XILINX®

XAPP372 (v1.0) September 19, 2003

CoolRunner-II Smart Card Reader

Summary

This application note describes the implementation of a Smart Card Reader design with a CoolRunner™-II CPLD. Different from most of the software-based smart card reader computer systems, this CoolRunner-II CPLD implementation is a hardware solution. There is no software development needed in this design. This application note explains the low-level protocol of the Smart Card Reader and its hardware implementation.

CoolRunner-II devices are the latest CPLDs from Xilinx that offer both low power and high-speed. A VHDL code for Smart Card Reader design is available with this application note: see **Source Code**, page 15

Introduction

A smart card is a credit card sized plastic card with an embedded microprocessor and memory and is used for identification, access, and conducting financial transactions.

Figure 1: **Smart Card Reader**

Acting like a mini-computer, smart cards allow money and information to be electronically stored and transferred in a secure but portable medium. When inserted into a reader or passed over a scanner, the smart card transfers data to and from a central computer. Overall, it is a replacement for old means of retaining data and transacting business.

There are two fundamental sides of development for any smart card application: the host-side and card-side. Software programming tends to comprise most of the effort involved in smart card development. The host software runs on a computer connected to a smart card, and the card software runs on the card as a counterpart to the host software.

In this application note we use a CoolRunner-II CPLD to build a smart card reader, creating a host-side design to replace a host computer system. The function of a smart card reader is to read the card content and display the decoded information on a character LCD display. Unlike standard smart card readers, though, this CoolRunner-II reader relies on hardware design rather than software programming to perform these tasks.

Smart Card Reader Block Diagram

The block diagram of the CoolRunner-II smart card reader is shown in Figure 2. The dashed area shows the logic that is contained in the CoolRunner-II CPLD. All the other blocks are external devices that can be obtained commercially. The CPLD logic blocks and the external devices in this diagram are briefly described in the following section.

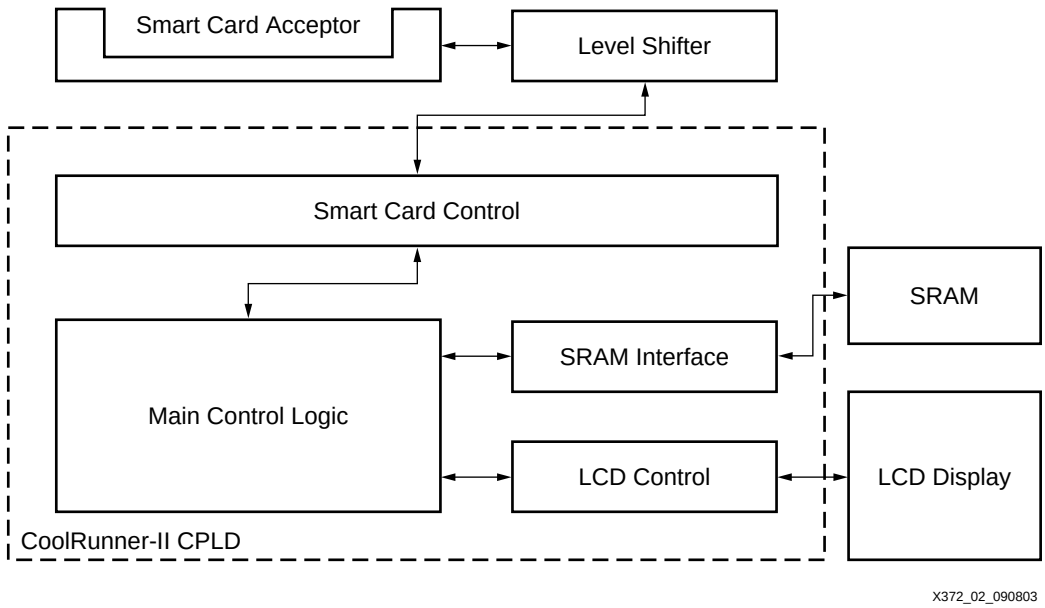

Figure 2: **Smart Card Reader Block Diagram**

CoolRunner-II CPLD Modules

Main Control Logic

The Main Control Logic block provides the system flow control. It reads data from smart card control, writes to SRAM, activates LCD control and sends decoded information to the LCD control.

Smart Card Control

Smart Card Control logic communicates with the smart card through the external level shifter. It uses a predefined communication protocol and commands. The data length for each field is also hard coded in this design.

SRAM Interface

SRAM Interface logic interfaces to the external SRAM. This logic block controls the SRAM read/write addressing. Smart card data is saved in SRAM and later fetched and decoded for the LCD display.

LCD Control

LCD control logic interfaces to the LCD Display. This logic block accepts the decoded data and writes to the LCD Display. Refer to **XAPP431, CoolRunner-II Character LCD Display Controller**, for detailed information.

External Devices

Level Shifter

The On semiconductor NCN6011 is a level shifter (analog device) to translate the voltages between a smart card and CoolRunner-II CPLD. This device handles all the 5V signals needed for the smart card and 1.8V signals for the CoolRunner-II CPLD. It is transparent to the smart card control logic in the CPLD.

LCD Display

The OKAYA RC1602ARS is a 16-character x 2-line, dot matrix, liquid crystal display module. It has an on-board controller and LSI drivers, which display alpha numerics, Japanese KATA KANA characters and a wide variety of other symbols in 5 x 7 dot matrix.

SRAM

A low power, 32k x 8 ISSI IS61LV256 SRAM is used in the module memory. The functionality of the SRAM does not require additional explanation. For more in-depth SRAM information refer to the ISSI SRAM data sheet.

Smart Card Acceptor

The Amphenol C702 10M008 2834 is a low cost smart card acceptor. It provides a direct sliding contact between the acceptor and the smart card. This acceptor has a “NC closed card present switch” which opens when the card is fully inserted. The switch activates the smart card reader operation.

Smart Card Standard ISO 7816

Smart Card is defined by the international standard ISO 7816. The first two parts cover smart card’s physical dimensions and locations of the chip contacts. A smart card image and its contacts are shown in Figure 3.

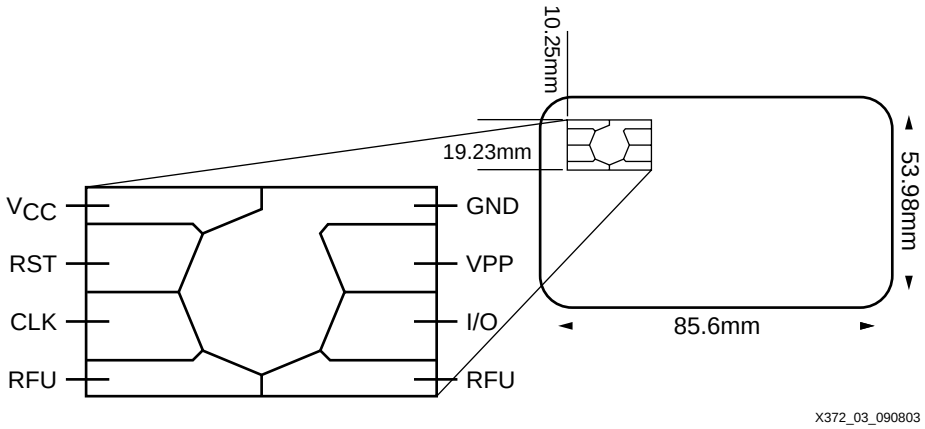

Figure 3: **Smart Card Image and Contacts**

ISO 7816-3 & -4 govern the electronic signals, transmission protocols and inter-industry commands for interchange. In this application note we limit the discussion to its transmission protocols and some basic commands.

ISO 7816-5 to –8 cover the number system, data elements, card SQL and security commands. These parts are not used by this reference design and will not be discussed in this application note.

Operating Procedure

This section details the ISO 7816-3 inter-operation between the smart card and the host device.

- Connection and activation of the contacts
- Reset of the card
- Answer to Reset
- The T=0 communication protocol

Connection and activation of the contacts

The activation of the contacts by the host device consists of the consecutive operations:

- RST is L
- VCC is powered
- I/O in the interface device is in reception mode
- VPP is raised to idle state
- CLK is provided with a suitable, stable clock

Reset of the card

The host device initiates a reset to smart card and the card responds with an Answer to Reset within 40000 clock cycles with Reset in state H. If an Answer to Reset does not occur, the Reset returns to state L and the smart card contacts are deactivated by the host.

Answer to Reset

There are two types of transmission defined in ISO 8616-3 for answer to reset: asynchronous and synchronous transmission. In this application note we only discuss asynchronous transmission.

In asynchronous transmission, characters are transmitted on the I/O line in an asynchronous half-duplex mode. The normal bit duration used on I/O is defined as one Elementary Time Unit (etu). The initial etu is 372/fi second where fi is in Hertz. All are initially operated with fi in the range of 1 MHz to 5 MHz.

A character consists of ten consecutive bits plus a guard time as follows:

- Start bit, used for character frame synchronization
- 8 data bits of information
- Parity bit, even parity for error detection

The guard time is separation between characters. Figure 4 is a diagram for asynchronous character frame.

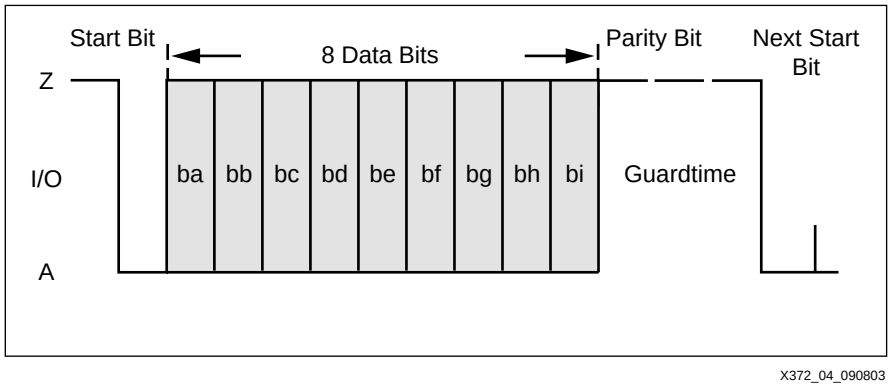

Figure 4: **Asynchronous Character Frame**

The Answer to Reset is at most 33 characters and consists of 5 fields,

- The initial character (TS)
- The format character (TO)
- The interface characters (TAji, TBji, TCji, TDji)
- The historical characters (T1, T2 ... TK)
- The check character (TCK)

Each of these fields is sent in order as shown in Figure 5.

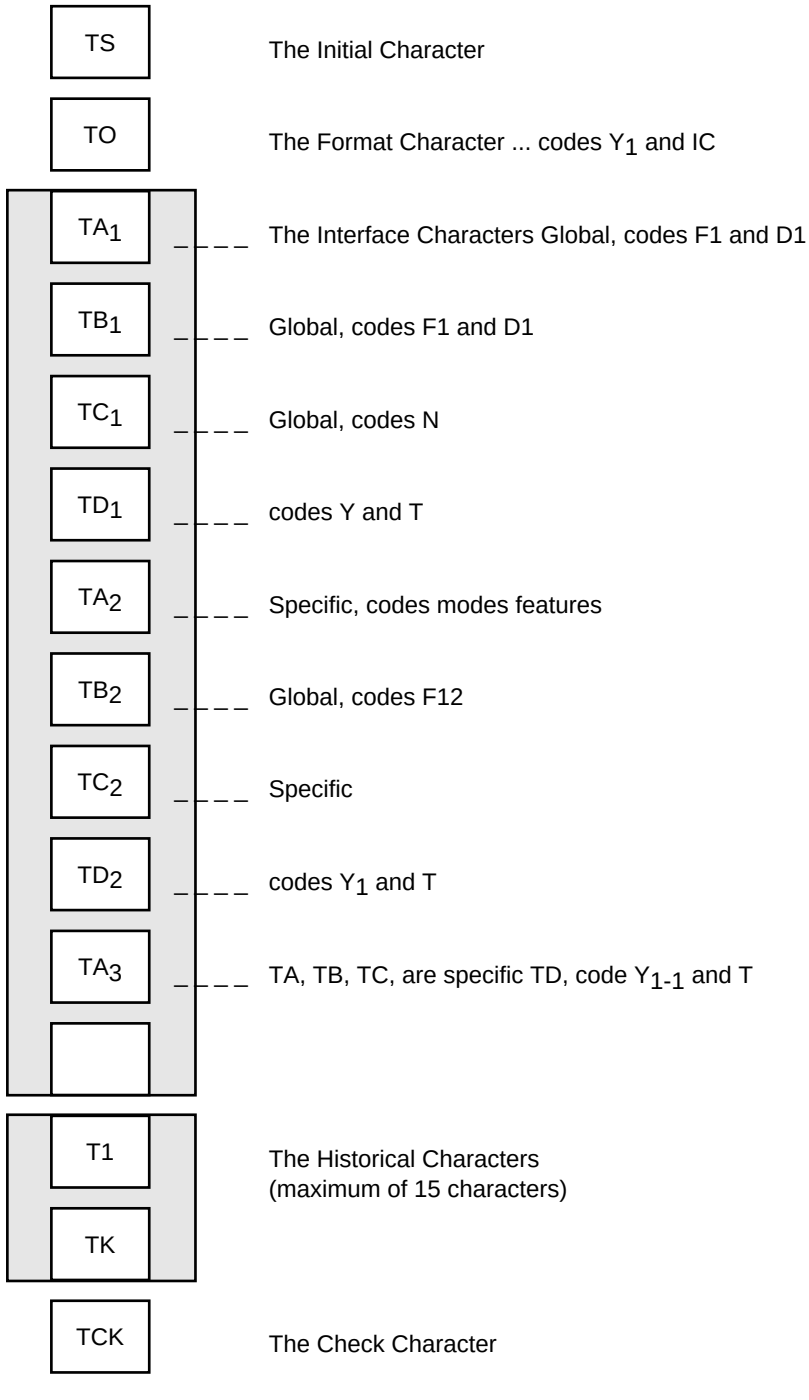

Figure 5: **Answer to Reset Configuration**

The initial character TS determines the data transmission rate and also determines the sense of the logic. The format of the TS character is shown in Figure 6. This shows the two possibilities of the direct and inverse convention, where logic level one is A, Ba is MSB for inverse and logic level one is Z, and Ba is LSB for the direct convention.

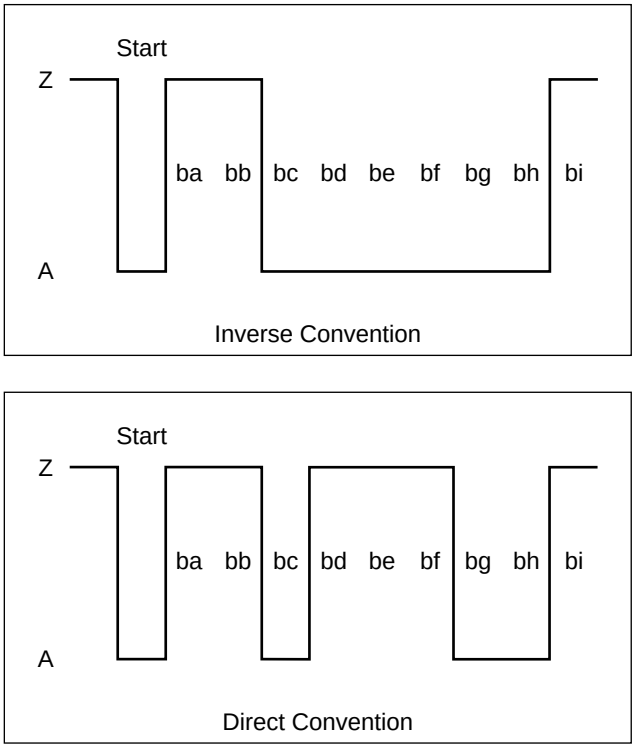

Figure 6: **Initial Character TS**

The format character TO provides information necessary to interpret the remaining answer to reset characters. See TO in Figure 7. the most significant half byte, b8 to b5, indicates the presence of TA1 to TD1. The least significant half byte, b4 to b1, indicates the number of historical characters.

TA1 defines the basic characters of the serial transmission. F1 is the clock rate conversion factor and D1 is the bit-rate adjustment factor. F1 and D1 are compared against a table in the ISO 7816-3 standard to achieve actual values of F and D in the table to define the actual work etu.

TB1 is used to define the EPROM programming voltage and current. TC1 provides the value of N, which defines the extra guard time to be used between successive characters. The first half byte of TD1 indicates the presence of TA2 to TD2. The second half byte of TD1 indicates the protocol type T=0 to T=15.

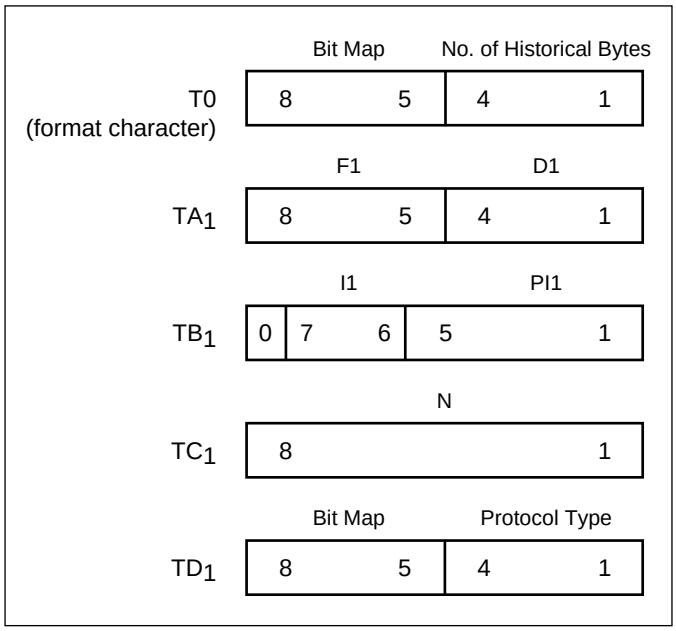

Figure 7: **Format and Interface Characters**

The historical characters may be used to convey information relating to the life cycle of the card. The check character should not be sent when only the T=0 protocol is indicated in the answer to reset. In all other cases TCK is sent as the last character of the answer to reset. The value of TCK is such that the exclusive-or of all bytes from T0 to TCK included is equal to zero.

The T=0 Communication Protocol

The interface device always initiates the command for the T=0 protocol. Interaction between the interface device and the card results in successive commands and responses. The message flow for the T=0 protocol is shown in Figure 8.

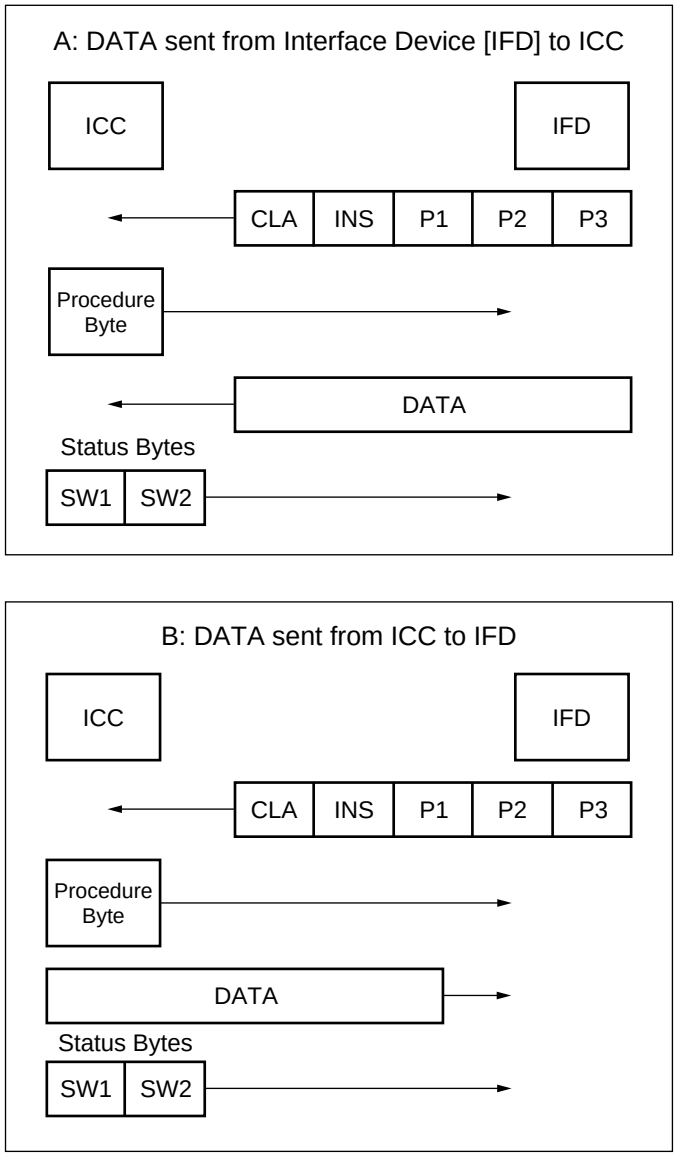

Figure 8: **The T=0 Protocol**

In Figure 8. IFD is the smart card controller and ICC is the smart card. The command header consists of the following 5 bytes,

- CLA, the instruction class
- NS, the instruction code
- P1, instruction code qualifier (e.g. memory address)
- P2, additional INS code qualifier
- P3, the length of the data block

The response from the card has two status bytes, SW1 and SW2, to indicate the current card status. The normal response is SW1, SW2 = 90, 00 hex. When SW1 = 6x or 9x various error conditions are reported by the card.

Table 1 and Table 2 show some of the CPA classes and INS commands. In this design we use ISO 7816-4 instruction class 80 and some basic INS codes such as A4, Select File, B2, Read Record and C0, and Get Response to read all the information we need.

Table 1: **CLA Instruction Set**

CLA Type	Instruction Set
0X	ISO 7816-4 instructions
10 to 7F	Reserved for future use
8X or 9X	ISO 7816-4 instructions
AX	Application/Vender specific instructions
B0 to CF	ISO 7816-4 instructions
D0 to FE	Application/Vender specific instructions
FF	Reserved for protocol type selection

Table 2: **INS Commands**

INS Value	Command Name
0E	Erase Binary
20	Verify
82	External Authentication
88	Internal Authentication
A4	Select File
B0	Read Binary
B2	Read Record
C0	Get Response
C2	Envelope
D0	Write Binary

CoolRunner-II Implementation

The CoolRunner-II smart card reader design uses the Advanced Card System ACOS1 microprocessor-based card. The information read from the card includes name, gender, status, age, and bank balance. gender, status and age are encoded in the same data record.

The initial character TS was programmed to direct convention, and format character and interface characters are predefined. T=0 protocol is used so there is no TCK in answer to reset. There are 19 bytes, including historical characters transmitted for answer to reset. To simplify the design, we count bytes received from the smart card with the CoolRunner-II to determine the valid data to be used.

There will be no parity check for each character frame and no branch operations to handle different protocol or bytes received. In this situation only the ACOS1 card can be used for this smart card reader.

Smart Card Control

A smart card control block diagram and state machine are shown in Figure 9 and Figure 10. When the card is inserted into the card acceptor, the switch turns on to enable the smart card state machine. By following the ISO 7816-3 standard sequence to activate smart card contacts,

the CoolRunner-II device sets Reset to high, goes to the Wait state, and waits for a low signal from the card, the start bit of the TS for Answer to Reset.

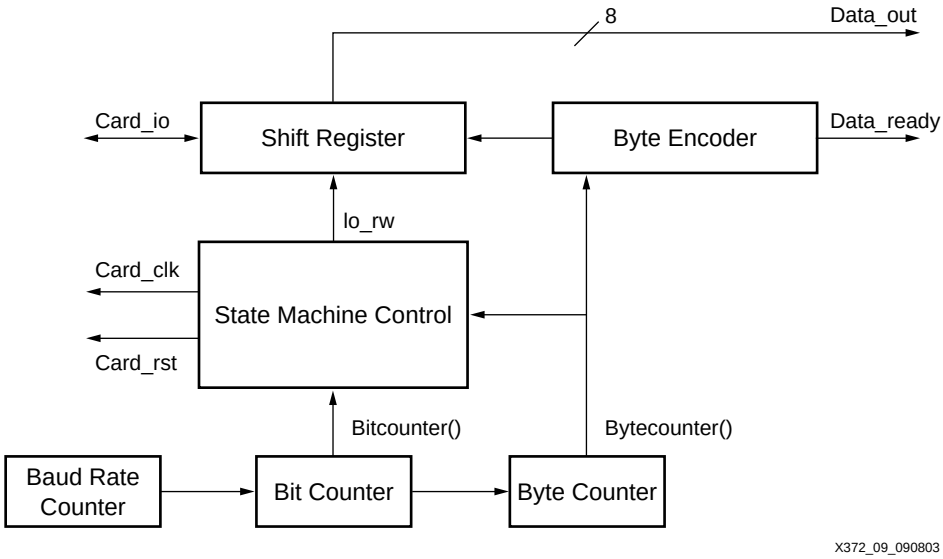

Figure 9: **Smart Card Control Block Diagram**

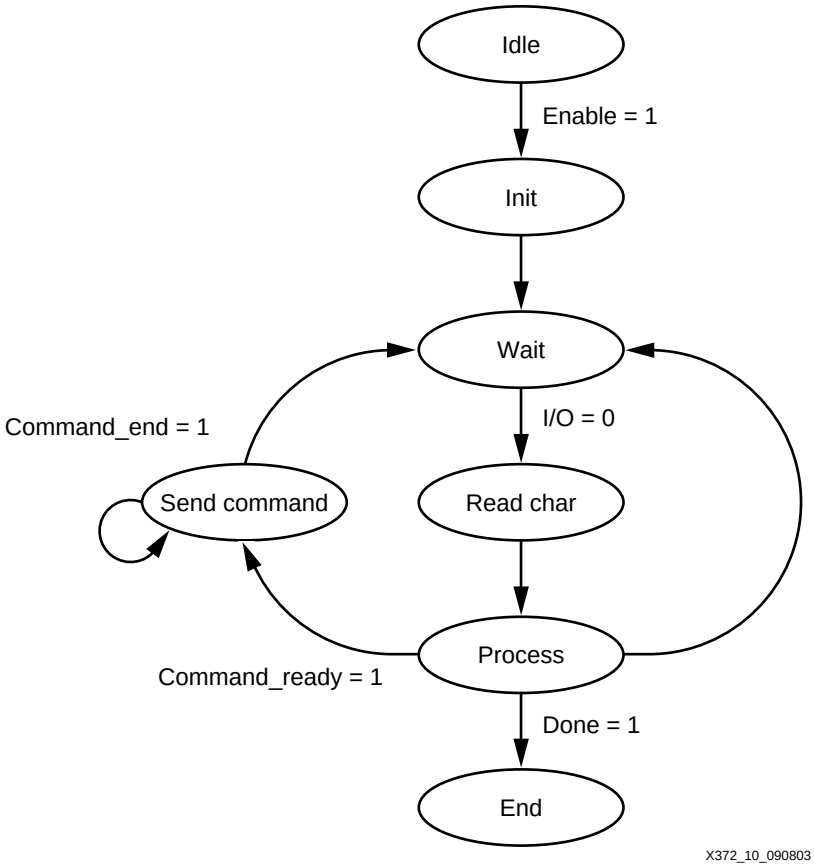

Figure 10: **Smart Card Control State Machine**

There is a baud rate counter counting to 372 for every bit received or sent to synchronize the data transmission. The received bits are sampled at the 186th count, which is in the center of each bit received.

Two other counters are used, one to count bits for each character and one to count bytes received or sent. The byte number is also used to determine the end of the read data cycle or the send command cycle.

For this preprogrammed ACOS1 smart card, the state machine will set to the send command state after 19 Answer to Reset characters are received. The smart card controller is now ready to send commands and receive responses based on the T=0 protocol. There is decoder logic to check the character counts to determine when to send a command, what command to send, and how many characters will be received by the request.

The first command sent to the smart card contains 5 bytes: 80, A4, 00, 00 and 02. After one procedure byte (A4) is echoed from the smart card, the controller sends two data bytes (F0, 00) and the smart card responds with two status bytes (91, 00). As is clear from the T=0 command table, A4 selects the file address and F0, 00 is the selected file address.

The subsequent commands are to select data records and to retrieve data. All the commands and data lengths are already defined and fixed. In this design, io_rw controls the shift register to read data from card_io or shift data from data encoder to card_io. All operations are based on the byte count.

In this design, the name record is between bytes 38 and 69. gender, status and age records are bytes 92, 93 and 94. The bank balance record are bytes 126 and 127. A data_ready signal is enabled for these bytes to filter out unused data. This signal is used for the SRAM interface to write the smart card data to the SRAM.

Main Control Logic

A main state machine in this block controls the ordering of the functions. It also reads the SRAM data and decodes it before sending it to the LCD control logic. There is also a delay counter to separate the data sent to the LCD display, so there will be 1 to 2 seconds between each piece of information displayed on the LCD.

A block diagram is shown in Figure 11 and its flow chart is shown in Figure 12. The state machine powers up in an idle state and waits for the smart card controller to read the data and save it to SRAM. When the smartcard_done bit goes high it will go to the standby state and wait for the lcd_ready.

The card name has the same ASCII format as the LCD display so there is no need to decode the characters. All characters read from SRAM will be dumped to the LCD Display.

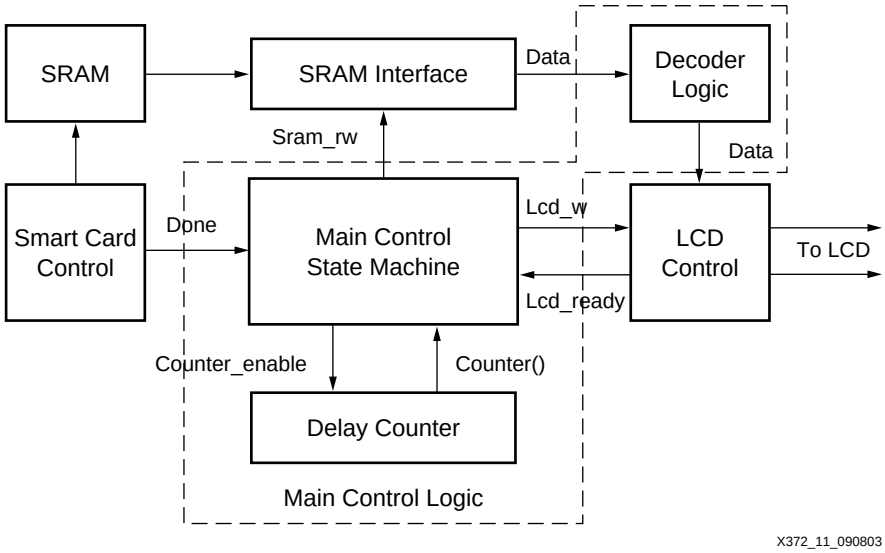

Figure 11: **Main Control Logic Block Diagram**

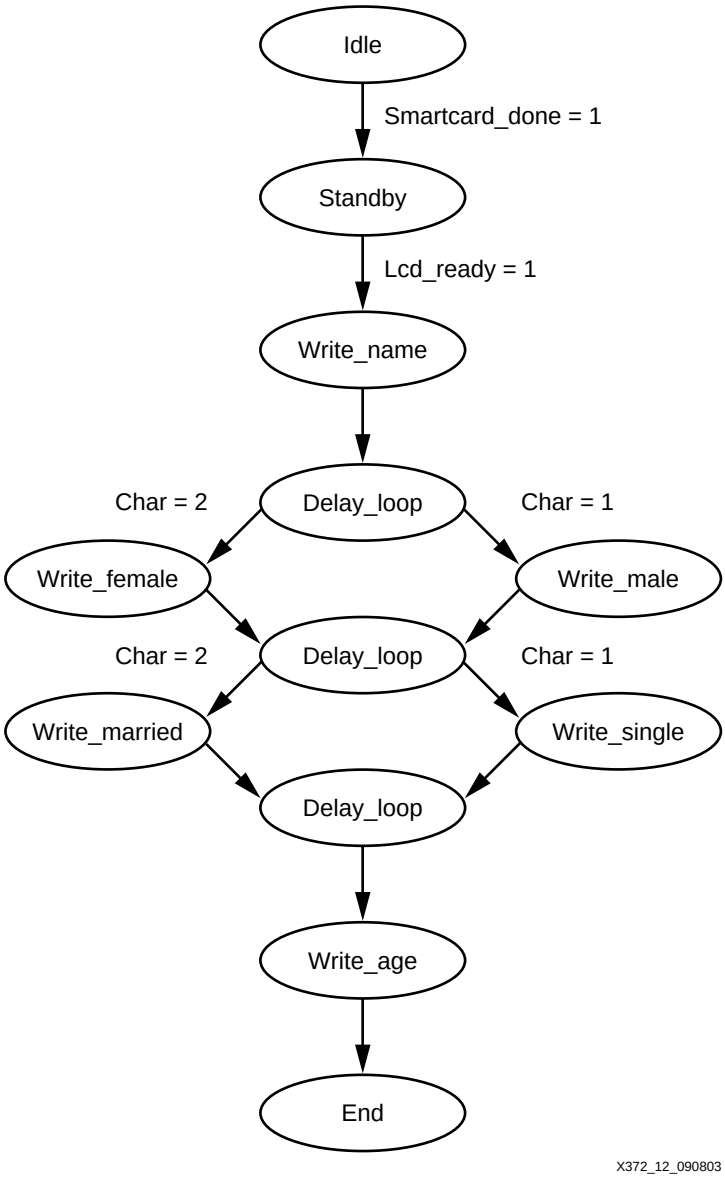

Figure 12: **Main Control Logic Flow Chart**

The next data record is decoded gender information. The data value is one for male, two for female. After gender comes status information, also decoded as one and two (one for single, two for married).

The age information is saved as binary value in the smart card. It has to be converted to ASCII digits before getting sent to the LCD controller. A binary-to-digital module is separated from the top module, which is for the ASCII coding function.

SRAM Interface

The SRAM interface is controlled by the main control logic. The signal sram_w is always kept high as write enable during smart card reading. The address counter will be reset when the main control state machine enters the standby state for writing data to the LCD display. The data is then retrieved in the order it was saved.

LCD Control

LCD control logic is used to initialize and pass decoded data to the LCD display. This module uses simplified timing to control the LCD display. Every character write cycle is set to 30000 clock periods and is much higher than the few hundred microsecond LCD controller specification. The detailed description of the LCD display controller can be found in XAPP CoolRunner-II Character LCD Display Controller.

Source Code

THIRD PARTIES MAY HAVE PATENTS ON THE CODE PROVIDED. BY PROVIDING THIS CODE AS ONE POSSIBLE IMPLEMENTATION OF THIS DESIGN, XILINX IS MAKING NO REPRESENTATION THAT THE PROVIDED IMPLEMENTATION OF THIS DESIGN IS FREE FROM ANY CLAIMS OF INFRINGEMENT BY ANY THIRD PARTY. XILINX EXPRESSLY DISCLAIMS ANY WARRANTY OR CONDITIONS, EXPRESS, IMPLIED, STATUTORY OR OTHERWISE, AND XILINX SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR A PARTICULAR PURPOSE, THE ADEQUACY OF THE IMPLEMENTATION, INCLUDING BUT NOT LIMITED TO ANY WARRANTY OR REPRESENTATION THAT THE IMPLEMENTATION IS FREE FROM CLAIMS OF ANY THIRD PARTY. FURTHERMORE, XILINX IS PROVIDING THIS REFERENCE DESIGN "AS IS" AS A COURTESY TO YOU.

XAPP372 –http://www.xilinx.com/products/xaw/coolvhdlq.htm

Conclusion

This document has explained outlined some of the smart card protocol and has explained how to implement it using a CoolRunner-II, hardware-based solution. Although this Smart Card Reader design is a simplified version with no error checking or frills, it is a good reference for users needing to approach for smart card applications with minimal software development.

References

1. International Standards Organization, ISO 7816
2. Scott B. Guthery & Timothy M. Jurgensen, Smart Card Developer's Kit
3. David B Everett, Smart Card Tutorial
4. Advanced card systems Ltd., ACS software development kit

Further Reading

Application Notes

http://www.xilinx.com/xapp/xapp371.pdf (Galois Field GF (2^m) Multiplier)
http://www.xilinx.com/xapp/xapp372.pdf (Smart Card Reader)
http://www.xilinx.com/xapp/xapp373.pdf (Avoiding CPLD Brownout)
http://www.xilinx.com/xapp/xapp374.pdf (CryptoBlaze)
http://www.xilinx.com/xapp/xapp375.pdf (Timing Model)
http://www.xilinx.com/xapp/xapp376.pdf (Logic Engine)
http://www.xilinx.com/xapp/xapp377.pdf (Low Power Design)
http://www.xilinx.com/xapp/xapp378.pdf (Advanced Features)
http://www.xilinx.com/xapp/xapp379.pdf (High Speed Design)
http://www.xilinx.com/xapp/xapp380.pdf (Cross Point Switch)
http://www.xilinx.com/xapp/xapp381.pdf (Demo Board)
http://www.xilinx.com/xapp/xapp382.pdf (I/O Characteristics)
http://www.xilinx.com/xapp/xapp383.pdf (Single Error Correction Double Error Detection)
http://www.xilinx.com/xapp/xapp384.pdf (DDR SDRAM Interface)
http://www.xilinx.com/xapp/xapp387.pdf (PicoBlaze Microcontroller)

http://www.xilinx.com/xapp/xapp388.pdf (On the Fly Reconfiguration)
http://www.xilinx.com/xapp/xapp389.pdf (Powering CoolRunner-II CPLDs)
http://www.xilinx.com/xapp/xapp393.pdf (8051 Microcontroller Interface)
http://www.xilinx.com/xapp/xapp394.pdf (Interfacing with Mobile SDRAM)
http://www.xilinx.com/xapp/xapp395.pdf (Using DataGATE)
http://www.xilinx.com/xapp/xapp398.pdf (CompactFlash Card Interface)

CoolRunner-II Data Sheets

http://direct.xilinx.com/bvdocs/publications/ds090.pdf (CoolRunner-II Family Datasheet)
http://direct.xilinx.com/bvdocs/publications/ds091.pdf (XC2C32 Datasheet)
http://direct.xilinx.com/bvdocs/publications/ds092.pdf (XC2C64 Datasheet)
http://direct.xilinx.com/bvdocs/publications/ds093.pdf (XC2C128 Datasheet)
http://direct.xilinx.com/bvdocs/publications/ds094.pdf (XC2C256 Datasheet)
http://direct.xilinx.com/bvdocs/publications/ds095.pdf (XC2C384 Datasheet)
http://direct.xilinx.com/bvdocs/publications/ds096.pdf (XC2C512 Datasheet)

CoolRunner-II White Papers

http://www.xilinx.com/publications/products/cool2/wp_pdf/wp165.pdf (Chip Scale Packaging)
http://www.xilinx.com/publications/whitepapers/wp_pdf/wp170.pdf (Security)
http://www.xilinx.com/publications/whitepapers/wp_pdf/wp197.pdf (Cipher Stream Protocol)
http://www.xilinx.com/publications/whitepapers/wp_pdf/wp198.pdf (Cell Phone Handsets)

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
11/19/03	1.0	Initial Xilinx release.