

**Power Electronics Building Blocks**  
**“Plug and Play”**  
**Hardware and Software Control Architectures**

Test Plan for the Universal Controller Hardware

Prepared By:  
Jerry Francis

12 July 2001

## 1. INTRODUCTION

This test plan documents the test procedure used to test the Universal Controller board for the Power Electronics Building Blocks Plug and Play and System Integration project. The Universal Controller is a digital board with two analog signals. It contains an FPGA, a DSP, a PCI interface, two fiber optic daisy-chain interfaces, two hex displays, a digital to analog converter, several general purpose IO connectors, two EEPROMs accessible to the DSP and FPGA, and a configuration EEPROM used to configure the FPGA.

## 2. COMPONENTS

### 2.1. Download and Start FPGA

The user must be able to configure the FPGA with code. There are two ways to program the FPGA. One is via the SelectMap interface and the other is via the JTAG interface. In order to accomplish this, several software and hardware components are required. First, the cable to program the FPGA must be present. This cable is the Xilinx MultiLinx cable. This cable supports both programming methods. This task also requires Xilinx Foundation express software and Xilinx ChipScope ILA software. The user must use the constraints file from the VHDL code for the universal controller, write VHDL code that does not use any shared pins, such as the busses (These are tristated), and just changes one pin. (A simple T-Flip-Flop behavior with divide by 65536 counter should be sufficient). All other IO pins should be tri-stated at this time. When the FPGA pin toggles, the code has downloaded to the FPGA.

#### 2.1.1. Configure FPGA using JTAG

The FPGA should be able to use JTAG for configuration. JTAG uses a serial daisy-chain interface with four connections.

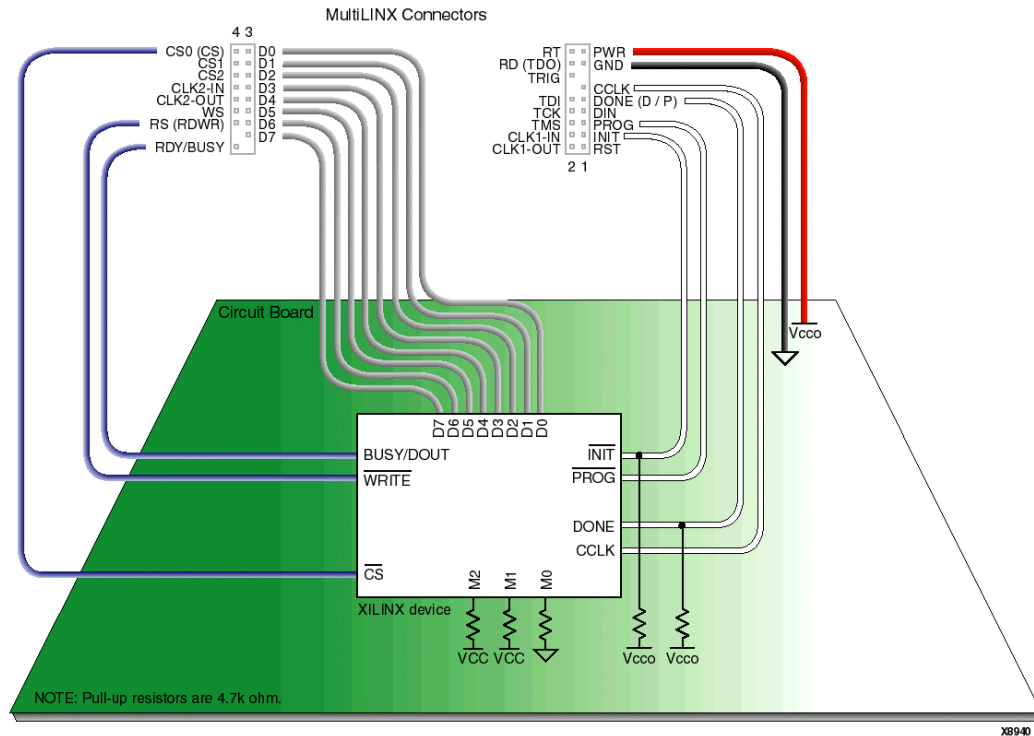
- ❑ Make sure the power supply to the Universal Controller is off.
- ❑ Tie pin 3 of JM34 to 3.3 Volts to disable the FPGA Config FLASH (U24)
- ❑ Attach Multilinx cable pins for JTAG configuration (TCK on pin JM12-11, TMS on pin JM12-13, TDI on pin JM12-9, TDO on pin JM12-3, PWR on pin JM12-2, GND on pin JM12-4). The PWR pin is connected to 2.5 volts due to power supply constraints. Even though the multilinx cable is capable of using 3.3 Volts and 5 Volts, DO NOT CONNECT THE MULTILINX CABLE TO ANY VOLTAGE SOURCE OTHER THAN 2.5 VOLTS. Otherwise, it will damage the power supplies on the Universal Controller.

- ❑ Apply the 2.5 Volt power to the Universal Controller board.
- ❑ Using ChipScope ILA software, download the configuration to the FPGA.
- ❑ Using an oscilloscope, check to see that there is a square wave on the pin defined in the VHDL file that was downloaded.
- ❑ Remove 2.5 Volt power before disconnecting the MultiLinx cable.

### 2.1.2. Configure FPGA using SelectMap

The FPGA should be able to use SelectMap for its configuration. SelectMap is a fast way to configure the FPGA and ensure that the configuration has successfully downloaded to the FPGA.

- ❑ Make sure the Universal Controller power supply is off.
- ❑ Connect the Multilinx Cable to the JM12 and JM34 connectors according to the picture below.



### 2.1.3. Retrieve Configuration using SelectMap

The configuration information can be uploaded from the FPGA to the software. This allows it to be compared to the original code to ensure that the FPGA configuration has not changed. It is useful for verification. SelectMap is also important because the configuration flash for the FPGA will use this on power-on to configure the FPGA.

#### **2.1.4. Use ChipScope and SelectMap to do simple debugging on FPGA**

In order to simplify future debugging, it is desirable to enable a trigger for the FPGA using the ChipScope ILA software. This software will enable the FPGA to implement a very simple “Logic Analyzer” based on CLBs. This will be useful for trapping faulty states. This involves embedding a trigger block into the VHDL code. Xilinx provides this block. The ChipScope software will capture the state only when this block is used.

#### **2.1.5. Configure FPGA Flash using JTAG**

The configuration flash uses only JTAG for configuration. It is important to ensure that the flash is configured and that it can configure the FPGA on startup. In order to do this, configure the FLASH using JTAG, and then remove the JTAG programmer (with the power off). Apply power to the board. When the board starts up, the FPGA will be configured. Check to see if the square wave is present on the FPGA output.

### **2.2. Download and Start DSP**

In order to configure the DSP, the FPGA must be configured. If the FPGA is not configured, and is tri-stated, the DSP will remain in the reset state via a pull down resistor until the FPGA sets the reset pin high. Anomaly 22 for the ADSP-21160 states that the FPGA must enable the DSP clock for 10mS, and then disable it, and then enable it again for the onboard PLL of the DSP to lock. This must be done using the FPGA and the DSPCLKEN signal of the FPGA. During this time, the FPGA must be running off the fiber optic transceiver clock. Prior to enabling the DSP, the DIP switches must be configured. The DSP ID should be ‘000’. EBOOT should be ‘1’, and RPBA should be ‘0’. All the BRx pins must be open, and the CLUSTER CLOCK enable must be low. To do this, set the DIP switches to the following values:

J4: ‘00000001’

J3: ‘1111’

The other two DIP switches are not used for this phase of testing.

The FPGA will release the RESET line, and the DSP will boot using JTAG. The JTAG connector must be attached to J6 in order to configure the DSP.

When the DSP is configured, it should change a flag pin. This pin should be read by the FPGA and the value should be placed on an FPGA IO pin. The DSP should toggle this pin at a frequency less than 10 KHz (Toggle once every 8000 clock cycles at minimum) to ensure that the pin is not toggling due to high-frequency ringing or clock skew problems.

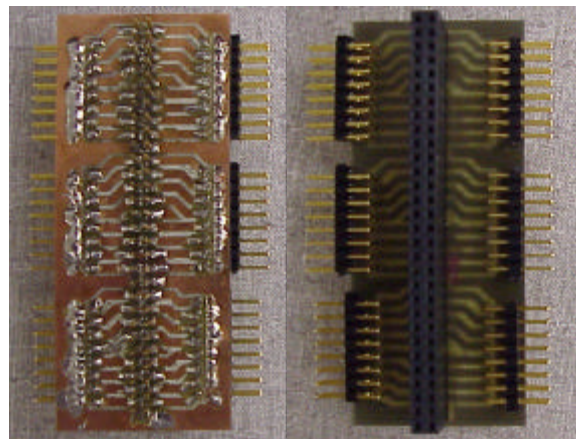
### 2.3. DIP Switches and HEX Displays

The HEX display module should be tested as a memory mapped peripheral in the DSP memory space. The APEX-ICE or SUMMIT-ICE debugger should be used to write to the DSP. The user should write several values to the hex display such as the sequence 00,01,02,04,08,10,20,40,80. They should also check the blank input by writing to the control register for the hex display. In order to do this test, the HEX display has to be written as well as the DSP interface. The control registers should also be able to be read.

Testing the DIP switches will ensure that the peripheral manager works. The dip switches can be read from the DSP memory. The same sequence should be used here as with the HEX displays to ensure that all dip switches work.

### 2.4. General Purpose Connectors

The general purpose connectors are bi-directional. They are programmable using VHDL. The VHDL is implemented as a set of registers. Depending on the application, this can be made faster. The best way to test the connectors is using a counter connected to the logic analyzer. The DSP should count up and send the values to the general purpose connector. The logic analyzer can see this via the interface board that connects the logic analyzer to the general purpose header.



Secondly, a loop back test should be performed by connecting every two IO pins together. The FPGA should 'count' from 0x0 to 0xFF...FFF, making comparisons on the loop back every time. An inconsistency will show that the pin connection is faulty. After this, the directions should be reversed, and the procedure should be repeated.

After doing this, connect the loop back, and use the DSP to read and write values from the memory location of the FPGA pins (DSP controls values of pins and checks to ensure that the loop back is successful). Upon an error, the HEX display should be set to the pin that has an error.

## **2.5. PMC (PCI Mezzanine Card)**

The PCI interface is more complicated than the previous interfaces due to several issues. Firstly, the Cypress chip needs to be verified. This can be done by reading and writing from its memory registers (regardless of their values). Secondly, the proper configuration needs to be determined for the Cypress chip, and that configuration needs to be programmed into the chip. Thirdly, there is a host board between the PMC card and the PC. The connections need to be verified. Fourthly, a small driver needs to be written for 98 or NT so that the PCI device can be addressed and used from the PC. Finally, use the driver to modify (count up) the hex display on the controller once every second. See Microsoft knowledgebase article Q152044 – HOWTO: Create a PCI Device Driver for Windows NT. The Windows NT DDK is required in order to do this. Refer to a very useful book, [The Windows NT Device Driver Book, A Guide for Programmers](#) by Baker.

## **2.6. Fiber Optic Interface**

The fiber interface can be tested using the old board and the new board. The communication can be monitored using the logic analyzer. Simply send a pattern from the old board, and have the new board do some simple manipulation on it (add 1). Command and data communications must be tested. The PESNet protocol is non-existent at this point in either controller. Once this works, the fiber optic interface works. Repeat this test for the redundant pair (FIBER2). Use the DSP to send the data out of the fiber port.

## **2.7. Digital to Analog Converter**

Using the DSP, generate a sine wave on the DAC. There are 4096 different voltage values, and the output should span all of them at a reasonable frequency. Use the DSP memory map to read back the value on the DAC. Repeat this for the other channel.

## **2.8. Flash EEPROMs**

To program the flash PROM, use the DSP to write incremental addresses. After this, read them back and subtract every two pairs. The result should be one. Power off the controller board, and power on. Repeat this test without programming it. A difference other than one indicates an error. Show this error on the FPGA HEX. This can be done for both PROMS.

## **2.9. Serial Communications Interface**

Use the old board to perform a test similar to the Fiber test described earlier.

## **2.10. Multiprocessor Bus**

The multiprocessor bus is a bus that connects two Universal Controllers together. This bus contains pins that allow several DSPs to work in parallel. It also allows for the FPGAs to communicate with each other via a simple serial communication protocol in the FPGA. The test of this involves writing code for two DSPs, using the Host Port Interface (the FPGA accessing DSP internal memory), and DMA transfers initiated by another DSP and by the FPGA.

## **2.11. FPGA Internal Functions**

These functions are things such as counters, interrupt controllers, watchdog timers, serial controllers, Virtex SelectRAM, Lower level protocol for PesNET, communication between FPGAs, Host Port Interface and FPGA status registers.

### 3. SCHEDULE

The following table summarizes the task allocation.

Task	Person	Date
2.1 Download and Start FPGA	Jerry	
2.2 Download and Start DSP	Jerry	
2.3 DIP Switches and HEX Displays	Jerry	
2.4 General Purpose Connectors	Jinghong	
2.5 PMC (PCI Mezzanine Card)	Jinghong	
2.6 Fiber Optic Interface	Jinghong	
2.7 Digital to Analog Converter	Jerry	
2.8 Flash EEPROMs	Jerry	
2.9 Serial Communications Interface	Jerry	
2.10 Multiprocessor Bus	Jerry/Jinghong	
2.11 FPGA Internal Functions	Jerry/Jinghong	