The QF^2-pre Guide

Dr. John A. Jones

Questions / Corrections / Requests? - Email me: **qf2-pre@jxj.me.uk**

Spartan-6 Bootloader Firmware

SHA256: XXXXXXXXX
Build date: XXXXXXXX

Spartan-6 Runtime Firmware

SHA256: XXXXXXXXX
Build date: XXXXXXXX

THERE IS NO GUARANTEE THAT ANY OTHER FIRMWARE VERSION THAN THE ABOVE WILL MATCH THIS DOCUMENT.

If you are using a firmware listed in the revision history below, the python tools will identify the firmware loaded in the QF^2-pre over the network, and will attempt to provide consistent behavior for the end user no matter which version is in use. It is possible that features may not be available in different firmware / tool revisions, so you may need to update the firmware and / or software to a particular revision to use a feature. If you attempt to use a feature that is unavailable, the Python tools will throw an exception.

If in doubt, you can use *update_version.py* to get the latest tools, firmware and examples. It will overwrite conflicting files in the local directory, so if a fresh copy of the tools is desired, it should be executed from an empty directory. The script can be found at:

http://ADDRESSTOADD

Items highlighted in yellow like this are future features are currently in active development / test.

Revision History

2017-8-1

Bootloader SHA256:
Bootloader build date:
Runtime SHA256:
Runtime build date:

Comments: Initial version.
Bootloader changes: None.
Runtime changes: None.
Tool changes: None.
Example changes: None.

Known Issues / Limitations

- FMC sites are 'swapped', and therefore incompatible with some double FMCs (i.e. the HPC FMC should be the top site, and the LPC the bottom). This was a known consequence of the Kintex-7 FPGA banking.

- Spartan-6 fiber ethernet interface is tested, but is not yet integrated with the firmware.

- Some virtual channels between the Spartan-6 and Kintex-7 are not yet implemented.

- Microphone interface not yet implemented.

1 – Hardware

1.1 - Overview

The QF^2-pre is a double FMC carrier. It has one HPC mezzanine site, and one LPC site. The total FMC I/O count is 228 lines. Both are fully allocated aside from the serial links, which are unused. There are two FPGAs on the board:

- A Spartan-6 LX45T is used to provide the core functions of the board. It implements a boot system, ethernet communications (ARP, PING, UDP & IP MULTICAST), power management and power supply clock phasing and board monitoring. It has a non-volatile configuration space in its PROM, and maintains several Kintex-7 firmwares which can be loaded automatically at boot. The Spartan-6 also manages the FMC and QSFP I^{2}C interfaces and the FMC JTAG interfaces and dedicated clocks.

- A Kintex-7 160T / 325T is used as an application FPGA. It is mostly dedicated to interfacing to the FMCs and MFCs.

The two FPGAs are connected by a bidirectional LVDS link operating at cloase to a 1Gb/s line rate. The separation of functionality between the FPGAs significantly reduces the overhead in the Kintex-7 firmware as e.g. network interfaces are already implemented in the Spartan-6, and ensures the board is always up and cannot be 'bricked'.

The board has a modular power system, with independent supplies for the Spartan-6, Kintex-7 and FMCs, which allows the board to power on and off in stages while maintaining communication with a host over ethernet. The FMC VADJ supply voltages are digitally programmable, but are currently locked in the firmware at +1.8V for the HPC FMC (Kintex-7 HP I/O) and +2.5V for the LPC FMC (Kintex-7 HR I/O), supporting LVDS by default.

1.2 – Design Philosophy

FPGA technology (and electronics in general) have become a great deal more complicated over the past decade. This puts many of the device features outside the expertise of the layperson, or many more experienced designers. This board was in part designed to combat these issues while still providing high performance at a reasonable price point.

The governing philosophy of its design is:

- **Ensure mistakes are in-system recoverable:** The Spartan-6 firmware has a bootloader that is optionally write-protected in hardware, a highly recommended practice for inaccessible systems. As such, even a complete corruption can be recovered from by power-cycling to return to the bootloader.

- **Provide communications and diagnostic interfaces to support development and testing:** Often when we develop a new application firmwares, a lot of effort is expended developing as stable communication interface to the board. Further, it can be a challenge to identify whether faults are at the hardware-level or in the new firmware. To avoid this, many diagnostic features are implemented in the Spartan-6 FPGA. Their output (which has been field-tested and verified) can be cross-checked against the user’s application firmware to identify issues during

development. In addition, a Kintex-7 demonstration firmware and software framework is provided to help aid in identifying hardware faults.

- **Make it as easy as possible for a user to access the powerful features of the hardware without writing doing everything from scratch:** Many of the QF^2-pre features such as board monitoring, power supply phasing, remote programming and debug, over temperature shutdown, protected bootloader, etc... are available via a simple python interface. They are managed by the Spartan-6 FPGA, so you don't have to write any firmware to use them, and you aren't at risk of breaking them when you develop your application firmware.

- **Conversely, allow for power users to access the raw data:** Perhaps you don't want to use the features in the Spartan-6 firmware, or want to do something slightly different to the typical user such as implementing network protocols that the Spartan-6 doesn't support. This can be achieved by using bypass controls built into the Spartan-6 runtime that will forward the 'raw' data for different features to the Kintex-7.

If there is something you think this board isn't capable of but should be, we'd be interested in hearing about it – hopefully it's something that can be incorporated into later revisions of firmware (or if necessary, hardware) for the benefit of all. The firmware and hardware design are derived from experiences in many projects and progressively improved, so that every current and future user can benefit. Backward-compatibility is maintained whenever possible, or an equivalent feature provided otherwise.

The core Spartan-6 fimware is currently closed-source. The main reason for this is to maintain the critical infrastructure of the board with a clean development and test cycle. The Spartan-6 firmware is non-trivial, and it is important for all users that any issues can be easily recognized and addressed. The Spartan-6 firmware contains several diagnostic features which aid in the recovery of malfunctioning hardware and understanding what went 'wrong'. If multiple firmware variants written by users were running in the field, it would be impossible for us to provide effective technical support.

1.2 - Hardware Feature Summary

1.2.1 – Application Resources

- Kintex-7 160T / 325T FPGA.
- Eight 10.3Gb/s serial links to application FPGA via QSFP+.
- External RF 50-ohm GTX transceiver clock input via U.FL connector.
- Dedicated 125MHz MEMS oscillator for Kintex-7 serial links.
- One fully populated (except serial links) HPC FMC with programmable VADJ (typically +1.8V).
- One fully populated (except serial links) LPC FMC with programmable VADJ (typically +2.5V).
- Two Hirose FH29 ZIF connectors connected to Kintex-7 HR banks with VCCO @ +2.5V.
- Two orange LEDs.
- Independent main power system.

1.2.2 – Shared Resources

- Two reconfigurable SI570 oscillators for serial links (shared to both Kintex-7 and Spartan-6 FPGAs). Default frequency is 156.25MHz.

1.2.3 – System Resources

- Supply voltage +6V - +14.5V.
- Spartan-6 LX45T system FPGA.
- Four 3.2Gb/s links to system FPGA via QSFP.
- Dedicated 125MHz oscillator for Spartan-6 serial links.
- Gigabit Ethernet over copper.
- Recovery, diagnostic and user-defined push buttons with jumper bypass for chassis switches.
- uSD card support.
- Supply voltage, supply current, ambient humidity, barometric pressure and magnetic field monitoring.
- Board and device temperature montoring, with automatic thermal overload protection for Kintex-7 FPGA, independent of internal XADC.
- Atmel cryptographic security chip (key authentication, certificates, unique ID, TRNG, OTP memory).
- 256Mb boot PROM with write-protected bootloader.
- 9-axis IMU.
- PDM microphone.
- Speaker and headphone socket.
- ATX fan header with tach & PWM support.
- Four tri-colour LEDs.

1.3 - Spartan-6 Protected Bootloader & Runtime Firmwares

There are two versions of the Spartan-6 firmware: a protected bootloader firmware and a runtime firmware.

The bootloader is a simplified Spartan-6 firmware that resides in the first 32 x 64kB sectors of the on-board PROM. It always loads before anything else and contains the following components:

- Ethernet support with non-volatile network settings stored in the PROM.
- A system I^2C controller that ensures the board is in a known state with main power turned off, and provides monitoring information.
- A programming interface for the on-board PROM.
- An audio diagnostic output.
- An ICAP reboot controller that can trigger a reload of the Spartan-6 firmware.

The firmware is complete enough to allow reconfiguration of the runtime image of the Spartan-6 and to trigger reloading into the image. As such, it can provide a protected fail safe in the event of a corrupted runtime firmware. The PROM settings for the bootloader and runtime are independent from each other.

For normal operation, the board can be started either by loading the bootloader and waiting for a user command via ethernet to load the runtime firmware, or by automatically jumping to the runtime firmware immediately. The mode in use is determined by the bootloader register settings.

NOTE: In situations where the recovery mode button is typically inaccessible, ***it is highly recommended that the bootloader is configured to await an external command before entering into the runtime firmware and that the lower 32 sectors of the PROM are write-protected***. This will allow the board to be recovered by power-cycling in the event of a corrupted runtime firmware

or configuration, as the bootloader network settings and firmware are completely independent of the runtime.

NOTE: By default, once the write-protect for the PROM has been enabled in software, ***it cannot be disabled again without physically applying a jumper to the header next to the Spartan-6.***

Irrespective of the mode the bootloader is configured in, it can be also be forced into a recovery mode using the bootloader firmware by holding down the recovery mode button before power is applied to the board, or similarly using an external switch attached to the appropriate jumper (for cases where e.g. the board is mounted in a chassis or case). This will force the bootloader to use it's default register and network settings and disable automatic reboot into the runtime firmware. This is further documented in the recovery mode section below.

The runtime firmware contains all the additional features documented below in addition to those of the bootloader.

1.4 - Copper and Fiber Ethernet Support

The current Spartan-6 firmware supports copper ethernet operating at 1Gb/s using a Micrel KSZ9031RNX RGMII PHY. The firmware implementation is gigabit-only and does not work at 10Mb/s or 100Mb/s interface speeds.

Internally, the Spartan-6 firmware supports 4-channel fiber ethernet through the Spartan-6, and implements a basic ethernet switch with limited arp caching between the 5 ports. The Spartan-6 acts as an additional internal endpoint on the switch. Other devices can therefore be chained without additional external switches, or additional connections can be used for redundancy.

The Spartan-6 ethernet interface has a UDP core which eliminates some of the network firmware infrastructure required in the Kintex-7. It supports:

- ARP
- ICMP PING.
- A single multicast UDP channel that is forwarded to the Kintex-7 FPGA. It does not support registration, which must be statically configured in the attached network.
- UDP echo on port 7.
- Board identification on UDP port 50000.
- Spartan-6 configuration space on UDP port 50001.
- Kintex-7 / FMC JTAG & PROM SPI sequencer (25MHz default speed) on port 50002.
- Kintex-7 UDP port forwarding over ports 50008, 50009, 50010, 50011.

All data channels operate by sending / receiving UDP packets to / from these ports. DHCP and TCP are not currently supported.

A channel multiplexer in the Spartan-6 separates the streams and records the source UDP port and source IP for each channel separately, and sends subsequent responses to that target after receiving each request. Therefore, ***each individual channel can can have an independent master, but a single channel must only have one.*** The architecture is designed so that most single-packet transactions are atomic. Therefore a board is recoverable even in the event two masters collide.

Keeping the channels separate allows one to for example run a Xilinx Chipscope interface to the

Kintex-7 on port 50002 using the included Xilinx Virtual Cable interface script, while operating the board normally using the other channels.

The PROM / JTAG interface is stable, and is forward and backward compatible with past and future Spartan-6 firmwares.

To conserve power, serial link channels can be turned on and off depending on the application by changing the Spartan-6 configuration settings.

1.5 - Board LEDs

The board has six standard LEDs. Two are located near the ZIF sockets near the top of the board and are connected to the Kintex-7. The other four are connected to the Spartan-6 and are located at the four corners of the board. The Kintex-7 LEDs emit orange light, and the Spartan-6 LEDs are tri-color and can therefore produce colors across the visible spectrum.

1.5.1 Kintex-7 LED Interpretation

Kintex-7 LED usage is application-specific. They emit orange light.

1.5.2 Spartan-6 Bootloader LED Interpretation

The default interpretation of the Spartan-6 LEDs is shown below. It is possible to transfer control of these LEDs to the Kintex-7 FPGA using a virtual communication channel, as described later in this document.

Top-left (by headphone socket):

Red: PROM configuration checksum failure - using default firmware and default network settings.
Blue: Recovery mode - using default firmware and default network settings.
Alternate flashing blue / green : Hybrid recovery mode - using default firmware settings with PROM network settings.
Green: PROM configuration checksum pass - using PROM settings.

Bottom-left (by ethernet socket):

Flashing red: Spartan-6 programmed.
Blue: UDP ethernet transaction in progress.

Top-right (by top FMC):

Pink: Indicates bootloader firmware.

Bottom-right (by bottom FMC):

Pink: Indicates bootloader firmware.

1.5.2 Spartan-6 Runtime LED Interpretation

Top-left (by headphone connector):

Red: PROM configuration checksum failure - using default firmware and default network settings.
Green: PROM configuration checksum pass - using PROM settings.

Bottom-left (by ethernet connector):

Flashing green: Spartan-6 programmed.
Flashing blue: UDP ethernet transaction in progress.

Top-right (by top FMC):

Unused.

Bottom-right (by bottom FMC):

Unused.

1.6 - High-Performance Clocking

There are four different high-performance clock sources on the board. These clocks are solely for the purpose of driving the serial links on the Kintex-7 and Spartan-6. Usage of an appropriate source depends on a user's application. The four sources are:

Local oscillator (SiTime MEMS oscillator): This can be populated by either a SIT912X one-time programmable oscillator (+3.3V, LVDS), or a SiTime DCXO. The Spartan-6 and Kintex-7 each have a separate oscillator site which is by default fitted with a 125MHz SIT9121 (alternative values can be fitted on request).

Pros: Easy to use, low power.
Cons: Frequency is one-time programmable.

External clock: A dedicated 50-ohm RF clock can be provided directly to the Kintex-7 via a U.FL connector.

Pros: Simple to use, useful for test and characterization, can be used to synchronize to a global system clock.
Cons: Requires a high-quality reference to be provided off-board.

SI57X programmable oscillators / VCXO: These generate programmable frequency references which can also be manipulated dynamically over I^2C at up to 10kHz for frequency tracking applications. They are configured over I^2C via the Spartan-6. The initial settings are part of the Spartan-6 configuration data and the devices can be automatically enabled at startup.

Pros: Programmable frequency, can be used to track a reference.
Cons: Relatively complex to configure and operate.

1.7 – Board Health Monitoring

The QF^2-pre provides many options for monitoring the board health of the system. It monitors all the supply voltages on the board, and the current consumption of all the primary switching regulators. It also has an ATX fan header which is capable of PWM modulation and fan speed (tachometer) measurement. The humidity sensor can be used to detect the presence of water in the electronics environment from e.g. a faulty cooling system. Barometric pressure sensing, tri-axis magnetic field sensing and a 9-axis IMU are also included. Volatile chemical sensing is intended to be introduced as well once the appropriate part become available.

All the available board monitoring devices are polled by the Spartan-6 (in both bootloader and runtime firmwares) at five second intervals. This interval cannot be changed by the user and represents a compromise between responding to events promptly and excessive polling of the system I^{2}C bus.

The monitoring data can be read back from the Spartan-6 configuration space, and can also be forwarded to the Kintex-7 FPGA over its own virtual communication channel should the user want direct access to it in their application firmware.

1.8 – Over Temperature Shutdown

In order to protect the hardware from excessive temperatures (and in particular the Kintex-7 FPGA), the Spartan-6 runtime firmware features an always-on over-temperature shutdown feature which cannot be disabled. The Spartan-6 will turn off all the main power regulators on the board if the temperature of the Kintex-7 FPGA exceeds a fixed threshold of 80C (i.e. 5C less than the recommended maximum temperature of a commercial grade part).

In addition to this, it is ***highly recommended*** that the internal over-temperature shutdown feature of the Kintex-7 FPGA is enabled to provide an extra layer of safety.

NOTE: The example application firmware for the Kintex-7 also demonstrates emergency shutdown notification from the Kintex-7 to the Spartan-6.

NOTE: The Spartan-6 can be configured to also shut down main power on other monitoring events, such as out-of-bounds supply voltages.

1.9 – Diagnostic Audio Output

For situations where the network settings of a board are unknown and recovery mode is undesirable, or diagnostic information is desired but a computer is unavailable, the diagnostic mode button can be used. Pressing it will trigger an audio output on the on-board speaker, or via a pair of headphones if they are plugged in, providing information about the current board status.

Currently, the audio output provides information on the current IP and MAC address of the board. This feature set will be expanded at a later date.

1.10 – FMC and QSFP I^{2}C Access

The I^{2}C interfaces for the FMC sites and QSFPs are shared with the system I^{2}C bus via a multiplexer. There are two ways they can be accessed:

- A diagnostic interface in the Spartan-6 which is accessible using the example Python code below.
- A control channel which can be enabled in the Spartan-6 settings, which allows the Kintex-7 to directly drive those interfaces.

NOTE: FMC and QSFP access from the Kintex-7 is demonstrated in the example Kintex-7 firmware.

2.0 – Usage

2.1 – Turning the Board on for the First Time

If you are receiving a new board, you should find that the PROM is already programmed with firmware and settings that match the firmware defaults.

To begin with one should apply power to the board either using a supply jack (center positive) or the header terminals.

Once power is applied, the board LEDs should flash orange by the Ethernet connector (indicating the bootloader is running on the Spartan-6) and the LED by the headphone jack should be green, indicating the board is using the PROM configuration settings. Other behaviors can happen if the board has already been user-configured, so if you see something else, consult the LED section.

The board should be connected to a network-capable computer with a static IPv4 address in the ***192.168.1.X*** subnet, ***initially via the copper interface as the optical interface is disabled by default***. Communicating with the board over Ethernet involves targeting the IP address ***192.168.1.127***. The board's MAC address is set to ***AA:BB:CC:DD:EE:FF*** by default. A simple test would be to ping the board using that IP to ensure communication is established before using other scripts / tools.

NOTE: It is recommended that a ***minimum*** of a 15W power supply is used, as the use of all the board features (in particular the serial links and QSFPs) will demand too much from lesser supplies and may create undesired / unexpected behavior. The supply voltage is user-defined between +6V and +14.5V, but be aware that the FMC standard expects a +12V rail, so the user should check that any FMCs are compatible with non-standard voltages before use. Lower supply voltages will increase the overall efficiency of the board power system and lower overall heat dissipation.

2.2 – Recovery Mode

There are two ways a board will definitely start up with default settings:

- If the PROM settings are corrupted or not set the bootloader will fail to detect a valid checksum (verified using the Fletcher-16 algorithm) and will use its default settings instead.
- The user presses the recovery button.

There are several ways of using the recovery button:

- **Holding the button down before and during power-up:** This will load the bootloader firmware and force the Spartan-6 bootloader to use its default settings, which will prevent it from automatically launching the runtime firmware. It will also reset the IP address to 192.168.1.127 and MAC address to AA:BB:CC:DD:EE:FF.

- **Depressing for fewer than approximately three seconds, then releasing the recovery button:** This has no effect in the runtime firmware, but will alternate between using the PROM network settings and the firmware defaults while the bootloader firmware is running.

- **Depressing for more than approximately three seconds, then releasing the recovery button:** This will trigger a reboot of the Spartan-6 into the bootloader firmware irrespective of

which firmware is currently running in the Spartan-6. The PROM settings will be bypassed and the firmware defaults used instead. Automatic restart into the runtime firmware is disabled.

NOTE: Depressing the recovery button while the bootloader firmware is loading load stops the bootloader firmware from automatically transitioning back to the runtime firmware.

NOTE: The Ethernet connections will drop out for several seconds whenever the Spartan-6 is reloaded.

NOTE: It is easy to check which mode the board is in by inspecting the LEDs.

2.3 – Spartan-6 Board Identification

Communication with port 50000 can be used to identify information about the firmware running in the board. It can also be used to trigger a Spartan-6 firmware reload.

The interface is used by sending a single byte message to the port, with the following meanings:

0x01: Restart the Spartan-6 using the bootloader firmware (no response packet will be sent).
0x81: Restart the Spartan-6 using the runtime firmware (no response packet will be sent).
0x00: Query the board identification information.

A query message will lead to the following response packet:

Byte 0: Identification format (currently zero).
Byte 1: Board identification (1 == QF^2-pre).
Byte 2: Active Spartan-6 firmware (0 == bootloader, 1 == runtime).
Bytes 3 – 6: Bootloader firmware build date.
Bytes 7 – 38: Bootloader firmware SHA256.
Bytes 39 – 44: Runtime firmware build date.
Bytes 45 – 76: Runtime firmware SHA256.

2.4 – Spartan-6 Configuration

The Spartan-6 configuration bus uses a simple 'write all registers + mask' process. While there are more complicated approaches, it's simple, synthesizes cleanly and is easy to understand. To write data values to the Spartan-6 registers, one sizes two blocks of bytes the size of the entire register space, then sets the values in one and the corresponding mask for the bits to change in the other. Sending the packet to 50001 will trigger a response packet containing the entire read-back register space. In order to read the current board status, one sends a packet with the mask set to all zeros.

The structure of the Spartan-6 register write space is as follows:

The structure of the Spartan-6 register read space is as follows:

NOTE: The register space stored in the PROM has the exact same format as the register space

accessible via port 50001, with the addition of a network settings block which can only be changed by PROM configuration.

3.0 – Software and Firmware

3.1 – Getting the Latest Version

update_version.py can be used to get the latest software and firmware. This downloads a complete new copy of the QF^2 software and firmware from an online server into the local directory. It contains a complete history of the firmware revisions, tools and examples.

NOTE: This ***will*** overwrite conflicting files in the local directory, so if a fresh copy of the tools is desired without deleting a previous copy, it should be executed from an empty directory.

3.2 - Python Tools

The Python tools provide a comprehensive set of interface scripts for programming and configuring and the board.

store_spartan_6_image.py

Stores a new Spartan-6 firmware image in the PROM. Also embeds an SHA256 hash of the bitstream and a storage and build date stamp after the bitstream itself. The new image can be activated by power cycling the board, using the recovery mode button, or a software reboot via port 50000.

Arguments:

-t [TARGET]: IP address of board to communicated with (default == 192.168.1.127).
-b [BITFILE]: Name of bitfile to download into PROM.
-l: Unlock PROM space before write, and lock again afterwards (default == disabled).
-r: Reboot the board automatically and verify restart after programming (default == no reboot).
-X: Program the bootloader image (default == program the runtime image).

NOTE: The hash of the bitstream is calculated from just the bitstream itself, ***not*** the entire bitfile. They should not be assumed to be equal.

verify_spartan_6_image.py

Verifies the current Spartan-6 firmware image in the PROM.

Arguments:

-t [TARGET]: IP address of board to communicated with (default == 192.168.1.127).
-b [BITFILE]: Name of bitfile to compare with PROM.
-X: Verify the bootloader image (default == verify the runtime image).

store_spartan_6_configuration.py

Stores a new non-volatile Spartan-6 configuration in the PROM.

Arguments:

-t [TARGET]: IP address of board to communicated with (default == 192.168.1.127).
-i [IP]: New IP address for board (default == current value unless PROM is corrupt, in which case 192.168.1.127).
-m [MAC]: New MAC address for board (default == current value unless PROM is corrupt, in which case AA:BB:CC:DD:EE:FF).
-s [HASH]: SHA256 of Kintex-7 bitstream to load from PROM on boot (default == current value unless PROM is corrupt, in which case NULL).
-X: Store the bootloader configuration (default == store the runtime configuration).
-r: Reboot the board automatically and verify restart after programming (default == no reboot).

NOTE: The new settings will not take effect without a reboot.
NOTE: If you use '-r' and have changed the network settings, the restart verification may fail if you have e.g. changed the board to a different subnet.

verify_spartan_6_configuration.py

Downloads the current Spartan-6 configuration, verifies checksum, and displays current settings.

Arguments:

-t [TARGET]: IP address of board to communicated with (default == 192.168.1.127).
-b [BITFILE]: Name of bitfile to compare with PROM.
-X: Verify the bootloader configuration (default == verify the runtime configuration).

add_kintex_7_image.py

Programs a Kintex-7 image into the PROM. Adds an entry to the Kintex-7 image table after verification.

Arguments:

-t [TARGET]: IP address of board to communicated with (default == 192.168.1.127).
-b [BITFILE]: Name of bitfile to download into PROM.

erase_kintex_7_image.py

Erases a Kintex-7 image from the PROM. It does this by removing the entry for the image from the image table.

Arguments:

-t [TARGET]: IP address of board to communicated with (default == 192.168.1.127).
-s [HASH]: SHA256 identifier hash for the bitstream.

erase_kintex_7_image_table.py

Erases the entire Kintex-7 image table.

Arguments:

-t [TARGET]: IP address of board to communicated with (default == 192.168.1.127).

list_kintex_7_images.py

Displays all the Kintex-7 images in the PROM.

Arguments:

-t [TARGET]: IP address of board to communicated with (default == 192.168.1.127).

program_kintex_7.py

Streams a new Kintex-7 firmware into the FPGA over ethernet. Bypasses PROM.

Arguments:

-t [TARGET]: IP address of board to communicated with (default == 192.168.1.127).
-b [BITFILE]: Name of bitfile to load into Kintex-7 FPGA.

xilinx_virtual_cable.py

Implements a Xilinx Virtual Cable (XVC) interface that can be used by the Xilinx tools to interface with the Kintex-7 FPGA over ethernet without using a separate programming cable.

In Vivado, connection to this service can be done by running the following commands in the Vivado Hardware Manager Tcl console:

connect_hw_server
open_hw_target -xvc_url 127.0.0.1:2542

NOTE: If you have trouble getting this to work, make sure there is no hardware USB / JTAG pod connected to the computer running the tools and script. Vivado can become confused if multiple interfaces are available or multiple hardware servers are running.

NOTE: The other scripts listed here should not be used at the same time as this one.

Arguments:

-t [TARGET]: IP address of board to communicated with (default == 192.168.1.127).

3.3 – Python Examples

The examples in this section are capable of interrogating the firmware running in the board, and

will adapt accordingly if compatible. If not, the software will throw an exception indicating an incompatible firmware.

read_fmc_prom.py

Example script demonstrating read / write of FMC PROMs over I^2C.

Arguments:

-t [TARGET]: IP address of board to communicated with (default == 192.168.1.127).

read_monitors.py

Demonstrates reading the board monitors on the board in a continuous loop.

Arguments:

-t [TARGET]: IP address of board to communicated with (default == 192.168.1.127).

udp_loopback_test.py

Sends packets to the Spartan-6 UDP echo port and verifies the echo packet matches the one that was sent.

Arguments:

-t [TARGET]: IP address of board to communicated with (default == 192.168.1.127).

3.4 – Kintex-7 Firmware

Example firmware modules for the Kintex-7 are described below. They can be used as a starting point for application development, and are provided ‘as-is’ without any guarantee of behavior. They are designed to be compatible with the latest version of the Spartan-6 runtime.

Appendix A – Recovering a Corrupted Bootloader

In the rare event that the bootloader is corrupted and you cannot communicate with the Spartan-6, the board has a standard Xilinx JTAG connector that is directly connected to the FPGA. The procedure to recover is:

- Reprogram the Spartan-6 using the bootloader firmware image using a JTAG programming cable attached to the header on the board (near the Ethernet).
- Verify the LEDs operate as expected – it should be noted that the firmware will still look at the PROM configuration settings upon startup, so you may wish to hold the recovery button down while programming the FPGA to force the firmware to use the default settings.
- Use the Python tools to reload the bootloader image into the PROM and the corresponding bootloader network settings. It is recommended to also verify the runtime image and configuration are as expected.

NOTE: Do ***NOT*** use the Xilinx tools to program the PROM. ***They are extremely primitive, do not understand the data structure and will cause further corruption***.