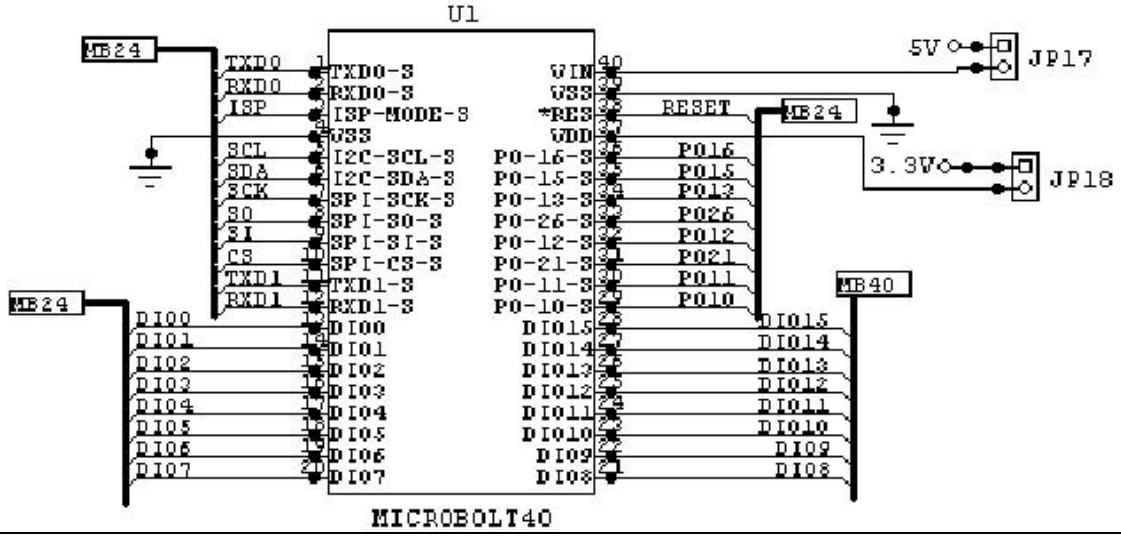
Micromint, Inc	**AN808**
	MicroBolt
Analog to Digital Conversion with the MicroBolt	10/9/2005

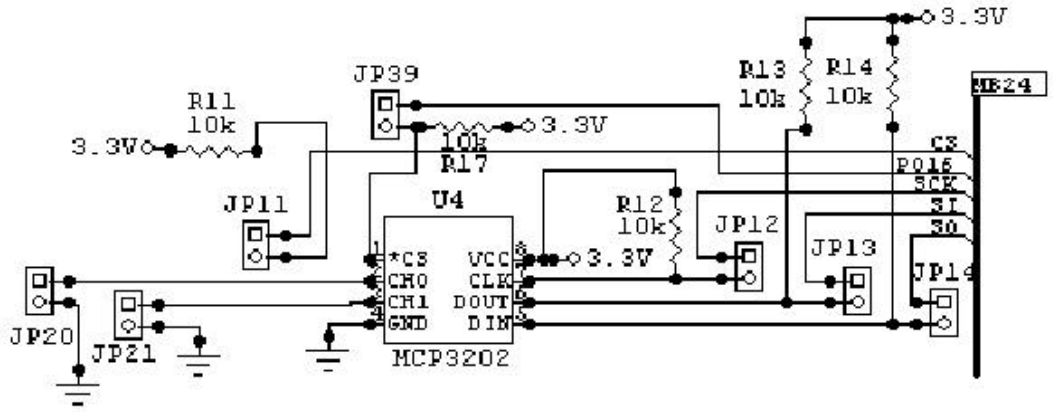
Introduction:
This application notes demonstrates how to add analog to digital conversion to the MicroBolt embedded module.

Background:
Many applications require the reading of an analog voltage by the MicroBolt. The MicroBolt must convert this analog voltage to a digital reading in order to use and operate on the voltage reading in its program. Due to the varying analog to digital requirements of different applications (e.g., speed, number of bits, number of channels, single ended or differential, etc.), a single analog to digital converter (ADC) device will not fulfill each applications need. It's this reason that the MicroBolt does not contain an onboard ADC. The MicroBolt does however provide the user with industry standard serial busses, which allow the user to hook up an external ADC. This allows the user to select the appropriate ADC for their application.

In this example, the MicroBolt will control a Microchip MCP3202, 2-Channel, 12-bit ADC over the SPI serial bus.

Schematic:





How it works:
This ImageCraft ICCARM demo project is used with the MicroBolt development board and interfaces the MicroBolt to a Microchip 12 bit, 2 channel, A to D converter; the MCP3202. The MCP3202 is controlled by the MicroBolt's SPI serial bus at an approximate frequency of 170 KHz.

The MicroBolt is the SPI master and the MCP3202 is an SPI slave.

The MicroBolt Serial Debugger is also utilized to allow reading of the 2 A to D channels via a PC application over an RS232 serial connection at 115200 Baud.

MCP3202 2-Channel 12-bit ADC info:
The MCP3202 (U4) is a successive approximation 12-bit Analog to Digital Converter with on board sample and hold circuitry. It is programmable to provide a single pseudo-differential input pair or dual single-ended inputs. Communication to the ADC is done using MicroBolt's SPI bus. The ADC's chip select can be connected to P0.16 of the MicroBolt by adding a jumper to JP39. The clock signal can be connected to P0.4 of the MicroBolt by adding a jumper to JP12. In order for the SPI bus to operate JP11 must have a jumper connected to it. JP11 pulls P0.7 of the MicroBolt up to 3.3V through a 10k

Required tools and/or parts:
-Microchip 8 pin MCP3202 A to D converter (can be purchased from Digikey)
-MicroBolt development kit (Board, cables, power supply, schematics, etc.)
-MicroBolt Serial Debugger (On MicroBolt development kit CD or Micromint website)

Instructions:
1. Plug MCP3202 A to D converter into U4 socket on MicroBolt development board.

2. Signals SI and SO on JP13 and JP14 must be crossed in order for the SPI bus to work properly. Reference the MicroBolt development board schematic on the MicroBolt datasheet and wire wrap JP13 pin 1 to JP14 pin 2 and JP14 pin 1 to JP13 pin 2. Leave the plastic jumper blocks off from JP13 and JP14. This fixes the SPI routing to the MCP3202.

3. Connect an analog signal to JP20 and another to JP21; a potentiometer works well for testing purposes.

4. Connect serial port cable from PC to J1 of MicroBolt development board (verify COM port jumpers via the MicroBolt datasheet)

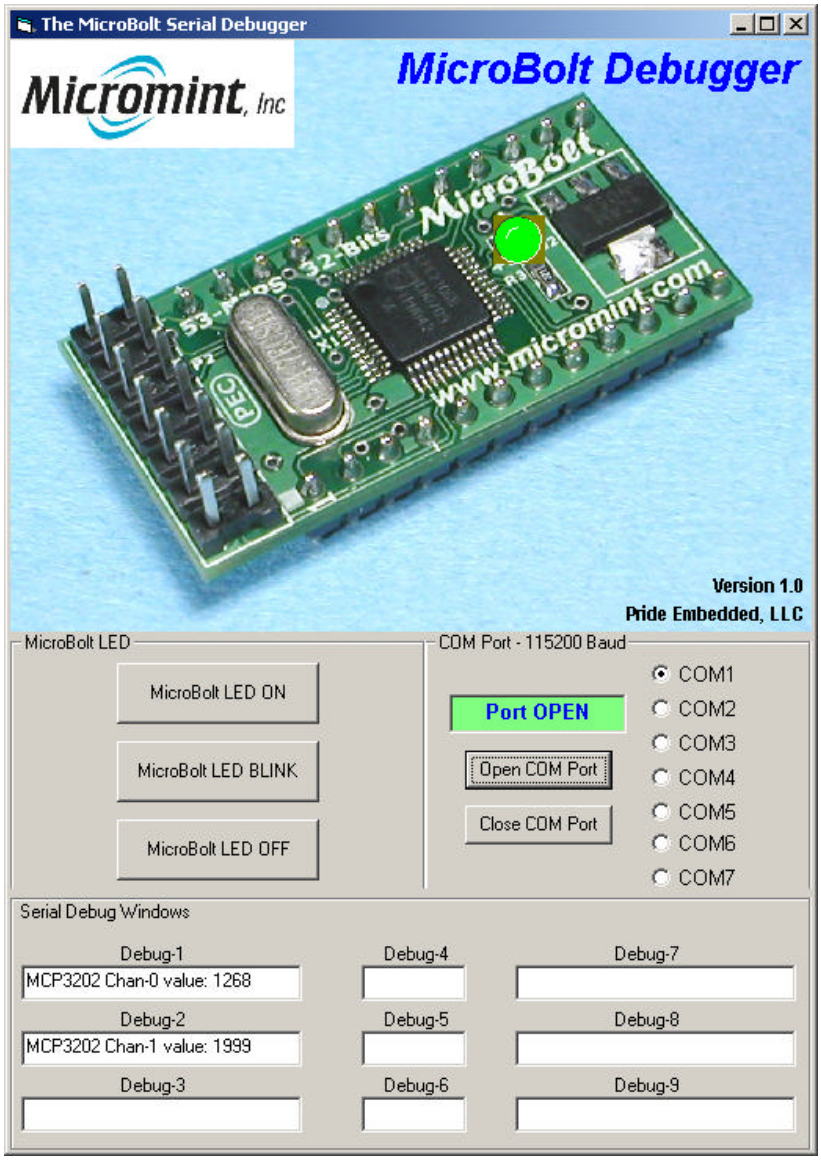
5. Power up the board and download the demo project to the board.

6. Open up the MicroBolt Serial Debugger, select the desired COM port, and open it.

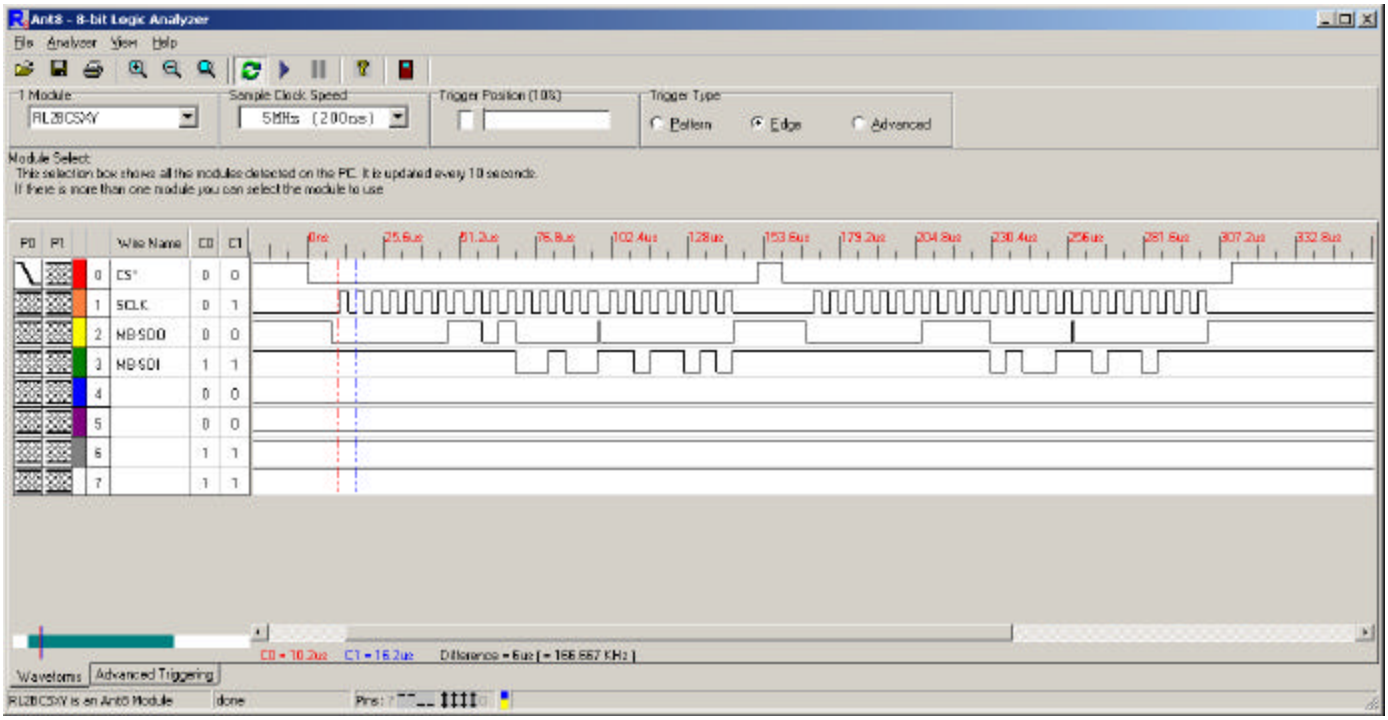
7. A to D data should now be displayed in debug windows 1 and 2 (See below MicroBolt Serial Debugger screenshot).

8. Vary the analog input voltage and verify the displayed data in the debugger changes accordingly.

MicroBolt Serial Debugger screenshot:



Waveforms:
The following waveform capture shows two reads happening over the MicroBolt's SPI bus. First channel-0 is read, and then channel-1 is read.



Program Listing:

```
/*
|-----------------------------------------------------------------------------------------
|-----------------------------------------------------------------------------------------
| File Name                     : MicroBolt_AtoD.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_AtoD.h"
#include "SerialDebugger.h"

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

```
*/
void main(void)
  {
  unsigned int Delay = 0;
  unsigned char Count = 0;
  unsigned short AtoDvalueChan0, AtoDvalueChan1;

  __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

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

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

/*
|-------------------------
| Config GPIO
|-------------------------
*/
  PCB_PINSEL0=0x00005505;        // Setup with ICCARM App builder - MicroBolt_AtoD.bcf (UART0 and SPI pins)
  PCB_PINSEL1=0x55400000;        //                                                    (Secondary JTAG pins)

  GPIO_IODIR |= MICROBOLT_LED;                  // Setup MicroBolt LED as output
  GPIO_IODIR |= A_TO_D_CHIP_SELECT;             // Setup A to D chip select - P0.16

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

  GPIO_IOSET = A_TO_D_CHIP_SELECT;              // Chip select high to start with - Disabled

/*
|-------------------------
| Config SPI (Master)
|-------------------------
*/
 SPI_SPCR=0x00000020;                           // Setup edges
 SPI_SPCCR=0x00000052;                          // Clock frequency 170 KHz

/*
|-------------------------
| 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
    {
    AtoDvalueChan0 = ReadAtoD(CHANNEL_0);                           // Read A to D channel 0
    AtoDvalueChan1 = ReadAtoD(CHANNEL_1);                           // Read A to D channel 1

// Output string to MicroBolt Serial Debugger Debug window 1

    SerialDebug(1,"MCP3202 Chan-0 value: %d", AtoDvalueChan0);

// Output string to MicroBolt Serial Debugger Debug window 2

    SerialDebug(2,"MCP3202 Chan-1 value: %d", AtoDvalueChan1);

    for (Delay = 0; Delay < 870000; Delay++);                       // Delay for 100 mS

    Count++;                                                        // Now simply blink the serial debugger LED
    if (Count == 10)
      {
      SerialDebugLed(LED_ON);
      GPIO_IOSET = MICROBOLT_LED;
      }
    else
    if (Count == 20)
      {
      Count = 0;
      SerialDebugLed(LED_OFF);
      GPIO_IOCLR = MICROBOLT_LED;
      }
    }
}

/*
|-------------------------------------------------------------------------------------------
|     Function        :   ReadAtoD
|     Inputs          :   None
|     Outputs         :   Analog value from the A to D
|     Purpose         :   Read A to D values from MCP3202
|     Author          :   Micromint, Inc.
|-------------------------------------------------------------------------------------------
*/
unsigned short ReadAtoD(unsigned char AtoDchannelNumber)
  {
        unsigned char AnalogToDigitalMsbValue;
        unsigned char AnalogToDigitalLsbValue;
        unsigned short AnalogToDigital12bitValue;

        GPIO_IOCLR = A_TO_D_CHIP_SELECT;                                    // Chip select low - Enable
        Delay10uS();                                                        // Setup time

        SPI_SPDR = A_TO_D_START_SINGLE;                                     // Send out first byte to A to D
        while(!(SPI_SPSR & SPIF));                                          // Wait for completion

  if (AtoDchannelNumber == CHANNEL_0)                 // Send out 2nd byte, data depends on channel number
      {
           SPI_SPDR = A_TO_D_CHANNEL_0;
      }
  else
      {
           SPI_SPDR = A_TO_D_CHANNEL_1;
      }
        while(!(SPI_SPSR & SPIF));                                      // Wait for completion
```

```
// Got the first desired byte (MSB) of A to D data, now store it

        AnalogToDigitalMsbValue =  SPI_SPDR;

// Send out 3rd byte of data so as to get the second A to D byte

        SPI_SPDR = A_TO_D_NULL_WAIT_FOR_DATA;
        while(!(SPI_SPSR & SPIF));                                      // Wait for completion

// Got the second byte (LSB) of A to D data, now store it

        AnalogToDigitalLsbValue =  SPI_SPDR;

        Delay10uS();                                                    // Hold time
        GPIO_IOSET = A_TO_D_CHIP_SELECT;                                // Chip select high - Disable

// Now create 12 bit A to D value...
// Drop top 4 bits from MSB for 12 bit value

        AnalogToDigitalMsbValue = AnalogToDigitalMsbValue & 0x0F;

// Store MSB by shifting up 8

        AnalogToDigital12bitValue = (unsigned short)AnalogToDigitalMsbValue << 8;

// OR in the LSB to get the final value

        AnalogToDigital12bitValue = AnalogToDigital12bitValue | (unsigned short) AnalogToDigitalLsbValue;

// 12 bit A to D value read from MCP3202, return it

        return(AnalogToDigital12bitValue);
  }

/*
|-------------------------------------------------------------------------------------------
|     Function        :   Delay10uS
|     Inputs          :   None
|     Outputs         :   None
|     Purpose         :   Delay for 10 uS when PLLCFG is set to 0x23 (58.98 MHz)
|     Author          :   Micromint, Inc.
|-------------------------------------------------------------------------------------------
*/

void Delay10uS(void)
  {
        volatile unsigned int Delay;

  for (Delay = 0; Delay < 35; Delay++);          // Delay for 10 uS
  }

/*
|-------------------------------------------------------------------------------------------
|     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
  }
```