

I²C on the eDemo board

Bob Alkire



I²C, SMBus or 2 Wire interface (TWI) are similar implementations of a synchronous half duplex multi-master interchip interface. The eDemoboard rev 5 SDK now has firmware which supports I²C master.

Microchip made a very nice demo board that I used for testing the eDemo I²C. This is the PICKit I²C serial demo board and it hosts a wide variety of chips with I²C interface. Digikey and Mouser have this board PKSERIAL-I²C1 for \$25. Data sheet can be downloaded from the Microchip website.

The board has:

- 24LC02B 2Kbit serial EEPROM
- MCP9801 Temperature Sensor
- MCP3221 12 bit A/D Converter
- TC1321 10 bit D/A Converter
- MCP23008 8 bit I/O Expander (connected to LEDs)

The test code, TWITest.java, exercises each of these devices.

The firmware is for the rev 5.0 SDK only (Red) and will not work on earlier versions. It can work with multiple masters but cannot be addressed as a slave device itself.

The TWI Java API is located in EDemoBoard.java, although most of the hard work is done in EDemoController.java.

Create or use the existing eDemo instance with:

```
EDemoBoard edemo = EDemoBoard.getInstance();
```

Send data with:

```
edemo.sendTWI(SLA, snd, sndlen);
```

Receive (only) data with:

```
edemo.receiveTWI(SLA, rcv, rcvlen);
```

Or transmit and then receive with:

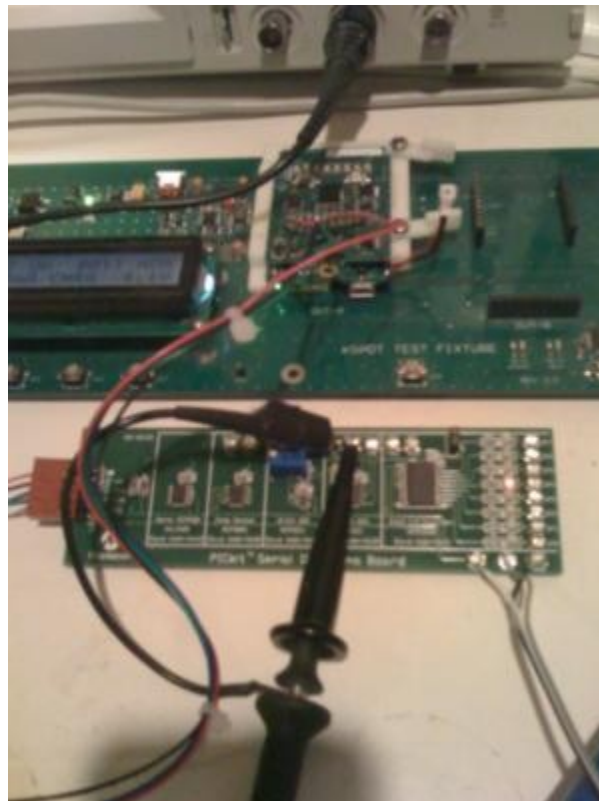
```
edemo.transceiveTWI(SLA, snd, sndlen, rcv, rcvlen);
```

Hooking up the PICKit

The next step is to hook up the board using the following connections:

EDEMO	PICKIT
+3V	+V
GND	GND
D2	SDA
D3	SCK

The eDemo board will provide +3V to the board. You will also need to add two 4.7K surface mount resistors (0805 works) to pull up the SDA and SCK lines. These are located near the connector on the PICKIT.



You should edit the TWITest.java file and make sure if this is the first time this has been run, find the line:

```
private final static boolean WRITE_ME = false;
```

and change it to true. You may want to change back to false afterwards to check the non-volatile memory or if your concerned of to many writes to EEPROM.

Once the board is connected, you can now deploy the Java application, by:

```
ant deploy run
```

After compiling you should see output:

```
-run-spotclient-once:
[java] SPOT Client starting...
[java] [waiting for reset]

[java] Sun SPOT bootloader (blue-080626red-xxxxxx)
[java] SPOT serial number = 0014.4F01.0000.0FBB

[java] Squawk VM Starting (blue-080626red-xxxxxx) ...
[java] Start EDemoTWI Test
[java] Reading "Hello World" from EEPROM
[java] Temperature: 27.5C
[java] ADC: 3756
[java] Temperature: 26.5C
[java] ADC: 3757
```

The DAC output can be viewed on an oscilloscope as a slow sawtooth with a one second hold.



The ADC will continually output values and can be varied by adjusting the trimmer potentiometer on the PICKit. The LEDs will scan from GP0 to GP7 slowly.

eDemo TWI theory of operation

The eDemo will use the D2 and D3 pins for SDA and SCK of the TWI interface. These are the Port C bit 4 and Port C bit 5 pins. When the TWI is active, the SCK pin is an open drain output and the SDA pin is bidirectional with open drain output. They require pull up resistors to +3V and the resistors are not part of the eDemo board. The signal levels are also 3V logic and will require a voltage translation IC for running at 5V; otherwise the Atmega can be damaged.

Most of the work is done inside of the TWI interrupt routine. This routine is similar to one in the Atmel application note AVR315. The routine stores and retrieves data through the `twi_master` structure:

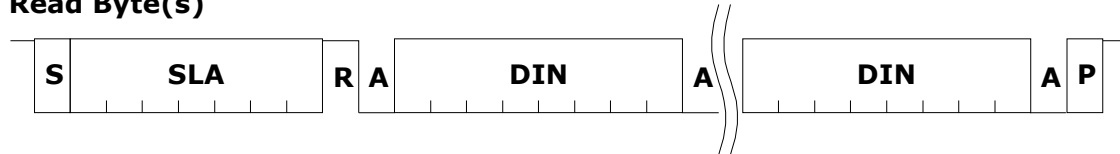
```
struct {
    uint8_t sla; // Slave address + Read/Write bit
    uint8_t lenA; // Length in bytes of first packet
    uint8_t lenB; // Length in bytes of restart packet (msb is restart
read/write)
    uint8_t buffer[TWI_BUFFER_SIZE];
    uint8_t control; // Control byte
    uint8_t state; // Error state
} twi_master;
```

This structure is accessed as a byte array from Java. The `sla` is the 7 bit slave address of the slave device your are talking to. The LSB of the SLA is Read/Write bit for the packet. The length of the packet in bytes is in `lenA`. A TWI packet can be restart and reuse the SLA with another Read/Write bit. The number of bytes of data sent or received after the restart is in `lenB` with the Read/Write as the MSB of `lenB`. Data is read from and stored to the buffer.

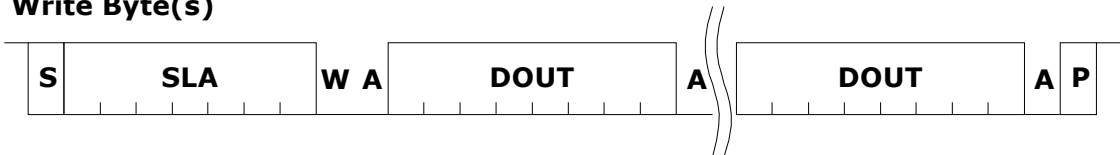
The MSB of the control byte is the interrupt mask, that is an interrupt can be generated to the ARM9 when TWI completes. The state byte indicates whether or not the transaction was successful. Once the buffer is set up, the TWCR register is written and the TWI transaction starts. It is busy as long as the TWI interrupt is set.

Protocol for TWI is:

Read Byte(s)



Write Byte(s)



Restart Write then Read Byte(s)



S = Start bit

SLA = Slave address

R = Read bit (1)

W = Write bit (0)

A = Acknowledge (1=No Acknowledgement)

DIN = Data input from the slave to the master

DOUT = Data output to the slave from the master

R = Restart bit

P = Stop bit

For more detailed information about I2C and TWI, go here:

http://www.nxp.com/acrobat_download/literature/9398/39340011.pdf

http://www.atmel.com/dyn/resources/prod_documents/doc2564.pdf

About Sun Labs

Established in 1990, Sun Microsystems Laboratories is the applied research and advanced development arm of Sun Microsystems, Inc., with locations in California and Massachusetts. Sun Labs is one of the ways Sun invests in the future, and is responsible for many of the technology advancements that have made Sun a technology powerhouse—including asynchronous and high-speed circuits, optical interconnects, 3rd-generation Web technologies, sensors, network scaling and Java technologies. Although many companies have R&D groups, Sun Labs can claim one of the highest rates of technology transfer in the industry.

Sun Microsystems, Inc. 4150 Network Circle, Santa Clara, CA 95054 USA Phone 1-650-960-1300 or 1-800-555-9SUN Web sun.com

