CIRRUS LOGIC, Inc., 1463 Centre Pointe Drive, Milpitas, CA 95035 Telephone 408.945.8300 Telex 171918 FAX 408.263.5682

CIRRUS LOGIC

APPLICATION NOTE
AN–CD2

CL-CD180 Interrupts

Data Communications Product Group
CIRRUS LOGIC, Inc.

August, 1988

Scope and Applicability

This application note explains the interrupt features of the CL-CD180. It will be useful to designers who need a more detailed description of those features and their interaction than is given in the CL-CD180 Data Sheet.

Related Documents

Readers are referred to the CL-CD180 Preliminary Data Sheet, June, 1988, as well as the application note AN-CD1, CL-CD180 Flow Control.

Note:

This document describes a potential application of CIRRUS LOGIC integrated circuits. No warrantee is given for the suitability of the circuitry or program code described herein for any purpose other than demonstrating functional operation. CIRRUS LOGIC believes this information is accurate and reliable. However, it is subject to change without notice. No responsibility is assumed by CIRRUS LOGIC for the use of this information; nor for infringements of patents or other rights of third parties. This document implies no license under patents or copyrights.

The architecture and interrupt schemes of the CL-CD180 are patent pending.

1. INTRODUCTION

The CD180 is an 8 channel asynchronous communications controller based on a Cirrus Logic proprietary, firmware programmed micro-controller engine. The power of this design approach in conjunction with some specific hardware innovations allows the device to provide exceptional data transfer performance in an interrupt driven environment. This application note explains the unique interrupt features of the CD180.

2. INTERRUPT VECTORING

The CD180 employs a system of vectored interrupts. It asserts interrupts to the host on any of its 3 interrupt request lines. Those interrupt requests convey requests for service to the host for enabled interrupt sources in the Receive, Transmit and Modem Signal Change interrupt groups. When the interrupt acknowledge signal is asserted to the CD180 by the host, it drives a modified interrupt vector onto the data bus. Bits 3 through 7 of the interrupt vector must be pre-defined by the host system (saved in the CD180 in the Global Interrupt Vector Register). Bits 0 and 1 of the vector provided by the CD180 identify the interrupt group number. Group 1 is the Modem signal change group and is coded 01, group 2 is for transmit interrupts and is coded 10. Group 3 is for receive interrupts and is coded 11. For Receive interrupts an additional bit, bit 2, indicates whether the interrupt is of the "good data" type or of the exception type. (More about that in Section 1.5.) Group interrupt code 00 is not used. Bit 2 of the interrupt vector is always 0 for Group 1 and Group 2 interrupt vectors.

When the host responds to a CD180 interrupt, it must convey to the CD180 which interrupt group is being acknowledged. It does this by placing an interrupt acknowledge code on the address bus while asserting the interrupt_acknowledge signal (IACKIN*). The interrupt acknowledge code must correspond to the one assigned for the interrupt group being acknowledged.

The CD180 contains a 3 entry content addressable memory. Each entry corresponds to one of the interrupt Groups. The user must initialize this CAM, the Priority Interrupt Level Registers, with the codes which will be present on the address bus when it acknowledges CD180 interrupts. Each PILR code must be unique. PILR bits 0 through 6 are compared with address bus bits 0 through 6 respectively. PILR bit 7 must be a 1. In applications using multiple CD180s in an interrupt daisy chain, each CD180 must be programmed with the same PILR codes. By recognizing that the address bus matches one of it's Priority Interrupt Level Registers when IACKIN* is asserted, the CD180 knows which of its interrupt types is being acknowledged.

If the address does not match one of the PILR codes, the assertion of IACKIN* has no effect. In systems employing more than one CD180, the IACKOUT* pin of one CD180 is connected to the IACKIN* pin of the next, forming an interrupt acknowledge daisy chain. The host drives the IACKIN* of the CD180 at the head of the chain. When a CD180 sees a valid interrupt acknowledgment for one of it's interrupt Groups, but does not have that interrupt asserted, it passes on the interrupt acknowledge by asserting IACKOUT*. The CD180 employs a fair share mechanism to provide equal interrupt service to all CD180s in a daisy chain. This is explained in Section 3.

3. SCANNING AND "FAIRNESS"

The CD180 scans its 8 full duplex channels for conditions which the host has enabled (on an individual channel basis) as active sources of interrupts. It performs 3 independent scans, one for each interrupt group. The CD180 doesn't know or assume the relative priorities to the host of its 3 interrupt groups. By scanning each group independently and asserting a group's interrupt on a separate interrupt request pin, the user may choose how to deal with the three interrupt groups. The CD180 supports concurrent interrupts on all three groups, and the host may nest them in any order.

The CD180 performs the interrupt scans sequentially by channel number. A channel scan pointer is maintained for each interrupt group. The pointer is advanced in a circular fashion through the 8 channels as the scans progress. This guarantees that each channel in the CD180 has an opportunity for interrupt service before any channel is serviced twice.

The CD180 is designed to work effectively in systems supporting more than 8 channels. Several CD180s may be used, all sharing 3 common interrupt request lines. (The interrupt request outputs are "open-drain" and require an external pullup resistor. A value of 4700 ohms is suggested.) The CD180 has both an interrupt acknowledge in and an interrupt acknowledge out, IACKIN* and IACKOUT*. These pins support a "fair share" interrupt acknowledge daisy chain.

In conventional daisy chains, the positional priority implicit in the order of connection of the devices in the chain is fixed and absolute. If multiple devices in the chain sharing a common interrupt request type or priority level jointly assert that interrupt type, the device closest to the head of the chain will always be the one accepting the acknowledgment and receiving service. It is possible (and in many instances likely) for a device or devices down the chain to receive no interrupt service as successive requests from devices near the head of the chain preempt service from those near the tail of the chain.

In contrast, CD180s maintain an interrupt request enable status or "fair" state. The common interrupt request outputs of the CD180s are active low or negative true. Any one device asserting the interrupt request output causes the common interrupt request line to go low. More than one device may pull the interrupt request line low, but if none of the devices is asserting its interrupt request output, the line is pulled high to its unasserted state by the external pullup resistor.

The "fair" state of each CD180 is maintained by the following algorithm. The CD180 observes the logic state of the shared interrupt request line. Whenever that line is observed to be unasserted, i.e. no CD180 is requesting interrupt service, the fair state is set true. This permits the assertion of an interrupt by the CD180. When that CD180 has cause to request interrupt service, it examines the fair state. If true, it asserts the interrupt. The fair state will be set to false when the interrupt is asserted by the CD180. In the interval from the time the fair state is set true through the time the CD180s interrupt request is acknowledged, one or more other CD180s may independently assert the same interrupt. When the host system acknowledges the interrupt, its acknowledge signal is applied to the first CD180 in the chain. If that CD180 has the interrupt asserted, it accepts the acknowledgement, de-asserting its interrupt request. If not, it passes it on to the next CD180 in the chain.

Now, if more than one CD180 is asserting the interrupt request, the interrupt request line will remain in its active state. The CD180 having received service may not request service again because its fair state is false and will remain false until the shared interrupt request line is observed to be unasserted. When the host completes the servicing of the current interrupt, it will issue another acknowledgment. The acknowledgment is passed through the daisy chain to the CD180 closest to the top of the chain having its interrupt asserted. This process repeats as many times as necessary, until all CD180s requesting service have been serviced, whereupon the interrupt request line goes to its unasserted state, and all CD180s will set their "fair" state to true. This will allow them to assert new interrupts initiating another round of interrupt service.

The fairness mechanism is duplicated for each interrupt group.

4. CONTEXT ORIENTED ADDRESSING AND INTERRUPTS

The large number of internal registers in the CD180 are accessed in one of two addressing modes. These modes are global and channel oriented. Address input A6 determines the mode. If it is 1, the mode is a global address, if it is a 0 the address is a channel oriented address. Global addresses access registers and functions which are not specific to any one channel. Channel oriented addresses are used for data and functions specific to a channel. A channel number is supplied by the CD180 during the channel oriented access. There are several internal sources for the channel number. The source which supplies the channel number is determined by the current context. There are three different interrupt contexts (one for each interrupt group) in addition to the background context. Contexts are nested as interrupts nest. When an interrupt is acknowledged the CD180 pushes the previous context onto an internal stack and enters a new context for the acknowledged interrupt. When an interrupt service is concluded by the mandatory write to the End Of Interrupt Register (a dummy register which exists to signal the end of the interrupt context), the context stack is popped, restoring the previous context.

Within an interrupt context, the CD180 supplies the channel number for all channel oriented accesses. That channel number was set up by the interrupt scanner when it identified an enabled interrupt condition for that channel. The user may not modify the channel number used for channel accesses while within an interrupt service context. The channel number used in the context is provided to the host as data in the Global Interrupting Channel Register.

5. TRANSMIT AND RECEIVE FIFO's

The CD180 has three 8 byte FIFO's per channel. There is one FIFO each for Transmit Data, Receive Data and Receive Status. Each data character received has it's own unique status FIFO entry.

In addition to the FIFO's, there is a holding register and a shift register for each channel for both transmit and receive.

Within the interrupt context for a transmit interrupt, data is moved to the transmit FIFO for the interrupting channel by successive writes to the global Transmit Data Register. The transmit FIFO is always empty when a transmit interrupt is asserted, so the user may transfer up to 8 bytes of data during a transmit interrupt. The user needn't transfer any data and may disable further transmit interrupts from that channel when responding to the interrupt. The Transmit Data Register must be written to only during transmit interrupt service.

Within the interrupt context for a receive interrupt, data is moved from the receive FIFO of the interrupting channel by successive reads from the global Receive Data Register. If the character status must be read (during a receive exception interrupt) it is obtained by reading the global Receive Character Status Register. As with the TDR, use of the RDR and RCSR are limited to the receive interrupt context.

6. RECEIVE INTERRUPTS AND "GOOD DATA"

The CD180 improves system performance in receiving data by removing 2 levels of character processing overhead from the host. It eliminates the need to examine the arriving characters to detect in band flow control requests by the remote devices connected to the serial channels. It does this by recognizing the pre-assigned flow control characters (or character sequences) and implementing the flow control function itself. (This is fully described in Cirrus Logic Application Note AN-CD1).

It also examines the arriving data and manages it as a queue of data with exception conditions, data matching special characters, and "good" data. Characters with an exception status and recognized special characters are transfered to the host by the Receive Exception interrupt. A unique vector informs the host of this case. It may be sufficient, when servicing a Receive Exception interrupt, to read only the interrupting channel number and the character status. The receive exception interrupt is a one character interrupt.

When the FIFO is headed by a contiguous string of characters of the good data type, the receive interrupt will be of the Good Data type. The number of contiguous bytes of good data waiting in the FIFO is contained in the Receive Data Count Register. The host may simply read the Receive Data Register repeatedly to transfer the number of good characters available.

The host may choose whether or not the 4 user defined special characters are to be recognized and reported by the exception interrupt type. The receive FIFO threshold is programmable allowing control over the amount of time the host has to service a given interrupt without incurring a receiver overrun. Additional characters received beyond the Receive Threshold are included in the Receive Data Count if they add to an outstanding Good Data Interrupt. In this way, maximum use is made of the Receive FIFO depth.

7. HARDWARE INTERFACE

The hardware interrupt interface consists of 5 pins. Three are the open-drain int IREQ2* and IREQ1*. The Interrupt acknowledge input and output pins . complete the interface. These are all negative true or active low signals. The us pullup resistors on the IREQ signals. A value of 4700 ohms is suggested.

8. MULTIPLE CD180s

Multiple CD180s are designed to share three interrupt request lines and have corresponding IREQ signa connected in parallel. The user will provide an interrupt acknowledge signal for the CD180 interrupts and connect this signal to the IACKIN* pin of one of the CD180s. Additional CD180s will be connected daisy chain fashion. The IACKOUT* of the first CD180 will be connected to the IACKIN* of the next. The IACKOUT* of the last CD180 in the chain is left unconnected.