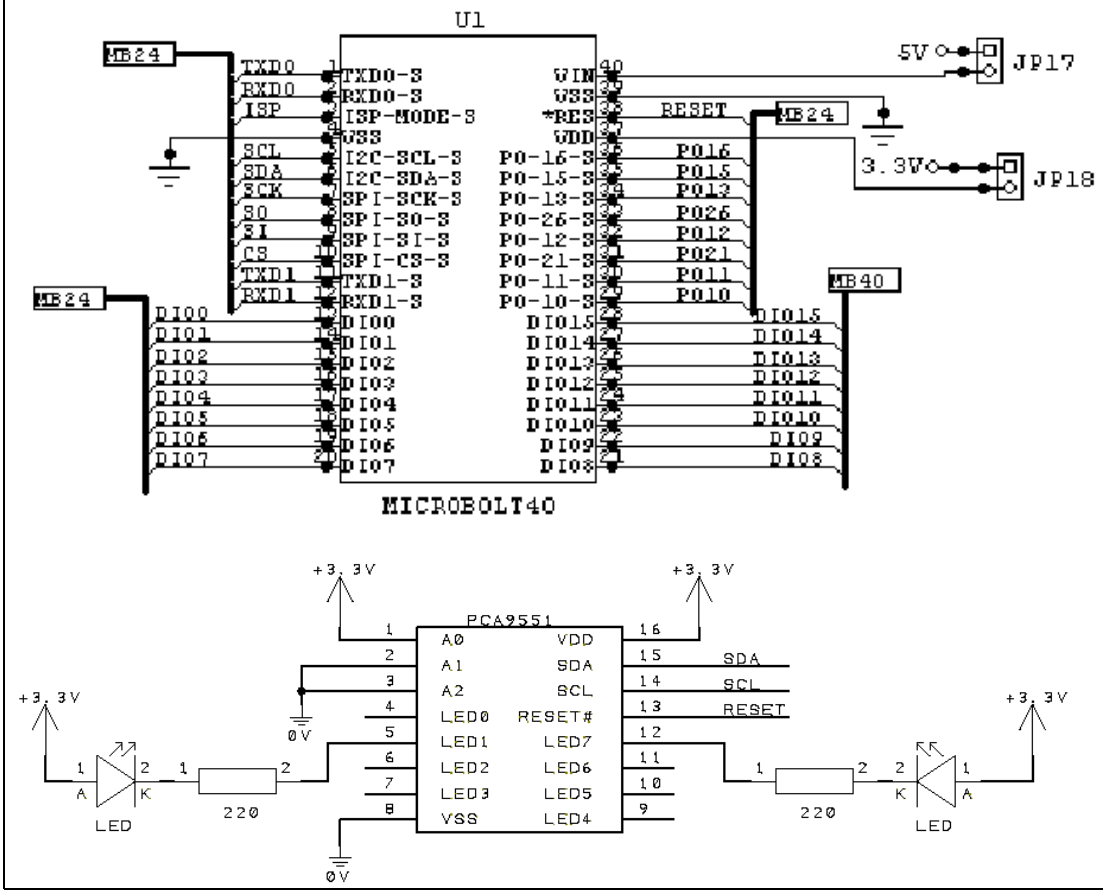
Micromint, Inc	**AN818**
	MicroBolt
The MicroBolt as an I²C Master	12/30/2005

Introduction:
This application notes demonstrates how to use the MicroBolt as an I²C Master on the I²C serial bus.

Background:
The Philips 2 wire I²C Serial bus is a very popular 2-wire network. Many devices are available that support connection to this serial bus. Since the MicroBolt is based off from a Philips LPC2106 controller, its I²C implementation is highly integrated. For this reason, the MicroBolt can live on the I²C bus with very little user written code since most of the I²C functionality is built into the LPC2106 hardware.

For more information on the Philips I²C Serial bus 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. The MicroBolt is setup as a master on the I²C bus and a Philips PCA9551 LED driver is setup as a slave. The PCA9551 slave address is set to 0x61. The PCA9551 was wired onto the MicroBolt development board prototyping area and drives two LEDs wired to the LED-1 and LED-7 outputs. The MicroBolt sets up the PCA9551 via writing to it over the I²C bus and monitors it via reads over the I²C bus. The MicroBolt reads the input register of the PCA9551 continuously and toggles its own LED to match the PCA9551 LED-1 and LED-7 outputs. This provides read and write examples for the MicroBolt as an I²C master.

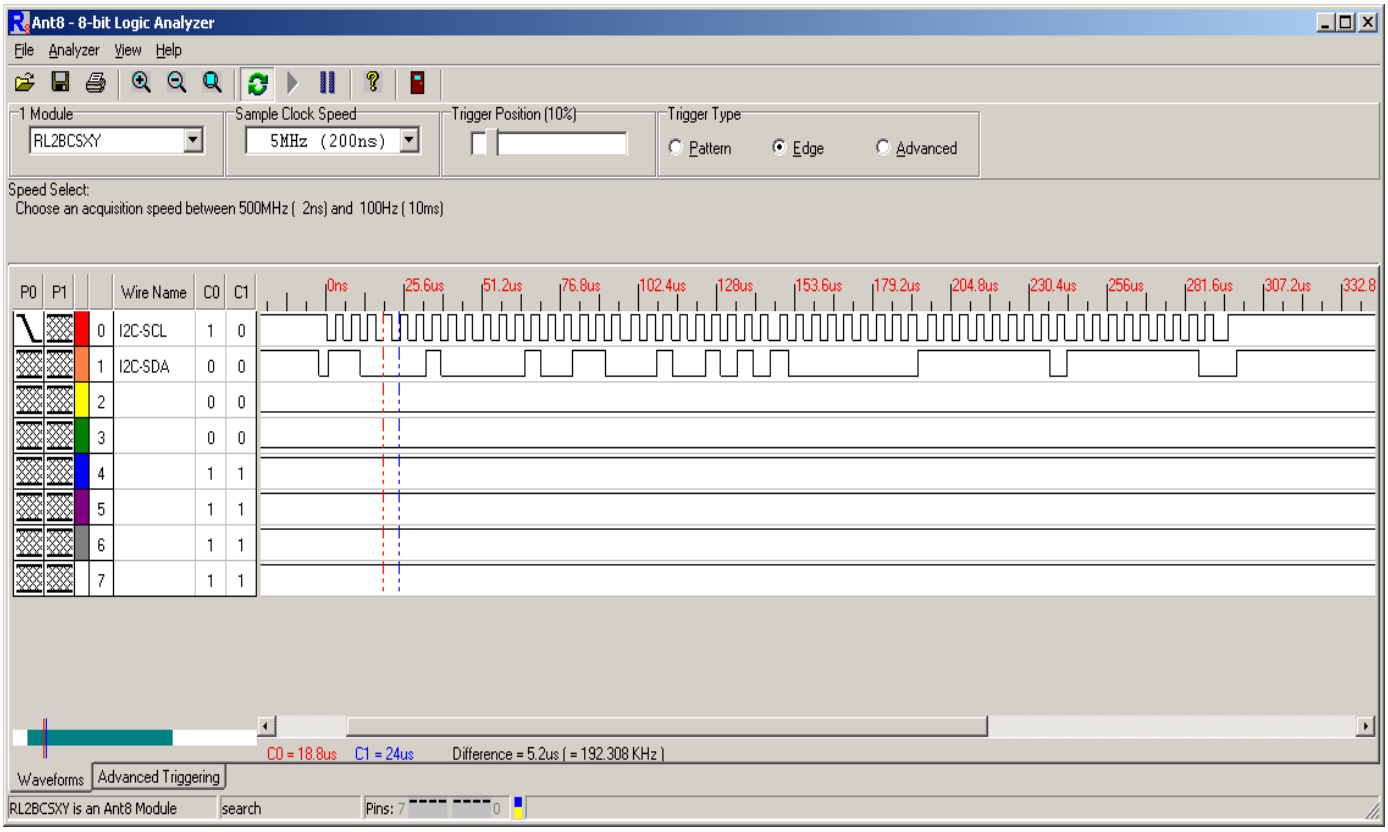
All I²C parsing occurs in the I²C interrupt handler. The example code contains an I²C buffer (array of 20 bytes) that can be used for transmitting and receiving bytes over the I²C bus. An index points to the buffer allowing the buffer to be incremented or decremented at will. Other variables for packet size and packet type are provided as an easy way to setup data for the I²C bus.

The I²C slave framework of this project allows the user to configure the I²C master as needed. The I²C buffer, index pointer, and valid states have been properly setup and example packets controlling the PCA9551 show a simple I²C master implementation.

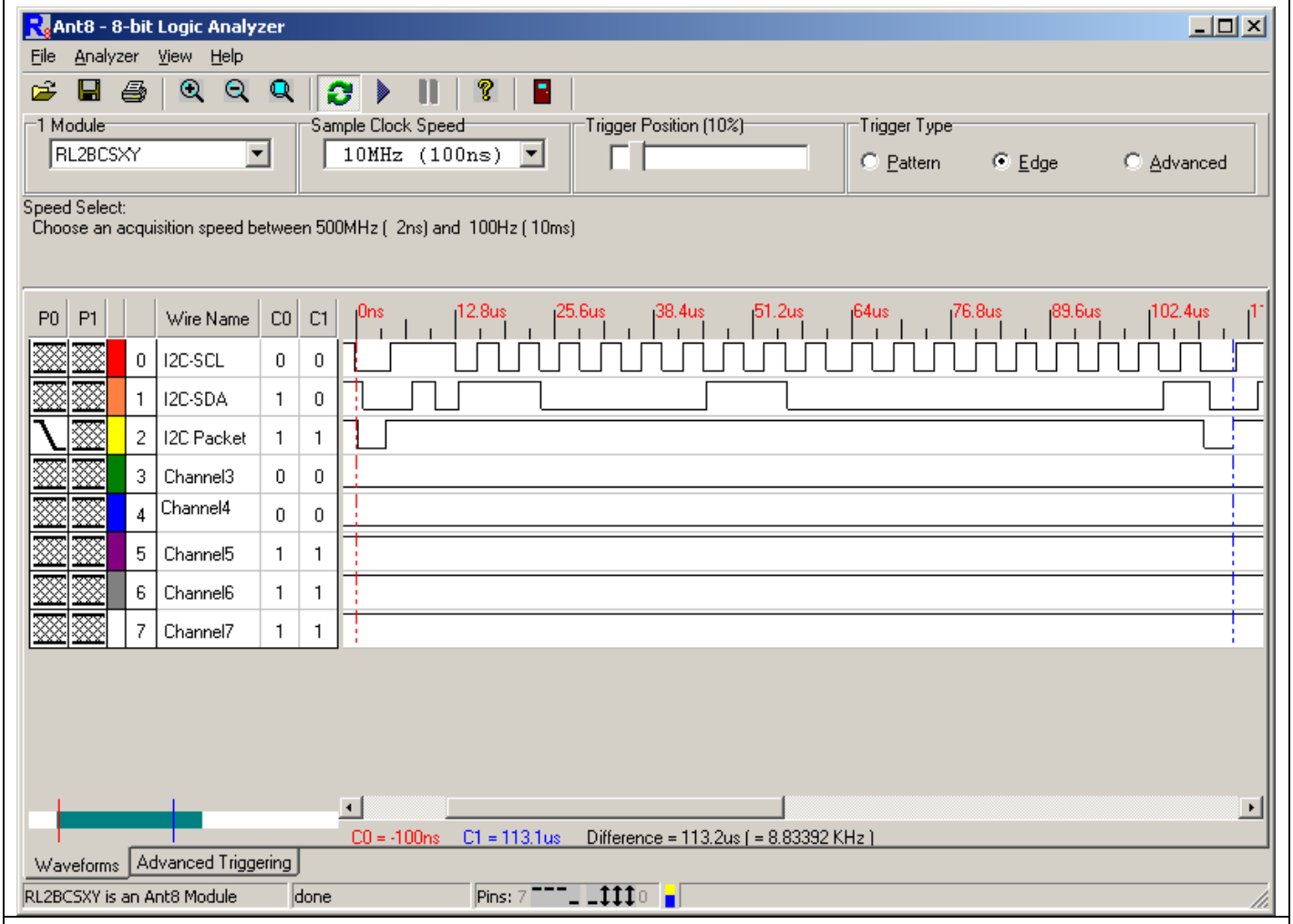
Note: Be sure to install jumpers JP37 and JP38 to enable I²C on the MicroBolt development board.

Waveforms:

The following waveform capture shows 6 bytes being written from the MicroBolt to the PCA9551 over the I²C bus. The first byte is the PCA9551 slave address of 0x61 and the next 5 bytes are associated data as found in the demo code.



The following waveform capture shows 2 bytes for a read from the PCA9551 by the MicroBolt over the I²C bus. The first byte is the PCA9551 slave address of 0x61 and the next byte is the input register status of the PCA9551.



Program Listing:

```
/*
|-----------------------------------------------------------------------------------------
|-----------------------------------------------------------------------------------------
| File Name                     : MicroBolt_I2C_Master.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_Master.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)
  {

  __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_Master.bcf (I2C)
  PCB_PINSEL1=0x55400000;               //                                   (Secondary JTAG pins)

  GPIO_IODIR |= MICROBOLT_LED;                  // Setup MicroBolt LED as output
  GPIO IODIR |= P0 9;                           // P0 9 for debug
```

```
  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 PCA9551 Slave device
|---------------------------------
*/                                 // Initialize I2C buffers with desired data for PCA9551 slave
                                   // The below configuration toggles the PCA9551 outputs 3 times a second
                                   // For this application, only LED-1 and LED-7 are wired up
  I2cSlaveAddress = I2C_SLAVE_ADDR_PCA9551;  // Address the PCA9551
  I2cPacketDataSize = 5;                    // How many bytes from the I2c buffer to send
  I2cBuffer[0] = 0x13;                      // Control register sets starting address and autoincrement to on
  I2cBuffer[1] = 0x10;                      // Prescaler 1 value
  I2cBuffer[2] = 0x80;                      // PWM-1 value
  I2cBuffer[3] = 0xFF;                      // LED 0 to 3 assigned to PWM-1
  I2cBuffer[4] = 0xFF;                      // LED 4 to 7 assigned to PWM-1
  I2cStart(I2C_WRITE);                      // Send out an I2C Start condition for a Write packet
  while(I2cPacketInProgress == TRUE);       // Wait here for I2C packet to complete

/*
|-----------------------------------------------------
| Send a new control byte to PCA9551 I2C Slave device
|-----------------------------------------------------
*/
  I2cSlaveAddress = I2C_SLAVE_ADDR_PCA9551;     // Address the PCA9551
  I2cPacketDataSize = 1;                        // How many bytes from the I2c buffer to send
  I2cBuffer[0] = 0x00;                          // Control register set to 0, turn off autoincrement
  I2cStart(I2C_WRITE);                          // Send out an I2C Start condition for a Write packet
  while(I2cPacketInProgress == TRUE);           // Wait here for previous I2C packet to complete

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

  while(1)                                      // Do this forever
    {
/*
|-------------------------------------------------
| Read PCA9551 I2C Slave device's input register
|-------------------------------------------------
*/
    I2cSlaveAddress = I2C_SLAVE_ADDR_PCA9551;   // Address the PCA9551
    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

    if (I2cBuffer[0] == 0x82)           // LED-1 and LED-7 read as ON from the PCA9551 via the I2C buffer?
        {
        GPIO_IOCLR = MICROBOLT_LED;     // Yes, turn off the MicroBolt LED to match LED-1 and LED-7
         }
   else
        {
        GPIO_IOSET = MICROBOLT_LED;      // No, turn on the MicroBolt LED to match LED-1 and LED-7
        }
    }
  }

/*
|-------------------------------------------------------------------------------------------
|     Function        :   I2c_ISR
|     Inputs          :   None
```

```
|     Outputs         :   None
|     Purpose         :   I2C interrupt and command processing
|     Author          :   Micromint, Inc.
|-------------------------------------------------------------------------------
*/

#pragma interrupt_handler I2c_ISR

void I2c_ISR(void)
{
  {

  See ImageCraft ICCARM demo project for the rest of the code

  }
```