CIRRUS LOGIC®

FEATURES

- **Eight full-duplex asynchronous channels supporting data rates up to 64 kbps**
- **Register-based interrupt acknowledges eliminate need for separate interrupt acknowledge signals**
- **Automatic prioritization scheme allows device to respond to an interrupt acknowledge with the highest internal interrupt pending (host-programmable)**
- **Sophisticated interrupt schemes**
  - — Vectored Interrupts
  - — *Fair Share* ™ Interrupts
  - — *Good Data* ™ Interrupts for improved throughput
  - — Simultaneous interrupt requests for three classes of interrupts: Rx, Tx, and modem state changes
- **Independent baud-rate generators for each channel/direction**
- **Improved host/controller software interface**
- **Generation and detection of special characters**
- **On-chip flow control**
  - — In-band (Xon, Xoff generation and detection)
  - — Out-of-band (DTR/DSR or RTS/CTS)
- **On-chip FIFO — 8 bytes each for Rx, Tx, and Status**
- **Line break detection and generation**
- **Multiple-chip daisy-chain cascading feature**
- **Odd, even, forced, or no parity**
- **Four modem/general-purpose I/O signals per channel**
- **System clock up to 12.5 MHz**
- **CMOS technology in 84-pin PLCC**

Intelligent Octal-Channel Asynchronous Communications Controller

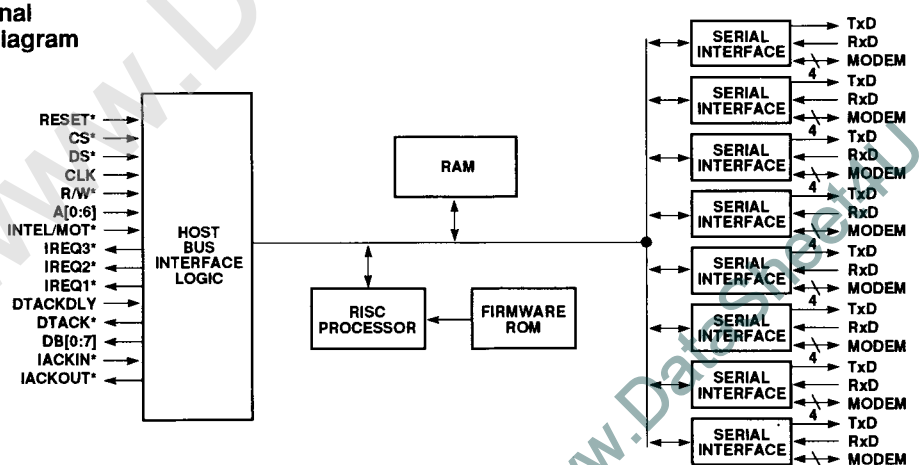
OVERVIEW

The CL-CD180 is an I/O controller capable of controlling eight full-duplex channels transferring data at rates up to 64 kbps. The advantage of the CL-CD180 lies in its ability to move data efficiently from the serial channels to the host. This results in an order-of-magnitude improvement in system-level throughput and a reduction in overhead on the host CPU.

To increase the overall data throughput of the system, the chip relies on a combination of features. Most important are the buffers for transmit and receive data. Each serial channel has three 8-byte FIFOs — one each for transmit, receive, and receive exception status. The Receive FIFOs have programmable thresholds to minimize interrupt latency requirements.

(cont. next page)

Functional Block Diagram

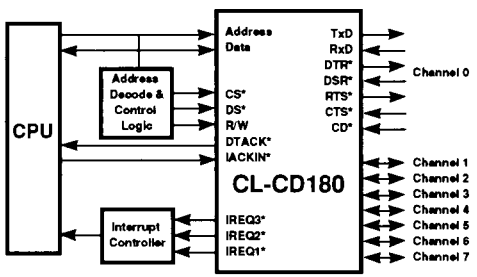


Before beginning any new design with this device, please contact Cirrus Logic, Inc., for the latest errata information. See the back cover of this document for sales office locations and phone numbers. This data sheet applies to Revision 'C' or later devices.

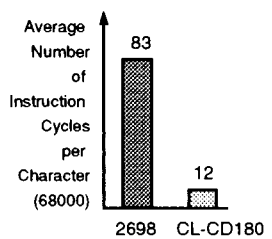
OVERVIEW (cont.)

The CL-CD180 is based on a high-performance proprietary RISC processor architecture developed by Cirrus Logic specifically for data communication applications. This processor executes all instructions in one clock cycle, and uses a register window architecture to ensure zero-overhead context switch for each type of internal interrupt.

The CL-CD180 is fabricated in an advanced CMOS process. The chip's high throughput, low power consumption, and high level of integration permit system designs with minimum parts count, maximum performance, and maximum reliability.



Typical CL-CD180 Host CPU Interface



CL-CD180 Performance

CL-CD180 Features/Benefits

Unique Features	Benefits
■ ***Three 8-byte FIFOs per channel***	Greatly reduces real-time interrupt response time requirement of the host CPU. Simplifies system tasks in a real-time multi-tasking environment.
■ ***User-programmable receive FIFO interrupt threshold***	Enables tailoring of interrupt conditions to different system requirements. Speeds software development.
■ ***Data Interrupt for transferring multiple bytes of data***	Reduces host time required to transfer data and significantly improves system performance. 'Frees-up' bandwidth for host to perform higher-level system tasks.
■ ***Interrupt vectoring by device ID and type of service required***	Permits direct jump into proper interrupt service routine, improving overall system performance.
■ ***Automatic flow control***	Real-time control of data flow reduces risk of losing valuable data.

CIRRUS LOGIC

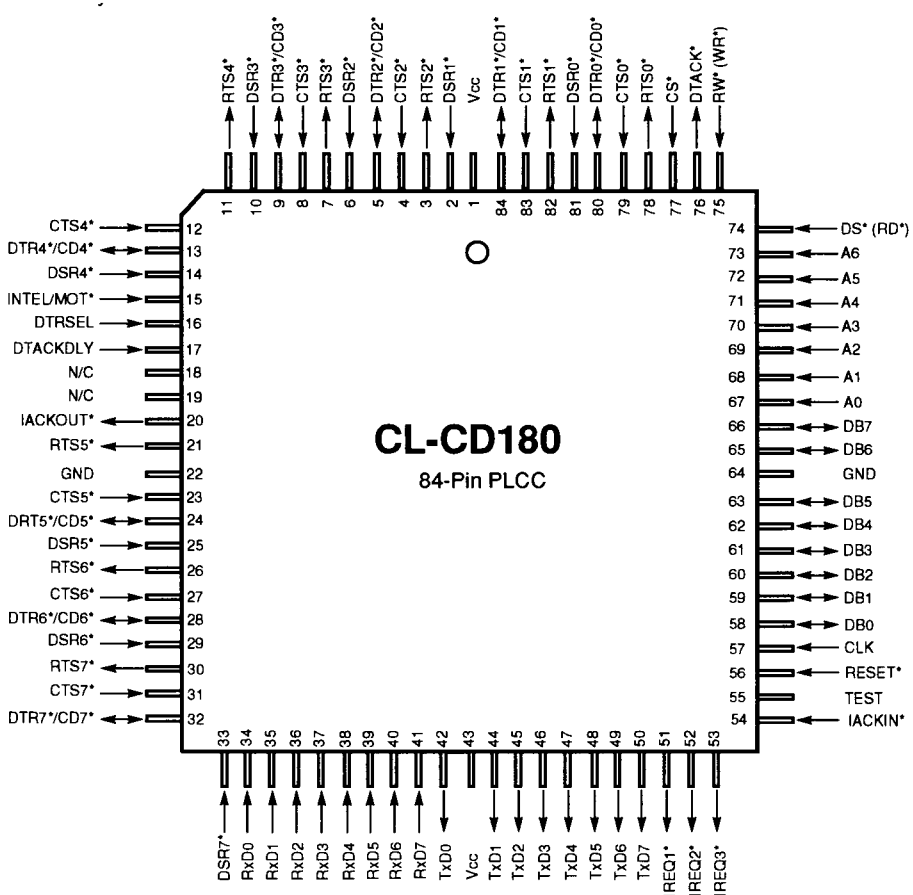
Table of Contents

Table of Contents *(cont.)*

1. PIN INFORMATION

The CL-CD180 is available in a 84-pin plastic leaded chip carrier (PLCC) device configuration, shown below.

1.1 Pin Diagram



1.2 Pin Assignments

The following conventions are used in the table below: (*) denotes an active-low signal; I = Input, I/O = Input/Output, O = Output, and OD = Open Drain.

Symbol	Pin #	# of Pins	Type
General			
CLK	57	1	I
RESET*	56	1	I
Microprocessor Interface			
A[0:6]	67:73	7	I
DB[0:7]	58:63,65:66	8	I/O
CS*	77	1	I
DS* (RD*)	74	1	I
R/W* (WR*)	75	1	I
DTACK*	76	1	OD
DTACKDLY	17	1	I
INTEL/MOT*	15	1	I
Service Request Interface			
IACKIN*	54	1	I
IACKOUT*	20	1	O
IREQ3*	53	1	OD
IREQ2*	52	1	OD
IREQ1*	51	1	OD

NOTE: Both the CLK and RESET* Pins have a V_{IH} specification of 2.7 volts. A 1-KΩ pullup, or use of a driver of logic families that are specified by their manufacturers as providing a V_{OH} of at least 3.0 volts (such as advanced CMOS, advanced Schottky, or others) when driving a micro-amp load, is recommended.

Symbol	Pin #	# of Pins	Type
Communications Interface			
RxD[0:7]	34:41	8	I
TxD[0:7]	42,44:50	8	O
DSR[0:7]*	81,2,6,10, 14,25,29,33	8	I
DTR[0:7]*/ CD[0:7]	80,84,5,9, 13,24,28,32	8	I/O
CTS[0:7]*	79,83,4,8, 12,23,27,31	8	I
RTS[0:7]*	78,82,3,7, 11,21,26,30	8	O
DTRSEL	16	1	I
Miscellaneous			
N/C	18,19	2	–
V_{CC}	1,43	2	–
GND	22,64	2	–
Test	55	1	–

1.3 Pin Descriptions

Symbol	Number	Type	Description
General			
CLK	57	I	**SYSTEM CLOCK:** Input for 1x clock signal.
RESET*	56	I	**RESET:** Resets CL-CD180. All internal registers are cleared or initialized.
Microprocessor Interface			
A[0:6]	67:73	I	**ADDRESS:** Address inputs, used to select the various internal registers of the CL-CD180.
DB[0:7]	58:63,65:66	I/O	**DATA BUS I/O.**
CS*	77	I	**CHIP SELECT:** Must be low for all reads and writes to the CL-CD180, but not for service acknowledgment cycles. Must never be low when IACKIN* is low.
DS* (RD*)	74	I	**DATA STROBE:** When the INTEL/MOT* Pin is low, this signal is used to control access to the CL-CD180, and data hold time on the bus. When the INTEL/MOT* Pin is high, this pin performs the same function for read cycles and service acknowledgment cycles.
R/W* (WR*)	75	I	**READ/WRITE:** When the INTEL/MOT* Pin is low, this signal controls whether the current bus cycle is a read or a write. When the INTEL/MOT* Pin is high, this signal strobes data into the CL-CD180 on write cycles only.
DTACK*	76	OD	**DATA TRANSFER ACKNOWLEDGE:** Open-drain output. This signal indicates the completion of an internal bus cycle within the CL-CD180. It can be used to insert wait states by the host. Note that the bus cycles are of fixed length, and if the bus interface is correctly designed, DTACK* is not required to insert wait states.
DTACKDLY	17	I	**DTACK DELAY:** Controls the time of assertion of DTACK to allow 'fine tuning' of the number of wait states inserted. When low, DTACK asserts earlier than when high.

1.3 Pin Descriptions *(cont.)*

Symbol	Number	Type	Description
INTEL/MOT*	15	I	**INTEL/MOT*:** Selects either of two bus-handshake styles. When low, DS* acts as Data Strobe, and the R/W* Pin acts as Read/Write. When INTEL/MOT* is high, the two pins act as RD* Strobe and WR* Strobe. INTEL/MOT* does not affect the timing of the bus interface, only the logical meaning of these two pins. INTEL/MOT* may be tied either high or low, but should not be changed during regular operation.
Service Request Interface			
IACKIN*	54	I	**ACKNOWLEDGMENT (SERVICE) INPUT:** Must be low only during service acknowledge bus cycles. Must never be low when CS* is low.
IACKOUT*	20	O	**ACKNOWLEDGMENT (SERVICE) OUTPUT:** Goes low whenever the CL-CD180 recognizes a valid acknowledgment is occurring (either hardware- or register-based) that is not for the CL-CD180. In daisy-chain applications, the IACKOUT* should be connected to the IACKIN* of the next CL-CD180.
IREQ3*	53	OD	**RECEIVE REQUEST OUTPUT:** Asserts whenever the CL-CD180 has a receive condition requiring service. Negates whenever a service acknowledgment of the receive type occurs.
IREQ2*	52	OD	**TRANSMIT REQUEST OUTPUT:** Asserts whenever the CL-CD180 has a transmit condition requiring service. Negates whenever a service acknowledgment of the transmit type occurs.
IREQ1*	51	OD	**MODEM REQUEST OUTPUT:** Asserts whenever the CL-CD180 has a modem signal change condition requiring service. Negates whenever a service acknowledgment of the modem signal change type occurs.
Communications Interface			
RxD[0:7]	34:41	I	**RECEIVED DATA INPUTS.**
TxD[0:7]	42,44:50	O	**TRANSMITTED DATA OUTPUTS.**

1.3 Pin Descriptions *(cont.)*

Symbol	Number	Type	Description
NOTE: The following 'modem control' signals are named arbitrarily. The CD* Signal is a general-purpose input. The DSR* and DTR* Signals can be used by the CL-CD180 receiver for handshake or flow control, or may be used as general-purpose inputs and outputs. The RTS* and CTS* Signals can be used by the CL-CD180 transmitter as handshake or flow control, or may be used as general-purpose inputs and outputs. In all cases, the CL-CD180 can be programmed to generate interrupts whenever the input pins change state in a specified direction.			
DSR*[0:7]	81,2,6,10, 14,25,29,33	I	**DATA SET READY INPUTS:** May be used to control the Receive Shift Register for flow-control purposes, or may be used as general-purpose inputs.
DTR*[0:7]/ CD*[0:7]	80,84,5,9, 13,24,28,32	I/O	**DATA TERMINAL READY /CARRIER DETECT:** Depending on the state of the DTRSEL input, these pins are either DTR outputs or CD inputs. When selected as DTR (DTRSEL = high), and if enabled by MCOR1, they are used by the receiver to indicate that the Receive FIFO has exceeded a user-defined threshold; in other words, as a signal to flow-control the remote sender. These pins may also be used as general-purpose outputs. When selected as CD (DTRSEL = low), they become CD inputs. They can also be used as general-purpose inputs.
CTS*[0:7]	79,83,4,8, 12,23,27,31	I	**CLEAR-TO-SEND INPUTS:** Used by the transmitter as a permission-to-send, or may be used as general-purpose inputs.
RTS*[0:7]	78,82,3,7, 11,21,26,30	O	**REQUEST-TO-SEND OUTPUTS:** Used by the transmitter to indicate that there is data to be sent. May be used as general-purpose outputs.
DTRSEL	16	I	**DTR SELECT:** This input sets the mode for the DTR*/CD* pins. When DTRSEL is high, the DTR*/CD* pins implement the DTR* output; when low, the DTR*/CD* Pins become CD* inputs.
Miscellaneous			
N/C	18,19	–	**NO CONNECT:** Make no connections to these pins.
V_{CC}	1,43	–	**+5V.**
GND	22,64	–	**GROUND.**
TEST	55	–	**TEST:** This is a test pin and should be connected to ground.

2. FUNCTIONAL DESCRIPTION

2.1 Introduction

The CL-CD180 I/O coprocessor controls eight full-duplex channels that transfer data at rates up to 64 kbps. The CL-CD180 moves data efficiently between the serial channels and the host, resulting in a great improvement in system-level throughput and a reduction in overhead on the host CPU. This improvement is obtained by reducing the number of service requests (interrupts) the host must respond to and reducing the complexity and time required to handle each service request.

The CL-CD180 relies on a combination of features to achieve reduction in the number and complexity of service requests. Most important are the buffers for transmit and receive data. Each serial channel has three 8-byte FIFOs — one each for transmit, receive, and receive-exception status. The Receive FIFOs have programmable thresholds to minimize interrupt latency requirements. The vectored service requests and the Good Data™ Interrupt allow the host system to immediately transfer data upon beginning processing of a service request, without tedious checking of flags and error conditions.

The CL-CD180 is based on a high-performance, proprietary RISC processor architecture developed by Cirrus Logic specifically for data communications applications. The CL-CD180 processor executes all instructions in one clock cycle, and it uses a register window architecture to ensure zero-overhead context switch for each type of internal interrupt. The instruction set of this processor is optimized for bit-oriented tasks that, combined with instantaneous response to sending or receiving one bit, allow highly efficient processing of characters. All firmware for the CL-CD180 processor is contained in an on-chip ROM, and requires no user programming.

The CL-CD180 processor is assisted in its task by specialized peripheral logic. Serial data transmission and reception is handled by 'bit engines'. Each channel has a bit engine for transmitting and another for receiving. While each engine handles all bit-level timing, bit-to-character assembly is done in firmware. Bits are passed to the CL-CD180 processor by internal interrupts over a special bus dedicated to this purpose. Special internal-interrupt context hardware reduces overhead on internal interrupts to zero by pointing to the correct register window for every possible context, and a unique Global Index Register eliminates address calculations by always pointing to the current channel. External service requests to the host system are also hardware-assisted. There is a queue for each of the three classes of external service requests, and the request/acknowledgment mechanism is entirely in hardware to minimize response time.

The CL-CD180 processor assembles bits into characters, checks parity and formatting parameters, and stores the data in the FIFOs as required. FIFOs are maintained as RAM-based structures, and both the local CL-CD180 processor and the host access them via Pointer Registers by an Indexed Addressing Mode.

The CL-CD180 communicates with the host via service requests and service acknowledgments. Service requests can be handled either as interrupts or by polling. Regardless of the method used, the CL-CD180 has features to minimize both the number of requests to be serviced and the time required to service them. The number of service requests is reduced by the FIFOs since a service request is required only every eight characters. To reduce the time required per request, the CL-CD180 supplies separate vectors for four different types of service requests. This reduces the time required by the host CPU to determine what action to take. For example, there is a unique vector for Good Data so that the host wastes no time checking status bits for error conditions. If there is an error condition, the CL-CD180 supplies a unique vector pointing to the error-handling routine. Other vectors report transmit status and modem signal change.

Service requests to the host system are implemented on the CL-CD180 by three hardware service request state machines. Each machine has the ability to 'queue-up' multiple requests. The state machines are designed to offer the fastest response possible. Whenever the CL-CD180 processor determines that a condition needs a service request, it queues the request with the appropriate state machine. The state machine posts the external request, monitors acknowledgment

cycles from the host, and informs the CL-CD180 processor when a valid service acknowledgment has been completely serviced. This allows the CL-CD180 to correctly maintain the internal context for processing the channel being serviced.

Because the CL-CD180 processor processes every character sent or received, features such as Automatic Flow Control and Special Character Recognition are easily implemented. This reduces the processing burden on the host system. Both In-Band (Xon, Xoff) and Out-of-Band (RTS/CTS, DTR/DSR) Flow Control Modes are supported. For In-Band Flow Control, the CL-CD180 automatically starts and stops its transmitter when the remote unit sends flow-control characters. The CL-CD180 makes it easy for the local host to flow-control the remote via the 'Send Special Character' commands. For Out-of-Band Flow Control, the transmitter will optionally assert RTS and monitor CTS for permission to send, and assert/negate DTR when the Receive FIFO reaches a user-definable threshold. DSR may be used to gate the receiver on and off. Together, the In-Band and Out-of-Band features allow the data flow to be controlled in real time with minimum or no host intervention, and they also prevent loss of data.

Systems with multiple CL-CD180s are easily implemented, with no external glue, via a daisy-chain scheme. A Fair Share feature ensures equal access for all service requests, both within one CL-CD180 and across multiple devices. Alternately, multiple CL-CD180s may be operated in parallel as independent devices.

Serial channels on the CL-CD180 are entirely independent of one another. Any channel may be programmed to a combination of features regardless of the state of other channels. Bit-rate generators are programmed by loading a divisor value, so the transmitters and receivers can each operate at any standard or non-standard data rate.

The CL-CD180 can detect the received line-break condition, send break characters of any length, and transmit delays. This is done via transmit commands embedded in the Transmit Data Stream. The CL-CD180 can also be programmed to detect user-defined special characters and generate a special service request to the host. Parity checking is performed automatically, but can be overridden by the host to force parity errors for test purposes. Character length and Stop Bit length are also programmable per-channel.

Modem pins on the CL-CD180 are general-purpose, i.e., they are not hard-wired into the UART functions. If modem pins are not needed to interface to actual modems, they can be used as general-purpose I/O pins. In either case they are readable and writable directly by the host system. In addition, the CL-CD180 can be programmed to monitor levels on modem input pins and generate service requests to the host upon detecting a specified change.

The CL-CD180 is fabricated in an advanced CMOS process. Its high throughput, low-power consumption, and high level of integration permits system designs with minimum parts count, maximum performance, and greater reliability.

There is a significant difference between the CL-CD180 and conventional dumb UARTs; the CL-CD180 is more efficient and is truly intelligent, even when operating in a polled environment. Systems built with the CL-CD180 interface between the host and the I/O device at a higher level than systems built with conventional UARTs. For example, with a dumb UART, the host must test each channel for presence of data, a process that is time-consuming. With the CL-CD180, the host queries the entire serial I/O subsystem for the presence of data. If data is present, the CL-CD180 determines which channel it is on, and whether it is good or erroneous. Thus, using the CL-CD180, the host-peripheral interface is easier to implement, faster, and more efficient.

2.2 Internal Operation

The internal architecture of the CL-CD180 is shown in Figure 2–1. The foundation of the design is a custom-designed CPU that Cirrus Logic has developed especially for this application. This CPU is optimized for bit-oriented tasks associated with UART functions, and it has a set of registers for each channel, arranged in a register window architecture. These registers and the ALU are eight bits wide. The CL-CD180 processor has a 16-bit instruction word that it retrieves from an on-chip ROM. Every instruction is one word long and executed in one-clock cycle.

Whenever an internal interrupt occurs (from a bit engine), the CL-CD180 processor automatically switches context to that channel's block of registers. No time is lost in saving any machine state. The CL-CD180 processor executes the instructions necessary to handle that bit (typically three to six instructions) and then returns to the context it was in prior to the internal interrupt. All internal interrupts are at the same priority level; the interrupt handler block ensures Fair Share access across channels.

Each channel's serial interface logic consists of a Receive-bit Engine, a Transmit-bit Engine, a Receive-baud-rate Generator, a Transmit-baud-rate Generator, and a Timer. The Receive-bit Engine samples the state of the RxD Pin at the time indicated by the Receive-baud-rate Generator, and it reports this value to the CL-CD180 processor as an interrupt. The Transmit-bit Engine works in a similar manner, but with a slight difference. At the baud rate tick, it outputs the next bit and generates an interrupt to the CL-CD180 processor requesting the following bit.

The baud-rate generators are 16-bit dividers that operate from a master clock, which is the system clock divided by 16. All baud-rate generators are independent, so a channel can send and receive at any speed. In addition to the baud-rate generators, there are two channel timers for each channel. One is an 8-bit divider, operating off the master prescaler timer tick. This timer is used to time-out partially full FIFOs to avoid 'stale' data. The other is used to time embedded delays in the Transmit Data Stream.

All eight channels are continuously scanned by internal logic that generates interrupts to the CL-CD180 processor in a 'fair' manner. This Fair Share Interrupt feature is the same as the mechanism used to share service requests across multiple devices. Whenever two or more channels are contending for interrupt service, the channel that is serviced first will not assert again until all other currently pending channels have been serviced. This prevents a fast, 64-kbps channel from 'hogging' service from a slow 1200-bps channel, yet it allows the faster channel the additional service it needs to support its higher speed. This allows more overall throughput than a 'round-robin' or an 'equal-access' method would provide.

Service requests for the host are handled by fast, dedicated logic on each of the three levels provided. Whenever the CL-CD180 processor detects a condition requiring external-host service, it queues the request with the service-request machine for that level. This machine asserts the External Request Pin, and it watches for a service acknowledgment of the same level. When a service acknowledgment is sensed, the machine automatically provides the vector to the host and sets up the internal context of the CL-CD180 for service. Upon completion of the service, the machine restores the normal context. The queue for service requests is two deep, so in a busy system there can be another request immediately pending when the first one is completed. This method avoids any delay between requests, and improves overall efficiency.

Modem I/O signals are implemented as 'conventional' input-output circuits, readable, and writable by either the on-chip or the host CPU. This allows maximum flexibility in using these signals either in the conventional way, or for any other I/O function desired. When the CL-CD180 processor is using these pins to implement flow-control functions, it reads them under software control and implements the function that way. There is no direct hardware association between the modem pins and the serial I/O hardware.

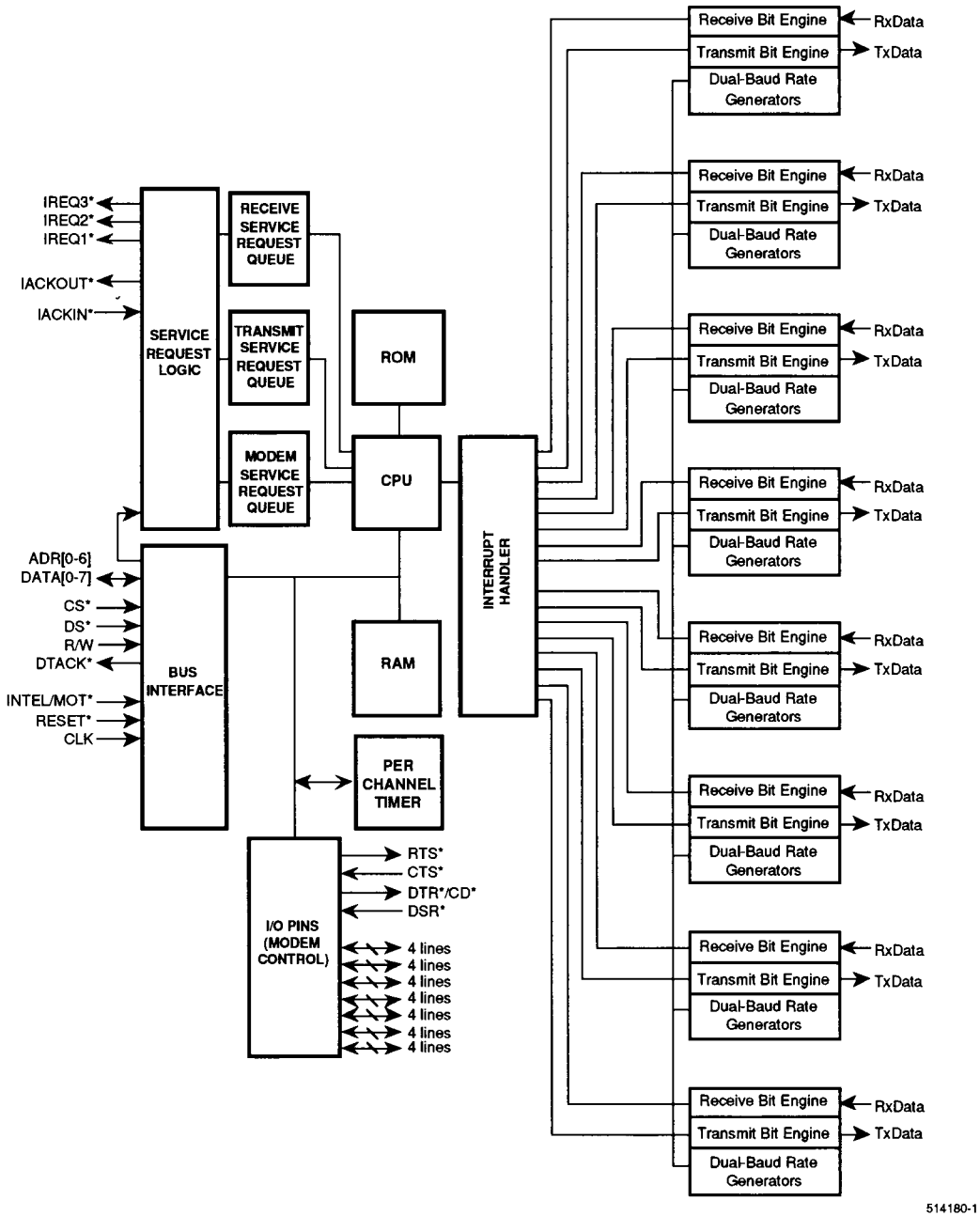


Figure 2–1. Internal Block Diagram

The CL-CD180 workload can be divided into two categories:

- Bit-to-character conversion (and vice versa) — the 'traditional' UART function.
- Character-level processing such as flow control, FIFO management, and host interface functions.

The CL-CD180 internal processor handles all these tasks in firmware. A foreground/background scheme is used: foreground for internal bit-engine interrupts and background for everything else. This internal structure represented in Figure 2–2, shows how the foreground communicates with the background. Foreground code handles bit-to-character assembly for receive, and character-to-bit disassembly for transmit. In either case a Holding Register, together with a Full/Empty Bit, acts as the 'gateway' between the interrupt-driven foreground and the polling-loop background code.

The background code executes the polling loop as shown in Figure 2–3. After power-on reset, the software runs continuously in an inner and an outer loop. Lower-priority tasks are handled in the outer loop, and higher-priority tasks are handled in the inner loop. The highest-priority tasks are bit events, which are handled by foreground (i.e., interrupt-driven) code.

The inner loop executes eight times as often as the outer loop. It checks each channel's full/empty bits to sense if another character needs to be moved. It first checks receive, and if there is a character to be moved, it is moved and execution moves on to the next channel. If receive data needs no processing, then transmit is checked. This mechanism gives a slightly higher priority to receive than to transmit; and it is desirable because missing a receive character is a fatal error and being late in transmitting one is not an error. (The effect of this may be observed by programming the CL-CD180 for higher-than-rated serial baud rates and providing a source of receive traffic with virtually 100-percent loading. As the CL-CD180 is heavily loaded, it will leave short gaps between transmit characters because the firmware is following the 'receive' path through the code. Refer to Section 3.2.3 for details on maximum performance and maximum line speed).

After eight passes through the inner loop (i.e., checking all eight channels for data), one pass is made through the outer loop. This pass checks one channel for host commands (such as 'Send Special Character'), timer functions, and a condition that requires posting an external service request (e.g., Receive FIFO full, Transmit FIFO empty, modem signal change, etc.). If required, the firmware posts the service request within the queue of the appropriate service-request logic. It then continues normal operation, until the host responds to the service request. After a single pass through the outer loop, eight passes through the inner loop are again made.

In most cases the CL-CD180 checks the appropriate bit in RAM to determine which options are enabled and modifies its processing accordingly. Some control bits must be interpreted and moved by CL-CD180 firmware from their location in option bit registers to other locations in the chip. Therefore, the host must notify the CL-CD180 when these bits are modified. The CL-CD180 will then alter the channel as commanded. Refer to Section 4.2 for details on channel command functions.

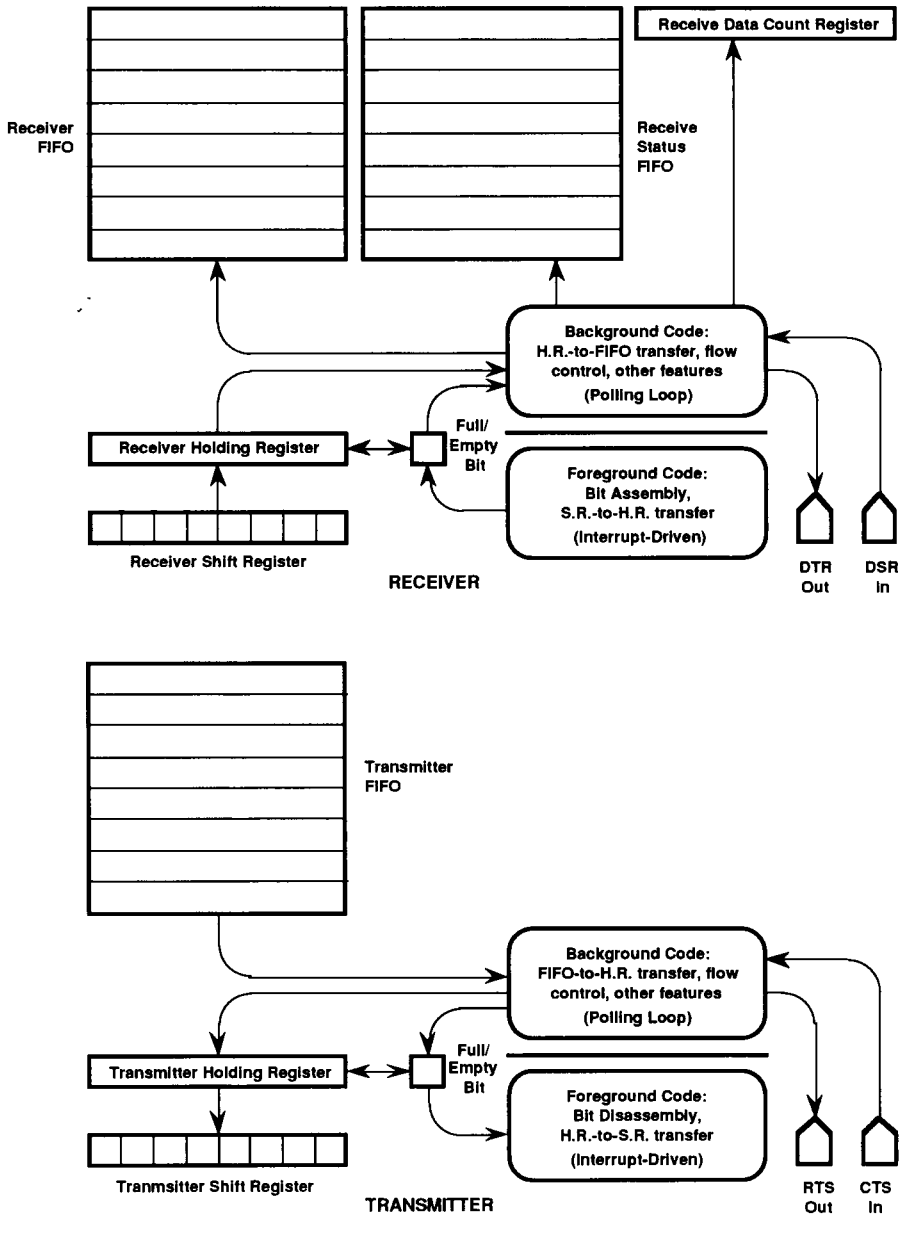
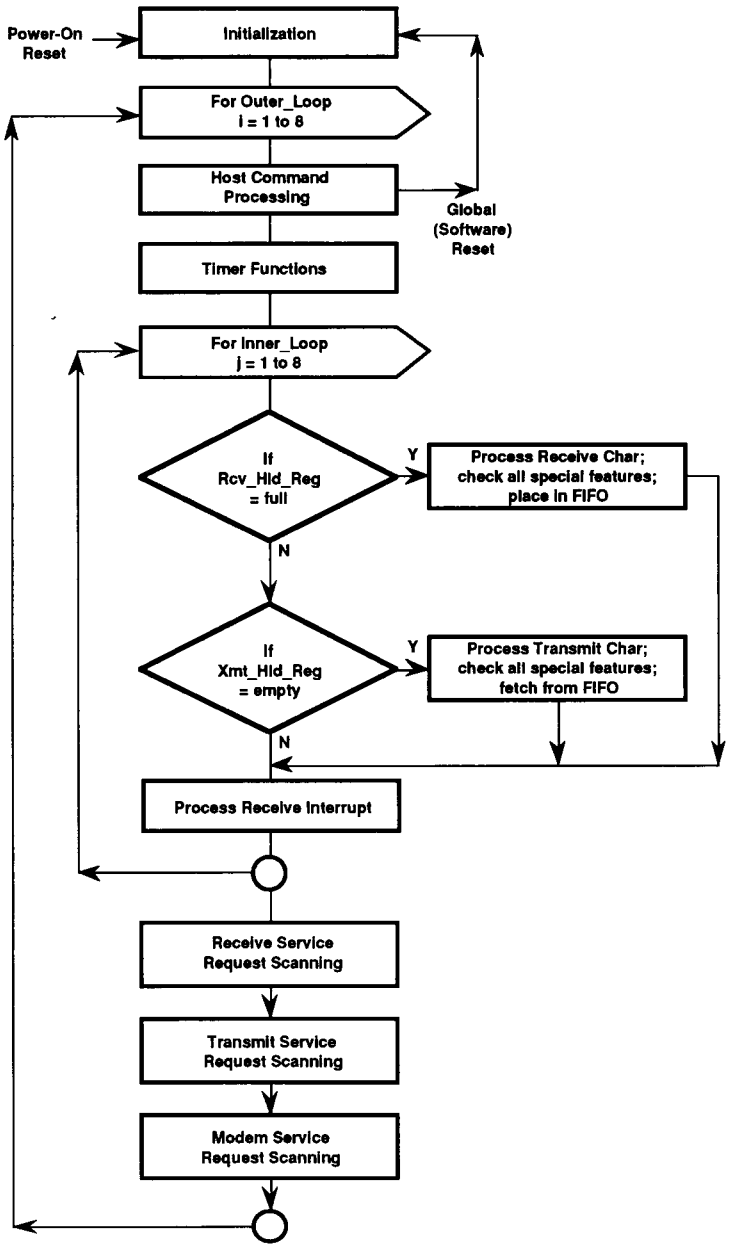


Figure 2–2. Foreground/Background Internal Structure



514180-3

Figure 2–3. Internal Operation Flow Chart

2.3 Service Request And Interrupt Operation

The CL-CD180 enhances design efficiency because it is an intelligent device that more closely resembles an add-in controller board than a mere collection of TTL. Conventional UARTs are basically passive, 'dumb' logic. For example, when polling a device for channels requiring service, each channel is not individually tested. Because of this, certain restrictions are placed on when and how FIFOs are accessed. The CL-CD180 processor must determine what the host is doing, and when to manage the queue of events correctly and efficiently.

NOTE: Any revision after B of the CL-CD180 has had additional features added to enhance its usefulness. These new features relate to interrupt processing and are controlled by two new registers: the Service Request Control Register (SRCR) and the Service Request Status Register (SRSR). Please see the descriptions in this section and the register descriptions in Section 6 for complete details.

Interrupt-Driven Versus Polled

Choosing the software interface, interrupt-driven versus polled, is critical to overall system performance. This choice also affects how the software is written. In hardware implementation, a programmer has a choice of Mixed Mode, i.e., when to poll versus when to be interrupt-driven. Mixed-mode Operation allows a programmer to optimize the efficiency of the system according to changing needs. The advantages of each method are discussed in Section 2.5.

2.3.1 Theory of Operation

The CL-CD180 has three independent service request levels, one for each of the three categories — Receive, Transmit, and Modem Signal Change. The priority of these lines is not fixed, but may be determined in one of the following three ways:

- It may be set within the CL-CD180 by the AutoPriority Option Bits.
- A system designer may assign priorities by the manner in which the three service request lines are connected to the host interrupt controller.
- Under software control, the host system may define and redefine the order of service requests.

The Service Request Interface to the host is implemented with five signals — IREQ1*, IREQ2*, IREQ3*, IACKIN*, and IACKOUT*. IREQ1*, IREQ2*, and IREQ3* are asserted when a service request is pending; IACKIN* is asserted during service-acknowledgment cycles; and IACKOUT* is used in multiple–CL-CD180 designs to share service requests and daisy-chain acknowledgments.

Whenever the CL-CD180 processor determines that one or more channels need service from the host, it loads the appropriate service-request state machine with the information about the type of request. The service-request state machine for that level will then assert its request signal. Note that all three request signals can be active at the same time. At this point, the CL-CD180 has not determined which request should be handled first — it simply asserts any and all lines, as required by the status of various channels. (This is true even if the AutoPri Option is enabled; AutoPri takes effect when a service request is acknowledged, and at that time the CL-CD180 determines which is the most important request.)

The host, after noticing that one or more of the three service request pins are active — either because the host was interrupted or it polled an external or internal CL-CD180 status register — decides which of the requests (if more than one is active) it will service first. The host begins the service operation by issuing a Service Acknowledge Cycle. The purpose of this cycle is to cause the CL-CD180 to set up its internal state for that type of request. (Note that if AutoPri is set, the host need not determine which level of service request to acknowledge; it simply acknowledges the CL-CD180 request and the CL-CD180 will return the vector for the highest-priority active request.)

If AutoPri is not being used, the CL-CD180 needs to be informed which one of the three possible pending requests the host wants to acknowledge. There are two different ways CL-CD180 can be informed of this — hardware and software.

The hardware method is based on the value in the address bus. The CL-CD180 determines the type of request being acknowledged by the value placed in the address bus during the acknowledge cycle. This is the method used by Motorola®-family processors. The host places the level of interrupt

being serviced on the low-order address bits during an interrupt acknowledgment cycle. When the host performs a Service Acknowledge Cycle, the CL-CD180 compares the value on the address bus with the three unique values stored in three internal registers — the Priority Interrupt Level Register 1 (PILR1) for modem requests, the Priority Interrupt Level Register 2 (PILR2) for transmit requests, and the Priority Interrupt Level Register 3 (PILR3) for receive requests. These values are set by the user at system initialization. A match will occur on only one of these registers, and this informs the CL-CD180 of the type of request being acknowledged.

In most circumstances the address bus should not have a value that does not match one of the three PILR values during an acknowledgment cycle. This will cause the CL-CD180 to not recognize that any bus cycle is occurring, and it will not assert DTACK*, or terminate the cycle, or take any other action. Doing this will not affect the CL-CD180, but the system must have some other provision to terminate the bus cycle. If, for example, the CL-CD180 shares an interrupt level with another device, different values on the address bus should be used to control responses to an acknowledgment, but the bus cycle should terminate in a usable way.

Service acknowledgments can also be performed by software. The host simply reads one of three Request Acknowledge Registers, and the CL-CD180 performs as if a hardware service acknowledge cycle had been executed.

Regardless of the method of acknowledgment used, within the CL-CD180, each service request state machine makes the following determination: if it has an internal service request pending, and there is a service acknowledge of the same type, it asserts its internal-acknowledge-accepted signal back to the Service Request Controller logic, negates the Service Request Output Pin, and holds its acknowledge-out daisy chain in a negated state. It also drives the value in the Global Interrupt Vector Register (GIVR) onto the data bus, for the host to read as part of the Service Acknowledge Cycle. The GIVR value placed on the bus during the Service Acknowledge Cycle serves two purposes. The least-significant three bits of GIVR indicate which of the four types of service requests are occurring. The upper-five bits are user-defined and serve to identify, in daisy-chained CL-CD180 systems, which of the multiple CL-CD180s is active.

If the service request state machine does not have a service request pending, and there is a software acknowledgment or address bus match, it passes the service acknowledgment down the chain by asserting IACKOUT*. If there is no match, the state machine remains idle.

If a service request is pending and the Receive Service Request is to be handled, the CL-CD180 is notified because the three PILR registers have different values in them; therefore, only one match (receive service, in this case) occurred. The internal grant from the service request state machine causes the receive service type code and active channel number (previously stored at the time the request was posted by the CL-CD180 processor) to be pushed onto the service request stack. This automatically causes the FIFO pointers to be set up for the active channel, with no host intervention.

The host, at this point, has all the information needed to handle the service request. It determines the exact type of service being requested (Transmit, Receive Good Data, Receive Exception, or Modem Signal Change) and which of the multiple CL-CD180s is requesting service. It gets the channel number by reading the Global Interrupting Channel Register (GICR) and then proceeds to service the request. At the completion of the service, the host performs a dummy write to the CL-CD180 End-Of-Interrupt Register (EOIR), that causes the CL-CD180 to exit its internal service request state by popping the service request stack. At this time the CL-CD180 is ready to be serviced on another of its outstanding requests. If another request of the same level is pending, two clock periods after the write to EOIR are required for the CL-CD180 to re-assert the request line.

Because the CL-CD180 has a service request stack, it can support nested-service requests. For example, the host can be in middle of a Transmit Service Request, detect that Receive Service Request has asserted, process the Receive Service Request, and after exiting the receive service routine, resume the Transmit Service Request. The CL-CD180 stack is three deep, so all three types (one of each) can be nested if desired. The current

service request context (i.e., the stack) is readable in the Service Request Status Register.

The Global Interrupting Channel Registers (GICR) are actually three registers that provide the number of the channel requesting service. Reading any of these registers will cause the CL-CD180 to mask in three bits, specifying the channel number of the currently active channel. Normally these registers are read by the host when it is handling a service request. In this case, the three bits will be the number of the channel requesting service. If any of the three GICR Registers are read when the CL-CD180 is not in a service-request context, the three bits will be the current value in the CAR. The current channel number is masked into the contents of Bits 4:2 of this register by the CL-CD180 when it is read by the host. The actual contents of the register are not modified.

These three registers are provided as a convenience to the user. In most applications, the user will only use one of these locations, and set the register to some arbitrary value. However, it may be useful to sometimes record information about the state of the CL-CD180 (or the software driving it) that is associated with each of the three service-request types. In this case, the user may store whatever information is desired in the unused bits. Then, when entering a service routine, the software can check these bits to find what state they were left in, and this could be used as a 'sub-vector'.

2.3.2 Internal Implementation of the Service Request Logic

As discussed above, the heart of each service request level is an asynchronous state machine. This state machine has three inputs:

- MATCH from the Priority Interrupt Level Register comparator,
- IACKIN* from the host system, and
- INTERNAL_REQUEST from the CL-CD180.

NOTE: Software acknowledgments (reads from the Service Request Acknowledge Registers), in effect, force the MATCH value true for their respective level.

It also has three outputs:

- Svc_Req to the host system,
- INTERNAL_GRANT to the CL-CD180, and
- IACKOUT*, which is combined with the other two IACKOUT* Signals to provide IACKOUT* to the next CL-CD180 in the daisy chain.

Figure 2–4 shows logic implemented by the state machine, which is described in Table 2–1.

Table 2–1. State Machine Logic

State Name	Output Condition	Comments

IDLE
IF (INTERNAL_REQUEST = 1)
ELSE IF (IACKIN* = 1 & MATCH =1)
ELSE
	all outputs inactive
	GoTo REQ_ACTIVE
	GoTo PASS_ACK
	Stay at IDLE
		
		; normal 'resting' state
		; pass this acknowledge
		; wait here

REQ_ACTIVE
IF (IACKIN* = 1 & MATCH =1)
IF (IACKIN* = 1 & MATCH =0)
ELSE
	
	GoTo KEEP_ACK
	Stay at REQ_ACTIVE
	Stay at REQ_ACTIVE
		request asserted
		; keep this acknowledge
		; wait here, ACK is for some other level (†)
		; wait here

PASS_ACK
IF (IACKIN* = 0)
ELSE
	
	GoTo IDLE
	Stay at PASS_ACK
		IACKOUT* asserted
		; return when IACK is gone
		; wait here while IACK active

KEEP_ACK
IF (IACKIN* = 0)
ELSE
	GoTo IDLE
	Stay at KEEP_ACK
		INTERNAL_GRANT asserted
		; return when IACK is gone
		; wait here while IACK active

NOTE: The (†) denotes the point at which, if there is no match, the CL-CD180 determines *not* to pass the IACK down the daisy chain. It does this for two reasons: first, it is unacceptable to have the IACKOUT* 'glitch' low; and second, the state machine should be as fast as possible. When the state machine senses an IACKIN* and match is not valid, it cannot conclude that it should assert IACKOUT*; the IACKIN* may be for one of the other two service requests levels. It could wait for the results of the other two MATCH comparators; however, this would complicate, and therefore slow down, the response of the state machine. The reason this complication would cause delay is (to implement the logical function 'assert IACKOUT* if no match') it must determine how long to wait before declaring a no-match condition. To implement this delay function, a synchronous state machine would be required, which at a 15-MHz clock, would mean a delay of several hundred nanoseconds from IACKIN* to IACKOUT*, instead of the 65 ns currently specified.

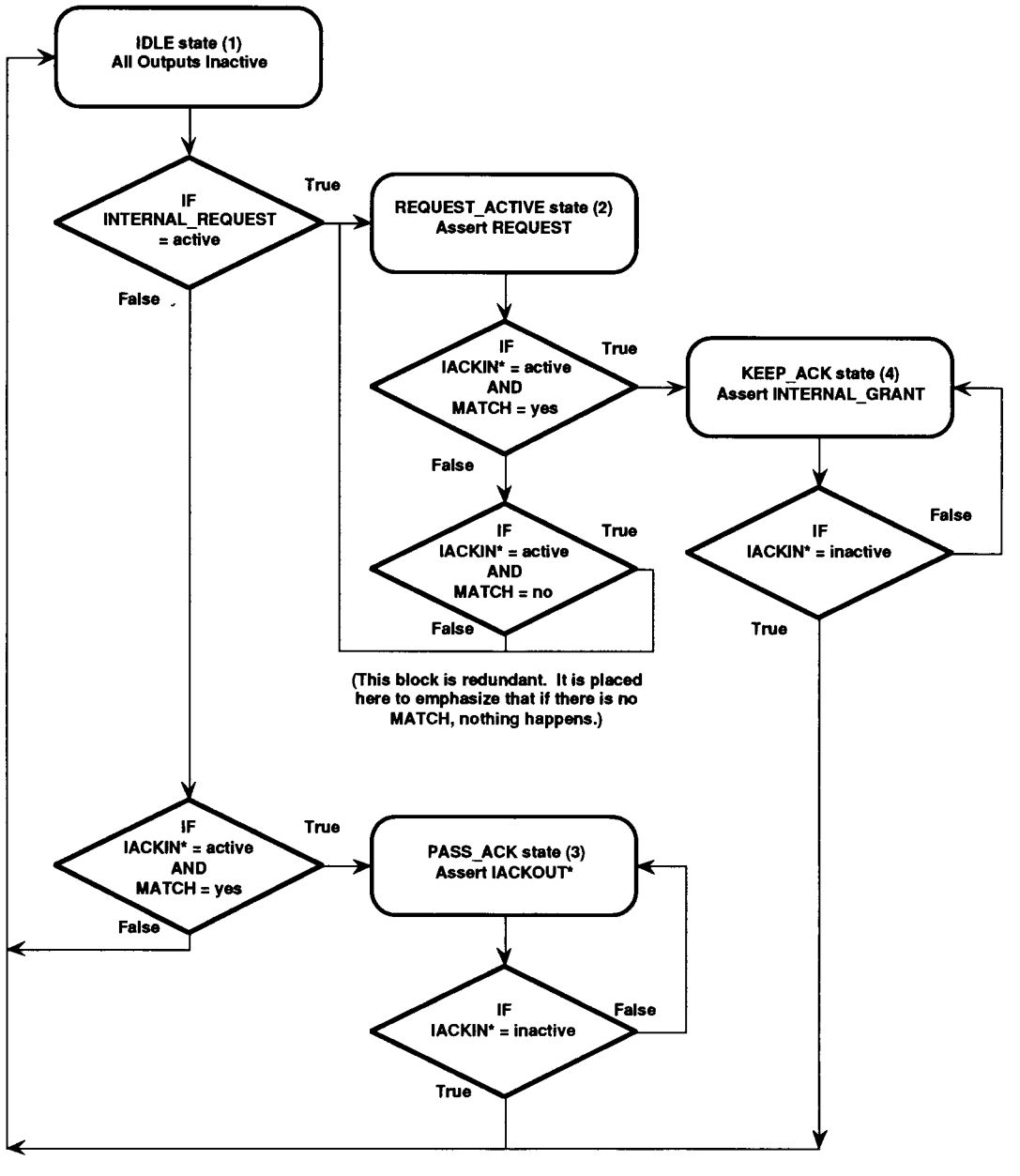
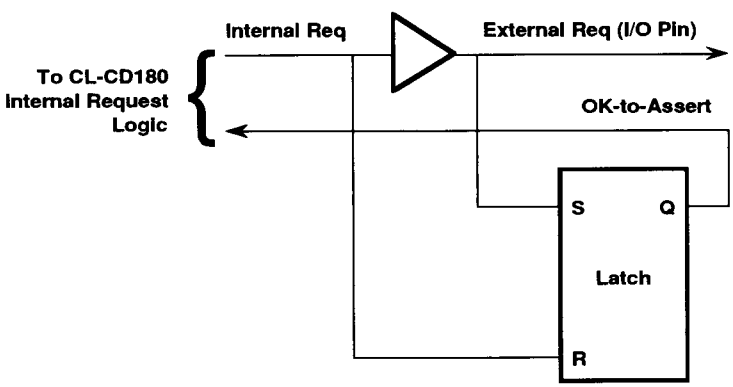


Figure 2–4. Internal Service Acknowledge Decision Tree

2.3.3 Priorities and Fair Share™

The CL-CD180 implements a Fair Share mechanism to ensure that all channels receive equal service, without any 'data starvation'. Fair Share works automatically among the channels in one chip and across multiple chips.

Figure 2–5 shows a Fair Share Operation block diagram. On each of the three service request lines, the CL-CD180 monitors both the internal and external value of the line. (The external value can differ because, in multiple-CL-CD180 applications, it can be driven by other CL-CD180s.) At the end of a service acknowledgment bus cycle, the CL-CD180 checks the state of both request values. If they are different, the CL-CD180 determines that there is another part also driving the request line, and it will not re-assert its own request line until the external request has gone inactive. This inactive level means every other CL-CD180 with a pending request has been serviced; therefore, it is now okay to re-assert requests without 'hogging' the host's attention.



514180-5

Figure 2–5. Internal Fair Share™ Operation

2.4 Types of Service Requests

The categories of service requests that a CL-CD180 can generate are explained below. Each channel's transmitter, receiver, and modem pins require service from the host occasionally; however, each category of service request conditions can tolerate different latencies in being serviced. Conditions for service requests fall into three basic categories:

- Data is received from the remote device and needs to be transferred to the host.
- Data from the host can be given to the Transmitter FIFO, which is now empty.
- A modem signal changes state.

Three separate service request levels are provided to support the following three categories:

Source	Pin Name	Request Match Register Name
Receive Data	IREQ3*	PILR3
Transmit Data	IREQ2*	PILR2
Modem Signal Change	IREQ1*	PILR1

2.4.1 Receive Service Requests

The Receive Service Request is unique as it has two sub-types; i.e., it is capable of returning one of the two different vectors during a service request acknowledge cycle. The two sub-types are — 'Receive Good Data' and 'Receive Exception'. The reason there are two types within one category of service request is that, while Good Data and Exceptions require different handling, they are both of equal priority, and need to be serviced in the order they are received. For example, suppose two good characters are received, then an exception character, and then another good character. There must be a service request for the first two bytes of Good Data, then for the Exception, and then for more Good Data. If Exception Service Request is at a different level, the exception character will be processed either before or after the Good Data, and not in sequence as it should be. This method also allows the Receive Good Data-handling routine in the host to be very fast and efficient, since it only has to move 'N' bytes to a buffer. All special-case conditions can be put in a separate handler, where they will not slow down normal data transfers.

Exception characters are characters with errors or that match the defined special characters, line breaks, and certain time-out conditions.

Data must *not* be read from the Receive FIFO or the Receive Status FIFO except when the CL-CD180 is within the context of a Receive Data Service Request.

2.4.1.1 Receive Good Data™

A Receive Good Data Service Request is asserted for any of the following three conditions:

1) RxFIFO threshold reached, and the FIFO contains Good Data.
2) RxFIFO threshold not reached, but the FIFO contains Good Data, and the Receive Data Timer times-out.
3) RxFIFO threshold not reached, but the FIFO contains Good Data, and the newly arrived data contains an exception condition.

When any of these conditions occur, the modified service request vector indicates to the host that the service request is for Good Data. The CL-CD180 continues to add bytes to the FIFO, and it increments the Count Register for each good byte added, and this allows for optimally efficient use of the FIFO.

It is not necessary to accept any or all of the Good Data that is available when a Good Data Interrupt is received. If a host buffer is too full to accept eight bytes, a smaller number (even 0) can be read, the service request context left, and the host buffer handled first. The CL-CD180 will again generate another Good Data Service Request when any of the three conditions listed above are met.

If the condition which caused the request in the first place remains true, the CL-CD180 quickly generates another service request. If no data is read, this is always the case. If some, but not all, of the available data is read, Conditions 1 and 2 will not be true, but Condition 3 may be if an exception condition was the cause of the Good Data Interrupt. If this becomes a problem, one solution is to temporarily disable receiving interrupts on that channel. To avoid FIFO overflow, do not disable the channel for too long.

2.4.1.2 Receive Exception

Unusual or exception conditions are reported to the host one character at a time through the Receive Exception Service Request. As with normal receive processing, the host determines the requesting channel by reading the GICR. It can then determine the specific exception(s) by reading the Receive Character Status Register.

Exception conditions are generated for parity errors, framing errors, FIFO overrun, special character recognition, break detect, and for a special feature called the 'No New Data Timer' (NNDT).

NNDT is a receive timer option to generate a service request for the first receive data time-out following the transfer of all data from the FIFO to the host. It is often useful, when managing relatively large I/O buffers, for an I/O processor to determine that 'no data has arrived lately'. This event is used to transfer the contents of the local buffer that has been storing data from the CL-CD180 FIFO for host-system processing.

This service request is a receive exception subtype, and can be used to signal that it is time to transfer the buffer. This feature can be enabled or disabled by controlling the NNDT Bit in the Service Request Enable Register. As shown in Figure 2–6, every time a received character is loaded into the FIFO, the timer is restarted. If the timer times-out, the CL-CD180 checks if there is any data in the FIFO. If there is, a Good Data Service Request is posted to avoid 'stale data'. If there is no data in the FIFO, the CL-CD180 checks that NNDT is enabled and 'armed'. Arming occurs when the last character is transferred out of the FIFO to the host. If NNDT is on and armed, a Receive Exception Service Request is posted to inform the host of this event. Note that the NNDT is not armed if the last character removed from the FIFO was an exception character.

Every Receive Exception is a unique, one-character event. The Receive Data Count Register has no meaning, unlike the Receive Good Data case, the Status Byte in the receive exception handling routine must be read. The Receive Data Count Register and the associated data character will be discarded by the CL-CD180 at the end of the service routine. The Status Byte must be read before reading the Data Byte. Once the Data Register is read, the Status Byte is no longer available.

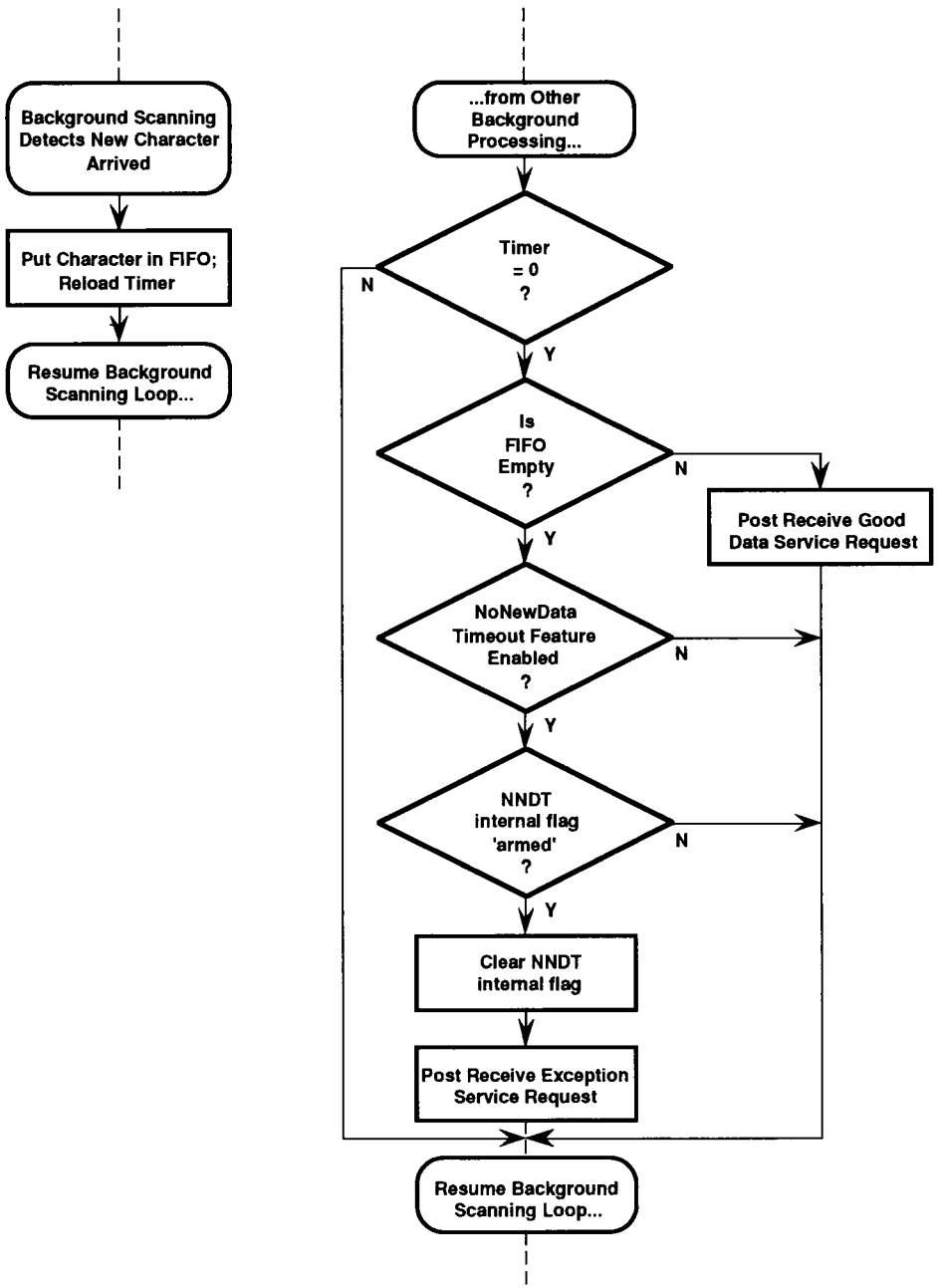


Figure 2–6. Receive Timer Operation

2.4.2 Transmit Service Requests

Each transmitter contains eight bytes of Transmit FIFO in addition to the Transmit Holding Register and the Transmit Shift Register. As data is being transmitted, the FIFO status is being monitored by the CL-CD180. A service request is invoked for one of the following conditions:

- Transmit FIFO Empty — When the Transmit FIFO is empty, there is still one character in the Transmit Holding Register and one character in the Transmit Shift Register. The host has two character times to respond to this request without causing a gap in the Transmit Data Stream.
- Transmitter Empty — The Transmit FIFO, Transmit Holding Register, and the Transmit Shift Registers are now all empty. This signifies that all characters written to the FIFO have been completely transmitted.

The host can select which one of these causes a Transmit Service Request, and it will be used by programming the options in the Service Request Enable Register (SRER).

Data must *not* be put into the Transmit FIFO at any time other than when the CL-CD180 is in a Transmit Service Request context for that channel. During a transmit service, characters (up to eight) are placed into the FIFO via the Transmit Data Register (TDR).

2.4.3 Modem Signal Change Service Requests

The CL-CD180 may be programmed to assert a service request when a channel's modem input signals have changed states. The change-detect options are programmed in the Modem Change Option Registers. Individual modem pin service requests are enabled by setting the corresponding bits in the Service Request Enable Register.

The host must read the Modem Change Register during a modem change service to determine which modem signal changes were detected. This is indicated by a '1' in the appropriate bit location. The Modem Change Register must be reset to a '0' by the host before exiting the service request because the CL-CD180 does not do this. Refer to Section 4.4 for more details.

2.4.3.1 Using Modem Pins as Input/Output

The pins labelled as modem pins are general-purpose I/O pins that can be controlled by either the CL-CD180 processor or the host system. There is no direct, hardwired connection from any modem pin directly to a transmitter or a receiver. This means that these pins can be used for general-purpose I/O if they are not needed for modem-control purposes. See Section 4.4 for more details.

2.5 Implementing Service Requests

The CL-CD180 is designed to easily interface with any processor, yet be efficient and flexible enough to provide maximum throughput. The CL-CD180 generates service requests and waits for acknowledgments of these from the host. However, service requests may be implemented in either hardware or software; likewise, acknowledgments can be affected either way to offer maximum advantages to the system designer and programmer. This interfacing can begrouped as various steps.

Service requests must be 'noticed' by the host system before they can be acted on, and this can be done the following three ways:

1. Provide three levels of interrupt support, with three separate levels and three separate vectors. This is well-suited to Motorola® 680X0 processors.
2. Provide a single level of interrupt support; this is an effective method when using 8-bit processors such as the Z-80 and many Intel® microprocessors.
3. Poll the device directly in software.

Once the host has 'noticed' the service request, it has the following two choices for acknowledging the request and begining to service it:

a. Acknowledge the request via a hardware-based service acknowledgment, as is typically done in interrupt-driven systems.
b. Acknowledge the request in software by reading from a register in the CL-CD180.

Table 2–2. Service Request Methods

		How the host detects the Service Request		
		1. Three-level Hardware Interrupt	**2. Single-level Hardware Interrupt**	**3. Software Polling**
How the host acknowledges the Interrupt	**a. Hardware-based service acknowledge**	1a Full Interrupt – Type A	2a Not recommended (Inefficient)	3a Not recommended (Inefficient)
	b. Software-based service acknowledge	1b Full Interrupt – Type B	2b Single Interrupt	3b Software Polled

Thus, there are six theoretically possible options for interfacing the CL-CD180 to the host system. Two of the methods (2a and 3a) are not practical to implement without external hardware, and offer no performance advantage. Each of the other four methods has advantages and drawbacks depending on the type of host CPU being used and whether or not that host CPU supports more than one CL-CD180. The four methods used are listed in Table 2–2.

1a. This method is called 'Full Interrupt – Type A'. The system is fully interrupt driven with acknowledgments in hardware. It requires a host with at least three interrupt priority levels available and the ability to acknowledge on multiple levels. This is the technique used by Motorola 680X0 processors. It is the most efficient method when the host CPU has a relatively fast interrupt context switch time and when the host CPU has duties other than driving the CL-CD180s.

1b. This method is called 'Full Interrupt – Type B'. It still has three levels of interrupt, but provides a single acknowledgment level. It is commonly used in Intel-type processor systems where there is an 8259A interrupt controller. The 8259A receives the three levels of interrupt, but it provides its own vector to the host rather than that of the CL-CD180s. The host then acknowledges the CL-CD180s Service Request by reading the Vector Register.

2b. This method is called 'Single Interrupt', and is best-suited to systems having only a single interrupt input, such as most 8-bit microprocessors. After the host has received its interrupt and is entering its interrupt service routine, it reads the CL-CD180 to see which of the three types of service requests is responsible for the interrupt. It then acknowledges the interrupt by reading the appropriate Request Acknowledge Register. Note that the single interrupt signal must be generated via the logical OR of the three request outputs with external output gates, not by 'wire-OR'ing' them.

3b. This method is called 'Software Polled'. Polling is often used in situations where the host system is primarily dedicated to servicing the serial channels and has few other tasks to perform. It is often better when the host CPU has a long interrupt context switch time. In this method, the host periodically checks the CL-CD180s to determine if any service requests are pending. If they are, the host acknowledges them in software and proceeds with the service.

One of the advantages of the CL-CD180 is that it allows the use of any of the above techniques, or a combination thereof. Such a combination is referred to as 'Mixed-mode Operation'. In a typical mixed-mode design, normal interrupts are used to signal to the host that service is required. After the

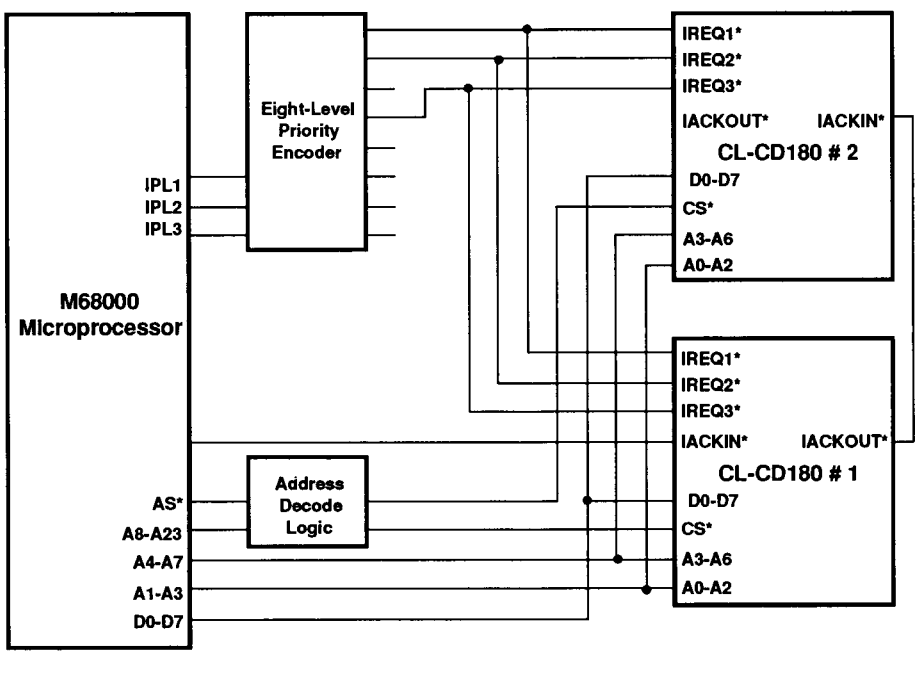
host enters its interrupt service routine, it services the CL-CD180 that generated the service request. The host then polls the CL-CD180s to determine if more channels require service. Frequently this will be the case. When the host finds a channel requiring service, it handles it in the usual manner, and then proceeds to poll again for more service requests. This process continues until all CL-CD180s have been handled. Because the host is not exiting and re-entering its own interrupt context each time, much host CPU time is saved, resulting in even faster overall performance.

A mixed-mode design has the advantage that the software has complete control of whether to be fully interrupt driven or to poll in certain circumstances. A mixed-mode design is recommended to tune a system for optimum performance.

A CL-CD180 evaluation board can be employed to analyze CL-CD180 performance and evaluate different software implementations. Cirrus Logic testing (in an AT-compatible '386 machine) found that a mixed-mode system provided the highest overall throughput with minimum host CPU loading. This was generally found to be the case with host processors that have relatively long interrupt response times, such as the Intel '386.

2.5.1 Method 1a — Full Interrupt – Type A, Three-Level Interrupt with Three-Level Acknowledge

This method is illustrated in Figure 2–7. It is best-suited for 680X0-family processors. The three CL-CD180 service request lines are connected to the Interrupt Priority Encoder. When the host performs an interrupt acknowledgment cycle, the CL-CD180 responds with its vector. The host uses this vector to jump directly to the appropriate service routine. The other methods can also be used with a 680X0-based system.



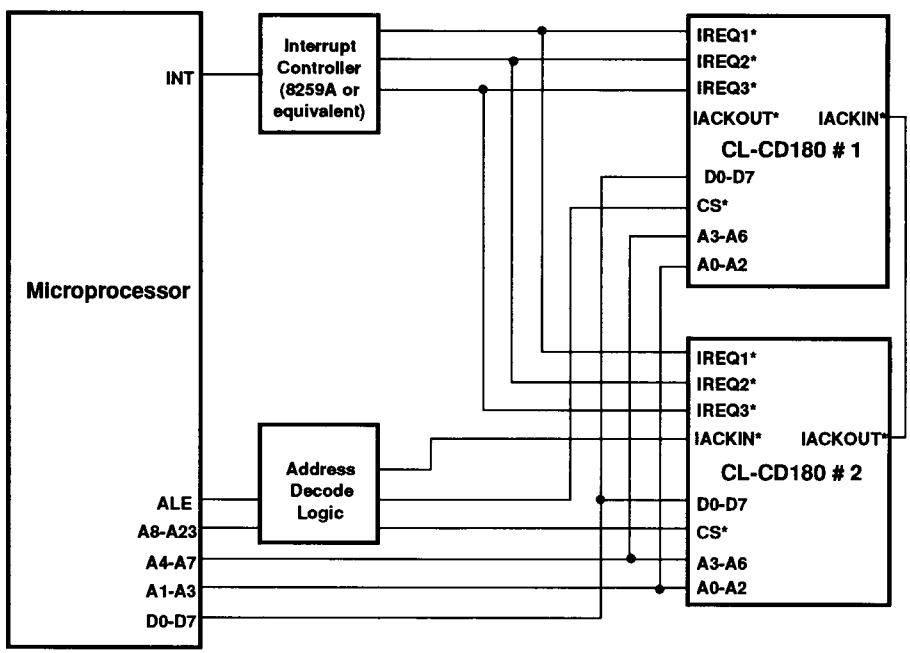
514180-7

Figure 2–7. Three-Level Interrupt with Three-Level Acknowledge Example

2.5.2 Method 1b — Full Interrupt – Type B, Three-Level Interrupt with Single-Level Acknowledge

This method is illustrated in Figure 2–8. It is useful with 80X86 systems that use the 8259A Interrupt Controller. Since the 8259A supplies its own vector to the host when an INTA cycle occurs, the host can simply read the CL-CD180's vector by the method described in the polled interface example or a separate chip select decode can be provided to drive the IACKIN* input.

After the 8259A has supplied a vector to the 80X86 host CPU, the host performs a software acknowledgment to the CL-CD180, which transfers the CL-CD180 vector to the host and allows the service request to be processed.

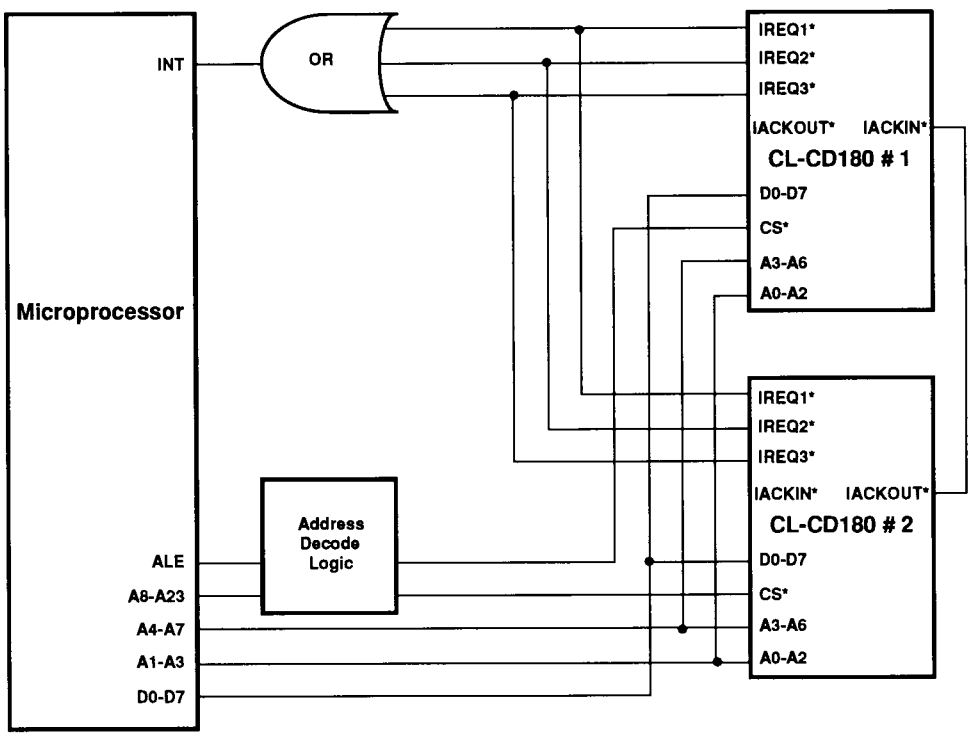


514180-8

Figure 2–8. Three-Level Interrupt with Single-Level Acknowledge Example

2.5.3 Method 2b — Interrupt Interface, Single-Level Interrupt with Single-Level Acknowledge

This method is illustrated in Figure 2–9. It is best-suited to host systems having a single interrupt input. The three service request lines from the CL-CD180 are run through an 'OR' gate to the host's interrupt input. When an interrupt occurs, the host system polls the CL-CD180s to determine which of the three levels it was, and acknowledges it accordingly.



514180-9

Figure 2–9. Single-Level Interrupt with Single-Level Acknowledge Example

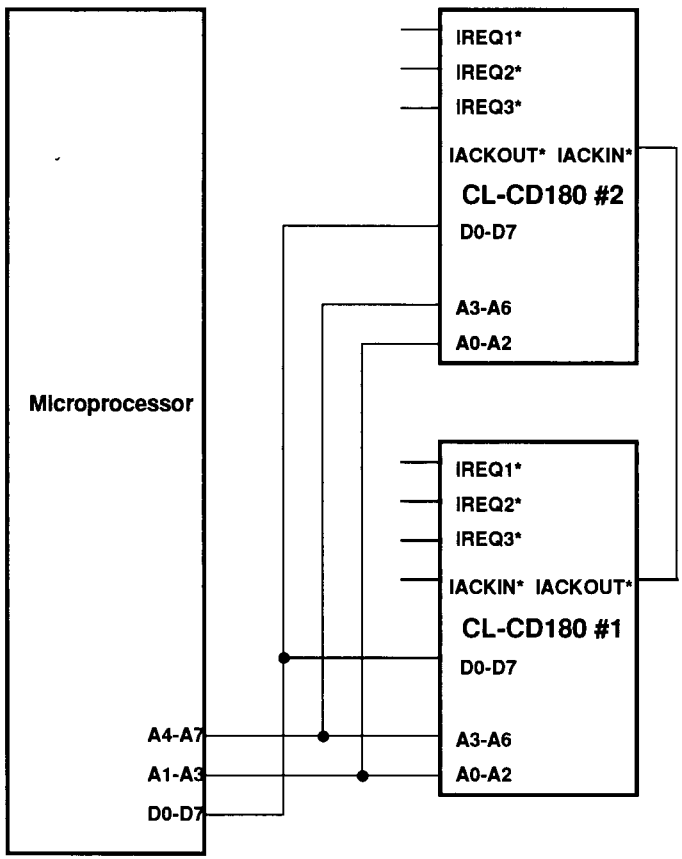
2.5.4 Method 3b — Polled Interface

This method is illustrated in Figure 2–10. Polled operation can be used with any type of host CPU, or it can be used in combination with interrupts to provide a mixed-mode system optimized for a particular application. In a polled system, the host reads the Service Request Status Register (SRSR) within the CL-CD180 to determine whether there are any channels that need service. (Note that unlike traditional UARTs, only one register needs to be read to determine if there are any channels in any device that need attention, and this saves time).

If the host finds channels needing service, it acknowledges the desired type by reading one of the three Request Acknowledge Registers. These provide a vector that can be used to jump directly to the correct service routine. Processing from this point proceeds as in the case of interrupt-driven operation. Note that the difference between this method and Method 2b lies in how the host system becomes aware of the need to service the CL-CD180. In Method 2b a single interrupt starts the process. In Method 3b the host polls periodically. The two methods can be combined — an interrupt triggers the first service, but the host continues to poll until any other pending requests have been serviced.

There is a difference between the CL-CD180 and conventional dumb UARTs that makes the CL-CD180 more efficient even when operating in a polled environment. With a dumb UART, the host polls each channel in turn to determine whether it has any data. With the CL-CD180, the host polls the CL-CD180s as a group for whether it has data. If it does, the CL-CD180s will indicate the channel, rather than the host testing each channel in turn. In fact, it is not possible for the host to dictate which channel is to be serviced; the CL-CD180 determines this order. This minimizes both the number of polling steps required and the amount of time each needs, and it also ensures fair, balanced service of all channels.

There are several ways a host system can poll the CL-CD180, and each method has certain advantages. The most direct method is to read the Service Request Status Register (SRSR). This register contains three bits that indicate whether there is a request pending for receive, transmit, or modem signal change, on the CL-CD180 being read. There are three more bits that provide the same information for all CL-CD180s in the system — these three bits reflect the state of the wire-OR'ed external request lines. Thus a single read operation can determine if there is any activity.

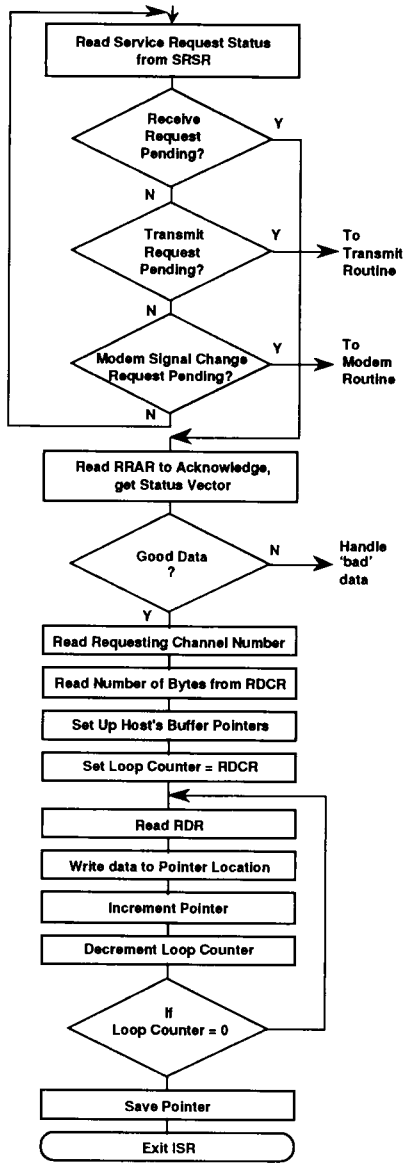


514180-10

Figure 2–10. Simple Software Polled Interface Example

2.5.5 Comparison of Interrupt and Polled Code Sequences

Figure 2–11 and Figure 2–12 show the code sequences for polled and interrupt service request methods.



514180-11

Figure 2–11. Polled Code Sequence

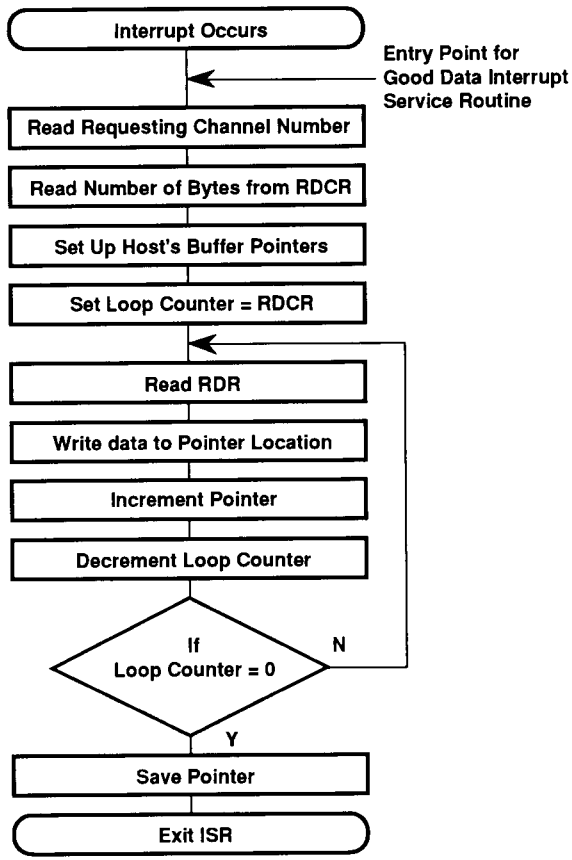


Figure 2–12. Interrupt Code Sequence

2.5.6 Cascading Service Requests with Multiple CL-CD180s

Regardless of the method used to support service requests, multiple CL-CD180s may be cascaded by tying together all IREQ1* lines, all IREQ2* lines, and IREQ3* lines. These lines are open drain so they may be wire-OR'ed. The CL-CD180s are then daisy-chained by simply connecting the IACKOUT* of one device to the IACKIN* of the next.

The host knows which CL-CD180 is requesting service by the value returned by the CL-CD180 from its Global Interrupt Vector Register. Up to 32 CL-CD180s may be cascaded in any one daisy chain in this manner. The maximum number of CL-CD180s can be very large, since multiple daisy chains are possible. The 32-per-daisy-chain limit is set by the fact that there are five bits in the GIVR that can be used to identify which CL-CD180 responded to the service request acknowledge cycle. The user must program different values into the upper-five bits of each CL-CD180s GIVR.

Note that thirty-two CL-CD180s is the logical limit per daisy chain. Since it takes over 1000 ns for an acknowledgment to ripple down 32 devices, it may not be efficient to have one long chain in heavy-traffic applications.

NOTE: In some systems that daisy-chain many CL-CD180 devices, a potential timing hazard exists if the host processor does not allow sufficient time for the removal of the IACKIN*/IACKOUT* daisy-chain signal to propagate through all devices. In the event that the host processor begins I/O operations with another section of logic and applies DS* (RD* or WR* in an Intel environment) while an active IACKIN* is being applied to a CL-CD180 due to propagation delay time, unpredictable results can occur. This constitutes an illegal acknowledge cycle. The failure mode is most often a cessation of service requests from the device, especially of the type that was being serviced when the illegal access occurred. Care must be taken to assure that the 35-ns propagation delay per device is included in any wait-state generation.

2.5.7 Multiple CL-CD180s without Cascading

It is possible to interface several CL-CD180s without using the cascade feature. There is an advantage to this because as there is less delay incurred while waiting for the service acknowledgment to ripple down a chain of devices. There are two possible disadvantages. If each of the CL-CD180's three service request lines has a separate input to the interrupt controller, the interrupt controller is more complex, and the Fair Share feature does not work. If the service request lines are wire-OR'ed, Fair Share works, but the host has to test each CL-CD180 in turn to see which one generated the service request. To implement this method, simply connect the CL-CD180 address and data lines in the usual manner.

2.5.8 Acknowledging Service Requests

As mentioned in Section 2.5, two different methods can be used to acknowledge a service request. One method is hardware-based, and the other is software-based. The hardware-based mechanism is a specific type of bus cycle that uses the IACKIN* and IACKOUT* Signals and the PILR registers in the CL-CD180. An acknowledge cycle is defined where IACKIN* and DS* are active and CS* is inactive. This method is used by processors that perform interrupt acknowledge cycles, such as the 680X0.

The software-based mechanism uses three registers — Receive Request Acknowledge Register, Transmit Request Acknowledge Register, and Modem Request Acknowledge Register. Reading any of these registers has the effect of acknowledging a service request, and the data read will be the appropriate vector, i.e., the contents of the Global Interrupt Request Vector Register. The low-three bits of this register will be modified to indicate the specific type of interrupt being acknowledged.

If the host reads these registers when no service request is pending, either of two things will happen. If daisy chaining of acknowledgments is enabled, the IACKOUT* Pin of the CL-CD180 will assert. If daisy chaining is not enabled, the part will supply a vector with the low-three bits set to a '0'. Thus, it is possible to 'fish' for service requests, i.e., to acknowledge each CL-CD180 in turn until a non-zero vector is received.

'Fishing' is not usually an efficient software technique, but may be useful in some circumstances. For example, in systems that are normally interrupt-driven, but where interrupts are not available for diagnostics or other reasons, the host can determine if a service request is pending by reading the appropriate Request Acknowledge Register. The CL-CD180 must be configured not to daisy-chain; in this case it will return a vector if a request is pending, or '00' if no request is pending. The host may try all three levels of request in turn. This method will work for either single CL-CD180s or multiple devices. In multiple-device systems, either disable daisy-chaining on all devices and 'fish' each individually, or disable daisy-chaining on the last device only and 'fish' the device at the beginning of the chain.

Both methods of acknowledging service requests may be used interchangeably. It is usually advantageous to use Mixed Mode. For example, after receiving an interrupt and servicing it in the normal manner, the host should read the Service Request Status Register (SRSR) to see if other requests are pending. If so, the host can acknowledge by reading the appropriate Request Acknowledge Register (RRAR, TRAR, and MRAR) and proceed to service the request. This avoids the time required for the host to exit its interrupt routine, only to re-enter it immediately for the next request.

3. SYSTEM BUS INTERFACE AND SYSTEM CLOCK

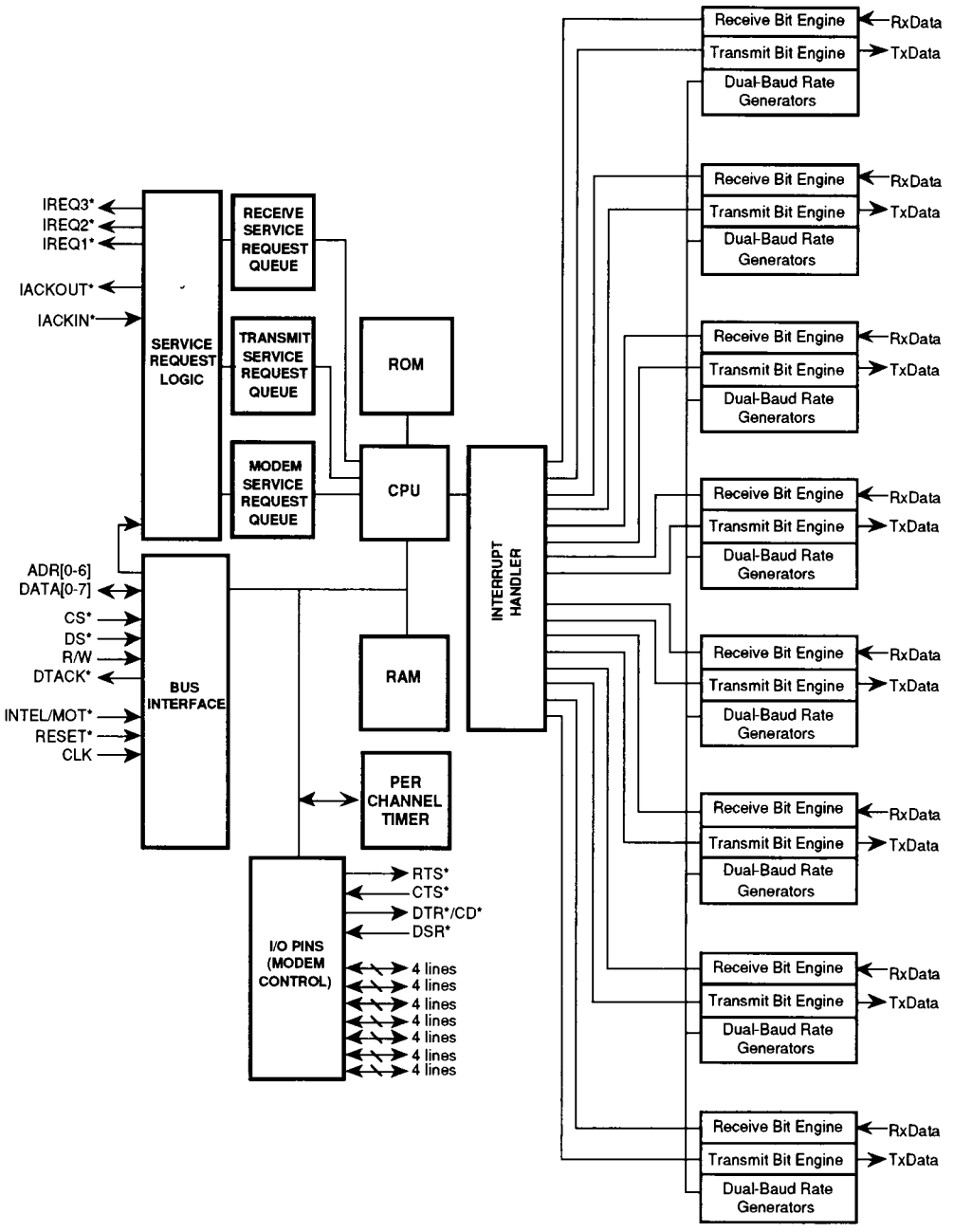


Figure 3–1. Internal Block Diagram

3.1 System Interface Considerations

When using the CL-CD180, two areas where system architects, designers, and programmers should consider options are system clock speed, and Un-clocked versus Clocked-host Bus Interface.

3.2 System Clock and Bit Rate Options

3.2.1 System Clock

System clock is a high-frequency clock (supplied by the user) used by the CL-CD180 to derive all the necessary timing. The CL-CD180 is capable of handling system clock levels of TTL-compatible voltage swings; however, the V_{IL} and V_{IH} specifications are not identical to all families of TTL logic. Specifically, the clock signal (and the reset signal) have lower V_{IL} and higher V_{IH} than the worst-case specifications of some TTL families. In general, any TTL family is adequate if not heavily loaded. Refer to the DC Specifications in Section 7.3 for details.

The CL-CD180 operates from an external clock source. The external clock is 1x the rated frequency. The 1x clock input is strictly specified in terms of rise and fall time, duty cycle, and V_{IL} and V_{IH}. Commercially-available oscillator modules typically have poor duty-cycle ratios (60/40) and poor V_{OH}, therefore it is advisable to specify a 55/45 duty cycle oscillator and to buffer the output through a 74HC-type device. Conversely, the oscillator used could be twice the rated frequency which is subsequently divided by two via a 74HC74 (or similar); this provides the proper duty cycle and V_{IH} and V_{OH}.

The CL-CD180 can be operated from the main system clock or its own clock. Operation from the main system clock can reduce the number of clocks required, and it allows the bus interface between the system and the CL-CD180 to be clocked, but in general, typical system clock speeds are not exact baud-rate multiples. As bit rates are derived from the clock, it is important to take this into consideration when selecting a clock value. If exact baud rates are needed, or the system clock is not a convenient value, the CL-CD180 must be supplied with its own clock or crystal.

3.2.2 Bit Rate Options

The CL-CD180 supports independent transmitter and receiver bit rates on each of its eight channels. The bit rate is determined by a 16-bit period value (divisor) stored in the Transmitter Bit Rate Period Registers (TBPRH and TBPRL) or in the Receiver Bit Rate Period Registers (RBPRH and RBPRL). These registers establish the period of the corresponding Transmitter and Receiver Bit Rate Counters. The value to be loaded to set a given bit rate is determined by the following equation:

$$\text{Bit Rate Divisor} = \frac{(\text{CLK frequency (in Hertz)})}{(16 \times \text{desired Bit Rate (in bits per second)})}$$

This equation may yield a non-integer result. The nearest integer value is the optimum choice for that bit rate and system clock combination. The value loaded in the Bit Rate Period Registers must be that integer expressed as a 16-bit binary value. If rounding is necessary, the percentage bit rate error may be calculated by:

$$(\text{Bit Rate Divisor} - \text{Integer}) \times 100 / \text{Bit Rate Divisor}$$

The popular bit rates and their corresponding divisors at various system clock rates are shown in Table 3–1.

If operation at 57.6 and 64 kbps is required, a clock of 12.5 MHz must be used.

Table 3–1. Possible Clock Speeds For Exact Baud Rates

Max Baud Rate (kbps)	Baud Rate Divisor Value								
	12	13	14	15	16	17	18	19	20
	Notes 1, 2, and 3				Note 1				
64	12.2880								
57.6	11.0592	11.9808							
56	10.7520	11.6480	12.5440						
38.4	7.3728	7.9872	8.6016	9.2160	9.8304	10.4448	11.0592	11.6736	12.2880

NOTES:

1) All clock speeds are shown as net clock speeds in MHz.
2) Divisors less than 16 may not produce 100% throughput in all cases.
3) Divisors less than 12 may result in errors if all eight channels are active.

Table 3–2 shows possible clock values for various baud rates and divisors. Not every combination is legal in all cases. Refer to Section 3.2.3 for information on throughput limits.

Table 3–2. Divisors For Standard Baud Rates For Various Clock Speeds

Baud Rate	Clock Speeds		
	11.0592 MHz	**9.8304 MHz**	**9.216 MHz**
64 kbps[a]	N/A	N/A	9
57.6 kbps	12	N/A	10
38.4 kbps	18	16	15
19.2 kbps	36	32	30
14.4 kbps	48	43	40
9600 bps	72	64	60
7200 bps	96	85	80
4800 bps	144	128	120
2400 bps	288	256	240
1200 bps	576	512	480
600 bps	1152	1024	960
300 bps	2304	2048	1920
150 bps	4608	4096	3840
110 bps	6283	5585	5236
75 bps	9216	8192	7680

[a] Note that there are no perfect divisors for 64 kbps at any of these standard frequencies except 9.216 MHz. However, serial performance at 64 kbps with a 9.216 MHz clock will be unacceptable and is *not* recommended.

3.2.3 Maximum Throughput Limits

The CL-CD180 is internally a fully static, synchronous design. Consequently, the maximum data rate handled by CL-CD180 is determined by the clock speed at which it is operating. There are a fixed number of CL-CD180 processor cycles required to process each bit and character; a slower CL-CD180 processor rate equates to a slower bit rate. The minimum clock frequency required can be determined by the data rate needed to be supported.

In general, the CL-CD180 can maintain 100% full-duplex throughput when divisors of 16 or greater are used. For a given master clock frequency, this limitation can be used to determine the maximum bit rate at which the system can sustain 100% throughput on both receive and transmit. Divisors as small as 12 may be used, however a degradation in throughput will be observed. This degradation will be seen as gaps between transmit characters and are, in effect, extra long stop bits. This is a fail-safe condition. Divisors smaller than 12 may work in an application if less than eight channels are enabled.

WARNING: Extensive testing is required in the user's end application to determine the maximum serial performance that can be sustained without error in that environment. Cirrus Logic only guarantees that the device will not fail at bit rates up to 64 kbps when operating at 12.5 MHz and with eight channels operating.

Lab testing at Cirrus Logic has shown the following performance characteristics in an IBM-PC compatible environment with a 33-MHz 80486 processor:

Table 3–3. Performance vs. Operating Frequency

Clock Frequency	Maximum Data Rate	Typical Throughput (Receive/Transmit)
8 MHz	38.4 kbps	100%/66%
10 MHz	38.4 kbps	100%/100%
12.5 MHz	57.6 kbps	100%/65%
12.5 MHz	64 kbps	100%/54%

3.3 CL-CD180 Basic Bus Interface and Addressing

The CL-CD180 is addressed through an active-low Chip Select (CS*) in conjunction with seven Address Inputs A[0:6] that are mapped CL-CD180 internal addresses in two addressing modes — global and channel. In Channel Addressing Mode, the bits defining the channel to be accessed are provided from the Channel Access Register (CAR) within the CL-CD180.

The most-significant Address Input (A6) performs the selection between global- and channel-specific addresses. If this bit is a '1', the address is global, and is not associated with any specific channel. If this bit is a '0', the address is channel-related.

With the exception of the FIFOs, all channel-specific registers are accessed by first setting the desired channel number in the low-three bits of the Channel Access Register. FIFOs may only be accessed within the context of a service routine. Attempting to force access to a particular FIFO by setting the CAR will cause unpredictable and incorrect results. Within the context of a service request, the effective channel access value is automatically controlled by the CL-CD180, thus the CAR should not be modified by the host system during service-request processing.

The advantage of this method is that the host never performs any address computation to access the CL-CD180 during service requests. Because only the registers specific for the active channel (i.e., the one being serviced) are accessible to the host within a service request routine. An automatic indexing feature handles this, thus avoiding any burden on the host. Refer to Section 6.3 on Indexed Indirect Registers for details.

3.3.1 Intel® Versus Motorola® Interface Signals and Addressing

The CL-CD180 supports two bus handshake methods. One is patterned after the Motorola 680X0-family processors, and the other after Intel 80X86-bus interfaces. Bus interface selection is achieved via the INTEL/MOT* Signal. When this

signal is 'high', the Intel Bus Interface is selected, and when this signal is 'low', the Motorola Bus Interface is selected. This selection affects the logical meaning of two pins, but has no effect on bus timing.

The two signals having dual meaning are RD* versus DS*, and WR* versus R/W*. When the Intel Bus Interface is selected, these two pins function as RD* and WR*. These pins can be connected to either the IOR* and IOW*, or to MEMRD* and MEMWR* depending whether the CL-CD180 is mapped into memory or I/O space. These pins then serve to select the CL-CD180, and when either is active (along with CS* or IACKIN*) the CL-CD180 considers itself selected. CS* and IACKIN* must never be active at the same time.

When the Motorola Bus Interface is selected, these two signals function as DS* and R/W*. DS* must be asserted (along with CS* or IACKIN*) for all types of cycles, and R/W* should be low when writing to the device.

In either case, the choice of bus interface is entirely up to the user. This feature is for users convenience, and to accommodate the address and bus-control logic being used. The CL-CD180 has an 8-bit data bus, and it is a common practice (when connecting 8-bit peripherals to 16- or 32-bit systems) to connect them to only one lane, or one byte position. Thus, the CL-CD180 Registers will appear in the host's address space only at every other byte address. The most common practice is to connect the CL-CD180 to the portion of the data bus labelled D0-D7. For the little-endian processors, such as Intel's, the CL-CD180 will appear at even addresses (A0 = 0). For big-endian processors, such as Motorola's, the CL-CD180 will appear at odd addresses.

3.3.2 Un-Clocked Versus Clocked Bus Interface

Depending on the type and speed of the host processor, another important choice is determining the system bus interface to be clocked or un-clocked with the host CPU clock. Because there is a single clock for both the bus interface and bit-rate generation, the decision to use either Clocked or Un-clocked Bus Interface is affected by whether exact bit rates are required. Most applications do not require exact bit rates, and operate with rates varying by one percent or so. If exact bit rates are required, the clock speed must be a baud-rate multiple.

One method of bus interfacing may be preferable to another in certain applications. Although the easiest way to interface to the CL-CD180 is by using the un-clocked handshake supplied by DTACK*, in some cases it may be better to design a clocked interface. The latter is true if the host system is running at the same clock speed (or a multiple) of the CL-CD180 speed.

Un-Clocked Bus Interface

An Un-clocked Bus Interface is the easiest interface to implement. Simply connect the address, data, and control lines in the customary manner, and use DTACK* to control the number of wait states either by connecting it to the processor's DTACK* (if it has one), or by feeding into a wait-state generator. Figure 3–2 shows a typical Un-clocked Bus Interface.

The maximum bus cycle time is two clock periods plus 10 ns, though typically less because this specification is based on worst-case internal synchronization delays. Using DTACK* saves time; however, it is permissible to hard-wire the wait-state generator for the maximum time.

Clocked Bus Interface

The CL-CD180 bus interface is controlled by a state machine that samples on the falling edge of the clock. External strobes (CS*, DS*, and R/W*; or CS*, and RD* or WR*) that meet the setup time requirement cause a bus cycle to begin. The external interface can be designed to meet these setup time requirements, and to have shorter CL-CD180 access cycles. Figure 3–3 shows a typical Clocked Bus Interface.

A bus cycle consists of two half-clock periods. During the clock-low period, the transaction is set up internally, and the local bus arbitration occurs. During the clock-high period, the read or write transaction to RAM occurs. On write cycles, the data from the host is latched internally on the low-to-high clock transition. On read cycles, the data is available shortly after the end of the clock-high period.

Read and write cycles differ slightly in timing; during a write, it is permissible to remove the WR* or DS* relatively early during the high-clock period, however, this cannot be done during read cycles. The RD* or DS* Strobe is used as an output enable, and must remain low for the data to appear on the external data bus.

Service request acknowledgment cycles follow a different timing than ordinary read cycles. First, it is necessary to have the address stable before asserting IACKIN*. Second, the setup time from IACKIN* and DS* (or RD*) going low to the falling clock edge is longer due to additional internal logic involved in service request acknowledge cycles.

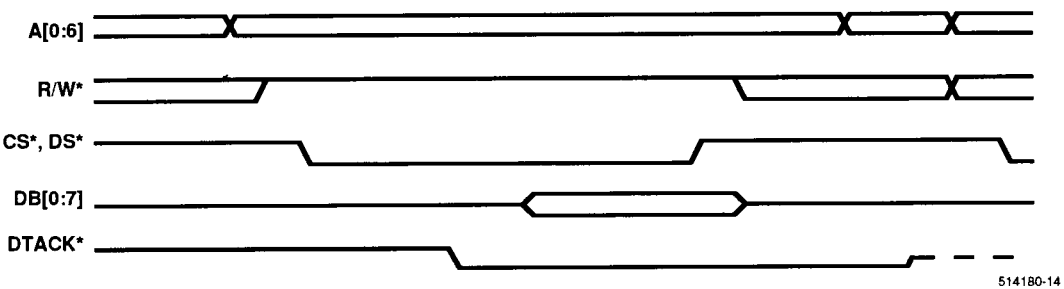


Figure 3–2. Typical Un-Clocked Bus Interface

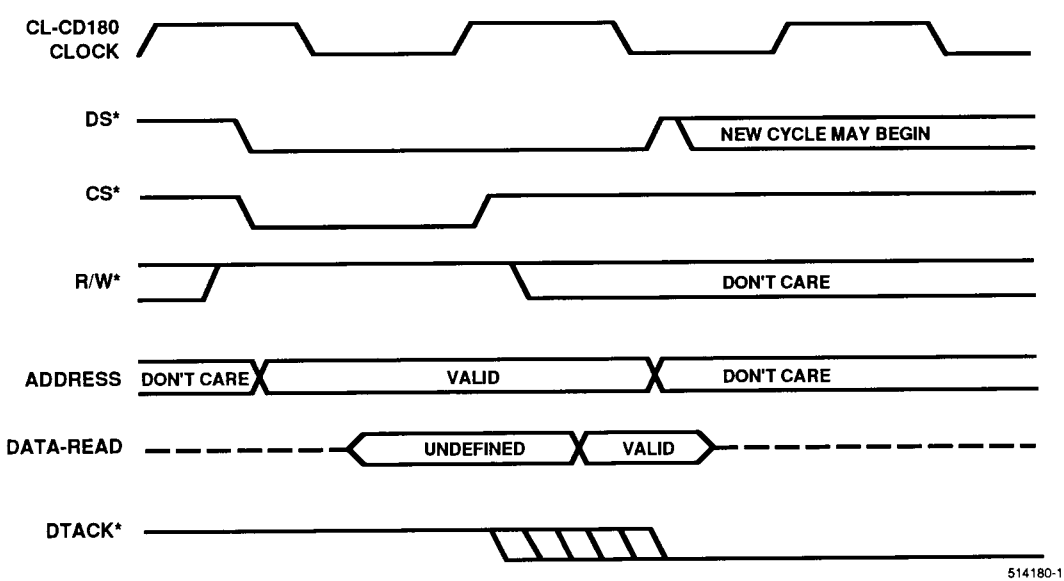


Figure 3–3. Typical Clocked Bus Interface

3.4 Interface Examples

There are some general design considerations when interfacing the CL-CD180 to any host environment.

The three Service Request Pins (IREQ1*, IREQ2*, and IREQ3*) may change at any time, and this can introduce metastability problems if the interrupt controller requires clocked signals. Care should be taken when designing to make sure that all signals are stable when needed.

The Service Request Pin of the type being acknowledged is negated at the end of the service acknowledgment bus cycle. Often, during the course of servicing one channel, another channel will reach a state where a request would assert, e.g., while servicing receive on channel one, channel two's FIFO fills. The Service Request Bits in the Service Request Status Register (SRSR) will not re-assert until approximately two clock periods after the host completes its write to the End-Of-Interrupt Register (EOIR). In polled or mixed-mode systems, to determine whether another service request of the same level is pending, and to make sure that the host does not re-read the SRSR too quickly, insert a No-Operation (or similar) instruction.

Performing an 'invalid' service acknowledgment bus cycle on the CL-CD180 is permissible, but it can cause problems in certain circumstances. An Invalid Service Acknowledgment is an acknowledgment for which there is no request pending.

If a service request acknowledgment bus cycle is performed by the host when no service request is pending, either of two things can happen. If the value on the address bus matches one of the three values in the three match (PILR) registers, and daisy chaining is enabled, the CL-CD180 assumes that another device down the daisy chain should receive the request, and asserts its IACKOUT* Pin. This propagates down the CL-CD180 chain until eventually the last CL-CD180 asserts its IACKOUT*. At this point, the system waits endlessly unless the bus cycle terminates. The best method is to connect the IACKOUT* of the last CL-CD180 in the chain to a bus-error input on the host. If there are multiple CL-CD180s that are not cascaded, the IACKOUT* Signals should be OR'ed together through a gate or a PAL.

If an acknowledgment occurs and the value on the address bus does not match any of the Match Registers, the first CL-CD180 in the chain does not pass it along or assert DTACK* and the system waits endlessly unless there is a bus time-out or other mechanism to detect this condition. In either of these circumstances, the 'value' on the data bus is likely to be FFh because the bus is floating (this is system dependent). To make a robust design, do not use FFh as a valid Global Service Vector Register (GSVR) value. If daisy chaining is not enabled, then the CL-CD180 will return a vector of '00' for invalid acknowledgments.

3.4.1 Interfacing to 80X86-Family Processors

The Intel 80X86 family processors often use the 8259A as the interrupt controller, that supplies its own vector during the INTA Cycle. The easiest way to interface the CL-CD180 to an Intel processor is by Mixed Mode, as described in Section 2.5.

There is one 'bug' in the 8259A to be aware of. The 8259A can change the prioritizing of its eight inputs, which can result in one of its acknowledge outputs going low briefly (~30 ns) if an input changes at a certain time. This typically happens if a higher-priority input to the 8259A asserts when the 8259A is about to issue an acknowledge to a lower-priority device. If this occurs at the beginning of a cycle, this brief pulse can cause the CL-CD180 (and other devices) to malfunction. Care should be taken to make sure that this does not happen. See *Intel 8259A Data Sheet* for details.

3.4.2 Interfacing to 680X0-Family Processors

The 68000-family interface is quite straightforward. The three service request lines go through a priority encoder to the 680X0 IPL inputs. The CL-CD180s IACKIN* Pin is driven by a decoder.

When the 680X0 performs an Interrupt Acknowledge Cycle, it drives its address lines A1, A2, and A3 with a three-bit value indicating the level being serviced. The other address lines are set to a 1. If the level being serviced corresponds to a level assigned to the CL-CD180, external decoding logic

should assert the CL-CD180 IACKIN* Pin. The value on address lines A0 to A7 has been programmed into the PILR registers, so the CL-CD180 recognizes the acknowledgment and proceeds as described in the Service Request Section 2.3.1.

All CL-CD180 service requests can also be routed to a single interrupt level by using a Mixed-mode Interface, as described in Section 2.5.

3.4.3 Interfacing to the VME Bus

The CL-CD180 can be directly interfaced to the VME Bus, and requires only a small amount of logic to complete the interface. This is necessary because service request acknowledgment works differently on the VME Bus than on the CL-CD180. VME defines seven levels of interrupts; each level can be shared among multiple VME cards. During an Interrupt Acknowledge Cycle, the VME Bus provides three bits on the address bus, indicating the level being acknowledged (A1-A3). Each VME card must pass along an interrupt on all levels it is not using but the CL-CD180 does not automatically pass an interrupt acknowledgment.

To recognize how this difference can cause a problem, suppose that the three Service Request lines from the CL-CD180 are connected to levels 7, 6, and 5 of the VME Bus (see Figure 3–4). Also, attach a 74XX244 so that during an Interrupt Acknowledgment Cycle provides an 8-bit code consisting of the three address bits plus five more hard-wired bits to the CL-CD180. Now, whenever an acknowledgment of a level 5, 6, or 7 interrupt occurs, the CL-CD180 either responds or passes the acknowledgment properly. If an acknowledgment occurs on levels 1-4, the daisy chain 'breaks' because the CL-CD180 does not recognize a match.

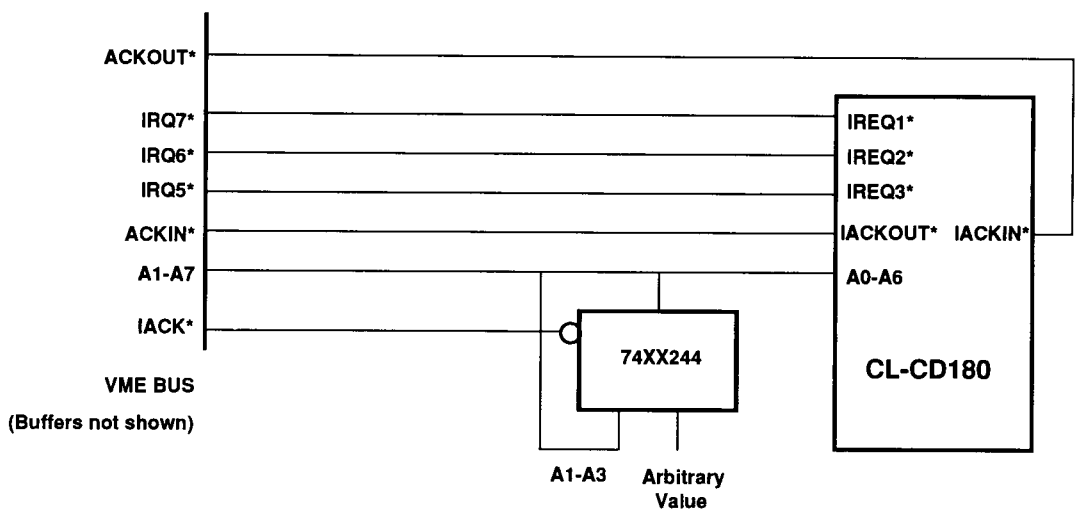
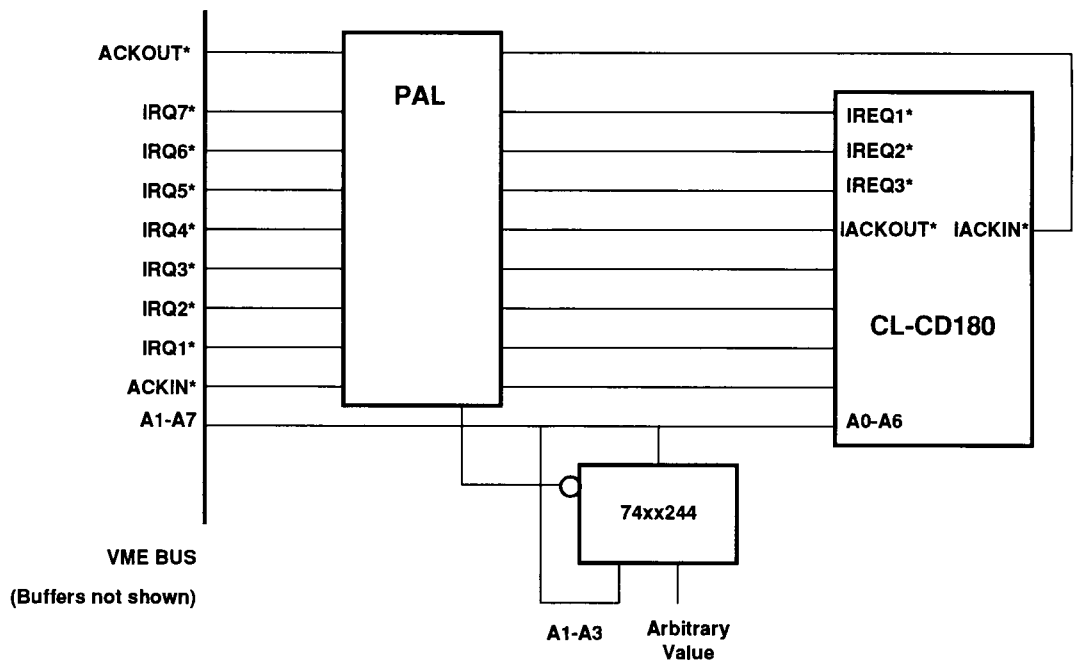


Figure 3–4. Incorrect VME Interface

This condition can be easily rectified, as shown in Figure 3–5. A PAL is used to assert IACKOUT* whenever IACKIN* occurs on a level not being used by the CL-CD180. The PAL is programmed for fixed levels. For example, if the current VME Bus Interrupt level is 1-4, the PAL asserts IACKOUT* whenever IACKIN* is active. If the current level is 5-7, the PAL asserts IACKOUT* when IACKOUT* from the CL-CD180 is active. If desired, the assignment of VME Interrupt levels to the CL-CD180 can be field-programmable by supplying additional inputs to the PAL, indicating the levels being used by the CL-CD180.



514180-17

Figure 3–5. Correct VME Interface

4. SERIAL INTERFACES

4.1 Receiver Operation

4.1.1 Basic Operation

All receivers are disabled upon master reset. To prepare a receiver, first initialize and then enable it. Once initialized and enabled, the receiver monitors the RxD Line and waits for a high-to-low transition, which indicates a Start Bit. This sampling is performed at one-eighth of the System-clock Rate regardless of the Programmed Bit Rate, and it provides accuracy of synchronization with the incoming data. See Figure 4–1 below for CL-CD180 bit synchronization. Once a transition is detected, the receiver checks the RxD Input state again (a half-bit time later) to validate that it is a Start Bit. A valid Start Bit is defined to be a 'space' or a logic '0'. If the RxD Input is no longer a 'space' then a false Start Bit is assumed, and the receiver resumes the search for a high-to-low transition. If a valid Start Bit is detected, the RxD Input is sampled at one-bit time intervals in the middle of the bit to ensure stable data. Characters are assembled according to the programmed content of the Channel Option Register (COR1). Valid character framing (presence of a Stop Bit), and Optional Parity Bits are checked. After a character is assembled, it is placed in a temporary Holding Register. Then the CL-CD180 processor checks for error conditions, FIFO overrun, and special character match before placing the character and its corresponding status into the Receive and Status FIFOs.

4.1.2 Receive FIFO Operation

Eight bytes of FIFO are assigned to each receiver for data storage, in addition to the Receive Holding Register and the Receive Shift Register. The CL-CD180 can be programmed to generate a service request once the number of data bytes received and stored in the FIFO reaches a programmed threshold. See Figure 4–2 for Receive Operation. The Receive FIFO Service Request threshold can be selected by programming the RxTH Bits 3:0 in the Channel Option Register 3. A service request threshold of one-to-eight characters can be selected. Once this threshold is defined, a service request will be automatically triggered when the condition is met. It is possible that by the time the host responds to the service request, there will be more data in the FIFO than the threshold level.

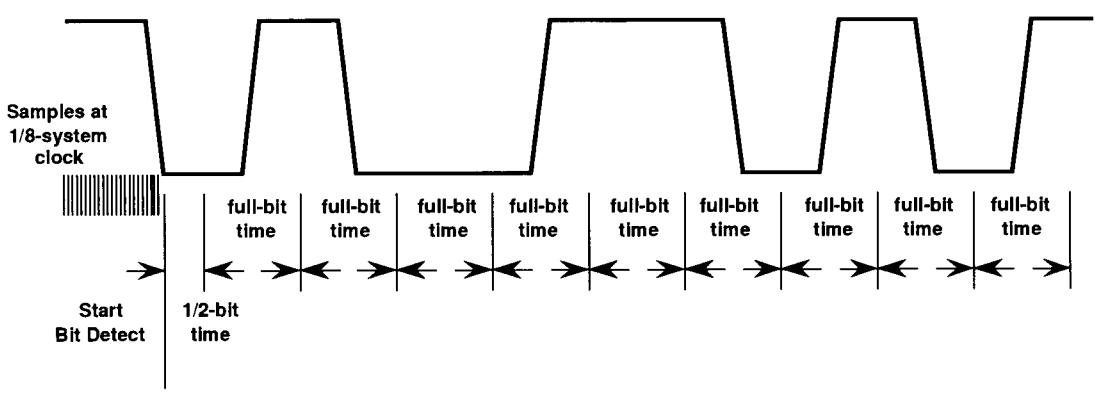


Figure 4–1. Bit Synchronization in CL-CD180

An overrun condition occurs when the new data arrives, but the Receive FIFO and the Receive Holding Register are both full. The new data is lost and the overrun indication is flagged on the character in the Holding Register. That character and its status including the overrun indication will eventually be transferred to the host by a Receive Exception Service Request. Note that this character is good, and is the last character received before the overrun occurred.

Receiver Service Requests are enabled or disabled via the Receive Data Bit in the Interrupt Enable Register (IER). Receive Data Bit, when set to a '1', enables service requests to be asserted for the above causes.

The Prescaler Period Counter is a 16-bit counter clocked by the system clock. If the system clock is a 10-MHz clock, the maximum count will establish a clock tick every 6.5536 ms. The Prescaler Period should be set to generate a minimum tick period of 1.0 ms. The Receive Time-out Counter is an 8-bit counter decremental on every tick of the Prescaler Period Counter. At the maximum count per tick, the maximum time-out period is 1.671 seconds.

The Receive Time-out is always enabled to transfer data when the Receive Data Service Request is enabled. From the system applications viewpoint, this time-out function is important for asynchronous data transmission. This is especially true when a FIFO is in use and a service request threshold for the FIFO is set greater than one character. The Timer Service Request will eliminate long response times when excessive delay between characters occurs caused either by the remote operator or due to the line being disabled. The 'No New Data' Timer Service Request, which occurs after all data is transferred to the host, may be used to manage transfers from the host's receive data buffers.

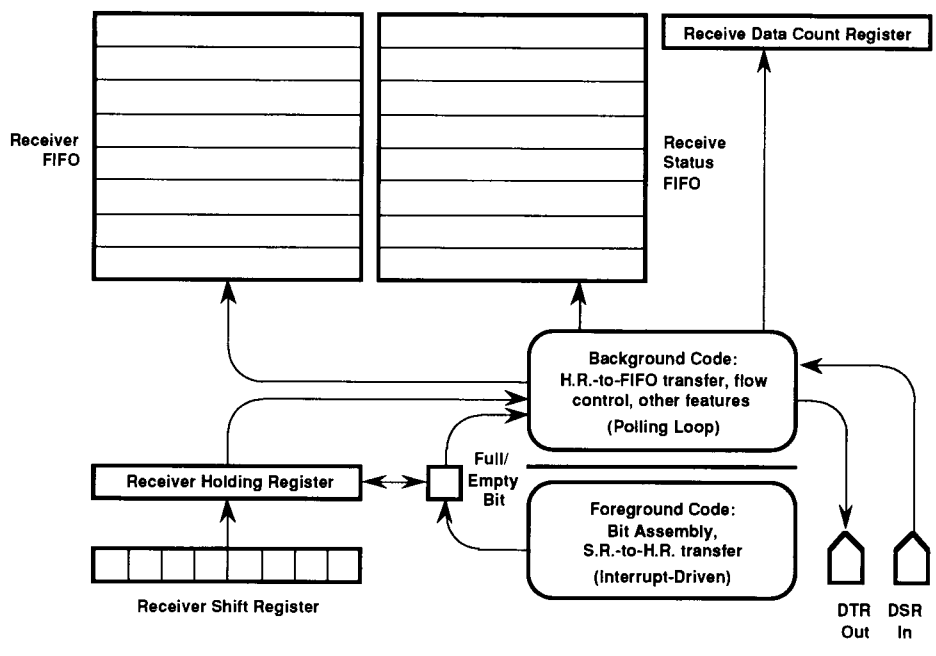


Figure 4–2. Receive Operation

4.1.3 FIFO Timer Operations

The CL-CD180 uses the Receive FIFO Timer for two purposes. The first is to avoid 'stuck' (or 'stale') data in the FIFO caused by not receiving enough characters to trip the threshold, which causes a service request to be issued. The second is to signal the host that there has been a relatively long pause in received data. It is useful for the host to know that 'no data has arrived lately' when managing relatively large I/O buffers. This event flushes the buffer up to the host for processing.

To avoid 'stuck' data, each time the CL-CD180 moves a character into a channel's Receive FIFO, it sets the channel's Receive FIFO Timer to the value contained in the channel's Receive Time-out Period Register (RTPR). If the timer expires before new data arrives, a Receive Good Data sub-type service request will be asserted for the channel if the Receive Data Enable Bit in the IER is set.

The other receive timer option is to generate a service request for the first Receive Data Time-out following the transfer of all data from the channel to the host. This is called the No New Data Time-out (NNDT). This service request is a Receive Exception sub-type with a status type of 'Time-out Exception'. There is no data character associated with the Time-out Exception status. This option can be enabled or disabled by controlling the NNDT Bit in the IER.

If enough data arrives to fill the Receive FIFO to the level set by the RxTh Bits in COR3, or if a special character arrives in the Receive FIFO and the RxSC Bit of IER is set, the channel will assert the Receive Data Service Request without waiting for the timer to expire.

If the timer times-out and the FIFO is not empty, the 'stale data' condition has occurred, and the device posts a Receive Good Data Interrupt. If the timer times-out and there is no data, two conditions are checked. First, a test is made to see if the feature is enabled, if it is true, then another flag is tested to make sure this is the first time the condition has occurred. If this is true, a Receive Exception Service Request is posted. (The NNDT internal flag is armed when the FIFO is emptied).

4.1.4 Receive Service Requests

The Receive Service Request is unique as it has two sub-types; i.e., it is capable of returning one of two different vectors during a service request acknowledge cycle. The two sub-types are Receive Good Data and Receive Exception. The reason there are two types within one category of service request is because while Good Data and Exceptions require different handling, they are both of equal priority and need to be serviced in the order they were received. Suppose, for example, two good characters are received, then an erroneous character, then another good character, then there must be a service request for the first two bytes of Good Data, then for the Exception, and then for more Good Data. If Exception Service Requests were at a different level, the erroneous character would be processed either before or after the Good Data, not in sequence as it should be. Receiver Service Requests will be invoked under several conditions.

Conditions that cause a Receive Good Data Service Request are:

- Receive FIFO threshold reached or exceeded
- Receive FIFO time-out — interval between character receptions exceeds time-out value

Conditions that cause a Receive Exception Service Request are:

- Receive erroneous data (parity error)
- Framing error (No Stop Bit)
- No data received time-out (optional)
- Special character detection
- Break detect

NOTE: Data cannot be read from the Receive FIFO or the Receive Status FIFO except when the CL-CD180 is within the context of a Receive Data Service Request for a specific channel.

4.1.5 Receive Good Data™ Service Request

A Receive Good Data Service Request is asserted for any of the following three conditions:

1) Receive FIFO threshold reached, and the FIFO contains Good Data.
2) Receive FIFO threshold not reached, but the FIFO contains Good Data and the Receive Data Timer times-out.
3) Receive FIFO threshold not reached, but the FIFO contains Good Data and the newly arrived data contains an exception condition.

When any of these conditions occur, the modified service request vector indicates to the host that the service request is for Good Data.

It is not necessary to take all or any of the available Good Data when a Good Data Service Request is received. If a host buffer is too full to accept eight bytes, a smaller number (even a '0') can be read. Service request context is then left, and the host buffer is dealt with first. The CL-CD180 will generate another Good Data Service Request when any of the three conditions listed above are met.

The CL-CD180 will immediately generate another service request if the condition that caused it in the first place remains true. If no data is read, this is always the case. If some, but not all of the available data is read, Conditions 1 and 2 will not be true, but Condition 3 may be true if an exception condition caused the Good Data Service Request. If this is a problem, one solution is to temporarily disable Receive Service Requests on that channel. To avoid FIFO overflow, do not delay handling the channel for too long.

4.1.6 Receive Exception Service Request

Unusual or exception conditions are reported to the host one character at a time through the Receive Exception Service Request. As with normal receive processing, the host determines the requesting channel by reading the GICR. It can then determine the specific exception(s) by reading the Receive Character Status Register before performing the appropriate action. Receive Exceptions are always one-byte deep; multiple bytes of exception conditions will cause multiple Receive Exception Service Requests.

For many exceptions it will not be necessary to read the Receive Data Register once the Receive Status Register is read. For example, if special character detection is enabled, and the service request is for recognition of a special character, the character is known by definition because the exception code indicates which character or character sequence was detected.

However, for every exception a byte is placed in the Data FIFO, even though the contents of that byte may be suspect data, and the byte is discarded at the end of the exception service routine regardless of whether it was read by the host or not. This is done to keep the Status and Data FIFOs in lockstep with each other. This is different in the case of a Receive Good Data Service Request where the user is free to read as many or as few bytes as desired.

Regardless of the number or type of exceptions occurring, they will be reported to the host one character at a time; i.e., the number-of-bytes value in the Receive Data Count Register is not meaningful. Since every error is reported individually, there is no Receive Time-out Exception generated if the only characters in the FIFOs are error or exception characters.

4.1.7 Types of Errors

There are four types of errors recognized by the CL-CD180: parity, framing, line break, and overrun. If parity checking is enabled, parity errors will be logged in the Status FIFO and the suspect data will be placed in the Receive Data FIFO. An error is also logged for framing, i.e., absence of a Stop Bit. In these cases, the suspect character is in the Receive Data FIFO and the appropriate status byte is placed in the Status FIFO.

When a line-break condition is recognized (zero data with zero parity, and no Stop Bit), one NULL (00) character is loaded into the Receive FIFO, and a break status is recorded in the Status FIFO. Note that if odd parity is set and the bits received are all zeroes, it is marked as both a break character and a parity error. Generally when a break character is received, pre-set parity error may be ignored. No further FIFO entries will be made until normal-character reception is resumed, i.e., a Start Bit is found. The line must go high and then back to low for this to occur.

Multiple errors in one byte are possible because the CL-CD180 evaluates the characters bit-by-bit as it receives them. For example, a parity error will be detected and flagged before the CL-CD180 recognizes that a framing error has occurred. Parity plus framing or parity plus break error can occur, but framing plus a break error cannot occur because, if a character is received with every bit equal to a '0', it is marked as a break character. If some bits are a '1', but the Stop Bit is missing, i.e., a '0', it is marked as a framing error. Thus, any one character cannot have both framing and break errors.

The length of the Stop Bit is not checked by CL-CD180. Any Stop Bit long enough to be sampled in mid-bit time as a '1' will be interpreted as a valid Stop Bit. In addition to all of the other errors, if an overrun occurs, the Overrun Error Bit will be set along with other error bits.

4.1.8 Types of Exceptions

4.1.8.1 Special Character Recognition

'Special Character Recognition' is a feature found only on the CL-CD180 and other Cirrus Logic data communications controllers. The on-chip processor compares every good character received with user-defined special characters stored in registers on the chip. Both single-character and two-character sequence recognition is possible. This capability has several applications, including In-Band Flow Control. Special-character matches are reported to the host via a Receive Exception Service Request.

Four Special Character Registers are provided per channel, allowing received characters to be compared to as many as four special characters. However, these four registers are shared between Receive Special Character Detection and the Send Special Character Command, so some planning is required for using these characters.

The full set of features and options available as part of Special Character Recognition allows for Xon/Xoff flow-control to be implemented transparently to the host, and at the same time, detect either of two other special characters in the data stream and alert the host of their arrival.

The user may individually enable any CL-CD180 channel to recognize special characters. There are six bits used to control the various recognition and flow-control modes.

The following four registers are used to control character recognition:

Bit Name	Register	Function
SCDE	COR3	Enables detection of special characters. Must be set for In-Band Flow Control to work.
RxSC	IER	Enables generation of service requests. Cannot be overridden by other bits. Does not need to be set for In-Band Flow Control to work.
XonCH	COR3	Controls single- versus double-character matching.
XoffCH	COR3	Controls single- versus double-character matching.

The following table shows the effects of XonCH and XoffCH:

XonCH	XoffCH	Characters matched
0	0	Match on: any of SCHR1-4
0	1	Match on: SCHR1 or SCHR3 or (SCHR2 and SCHR4)
1	0	Match on: (SCHR1 and SCHR3) or SCHR2 or SCHR4
1	1	Match on: (SCHR1 and SCHR3) or (SCHR2 and SCHR4)

NOTE: The two-character pairs may share a common first character; however, the same character must be programmed in both SCHR1 and SCHR2.

Single- versus double-character recognition is controlled by XonCH and XoffCH. If single-character compare is enabled, the CL-CD180 will compare data in the data stream against the four special characters stored in the Special Character Registers (SCHR1-4). If fewer than four special characters are required, the unused Special Character Register(s) should be disabled by duplicating the pattern to be matched in the unneeded register. When reporting a special character, the CL-CD180 always reports the lowest-number Special Character Register that matches.

To set up Special Character Recognition, first set the characters to be matched in Registers SCHR1-4, then set XonCH and XoffCH according to the length of match wanted. Set the SCDE Bit, and lastly enable service requests by setting RxSC.

Special characters are reported to the host by placing the appropriate status word in the Status FIFO and the recognized special character in the Receive Data FIFO. In the case of a two-character sequence, only the second character will be stored in the Receive FIFO. This is because there is room only for one character and preserving both is not needed as these characters are user-defined.

4.1.8.2 Flow-Control Characters

Automatic In-Band Flow Control of the CL-CD180 transmitter is a subset of the Special Character Recognition capability, so to understand both these features is important. Refer to Section 4.2 for Transmitter Operation. Flow-control characters and operation are programmable on a per-channel basis. This is important to operating systems that allow users to configure their own terminal settings independently.

Because the CL-CD180 performs flow-control functions before the data is passed to the host, the response time required of the host to avoid data overrun is greatly reduced. Additionally, the flow-control characters can be stripped from the data stream, relieving the host from processing them.

To use automatic flow-control, the Special Character Detection (SCDE) must be enabled via Bit 4 of Channel Option Register 3 (COR3). This causes all error-free received data to be compared for a match with the Special Character Registers (SCHR1-4). In addition, flow-control must be enabled via Transmit In-Band Enable (TxIBE, Bit 6) of COR2. This causes the special characters to be interpreted as flow-control characters. For single-character flow-control sequences, SCHR1 is used as Xon and SCHR2 as Xoff. SCHR3-4 are available for use as normal special-detect characters. If two-character sequences are enabled via XoffCH and XonCH (Bits 6 and 7) of COR3, SCHR1 and SCHR3 form the Xon sequence, and SCHR2 and SCHR4 form the Xoff sequence.

If flow-control characters are passed to the host, they are marked as special characters 1 or 2 in the Receive Channel Status Register (RCSR). If a two-character sequence is detected, it is compressed to the second character and a status indicating a match of the first character is set. A valid two-character sequence requires that both characters be received without error; if an error occurs on the second character the first character is treated as a normal character, and this does not affect non-flow control special character detection. Bits affecting flow control are summarized below.

Bit Name	Register	Function
SCDE	COR4	Enables Special Character Recognition.
TxIBE	COR2	Enables Automatic Transmitter Flow-Control.
FCT	COR3	Sets Transparency Mode of flow-control.

XonCH	XoffCH	Xon	Xoff
0	0	SCHR1	SCHR2
0	1	SCHR1	(SCHR2 and SCHR4)
1	0	(SCHR1 and SCHR3)	SCHR2
1	1	(SCHR1 and SCHR3)	(SCHR2 and SCHR4)

The FCT Bit controls whether flow-control characters are passed on to the host. It has meaning only when In-Band Flow Control is enabled, i.e., TxIBE is set. When the CL-CD180 receives a flow-control character or character sequence and FCT is a '0', it will start or stop the transmitter, as required, and pass the character onto the host as a Receive Exception. Since there is a one-to-one correspondence between the Status and Receive FIFO, the flow-control character detected will be stored in the Receive FIFO, and a status byte indicating special-character detect will be stored in the Status FIFO. If FCT is a '0', RxSC must be set to enable service requests to be issued to the host. Otherwise, flow-control characters cannot be passed as Receive Exceptions and will instead be passed as Good Data.

If the FCT Bit is a '1', the CL-CD180 will still start or stop the transmitter, as required, but the character(s) will be discarded, and no exception will be posted. In either case, the flow-control status of the transmitter (on or off) is maintained by the CL-CD180 in the Channel Control Status Register (CCSR).

The FCT Bit makes it possible to support 'escaping' of flow-control characters. Some systems follow a convention where two identical flow-control characters in a row indicates that flow control is not to be performed, but rather one flow-control character is to be kept in the normal received-data stream, and the other 'escape' character is to be discarded. If the CL-CD180 is in such a system, set the FCT Bit to a '0', allowing flow-control characters to pass onto the host. When the host detects two flow-control characters in a row, it simply restores the proper flow-control state of the channel and discards one of the characters. However, for most systems the FCT Bit can be set to a '1', reducing loading on the host.

4.1.8.3 No New Data Received Time-out

It is sometimes useful for the host to sense that 'no data has arrived lately', when managing relatively large I/O buffers. This event is used to flush the buffer up to the host for processing. One of the receive timer options, No New Data Time-out (NNDT), generates a service request for the first Receive Data Time-out following the transfer of all data from the channel to the host. This service request is a Receive Exception sub-type, and can be enabled or disabled by controlling the NNDT Bit in the IER. Refer to Figure 4–3 for the timer logic.

The timer is started only on data arrival. If the CL-CD180 processor determines that the Receive FIFO is empty, the timer has expired, and there has been a previous receipt of Good Data (and the timer feature is enabled), a Receive Exception will occur with a status indicating that a time-out has occurred.

If the last Receive Exception Service Request was triggered by a time-out (to avoid 'stale' data) the No New Data Time-out Service Request will occur immediately after the Data Transfer Service Request completes. If the last service request was triggered by reaching the threshold, the timer still has to expire so that some time will pass before the No New Data Service Request occurs. Likewise, if the last service request was triggered by some other error, such as parity, the timer still has to expire so that some time will pass before the No New Data Service Request occurs.

No New Data Function should not be confused with the time-out that occurs when there is Good Data in the FIFO but the threshold has not been reached and the timer expires. This event is a Receive Good Data Service Request, and not a Receive Exception event. Timing-out to transfer Good Data before it becomes 'stale' is standard, and it cannot be turned off by the user.

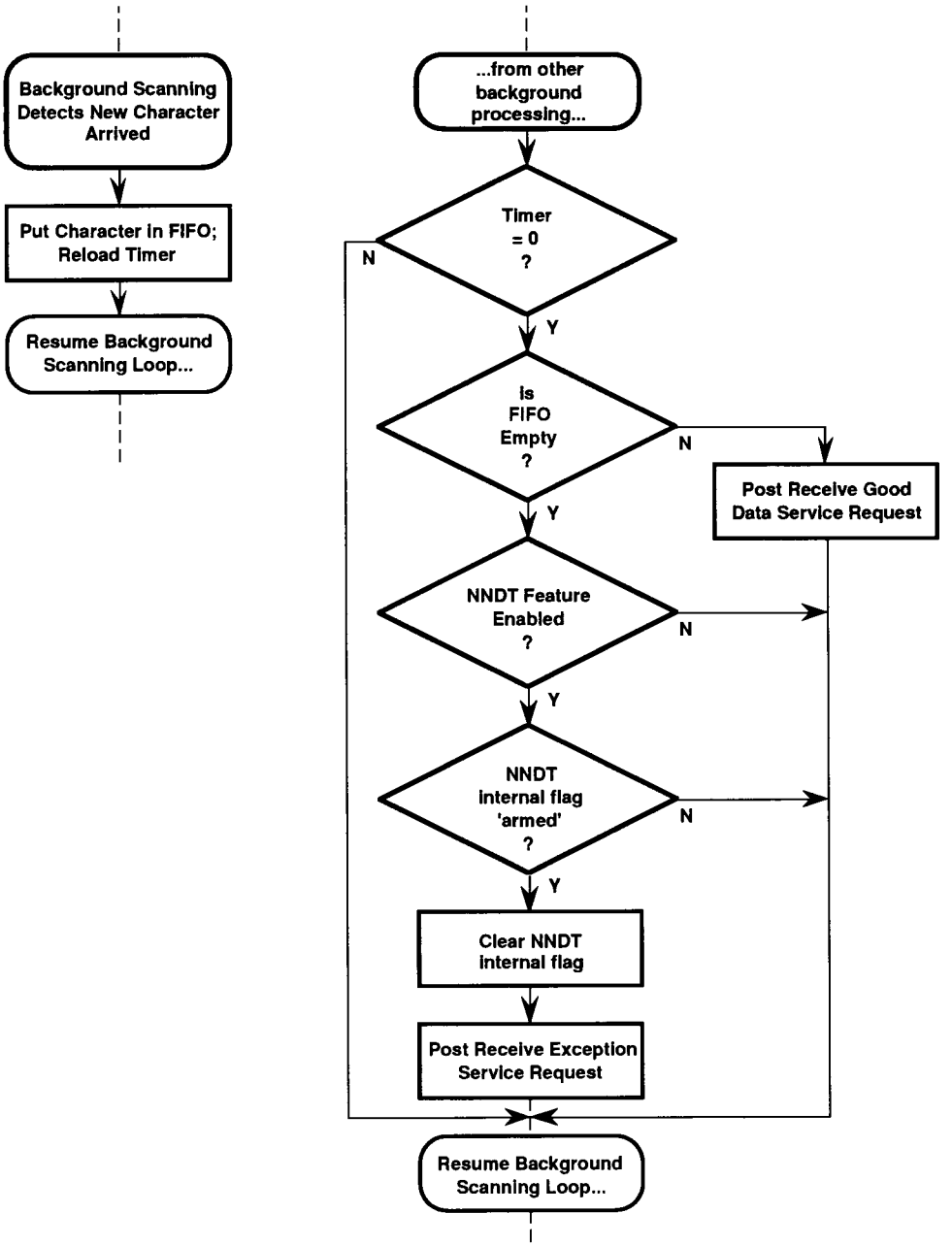


Figure 4–3. No New Data Timer Logic

4.1.9 Programming Notes

If a special condition (e.g., framing or a parity error) occurred on a special character, the CL-CD180 will not interpret this character as matched. Flow-control characters that are processed and discarded because FCT is set never cause an overrun.

Special Character Recognition only occurs on characters that have no other problems or errors. There is one case where the CL-CD180 will not find a special character even though the character has been correctly received. If a good character arrives as the ninth character (e.g., the FIFO is full), it stays in a Holding Register. If another character arrives, the good character in the Holding Register will have its status marked as 'overflow', indicating that it is the last good character received; however, it will not be recognized as a special character.

There are two cases where the CL-CD180 might not detect a two-character sequence. If the first character has been found, but no other character has been received for a long period of time and the Receive Time-out event occurs, no match will be found because the first character will have been flushed up to the host. If special-character detection is disabled by clearing SCDE just when the CL-CD180 has received the first two-character special-character sequence, but has not received the second character yet, the first character will be lost.

4.2 Transmitter Operation

4.2.1 Basic Operation

Refer to Figure 4–4 for a diagram of transmitter operation. Upon power-on reset, all transmitters are disabled with their Transmit Output held in the 'Mark' or a logic '1' condition. Other channel parameters are undefined. The minimum configuration of a channel for transmission consists of specifying the bit rate, parity, and number of Stop Bits. In-band and Out-of-Band Flow Control should also be set as desired. Next, set either (or both) of the service request enable bits. Then issue the Transmit Enable Command and either of two service request enable bits. For normal operation set the TxRDY Bit, which will cause a service request to be issued when the FIFO is empty. Since on power-up the FIFO is empty, a service request will be received (less than 1 millisecond), and at that time data can be transferred to the FIFO. Data can not be transferred to the FIFO as part of channel initialization; instead one has to be in the service-request routine to do this. Refer to the Section 2.3 for details.

Once the channel is initialized and serviced, and a character is written into the Transmit FIFO, the transmitter starts to transmit by first sending the Start Bit (space or a logic '0') followed by the data character according to predefined character length, least significant bit first. An optional parity bit (none, odd, even, or forced) is appended followed by the final Stop Bit (a logic '1' or a 'Mark'). The length of the Stop Bit can be one, one-and-a-half, two, or two-and-a-half bit-times long.

The transmitter will continue sending characters one after the other as long as the Transmit FIFO is not empty. When the Transmit FIFO becomes empty and the last character has been sent, the transmitter will stop transmission and will hold the TxD Output in the 'Mark' (1) condition. Transmission will resume as soon as there is another character in the FIFO.

In some cases it must be determined if the channel is completely done transmitting the last bit of the last character — for instance, before changing the bit rate. In such a case, the service request is to be issued only when the last character has been sent, rather than when the FIFO is empty. In this case, instead of setting the TxRDY Bit, set the TxMpty Bit. This will cause a service request to be issued only when the transmitter is completely empty.

For details on Transmitter Flow-control Operation, refer to the Section 4.3.

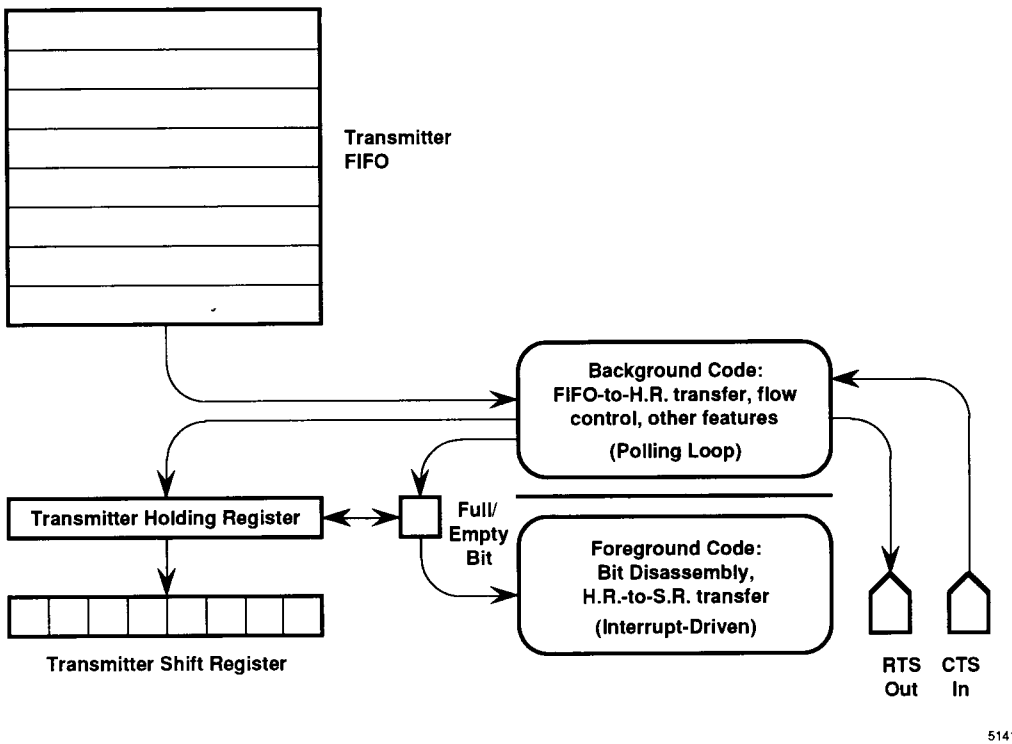


Figure 4–4. Transmitter Operation

4.2.2 FIFO Operation

An eight-byte FIFO is provided for each transmit channel. In addition to the eight-byte FIFO, the CL-CD180 also contains a Transmit Holding Register and the Transmit Shift Register for each channel. However, when servicing a Transmit Service Request, only up to eight characters can be written into the Transmit Data Register (TDR) consecutively.

4.2.3 Transmit Service Requests

Generating a Transmit Service Request depends on control bits in the Interrupt Enable Register (IER). Setting the TxRdy Bit of the IER specifies that a Transmit Service Request be generated when the FIFO is empty. When this condition occurs, there is still one character in the Transmit Holding Register and another character in the Transmit Shift Register. The host CPU, therefore, has up to two-character times to respond before the transmitter output goes into the idle (Mark) condition.

Setting the TxMpty Bit instead of the TxRdy Bit of the IER specifies that a Transmit Service Request be generated only when the FIFO, the Transmit Holding Register, and the Transmit Shift Register are empty. When this condition occurs, it means that all characters have been completely transmitted and the channel can now be re-configured. It is recommended that one of the two bits be set as needed, but do not set both bits at the same time.

End of a service request must be signalled to the CL-CD180 by writing to the End-of-Interrupt Register (EOIR).

4.2.4 Special Transmitter Commands

The CL-CD180 is capable of sending special characters preemptively (bypassing the FIFO): sending break characters and inserting delays or pauses either between characters or to lengthen a break. There are two basic mechanisms the CL-CD180 uses for these 'Send Special Character' and 'Embedded Transmit Command' functions.

4.2.5 Special Character Transmission Via Send Special Character Command

Selected special characters, or two-character sequences, may be transmitted preemptively by setting the appropriate bits in the Channel Command Register (CCR). The Send Special Character (SEND SP CH) Bit of the CCR, when set, initiates the Send Special Character Command. SSPC0-2 Bits of the CCR then specify which character or two-character sequence is to be used. The choice of a single- or two-character sequence is determined by the XonCH and XoffCH Bits of COR3.

When a Send Special Character Command is given, the CL-CD180 will insert the special character(s) into the data stream immediately following the current character in the Transmit Holding Register. Thus, it is ensured that the special character will begin transmitting within two-character times after the command is issued. The Send Special Character Command overrides all other flow-control modes, including the state of TXEN and CTS*. Generally this is the preferred case. However, sample CTS* or CD* in some applications to determine if it is okay to send a character before invoking the Send Special Character Command.

The CCR is reset by the CL-CD180 as an acknowledgment of the command. A new command must not be issued if the CCR contents are non-zero. A send special character command will be recognized and cleared within 125 µs (at 15 MHz, proportionally longer at lower clock speeds), unless a break is being sent. If a break is being sent, the special character will not be sent until after the break time is complete.

4.2.6 Embedded Transmit Commands

The CL-CD180 may be enabled to recognize certain 'escape' sequences as commands embedded in the Transmit Data Stream. These commands are issued to introduce a time delay between characters, to insert an idle period during the transmission, or to send a break on the line.

These capabilities are enabled on a per-channel basis by setting the Embedded Transmit Command (ETC) Bit in the Channel Option Register 2 (COR2). The 'null' (00) character is used as the controlling character to initiate the special action. To preserve data transparency, two mechanisms are provided to allow the null character to be sent as data. If the host must transmit a null character as data, either the ETC Mode may be disabled, or the null character may be preceded by a null, i.e., '00 00' will cause one-null character to be sent. If the ETC Bit is not set, the '00' character has no effect, and it may be sent as ordinary data. ETC Mode may be enabled or disabled 'on-the-fly'.

The CL-CD180 uses the Transmit Timer to generate time delays between characters in the output data stream. It is also used to extend the duration of a line-break transmit condition when the delay is inserted between the 'Start Break' and 'Stop Break' embedded-transmit commands. All of the timers count ticks are determined by the Prescaler Counter. The two eight-bit Prescaler Period Registers (PPRH and PPRL) determine the real-time

length of a tick. A tick is the period of the CL-CD180 System Clock Input (CLK) multiplied by the Period Registers' contents.

4.2.7 Sending Breaks

Line breaks may be sent by embedding the following sequences in the data stream (all values are given in Hex):

00 81 **Send Break:** Enter line-break condition for at least one character time. The line will enter the break condition and stay there until one of the following conditions is met:

1) Another character needs to be sent.
2) If the Insert Delay Special Character Sequence immediately follows the Send Break Sequence, the duration of the break transmission is extended by the amount of the programmed delay. The Insert Delay Sequence is: 00 82 xx. This inserts a delay of 'xx' (interpreted as an unsigned binary number) times the programmed timer 'tick' set by the Prescaler Period Registers. Multiple insert delay commands can be executed consecutively by the CL-CD180 to allow delays of arbitrarily long length. If 'xx' is a zero, no delay is inserted.
3) The Stop Break Sequence '00 83' is encountered next. This sequence is optional, and exists to provide a way to terminate a break without actually sending another character. If another character is being sent anyway, no Stop Break is required.

If there is no more data to be sent, the TxD Pin remains in the state it was left in by the last character. Since the Stop Bit is always a '1', the line will be left in the idle state after any character, except for the break character. The break character leaves the line in the '0' state until more data needs to be sent. Long breaks can be sent by simply sending one break and then waiting. To terminate the break, send the Stop Break Sequence or send another character.

Sending long breaks has precedence over the Send Special Character Command, i.e., the time delay duration must pass before the special character will be sent.

4.2.8 Sending Inter-Character Delays

In some applications it is desirable to pause between characters. For example, certain types of electro-mechanical teletype equipment cannot handle characters continuously at their specified bit rate. To accommodate this, the CL-CD180 allows insertion of a delay between characters.

The user embeds an escape sequence into the Output Data Stream to generate delays between characters. When the CL-CD180 encounters the Insert Delay Escape Sequence, it sets the Transmit Timer to the value contained in the Escape Sequence. When the timer expires, the CL-CD180 loads the next character into the Transmit Shift Register and resumes output (unless the next character begins another Escape Sequence). The Escape Sequence for an inserted delay consists of three characters: '00', '82', and 'tt'. The time-out value 'tt' is expressed in timer ticks.

4.2.9 Summary of Special Transmitter Commands

The ETC Bit in COR2 must be set to enable the following functions:

Char. Sequence	Effect
00 00	Send one-null character.
00 81h	Send one-character time of line break.
00 82h xxh	Delay for 'xx' prescaler time ticks (i.e., Transmit Timer Value is 'xx').
00 83h	Stop break.

4.3 Flow Control

Variations in response times and system data transfer rates between systems communicating across asynchronous interfaces give rise to a need to control the flow of data between them. Systems typically are implemented with a receive buffer for temporary storage of data. When this buffer is nearly full, the receiving computer 'flow-controls' the remote transmitter. When, after processing the existing data, more buffer space is available for the receive process, the receiving computer signals the remote to resume transmission.

Flow control is implemented in one of two ways — 'out-of-band' or 'in-band' signaling. Out-of-band signaling is a hardware-based mechanism, performed via extra wires such as the RTS/CTS and DSR/DTR pairs. It has the advantage of complete independence from the data stream. However, it is not always possible to provide all of the wires necessary to support Out-of-Band Flow Control. Also standards for implementing Out-of-Band Flow Control vary widely.

In-Band Flow Control works by inserting special flow-control characters into the stream of data being sent. It has the advantage that only the data circuit is required, thus only two wires are needed. The disadvantage of In-Band Flow Control is that the two communicating computers must perform additional functions, specifically, they must monitor the data stream for flow-control characters and take the appropriate action. This can be quite burdensome because the host computer that receives a flow-control command must recognize this event quickly and respond in a timely manner to avoid overrun at the remote receiver.

Although there are advantages and disadvantages to each system, in general the trend is toward In-Band Flow Control. This is because it is more useful than Out-of-Band Flow Control over a wider range of applications, such as communication via modems.

The CL-CD180 provides significant performance advantages over conventional solutions during both the receive processing of and the transmission of flow-control characters. It does this by handling almost all flow control automatically, without host intervention. It also provides tools to make host intervention, when required, much easier. Because the CL-CD180 performs flow-control functions automatically, before the data is passed to the host, the response time required of the host is substantially reduced. The possibility of data overrun is also reduced. Additionally, the flow-control characters themselves can be stripped from the data stream, relieving the host from processing them. The flow-control status of the transmitter is always available to the Host as a bit in the Channel Control Status Register (CCSR).

4.3.1 Receiver Flow Control

The CL-CD180 provides both In-Band (Xon/Xoff)) and Out-of-Band Flow Control functions for ensuring that the receiver does not overflow. In-Band Flow Control is semi-automatic and helps the host manage its buffer size. Out-of-Band Flow Control is fully automatic and can be used to prevent the CL-CD180 Receive FIFO from overflowing. Figure 4–5 diagrams the receiver flow-control logic.

When the CL-CD180 receiver is too busy, the transmitter can be used to send Xoff/Xon to the remote device. This Receiver Software (In-Band) Flow Control is covered in Section 4.3.3.

The CL-CD180 transmitter can be controlled by the remote device. This Transmitter Software (In-Band) Flow Control is covered in Section 4.3.6.

The current flow-control status is always available to the host. It is stored in the Channel Control Status Register (CCSR). Two bits, Receive Flow-on and Receive Flow-off, show whether the last flow-control command sent by the CL-CD180 was on or off. As long as the receiver is enabled, the CL-CD180 will continue to receive any data sent regardless of whether it has requested the remote to shut off.

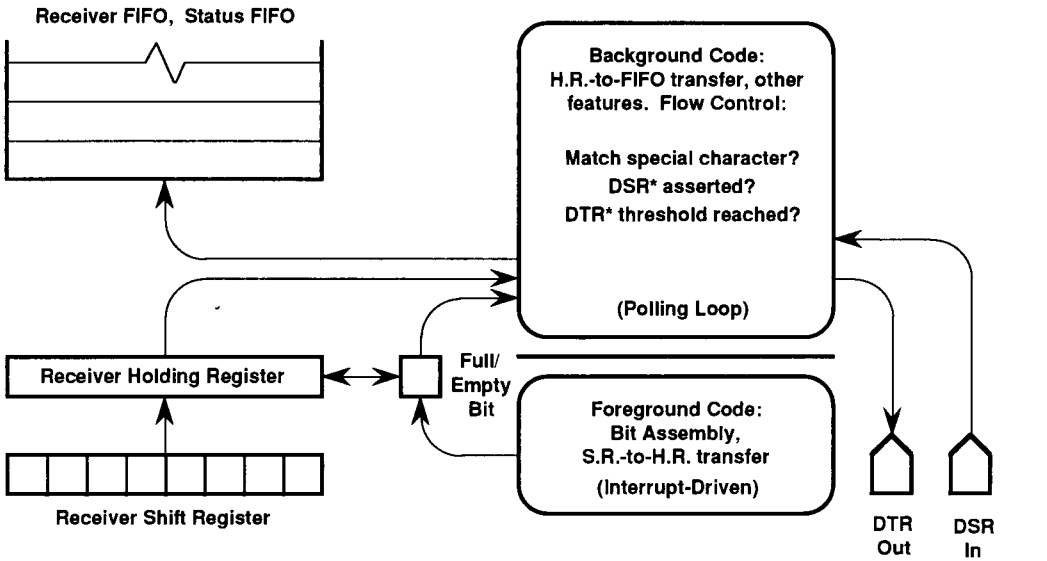


Figure 4–5. Receiver Flow-Control Logic

4.3.2 Receiver Hardware (Out-of-Band) Flow Control

Out-of-Band Flow Control uses the Modem Handshake Signal (DTR*) to control the flow of data. Whenever the Receive FIFO reaches a user-defined threshold, DTR* will be negated. This event can be used to signal the remote to stop sending characters. The threshold is set by four bits in the Modem Control Option Register 1, and can be any level from one to eight, or disabled. The DTR* Pin will also be negated whenever DTR* Mode is set and the channel is disabled or reset. If DTR* Mode is not set, the DTR* Pin is not changed by the CL-CD180, and remains at whatever value the host sets it to.

While it is possible to set the DTR* threshold lower than the service request threshold, the part will operate as though the DTR* threshold was the same as the service request threshold. If the DTR* threshold is set lower, it will be ignored, and DTR* will negate when the service request threshold is reached. If desired, set the DTR* threshold to a '1', and then it will 'track' the other threshold automatically.

The receiver monitors the state of DSR* (if enabled) and ignores data on the Receive Data Pin if DSR* is negated. This feature is controlled by the DsrAE Bit, Bit 0, of Channel Option Register 2 (COR2).

4.3.3 Receiver Software (In-Band) Flow Control

Host receive buffers often cannot keep pace with data being received. The CL-CD180 transmitter can be used to send flow-control characters to the remote device. This avoids over-flowing the receive buffers in the host. However, transmitting flow-control characters is an additional complication and source of delay when using conventional devices. As the host's receive buffer becomes full, the transmit process must be flagged to insert a flow-control character (or sequence) in the Transmit Data Stream. Any data already in the Transmit FIFO will be transmitted ahead of the flow-control character, increasing the response time at the remote end.

With the CL-CD180, In-Band Flow Control of the remote system is semi-automatic; two commands

(Send Xon, Send Xoff) can be issued by the host whenever the host wants to flow-control the remote. These special commands make host programming and buffer management easier because it allows the flow-control character(s) to be sent as the next character, regardless of the contents of the Transmit FIFO or host transmit buffers.

Flow-control characters are transmitted via the send special character command in the Channel Control Register (CCR). The lower-three bits in the command determine which of the four-special characters are to be sent. If two-character flow control sequences are enabled, requesting either SCHR1 or SCHR2 causes the appropriate two-character sequence to be transmitted. Refer to Section 4.2.5 for Special Character Definition details. Special characters are transmitted regardless of the state of transmit enable or transmit flow control. Transmitting flow-control characters can be handled independently of the current state of the transmit channel. In sending special characters, the CL-CD180 bypasses any data already in the Transmit FIFO, thereby minimizing delay in transmitting flow-control characters. The maximum delay is two-character times. However, if a break is currently being transmitted, the CL-CD180 will wait for the break transmission to terminate before the special character is transmitted, regardless of the length of the break.

The CL-CD180 keeps a copy of the current state of the receive flow in the CCSR. Two bits are used to indicate the current state of the channel regarding flow control: RxFloff and RxFlon. RxFloff and RxFlon are meaningful only when the CL-CD180 is flow-controlling the remote. Whenever an Xoff is transmitted, RxFlon is cleared and RxFloff is set. When a subsequent Xon is transmitted, RxFloff is cleared and RxFlon is set. When data is received from the remote, RxFlon is cleared.

The '00' state is provided as an aid to the programmer in determining whether there might be a problem in a communications link. If RxFlon remains set during normal operation, it could indicate that the remote did not correctly receive the last Xon.

If flow-control characters are sent by the host by embedding them in the Transmit FIFO rather than using the Send Special Character Function, the CL-CD180 flow-control logic does not sense them, and the CCSR is not affected.

The table below summarizes the meaning of RxFloff and RxFlon.

RxFloff	RxFlon	Meaning
1	1	Illegal Mode.
1	0	Xoff is last flow-control character sent (flow off).
0	1	Xon is last flow-control character sent (flow on).
0	0	Flow is on, data has been received.

4.3.4 Transmitter Flow Control

The CL-CD180 provides both automatic In-Band (Xon/Xoff) and Out-of-Band Flow Control functions. In-Band Flow Control recognizes special characters or character sequences for Xon and Xoff control embedded in the data stream. Out-of-Band Flow Control uses the modem handshake signals, RTS/CTS, to control the flow of data. Both types of flow control are implemented between the Transmit FIFO and the Transmit Holding Register, not between the Transmit Holding Register and the Transmit Shift Register. Figure 4–6 diagrams the transmitter flow-control logic.

All automatic flow-control functions are controlled by bits in Channel Option Register 2 (COR2), except DTR threshold, which is controlled by Modem Change Option Register 1 (MCOR1). Channel enable and flow-control status is stored in the Channel Control Status Register (CCSR). A TxEn Bit shows the enabled status of the channel's transmitter. Two bits, TxFloff and TxFlon, are used to indicate the current state of the channels' flow control.

Once the Automatic Flow-Control Modes are invoked by the host, all actions will be transparent to the host. If receipt of flow-control characters by the host is not desired, the Flow-Control Transparency Bit of COR3 may be set to not pass received flow-control characters onto the host. If TxIBE is set, the CL-CD180 will implement the flow-control function on the transmitter regardless of the FCT Mode. The host can review the status of the channel by reading the Channel Control Status Register.

If flow-control status is needed by the host, the SCDE and RxSC Control Bits must be set and the FCT Bit must not be set. A special character detect status and the special character will be presented to the host by a Receive Exception Service Request. If the host wishes to manually flow-control the transmitter, it can do so by using the TxEn Bit, which will stop transmission after the current character completes.

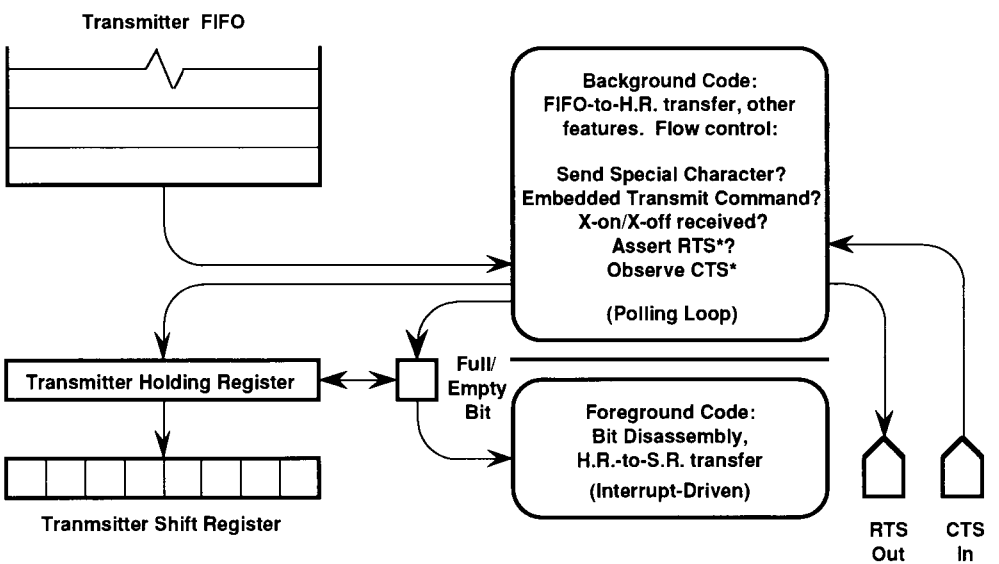


Figure 4–6. Transmitter Flow-Control Logic

4.3.5 Transmitter Hardware (Out-of-Band) Flow Control

Transmit out-of-band flow control is performed automatically by the CL-CD180 via the CTS* Pin, if the CTS Auto Enable (CtsAE) Mode is enabled in Bit 1 of COR2. In this mode, before a character from the FIFO is transmitted, the CTS* Pin will be tested, and, if inactive, transmission will be delayed. Since flow control is implemented between the FIFO and the Transmit Holding Register, when CTS* is negated, it is possible to get both the current character being sent and the character in the Transmit Holding Register.

However, the Send Special Character Command (e.g., Xon and Xoff) will override CTS* inactive. This is generally preferred; however, in some applications sample CTS* or CD* before sending a special character.

To complete the handshake with a remote device, an RTS Automatic Output (RtsAO, Bit 2) Mode is also provided. This causes the RTS Pin to be asserted throughout any data transmission: normal, break, and special characters. The RTS Pin is activated whenever there is data in the FIFO and transmitter registers. It is held active until after the last Stop Bit of the last character is transmitted.

4.3.6 Transmitter Software (In-Band) Flow Control

The CL-CD180 transmitter can be programmed to respond automatically to flow-control characters received by the receiver. This feature requires no host assistance and substantially reduces host processing requirements. If this Automatic Mode is enabled, when the remote unit transmits an Xoff character to the CL-CD180 (to prompt the CL-CD180 to suspend transmission), the CL-CD180 will terminate the transmission. The CL-CD180 may require approximately 500 microseconds (~2 character-times at 38.4 kbps) after receipt of the Stop Bit to recognize that the character it has received is a flow-control character and set its internal flag to stop transmission. Transmission actually stops as soon as the characters in the Transmit Shift Register and Transmit Holding Register are shifted out.

To enable In-Band Flow Control, two bits must be set. First, the Special Character Detection (SCDE) must be enabled via Bit 4 of Channel Option Register 3 (COR3). This causes all error-free received data to be compared for a match with the Special Character Registers (SCHR1-4). Second, flow control is enabled via Transmit In-Band Enable

(TxIBE, Bit 6) of COR2, the special characters are interpreted as flow-control characters.

Different flow-control protocols use either single- or two-character sequences for the Xon and Xoff functions. For single-character flow-control sequences SCHR1 is used as Xon, SCHR2 as Xoff, and SCHR3-4 as normal special detect characters. If two-character sequences are enabled, via XoffCH and XonCH (Bits 6 and 7) of COR3, SCHR1 and SCHR3 form the Xon sequence and SCHR2 and SCHR4 form the Xoff sequence.

Many operating systems allow users to define their own terminal's flow-control settings independently. The CL-CD180 allows flow-control characters to be programmed on a per-channel basis.

The FCT Bit controls whether flow-control characters are passed on to the host. When the CL-CD180 receives a flow-control character or character sequence and FCT is a '0', it will start or stop the transmitter as required, and pass the character on to the host as a Receive Exception Service Request. Since there is a one-to-one correspondence between the Status FIFO and the Receive Data FIFO, the flow-control character detected will be stored in the Receive Data FIFO, and a status byte, indicating special character detect, will be stored in the Status FIFO.

If the FCT Bit is a '1', the CL-CD180 will still start or stop the transmitter as required, but the character will be discarded, and no exception will be posted. In either case, the flow-control status of the transmitter (on or off) is maintained by the CL-CD180 in the Channel Control Status Register (CCSR).

If flow-control characters are passed to the host, they are marked as special characters 1 or 2 in the Receive Channel Status Register (RCSR). If a two-character sequence is detected, it is compressed to the second character and a status indicating a match of the first character is set. A valid two-character sequence requires that both characters be received without error. If an error occurs on the second character, the first character is treated as a normal character, and the second character is reported as an error via a Receive Exception Service Request.

Bits affecting flow control are summarized in the table below:

Bit Name	Register	Function
SCDE	COR4	Enables Special Character Recognition.
TxIBE	COR2	Enables Automatic-transmitter Flow Control.
FCT	COR3	Sets Transparency Mode of flow control.
IXM	COR2	Sets implied Xon Mode

XonCH	XoffCH	Xon	Xoff
0	0	SCHR1	SCHR2
0	1	SCHR1	(SCHR2 and SCHR4)
1	0	(SCHR1 and SCHR3)	SCHR2
1	1	(SCHR1 and SCHR3)	(SCHR2 and SCHR4)

The remote device can signal the CL-CD180 to resume transmission in one of two ways depending on the setting of the Implied Xon Mode (IXM) option Bit COR2. When the IXM Bit is set, the CL-CD180 will resume transmission upon receipt of any character, i.e., each character is an implied Xon. In Implied Xon Mode it is assumed that if the remote is capable of transmitting data, it is able to receive as well. If a character is treated as an implied Xon, no special status is recorded in the RCSR, and the TxFlon Bit is not set in the CCSR. An implied Xon character will not be stripped if flow-control transparency is enabled.

When the IXM Bit is not set, the CL-CD180 will only resume transmission upon receipt of an Xon character. In addition, the host may force a resumption of transmission by issuing a Transmit Enable Command, which will clear the TxFloff Bit.

The Xon and Xoff characters or character sequences are equal in a Toggle Mode. There is no special bit to enable this mode. The CL-CD180 detects this mode whenever the Xon character equals the Xoff character, and it implements Toggle Mode automatically.

In Toggle Mode, whenever the special character is received, the current state of flow control is toggled. If flow control transparency is set, the character is dropped. If not in flow-control transparency, the character is passed to the host. If it is a single character, the special character status is '1' and the character is put in the Receive Data FIFO. In two-character sequence, the second character is placed in the Receive Data FIFO along with special character '1' in the Status FIFO.

The TxFloff and TxFlon Bits indicate channel status when the remote device is flow-controlling the CL-CD180 transmitter. When the remote requests the CL-CD180 to stop transmission, the CL-CD180 will set the TxFloff Status Bit in the CCSR. If TxFloff is set, the last flow-control character received was a flow-off. When the remote sends an explicit flow-on character, the CL-CD180 will clear the TxFloff Bit, and set the TxFlon Bit. (If flow is resumed because of implied Xon, TxFloff will be cleared, but TxFlon will not be set). When the CL-CD180 resumes transmission, the TxFlon Bit will be cleared. Transmit Flow Status Bits will also be cleared by enabling or disabling the transmitter or resetting the channel.

This is summarized in the table below:

TxFloff	TxFlon	Meaning
1	1	Illegal.
1	0	Transmitter is flow-controlled off.
0	1	Transmitter on, no data sent yet.
0	0	Transmitter on, CL-CD180 has sent data, or implied Xon has occurred. This is also the 'normal' state of these bits when flow control is not being used.

4.4 Modem Signals and General-Purpose I/O

Each channel of the CL-CD180 has four pins that can be used either as modem-control or general-purpose input/output pins. The modem signal names assigned to these four pins have been chosen to provide an easy reference for systems designers. In fact, they are all simply general purpose inputs and outputs (if automatic out-of-band flow-control is not used) that can be individually controlled via the modem signal value register(s). Since they are general purpose, system designers may choose to connect the pins in any way that suits the application.

However, when the system software design chooses to make use of the automatic out-of-band flow control with the pins, then the signal naming convention no longer holds true in some cases, depending on whether the device is used as DCE or DTE. In this case, it is best to think of the pins in terms of their actual uses within the CL-CD180 and connect them accordingly, without regard to their names. The RTS* and CTS* Pins are associated with the transmitter and the DTR* and DSR* Pins are associated with the receiver. The table below shows Cirrus Logic's recommended signal hook-up if automatic, out-of-band flow control is desired.

DCE	DTE	CL-CD180 Pins	Out-of-Band Flow Control
CTS		DTR	Signal remote to transmit
RTS			Not implemented in this direction
	RTS	RTS	Request remote permission to transmit
	CTS	CTS	Enable transmitter

For example, if the CL-CD180 is designed to be a DCE and automatic out-of-band flow control is desired, the pin labeled DTR should be connected to remote CTS input. If the CL-CD180 is to be used as the DTE side, then the CL-CD180 CTS output would be connected to the remote CTS input.

Note that if automatic out-of-band flow control is implemented, the activity of DTR and DSR Pins do not implement the function assigned to those signal names by the signalling conventions of the CCITT and other standards organization. These names would only apply to these pins if they are under program control and not under automatic CL-CD180 control. In fact, the "DTR" function, as defined, enables the modem to go on- and off-line, depending on the state of the pin. If automatic control is used, then DTR would go inactive when the receive FIFO reached the programmed threshold thus causing the modem to drop the connection (carrier) to the remote, which would not be the correct function. Refer to Section 4.3 for details on operation of modem pins in flow-control applications.

Modem Control Pins	Function
RTS*	Request to Send (general-purpose output).
CTS*	Clear to Send (general-purpose input).
DTR*	Data Terminal Ready (carrier detect/general-purpose input/output†).
DSR*	Data Set Ready (general-purpose input).
CD*	Carrier Detect (general-purpose input†).

† Depends on the setting of DTRSEL input: if DTRSEL = '1', DTR/CD Pin is output and thus DTR*; if DTRSEL = '0', DTR/CD Pin is input and thus CD*.

Modem pins are implemented as I/O ports accessible by either the CL-CD180 processor or the host. The modem pins are not connected directly to the transmit or receive hardware. When a user programs out-of-band modem functions to be active, the CL-CD180 processor will read from and write to these pins. Specifically, when RTS* and CTS* are being used for transmit flow control, the CL-CD180 processor will assert RTS* and sense CTS*, as required. Likewise, when configured to do so, the Receive FIFO will negate DTR* when full. The host should not be allowed to re-assert it inadvertently. The host is not 'locked out' of accessing these bits; care should be taken so that these bits are not written to, causing the system to malfunction.

The user has direct control over the RTS* and DTR* Outputs and can sense the state of CTS*, CD*, and DSR* Inputs through the Modem Signal Value Register (MSVR). Since the host is accessing these pins directly, there is no delay in the host's ability to detect a level change. DTR* and CD* depend on the state of the DTRSEL input.

When the CL-CD180 is programmed to detect level changes and generate service requests when level changes occur, it does so in firmware by reading the pins and comparing to a previously stored value. This function is performed in the main timing loop of the firmware; the maximum time required to detect a level change under worst-case conditions is approximately 2 milliseconds. When the CL-CD180 is performing this function, the modem pins are periodically sampled rather than continuously monitored; as such they have very little sensitivity to noise, which is desirable in data communication applications. However, in extremely noisy applications, re-read a modem line which has caused a Modem Signal Change Service Request to verify that it has indeed changed and is not merely malfunctioning. This will eliminate even the slight possibility of a noise pulse causing erratic operation.

When the CL-CD180 is monitoring modem pins to control transmit or receive functions, it does not rely on the previously stored value, but checks the pins at the appropriate time. Thus, there is very little delay in this response. For example, before deciding to transmit another character, it will examine the CTS* Pin at that time. (The CL-CD180 makes this decision when moving characters from the FIFO to the Holding Register, not from the Holding Register to the Shift Register.) Refer to Section 4.3 for flow-control details.

Note that the logical sense of the modem bits is inverted; i.e., writing a '1' to the MSVR causes the output pin to go to nominal zero volts. Likewise, a low-voltage input will be sensed as a '1'.

4.4.1 Generating Service Requests with Modem Pins

The CL-CD180 can generate service requests when any one of the input pins changes state. Either or both edges may be detected by setting bits in the two Modem Change Option Registers (MCOR1 and MCOR2). For each pin, the user can individually enable on-to-off or off-to-on transition detection of the inputs. When the CL-CD180 detects such a transition, it sets the corresponding bit in the Modem Change Register. If the corresponding bit in the channel's Interrupt Enable Register is set, the CL-CD180 will assert its IREQ1* Output. The user must clear the Modem Change Register during the service request service routine before writing to the EOIR.

The CL-CD180 performs this task by reading the modem input signals and comparing the current value with the value read in the last pass through the outer scanning loop. Because this is the lowest-priority event in the CL-CD180 scanning loop, changes may not be detected unless they are several hundred microseconds long. Modem Input Pins can be used for purposes such as detecting the closing of a switch. However, the relatively slow speed of response should be taken into account when using Modem Input Pins for this purpose. The CL-CD180 does not latch the Modem Input Signals.

4.4.2 Using Modem Pins as General-Purpose I/O

Since the modem pins can be directly accessed by the host, they can be used as general-purpose I/O pins if they are not needed for flow control or modem interfacing. Simply read from and write to them as any I/O port.

4.5 Testing the CL-CD180 — Loopback Tests

The CL-CD180 performs a basic internal self-test whenever it is reset. This test provides a reasonable degree of confidence that the CL-CD180 is functioning satisfactorily. There are two additional tests that can be performed by the user to further ensure complete functionality. These two test modes are Local and Remote Loopback. Used together with diagnostic firmware in the host system, the Loopback Modes provide very thorough test coverage of all CL-CD180 functional blocks: the CL-CD180 processor, ROM, RAM, bus interface, transmitters/receivers, and random logic.

Local Loopback Mode

Local Loopback Mode is a 'silent' loopback, i.e., data being sent by the transmitter is internally connected to the receiver without reaching the external TxD Pin. Generally, this is advantageous because it allows diagnostic software to operate without causing unwanted effects on any remote device that may be connected to the serial line. During local loopback, the TxD Pin is in the 'mark' (a logic '1') state. If non-silent loopback is also needed, it can be easily implemented externally with an AND gate or a jumper plug on the serial connector.

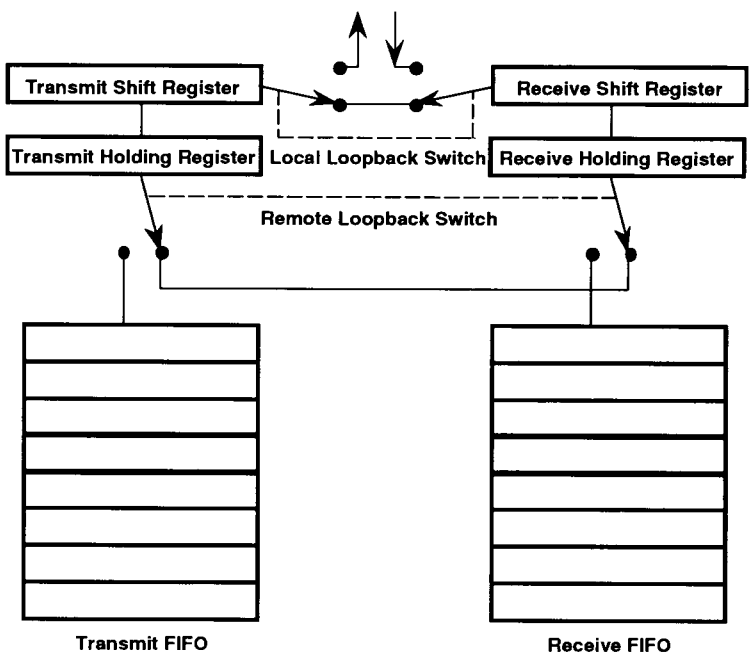
Local Loopback Mode is invoked by setting the LLM Bit in the Channel Option Register 2 (COR2) and then issuing a channel command to tell the CL-CD180 that COR2 has changed. When in this mode, the channels TxD Output is internally looped back to the channel's RxD Input. However, all other channel parameters including modem pins continue to work independently and normally. Receive special character recognition, overflow handling, and other options may be tested by using the Local Loopback Mode and transmitting the appropriate character sequences. As shown in Figure 4–7, the loopback connection is directly from the TxD Signal to the RxD Signal, i.e., all transmit and receive logic is tested except the actual I/O buffers.

Remote Loopback Mode

Remote Loopback Mode is provided to support testing of devices connected to the serial lines. Remote Loopback is invoked by setting the RLM Bit in the Channel Option Register 2 (COR2). When in this mode, the CL-CD180 will echo the received data to the transmitter for transmission back to the sender. The received data will not be passed on to the host.

When in Remote Loopback Mode, the transmitter continues to run as defined by its own Baud Rate Registers, not the values being used by the receiver. The CL-CD180 receives a complete character, strips off Start, Stop, and Parity Bits, and then re-transmits it with Parity, Length, and Stop Bit Output options as defined in COR1. Thus, it is possible to change baud rate. However, this can result in receiver overflow. In general, when programming for Remote Loopback Operation, the Transmit Bit Rate should be as fast or slightly faster than the expected receive rate to avoid possible overrun and loss of data. The number of Stop Bits should be set to a one, rather than one-and-a-half or two, if the application permits it. This ensures that the effective transmit rate is faster than the receive rate.

As shown in Figure 4–7, Remote Loopback is done at the character level and not the bit level. The Receive and Transmit FIFOs are not used in Remote Loopback. Characters are transferred directly from the Receive Holding Register to the Transmit Holding Register. For a diagnostic mode that tests the FIFOs, other logic is needed to be implemented by programming the host system to transfer received characters from the Receive FIFO to the Transmit FIFO. This will permit full testing of FIFO thresholds, service request logic, special character operation, etc.



514180-24

Figure 4–7. Local and Remote Loopback Logic

5. PROGRAMMING

5.1 Types of Registers

The CL-CD180 contains three types of registers:

- Global Registers — registers not specific to a particular channel
- Indexed Indirect Registers — special registers that are mapped to unique functions
- Channel Registers — registers specific to each channel

Global Registers

Global Registers contain information common to all channels. They are used primarily for passing vectors and setting-up service request handling.

Indexed Indirect Registers

Indexed Indirect Registers are special registers that either point to the FIFOs or signal the end-of-service request processing. The Indexed Indirect Registers are used primarily to transfer data to and from the serial channel FIFOs. Such transfers can be done only during a service request. When service requests are being serviced by the host, a context-switching technique is used by the CL-CD180 to reduce the number of cycles needed by the host to transfer data to and from the CL-CD180. The CL-CD180 makes available to the host all the registers pertaining to the channel requesting service by mapping them through to the Indexed Indirect Register addresses. This removes the burden on the host of keeping different addresses according to which channel is being accessed.

FIFO information is channeled through either the Receive Data Register, the Receive Character Status Register, or the Transmit Data Register of the Indexed Indirect Register set. Use of the Indexed Indirect Registers is valid only during appropriate service requests; the Transmit Data Register can be accessed during Transmit Service Requests, but not during Receive or Modem Service Requests. The Channel Access Register's (CAR) content is left unchanged from the value last set by the user, but it is not used in a service request context. The CAR should not be modified during a service request. During a service request, only access the channel that has caused the service request to be issued (as defined by the Global Interrupting Channel Register).

Channel Registers

Channel Registers are used to store parameters specific to each channel, such as bit rates, special character processing, and modem options. When not actively involved in a service request, each channel can be accessed at any time, independently of the other channels. Channel Registers can be accessed by first writing the number of the channel to be accessed into the Channel Access Register. The channel number in the CAR is used by the CL-CD180 as part of the Channel Register Address.

Individual CL-CD180 Registers are addressed via a seven-bit address contained in Address Bus Bits A6-A0. Address Bit A6 set to a '1' selects the Global Registers, and when set to a '0' selects the Channel Registers. When the CL-CD180 is not in a service request context, the active channel is defined in the CAR. The contents of the CAR then become part of the Address Field (along with A0-A5) needed to access the Channel Register file.

Off-Limit Registers

The CL-CD180 communicates to the host via shared access to its on-chip RAM. Of the 128-byte locations in the CL-CD180 address range, only 41 locations are defined as registers available to the host. The rest are used by the CL-CD180 for internal variable storage. Users should not access these registers because doing so may cause the CL-CD180 to malfunction.

5.2 Access Duty Cycle

The host access to the CL-CD180 appears to be a simple static read or write cycle, but the actual access occurs by arbitrating for the local (on-chip) bus and 'stealing' one-bus cycle. This is completely hidden from the user in normal circumstances, and successive accesses to the CL-CD180 may be done 'back-to-back' with no delay. However, if the host were to repetitively read from (or write to) the CL-CD180 as fast as possible over many cycles, enough CL-CD180 internal bus cycles would be 'stolen' that the CL-CD180 processor might not be able to keep pace with its processing. This situation could only

occur if the host was continuously testing a bit while waiting for it to change state. If there is a requirement to do something similar, a delay should be inserted in the host code so that the net-duty cycle of accesses is less than ten percent. This limitation applies only when the CL-CD180 is sending and receiving data on one or more channels. When initializing or re-configuring a channel, these registers can be written to at a fast pace.

5.3 Accessing FIFOs Versus Other Registers

The FIFO storage array is under the control of the CL-CD180 at all times. This is necessary to ensure that the FIFO is available for the CL-CD180 processor to access whenever needed. During normal operation, the CL-CD180 processor sets the FIFO pointers to the value required to transfer data, regardless of the value placed in the Channel Access Register (CAR) by the user. Therefore, the user cannot access the FIFOs in this manner.

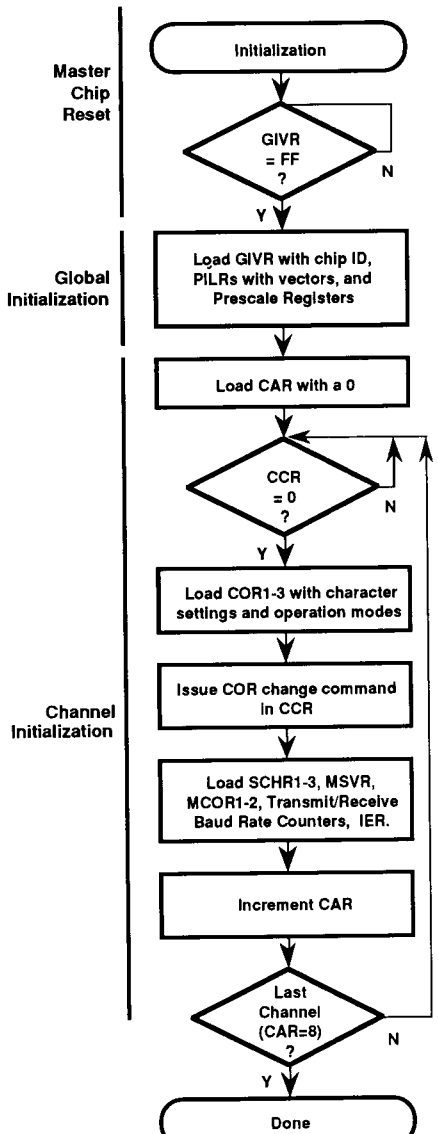
FIFOs may only be accessed in the context of an active Service Request. At this time only the CL-CD180 processor causes the FIFO pointers to be set to the appropriate value for the channel being serviced. FIFOs are then accessed via the Indirect Indexed Registers.

5.4 Initialization

The CL-CD180 initialization begins with a mandatory hardware reset applied through the active-low RESET* Input. The System Clock (CLK) Input must be active during the hardware reset, and the reset duration must be at least five clock periods. It is not necessary to synchronize RESET* Input with CLK. Refer to Figure 5–1.

Immediately following the hardware reset, the CL-CD180 goes through a firmware initialization, reaching an Idle Mode within 500 µs. This may be verified by the host by reading the Global Interrupt Vector Register and finding its contents to be FF Hex. Upon internal reset completion, the user may then configure the CL-CD180 for the desired channel functions.

A software reset may be performed by setting certain bits in the Channel Command Register (CCR). Setting Bits 7 and 0 to a '1' will reset all channels. This is done by forcing the CL-CD180 processor to jump to the same power-up sequence that it uses upon hardware reset. Whether the reset is caused by hardware or software, the CL-CD180 does not initialize every register and RAM location to a defined value. The only sure state is that all channels will be inactive, no service requests will be pending, and the Global Interrupt Vector Register will be FF Hex.



After either a hardware reset via the RESET Pin or a software reset via a CCR command, wait until the GIVR = xFF before proceeding with chip initialization.

When the CL-CD180 is ready, begin by loading the GIVR with the chip ID if there are more than one CL-CD180 in the system. Load the Service Match Registers with the vectors that will be used during service acknowledge cycles. Load the Prescale Registers with the value chosen for the basic time count for timer operations.

In preparation for channel initialization, load the CAR with a '0' to access Channel Zero Registers.

Before making any changes to the Channel Option Registers, wait for the CCR to contain a value of zero to ensure that the CL-CD180 is not processing a previous change command for that channel.

Load the Channel Option Registers with the values to enable the desired modes of operation and character parameters such as parity, stop bits, etc.

Inform the CL-CD180 that one or more Channel Option Registers have changed via the COR Change Command.

Load the Special Character Registers with the patterns that will be used during special character operations; the Modem Change Option Registers for modem interrupt conditions; the MSVR with the states of DTR/RTS as necessary and the baud rate constants for Transmit and Receive Baud Rate Generators. Set the appropriate bits in the SRER Register for the interrupt conditions desired.

If more channels, go back to the top of the loop.

514180-25

Figure 5–1. Initialization

5.5 Global Initialization

The user must initialize the CL-CD180 by programming the following Global Registers before starting normal operations on the ports — Prescaler Period Registers, the Global Interrupt Vector Register, and the three Service Match Registers (PILR).

5.6 Service Request Initialization

To prepare the CL-CD180 for service requests the following registers must be initialized:

- Global Interrupt Vector Register (GIVR)
- three Service Match Registers (PILR1, PILR2, PILR3)
- Global Interrupting Channel Registers (GICR)

The Global Interrupt Vector Register consists of five bits of user-supplied information, and three bits of CL-CD180–supplied service request group information. This concatenated vector supplied by the CL-CD180 during a service-request-acknowledgment cycle will direct the host to the proper service request subroutine. The host writes the five MSBs into the GIVR during initialization. These five bits may be either a chip ID number or a code that is appropriate for service request handling. In multiple-cascaded–CL-CD180 applications, these five bits must have a unique value for each CL-CD180 to identify which CL-CD180 is responding to a service request cycle.

Three registers in the Global Register set — Modem Service Match Register (PILR1), Transmit Service Match Register (PILR2), and Receive Service Match Register (PILR3) store the service request values for the three types of service requests. These levels are used to match with the value that appears on the address bus during a service-request-acknowledgment cycle. Since these levels are system dependent, the user must initialize these registers with the proper values.

The following three registers are used to provide the channel number of the channel requesting service — GICR1, GICR2, and GICR3. Reading any of these registers causes the CL-CD180 to 'mask-in' three bits specifying the channel number of the currently active channel. Normally these registers are read by the host when it is handling a service request. In this case, the three bits will be the number of the channel requesting service. If any of the three GICR Registers are read when the CL-CD180 is not in a service request context, the three bits will be the current value in the CAR.

Bits 4:2 are masked into the contents of these registers by the CL-CD180 when it is read by the host. The actual contents of the register are not modified.

These three registers are provided as a convenience to the user. In most applications the user will only use one of these locations and set the register to an arbitrary value. However, in some cases it may be useful to be able to record information about the state of the CL-CD180 (or the software driving it) that is associated with each of the three service request types. In this case, the user may store desired information in the unused bits. When entering a service routine, the software can check these bits (a 'sub-vector') to read recorded states.

5.7 Prescaler

The Prescaler Period Register (PPR) determines the fundamental 'tick' rate for all CL-CD180 on-chip timers, the Receiver Data Time-out and Transmitter Real-time Delay Timers. The PPR counts Clock (CLK) periods, and the minimum PPR value used must guarantee a 'tick' length of not less than 1.0 milliseconds. This requires a minimum value of 2666h for 9.830-MHz CLK, or 2710h for 10-MHz CLK. When operating at lower clock speeds than this, a value of at least 2666h must be used. As shown in the Internal Operation Flow Chart, Figure 2–3, processing timer events is in the outer (lowest priority) loop of the CL-CD180 firmware. A timer tick that is too short may result in two ticks occurring within one pass through the outer loop; this would result in missing one tick. This is not fatal, but it would result in inaccurate timings.

5.8 Channel Initialization and Changes

Prior to enabling the individual channels, program the Channel Registers with desired channel options and parameters such as character lengths, parity type, Receive FIFO thresholds, modem signal detection levels, bit rates, etc. When ready to begin, enable service requests.

Channel initialization is accomplished by first writing to the CAR Register with the number of the channel to be programmed. This channel number will automatically become part of the address for subsequent channel register programming. The host can use the same set of register addresses for all channels, thus eliminating the need to calculate addresses.

Certain channel options are controlled by the three Channel Option Registers. All changes to the Channel Option Registers must be accompanied by setting the appropriate Channel Option Register 'changed' Bits in the Channel Command Register (CCR). The CL-CD180 processor regularly samples the CCR for any value that is not a '0'. If the CCR is not a '0', the CL-CD180 decodes the command or commands, acts on them, and clears the CCR to signify completion of the commands. New commands must not be issued until any existing commands have been completed.

5.9 Transmitting Data

When transmitting data, a service request is received when the Transmit FIFO is empty. The number of the channel requesting service (i.e., the one with the empty FIFO) is available from the GICR. If there is more data to be sent, transfer up to eight bytes to the FIFO. If no data is available, disable the channel. The easiest way to accomplish this is by clearing the appropriate bit in the Interrupt Enable Register (IER). When new data is available, re-enable the channel via the IER, and a new service request for transmit data is received. At that time, transfer the data to the FIFO. Channels can be enabled or disabled by giving enable and disable commands via the Channel Command Register (CCR), but it is a slower process.

In some cases, it is necessary to know when a channel has sent the last bit of the last character rather than an empty FIFO. One example would be when changing bit rates. Two bits in the Interrupt Enable Register (IER), TxMpty and TxRdy, control the exact conditions for generating a service request. TxRdy indicates when the FIFO is empty, and TxMpty indicates when the last bit has been sent. It is acceptable to have both bits set but proper operation will be achieved by switching from the FIFO empty status to the transmitter empty status when it is necessary to know that all data has been completely sent. If they are set, the FIFO Empty Service Request will always occur first. If there is no more data to be sent, the Transmitter Empty Service Request will be received later, but in the mean time, FIFO empty requests may also be received. Once the last bit of the last character has been sent, a channel can be re-configured.

5.10 Receiving Data

When receiving data, a service request will be sent (for Good Data) when either the number of received bytes meets the threshold level, the Receive Time-out expires, or there is Good Data followed by a Receive Exception Condition (the CL-CD180 must transfer all the Good Data before giving the Exception). In all cases, the service-request routine reads the channel number requesting service (from GICR) and the number of bytes available (which can be more, the same, or less than the number set as the threshold) from the Receive Data Count Register (RDCR), and proceeds to transfer that many bytes, if possible.

It is not necessary to transfer as many bytes as are available or any bytes at all. If the host's buffer is nearly or completely full, the host can accept only those bytes it has room for, disable Receive Service Requests, exit the Service Request Routine, process the buffer, enable Receive Service Requests, and wait for the next service request. If no bytes are transferred during a Receive Service Request, and Receive Service Requests are still enabled, the CL-CD180 will immediately re-request service because the internal conditions that caused the request to be issued are still true. The host may either disable service requests or suspend host service request processing; however, both of these options should be implemented carefully as suspending service requests may result in an overflow condition if the suspension lasts too long.

6. DETAILED REGISTER DESCRIPTIONS

6.1 Register Map Quick Reference

Name	Description	Access	Binary Address	Hex Address (8 bit)[1]	Hex Address (Intel®)[2]	Hex Address (Motorola®)[3]	Page
GLOBAL REGISTERS							
GFRCR	Global Firmware Revision Code Register	R/W	110 1011	$6B	$D6	$D7	77
SRCR	Service Request Configuration Register	R/W	110 0110	$66	$CC	$CD	77
PPRH	Prescaler Period Register High	R/W	111 0000	$70	$E0	$E1	80
PPRL	Prescaler Period Register Low	R/W	111 0001	$71	$E2	$E3	80
PILR1	Modem Service Match Register	R/W	110 0001	$61	$C2	$C3	80
PILR2	Transmit Service Match Register	R/W	110 0010	$62	$C4	$C5	80
PILR3	Receive Service Match Register	R/W	110 0011	$63	$C6	$C7	81
GIVR	Global Interrupt Vector Register	R/W	100 0000	$40	$80	$81	82
SRSR	Service Request Status Register	R	110 0101	$65	$CA	$CB	82
MRAR	Modem Request Acknowledge Register	R	111 0101	$75	$EA	$EB	83
TRAR	Transmit Request Acknowledge Register	R	111 0110	$76	$EC	$ED	83
RRAR	Receive Request Acknowledge Register	R	111 0111	$77	$EE	$EF	83
GICR1	Global Interrupting Channel Register 1	R/W	100 0001	$41	$82	$83	83
GICR2	Global Interrupting Channel Register 2	R/W	100 0010	$42	$84	$85	83
GICR3	Global Interrupting Channel Register 3	R/W	100 0011	$43	$86	$87	83
CAR	Channel Access Register	R/W	110 0100	$64	$C8	$C9	84
INDEXED INDIRECT REGISTERS							
RDCR	Receive Data Count Register	R	000 0111	$07	$0E	$0F	85
RDR	Receiver Data Register	R	111 1000	$78	$F0	$F1	86
RCSR	Receiver Character Status Register	R	111 1010	$7A	$F4	$F5	86
TDR	Transmit Data Register	W	111 1011	$7B	$F6	$F7	87
EOIR	End-of-Interrupt Register	W	111 1111	$7F	$FE	$FF	87
CHANNEL REGISTERS							
IER	Interrupt Enable Register	R/W	000 0010	$02	$04	$05	88
CCR	Channel Command Register	R/W	000 0001	$01	$02	$03	89
COR1	Channel Option Register 1	R/W	000 0011	$03	$06	$07	92
COR2	Channel Option Register 2	R/W	000 0100	$04	$08	$09	93
COR3	Channel Option Register 3	R/W	000 0101	$05	$0A	$0B	94
CCSR	Channel Control Status Register	R	000 0110	$06	$0C	$0D	95
RBR	Receiver Bit Register	R	011 0011	$33	$66	$67	96
RTPR	Receive Time-out Period Register	R/W	001 1000	$18	$30	$31	96
RBPRH	Receive Bit Rate Period Register High	R/W	011 0001	$31	$62	$63	96
RBPRL	Receive Bit Rate Period Register Low	R/W	011 0010	$32	$64	$65	96
TBPRH	Transmit Bit Rate Period Register High	R/W	011 1001	$39	$72	$73	97
TBPRL	Transmit Bit Rate Period Register Low	R/W	011 1010	$3A	$74	$75	97

6.1 Register Map Quick Reference *(cont.)*

Name	Description	Access	Binary Address	Hex Address (8 bit)[1]	Hex Address (Intel)[2]	Hex Address (Motorola)[3]	Page
CHANNEL REGISTERS *(cont.)*							
SCHR1	Special Character Register 1	R/W	000 1001	$09	$12	$13	97
SCHR2	Special Character Register 2	R/W	000 1010	$0A	$14	$15	97
SCHR3	Special Character Register 3	R/W	000 1011	$0B	$16	$17	98
SCHR4	Special Character Register 4	R/W	000 1100	$0C	$18	$19	98
MCR	Modem Change Register	R/W	001 0010	$12	$24	$25	99
MCOR1	Modem Change Option Register 1	R/W	001 0000	$10	$20	$21	100
MCOR2	Modem Change Option Register 2	R/W	001 0001	$11	$22	$23	101
MSVR	Modem Signal Value Register	R/W	010 1000	$28	$50	$51	101
MSVRTS	Modem Signal Value – Request To Send	W	010 1001	$29	$52	$53	102
MSVDTR	Modem Signal Value – Data Terminal Ready	W	010 1010	$2A	$54	$55	102

NOTES:

1) Hex Address for 8-bit processor.
2) Address for Intel-style processor, see below.
3) Address for Motorola-style processor, see below.

In the above register map, the binary addresses are shown relative to the CL-CD180 address lines. In 16- and 32-bit systems, it is a common practice to connect eight-bit peripherals to only one byte lane. In 16-bit systems, the CL-CD180 appears at every other address, i.e., A0 in the CL-CD180 is connected to A1 in the host. In 32-bit systems, the CL-CD180 appears at every fourth address, i.e., A0 in the CL-CD180 is connected to A2 in the host. In both of these cases, the addresses used by a programmer will be different than what is shown.

For instance, in a 16-bit Motorola 68000-based system (or other 'big-endian' processors), the CL-CD180 is placed on data lines D0-D7, which are at odd addresses in the Motorola manner of addressing. The A0 in the CL-CD180 is connected to A1 of the 68000. Thus, CL-CD180 address $40 becomes $81 to a programmer. It is 'left-shifted' one bit, and A0 must be '1' for low-byte (D0-D7) accesses.

In a 16-bit Intel system (or other 'little-endian' processors), the CL-CD180 is again placed on data lines D0-D7, but these are at even addresses. The A0 in CL-CD180 is connected to the A1 in the host, but the host's A0 must be a '0' to access data lines D0-D7.

Many 32-bit processors have internal logic to 'steer' the data to the correct pins regardless of address value. However, if the processor employed does not, a scheme similar to the one described for 16-bit machines can be used, except that the CL-CD180 addresses are shifted two bits instead of one.

Even though not all of the CL-CD180 registers are intended to be read/write, there is no hardware mechanism to prevent the user from writing to them. The registers should, in some cases, not be written to by the host. See the individual register descriptions for details.

6.2 Global Registers

Global Registers provide a function common to all channels. There are two groups of Global Registers: ones that control the configuration of the CL-CD180 and ones that control service requests/interrupts.

6.2.1 Miscellaneous Registers

Global Firmware Revision Code Register (GFRCR) ($6B) — Read/Write

Firmware Revision Code

This register is initialized by the firmware during the power-on reset initialization routine to contain the current firmware version code of the CL-CD180. A Revision 'B' CL-CD180 is set to $81, and Revision 'C' is set to $82.

This register is a RAM location and may be modified by the user. The CL-CD180 sets it to the defined value only when a hardware or software reset is performed, and its contents are otherwise ignored. This value can be modified to indicate the configuration status of the CL-CD180, or to indicate any other requirement.

6.2.2 Configuration Registers

Service Request Configuration Register (SRCR) ($66) — Read/Write

PkgTyp	RegAckEn	DaisyEn	GlobPri	UnFair	Reserved	AutoPri	PriSel

This register configures the CL-CD180 depending on the method chosen for handling service requests. In addition to the 'traditional' interrupt-based host interface, writing the appropriate bits in this register provides for software- rather than hardware-based service request acknowledgments, fixes service request priorities in either of two ways, and controls Fair Share Interrupt operation. This register preserves compatibility with existing CL-CD180 software. For this reason, this register defaults to all zeroes and must be enabled for each new feature as desired.

RegAckEn and DaisyEn Bits are related to each other, and perform service-request acknowledgments by accessing registers within the CL-CD180 instead of asserting hardware signals.

Service requests are prioritized by four other bits. AutoPri enables the priority scheme; PriSel, GlobPri, and UnFair determine the specific priority to be used.

Bit	Description
Bit 7	**PkgTyp:** This read-only bit indicates the CL-CD180 package type. This bit is a '0' for the 84-pin PLCC.
Bit 6	**RegAckEn:** Enables register-based service-request acknowledgments. If this bit is a '0', register-based acknowledgments are not accepted. In this case, the results of a read of any of the service-acknowledgment registers are undefined. This is the default state of RegAckEn, and it ensures compatibility with earlier versions of the CL-CD180. When RegAckEn is enabled, register-based acknowledges allow the user's software to acknowledge a service request by reading from a register rather than by driving the external IACKIN* Signal. This is convenient in applications where interrupts are not supported or where polling is preferred. Setting this bit does not disable the function of the IACKIN* Signal.
Bit 5	**DaisyEn:** Enables daisy-chaining of register-based service acknowledgments. When DaisyEn is a '1', a CL-CD180 being addressed with a register-based service acknowledgment (a read takes place from a register-acknowledgment address) for which it has a pending request, will place the contents of the Global Interrupt Vector Register modified by the service type on the data bus. When DaisyEn is a '1', a CL-CD180 being addressed with a register-based service acknowledgment, for which it does not have a pending service request, asserts IACKOUT* to pass the acknowledgment down the daisy chain. The next CL-CD180 in the chain will see the acknowledgment as an IACKIN* acknowledgment. The Service Request Acknowledge Register addresses must be placed in the corresponding Service Match Registers (PILR1, PILR2, and PILR3) as part of the user setup for daisy-chaining of register-based service acknowledgments. If daisy-chaining of register-based service acknowledgments is not used, the Service Match Registers may be programmed with any address codes that the user finds convenient for use with the 'normal' IACKIN* service-acknowledge mechanism. If DaisyEn is a '0' and a CL-CD180 is addressed with a register-based service acknowledgment for which it does not have a pending service request, it will respond by providing an interrupt vector with a modification code of '000'. The addressed CL-CD180 treats this as an interrupt acknowledge cycle, but with passing inhibited, it must 'take' the acknowledge with an ACK level of '00' (none of the interrupt types). RegAckEn must be a '1' to enable register-based service acknowledgments. DaisyEn has no effect on daisy-chain operation of the regular IACKIN*/IACKOUT* chain.

Bit	Description *(cont.)*
Bit 4	**GlobPri:** When AutoPri is used, if GlobPri is set to a '1', the CL-CD180 will prioritize across multiple CL-CD180s sharing IREQ lines. If GlobPri is set to a '0', the CL-CD180 will accept the acknowledge for the highest priority on-chip interrupt. In both cases, automatic prioritizing is only done on type 1 (normally the modem signal change type) interrupt acknowledgments through the IACKIN* mechanism or the register-based acknowledge mechanism. When using GlobPri and AutoPri, it is possible to use the CL-CD180 with the three IREQ lines wire-OR'ed together. In this configuration, with any interrupt request asserted, the global values of all requests will appear asserted. GlobPri should be a '0' to force prioritization among the interrupt sources on-chip. When no on-chip interrupts are pending, the acknowledgment will be subject to daisy-chaining. See DaisyEn description.
Bit 3	**UnFair:** Fairness Override Bit. If UnFair is a '0', normal Fair Share Interrupt control is performed. If UnFair is a '1', the fair bits are all forced to a '1', disabling the Fair Share mechanism. This is useful when the AutoPriority Option is used, and the different IREQ lines are wire-OR'ed together.
Bit 2	Reserved. Must be a '0'.
Bit 1	**AutoPri:** When set, indicates that the CL-CD180 should prioritize service requests in the manner selected by the PriSel Bit. In conjunction with the GlobPri Bit, either local (within the chip) or global (across daisy-chained chips) prioritization is done. With AutoPri set, auto-prioritization is performed only when a type 1 (modem) interrupt acknowledgment is recognized. Acknowledgments of type 2 (transmit) and 3 (receive) interrupts continue to be unique and specific even with AutoPri set. This offers a form of local override to Auto-prioritization for Transmit or Receive Service Request when continuing a second-priority service routine. If not set, the user must indicate the service request being acknowledged by the choice of service request acknowledge register. AutoPri x GlobPri => look at IREQin to prioritize globally. AutoPri x GlobPri* => look at IREQ to prioritize locally.
Bit 0	**PriSel:** Prioritized interrupt order option. If AutoPri is set, PriSel selects the highest-priority service request. If PriSel is a '0', receive requests have the highest priority. If PriSel is a '1', transmit requests have the highest priority. Modem signal change request priority is fixed at the lowest priority.

Prescaler Period Register – High (PPRH) ($70), Low (PPRL) ($71) — Read/Write

Binary Value

These two registers provide the initialization value for the Timer Prescaler that is clocked by the system clock. This establishes the clock for the various on-chip timers.

The value loaded into these registers must establish a clock period of at least 1.0 ms. For a clock speed of 10 MHz, the value must be 10,000 (decimal) or larger. The values in these registers will be programmed to be FF (Hex) automatically upon a hardware reset.

Modem Service Match Register (PILR1) ($61) — Read/Write

1	Binary Value

This register must contain the value for Modem Signal Change Service Requests that will be presented on the Address Bus A0-A6 by the host to indicate the type of service request being acknowledged when IACKIN* is asserted. This register, along with the other two Match Registers, is compared to the value on the Address Bus during acknowledgment cycles so that the CL-CD180 can determine the service request being acknowledged by the host.

Bit 7 must be programmed to a '1'. The CL-CD180 compares all eight bits internally, but there are only seven address lines. Bits 6:0 of the register are compared to A6:A0 of the Address Bus. Bit 7 of the register is compared with a logic '1'.

Within any one CL-CD180, the three Match Registers must have unique values. In multiple CL-CD180 designs where service acknowledgments are cascaded, all Match Registers of the same type (e.g., Modem) must have the same value.

In designs using register-based service acknowledgments (RRAR, TRAR, and MRAR), the addresses of these registers must be placed in the equivalent Match Register so that PILR1 contains $75.

Transmit Service Match Register (PILR2) ($62) — Read/Write

1	Binary Value

This register must contain the value for Transmit Data Service Requests that will be presented on the Address Bus A0-A6 by the host to indicate the type of service request being acknowledged when IACKIN* is asserted. This register, along with the other two Match Registers, is compared to the value on the Address Bus during acknowledgment cycles so that the CL-CD180 can determine the service request being acknowledged by the host.

Bit 7 must be programmed to a '1'. The CL-CD180 compares all eight bits internally, but there are only seven address lines. Bits 6:0 of the register are compared to A6:A0 of the Address Bus. Bit 7 of the register is compared with a logic '1'.

Within any one CL-CD180, the three Match Registers must have unique values. In multiple-CL-CD180 designs where service acknowledgments are cascaded, all Match Registers of the same type (e.g., Transmit) must have the same value.

In designs using register-based service acknowledgments (RRAR, TRAR, and MRAR), the addresses of these registers must be placed in the equivalent Match Register so that PILR2 contains $76.

Receive Service Match Register (PILR3) ($63) — Read/Write

1	Binary Value

This register must contain the value for Receive Data Service Requests that will be presented on the Address Bus A0-A6 by the host to indicate the type of service request being acknowledged when IACKIN* is asserted. This register, along with the other two Match Registers, is compared to the value on the Address Bus during acknowledgment cycles so that the CL-CD180 can determine the service request being acknowledged by the host.

Bit 7 must be programmed to a '1'. The CL-CD180 compares all eight bits internally, but there are only seven address lines. Bits 6:0 of the register are compared to A6:A0 of the Address Bus. Bit 7 of the register is compared with a logic '1'.

Within any one CL-CD180, the three Match Registers must have unique values. In multiple-CL-CD180 designs where service acknowledgments are cascaded, all Match Registers of the same type (e.g., Receive) must have the same value.

In designs using register-based service acknowledgments (RRAR, TRAR, and MRAR), the addresses of these registers must be placed in the equivalent Match Register so that PILR3 contains $77.

Global Interrupt Vector Register (GIVR) ($40) — Read/Write

Binary Value	IT2	IT1	IT0

Bit	Description
Bits 7:3	These bits are user-defined. However, in a multiple-chip design, these five bits must have a unique value in each CL-CD180 to identify which CL-CD180 is returning a vector during service acknowledgments. When writing to this register, write eight bits at once; the CL-CD180 will modify the low-three bits automatically. Note that if this register is read in a normal manner, the original eight bits will be read and the modified bits from the last acknowledgment cycle will not be preserved.
Bits 2:0	These three bits indicate the group/type of service request occurring. These bit are supplied by the CL-CD180 during an acknowledgment cycle.

IT2	IT1	IT0	Value	Group/Type
0	0	0	0	No Request Pending *
0	0	1	1	Modem Signal Change Service Request
0	1	0	2	Transmit Data Service Request
0	1	1	3	Receive Good Data Service Request
1	0	0	4	Reserved
1	0	1	5	Reserved
1	1	0	6	Reserved
1	1	1	7	Receive Exception Service Request

* This code is returned by the CL-CD180 only when RegAckEn is set, and DaisyEn is not set. In this condition, the CL-CD180 must provide a vector when acknowledged. If the CL-CD180 receives an acknowledgment for which it does not have a request pending, it will return '000'.

6.2.3 Service Request/Interrupt Control Registers

Service Request Status Register (SRSR) ($65) — Read Only

ilvl [1]	ilvl [0]	IREQ3ext	IREQ3int	IREQ2ext	IREQ2int	IREQ1ext	IREQ1int

The i-level Bits, ilvl[1] and ilvl[0], are the current context code from the service request context stack. They are encoded as follows:

ilvl[1:0]	Context
00	Not in a service request context
11	CL-CD180 is in a Receive Service Request context
10	CL-CD180 is in a Transmit Service Request context
01	CL-CD180 is in a Modem Service Request context

An accepted interrupt acknowledge cycle pushes a new context onto the stack.

NOTE: The IREQ Status Bits are positive true, and the IREQ* Pins are negative true. The '...int' (internal) values are local to the chip being read, and the '...ext' (external) values are present externally on the pin, i.e., the result of the wire-OR'ed function.

Modem Request Acknowledge Register (MRAR) ($75) — Read Only

Transmit Request Acknowledge Register (TRAR) ($76) — Read Only

Receive Request Acknowledge Register (RRAR) ($77) — Read Only

Modified Interrupt Vector provided on read

The Service Request Acknowledge Registers are read-only registers that return an appropriate interrupt vector when read. Reading one of these registers has the effect of a service acknowledgment cycle in the CL-CD180 (not necessarily the one addressed; it may be one further down the daisy chain). The vector supplied on the data bus during the cycle is described under the Global Service Vector Register description. RegAckEn must be set for these registers to operate properly.

Global Interrupting Channel Registers 1, 2, 3 (GICR) ($41-$43) — Read/Write

Binary Value	C2	C1	C0	Binary Value

There are three registers used to provide the channel number of the channel requesting service. Reading any of these registers will cause the CL-CD180 to 'mask-in' three bits, specifying the channel number of the currently active channel. Normally these registers are read by the host when it is handling a service request. In this case, the three bits will be the number of the channel requesting service. If any of the three GICR Registers are read when the CL-CD180 is not in a service request context, the three bits will be the current value in the CAR. Bits 4:2 are masked into the contents of this register by the CL-CD180 when it is read by the host. The actual contents of the register are not modified.

These three registers are provided as a convenience to the user. In most applications, the user will only use one of these locations, and will set the register to an arbitrary value. All types of service routines would use this register. However, in some cases it may be useful to be able to record information about the state of the CL-CD180 (or the software driving it) that is associated with each of the three service request types. In this case, the user may associate an individual register with each level of service request, and store whatever information is desired in the unused bits. When entering a service routine, the software can check these bits (a sub-vector) to read recorded states.

Bit	Description
Bits 7:5	User-defined. Set to a specific value by the user.
Bits 4:2	Defines the service requesting channel number. **C2 C1 C0 Channel Number** 0 0 0 Channel 0 0 0 1 Channel 1 0 1 0 Channel 2 0 1 1 Channel 3 1 0 0 Channel 4 1 0 1 Channel 5 1 1 0 Channel 6 1 1 1 Channel 7
Bits 1:0	User-defined. Set to a specific value by the user.

Channel Access Register (CAR) ($64) — Read/Write

Reserved	Reserved	Reserved	Reserved	A7 (0)	C2	C1	C0

This register contains the channel number used for channel-oriented host read or write operations when the host is not in a service request service routine. When the CL-CD180 and the host are in a service request routine, the CL-CD180 supplies the service-requesting channel number via the Global Interrupting Channel Register. The Channel Access Register contents are not used during service request. The host service request routine is restricted to accessing only the register set of the service-requesting channel and the Global Registers.

The Channel Access Register is used by the host when the host is setting up or modifying the configuration of the channel. It is also used to issue certain channel-specific commands such as sending a flow-control character.

Bit	Description
Bits 7:4	Reserved, must be a '0'.
Bit 3	Internally, to the CL-CD180, this is Address Bit 7. This bit completes the external to internal CL-CD180 register address mapping, but it is only to be used for test purposes. In normal operation, this bit should always be a '0'.
Bits 2:0	Channel number

C2	C1	C0	Channel Number
0	0	0	Channel 0
0	0	1	Channel 1
0	1	0	Channel 2
0	1	1	Channel 3
1	0	0	Channel 4
1	0	1	Channel 5
1	1	0	Channel 6
1	1	1	Channel 7

6.3 Indexed Indirect Registers

Certain registers are specially designed to facilitate service-request handling. These registers do not exist as distinct registers, and can be thought of as pointers. These registers provide functions that are valid only during service-request service routines, and they must not be accessed at other times.

Three of the registers are actually pointers to the Transmit and Receive FIFOs, i.e., when referenced they cause the appropriate FIFO to be accessed. These registers are: Receive Data Register, Receive Character Status Register, and Transmit Data Register.

The CL-CD180 maintains all channel-specific information. During data transfer between the host and the CL-CD180, the CL-CD180 uses a context-switching technique to switch the proper channel-specific information into the Global Registers for use by the host. This reduces the processing burden on the host by eliminating the need to calculate address offsets.

6.3.1 Receive Data Count Register (RDCR) ($07) — Read Only

0	0	0	0	CT3	CT2	CT1	CT0

Bit	Description
Bits 7:4	Reserved, must be a '0'.
Bits 3:0	Specifies the number of Good Data bytes for transfer from the Receive FIFO at the time of service request. This may be larger or smaller than the threshold level set by the user. This register reflects the actual amount of data available, which can be greater than the threshold level if service-request response is slow, or less than the threshold if some other event (such as an error condition) has caused the Receive Good Data Interrupt. This register need only be read when receiving Good Data; by default all exceptions are one character, and the value in this register during a Receive Exception is not defined or meaningful. The RDCR will contain a zero if the current service request is for the NNDT case.

CT3	CT2	CT1	CT0	Number of good bytes
0	0	0	0	will not occur
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1001	to	1111		will not occur

6.3.2 Receive Data Register (RDR) ($78) — Read Only

D7	D6	D5	D4	D3	D2	D1	D0

This register accesses the Receive Data FIFO for the channel. It is used by all channels to transfer Receive FIFO data to the host. Successive reads transfer bytes from the FIFO to the host. Reading this register increments an internal pointer to the Data and Status FIFOs. During service-request routines for Good Data, this is the only register that must be read. During service-request routines for Receive Exception, the Receive Status Register must be read first, then this register may be read. If both the RCSR and this register are to be read, the RCSR must be read first because reading this register causes the FIFOs to 'pop'.

Any attempt to write to this register will cause unpredictable results.

6.3.3 Receive Character Status Register (RCSR) ($7A) — Read Only

Time-out	SC Det2	SC Det1	SC Det0	Break	PE	FE	OE

This register accesses the status information for the current receive character.

Bit	Description
Bit 7	**Time-out:** Indicates that the Receive FIFO is empty, and no data has been received within the receive time-out period. There is no data character associated with this status and no other status bits are valid if the Time-out Bit is set. Must be armed by the NNDT bit in IER.
Bits 6:4	**Special Character Detect (SCD0-2):** **SCD2 SCD1 SCD0 Status** 0 0 0 None detected 0 0 1 Special Character 1 or Special Character 1 and 3 sequence matched (only if Special Character 1 and 3 sequence is enabled). 0 1 0 Special Character 2 or Special Character 2 and 4 sequence matched (only if Special Character 1 and 3 sequence is enabled). 0 1 1 Special Character 3 (only if Special Character 1 and 3 sequence is not enabled). 1 0 0 Special Character 4 (only if Special Character 2 and 4 sequence is not enabled). **NOTE:** No special-character match is performed if any type of error occurs. The second character of a two-character sequence cannot cause a receiver overrun.
Bit 3	**Break:** Indicates that a break has been detected.
Bit 2	**Parity Error:** Indicates that a parity error has been detected.
Bit 1	**Framing Error:** Indicates that a bad Stop Bit has been detected.

Bit	Description *(cont.)*
Bit 0	**Overrun Error:** Indicates that new data has arrived but the CL-CD180 FIFO and Holding Registers are full. The new data is lost and the overrun indication is flagged on the last character received before the overrun occurred.

Multiple errors in one byte are possible because the CL-CD180 evaluates the character bit-by-bit as it receives it. For example, a parity error will be detected and flagged before a framing error. If a character is received with every bit (including the stop bit) equal to a '0', it is marked as a line-break. If some bits are a '1', but the Stop Bit is 'missing' a '0', it is marked as a framing error. If odd parity is set and the bits received are all zeroes, it is marked as both a break character and a parity error. In addition to any other bits, the Overrun Bit will be set if an overrun has occurred.

Any attempt to write to this register will cause unpredictable results.

6.3.4 Transmit Data Register (TDR) ($7B) — Write Only

D7	D6	D5	D4	D3	D2	D1	D0

When servicing a Transmit Data Service Request, the Transmit Data Register accesses the Transmit FIFO of the service-requesting channel. Data is written to the Transmit Data Register by the host; the CL-CD180 automatic FIFO pointer mechanism will place the data into the service-requesting channel's Transmit Character FIFO. Up to eight bytes of data may be written into the TDR during Transmit Data Service Request.

Any attempt to read from this register will cause unpredictable results.

6.3.5 End of Interrupt Routine Register (EOIR) ($7F) — Write Only

Irrelevant Value

This is a dummy register, and must be written to by the host's service request routine to signal to the CL-CD180 that the current service-request service is concluded. This must be the last access to the CL-CD180 during a service-request routine. Writing to this register will generate an internal End-of-Service Signal, which 'pops' the CL-CD180's service-request-context stack, allowing the CL-CD180 to resume normal processing and also service other channels. Service-request contexts may be nested, as explained in Section 2.4; i.e., one can respond to and service a higher-priority event while in the middle of a lower-priority service request routine (nesting subroutine calls within other subroutines).

Any attempt to read from this register will cause unpredictable results.

6.4 Channel Registers

There are eight sets of Channel Registers, but only one set is available at any given time. This offers the software-simplifying advantage that a given register is at the same address regardless of the channel number. To access a given channel's registers, first point to them by writing the channel number to the Channel Access Register.

6.4.1 Interrupt Enable Register (IER) ($02) — Read/Write

DSR	CD	CTS	RxD	RxSC	TxRdy	TxMpty	NNDT

A '1' in each bit position enables service request generation for the associated cause.

Bit	Description
Bit 7	**Data-Set-Ready (DSR) Service Request:** When enabled, generates a Modem-Change Service Request on the selected level changes of the DSR Input.
Bit 6	**Carrier Detect (CD) Service Request:** When enabled, generates a Modem-Change Service Request on the selected level changes of the CD Input.
Bit 5	**Clear-To-Send (CTS) Service Request:** When enabled, generates a Modem-Change Service Request on the selected level changes of the CTS Input.
Bit 4	**Receive Data Service Request:** When enabled, the Receive Data Service Request is generated for receive data and Receive Exceptions.
Bit 3	**Receive Special Character (RxSC) Service Request:** When enabled, the Receive Data Exception Service Request is generated when a received character matches one of the four user-defined special characters. When disabled, Receive Exceptions will be generated for error conditions and time-outs only. If flow-control transparency is set, flow-control characters will be stripped, and no Receive Special Character Exceptions will occur.
Bit 2	**Transmit Ready (TxRdy) Service Request:** When enabled, the transmitter will generate a service request when the Transmit FIFO becomes empty. Set this bit when first beginning transmission on a channel, and before attempting to write data to the Transmit FIFO. Enabling the service request will cause an immediate Transmit Service Request, allowing it to write data into the Transmit FIFO in the usual manner. This bit may be set and cleared as needed to regulate the assertion of Transmit Data Service Requests on each channel. This technique is preferred over disabling the transmitter.
Bit 1	**Transmitter Empty (TxMpty) Service Request:** When enabled, a service request is generated when the Transmit FIFO, the Transmit Holding Register, and the Transmit Shift Register are all empty. This mode is provided to allow the host to determine when all bits have been sent and it is safe to alter a channel's configuration.
Bit 0	**No New Data Time-out (NNDT) Service Request:** When enabled, a Receive Exception Service Request is generated after the completion of data transfer from the CL-CD180 to the host. This feature assists in buffer management by providing a notice of a gap in the Receive Data Stream longer than the time-out period.

6.4.2 Channel Command Register (CCR) ($01) — Read/Write

RESET CHAN	COR CHNG	SEND SP CH	CHAN CTL	D3	D2	D1	D0

The CCR is a special register used to prompt the CL-CD180 processor to indicate if any channel parameters have changed. Bits are set in the CCR to indicate which of several commands to carry out. The CL-CD180 processor notes changes in these bits and makes the required adjustments to the hardware; this process can take from microseconds to milliseconds. Therefore, it is important that the host CPU waits until the CL-CD180 processor has finished the current command before issuing any more commands, or continuing with any operation that the command will affect. For example, after setting the Local Loopback Bit in COR2, the host must wait until the command is complete before resuming transmission. If the host does not wait, characters may not be properly looped back.

The CL-CD180 processor indicates completion by clearing the CCR.

Bit	Description
Bit 7	Reset Channel Command.
Bit 6	Channel Option Register Command.
Bit 5	Send Special Character(s) Command.
Bit 4	Channel Control Command.
Bits 3:0	Defined by the type of command being issued; see the following descriptions.

Reset Channel Command

RESET CHAN	0	0	0	0	0	0	TYPE

This is a software reset command. There are two types of reset — Channel Reset (type 0), which resets only the current channel, and Global Reset (type 1), which resets the entire part to its power-up condition. When the channel reset command is issued, the CL-CD180 disables the transmitter and the receiver and clears the Data and Status FIFOs of the channel. Channel parameters will not be affected by a Channel Reset.

Bit	Description
Bit 7	Reset Channel Command, must be a '1'.
Bits 6:1	Not used. Must be a '0'.
Bit 0	**Reset Type:** If the Reset Type Bit is a '0', a software reset of the channel is performed. The transmitter and receiver are disabled, and all FIFOs are cleared (flushed). If the Reset Type Bit is a '1', an on-chip firmware initialization of all channels is performed. All channel and global parameters are reset to their power-on reset condition.

Channel Option Register Change Command

0	COR CHG	0	0	COR3	COR2	COR1	N/U

Changes made to some Channel Option Register Bits must be signalled to the CL-CD180 by this command. Any combination of COR changes may be indicated by one command. All of the bits in COR3 take effect immediately, and all of the bits in COR2 (except LLM) take effect immediately. In other words, when changing COR3 or any of COR2 (except LLM), it is not necessary to issue a Channel Option Register Change Command. However, to preserve compatibility with older CL-CD180 designs, it is acceptable to set these bits.

Bit	Description
Bit 7	Must be a '0'.
Bit 6	Channel Option Register Change Command, must be a '1'.
Bits 5:4	Must be a '0'.
Bit 3	Channel Option Register 3 changed (no longer required).
Bit 2	Channel Option Register 2 changed (required only for Local Loopback Mode change).
Bit 1	Channel Option Register 1 changed.
Bit 0	Not used.

Send Special Character(s) Command

0	0	SEND SP CH	0	0	SSPC2	SSPC1	SSPC0

Bit	Description
Bits 7:6	Must be a '0'.
Bit 5	Send Special Character(s) Command, must be a '1'.
Bits 4:3	Must be a '0'.

Bit	Description *(cont.)*

Bits 2:0 — Special Character Select

SSPC2	SSPC1	SSPC0	Function
0	0	0	Do not use
0	0	1	Send Special Character 1, or characters 1 and 3 in sequence if COR3 [XonCH] defines a two-character sequence.
0	1	0	Send Special Character 2, or characters 2 and 4 in sequence if COR3 [XoffCH] defines a two-character sequence.
0	1	1	Send Special Character 3
1	0	0	Send Special Character 4
1	0	1	Do not use
1	1	0	Do not use
1	1	1	Do not use

Channel Control Command

0	0	0	CHAN CTL	XMTR EN	XMTR DIS	RCVR EN	RCVR DIS

Bit	Description
Bits 7:5	Must be a '0'.
Bit 4	Channel Control Command, must be a '1'.
Bit 3	Transmitter Enable
Bit 2	Transmitter Disable
Bit 1	Receiver Enable
Bit 0	Receiver Disable

When turning the receiver or transmitter on or off, it is faster to simply enable and disable service requests (IER) rather than using the Channel Control Command.

Channel Option Register 1 (COR1) ($03) — Read/Write

Parity	ParM1	ParM0	Ignore	Stop 1	Stop 0	CHL 1	CHL 0

Changes to this register must be signalled via the Channel Command Register.

Bit	Description
Bit 7	**Parity:** 1 = odd parity. 0 = even parity.
Bits 6:5	**Parity Mode 1 and 0:** Defines Parity Mode for both the transmitter and the receiver. **ParM1 ParM0 Parity** 0 0 No parity 0 1 Force parity (odd parity = force 1, even = force 0) 1 0 Normal parity 1 1 Not used
Bit 4	**Ignore:** Ignore parity 0 = Evaluate parity on received characters. 1 = Do not evaluate parity on received characters.
Bits 3:2	**Stop Bit Length:** Specifies the length of the Stop Bit. **Stop1 Stop0 Stop Bit** 0 0 1 Stop Bit 0 1 1 1/2 Stop Bits 1 0 2 Stop Bits 1 1 2 1/2 Stop Bits
Bits 1:0	**Character Length:** **CHL1 CHL0 Character Length** 0 0 5 bits 0 1 6 bits 1 0 7 bits 1 1 8 bits

Channel Option Register 2 (COR2) ($04) — Read/Write

IXM	TxIBE	ETC	LLM	RLM	RtsAO	CtsAE	DsrAE

Changes only to Bit 4 (LLM) of this register must be signalled via the Channel Command Register.

Bit	Description
Bit 7	**Implied Xon Mode (IXM):** This bit has meaning only when in the automatic Transmit In-Band Flow-control Mode. During Transmit In-Band Flow-control Mode, the CL-CD180 stops transmission upon detection of an Xoff character or character sequence. The IXM Bit determines whether the CL-CD180 should restart transmission based on receipt of an Xon character or any character. When IXM Bit is set, the CL-CD180 will restart transmission upon detection of any character. When IXM Bit is not set, the CL-CD180 will wait for the Xon character or character sequence to restart the transmission.
Bit 6	**Transmit In-Band (Xon/Xoff) Flow Control Automatic Enable (TxIBE):** The CL-CD180 in the Transmitting Mode is flow-controlled by the remote. Upon receipt of the Xoff character, the CL-CD180 terminates transmission after the current character in the Transmit Shift Register, and the character in the Transmit Holding Register is sent. The CL-CD180 will resume transmission upon receipt of the Xon character, or any character, depending on the state of the IXM Bit.
Bit 5	**Embedded Transmitter Command Enable (ETC):** If set, the embedded special transmitter command functions are enabled.
Bit 4	**Local Loopback Mode (LLM):** 1 = Enables the Local Loopback Mode. 0 = Disables the Local Loopback Mode.
Bit 3	**Remote Loopback Mode (RLM):** 1 = Enables the Remote Loopback Mode. 0 = Disables the Remote Loopback Mode.
Bit 2	**RTS Automatic Output Enable (RtsAO):** When set, if the channel is enabled, the CL-CD180 will automatically assert the RTS* Output when it has characters to send. If CtsAE is also set, it will wait for CTS* to respond prior to transmission.
Bit 1	**CTS Automatic Enable (CtsAE):** Enables the CTS* Input to be used as automatic transmitter enable or disable.
Bit 0	**DSR Automatic Enable (DsrAE):** Enables the DSR* Input as automatic receiver enable or disable.

Channel Option Register 3 (COR3) ($05) — Read/Write

Xon CH	Xoff CH	FCT	SCDE	RxTH3	RxTH2	RxTH1	RxTH0

Changes to this register do not have to be signalled via the CCR.

Bit	Description
Bit 7	**Xon Character Definition:** 0 = Xon Character is a single-character code, and it is defined by Special Character. 1 = Xon Character is a double-character sequence, and it is defined by Special Characters 1 and 3.
Bit 6	**Xoff Character Definition:** 0 = Xoff Character is a single-character code, and it is defined by Special Character 2. 1 = Xoff Character is a double-character sequence, and it is defined by Special Characters 2 and 4.
Bit 5	**Flow-Control Transparency (FCT) Mode:** 0 = Flow-control characters received will be given to the host by Receive Exception Service Requests. 1 = Flow-control characters received will not be given to the host by Receive Exception Service Requests.
Bit 4	**Special-Character Detection Enable:** 0 = Special-Character Status detection is disabled. 1 = Special-Character Status detection is enabled.
Bits 3:0	**RxFIFO Threshold:**

RxTh3	RxTh2	RxTh1	RxTH0	Status
0	0	0	0	Do not use
0	0	0	1	1 character
0	0	1	0	2 characters
0	0	1	1	3 characters
0	1	0	0	4 characters
0	1	0	1	5 characters
0	1	1	0	6 characters
0	1	1	1	7 characters
1	0	0	0	8 characters
1001	to	1111		Reserved, do not use.

6.4.3 Channel Control Status Register (CCSR) ($06) — Read Only

RxEN	RxFloff	RxFlon	N/U	TxEN	TxFloff	TxFlon	N/U

This Status Register stores the current state of the channel. It may be read by the host at any time. If the host determines that a flow-control state is inappropriate, it may be cleared by enabling or disabling the transmitter or receiver by CCR command.

Bit	Description
Bit 7	**RxEn Receiver Enable:** 0 = Receiver is disabled. 1 = Receiver is enabled.
Bit 6	**RxFloff Receive Flow-off:** 0 = Normal 1 = The CL-CD180 has requested the remote to stop transmission (Send Xoff Command has been given to the channel). This bit will be reset when the CL-CD180 has requested the remote to restart transmission, or when the receiver is enabled or disabled, or when the channel is reset.
Bit 5	**RxFlon Receive Flow-on:** 0 = Normal 1 = The CL-CD180 has requested the remote to restart character transmission (Send Xon Command has been given to the channel). This bit is reset when the next (non-flow control) character is received, or when the receiver is enabled or disabled, or when the channel is reset.
Bit 4	Not used.
Bit 3	**TxEn Transmitter Enable:** 0 = Transmitter is disabled. 1 = Transmitter is enabled.
Bit 2	**TxFloff Transmit Flow-off:** 0 = Normal 1 = The CL-CD180 has been requested by the remote to stop transmission. This bit is reset when the CL-CD180 receives a request to resume transmission, or when the transmitter is enabled or disabled, or when the channel is reset.
Bit 1	**TxFlon Transmit Flow-on:** 0 = Normal 1 = The CL-CD180 has been requested by the remote to resume transmission. This bit is reset once character transmission is resumed, or when the transmitter is enabled or disabled, or when the channel is reset.
Bit 0	Not used.

6.4.4 Receiver Bit Register (RBR) ($33) — Read Only

Reserved	RxD	Start Hunt	Reserved	Reserved	Reserved	Reserved	Reserved

This register monitors certain functions of the actual receive hardware. It should *never* be written to as this will cause the CL-CD180 to fail. Only two of the bits are defined herein; however, the other bit positions can change value, so these bits should be 'masked-out' before testing.

Bit 6 is the sampled state of the RxD Pin, as sampled at the last bit-rate clock edge. This is not the actual RxD Input, as RxD cannot be sampled in real time. If no data has been received for a period of time, this bit will still reflect the last sampled state of the line at the end of the last character. This is because the line is not sampled when the CL-CD180 is looking for the Start Bit of a new character.

Bit 5 indicates whether the CL-CD180 is looking for a Start Bit. If Bit 5 is a '1', it is looking. If Bit 5 is a '0', it is receiving a character.

6.4.5 Receive Time-out Period Register (RTPR) ($18) — Read/Write

Receiver Data Time-out Period

This register defines the time period for two functions related to the Receive FIFO. As each character is moved to the Receive FIFO, the Receive Timer is reloaded with the Receive Data Time-out Period. The Receive Timer is then decremental on each tick of the Prescaler Counter. If the Receive Timer reaches a '0', it causes a Receive Good Data Service Request.

There is another optional feature called No New Data Time-out. When enabled, the Receive Timer will generate a Receive Exception if the timer expires after the last data is transferred from the FIFO to the host. This is intended to tell the host that no more data is arriving, and to go ahead and process the buffer.

The Receive Time-out Period Register defines the time-out period for both of these functions. It counts in time increments defined by the prescaler.

6.4.6 Receive Bit Rate Period Register – High Byte (RBPRH) ($31) and Low Byte (RBPRL) ($32) — Read/Write

Receive Bit Rate Divisor Byte

These two registers contain the 16-bit pre-load value for the Receive Bit Rate Counter. This count establishes the basic Receiver Clock Rate, which must be 16 times the desired Receiver Bit Rate. These registers are reset to a '0' by RESET*. The period established for the 16 times Receiver Clock Rate is equal to the RBPR 16-bit binary value times the System Clock (CLK) Period.

6.4.7 Transmit Bit Rate Period Register – High Byte (TBPRH) ($39) and Low Byte (TBPRL) ($3A) — Read/Write

Transmit Bit Rate Divisor Byte

These two registers contain the 16-bit pre-load value for the Transmit Bit Rate Counter. This count establishes the Transmitter Clock Rate, which must be 16 times the desired Transmitter Bit Rate. The precise period established for the 16 times Transmitter Clock is equal to the RBPR 16-bit binary value times the System Clock (CLK) Period. These registers are reset to a '0' by RESET*.

6.4.8 Special Character Register 1 (SCHR1) ($09) — Read/Write

Special Character 1

This register stores the right-justified bit pattern for Special Character 1. Unused bits must be a '0'. During receive, this character is one of the four characters compared with the received data for special-character recognition. If a match occurs with one of these four characters, it is noted in the Receiver Status FIFO entry accompanying the received character unless a double-character compare is enabled. In this case, the Receive Status FIFO entry will not be made until both characters are compared and matched.

During transmit, this register contains the characters that are sent as a result of the Send Special Character 1 command. If two-character sequences are enabled, Characters 1 and 3 will be sent.

Special Character 1 defines the Xon character or the first-half of the Xon-character sequence. The second half is Special Character Register 3.

6.4.9 Special Character Register 2 (SCHR2) ($0A) — Read/Write

Special Character 2

This register stores the right-justified bit pattern for Special Character 2. Unused bits must be a '0'. During receive, this character is one of the four characters compared with the received data for special-character recognition. If a match occurs with one of these four characters, it is noted in the Receiver Status FIFO entry accompanying the received character unless a double-character compare is enabled. In this case, the Receive Status FIFO entry will not be made until both characters are compared.

During transmit, this register contains the characters that are sent as a result of the Send Special Character 2 command. If two-character sequences are enabled, Characters 2 and 4 will be sent.

Special Character 2 defines the Xoff character or the first-half of the Xoff-character sequence.

6.4.10 Special Character Register 3 (SCHR3) ($0B) — Read/Write

Special Character 3

This register stores the right-justified bit pattern for Special Character 3. Unused bits must be a '0'. During receive, this character is one of the four characters compared with the received data for special character recognition. If a match occurs with one of these four characters, it is noted in the Receiver Status FIFO entry accompanying the received character unless a double-character compare is enabled. In this case, the Receive Status FIFO entry will not be made until both characters are compared.

During transmit, this register contains the characters that are sent as a result of the Send Special Character 3 command.

Special Character 3 may be the second-half of the Xon-character sequence.

6.4.11 Special Character Register 4 (SCHR4) ($0C) — Read/Write

Special Character 4

This register stores the right-justified bit pattern for Special Character 4. Unused bits must be a '0'. During receive, this character is one of the four characters compared with the received data for special character recognition. If a match occurs with one of these four characters, it is noted in the Receiver Status FIFO entry accompanying the received character unless a double-character compare is enabled. In this case, the Receive Status FIFO entry will not be made until both characters are compared.

During transmit, this register contains the characters that are sent as a result of the Send Special Character 4 command.

Special Character 4 may be the second-half of the Xoff-character sequence.

6.4.12 Modem Change Register (MCR) ($12) — Read/Write

DSR chg	CDchg	CTSchg	0	0	0	0	0

The CL-CD180 sets bits in this register when it recognizes a level change on a modem pin, as programmed by the Modem Change Option Registers. Changes detected will be a cause for asserting the Modem Service Request if corresponding Service Request Enable Bits are set. Once the service request is asserted, updates to this register are inhibited until End-of-Interrupt Register (EOIR) is written at the end of the Modem Service Request Routine. The host must clear these register bits during the service routine.

Bit	Description
Bit 7	**DSR Changed:** A logic '1' denotes that the Data-Set-Ready Input has changed state.
Bit 6	**CD Changed:** A logic '1' denotes that the Carrier Detect Input has changed state.
Bit 5	**CTS Changed:** A logic '1' denotes that the Clear-to-Send Input has changed state.
Bits 4:0	Must be a '0'.

Modem Change Option Register 1 (MCOR1) ($10) — Read/Write

DSRzd	CDzd	CTSzd	0	DTRth3	DTRth2	DTRth1	DTRth0

This register is used to define the current state change options to be monitored.

Bit	Description
Bit 7	**DSRzd is a '1':** Detect high-to-low voltage transition on DSR* Input (zero-to-one transition of DSR (MSVR) Bit).
Bit 6	**CDzd is a '1':** Detect high-to-low voltage transition on CD* Input (zero-to-one transition of CD (MSVR) Bit).
Bit 5	**CTSzd is a '1':** Detect high-to-low voltage transition on CTS* Input (zero-to-one transition of CTS (MSVR) Bit).
Bit 4	Must be a '0'.
Bits 3:0	Defines the threshold level that causes negation of DTR* when this flow-control option is specified. Normally, this level should be equal to or higher than the service-request level threshold as set in COR3. If it is set lower than the service-request threshold, it will default to the service-request threshold level.

DTRth3	DTRth2	DTRth1	DTRth0	Function
0	0	0	0	Automatic DTR Mode disabled
0	0	0	1	1 character
0	0	1	0	2 character
0	0	1	1	3 character
0	1	0	0	4 character
0	1	0	1	5 character
0	1	1	0	6 character
0	1	1	1	7 character
1	0	0	0	8 character

Modem Change Option Register 2 (MCOR2) ($11) — Read/Write

DSRod	CDod	CTSod	0	0	0	0	0

This register is used to define the current state change options to be monitored.

Bit	Description
Bit 7	**DSRod is a '1':** Detect low-to-high transition on DSR* Input (one-to-zero transition DSR (MSVR) Bit).
Bit 6	**CDod is a '1':** Detect low-to-high transition on CD* Input (one-to-zero transition of CD (MSVR) Bit).
Bit 5	**CTSod is a '1':** Detect low-to-high transition on CTS* Input (one-to-zero transition of CTS (MSVR) Bit).
Bits 4:0	Must be a '0'.

6.4.13 Modem Signal Value Register (MSVR) ($28) — Read/Write

DSR	CD	CTS	N/U	N/U	N/U	DTR	RTS

This register is read to determine the current input levels on the Modem Input Pins. It is written to supply an output value for the RTS* and DTR* Pins. The register bits have the opposite polarities from the actual states on the individual pins. Writing a '1' causes the pin to go to nominal zero volts.

Bit	Description
Bit 7	**DSR:** Current state of Data-Set-Ready Input.
Bit 6	**CD:** Current state of Carrier Detect Input.
Bit 5	**CTS:** Current state of Clear-to-Send Input.
Bits 4:2	Not used.
Bit 1	**DTR:** Current state of Data-Terminal-Ready Output.
Bit 0	**RTS:** Current state of Request-to-Send Output.

6.4.14 Modem Signal Value Request-To-Send (MSVRTS) ($29) — Write Only

0	0	0	0	0	0	0	RTS

In the Modem Signal Value Register, a write to either RTS or DTR affects the state of the other one. This can be a problem when the CL-CD180 is using one of these signals for flow control and the other one needs to be used under host control. This register writes to RTS without affecting the state of any other bits. RTS is at Bit 0.

6.4.15 Modem Signal Value Data-Terminal-Ready (MSVDTR) ($2A) — Write Only

0	0	0	0	0	0	DTR	0

In the Modem Signal Value Register, a write to either RTS or DTR affects the state of the other one. This can be a problem when the CL-CD180 is using one of these signals for flow control and the other one needs to be used under host control. This register writes to DTR without affecting the state of any other bits. DTR is at Bit 1.

7. ELECTRICAL SPECIFICATIONS

7.1 Absolute Maximum Ratings

Operating Ambient Temperature ..... 0°C to 70°C

Storage Temperature ..... -65°C to 150°C

All voltages, with respect to ground ..... -0.5 volts to V_{cc} + 0.5 volts

Supply Voltage (V_{cc}) ..... +7.0 volts

Power Dissipation ..... 0.5 watt

NOTE: Stress above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

7.2 Recommended Operating Conditions

Supply Voltage (V_{cc}) ..... 5 volts ± 5%

Operating free-air ambient temperature ..... $0°C < T_A < 70°C$

System Clock ..... 12.5 MHz

7.3 DC Electrical Characteristics

(@ V_{CC} = 5 volts ± 5%, T_A = 0°C to 70°C)

NOTE: Before beginning any new design with this device, please contact Cirrus Logic, Inc., for the latest errata information. See the back cover of this document for sales office locations and phone numbers.

Symbol	Parameter	MIN	MAX	Units	Conditions
V_{IL}	Input Low Voltage	-0.5	0.8	V	
V_{IH}	Input High Voltage	2.0	V_{cc}	V	(See note below)
V_{OL}	Output Low Voltage		0.4	V	I_{OL} = 8 mA
V_{OH}	Output High Voltage	2.4	V_{cc}	V	I_{OH} = -8 mA
I_{IL}	Input Leakage Current	-10	10	µA	$0 < V_{in} < V_{cc}$
I_{LL}	Data Bus three-state leakage current	-10	10	µA	$0 < V_{out} < V_{cc}$
I_{OC}	Open Drain Output Leakage	-10	10	µA	$0 < V_{out} < V_{cc}$
I_{CC}	Power Supply Current		75	mA	CLK = 12.5 MHz

7.3 DC Electrical Characteristics *(cont.)*

Symbol	Parameter	MIN	MAX	Units	Conditions
C_{in}	Input Capacitance		10	pF	
C_{out}	Output Capacitance		10	pF	

NOTE: Clock and RESET* V_{IH} MIN = 2.7 volts.

7.4 Index of Timing Information

7.5 AC Electrical Characteristics

Internally, the CL-CD180 is a fully clocked design; however, the hardware interface to the CL-CD180 may be either un-clocked or clocked. An un-clocked interface is generally easier to implement, especially if the CL-CD180 and its host are operating at different clock speeds. A clocked interface may be faster in some applications.

7.5.1 Clocked Bus Interface

Data transfers to or from the device occur in two steps. The first step occurs during the clock-low time. If the read/write state machine detects that it is time to do a cycle, it acquires the internal bus. The second step, that of actually transferring the data, occurs during the clock-high time. The cycle is complete at the end of the clock-high time.

The read/write state machine determines that it is time to do a cycle when there is a falling edge on the clock and both CS* and DS* are low. There is a specified setup time which must be met to guarantee that the cycle will begin. If this setup is not met, the cycle will occur one clock later. If the cycle is recognized, arbitration for the internal bus is done during the clock-low time. Addresses (and data, if a write cycle) must meet another setup time specification to the rising edge of the clock for the actual data transfer to occur properly during the clock-high time. In addition, the addresses must remain valid throughout the clock-high time, as specified. If the cycle is a write cycle, data must remain valid as specified. If the cycle is a read cycle, data is guaranteed valid for a specified time after the rising edge of the clock.

Service Acknowledge Cycles are a special case of read cycles. The service acknowledge 'read' (which returns the Global Service Request Vector value to the host) is started when the read/write state machine detects both DS* and another internal signal derived from both IACKIN* and DS*. There are two possible worst-case paths to consider when determining whether DS* and IACKIN* meet the necessary setup times to guarantee recognition on a particular clock edge. The longest path is DS*; it must propagate through a gate, an 8-bit comparator, a state machine, and another gate before arriving at the read/write state machine. The setup time for this is given in Table 7–1.

The other critical path is IACKIN*; it must pass through a state machine and a gate before arriving at the read/write state machine. The setup time to guarantee recognition on a particular clock edge is given in Table 7–1. Intel-style pin names are shown in {curly brackets}. All times are in nanoseconds, unless otherwise specified.

Table 7–1. Clocked Timings

Number in Figures	Description	MIN[a]	MAX[a]	Notes
1	Setup, DS*{RD*} and CS* low to CLK low, for read or write cycle to start ('ordinary' reads and all writes)	15		b
2	Setup, DS* {RD*} low to CLK low, for Service Acknowledge Cycle to start (IACKIN* Cycles and read cycles from Acknowledge Registers)	30		c
3	Setup, IACKIN* low to CLK low for cycle to start	15		
4	Setup, Address Valid to CS* and DS* low	5		
5	Setup, Address Valid to DS* (service acknowledge cycles)	7		d
6	Setup, Write Data Valid to CLK high	0		
7	Setup, R/W* {RD*, WR*} stable to DS* and CS* low (read, write cycles)	0		b, e
8	(DS* and CS*), or (RD* and CS*), or (WR* and CS*), high	10		f, g
9	Hold time, CS* low after CLK high (read, write cycles)	40		h
10	Hold time, DS* {RD*} after valid data	0	Infinity	h
11	Hold time, Address Valid after CLK high	40		h
12	Hold time, Write Data Valid after CLK high	25		
13	Hold time, IACKIN* low after next CLK low	7		i
14	Clock Period (T_{CLK})	80	500	j
15	Clock Low Time	37.6	250	j
16	Clock High Time	42.4	250	j
	Clock Duty Cycle	47%	53%	
17	Clock Rise/Fall time		3	k
18	RESET pulse width (after power is good and clock is stable)	5 clock periods		
19	Data Bus out of Hi-Z after CLK low	0		l
20	Read Data Valid after CLK high		66	
21	IACKIN* to IACKOUT* propagation delay		30	
22	IACKOUT* high after IACKIN* high		30	
23	DS* {RD*} high to data bus three-state	0	25	
24	DTACK* assert after CLK high (DTACKDLY = 0)		35	
25	DTACK* assert after CLK low (DTACKDLY = 1)		30	
26	DTACK* negate after DS* {RD* or WR*} negation		20	

Table 7–1. Clocked Timings *(cont.)*

Number In Figures	Description	MIN[a]	MAX[a]	Notes
27	IACKOUT* assert after CS* and DS* active on register acknowledge cycle with no match		40	m
28	DTACK* active pull-up time			n
29	IACKOUT* high after end of cycle		30	

[a] Unless otherwise noted, all values are in nanoseconds (ns).

[b] The reference to DS* and CS* refers to whichever one goes active last; that is, both signals must meet the setup time requirement.

[c] Enabling the Register Acknowledge ('regack') feature changes the timing somewhat, even on cycles where 'regack' is not being used.

[d] Calculated value; guaranteed by design, but not tested.

[e] For Motorola-style interface, refers to R/W*.
For Intel-style interface, refers to RD* or WR* (whichever is inactive for that cycle).

[f] A cycle must positively end before another begins; that is, control signals shall return to states such that no cycle is pending or active.

[g] Guaranteed by design, but not tested.

[h] During Register Based Acknowledge cycles, these signals must be held in the correct state until valid data is presented by the device, as indicated by DTACK* going active. Note that in daisy-chain applications, the response from the chain may be quite long due to the IACKIN*-IACKOUT* propagation delay required for the actual interrupting device to receive the select (IACKIN*). Waiting for the active DTACK* from the chain will eliminate any timing problems relating to these parameters.

[i] IACKIN* must be low for at least one clock period plus setup and hold times if there is only one CL-CD180 in the daisy chain. If there is more than one CL-CD180 in a daisy chain, IACKIN* must be low until it has rippled all the way down the chain.

[j] When using the clock out (CKOUT) of one CL-CD180 to drive subsequent CL-CD180s (such as in daisy-chain environments), CKOUT is skewed (delayed) by 3 ns from the internal clock. Therefore, on subsequent CL-CD180s, setup times are improved by 3 ns and hold times are derated by 3 ns.

[k] For clock periods greater than 100 ns (10 MHz or less clock), rise and fall time may be 5 ns maximum.

[l] Greater than a '0' by design, but not tested.

[m] This is the time for IACKOUT* to assert on register acknowledge cycles. IACKOUT* asserts if the device determines the acknowledgment is not intended for that part. If IACKOUT* asserts, the device does not drive the data bus or assert DTACK*. These functions are left to a device further down the daisy chain that accepts the acknowledge cycle.

[n] DTACK* sources current (drives 'high') until the voltage on the DTACK* line is approximately 1.5 volts. Then DTACK* goes to an 'open-drain' (high-impedance) state.

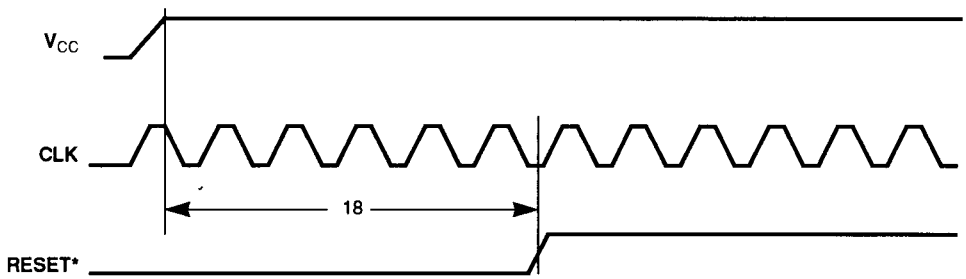


Figure 7–1. Clocked Bus Interface Reset

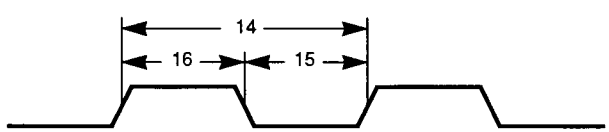


Figure 7–2. Clocked Bus Interface Clocks

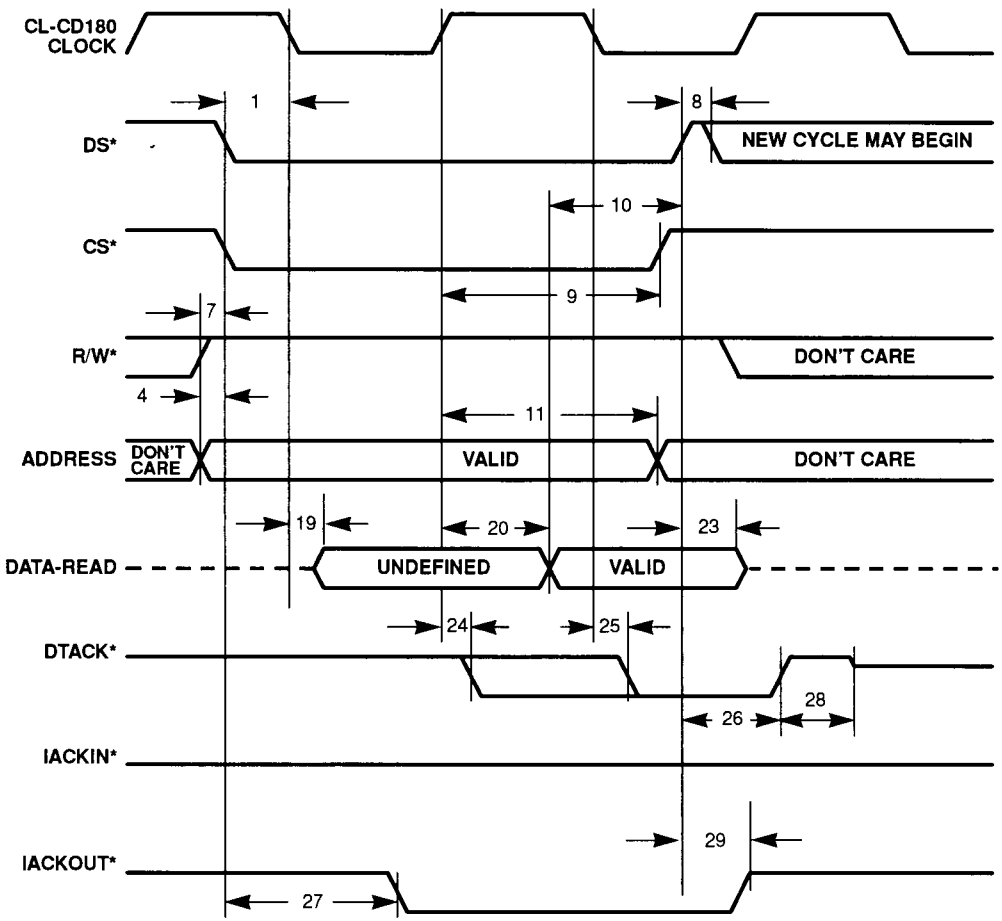


Figure 7–3. Clocked Bus Interface Read Cycle, Motorola®-Style Handshake

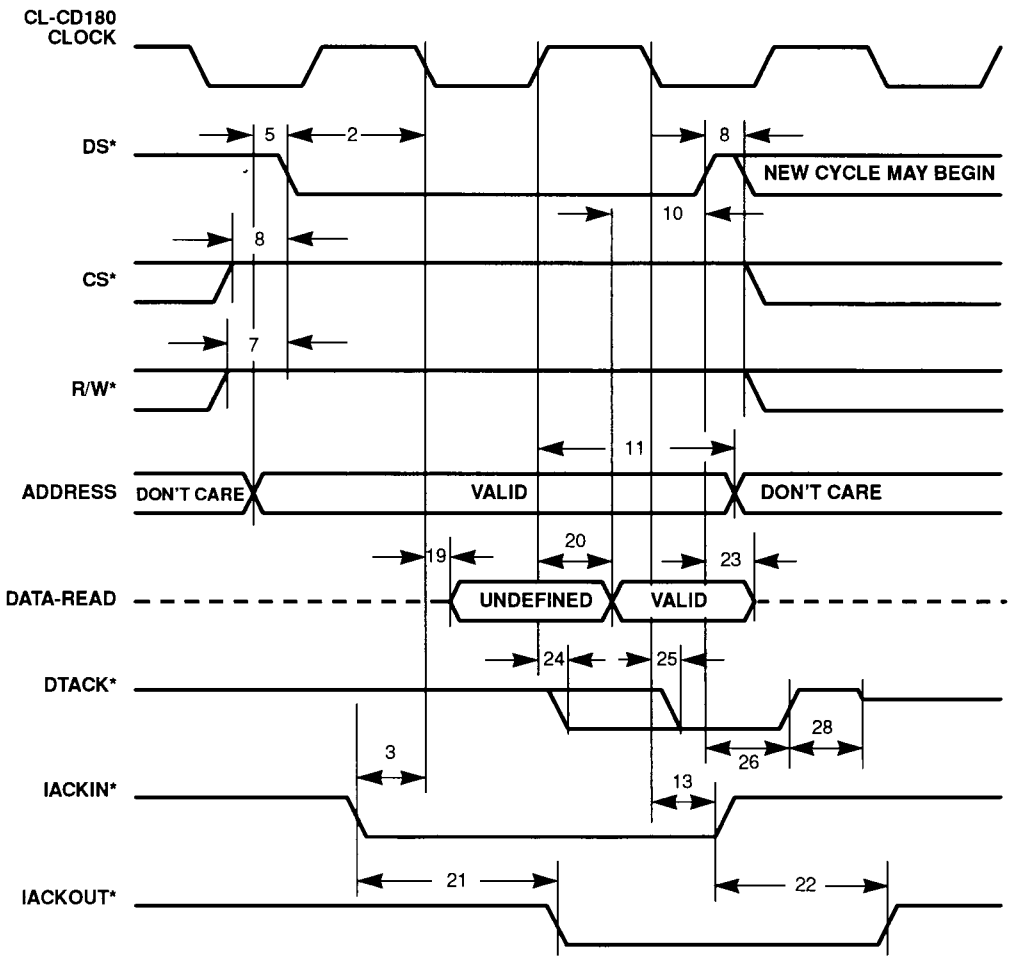


Figure 7–4. Clocked Bus Interface Service Acknowledgment Cycle, Motorola®-Style Handshake

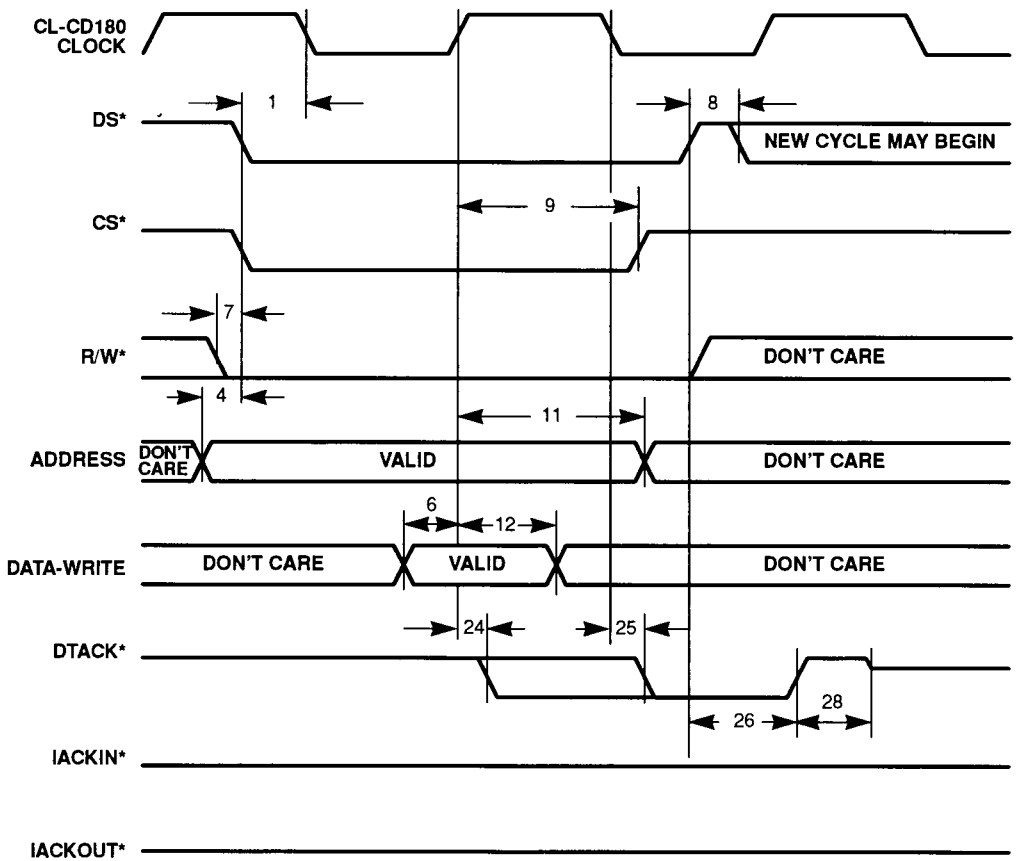


Figure 7–5. Clocked Bus Interface Write Cycle, Motorola®-Style Handshake

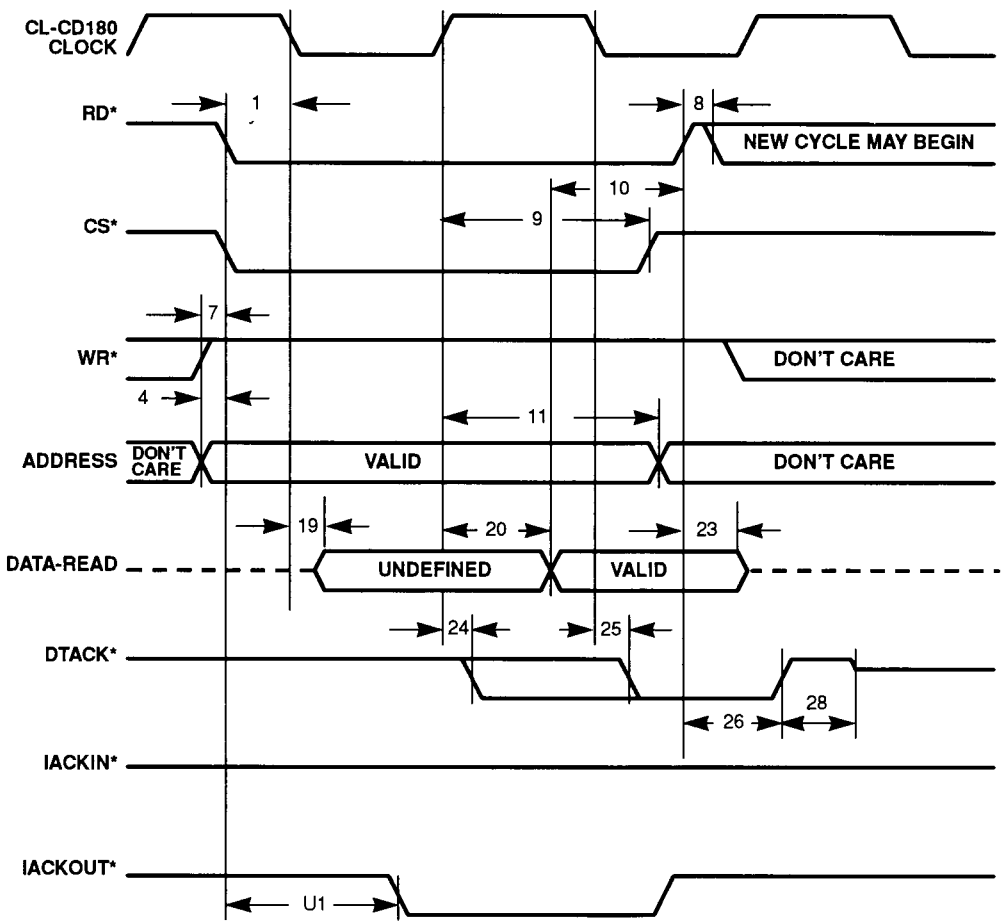


Figure 7–6. Clocked Bus Interface Read Cycle, Intel®-Style Handshake

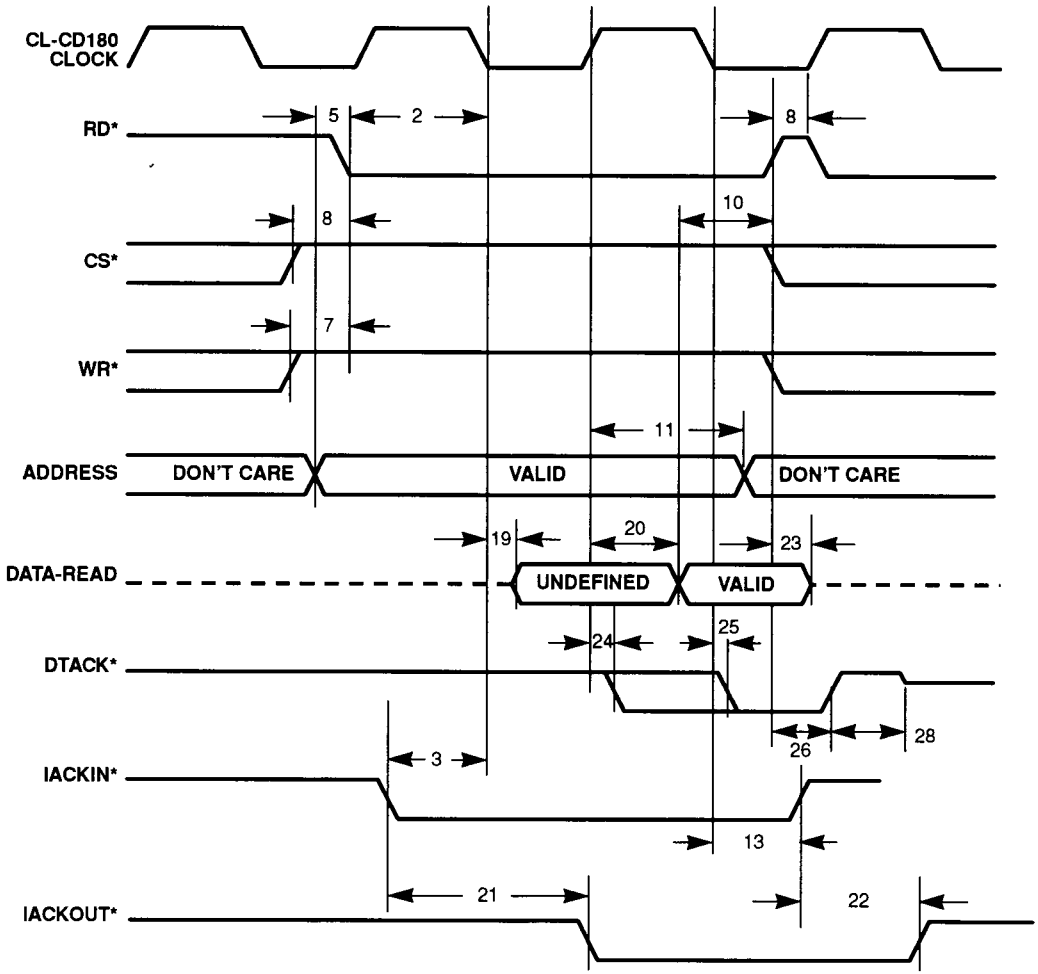


Figure 7–7. Clocked Bus Interface Service Acknowledgment Cycle, Intel®-Style Handshake

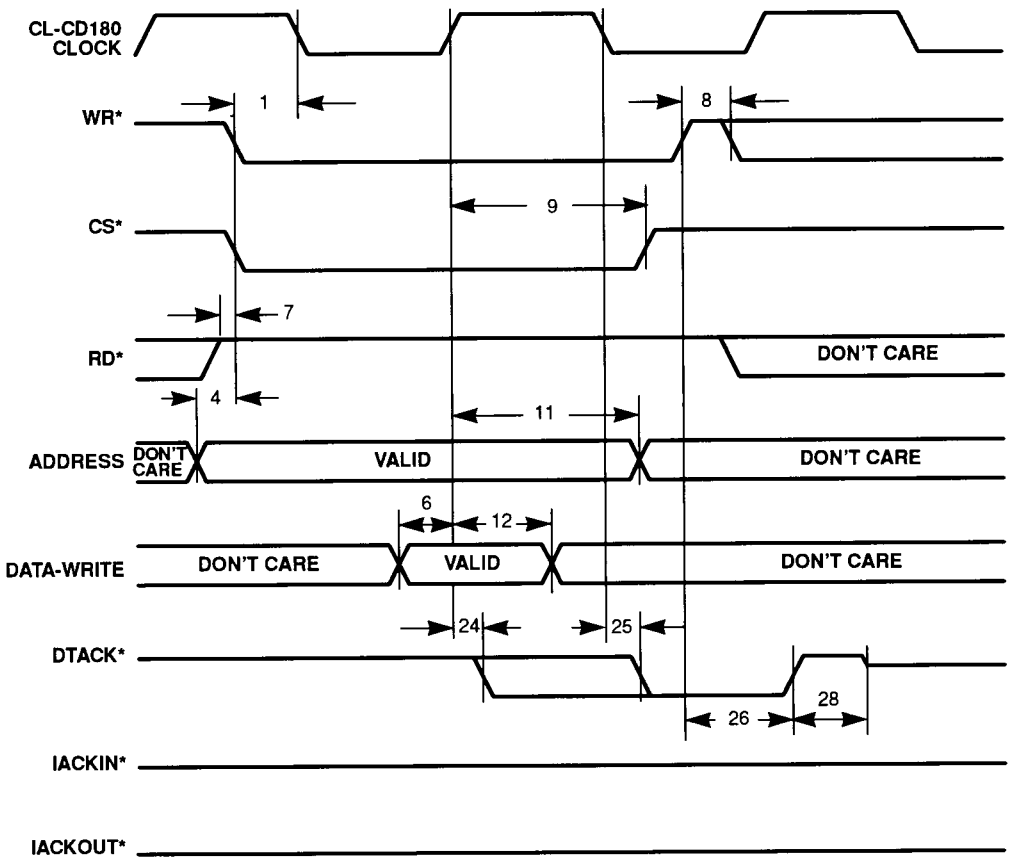


Figure 7–8. Clocked Bus Interface Write Cycle, Intel®-Style Handshake

7.5.2 Un-Clocked Bus Interface

Un-clocked timing diagrams represent worst-case synchronization delays. That is, they reflect the maximum number of clock cycles required to complete the operation.

Internally, the CL-CD180 fully synchronizes all signals; thus, the user need not be concerned with setup times or metastability. The vast majority of CL-CD180 designs employ an un-clocked Bus Interface.

All times are based on a master clock (CLK) of 15 MHz. All times are measured in nanoseconds. Intel-style handshake signals (where appropriate) are shown in {curly brackets}.

Table 7–2. Un-Clocked Timings

Number	Description	MIN[a]	MAX[a]	Notes
1	Setup time, address to CS*, DS* {CS*, RD* or WR*}		5	b
2	Setup time, R/W* to CS* or DS*	0		b
3	Hold time, address after CS* or DS* {CS* or RD* or WR*}	0		c, d
4	R/W* hold time after CS* and DS*	5		c, d
5	Delay time, DTACK* assert to valid read data: If DTACKDLY = 0 If DTACKDLY = 1		 31 -30	
6	DTACK* assert after CS* or DS* {RD*} or IACKIN* If DTACKDLY = 0 If DTACKDLY = 1		 150 175	b, e
7	Hold time, read data after CS* and DS*{RD*} high	1	30	c, f, g
8	CS* or DS* {RD*} high from DTACK* low If DTACKDLY = 0 If DTACKDLY = 1	 31 3		d, h, i, g
9	DTACK* inactive from (CS* or IACKIN*) or DS* high		30	c, j, d
10	DS* {RD*} high pulse width	10		d
11	Setup time, Address to IACKIN*	23		k, l
12	Setup time, write data to DS* {or WR*} low		-10	
13	Hold time, write data after DS* {or WR*} high	0		
14	x_REQ* deassert after DTACK* asserted		2 Tclk + 50	m
15	Setup time, R/W* {WR*} and CS* to IACKIN* low	0		n
16	x_REQ* reassert delay after write to EOSRR		2 Tclk + 50	o, p
17	IACKIN* assert/deassert to IACKOUT* assert/deassert prop delay		30	
18	Data bus out of high-impedance after DS* {RD*} low	5		q
19	Setup time, address to DS* {RD*} during acknowledge cycles	7		
20	IACKOUT* assert after CS* and DS* {RD*} active on register acknowledge cycles with no match		40	r
21	DTACK* active pull-up time			s

[a] Unless otherwise noted, all values are in nanoseconds (ns).

[b] During read cycles, CS* and DS* {RD*} are gated together internally. This specification is with respect to whichever goes active (low) last.

[c] During read cycles, CS* and DS* {RD*} are gated together internally. This specification is with respect to whichever goes inactive (high) last.

[d] This specification is with respect to whichever goes inactive (high) last.

[e] The values given is for 15-MHz operation. The time depends on system clock rate and the chosen DTACKDLY option. The actual time in any case can be determined by the formula:
If DTACKDLY = 0, then the time is 1.5(Tclk) + 43 ns
If DTACKDLY = 1, then the time is 2.0(Tclk) + 48 ns

[f] This specification is with respect to whichever of IACKIN* and DS* {RD*} goes active (low) last.

[g] The data bus is three-stated immediately after removal of DS* {RD*}. The device is guaranteed to be off the bus by the specified maximum time. The time can be as short as the minimum time. The hardware design should assure that the data has been read before DS* {RD*} is removed.

[h] In multiple–CL-CD180 designs, the Interrupt Acknowledge cycle must be long enough to accommodate the IACKIN* to IACKOUT* daisy-chain propagation delay from the first to the last CL-CD180. IACKIN* must remain low until after DTACK* asserts.

[i] For Acknowledge cycles, this specification refers to IACKIN* instead of CS*.

[j] During Interrupt Acknowledge cycles, IACKIN* is asserted instead of CS*; CS* should remain high. Note that IACKIN* timing is not always the same as CS*.

[k] During acknowledge cycles, addresses must propagate through the Service Match Registers. If a service request is pending on this CL-CD180, the match must finish before IACKIN* asserts. This is ensured by the specifications.

[l] This specification is with respect to IACKIN* only.

[m] This specification refers to one of Receive, Transfer, or Modem Service Request Outputs (RREQ*, TREQ*, MREQ*).

[n] This specification is with respect to DS*. CS* and R/W* must be high before the assertion of DS* to avoid the possibility of the CL-CD180 misinterpreting the cycle as a read or write.

[o] This is the time required to reassert a service request if the internal conditions of the CL-CD180 are such that the request should be asserted.

[p] This specification refers to one of Receive, Transfer, or Modem Service Request Outputs (RREQ*, TREQ*, MREQ*).

[q] The data bus is guaranteed to become active after DS* {RD*} low and before data is valid.

[r] This is the time for IACKOUT* to assert on register acknowledge cycles. IACKOUT* asserts if the part determines the acknowledgment is not intended for that part. If IACKOUT* asserts, the part does not drive the data bus or assert DTACK*. These functions are left to a device further down the daisy chain that accepts the acknowledge cycle.

[s] DTACK* sources current (drives 'high') until the voltage on the DTACK* line reaches 1.5V. At that time, DTACK* switches to an 'open-drain' (high-impedance) state.

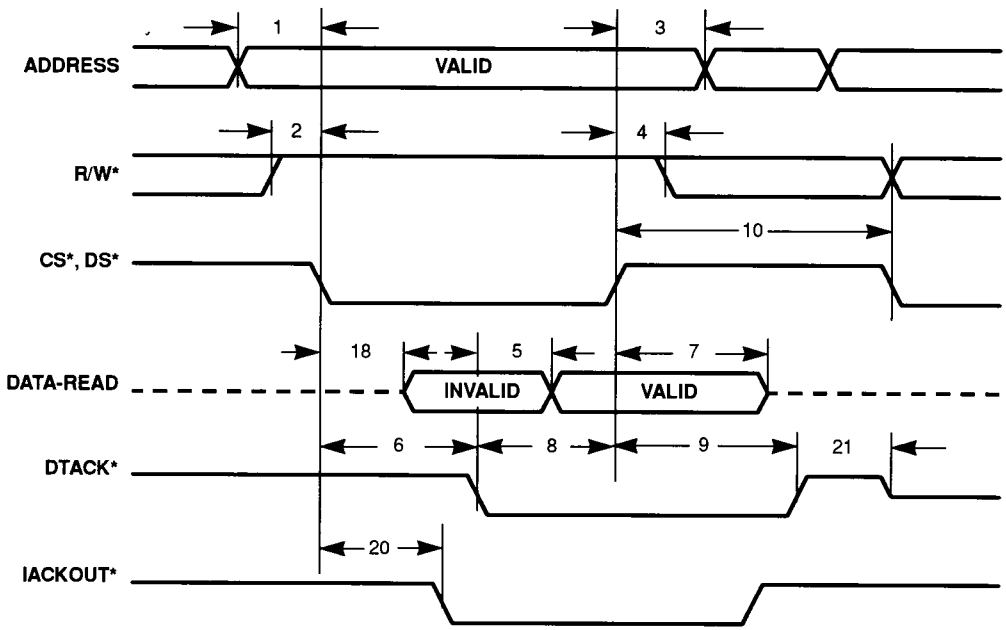


Figure 7–9. Un-Clocked Bus Interface Read Cycle, Motorola®-Style Handshake

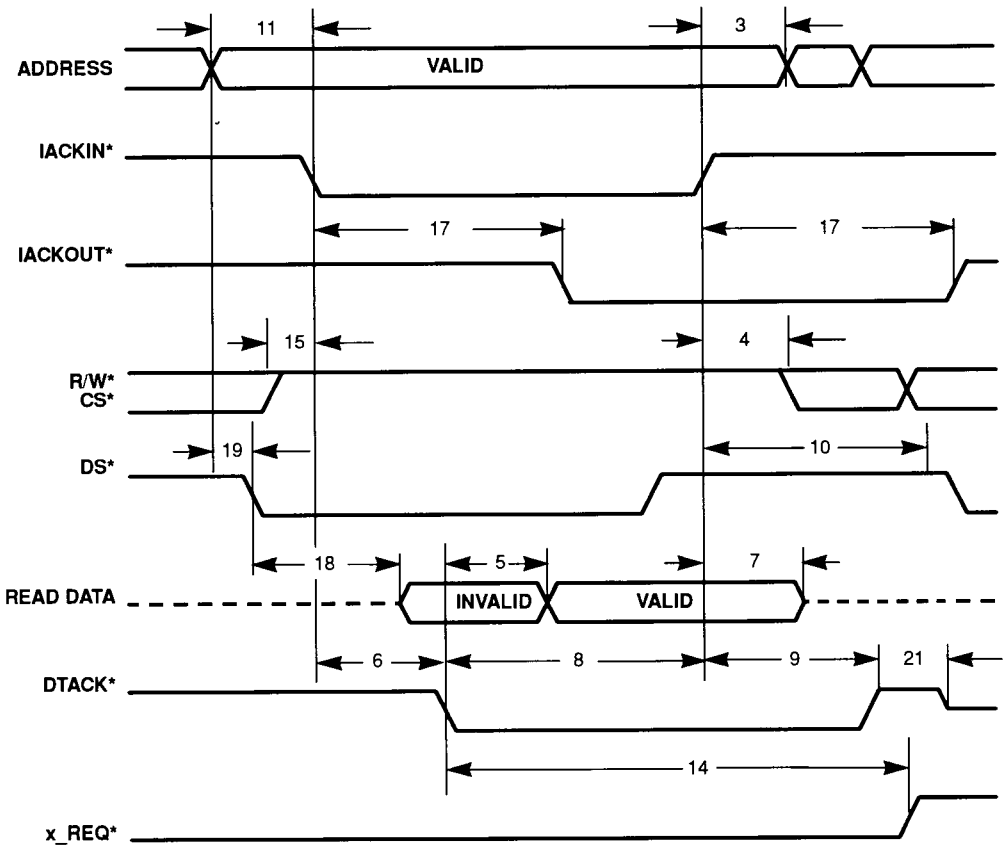


Figure 7–10. Un-Clocked Bus Interface Service Acknowledgment Cycle, Motorola®-Style Handshake

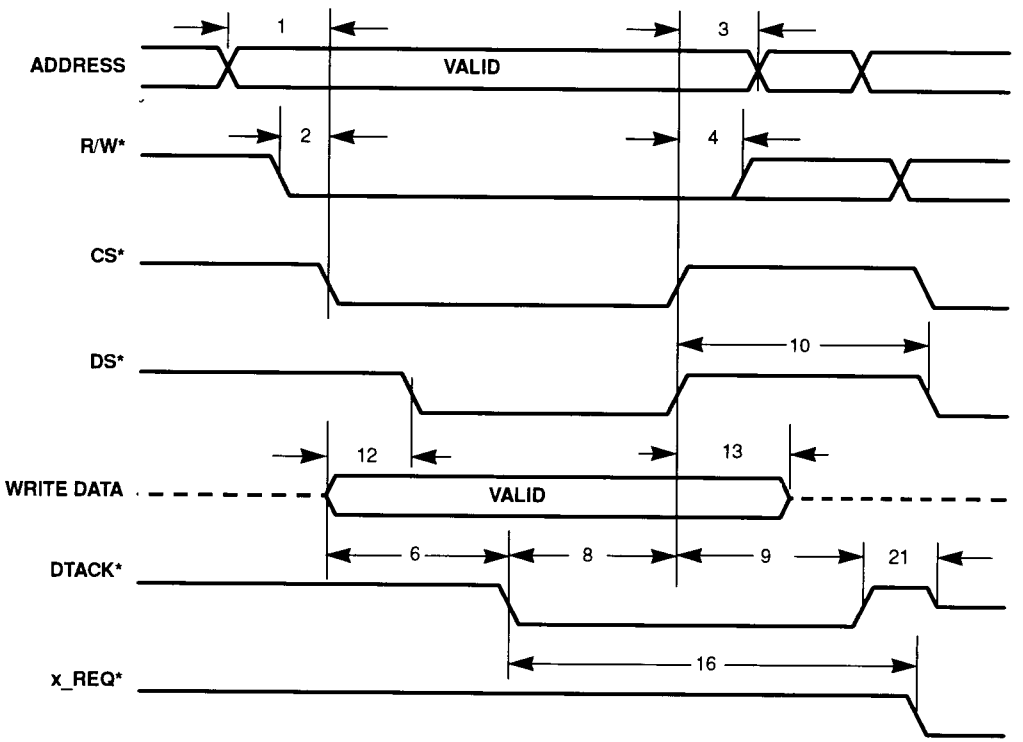


Figure 7–11. Un-Clocked Bus Interface Write Cycle, Motorola®-Style Handshake

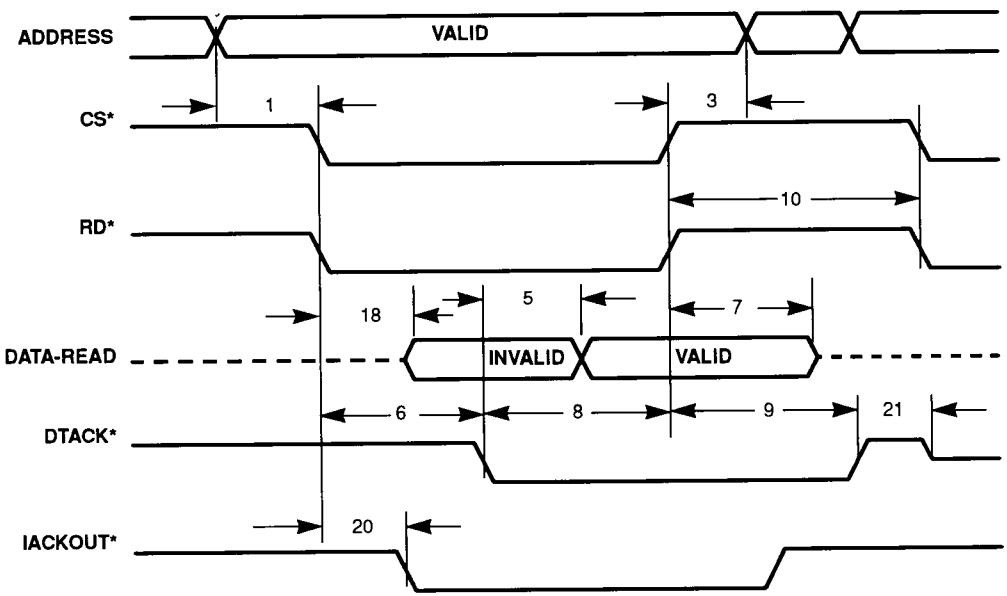


Figure 7–12. Un-Clocked Bus Interface Read Cycle, Intel®-Style Handshake

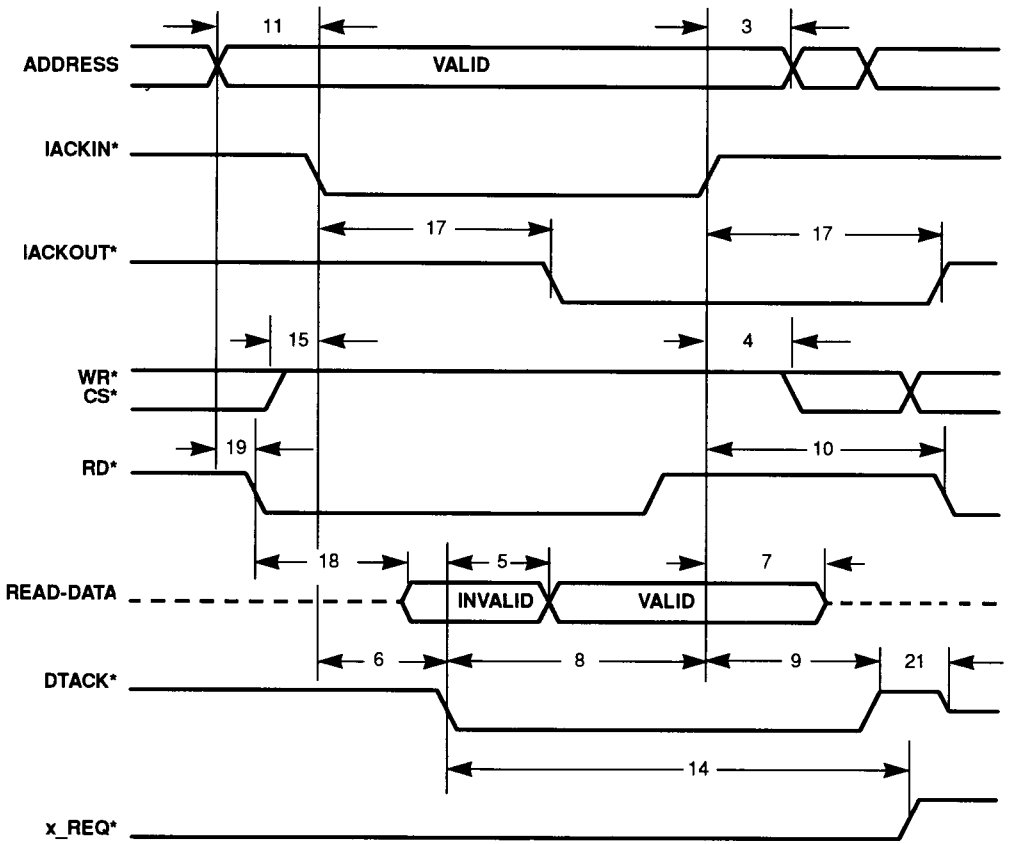


Figure 7–13. Un-Clocked Bus Interface Service Acknowledgment Cycle, Intel®-Style Handshake

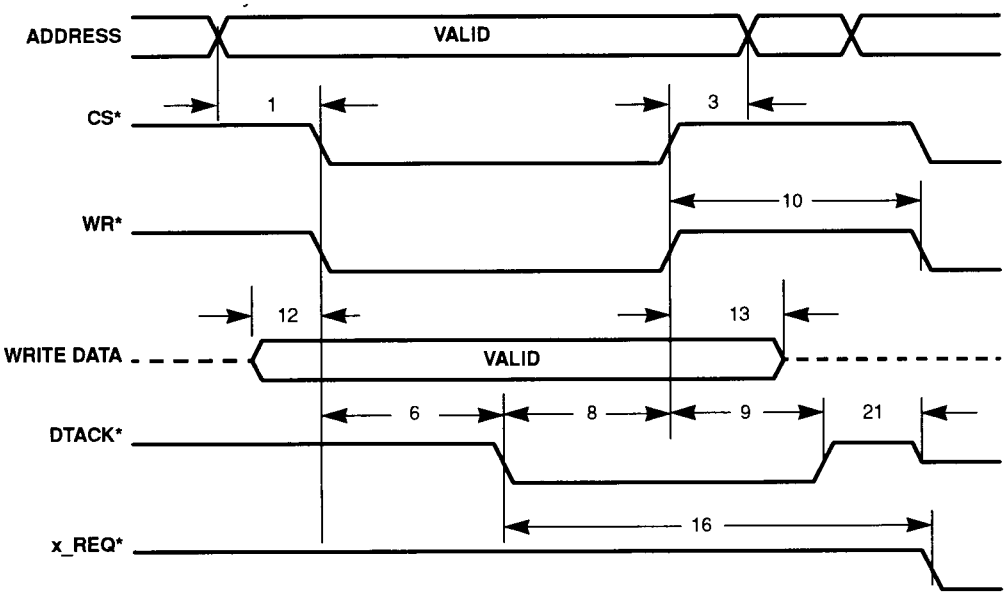
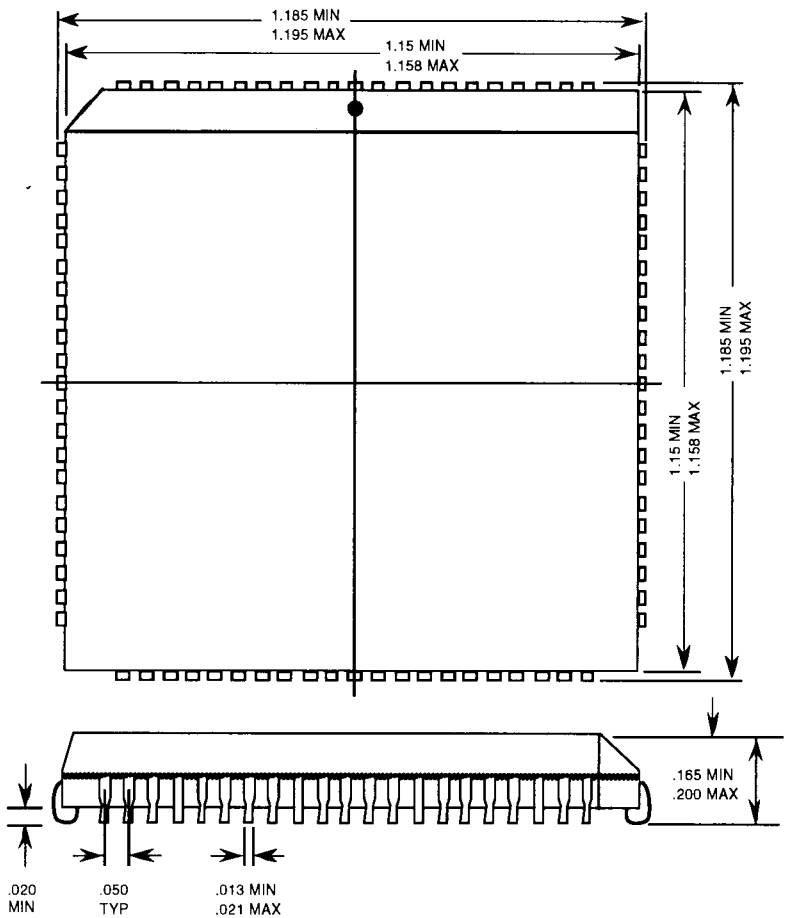


Figure 7–14. Un-Clocked Bus Interface Write Cycle, Intel®-Style Handshake

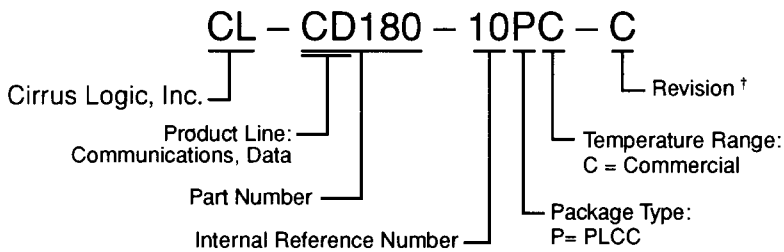
8. PACKAGE DIMENSIONS — 84-Pin PLCC



NOTE: The dimensions are in inches.

9. ORDERING INFORMATION

The order number for the 100-pin part is:



† Contact Cirrus Logic, Inc. for up-to-date information on revisions.

Appendix A

Differences Between Revision B and Revision C

The CL-CD180 Revision C is an enhanced version of the CL-CD180 Revision B octal UART. Several new features have been provided that add flexibility to both the hardware and software interface of the device. The Revision C device provides a 25% performance increase over the Revision B device, the result of increasing the clock speed from 10 MHz maximum to 12.5 MHz in Revision C. The clock input for a Revision C device requires a 47%/53% (or better) duty cycle (see Section 7 of this data book).

The pinout of the CL-CD180 Revision C version is identical to the CL-CD180 Revision B device.

The GFRCR Register of the CL-CD180 Revision C device indicates its version by returning a value of hex 82 when read. The bit definition is as follows:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
1	0	0	0	0	0	1	0

The sections below describe the new features and capabilities of the Revision C device.

A.1 MSVDTR and MSVRTS — Separated Control of DTR and RTS Outputs

In Revision C devices, two new registers have been added to aid manual control of the DTR and RTS outputs. They are the MSVDTR and MSVRTS Registers (address hex 2A and 29, respectively). In the Revision B device, it is not possible to write to either DTR or RTS without affecting the other. These new registers in Revision C allow individual access to the respective modem output pin.

MSVDTR Register:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
X	X	X	X	X	X	DTR	X

MSVRTS Register:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
X	X	X	X	X	X	X	RTS

A.2 Active DTACK* Release

The Revision C device provides an active driver on the DTACK* output to reduce the signal rise time on the trailing edge of the DTACK* strobe. In the CL-CD180 Revision B device, DTACK* was simply released (three-stated) at the end of the DTACK cycle, and it relied on the external pull-up resistor to pull the signal to the logic '1' state. Depending on the value of the resistor chosen and the number of devices on the DTACK* line, signal rise times could be quite long. The Revision C implements a special driver on the DTACK* output that actively drives the DTACK* to the logic '1' state and then immediately releases it so that it does not interfere with other devices on the shared signal. The pull-up resistor, however, is still required to keep the signal at the logic '1' state when it is not being driven.

A.3 Register-Based Acknowledge

Five new registers have been added to Revision C to provide a more flexible host interface. Three of these new registers have been provided for systems that do not generate the IACKIN* Signal as a normal part of an interrupt service routine or that do not implement an interrupt-based system. These registers allow the interrupt acknowledge cycle to be initiated by performing a simple register read cycle. Thus, they are in the regular register address map and are selected in the same way as any other register. For easier implementation of polled systems, the two other new registers provide interrupt status; no external logic is necessary. These two new registers also allow interrupt configuration to enable the new features.

These five registers are: Receive Request Acknowledge Register (RRAR), Transmit Request Acknowledge Register (TRAR), Modem Request Acknowledge Register (MRAR), Service Request Configuration Register (SRCR), and Service Request Status Register (SRSR).

A.3.1 Receive Request Acknowledge Register (RRAR) Address = 77 (hex) (Read Only), Transmit Request Acknowledge Register (TRAR) Address = 76 (hex) (Read Only), Modem Request Acknowledge Register (MMAR) Address = 75 (hex) (Read Only)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
X	X	X	X	X	IT2	IT1	IT0

These three virtual registers provide a means to acknowledge a service request without implementing an ACKIN cycle. For example, if the Revision C device is requesting service for a transmit condition, a read from the TRAR address will put the device in the service acknowledge context in the same manner as an IACKIN* cycle (with a proper match for the PILRx Registers) would have done on the Revision B device. The vector that will be returned during the read cycle will be the same as is provided by the GIVR (now GSVR) Register during the IACKIN* cycle. The value will be whatever was previously loaded into the five most significant bits of the GSVR with the three remaining bits (IT2-IT0) providing the request type vector. For receive service requests, the host reads from the RRAR address, and for modem requests, it reads from the MRAR address. At the end of the service routine, host software must still perform the dummy write to the EOIR (now EOSRR) Register to terminate the service and take the device out of the service context. There is a slightly longer address setup time for accesses from the RRAR, TRAR, and MRAR; these will be provided when device characterization is complete but will be approximately 15 to 20 ns as opposed to the 6 to 10 ns required for a normal register read/write.

NOTE: These registers are only accessible if register-based acknowledges are enabled in the Service Request Configuration Register (see below). If not enabled this way, reading from these registers will have unpredictable results.

A.3.2 Service Request Configuration Register (SRCR) Address = 66 (hex)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	RegAckEn	DaisyEn	GlobPri	Unfair	not used	AutoPri	PriSel

This register is used to enable any or all of the new options available for service requests in the CL-CD180 Revision C. Each function is described below.

RegAckEn — Register-Based Acknowledge Enable

If this bit is '0', register-based acknowledgments are not accepted and the results of a read of any of the service acknowledgment registers is undefined. This enable bit provides backwards compatibility with Revision-B-based software. To enable any of the new features, this bit must be set so that previous versions of a software driver will not inadvertently activate the register-based acknowledge.

DaisyEn — Enable Daisy-Chain Register-Based Acknowledgments

When DaisyEn is a '1', a Revision C device addressed with a register-based service acknowledgment (a read occurs from a register acknowledgment address) for which the device does not have a pending service request, asserts IACKOUT* to pass the acknowledgment down the daisy chain. The next CL-CD180 in the chain will see the acknowledgment as an IACKIN* acknowledgment. The Service Request Acknowledge Register addresses *must* be placed in the PILR Registers as part of the user setup for daisy-chaining of register-based service acknowledgments. If daisy-chaining of register-based service acknowledgments is not used, the PILR Registers may be programmed with any address codes the user chooses for use with the 'normal' IACKIN* service acknowledge mechanism.

If DaisyEn is a '0', and a Revision C device is addressed with a register-based service acknowledgment for which the device does not have a pending service request, it will respond by providing an interrupt vector with a modification code of '000'.

RegAckEn must be a '1' to enable register-based service acknowledgments.

GlobPri — Set Mode of Global (Across Multiple CL-CD180 Revision C Devices) Priority

When AutoPri is used, if GlobPri is a '1', then the Revision C device will prioritize across multiple Revision C devices sharing REQ lines. If GlobPri is a '0', then the device accepts the acknowledge for the highest priority on-chip interrupt. In both cases, automatic prioritizing is only performed for Type 1 (the modem signal change type) interrupt acknowledgments. Both the IACKIN* mechanism and the register-based acknowledge mechanism are subject to AutoPri treatment.

UnFair — Fairness Override Bit

If UnFair is a '0', normal fair-share interrupt control is performed. If UnFair is a '1', the fair bits are all forced to '1', disabling the fair-share mechanism. This is useful when the Automatic Priority Option is used and the different REQ lines are wire-OR'ed together.

AutoPri — Automatic Prioritizing Select

When this bit is set, it means that the device should prioritize service requests in the manner selected by the PriSel Bit. In conjunction with the GlobPri Bit, either local (within the chip) or global (across daisy-chained chips) prioritization is done. With AutoPri set, automatic prioritization is performed when a MREQ interrupt is acknowledged. Acknowledgments of TREQ and RREQ interrupts continue to be unique and specific even with AutoPri set. This offers a form of local override to automatic prioritization for transmit or receive service when the user wishes to continue in a specific priority service routine.

If AutoPri is not set, the user must indicate the service request being acknowledged by the choice of Service Request Acknowledge Register.

PriSel — Prioritized Service Request Order Select

If AutoPri is set, the PriSel Bit selects the highest priority service request. If PriSel is a '0', receive requests have the highest priority. If PriSel is a '1', transmit requests have the highest priority. Modem Signal change request priority is fixed at the lowest priority.

A.3.3 Service Request Status Register (SRSR) Address = 65 (hex)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ILV[1]	ILV[0]	RREQext	RREQint	TREQext	TREQint	MREQext	MREQint

This register provides information about the current state of the device as related to service requests. Three different types of information is available from this register:

TREQint, RREQint, MREQint — Internal Requests Active

If the Revision C device has any of the service requests posted, the corresponding bit will be set.

TREQext, RREQext, MREQext — External Requests Active

These three bits reflect the current state of the external request pins. If a request is active (whether internal or external), its corresponding bit will be set. For example, if the TREQext is set and the TREQint is not, then another device in the chain is requesting service. However, if both the TREQint and TREQext are set, then at least the device being queried is requesting service (another device in the chain might also be requesting service on this same level).

ILV[1], ILV[0] — Internal Service Context Code

These two bits indicate the current context code of the service request context stack. The encoding is as follows:

ILV[1]	ILV[0]	Context
0	0	Not in a service request context
1	1	In a receive service request context
1	0	In a transmit service request context
0	1	In a modem service request context

A.4 R/W* Hold Time After CS* and DS*

This AC parameter (Number 4 in Table 7–2) has been changed to 5 ns (minimum) in the Revision C device.

Direct Sales Offices

Domestic

N. CALIFORNIA
San Jose
TEL: 408/436-7110
FAX: 408/437-8960

S. CALIFORNIA
Tustin
TEL: 714/258-8303
FAX: 714/258-8307

Thousand Oaks
TEL: 805/371-5381
FAX: 805/371-5382

ROCKY MOUNTAIN AREA
Denver, CO
TEL: 303/786-9696
FAX: 303/786-9695

SOUTH CENTRAL AREA
Austin, TX
TEL: 512/794-8490
FAX: 512/794-8069

Plano, TX
TEL: 214/985-2334
FAX: 214/964-3119

CENTRAL AREA
Chicago, IL
TEL: 708/490-5940
FAX: 708/490-5942

NORTHEASTERN AREA
Andover, MA
TEL: 508/474-9300
FAX: 508/474-9149

New Brunswick, NJ
TEL: 908/603-7757
FAX: 908/603-7756

SOUTH EASTERN AREA
Boca Raton, FL
TEL: 407/362-5225
FAX: 407/362-5235

International

GERMANY
Herrsching
TEL: 49/08152-2030
FAX: 49/08152-6211

JAPAN
Tokyo
TEL: 81/3-3340-9111
FAX: 81/3-3340-9120

SINGAPORE
TEL: 65/3532122
FAX: 65/3532166

TAIWAN
Taipei
TEL: 886/2-718-4533
FAX: 886/2-718-4526

UNITED KINGDOM
Hertfordshire, England
TEL: 44/0727-872424
FAX: 44/0727-875919

The Company

Cirrus Logic, Inc., produces high-integration peripheral controller circuits for mass storage, graphics, and data communications. Our products are used in leading-edge personal computers, engineering workstations, and office automation equipment.

The Cirrus Logic formula combines innovative architectures in silicon with system design expertise. We deliver complete solutions — chips, software, evaluation boards, and manufacturing kits — on-time, to help you win in the marketplace.

Cirrus Logic's fabless manufacturing strategy, unique in the semiconductor industry, employs a full manufacturing infrastructure to ensure maximum product quality, availability and value for our customers.

Talk to our systems and applications specialists; see how you can benefit from a new kind of semiconductor company.

CIRRUS LOGIC, Inc., 3100 West Warren Ave. Fremont, CA 94538
TEL: 510/623-8300 FAX: 510/226-2180

514180-004