OMAP5910 Low-Power System Design

Thanh Tran, Ph.D. and Mike Blaskovich *DSP/EEE Catalog, OMAP Applications*

ABSTRACT

The OMAP5910 is a true system-on-a-chip device, which consists of ARM925T MPU and C55X DSP cores. The device has many advanced power management modes that enable system designers to develop a very low-power, multimedia communication device. This application note shows low-power design techniques and outlines the steps required to put the OMAP™ device in the lowest power consumption mode and to wake it up. Also, system hardware design, a low-power code example, and experimental results are being demonstrated.

This application report contains project code that can be downloaded from http://www.ti.com/lit/zip/SPRA954.

Contents

List of Figures

List of Tables

1 Introduction

This document is intended to demonstrate a low-power system design utilizing the OMAP5910 dual-core processor and the TPS65010 power and battery management device. This document is divided into four different parts, where the first part covers the OMAP5910 power management capabilities, the second part provides an overview of the TPS65010 device and how it is being interfaced with the OMAP, the third part shows the hardware system design which includes schematics, and finally, the last part gives a code example to put the OMAP5910 in the lowest power consumption mode and to wake it up. The last part also includes power measurements for different operating conditions within the OMAP itself, e.g., running an algorithm on the DSP while doing nothing on the ARM, shutting down the traffic controller, and so on.

2 Design Description

The OMAP5910 has multiple low-power modes (awake, big sleep and deep sleep); transitions between these different modes are being handled by the ultra low-power module (ULPD), internal to the OMAP5910. As shown in Figure 1, the OMAP5910 device architecture consists of the C55x DSP, ARM925, System DMA, traffic controller, integrated SRAM, and many peripherals; refer to *OMAP5910 Dual-Core Processor Data Manual* (literature number SPRS197) and *OMAP5910 Dual-Core Processor Technical Reference Manual* (literature number SPRU602) for more detailed description of these modules. For power management, the ULPD is the most critical module. It performs the most important task, that is, controlling the entire device by setting all the power states and clock domains associated with each individual module.

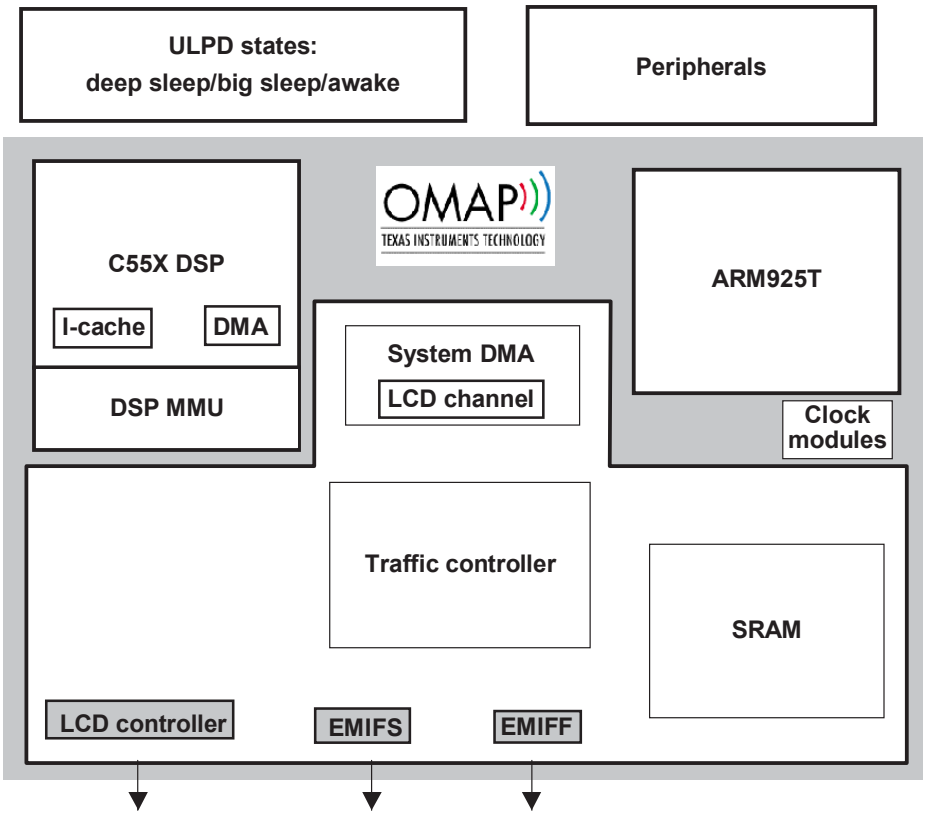

Figure 1. OMAP5910 Device Architecture

Figure 2 demonstrates the communication between the OMAP5910 and the power management device to transition to different power states (Deep Sleep, Big Sleep, or Awake).

As shown in Figure 2, the ULPD drives the LOW_PWR signal to indicate that it is going into deep sleep. The TPS65010 power device responds by lowering the core voltage to reduce the power consumption. ULPD de-asserts the LOW_PWR to go back to the awake state.

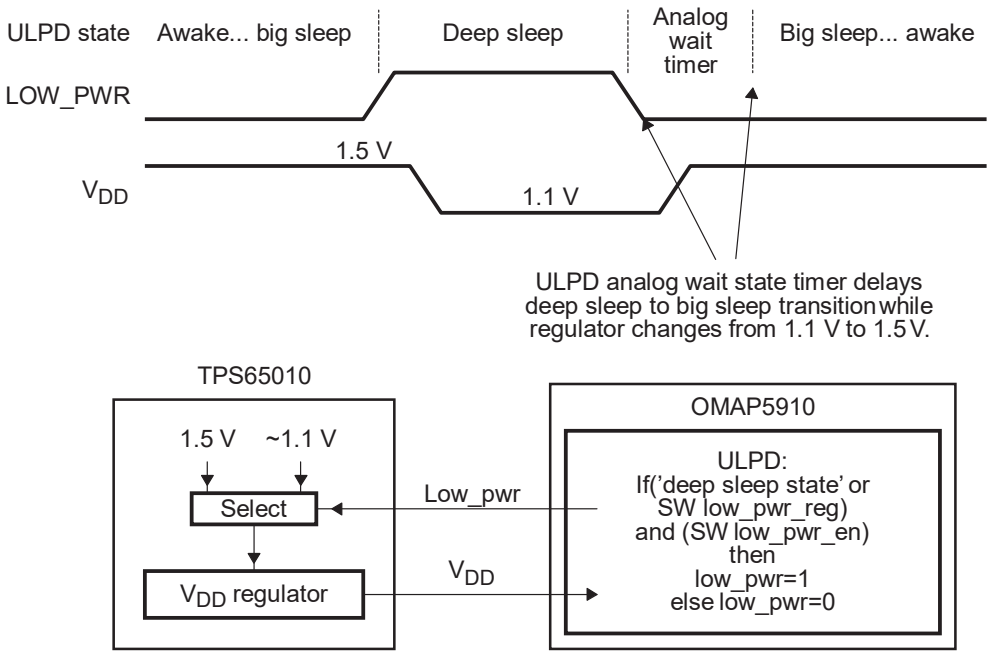

Figure 2. UPLD Characteristics

2.1 Low-Power System Architecture

Figure 3 shows the architecture of the design and how the interface between the power management device TPS65010 and the OMAP5910 is implemented. The low-power system consists of two main parts: the OMAP5910 processor and the TPS65010 power and battery management IC. Other components such as system memory, USB, keyboard, mouse, Ethernet and other peripherals are not included in this application note.

The TPS65010 provides two highly efficient step-down converters targeted at providing the core voltage (VCC_CORE) and I/O rails (VCC_IO), and two integrated LDOs allowing designers to create low noise voltage outputs to drive noise-sensitive circuits. One important feature is that the OMAP can drive the LOW_PWR control pin to lower the core voltage while it is transitioning to deep sleep mode.

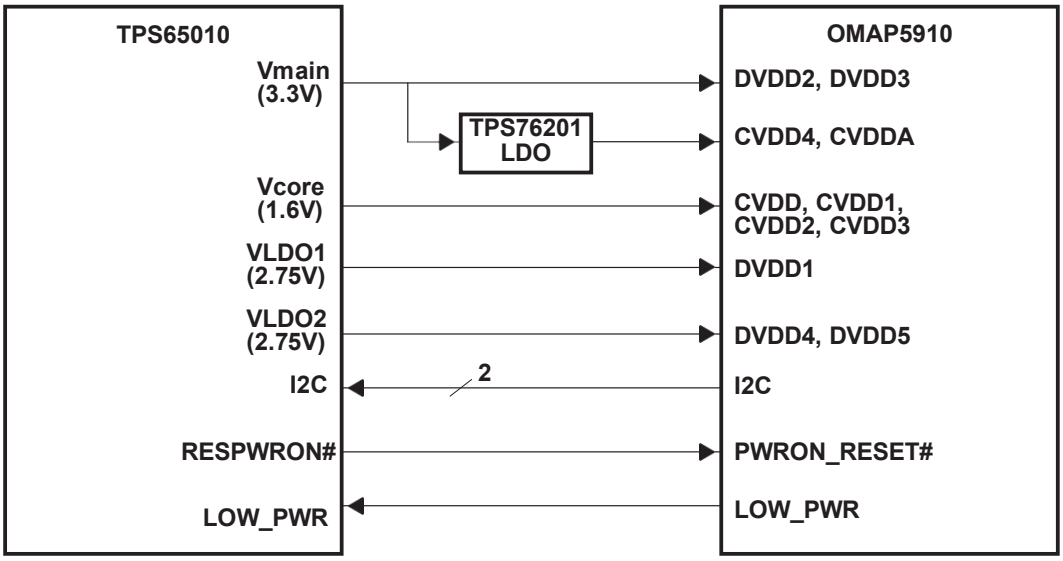

Figure 3. OMAP Low-Power System Architecture

2.2 OMAP5910 Power Management

As mentioned in the Introduction, the ULPD block of the OMAP design is responsible for the power management, which includes performing the power on reset, managing system reset, calibrating the 32-kHz oscillator, managing the low-power mode transitions, performing the wake-up of the external oscillator, performing the 12-MHz/32-kHz switch, and other control functions. Refer to *OMAP5910 Dual-Core Processor Data Manual* (literature number SPRS197) and *OMAP5910 Dual-Core Processor Technical Reference Manual* (literature number SPRU602) for detailed discussion of the configurations.

2.2.1 Power On Reset and Reset Management

The PWRON_RESET# signal is an active-low asynchronous reset input responsible for the reset of the entire OMAP5910 device. When using an external crystal oscillator for the 32 kHz clock, the PWRON_RESET# must be asserted low a minimum of two 32-kHz clock cycles longer than the worst-case start-up time of the oscillator. Internally, the ULPD synchronizes the RESPWRON input signal and generates two internal functional resets called **nreset** and **omapnrst**. At power on reset, both nreset and omapnrst are low. AWAKE state is achieved after 1024 32-kHz clock cycles. nreset is released first after 20 more clock cycles of the 12-MHz clock input. After nreset is released, the omapnrst waits for 30 additional 12-MHz clock cycles before going inactive.

$$T_{min} = (T_{osc-max} + 2 \times 1/32\ \text{kHz}) + ((1024 \times 1/32\ \text{kHz}) + (20 \times 1/12\ \text{MHz})) + (30 \times 1/12\ \text{MHz})$$
$$= T_{osc-max} + 0.0321\ \text{sec}$$

where T_{min} is the minimum OMAP wakeup time from the power on reset for 12 MHz input clock and $T_{osc-max}$ is the maximum startup time for the external 32 kHz oscillator.

2.2.2 OMAP5910 Low-Power Modes

- Big sleep (12 MHz and 32 kHz are on and digital phase-locked loop or DPLL1 off), second lowest power consumption
- Deep sleep (only 32 kHz on, all other components off), lowest power consumption

In the big sleep mode, the 12-MHz and 32-kHz clocks are running. The 12-MHz clock is distributed only to the 48-MHz DPLL and to one, or both, of the 12-MHz output pins (MCLK and/or BCLK). The 48-MHz DPLL and the 48-MHz clock may also be distributed on-chip to a requesting peripheral. The ARM and DSP are in their idle modes.

Deep sleep mode is the lowest power state of the OMAP5910. In this mode, the OMAP5910 processor is operating solely on its 32-kHz input clock. The ULPD has no requests for a 12-MHz or 48-MHz clock and, therefore, it will shut down the 12-MHz oscillator. In this mode, the ARM and DSP are in their idle modes and all on-chip PLLs are off. As shown in Figure 1, the ULPD can signal the entry of deep sleep mode by asserting the *LOW_PWR* signal (this is an alternate pin-multiplexing of armio_5) automatically. This indication forces the external voltage regulator, TPS65010, to go to standby or to lower the core voltage to 1.1V. While in deep sleep mode, the 32-kHz clock provides the necessary on-chip clocking to the peripherals that can wake up the processor (such as UART2, the keypad, the MPUIOs). Also, there are two external hardware requests (MCLKREQ and BCLKREQ) that would cause an exit from the deep sleep state, when one or both of the requests are active. Figure 4 shows a flowchart of the low-power mode transitions.

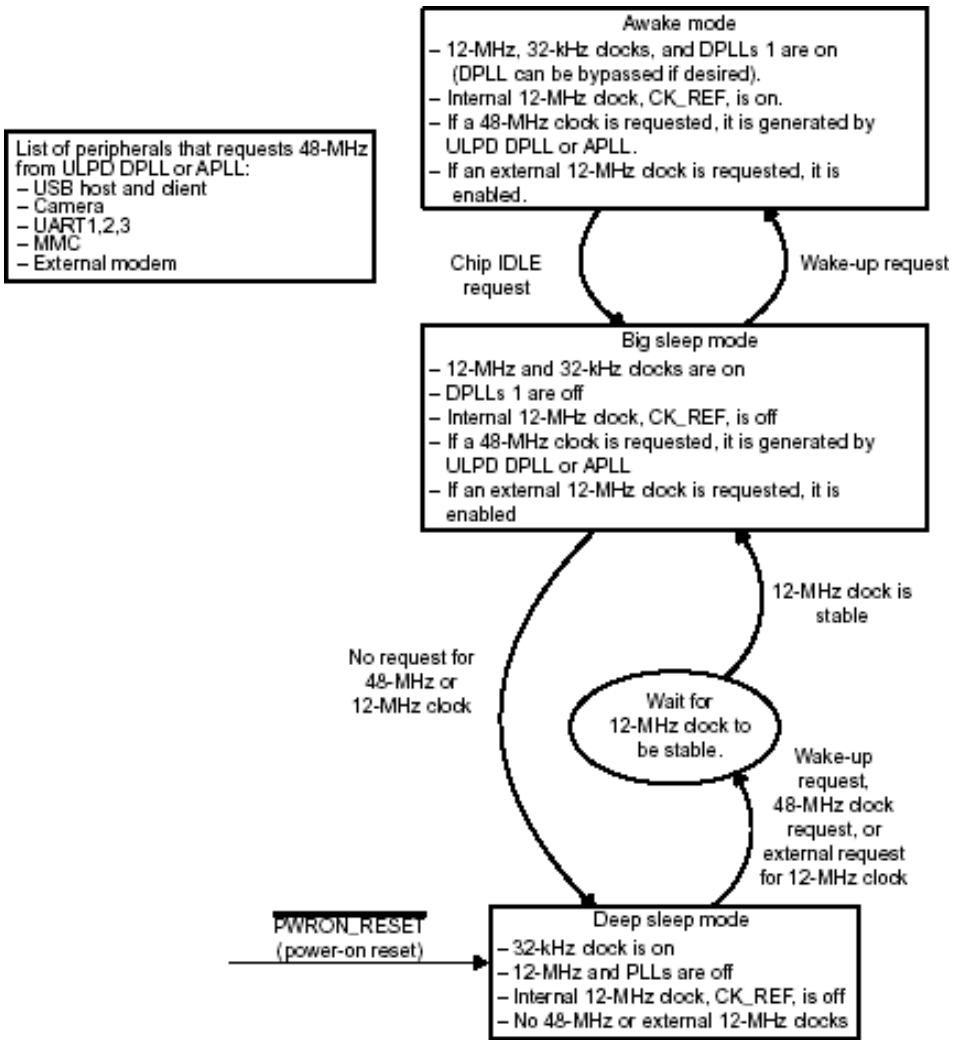

Figure 4. Power Management Mode Transitions

2.3 TPS65010 Power and Battery Management Device

The TPS65010 [5] consists of two high-efficiency, step-down converters and two 200mA LDOs. The two switchers, Vmain (3.3V) and Vcore (1.6), generate voltage outputs that are higher than 90% and 80% efficient at 100mA level, respectively [5]. The disadvantages of using switchers in the design are radiated switching noise and higher output ripple, which may interfere with other noise sensitive circuits, such as audio, video, SDRAM, and phase-locked loop (PLL).

In this application, the LDOs are being utilized to drive SDRAM, FLASH, analog PLL and digital PLL sections of the OMAP. The switcher outputs are only connected to the core CPU and the I/O voltages as shown in Figure 2.

2.3.1 TPS65010 Low-Power Mode

The configuration registers are accessible via I2C bus. To enable the low-power mode, the enable low-power bit (ENABLE_LP, bit 3 of the VDCDC1 register) must be set and the LOW_PWR pin must be driven high by the OMAP. Internally, the TPS65010 uses the rising edge of the internal signal formed by a logical AND of the LOW_PWR and ENBLE_LP signals before entering the low power mode.

In this mode, the Vmain switching converter remains active but the Vcore converter may be disabled by setting the LP_COREOFF bit in the VDCDC2 register. If left-enabled, the Vcore voltage is set to the value predefined by the CORELP0/1 bits in the VDCDC2 register.

For the LDO outputs in low power mode, the LDO1OFF/nSLP and LDO2OFF/nSLP bits in the VREGS1 register determine whether the LDOs are turned off or put in a reduced power mode, where the internal transient speed-up circuitry is disabled in order to minimize the quiescent current.

Refer to [5] for detailed description of all the configuration registers and modes.

2.4 System Design

Figure 5, Figure 6, and Figure 7 show how the OMAP5910 power supply rails are connected to the power management device. This design only demonstrates the power and power management capabilities of the OMAP, so it is up to the designers to determine what other external components are required for implementing a specific application. Table 1 shows the signals associated with the power management interface.

Table 1. OMAP5910 and TPS65010 Interface

OMAP5910	TPS65010	Schematic Ref.	Description
CVDDA, CVDD4		U1300	Low noise supply for DPLL and APLL circuits
CVDD, CVDD1, CVDD2, CVDD3	Vcore	VCC_CPU	High efficient 1.6V core voltage
DVDD1	VLDO1	VCC_I/O	Low noise 2.75V I/O voltage
DVDD4, DVDD5	VLDO1	VCC_MEM	Low noise 2.75V memory voltage
DVDD2, DVDD3	Vmain	VCC_3V3	High efficiency 3.3V I/O voltage
I^2C Master	I^2C Slave		OMAP is the I^2C master
PWRON_RESET	$\overline{\text{RESPWRON}}$	nRESPWRON	Active low asynchronous reset
LOW_PWR	LOW_PWR	LOW_PWR	Deep Sleep indication driven by the OMAP5910

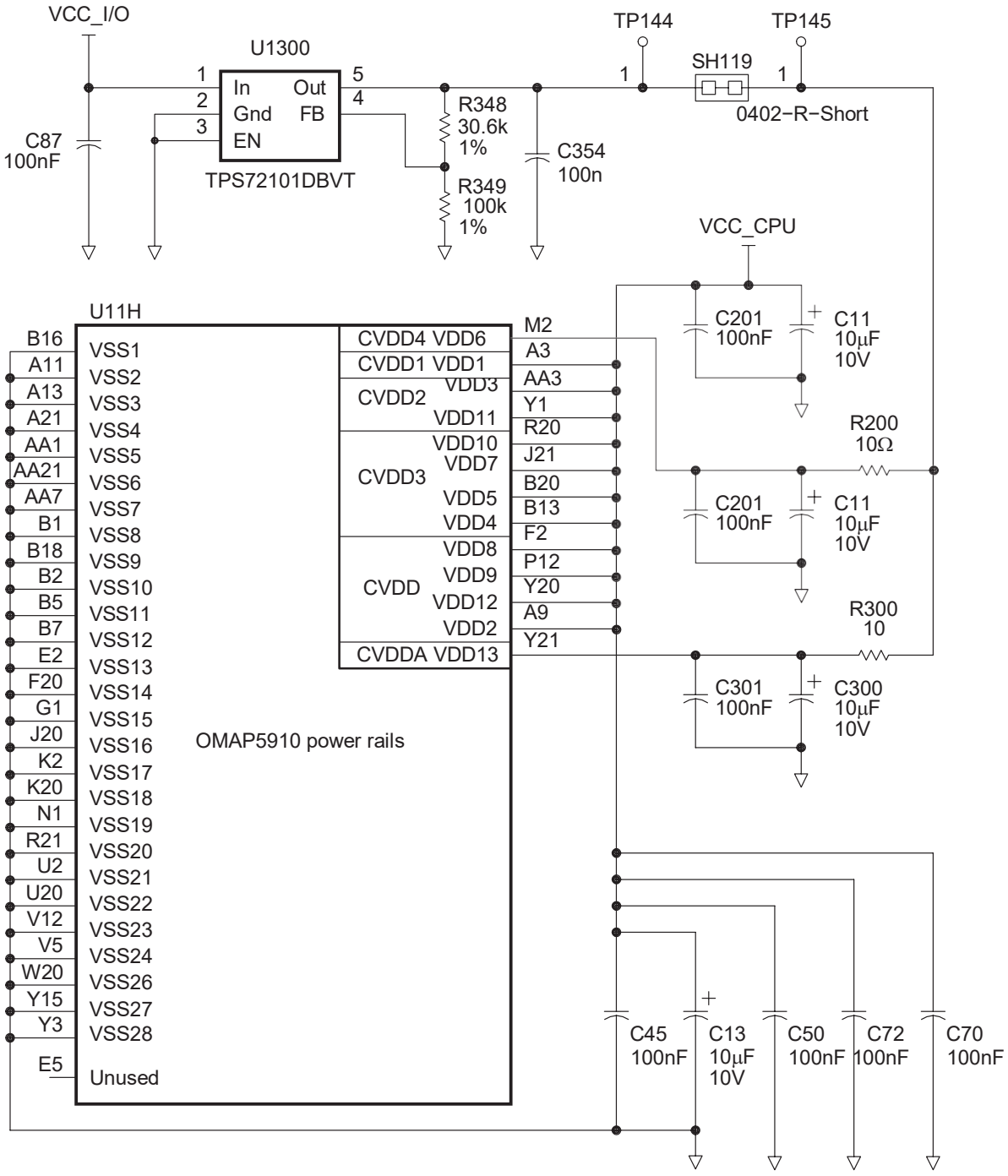

Figure 5. OMAP5910 Core/PLL Voltages

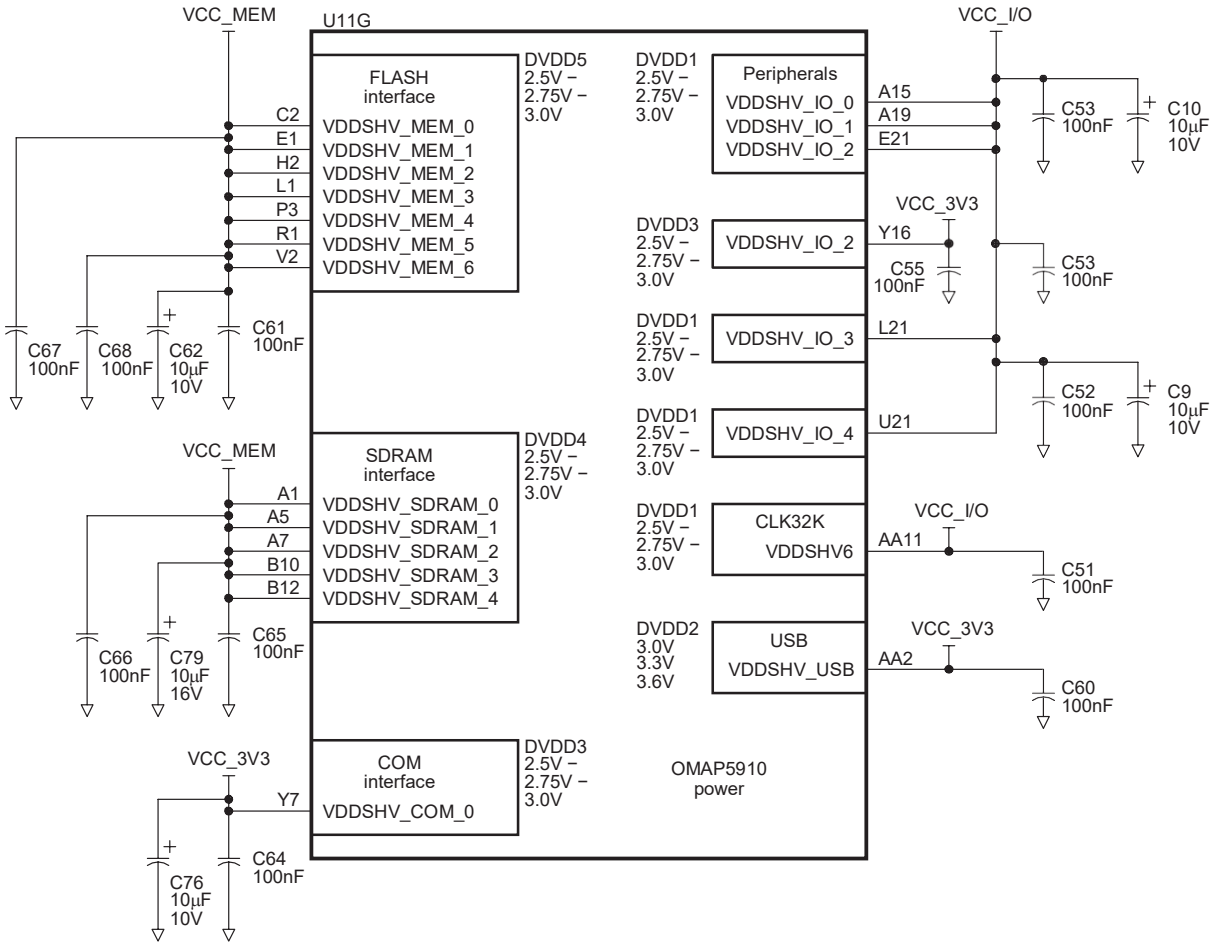

Figure 6. OMAP5910 I/O, Memory and USB Voltages

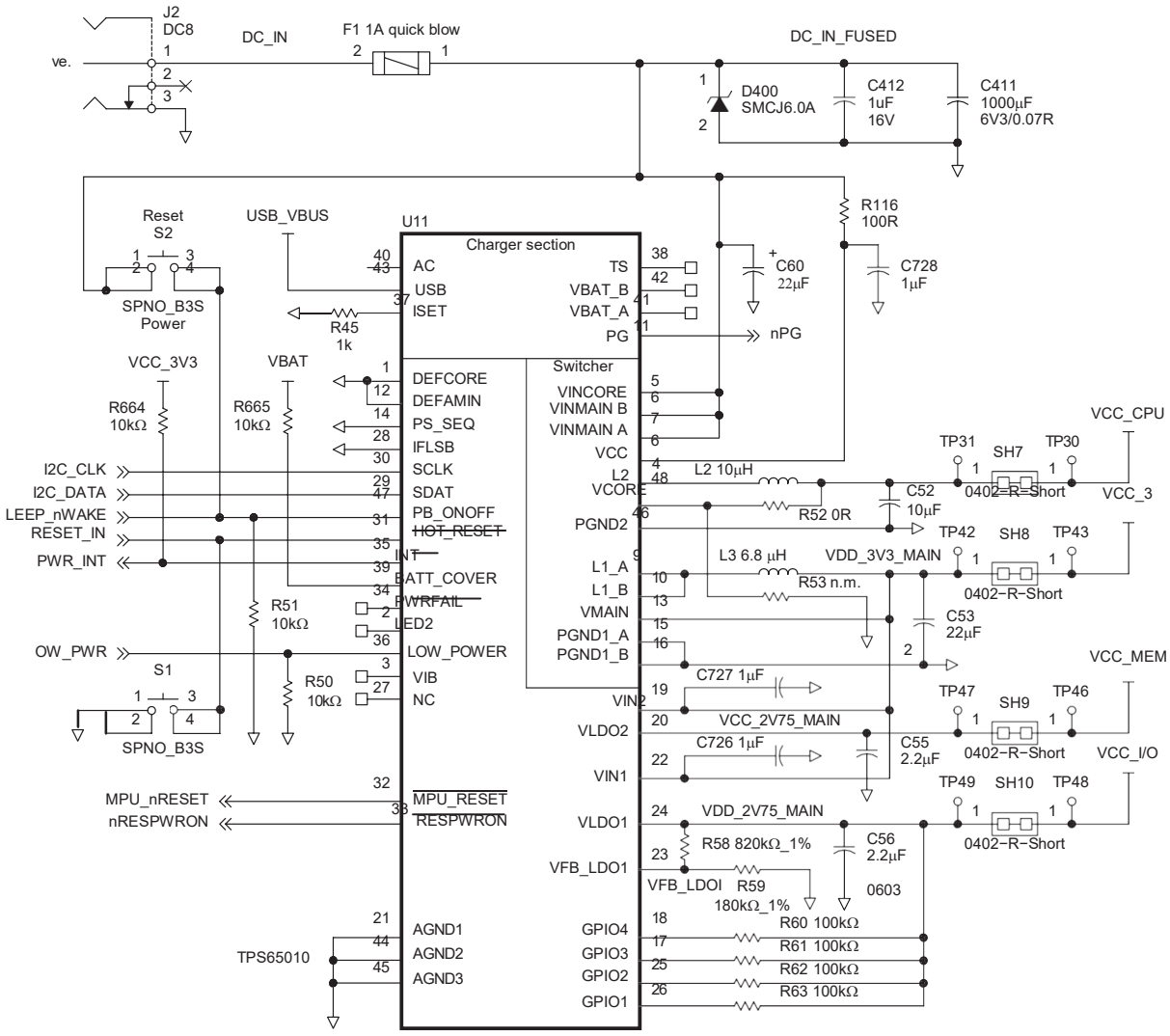

Figure 7. TPS65010 Power Management Circuit

2.5 System Software

The OMAP5910 deep sleep and awake software example was developed by Mike Blaskovich [3] for the Innovator Development Platform [4]. The code example shown in Appendix A requires the OMAP Code Composer Studio Version 2.1 or later to run. The entire project file is posted on the website (www.omap.com).

2.5.1 OMAP5910 Deep Sleep Program Flow

The sleep code example performs the following functions:

1. OMAP5910 start-up code
2. *LOW_PWR* signal enabling
3. OMAP5910 DSP shutdown
4. Wakeup event setup
5. ARM and traffic controller shutdown
6. Deep sleep entry
7. Wakeup by keypad on innovator

2.5.1.1 OMAP5910 Start-Up Code

The start-up code first places the OMAP5910 into its OMAP5910 pin-configuration mode by calling the Set5910Mode routine. For simplicity, each pin uses the default pin multiplexing. The OMAP5910 is then placed into synchronous scaleable mode. The write to the ARM_CKCTL register controls the clock divisors used by various portions of the OMAP5910. For example, the DSPMMU will receive the DPLL1 clock divided by 2 (72 MHz), the traffic controller (TC) will also receive a divided-by-2 clock (72 MHz), the ARM and DSP will receive the full 144 MHz, the LCD clock is 72 MHz, and the peripheral clock is 36 MHz. The write to ARM_IDLECT2 enables the distribution of clocks to the DMA, DSP, timers, etc.

```
Set5910Mode();          // Setup Configuration Space for 5910 Mode
    ARM_SYSST=0x1000;   // Set Sync-Scaleable Mode
    ARM_CKCTL=0x3506;   // DSPMMU/2,TC/2,ARM/1,DSP/1,LCD/2,PER/4
    DPLL1_CONTROL_REG=0x2610;              // Spin up to 144 MHz
    while (ARM_CKCTL&0x1==0);              // wait for DPLL lock
    ARM_IDLECT2 = 0x6FF;           // Start most clocks (to be disabled later)
    ARM_RSTCT2 = 0x0001;           // Remove Reset from ARM peripherals
```

2.5.1.2 LOW_PWR Signal Enabling

This section of code enables the *LOW_PWR* signal to be output on pin T20. The *LOW_PWR* signal is an alternate function of the *armio_5* signal. The pin-multiplexing for this pin is controlled by setting FUNC_MUX_CTRL_7<14:12>='001'. In addition to the pin-multiplexing change, the *LOW_PWR* signal must be enabled by a write to the ULPD register POWER_CTRL_REG at offset 0x50 in the ULPD register space.

```
ULPD_POWER_CTRL_REG = 0x0001; // Enable low_pwr signal output during Deep Sleep
     stmpval=FUNC_MUX_CTRL_7;              // Read Value from FUNC_MUX_CTRL_7
     stmpval &= 0xFFFF8FFF;                // clear bit locations 14:12
     stmpval |= 0x00001000;         // <14:12>='001' selects low_pwr signal on armio_5
     FUNC_MUX_CTRL_7 = stmpval;
```

2.5.1.3 OMAP5910 DSP Shutdown

The next segment of code places the OMAP5910's C55x DSP into idle mode. To stop the DSP, it is simply placed into reset and the DSP clock disabled. Additionally, the clocks to the DSP peripherals must all be stopped by setting the DSP_IDLECT2 register to 0x0.

```
ARM_RSTCT1 |= 0x2;            // Set DSP_EN=1, remove reset from DSP block
     ARM_RSTCT1 &= ~0x2;       // Set DSP_EN=0, put DSP block back into reset
     ARM_CKCTL  &= ~0x2000;    // Set EN_DSPCK=0, stop DSP block clock
     DSP_IDLECT2 = 0x0;        // Stop any DSP domain clocks
```

2.5.1.4 Wakeup Event Setup

This segment of code will enable the OMAP5910 to exit deep sleep mode by means of a wakeup event. The event that is used is the Innovator keypad. A wakeup event is simply an unmasked interrupt from one of the 32 kHz-enabled peripherals. Because the keypad interrupt is an edge-triggered event, it can simply be cleared by writing a '0' to the appropriate bit of the ITR register of the interrupt handler. A keypad press will provide an interrupt to bit-1 of the second level interrupt handler.

```
// Setup a wakeup event. Wake from any keypad press
     INTH2_ITR &= ~0x2;       // Clear any pending keypad interrupt
     INTH1_ITR &= ~0x1;       // Clear pending 2nd level interrupt
     INTH2_MASK=0xFFFFFFFD;          // Enable a keypad interrupt to wakeup device
     INTH1_MASK=0xFFFFFFFE;          // Permit 2nd level interrupt to wakeup device
```

2.5.1.5 ARM and Traffic Controller Shutdown

The next code segment prepares the ARM and traffic controller to enter deep sleep. If it is running, the ARM watchdog must be disabled.

```
WATCHDOG_TIMER_MODE=0x00F5;       // If running, disable ARM watchdog timer
     WATCHDOG_TIMER_MODE=0x00A0;
```

The remainder of the ARM shutdown has been done in an assembly routine named 5910DeepSleep. Rather than describing each line of the assembly code, the following steps describe the events that are occurring:

1. Clear EMIFS global power-down enable bit (PDE) in EMIFS_CONFIG register
2. Set IMIF power-down bit (PWD_EN) in EMIFS_CONFIG register
3. Set CLK, PWD and SLRF bits of SDRAM controller
4. Set EMIFS global power-down enable bit (PDE) in EMIFS_CONFIG register
5. Put DPLL in Bypass
6. Stop the GPIO (XOR), API, HSAB, LB, and LCD clocks if running

Steps 1−4 are required to shut down the traffic controller. This sequence of operations works around an errata that requires the PDE bit of the EMIFS_CONFIG register to be set to ‘0’ prior to beginning the shutdown of the traffic controller (reference OMAP5910 Errata). Setting PDE=1 must be the last step toward idling the traffic controller.

In step 3, setting the CLK bit of the SDRAM_CONFIG register tells OMAP5910 to stop the SDRAM clock during idle. PWD tells the SDRAM controller it can enter idle mode when the SDRAM controller is inactive. Setting the SLRF bit places the SDRAM into self-refresh.

Step 5 places DPLL1 into bypass. At this point, all OMAP5910 internal clocks are derived directly from the 12 MHz oscillator.

Step 6 is a write to the DSP_IDLECT2 register to manually assure that the clocks to the GPIO, API, high-speed access bus (HSAB), local bus (LB) and LCD. These clocks are not automatically stopped when the ARM Idle instruction is executed, so they must be stopped by this manual operation.

2.5.1.6 Deep Sleep Mode Entry

The entry to deep sleep mode is done by writing to the ARM_IDLECT1 register. This register controls two things: the entry of the ARM into its idle mode (bit 11, SETARM_IDLE); and a number of bits that cause a number of other clocks to idle in conjunction with the ARM transition into idle mode. The write to ARM_IDLECT1 with the SETARM_IDLE bit set to ‘1’ is the last instruction before entry to deep sleep mode. This instruction should be followed by 20 NOPs:

```
; Idle all of the ARM-based clocks and Go to sleep
      ldr    r0, arm_idlect1_reg               // arm_idlect1_reg is 0xFFFECE04
      ldr    r1, arm_idlect1_reg_sleepmask     // arm_idlect1_reg_sleepmask is 0x0EC7
      strhr1,    [r0]               // 16-bit write operation. OMAP enters Deep Sleep
      nop
      nop
      :
      :
```

2.5.1.7 Wakeup By Keypad on Innovator

Innovator has a black 4-way switch on the front. Touch this switch and the device will wake up. Code execution will resume in the nop instructions that followed the ARM_IDLECT1 write that caused entry into deep sleep. In a system with more than one wakeup source, it will be necessary to check the interrupt status bits to determine the source of the wakeup.

2.5.2 Running the Code Under Code Composer Studio

After building the code using Code Composer Studio and loading the program, the system is ready for test. The OMAP5910Sleep program runs in two segments with user intervention required between the two segments. It follows a flow where it will:

1. Put the OMAP5910 to sleep
2. Wake up the OMAP5910 after a keypad press

When you hit Run, the program will execute until the OMAP5910 is in deep sleep. The output window will display the following messages:

DSP Idled
(appears immediately after the DSP has been idled as described in section 2.5.1.3)

Processor about to enter Deep Sleep mode
(appears before calling 5910DeepSleep assembly routine described in section 2.5.1.6)

When deep sleep mode is entered, the program will still be running, but will appear to be “stuck” – it is waiting for a wakeup event. By pressing a button associated with the keypad, it will wake up and the following message will be displayed:

Processor Awake

To run the code again, simply select ‘Restart’ from the Debug pulldown menu of Code Composer Studio.

2.5.3 Observing Deep Sleep Entry

Two methods can be easily observed when the OMAP5910 enters deep sleep mode: the 12-MHz oscillator stops toggling AND the *LOW_PWR* signal is asserted. The 12-MHz oscillator restarts and the LOW_PWR gets deasserted as soon as the OMAP5910 exits deep sleep mode. In this deep sleep example code, it will be asserted immediately upon execution of the write to ARM_IDLECT1 as described in section 2.5.1.6. The *LOW_PWR* signal will remain asserted until there is a keypad press.

To observe LOW_PWR on an Innovator, connect a scope probe (or a multimeter) to either the break-out-board location B28, or observe it at resistor R43. The R43 resistor is located on the opposite side of the processor module from the OMAP5910. It is unmarked but is easily located (see Figure 8).

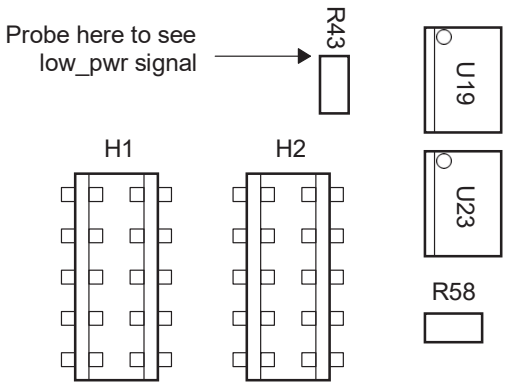

Figure 8. Innovator Settings for Deep Sleep

2.6 Experimental Results

The OMAP5910 power consumptions (core plus I/O) were thoroughly measured and the results are shown in Table 2 [6].

Table 2. OMAP5910 Current Consumption

Module	Measured Results	Frequency MHz	Total mA	Comments
DSP mA/MHz	0.575	150	86.3	DSP runs GSM FR vocoder from Icache, all DSP domains active, but with no DMA activity. Includes DSP MMU.
	0.457	150	68.6	DSP runs GSM FR vocoder from internal RAM. Includes DSP MMU.
MPU mA/MHz	0.310	150	46.5	MPU running EPOC
TC mA/MHz	0.191	75	14.3	TC, TIPB, and DMA clocks on (IDLIF_ARM = 0) and DMA auto-idle on.
CLKM mA/MHz	0.068	150	10.2	12-MHz oscillator, DPLL1, and clock circuits
Total with DSP program in external memory: 157.3 mA				
Total with DSP program in internal memory: 139.6 mA				

3 Conclusions

This application note provided an example for system engineers to design a low-power, multimedia communication device utilizing the OMAP5910 and the TPS65010 power management device. This document also provides a code example developed for the Innovator Development Platform to put the device in deep sleep and to wake it up. Finally, it includes experimental data, which show the low-power consumption associated with different modules within the OMAP5910 itself.

This design is only a proof-of-concept (not a complete reference design) to help designers to do system planning and power budgeting of the OMAP5910 and the power management device.

4 References

1. *OMAP5910 Dual-Core Processor Data Manual* (literature number SPRS197)
2. *OMAP5910 Dual-Core Processor Technical Reference Manual* (literature number SPRU602)
3. Mike Blaskovich*, OMAP1510 Deep Sleep Example Version 0.21,* Texas Instruments Inc., May 2003
4. Productivity Systems, Inc.*, Innovator Development Kit for the Texas Instruments OMAP Platform*

5. Texas Instruments Inc., *TPS65010 Power and Battery Management IC for Li-ION Powered Systems*
6. Texas Instruments Inc., *OMAP5910 Power Consumption Presentation,* May 2003

Appendix A OMAP5910 Code [3]

```
// OMAP5910sleep.c
// Mike Blaskovich
// Revision 1.2
//
// History:
//    Rev 1.0  put OMAP5910 to sleep
//    Rev 1.1  added wakeup from keypad
//    Rev 1.2  added additional comments
#include ”sleep1.h”
int main()
{
   unsigned int stmpval;
   void Set5910Mode();
/*****************************************************************************
*/
/* This block of code has nothing to do with putting the processor to sleep. It is
*/
/* starting clocks, putting the device into 5910 mode, removing peripherals from reset
*/
/*****************************************************************************
*/
   Set5910Mode();                  // Setup Configuration Space for 1510 Mode
   ARM_SYSST=0x1000;               // Set Sync-Scaleable Mode
   ARM_CKCTL=0x3506;               // DSPMMU/2,TC/2,ARM/1,DSP/1,LCD/2,PER/4
   DPLL1_CONTROL_REG=0x2610;       // Spin up to 144MHz
   while (ARM_CKCTL&0x1==0);       // wait for DPLL lock
   ARM_IDLECT2 = 0x6FF;            // Start most clocks (to be disabled later)
   ARM_RSTCT2 = 0x0001;            // Remove Reset from ARM peripherals
/*****************************************************************************
*/
/*****************************************************************************
*/
/* Make the low_pwr signal output on the armio_5 pin location
*/
/*****************************************************************************
*/
   ULPD_POWER_CTRL_REG = 0x0001;   // Enable low_pwr signal output during Deep Sleep
                                   // by bit0=1 in ULPD POWER_CTRL_REG (offset 0x50)
   stmpval=FUNC_MUX_CTRL_7;        // Read Value from FUNC_MUX_CTRL_7
   stmpval &= 0xFFFF8FFF;          // clear bit locations 14:12
   stmpval |= 0x00001000;          // <14:12>=’001’ selects low_pwr signal on armio_5
   FUNC_MUX_CTRL_7 = stmpval;
/*****************************************************************************
*/
/*****************************************************************************
*/
```

```
/* IDLE the DSP
*/
/*****************************************************************************************
*/
   ARM_RSTCT1 |= 0x2;               // Set DSP_EN=1, remove reset from DSP block
   ARM_RSTCT1 &= ~0x2;              // Set DSP_EN=0, put DSP block back into reset
   ARM_CKCTL  &= ~0x2000;           // Set EN_DSPCK=0, stop DSP block clock
   DSP_IDLECT2 = 0x0;               // Stop any DSP domain clocks
   printf("DSP Idled\n");
/*****************************************************************************************
*/
/*****************************************************************************************
*/
/* Setup a wakeup source. Wakeup only from a keypad press
*/
/*****************************************************************************************
*/
   INTH2_ITR &= ~0x2;               // Clear any pending keypad interrupt //
   INTH1_ITR &= ~0x1;               // Clear pending 2nd level interrupt //
   INTH2_MASK=0xFFFFFFFD;           // Enable a keypad interrupt to wakeup device //
   INTH1_MASK=0xFFFFFFFE;           // Permit 2nd level interrupt to wakeup device //
/*****************************************************************************************
*/
/* Shut down the ARM and TC domains
*/
/*****************************************************************************************
*/
   WATCHDOG_TIMER_MODE=0x00F5;      // If running, disable ARM watchdog timer
   WATCHDOG_TIMER_MODE=0x00A0;
   ULPD_CLOCK_CTRL_REG |= 0x20;     // Make sure USB Client Clock is stopped
   printf("Processor about to enter Deep Sleep mode\n");
   OMAPDeepSleep();

   /* If wakeup occurs, the next line will be executed */
   printf("Processor Awake\n");
}
/*****************************************************************************************
*/
/* Put device in OMAP5910 Mode. All pins with default pin muxing.                 */
/*****************************************************************************************
*/
void Set5910Mode()
{
  FUNC_MUX_CTRL_0   =   0x00000000;
  FUNC_MUX_CTRL_1   =   0x00000000;
  FUNC_MUX_CTRL_2   =   0x00000000;
  FUNC_MUX_CTRL_3 = 0x00000000;
```

```
  FUNC_MUX_CTRL_4     =  0x00000000;
  FUNC_MUX_CTRL_5     =  0x00000000;
  FUNC_MUX_CTRL_6     =  0x00000000;
  FUNC_MUX_CTRL_7     =  0x00000000;
  FUNC_MUX_CTRL_8     =  0x00000000;
  FUNC_MUX_CTRL_9     =  0x00000000;
  FUNC_MUX_CTRL_A     =  0x00000000;
  FUNC_MUX_CTRL_B     =  0x00000000;
  FUNC_MUX_CTRL_C     =  0x00000000;
  FUNC_MUX_CTRL_D     =  0x00000000;
  PULL_DWN_CTRL_0     =  0xFFFFFFFF;
  PULL_DWN_CTRL_1     =  0xD8FDC000;
  PULL_DWN_CTRL_2     =  0x00280C2B;
  PULL_DWN_CTRL_3     =  0xFFFC00FE;
  GATE_INH_CTRL_0     =  0x00000000;
  VOLTAGE_CTRL_0      =  0x00000007;
  TEST_DBG_CTRL_0     =  0x00000007;
  MOD_CONF_CTRL_0     =  0x0D000000;
  COMP_MODE_CTRL_0    =  0x0000EAEF;
}
```

A.1 Library

```
// sleep1.h
// Registers used by OMAP5910sleep.c
typedef unsigned short UInt16;
typedef unsigned int   UInt32;
#define ARM_CKCTL             *((volatile UInt16 *) 0xFFFECE00) /* ARM Clock Control Reg */
#define ARM_IDLECT1           *((volatile UInt16 *) 0xFFFECE04) /* ARM Idle Control Reg 1 */
#define ARM_IDLECT2           *((volatile UInt16 *) 0xFFFECE08) /* ARM Idle Control Reg 2 */
#define ARM_RSTCT1            *((volatile UInt16 *) 0xFFFECE10) /* ARM Reset Control Reg 1 */
#define ARM_RSTCT2            *((volatile UInt16 *) 0xFFFECE14) /* ARM Reset Control Reg 2 */
#define ARM_SYSST             *((volatile UInt16 *) 0xFFFECE18) /* ARM System Status Register
*/
#define DSP_IDLECT2           *((volatile UInt16 *) 0xE1008008) /* DSP Idle Control Reg 2 */
#define ULPD_CLOCK_CTRL_REG   *((volatile UInt16 *) 0xFFFE0830) /* ULPD additional clock
controls */
#define ULPD_POWER_CTRL_REG   *((volatile UInt16 *) 0xFFFE0850) /* ULPD Power Control, low_pwr
enable */
#define EMIFS_CONFIG_REG      *((volatile UInt32 *) 0xFFFECC0C) /* Memory I/F Control Register
*/
#define EMIFF_SDRAM_CONFIG    *((volatile UInt32 *) 0xFFFECC20) /* SDRAM Config Register */
#define WATCHDOG_TIMER_MODE   *((volatile UInt16 *) 0xFFFEC808) /* ARM Watchdog timer mode */
#define DPLL1_CONTROL_REG     *((volatile UInt16 *) 0xFFFECF00) /* DPLL1 Control Reg */
#define INTH1_ITR             *((volatile UInt32 *) 0xFFFECB00) /* 2nd Level Interrupt
register */
```

```
#define INTH1_MASK            *((volatile UInt32 *) 0xFFFECB04) /* 2nd Level Interrupt handler
mask */
#define INTH2_ITR             *((volatile UInt32 *) 0xFFFE0000) /* 2nd Level Interrupt
register */
#define INTH2_MASK            *((volatile UInt32 *) 0xFFFE0004) /* 2nd Level Interrupt handler
mask */
#define FUNC_MUX_CTRL_0       *((volatile UInt32 *) 0xFFFE1000) /* OMAP5910 Configuration
Registers */
#define FUNC_MUX_CTRL_1       *((volatile UInt32 *) 0xFFFE1004)
#define FUNC_MUX_CTRL_2       *((volatile UInt32 *) 0xFFFE1008)
#define COMP_MODE_CTRL_0      *((volatile UInt32 *) 0xFFFE100C)
#define FUNC_MUX_CTRL_3       *((volatile UInt32 *) 0xFFFE1010)
#define FUNC_MUX_CTRL_4       *((volatile UInt32 *) 0xFFFE1014)
#define FUNC_MUX_CTRL_5       *((volatile UInt32 *) 0xFFFE1018)
#define FUNC_MUX_CTRL_6       *((volatile UInt32 *) 0xFFFE101C)
#define FUNC_MUX_CTRL_7       *((volatile UInt32 *) 0xFFFE1020)
#define FUNC_MUX_CTRL_8       *((volatile UInt32 *) 0xFFFE1024)
#define FUNC_MUX_CTRL_9       *((volatile UInt32 *) 0xFFFE1028)
#define FUNC_MUX_CTRL_A       *((volatile UInt32 *) 0xFFFE102C)
#define FUNC_MUX_CTRL_B       *((volatile UInt32 *) 0xFFFE1030)
#define FUNC_MUX_CTRL_C       *((volatile UInt32 *) 0xFFFE1034)
#define FUNC_MUX_CTRL_D       *((volatile UInt32 *) 0xFFFE1038)
#define PULL_DWN_CTRL_0       *((volatile UInt32 *) 0xFFFE1040)
#define PULL_DWN_CTRL_1       *((volatile UInt32 *) 0xFFFE1044)
#define PULL_DWN_CTRL_2       *((volatile UInt32 *) 0xFFFE1048)
#define PULL_DWN_CTRL_3       *((volatile UInt32 *) 0xFFFE104C)
#define GATE_INH_CTRL_0       *((volatile UInt32 *) 0xFFFE1050)
#define VOLTAGE_CTRL_0        *((volatile UInt32 *) 0xFFFE1060)
#define TEST_DBG_CTRL_0       *((volatile UInt32 *) 0xFFFE1070)
#define MOD_CONF_CTRL_0       *((volatile UInt32 *) 0xFFFE1080)
```

A.2 Assembly Code

```
.global    _OMAPDeepSleep
 .sect ".text"

_OMAPDeepSleep:
    ; Clear EMIFS global power-down enable bit (PDE) in EMIFS_CONFIG register
    ldr     r0, emifs_config_reg
    ldr     r1, [r0]
    ldr     r2, emifs_config_reg_pde_mask
    and     r1, r1, r2
    str     r1, [r0]
    ; Set IMIF power-down bit (PWD_EN) in EMIFS_CONFIG register
    ldr     r1, [r0]
    orr     r1, r1, #0x00000004
```

```
str     r1, [r0]
; Set CLK, PWD and SLRF bits of SDRAM Controller
ldr     r0, emiff_sdram_config_reg
ldr     r1, [r0]
orr     r1, r1, #0x0C000001
str     r1, [r0]
; Set EMIFS global power-down enable bit (PDE) in EMIFS_CONFIG register
ldr     r0, emifs_config_reg
ldr r1, [r0]
orr r1, r1, #0x08
str r1, [r0]
; Put DPLL in Bypass
ldr r0, dpll1_ctl_reg
ldrh r1, [r0]
ldr     r2, dpll1_ctl_reg_mask
and     r1, r1, r2
strh    r1, [r0]
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
; Stop the GPIO (XOR), API, HSAB, LB, and LCD clocks if running
ldr     r0, arm_idlect2_reg
ldrh    r1, [r0]
ldr     r2, arm_idlect2_reg_stopmask
and     r1, r1, r2
strh    r1, [r0]
; Idle all of the ARM-based clocks and Go to sleep
ldr     r0, arm_idlect1_reg
ldr     r1, arm_idlect1_reg_sleepmask
strh    r1, [r0]
nop
nop
nop
nop
nop
nop
```

```
    nop
    nop
    nop
    nop
    nop
    nop
    nop
    nop
    nop
    nop
    nop
    nop
    nop
    nop
    nop
    nop
    nop
    nop
    nop
    nop
    nop

    mov pc, lr
emifs_config_reg                .word 0xFFFECC0C
emiff_sdram_config_reg          .word 0xFFFECC20
arm_idlect1_reg                 .word 0xFFFECE04
arm_idlect2_reg                 .word 0xFFFECE08
dpll1_ctl_reg                   .word 0xFFFECF00
dpll1_ctl_reg_mask              .word 0xFFFFFFE3
emifs_config_reg_pde_mask    .word    0xFFFFFFF7
arm_idlect1_reg_val             .word 0x000006C7
emiff_sdram_config_reg_wakemask .word 0xF3FFFFFE
arm_idlect2_reg_stopmask     .word    0xFFFFF987
arm_idlect1_reg_sleepmask       .word 0x00000EC7
```

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale (www.ti.com/legal/termsofsale.html) or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265