // power-off mode 1, wake-up by sleep timer
 else if(mode == PDM_POWERDOWN)  pdparam[0] = 0x2A;  // power-off mode 0, wake-up by gpio (ssb)
 else if(mode == PDM_STANDBY)    pdparam[0] = 0x24;  // standby mode,     wake-up by gpio (ssb)
 else                            return;             // nothing to do

 // sleep timer wake-up time in [ms]
 sleeptime_ms = 1000*(u32_t)sleeptime_sec;
 pdparam[1] = LS0_BYTE(sleeptime_ms);
 pdparam[2] = LS1_BYTE(sleeptime_ms);
 pdparam[3] = LS2_BYTE(sleeptime_ms);
 pdparam[4] = LS3_BYTE(sleeptime_ms);

 // send HWME-SET request and check HWME-SET confirm
```

```
 if(HWME_SET_request_sync(HWME_POWERCON, 5, pdparam))
  BSP_Error();

 } // End of CA8210_PowerDown()
```

The function CA8210_Wakeup() wakes the CA-8210 from the low power mode by asserting the SPI chip select signal if required. It checks that the chip has woken up correctly and re-initialises some hardware-related register default values with the API function TDME-ChipInit().

```
/*******************************************************************************/
/*******************************************************************************/
/****** CA8210_Wakeup()                                                   ******/
/*******************************************************************************/
/****** Brief:  CA8210 Wake-Up from Low Power Mode                         ******/
/*******************************************************************************/
/****** Param:  -                                                         ******/
/*******************************************************************************/
/****** Return: -                                                         ******/
/*******************************************************************************/
/*******************************************************************************/
void CA8210_Wakeup(u8_t mode)
 {
 u8_t condition;

 // wake up CA8210 by GPIO by toggling the SSB chip select signal of the SPI interface
 // this avoids having to connect a seperate i/o purely for wake-up
 if((mode == PDM_STANDBY) || (mode == PDM_POWERDOWN))
  {
  // set GPIO for SSB low
  BSP_SetRFSSBLow();
  // wait at least 50 us as signal change has to pass CA8210 de-glitch filter
  BSP_Wait_us(50);
  // set GPIO for SSB high
  BSP_SetRFSSBHigh();
  }

 // wait for HWME-WAKEUP indication coming from CA8210
 // signal error if no response after a timeout of 10 ms
 if(CA8210_WaitforResponse(SPI_HWME_WAKEUP_INDICATION, 10))
  {
  BSP_Error();
  return;
  }

 // re-initialise CA8210 chip if no data retention
 if((mode == PDM_POWEROFF) || (mode == PDM_POWERDOWN))
  TDME_ChipInit();

 } // End of CA8210_Wakeup()
```

The function CA8210_InitPIB() gives an example of how to (re-)initialise the CA-8210 MAC PIB and should be used when exiting a low power mode without data retention. For most MAC configurations only few essential PIB attributes have to be re-initialised. This usually includes basic network parameters such as the IEEE802.15.4 channel, PANId and device addresses.

Here the parameters APP_CHANNEL, APP_PANID, APP_LONGADDRESS and APP_SHORTADDRESS are assumed to be global variables (or defines) passed from higher layers to the MAC layer.

```
/*****************************************************************************/
/*****************************************************************************/
/****** CA8210_InitPIB()                                                 ******/
/*****************************************************************************/
/****** Brief: (Re-)Initialisation of MAC PIB                            ******/
/*****************************************************************************/
/****** Param:  -                                                        ******/
/*****************************************************************************/
/****** Return: -                                                        ******/
/*****************************************************************************/
/*****************************************************************************/
void CA8210_InitPIB(void)
 {
 u16_t ptime;
 u8_t  param;

 MLME_RESET_request_sync(1)    // reset MAC PIB, SetDefaultPIB = TRUE

 MLME_SET_request_sync(phyCurrentChannel,  0, 1, &APP_CHANNEL);       // set Channel
 MLME_SET_request_sync(macPANId,           0, 2, &APP_PANID);         // set local PANId
 MLME_SET_request_sync(nsIEEEAddress,      0, 8, APP_LONGADDRESS);    // set local long address
 MLME_SET_request_sync(macShortAddress,    0, 2, &APP_SHORTADDRESS);  // set local short address

 } // End of CA8210_InitPIB()
```

6 References

[1] IEEE Std 802.15.4™-2006: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs)

[2] IEEE Std 802.15.4™-2003: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs)

[3] Cascoda Ltd, “IEEE802.15.4 Transceiver CA-8210 Datasheet”,
URL: https://www.cascoda.com/products/ca-8210/datasheet/

Cascoda Ltd
1 Venture Road
Southampton Science Park
Southampton, SO16 7NP
United Kingdom
Tel.: +44 (0) 2380 111797
Email: info@cascoda.com