Freescale Semiconductor
Application Note

Document Number: AN3932
Rev. 0.0, 11/2009

MC13224 Configuration and Operation with the Buck Regulator

1 Introduction

The Freescale MC13224 is a wireless communication platform that incorporates a complete, low power, 2.4 GHz radio frequency transceiver and a 32-bit ARM7 MCU with peripherals into a 99-pin LGA Platform-in-Package (PiP). The MC13224V solution can be used for wireless applications complying with the 2.4 GHz IEEE® 802.15.4 Standard.

The Freescale MC13224 has an optional buck voltage regulator to maximize battery efficiency for power sensitive applications. This application note describes how to configure the device hardware and describes software for use of the buck regulator.

To illustrate the use of the buck regulator:

- The MC1322x Low Power Board “LPB” is used as the hardware platform because it provides the buck regulator components.
- The “Low Power Bell TX” demo application based on the 1322x SMAC Codebase is used as the software source.

Contents

NOTE

The methods described in this note apply to other HW implementations and Freescale Software stacks (e.g. BeeStack, MAC and BeeStack Consumer).

Codebases supporting Buck Regulator are:

- BeeKit ARM7 MAC Codebase 1.1.4 or later,
- BeeKit ARM7 SynkroRF: 1.2.10 or later,
- BeeKit ARM7 BeeStack Consumer: 1.1.3 or later,
- BeeKit ARM7 BeeStack 3.0.5. or later and
- BeeKit ARM7 SMAC Codebase 1.1.3 or later

2 Buck Regulator Hardware Configuration and Use

This section describes the hardware configuration for use of the buck regulator and briefly provides an overview of the rules for using, enabling, and disabling the buck. It is not the intent of this description to provide a detailed description of the buck regulator; the user is directed to the Freescale MC1322x Reference Manual, Section 3.2.3, for a thorough description of the buck regulator.

2.1 Hardware Configuration

There are two pins that supply VDD voltage to the MC13224; i.e., VBATT and LREG_BK_FB. These pins can be configured for two modes of operation:

- Tied common (buck not used) - VDD is the source voltage to all onboard circuitry and regulators
- Buck regulator used - LREG_BK_FB becomes the buck regulated output voltage. LREG_BK_FB feeds the analog and FLASH (NVM) voltage regulators. As a result, the buck saves power when the radio or the NVM is in use.

Figure 1 shows the hardware configuration for use of the buck regulator. A 100 µH inductor and a 10 µF capacitor form the buck switching network, and a Schottky diode is added to clamp any overshoot on the drive side of the inductor to a diode drop above VBATT (see the MC1322x Reference Manual, Section 3.2.3.1 for details of the circuit).

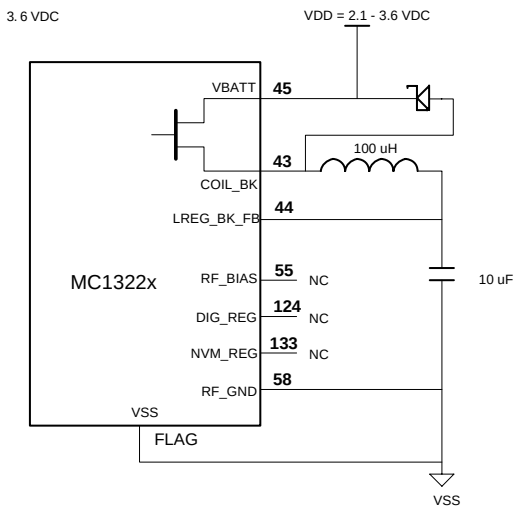

Figure 1. Using the Buck Regulator with the MC13224

2.2 Buck Usage Rules

Operation of the buck regulator is controlled by the MC13224 SYS_CNTL and VREG_CNTL Registers. General rules for use of the buck include:

- The buck mode must be enabled and disabled as required.
- The buck is disabled during low power modes.
- The buck enable modes include -
  - — Normal switching mode - usable when the source VDD voltage is about 2.5 V to 3.6 V
  - — Bypass mode - the high-side P-channel switching transistor is simply turned-on (not switched) to supply current to the LREG_BK_FB input.
- The VDD voltage can be monitored via the ADC block to determine when to transfer from switching mode to bypass mode as the battery voltage decreases.

Refer to the MC1322x Reference Manual, Section 3.2.3.2 for details on the control mechanisms and procedures.

3 MC13224 Application Configuration

This section outlines configuration of a MC13224 application on the Freescale BeeKit development environment. Understanding this procedure is necessary before describing the changes done to an application to support buck regulator operation.

The target software for modification is the “Low Power Bell Demonstration” which actually consists of two applications: “Low Power Bell TX” and “Low Power Bell RX”:

- Only the Low Power Bell TX application is modified - only one target is necessary to demonstrate use of the buck regulator
- The MC1322x low power board or LPB is the target for the Low Power Bell TX application because it supports the buck hardware
- The MC1322x network control board or NCB is the target for the Low Power Bell RX application

The following subsections describe how to generate, compile and execute the Low Power Bell demo. The instructions first apply to an LPB NOT CONFIGURED for the buck regulator. Subsequent information covers operation on an LPB after it has been modified to support the buck regulator.

3.1 Generating the Project In BeeKit

To generate the project in BeeKit:

1. Open BeeKit (Start->Programs->Freescale BeeKit->Freescale BeeKit).
2. In BeeKit, select the MC1322x SMAC codebase (File menu -> Select Codebase...)
3. Click on File -> New Project and choose the Low Power Bell TX template.
4. Follow the steps as displayed in the New Project wizard.
5. Add the Low Power Bell RX project to the solution.
6. Configure the properties as needed.

NOTE

Be sure to configure the Low Power Bell TX and Low Power Bell RX applications both on the same channel.

7. Validate the solution.
8. Export the solution.

For further details on how to use BeeKit, see the BeeKit Wireless Connectivity Toolkit User’s Guide (BKWCTKUG).

3.2 Open, Compile and Execute the Low Power Bell Application (before HW LPB modifications)

Using the IAR Embedded Workbench, open the solution work space (My Solution.eww).

3.2.1 Load “Low Power Bell TX” on a LPB

1. Choose the “Low Power Bell TX” tab.
2. Select “Release” in the Workspace window.
3. Click on Project -> Rebuild All.
4. Use the JTAG interface to load the Low Power Bell TX application on the LPB board being used as the Low Power Bell TX - Connect the JTAG interface to the board, then click the Debug button at the IAR Embedded Workbench IDE.
5. Close the debugging session and move the JTAG connector to an NCB.

3.2.2 Load “Low Power Bell RX” on a NCB

1. Choose the “Low Power Bell RX” tab.
2. Click on Project -> Rebuild All.
3. Use the JTAG interface to load the Low Power Bell RX application on the NCB board being used as the Low Power Bell RX - Click the IAR Debug button to download the application to the board.

3.2.3 Execute Application

Both boards are now ready to run the application:

1. Reset both boards.
2. Pressing SW1 on the LPB-TX results in activation of the buzzer on the LPB-RX.

For additional information on how to generate and run this application refer to “MC1322x SMAC Demonstration Application User’s Guide“, Chapter 5 - Low Power Bell Demonstration

4 Buck Regulator Enable

To configure the buck regulator, two registers from the MC13224 Clock/Reset/Power Module “CRM” are used:

- System Control Register (SYS_CNTL)
- Voltage Regulator Control Register (VREG_CNTL)

4.1 System Control Register (SYS_CNTL)

The PWR_SOURCE[1:0] field of SYS_CNTL (Bit [1:0]) selects the system power source. This value selects the type of supply that will be powering the system (see Table 1).

NOTE

See the MC1322x Reference Manual, Section 3.2.3.2

- This value is retained until a hard or soft reset (write once)
- On POR start-up, a VBATT battery supply is assumed (default) unless over-programmed.

- On any subsequent wake-up from hibernate or doze, the system will immediately use the over-programmed supply configuration
- Writes to this field should be done via a read-modify-write.
- This field must be written to 0x1 to use the buck

Table 1. Power Supply Selections

PWR_SOURCE	Power Source	Detail
2'b00	VBATT	Any enabling of the Buck regulator will be ignored. (default)
2'b01	Buck Regulation	Enables buck regulator
2'b10	Reserved	N/A
2'b11	Reserved	N/A

4.2 Voltage Regulator Control Register (VREG_CNTL)

The other relevant register, VREG_CNTL, controls operation the power supply regulators.

NOTE

See the MC1322x Reference Manual, Section 5.9.18, "Voltage Regulator Control (VREG_CNTL)"

- The register contents are not retained in sleep mode.
- The VREG_CNTL register configures -
  - BUCK_CLKDIV[3:0] - buck regulator switching frequency
  - VREG_1P8V_EN - NVM 1.8V voltage regulator enable
  - VREG_1P5V_SEL[1:0] - Analog 1.5V voltage regulator current select field
  - VREG_1P5V_EN[1:0] - Analog 1.5V voltage regulator enable field
  - BUCK_BYPASS_EN - Buck regulator bypass enable
  - BUCK_SYNC_REC_EN - Buck synchronous rectifier voltage regulator enable
  - BUCK_EN - Buck voltage regulator enable

4.3 Code for Enabling Buck Regulator

After boot, the application initialization code runs and leaves the device in a mode for "normal" or non-buck regulator mode. After executing the radio init function included in the library (RadioInit for SMAC):

- The SYS_CNTL Register PWR_SOURCE[1:0] field is default (VBATT single source)
- The VREG_CNTL Register has the following configuration:
  - BUCK_CLKDIV = 0xF
  - VREG_1P8V_EN = 0 (NVM disabled)
  - VREG_1P5V_SEL = 11 (40mA selected)
  - VREG_1P5V_EN = 00 (disabled)

- BUCK_BYPASS_EN = 0 (disabled)
- BUCK_SYNC_REC_EN = 0 (disabled)
- BUCK_EN = 0 (disabled)

In order to enable the buck regulator in the Low Power Bell TX application:

- Buck Regulation must be enabled in the SYS_CNTL Register - field PWR_SOURCE[1:0] = 01
- The VREG_CNTL Register must be modified as designated below in bold letters -
  - BUCK_CLKDIV = 0xF
  - VREG_1P8V_EN = 0 (disabled)
  - VREG_1P5V_SEL = 11 (40mA selected)
  - **VREG_1P5V_EN = 11 (radio fully enabled)**
  - BUCK_BYPASS_EN = 0 (disabled)
  - **BUCK_SYNC_REC_EN = 1(enabled)**
  - **BUCK_EN = 1(enabled)**

Therefore to provide the buck initialization, the following lines of code are added at the end of the initialization function “low_power_bell_app_init” as follows:

```
void low_power_bell_app_init(void)
{
      ...
      CRM_REGS_P->SysCntl |= 0x00000001; //Enable buck regulator as power source
      CRM_REGS_P->VregCntl = 0x00000F7B; //buck enabled
      u8BuckState = gBuckEnabledSt_c;
}
```

After adding the code as explained above, compile and download and run the application to a modified LPB.

When running the application with the buck enabled you will observe a waveform at the COIL_BK pin as shown in Figure 2. The application will continue to work as previously described.

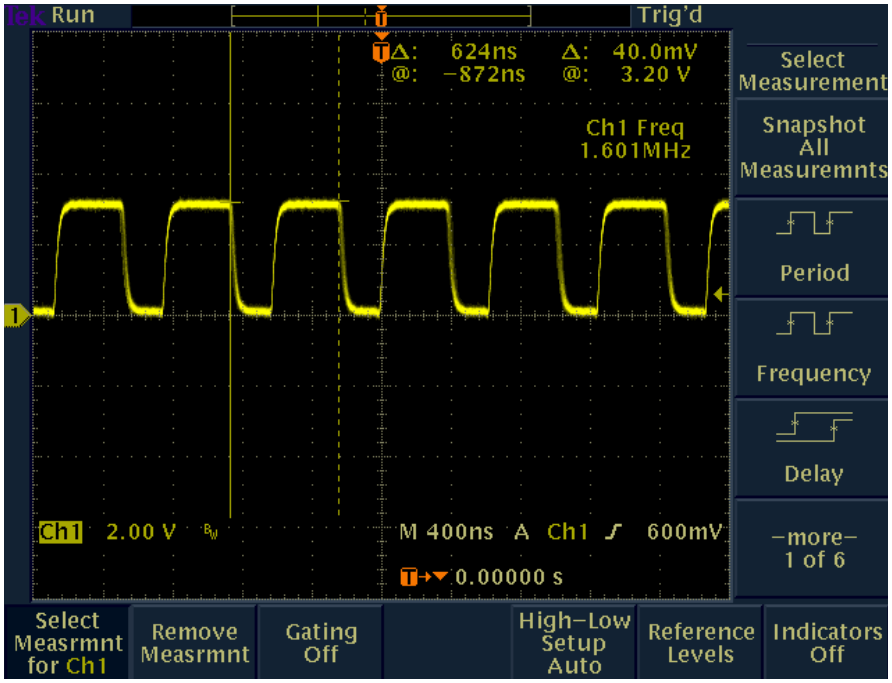

Figure 2. Signal at COIL_BK (Buck Regulator Enabled)

5 Buck Regulator Bypass

Once the LPB is modified to support the buck regulator, there is an option to enable “bypass” mode. In bypass mode, the onboard switcher function is disabled but the high-side transistor is fully turned-on providing an “always on” connection to VBATT.

This is achieved by modifying the VREG_CNTL register as shown in the following line of code:

```
CRM_REGS_P->VregCntl = 0x00000F7C; //buck bypass
```

Where the value 0x00000F7C equates to:

- BUCK_CLKDIV = 0xF
- VREG_1P8V_EN = 0 (disabled)
- VREG_1P5V_SEL = 11 (40mA selected)
- VREG_1P5V_EN = 11 (radio fully enabled)
- BUCK_BYPASS_EN = 1 (enabled)
- BUCK_SYNC_REC_EN = 0 (disabled)
- BUCK_EN = 0 (disabled)

After adding the code as explained above, compile, download and run the application on a modified LPB.

When running the application with the buck bypassed, the waveform observed at the COIL_BK pin shows a constant DC signal as shown in Figure 3. The application will continue to work as previously described.

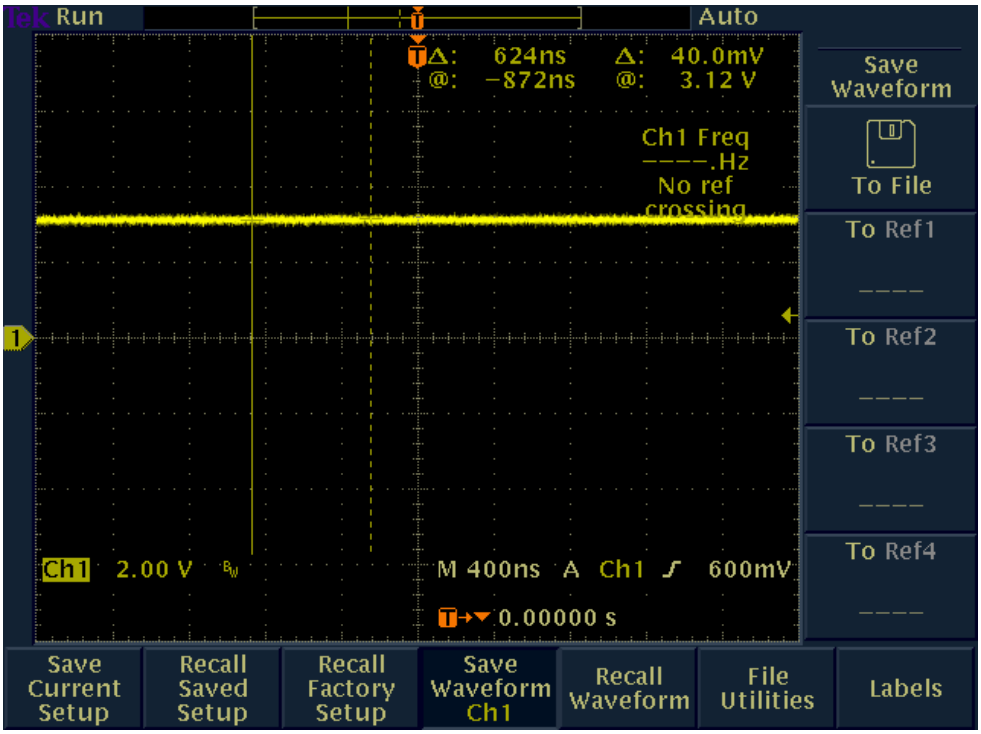

Figure 3. Signal at COIL_BK **when Buck Regulator Is Bypassed**

6 Monitoring Battery Voltage

The buck regulator is used as a DC switcher in the ~2.5-3.6 VDC input supply range. If the battery voltage drops below 2.5 V then the buck regulator should be bypassed.

The MC13224 has two 12-bit analog-to-digital converters (ADCs) that share 8 input channels and an additional channel for ADC1 is internally connected to a fixed “Battery Reference” voltage of 1.2Vdc. This capability can be used to monitor the battery voltage (VBATT).

The VREFh reference voltage to the ADCs can be enabled internally to VBATT or connected externally to VBATT. As the battery voltage (the supply voltage) drops, the VREFh also drops and sampling the fixed 1.2 V reference voltage will result in a higher ADC reading as a result. In the ADC control registers, the Battery Voltage Upper Trip Point Register (ADC_BAT_COMP_OVER) can be used to detect a low supply voltage trip point.

When considering the voltages, the fixed voltage is 1.2 V and the full scale voltage for the 12-bit ADC (VREFh) will be 2.5 V for the trigger condition. Using ratios, a full scale ADC reading would be 0xFFF (4095_{dec}) for 2.5 V and the sampled fixed voltage of 1.2 V would be 0x7AD (1965_{dec})

```
...
#define gBattRefInMilivolts_c    (1200) // Internal reference is 1.2 V
#define gMaxAdcCoun_c            (4095) // 12 Bits ADC then ((2^12)-1)
#define gMinValForBuckEnableInMilivolts_c (2500) // 2.5 V
#define gHysteresisMilivolts_c   (100)
#define gVBattAdcValAt2p5V_c
((gBattRefInMilivolts_c*gMaxAdcCoun_c)/(gMinValForBuckEnableInMilivolts_c))
```

```
#define gRestoreBuckAdcCounts_c
((gBattRefInMilivolts_c*gMaxAdcCoun_c)/(gMinValForBuckEnableInMilivolts_c+gHysteresisMilivolt
s_c))
#define gBattOverMask_c          (0x0001 << 9)
#define gBattUnderMask_c         (0x0001 << 8)
#define gTimeBetweenSamples_c    (9999)
...
```

You will need a function for initializing and configuring the ADC to take readings from the internal battery reference in order to monitor Battery Voltage.

```
...
/*****************************************************************************
* ADC_Setup
*
* This function configures the analog to digital converter.
*****************************************************************************/
static void ADC_Setup(void)
{
  AdcConfig_t sAdc_Config;
  AdcConvCtrl_t adcConvCtrl;
  AdcCompCtrl_t adcCompCtrl;
  Adc_Init();
  Adc_Reset();
  (void)Adc_SetFifoCtrl(FIFO_DEEP); /*Sets 7 to the fifo treshold*/
  adcCompCtrl.adcChannel = gAdcBatt_c;
  adcCompCtrl.adcCompType = gAdcCompLess_c;
  adcCompCtrl.adcCompVal = gRestoreBuckAdcCounts_c;
  (void)Adc_SetCompCtrl(&adcCompCtrl);
  adcCompCtrl.adcChannel = gAdcBatt_c;
  adcCompCtrl.adcCompType = gAdcCompGrater_c;
  adcCompCtrl.adcCompVal = gVBattAdcValAt2p5V_c;
  (void)Adc_SetCompCtrl(&adcCompCtrl);
  Adc_DefaultConfig(sAdc_Config, 24000); /* (1Mhz prescale clock, 300KHz ADC analog clock,
                                            8 us on time, 40 us conversion time,
                                            auto working mode, compare interrupt enabled,
                                            FIFO interrupt disabled)*/
  sAdc_Config.adcCompIrqEn = TRUE;
  (void)Adc_SetConfig(&sAdc_Config);
  Adc_TurnOn();
  adcConvCtrl.adcTmrOn = TRUE;
  adcConvCtrl.adcSeqIrqEn = TRUE;
  adcConvCtrl.adcChannels = 0;
  adcConvCtrl.adcChannels |= (1 << gAdcBatt_c);
  adcConvCtrl.adcTmBtwSamples = gTimeBetweenSamples_c; // 1s between samples
  adcConvCtrl.adcSeqMode = gAdcSeqOnTmrEv_c; //Timer mode
  adcConvCtrl.adcRefVoltage = gAdcBatteryRefVoltage_c;
  (void)Adc_SetConvCtrl(gAdcPrimary_c, &adcConvCtrl);
  (void)Adc_SetCallback(gAdcSeq1event_c,(AdcEvCallback_t)ADC_Callback);
  (void)Adc_SetCallback(gAdcCompEvent_c,(AdcEvCallback_t)ADC_CompCallback);
}
...
```

There are two callbacks involved `ADC_Callback` and `ADC_CompCallback`. `ADC_Callback` does not require to perform any action for monitoring while `ADC_CompCallback` should take care of the VBatt over/under events:

```
...
```

```
static volatile bool_t gIsSystemUnderVoltage;
...

/*****************************************************************************
*ADC_Callback
*
* Do nothing for this application
*****************************************************************************/
void ADC_Callback(void)
{
}

/*****************************************************************************
* ADC_CompCallback
*
* This function is called during ADC interrupt when VBatt over/under consition is reached.
* Deppending on the event the function sets or clears the gIsSystemUnderVoltage flag.
*****************************************************************************/
void ADC_CompCallback(void)
{
  uint16_t AdcChannelsTriggered;
  gIsSystemUnderVoltage = FALSE;
  AdcChannelsTriggered = Adc_ChnTriggered();
  if(gBattOverMask_c & AdcChannelsTriggered)
  {
    gIsSystemUnderVoltage = TRUE;
  }
}
```

Now the ADC ISR shall be initialized at function `low_power_bell_app_init`.

```
void low_power_bell_app_init(void)
{
...
  IntAssignHandler(gAdcInt_c, (IntHandlerFunc_t)Adc_Isr);
  ITC_SetPriority(gAdcInt_c, gItcNormalPriority_c);
  ITC_EnableInterrupt(gAdcInt_c);

  CRM_RegisterISR(gCrmKB4WuEvent_c, after_wakeup_isr);

  IntDisableIRQ();
  IntDisableFIQ();
...
}
```

Using these functions can provide a state machine to enable or bypass the buck regulator based on the battery voltage. The possible states can be enabled, disabled and bypassed. A sample of the function implementing this state machine, `setBuckState`, is shown here:

```
...
typedef enum buckRegStates_tag{
  gBuckDisabledSt_c = 0,
  gBuckBypassSt_c,
  gBuckEnabledSt_c,
```

```
}buckRegStates_t;

static buckRegStates_t u8BuckState;
...

/*****************************************************************************
* setBuckState
*
* This function switches between buck_bypass and enabled modes according to the
* measured vbatt voltage at ADC
*****************************************************************************/
void setBuckState(buckRegStates_t buckDesiredState)
{
  if(gBuckDisabledSt_c == buckDesiredState)
    return;
  if(gBuckBypassSt_c == buckDesiredState)
  {
    if(gBuckEnabledSt_c == u8BuckState)
    {
      CRM_REGS_P->VregCntl = 0x00000F04;//set buck bypass and disable nvm and radio regs*/
      CRM_REGS_P->VregCntl = 0x00000F7C;//buck bypass and radio reg*/
      u8BuckState = gBuckBypassSt_c ;
    }
  }
  else
  {
    if(gBuckBypassSt_c == u8BuckState)
    {
      CRM_REGS_P->VregCntl = 0x00000F7B;//buck enabled
      u8BuckState = gBuckEnabledSt_c ;
    }
  }
}
```

NOTE

When switching from Enable to Bypass state the NVM and Radio regulators are disabled during the transition.

Now the buck regulator can be configured to be bypassed at the end of the initializing function `low_power_bell_app_init` function and the state variable "`u8BuckState`" initialized to "`gBuckBypassSt_c`".

```
void low_power_bell_app_init(void)
{
      ...
      CRM_REGS_P->VregCntl = 0x00000F7C;//buck bypass*/
      u8BuckState = gBuckBypassSt_c;
}
```

And "`setBuckState`" could be called on a periodic task in this case, it would be located at the "`low_power_bell_process`" function.

```
void low_power_bell_process(void)
{
  static low_pow_bell_states_t u8AppState = IDLE_LP_BELL_ST;
```

```
  static uint8_t num_retries = 0;

#if(gUseLowPowerMode_c)
  crmSleepCtrl_t SleepCtl;
  uint32_t u32LoopDiv;
#endif

  (void)process_radio_msg();
  if(TRUE == gbDataIndicationFlag)
  {
    gbDataIndicationFlag = FALSE;
    process_incoming_msg();
  }
  else
  {
  }

  if ( gBuckDisabledSt_c != u8BuckState )
  {
    if(TRUE == gIsSystemUnderVoltage)
    {
      setBuckState(gBuckBypassSt_c);
    }
    else
    {
      setBuckState(gBuckEnabledSt_c);
    }
  }

#if OTAP_ENABLED == TRUE
 if(gbOtapExecute)
 {
    OTAP_execute();
 }
 else
#endif
      ...
}
```

With this implementation, the buck regulator will be automatically enabled or bypassed based on the battery voltage.

7 Considerations for Low Power Modes

This section covers how to manage the buck regulator when operating in low power modes like Doze or Hibernate. Since the VREG_CNTL register contents are not retained in sleep mode, they need to be restored to their desired contents after waking up.

7.1 Setting Low Power

A Low Power option (disable) is added to the Low Power Bell demonstration application. To set low power module enable on an SMAC application, configure as desired the appropriate defines in the PWR_Config.c file:

```
#define gUseLowPowerMode_c          TRUE
#define gMCURetentionMode_c         TRUE
#define gMCUPadRetentionMode_c      TRUE
#define gRAMRetentionMode_c         gRamRet96k_c
```

The Low Power module can also be enabled in BeeKit:

1. When generating the project as described in Section “CROSS REFERENCE to 3.1”: at Step 4 check the Low Power Module check box when running the wizard as show at Figure 4

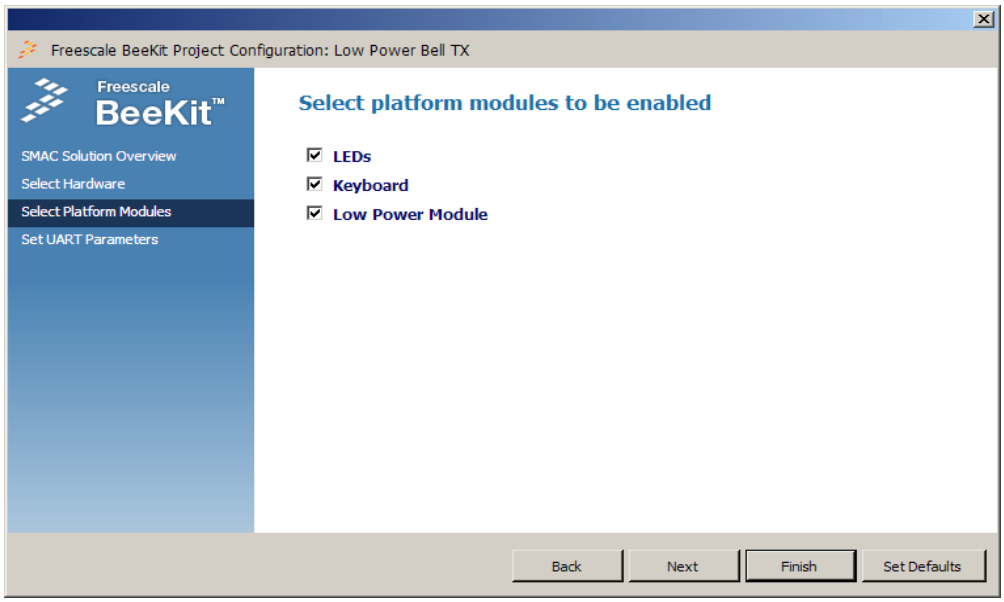

Figure 4. Enabling Low Power Module at BeeKit wizard

2. Edit the property and export.

7.2 Restoring Buck Regulator Settings

With the Low Power Bell TX application is configured for using low power operation, it must handle the re-configuration of the buck regulator settings after each wake-up event (configuration is not retained during sleep periods). This can be achieved by setting the buck regulator to bypass state after the Radio has been initialized as consequence of the system waking up. The code excerpt below shows how the "`low_power_bell_process`" function was modified to accomplish this.

```
void low_power_bell_process(void)
{
      ...
case LOW_POW_BELL_ST:
      {
        if(is_tx_msg_final_state(TX_msg) && is_rx_msg_final_state(RX_msg))
        {
          TurnOffLeds();
#if(gUseLowPowerMode_c)
            Adc_TurnOff();
            SleepCtl.ramRet = gRAMRetentionMode_c;
            SleepCtl.mcuRet = gMcuRet_c;
            SleepCtl.pfToDoBeforeSleep = NULL;
            MLMEHibernateRequest(gRingOsc2khz_c, SleepCtl);
            Adc_TurnOn();
            u32LoopDiv = ((gDigitalClock_RN_c<<25) + gDigitalClock_RAFC_c);
            RadioInit(PLATFORM_CLOCK, gDigitalClock_PN_c, u32LoopDiv);
            CRM_REGS_P->VregCntl = 0x00000F7C;//buck bypass*/
            u8BuckState = gBuckBypassSt_c;
            (void)MLMEPAOutputAdjust(gDefaultPowerLevel_c);
            MLMESetChannelRequest(CHANNEL_NUMBER);
#endif
        }
      }
      break;
      ...
}
```

With this change the buck will be set to bypass just after awaking and it will be enabled when the Battery Voltage is bigger than 2.5 V during execution of the "`switch_buck_bypass_enable`" function.

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064, Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

***For Literature Requests Only*:**
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-521-6274 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

Document Number: AN3932
Rev. 0.0
11/2009

freescale™
semiconductor