Power Datasheet PWR V 0.2

Copyright © 2005-2012 Cypress Semiconductor Corporation. All Rights Reserved.

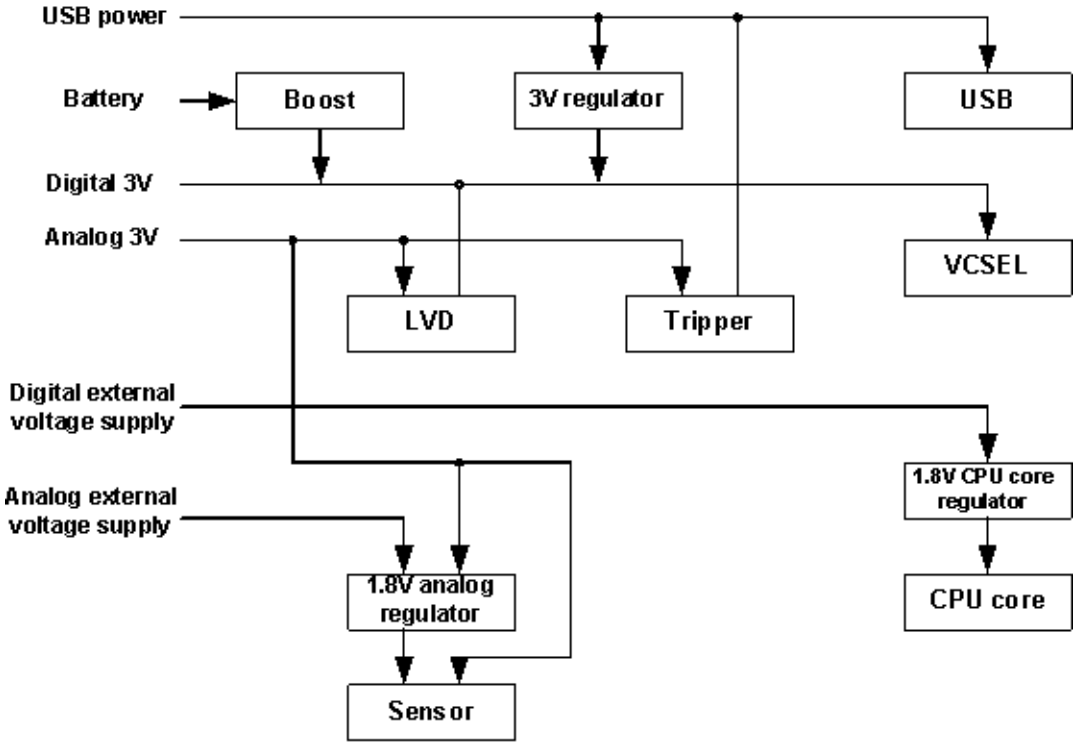
Resources	API Memory (Bytes)		Pins (per External I/O)
	Flash	RAM	
CYONS2000, CYONS2001, CYONS2010, CYONS2011, CYONS2100, CYONS2101, CYONS2110, CYONSFN2051, CYONSFN2053, CYONSFN2061, CYONSFN2151, CYONSFN2161, CYONSFN2162, CYONSTB2010, CYONSTB2011			
	1207	17	-

Features and Overview

- Power switchover detecting input voltage change
- Single or dual cell battery supply
- An internal Current DAC is used for estimating battery charge

The Power User Module, in combination with the on board full speed USB on CYONS2xxx devices provides a one chip solution for driving a USB mouse. Wired, wireless, and hybrid mouse solutions are all supported. Wireless applications are supported by a power system capable of operating from a battery. The Power User Module supports hybrid mice by allowing the mouse to seamlessly transition from battery powered wireless operation to USB powered wired operation and vice versa.

Figure 1. Figure 1. Power System Block Diagram



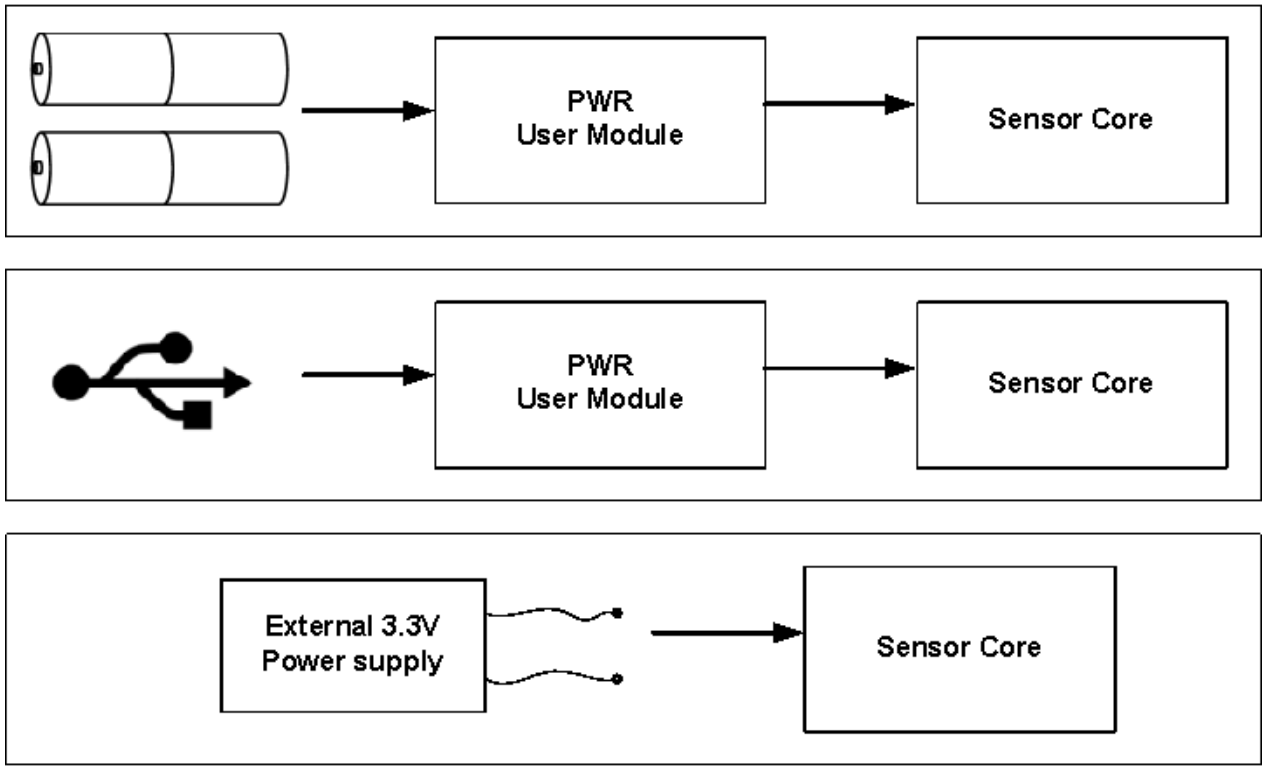
Functional Description

The purpose of the Power User Module is to give you access to the unique power supply features of the CYONS2xxx system. The CYONS2xxx is capable of operating from one or two batteries or from a USB interface. It can also run off of a fixed 3.3 volt supply.

The device can be configured for one of the following:

- Wired (power is supplied by the USB bus)
- Wireless (battery power supply for radio application)
- Dual wired/wireless (battery and USB power are hot swappable)
- External 3.3V supply

Figure 2. Application Diagram for Radio Wireless and Wired USB Mouse Devices



Power System Modes

Wired mode. In this mode, CYONS2xxx is powered only from VDD5V. In a typical USB application, this pin is connected to V_{BUS}, the USB 5V power pin. In wired operation, the 3.3V regulator is active. Selection of wired mode in the PWR User Module Parameter area causes the 3.3V regulator to be active by default.

The boost regulator should be off in this configuration. Selection of wired mode in the PWR User Module Parameter area causes the boost regulator to be disabled by default. However, it is the designer's responsibility to ensure that BOOST_GND, BOOST_IN, and BATT signals are connected to DVSS.

Wireless mode. In this mode, the CYONS2xxx is powered only from the boost regulator using the BOOST_IN power input. In a typical wireless operation, BOOST_IN is connected to an inductor, BOOST_GND to the battery's negative terminal, and BATT directly to the battery output.

In wireless mode, the boost regulator should be enabled, and the 3.3V regulator should be disabled. Both of these are set by default when wireless mode is selected in the PWR User Module Parameter area. However, for proper operation, the designer must connect VDD5V to DVSS.

External power supply mode. In this mode, both of the internal regulators are disabled, and the chip is powered from a clean, regulated, external supply.

In external mode, both the boost regulator and the 3.3V regulator should be disabled. Selection of external mode in the PWR User Module Parameter area disables the two regulators by default. For proper operation, the designer must connect BOOST_GND, BOOST_IN, BATT, and VDD5V to DVSS.

Wired and Wireless mode.In this mode, the CYONS2xxx is allowed to transition between wired and wireless operation, by dynamically changing the configuration of the two regulators in response to which power sources are present.

If both sources are present, then the boost regulator is disabled, and the chip is powered from the VDD5V input. If the VDD5V input is removed, the UM responds to the trip interrupt by enabling the boost regulator. When the VDD5V input is replaced, the UM switches operation back to the 3.3V regulator, to preserve battery life.

Hybrid Operation

When the battery is inserted, the boost turns on. The CPU has an opportunity to monitor the battery insertion and configure the boost through the boost registers to make the boost go into sleep mode. When USB supply is ramped down on detecting the tripper going to 0, core is issued a wakeup interrupt. The boost is switched on and is in the active state. The 1.8V regulator is turned off. Power down signals are asserted to the analog core and analog part enable is deasserted. If the core is in sleep mode it wakes up and provides the 8 MHz clock for digital blocks. The tripper interrupt is asserted and is propagated to core.

When USB supply is ramped up, on detecting the tripper going to 1, the core is issued a wake up interrupt. The 1.8V regulator is turned off.

Figure 3. Figure 3. Power Mode State Diagram for Wired Switchover

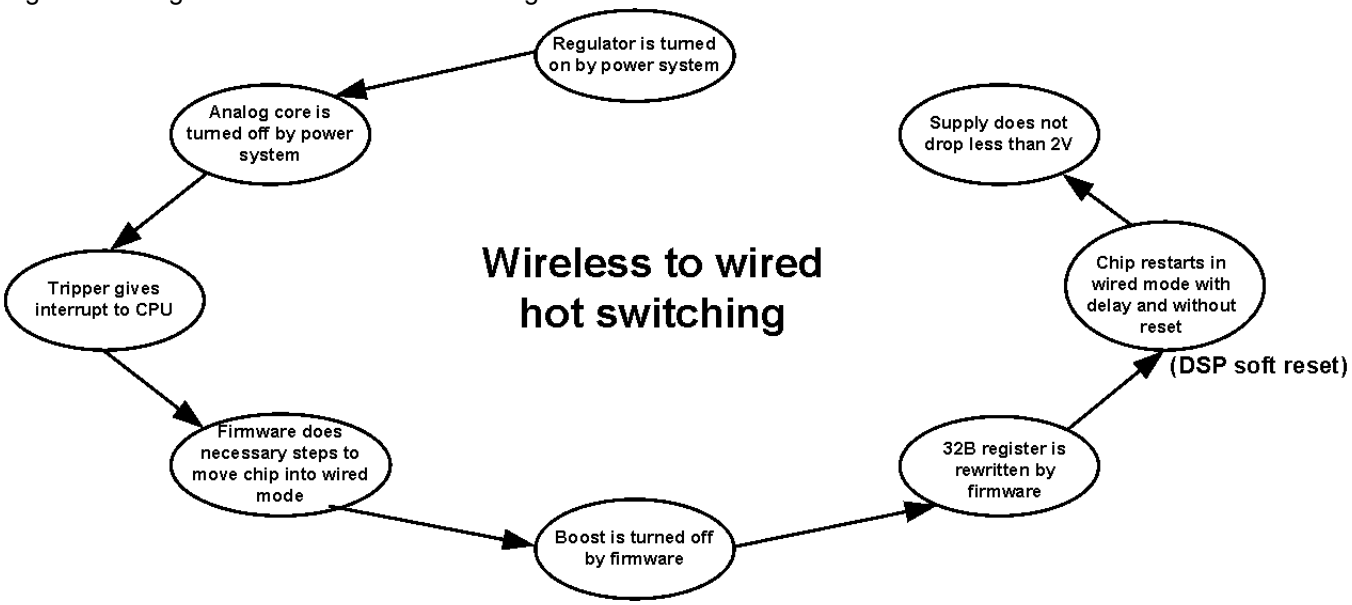
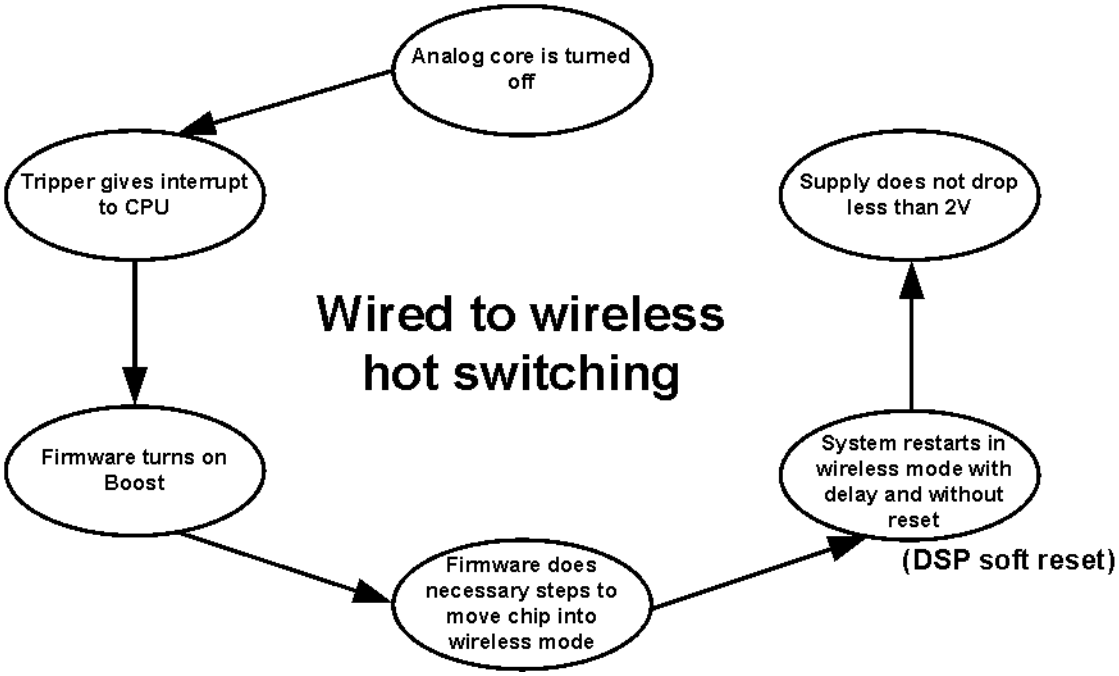


Figure 4. Figure 4. Power Mode State Diagram for Wired Switch



Boost Regulator

The Boost regulator is a device that allows conversion of low voltage (for example, battery) inputs to 3.3V operating levels.This block architecture uses a single inductor and internal rectification to increase or decrease voltage from input to output as required. An inductor is the only additional external component required to implement the converter. External bypass capacitors on Vdd are also required - these capacitors are the same components required when the device is powered by a conventional power supply but are specified more completely to assure proper operation. Rectification is achieved internally rather than requiring additional external Shottky diodes.

3.3V Regulator

This is a linear voltage regulator designed to convert a 5.0V USB supply voltage to 3.3V (in wired applications). The regulator input voltage can vary between 4V and 5.5V. The nominal regulator output is 3.3V. It has been designed such that it never exceeds 3.6V. In wired applications, the Boost circuit is turned off and the regulator supplies the power. The USB pin is used to switch power domain from boost to regulators. When USB power is not supplied in a wireless application, the USB pin is pulled down (1000 kΩ).

DC and AC Electrical Characteristics

See the device datasheet for your Ovation device for the electrical characteristics of the Navigation block.

Placement

The Power User Module can be placed in the dedicated block only.

Parameters and Resources

Only the Power system configuration parameter must be set.

Power System Configuration

This parameter chooses power source and actually sets device configuration. The available choices vary by device. Possible choices are:

Parameter	CYONS2000, CYONS2100, CYONS2010, CYONSTB2010	CYONS2001, CYONS2101, CYONS2011, CYONSTB2011	CYONS2110, CYONSFN2162	CYONSFN2051, CYONSFN2053, CYONSFN2061,CYONSFN2151, CYONSFN2161
5V USB	● (default)		●	
3.3V supply	●	●	●	●
Single cell battery		●	●	●
Dual cell battery		● (default)	●	● (default)
5V USB/Single cell battery			●	
5V USB/Dual cell battery			● (default)	

Application Programming Interface

The Application Programming Interface (API) functions are provided as part of the user module to allow you to deal with the module at a higher level. This section specifies the interface to each function together with related constants provided by the include files.

Note

****** In this, as in all user module APIs, the values of the A and X register may be altered by calling an API function. It is the responsibility of the calling function to preserve the values of A and X before the call if those values are required after the call. This "registers are volatile" policy was selected for efficiency reasons and has been in force since version 1.0 of PSoC Designer. The C compiler automatically takes care of this requirement. Assembly language programmers must also ensure their code observes the policy. Though some user module API functions may leave A and X unchanged, there is no guarantee they may do so in the future.

For Large Memory Model devices, it is also the caller's responsibility to preserve any value in the CUR_PP, IDX_PP, MVR_PP, and MVW_PP registers. Even though some of these registers may not be modified now, there is no guarantee that will remain the case in future releases.

Not all API functions are supported for all devices. The following table defines the applicability of each API for each part number. • - Available on this part.

Table 1. API Applicability vs. Part Number

APIs	CYONS2000, CYONS2100, CYONS2010, CYONSTB2010	CYONS2001, CYONS2101, CYONS2011, CYONSTB2011	CYONS2110, CYONSFN2162	CYONSFN2051, CYONSFN2053, CYONSFN2061, CYONSFN2151, CYONSFN2161
PWR_Start()	●	●	●	●
PWR_Stop()	●	●	●	●
PWR_5VRegOn()	●		●	
PWR_5VRegOff()	●		●	
PWR_BoostOn()		●	●	●
PWR_BoostOff()		●	●	●
PWR_5VRegTripEnableInt()			●	
PWR_5VRegTripDisableInt()			●	
PWR_iReadBattMonitor()		●	●	●
PWR_BoostUnderVoltageEnableInt()		●	●	●
PWR_BoostUnderVoltageDisableInt()		●	●	●
PWR_fHad5VRegTrip()			●	

APIs	CYONS2000, CYONS2100, CYONS2010, CYONSTB2010	CYONS2001, CYONS2101, CYONS2011, CYONSTB2011	CYONS2110, CYONSFN2162	CYONSFN2051, CYONSFN2053, CYONSFN2061, CYONSFN2151, CYONSFN2161
PWR_fHadBoostUnderVoltage()		●	●	●
PWR_fCheckTripperOutput()			●	
PWR_fCheckLvdOutput()	●	●	●	●
PWR_fCheckStartupDone()	●	●	●	●
PWR_fCheckInitDone()	●	●	●	●
PWR_fCheckBoostVoltageReached()		●	●	●
PWR_EnableUSBSupplyWake()	●		●	
PWR_DisableUSBSupplyWake()	●		●	
PWR_ShutDown()	●	●	●	●
PWR_Startup()	●	●	●	●

PWR_Start

Description:

Provides initial configuration: turns on boost or 3.3V linear regulator and initializes battery supply.

C Prototype:

```
void  PWR_Start(void);
```

Assembly:

```
lcall  PWR_Start
```

Parameters:

None

Return Value:

None

Side Effects:

See Note ** at the beginning of the API section

PWR_Stop

Description:

Turns off boost and 3.3V linear regulator.

C Prototype:

```
void  PWR_Stop(void);
```

Assembly:

```
lcall  PWR_Stop
```

Parameters:

None

Return Value:

None

Side Effects:

See Note ** at the beginning of the API section

PWR_5VRegOn

Description:

Turns on the 5V regulator circuitry.

C Prototype:

```
void  PWR_5VRegOn(void);
```

Assembly:

```
lcall  PWR_5VRegOn
```

Parameters:

None

Return Value:

None

Side Effects:

See Note ** at the beginning of the API section

PWR_5VRegOff

Description:

Turns off the 5V regulator circuitry.

C Prototype:

```
void  PWR_5VRegOff(void);
```

Assembly:

```
lcall  PWR_5VRegOff
```

Parameters:

None

Return Value:

None

Side Effects:

See Note ** at the beginning of the API section

PWR_BoostOn

Description:

Turns on the boost circuitry.

C Prototype:

```
void PWR_BoostOn(void);
```

Assembly:

```
lcall PWR_BoostOn
```

Parameters:

None

Return Value:

None

Side Effects

See Note ** at the beginning of the API section

PWR_BoostOff

Description:

Turns off the boost circuitry.

C Prototype:

```
void PWR_BoostOff(void);
```

Assembly:

```
lcall PWR_BoostOff
```

Parameters:

None

Return Value:

None

Side Effects

See Note ** at the beginning of the API section

PWR_5VRegTripEnableInt

Description:

Enables the 5V regulator trip interrupt.

C Prototype:

```
void PWR_5VRegTripEnableInt(void);
```

Assembly:

```
lcall PWR_5VRegTripEnableInt
```

Parameter:

None

Return Value:

None

Side Effects:

See Note ** at the beginning of the API section

PWR_5VRegTripDisableInt

Description:

Disables the 5V regulator trip interrupt .

C Prototype:

```
void PWR_5VRegTripDisableInt(void);
```

Assembly:

```
lcall PWR_5VRegTripDisableInt
```

Parameters:

None

Return Value:

None

Side Effects:

See Note ** at the beginning of the API section

PWR_iReadBattMonitor

Description:

Returns scaled DAC code of battery charge state

C Prototype:

```
INT PWR_iReadBattMonitor(void);
```

Assembly:

```
lcall PWR_iReadBattMonitor
;now you can read result through X(MSB) and A(LSB)
```

Parameters:

None

Return Value:

Returns scaled DAC code of battery charge state or corresponding error code through X (MSB) and A (LSB) registers.

Returned Value	Description
-4	DAC code at 0.8V is not defined
-3	DAC code at 1.2V is not defined
-2	DAC code of battery charge estimation is not defined
-1	Zero or negative slope is derived during calculation
[0;32767]	Corresponding code of battery charge state

Side Effects:

See Note ** at the beginning of the API section

PWR_BoostUnderVoltageEnableInt

Description:

Enables Boost under-voltage interrupt.

C Prototype:

```
void PWR_BoostUnderVoltageEnableInt(void);
```

Assembly:

```
lcall PWR_BoostUnderVoltageEnableInt
```

Parameters:

None

Return Value:

None

Side Effects:

See Note ** at the beginning of the API section

PWR_BoostUnderVoltageDisableInt

Description:

Disables Boost under-voltage interrupt.

C Prototype:

```
void PWR_BoostUnderVoltageDisableInt(void);
```

Assembly:

```
lcall PWR_BoostUnderVoltageDisableInt
```

Parameters:

None

Return Value:

None

Side Effects:

See Note ** at the beginning of the API section

PWR_fHad5VRegTrip

Description:

Checks 5V tripper interrupt status bit. Clears status bit if it is set.

C Prototype:

```
BOOL PWR_fHad5VRegTrip(void);
```

Assembly:

```
lcall PWR_fHad5VRegTrip
```

Parameters:

None

Return Value:

Returns (through A register) true if tripper circuit interrupt status bit is set, false - otherwise.

Side Effects:

See Note ** at the beginning of the API section

PWR_fHadBoostUnderVoltage

Description:

Checks boost under-voltage interrupt status bit.Clears status bit if it is set.

C Prototype:

```
BOOL PWR_fHadBoostUnderVoltage(void);
```

Assembly:

```
lcall PWR_fHadBoostUnderVoltage
```

Parameters:

None

Return Value:

Returns (through A register) true if tripper circuit interrupt status bit is set, false - otherwise.

Side Effects:

See Note ** at the beginning of the API section

PWR_fCheckTripperOutput

Description:

Indicates the filtered tripper output.

C Prototype:

```
BOOL PWR_fCheckTripperOutput(void);
```

Assembly:

```
lcall PWR_fCheckTripperOutput
```

Parameters:

None

Return Value:

Returns (through A register) true if USB power is above 3V Tripper trip point, returns false if USB power goes below 3V Tripper point.

Side Effects:

See Note ** at the beginning of the API section

PWR_fCheckLvdOutput

Description:

Indicates the filtered LVD signal output.

C Prototype:

```
BOOL PWR_fCheckLvdOutput(void);
```

Assembly:

```
lcall PWR_fCheckLvdOutput
```

Parameters:

None

Return Value:

Returns (through A register) true if power supply is above LVD trip point, returns false if power is below LVD trip point.

Side Effects:

See Note ** at the beginning of the API section

PWR_fCheckStartupDone

Description:

Indicates the power system startup is done (after a power switching).

C Prototype:

```
BOOL PWR_fCheckStartupDone(void);
```

Assembly:

```
lcall PWR_fCheckStartupDone
```

Parameters:

None

Return Value:

Returns (through A register) true if power system startup bit is set, false otherwise.

Side Effects:

See Note ** at the beginning of the API section.

PWR_fCheckInitDone

Description:

Indicates if initial power up done.

C Prototype:

```
BOOL PWR_fCheckInitDone(void);
```

Assembly:

```
lcall PWR_fCheckInitDone
```

Parameters:

None

Return Value:

Returns (through A register) true if initial power up bit is set false otherwise.

Side Effects:

See Note ** at the beginning of the API section

PWR_fCheckBoostVoltageReached

Description:

Indicates if battery is absent or is in low level condition.

C Prototype:

```
BOOL PWR_fCheckBoostVoltageReached(void);
```

Assembly:

```
lcall PWR_fCheckBoostVoltageReached
```

Parameters:

None

Return Value:

Returns (through A register) true if low battery condition bit is set false otherwise.

Side Effects:

See Note ** at the beginning of the API section

PWR_EnableUSBSupplyWake

Description:

Enables the wakeup to Krypton when USB supply switches.

C Prototype:

```
void PWR_EnableUSBSupplyWake(void);
```

Assembly:

```
lcall PWR_EnableUSBSupplyWake
```

Parameters:

None

Return Value:

None

Side Effects:

See Note ** at the beginning of the API section

PWR_DisableUSBSupplyWake

Description:

Disables the wakeup to Krypton when USB supply switches.

C Prototype:

```
void PWR_DisableUSBSupplyWake(void);
```

Assembly:

```
lcall PWR_DisableUSBSupplyWake
```

Parameters:

None

Return Value:

None

Side Effects:

See Note ** at the beginning of the API section

PWR_ShutDown

Description:

Initiates power down sequence of power system.

C Prototype:

```
void PWR_ShutDown(void);
```

Assembly:

```
lcall PWR_ShutDown
```

Parameters:

None

Return Value:

None

Side Effects:

See Note ** at the beginning of the API section

PWR_Startup

Description:

Brings up the power system.

C Prototype:

```
void PWR_Startup(void);
```

Assembly:

```
lcall PWR_Startup
```

Parameters:

None

Return Value:

None

Side Effects:

See Note ** at the beginning of the API section

Sample Firmware Source Code

The C code illustrated here reads the battery voltage on the VBATT pin. Not all devices support the iReadBattMonitor API.

The C code illustrated here reads the battery voltage on the VBATT pin:

```
    int iBatteryVoltage;

    iBatteryVoltage = PWR_iReadBattMonitor();

// iBatteryVoltage now holds integer value of 100*voltage. For example, if
// the battery voltage is 1.13 volts, iBatteryVoltage will equal 113.
```

The Assembly code illustrated here reads the battery voltage on the VBATT pin. Not all devices support the iReadBattMonitor API.

```
export _main

area bss(RAM)          ; inform assembler that variables follow
iBatteryVoltage: blk 2 ; battery voltage variable
area text
_main:

; Insert your main assembly code here.


LCALL _PWR_iReadBattMonitor
; call function to read battery voltage
; upon return, stack contains high and low bytes of battery voltage * 100. ; For example,
if the battery voltage is 3.06 volts, the scaled battery
; voltage will equal 306. The value at the stack pointer will contain the
; lower byte of this value, ie 0x32, and the value at SP+1 will contain the
; upper byte of this value, ie 0x01.

mov X, SP
mov A, [X]
;LOW byte should be specified as:
mov [iBatteryVoltage+1], A

inc x
```

```
mov A, [X]
;HIGH byte of the variable should be specified as:
mov [iBatteryVoltage], A
```

Configuration Registers

The following registers are configured in this UM. Symbolic names for these registers are defined in the user module instance C and assembly language interface files (the *.h* and *.inc* files).

Table 2. CONTROL_REGISTER1, Bank 3

Bit	7	6	5	4	3	2	1	0
Value	Reserved				boost mode	extmode	clksel1	clksel0

boost mode - This bit enables boost operation.

extmode - This bit enables external mode control.

clksel1 - Clock frequency select 1.

clksel0 - Clock frequency select 0.

Table 3. INTR_MASK_REG, Bank 2

Bit	7	6	5	4	3	2	1	0
Value	Reserved			USBPwrWake Mask	StChgd Mask	Reserved	TripFilOut Mask	vcselErr Mask

USBPwrWake Mask - This bit is the USB power supply interrupt mask. When set to 1, this bit disables the interrupt generation on USB power supply switch.

StChgd Mask - This bit is the mask for StChgd. This mask bit does not affect the status register (StChgd status) assertion. It only affects the PSoC core interrupt assertion.

TripFilOut Mask - Whenever power supply switching happens from wireless to wired, this signal toggles from 0 to 1. Whenever power supply switching happens from wired to wireless, this signal toggles from 1 to 0. During both the transitions, interrupt is set.

vcselErr Mask - Whenever this interrupt is set, it means that VCSEL is ON for more than the desired time interval. This signal makes a transition from 0 to 1 to indicate error.

Version History

Version	Originator	Description
0.2	DHA	Added Version History
0.2.b	DHA	Updated the "Configuration Registers" section according to the latest Ovation ONS II Technical Reference Manual.

Note PSoC Designer 5.1 introduces a Version History in all user module datasheets. This section documents high level descriptions of the differences between the current and previous user module versions.

Copyright © 2005-2012 Cypress Semiconductor Corporation. The information contained herein is subject to change without notice. Cypress Semiconductor Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in a Cypress product. Nor does it convey or imply any license under patent or other rights. Cypress products are not warranted nor intended to be used for medical, life support, life saving, critical control or safety applications, unless pursuant to an express written agreement with Cypress. Furthermore, Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress products in life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

PSoC Designer™ and Programmable System-on-Chip™ are trademarks and PSoC® is a registered trademark of Cypress Semiconductor Corp. All other trademarks or registered trademarks referenced herein are property of the respective corporations.

Any Source Code (software and/or firmware) is owned by Cypress Semiconductor Corporation (Cypress) and is protected by and subject to worldwide patent protection (United States and foreign), United States copyright laws and international treaty provisions. Cypress hereby grants to licensee a personal, non-exclusive, non-transferable license to copy, use, modify, create derivative works of, and compile the Cypress Source Code and derivative works for the sole purpose of creating custom software and or firmware in support of licensee product to be used only in conjunction with a Cypress integrated circuit as specified in the applicable agreement. Any reproduction, modification, translation, compilation, or representation of this Source Code except as specified above is prohibited without the express written permission of Cypress.

Disclaimer: CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Cypress reserves the right to make changes without further notice to the materials described herein. Cypress does not assume any liability arising out of the application or use of any product or circuit described herein. Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress' product in a life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

Use may be limited by and subject to the applicable Cypress software license agreement.