

## Introduction

This section provides an overview of the MachXO2 Embedded Configuration Access (ECA) demo running from the LatticeECP3™ Versa Evaluation Board to the MachXO2™ Pico Evaluation Board. The demo shows how the MachXO2 Embedded Configuration Access reference design is implemented on a microcontroller (in this case the LatticeMico32™) and accesses the MachXO2 via an I<sup>2</sup>C bus to perform the following operations:

- Program the MachXO2 with a JEDEC file
- Read usercode and other device registers
- Update EBR init values in the User Flash Memory (UFM)
- Read/write pages in the UFM

*Note: See TN1204, [MachXO2 Programming and Configuration Usage Guide](#) and TN1205, [Using User Flash Memory and Hardened Control Functions in MachXO2 Devices](#), for further information on the MachXO2 configuration command capabilities.*

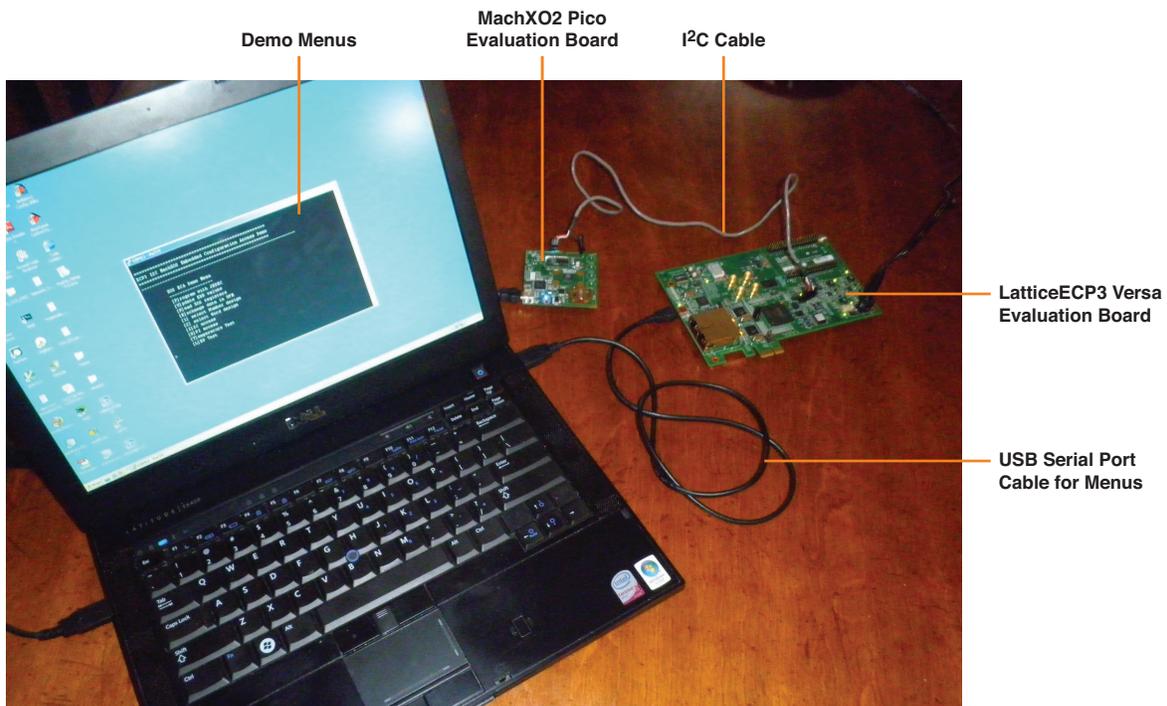
The demo shows the capabilities of programming a blank device with an initial design (or reprogramming with an updated design) as well as reading the MachXO2 DeviceID, usercode and other documented capabilities. These features are especially attractive in embedded systems where the MachXO2 device needs to be programmed or configured at run time or in pin-limited packages where a separate MachXO2 JTAG interface is not available.

The reference design provides an alternate method to the traditional ispVM™ Embedded tools. The ispVM Embedded package is adapted from the Windows/Linux based ispVM tool, which is targeted for operation as an application on a PC. The assumptions of operating in a PC environment do not always port well to the traditional embedded microcontroller environment, where such configuration of an MachXO2 would be applicable. This demo shows the routines performing configuration operations through pre-defined (and tested) methods (i.e. these ECA routines). Reusing these 'C' routines will be more economical to the user than incurring the expense of coding, testing and debugging custom routines based on TN1204, [MachXO2 Programming and Configuration Usage Guide](#) and TN1205, [Using User Flash Memory and Hardened Control Functions in MachXO2 Devices](#).

This demo provides a hands-on example for compiling the MachXO2 Embedded Configuration Access reference design routines, running the routines, accessing a MachXO2 device via I<sup>2</sup>C and performing various configuration operations on the device.

Figure 1 shows the LatticeECP3/MachXO2 I<sup>2</sup>C demo setup.

Figure 1. LatticeECP3/MachXO2 I<sup>2</sup>C Demo Setup



The serial port from the LatticeECP3 Versa Evaluation Board provides the user interface. The serial port is implemented using a USB connection. A mini-USB cable connects the LatticeECP3 Versa Evaluation Board to a PC. The PC runs a USB to a virtual COM port program that allows the HyperTerminal to connect to the LatticeECP3 Versa Evaluation Board over the USB cable.

The LatticeMico32 soft microprocessor in the LatticeECP3 device interfaces to the MachXO2 device with an I<sup>2</sup>C bus implemented as a three-wire cable between the two boards.

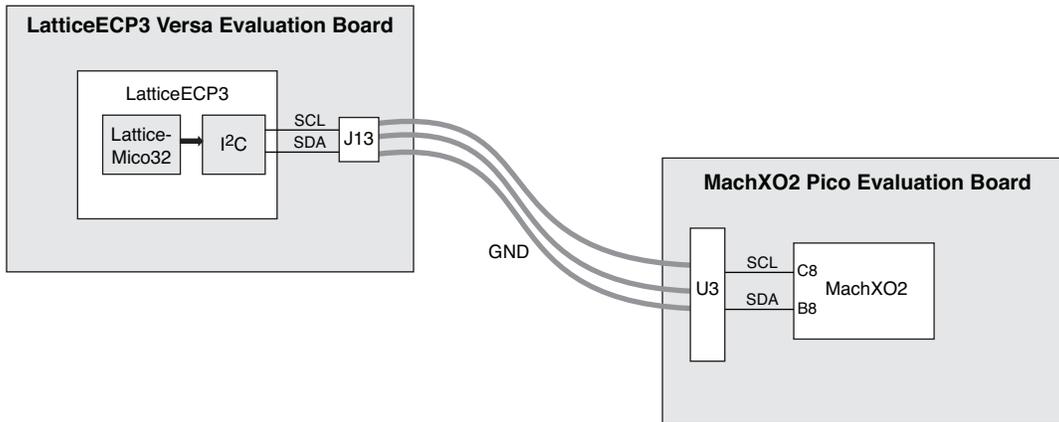
## Implementation

The purpose of the MachXO2 Embedded Configuration Access demo is to provide a working example of accessing MachXO2 Embedded Function Block (EFB) configuration logic via an I<sup>2</sup>C interface. The demo uses interactive menus to allow the user to try various use-case operations that would represent typical embedded accesses to the MachXO2:

- Program the MachXO2 with a JEDEC file
- Read usercode and other device registers
- Update EBR init values in the UFM
- Read/write pages in the UFM

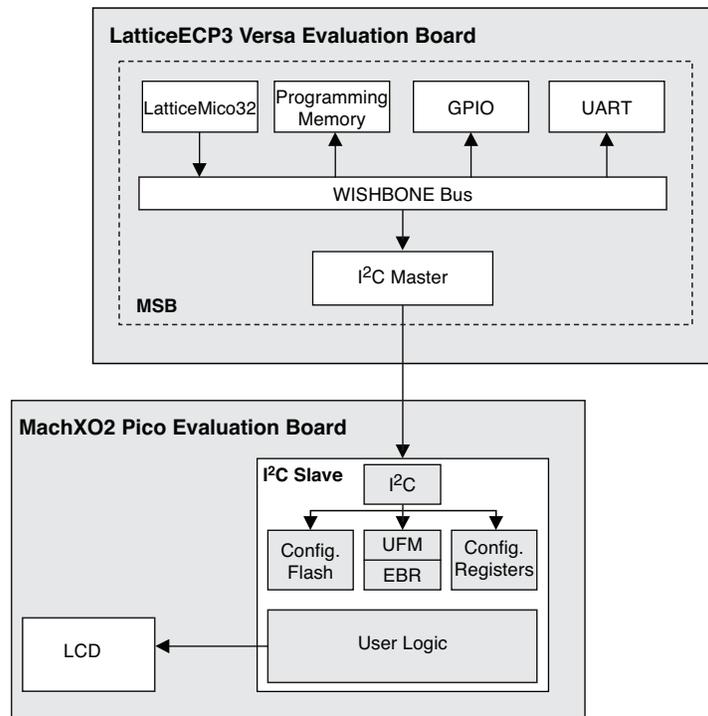
This reference design is implemented using the LatticeECP3 Versa Evaluation Board as the system control processor and the MachXO2 Pico Evaluation Board as the target MachXO2 device to access. A cable connects the I<sup>2</sup>C buses between the two boards. See [Appendix A. I2C Connector on the LatticeECP3 Versa Evaluation Board](#) and [Appendix B. I2C Connector on the MachXO2 Pico Evaluation Board](#) for specific board connector details.

**Figure 2. I<sup>2</sup>C Connection Between Evaluation Boards**



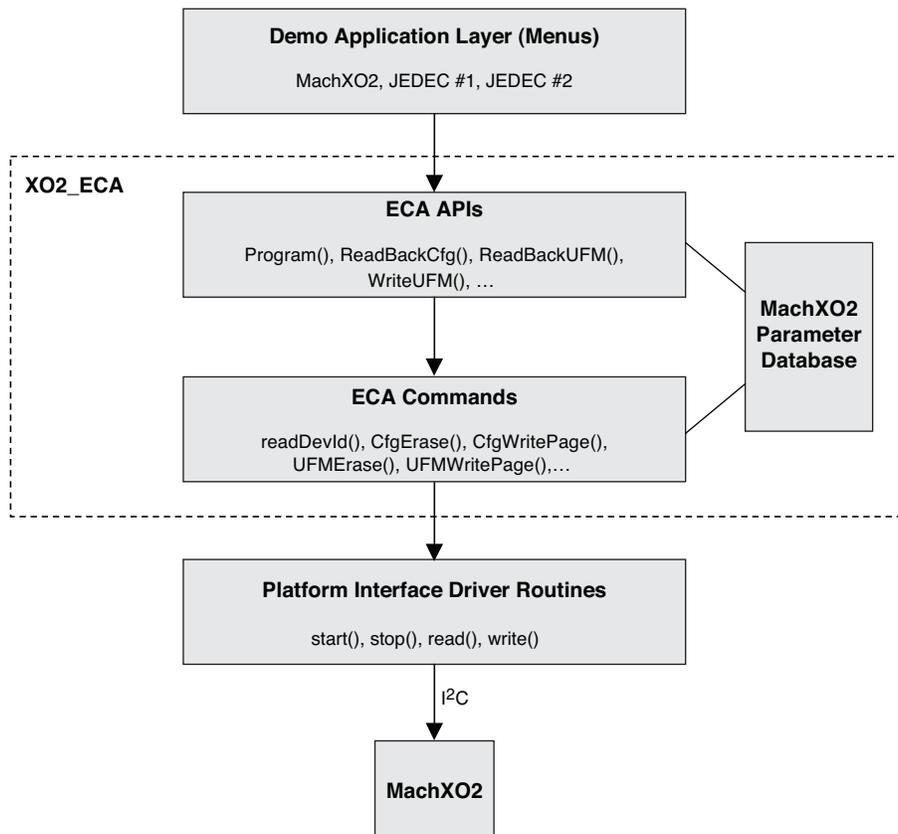
The user menus are provided by the LatticeMico32 demo software running on the LatticeECP3 Versa Evaluation Board. The user selects various demo options and the commands are executed over the I<sup>2</sup>C bus to perform the specific MachXO2 operation. Figure 3 shows the hardware/IP architecture of the demo.

**Figure 3. Demo Architecture**



The LatticeMico32 MSB processor system runs the entire user interface software from on-chip program memory. The processor system also contains the UART component that provides the serial port communications to the external PC. An I<sup>2</sup>C master component provides access to the MachXO2 configuration block via the built-in I<sup>2</sup>C port. The program memory of the MSB contains all the MachXO2 access drivers, commands and two JEDEC files for programming the MachXO2 device with differently behaving designs, in order to visually see that a new design has been loaded.

Figure 4. Software Architecture



The flow of control is handled entirely from the menus in the user application layer. The user invokes ECA API routines using the menu options. The menu options call the APIs and pass a handle to the specific MachXO2 device to on which the operations will be performed. The ECA APIs implement the algorithms specified in TN1204, [MachXO2 Programming and Configuration Usage Guide](#) and TN1205, [Using User Flash Memory and Hardened Control Functions in MachXO2 Devices](#) for performing macro operations such as erase configuration Flash, read usercode, etc. The ECA APIs in turn call ECA commands, which implement the specific sequence of configuration command opcodes and operands, as per TN1204 and TN1205, for doing one specific operation. The ECA command routines then invoke, via function pointers in the MachXO2 handle, the specific interface driver routines to start a transfer, write bytes/read bytes and stop a transfer. The platform-specific driver routines are then invoked to perform the I<sup>2</sup>C bus transfers to the specific MachXO2 device.

The XO2\_ECA routines are provided as source code and are built into the menu application running on the LatticeECP3 Versa Evaluation Board. An internal data structure holds the specific parameters for each supported MachXO2 device, such as the number of pages in the UFM, or configuration Flash erase time, etc. These parameters are used by the APIs and commands when performing operations.

### Known Limitations

This section describes some of the limitations of the MachXO2 Embedded Configuration Access demo.

#### Compression

The JEDEC data could be compressed, greatly reducing code storage requirements. A simple run-length encoding compression routine has been shown to be very effective (2x to 4x compression) but is not implemented in this version.

### Only I<sup>2</sup>C

Only I<sup>2</sup>C is demonstrated at this time. The software framework is set up to support other interfaces, such as SPI, to the MachXO2 device, but only a single I<sup>2</sup>C interface is used in this demo.

### Transparent Mode Programming

The MachXO2 Pico Evaluation Board only supports MachXO2 Flash programming in Transparent mode. The power to the I<sup>2</sup>C bus on the board is controlled by a pin on the MachXO2 device. If the MachXO2 does not have a user design in it to drive this output pin high, the I<sup>2</sup>C bus will not have the required pull-ups, and other devices connected to the I<sup>2</sup>C bus will not operate properly. Therefore, Offline mode cannot be demonstrated with a standard MachXO2 Pico Evaluation Board because the user design is halted, and all MachXO2 I/Os default to tri-state output low, disabling the I<sup>2</sup>C. A hardware modification can be made to the MachXO2 Pico Evaluation Board that provides constant pull-up voltage to the I<sup>2</sup>C bus. See [Appendix C. I2C Bus Pull-up Modification](#) for this modification if offline mode operation is desired.

## Setup

This section describes how to connect and set up the MachXO2 Embedded Configuration Access demo.

### General Warning

Please observe the following important safety items:

- Follow ESD precautions
- Use only the power supply provided with the LatticeECP3 Versa Evaluation Board. Do not use a generic 12V supply.

### Required Demo Components

- LatticeECP3 Versa Evaluation Board
- 12V DC power supply for LatticeECP3 Versa Evaluation Board
- Mini USB cable for LatticeECP3 Versa Evaluation Board
- MachXO2 Pico Evaluation Board
- Mini USB cable to power the MachXO2 Pico Evaluation Board
- I<sup>2</sup>C cable (three-wire cable connecting the LatticeECP3 Versa Evaluation Board to the MachXO2 Pico Evaluation Board – see [Appendix A. I2C Connector on the LatticeECP3 Versa Evaluation Board](#) and [Appendix B. I2C Connector on the MachXO2 Pico Evaluation Board](#))

### Project Setup

1. Unzip the reference design package.

The demo bitstream that needs to be loaded into the LatticeECP3 Versa Evaluation Board is located in the Bitstreams directory.

### LatticeECP3 Versa Evaluation Board Installation

1. Load the bitstream located in the top-level Bitstreams directory. This contains all the demo menus, MachXO2 I<sup>2</sup>C access routines and two MachXO2 JEDEC designs to be loaded into the MachXO2.
2. Assemble the I<sup>2</sup>C cable.
3. Connect the USB cable to the PC for serial port access to the menus.

### MachXO2 Pico Evaluation Board Installation

1. Connect the I<sup>2</sup>C cable.

2. Connect the USB cable from the MachXO2 Pico Evaluation Board to the PC for power.
3. Install an initial JEDEC design into the MachXO2 Pico Evaluation Board. The board does not provide I<sup>2</sup>C bus power by default. A design needs to be loaded into the MachXO2 device that asserts the power enable control line in order for the LatticeECP3 device to see the I<sup>2</sup>C bus. The JEDEC file to load is in the Bitstreams directory. Load XO2\_I2C\_slave\_XO2\_numbers.jed.

## FTDI COM Port Driver Installation

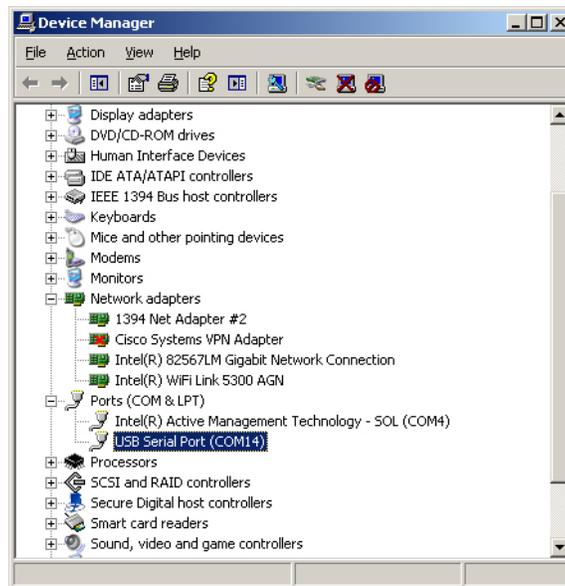
The next step is to connect the USB cable to the mini-USB connector on the front of the LatticeECP3 Versa Evaluation Board. Connect the other end to an available USB slot on your PC. This will provide the connection to the menus on the LatticeECP3 Versa Evaluation Board. You will need to install a driver provided by Future Technology Devices International (FTDI), the manufacturer of the FT232RL device used on the LatticeECP3 Versa Evaluation Board that converts the RS-232 UART output to USB). The driver converts the PC USB port to a COM port so that a serial terminal program (HyperTerminal) can open the USB port and communicate over it as if it were an RS-232 port. The FTDI driver installation program can be downloaded from the FTDI website.

*NOTE: Windows Vista and Windows 7 do not ship with HyperTerminal. Users will need to install a suitable serial port terminal emulator such as PuTTY 0.60 (or newer) or the XP version of HyperTerminal.*

Once the driver is installed, power up the LatticeECP3 Versa Evaluation Board by connecting the power supply.

To determine which COM port the USB serial device is mapped to, open **My Computer > Properties > Hardware > Device Manager** and select **Ports** to see available serial ports and choose the one that is the USB Serial Port. See the Figure 5 as an example.

**Figure 5. Finding New Comm Port**



Open your terminal program and select the new USB serial port. Set the communication settings to: 115000 baud, 8 data bits, No Parity, 1 stop bit (115000,8,N,1). Figures 6 and 7 show these settings for a PuTTY session.

Figure 6. Opening Comm Port with PuTTY

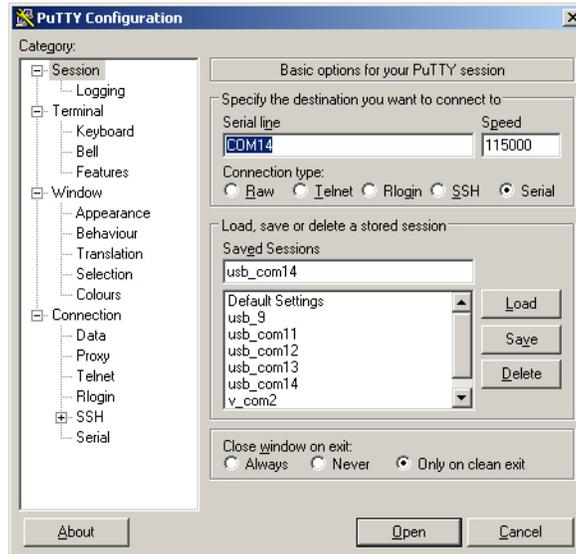
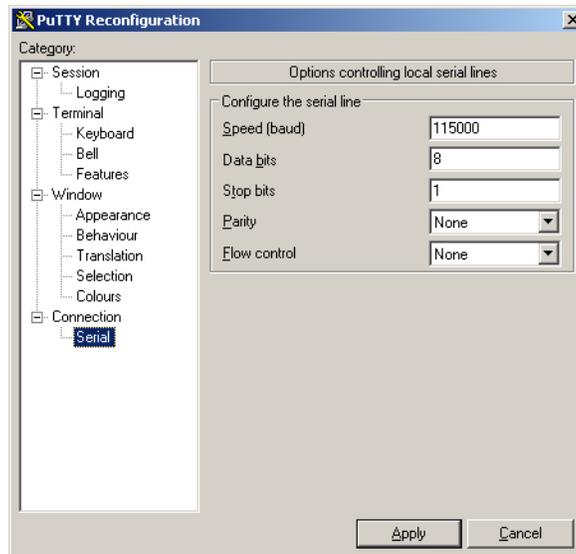


Figure 7. Comm Port Settings



After the serial port settings are configured, and power is applied to the LatticeECP3 Versa Evaluation Board, the main menu will appear (press the LatticeECP3 Versa Evaluation Board reset button to redisplay).

## Hardware Settings

For demo operation, there is no hardware configuration necessary other than connecting the I<sup>2</sup>C cable and USB cables. This section is provided for reference or in case the LatticeECP3 Versa Evaluation Board or MachXO2 Pico Evaluation Board has been used in other applications and their current configuration is in doubt.

### LatticeECP3 Versa Evaluation Board

The only hardware indicator of demo activity is the red LED D25 blinking to indicate the system clock is running. This indicates that the bitstream has loaded correctly and that basic logic is functional. It does not reflect the I<sup>2</sup>C or LatticeMico32 operating status.

No other LED indicators are used at this time.

## MachXO2 Pico Evaluation Board

The LCD is used to indicate the particular JEDEC file design that has been loaded into configuration Flash.

## Demo Operation

All user interaction and demo control is through the text menus provided by the LatticeMico32 soft microprocessor. This section will show and discuss the available menu options that demonstrate the MachXO2 configuration accesses.

## Background

There are two MachXO2 designs provided in the reference design package. Both have the same basic design. They read bytes from an initialized EBR memory and display four characters at a time on the LCD display. One design uses an EBR initialized with hex digits, for example '0123', '4567', 'ABCD'. The other design uses an EBR initialized with four character words (that can be spelled using only alphabetic digits of hex numbers), for example 'DEAD' or 'CAFÉ' or 'FEED'. Each design also has a unique usercode. These differences allow the user to see the results of loading different designs via I<sup>2</sup>C and reading back the configuration registers to verify the proper design is loaded (i.e., usercode value).

### XO2\_numbers Implementation

- LCD display: "0123", "4567", "1111", "2222", etc.
- EBR init file: Hardware\XO2\_Pico\Source\numbers.mem
- UserCode: 0x01234567
- TracerID: 0x12

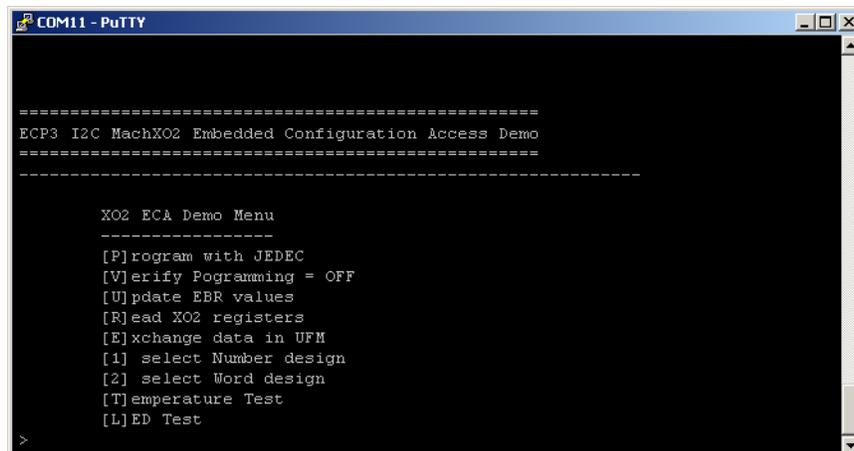
### XO2\_words Implementation

- LCD display: "dead", "beef", "abba", "café", "feed", etc.
- EBR init file: Hardware\XO2\_Pico\Source\words.mem
- UserCode: 0xDEADBEEF
- TracerID: 0xAB

## Menu Screen

The menu screen is the starting point for running the demo software. The user enters the letter of the operation to execute.

**Figure 8. Menu Screen**



```
COM11 - PuTTY
=====
ECP3 I2C MachXO2 Embedded Configuration Access Demo
=====
-----
XO2 ECA Demo Menu
-----
[P]rogram with JEDEC
[W]erify Programming = OFF
[U]pdate EBR values
[R]ead XO2 registers
[E]xchange data in UFM
[1] select Number design
[2] select Word design
[T]emperature Test
[L]ED Test
>
```

## LED Test

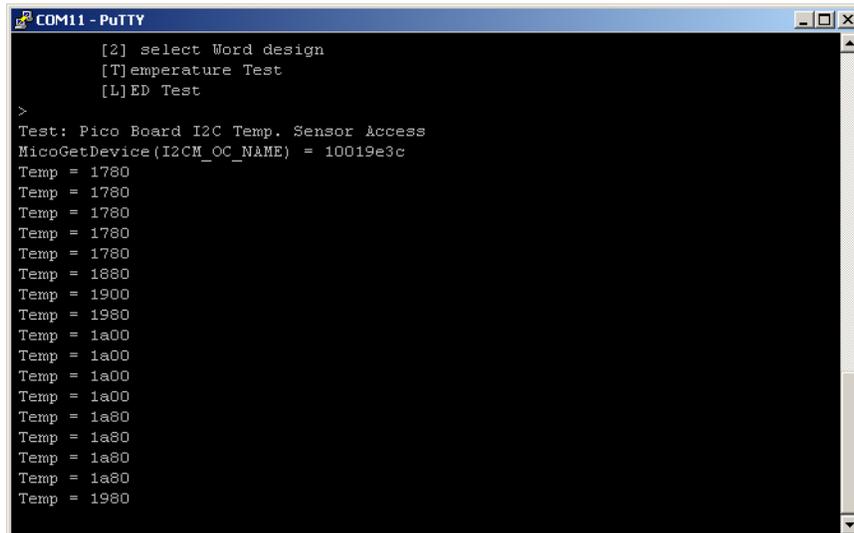
The LED test simply demonstrates that the LatticeECP3 is working and that the FPGA and software running on the LatticeMico32 are functioning correctly. The test blinks a visual indicator on the LatticeECP3 Versa Evaluation Board to show that the software is working, even if I<sup>2</sup>C portions are having trouble. The eight discrete LEDs on the left side light (move) up and down and the segments on the 14-segment display light, indicating proper functioning of the LatticeECP3 software and hardware.

```
-----  
Running ECP3 Versa Board LED Test...  
-----
```

## Temperature Test

The Temperature test demonstrates reading, over the I<sup>2</sup>C cable, the Texas Instruments temperature sensor device on the MachXO2 Pico Evaluation Board. This provides a way to validate proper I<sup>2</sup>C bus operations using a third-party device. The test prints the 12-bit register value from reading the temperature. It is not converted to Fahrenheit or Celsius; just a raw hex value. Placing a finger on the device will dynamically change the reading.

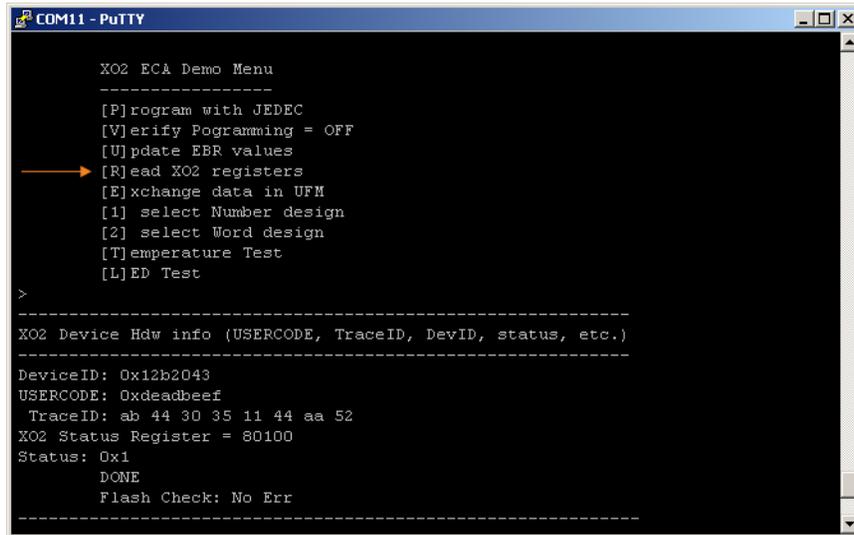
**Figure 9. I<sup>2</sup>C Temperature Display**



## Read MachXO2 Registers

Reading the MachXO2 configuration registers demonstrates how a host processor could interrogate the specific MachXO2 device or design (usercode) that is running. System operation could then be altered based on the information obtained, such as limiting features if the usercode indicates a "base" model system.

Figure 10. Read MachXO2 Configuration Registers



```

COM11 - PuTTY
XO2 ECA Demo Menu
-----
[P]rogram with JEDEC
[V]erify Programming = OFF
[U]pdate EBR values
[R]ead XO2 registers
[E]xchange data in UFM
[1] select Number design
[2] select Word design
[T]emperature Test
[L]ED Test
>
-----
XO2 Device Hdw info (USERCODE, TraceID, DevID, status, etc.)
-----
DeviceID: 0x12b2043
USERCODE: 0xdeadbeef
TraceID: ab 44 30 35 11 44 aa 52
XO2 Status Register = 80100
Status: 0x1
      DONE
      Flash Check: No Err
-----

```

For the demonstration, the usercode is different for each MachXO2 design. The usercode value indicates:

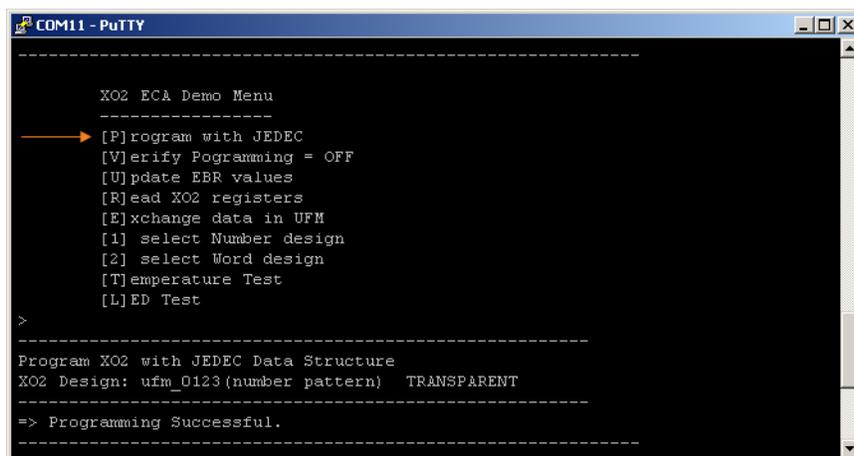
- 0x01234567 – Design that prints number patterns to the LCD display
- 0xdeadbeef – Design that prints “words” to the LCD display

The status registers are also read and decoded to identify errors that could occur during operation.

### Program with JEDEC Data

This operation reprograms the entire contents of the MachXO2 Flash (configuration, UFM and Feature Row sectors) with the JEDEC file data. The design to load is chosen with ‘1’ or ‘2’. ‘1’ selects loading the MachXO2 design that displays digits on the 4-character LCD display. ‘2’ selects loading the MachXO2 design that displays words, using the letter digits of hexadecimal, on the 4-character LCD display.

Figure 11. Program with JEDEC File



```

COM11 - PuTTY
-----
XO2 ECA Demo Menu
-----
[P]rogram with JEDEC
[V]erify Programming = OFF
[U]pdate EBR values
[R]ead XO2 registers
[E]xchange data in UFM
[1] select Number design
[2] select Word design
[T]emperature Test
[L]ED Test
>
-----
Program XO2 with JEDEC Data Structure
XO2 Design: ufm_0123 (number pattern) TRANSPARENT
-----
=> Programming Successful.
-----

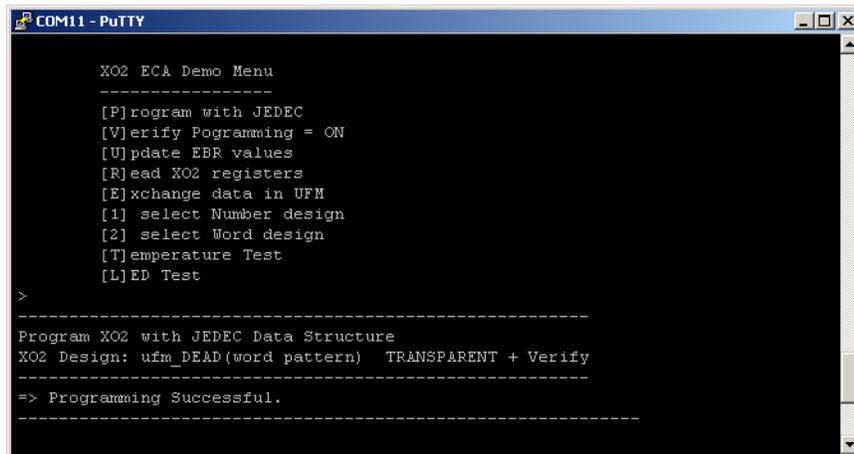
```

In this example, design 1 (the initial default) is loaded. The LCD display will now show patterns of digits (0-9), read from the EBR init values. For example: ‘0000’, ‘1111’, ‘2222’.

## Verify

The Verify menu option enables checking of the programmed Flash. By default, it is turned off. To enable the verification of programmed JEDEC data, press **V**. After each Flash sector is programmed, the pages will be read back one at a time and compared with the programmed data. If an error is detected, programming aborts with an error.

Figure 12. Program with Verify



```

COM11 - PuTTY

XO2 ECA Demo Menu
-----
[P]rogram with JEDEC
[V]erify Programming = ON
[U]pdate EBR values
[R]ead XO2 registers
[E]xchange data in UFM
[1] select Number design
[2] select Word design
[T]emperature Test
[L]ED Test
>

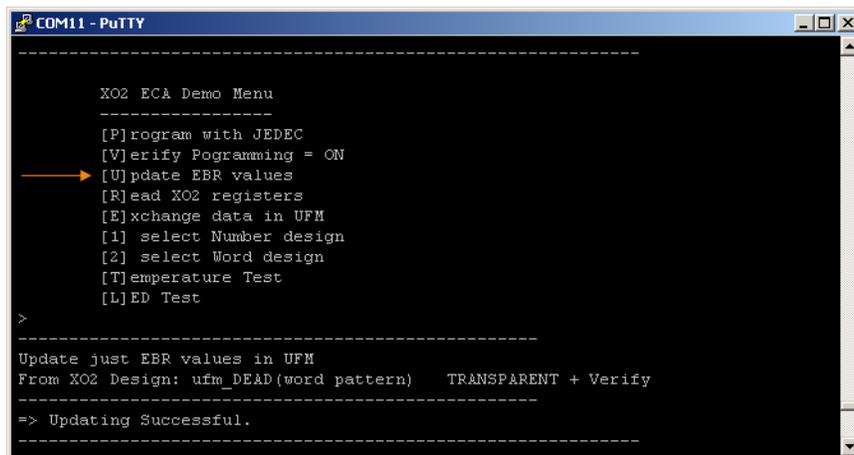
-----
Program XO2 with JEDEC Data Structure
XO2 Design: ufm_DEAD(word pattern)  TRANSPARENT + Verify
-----
=> Programming Successful.
-----

```

## Update EBR Values

This selection reprograms the MachXO2 UFM sector with the JEDEC file UFM data, effectively changing the behavior of the example MachXO2 hardware design by changing the EBR contents. The EBR contents are either loaded with the number pattern (0-9) or the word pattern (abcd) based on previous selections of '1' or '2'.

Figure 13. Update UFM Contents



```

COM11 - PuTTY

XO2 ECA Demo Menu
-----
[P]rogram with JEDEC
[V]erify Programming = ON
[U]pdate EBR values
[R]ead XO2 registers
[E]xchange data in UFM
[1] select Number design
[2] select Word design
[T]emperature Test
[L]ED Test
>

-----
Update just EBR values in UFM
From XO2 Design: ufm_DEAD(word pattern)  TRANSPARENT + Verify
-----
=> Updating Successful.
-----

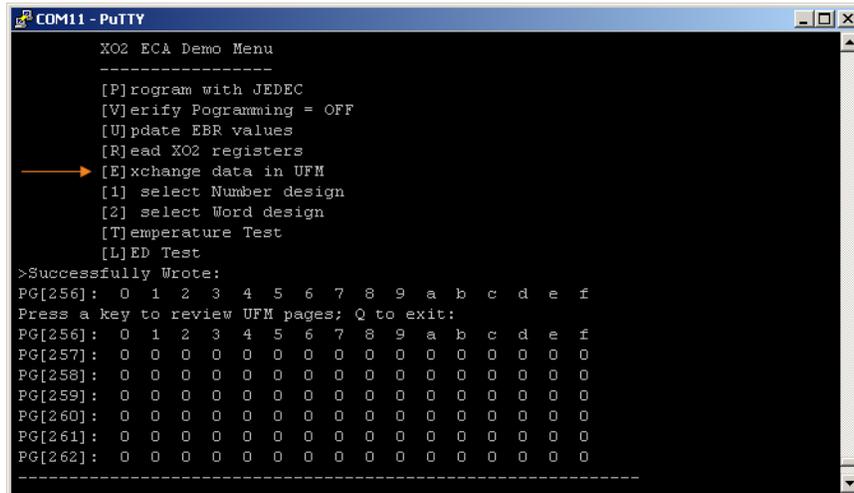
```

In this example, the EBR contents in the UFM are reloaded with the data for the word pattern. The configuration Flash sector remains unchanged and the design continues to operate in hardware. The only change is that once programming the UFM EBR completes, the LCD will now display words, instead of the previous number pattern.

## Exchange Data in the UFM

This option allows you to program a page at a time into the UFM. An incrementing pattern is used as the data to program. After writing the page, you then can display the contents of the UFM.

Figure 14. UFM Page Access



For this demo, UFM writable pages start at page number 256 to avoid overwriting the EBR initial values used in the design. In a real system, the user needs to be aware of how much EBR memory is used, and where the first available page begins. See TN1205, [Using User Flash Memory and Hardened Control Functions in MachXO2 Devices](#), for a detailed description and example of planning UFM page use.

## Building Hardware Designs

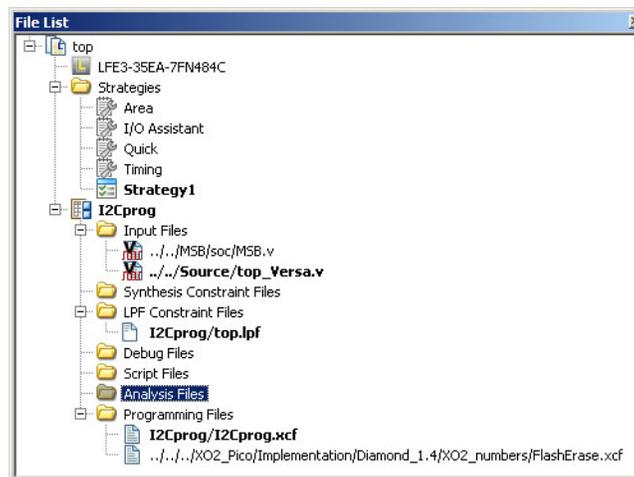
The pre-built LatticeECP3 bitstream containing all the demo menus, MachXO2 programming files, etc., is included in the demo package in the top-level bitstreams directory. Only this needs to be loaded into the LatticeECP3 Versa Evaluation Board SPI Flash for immediate demo operation. This section discusses how to rebuild the projects for the MachXO2 and LatticeECP3 if changes or experimentation are to be done, or just to understand the overall flow.

### LatticeECP3 Lattice Diamond® Project

The LatticeECP3 project is located in Hardware\ECP3\_Versa\Implementation\Diamond\_3.3.

The project builds the LatticeMico32 system and connects the I<sup>2</sup>C master component to the proper connector on the LatticeECP3 Versa Evaluation Board for interfacing to the MachXO2 Pico Evaluation Board.

Figure 15. LatticeECP3 Diamond Project



Building the project is straightforward, although the MSB platform needs to be generated first and the program memory initialized with the demo menu code and the MachXO2 ECA routines, as well as the JEDEC data for the two MachXO2 designs.

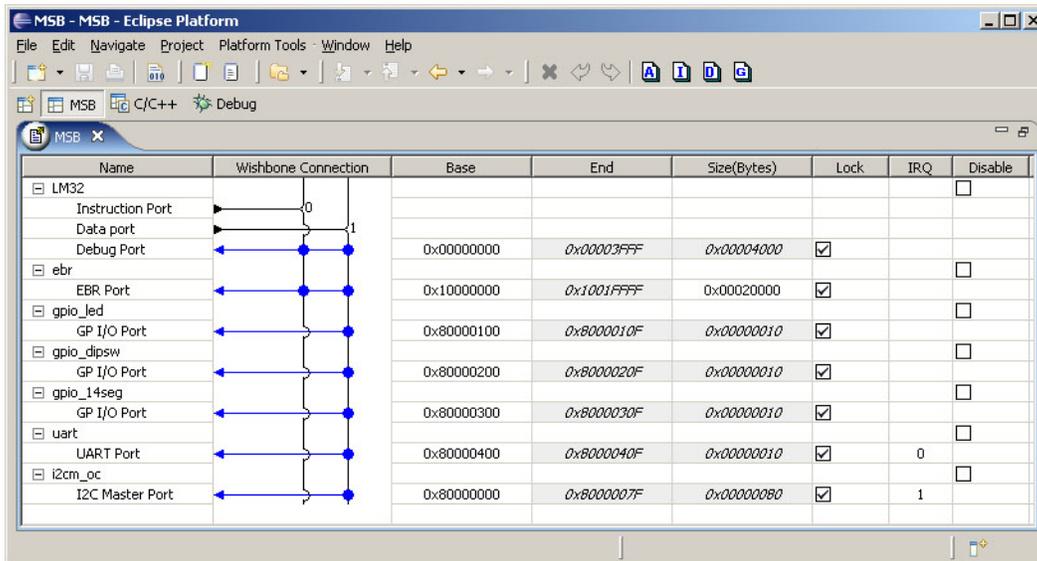
Programming is accomplished with the included Programmer .XCF file.

A FlashErase.xcf file is also included which will fully erase the MachXO2 to simulate I<sup>2</sup>C access to a blank device.

## LatticeECP3 LatticeMico32 Platform and Software

The LatticeECP3 LatticeMico32 MSB platform emulates a typical embedded processor system. The 32-bit LatticeMico32 processor accesses on-chip memory for program and data storage. It uses the UART for user I/O and the I<sup>2</sup>C master component to access the MachXO2 device.

**Figure 16. LatticeECP3 LatticeMico32 Platform**



The LatticeMico32 MSB platform is located in:

Hardware\ECP3\_Versa\MSB\soc\MSB.msb

The 'C' project is located in:

\Software\Mico32\XO2ECA

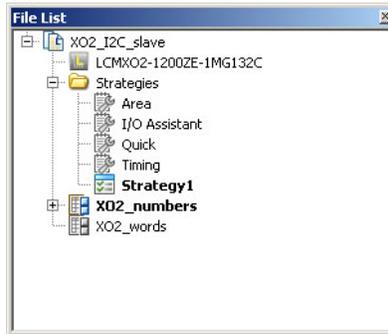
After building the 'C' source code, be sure to deploy the software to the on-chip memory initialization file: Software\Mico32\prog.mem and update the memory initialization file path by double-clicking on the ebr component in MSB perspective. Run Generator to update the path in the MSB.v instantiated in the Diamond project.

## MachXO2 Diamond Projects

There are two MachXO2 projects that produce different visual results, so the user can see and verify that a different design has been loaded into the MachXO2. The MachXO2 projects for LSE (Lattice Synthesis Engine) synthesis tool are located in \Hardware\XO2\_Pico\Implementation\Diamond\_3.3\.

Open the XO2\_I2C\_slave.ldf Diamond project file located there. One MachXO2 implementation (project) is called XO2\_numbers, the other is XO2\_words. Both share common source files. The main difference is in the EBR init value file contents and preferences to change the usercode value.

Figure 17. MachXO2 Diamond Project



Each builds a JEDEC file in the respective project directory:

- XO2\_numbers\XO2\_I2C\_slave\_XO2\_numbers.jed
- XO2\_words\XO2\_I2C\_slave\_XO2\_words.jed

The JEDEC file contents are converted into the data structure needed by the MachXO2 ECA software routines. The conversion is done using a Windows utility `jed2ecaC.exe`, discussed in the following section.

The JEDEC files can also be programmed directly into the MachXO2 using the Programmer .XCF files included in each project.

## JEDEC File Conversion

The MachXO2 JEDEC file needs to be converted into a more compact, 'C' compilable format, for inclusion directly in the embedded system that will be programming the MachXO2 device. The ECA routines do not expect file system access, and so do not read the JEDEC file directly. The ECA routines expect the JEDEC file contents to be converted to binary 'C' arrays, for simple and easy access to the bytes needed to be written into the MachXO2 configuration Flash sector and/or UFM sector.

*Note: If the targeted embedded system does not have a file system, and space is not an issue, JEDEC files can be left in their native form, and a modified version of this conversion utility could be implemented to read JEDEC files and convert directly to the C array needed by the ECA routines. Systems running Linux or having network file access may opt for this method. The examples provided here are for a more traditional, resource limited embedded system that need to store all MachXO2 JEDEC data in program data memory.*

An ANSI 'C' conversion utility is provided in source format so it can be built on any system, and adapted as needed to parsing the MachXO2 JEDEC file. The utility is named `jed2ecaC`, meaning it converts a JEDEC file into an MachXO2 ECA 'C' data structure containing arrays for the Configuration, UFM and Feature Row sector data. The conversion program is located in: `Software\PC\jed2ecaC`.

See the `ReadMe.txt` file for information about running.

Execute the `convert.bat` batch program to generate the files needed by the demo and copy them into the LatticeMico32 software build directory.

## References

- TN1204, [MachXO2 Programming and Configuration Usage Guide](#)
- TN1205, [Using User Flash Memory and Hardened Control Functions in MachXO2 Devices](#)
- EB56, [LatticeECP3 Versa Evaluation Board User's Guide](#)
- EB61, [MachXO2 Pico Development Kit User's Guide](#)
- Digital Temperature Sensor with I<sup>2</sup>C Interface (Texas Instruments Inc.)

---

## Technical Support Assistance

e-mail: [techsupport@latticesemi.com](mailto:techsupport@latticesemi.com)

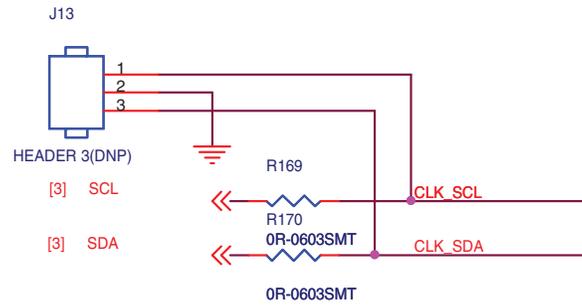
Internet: [www.latticesemi.com](http://www.latticesemi.com)

## Revision History

Date	Version	Change Summary
January 2015	1.1	Updated to support Diamond 3.3 design software for MachXO2.
		Updated to support LSE for MachXO2.
		Updated <a href="#">LatticeECP3 LatticeMico32 Platform and Software</a> section. Updated information on deploying the software to the on-chip memory initialization file.
		Updated <a href="#">Technical Support Assistance</a>
		Updated <a href="#">Appendix A. I2C Connector on the LatticeECP3 Versa Evaluation Board</a> section. Updated Figure 18, I2C Connector on the LatticeECP3 Versa Evaluation Board.
May 2012	01.0	Initial release.

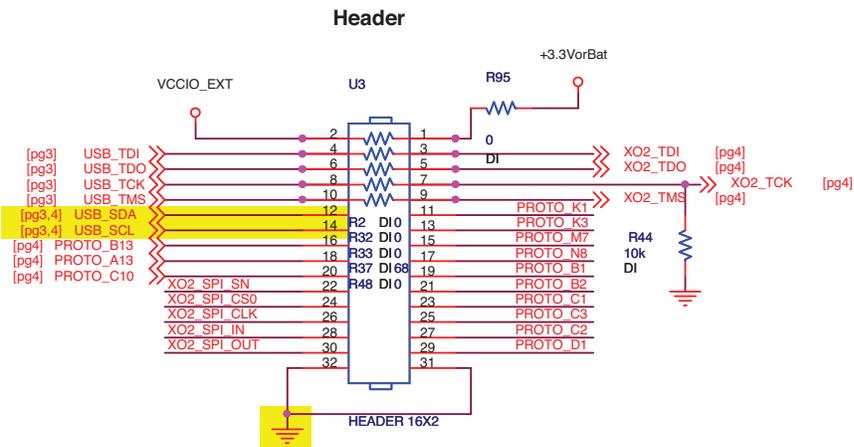
## Appendix A. I<sup>2</sup>C Connector on the LatticeECP3 Versa Evaluation Board

Figure 18. I<sup>2</sup>C Connector on the LatticeECP3 Versa Evaluation Board



## Appendix B. I<sup>2</sup>C Connector on the MachXO2 Pico Evaluation Board

Figure 19. I<sup>2</sup>C Connector on the MachXO2 Pico Evaluation Board

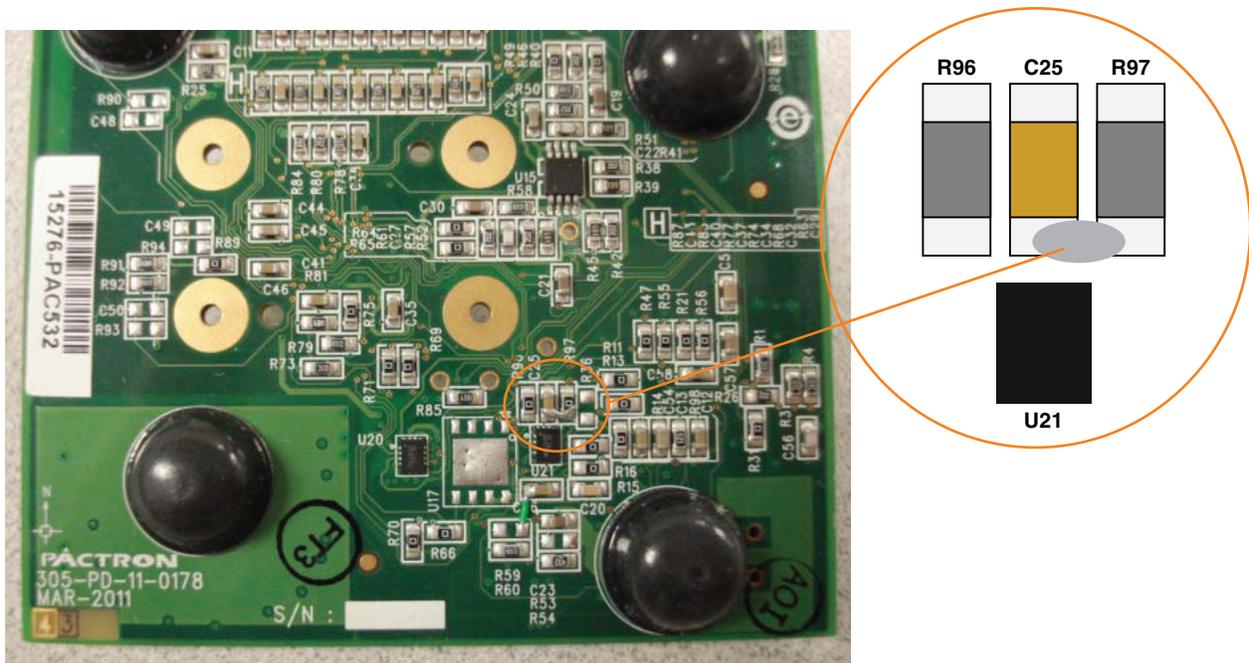


## Appendix C. I<sup>2</sup>C Bus Pull-up Modification

The MachXO2 Pico Evaluation Board has a feature that enables low power mode by disabling unused peripheral devices. The 3.3 V power supply for certain devices is controlled by an output pin of the MachXO2 device. The 3.3 V supply powers the I<sup>2</sup>C bus pull-up resistors and I<sup>2</sup>C temperature device. Enabling this low power mode unfortunately disables the I<sup>2</sup>C bus. When the MachXO2 device is unprogrammed/blank, all outputs are tri-stated pulled low, thus the power to the I<sup>2</sup>C bus is turned off and the MachXO2 device is not accessible from the I<sup>2</sup>C Master (LatticeECP3 Versa Evaluation Board).

In order to operate the demo in Offline mode on the MachXO2 Pico Evaluation Board, the following hardware modification can be made to ensure the I<sup>2</sup>C bus is always powered. Solder a short between C25 and R97 to enable the I<sup>2</sup>C and SPI power supply rail provide constant 3.3 V to the I<sup>2</sup>C pull-up resistors. The following diagram illustrates the location of the modification on the back side of the board.

**Figure 20. MachXO2 Pico Evaluation Board I<sup>2</sup>C Bus Power Enable**



The demo software on the LatticeECP3 Versa Evaluation Board also needs to be rebuilt in order to enable the offline mode menu option. Uncomment the following line in main.c:

```
#define ENABLE_ALL_MODES
```

Rebuild the Mico C/C++ project and deploy the software prog.mem so that when rebuilding the LatticeECP3 design, the new demo software is loaded into the EBR init values.