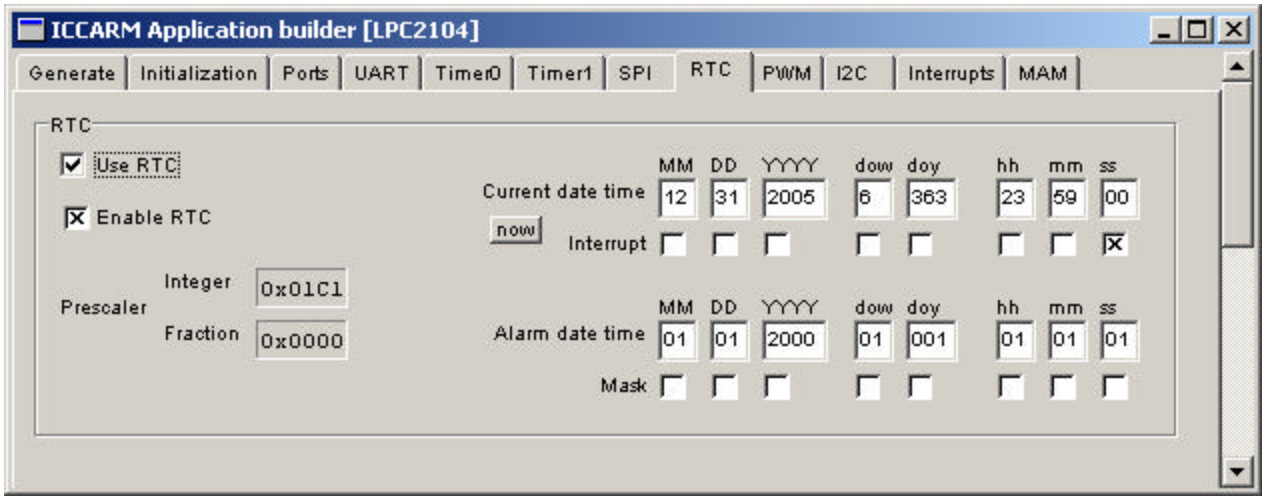
Micromint, Inc	**AN807**
	MicroBolt
Real Time Clock Calendar on the MicroBolt	10/7/2005

Introduction:
This application notes demonstrates how to use the real time clock calendar (RTC) on the MicroBolt.

Background:
The MicroBolt has an available real time clock calendar for keeping track of the current time and date.

How it works:
This ImageCraft ICCARM demo project utilizes the MicroBolt's onboard Real Time Clock (RTC) and outputs the time and date via the serial port to the MicroBolt Serial Debugger. The RTC is setup via the ImageCraft Application builder utility (see below screenshot), and the generated code from App Builder was pasted into the demo project. The RTC is setup to generate an interrupt every second for which the time and date is then outputted.

Screenshot:



The flexibility of the RTC allows for the use of other crystal frequencies besides 32.768 KHz. This allows the MicroBolt's 14.7456 MHz peripheral clock to feed the RTC for accurate RTC functionality. View the LPC2106 datasheet for more information.

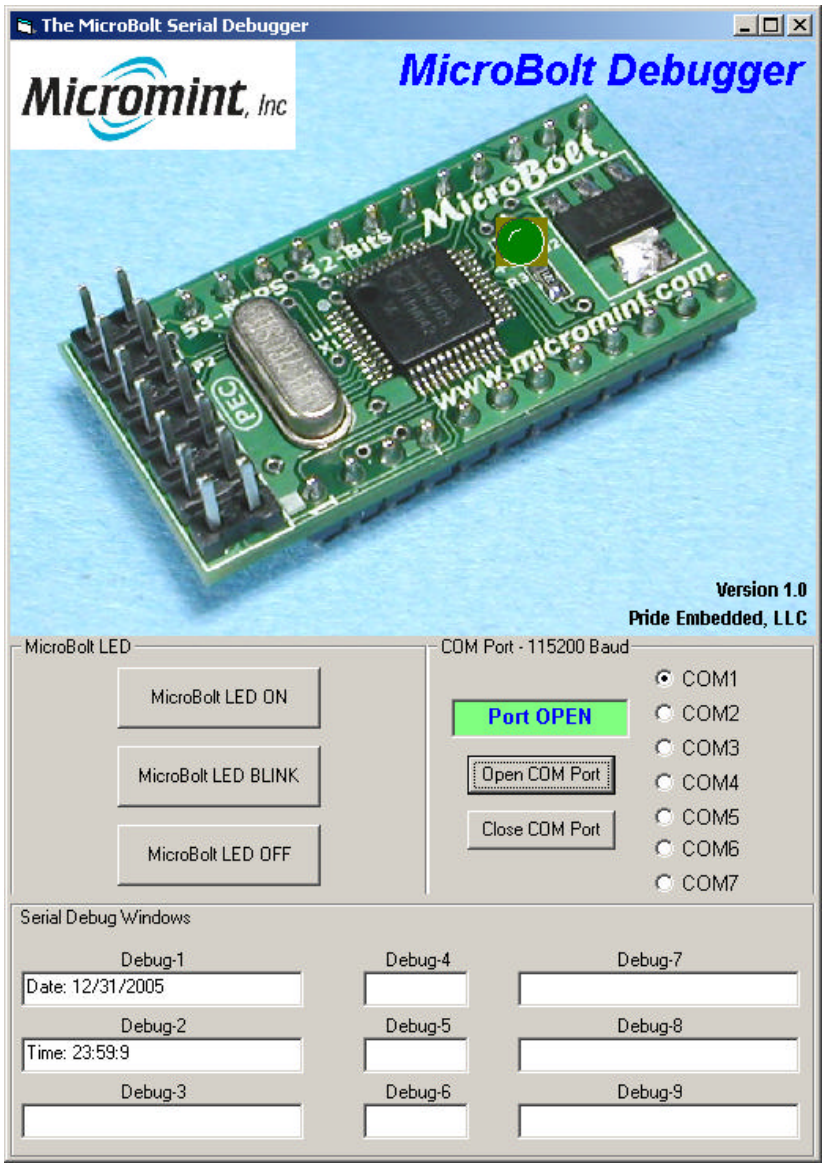
If the RTC must be battery backed, then the MicroBolt's LPC2106 controller can be put into low power mode. View the LPC2106 datasheet for more information.

To Test:
On the MicroBolt development board, setup the UART-0 configuration jumpers accordingly (see MicroBolt datasheet) and connect a serial cable from J1 to the PC.

Now, open up the MicroBolt Serial Debugger application (if you don't have a copy of the MicroBolt Serial Debugger, you can

find it on the MicroBolt CD or the Micromint website). Open the COM port and view the time and date coming from the MicroBolt. The time and date is initially set to Dec 31, 2005 - 23:59:00 PM to show time roll overs. You'll also see the MicroBolt's onboard LED blink every RTC second interrupt. See below screenshot.

Screenshot:



Program Listing:

```
/*
|-----------------------------------------------------------------------------------------
|-----------------------------------------------------------------------------------------
| File Name                     : MicroBolt_Rtc.c
| Author                        : Micromint, Inc.
| Copyright                     : Copyright © 2005, Micromint, Inc.
| Creation Date                 : 9/30/05
| Version                       : 1.00
| Spaces per tab                : 2
| Description                   : Main C file
| Revision                      : Initial
|-----------------------------------------------------------------------------------------
|-----------------------------------------------------------------------------------------
*/
```

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

#include <ARM/philips/lpc210x.h>
#include <arm_macros.h>
#include "MicroBolt_Rtc.h"
#include "SerialDebugger.h"

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

void main(void)
  {
  __DISABLE_INTERRUPT();                        // Disable all interrupts

/*
|-------------------------
| Config 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

/*
|-------------------------
| Config 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

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

/*
|-------------------------
| 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)rtc_isr;               // Assign the RTC ISR vector address
  VICVectCntl1 = INTERRUPT_CHANNEL_FOR_RTC;     // Assign the VIC address to the actual interrupt
  VICIntEnable = INTERRUPT_ENABLE_FOR_RTC;      // Enable the interrupt

/*
|-------------------------
| Config GPIO
|-------------------------
*/
  PCB_PINSEL0=0x00000005;                     // Setup with ICCARM App builder - MicroBolt_Rtc.bcf (UART0)
  PCB_PINSEL1=0x55400000;                     //                                 (Secondary JTAG pins)

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

```
  GPIO_IOCLR=0xffffffff;                        // Clear all pins to start with

/*
|-------------------------
| Config RTC
|-------------------------
*/
  RTC_CCR=0;                    // Configure of RTC done via Imagecraft App builder, open .bcf file to edit
  RTC_CIIR=0x00000001;
  RTC_AMR=0x00000000;
                                // Set the time and date for the RTC - Dec 31, 2005 - 23:59:00 PM
  RTC_SEC=0x00000000;
  RTC_MIN=0x0000003B;
  RTC_HOUR=0x00000017;
  RTC_DOM=0x0000001F;
  RTC_DOW=0x00000006;
  RTC_DOY=0x0000016B;
  RTC_MONTH=0x0000000C;
  RTC_YEAR=0x000007D5;

  RTC_ALSEC=0x00000001;                         // Alarm setting if applicable
  RTC_ALMIN=0x00000001;
  RTC_ALHOUR=0x00000001;
  RTC_ALDOM=0x00000001;
  RTC_ALDOW=0x00000001;
  RTC_ALDOY=0x00000001;
  RTC_ALMON=0x00000001;
  RTC_ALYEAR=0x000007D0;

// Configure the clock divisor for use with MicroBolt crystal - 14.7456 MHz/32.768 KHz

  RTC_PREINT=0x000001C1;

  RTC_PREFRAC=0x00000000;                       // Remainder of above division

  RTC_CCR=0x00000001;                           // Start the RTC

/*
|-------------------------
| Config UART-O
|-------------------------
*/
  UART0_IER = 0x00000001;                        // Receive interrupts
  UART0_FCR = 0x00000001;                        // Enable the fifos
  UART0_LCR = 0x00000083;                        // Enable the divisor
  UART0_DLM = 0;                                // Divisor latch MSB (for baud rates < 4800)
  UART0_DLL = BAUD_RATE_115200;                 // Divisor latch LSB
  UART0_LCR = 0x00000003;                       // Close it, then UART works with divisor

  __ENABLE_INTERRUPT();                         // Enable all interrupts

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

  while(1)                                      // Do this forever
    {
    }
  }

/*
|-------------------------------------------------------------------------------------------
|     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;                     // Disable the PLL interrupt, not needed anymore
  VICVectAddr = VIC_ACK;                       // Acknowledge Interrupt
  }

/*
|-------------------------------------------------------------------------------------------
|     Function        :   rtc_isr
|     Inputs          :   None
|     Outputs         :   None
|     Purpose         :   Interrupt every second
|     Author          :   Micromint, Inc.
|-------------------------------------------------------------------------------------------
*/

#pragma interrupt_handler rtc_isr

void rtc_isr(void)
  {
  unsigned int Delay;

  SerialDebugLed(LED_ON);                                          // MicroBolt Serial Debugger LED on

// Output string to MicroBolt Serial Debugger Debug window 1

  SerialDebug(1,"Date: %d/%d/%d",RTC_MONTH, RTC_DOM, RTC_YEAR);

// Output string to MicroBolt Serial Debugger Debug window 2

  SerialDebug(2,"Time: %d:%d:%d",RTC_HOUR, RTC_MIN, RTC_SEC);

  GPIO_IOSET = MICROBOLT_LED;                                      // MicroBolt LED on
  for (Delay = 0; Delay < 100000; Delay++);                        // Delay for a few
  GPIO_IOCLR = MICROBOLT_LED;                                      // MicroBolt LED off

  SerialDebugLed(LED_OFF);                                         // MicroBolt Serial Debugger LED off

  RTC_ILR = RTC_CLR;                                               // Clear RTC interrupt
  VICVectAddr = VIC_ACK;                                           // Acknowledge Interrupt
  }
```