XAPP632 (v1.0) May 13, 2002

Programming an FPGA via E-mail

Author: Marc Defossez

Summary

This application note describes the process to program and reprogram an FPGA through an intranet or Internet connection. Many variations to this theme are possible. The example application is a theoretical external controller with the intent on making readers aware of the capabilities. Modules of the example applications have been hardware and software tested; this application note is the sum of all tested design parts.

Introduction

The most commonly used application on the Internet or intranet is sending and receiving E-mail. The ability to use this same application to program or reprogram an FPGA is an advantage when updating and servicing applications using FPGAs. Different solutions are discussed in this application note. Some solutions need an external Internet/intranet aware microprocessor. In the reference design example the SX52BD processor from Ubicom is used, but other components, IP2022, or manufacturers, as Rabbit, Lantronix or Zilog, can also be used.

Internet Protocol Stack

Communication over the Internet/intranet is standardized. These standards are Request For Change (RFC) documents and can be retrieved at **http://www.ietf.org**. The Internet/intranet working mechanism is organized as a protocol stack (Figure 1). A bundled stack is a set of independent programs, called layers, allowing information to flow from the physical interface (physical layer) to the user application (application layer) and vice versa.

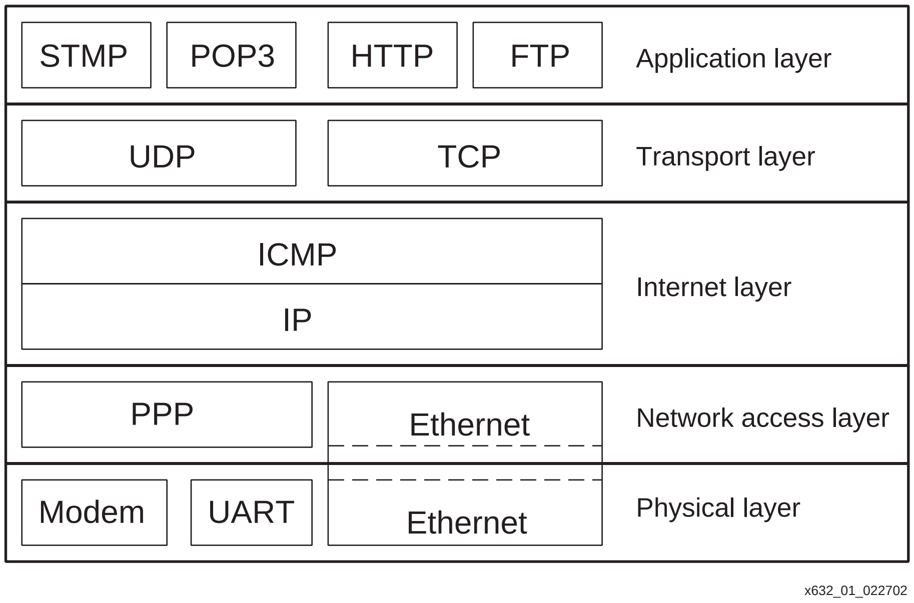


Figure 1: Internet Layer Protocol

Each layer is independent of the upper or lower layers and takes only incoming data, processes it, and passes it through to the upper or lower layer.

- Physical Layer - This is the hardware, such as coaxial, fiber, and telephone or radio waves or fiber optics used for communication between devices and connected to Modem-UART combinations or Ethernet controllers.
- Network Access Layer - This layer manages the incoming protocol (such as Ethernet), encapsulates packets from higher-layer levels, or dismantles the access layer protocol for the higher-layer protocol. Examples are: PPP (modem and UART) or Ethernet.
- Internet Layer - This layer is also called Internet Protocol (IP). The service protocol, Internet Control Message Protocol (ICMP) can also be located at this level.

  The IP protocol provides packet routing. It does not provide error recovery nor does it provide flow control. Another activity of this protocol is to break larger data blocks into fragments at the transmitter side and re-assemble them at the receiver side.

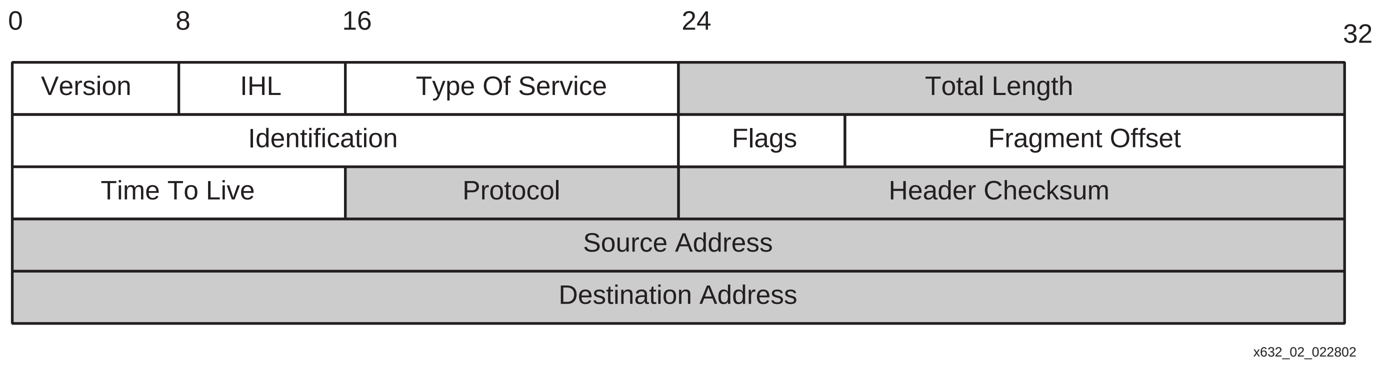


Figure 2: IP Header Format

The ICMP protocol is an extension to the IP protocol and is used for diagnostic messages. The most widely used ICMP command is the ping utility.

The implementation of the entire protocol is not necessary for the functioning of the Internet Protocol Stack, but ping should be implemented. Ping is implemented in this application note.

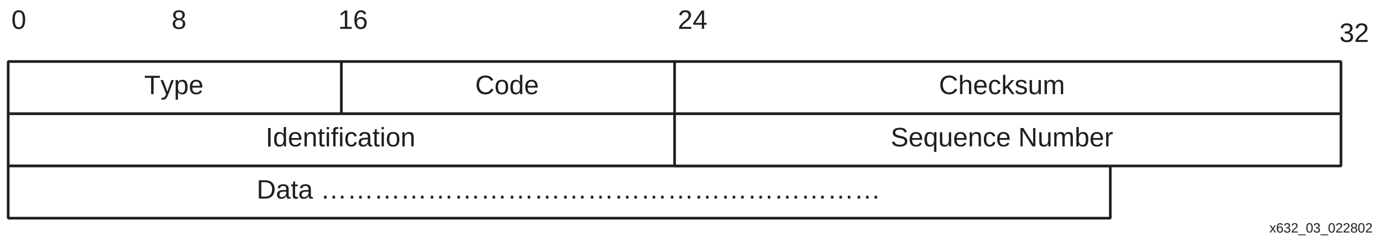


Figure 3: ICMP Layer Format

- Transport Layer - The two protocols taking nearly all the Internet/intranet traffic are TCP and UDP.
  - Transmission Control Protocol (TCP) - This is the default and most widely used standard of Internet data traffic. It is used as the lower layer of other protocols (HTTP, SMTP, and POP3).
  - User Datagram Protocol (UDP) - This is less used in recent applications because it is not as reliable as TCP.

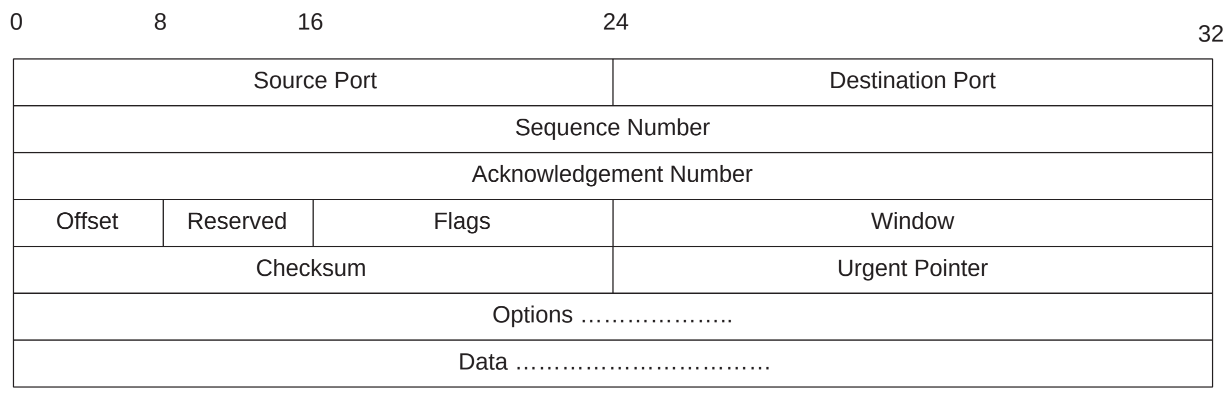
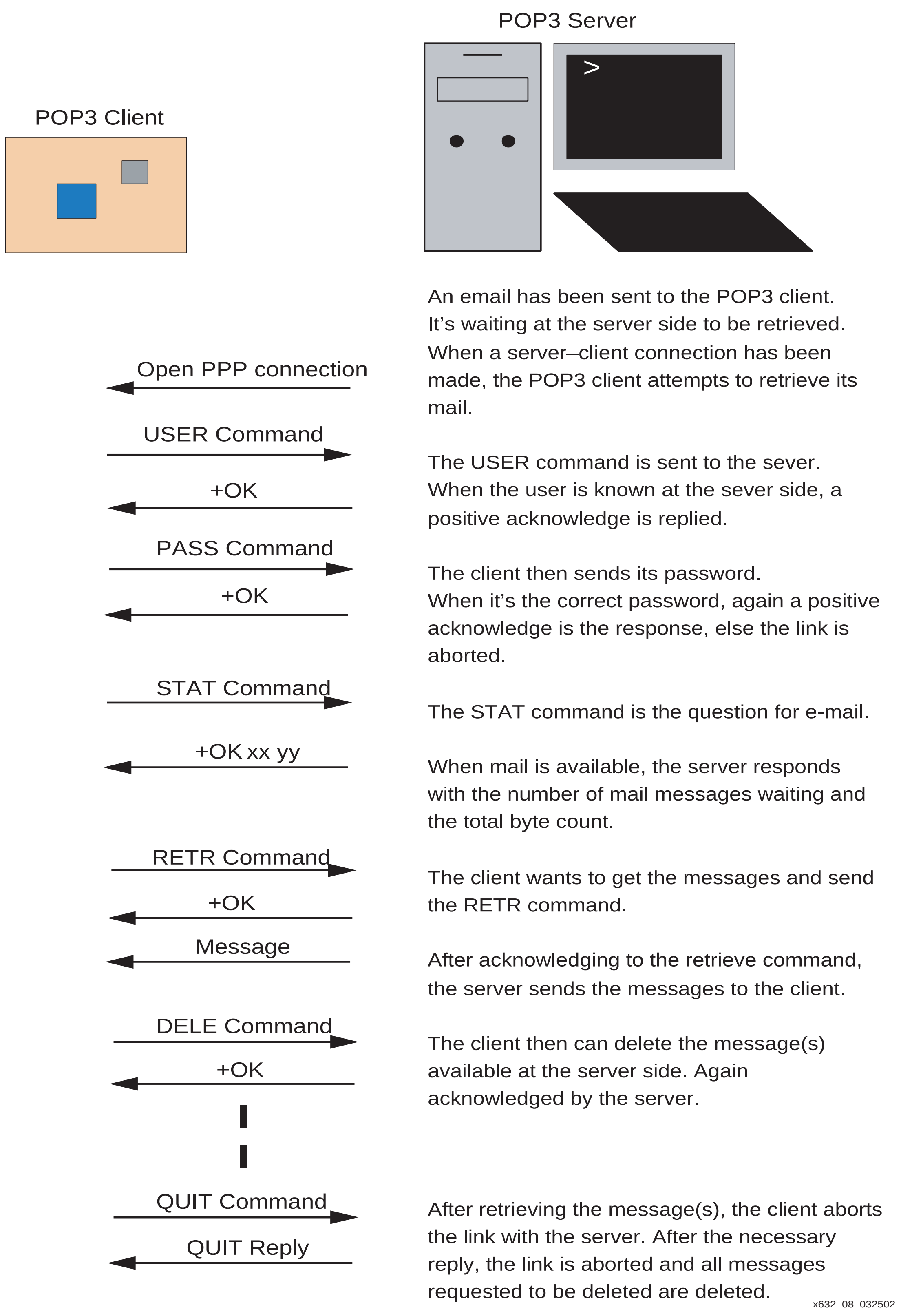


Figure 4: TCP Header Format

- Application Layer - Protocols such as HTTP, FTP, POP3, SMTP are found in the application layer and are integrated into widely used Internet browser and E-mail client programs. Note that since this application note is based on E-mail, only E-mail protocols are discussed.

E-mail Principles

Two most commonly used client-server protocols for sending and receiving mail messages across the Internet are SMTP and POP3. The client is usually a desktop E-mail program and the server is the Internet Service Provider (ISP) SMTP server. Both SMTP and POP3 client-servers are transaction-oriented around a simple text-based protocol. The reference design is acting as a POP3 client, only retrieving E-mail from POP3 a server. A POP3 connection is initiated by the client, after a connection with the server has been established, asking for E-mail messages. Each step in the ongoing communication must get a reply (action) from the opposite side (server).



An email has been sent to the POP3 client. It's waiting at the server side to be retrieved. When a server–client connection has been made, the POP3 client attempts to retrieve its mail.

The USER command is sent to the sever. When the user is known at the sever side, a positive acknowledge is replied.

The client then sends its password. When it's the correct password, again a positive acknowledge is the response, else the link is aborted.

The STAT command is the question for e-mail.

When mail is available, the server responds with the number of mail messages waiting and the total byte count.

The client wants to get the messages and send the RETR command.

After acknowledging to the retrieve command, the server sends the messages to the client.

The client then can delete the message(s) available at the server side. Again acknowledged by the server.

After retrieving the message(s), the client aborts the link with the server. After the necessary reply, the link is aborted and all messages requested to be deleted are deleted.

Figure 5: **POP_3 Communication**

External Controller SX52BD

The Ubicom SX52BD is a small controller based on RISC architecture. High-speed computation, flexible I/O control, and efficient data manipulation at speeds up to 100 MHz. Most of the instructions are optimized into single-cycle events. Together with ready-made software modules (Virtual Peripheral™), there are many Internet/intranet possibilities. A data sheet of these devices is available on the Ubicom web site: **http://www.ubicom.com/products/sx/index.html**. The software module used in this reference design example is the freely available SX-Stack. With some small modifications and additions, the module is used to program an FPGA.

Operation

The implementation has been tested with a SXB-TCP01-02 board of Ubicom, containing a SX52BD processor and two UART ports.

Download the E-mail server Assembler source code, eSXv1_0_4.src, from the Ubicom web site. Select the POP3 demo and POP3 debug features in the file. The POP3 debug facilities send the incoming E-mail content on the PPP port unmodified to the debug port. The modifications to the code are:

- The need exists to implement a mechanism to not take every possible E-mail as an FPGA bitstream. The mechanism used here gives the mail containing the bitstream a specific subject header.
- A clock pulse must be made along with each bit of data, emulating a slave serial download.
- The done pin must be used as input to the microcontroller to abort the intranet/Internet connection.

Some hints when modifying the Assembler source code include:

- When using Windows 2000 or Linux, set or unset the WIN32 and WIN98 options.
- The second UART, debug UART, is used when PO3DEBUG is uncommented in the options section of the code.
- The default IP address of the demo can be modified when changing POP3address1, POP3address2, and POP3address3 in the POP3 constants section.
- The ports used by the second UART can be changed to another location. Search for "debug_rx_pin" and "debug_tx_pin" and change the pin numbers accordingly.
- The POP3 user and password can be modified. Look for "USER_eSX" and "PASS_eSX" in the canned ppp section of the code.
- The subroutine used to initialize the POP3 demo can be found from line 2164 of the source code. It reads:

```
2164   IFDEF POP3DEMO
2165   ;-----------------------------------------------------------------
2166   ;Subroutine: AppInit
```

- The POP3 application starts from line 4439, and reads:

```
4439   ;=======================================================
4440   ; TCP POP3 Demo Functions
4441   ;=======================================================
```

This is the POP3 state machine and its subroutines.

- Here modifications can be made to recognize the E-mail header "Subject: …" and to pass bits or transfer characters. The bitstream to download into the FPGA can be sent as a binary attachment to the mail message or can first be converted with the Xilinx tools to ASCII and then incorporated into the body of the E-mail. Depending on the method used, the Assembler source code must be modified accordingly.
- To generate along with each data bit a clock pulse the debug UART code can be modified slightly.

Be aware that this small processor is at its maximum performance when running the TCP/IP, POP3 application and the data memory is nearly completely used. This small controller can only work via a UART (PPP) port. A better solution is found in the IP2022 processor of the same manufacturer Ubicom.

Internal Controller

It is also possible to configure an FPGA by means of the internal available microcontroller, whether it is the MicroBlaze™ soft core or PowerPC hard-core controller.

It will be slightly more complex and will need some external logic.

With the use of an internal controller, peripherals must be downloaded. Even the controller must be downloaded when using the MicroBlaze soft core. Controller and peripherals must become operational before any hardware operation, such as contacting the Internet/intranet is possible.

A possible setup to download new configurations from the Internet is shown in Figure 6.

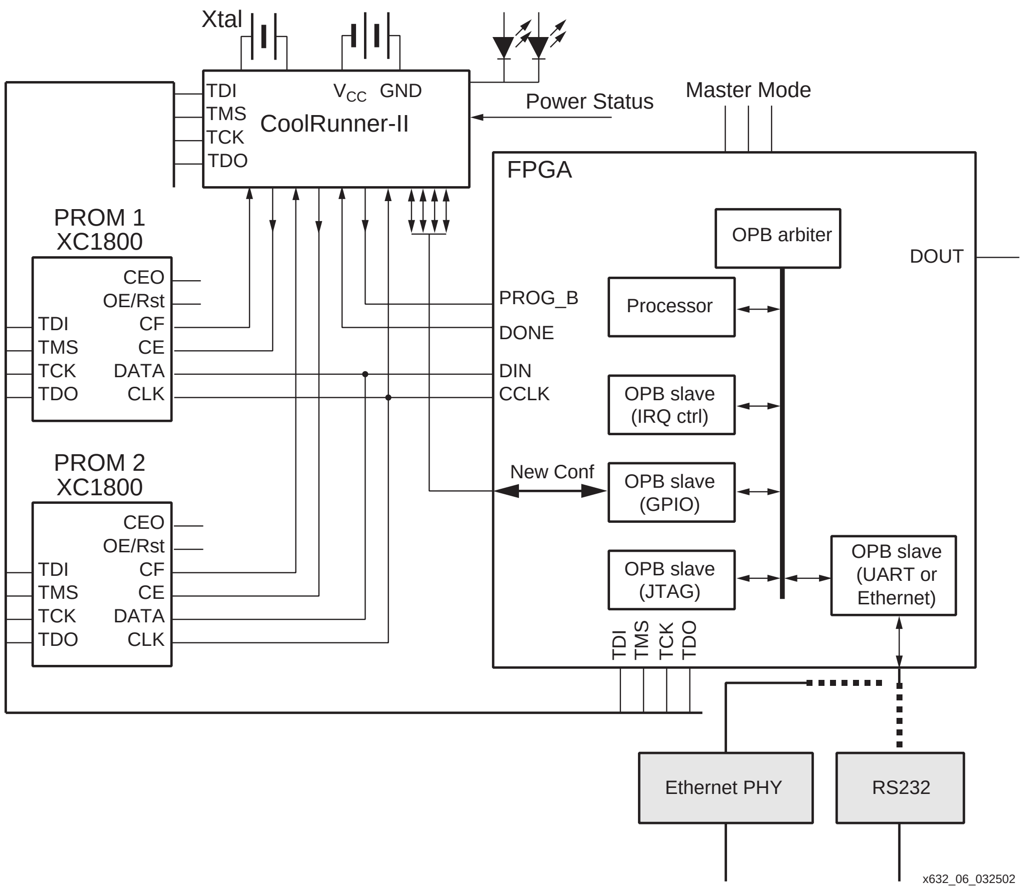


Figure 6: **Internal Download**

Operation

A basic FPGA design must be made. This basic design can be the processor with its peripherals, or the complete initial FPGA design.

- In case the initial FPGA design only contains the processor and some peripherals, once the application is started the real design must be downloaded from the network. When the FPGA contains an operational design, the connection to the network can be delayed until it is necessary to download a new bitstream.

In both cases, the controller needs to run a TCP/IP stack and POP3 application. This is available when an OS is implemented on the processor, or else it must be ported onto the processor as separate code.

At least two possible semi-permanent memory (Flash PROM) banks must be designed. It is possible that one bank has multiple Flash memory devices, depending the size of the used FPGA or the number of FPGAs in master-slave configuration mode, for example, a XC2VP20 bank needs two XC18V04 devices. This example uses the FPGA's JTAG port to configure the Flash memories (the CPLD can also be configured through the FPGA). However, a processor peripheral to control the JTAG port must be designed.

The bitstream received at the FPGA must be JTAG prepared; it must contain the necessary command to generate a pulse onto the CF output once the PROM is programmed. The CF pin of the PROM allows a JTAG CONFIG instruction to initiate FPGA configuration without powering down the FPGA. It is an open-drain output that is pulsed Low by the JTAG CONFIG command `11101110`. A download controller device (here a CoolRunner™-II CPLD) is used to control the source of the data stream to the FPGA or FPGAs. The FPGA must be set in Master Serial mode; other FPGAs can be connected in Slave Serial mode onto the first FPGA.

Download Controller

An 8-bit register expressing the states of inputs and outputs is the major part of this design. Two down counters are needed, one for the number of download retry loops and one with the total bitstream length. The register is shown in Table 1.

Table 1: **The Register**

Valid Conf_2	Done2	CEN_2	New Conf_2	Valid Conf_1	Done1	CEN_1	New Conf_1	Power Status
0	0	1	0	1	0	1	1	0
Internal	Input	Output	Input	Internal	Input	Output	Input	Input

Flash PROM(s) and the CoolRunner-II CPLD are factory programmed. PROM 1 contains a basic FPGA code and PROM 2 is empty. The CPLD has the controller design, with some registers preset to indicate that PROM 0 contains valid code.

Controller Function

The controller is a state machine. The flowchart of the process is shown in Figure 7.

1. After power-up of the application, the download controller starts in a predefined state and waits for a valid power status. This is a way to indicate that all power supplies have reached a valid voltage level and FPGA download can start.
2. Check that "Valid Configuration" is performed. At first use "Valid Conf_1" is factory pre-programmed.
3. Enable the PROM with the known configuration.
4. Initiate the counters.
    a. "LoopCnt" is a small down counter to retry the download operation a few times when necessary.
    b. "DldCnt" counts down at the rate of the bitstream.
5. Give a pulse on the FPGA PROGRAM pin.
6. The FPGA set in Master Serial mode issues a clock waveform on the CCLK pin and retrieves data from the PROM(s) at the same time the CCLK clock is fed into the CPLD controller and the CCLK pulses make the "DldCnt" count down:
7. The FPGA when correctly downloaded brings the DONE pin High. This state is latched into the CPLD controller.
8. A few clock pulses later the "DldCnt" down counter reaches zero and the DONE status can be controlled.

9. When DONE is High:
   - The respective "Valid Config" bit is set.
   - The "New Config" bit is reset.
   - The DONE status is reset to zero.
   - The PROM is disabled.
10. The FPGA is operational.
11. When the DONE status bit is not High, the "loopCnt" is decremented and the state machine returns to state 4b.
12. When unsuccessful in downloading until "LoopCnt" is reaching "0."
   - Reset respective "Valid Config" bit.
   - Reset respective "New Config" bit.
   - Blink an LED to indicate that configuration of the FPGA is not possible.
13. When a new bitstream is loaded into semi-permanent memory devices during (or at the beginning), the processor handling the download must indicate that a new configuration will become available by setting the respective "New Config" bit in the CPLD controller.
   - The processor reads the "Valid Config" and New Config" bits to know in what Flash PROM bank the new configuration must be stored.
14. At the end of the JTAG programming, the PROM generates a CF pulse.
15. This pulse triggers the state machine and a download starts.
   - When DONE goes High, the new configuration becomes the default one by setting the "Valid Config" bit.
   - When DONE does not go High, a set of trial downloads is executed. The previous "good" configuration is downloaded when the DONE status is false after the different retries.

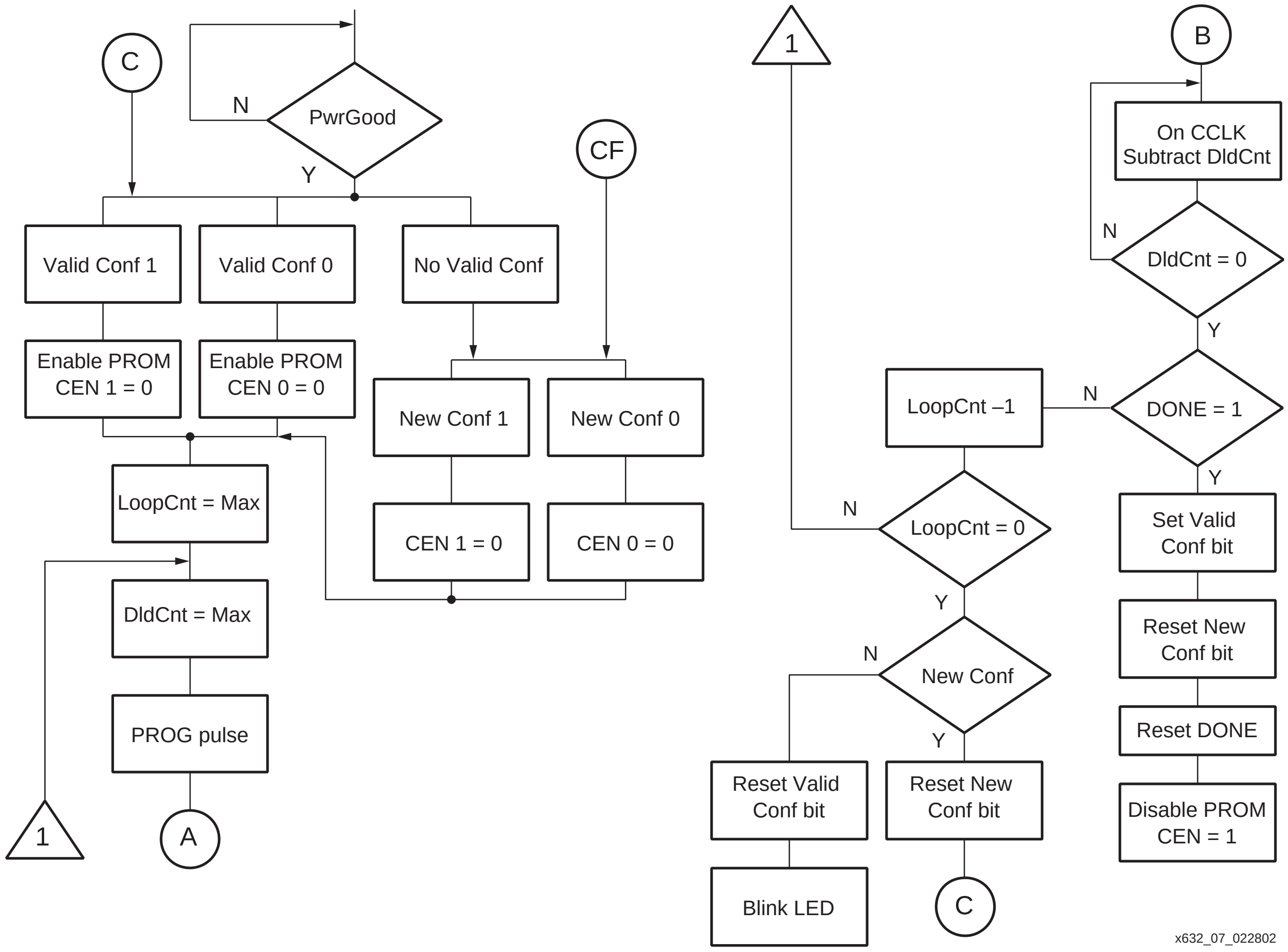


Figure 7: **Controller Flow Chart**

When the state machine arrives at point A, it stops operation. See Figure 7. The FPGA will now generate CCLK pulses, starting from point B, decrementing the "DldCnt" counter until "0." Now the controller state machine takes over.

Reference Design

To obtain the SX52BD Assembler reference code for this design, log into the Ubicom support portal at: **http://support.software911.com/ubicom/login.asp?op=8**

Conclusion

FPGA programming and reprogramming over the Internet or an intranet system is possible and quite simple. This application note and accompanying reference design allows designers to use this protocol today.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
05/13/02	1.0	Xilinx initial release.