Application Note

USING THE CRYSTAL™ CS8900 IN 8-BIT MODE

By James Ayres

Introduction

The CS8900 is a good candidate for designs with an 8-bit data bus. Because of its small size and built-in filters the chip will take up a minimum of board space while providing a cost effective, high performance Ethernet connection. This application note shows how to use the CS8900 in 8 bit mode, including software information for the programmer and a typical connection diagram for the design engineer.

References

The designer should familiarize himself with Chapter 5 of the CS8900 Technical Reference Manual, ***Low cost, high performance Ethernet Controller for non-ISA systems***. This chapter is a reference on how to easily connect the chip to a non-ISA processor. It includes a reference connection diagram with glue logic connecting the CS8900 to an MC68302 microcontroller. Chapter 5 will contain much of the data needed for the design engineer. The data sheet is the source for functional descriptions of the registers, receive operation, transmit operation, timing etc. Only the 8-bit specific issues will be covered here.

Software Drivers

There are many software drivers available for the CS8900 in 16-bit mode, including VxWorks™, Psos®, Linux®, Packet Driver and ATI Nucleus. Source code for the VxWorks, Linux, Psos and an 8-bit version of the Packet Driver are available on the Cirrus Website. The VxWorks driver, in particular, is a good starting point for writing a custom driver in C. It is very well documented and very modular.

I/O Ports

In 8 bit mode the CS8900 is accessed through its eight 16 bit I/O ports.

Offset	Type	Description
0000h	Read/Write	Receive/Transmit Data (Port 0)
0002h	Read/Write	Receive/Transmit Data (Port 1)
0004h	Write-only	TxCMD (Transmit Command)
0006h	Write-only	TxLength (Transmit Length)
0008h	Read-only	Interrupt Status Queue
000Ah	Read/Write	PacketPage Pointer
000Ch	Read/Write	PacketPage Data (Port 0)
000Eh	Read/Write	PacketPage Data (Port 1)

Table 1. I/O Mode Mapping

In a non-ISA system these ports are usually memory mapped into standard system memory. Please note that the driver should read or write both bytes when accessing any CS8900 status or event register.

Frame Transmission

Transmission and reception of frames is done through these data ports. The basic steps in transmitting a frame are 1) bid for buffer space on the chip by writing the transmit command to the TxCMD port and the length to TxLength port then checking the BusSt register. 2) if space is available begin writing the data, a byte at a time, to Receive/Transmit data port 0. Refer to section ***4.10, I/O Space Operation*** of the data sheet for more details.

Cirrus Logic, Inc.
P.O. Box 17847, Austin, Texas 78760
(512) 445 7222 FAX: (512) 445 7581
http://www.crystal.com

For instance, the CS8900 is at its default I/O location of 300h. To transmit a frame that was 81 bytes in length the driver would first write the transmit command 00C0h (Start transmitting after all bytes transferred) to the TxCMD port. This is done by writing the low order byte, C0h, to 304h then writing the high order byte, 00h, to 305h. Next write 0051h (81 decimal) to the TxLENGTH port. Low byte, 51h, to 306h then high byte, 00h, to 307h.

Now check to see if the space was available. This is done by checking the BusST register, bit 8. Write 0138h to 30Ah (Packet Page Pointer) then read the word at 30Ch (Packet Page Data Port 0). If bit 8 (Rdy4TxNow) is set then you can start transferring data to Transmit Data Port 0. Do so in the following manner: write the first byte to 300h, the second byte to 301h, byte 3 to 300h, byte 4 to 301h and so on until the whole frame is written. The chip will automatically send the frame after the last byte is transferred.

Frame Reception

Reception is also straightforward. The host is notified of an incoming frame through an interrupt or the host can poll the chip to see when a frame is available. When the host is aware of an incoming frame the software should read the frame data following these steps (assuming I/O base 300h):

- read the interrupt status queue at 308h (read low order byte at 308, then high order byte at 309) to determine the event and type of frame.
- read the RxStatus word (same data as RxEvent, register) from data port 0. Read this high order byte 301h first, then low order byte 300h.
  Note: it is very important to read the RxStatus and RxLength high order byte first.
- read the RxLength word (the frame length) from data port 0. Read this high order byte 301h first, then low order byte 300h.
  Note: it is very important to read the RxStatus and RxLength high order byte first.
- begin reading the frame data 300h then 301h, 300h then 301h until the entire frame has been transferred to host memory.

Schematic and Layout Review Service

Prevent problems early in the design phase of your product. Have your schematic or layout reviewed free of charge by our experts before you build your board. Call Applications Engineering at (512) 442-7555 or send e-mail to ethernet@crystal.cirrus.com.

Unsupported functions in 8 bit mode

- The DMA engine only uses 16 bit memory accesses and does not support 8 bit transfers.
- The packet page pointer has an auto increment feature that cannot be used in 8 bit mode.
- An EEPROM is not supported. Most 8 bit designs should not require one and can eliminate the added cost.

Typical Connection Diagram