Micromint, Inc	**AN814**
	MicroBolt
Using the MicroBolt Serial Debugger	10/7/2005

Introduction:
This application notes demonstrates how to send and receive ASCII text to and from UART-0 on the MicroBolt via the MicroBolt Serial Debugger PC application.

Background:
The MicroBolt Serial Debugger provides useful debugging over the MicroBolt's high speed serial port at 115 K Baud. The application is free from Micromint and is a simple alternative for those who do not require high-end and expensive JTAG debugging solutions.

How it works:
This ImageCraft ICCARM demo project outputs data to the MicroBolt Serial Debugger and also accepts a few serial commands to control the MicroBolt's onboard LED. These commands relate to the MicroBolt LED buttons in the PC application. This code demonstrates the transmitting and receiving of serial date via the MicroBolt.

As shown in other MicroBolt demo projects (e.g., MicroBolt real time clock demo project), this serial debug support code can be placed in a separate C file and included in your application to allow for seamless serial debugging.

There are 9 serial debug windows available for serial debug at 115200 baud and one virtual LED.

To send data to a serial window, all that is required is the predefined debug statements and the desired variable you wish to see/debug. To output to a serial debug window, the window number must be defined then your variable or data. This is very similar to a printf with the addition of the debug window selection.

Here is an example of outputting a variable called "Test" to debug window 1:

```
SerialDebug(1,"%d", Test);
```

As shown above a 1 defines debug window number one and Test is the data to be displayed.

Even simpler, here is "Hello World!" to debug window 8:

```
SerialDebug(8,"Hello World!");
```

For the serial debug LED, here's how to turn it on:

```
SerialDebugLed(LED_ON);
```

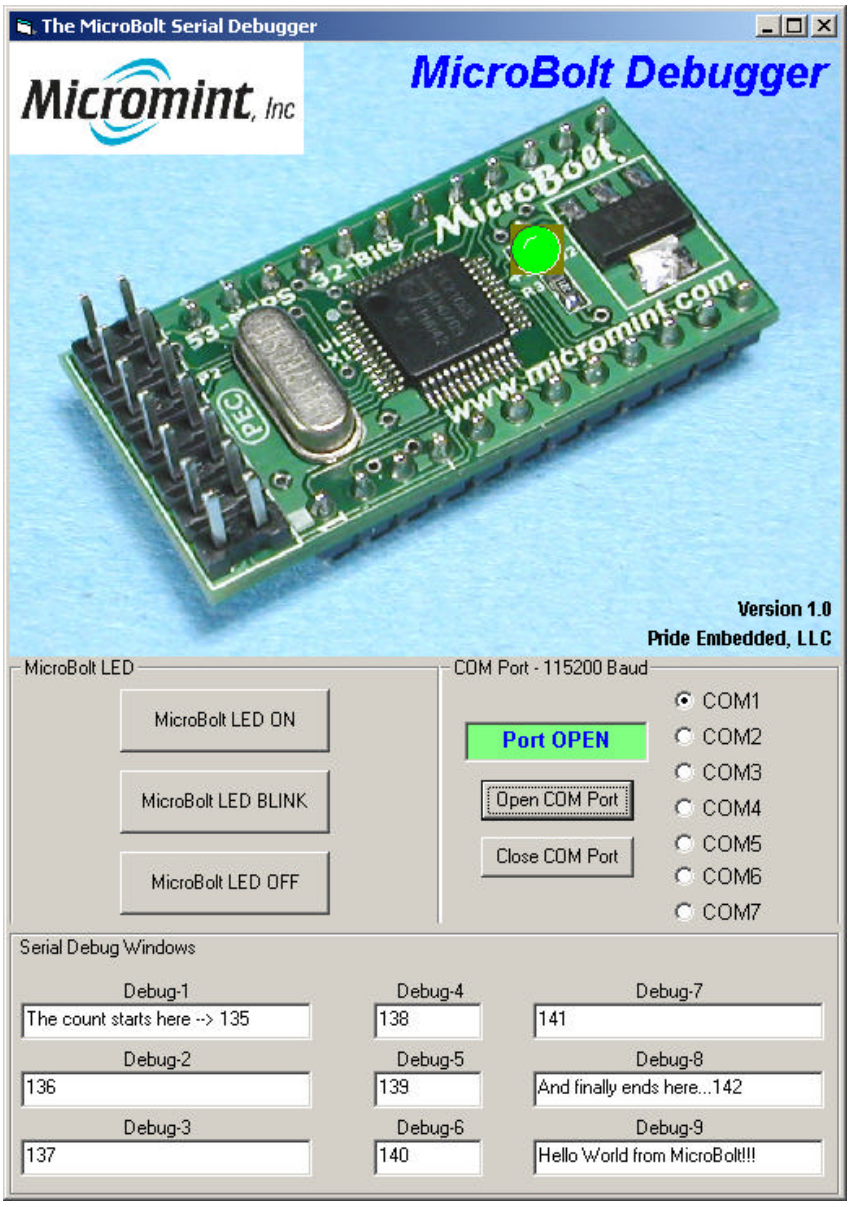
And turn it off:

```
SerialDebugLed(LED_OFF);
```

To test:

Once installed (MicroBolt debugger application .zip file is in the project directory), open up the MicroBolt Serial Debugger PC application and start debugging by connecting up the PC's serial port to the UART-0 serial port of the MicroBolt. What you'll see for data is reflected in the below screenshot.

Screenshot:



Program Listing:

```
/*
|-----------------------------------------------------------------------------------------
|-----------------------------------------------------------------------------------------
| File Name                     : MicroBoltSerialDebugger.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 <STDIO.h>
#include <STDARG.h>

#include "MicroBoltSerialDebugger.h"

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

extern int _textmode;                           // This is defined in the ICCARM library
unsigned char TestLedFlag = LED_BLINK;

void main(void)
  {

  unsigned int Delay;
  unsigned int Count = 0;
  unsigned char Count2 = 0;

  __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 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 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)Uart0_ISR;           // Assign the UART-0 ISR vector address
  VICVectCntl1 = INTERRUPT_CHANNEL_FOR_UART0;   // Assign the VIC address to the actual interrupt
  VICIntEnable = INTERRUPT_ENABLE_FOR_UART0;    // Enable the interrupt

/*
|-------------------------
| Configure GPIO
|-------------------------
*/
  GPIO_IODIR |= MICROBOLT_LED;                  // Setup MicroBolt LED as output
  PCB_PINSEL0 |= P0_0_UART_0_TX;                // Setup P0.0 to alternate function UART0-TX
  PCB_PINSEL0 |= P0_1_UART_0_RX;                // Setup P0.1 to alternate function UART0-RX
  GPIO_IOCLR=0xffffffff;                        // Clear all pins to start with

/*
|-------------------------
| Configure 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
    {
    Count++;                                            // Increment test counter

    SerialDebug(1,"The count starts here --> %d", Count); // Output string to Debug window 1
    SerialDebug(2,"%d", Count+1);                       // Output string to Debug window 2
    SerialDebug(3,"%d", Count+2);                       // Output string to Debug window 3
    SerialDebug(4,"%d", Count+3);                       // Output string to Debug window 4
    SerialDebug(5,"%d", Count+4);                       // Output string to Debug window 5
    SerialDebug(6,"%d", Count+5);                       // Output string to Debug window 6
    SerialDebug(7,"%d", Count+6);                       // Output string to Debug window 7
    SerialDebug(8,"And finally ends here...%d", Count+7); // Output string to Debug window 8
    SerialDebug(9,"Hello World from MicroBolt!!!");     // Output string to Debug window 9

    for (Delay = 0; Delay < 370000; Delay++);           // Delay for a few

/*
|-------------------------
| LED control
|-------------------------
*/
    if (TestLedFlag == LED_BLINK)
      {
      Count2++;                                         // Now simply blink the serial debugger LED
      if (Count2 == 10)
        {
        GPIO_IOSET = MICROBOLT_LED;
        SerialDebugLed(LED_ON);
        }
      else
      if (Count2 == 20)
        {
        Count2 = 0;
        GPIO_IOCLR = MICROBOLT_LED;
        SerialDebugLed(LED_OFF);
        }
      }
    else
```

```
    if (TestLedFlag == LED_ON)
      {
      GPIO_IOSET = MICROBOLT_LED;                       // MicroBolt LED On
      SerialDebugLed(LED_ON);
      }
    else
    if (TestLedFlag == LED_OFF)
      {
      GPIO_IOCLR = MICROBOLT_LED;                       // MicroBolt LED Off
      SerialDebugLed(LED_OFF);
      }
    }
}

/*
|-------------------------------------------------------------------------------------------
|     Function        :   SerialDebug
|     Inputs          :   Printf for serial debugger
|     Outputs         :   None
|     Purpose         :   Transmit data to the MicroBolt Serial Debugger
|     Author          :   Micromint, Inc.
|-------------------------------------------------------------------------------------------
*/

void SerialDebug(char DebugBoxNum, char *fmt,...)
    {
    va_list va;
    va_start(va, fmt);
    printf("%d", DebugBoxNum);
    vprintf(fmt,va);
    printf("\n");
    }

/*
|-------------------------------------------------------------------------------------------
|     Function        :   SerialDebugLed
|     Inputs          :   LED for serial debugger
|     Outputs         :   None
|     Purpose         :   Transmit data to the MicroBolt Serial Debugger
|     Author          :   Micromint, Inc.
|-------------------------------------------------------------------------------------------
*/

void SerialDebugLed(char LedOnOrOff)
    {
    if (LedOnOrOff == LED_ON)
      {
      printf("L1\n");
      }
    else
    if (LedOnOrOff == LED_OFF)
      {
      printf("L0\n");
      }
    }

/*
|-------------------------------------------------------------------------------------------
|     Function        :   putchar
|     Inputs          :   Byte for transmission
|     Outputs         :   None
|     Purpose         :   Transmit a byte via UART0 to enable printf
|     Author          :   Micromint, Inc.
|-------------------------------------------------------------------------------------------
*/

int putchar(char SerialByte)
  {
  _textmode = 1;                                  // Turn carriage return line feed on for printf
  if (_textmode == 1 && SerialByte == '\n')
      {
      putchar('\r');
      }
  while (!(UART0_LSR & 0x20));
```

```
  UART0_THR = SerialByte;
  return SerialByte;
  }

/*
|-------------------------------------------------------------------------------------------
|     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        :   Uart0_ISR
|     Inputs          :   None
|     Outputs         :   None
|     Purpose         :   Interrupt service routine for Uart-0
|     Author                      :   Micromint, Inc.
|-------------------------------------------------------------------------------------------
*/

#pragma interrupt_handler Uart0_ISR

void Uart0_ISR (void)
  {
  static unsigned char CharRxCount = 0;
  static unsigned char SerialBuffer[20];

  switch(UART0_IIR & 0x0F)
      {
      case 0x04:                                        // Receive data available

/*
|-----------------------------------------------------
|     Get serial port data and check for commands
|-----------------------------------------------------
*/
      SerialBuffer[CharRxCount] = UART0_RBR;

      if (SerialBuffer[0] == 'L')
          {
          if (CharRxCount == 2)
              {
              switch(SerialBuffer[1])
                  {
                  case 'A':
                      {
                      TestLedFlag = LED_ON;
                      break;
                      }
                  case 'B':
                      {
                      TestLedFlag = LED_BLINK;
                      break;
                      }
                  case 'C':
                      {
                      TestLedFlag = LED_OFF;
                      break;
                      }
                  default:
```

```
                    {
                    break;
                    }
                }
            CharRxCount = 0;
            }
        else
            {
            CharRxCount++;
            }
        }
    else
        {
        CharRxCount = 0;
        }
    break;

    default:
    break;
    }
  VICVectAddr = VIC_ACK;                          // Acknowledge Interrupt
  }
```