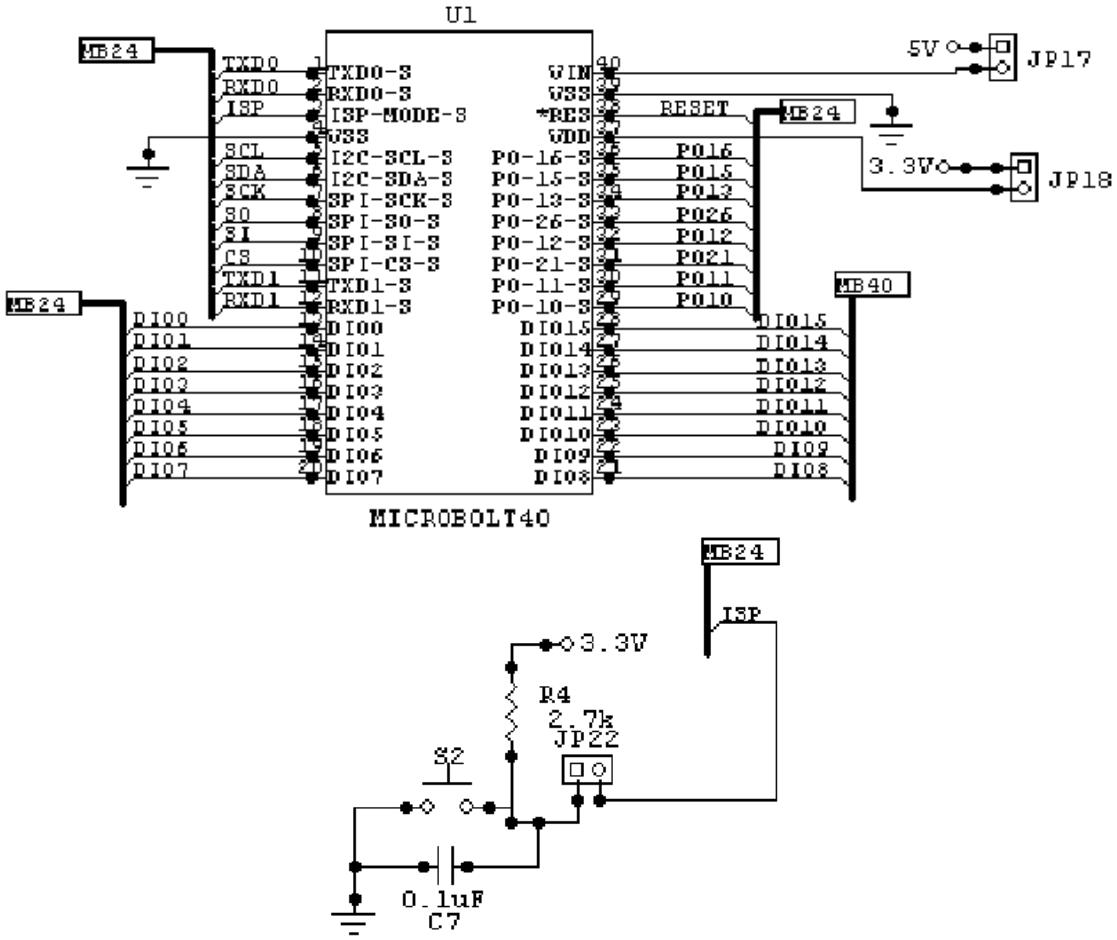
Micromint, Inc	**AN821**
	MicroBolt
MicroBolt Low Power Modes	12/30/2005

Introduction:
This application notes demonstrates how to access the two lower power modes offered by the MicroBolt.

Background:
The MicroBolt supports two low power modes to aide in the reduction of power. These low power modes can provide crucial current reduction for power sensitive applications such as battery-operated devices that incorporate a MicroBolt.

Schematic:



How it works:

This ImageCraft ICCARM demo project demonstrates the MicroBolt's low power modes. The MicroBolt supports two reduced power modes: Idle mode and Power Down mode. The example code starts up the MicroBolt in normal mode, then places it in Power down mode after blinking the MicroBolt LED 10 times. To come out of Power down mode, you must press the push button switch on the MicroBolt development board labeled S2. This switch is tied to the external interrupt EINT1. Once the MicroBolt comes out of Power down mode, Timer-0 is used to blink the MicroBolt LED every 10 seconds. When the LED is off, the MicroBolt is placed into Low power mode. When it is on, Timer-0 has woken up the MicroBolt and normal mode is activated.

The demo project was constructed so as to easily take current measurements of the MicroBolt during the different power modes. The resulting current draw for the various mode is as follows (sample size of 1):

Normal mode, approximately 53 mA
Idle mode, approximately 17 mA
Power down mode, approximately 12 mA

In Idle mode, execution of instructions is suspended until either a reset or interrupt occurs. Peripheral functions continue operation during Idle mode and may generate interrupts to cause the processor to resume execution. Idle mode eliminates power used by the processor itself, memory systems and related controllers, and internal buses.

In Power Down mode, the oscillator is shut down and the LPC2106 receives no internal clocks. The processor state and registers, peripheral registers, and internal SRAM values are preserved throughout Power Down mode and the logic levels of chip pins remain static. The Power Down mode can be terminated and normal operation resumed by either a Reset or the external interrupts, EINT0, EINT1, and EINT2. Since all dynamic operation of the chip is suspended, Power Down mode reduces LPC2106 power consumption to nearly zero.

Wakeup from Power Down or Idle modes via an interrupt resumes program execution in such a way that no instructions are lost, incomplete, or repeated.

A Power Control for Peripherals feature allows individual peripherals to be turned off if they are not needed in the application, resulting in additional power savings. This is shown in the example code.

Sources:
Philips LPC2106 datasheet

Program Listing:

```
/*
|-----------------------------------------------------------------------------------------
|-----------------------------------------------------------------------------------------
| File Name                     : MicroBoltLowPowerModes.c
| Author                        : Micromint, Inc.
| Copyright                     : Copyright © 2005, Micromint, Inc.
| Creation Date                 : 10/16/05
| Version                       : 1.00
| Spaces per tab                : 2
| Description                   : Main C file
| Revision                      : Initial
|-----------------------------------------------------------------------------------------
|-----------------------------------------------------------------------------------------
*/

/*
|-------------------------------------------------
|     Includes
|-------------------------------------------------
*/

#include <ARM/philips/lpc210x.h>
#include <arm_macros.h>

#include "MicroBoltLowPowerModes.h"
```

```
/*
|-------------------------------------------------------------------------------------------
|     Function        :   main
|     Inputs          :   None
|     Outputs         :   None
|     Purpose         :   Main function for system
|     Author          :   Micromint, Inc.
|-------------------------------------------------------------------------------------------
*/

void main(void)
  {
  unsigned int Delay;
  unsigned char Count;

/*
|-----------------------------------------------------
|     MicroBolt hardware setup
|-----------------------------------------------------
*/

  __DISABLE_INTERRUPT();                        // Disable all interrupts

/*
|-------------------------
| Configure MAM
|-------------------------
*/
  MAM_CR = 0x00;                                // Turn MAM off (default)
  MAM_TIM = 0x04;                               // Set flash timing to 4 clock cycles

  MAM_CR = 0x02;                                // Fully enable the Memory Accleration Module

/*
|-------------------------
| Configure VIC
|-------------------------
*/
  VICVectAddr0 = (unsigned)pll_isr;             // Assign the PLL lock ISR vector address
  VICVectCntl0 = INTERRUPT_CHANNEL_FOR_PLL;     // Assign the VIC address to the actual interrupt
  VICIntEnable = INTERRUPT_ENABLE_FOR_PLL;      // Enable the interrupt

  VICVectAddr1 = (unsigned)ExtInt1_ISR;         // Assign the EINT1 ISR function to VIC priority 1
  VICVectCntl1 = INTERRUPT_CHANNEL_FOR_EINT1;   // Assign the VIC channel EINT1 to interrupt priority 1
  VICIntEnable |= INTERRUPT_ENABLE_FOR_EINT1;   // Enable the interrupt

  VICVectAddr2 = (unsigned)Timer0_ISR;          // Assign the Timer-0 ISR function to VIC priority 2
  VICVectCntl2 = INTERRUPT_CHANNEL_FOR_TIMER0;  // Assign the VIC channel Timer-0 to interrupt priority 2
  VICIntEnable |= INTERRUPT_ENABLE_FOR_TIMER0;  // Enable the Timer-0 interrupt

/*
|-------------------------
| Configure PLL and CCLK
|-------------------------
*/
  SCB_PLLCFG |= 0x23;                            // Set to 59 MHz (0x03 is multiply value of 4)
  SCB_PLLCON |= 0x01;                           // Enable the PLL
  SCB_PLLFEED = 0xAA;                            // Shadow register copy to enable changes
  SCB_PLLFEED = 0x55;                           // in PLLCON and PLLCFG

/*
|-------------------------
| Configure PCLK
|-------------------------
*/
  SCB_VPBDIV = 0;                   // Peripheral clock is 1/4th Processor clock which equals 14.7456 MHz

/*
|-------------------------
| Configure GPIO
|-------------------------
*/
  PCB_PINSEL0=0x00000000;           // JTAG is via secondary port (Configured via Imagecraft App builder)
  PCB_PINSEL1=0x55400000;
```

```
  GPIO_IODIR=(0x00000000<<16)|                  // Make all inputs to start with
             0x00000000;

  GPIO_IODIR |= MICROBOLT_LED;                  // Setup MicroBolt LED as output

  GPIO_IOCLR = MICROBOLT_LED;                   // MicroBolt LED off

  PCB_PINSEL0 |= P0_14_EXTERNAL_INTERRUPT_1;    // Setup P0.14 to alternate function EINT1

/*
|-------------------------
| Configure Timer-0
|-------------------------
*/
  T0_MR0 = 0x08C3F087;                          // Match register 0 value for timer rate = 10 Seconds

  T0_MCR = 0x00000003;                          // Interrupt and Reset on MR0
  T0_TCR = 0000000001;                          // Timer0 Enable

/*
|-------------------------
| Configure Idle Mode
|-------------------------
*/
  SCB_PCONP |=  POWER_CONTROL_TIMER_0;          // Timer-0 will function in idle mode and wake up MicroBolt

  SCB_PCONP &= ~POWER_CONTROL_TIMER_1;          // Timer-1 will not function in idle mode
  SCB_PCONP &= ~POWER_CONTROL_UART_0;           // UART-0 will not function in idle mode
  SCB_PCONP &= ~POWER_CONTROL_UART_1;           // UART-0 will not function in idle mode
  SCB_PCONP &= ~POWER_CONTROL_PWM_0;            // PWM-0 will not function in idle mode
  SCB_PCONP &= ~POWER_CONTROL_I2C;              // I2C serial channel will not function in idle mode
  SCB_PCONP &= ~POWER_CONTROL_SPI;              // SPI serial channel will not function in idle mode
  SCB_PCONP &= ~POWER_CONTROL_RTC;              // Real time clock will not function in idle mode

/*
|--------------------------
| Configure Power down Mode
|--------------------------
*/
  SCB_EXTWAKE = EXT_WAKE_1;                     // EINT1 external interrupt pin can wake up MicroBolt from
                                                // the Power down mode

  __ENABLE_INTERRUPT();                         // Enable all interrupts

/*
|-------------------------------------------------------------------------------
|     Start of application
|-------------------------------------------------------------------------------
*/

  // Blink the MicroBolt LED 10 times to signify Power down mode is coming

  for (Count = 0; Count < 10; Count++)
      {
      GPIO_IOSET = MICROBOLT_LED;               // MicroBolt LED On
      for (Delay = 0; Delay < 500000; Delay++);
      GPIO_IOCLR = MICROBOLT_LED;               // MicroBolt LED Off
      for (Delay = 0; Delay < 500000; Delay++);
      };
/*
|*************************
| Start Power Down Mode
|*************************
*/
  SCB_PCON = POWER_DOWN_MODE;            // Go into Power down mode, wake up is only done via EINT1...

/*
|*************************
| Loop here forever
|*************************              // Only get here if EINT1 wakes up MicroBolt from Power down mode.
*/
  while(1)                               // Do this forever
    {
```

```
    GPIO_IOSET = MICROBOLT_LED;                 // MicroBolt LED On

    for (Count = 0; Count < 10; Count++)        // Blink MicroBolt LED 10 times in approximately 5 seconds
        {
        for (Delay = 0; Delay < 3000000; Delay++);
        };

    GPIO_IOCLR = MICROBOLT_LED;                 // MicroBolt LED Off

/*
|*************************
| Start Idle Mode
|*************************
*/
    SCB_PCON = IDLE_MODE;                       // Done with work, now enter idle mode and go to sleep
                                                // until Timer-0 wakes us up...
    }
  }

/*
|-----------------------------------------------------
|     Functions
|-----------------------------------------------------
*/

/*
|-------------------------------------------------------------------------------------------
|     Function        :   Timer0_ISR
|     Inputs          :   None
|     Outputs         :   None
|     Purpose         :   Interrupt service routine for Timer-0 that wakes up MicroBolt
|     Author          :   Micromint, Inc.
|-------------------------------------------------------------------------------------------
*/

#pragma interrupt_handler Timer0_ISR

void Timer0_ISR (void)
  {                                             // Now we're out of Idle mode

  T0_IR = TIMER_CLR;                            // Clear Timer-0 interrupt flag
  VICVectAddr = VIC_ACK;                        // Acknowledge Interrupt
  }

/*
|-------------------------------------------------------------------------------------------
|     Function        :   ExtInt1_ISR
|     Inputs          :   None
|     Outputs         :   None
|     Purpose         :   Interrupt service routine for EINT1 that wakes up MicroBolt
|     Author          :   Micromint, Inc.
|-------------------------------------------------------------------------------------------
*/

#pragma interrupt_handler ExtInt1_ISR

void ExtInt1_ISR(void)                  // Come here whenever there is a low level on EINT1
  {
                                        // This interrupt takes the MicroBolt out of Power down mode
  SCB_PLLFEED = 0xAA;                  // Turn the PLL back on; Shadow register copy to enable changes
  SCB_PLLFEED = 0x55;                   // in PLLCON and PLLCFG

  SCB_EXTINT |= EXTINT_CLR;             // Clear interrupt
  VICVectAddr = VIC_ACK;                // Acknowledge Interrupt
  }

/*
|-------------------------------------------------------------------------------------------
|     Function        :   pll_isr
|     Inputs          :   None
|     Outputs         :   None
|     Purpose         :   Once PLL has locked, connect it and use for system clock
|     Author          :   Micromint, Inc.
|-------------------------------------------------------------------------------------------
```

```
*/

#pragma interrupt_handler pll_isr

void pll_isr(void)
  {
  SCB_PLLCON |= 0x02;                          // Connect the PLL
  SCB_PLLFEED = 0xAA;                          // Shadow register copy to enable changes
  SCB_PLLFEED = 0x55;                          // in PLLCON and PLLCFG
  VICIntEnClear = PLL_CLR;                     // Clear the PLL interrupt
  VICVectAddr = VIC_ACK;                       // Acknowledge Interrupt
  }
```