



MachXO2 Hardened I²C Master/Slave Demo

User's Guide

Introduction

Every MachXO2™ device contains two hardened I²C IP cores designated as the “primary” and “secondary” I²C IP cores. Either of the two cores can be operated as an I²C master or as an I²C slave. The difference between the two cores is that the primary core has pre-assigned I/O pins while the ports of the secondary core can be assigned by designers to any general purpose I/O. In addition, the primary core also has access to the configuration logic of the MachXO2 devices.

When an I²C core is a master it can control other devices on the I²C bus through the physical interface. When a core is the slave, the device can provide peripheral expansion to an I²C master. For more information on the hardened I²C core please refer to the [MachXO2 Family Data Sheet](#).

This design demonstrates the usage of the hardened “primary” I²C core as both master and slave on two MachXO2 Pico Evaluation Boards. One Pico board is configured as I²C master and the other as I²C slave. For user interaction with the demo, the I²C master design includes a UART and capsense button controller logics. The I²C master board communicates with the I²C slave board based on the user inputs through the UART terminal or capsense buttons.

Demo design hardware requirements:

- Two MachXO2 Pico Evaluation Boards
- Windows PC for implementing the demo project and downloading the bitstream
- USB download cable
- Connecting wires to make connection between the I²C signals of the two boards

Demo design software requirements:

- Lattice Diamond® design software version 1.4 (or later)
- HyperTerminal or any other terminal emulator (communications) program to interact with the I²C master demo board

Demonstration Design

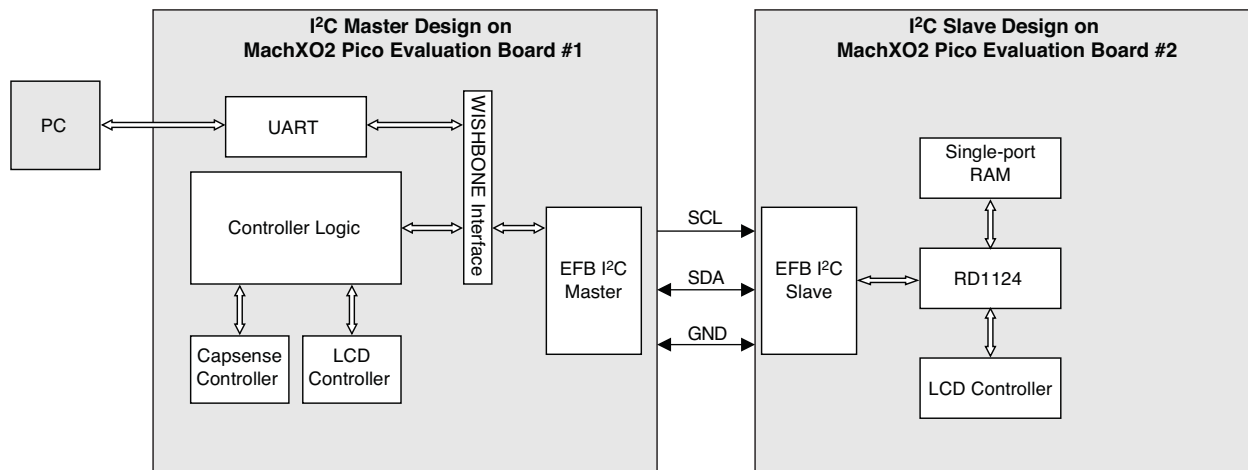
The MachXO2 Hardened I²C Master/Slave Demo consists of two designs:

- I²C master demo design
- I²C slave demo design

Users can interact with the I²C master demo design through the terminal of a PC or through the capacitive touch sense buttons. The I²C master demo design responds to the user inputs by writing data to or reading data from the I²C slave demo design using the hardened I²C primary IP core. The I²C master demo design also has a LCD controller logic which displays the I²C transaction data on the LCD.

The I²C slave demo design incorporates RD1124, I²C Slave Peripheral Using Embedded Function Block. One of the GPIO output ports of RD1124 is connected to LCD controller logic and the memory port of RD1124 is connected to a single-port RAM, implemented in Embedded Block RAM (EBR). Figure 1 shows the block diagram of this demo design.

Figure 1. MachXO2 Hardened I²C Master/Slave Demo Block Diagram



MachXO2 Hardened I²C Master Design

Controller Logic

The controller logic controls the different modules of the I²C master design. Users can interact with the controller logic through the UART terminal or capsense buttons. The controller logic generates the necessary WISHBONE signals to interact with the hardened I²C master module based on the input from the user.

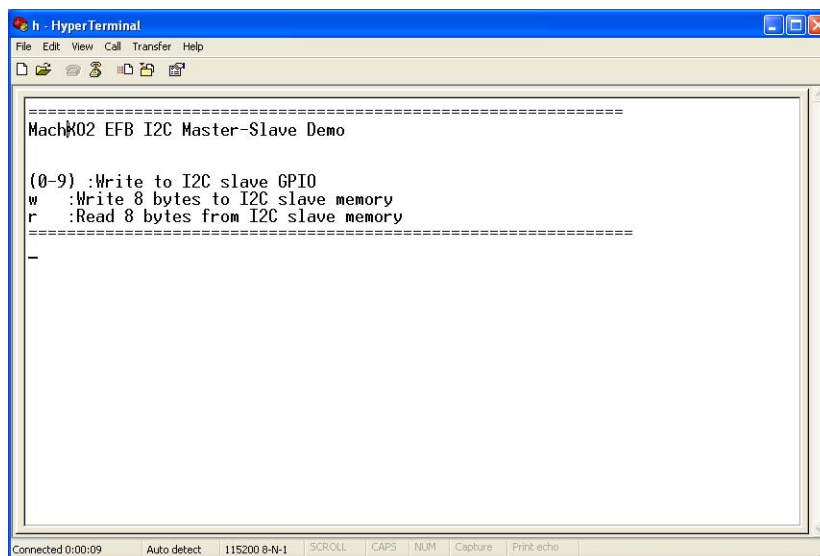
EFB I²C Master

The MachXO2 Embedded Function Block (EFB) hardened IP is configured to use the primary I²C core as I²C master with 7-bit addressing mode and operates at a data transfer speed of 50 KHz. The I²C core interacts with the controller logic through the WISHBONE interface.

UART

Users can interact with the controller logic through the UART module. Figure 2 shows the menu that is displayed on the PC terminal window once reset is de-asserted.

Figure 2. PC Terminal User Menu



```

=====
MachXO2 EFB I2C Master-Slave Demo

(0-9) :Write to I2C slave GPIO
w      :Write 8 bytes to I2C slave memory
r      :Read 8 bytes from I2C slave memory
=====
  
```

The options available are:

- **(0-9)**: Displays the key entered on the LCD screen of both I²C master and slave boards.
- **w**: Writes eight bytes of data to the I²C slave memory, starting at address 0. The data written in this case is fixed to 1-8 and the address will be automatically incremented by the I²C slave design. The data being transmitted will be displayed on the LCD screen of both the I²C master and slave boards.
- **r**: Reads the previously written eight bytes of data (1-8) from the I²C slave memory starting from location 0 and displays on the I²C master board's LCD and on the PC terminal. There will be no display on the I²C slave board's LCD screen. If no data is written prior to issuing this command, then a read will return eight bytes of 0s.

Capsense buttons

The user can interact to the control logic through the on-board capsense buttons on the I²C master board too. There are four capsense buttons on-board and their functionality for this demo is:

- **Button 1**: Displays '1111' on the LCD screen of both the I²C master and slave boards.
- **Button 2**: Displays '2222' on the LCD screen of both the I²C master and slave boards.
- **Button 3**: Writes eight bytes of data to the I²C slave memory, starting at address 0. The data written in this case is fixed to 1-8 and the address will be automatically incremented by the I²C slave design. The data being transmitted will be displayed on the LCD screen of both the I²C master and slave boards.
- **Button 4**: Reads the previously written eight bytes of data (1-8) from the I²C slave memory starting from location 0 and displays on the I²C master board's LCD and on the PC terminal. There will be no display on the I²C slave board's LCD screen. If no data is written prior to issuing this command, then a read will return eight bytes of 0s.

MachXO2 Hardened I²C Slave Design

EFB I²C Slave

The MachXO2 EFB hardened IP is configured to use the primary I²C core as I²C slave with 7-bit addressing mode. The I²C core interacts with the controller logic through the WISHBONE interface. In this design, the I²C slave address is set as '0001001 B'.

RD1124

This design provides eight bytes of I/O port and memory interface, which is controlled through the I²C slave interface.

There are four single-byte input ports and four single-byte output ports. This design interfaces with an embedded memory block. The memory read and write operations are controlled by the I²C commands. The operation of the design will be activated only when an enable command is received by the I²C slave. One of the GPIO output ports is connected to the LCD controller logic.

Based on the command received from the I²C master, data from the I²C master is either displayed on the LCD, written to the memory or read back from the memory. No RD1124 interrupts are used in this demo.

RD1124 Commands

Table 1 shows the different commands used in this demo by the I²C master design to order to access the RD1124 I²C slave. For more information on the command structures and sequencing, please refer to the RD1124 documentation.

Table 1. RD1124 Commands

Operation	Command	Operands	Data	Number of Bytes in Command
Enable	0x06	—	—	1
Disable	0x04	—	—	
Write GPO	0x01	Port # (0 to 3)	1 byte write data	3
Write Memory	0x02	1 byte address	8 bytes write data	10
Read Memory	0x0B	1 byte address	8 bytes read data	

Port Description and Pin Assignments

I²C Master Design

Table 2. I²C Master Design Pinouts

Port	I/O	Description	MachXO2 Pin
rst_n	Input	Active low reset signal	N3
SCL	Input	I ² C serial clock	C8
SDA	Input	I ² C serial data	B8
uartSIN	Input	UART serial input	E1
uartSOUT	Output	UART serial output	E2
LCD_COM0	Output	LCD com signal	B14
LCD_COM1	Output	LCD com signal	C13
LCD_COM2	Output	LCD com signal	C14
LCD_COM3	Output	LCD com signal	D12
LCD_5	Output	LCD segment signal	J12
LCD_6	Output	LCD segment signal	J14
LCD_7	Output	LCD segment signal	J13
LCD_8	Output	LCD segment signal	K12
LCD_9	Output	LCD segment signal	K13
LCD_10	Output	LCD segment signal	K14
LCD_11	Output	LCD segment signal	L14
LCD_12	Output	LCD segment signal	M13
cap_btn1	Input	Capsense button1	M10
cap_btn2	Input	Capsense button2	P11
cap_btn3	Input	Capsense button3	M11
cap_btn4	Input	Capsense button4	P12
enable1	Output	I ² C/SPI power enable signal	P2
enable2	Output	I ² C/SPI power enable signal	N2

I²C Slave Design

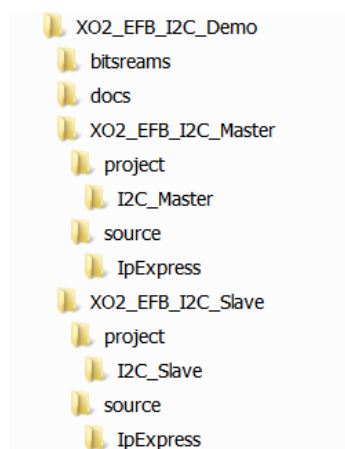
Table 3. I²C Slave Design Pinout

Port	I/O	Description	MachXO2 Pin
rst_n	Input	Active low reset signal	N3
SCL	Input	I ² C serial clock	C8
SDA	Input	I ² C serial data	B8
LCD_COM0	Output	LCD com signal	B14
LCD_COM1	Output	LCD com signal	C13
LCD_COM2	Output	LCD com signal	C14
LCD_COM3	Output	LCD com signal	D12
LCD_5	Output	LCD segment signal	J12
LCD_6	Output	LCD segment signal	J14
LCD_7	Output	LCD segment signal	J13
LCD_8	Output	LCD segment signal	K12
LCD_9	Output	LCD segment signal	K13
LCD_10	Output	LCD segment signal	K14
LCD_11	Output	LCD segment signal	L14
LCD_12	Output	LCD segment signal	M13
enable1	Output	I ² C/SPI power enable signal	P2
enable2	Output	I ² C/SPI power enable signal	N2

Demo Directory Structure

The directory structure of the MachXO2 hardened I²C Master/Slave demo is shown in Figure 3. The demo package includes four top-level folders: bitstreams, docs, XO2_EFB_I2C_Master and XO2_EFB_I2C_Slave, described below.

Figure 3. MachXO2 Hardened I²C Master/Slave Demo Directory Structure



Bitstreams Folder

This folder includes the bitstreams for the I²C master and slave designs.

Docs Folder

This folder includes the MachXO2 Hardened I²C Master/Slave Demo User's Guide and the read_me file. The read_me file contains a list of files for the demo and the steps to regenerate the bitstream.

XO2_EFB_I2C_Master Folder

This folder includes the following subfolders:

- **Project** – This folder includes the implementation project files for the Diamond design software and other necessary files including the place and route (PAR) preference and the post-route trace preference files for the I²C master design.
- **Source** – This folder includes all RTL source files used the I²C master design. This folder also includes the following subfolder:
 - **IPexpress** – This folder contains the IPexpress™ generated files for the IP cores, EFB and other memory modules in the design. You can use the IPexpress tool in the Diamond software to modify the IP cores and regenerate these files.

XO2_EFB_I2C_Slave Folder

This folder includes the following subfolders:

- **Project** – This folder includes the implementation project files for the Diamond design software and other necessary files including the place and route (PAR) preference and the post-route trace preference files for the I²C slave design.
- **Source** – This folder includes all RTL source files used for the I²C slave design. This folder also includes the following subfolder:
 - **IPexpress** – This folder contains the IPexpress generated files for the IP cores, EFB and other memory modules in the design. You can use the IPexpress tool in the Diamond software to modify the IP cores and regenerate these files.

Setting up the Boards

Drivers and Firmware

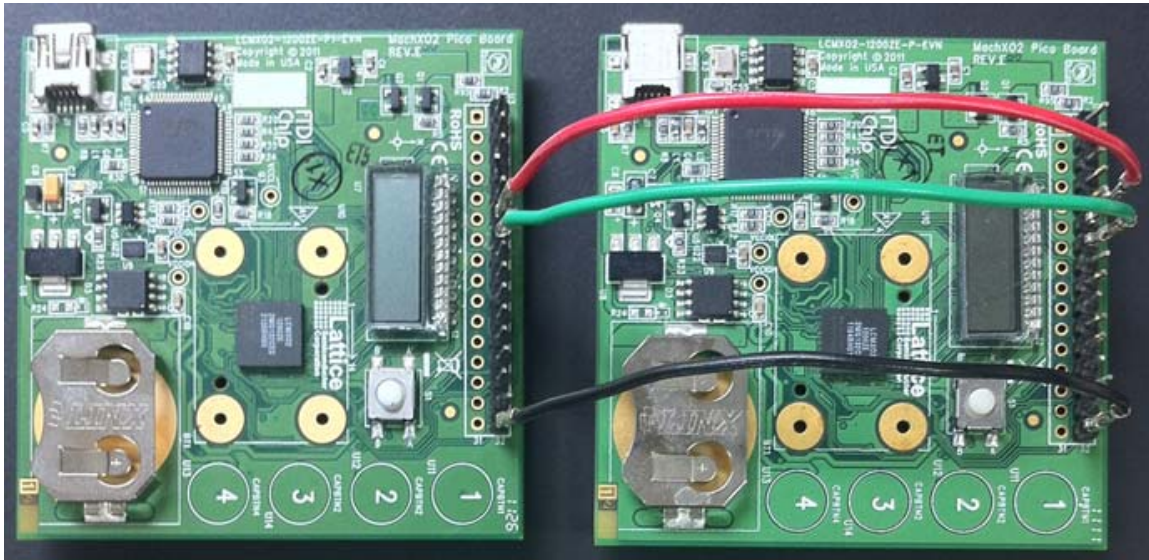
Before you begin, you will need to obtain the necessary hardware drivers for Windows from the Lattice website.

1. Browse to www.latticesemi.com/alpha-mxo2-pico-kit and locate the hardware device drivers for the USB interface.
2. Download the ZIP file to your system and unzip it to a location on your PC.

Connecting the Two MachXO2 Pico Evaluation Boards

1. The following pins of the header U3 on the two MachXO2 Pico Evaluation Boards must be connected to each other as shown in Figure 4.
 - SCL – Connect pin C8 of the two Pico boards to each other
 - SDA – Connect pin B8 of the two Pico boards to each other
 - Ground – Connect pin GND of the two Pico boards to each other

Figure 4. Evaluation Board Connections



2. Connect the evaluation boards to your PC using a USB cable. The USB connector on the board includes reference designator J1. Once the connection is made, a blue LED with reference designator D2 will illuminate.
3. If you are prompted “Windows may connect to Windows Update”, select **No, not this time** from available options and click **Next** to proceed with the installation.
4. Choose the **Install from specific location (Advanced)** option and click **Next**.
5. Select **Search for the best driver in these locations** and click the **Browse** button to browse to the Windows driver folder created earlier. Select the **CDM 2.04.06 WHQL Certified** folder and click **OK**.
6. **Click Next**. A screen will display as Windows copies the required driver files. Windows will display a message indicating that the installation was successful.

Programming the Boards

Using ispVM™ System software, users can scan and perform JTAG operations, including programming the MachXO2 device. Program the two boards, one at a time, using the bitstreams in the following locations:

1. Program one of the MachXO2 Pico Evaluation Boards with the jed file in the following location:
`.\XO2_EFB_I2C_Demo\bitsreams\I2C_Slave_I2C_Slave.jed`
2. Program the second MachXO2 Pico Evaluation Board with the jed file in the following location:
`.\XO2_EFB_I2C_Demo\bitsreams\I2C_Master_I2C_Master.jed`

Once both the boards are programmed, power-up the I²C slave board first and then the I²C master board.

Setting up Windows HyperTerminal

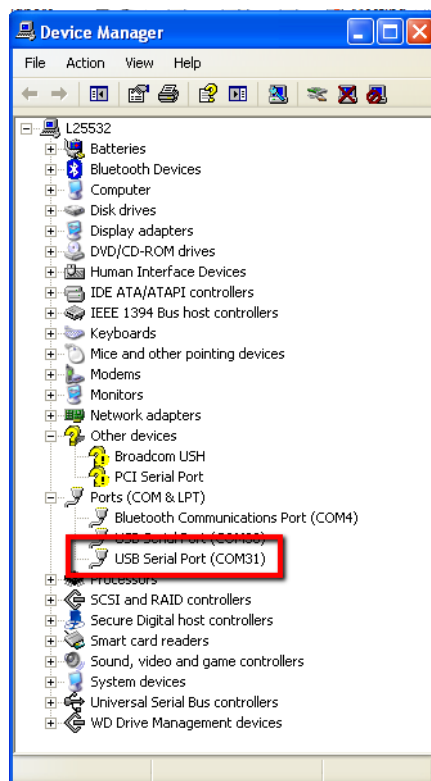
You will use a terminal program to communicate with the I²C master board. The following instructions describe the Windows HyperTerminal program which is found on most Windows PCs. You may use another terminal program but setup will be somewhat different. Windows 7 does not include HyperTerminal. Tera Term has been verified to work with Windows 7.

Note: This step uses the procedure for Windows XP users. Steps may vary slightly if using another Windows version.

1. From the **Start** menu, select **Control Panel > System**. The “System Properties” dialog box appears.

2. Select the **Hardware** tab and click **Device Manager**. The “Device Manager” dialog box appears.

Figure 5. Device Manager – COM Port



3. Expand the **Ports (COM & LPT)** entry and note the COM port number for the USB Serial Port.
4. From the **Start** menu, select **Programs > Accessories > Communications > HyperTerminal**. The HyperTerminal application and a “Connection Description” dialog box appear.

Figure 6. New Connection – COM Port



5. Specify a Name and Icon for the new connection. Click **OK**. The “Connect To” dialog box appears.
6. Select the COM port identified in Step 3 from the Connect using: list. Click **OK**.

Figure 7. Selecting the COM Port



7. The “COMn Properties” dialog appears where “n” is the COM port selected from the list.

8. Select the following Port Settings and click **OK**.

Bits per second: **115200**

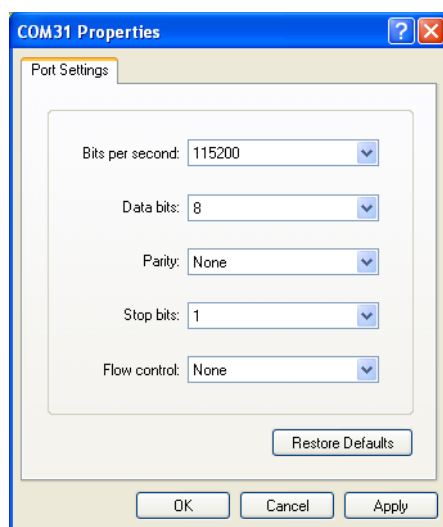
Data bits: **8**

Parity: **None**

Stop bits: **1**

Flow control: **None**

Figure 8. COM Port Properties



9. The HyperTerminal window appears.

10. Press the S1 push-button (GSR) of the I²C master board. The I²C Master/Slave demo main menu appears.

Running the Demo

On successfully setting up the two MachXO2 evaluation boards, various operations of the demo can be done through the HyperTerminal or using the on-board capsense buttons of the I²C master board.

Pressing (0-9)

Figures 9 and 10 show the activity on the HyperTerminal window and the on-board I²C transactions when a key between 0-9 is pressed (in this example key '3' was pressed).

I²C Data flow

Table 4 shows the necessary I²C packet structure and the different RD1124 commands accessed for this operation.

Table 4. I²C Packet Structure when UART Input is '3'

Issue Start	Slave Address (binary)	W/R	RD1124 Command (Hex)	Operand (Hex)	Data (Hex)	Issue Stop
Y	000_1001	W	06 (enable)	—	—	Y
Y	000_1001	W	01 (write GPIO)	00	03	Y
Y	000_1001	W	04 (disable)	—	—	Y

Figure 9. Activity on HyperTerminal when a Key Between 0-9 is Pressed

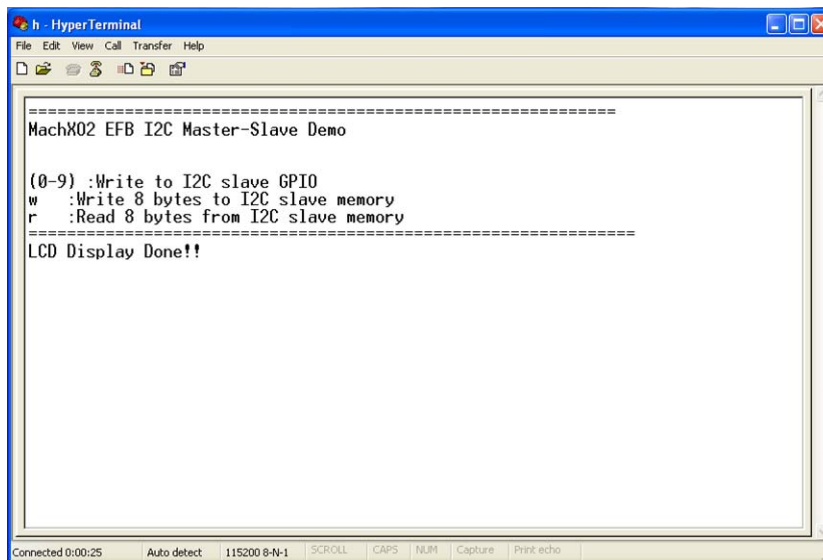
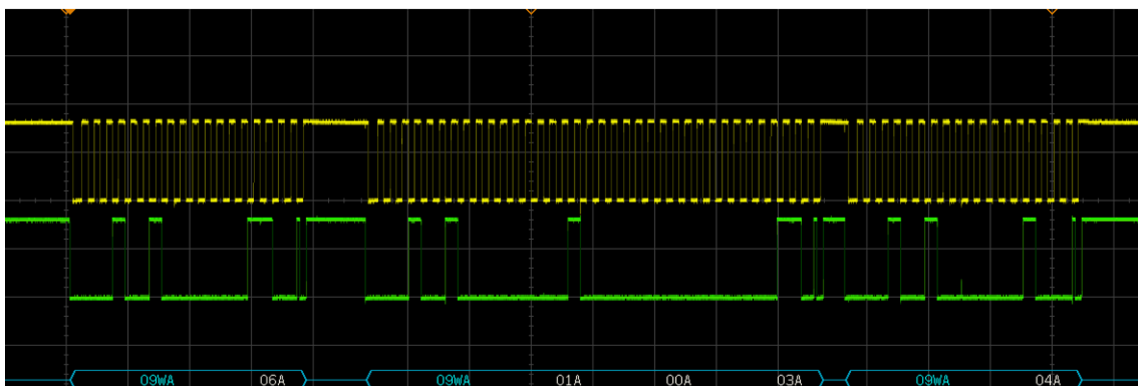


Figure 10. On-board I²C Signal Activities when a Key Between 0-9 is Pressed



Note: In Figure 10, the yellow signal indicates SCL, the green signal indicates SDA and the blue bus indicates the decoded value of the SDA line.

Pressing 'w'

Figures 11 and 12 show the activity on the HyperTerminal window and the onboard I²C transactions when the key 'w' is pressed.

I²C Data Flow

Table 5 shows the necessary I²C packet structure and the different RD1124 commands accessed for this operation

Table 5. I²C packet Structure when UART Input is 'w'

Issue Start	Slave Address (Binary)	W/R	RD1124 Command (Hex)	Operand (Hex)	Data (Hex)	Issue Stop
Y	000_1001	W	06 (enable)	—	—	Y
Y	000_1001	W	02 (write memory)	00 (start address)	01 02 03 04 05 06 07 08	Y
Y	000_1001	W	04 (disable)	—	—	Y

Figure 11. Activity on HyperTerminal when the Key 'w' is Pressed

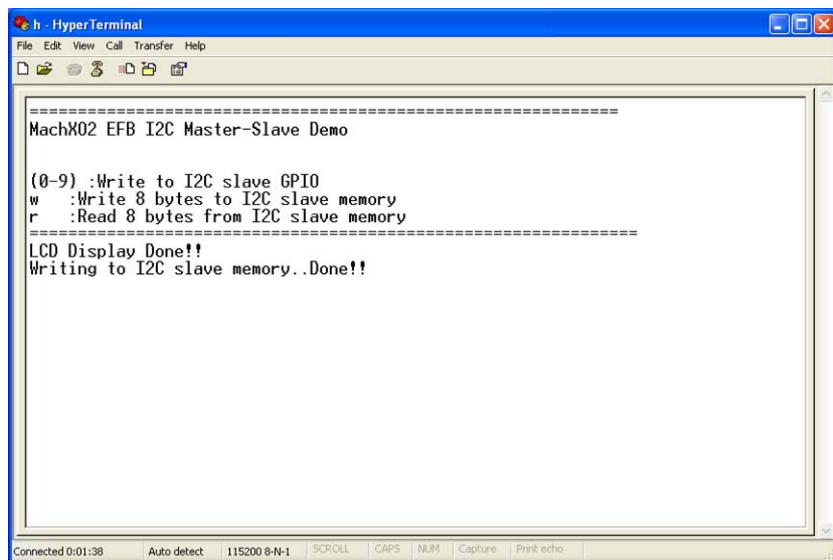
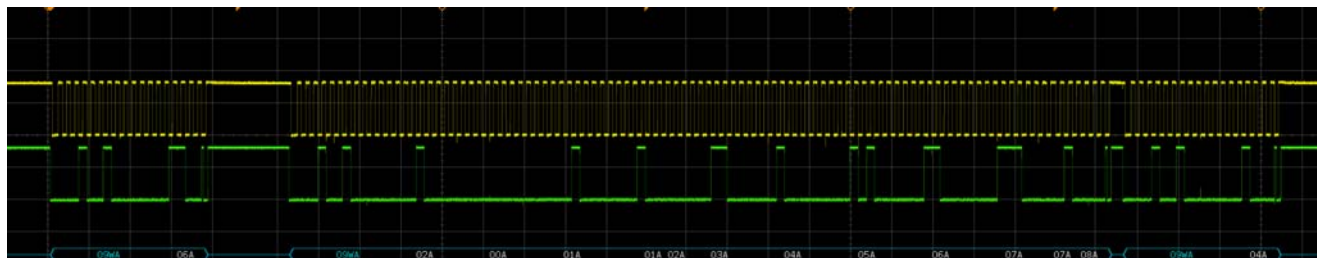


Figure 12. On-board I²C Signal Activities when the Key 'w' is Pressed



Pressing 'r'

Figures 13 and 14 show the activity on the HyperTerminal window and the on-board I²C transactions when the key 'r' is pressed.

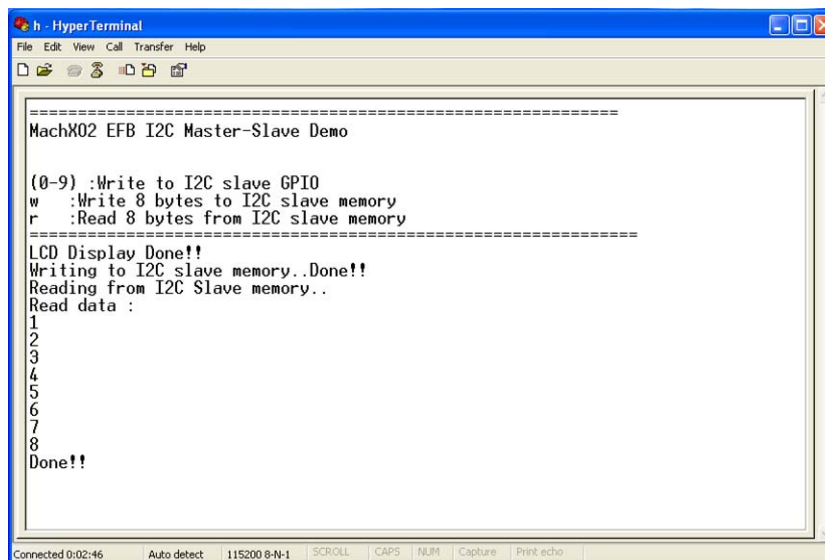
I²C Data flow

Table 6 shows the necessary I²C packet structure and the different RD1124 commands accessed for this operation

Table 6. I²C Packet Structure when UART Input is 'r'

Issue Start	Slave Address (Binary)	W/R	RD1124 Command (Hex)	Operand (Hex)	Data (Hex)	Issue Stop
Y	000_1001	W	06 (enable)	—	—	Y
Y	000_1001	W	0B (read memory)	00 (start address)	—	N
Y	000_1001	R	—	—	01 02 03 04 05 06 07 08	Y
Y	000_1001	W	04 (disable)	—	—	Y

Figure 13. Activity on HyperTerminal when the Key 'r' is Pressed

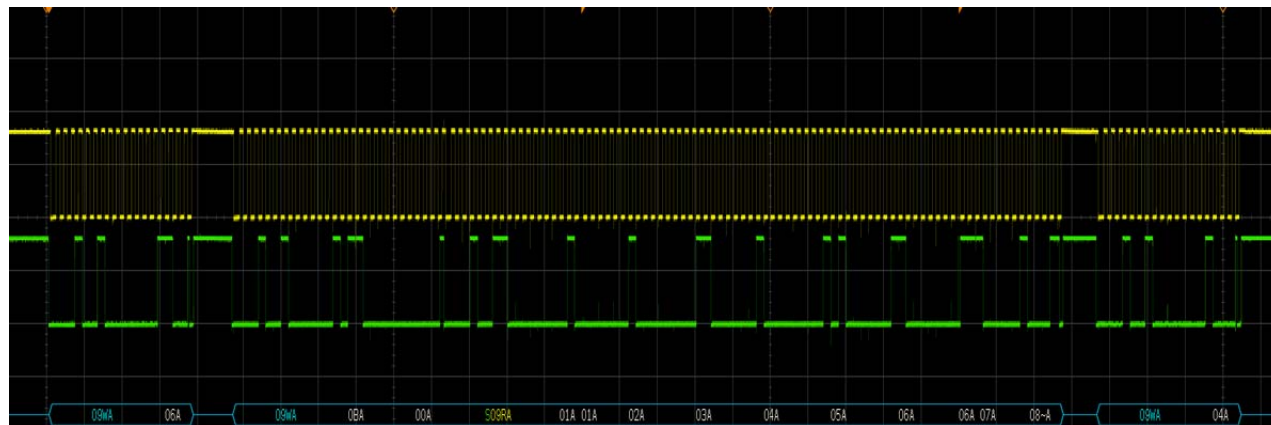


```

=====
MachX02 EFB I2C Master-Slave Demo

(0-9) :Write to I2C slave GPIO
w :Write 8 bytes to I2C slave memory
r :Read 8 bytes from I2C slave memory
=====
LCD Display Done!!
Writing to I2C slave memory..Done!!
Reading from I2C Slave memory..
Read data :
1
2
3
4
5
6
7
8
Done!!
  
```

Figure 14. On-board I²C Signal Activities when the Key 'r' is Pressed



Technical Support Assistance

Hotline: 1-800-LATTICE (North America)
+1-503-268-8001 (Outside North America)

e-mail: techsupport@latticesemi.com

Internet: www.latticesemi.com

Revision History

Date	Version	Change Summary
April 2012	01.0	Initial release.