

# Microtronix Avalon I<sup>2</sup>C

---

User Manual



9-1510 Woodcock St.  
London, ON  
Canada N5H 5S1  
[www.microtronix.com](http://www.microtronix.com)



This user guide provides basic information about using the Microtronix Avalon I<sup>2</sup>C IP. The following table shows the document revision history.

Date	Rev	Description
March 2006	1.0	Initial release
July 2008	1.1	Added slave address match & minor edits

## How to Contact Microtronix

### *E-mail*

Sales Information: [sales@microtronix.com](mailto:sales@microtronix.com)

Support Information: [support@microtronix.com](mailto:support@microtronix.com)

### *Website*

General Website: <http://www.microtronix.com>

### *Phone Numbers*

General: 519-690-0091

Fax: 519-690-0092

## Typographic Conventions

Visual Clue	Meaning
Path/Filename	A path or filename
[SOPC Builder]\$ <cmd>	A command that should be run from within the Cygwin Environment.
Code	Sample code.
←	Indicates that there is no break between the current line and the next line.

## Table of Contents

---

How to Contact Microtronix .....	2
E-mail .....	2
Website .....	2
Phone Numbers .....	2
Typographic Conventions .....	2
Introduction .....	4
SOPC Builder Setup .....	4
Register Map .....	4
Control Register .....	5
Status Register.....	5
Address Register .....	6
Transmit Buffer.....	6
Receive Buffer.....	7
Software.....	7
Simulation .....	9
Verification .....	9

## Introduction

The I<sup>2</sup>C-bus is a simple two wire bi-directional interface developed for inter-IC communication. Many semiconductor vendors offer a wide range of I<sup>2</sup>C-devices, like EEPROM memories, I/O-ports, temperature sensors, analog / digital converters, etc.

The I<sup>2</sup>C-core is an Avalon slave peripheral that allows Avalon master peripherals, like the Altera Nios II processor, to interface with I<sup>2</sup>C-peripherals. It has the following features:

- (Single) I<sup>2</sup>C-Master Transmit / Receive Mode.
- I<sup>2</sup>C-Slave Transmit / Receive Mode.
- Fixed 8-bits data transfers.
- 7-bits addressing format.
- Own address and general call address detection.
- Single byte transmit and receive buffer.
- Input filter.
- Three transmission speeds
  - Standard Mode (100 KHz)
  - Fast Mode (400 KHz)
  - High-Speed Mode (1 MHz)
- Meets the Philips I<sup>2</sup>C-bus specification version 2.1

The I<sup>2</sup>C core is SOPC Builder ready and integrates easily into any SOPC Builder generated system.

## SOPC Builder Setup

Before using the I<sup>2</sup>C component in SOPC Builder, its directory must be added to the SOPC Builder search path. In SOPC Builder, select Options from the Tools menu. Select IP Search Path under Category. Click Add and browse to the ip\nios2\_ip directory of the Quartus II installation (default for 8.0 is C:\altera\80\ip\sopc\_builder\_ip). Click Open then Finish. SOPC Builder must be restarted for this change to take effect.

## Register Map

The I<sup>2</sup>C core has four 8-bits registers shown in Table 1. Note: SOPC Component uses Avalon native alignment, registers are 32-bit aligned in a Nios II system.

**Table 1: Register Map**

Offset	Name	RW
0	Control Register	RW
1	Status Register	R
2	Address Register	RW
3	Transmit Buffer	W
3	Receive Buffer	R

**Control Register**

The control register controls the operation of the I<sup>2</sup>C core. The control register can be read any time without changing the value of any bits. The control register bits are shown in Table 2.

**Table 2: Control Register**

Offset	Name	Description
0	I2C ENABLE	Setting this bit enables the I <sup>2</sup> C-interface.
1	I2C MODE	The two bits sets the I <sup>2</sup> C mode
2		00 – Slave Mode 01 – Master Standard Mode (100 KHz) 10 – Master Fast Mode (400 KHz) 11 – Master High-speed Mode (1 MHz)
3	I2C START	When is bit is set in master mode a (repeated) start condition is created. After the I <sup>2</sup> C core has applied the start condition this bit is automatic cleared and the I <sup>2</sup> C address is transmitted. In slave mode this bit has no influence.
4	I2C STOP	Setting this bit in master mode generates a stop condition. This bit is automatically cleared after the stop condition was applied. An interrupt is generated after the stop condition was successful executed. In slave mode this bit has no influence.
5	I2C RW	This bit selects an I <sup>2</sup> C read or write transfer. 0 - Write 1 - Read
6	I2C ACK	This bit is transmitted after the I <sup>2</sup> C data byte. 0 – ACK 1 – NACK
7	IRQ ENABLE	Setting this bit enables the Avalon interrupt

**Status Register**

The status register shows the status of the I<sup>2</sup>C core. Reading the status register doesn't change the value of the bits. Writing any value to the status register clears the interrupt bit.

**Table 3: Status Register**

Offset	Name	Description
0	READY	
1	I2C NACK	In master mode this bit is set when no acknowledge was received after an I <sup>2</sup> C address or write data transfer. The I <sup>2</sup> C core automatically generates a stop condition when no acknowledge was detected. In slave mode this bit is always cleared.
2	TX EMPTY	This bit indicates the status of the transmit buffer.
3	RX FULL	This bit indicated the status of the receive buffer.
4	I2C WRITE TRANSFER	This bit is set when a write transfer is performed.
5	I2C READ TRANSFER	This bit is set when a read transfer is performed.
6	I2C SLAVE ADDRESS MATCH	This bit is used in slave mode to indicate when the address on the I <sup>2</sup> C bus matches the content of the address register. This bit is cleared by writing to the status register. In master mode this bit is always cleared.
7	IRQ	The interrupt bit is asserted when; <ul style="list-style-type: none"> <li>- The transmit buffer is empty.</li> <li>- The receive buffer is full.</li> <li>- A write access was not acknowledged by the I<sup>2</sup>C slave in master mode.</li> </ul> After the stop condition was generated in master mode.

### **Address Register**

The address register holds the seven-bit I<sup>2</sup>C address.

In master mode the I<sup>2</sup>C address is transmitted after the start bit in the control register was set.

In slave mode the address register is compared with the address byte sent by the I<sup>2</sup>C master. If the addresses are equal, an acknowledgement is automatically sent by the I<sup>2</sup>C core to the I<sup>2</sup>C master. The slave address match bit is also set.

### **Transmit Buffer**

An Avalon master peripheral writes data to be transmitted into the transmit buffer. Writing to the transmit buffer automatically clears the transmit buffer empty bit and the interrupt.

In master mode the data in transmit buffer is directly send after the address byte. If the transmit buffer is empty, the transmit buffer empty bit in the status register is set and the interrupt is

asserted. The I<sup>2</sup>C core stalls the I<sup>2</sup>C transfer until new data is written in the transmit buffer or a repeated start or stop condition is selected.

In slave mode after the I<sup>2</sup>C slave address was received, the data in the transmit buffer is transferred to the I<sup>2</sup>C master. If no (new) transmit data is available, the transmit buffer empty bit is set and the interrupt is activated. The I<sup>2</sup>C core tells the I<sup>2</sup>C master to wait by forcing the I<sup>2</sup>C clock line low until new data had been written in the transmit buffer. After the data transfer the I<sup>2</sup>C -master sends an acknowledgement. An ACK indicates that the I<sup>2</sup>C -master wants to read more data from the I<sup>2</sup>C slave. A NACK tells the I<sup>2</sup>C slave that this was the last byte of the transfer. Now the I<sup>2</sup>C slave releases the I<sup>2</sup>C data line and the I<sup>2</sup>C master can generate start or stop condition.

### **Receive Buffer**

Received data is loaded by the I<sup>2</sup>C core into the receive buffer and the receive buffer full bit is set. When the receive buffer is read by the Avalon master peripheral, the interrupt bit and receive buffer full bit are cleared.

When the I<sup>2</sup>C core is in master mode and after it receives a data byte from the I<sup>2</sup>C slave, the acknowledge bit, located in the control register, is send. When the receive buffer is read and no start or stop condition is selected, then the I<sup>2</sup>C core executes another read transfer to the same I<sup>2</sup>C slave.

In slave mode the interrupt is asserted when the I<sup>2</sup>C core received new data from the I<sup>2</sup>C master and has loaded it in the receive buffer. The I<sup>2</sup>C core automatically acknowledges the transfer to the I<sup>2</sup>C master. If the receive buffer is full, indicated by the receive buffer full bit in the status register, the acknowledge bit is delayed by the I<sup>2</sup>C core. The I<sup>2</sup>C clock line to pulled low until the receive buffer was read by the Avalon master peripheral. Now the I<sup>2</sup>C core releases the I<sup>2</sup>C clock line and the I<sup>2</sup>C master continues with the read transfer.

## **Software**

The file `mtx_avalon_i2c_regs.h` declares functions for accessing the I<sup>2</sup>C-peripheral. These functions are listed in Table 4.

**Table 4: Software Routines**

<code>IORD_MTX_AVALON_I2C_CONTROL(base)</code>	Read Control Register
<code>IOWR_MTX_AVALON_I2C_CONTROL(base, data)</code>	Write Control Register
<code>IORD_MTX_AVALON_I2C_STATUS(base)</code>	Read Status Register
<code>IOWR_MTX_AVALON_I2C_STATUS(base, data)</code>	Write Status Register
<code>IORD_MTX_AVALON_I2C_ADDR(base)</code>	Read Address Register

IOWR_MTX_AVALON_I2C_ADDR(base, data)	Write Address Register
IOWR_MTX_AVALON_I2C_DATA(base, data)	Write Transmit Buffer
IORD_MTX_AVALON_I2C_DATA(base)	Read Transmit Buffer

The following code demonstrates writing 8 bytes to an I<sup>2</sup>C-EEPROM.

```
int eeeprom_write(unsigned char slave_addr, unsigned
char mem_addr)
{
    int m;

    // EEPROM Slave Address
    IOWR_MTX_AVALON_I2C_ADDR(I2C_BASE, slave_addr);

    // Start Condition
    IOWR_MTX_AVALON_I2C_CONTROL(I2C_BASE,
    MTX_AVALON_I2C_ENABLE |
    MTX_AVALON_I2C_MASTER_FAST |
    MTX_AVALON_I2C_START);

    // Wait for event
    while( !(IORD_MTX_AVALON_I2C_STATUS(I2C_BASE) &
    MTX_AVALON_I2C_IRQ) );

    // Clear interrupt
    IOWR_MTX_AVALON_I2C_STATUS(I2C_BASE, 0);

    // Check if EEPROM responded
    if ( IORD_MTX_AVALON_I2C_STATUS(I2C_BASE) &
    MTX_AVALON_I2C_NACK)
    {
        printf("EEPROM not found.\n");
        return 0;
    }

    // Write Address
    IOWR_MTX_AVALON_I2C_DATA(I2C_BASE, mem_addr);

    // Wait for event
    while( !(IORD_MTX_AVALON_I2C_STATUS(I2C_BASE) &
    MTX_AVALON_I2C_IRQ) );

    // Write 8 bytes to EEPROM
    for (m = 1 ; m < 9; m++)
    {
        // Fill Transmit Buffer
        IOWR_MTX_AVALON_I2C_DATA(I2C_BASE, m);

        // Wait for transmit buffer empty
        while( !(IORD_MTX_AVALON_I2C_STATUS(I2C_BASE) &
    MTX_AVALON_I2C_IRQ) );
    }

    // Stop Condition
    IOWR_MTX_AVALON_I2C_CONTROL(I2C_BASE,
    IORD_MTX_AVALON_I2C_CONTROL(I2C_BASE) |
```

```
    MTX_AVALON_I2C_STOP);

// Wait for event
while( !(IORD_MTX_AVALON_I2C_STATUS(I2C_BASE) &
    MTX_AVALON_I2C_IRQ) );

// Clear interrupt
IOWR_MTX_AVALON_I2C_STATUS(I2C_BASE, 0);

// Disable
IOWR_MTX_AVALON_I2C_CONTROL(I2C_BASE, 0);

return 1;
}
```

## Simulation

A precompiled simulation library is provided for performing simulations using ModelSim. The library is located in the <install\_dir>/simulation directory. Perform the following steps to simulate your design with the I<sup>2</sup>C controller.

1. Launch ModelSim
2. Map the I<sup>2</sup>C library. At the ModelSim prompt type;  
vmap mtx\_avalon\_i2c\_top  
<install\_dir>/simulation/mtx\_avalon\_i2c\_top

If you use a newer version of ModelSim, you must refresh the precompiled library. At the Modelsim prompt type;  
vcom -refresh -work mtx\_avalon\_i2c\_top

3. Compile the I<sup>2</sup>C top level. Eg. If the I<sup>2</sup>C controller was named i2c in the SOPC Builder, then type;  
vcom -93 i2c.vhd
4. Compile all of the design files
5. Start the ModelSim simulation by typing;  
vsim -t ps -L mtx\_avalon\_i2c\_top <top\_level>

## Verification

The I<sup>2</sup>C core has been verified in hardware on the Sendero Evaluation Board Revision B.