XAPP1020 (v1.0) June 01, 2009

Post-Configuration Access to SPI Flash Memory with Virtex-5 FPGAs

Author: Daniel Cherry

Summary

Virtex®-5 FPGAs support direct configuration from industry-standard Serial Peripheral Interface (SPI) flash memories. After configuration, it is possible for a user application to read and write to this memory for general purpose use. However, before the application can communicate with the SPI flash memory, it must first instantiate the STARTUP_VIRTEX5 primitive to gain access to some of the signals connected to the memory. This application note explains the techniques used in a reference design to implement the STARTUP_VIRTEX5 primitive and how to interface with an external SPI flash memory after configuration is complete.

This application note includes:

- The instantiation process for implementing the STARTUP_VIRTEX5 primitive
- A software application to demonstrate successful post-configuration use of the SPI flash memory

The reference design targets the ML505 evaluation board that includes a Numonyx (formerly STMicroelectronics) M25P32 32 Mb serial flash memory. For custom applications, it is recommended to use an SPI flash memory that is supported by the iMPACT configuration software and confirm device functionality with iMPACT before testing the reference design discussed in this application note. The iMPACT configuration software is part of the ISE® design suite.

Introduction

Normally, the SPI flash memory is used only for configuration. This document describes a reference design that demonstrates how to provide a user application with access to the SPI flash memory after it is used to configure the FPGA. This dual-purpose use requires that the FPGA pins used to interface with the SPI flash memory during configuration must also be used by the user application. However, some of these pins are dedicated for configuration and cannot be reconfigured for user I/O. For these pins, access is provided through the STARTUP_VIRTEX5 primitive. Figure 1 shows the relationship between the dedicated pins, the STARTUP_VIRTEX5 primitive, and the dual-purpose pins.

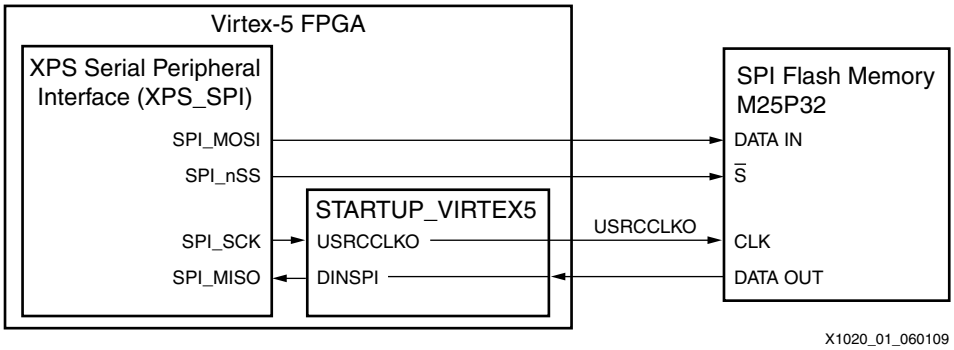


Figure 1: **SPI Interface Using STARTUP_VIRTEX5 on Dedicated FPGA Pins**

© 2009 Xilinx, Inc. XILINX, the Xilinx logo, Virtex, Spartan, ISE, and other designated brands included herein are trademarks of Xilinx in the United States and other countries. All other trademarks are the property of their respective owners.

When an SPI flash memory configures the FPGA, the STARTUP_VIRTEX5 primitive serves as the interface between the FPGA configuration logic and the data out and clock signals on the memory. To access the SPI flash memory post-configuration, the STARTUP_VIRTEX5 primitive must also be used.

The XPS SPI core used by the reference design demonstrates communication to the SPI device through the STARTUP_VIRTEX5 primitive. This helps to verify that the instantiation of the STARTUP_VIRTEX5 primitive is done correctly.

Requirements

Running the reference design requires the material described here:

Reference Design

See "Reference Design Files," page 8.

Software

The software requirements are:

- Xilinx® Platform Studio (XPS) 10.1.03
- Xilinx ISE software 10.1.03
- Microsoft HyperTerminal or equivalent terminal emulator software

Hardware

The Hardware requirements are:

- Xilinx ML505 evaluation board
- Numonyx M25P32 32 Mb serial flash memory (included on the ML505 evaluation board)
- USB cable or Parallel IV programming cable
- RS-232 9-pin straight-through cable

Reference Design Specifics

The reference design uses the Microblaze™ processor, the Microblaze processor debug module (MDM), the XPS SPI core, the XPS UART Lite core, and the XPS Interrupt Controller (XPS INTC). The IP cores are included with the EDK that is part of the ISE software.

Figure 2 shows the reference design block diagram. The address map used by the reference design is summarized in Table 1.

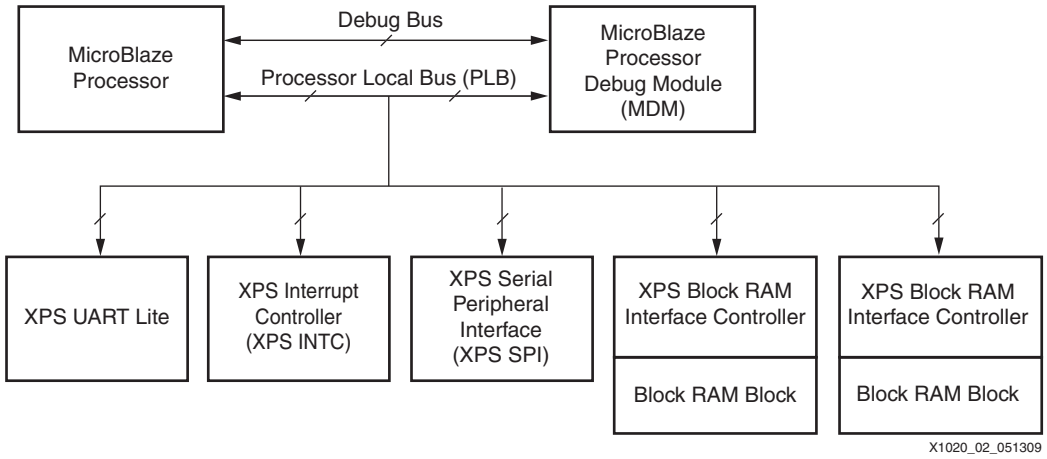


Figure 2: **Reference Design Block Diagram**

Table 1: **Reference Design Address Map**

Peripheral	Instance	Base Address	High Address
xps_uartlite (XPS UART Lite)	RS232_Uart_1	0x84000000	0x8400FFFF
mdm (MDM)	debug_module	0x84400000	0x8440FFFF
xps_intc (XPS INTC)	xps_intc_0	0x81800000	0x8180FFFF
xps_spi (XPS SPI)	xps_spi_0	0x81820000	0x8182007F

SPI Bus Overview

The SPI bus is a full-duplex, synchronous, serial data link. Devices on an SPI bus communicate using a master/slave relationship, where one device or process maintains control over the other devices attached to the SPI bus. The SPI bus is often used in FPGA-based designs to communicate between internal components because it requires fewer resources than a wider parallel bus.

The SPI bus uses four logic signals:

- Serial Clock (SCLK)
- Master Out, Slave In (MOSI/SIMO)
- Master In, Slave Out (MISO/SOMI)
- Slave Select (SS)

The STARTUP_VIRTEX5 Primitive

To interface to the SPI flash memory after configuration is complete, the appropriate pins must be connected from the STARTUP_VIRTEX5 primitive to the XPS SPI within the EDK design. The description for each of the STARTUP_VIRTEX5 ports is shown in Table 2.

Table 2: **Port Description for STARTUP_VIRTEX5 Primitive**

Port	Direction	Width	Function
EOS	Output	1	Active-High signal indicates the end of configuration
CFGCLK	Output	1	Configuration main clock output
CFGMCLK	Output	1	Configuration internal oscillator clock output
USRCCLKO	Input	1	Internal user CCLK
USRCCLKTS	Input	1	Internal user CCLK 3-state enable
USRDONEO	Input	1	Internal user DONE pin output control
USRDONETS	Input	1	User DONE 3-state enable
TCKSPI	Output	1	Internal access to the TCK configuration pin when using SPI flash memory configuration
DINSPI	Output	1	Internal access to the DIN configuration pin when using SPI flash memory configuration
GSR	Input	1	Active-High global Set/Reset signal
GTS	Input	1	Active-High global 3-State signal
CLK	Input	1	User startup clock

To read and write to a single SPI device that is attached to the configuration-dedicated pins, only the USRCCLKO and DINSPI ports are used. USRCCLKO attaches an FPGA CCLK pin that is routed to the external SPI device. DINSPI relays the data input from the SPI device when a read from the SPI device is requested. Figure 3 shows the instantiation of the STARTUP_VIRTEX5 primitive used by the reference design.

Component Instantiation

```
STARTUP_VIRTEX5_inst : STARTUP_VIRTEX5
   port map (
      CFGCLK => open,       -- Config logic clock 1-bit output
      CFGMCLK => open,      -- Config internal osc clock 1-bit output
      DINSPI => SPI_MISO,   -- DIN SPI PROM access 1-bit output
      EOS => open,          -- End of Startup 1-bit output
      TCKSPI => open,       -- TCK SPI PROM access 1-bit output
      CLK => open,          -- Clock input for start-up sequence
      GSR => '0',           -- Global Set/Reset input(GSR cannot be used for the port name)
      GTS => '0',           -- Global 3-state input (GTS cannot be used for the port name)
      USRCCLKO => SPI_SCK,  -- User CCLK 1-bit input
      USRCCLKTS => '0',     -- User CCLK 3-state, 1-bit input
      USRDONEO => open,     -- User Done 1-bit input
      USRDONETS => open     -- User Done 3-state, 1-bit input
   );
```

x1020_03_050109

Figure 3: **STARTUP_VIRTEX5 Instantiation**

Connecting the XPS_SPI to the STARTUP_VIRTEX5 Primitive

Different methods can be used to instantiate the STARTUP_VIRTEX5 primitive, but this reference design uses the ISE software project file named `projnav.ise` that is located in the `projnav` directory reference design files. The EDK project is instantiated using the `system.xmp` file that is located in the reference design root directory. The `system.xmp` file is used for interfacing with the processor peripherals. The resulting connections are shown in Figure 4.

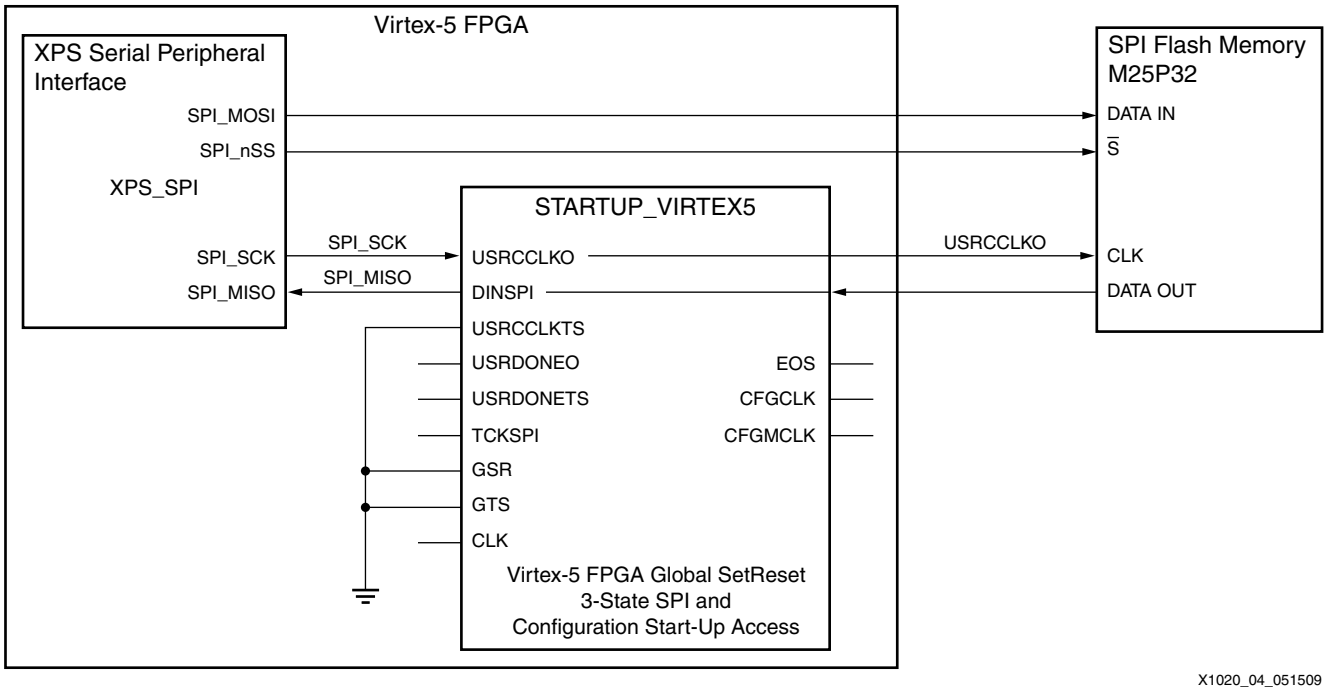


Figure 4: **SPI Signal Diagram after Instantiation**

The Software Application Summary

The software application executes automatically from the XPS block RAM memory. When the application starts, it performs some initialization tasks and then tests the SPI flash memory to verify that the connections via the startup block have been appropriately connected in the top level HDL source code. Results are output to the RS-232 serial port on the ML505 board and can be viewed using a HyperTerminal session running on a PC.

When the software application starts, it initializes the xps_spi software drivers and connects the SPI drivers to the interrupt subsystems so that interrupts are properly detected. An interrupt handler is also specified and is called whenever an SPI status occurs.

After the drivers are initialized and connected, the appropriate SPI options and slave selections are made to complete the setup for reading to and writing from the external SPI flash memory.

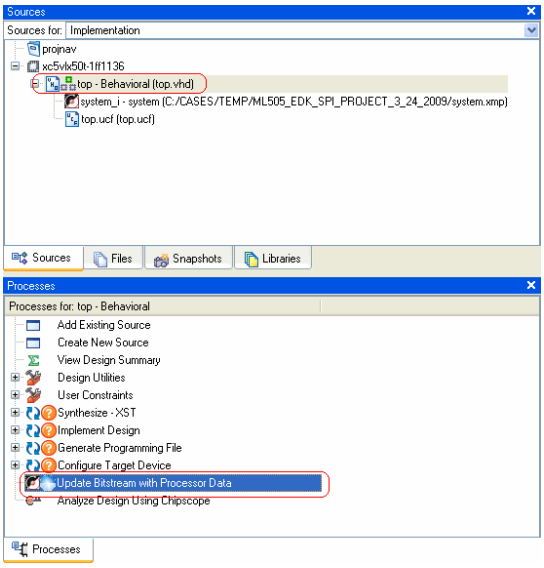
Before the application can write to a given sector in the SPI flash memory, that sector must first be erased. All sectors are erased except the beginning sectors of the memory where the bitstream is stored. Erasure allows the application to write to and then read from the contents of the memory.

After the device drivers are set up and the unused sectors are erased, the software reads to and writes from the sectors not used by the bitstream. The pass/fail results for erasing, writing, and reading the SPI flash memory are output to the RS-232 serial port for display on a PC running a terminal emulation program.

Building the Reference Design

Before executing the reference design, the bitstream is generated, the software application is compiled, and the `.mcs` file that is used to program the SPI flash memory is generated. The bitstream and reference design `.mcs` files for this system are available in the `ready_for_download/` directory under the project root directory.

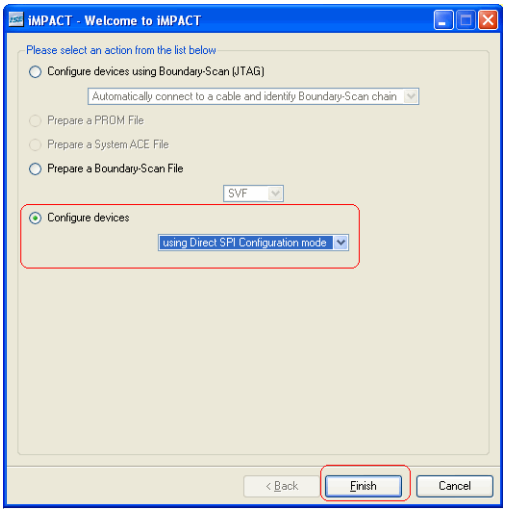
1. Use the ISE software to open the project file in the `/projnav` directory. From the Sources window, select `top.vhd` (Figure 5). In the Processes window, right click **Update Bitstream with Processor Data**. Select **Run**.



X1020_05_051509

Figure 5: **Generating the Bitstream**

2. Open iMPACT by right clicking **Configure Target Device** from the Processes window. Select **Run**.
3. Select **Configure devices** and choose **using Direct SPI Configuration mode** from the list. Click **Finish** (Figure 6). When the Add Device menu appears, select **Cancel**.



X1020_06_051509

Figure 6: **iMPACT Actions**

4. Double-click **PROM File Formatter** from the sources menu in the left hand screen. Select **3rd-Party SPI PROM** and PROM File Format **MCS**. In the Checksum Fill Value field, enter **FF**. In the Prom File Name field and Location field, enter a file name and directory path where the `.mcs` file is saved. Click **Next** (Figure 7).

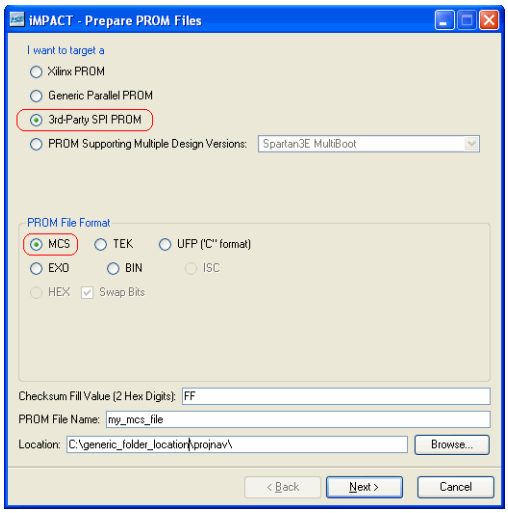


Figure 7: **PROM File Formatter Options**

5. From the Select an SPI PROM Density list, choose **32M**. Click **Next** (Figure 8). On the last page, click **Finish**.

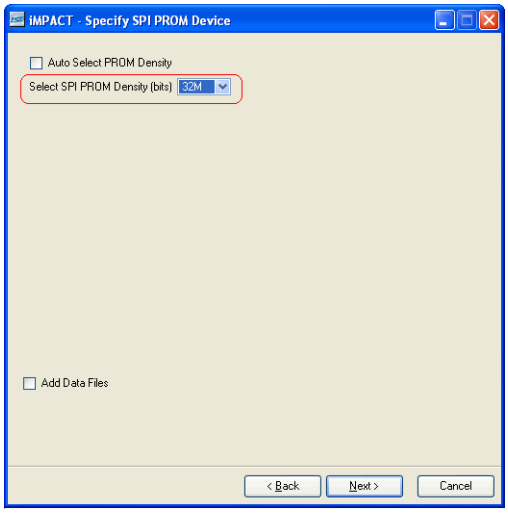


Figure 8: **PROM File Formatter Additional Options**

6. When iMPACT asks for a bitstream to add to the device chain, browse to the project directory `/projav` and select the `top_download.bit` file. No additional devices need to be added to the data stream for this reference design.
7. Right click in the white space of the PROM File Formatter window and select **Generate File**.
8. After the file is generated, connect the ML505 board, run a boundary scan, add the SPI flash memory to the Virtex-5 device, and download the `.mcs` file to the SPI flash memory.
9. On the ML505 board, set the configuration mode DIP switches to `001` for SPI configuration.
10. Connect the PC to the RS-232 serial port on the ML505 board.
11. On a PC, run HyperTerminal or a similar terminal emulation program.
12. Set the terminal to 9600 baud, 8 data bits, no parity, and no flow control. Figure 9 shows these settings in the HyperTerminal Properties window.

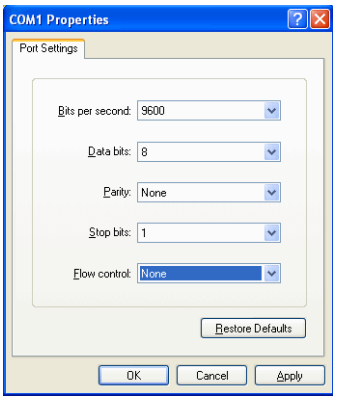


Figure 9: **HyperTerminal Settings**

Executing the Reference Design

To execute the reference design using the pre-build bitstream and the compiled software application, use the files inside the `ready_for_download/` directory in the project root directory and follow these steps:

1. Change to the `ready_for_download/` directory.
2. Use the iMPACT software to download the bitstream. Use the command: **`-batch ready_for_download.cmd`**
3. Verify that the configuration mode DIP switches on the ML505 Evaluation Board are set to `100`.
4. Power off the board and power it back on. After the bit file downloads, the software application executes out of block RAM, and HyperTerminal displays the results. Figure 10 shows the results if the communication with the SPI flash memory is successful.

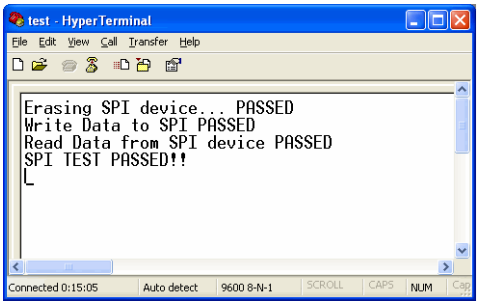


Figure 10: **HyperTerminal Output**

Conclusion

This application note describes how to implement a STARTUP_VIRTEX5 primitive to provide an interface to an external SPI flash memory post-configuration. The reference design targets the Virtex-5 FPGA on the ML505 evaluation board and includes a design for the ISE software and an EDK design that demonstrates reads and writes to the SPI flash memory located on the ML505 evaluation board.

Reference Design Files

The reference design files can be downloaded from:

https://secure.xilinx.com/webreg/clickthrough.do?cid=131484

The reference design matrix is shown in Table 3.

Table 3: **Reference Design Matrix**

Parameter	Description
Developer Name	Xilinx
Target Devices (stepping level, ES, production, speed grades)	Virtex-5 FPGAs
Source Code Provided	Yes
Source Code Format	VHDL
Design Uses Code/IP from an Existing Reference Design/Application Note, Third Party, or CORE Generator™ software	Yes
Simulation	
Functional Simulation Performed	No
Timing Simulation Performed	No
Testbench Used for Functional Simulations Provided	No
Testbench Format	
Simulator Software Used/Version (e.g., ISE software, Mentor, Cadence, other)	ISE software
SPICE/IBIS Simulations	No
Implementation	
Synthesis Software Tools Used/Version	XST version 10.1, service pack 3
Implementation Software Tools Used/Versions	ISE and EDK version 10.1, service pack 3
Static Timing Analysis Performed	Yes
Hardware Verification	
Hardware Verified	Yes
Hardware Platform Used for Verification	Xilinx ML505 evaluation board

Additional Resources

The following resources provide additional information about the subject covered in this application note:

1. DS444, *Block RAM (BRAM) Block, Data Sheet*.
2. DS570, *XPS Serial Peripheral Interface (SPI), Data Sheet*.
3. DS571, *XPS UART Lite, Data Sheet*.
4. DS572, *XPS Interrupt Controller, Data Sheet*.
5. DS596, *XPS Block RAM (BRAM) Interface Controller, Data Sheet*.
6. DS641, *MicroBlaze Debug Module (MDM), Data Sheet.*
7. UG081, *MicroBlaze Processor Reference Guide*.
8. UG191, Virtex-5 *FPGA Configuration User Guide.*
9. *Virtex-5 Libraries Guide for HDL Designs,* www.xilinx.com/itp/xilinx10/books/docs/virtex5_hdl/virtex5_hdl.pdf
10. *Xilinx Processor IP Library Device Driver API*, located in the EDK installation at $XILINX_EDK\doc\usenglish\xilinx_drivers_api_toc.htm.

Revision History

The following table shows the revision history for this document:

Date	Version	Description of Revisions
06/01/09	1.0	Initial Xilinx release.

Notice of Disclaimer

Xilinx is disclosing this Application Note to you “AS-IS” with no warranty of any kind. This Application Note is one possible implementation of this feature, application, or standard, and is subject to change without further notice from Xilinx. You are responsible for obtaining any rights you may require in connection with your use or implementation of this Application Note. XILINX MAKES NO REPRESENTATIONS OR WARRANTIES, WHETHER EXPRESS OR IMPLIED, STATUTORY OR OTHERWISE, INCLUDING, WITHOUT LIMITATION, IMPLIED WARRANTIES OF MERCHANTABILITY, NONINFRINGEMENT, OR FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT WILL XILINX BE LIABLE FOR ANY LOSS OF DATA, LOST PROFITS, OR FOR ANY SPECIAL, INCIDENTAL, CONSEQUENTIAL, OR INDIRECT DAMAGES ARISING FROM YOUR USE OF THIS APPLICATION NOTE.