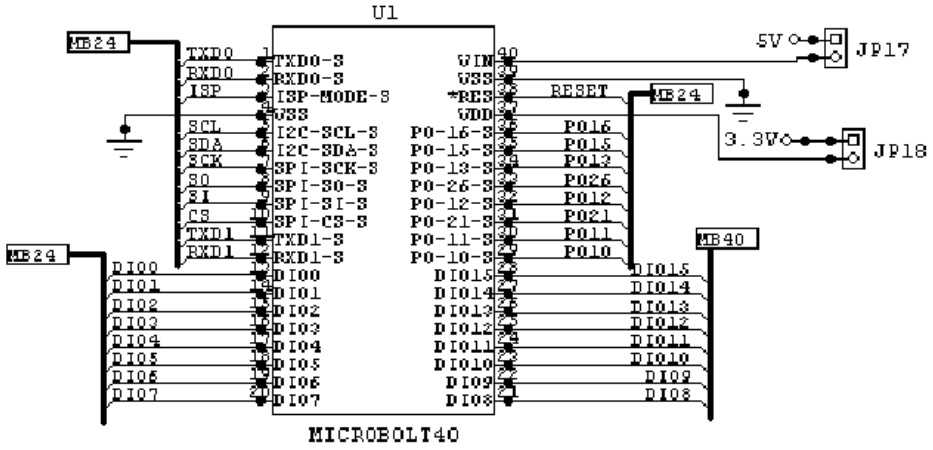
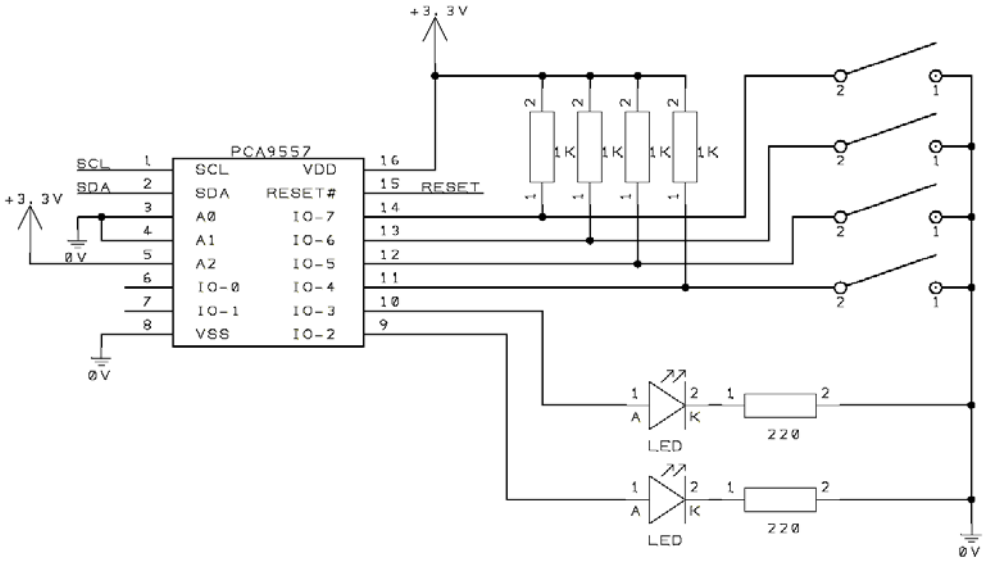
Micromint, Inc	**AN820**
	MicroBolt
The MicroBolt with I²C GPIO Expansion	12/30/2005

Introduction:
This application notes demonstrates how to expand the MicroBolt's GPIO via the Philips PCA9557 I²C 8-bit I/O port device.

Background:
The MicroBolt has 19 GPIO pins available to the user, but more may be required depending on the application. This application note shows how to add 8 more GPIO (General Purpose Input/Output) pins by interfacing to a Philips PCA9557 8-bit I²C I/O port device. For more information on the Philips PCA9557 visit the Philips site: http://www.semiconductors.philips.com/buses/i2c

Schematic:





How it works:

This ImageCraft ICCARM demo project utilizes the MicroBolt's I²C serial channel and demonstrates GPIO (general purpose input/output) pin expansion for the MicroBolt. The example code sets up the PCA9557 IO-2 and IO-3 port pins as outputs and the rest (IO-0, IO-1, IO-4, IO-5, IO-6, IO-7) as inputs. The PCA9557 IO-4 through IO-7 port pins are constantly scanned by the MicroBolt via the I²C bus, and depending on the value read, the PCA9557 IO-2 and IO-3 outputs are written to accordingly.

The following PCA9557 functions, and their example use, are shown below:

```
WritePCA9557IoPort(0x04);                  // Turn PCA9557 IO-2 LED On
PCA9557portValue = ReadPCA9557IoPort();    // Read 8-bit port value from the PCA9557 via I²C
```

As shown, for writing to the PCA9557, all that's required is the 8-bit value. For reading, all that is required is the function call and then the 8-bit data read from the port is returned.

This provides easy to use read and write functions for the MicroBolt when interfacing to the PCA9557 over the I²C bus.

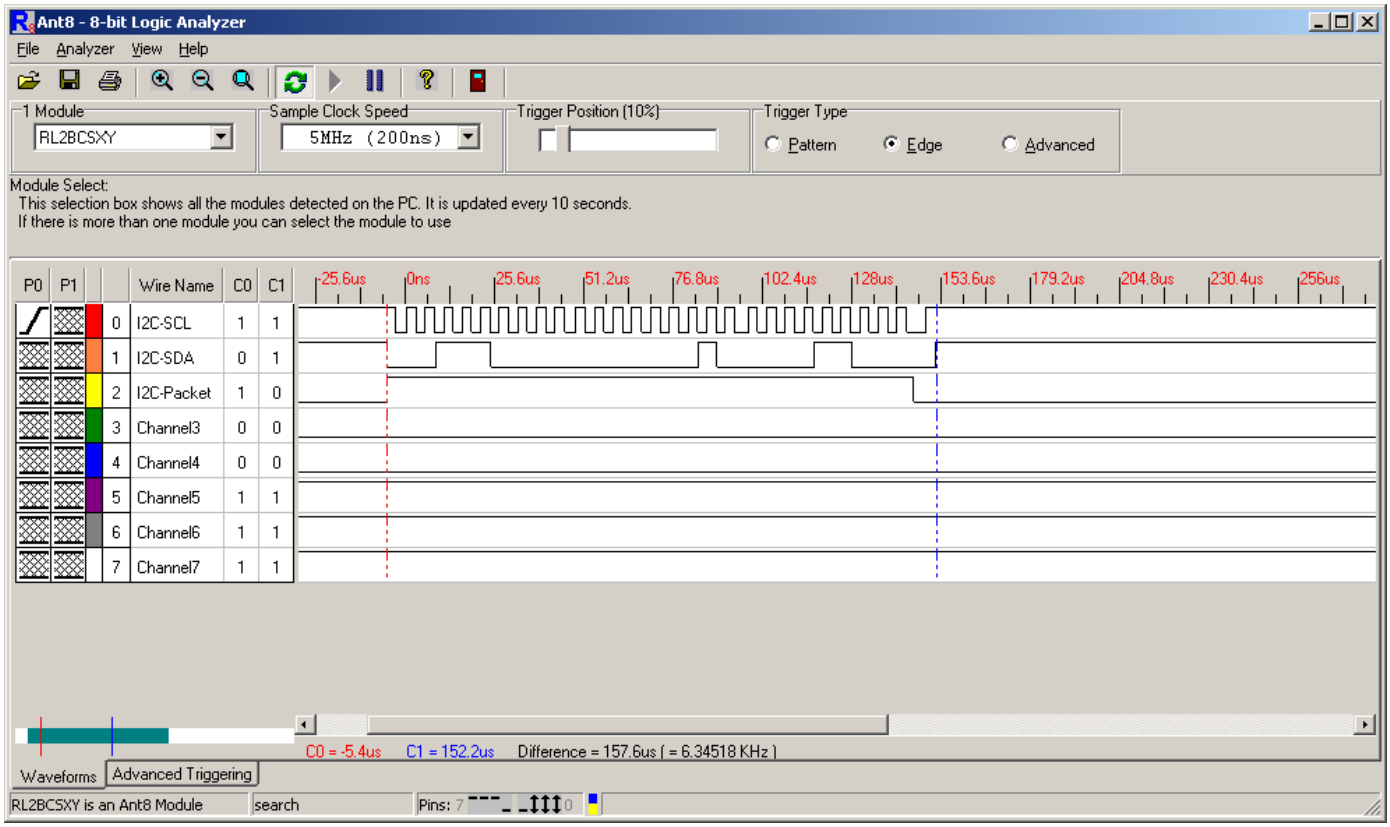
Notes:

The MicroBolt is setup as a master on the I²C bus and a Philips PCA9557 is setup as a slave. The PCA9557 slave address is set to 0x1C. The PCA9557 was wired onto the MicroBolt development board prototyping area and drives two LEDs via IO-2 and IO-3. IO-4 through IO-7 are inputs and are connected to a 4 position DIP Switch

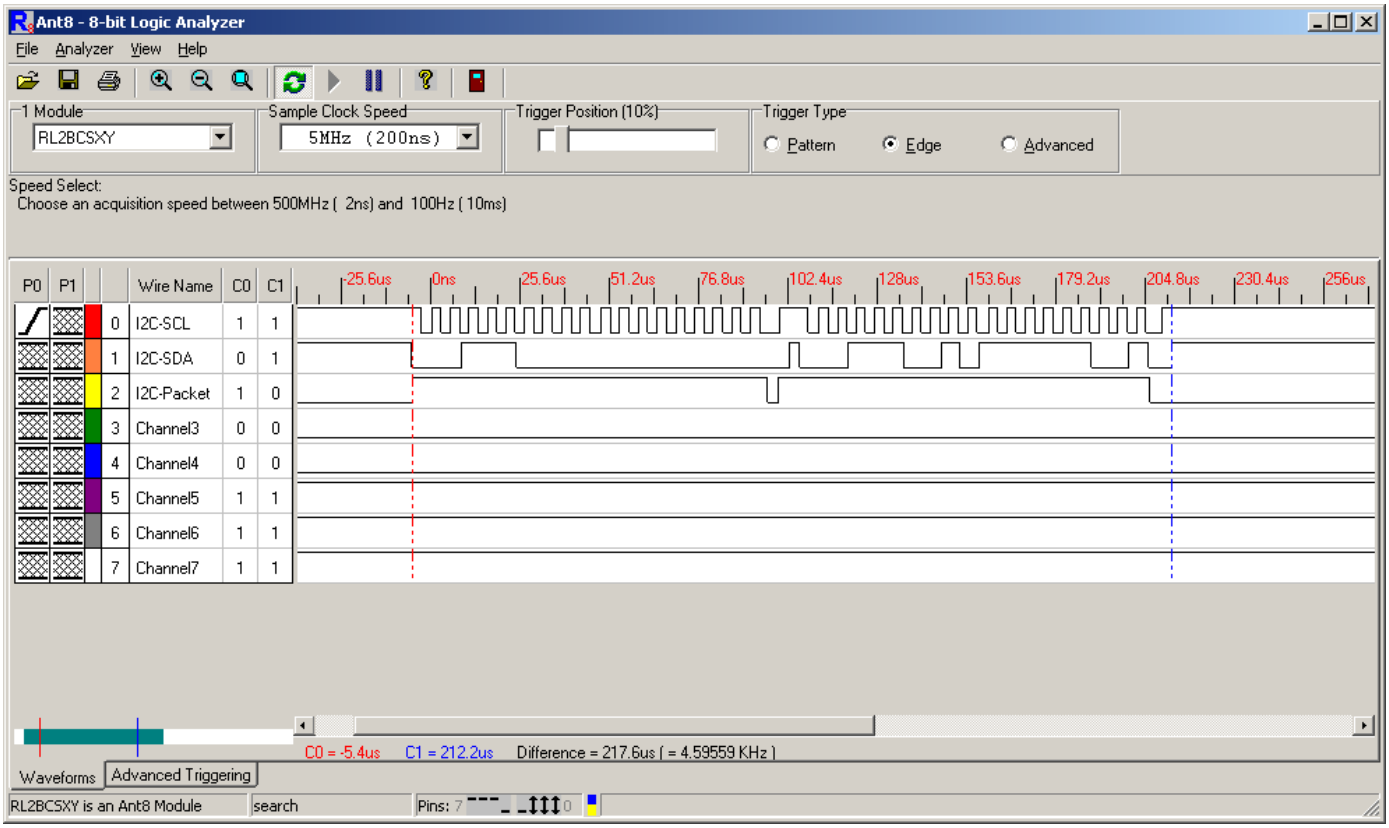
Be sure to install jumpers JP37 and JP38 to enable I²C on the MicroBolt development board.

Waveforms:

The following waveform capture shows 3 bytes being written from the MicroBolt to the PCA9557 over the I²C bus. The first byte is the PCA9557 slave address of 0x1C. The second byte is 0x01 for selecting the PCA9557 output register and the third is 0x0C, which turns on IO-2 and IO-3 of the PCA9557.



The following waveform capture shows two bytes being written and then one being read from the PCA9557 by the MicroBolt over the I²C bus. For the first I²C packet the first byte is the PCA9557 slave address of 0x1C and the second byte is 0x00 that selects the PCA9557 input register. For the second I²C packet the first byte is the PCA9557 slave address of 0x1C and the second byte is 0xFC which is the value read from the PCA9557 input register.



Program Listing:

```
/*
|-----------------------------------------------------------------------------------------
|-----------------------------------------------------------------------------------------
| File Name                     : MicroBolt_I2C_GPIO.c
| Author                        : Micromint, Inc.
| Copyright                     : Copyright © 2005, Micromint, Inc.
| Creation Date                 : 12/30/05
| 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_I2C_GPIO.h"

static unsigned char I2cSlaveAddress = 0;       // I2C slave address
```

```
static unsigned char I2cBuffer[20];             // I2C application buffer
static unsigned char I2cBufferIndex = 0;        // Index for I2C buffer
static unsigned char I2cPacketDataSize = 0;     // Number of data bytes for an I2C packet
static unsigned char I2cPacketType = 0;         // Read or Write packet type
static unsigned char I2cPacketInProgress = 0;   // I2C Packet in progress flag

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

void main(void)
  {
  unsigned char PCA9557portValue = 0;

  __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)I2c_ISR;              // Assign the I2C ISR vector address
  VICVectCntl1 = INTERRUPT_CHANNEL_FOR_I2C;     // Assign the VIC address to the actual interrupt
  VICIntEnable = INTERRUPT_ENABLE_FOR_I2C;      // Enable the interrupt

  __ENABLE_INTERRUPT();                         // Enable all interrupts

/*
|-------------------------
| Config GPIO
|-------------------------
*/
  PCB_PINSEL0=0x00000050;                 // Setup with ICCARM App builder - MicroBolt_I2C_GPIO.bcf (I2C)
  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 I2C Master
|-------------------------
*/
  I2C_I2SCLH = 37;                                // I2C clock is 200 KHz (14.7456 MHz/(SCLH + SCLL))
  I2C_I2SCLL = 37;

  I2C_I2CONSET = I2C_ENABLE_BIT;                  // Enable the I2C channel

/*
|----------------------------------------
| Config I2C PCA9557 Slave device I/O port
|----------------------------------------
*/                                                // Initialize I2C buffers with desired data for PCA9557
slave
                                                  // The below configuration sets up the PCA9557 port pins
  I2cSlaveAddress = I2C_SLAVE_ADDR_PCA9557;       // Address the PCA9557
  I2cPacketDataSize = 2;                          // How many bytes from the I2c buffer to send
  I2cBuffer[0] = 0x03; // Control register value 0x03 selects the configuration register to define I/O port
  I2cBuffer[1] = 0xF3; // IO-2 and IO-3 are outputs and the rest are inputs (0's outputs, 1's inputs)
  I2cStart(I2C_WRITE);                            // Send out an I2C Start condition for a Write packet
  while(I2cPacketInProgress == TRUE);             // Wait here for I2C packet to complete

/*
|-------------------------------------------------
| Config I2C PCA9557 Slave device polarity register
|-------------------------------------------------
*/                              // Initialize I2C buffers with desired data for PCA9557 slave
                                // The below configuration sets up the PCA9557 polarity inversion register
  I2cSlaveAddress = I2C_SLAVE_ADDR_PCA9557;       // Address the PCA9557
  I2cPacketDataSize = 2;                          // How many bytes from the I2c buffer to send
  I2cBuffer[0] = 0x02;          // Control register value 0x02 selects the polarity inversion register
  I2cBuffer[1] = 0x00;                            // All polarities match actual logic levels
  I2cStart(I2C_WRITE);                            // Send out an I2C Start condition for a Write packet
  while(I2cPacketInProgress == TRUE);             // Wait here for I2C packet to complete

/*
|-------------------------------------------------
|     Start of application
|-------------------------------------------------
*/
  while(1)                                        // Do this forever
    {

/*
|-----------------------------------------------
| Read PCA9557 I2C Slave device's input register
|-----------------------------------------------
*/
    PCA9557portValue = ReadPCA9557IoPort();       // Read 8-bit port value from the PCA9557 via I2C

/*
|-------------------------------------------------------------
| Write to the PCA9557 port based upon the value read above
|-------------------------------------------------------------
*/
     switch(PCA9557portValue & 0xF0)     // Take port value read from PCA9557 and mask off lower nibble
                        {                // High nibble IO-4 to IO-7 is now what's left
       case (0x10):                      // Is I0-4 the only one high?
            {
            WritePCA9557IoPort(0x04);   // Yes, Turn PCA9557 IO-2 LED On
            GPIO_IOCLR = MICROBOLT_LED; // Keep the MicroBolt LED Off
            break;
            }
       case (0x20):                      // Is I0-5 the only one high?
            {
            WritePCA9557IoPort(0x08);   // Yes, Turn PCA9557 IO-3 LED On
            GPIO_IOCLR = MICROBOLT_LED; // Keep the MicroBolt LED Off
            break;
            }
       case (0x40):                      // Is I0-6 the only one high?
```

```
            {
            WritePCA9557IoPort(0x0C);   // Yes, Turn PCA9557 IO-2 & IO-3 LED's On
            GPIO_IOCLR = MICROBOLT_LED; // Keep the MicroBolt LED Off
            break;
            }
        case (0x80):                             // Is I0-7 the only one high?
            {
            WritePCA9557IoPort(0x00);           // Yes, Turn PCA9557 IO-2 & IO-3 LED's Off
            GPIO_IOSET = MICROBOLT_LED;         // Turn the MicroBolt LED On
            break;
            }
        case (0xF0):                             // Are I0-4, IO-5, IO-6, and IO-7 all high?
            {
            WritePCA9557IoPort(0x0C);           // Yes, Turn PCA9557 IO-2 & IO-3 LED's On
            GPIO_IOSET = MICROBOLT_LED;         // Turn the MicroBolt LED On
            break;
            }
        default:                                // Any other combination of IO-4 through IO-7 ends up here
            {
            WritePCA9557IoPort(0x00);           // Turn PCA9557 IO-2 & IO-3 LED's Off
            GPIO_IOCLR = MICROBOLT_LED;         // Keep the MicroBolt LED Off
             break;
            }
         }
    }
  }

/*
|--------------------------------------------------------------------------------------------
|     Function        :   WritePCA9557IoPort
|     Inputs          :   IoPortPinValue
|     Outputs         :   None
|     Purpose         :   Write an 8 bit value to the PCA9557 IO port
|     Author          :   Micromint, Inc.
|--------------------------------------------------------------------------------------------
*/
void WritePCA9557IoPort(unsigned char IoPortPinValue)
  {
  I2cSlaveAddress = I2C_SLAVE_ADDR_PCA9557;     // Address the PCA9557
  I2cPacketDataSize = 2;                        // How many bytes from the I2c buffer to send
  I2cBuffer[0] = 0x01;                          // Control register value 0x01 selects the output register
  I2cBuffer[1] = IoPortPinValue;                // Desired IO port value
  I2cStart(I2C_WRITE);                          // Send out an I2C Start condition for a Write packet
  while(I2cPacketInProgress == TRUE);           // Wait here for I2C packet to complete
  }

/*
|--------------------------------------------------------------------------------------------
|     Function        :   ReadPCA9557IoPort
|     Inputs          :   None
|     Outputs         :   PCA9557 8-bit port value
|     Purpose         :   Read an 8 bit value from the PCA9557 IO port
|     Author          :   Micromint, Inc.
|--------------------------------------------------------------------------------------------
*/
unsigned char ReadPCA9557IoPort(void)
  {
  I2cSlaveAddress = I2C_SLAVE_ADDR_PCA9557;     // Address the PCA9557
  I2cPacketDataSize = 1;                        // How many bytes from the I2c buffer to send
  I2cBuffer[0] = 0x00;          // Control register value 0x00 selects input register for reading port pins
  I2cStart(I2C_WRITE);                          // Send out an I2C Start condition for a Write packet
  while(I2cPacketInProgress == TRUE);           // Wait here for previous I2C packet to complete

  I2cSlaveAddress = I2C_SLAVE_ADDR_PCA9557;     // Address the PCA9557
  I2cPacketDataSize = 1;                        // How many bytes from the I2c buffer to receive
  I2cStart(I2C_READ);                           // Send out an I2C Start condition for a Read packet
  while(I2cPacketInProgress == TRUE);           // Wait here for previous I2C packet to complete

  return(I2cBuffer[0]);                         // Return the byte read from the PCA9557 IO port
  }

  See ImageCraft ICCARM demo project for the rest of the code
```