CIRRUS LOGIC

CL-CD1400

Preliminary Data Sheet

FEATURES

Asynchronous Features

- **Software-programmable serial data rates up to 115.2 Kbit/sec. full-duplex[1]**
- **Twelve bytes of FIFO for each transmitter and each receiver, with programmable threshold for receive-FIFO-interrupt generation**
- **Improved interrupt schemes: *Good Data*™ *Interrupt*s eliminate the need for character status check**
- **Independent bit rate selection for transmit and receive on each channel**
- **User-programmable and automatic flow control modes for the serial channels:**
  - — In-band (software) flow control via single character (XON/XOFF)
  - — Out-of-band (hardware) flow control via RTS/CTS and DTR/DSR
- **Special character recognition and generation**

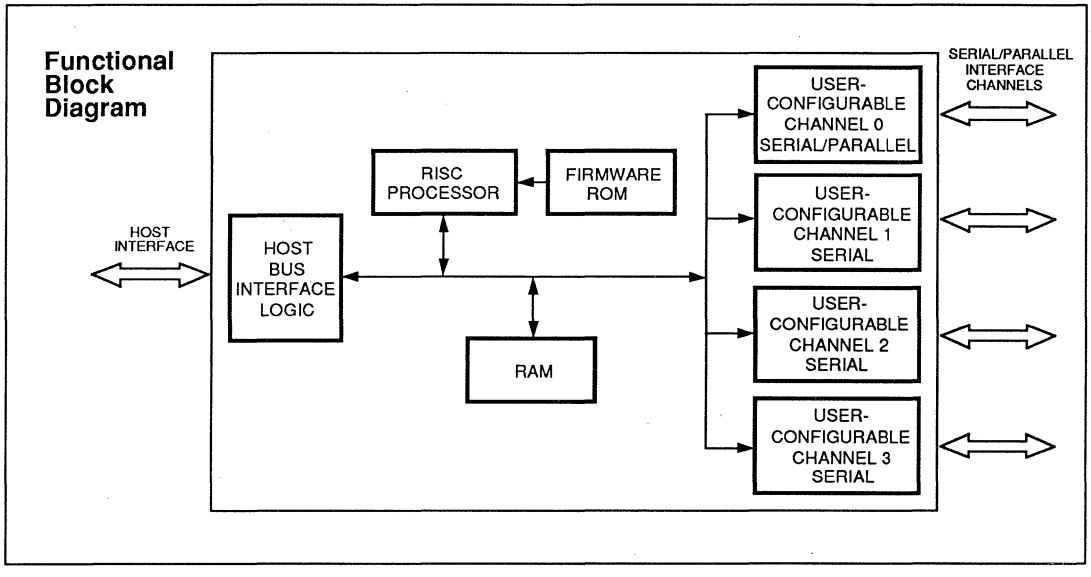
(cont.)

Four-channel Serial/Parallel Communications Engine with UNIX® Character Processing

OVERVIEW

The CL-CD1400 is a flexible asynchronous receiver/transmitter with four full-duplex serial channels, or three full-duplex serial channels and one high-speed bidirectional parallel channel. With optional special character processing capabilities, it is especially well-suited to UNIX applications. The CL-CD1400 is fabricated in an advanced-CMOS process and operates on a system clock of up to 20.2752 MHz.[2] Packaged in a 68-pin PLCC, its high throughput, low-power consumption and high level of integration permit system designs with minimum part-count, maximum performance and maximum reliability.

Functional Block Diagram



FEATURES (cont.)

- **Special character processing, particularly useful for UNIX-line-driver applications, optionally handled automatically by the CL-CD1400:**
  - — Automatic Expansion of NL to CR-NL
  - — Supports LNEXT and ISTRIP
  - — Ignore Break
  - — UNIX parity handling options:
    - Character removed from stream
    - Passed as good data
    - Replaced with null (00 hex)
    - Preceded with FF-00 hex
    - Passed as is with exception flagged
- **Line break detection and generation, with programmable choice of response/data pattern to the host**
- **Insertion of transmit delays in data stream**
- **One timer per channel for receive data time-out interrupt**
- **Local and remote maintenance loopback modes**
- **Six modem control signals-per-channel (DTR, DSR, RTS, CTS, CD, RI); CD and RI signals not available if the parallel channel is used**
- **Five to eight data bits per character plus optional parity**
- **Odd, even, no or forced parity**
- **1, 1.5 or 2 stop bits**

Parallel Features

- **Parallel data rate up to 20 Kbyte/sec.**
- **Thirty-byte FIFO**
- **Programmable strobe pulse widths**
- **Automatic generation and recognition of hand-shake control signals**
- **Compatible with Centronics®-interface specifications**

CONFIGURATION EXAMPLES

Figures 1–1 through 1–4 are functional block diagrams of four possible configurations that can be implemented with the CL-CD1400. The first is a typical workstation with printer, mouse, keyboard and modem ports, a mode which includes a single parallel port and three serial channels with modem control. Figure 1–2 illustrates one channel with complete modem control and three channels with partial modem control; the third configuration is four serial channels and one bi-directional general-purpose port with four input and four output pins. Figure 1–4 shows a quad serial mode of four channels with complete modem control. All modes of operation are software programmable via control registers within the CL-CD1400.

NOTES: [1] A minimum clock frequency is required to run all four serial channels at a 115.2K Bits data rate. Refer to the AC characteristics for complete information on device timing.

[2] 100% throughput is guaranteed up to 70K baud for full-duplex operation on all four channels simultaneously. 115.2K baud is achievable at reduced throughput. Refer to Section 4 for details.

TABLE OF CONTENTS

TABLE OF CONTENTS (cont.)

LIST OF FIGURES

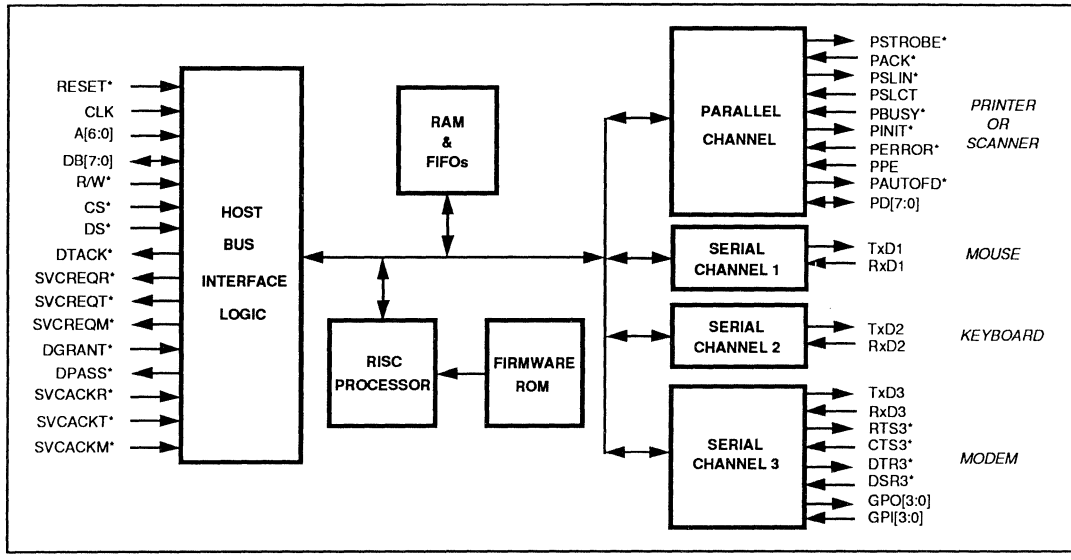


Figure 1–1. Workstation: Printer, Keyboard, Mouse and Modem Ports

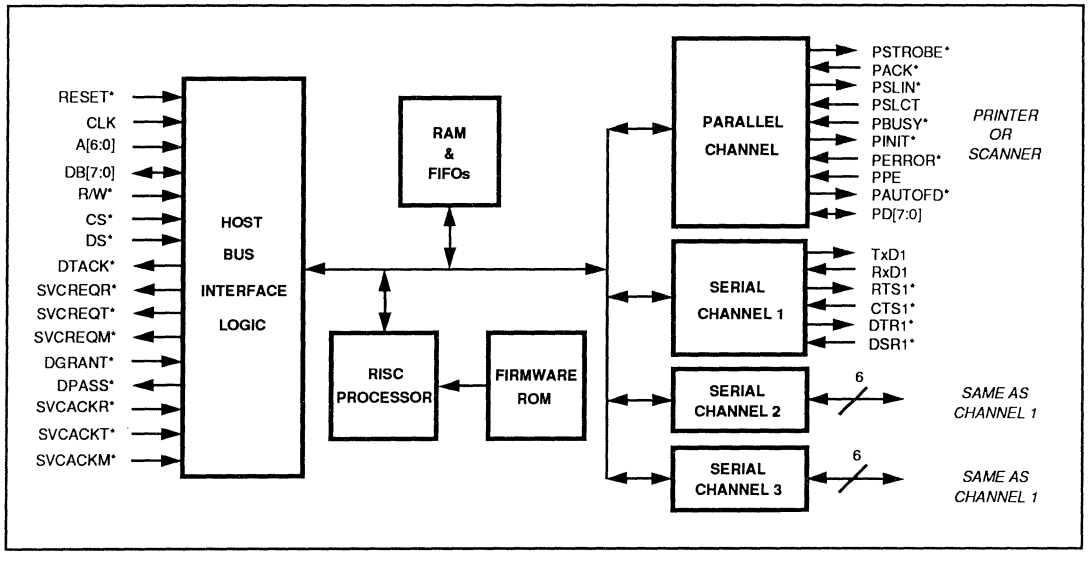


Figure 1–2. Three Serial Ports and One Parallel Port

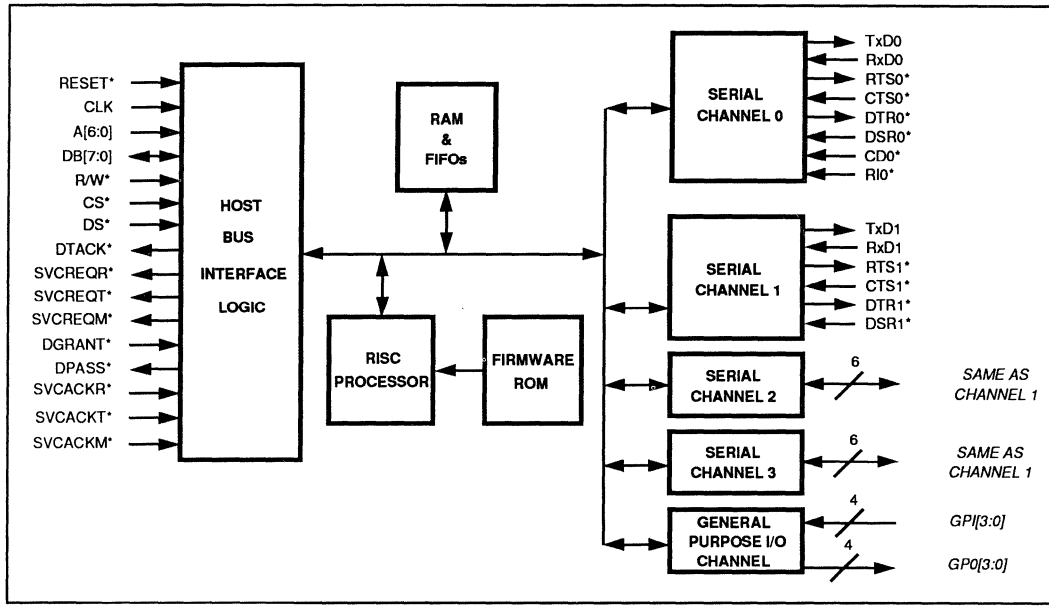


Figure 1–3. One Full-Modem Port, Three Serial Channels and One Eight-Bit I/O Port

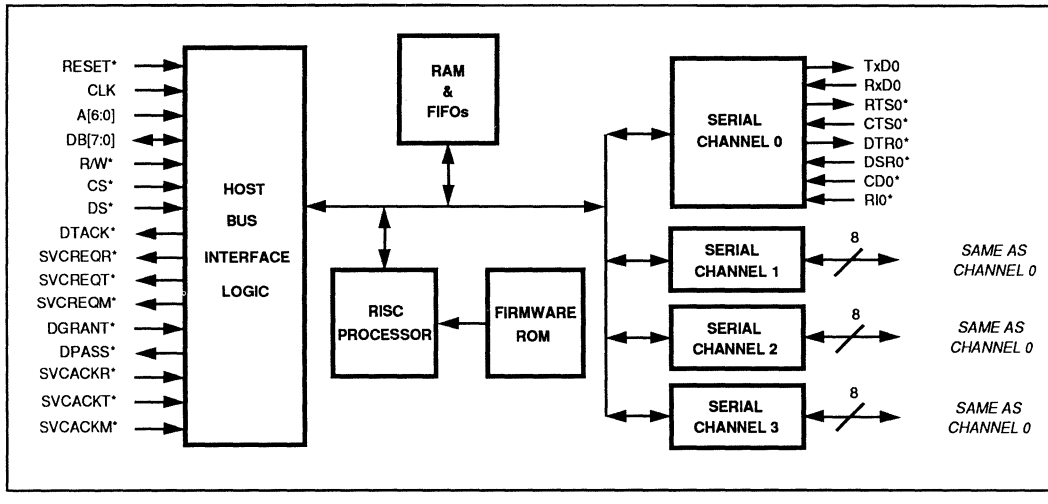
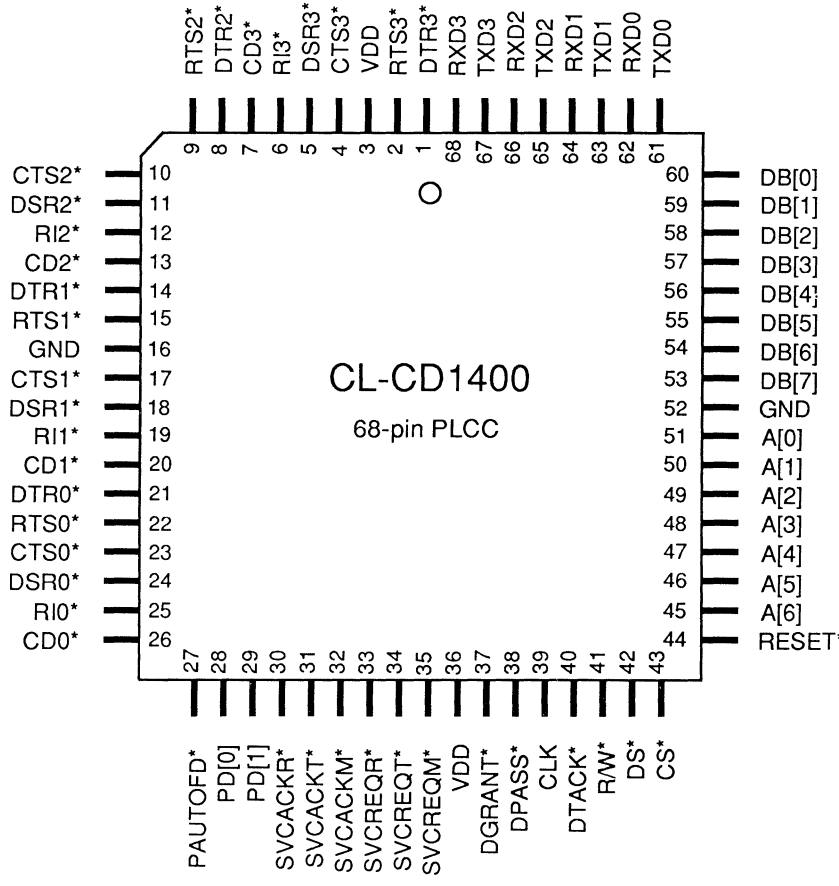


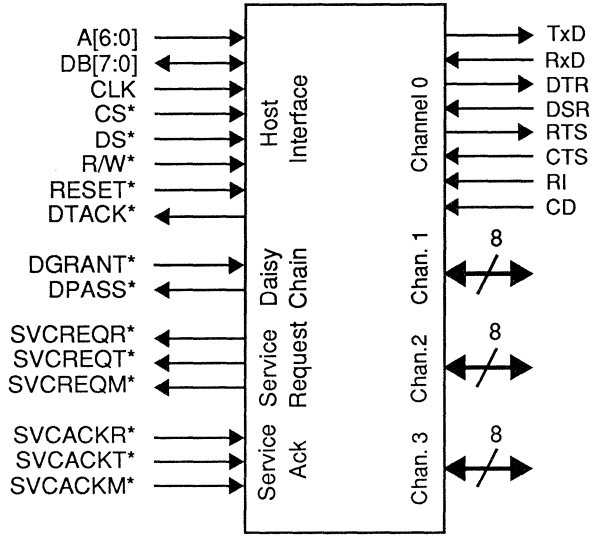
Figure 1–4. Four Full-Modem Ports

1. PIN INFORMATION

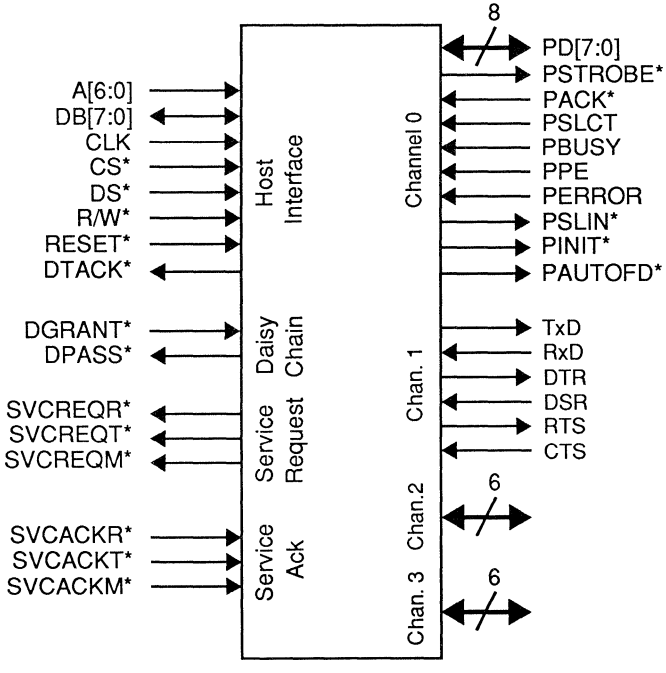
1.1 Pin Diagram for the 68-Pin PLCC Package



1.2 Pin Functions – Major Operational Modes



Pin Functions, Four Serial Channel Mode



Pin Functions, Three Serial/One Parallel Channel Mode

1.3 Pin List

The (*) after a name denotes an active low signal. Signal names in parenthesis are for the parallel channel.

I = Input; I/O = Input/Output; O = Output;

OD = Open Drain

PIN NAME	DIR	# OF PINS	PIN #
RESET*	I	1	44
CLK	I	1	39
CS*	I	1	43
DS*	I	1	42
R/W*	I	1	41
DTACK*	OD	1	40
A[6:0]	I	7	45-51
DB[7:0]	I/O	8	53-60
SVCREQR*	OD	1	33
SVCREQT*	OD	1	34
SVCREQM*	OD	1	35
SVCACKR*	I	1	30
SVCACKT*	I	1	31
SVCACKM*	I	1	32
DGRANT*	I	1	37
DPASS*	O	1	38
TXD0 (PSTROBE*)	O	1	61
RXD0 (PACK*)	I	1	62
RTS0* (PSLIN*)	O	1	22
CTS0* (PSLCT)	I	1	23
DSR0*/ (PBUSY)	I	1	24
DTR0* (PINIT*)	O	1	21
CD0* (PERROR*)	I	1	26
RI0* (PPE)	I	1	25
PAUTOFD*	O	1	27

PIN NAME	DIR	# OF PINS	PIN #
TXD1	O	1	63
RXD1	I	1	64
RTS1*	O	1	15
CTS1*	I	1	17
DSR1*	I	1	18
DTR1*	O	1	14
CD1* (PD[2])	I/O	1	20
RI1* (PD[3])	i/O	1	19
TXD2	O	1	65
RXD2	I	1	66
RTS2*	O	1	9
CTS2*	I	1	10
DSR2*	I	1	11
DTR2*	O	1	8
CD2* (PD[4])	I/O	1	13
RI2* (PD[5])	I/O	1	12
TXD3	O	1	67
RXD3	I	1	68
RTS3*	O	1	2
CTS3*	I	1	4
DSR3*	I	1	5
DTR3*	O	1	1
CD3* (PD[6])	I/O	1	7
RI3* (PD[7])	I/O	1	6
PD[0]	I/O	1	28
PD[1]	I/O	1	29
VCC	I	2	3,36
GND	I	2	16,52

1.4 Pin Descriptions

Symbol	Pin Number	Type	Description
RESET*	44	I	RESET – Asynchronously resets the CL-CD1400. RESET* must be active for a minimum of ten system clocks. When RESET* is removed, the CL-CD1400 will perform a software initialization of its registers, disable all transmitters and receivers, and when complete, place the firmware revision number in the GFRCR.
CLK	39	I	CLOCK – System clock. The CL-CD1400 requires a nominal 20.2752-MHz clock for proper operation. The system clock is divided by two, internally, to generate all on-chip timing clocks.
CS*	43	I	Chip Select – When active, CS*, in conjunction with DS*, initiates a host I/O cycle with the CL-CD1400.
DS*	42	I	Data Strobe – During an active I/O cycle, DS* strobes data into on-chip registers during a write cycle or enables data onto the data bus during read cycles.
R/W*	41	I	Read/Write – R/W* sets the direction of the data transfer between the host and the CL-CD1400. When high, the cycle is a Read, and when low, the cycle is a Write.
DTACK*	40	OD	Data Transfer Acknowledge – When the CL-CD1400 has completed internal operations associated with a host I/O cycle, it activates DTACK* to indicate the end of the cycle. The host may terminate the cycle as soon as DTACK* becomes active.
A[6:0]	45-51	I	Address[6:0] – These signals select the on-chip register being accessed during a host I/O cycle.
DB[7:0]	53-60	I/O	Data Bus[7:0] – These eight bidirectional signals are the data interface between the host and internal CL-CD1400 registers.
SVCREQR*	33	OD	Service Request Receive – When the CL-CD1400 needs host service for one of the receivers, it activates this signal.

1.4 Pin Descriptions *(cont.)*

Symbol	Pin	Type	Description
SVCREQT*	34	OD	Service Request Transmit – When the CL-CD1400 needs host service for one of the transmitters, it activates this signal.
SVCREQM*	35	OD	Service Request Modem – The CL-CD1400 activates this signal when an enabled change occurs.
SVCACKR*	30	I	Service Acknowledge Receive – The host activates this signal to start a receive interrupt service. This is a special case read cycle, during which the CL-CD1400 places the contents of the receive interrupt vector register on the data bus.
SVCACKT*	31	I	Service Acknowledge Transmit – The host activates this signal to start a transmit interrupt service. This is a special case read cycle, during which the CL-CD1400 places the contents of the transmit interrupt vector register on the data bus.
SVCACKM*	32	I	Service Acknowledge Modem – The host activates this signal to start a modem interrupt service. This is a special case read cycle, during which the CL-CD1400 places the contents of the modem interrupt vector register on the data bus.
DGRANT*	37	I	Daisy Grant – This input, qualified with DS* and a valid service acknowledge (SVCACKR*, SVCACKT*, SVCACKM*), activates the CL-CD1400 service acknowledge cycle.
DPASS*	38	O	Daisy Pass – This output is driven low when no valid service request exists for the type of service acknowledge active. In multiple-CL-CD1400 designs, this signal is normally connected to the following CL-CD1400's DGRANT* input, forming a service acknowledge daisy chain.
TxD[3:0]	67,65,63,61	O	Transmit Data[3:0] – These output signals provide the serial transmit data stream for all four channels. When channel 0 is operating in parallel mode, TxD0 becomes PSTROBE* (see PSTROBE*).

1.4 Pin Descriptions *(cont.)*

Symbol	Pin	Type	Description
RxD[3:0]	68,66,64,62	I	Receive Data[3:0] – These input signals carry the serial bit streams into the CL-CD1400. When channel 0 is programmed for parallel operation, RxD0 becomes PACK* (see PACK*).
RTS[3:0]*	2,9,15,22	O	Request To Send[3:0] – The request to send output from each channel. These signals are controlled by Modem Signal Value Register 1 inside the CL-CD1400. RTS0* serves a dual purpose based on the mode of operation of channel 0 (see PSLIN*).
CTS[3:0]*	4,10,17,23	I	Clear To Send[3:0] – Clear To Send inputs for each channel. If enabled, these signals can control the transmitter, enabling transmission when active, and disabling transmission when inactive. CTS0* serves a dual purpose based on the mode of operation of channel 0 (see PSLCT*).
DSR[3:0]*	5,11,18,24	I	Data Set Ready[3:0] – Data Set Ready for each channel. DSR0* serves a dual purpose based on the mode of operation of channel 0 (see PBUSY*).
DTR[3:0]*	1,8,14,21	O	Data Terminal Ready[3:0] – Data Terminal Ready for each channel. These signals are under control of Modem Signal Value Register 2. DTR0* serves a dual purpose based on the mode of operation of channel 0 (see PINIT*).
CD[3:0]* PD[6], PD[4], PD[2], PERROR	7,13,20,26	I	Carrier Detect[3:0] – Carrier Detect for each channel. These signals can be monitored via the Modem Signal Value Registers. CD0* serves a dual purpose based on the mode of operation of Channel 0 (see PERROR). CD1*, CD2* and CD3* serve dual purposes as parallel data bits 2, 4 and 6 (PD[2], PD[4] and PD[6]) when channel zero is operating in parallel mode.
RI[3:0]* PD[7], PD[5], PD[3], PPE	6,12,19,25	I	Ring Indicator[3:0] – Ring Indicator for each channel. These signals can be monitored via the Modem Signal Value Registers. RI0* serves a dual purpose based on the mode of operation of channel 0 (see PPE). RI1*, RI2* and RI3* serve dual purposes as parallel data bits 3, 5 and 7 (PD[3], PD[5] and PD[7]) when channel zero is operating in parallel mode.

1.4 Pin Descriptions *(cont.)*

Symbol	Pin	Type	Description
PSTROBE*	61	O	Printer Strobe – This is the alternate function for TxD0 when channel 0 is programmed as a parallel port. When the port is selected for output (printer), PSTROBE* is driven active by the CL-CD1400 after a proper data set up time. Data is held for a proper hold time after PSTROBE* is deactivated. When channel 0 is programmed as an input (scanner) port, PSTROBE* acts as the acknowledge pin to signal completion of data reception.
PACK*	62	I	Printer Acknowledge – This is the alternate function of RxD0 when channel 0 is programmed as a parallel port. When the port is selected to output (printer), this signal is used by the CL-CD1400 to signal completion of data reception by the printer, and it will begin the next I/O cycle. When channel 0 is selected as input (scanner), PACK* is treated as the strobe input. Proper data set-up and hold times are required.
PSLIN*	22	O	Printer Select in
PINIT*	21	O	Printer Initialize
PAUTOFD*	27	O	Printer Autofeed These three signals are general-purpose outputs. Their state is controlled by the lower three bits of the PSVR register (see the register descriptions for detailed information on register bit assignments). PSLIN* and PINIT* are alternate functions for RTSO* and DTRO*, depending on the mode of operation on channel 0. PAUTOFD* is a single-function output pin.
PSLCT*	23	I	Printer Select
PPE*	25	I	Printer Paper Empty
PERROR	26	I	Printer Error These three signals are general-purpose inputs. Their state can be monitored via the upper four bits of the PSVR register. As with their modem input counterparts (CTS0*, RI0* and CD0*), a change in state can be programmed to generate a SVCREQM*. The function of these signals is automatically selected based on the mode of operation programmed for channel 0.

1.4 Pin Descriptions *(cont.)*

Symbol	Pin	Type	Description
PBUSY	24	I/O	Printer Busy PBUSY is a bi-directional signal: input when transmit is enabled, and output when receive is enabled. During receive data operations, the CL-CD1400 drives PBUSY active after receiving the strobe from the remote. When it has taken the data, it deasserts PBUSY and activates PACK*. During transmit data operations, the state of PBUSY is made available to the host via the PSVR register; however, it does not affect transfer operation and is not a handshake signal for this direction.
PD[0]	28	I/O	Parallel Data bit 0 – When channel 0 is operating in parallel mode, this pin provides the parallel data bit zero.
PD[1]	29	I/O	Parallel Data bit 1 – When channel 0 is operating in parallel mode, this pin provides the parallel data bit one.

2. REGISTERS

All communication with the CL-CD1400 takes place through a large array of registers. Registers are considered to be one of three types: global, virtual and per-channel. Global registers affect all channels within the device. Per-channel registers pertain only to the channel being referenced. Global registers are always available for host access; access to a particular channel's local registers requires selecting that channel's register set. Virtual registers are only available to the host during the context of a service routine. There are four sets of per-channel registers, one for each channel. Selection of the register set is accomplished by writing the channel number (0 - 3) into the Channel Access Register (CAR). This causes a "bank switch" action, allowing the registers of the selected channel to be accessed. At any given time, only one channel's registers are available. Once selected, this register set remains available until the CAR is changed by the host.

The tables on the following pages define the register symbols, names, read and write access modes, and the internal offset address for each register in the CL-CD1400. The offset address is applied to the address bus (A[6:0]) during a host I/O cycle to select a particular register. A detailed description of host interface is presented in Section 4.

In the register bit definitions immediately following the register tables, some registers are shown with two functions. In these cases, the first definition applies to channel 0's serial operation mode, and the second to parallel mode. For channels 1 through 3, only the function labeled "Serial" applies.

Section 5 presents a detailed description of register programming.

Note that the addresses are shown relative to the CL-CD1400's definition of address lines. In 16- and 32-bit systems, it is a common practice to connect 8-bit peripherals to only one byte lane. Thus, in 16-bit systems, the CL-CD1400 appears at every other address; for example, the CL-CD1400's A0 is connected to the host's A1. In 32-bit systems, the CL-CD1400 appears at every fourth address; (the CL-CD1400's A0 is connected to the host's A2). In either of these cases, the addresses used by the programmer will be different than what is shown.

For instance, in a 16-bit Motorola 68000-based system, the CL-CD1400 is placed on data lines D0-D7, which are at odd addresses in the Motorola manner of addressing. The CL-CD1400's A0 is connected to the 68000's A1, etc. Thus, CL-CD1400 address x'40 becomes x'81 to the programmer. It is 'left-shifted' 1 bit, and A0 must be '1' for low-byte (D0-D7) accesses.

2.1 CL-CD1400 Register Map

2.1.1 Global Registers

Symbol	Register Name	R/W	A[6:0]	(Hex)
GFRCR	Global Firmware Revision Code Register	R/W	100 0000	40
CAR	Channel Access Register	R/W	110 1000	68
GCR	Global Configuration Register	R/W	100 1011	4B
SVRR	Service Request Register	R	110 0111	67
RICR	Receive Interrupting Channel Register	R/W	100 0100	44
TICR	Transmit Interrupting Channel Register	R/W	100 0101	45
MICR	Modem Interrupting Channel Register	R/W	100 0110	46
RIR	Receive Interrupt Register	R/W	110 1011	6B
TIR	Transmit Interrupt Register	R/W	110 1010	6A
MIR	Modem Interrupt Register	R/W	110 1001	69
PPR	Prescale Period Register	R/W	111 1110	7E

2.1.2 Virtual Registers

Symbol	Register Name	R/W	A[6:0]	(Hex)
RIVR	Receive Interrupt Vector Register	R	100 0011	43
TIVR	Transmit Interrupt Vector Register	R	100 0010	42
MIVR	Modem Interrupt Vector Register	R	100 0001	41
TDR	Transmit Data Register	W	110 0011	63
RDSR	Receive Data/Status Register	R	110 0010	62
MISR	Modem Interrupt Status Register	R	100 1100	4C
EOSRR	End Of Service Request Register	W	110 0000	60

2.1.3 Channel Registers

Symbol	Register Name	R/W	A[6:0]	(Hex)
LIVR	Local Interrupt Vector Register	R/W	0011000	18
CCR	Channel Command Register	R/W	0000101	05
SRER	Service Request Enable Register	R/W	0000110	06
COR1	Channel Option Register 1	R/W	0001000	08
COR2	Channel Option Register 2	R/W	0001001	09
COR3	Channel Option Register 3	R/W	0001010	0A
COR4	Channel Option Register 4	R/W	0011110	1E
COR5	Channel Option Register 5	R/W	0011111	1F
CCSR	Channel Control Status Register	R	0001011	0B
RDCR	Received Data Count Register	R	0001110	0E
SCHR1	Special Character Register 1	R/W	0011010	1A
SCHR2	Special Character Register 2	R/W	0011011	1B
SCHR3	Special Character Register 3	R/W	0011100	1C
SCHR4	Special Character Register 4	R/W	0011101	1D
SCRL	Special Character Range, Low	R/W	0100010	22
SCRH	Special Character Range, High	R/W	0100011	23
LNC	LNext Character	R/W	0100100	24
MCOR1	Modem Change Option Register 1	R/W	0010101	15
MCOR2	Modem Change Option Register 2	R/W	0010110	16
RTPR	Receive Time-out Period Register	R/W	0100001	21
MSVR1	Modem Signal Value Register 1	R/W	1101100	6C
MSVR2	Modem Signal Value Register 2	R/W	1101101	6D
PSVR	Printer Signal Value Register	R/W	1101111	6F
RBPR	Receive Baud Rate Period Register	R/W	1111000	78
RCOR	Receive Clock Option Register	R/W	1111100	7C
TBPR	Transmit Baud Rate Period Register	R/W	1110010	72
TCOR	Transmit Clock Option Register	R/W	1110110	76

2.2 Register Definitions

2.2.1 Global Registers

Global Firmware Revision Code Register (GFRCR) 40 Read/Write

Firmware Revision Code

Channel Access Register (CAR) 68 Read/Write

n/u	n/u	n/u	n/u	n/u	n/u	C1	C0

Global Configuration Register (GCR) 4B Read/Write

P/S*	n/u	n/u	n/u	n/u	n/u	n/u	n/u

Service Request Register (SVRR) 67 Read Only

0	0	0	0	0	SRM	SRT	SRR

Receive Interrupting Channel Register RICR 44 Read/Write

X	X	X	X	C1	C0	X	X

Transmit Interrupting Channel Register TICR 45 Read/Write

X	X	X	X	C1	C0	X	X

Modem Interrupting Channel Register MICR 46 Read/Write

X	X	X	X	C1	C0	X	X

Receive Interrupt Register (RIR) 6B Read/Write

rxireq	rbusy	runfair	1	1	0	ch[1]	ch[0]

Transmit Interrupt Register (TIR) 6A Read/Write

txireq	tbusy	tunfair	1	0	0	ch[1]	ch[0]

Modem Interrupt Register (MIR) 69 Read/Write

mdireq	mbusy	munfair	0	1	0	ch[1]	ch[0]

Prescaler Period Register (PPR) 7E Read/Write

Binary Value

2.2.2 Virtual Registers

Receive Interrupt Vector Register (RIVR) 43 Read Only

X	X	X	X	X	IT2	IT1	IT0

Transmit Interrupt Vector Register (TIVR) 42 Read Only

X	X	X	X	X	IT2	IT1	IT0

Modem Interrupt Vector Register (MIVR) 41 Read Only

X	X	X	X	X	IT2	IT1	IT0

Transmit Data Register (TDR) 63 Write Only

Transmit Character

Receive Data/Status Register (RDSR) 62 Read Only

Received Character	Character

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

Modem Interrupt Status Register (MISR) 4C Read Only

DSRch	CTSch	RIch	CDch	0	0	0	0

End Of Service Request Register (EOSRR) 60 Write Only

X	X	X	X	X	X	X	X

2.2.3 Channel Registers

Local Interrupt Vector Register (LIVR) 18 Read/Write

X	X	X	X	X	IT2	IT1	IT0

Channel Command Register (CCR) 05 Read/Write

Res Chan	COR Chg	Send SC	Chan Ctl	D3	D2	D1	D0

Format 1: Reset Channel Command

Res Chan	0	0	0	0	0	FTF	Type

Format 2: Channel Option Register Change Command

0	COR Chg	0	0	COR3	COR2	COR1	n/u

Format 3: Send Special Character Command

0	0	Send SC	0	0	SSPC2	SSPC1	SSPC0

Format 4: Channel Control Command

0	0	0	Chan Ctl	XMT EN	XMT DIS	RCV EN	RCV DIS

Service Request Enable Register (SRER) 06 Read/Write

MdmCh	n/u	n/u	RxData	n/u	TxRdy	TxMpty	NNDT

Channel Option Register 1 (COR1) 08 Read/Write

Parity	ParM1	ParM0	Ignore	Stop1	Stop0	ChL1	ChL0

Channel Option Register 2 (COR2) 09 Read/Write

IXM	TxIBE	ETC	LLM	RLM	RtsAO	CtsAE	DsrAE

Channel Option Register 3 (COR3) 0A Read/Write

SCDRNG	SCD34	FCT	SCD12	RxTh3	RxTh2	RxTh1	RxTh0	Serial
n/u	n/u	n/u	RxTh4	RxTh3	RxTh2	RxTh1	RxTh0	Parallel

Channel Option Register 4 (COR4) 1E Read/Write

IGNCR	ICRNL	INLCR	IGNBRK	-BRKINT	PEH[2]	PEH[1]	PEH[0]

Channel Option Register 5 (COR5) 1F Read/Write

ISTRIP	LNE	CMOE	n/u	n/u	n/u	ONLCR	OCRNL

2.2.3 Channel Registers

Channel Control Status Register (CCSR) 0B Read Only

RxEn	RxFloff	RxFlon	n/u	TxEn	TxFloff	TxFlon	n/u	Serial
RxEn	n/u	n/u	n/u	TxEn	n/u	n/u	n/u	Parallel

Received Data Count Register (RDCR) 0E Read Only

0	0	0	0	CT3	CT2	CT1	CT0	Serial
0	0	0	CT4	CT3	CT2	CT1	CT0	Parallel

Special Character Register 1 (SCHR1) 1A Read/Write

Special Character 1

Special Character Register 2 (SCHR2) 1B Read/Write

Special Character 2

Special Character Register 3 (SCHR3) 1C Read/Write

Special Character 3

Special Character Register 4 (SCHR4) 1D Read/Write

Special Character 4

Special Character Range Low (SCRL) 22 Read/Write

Character Range Low

Special Character Range High (SCRH) 23 Read/Write

Character Range High

LNext Character (LNC) 24 Read/Write

LNext Character

Modem Change Option Register 1 (MCOR1) 15 Read/Write

DSRzd	CTSzd	Rlzd	CDzd	DTRth3	DTRth2	DTRth1	DTRth0	Serial
PBUSYzd	PSLCTzd	PPEzd	PERRORzd	n/u	n/u	n/u	n/u	Parallel

2.2.3 Channel Registers *(cont.)*

Modem Change Option Register 2 (MCOR2) 16 Read/Write

DSRod	CTSod	Riod	CDod	0	0	0	0	Serial
PBUSYod	PSLCTod	PPEod	PERRORod	n/u	n/u	n/u	n/u	Parallel

Receive Time-out Period Register (RTPR) 21 Read/Write

Binary Count Value

Modem Signal Value Register 1 (MSVR1) 6C Read/Write

DSR	CTS	RI	CD	PSTROBE†	0	n/u	RTS

Modem Signal Value Register 2 (MSVR2) 6D Read/Write

DSR	CTS	RI	CD	PSTROBE†	0	DTR	n/u

Printer Signal Value Register (PSVR) 6F Read/Write

PBUSY*	PSLCT*	PPE*	PERROR	PACK*	PAUTOFD	PINIT	PSLIN

Receive Baud Rate Period Register (RBPR) 78 Read/Write

Binary Divisor Value

Receive Clock Option Register (RCOR) 7C Read/Write

n/u	n/u	n/u	n/u	n/u	ClkSel2	ClkSel1	ClkSel0

Transmit Baud Rate Period Register (TBPR) 72 Read/Write

Binary Divisor Value

Transmit Clock Option Register (TCOR) 76 Read/Write

n/u	n/u	n/u	n/u	n/u	ClkSel2	ClkSel1	ClkSel0

† Bit 3 of MSVR1 and MSVR2 show the state of the PSTROBE output only on Channel 0.

3. ELECTRICAL SPECIFICATIONS

3.1 Absolute Maximum Ratings

Supply voltage (V_{CC}): ..... +7.0 Volts

Input voltages, with respect to ground: ..... -0.5 Volts to V_{CC} +0.5 Volts

Operating Temperature (T_A): ..... 0° C to 70° C

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

Power Dissipation: ..... 0.25 Watt

NOTE: Stresses 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.

3.2 Recommended Operating Conditions

Supply Voltage (V_{CC}): 5V ± 5%
Operating Free Air Ambient Temperature: 0° C < T_A < 70° C
System Clock: 20.2752 MHz

3.3 DC Electrical Characteristics

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

Symbol	Parameter	MIN	MAX	Units	Test Conditions
V_{IL}	Input Low Voltage	-0.5	0.8	V	
V_{IH}	Input High Voltage	2.0	V_{cc}	V	(See Notes below)
V_{OL}	Output Low Voltage		0.4	V	I_{OL} = 2.4 mA; (see Notes)
V_{OH}	Output High Voltage	2.4		V	I_{OH} = -400 µA
I_{IL}	Input Leakage Current	-10	10	µA	0 < V_{IN} < V_{CC}
I_{LL}	Data Bus 3-state Leakage current	-10	10	µA	0 < V_{OUT} < V_{CC}
I_{OC}	Open Drain Output Leakage current	-10	10	µA	0 < V_{OUT} < V_{CC}
I_{CC}	Power Supply Current		100	mA	CLK = 20.2752 MHz
C_{IN}	Input Capacitance		10	pF	
C_{OUT}	Output Capacitance		10	pF	

NOTES: 1) V_{OL} for open drain signals is 0.5V @ 16 mA sinking. V_{IH} is 2.7 V minimum on RESET* and CLK.

NOTES: *(cont.)* 2) While the CL-CD1400 is a highly dependable device, there are a few guidelines which will help to insure that the maximum possible level of overall system reliability is achieved. First, the PC board should be designed to provide maximum isolation of noise. A four-layer board is preferable, but a two-layer board will work if proper power and ground distribution is implemented. In either case, decoupling capacitors mounted close to the CL-CD1400 are strongly recommended. Noise typically occurs when either the CL-CD1400's data bus drivers come out of tristate to drive the bus during a read, or when an external bus buffer turns on during a write cycle. This noise, a rapid rate-of-change of supply current, causes 'ground bounce' in the power distribution traces. This ground bounce, a rise in the voltage of the ground pins, effectively raises the input logic thresholds of all devices in the vicinity, resulting in the possibility of a '1' being interpreted as a '0'.

To reduce the possibility of ground bounce affecting the operation of the CL-CD1400, we have specified the input-high voltage (V_{IH}) of the CLOCK and RESET pins at 2.7 volts, instead of the TTL-standard 2.0 volts. This eliminates any sensitivity to ground bounce, even in very noisy systems.

Although 2.7 volts is higher than the industry-standard 2.4 volt output (V_{OH}) specified for TTL, there are several simple ways to meet this specification. One choice is to use any of the available advanced-CMOS logic families (FACT, ACL, etc.). These CMOS output buffers will pull up close to V_{CC} when not heavily loaded. In addition, AS and ALS TTL may be used if the output of the TTL device is only driving one or two CMOS loads. As noted in the Texas Instruments *ALS/AS Logic Data Book* (1986), pages 4-18 and 4-19, the V_{OH} output of these families exceeds 3.0 volts at low-current loading. Other manufacturers publish similar data. Cirrus Logic recommends the use of one of these two options for the CLK input, to insure fast, clean edges. Note that The RESET pin may, if desired, be pulled up passively with a 1K ohm (or less) resistor.

3.4 AC Characteristics

3.4.1 Asynchronous Timing

Refer to the Figures 3–1 through 3–5 on the following pages for the reference numbers in the following table.

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

Ref. #	Fig.	Parameter	MIN	MAX	Unit
t_1	3–1	RESET* Low pulse width	10		T_{CLK}
t_2	3–3	Address setup time to CS* or DS*		-20	ns
t_3	3–3	R/W* setup time to CS* or DS*		-10	ns
t_4	3–3	Address hold time after CS*	0		ns
t_5	3–3	R/W* hold time after CS*	0		ns
t_6	3–3	DTACK* low to read data valid		10	ns
t_7	3–3	DTACK* low from CS* or DS[2]	3 T_{CLK}	$5T_{CLK}$+40	ns
t_8	3–3	Data Bus Tri-state after CS* or DS* high	0	30	ns
t_9	3–3	CS* or DGRANT* high from DTACK* low	0		ns
t_{10}	3–3	DTACK* inactive from CS* or DGRANT* and DS* high		40	ns
t_{11}	3–3	DS* high pulse width	10		ns
t_{12}	3–4	Write data valid from CS* and DS* low		$2T_{CLK}$	ns
t_{13}	3–4	Write data hold time after DS* high	0		ns
t_{14}	3–2	Clock period (TCLK)[1]	49.32	Note 3	ns
t_{15}	3–2	Clock low time[1]	24.66±5%		ns
t_{16}	3–2	Clock high time[1]	24.66±5%		ns
t_{17}	3–5	Propagation delay, DGRANT* and DS* to DPASS*		35	ns
t_{18}	3–5	Setup time, SVCACK* to DS* and DGRANT*	10		ns

NOTES: 1) Timing numbers for RESET* and CLK in the table above are valid for both asynchronous and synchronous specifications.

2) On host I/O cycles immediately following SVCACK* cycles and writes to EOSRR, DTACK* will be delayed by 1 μs. On systems that do not use DTACK* to signal the end of the I/O cycle, wait states or some other form of delay generation must be used to assure that the CL-CD1400 will not be accessed until after this time period.

3) As TCLK increases, device performance decreases. A minimum clock frequencey of 20 MHz is required to guarantee performance as specified. The recommended maximum TCLK is 1000 ns.

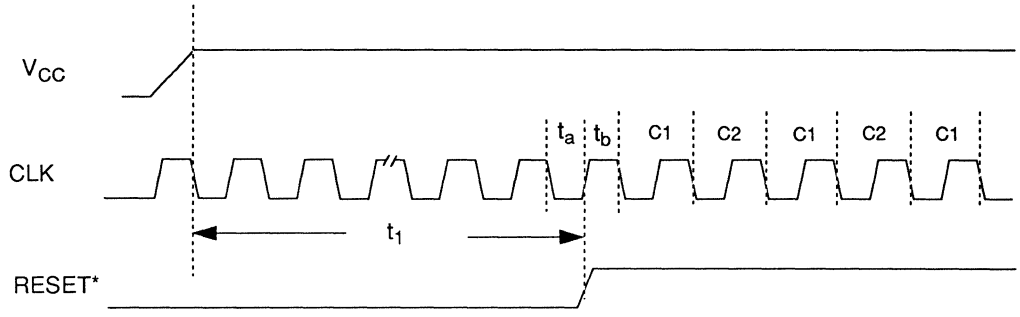


Figure 3–1. Reset Timing

NOTE: For synchronous systems, it is necessary to know the clock cycle number so that interface circuitry can stay in lock-step with the device. CLK numbers can be determined if RESET* is released within the range t_a - t_b; t_a is defined as 10 ns minimum after the falling edge of the clock; t_b is defined as 5 ns minimum before the next falling edge of the clock. If these conditions are met, the cycle starting after the second falling edge will be known to be C1. See the synchronous timing diagrams for additional information. Asynchronous systems need not be concerned with clock numbers.

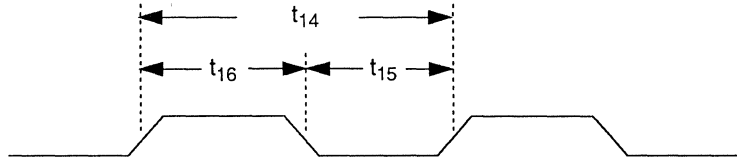


Figure 3–2. Clock Timing

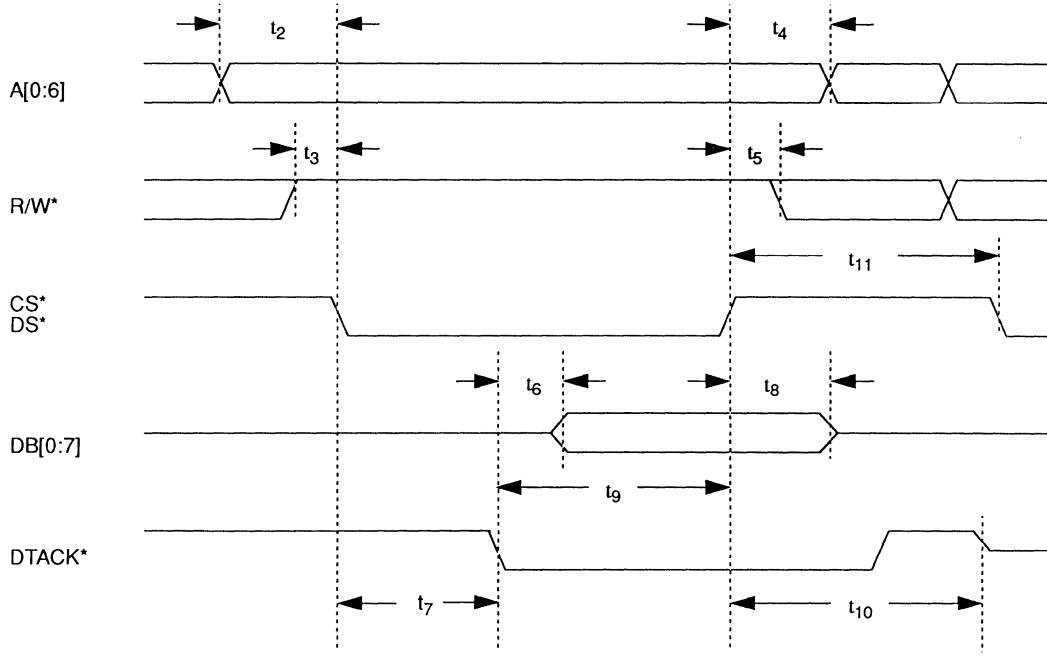


Figure 3–3. Asynchronous Read Cycle Timing

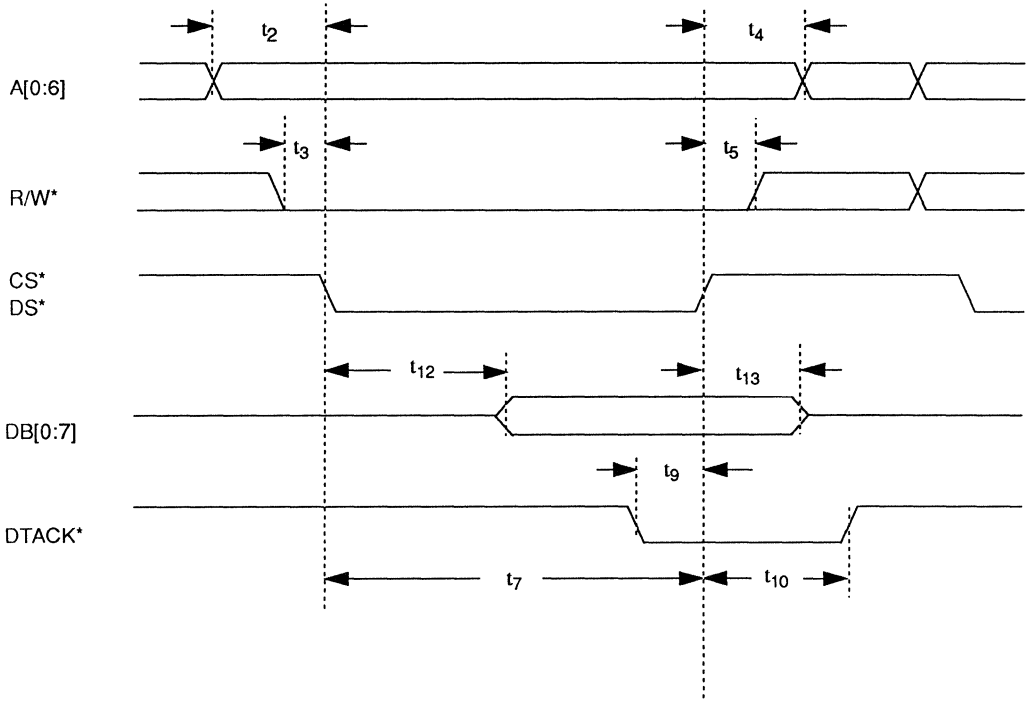


Figure 3–4. Asynchronous Write Cycle Timing

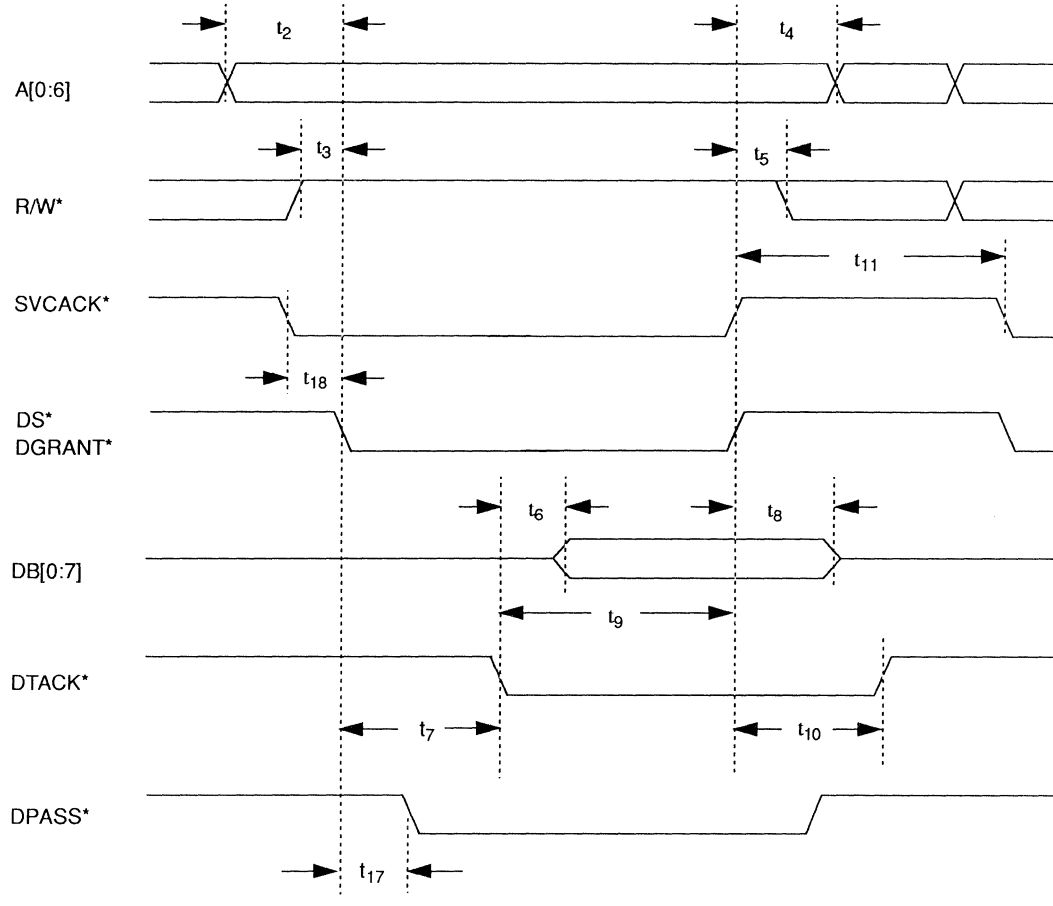


Figure 3–5. Asynchronous Service Acknowledge Cycle Timing

3.4.2 Synchronous Timing

Refer to Figures 3–6 through 3–8 on the following pages for the reference numbers in the table below.

Ref. #	Fig.	Parameter	MIN	MAX	Unit
t_1	3–6	Setup time, CS* and DS* to C1 falling edge	5		ns
t_2	3–6	Setup time, R/W* to C1 falling edge		-10	ns
t_3	3–6	Setup time, address valid to C1 falling edge		-20	ns
t_4	3–6	C3 falling edge to data valid		60	ns
t_5	3–6	DTACK* low from C4 falling edge		40	ns
t_6	3–6	CS* and DS* trailing edge to data bus high-impedance		30	ns
t_7	3–6	CS* and DS* inactive between host accesses	10		ns
t_8	3–6	Hold time, R/W* after C4 falling edge	20		ns
t_9	3–6	Hold time, address valid after C4 falling edge	0		ns
t_{10}	3–7	Setup time, write data valid to C3 falling edge	0		ns
t_{11}	3–8	Setup time, DS* and DGRANT* to C1 falling edge	20		ns
t_{12}	3–8	Setup time, SVCACK* to DS* and DGRANT*	10		ns
t_{13}	3–8	Hold time, write data valid after C4 falling edge	0		ns
t_{14}	3–8	Propagation delay, DS* and DGRANT* to DPASS*		35	ns

NOTE: On host I/O cycles immediately following SVCACK* cycles and writes to EOSRR, DTACK* will be delayed by 1 µs. On systems that do not use DTACK* to signal the end of the I/O cycle, wait states or some other form of delay generation must be used to assure that the CL-CD1400 will not be accessed until after this time period.

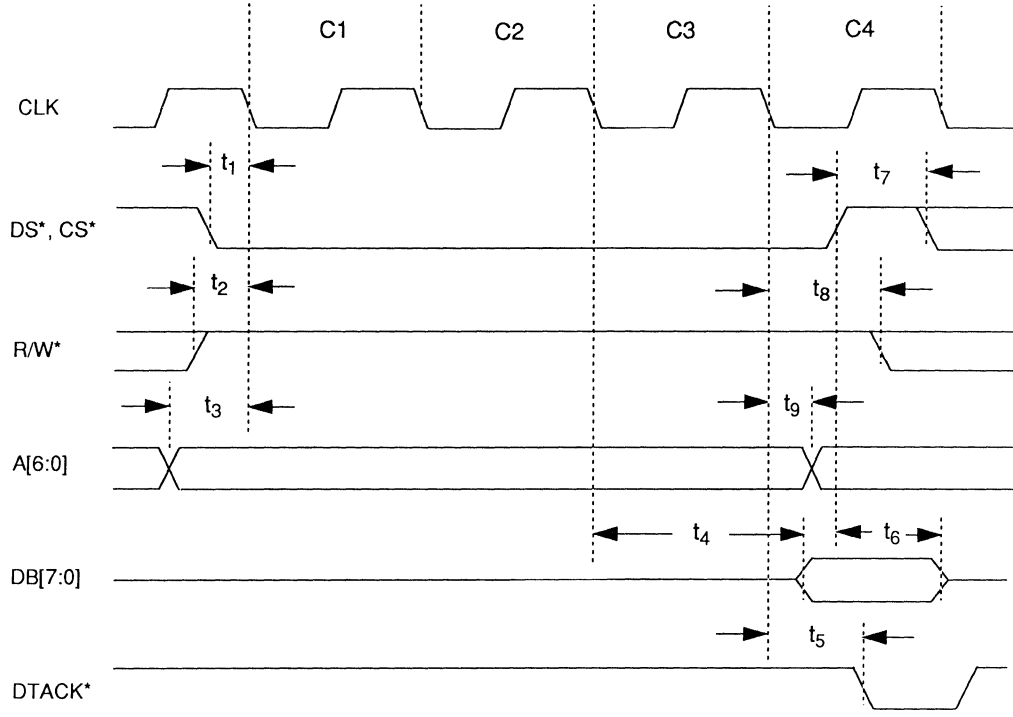


Figure 3–6. Synchronous Read Cycle Timing

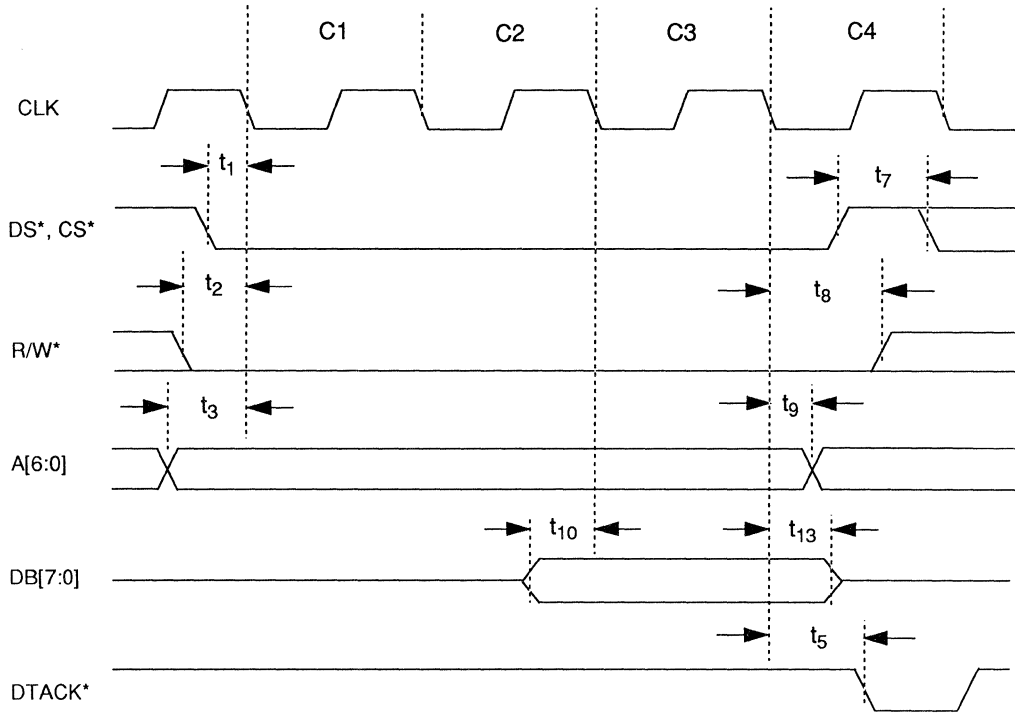


Figure 3–7. Synchronous Write Cycle Timing

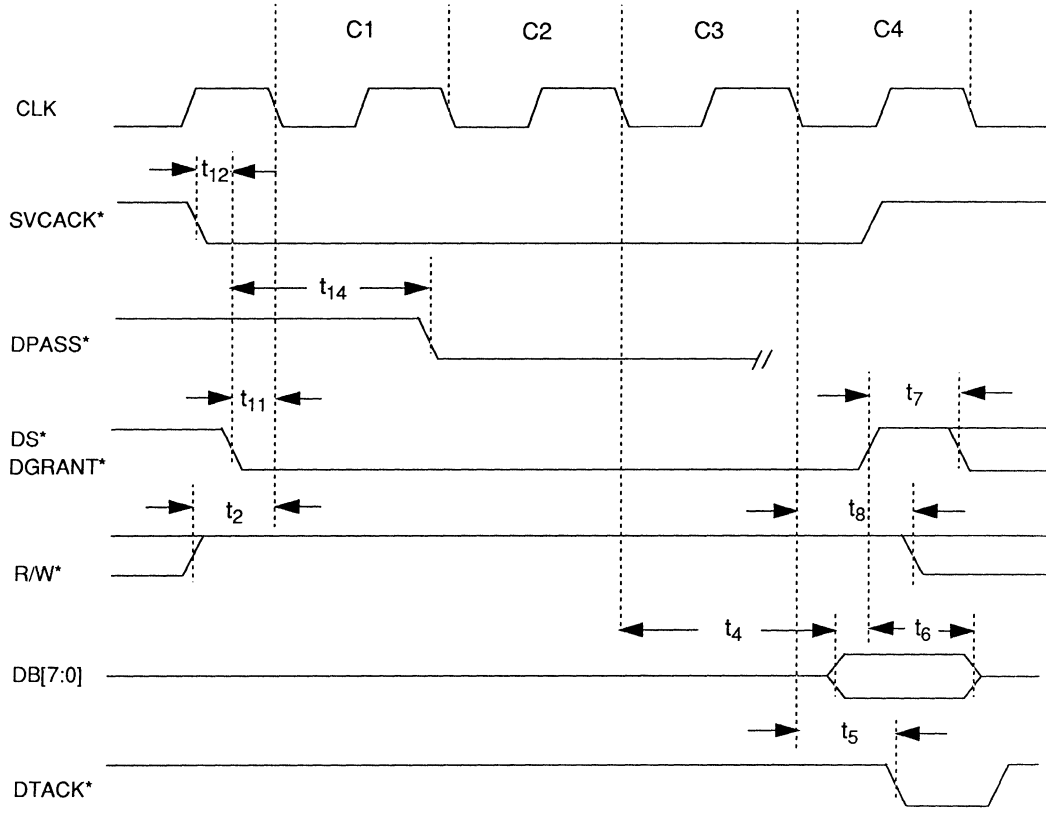


Figure 3–8. Synchronous Service Acknowledge Cycle Timing

3.4.3 Parallel Port Timing Specifications

Refer to Figures 3–9 and 3–10 for identification of reference numbers in the following table.

Note that the functions of PACK* and PSTROBE* are opposite depending on the direction of data movement, however the direction of PSTROBE* and PACK* does not change. The PACK* signal on the CL-CD1400 is always an input, and the PSTROBE* is always an output. The apparent function is changed by the external signals they are connected to and the direction of data movement. The tables below use the CL-CD1400 pin names for the signals.

The following table shows the timing specifications for the parallel port when it is programmed in the transmit mode. The PSTROBE* output provides the data strobe function and the PACK* input is connected to the acknowledge signal from the receiving device.

Transmit Timing (see Figure 3–9.)

Ref. #	Fig.	Parameter	MIN	MAX	Unit
t_{p1}	3–10	Setup time, PD[7:0] to PSTROBE* falling edge	200		ns
t_{p2}	3–10	Hold time, PD[7:0] after PSTROBE* rising edge[2]			
t_{p3}	3–10	PSTROBE* pulse width[1]			
t_{p4}	3–10	PACK* pulse width[3]	0.5		µs

The following table shows the timing specifications for the parallel port when it is programmed in the receive mode. The transmitting device connects its strobe output to the CL-CD1400 PACK* input and its acknowledge input to the CL-CD1400 PSTROBE* output.

Receive Timing (see Figure 3–10.)

Ref. #	Fig.	Parameter	MIN	MAX	Unit
t_{p5}	3–9	Setup time, PD[7:0] to PACK* falling edge	0		ns
t_{p6}	3–9	Hold time, PD[7:0] after PSTROBE* falling edge	8		µs
t_{p7}	3–9	PSTROBE* pulse width		Note 1	
t_{p8}	3–9	PACK* to PSTROBE* delay time		50 (typ)	µs
t_{p9}	3–9	PACK* pulse width[3]	0.5		µs

NOTES: 1) The width of the PSTROBE* pulse is set by the programmed value in the TCOR/TBPR register pair and will be equal to one bit-time. The recommended bit-time is approximately 10 µs.

2) PD[7:0] will be held until the receiver acknowledges the transfer by activating PACK*.

3) For highest performance, RCOR/RBPR should be programmed for a bit rate equal to 115.2 Kb/sec.

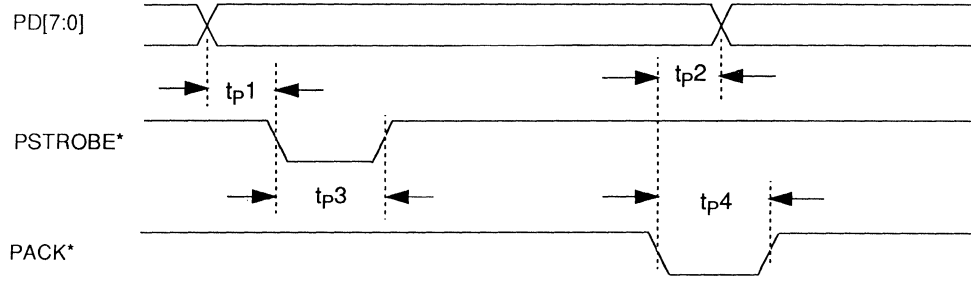


Figure 3–9. Parallel Port Transmit Timing

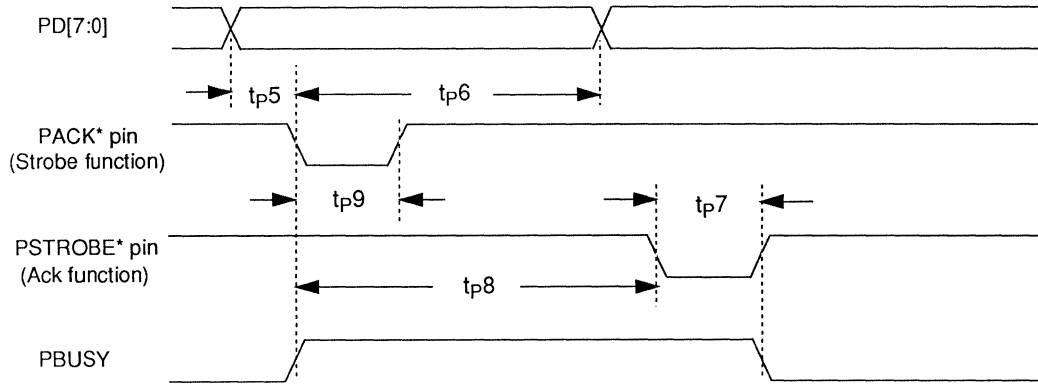


Figure 3–10. Parallel Port Receive Timing

4. FUNCTIONAL DESCRIPTION

4.1 Device Architecture

The CL-CD1400 can be described as a small computer system tailored to the function of sending and receiving serial and parallel data. It is made up of a RISC processor (MPU), RAM, ROM, host bus interface logic and serial data channels (one of which can function as a parallel port). It has special instructions and hardware that facilitate serial data manipulation.

The MPU is a true RISC processor. In addition to having a compact, efficient set of instructions, it has a 'windowed' architecture that allows it to handle one channel and its registers at a time. Before beginning any operations on a given channel, it loads an internal index register that forces all accesses to the appropriate set of registers. The index register becomes part of the internal address and allows direct addressing of the register bank and all hardware resources of the selected channel. No address computation is required in order to select the proper channel.

This same windowed scheme is carried through to the host interface as well (see Figure 4-2). For all channel specific accesses, the host first loads the Channel Access Register (CAR) with a pointer to the channel it wants to access. Thereafter, all read and write operations will take place with the proper channel. Host software need only define a register address once, and it will be valid for all channels because the CAR is used as part of the internal addressing.

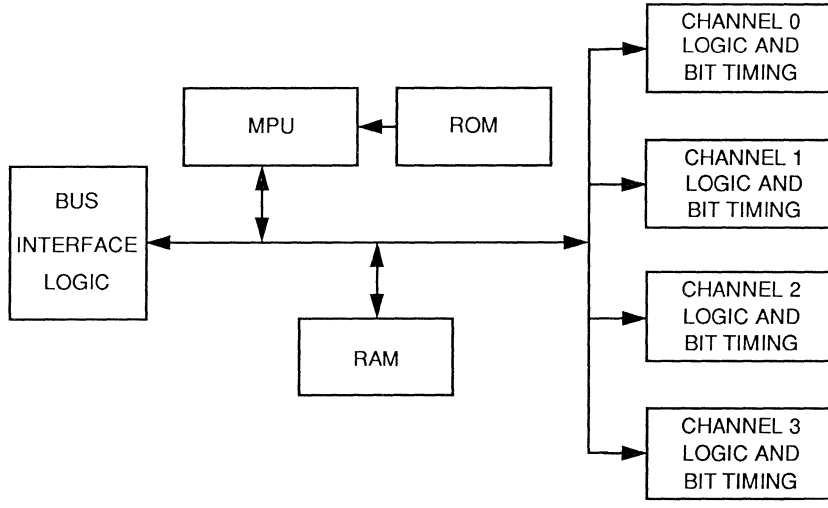


Figure 4–1. CL-CD1400 Functional Block Diagram

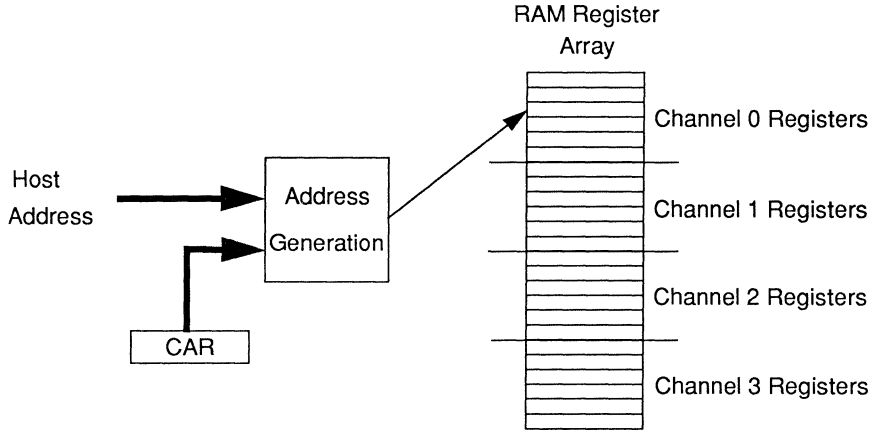


Figure 4–2. Internal Address Generation

The serial data channels are made of 'bit engines' that off-load the task of receiving and transmitting each bit from the MPU. The bit engines, after processing a complete bit, interrupt the MPU so that it can perform whatever task is required next. For example, when receiving data, the MPU will take the bit and add it to a character that is being assembled. When transmitting, it will give the bit-engine the next bit of the character being transmitted. Thus, the MPU does not need to concern itself with basic bit timing; this task is handled by the bit engines leaving it free to perform higher-level processing, such as detecting special characters.

When channel 0 is programmed to be a parallel port, the bit engines are used to set the timing of the handshake signals (PSTROBE*, PACK*).

4.2 Host Interface

The host interface to the CL-CD1400 is made up of an 8-bit bidirectional data bus, a 7-bit address bus and various strobes that identify the type of I/O cycle that is taking place. In most system designs, the I/O cycles will be merely normal host read and write cycles that activate the appropriate strobes. Although the strobe names and basic timing match that of the Motorola 68000 family, the CL-CD1400 easily fits into any CPU environment.

In most cases, when the host reads or writes an internal CL-CD1400 location, it actually accesses a location in a RAM array that serves as a bank of registers. Some locations, however, are mapped to actual hardware resources; for example, when a hard output signal is required, such as a service request output (in the SVRR), or when it is necessary to read the actual state of an input, such as a modem input.

The CL-CD1400 is, by design, a synchronous device. All internal operations take place on edges and levels (phases) of the internal clock. Note that the internal clock is generated by dividing the external (system) clock by two. When the host performs an I/O cycle with the CL-CD1400, its strobes, address and data are sampled on falling edges of the internal clock. As can be seen in the timing diagrams in Section 3, external control signals must meet setup times with respect to clock edges. Once a cycle has started, the sequence of

events is locked to the CL-CD1400's clock, with events (address setup, write data setup and read data available) occurring at predictable times.

It is not necessary, however, to design a synchronous interface to the CL-CD1400. In an asynchronous design, the Data Transfer Acknowledge (DTACK*) signal is used as an indication that the CL-CD1400 has completed the requested data transfer. Thus DTACK* can be an input to wait-state generation logic that will hold the host CPU until the operation is complete. If the strobes (Chip Select and Data Strobe – CS* and DS*) do not meet the minimum setup time with respect to a clock edge, the CL-CD1400 will not detect the I/O request, and the cycle will be delayed two full-system clock cycles, thus meeting the setup time. The I/O cycle will then commence and follow the predictable timing, with DTACK* signaling the end.

4.2.1 Host Read Cycles

Read cycles are initiated when the CL-CD1400 senses that both the CS* and DS* inputs are active and the Read/Write (R/W*) input is high. All strobes and address inputs must meet setup times as specified in the timing specifications in Section 3. It is important to note that both the CS* and DS* signals must be valid for a cycle to start, thus cycle times are measured from whichever of the two signals goes active last. The CL-CD1400 signals the fact that it has completed the read cycle (placing the data from the addressed register on the data bus pins), by activating the DTACK* signal. The read cycle is terminated when the host removes CS* and DS*.

4.2.2 Host Write Cycles

Write cycles timing and strobe activity is nearly identical to read cycles except that the R/W* signal must be held low. Write data, strobes and address inputs must meet setup and hold times as specified in the timing diagrams in Section 3. Again, the DTACK* signal is used to indicate that the cycle is complete and the CL-CD1400 has taken the data. Removing both CS* and DS* terminates the cycle.

4.2.3 Host Service Acknowledge Cycles

Service acknowledge cycles are a special-case read cycle. Timing is basically the same as a normal read cycle, and one of the SVCACK* inputs is activated instead of the CS* input (a little longer setup time is required on the SVCACK* input than on the CS* input). The data that the CL-CD1400 provides during the read cycle is the contents of the interrupt vector register associated with the type of request being acknowledged (RIVR for receive, TIVR for transmit and MIVR for modem) of the channel that is requesting service (see description of service request procedures later in this section). As with read and write cycles, DTACK* will indicate the end of the cycle and removing DS* and SVCACK* terminates the cycle.

An important fact to note about timing and service acknowledge cycles: when the host has completed the service routine and writes to the EOSRR register, a subsequent I/O cycle, if started immediately, will be delayed by approximately 1 µs. This is due to the time required by the internal processor to complete housekeeping activities associated with the switch out of the service acknowledge context. These activities are primarily FIFO pointer updates and restoration of the environment prior to the service request/service acknowledge procedure and must be completed before any internal registers are modified by the host. If the situation occurs that the host attempts an access before the internal procedures are complete, the CL-CD1400 will hold off the cycle until it is ready. In system designs which monitor DTACK*, this will not cause a problem; the cycle is extended until DTACK* becomes active, and the delay will automatically be met. If a system design does not monitor DTACK*, a mechanism must be provided to introduce the required delay.

4.3 Service Requests

From the host point of view, the CL-CD1400 operates in one of two modes: normal operation and service request/acknowledge. The normal mode of operation allows the host system to make changes and obtain current operating status on a global and per-channel basis. The service request/acknowledge mode is used when a particular channel needs service, for example when a receive FIFO has reached its programmed threshold and requires emptying. A unique behavior of the CL-CD1400 is that a service request can only be responded to after it has been placed in a service acknowledge "context". This context switch takes place when the request is acknowledged, either by activating the appropriate SVCACK* input pin, or by proper manipulation of two internal registers.

When the internal processor (MPU) detects a condition on a channel that requires host attention, it posts a service request internally and externally. The external request is the activation of one of the SVCREQ* output pins, depending on whether the type of service needed is for receive, transmit or modem signal change. Included with the internal request is a channel pointer that points to the channel requiring service. When the host service acknowledge begins, this pointer is loaded into the CAR, thus the request automatically services the proper channel. This is the purpose of the context switch, it prepares the CL-CD1400 for servicing of the proper channel. At the completion of the acknowledge procedure, the CL-CD1400 must be taken out of the acknowledge context by explicitly telling it that the procedure is complete, thus restoring the internal state to what it was before the context was switched.

It is important to remember that several of the registers within the CL-CD1400 can only be accessed when the context switch has been made and are referred to as "virtual" registers. For example, the host cannot directly place data in the transmit FIFO at any arbitrary time. It must wait for a transmit service request indicating that the FIFO is empty, and then acknowledge it. Once the acknowledge procedure has started, the transmit FIFO is available for loading.

The CL-CD1400 will make requests for service whenever an enabled need exists. The two basic ways in which the host can be made aware of these service requests is through hardware (interrupt), or software (polling internal CL-CD1400 registers). The method used will be dependent on the hardware/software design of the system; the CL-CD1400 functions well in both environments. This section discusses the trade-offs in choosing one or the other of the basic methods, and how the two can be combined for maximum performance.

4.3.1 Interrupt

The term "interrupt" is used as a generalized description of the method by which the CL-CD1400 gains the attention of the host CPU. It is used interchangeably with "service request" because the two really are the same function. "Interrupt" is often used to describe an unconditional response on the part of the host. Whether or not this is the case, the source is still the same – a service request from the CL-CD1400. The hardware signals generated by the CL-CD1400 (SVCREQR*, SVCREQT* and SVCREQM*) can be connected to the host CPU's interrupt generation/control facility and can cause it to invoke an interrupt service routine. The service routine can then begin servicing the CL-CD1400's request by starting an acknowledge sequence.

The SVCREQ* outputs can be connected to the host interrupt circuitry individually, thus using three unique interrupt level inputs, or they can be logically ORed together into a single interrupt and applied to one interrupt level input. In the latter case, the host may examine the SVRR register to determine which service requests are active. The method (single or multiple interrupts) chosen by the designer will be dependant on the system requirements and hardware and/or board space limitations; the CL-CD1400 places no restrictions on it. It is likely that interrupt latency will be slightly

shorter with the first method since the individual interrupt levels can cause a software vector directly to the correct service routine without first checking for the source of the interrupt.

No matter which interrupt method is used, the end result is the same. Once the host has recognized that a service request is active, a service acknowledge routine must be executed in order to satisfy the request. There are two ways in which to start the acknowledge and force the context switch: via four hardware input pins or by making specific modifications to internal registers.

4.3.1.1 Hardware-Activated Context Switch

The internal register manipulation that is involved in the context switch can be forced via the Service Acknowledge (SVCACK*) input pins on the CL-CD1400. There is one SVCACK* for each service request type: SVCACKR* for receive service requests, SVCACKT* for transmit service requests and SVCACKM* for modem signal change service requests. Each of these inputs is a special-case chip select that causes the MPU to set up the CL-CD1400 for servicing that particular service request type for the requesting channel. Note that the CS* input is not activated on service acknowledge cycles. Instead, the appropriate SVCACK* input and the DGRANT* inputs are used. DGRANT* will be discussed in the description of daisy-chaining multiple CL-CD1400s below. Figure 4–3 shows a generalized logic diagram of the hardware interface to the SVCACK* inputs. In the case of a service acknowledge, one of the SVCACK* address locations will be accessed instead of the CS* location.

To the host, the service acknowledge cycle is a read cycle. The data that the CL-CD1400 places on the bus during the read cycle is the contents of the interrupt vector register (RIVR, TIVR or MIVR) associated with the service acknowledge input that is active (SVCACKR*, SVCACKT* or SVCACKM*). The upper five bits of the vector register are whatever was previously loaded into the LIVR by the host; the lower three bits will be supplied by the CL-CD1400, indicating the type of interrupt (vector).

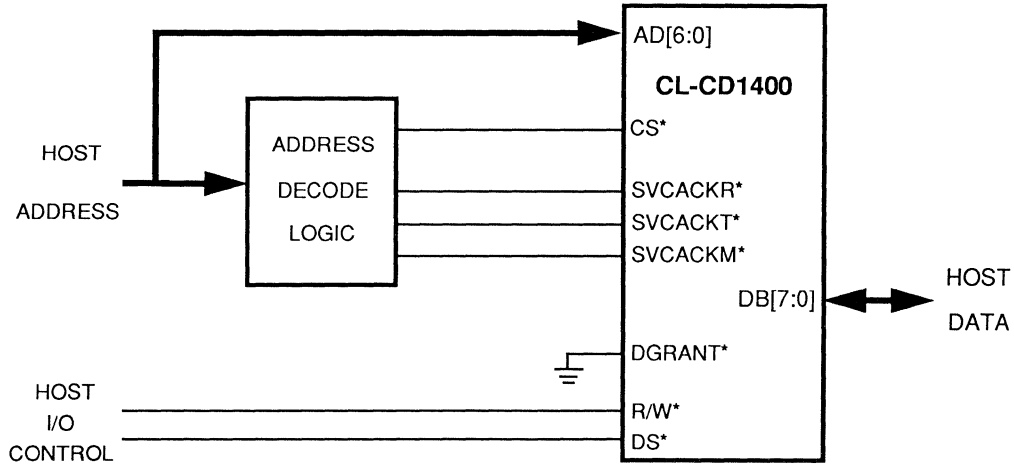


Figure 4–3. Control Signal Generation

At the time the CL-CD1400 is ready to post the service request, it copies the upper five bits of the LIVR into the appropriate vector register (RIVR, TIVR, MIVR) and then places the request type vector in the lower three bits. The following table shows the assignment of the request type bits.

Bit 2	Bit 1	Bit 0	Request Type
0	0	0	Not used
0	0	1	Group 1: Modem signal change service request
0	1	0	Group 2: Transmit data service request
0	1	1	Group 3: Received good data service request
1	0	0	Not used
1	0	1	Not used
1	1	0	Not used
1	1	1	Group 3: Received exception data service request

For transmit and modem service acknowledge cycles, the data in the lower three bits will be redundant to the host, since this information is known by the fact that the corresponding acknowledge has taken place. However, these bits will be of importance in the case of a receive data service acknowledge because they provide an indication of whether the request is for "good" data or exception data.

The value contained in the upper five bits of the LIVR can be used for a number of purposes. The primary purpose of the LIVR is as a source of a software vector that can be used by the host system as an index into a interrupt dispatch table. However, systems that can't use this or don't need it can use these bits for any purpose. In multiple-CL-CD1400 designs that use daisy-chaining, a logical value to place in these bits is a chip identification number. This will be discussed in more detail in the daisy-chaining description below. In a single-CL-CD1400 design or one that does not use daisy-chaining (unique address range for each device) and does not need the value in the LIVR as a vector for hardware interrupt response, a convenient use for these bits is channel encoding. Since each channel has its own LIVR, these five bits can have a unique value identifying the channel. By doing this, there is no need to read the RICR, TICR or MICR to determine the channel number, thus in a single I/O operation, the host knows both the type of interrupt and the number of the channel requesting service. In fact, with five bits available, systems with small numbers of CL-CD1400s can encode both the channel number and chip identification number in the LIVR.

Once all of the above has been completed, the CL-CD1400 is ready to be serviced for the type of interrupt that has been acknowledged. For example, if the interrupt was for receive good data, the host would read the RDCR register to determine the number of characters available in the receive FIFO, then read that many characters by successive reads from the RDSR. Other work, such as disabling future interrupts or changing channel parameters could also be performed at this time. Once all tasks involved in servicing the interrupt have been completed, one further operation *must* be performed. In order to inform the CL-CD1400 that the service acknowledge is complete, the host must write a dummy value to the EOSRR register. The data written does not matter; any value will do. What is important is the write operation itself. This

write forces the internal context switch back to normal operating mode.

Summary of Interrupt Driven Service Requests

In summary, the actions that take place during an interrupt request/service are:

1. Host senses service request via its interrupt request input from one of the CL-CD1400 service request outputs.
2. Host responds by performing a read cycle that activates the appropriate SVCACK* input pin.
3. Host decodes the value read from the vector register during step 2, making a decision on the type of service request (if necessary).
4. Host reads {R, T, M} ICR to determine channel number.
5. Host services the request (load transmit FIFO, read receive FIFO, etc.)
6. Host writes a dummy value to the EOSRR to terminate the service routine.

4.3.1.2 Software Activated Context Switch

It is possible, via host manipulation of some internal registers, to cause the context switch without activating any of the SVCACK* hardware inputs. The method used is the same as that which is used in a poll-mode-CL-CD1400 design. Once the host has detected the service request via its interrupt response circuitry, it can then follow the same procedures that a polling method would use once it had detected an active service request. Refer to the context switching description in the following section.

One reason a design might make use of this method is that there is limited board space available to provide the additional hardware address decoding required to generate the three SVCACK* and DGRANT* control signals. The system gains the advantage of not having to constantly check for active service requests by polling the CL-CD1400; it will be interrupted when a request is posted and can then examine internal CL-CD1400 registers to determine the source and channel number generating that request. If this method is chosen, the three SVCACK* and DGRANT* input pins should be tied inactive (logic '1') to prevent false activation of a service acknowledge cycle due to noise.

4.3.2 Polling

In poll mode, the hosts periodically checks the CL-CD1400 to see if there are any active service requests. If it detects any, it proceeds to service them via a software driven technique. There are several registers within the CL-CD1400 provided specifically to facilitate poll mode service request detection and acknowledgment. These are the SVRR, RIR, TIR, MIR, RIVR, TIVR and MIVR. Section 5 provides detailed bit definitions for these registers.

The SVRR (Service Request Register) is the master service request register. The least significant three bits (bits 2-0, SRM, SRT, and SRR) reflect the inverse of the state of the three service request output pins (SVCREQR*, SVCREQT* and SVCREQM*). For example, if bit 0 (SRR) is a "one", it indicates there is an active receive data service request, and that the SVCREQR* output pin is active (low). Thus, with a single read, the host can determine if the CL-CD1400 needs any service and, if so, which ones are active.

Each service request type has an interrupt request register; RIR for receive, TIR for transmit and MIR for modem. The RIR, TIR and MIR registers are special purpose registers that are used with the CAR to force the context switch and start a service acknowledge procedure. When a service request of a particular type is pending, the corresponding interrupt request register is set by the MPU with the appropriate data to cause the context switch to the requested type and the requesting channel. When the host is ready to service the request, it reads the contents of the

request register and copies it into the CAR. The action of writing this value into the CAR forces the context switch, and the CL-CD1400 is ready to be serviced. This is the same result as if a service acknowledge cycle had been performed with the SVCACK* pin. Each of the interrupt request registers provides the channel number that is requesting service in the least significant two bits. The most significant three bits provide status and control over internal interrupt sequencing. The middle three bits contain a code that is used by the MPU at the end of a hardware service acknowledge cycles (write to the EOSRR) to tell it which type of acknowledge cycle is ending. Each of the three registers has a unique code in these three bits that select the proper service acknowledge type.

At the end of a service request operation, the host must inform the CL-CD1400 that the request has been satisfied and take it out of the service request context. This is done by writing the value that was in the interrupt request register back into it after first clearing the upper two bits.

As with the hardware-driven request/acknowledge procedure, the virtual registers should only be accessed after the context switch has been made. Their contents are undefined until this time.

Summary of Poll Mode Service Requests

To summarize, the major steps involved in a poll mode service request/service acknowledge sequence are:

1. Host scans the SVRR periodically, checking the three least significant bits. If any of them are true ("1"), a service request is active.
2. Depending on which of the service request bits is active, read the appropriate interrupt request register (RIR, TIR or MIR) and copy the contents into the CAR.
3. Perform service routine.
4. Write the original value of the interrupt request register back after clearing the upper two bits.

4.3.3 Service Requests and Multiple CL-CD1400s

Multiple CL-CD1400s can be combined to form systems with more than four channels. There are a number of ways that two or more can be connected, but one way provides a more efficient service request/service acknowledge sequence by allowing the CL-CD1400s to arbitrate between themselves. This mode only works if hardware activated service acknowledges are being utilized.

The CL-CD1400 provides a means of "daisy-chaining" the service request and service acknowledgments of two or more devices together. This allows them to arbitrate and set priorities between themselves regarding which may post a particular type of service request. This is the Fair Share interrupt scheme. The Figure 4–4 shows the way in which two CL-CD1400s would be connected to enable the Fair Share function.

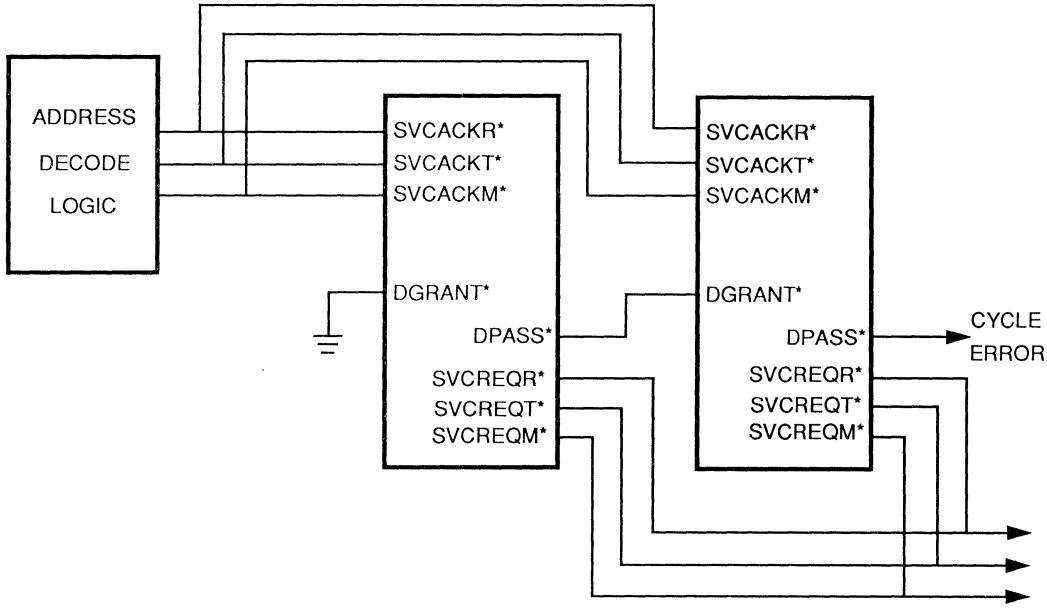


Figure 4–4. CL-CD1400 Daisy-Chain Connections

The open-drain request outputs of the two CL-CD1400s (SVCREQR*, SVCREQT* and SVCREQM*) are wire ORed together to form one request for each type. This allows each to monitor the state of the others' outputs. Also, each of the service acknowledge inputs (SVCACKR*, SVCACKT* and SVCACKM*) is also connected together to form one acknowledge of each type. The DGRANT* input of the first CL-CD1400 is connected to ground; the DPASS* output of the first CL-CD1400 drives the DGRANT* input of the second.

Before a request for service of a particular type is posted, the MPU checks the current state of the request output for that type. If it is inactive, indicating that no other CL-CD1400 is driving that level, a request can be posted, otherwise it will wait. This guarantees that each CL-CD1400 will have an opportunity to have this request type serviced when needed. When the host acknowledges the request, both CL-CD1400s will receive the acknowledge via the SVCACK* input. However, only the first will receive the DGRANT*. If it has an active request of this type pending, it will take the acknowledge and drive its vector register (RIVR, TIVR, MIVR) onto the data bus.

If it does not have a request pending, it will pass the DGRANT* input to the second CL-CD1400

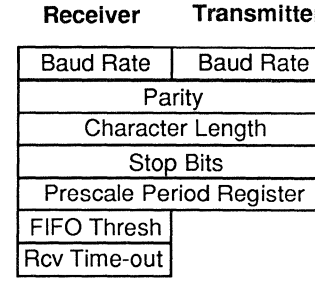
via the DPASS* output. Assuming that the second has an active request pending, it will take the acknowledge and drive its vector register onto the data bus.

As mentioned earlier, the upper five bits of the LIVR will reflect whatever the host loaded into them during its initialization of the CL-CD1400s. These bits must be used as a unique chip identification number so the host will know which CL-CD1400 responded to the service acknowledge. These five bits could be set to binary zero in the LIVR of the first CL-CD1400, and to binary 1 in the second. The host can easily test the bit to determine which device responded. Some examples of host service acknowledge software routines that show one way of performing this task are provided in Section 6.

CAUTION: If neither CL-CD1400 has a pending request, the DGRANT* will be passed by the second and neither will respond, thus causing the cycle to hang. The only time this could happen would be due to an error condition outside the CL-CD1400s that caused the host to respond to a request that was not made. A mechanism should be provided to terminate or abort the cycle if this error should occur. This can be accomplished with time-out circuitry or the DPASS* output of the second CL-CD1400 can activate an abort condition. Other devices may share the daisy-chain mechanism and could be connected to the DPASS* output of the second (or whichever is last) CL-CD1400 in the chain. The actual implementation is system-dependent, but it is important to provide some way for the host to know that the cycle did not complete normally, if no device exists at the end of the chain.

4.4 Serial Data Reception and Transmission

The CL-CD1400 has four channels, each with a receiver and a transmitter. Although a receiver and a transmitter pair are associated with each channel, in many respects they operate independently, sharing only parameter settings regarding character format, including length, parity type, if any, and number of stop bits. Each receiver and transmitter has its own baud rate generation function, allowing a channel to send at one rate and receive at another. Shared and independent parameters are shown in the diagram below:



Channel service needs, such as an empty transmit FIFO, are indicated to the host by one of three service request indicators: one for all receivers, one for all transmitters and one for all modem signal changes. The internal processor (MPU) scans each channel sequentially for service needs, posting a request when it detects a particular type. It continues the Fair Share scheme used in the external daisy-chain configuration by not allowing a channel to post another request of one type until all other channels have posted their requests of that type, if any. For example, if channel 0 is currently being serviced for a transmit request and channel 3 has one pending, the request from channel 3 will be posted before channel 0 is able to make another request for transmit service.

Each receiver and transmitter has a 12-character FIFO. The receiver has two additional character holding locations, the receive character holding register and the receiver shift register. The transmitter also has two additional locations, the transmitter holding register and transmitter shift register. The receive FIFO has a programmable threshold that sets the level at which a service request will be posted. When data reaches this 'high water' mark, a request will be made of the host to empty the FIFO. More details on this are provided later in the section describing receiver operation. Receive FIFOs also have a programmable threshold that, when reached, will

cause the DTR output to be deasserted (see the flow-control description).

In the asynchronous serial data protocol, a message consists of one 'character', made up of bits, either high or low, representing a one or zero value. A character can be made up of from five to eight bits plus an optional parity bit bracketed by a start bit and a stop bit. Each bit has a time duration that sets the data transmission rate, or baud rate. The start bit indicates the beginning of a character bit stream and is indicated by a transition from a logic '1' to a logic '0' (mark to space) on the transmission media. The start bit lasts one 'bit-time' and is immediately followed by the data bits (5 to 8), the parity, if any, and the stop bit.

As discussed previously, the CL-CD1400 incorporates special hardware to receive and transmit each bit. These are the 'bit engines'. They perform all timing associated with sending or receiving one serial data bit. A bit-engine behaves differently depending on whether it is sending or receiving. When a complete bit has been received, the bit-engine interrupts the MPU so that it can handle the bit on the character level. This usually entails its addition to the character being assembled. For transmitting, a transmit bit-engine interrupt causes the MPU to give it the next bit to be transmitted. The bit-engine interrupt happens at the end of a bit time, which has been timed by the engine, thus removing that duty from the MPU.

4.4.1 Receiver Operation

Each channel can be programmed to receive characters with several different parameters, such as character length, parity, number of stop bits, FIFO threshold and baud rate. Each receiver is independent of any other receiver. It may also be set to a different baud rate from its corresponding transmitter.

Before valid data can be received, the host must set up each channel by programming the desired operational parameters in the Channel Option Registers (COR1-COR5) and the baud rate generator registers (Receiver Clock Option Register and the Receiver Baud Rate Period Register - RCOR and RBPR). Once these are set, the channel is enabled by issuing the receiver enable command via the CCR and enabling service requests in the Service Request Enable Register (SRER).

Once a receiver is enabled, its bit-engine begins scanning the RxD input for a valid start bit. It does this by detecting a falling edge transition on the input. When the transition is detected, the bit-engine delays until the middle of the programmed bit time and checks the input again. If the input is still low, then the start bit is considered valid and character assembly begins. At each subsequent full bit time, the input is checked and its level recorded as the value of the next bit. If, at the center of the bit-time, the RxD input has returned to a mark state, then the start bit is considered invalid and the bit-engine goes back to the start bit detect mode. Following a valid start bit, the bit-engine begins receiving data bits. At the end of the programmed number of bits, following bits are checked for parity (if enabled) and a valid stop bit. A valid stop bit is defined as a mark or logic '1' on the input. If a valid stop bit is not detected, a framing error will be noted for the character. After a properly assembled (no framing error) character has been received, it is checked for several special conditions (see the section on special character handling and flow control), and the overrun condition before it is placed in the receive FIFO. If no errors or special character processing are required, the character is considered 'good' data and placed directly in the FIFO. If errors exist, it is placed in the FIFO as 'exception' data along with status indicating the type of error. As each good character is placed in the FIFO, the Receive Data Count Register (RDCR) is updated to reflect the number of good characters currently in the FIFO.

The receive FIFO has a programmable threshold that determines the level at which the CL-CD1400 will request receive data service. This level is programmed via the RxTh3-RxTh0 bits in Channel Option Register 3. The host may place the threshold at any number of characters from 1 to 12. Note that this only sets the level at which the CL-CD1400 will post a service request, not the

depth of the FIFO. When the host responds to a receive good data service request, it may read any number of characters out of the FIFO, from zero up to the number indicated in the RDCR before exiting the service routine. If the number read is zero, the CL-CD1400 will post another request for service almost immediately. If the number of characters read is less than the number indicated by the RDCR but enough such that the number in the FIFO falls below the threshold, a new request will not be made until the threshold is once again exceeded. The term 'almost immediately' is used above because, since the MPU scans the channels in a 'round-robin' fashion, another channel may post a receive service request before this channel again has the opportunity.

4.4.2 Receiver Timer Operations

Also associated with each receiver FIFO is a timer whose duration is set by the Receive Time-out Period Register (RTPR). This timer provides two services in relation to receive FIFO operation: a time-out to prevent 'stale' data in the FIFO and a time-out after the last character is taken out of the FIFO. The first type, type 1, will occur if the receive FIFO does not reach the set threshold before the programmed time period expires and the second, type 2, will occur if the timer expires and no new data has been placed in the FIFO after the last character was removed; this is called the No New Data Time-out (NNDT) service request.

The timer is driven by the prescaled clock generated by the Prescale Period Register (PPR) in the global register set. The timer is loaded with the value contained in the RTPR each time a character is placed in the receive FIFO or when the last character is removed from the FIFO. Each 'tick' of the prescaler decrements the timer. If the timer reaches zero and receiver interrupts are enabled, the MPU will generate a receive data service request for one of the two time-out conditions, depending on which is valid.

Type 1: If there are characters in the FIFO but the threshold level has not been reached, a good data service request will be posted when the timer expires. This function is provided to prevent data from remaining in the FIFO for long (potentially infinite) periods of time because the remote did not send enough data to fill the FIFO to the threshold level. This time-out cannot be disabled.

Type 2: If there is no data in the FIFO when the timer expires and the No New Data Time-out (NNDT) service request is enabled in the SRER, a receive exception service request will be posted with status indicating the time-out condition. This time-out is optional, and is provided so that host driver software can detect the possible end of a block of data and allows its buffers to be flushed to the higher, operating system level. The NNDT will be posted only on the first occurrence of a time-out after the FIFO becomes empty. Also note that the NNDT timer is *not* started if the last character removed from the FIFO was an exception character.

The flow chart in Figure 4–5 shows the timer process evaluation performed by the MPU when the timer reaches zero.

4.4.3 Receive Exceptions

Several conditions can cause the CL-CD1400 to evoke the receive exception service request. If an exception condition occurs, two bytes are placed in the receive FIFO. The first is the status indicating the type of error and the second is the data itself.

Exception data is given to the host one event at a time. That is, there will be a separate service request for each character that is received with some special condition. If, when an exception

condition occurs, the receive FIFO has good data in it, a good data receive service request will be posted immediately upon receipt of the bad data, regardless of the number of characters in the FIFO and the programmed threshold. This allows the host to remove the data in the FIFO ahead of the exception data so that the CL-CD1400 can post the service request for the error condition. Once the host terminates the service acknowledge procedure for the good data, a new service request will be posted for the exception data.

When the host acknowledges the receive exception service request, it reads the Receive Data/ Status Register (RDSR) first to get the status and second to get the data. Reading the data is optional: if the host does not read the FIFO twice during the service routine, the CL-CD1400 will update its internal FIFO pointers appropriately and discard the second byte. (Actually, the host need not read *any* data from the FIFO during an exception service acknowledge – the FIFO pointers will be updated correctly at the end of the service routine, discarding both the status and the data. Thus, the host must read at least the status, or it will be lost forever.)

Another special case of the exception data handling is for received line break conditions. A line break is a character with zero data and no parity or stop bit. In this case, a null (zero) character is placed in the FIFO with the break condition indicated in the accompanying status and a receive exception service request will be posted. However, regardless of the length of the break, only one character will be placed in the FIFO. Resumption of normal character reception will cause new data to again be placed in the FIFO.

Refer to the register definitions in Section 5 for a description of the status bits in the RDSR.

4.4.4 Transmitter Operation

Each of the four channels on the CL-CD1400 are capable of transmitting characters with a number of programmable characteristics such as length, parity and baud rate. The channels operate completely independently and settings in one will have no effect on the operation of another.

After being reset, from either hardware (RESET* input pin) or software (via the master reset command in the CCR), all transmitters are disabled with the TxD output held at a logic '1' condition. This is the off, or mark, condition of the asynchronous protocol. Before any operation of the transmitter can begin, the host must program the appropriate parameters in the Channel Option Registers (COR), the Clock Option register (TCOR) and Transmit Baud Rate Period register (TBPR). Once these registers are set, the channel is enabled by issuing a transmit enable command via the CCR and enabling service requests by setting the appropriate transmit enable request bit(s) in the Service Request Enable register (SRER). The channel will immediately post a transmit service request since its FIFO is empty. The host responds to the request by loading up to 12 characters into the transmit FIFO via the Transmit Data Register (TDR) after it places the CL-CD1400 in the service request acknowledge mode (see description of service request/service acknowledge procedures in Section 4.3). The transmitter does not begin transmitting the characters until the host terminates the service routine and writes the EOSRR. Transmission begins by sending a start bit (logic '0') followed by five to eight data bits (depending on the programmed value), least significant bit first. The last data bit is followed by the appropriate parity bit, if enabled, and a minimum of one stop bit. All bit transmission is handled by the transmit bit-engine with the MPU giving it each bit as it is required. If there are still characters in the FIFO, the next one will be transmitted immediately after the last stop bit of the previous character. This process continues until all characters in the FIFO have been transmitted. The CL-CD1400 will then post a service request for more data.

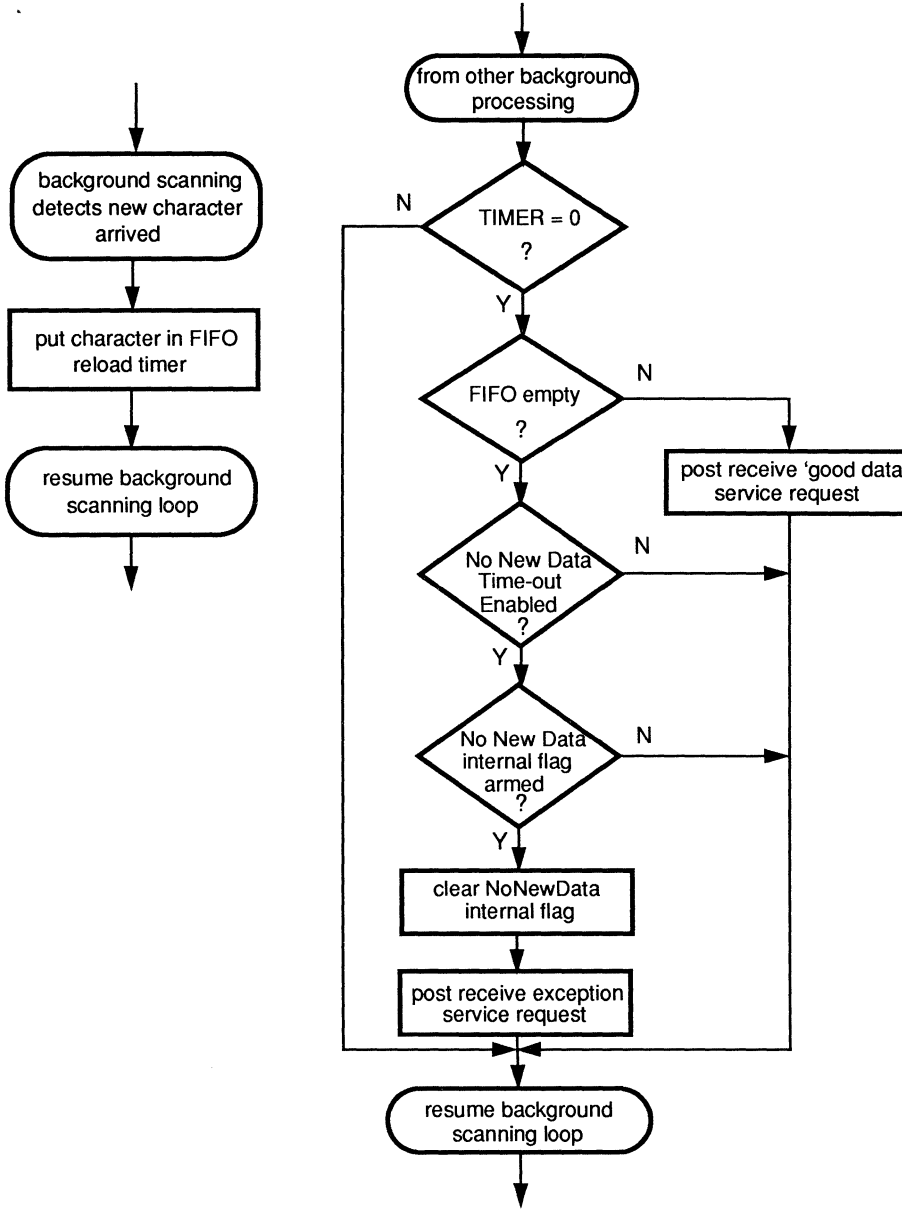


Figure 4–5. FIFO Timer Processing

There are actually 14 transmit character holding locations for each channel: 12 in the FIFO, one in the transmitter holding register and one in the transmitter shift register itself. The CL-CD1400 can be programmed, on a per channel basis, to request transmit data when one of two conditions exist: when the last character in the FIFO is transferred to the holding register or when the last data bit of the last character is shifted out of the transmitter shift register. The first option allows the host two character transmit times in which to reload the FIFO and prevent a transmit data underrun. This is the normal mode of operation. The second mode can be used to make sure the transmitter is empty before reconfiguring the channel. It is likely that the transmitter will underrun if the second option is chosen unless the host is sufficiently fast enough to respond to a transmit service request and reload the FIFO during the transmission of the stop bit(s) of the last character. If the transmitter underruns, it will continue to send stop bits (mark) until more data is placed in the FIFO. Normally, when a string of characters greater than 12 is being transmitted, host software will program the CL-CD1400 transmitter to post a service request when the FIFO becomes empty. When the last of the data to send has been placed in the FIFO, the service request enable is changed so that requests are made after the last character is sent. This allows the host to know when all the data is transmitted before disabling a channel. If a channel is disabled, any characters other than the one currently being transmitted will be held and the transmitter will enter the marking state. If the channel is subsequently re-enabled, any remaining data will be transmitted.

The transmitter is capable of performing several special functions such as break generation, inter-character delays and automatic flow control. These functions are discussed in the sections describing special character handling (embedded transmit commands) and flow control.

4.4.4.1 Transmitter Timer Operations

As with the receiver, the transmitter has a timer associated with it. This timer is used to generate the timing for the embedded transmit commands that send line breaks and inter-character delays. Whenever the MPU detects an embedded transmit command specifying the delay command, it loads the timer with the value contained in the parameter byte. This timer is decremented on each 'tick' of the prescaler timer until it reaches zero. At that time, the delay is terminated unless the next character in the FIFO is the beginning of another delay command sequence.

4.5 Flow Control

In all data communications applications, data is sent from one system to another via some protocol. Most systems have some method of buffering data for transmission and reception. In the asynchronous protocol, there is no way, at the protocol level, to determine the length of a data transmission therefore it is normally not possible to set aside a buffer area that is known to handle the entire length of the transmission. Also, the hardware receiving the data generally has a limited amount of buffer area, usually a FIFO, and if the host does not unload data at a fast enough pace, the buffer or FIFO may overflow. For these reasons, two methods are provided that can be used to stop the remote from sending data until room is once again available to receive data. This is known as flow control. Flow control can be in-band or out-of-band. In-band flow control makes use of special characters that can be sent to the host to stop data transmission. Out-of-band flow control are signals outside of the serial data channel that perform the same function. These are the Request To Send (RTS), Clear To Send (CTS) pair and the Data Set Ready (DSR) and Data Terminal Ready (DTR) pair. The CL-CD1400 can make use of either kind and has built in capabilities to do so automatically and/or semi-automatically (depending on direction and options chosen) without host intervention (or knowledge, if desired).

4.5.1 In-Band Flow Control

As mentioned, in-band flow control is implemented by special characters that are imbedded in the serial data stream, one to request that transmission stop and one to request resumption. The characters chosen can be any characters although conventionally the XON or DC1 (x'11) and XOFF or DC3 (x'13) characters if the ASCII character set is being used. The XOFF value designates the character that is to be used to stop data transmission and the XON character determines the character that is to be used to resume transmission. Whether or not the ASCII XON and XOFF characters are used, the CL-CD1400 allows the two characters to be set to any value that is appropriate to the system design by the value programmed in Special Character Registers 1 and 2 (SCHR1 and SCHR2). SCHR1 defines the XON character and SCHR2 defines the XOFF character.

4.5.1.1 Receiver In-Band Flow Control

When the host senses a need to flow control a sender, due to its receive buffer filling too fast to service, it can request that remote stop transmission by sending an XOFF character via the transmitter. This is accomplished by issuing a send special character 2 command via the Channel Command Register (CCR). The CL-CD1400 will then transmit whatever character is programmed in SCHR2. As discussed earlier, the send special character command is preemptive to data currently in the transmit FIFO, and thus the XOFF character will be transmitted after the currently transmitting character and the character in the transmitter holding register have been sent, a maximum delay of two character times. When the host is again ready to start receiving characters, it sends an XON character, also via a send special character command. This time, the CL-CD1400 is issued the command to send whatever is programmed in SCHR1. Send special character commands will override the remotes' flow-controlling of the CL-CD1400; in other words, even if the CL-CD1400 transmitter has been shut off by the remote, it can still send flow control characters.

The current state of the flow control condition is always made available to the host via the Channel Control Status Register (CCSR). In addition to the enabled/disabled status of the receiver and transmitter, the CCSR displays the flow control status. Two bits in the CCSR pertain to receiver flow control, RxFloff and RxFlon. Whenever the host issues the send special character 2 (send XOFF), the CL-CD1400 sets the RxFloff bit, indicating that it has requested the remote to stop transmission. When the host issues the send special character 1 (send XON) command, the RxFlon bit is set and RxFloff is reset. RxFlon remains set until the first character is received after the XON was transmitted. The table below shows the bit encoding for RxFloff and RxFlon.

RxFloff	RxFlon	Encoded Status
0	0	Transmission has resumed, receiver has been enabled/disabled or receiver is in default reset state
0	1	XON has been sent, transmission not yet restarted
1	0	XOFF has been sent
1	1	Not Used

The RxFloff/RxFlon bits are cleared whenever the receiver is disabled or enabled, regardless of the state of flow control when the disable/enable occurred.

NOTE: Regardless of the current state of RxFloff, the CL-CD1400 continues to receive characters. If the remote ignores or is slow to respond to the XOFF character, there is the possibility of overruns.

4.5.1.2 Transmitter In-Band Flow Control

The CL-CD1400 has the ability to automatically flow control its own transmitter when it receives the XON and XOFF characters, as programmed in SCHR1 and SCHR2. Control bits in Channel Option Registers 2 and 3 (COR2 and COR3) enable or disable various aspects of the automatic flow control.

In order for flow control characters to be acted upon, special character detection must be enabled via bit 4 (Special Character Detect 1 & 2 - SCD12 of COR3). When this bit is set, the CL-CD1400 will scan received characters for a match with one of the special characters programmed in SCHR1-SCHR2. If enabled via SCD12, and it has received a character matching the contents of SCHR2 (the XOFF character), the CL-CD1400 will then check to see if automatic transmit in-band flow control is enabled via bit 6 of COR2. If this function is enabled, the CL-CD1400 will cease transmission after the currently transmitting character and the character in the transmitter holding register, if any. If enabled, the CL-CD1400 will also attempt to match against errored characters. This function is enabled via the CMOE bit in COR5.

One other control bit in COR2 is involved in flow control activities. This is bit 7, the Implied XON mode, IXM. This bit determines what character will restart transmission after an automatic flow control has caused it to stop. If bit 7 is a zero, only a programmed XON character (SCHR1) will restart the transmitter; all other characters will be received and placed in the FIFO normally. If IXM is set, any character received will restart data transmission.

As with receiver flow control, the host can always determine the current state of the transmitter via two bits in the CCSR: TXFloff and TxFlon. When automatic in-band flow control is enabled and the CL-CD1400 receives and XOFF character, it sets TxFloff. When an XON character is received, TxFlon is set. Once transmission actually resumes, TxFlon is cleared. The encoding is shown in the table below.

TxFloff	TxFlon	Encoded Status
0	0	Transmission has resumed, transmitter has been enabled/disabled or transmitter is in default reset state
0	1	XON has been received, transmission not yet restarted
1	0	XOFF has been received, transmission has stopped
1	1	Not Used

The TxFloff/TxFlon bits are cleared whenever the transmitter is disabled or enabled, regardless of the state of flow control when the disable/enable occurred. This feature can be used to force resumption of transmission regardless of remote initiated flow control.

There is one final aspect of automatic in-band flow control: Flow Control Transparency (FCT) which is enabled/disabled by bit 5 in COR3. FCT determines whether or not remote flow control will be transparent to the host. If this bit is not set, in addition to stopping transmission when an XOFF is received, the CL-CD1400 will place the received XOFF character in the receive FIFO and inform the host of the reception via a receive exception service request. When the XON character is received, it too will be given to the host via an exception service request as well as restarting data transmission. If FCT is enabled, received flow control characters will control transmission but they will be discarded rather than be placed in the FIFO. If the host system software does not need to know when its transmit data has been stopped, this bit can be set to reduce the number of service requests that must be handled.

The following table summarizes the control bits in the Channel Option Registers that enable the various modes of in-band flow control.

Bit Name	Register	Function
SCD12	COR3	Enables recognition of special characters 1 and 2
FCT	COR3	Enables transparent flow control
TxIBE	COR2	Enables automatic transmitter in-band flow control
IXM	COR2	Enables implied XON mode

4.5.2 Out-of-Band Flow Control

Flow control can also be accomplished via the modem handshake signal pairs RTS/CTS and DSR/DTR. These are called out-of-band flow control because they are external to the data channel. The CL-CD1400 can be programmed to automatically respond to and generate out-of-band flow control via these signals.

4.5.2.1 Receiver Out-of-Band Flow Control

Along with the receiver FIFO threshold that sets the level at which the CL-CD1400 will post a service request, another threshold can be set that determines when it will automatically assert/deassert the DTR* output if so enabled. This is the DTR threshold and is enabled via the DTRth3-DTRth0 bits in Modem Change Option Register 1 (MCOR1). The level can be set for any number of characters from 0 to 12, with a threshold of 0 disabling the function. If the function and the receiver are enabled, the CL-CD1400 will automatically assert the DTR* output whenever the number of characters in the receive FIFO is less than the programmed number. Once the level reaches the threshold, DTR* will be deasserted. DTR* will be held in the deasserted state until the host removes enough characters from the FIFO to lower the level below the threshold.

In order for the receiver to operate properly, the DTR threshold must be set to a value equal to or higher than the receiver service request threshold. If the levels were reversed, normal character reception could not be completed because DTR* would always be deasserted before the receive FIFO threshold is reached, thus the host would not get a receive data service request until the receive FIFO time-out is reached. A serial data transmission performance limitation would result.

The DTR* output may also be controlled manually via bit 1 of Modem Signal Value Register 2 (MSVR2). Setting this bit to a '1' will assert the DTR* output.

4.5.2.2 Transmitter Out-of-Band Flow Control

Transmitter out-of-band flow control is implemented with three modem control signals: the RTS* output and the CTS* and DSR* inputs. The RTS* output can be programmed to automatically be asserted whenever there is data in the transmit FIFO and the transmitter is cleared to send. CTS* and DSR* can be enabled to automatically control the transmitter.

RTS Automatic Output (RtsAO) is enabled via bit 2 in COR2. If this bit is set, the CL-CD1400 will automatically assert the RTS* output when there is data in the FIFO to send. When the data has been sent and the FIFO is empty, RTS* will be deasserted until the host places more data in. If RtsAO is not set, the host software must control the RTS* output manually via Modem Signal Value Register 1 (MSVR1) if required by the remote.

The CTS* and DSR* inputs can also be monitored by the CL-CD1400 and used as a transmitter enable. The functions are enabled by setting bit 0 (DSR Automatic Enable - DsrAE) and/or bit 1 (CTS Automatic Enable - CtsAE) of COR2. These two functions operate independently but their control over the transmitter is the same. If the function is enabled, character transmission will occur only when the corresponding input signal is asserted. If the signal is deasserted during active transmission, the current character plus the character in the transmitter holding register, if any, will be transmitted and then transmission will cease. Thus, a minimum of one and a maximum of two characters may be transmitted after the control signal is deasserted. Transmission will resume when the signal(s) are reasserted.

The send special character command does not, however, sample the CTS* or DSR* inputs. If the host chooses to send one of the special characters, the character will be transmitted regardless of the state of these inputs. In most cases, this is desirable so that the host can flow control a remote even if it is itself flow controlled. If the state of CTS* and DSR* are important, they should be tested via MSVR1 before the special character send command is issued.

4.6 Receive Special Character Processing

The CL-CD1400 has several means of sending special characters and ways in which it processes these characters when it receives them. Some special characters can have fixed definitions and some can be user defined. The flow-chart at the end of this section defines the processing that the CL-CD1400 performs for receive data. The chart may aid in understanding the special character handling process.

4.6.1 UNIX Character Processing

The CL-CD1400 incorporates special character processing that can be of particular benefit in systems designed to run the UNIX® operating system. The processing performs some of the functions normally handled by the "line discipline" part of a serial device driver program. The effect of this is higher overall performance in serial communication than would otherwise be obtained because the character manipulation takes place at the hardware level without host action. This processing includes carriage return (CR) and new line (NL) substitution, programmable response to errored characters (framing, parity and overrun errors), the LNext function and ISTRIP. Each of the types of processing is optional; any, all or none of them can be enabled/disabled via control bits in the Channel Option Registers two, four and five (see the detailed register descriptions for the format of COR2, COR4 and COR5). This section gives detailed descriptions of each of the functions.

If channel 0 is programmed as a parallel channel, only the transmit special character processing occurs, such as repeat space and carriage return and new line translation.

4.6.1.1 Line Terminating Characters

The CL-CD1400 can be programmed to perform automatic substitution of carriage return (CR) and new line (NL) characters on both received and transmitted data. Received character processing has five unique substitutions based on the value of three bits in COR4 - IGNCR, ICRNL and INLCR (some combinations cause identical actions):

000	Do nothing – function not enabled
001	Received NL changed to CR
010	Received CR changed to NL
011	Received CR change to NL and received NL changed to CR
100	Received CR discarded
101	Received CR discarded and received NL changed to CR
110	Received CR discarded
111	Received CR discarded and received NL changed to CR

4.6.1.2 Errored Character Processing

The CL-CD1400 provides a number of ways to handle characters that are received with errors (parity, framing and overrun errors). If none of the special processing functions are enabled, errored characters are delivered to the host via a receive exception service request. Alternatively, these characters can be handled in one of the following ways, as defined by the PE[2:0] bits of COR4:

- Parity errors can be ignored – the character is placed in the FIFO as good data and is given to the host as any other received good data.
- An errored character can be replaced with a NULL (x'00) character in the FIFO.
- An errored character can be replaced in the FIFO with the three byte string **x'FF - 00 - character**. If this mode is enabled and an actual good x'FF character is received, it is replaced in the FIFO by two x'FF characters.
- An errored character can be discarded.

Received breaks are handled a little differently from other errored characters. They can be processed, based on the settings of the IGNBRK and -BRKINT bits in COR4, as:

- Reported as an errored character via a received exception service request.
- Replaced with a good NULL (x'00) character in the FIFO.
- Discarded

4.6.1.3 LNext

This function provides a means of "escaping" or ignoring any special meaning of special characters and treat them as normal data. The escape character is defined by the value in the LNC register. If the CL-CD1400 receives this character, it will put it and the next character in the FIFO without further processing. This allows, for example, a flow-control character to be received without it actually causing flow-control activity. LNext can be enabled to operate even on characters that are received with errors (parity, framing, overrun), otherwise errored characters are handled normally and the next character is not escaped.

4.6.1.4 ISTRIP

ISTRIP is a simple function that, if enabled, resets the most significant bit (bit 7) of all received good characters. If the character has a parity or framing error, the ISTRIP function does nothing and the character is given to the host via a normal receive exception service request.

4.6.2 Non-UNIX Receive Special Character Processing

In addition to the UNIX special character processing, the CL-CD1400 provides other special character recognition capabilities. The CL-CD1400 has four registers that define special characters, SCHR1-SCHR4. Two of these, SCHR1 and SCHR2 are used in flow control activities and were discussed in the flow control section. SCHR3 and SCHR4 define two additional special characters that the CL-CD1400 can scan for in the receive data stream. Recognition of special characters 3 and 4 are enabled by the SCD34 bit of COR3 (bit 76). If either of these are received, it is cause for a special character detect (receive exception) service request. It should be noted that if automatic in-band flow control is not enabled, SCHR1 and SCHR2 can still be used as special characters. They will be detected and reported as receive exceptions, they just won't cause any flow control activities to be invoked.

Another special character function is the range detect function. If this mode is enabled (via the SCDRNG bit in COR3), the CL-CD1400 will compare all received characters against the values in Special Character Range Low (SCRL) and Special Character Range High (SCHL). If the character received falls between these two values (inclusive), a special character detect service request will be posted.

The status shown in the RDSR indicates which of the special character recognition conditions were met and that caused the receive exception service request. See the RDSR description in the register definitions in Section 5 for the bit encoding.

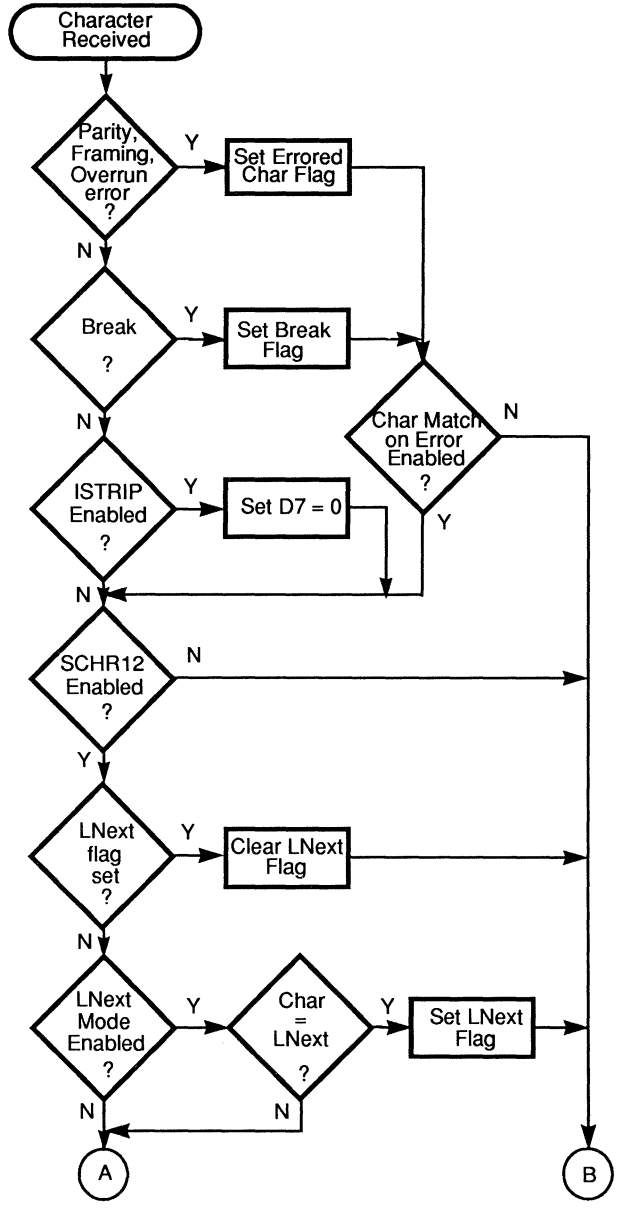


Figure 4–6. CL-CD1400 Receive Character Processing

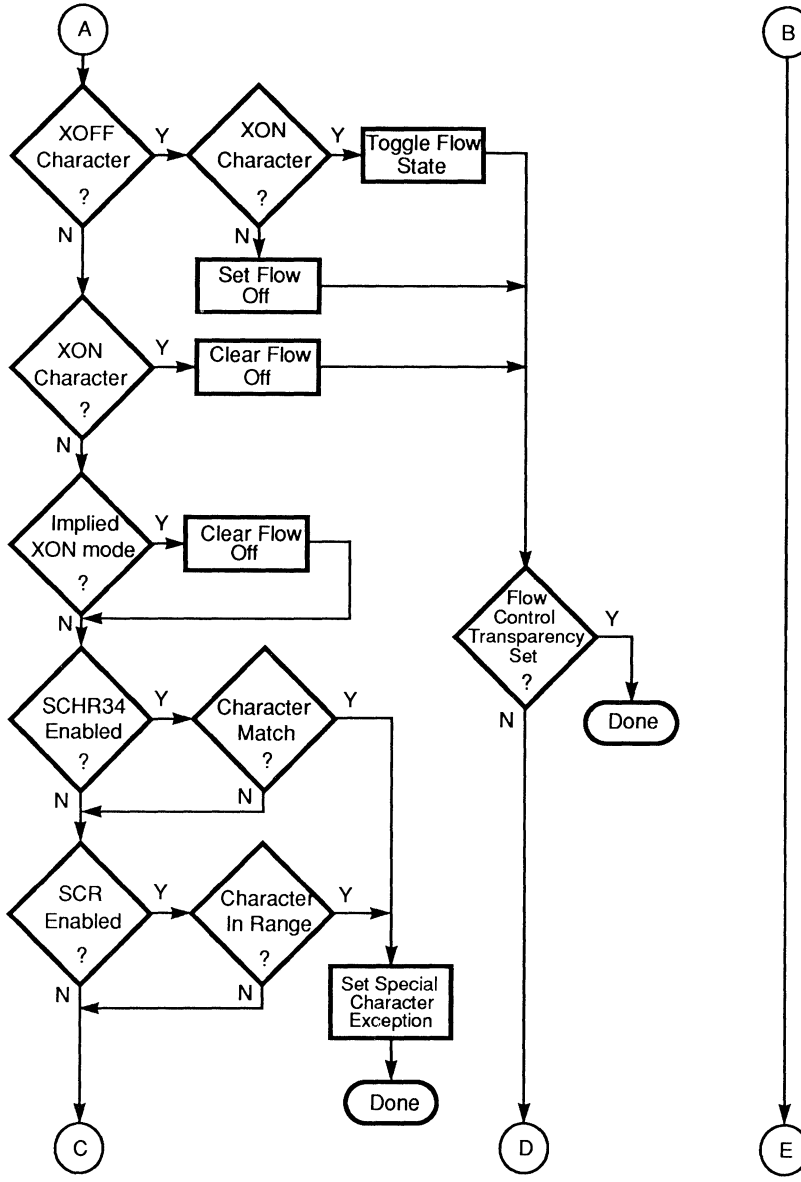


Figure 4–6. CL-CD1400 Receive Character Processing *(cont.)*

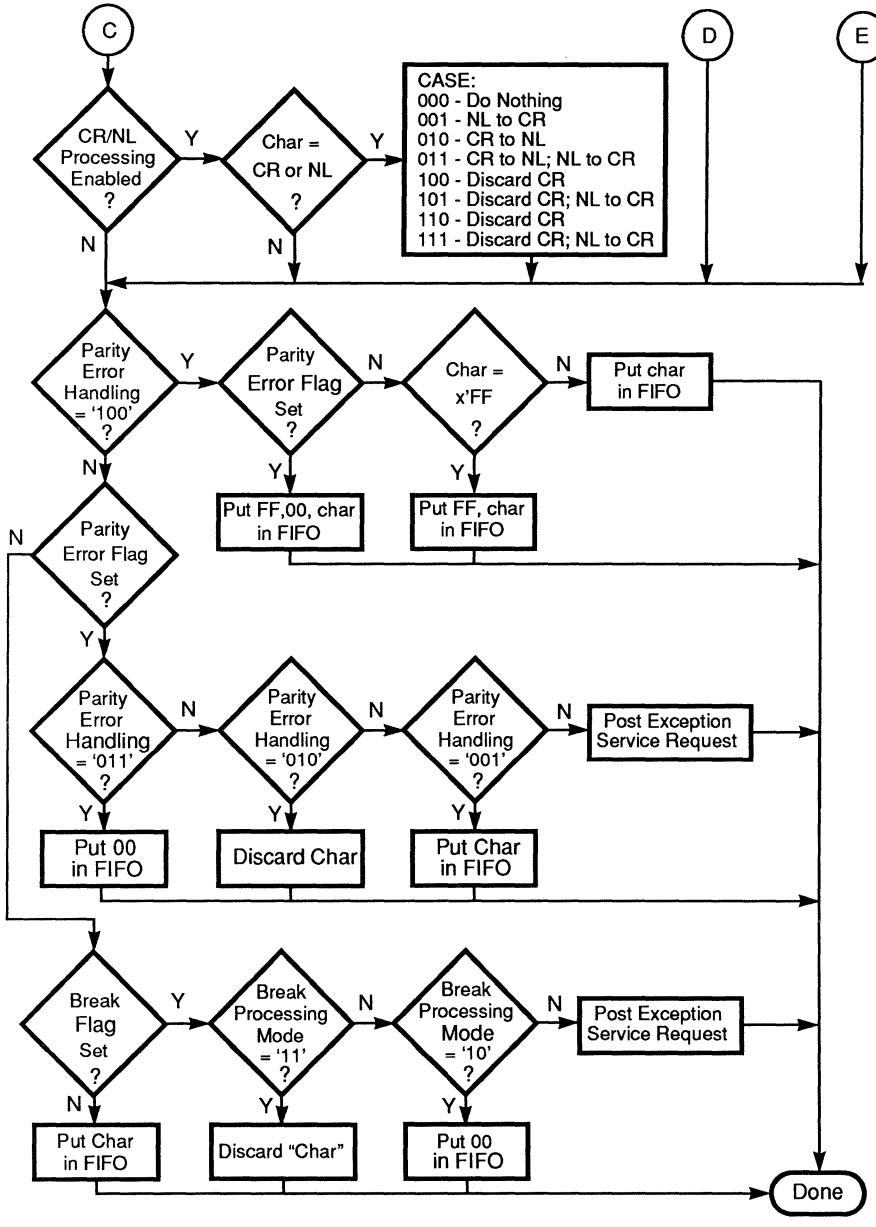


Figure 4–6. CL-CD1400 Receive Character Processing *(cont.)*

4.7 Transmit Special Character Processing

The CL-CD1400 also provides some special character handling on the transmit side; embedded transmit commands and direct commands that cause transmission of predefined special characters. A flow chart (Figure 4–7) is included at the end of this section to help describe the process of special character handling.

4.7.1 Line Terminating Characters

On transmit, there are four possible substitutions based on the setting of two flags in COR5 (bits 1 and 0 - ONLCR and OCRNL):

00 Do nothing - function not enabled

01 Change all <CR> characters to <NL>

10 Change all <NL> characters to <CR> <NL>

11 CR characters changed to NL or NL changed to <CR> <NL>

In the last case, where both flags are set, only one of the translations will take place. In other words, a CR that has been changed to NL will not then be changed into CRNL.

4.7.2 Embedded Transmit Commands

The CL-CD1400 has a special feature that optionally allows specific 'escape' character sequences in the transmit data stream to be interpreted as commands. These are called Embedded Transmit Commands (ETC) and are enabled by the ETC bit in Channel Option Register 2. They can be used to insert programmed time delays between characters and generate a line break on the transmit data output.

If enabled, an ETC is detected when the two- or three-character 'escape' sequence is detected in the transmit FIFO. An escape-character sequence is made up of the special escape character followed by the command character and an optional count for the delay period. The escape character is an all-zero (null) character. It should not be confused with the ASCII ESC character, which is 1B hex; for this discussion, ESC refers to the null character. Five commands are supported in the ETC command set:

ESC ESC: Send one **ESC** character. This command sequence is provided to allow the ESC character be sent alone. Thus, this "escapes" the escape when it is actually desired to send a null character.

ESC x'81: Send **BREAK**. This causes the transmitter to enter the line-break condition for at least one character time. Several conditions control the continuation and/or termination of the line break. If there is no more data in the FIFO following the send break command, the break will continue indefinitely until terminated by a stop break command. If there is an insert delay command (see the next command) immediately following the send break command, the break duration will be set by the value programmed in the delay command. Any other character in the FIFO immediately following the send break command carries an "implied" end of break condition, causing the break to be terminated and the next character to be sent.

ESC x'82 x'xx: Insert **delay**. This command will cause a delay between the previous character transmitted and the next character to be transmitted. The hex value contained in the third byte of the sequence determines the time of the delay based on the basic time period set by the Prescale Period Register (PPR). The value is treated as an unsigned binary value that is loaded into an internal counter. The counter is decremented once for each "tick" of the prescale period timer. Thus, if the PPR sets a basic timing period of 10 ms and the value set by the command is 100 (x'64), then a delay of 1 second will be generated. Multiple insert delay commands can be placed in the FIFO if time delays longer than that generated by a single delay period are needed.

This command is useful when a delay is required after sending a carriage return. A printer is an example of this type of situation. Often, the carriage return causes the printer to start a print cycle and the sending device must wait for the print to complete before sending the next line of text (unbuffered input). Using the insert delay command allows the delay to be performed automatically without the need for the host to time it. The delay command is placed in the FIFO directly following the carriage return and preceding the first data for the next line. The CL-CD1400 will automatically execute the delay following the carriage return and then start sending characters again.

Another useful application of the delay command is as a built-in timer that the host can use as an interrupt source causing it to periodically check its internal buffers for data to transmit. This assumes that the channel is not currently transmitting data. When the host services the transmit FIFO service request after a delay time-out, as set by the delay value, it can start transmission of a buffer if data is available or re-send the insert delay command and wait for the next service request. This removes the necessity of the host setting an internal timer interrupt to perform the same function.

ESC x'83: Stop **BREAK**. This command will terminate a break in progress regardless of other conditions. This command may be preceded by insert delay commands to set a specific, programmed break period if more than one character time is needed. Any character in the FIFO will cause the break to terminate. EXC x'83 is needed only if it is necessary to stop the break and there is no more data to be sent. A break will continue until another character is sent or the ESC x'83 is encountered in the FIFO.

ESC x'01- x'3F: Send **Repeat Space**. This command will cause the CL-CD1400 to send repeated space characters. The character following the ESC is interpreted as a binary count specifying the number of ASCII space (x'20) characters to send. The count must be in the range of x'01 through x'3F (1 - 63 decimal), inclusive.

4.7.3 Send Special Character Command

The CL-CD1400 has, as one of its host commands, a method of transmitting any one of the four special characters programmed in special character registers SCHR1-SCHR4. The command is issued via the CCR register with bit 5 set to a one and the least significant three bits encoding a selection of one of the four characters (see Section 5 for details of the bit-encoding). The function is preemptive, meaning that the selected character will be transmitted immediately following the currently transmitting character and a character in the transmitter holding register, if any. This preempts any characters in the transmit FIFO. If there are characters in the transmit FIFO, transmission of those will resume after the special character is sent.

One important use of this command is that it allows the host to flow-control a remote without having to wait for the transmit FIFO to empty before the flow control character can be put in. This is a special case of the normal transmitter operation of the CL-CD1400 in that the character can be sent without waiting for a transmit service request. The only requirement is that the transmitter be enabled (interrupts need not be enabled).

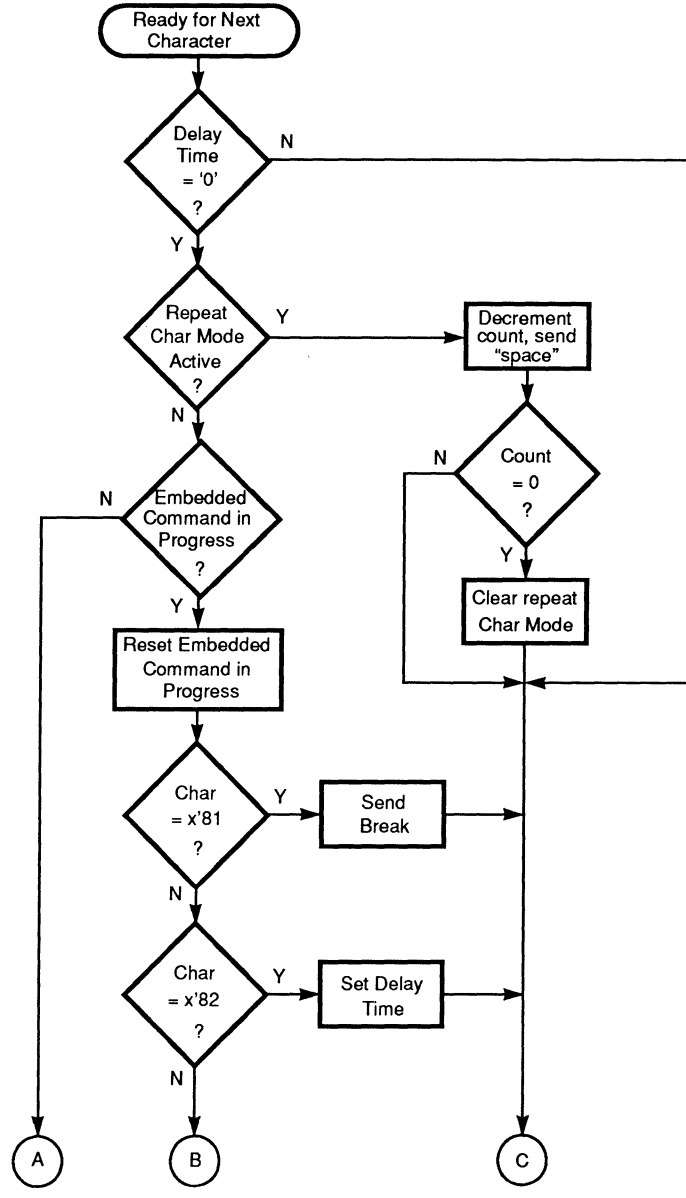


Figure 4–7. CL-CD1400 Transmit Character Processing

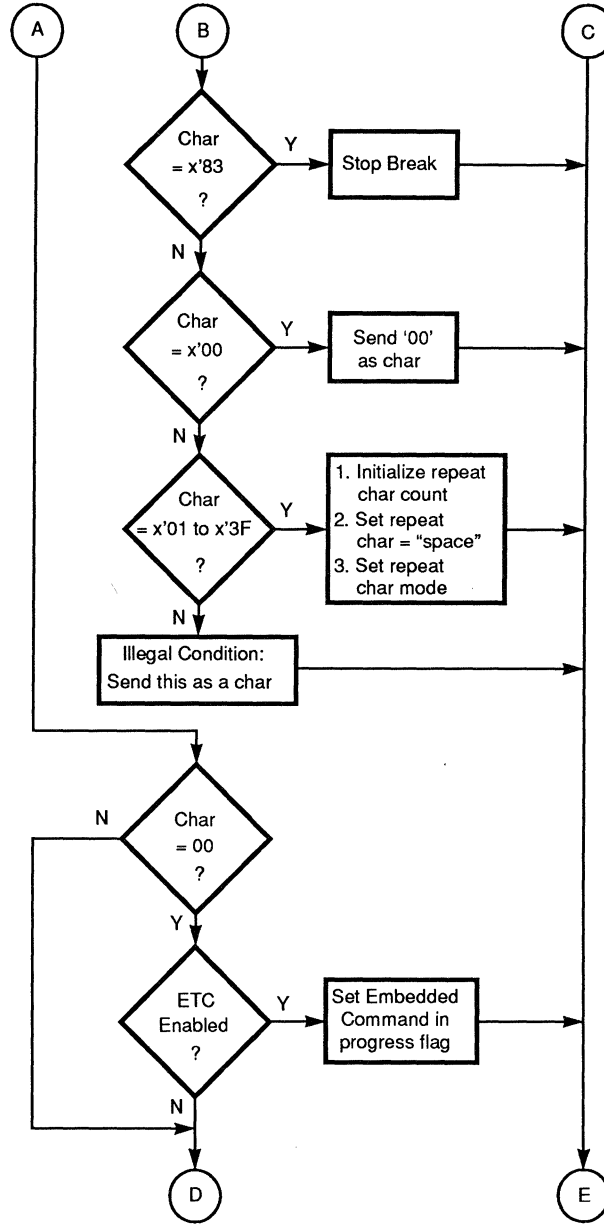


Figure 4–7. CL-CD1400 Transmit Character Processing *(cont.)*

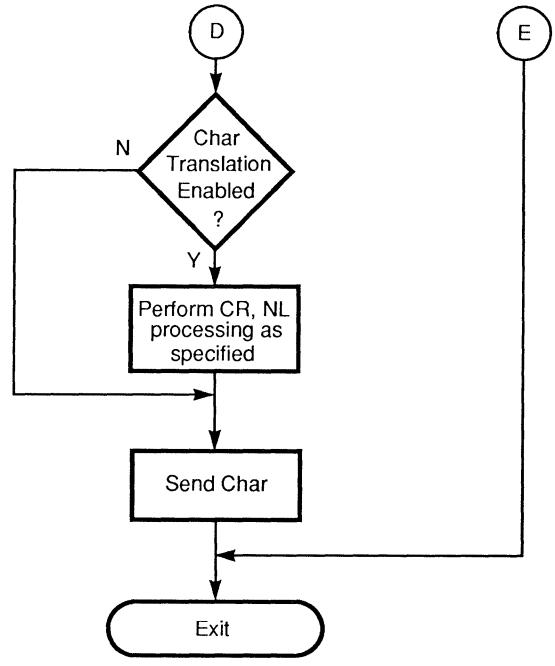


Figure 4–7. CL-CD1400 Transmit Character Processing *(cont.)*

4.8 Baud Rate Generation

The CL-CD1400 provides a separate baud rate generator for each direction of each channel. Each receive and transmit baud rate generator can be driven from one of five available clock sources. The source being used is selected by the value in the Receive Clock Option Register (RCOR) and Transmit Clock Option Register (TCOR). The selected clock is divided by the value in the Receive Baud Rate Period Register (RBPR) or Transmit Baud Rate Period Register (TBPR) to yield the desired bit rate.

The five clock sources are:

Clk0	System clock divided by 8, RCOR/TCOR = 0
Clk1	System clock divided by 32, RCOR/TCOR = 1
Clk2	System clock divided by 128, RCOR/TCOR = 2
Clk3	System clock divided by 512, RCOR/TCOR = 3
Clk4	System clock divided by 2048, RCOR/TCOR = 4

The system clock is the external clock driving the CLK input of the CL-CD1400. Two example baud rate tables are provided at the end of Section 6.

4.9 Diagnostic Facilities – Loopback

The CL-CD1400 provides the capability to perform loopback testing internally for both local and remote loopback modes. Loopback mode is enabled via the Local Loopback Mode (LLM) and Remote Loopback Mode (RLM) bits of COR2.

In local loopback, the output of the transmitter bit engine is connected directly to the input of the receiver bit engine and the input and output pins (TxD and RxD) are disconnected. The TxD output is left in the *mark* condition so that remote equipment does not see any line activity. Input conditions on the RxD are ignored. All channel parameters and service request functions are in effect and operate normally. If enabled, special characters are detected and acted upon and UNIX® translations take place.

Remote loopback mode causes the CL-CD1400 to echo any received data immediately back to the transmit output. This is done on a character-by-character basis rather than on a bit-by-bit basis; in other words characters are echoed once they have been completely received and assembled. Received data will not be placed in the FIFO, thus no data is given to the host. The received character will be re-transmitted with parity and stop bit options as defined by COR1. It is important to note that, if transmit baud rate is lower than receive baud rate, overrun errors and loss of data are likely to occur.

4.10 Parallel Channel Operations

Channel 0 is user-configurable as either a serial or a parallel port. The selection of operating modes is made by the value of the P/S* bit (bit 7) in the Global Configuration Register (GCR). After reset, channel 0 is configured as a serial port by default. Host software reconfigures the port to the parallel mode by setting this bit to a '1'. The port is capable of bi-directional operation, the direction being set by the enabled mode: transmit or receive. In the receive mode, the CL-CD1400 drives PBUSY as well as PSTROBE* automatically. PBUSY, however, is not a bi-directional handshake control: in transmit mode, it is not monitored by the CL-CD1400. PACK* is used as the acknowledge that the transfer has been completed.

It is important to note that when channel 0 is configured for parallel operation, several of the modem input signals of the other three channels are taken over for use by the parallel channel provide the necessary data and control/status signals required to support the parallel interface definition of the Centronics parallel specifications. These are the RI and CD inputs (since they are used for bidirectional data transfer, they are actually input/output signals). These six signals (RI[3:1]* and CD[3:1]*) and the two separate parallel data input/output signals (PD[0] and PD[1]) form the bi-directional parallel data port. Other modem input/output signals on channel 0 provide the control and status signals. The data transfer handshake is provided by the PSTROBE* output and the PACK* input. Note also that these two signals never change direction; PSTROBE* is always an output and PACK* is always an input. This is the normal signal name convention for parallel data transmit. For receive, the PSTROBE* output effectively becomes the transfer acknowledge (ACK) output function and the PACK* becomes the data strobe (STROBE) input function.

Unlike some parallel interface devices which require the host to control and generate the timing for the handshake signals directly, the CL-CD1400 automatically generates the PSTROBE* and PBUSY outputs and monitors the PACK* input. This removes nearly all host overhead in data transmission or reception other than putting data in, or taking data out of, the FIFO. Since parallel data movement is inherently half-duplex, the CL-CD1400 combines the serial receive and transmit FIFOs into one large, 30-byte FIFO. This further reduces host overhead by providing larger buffering and thus reduced service request activity.

The width of the PSTROBE* pulse is set by the programmed bit-time generated by the TCOR/TBPR register pair. The width can be any duration, but in no case should it be set to a value less than 15 µsec. For best overall performance, the RCOR/RBPR register pair should be set to generate a bit-time of 64K baud. PBUSY pulse width is dependant on the duration between the

strobe input to the CL-CD1400 and the acknowledge output (PACK* to PSTROBE* delay; see Section 3).

General operation of the parallel port is the same as the serial port. When the channel is in the transmit mode and the transmitter and transmit service requests are enabled, the CL-CD1400 will request service when ever the FIFO is empty. The host responds by writing up to 30 bytes into the FIFO. Carriage return and newline mapping occur on transmit data in the same manner as they do in serial mode. Only the send repeated space command of the embedded transmit command set is operative. The receiver also works the same way in parallel mode as it does in serial. The FIFO threshold can be set to trigger on any level between 1 and 30 inclusive. However, in the interest of highest possible performance, no receive special character processing occurs.

4.10.1 Transmit Operation

When the channel is enabled for transmit operation and service requests are enabled, the CL-CD1400 will transmit data whenever characters are in the FIFO. As with serial operation, service requests can be programmed to occur when the FIFO becomes empty or when the last character has been transmitted. The mode is determined by the type of service request enabled in the SRER register. Data transmission is controlled by the CL-CD1400 via the handshake signals PSTROBE* and PACK*.

When the FIFO has a byte to send, the CL-CD1400 puts the 8 bits of data on the parallel port. After a 200 nsec set-up time, the PSTROBE* signal is driven active (low) and remains active for the time specified by the values programmed in the TCOR/TBPR registers. PSTROBE* is then deactivated and the data is held until the PACK* input is driven active (low) by the receiving device. PACK* is the only signal monitored automatically by the CL-CD1400 for flow control purposes. PBUSY, however, is made available to the host via the PSVR register. Host software can detect the busy condition and disable the transmitter, if necessary. Once PACK* becomes inactive, the CL-CD1400 will place the next data byte on the port (if the FIFO is not empty), and the cycle repeats. If the receiving device is not able to accept the data, it may hold off the cycle indefinitely by not activating PACK*. This provides the parallel version of flow control.

Figure 4–8 shows a typical connection for a CL-CD1400 parallel transmit interface, which is a printer in this example.

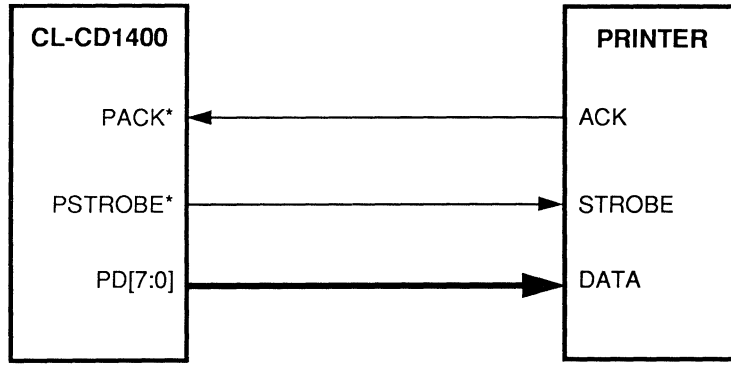


Figure 4–8. CL-CD1400 Parallel Data Transmit Connections

4.10.2 Receive Operation

Receive operation in parallel mode is also very much like the serial mode. The receive FIFO has a threshold setting that determines the level of data required to cause a receive service request. The threshold can be set anywhere from 1 to 30 characters. When the number of characters in the FIFO equals the set value of the threshold, the CL-CD1400 will post a receive service request.

The sequence of events in receiving a byte is the reverse of sending, the sending device places the data on the parallel data port; after an appropriate set-up time, it activates its strobe out. The strobe is connected to the PACK* input of the CL-CD1400. When the CL-CD1400 senses the active level on the PACK* input, it activates PBUSY to indicate that it is taking the data but is not yet done. Once it has taken the data, the it will activate its PSTROBE* output, which should be connected to the senders acknowledge input. At the end of the programmed pulse width duration, PSTROBE* and PBUSY are deactivated.

Flow control happens automatically in the receive direction. If the FIFO and the holding register are full when the next data byte is being received, the CL-CD1400 will maintain PBUSY in the active (high) state and will not activate PACK*. Once the host has serviced the receive FIFO and has removed at least one byte, thus making room for the current byte, the CL-CD1400 will complete the receive cycle by acknowledging the byte (activate PACK*) and deactivating PBUSY.

Figure 4–9 shows the connections between the CL-CD1400 and a sending device such as a scanner. This connection might also be seen in an application where the CL-CD1400 is the receiving device in a printer application.

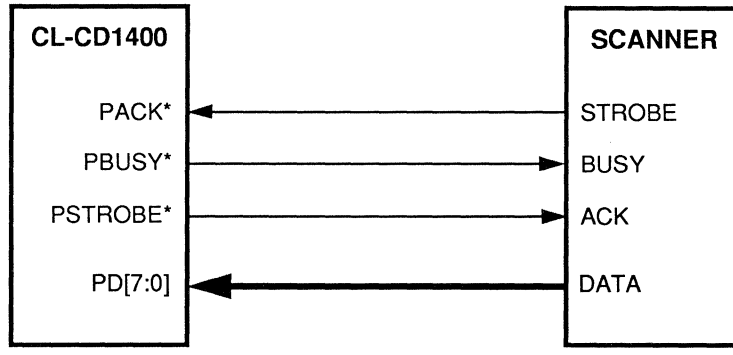


Figure 4–9. CL-CD1400 Parallel Data Receive Connections

4.11 Hardware Configurations

The simplicity of the host interface to the CL-CD1400 allows it to be built into systems making use of any of the microprocessors on the market, such as the Intel 80x86 family (8086, 80286, 80386, etc.), the Motorola family (68000, 68010, 68020, etc.), the National 32x32 family (32CG16, 32332, 32532, 32GX32, etc.), and the AMD29000.

4.11.1 Interfacing to an Intel Microprocessor-Based System

With very little extra logic, the CL-CD1400 can be interfaced to any system based on a processor in the Intel 80x86 family. The figure below shows a generalized view of an I/O mapped interface with an 80286 based system. In order to provide the proper strobes and controls, the IOR* and IOW* control strobes are used to synthesize the DS* and R/W* signals. DTACK* is used as an input to wait state generation logic that will hold the processor (if necessary) until the CL-CD1400 has completed the I/O request.

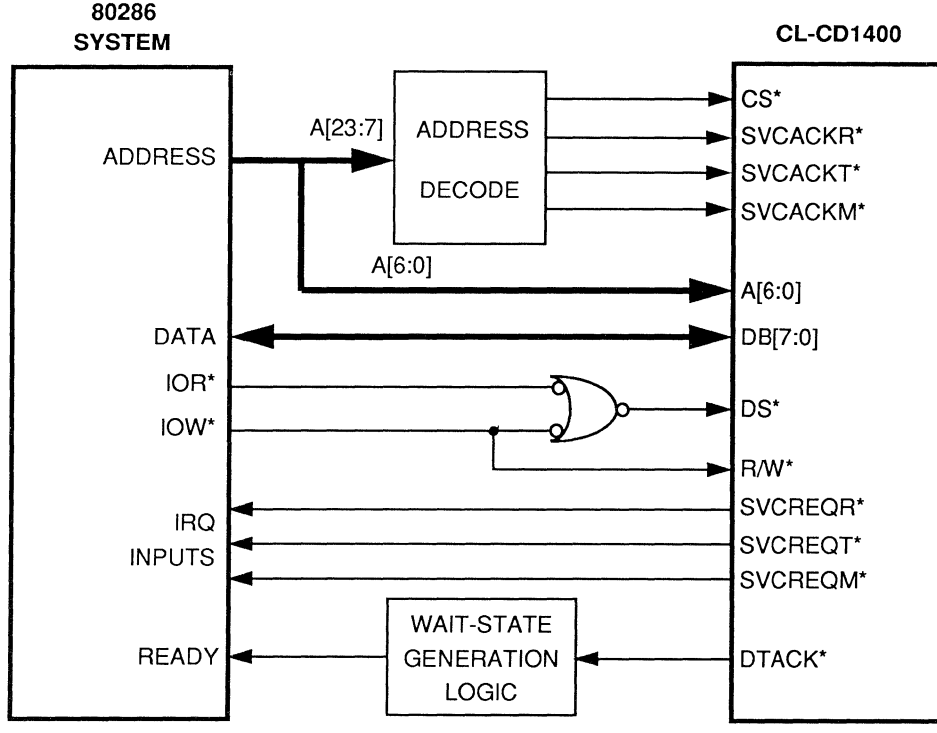


Figure 4–10. Intel 80x86 Family Interface

4.11.2 Interfacing to a Motorola Microprocessor-Based System

Interfacing to a 68000 family devise is very straight forward. The bus timing and the interface signal definitions very closely match those of the 68000 microprocessor, thus allowing direct connection in most cases. With later versions (68020, 68030), some additional logic is required to generate the DSACK0* and DSACK1* functions that replace the DTACK* on the earlier devices. The example below is a generalized interface to a 68020 device.

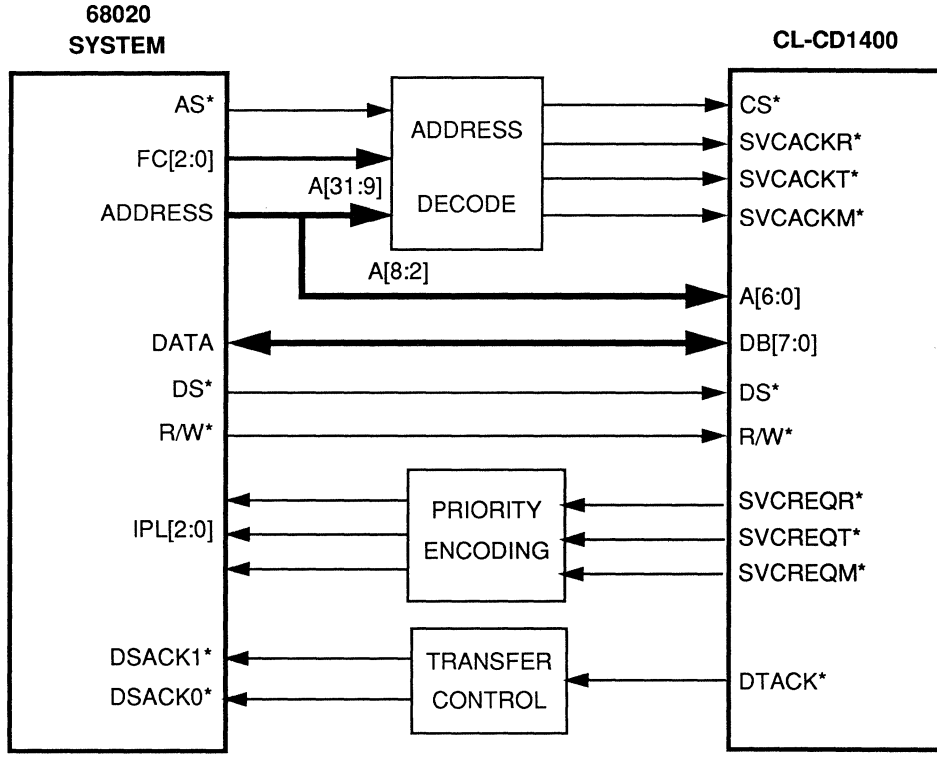


Figure 4–11. Motorola 68020 Interface

4.11.3 Interfacing to a National Semiconductor Microprocessor-Based System

The connections between the CL-CD1400 and a NS32000 (32GX320, 32CG16, etc.) embedded controller is also relatively simple. As with the Intel devices, cycles are controlled by the DS, CS and R/W signals that have been synthesized from the available I/O control signals and I/O cycle extensions (wait states) are generated by logic connected to the DTACK signal. Some additional consideration is required to implement memory-mapped I/O to prevent multiple read and write cycles with the CL-CD1400 FIFOs due to the pipelined architecture of the 32000 device but all of the necessary controls are available.

The figure below depicts a simplified interface example.

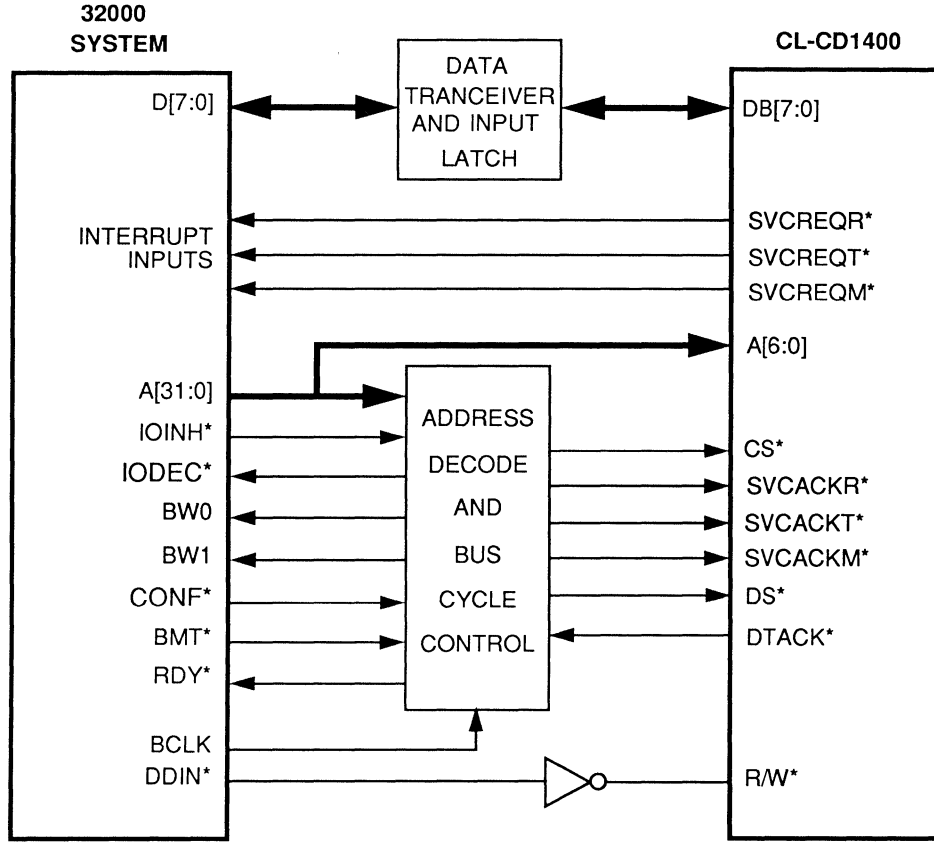


Figure 4–12. National 32000 Interface

4.12 Serial Data Performance

The maximum rate at which the CL-CD1400 can move serial data in and out of its internal registers is affected by many factors. The actual rate that is achievable by the bit engines themselves is approximately 250,000 bits per second. This is based on the smallest value of baud rate period at which the bit engines can operate. However, this is simply the speed that data can be shifted, not how fast characters can be assembled/disassembled and processed for any special character manipulation that is selected. The real limiting factor in the maximum data rate is the MPU (not taking into account host overhead in management of the receive and transmit FIFOs, which does have an impact on performance).

Recall that the bit engines operate at the bit level. When a bit engine has completed operation on a bit, whether it is related to receiving or transmitting, it interrupts the MPU indicating it is done with the task. The MPU performs the bit manipulation at the character level. If, for example, the bit engine interrupt is for a bit that has been received, the MPU must add the bit to the character being assembled and test for a complete character. Tasks performed during a bit engine interrupt are called foreground tasks. The background code will perform further manipulation of the character, such as checking parity or other error conditions, placing the character in the FIFO, generating a service request if appropriate, etc. All of these tasks require time, and as the number of interrupts from bit engines in a given time period increase due to higher data rates, the MPU has less time for the background tasks. This is compounded by requiring it to perform additional special tasks, such as UNIX character translations.

Internally, the MPU prioritizes its activities in the order of receive, transmit, modem. At the higher data rates, more time will be spent on receive characters than on transmit characters simply because there is less time overall between interrupts. When data rates reach a level at which the MPU cannot keep up, the transmitter will begin to show gaps between characters. This is the fail-safe condition; no data is lost. As the data rate continues to increase, inter-character gaps become longer and longer. At extremely high rates, the MPU begins to have trouble keeping up with receiver bit engine interrupts and the whole system breaks down and data is lost.

Extensive performance analysis has been performed on the CL-CD1400. The analysis was based on throughput at various serial data rates at a nominal system clock of 20 MHz. Throughput is measured as the amount of data transmitted over a given time period versus the theoretical maximum amount based on calculation. The difference between the two is made up of inter-character gaps.

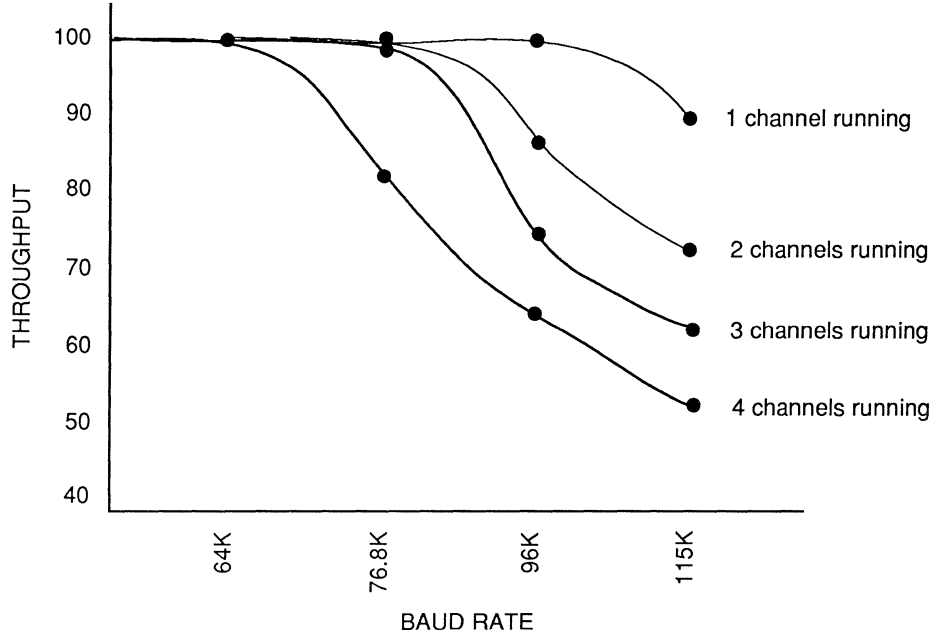
This analysis has shown that the device can support 100% throughput at 115.2K baud on all four channels in half-duplex mode (transmitting) and minimum optional character processing. This mode of operation might be found in terminal servers, for example, which tend to be half-duplex in operation. Nearly all high speed traffic is one direction, server to terminal, with relatively little traffic in the other direction, terminal keyboard to the server. Occasionally, the data movement will be reversed, such as for a file upload, but then the low duty-cycle traffic is also reversed. In a true full-duplex application, 100% throughput is achieved on all four channels up through 70K baud, also with minimum special options enabled. As the number of operating channels decreases, the maximum data rate at which 100% throughput can be maintained increases. One channel operating full-duplex can maintain 115.2K baud at 93% throughput.

The performance achieved in an application is dependent on a number of factors. As mentioned above, host performance is also a factor. If the host cannot respond to a transmit FIFO service request in time to keep it from running out of data, than the apparent throughput will decrease due to the inter-character gaps caused by the underrun. Of course, if the host cannot respond fast enough to a receive FIFO service request, overruns will result. High data rates increase the load on the host system as well as the MPU within the CL-CD1400. Another factor that has an obvious

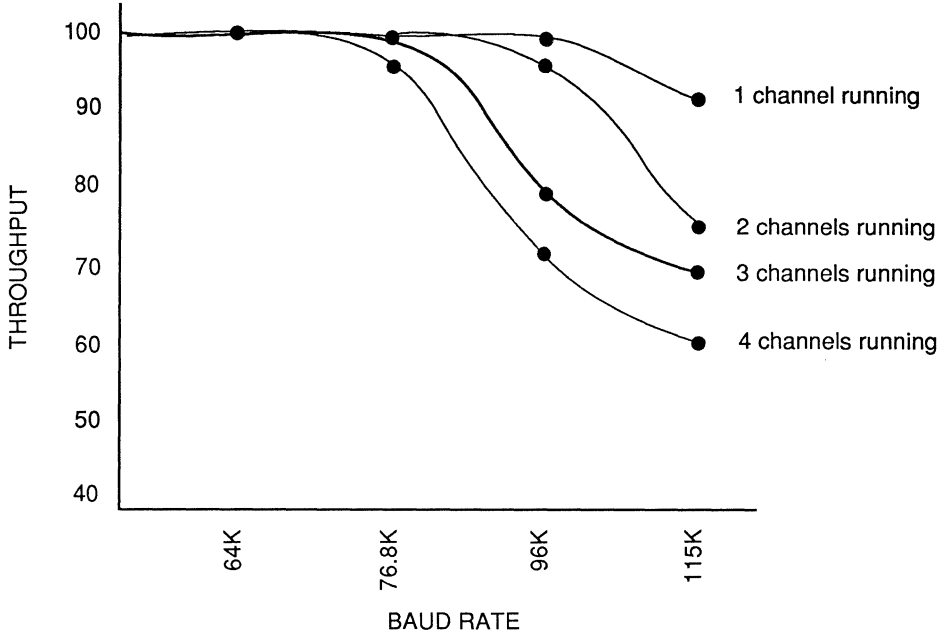
affect on performance is system clock speed. At the maximum recommended operating frequency of 20.2752 MHz, the throughput numbers shown in the diagrams will increase slightly. As system clock frequency decreases, there is a linear decrease in CL-CD1400 serial data performance, since the MPU is operating at slower pace.

The data that has been collected on CL-CD1400 is diagrammed in the following graphs. In order to show both best case and worst case conditions, the tests were run in two modes. The first mode turned on all possible special character processing, thus loading the MPU with the maximum amount of work necessary on each received character. The second mode did not enable any special character processing. The host in use for these tests was a PC/AT-type system based on a 25-MHz 80386 CPU. The overhead of the host service routines has been factored out of the data in order to show true CL-CD1400 performance levels.

Throughput with maximum character processing options enabled:



Throughput with minimum character processing options enabled:



5. DETAILED REGISTER DESCRIPTIONS

This section presents a complete, detailed description of each register. Registers have two formats: full eight bits, where the entire content defines a single function; or, the register is a collection of bits, grouped singly or in multiples, defining a function. In the second case, the descriptions break the register down into its component parts and describe the bits individually. The order of register presentation follows that given in the brief register descriptions in Section 2.

5.1 Global Registers

Global Firmware Revision Code Register (GFRCR) 40 Read Only

Firmware Revision Code

The GFRCR serves two purposes in the CL-CD1400. First, it displays the revision number of the firmware in the chip. When a revision to the CL-CD1400 is required, the revision number of the firmware is incremented by one. Beginning with Revision C, the code is 42. Later revisions will increment this by one; for example, revision D will be 43, and so on.

Secondly, this register can be used by the system programmer as an indication of when the internal processor has completed reset procedures, after either a power-on reset (via the RESET* input) or a software global reset (via the reset command in the CCR). Immediately after the reset operation begins, the internal CPU clears the register. When complete, and the CL-CD1400 is ready to accept host accesses, the register is loaded with the revision code.

Channel Access Register (CAR) 68 Read/Write

n/u	n/u	n/u	n/u	n/u	n/u	C1	C0

The CAR provides access to individual channels within the CL-CD1400. The least significant two bits of the register selects one of the four channels. Before any operation that affects a channel, this register must be loaded so that channel registers are available to the host. Bit 2 must always be 0.

C1	C0	Channel Selected
0	0	Channel 0
0	1	Channel 1
1	0	Channel 2
1	1	Channel 3

Global Configuration Register **(GCR)** **4B** **Read/Write**

P/S*	n/u	n/u	n/u	n/u	n/u	n/u	n/u

The GCR is used to define the mode of operation for channel 0. Resetting bit 7 will select serial mode; setting bit 7 selects parallel. The default mode selection after reset is serial.

P/S* = 0 Channel 0 mode is serial
P/S* = 1 Channel 0 mode is parallel

Service Request Register **(SVRR)** **67** **Read Only**

0	0	0	0	0	SRM	SRT	SRR

The SVRR reflects the inverse of the state of the service request pins (SVCREQR*, SVCREQT* and SVCREQM*). Its primary use is in polled systems, and it allows system software to determine what, if any, service requests are pending.

Bit	Description
7-3	Always 0
2	Service Request Modem; 1 indicates request pending
1	Service Request Transmit; 1 indicates request pending
0	Service Request Receive; 1 indicates request pending

Receive Interrupting Channel Register **RICR** **44** **Read/Write**

Transmit Interrupting Channel Register **TICR** **45** **Read/Write**

Modem Interrupting Channel Register **MICR** **46** **Read/Write**

X	X	X	X	C1	C0	X	X

These registers indicate the channel number that is currently being serviced by an active acknowledge cycle (whether polled or interrupt). Bits 3-2 (C1 and C0) are valid *only* during the context of a channel service routine; at any other time, their state is undefined. Host system software uses these registers to determine the number of the channel that originated the particular service request (receive, transmit or modem). The format of these registers is the same and the description is valid for each. The upper four bits and lower two bits are user defined and may be set to any value desired. When the register is read, these bits will be presented as defined by the user; C1 and C0 will be set by the CL-CD1400 to reflect the proper channel number.

Bit	Description
7-4	User Defined
3-2	Defines Channel Number

C1	C0	Channel Number
0	0	Channel 0
0	1	Channel 1
1	0	Channel 2
1	1	Channel 3

1-0 User defined

Receive Interrupt Register (RIR) 6B Read/Write

rdireq	rbusy	runfair	1	1	0	ch[1]	ch[0]

Transmit Interrupt Register (TIR) 6A Read/Write

tdireq	tbusy	tunfair	1	0	0	ch[1]	ch[0]

Modem Interrupt Register (MIR) 69 Read/Write

mdireq	mbusy	munfair	0	1	0	ch[1]	ch[0]

These registers are used during poll mode operation of the CL-CD1400. All three provide the same type of information for each of the three service requests. The functions of rxireq, txireq and mdireq have identical meanings, as do the group rbusy, tbusy and mbusy and the group runfair, tunfair and munfair. The least significant two bits indicate the number of the channel requesting service. Bits 2 through 4 are used internally by the CL-CD1400 to set the context of the service acknowledge cycle. See the description of poll mode operation in Section 4 for complete details.

Receive/Transmit/Modem Interrupt Registers *(cont.)*

rxireq
txireq
mdireq

These bits are set by the internal processor when service is required by a channel. They are a direct reflection of the inverse state of the SVCACK* pins, and they are not actually part of the register but are the active high output of the latch that drives the SVCACK* pins. The bits can be scanned by the host to detect an active service request. These bits are cleared by the internal processor at the beginning of the service acknowledge cycle (hardware service acknowledge) or by the host software when the poll mode cycle is terminated.

rbusy
tbusy
mbusy

These bits are set by the internal processor and remain set until the end of the service acknowledge cycle is indicated by either a write to the EOSRR (hardware service acknowledge), or cleared by the host software when the poll mode cycle is terminated. They signal the current state of the service acknowledge cycle. When cleared, the internal processor knows that it can assert another service request of this type.

runfair
tunfair
munfair

These bits are used by the internal processor to implement the "Fair Share" service request function. If this bit is set, the CL-CD1400 will not assert another service request of this type until the bit is cleared by a pulse on the external SVCACK* pin. These bits are not used in poll mode.

Bits 4-2

These bits define the context of the current service acknowledge cycle during poll mode and are fixed by hardware within the CL-CD1400. These bits must be replicated exactly when the register is copied to the CAR when activating a service acknowledge cycle. See the discussion of poll mode operation in Section 4 for a more detailed description.

ch[1] - ch[0]

These two bits encode the channel number of the requesting channel. During poll mode operation, when the RIR, TIR and MIR is copied into the CAR to start the service routine, ch[1:0] set the channel number that will be serviced.

Prescaler Period Register **(PPR)** **7E** **Read/Write**

Binary Value

The PPR sets the divisor that will be used to generate the time period for CL-CD1400 timer operations. It can be set to any value between 0 and 255 (x'FF). The PPR is clocked by the system clock prescaled (divided) by 512. Note that this value does not have any effect on baud rate generation. The time period generated by this register drives the receive timer and is used to activate the "no new data" and "receive data time-out" interrupts. See the receiver operation discussion in Section 4 for a description of receiver timer functions.

5.2 Virtual Registers

The CL-CD1400 has two operational contexts, a normal context which allows host access to most registers and any channel, and a service acknowledge context, allowing host access to some registers specific to the channel requesting service. This special set of registers is called virtual, because they are only available to host access and valid during this service acknowledge context; at all other times, their contents will be undefined and must *not* be written to by host software.

The use of virtual registers and context switching allows the CL-CD1400 to maintain all channel-specific information. The host need not make any changes to chip registers in order to access the registers pertinent to the channel being serviced.

The service acknowledge context is entered into in one of two ways; either via activation of one of the SVCACK* input pins (hardware activated), or via host software when the contents of any one of TIR,RIR, MIR is copied into the CAR by host software during a poll mode acknowledge cycle. See Section 4 for a discussion of the differences between these two modes.

Receive Interrupt Vector Register (RIVR) 43 Read Only

X	X	X	X	X	IT2	IT1	IT0

Transmit Interrupt Vector Register (TIVR) 42 Read Only

X	X	X	X	X	IT2	IT1	IT0

Modem Interrupt Vector Register (MIVR) 41 Read Only

X	X	X	X	X	IT2	IT1	IT0

These registers serve the same function for each of their respective service types – receive, transmit and modem. They provide information about the service request that is being acknowledged.

Receive/Transmit/Modem Interrupt Vector Registers *(cont.)*

The upper five bits are user defined, as programmed via the LIVR register of the channel being serviced; the lower three bits provide the service acknowledge vector and are OR'ed in by the CL-CD1400 when the register is read. The use of the TIVR and MIVR is optional if the value contained in the upper five bits is not needed by host software. The vector provided will be as indicated for the particular interrupt. IT2-IT0 will indicate that the service is for transmit in the case of the TIVR and modem for the MIVR. The value of these bits will be important when servicing a receive service request; IT2-IT0 will indicate whether the service request is for "good" data or "exception" data. The following table shows the encoding of IT2-IT0:

IT2	IT1	IT0	Encoding
0	0	0	Not used
0	0	1	Group 1: Modem signal change service request
0	1	0	Group 2: Transmit data service request
0	1	1	Group 3: Received good data service request
1	0	0	Not used
1	0	1	Not used
1	1	0	Not used
1	1	1	Group 3: Received exception data service request

Transmit Data Register **(TDR)** **63** **Write Only**

Transmit Character

The transmit data register is the host's port for writing to the transmit FIFO. When a channel is being serviced for a transmit service request, the host may write up to 12 characters into this register. The transmit data register must only be written to during the context of a transmit service acknowledge. Writing data to this location at any other time will have unpredictable results.

Receive Data/Status Register **(RDSR)** **62** **Read Only**

Received Character

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

The receive data and status register serves two purposes. During a receiver service acknowledge for good data, the RDSR provides access to the receive FIFO. The number of characters available in the FIFO is indicated by the receive data count register (RDCR), which will be described in the channel register section. Any number of characters, up to the value in the RDCR, may be read from the FIFO. All internal FIFO pointers are updated by the on-chip processor.

During a receive exception service acknowledge, the RDSR provides both the received character and the status that caused the exception condition. By definition, a receive exception service request will have only one character available (multiple receive exceptions will produce multiple service requests). The first read from the RDSR will provide the exception status, and the second read will provide the character. It is not necessary to read either of these values. If the service acknowledge is terminated without reading any data from the RDSR, the internal processor will update the FIFO pointers as if the status/character were read. The same is true if only the status is read. Overrun errors are an exception to this (see below).

Bit 7 Time-out – If the service request enable for time-out is set, this bit will indicate that no data has been received within the receive time-out period set by the receive time-out period register (RTPR) after the last character was removed.

Bits 6-4 Special Character Detect Encoding

SCDet2	SCDet1	SCDet0	Status
0	0	0	None Detected
0	0	1	Special Character 1 matched
0	1	0	Special Character 2 matched
0	1	1	Special Character 3 matched
1	0	0	Special Character 4 matched
1	0	1	Not used
1	1	0	Not used
1	1	1	Range Detect

NOTE: No special character matching is performed if either parity error (PE) or framing error (FE) are set.

Bit 3 Break – Indicates that a break was detected.

Bit 2 Parity Error – Indicates that a character was received with parity other than that programmed in COR 1.

Bit 1 Framing Error – Indicates that the character was received with a bad stop bit.

Receive Data/Status Register *(cont.)*

Bit 0 Overrun Error – This bit will be set if new data is received, but there is no space available in the FIFO and holding register. In this case, the character data is lost, and the overrun flag is applied to the last good data received before the overrun occurred. Thus, the character read on the subsequent read from the RDSR is good data and should not be discarded.

Modem Interrupt Status Register (MISR) 4C Read Only

DSRch	CTSch	RIch	CDch	0	0	0	0

The MISR provides the status indication of the reason for a modem service request. If either or both of the modem change modes (zero-to-one or one-to-zero transition) are enabled, such a change will cause a service request, and the signal that changed will be flagged in this register.

Bit 7 DSR change – An enabled transition on the Data Set Ready signal will cause this bit to be set and a modem service request posted.

Bit 6 CTS change – An enabled transition on the Clear To Send signal will cause this bit to be set and a modem service request posted.

Bit 5 RI change – An enabled transition on the Ring Indicator signal will cause this bit to be set and a modem service request posted.

Bit 4 CD change – An enabled transition on the Carrier Detect signal will cause this bit to be set and a modem service request posted.

Bits 3-0 These bits will always return zero.

End Of Service Request Register (EOSRR) 60 Write Only

X	X	X	X	X	X	X	X

The EOSRR register is a dummy location and is used to signal the end of a hardware service acknowledge procedure, activated via one of the SVCACK* pins. The data pattern written is a "don't care" value. The action of writing this location causes the CL-CD1400 to perform its internal switch out of the service acknowledge context. This register is only used during a hardware-activated service acknowledge and must not be written during poll mode operation.

5.3 Channel Registers

Each of the four channels has a set of registers that control aspects of its operation. In the information below, the register contents and offsets apply to any of the channels; the channel being accessed at any given time is controlled by the CAR. This holds true even during a service acknowledge context; the CAR points to the channel be serviced, whether it was loaded by the host (during poll mode operation) or by the CL-CD1400 itself (during hardware activated service acknowledge).

In a few of the following cases, some of the registers show two formats, one for serial and one for parallel. In these cases, the parallel register format applies only to channel zero, and only when the GCR has been programmed to place channel zero in parallel mode.

Local Interrupt Vector Register (LIVR) 18 Read/Write

X	X	X	X	X	IT2	IT1	IT0

The LIVR is used only during hardware-activated service acknowledge cycles. Host software loads any information it wants into the most significant five bits; the least significant three bits are not used. When the CL-CD1400 is setting up a service request, it overlays the five significant bits of the LIVR into appropriate interrupt vector register (RIVR, TIVR and MIVR) and sets the least significant three bits as required for the service request vector type. (See RIVR, TIVR and MIVR description earlier in this section).

Channel Command Register (CCR) 05 Read/Write

Res Chan	COR Chg	Send SC	Chan Ctl	D3	D2	D1	D0

Format 1: Reset Channel Command

Res Chan	0	0	0	0	0	FTF	Type

Format 2: Channel Option Register Change Command

0	COR Chg	0	0	COR3	COR2	COR1	0

Format 3: Send Special Character Command

0	0	Send SC	0	0	SSPC2	SSPC1	SSPC0

Format 4: Channel Control Command

0	0	0	Chan Ctl	XMT EN	XMT DIS	RCV EN	RCV DIS

The Channel Command Register is used to issue commands directly to the on-chip processor to control or change some channel and, in one case, global functions of the channel selected by the CAR. The upper four bits indicate which of four command types is being issued, and the lower four bits are parameters to those commands. At no time should more than one bit be set in the command type field. When the command has been executed by the CL-CD1400, it will zero out the CCR. Therefore, two consecutive commands must wait for the CCR to be cleared after the first is issued, before the second is issued.

Channel Command Register *(cont.)*

Format 1 – Reset Channel Command

When bit 7 is set, one of three types of reset operations are initiated, based on the value of the least significant two bits. Bit 0 sets the type of reset, either channel-only or full-chip, and bit 1 causes the FIFO of the selected channel to be flushed.

The two types of reset selected by bit 0 cause very different results. When bit 0 is a zero, the reset command effects only the selected channel. Resetting a channel disables both the receiver and transmitter, and all FIFOs are flushed (cleared). If bit 0 is a one, a full-chip reset is initiated. This reset will have the exact same results as a hardware reset caused by activation of the RESET* input pin. All channels are disabled, all FIFOs are flushed and all control registers set to their power-on reset state. The completion of the reset operation can be detected in the same manner as if a power-on or hardware reset had occurred; the GFRCR will change from zero to the value of the firmware revision. It should be noted that, at the beginning of the reset operation, the GFRCR will be cleared, but it may take some time for this to happen. Host software should wait for the GFRCR to go to zero, and then wait for it to go non-zero to indicate that the reset operation is complete.

The FTF (flush transmit FIFO) command, bit 1, will cause the transmit FIFO of the selected channel to be cleared. Any data in the FIFO will be lost.

The encoding of the bits for the reset channel command is:

Bit 7 Must be 1
Bits 6-2 Must be 0
Bits 1-0 Encoded as:

FTF	Type	Function
0	0	Reset current channel
0	1	Full CD1400 reset
1	0	Flush transmit FIFO of current channel
1	1	Not used

Format 2 – Channel Option Register Change Command

Bit 6, in combination with any of bits 1 through 3, will inform the MPU that a change has occurred in one of the Channel Option Registers, COR1, COR2 and/or COR3, respectively. It is permissible to indicate that more than one COR has changed.

This command exists so that changes in the COR registers will be noted by the MPU, allowing it to update its internal working register, since it keeps copies of the COR registers in its own shadow registers.

Bit 7 Must be 0
Bit 6 Must be 1
Bits 5-4 Must be 0
Bits 3-1 Encoded as:

COR3	COR2	COR1	Encoding
0	0	0	Not used
0	0	1	COR1 Changed
0	1	0	COR2 Changed
0	1	1	COR1 and COR2 Changed
1	0	0	COR3 Changed
1	0	1	COR3 and COR1 Changed
1	1	0	COR3 and COR2 Changed
1	1	1	COR1, COR2, and COR3 Changed

Bit 0 Must be 0

Format 3 – Send Special Character Command

This command causes one of the pre-programmed characters in the special character registers (SCHR1, SCHR2, SCHR3, SCHR4) to be sent preemptively. The character sent is selected by the settings of bits 2 through 0. "Preemptively" means that the special character will be sent immediately following the character in the transmitter holding register; it will not wait until the FIFO empties. Once the special character is sent, transmission of any characters remaining in the FIFO will proceed normally. The encoding of the bits is:

Bits 7-6 Must be 0
Bit 5 Must be 1
Bits 4-3 Must be 0

Format 3 – Send Special Character Command *(cont.)*

Bits 2-0 Encoded as:

SSPC2	SSPC1	SSPCO	Encoding
0	0	0	Not used
0	0	1	Send special character 1
0	1	0	Send special character 2
0	1	1	Send special character 3
1	0	0	Send special character 4
1	0	1	Not used
1	1	0	Not used
1	1	1	Not used

Format 4 – Channel Control Command

This command is used to activate or deactivate the transmitter and/or receiver of the selected channel, based on the values in bits 3 through 0. This command must be issued when a channel is being started for the first time. Once a channel is in use, it can be started and stopped using this command, though it is more efficient to use of the appropriate SRER bit in the Interrupt Enable Register. Multiple control commands can be issued at the same time; for example, both the transmitter and receiver can be enabled by setting both the XMT EN and RCV EN bits at the same time.

Issuing an enable/disable command does not affect any register programming of the selected channel. It does, however, affect the state of transmit flow-control. Issuing a disable or enable command to a channel whose transmitter has been flow-controlled by a remote (see the TxIBE bit in COR2) will restart transmission and clear the TxFloff bit in the CCSR. This ability is provided so that the host can override remote-generated flow control.

Bits 7-5 Must be 0

Bit 4 Must be 1

Bits 3-0 Select channel enable/disable activity:

XMT EN	XMT DIS	RCV EN	RCV DIS	Encoding
0	0	0	1	Disable receiver
0	0	1	0	Enable receiver
0	1	0	0	Disable transmitter
1	0	0	0	Enable transmitter
0	1	0	0	Disable transmitter and receiver
0	1	1	0	Disable transmitter, enable receiver
1	0	0	1	Enable transmitter, disable receiver
1	0	1	0	Enable transmitter and receiver

Service Request Enable Register (SRER) 06 Read/Write

MdmCh	0	0	RxData	0	TxRdy	TxMpty	NNDT

This register is used to enable the conditions that will cause the CL-CD1400 to post a service request via the SVRR and the SVCREQ* output pins. Each of the individual enable bits controls one type of service request.

Bits 7 MdmCh

This bit enables the Modem Change service request. When this bit is a one, any selected modem signal change conditions (as programmed by the MCOR1 and MCOR2 registers) will cause a Modem Service Request to be posted.

Bits 6-5 Must be zero.

Bit 4 RxData

The RxData enable bit enables the posting of receive service requests when characters have been received, and either the FIFO has reached the programmed threshold (as set by COR3), or the receive time-out period has expired.

Bit 3 Must be zero.

Bit 2-1 TxRdy and TxMpty

The transmitter can be enabled to post service requests on one of two conditions: either the FIFO is empty, or the transmitter shift register is empty.

The TxRdy bit enables the service request on the condition that the FIFO is empty. In this case, there are still two characters available for transmission before the transmitter underruns, one in the shift register and one in the holding register.

Service Request Enable Register *(cont.)*

The TxMpty bit enables the service request on the condition that the shift register is empty. The transmitter will underrun in this situation due to the latency that will be experienced between the time the service request is posted, and the time that the host is able to load the FIFO. Under normal operating conditions, this bit will be set and the TxRdy reset, when there is no more data to be transmitted, and the host wants to know when the last character has been sent before disabling the transmitter.

Bit 0 NNDT

The No New Data Time-out enable bit activates the optional exception service request when all data has been removed from the FIFO, and no new data has arrived after a pre-programmed delay period set by the value in the Receive Time-out Period Register (RTPR). The LIVR (or RIVR) will indicate a receive exception in the IT2-IT0 vector bits. There will be no data associated with this exception service request. A status bit in the RDSR (bit 7) will indicate that the service request is for an NNDT condition.

Channel Option Registers

The following five Channel Option registers are used to control many aspects of CL- CD1400 channel operation and enable special character processing features. COR4 and COR5 are used specifically for enabling the UNIX line discipline character handling functions.

Channel Option Register 1 (COR1) 08 Read/Write

Parity	ParM1	ParM0	Ignore	Stop1	Stop0	ChL1	ChL0

Bit 7 Parity Type

This bit selects the type of parity that is generated and checked if parity is enabled. A "1" selects odd parity and a "0" selects even parity.

Bits 6-5 Parity Mode 1 and Parity Mode 0

The parity mode bits define the parity operation for both the transmitter and receiver. The encoding is:

ParM1	ParM0	Function
0	0	No parity
0	1	Force parity (odd parity = force 1, even parity = force 0)
1	0	Normal parity
1	1	Not used

Bit 4 Ignore Parity

If this bit is set, the CL-CD1400 will ignore the parity on all incoming characters, thus not receive exception service requests will be generated if the parity is in error. If the bit is cleared, parity is evaluated.

Bit 3-2 Stop Bit Length

These two bits set the length, in bit times, of the stop bit for each character.

Stop1	Stop0	Number of Stop Bits
0	0	1 Stop bit
0	1	1.5 Stop bits
1	0	2 Stop bits
1	1	Not used

Bits 1-0 Character Length

ChL1 and ChL0 select the length of each character, in number of bits. The CL-D1400 receives and transmits the same length character, on a given channel, in the range of five to eight bits.

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)** **09** **Read/Write**

IXM	TxIBE	ETC	LLM	RLM	RtsAO	CtsAE	DsrAE

Bit 7 Implied X-ON Mode

The IXM bit enables the automatic resumption of character transmission upon the reception of any character other than the programmed X-OFF character. This bit only has meaning if the transmitter is in automatic in-band flow control mode as programmed by the TxIBE control bit. If this bit is reset and TxIBE is enabled, only the reception of an X-ON character will restart character transmission.

Channel Option Register 2 *(cont.)*

Bit 6 Enable Automatic In-band Transmit Flow-control

This bit enables the CL-CD1400's capability to examine error-free incoming characters looking for an X-OFF character (as programmed by SCHR2), if the special character match function is enabled (COR3, bit 4). If a match occurs, transmission will cease after the current characters in the transmitter shift register and transmitter holding register have been sent. Transmission will resume when an X-ON character (or any character, depending on the value of the IXM bit) is received or if a channel enable command is issued via the CCR.

Bit 5 Embedded Transmit Command Enable

If the ETC bit is set, the CL-CD1400 will examine characters in the transmit FIFO. If an embedded command is detected, it will be processed. See the embedded transmit command description in Section 3 for details of valid commands.

Bit 4 Local Loopback Mode

The LLM bit enables local loopback of the channel. This mode is generally used during system diagnostics. If this bit is set, the transmitter is internally "looped" back to the receiver and the TxD pins will be in the marking state. Data sent will be immediately received by the receiver. No data will appear on the TxD pin, and data on the RxD pin will be ignored.

Bit 3 Remote Loopback Mode

Remote loopback allows a remote system to test its serial data stream. If this function is enabled, the CL-CD1400 internally connects its receiver to the transmitter. Any data received is immediately echoed back. This mode is enabled by setting the RLM bit and disabled by clearing the bit.

Bit 2 Request To Send Automatic Output

The CL-CD1400 can automatically assert RTS when a channel is enabled (via transmit/receive enable command in the CCR). When the channel is disabled, RTS* will be negated. Setting RtsAO enables the function.

Bit 1 Clear To Send Automatic Enable

This bit enables the CTS* input to control transmitter operation. If CtsAE is set, and CTS is not asserted, character transmission will not proceed.

Bit 0 Data Set Ready Automatic Enable

As with CtsAE, DsrAE allows the DSR* input to control transmitter operation. Setting DsrAE enables the function.

Channel Option Register 3 **(COR3)** **0A** **Read/Write**

SCDRNG	SCD34	FCT	SCD12	RxTh3	RxTh2	RxTh1	RxTh0	Serial
0	0	0	RxTh4	RxTh3	RxTh2	RxTh1	RxTh0	Parallel

COR3 for Channel 0 has two formats, one for serial and one for parallel. Channels 3-1 have only the serial format. Both formats are described as follows:

Channel Option Register 3 – Serial

Bit 7 Special Character Detect Range

This bit enables range checking on received characters. If the character falls between a lower range set by the value stored in the SCRL register and an upper range set by the value stored in the SCRH register, inclusive, a receive exception service request will be posted with the status indicating a range detect (RDSR bits SCDet2-SCDet0 = 111).

Bit 6 Enable Special Character Detect on SCHR4-SCHR3

This bit controls whether or not the CL-CD1400 performs comparison on received characters against the values stored in registers SCHR4 and SCHR3. The comparison is enabled by a "1" in this location.

Bit 5 Flow Control Transparency

The FCT bit enables and disables transparent response to flow control characters received by the CL-CD1400. If FCT is set, received XON and XOFF characters will not be placed in the FIFO for the host. If in-band flow control is enabled, the characters will be acted upon. If FCT is not set, flow control characters will be acted upon, placed in the receive FIFO, and the host will be notified via a receive exception service request.

Bit 4 Enable Special Character Detect on SCHR2-SCHR1

This bit controls whether or not the CL-CD1400 compares received characters with the values stored in registers SCHR2 and SCHR1. A "1" enables compare. This bit must be set to enable automatic in-band flow control.

Bit 3-0 Receive FIFO Threshold

RxTh3	RxTh2	RxTh1	RxTh0	Receiver FIFO Threshold
0	0	0	0	Not used
0	0	0	1	1 Character
0	0	1	0	2 Characters
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
1	0	1	1	11 Characters
1	1	0	0	12 Characters
1	1	0	1	Not used
1	1	1	0	Not used
1	1	1	1	Not used

Channel Option Register 3 – Parallel

Bit 7-5 Not used
Bits 4-0 Receive FIFO Threshold

RxTh4	RxTh3	RxTh2	RxTh1	RxTh0	Receiver FIFO Threshold
0	0	0	0	0	Not used
0	0	0	0	1	1 Character
0	0	0	1	0	2 Characters
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
1	1	1	0	1	29 Characters
1	1	1	1	0	30 Characters
1	1	1	1	1	Not used

Channel Option Register 4 (COR4) 1E Read/Write

IGNCR	ICRNL	INLCR	IGNBRK	-BRKINT	PEH[2]	PEH[1]	PEH[0]

Bit 7-5 Carriage Return (CR) and New Line (NL) Processing
These three bits define the manner in which the CL-CD1400 will process received CR and NL characters (x'0D and x'0A). The table below shows the actions performed:

IGNCR	ICRNL	INLCR	Action
0	0	0	No action
0	0	1	Received NL changed to CR
0	1	0	Received CR changed to NL
0	1	1	Received CR changed to NL; NL changed to CR
1	0	0	Received CR discarded
1	0	1	Received CR discarded; NL changed to CR
1	1	0	Received CR discarded
1	1	1	Received CR discarded; NL changed to CR

Bit 4-3 Break Processing

The CL-CD1400 can handle received break characters in three ways:

IGNBRK	-BRKINT	Break Action
0	0	Received break generates an exception service request
0	1	Received break treated as a good NULL character
1	0	Not used
1	1	Received break discarded

Bit 2-0 Parity (P), Framing (F) and Overrun (O) Error Special Processing

As with break characters, the CL-CD1400 can treat errored characters in several different ways, if enabled:

PEH[2]	PEH[1]	PEH[0]	Action
0	0	0	Received P/F/O errored characters treated as exception data
0	0	1	Received P/F/O errored characters treated as good data
0	1	0	Received P/F/O errored characters discarded
0	1	1	Received P/F/O errored characters replaced with good NULL characters
1	0	0	Received P/F/O errored characters are replaced with the two character sequence x'FF-NULL. Good x'FF characters are replaced with the two character sequence x'FF-x'FF
1	0	1	Not used
1	1	0	Not used
1	1	1	Not used

Channel Option Register 5 **(COR5)** **1F** **Read/Write**

ISTRIP	LNE	CMOE	n/u	n/u	n/u	ONLCR	OCRNL

Channel Option Register 5 *(cont.)*

Bit 7 ISTRIP

The ISTRIP bit enables stripping of the most significant bit (bit7) on all received characters. A "1" in this position enables the function.

Bit 6 LNext Enable

When this bit is set, characters following an LNext character (as programmed by the LNC register) will not be processed as a special character.

Bit 5 Character Matching on Error

If this bit is set, character matching will occur on both good and errored characters. If the bit is cleared, matching will occur on good characters only.

Bits 4-2 Not used.

Bits 1-0 Carriage Return (CR) and New Line (NL) Processing – Transmit
These two bits define actions, if any, taken on characters in the transmit data stream.

ONLCR	OCRNL	Action
0	0	No action
0	1	Transmit CR changed to NL
1	0	Transmit NL changed to CR
1	1	Transmit CR changed to NL, NL changed to CRNL

Channel Control Status Register (CCSR) 0B Read Only

RxEN	RxFloff	RxFlon	n/u	TxEN	TXFloff	TxFlon	n/u	Serial

RxEN	n/u	n/u	n/u	TxEN	n/u	n/u	n/u	Parallel

The CCSR provides current status of the selected channel. The CCSR for channel 0 has two formats, one for serial operation and another for parallel operation.

Channel Control Status Register – Serial

Bit 7 Receiver Enabled

The RxEN bit is set when the receiver is enabled and cleared when it is disabled.

Bit 6 Receiver Flow Off

This bit indicates that the receiver has requested the remote to stop transmitting through the use of a send XOFF character via a send special character two command in the CCR. The bit will be cleared when a send special character one (XON) command is issued; the channel is either enabled or disabled or the channel is reset.

Bit 5 Receiver Flow On

When a send special character one (XON) command is issued via the CCR, this bit will be set. It will be cleared when one of three events has occurred: the first non-flow control character is received, the receiver is either enabled or disabled or the channel is reset.

Bit 4 Not used.

Bit 3 Transmitter Enabled
This bit is set when the transmitter is enabled and cleared when it is disabled.

Bit 2 Transmitter Flow Off

This bit indicates that the CL-CD1400 has been requested to stop transmission by the remote (received in-band flow control character XOFF). The bit is cleared when the CL-CD1400 requested to restart transmission (receives an XON character), the channel is either enabled or disabled or the channel is reset.

Bit 1 Transmitter Flow On

TxFlon is set when the CL-CD1400 has been requested to restart transmission (received an XON character). It is reset when transmission actually begins, when the channel is either enabled or disabled or when the channel is reset.

Bit 0 Not used.

Channel Control Status Register – Parallel

Bit 7 Receiver Enabled
The RxEN bit is set when the receiver is enabled and cleared when it is disabled.

Bits 6-4 Not used

Bit 3 Transmitter Enabled
This bit is set when the transmitter is enabled and cleared when it is disabled.

Bits 2-0 Not used

Received Data Count Register (RDCR) 0E Read Only

0	0	0	0	CT3	CT2	CT1	CT0	Serial

0	0	0	CT4	CT3	CT2	CT1	CT0	Parallel

Received Data Count Register *(cont.)*

The RDCR indicates the number of good characters currently in the received data FIFO. Host software can use this value as a loop counter when taking characters out of the FIFO. The value in this register is only valid during the context of a service request acknowledge. At other times, it may or may not give a true indication of the number of characters in the FIFO.

The register has two formats for channel 0, one for serial and one for parallel. Channels 1-3 have only the serial format.

Received Data Count Register – Serial

Bit 7-4 Always zero
Bits 3-0 Character count CT3-CT0

CT3	CT2	CT1	CT0	Number of characters in FIFO
0	0	0	0	Not used
0	0	0	1	1 Character
0	0	1	0	2 Characters
•	•	•	•	•
•	•	•	•	•
•	•	•	•	•
1	0	1	1	11 Characters
1	1	0	0	12 Characters
1	1	0	1	Not used
1	1	1	0	Not used
1	1	1	1	Not used

Received Data Count Register – Parallel

Bit 7-5 Always 0
Bits 4-0 Character count CT4-CT0

CT4	CT3	CT2	CT1	CT0	Number of Characters in FIFO
0	0	0	0	0	Not used
0	0	0	0	1	1 Character
0	0	0	1	0	2 Characters
•	•	•	•	•	•
•	•	•	•	•	•
•	•	•	•	•	•
1	1	1	0	1	29 Characters
1	1	1	1	0	30 Characters
1	1	1	1	1	Not used

Special Character Registers

The four special character registers, SCHR4-SCHR1, hold the character patterns that are used for various character matching and flow control functions. Each 8-bit character is right justified, that is, comparison takes place from right to left, and all bits are compared. Any unused bits must be zero. SCHR1 and SCHR2 serve the additional function of defining the XON and XOFF characters, respectively, used for in-band flow control.

Special Character Register 1 **(SCHR1)** **1A** **Read/Write**

Special Character 1

SCHR1 defines the XON character

Special Character Register 2 **(SCHR2)** **1B** **Read/Write**

Special Character 2

SCHR2 defines the XOFF character

Special Character Register 3 **(SCHR3)** **1C** **Read/Write**

Special Character 3

Special Character Register 4 **(SCHR4)** **1D** **Read/Write**

Special Character 4

Received Character Range Detection

If enabled (via bit 7 of COR3), the CL-CD1400 will check received characters to see if they fall within a range of values. Two registers set the range: SCRL and SCRH. Range checking occurs inclusive of the values programmed into these registers. If a received character is determined to be within the range, a special character detect exception service request will be posted. Bits 6-4 of the RDSR register will indicate a range detect by being set to 111. It should be noted that this range checking is performed in addition to normal special character detection on SCHR4-SCHR1.

Special Character Range Low **(SCRL)** **22** **Read/Write**

Character Range Low

SCRL set the lower inclusive value for range detection.

Special Character Range High **(SCRH)** **23** **Read/Write**

Character Range Low

SCRH sets the upper inclusive value for range detection.

LNext Character **(LNC)** **24** **Read/Write**

LNext Character

This register defines the LNext character. If the LNext function is enabled (bit 6 of COR5), the CL-CD1400 will examine received characters and compare them against this value. If a match occurs, this character and the following will be placed in the FIFO without any special processing. In effect, the LNext function causes the CL-CD1400 to ignore characters with special meaning, such as flow control characters. There are two exceptions. If the character following the LNext character is either a break or an errored character, LNext will be placed in the FIFO, and the following character will be treated as it normally would be for these error conditions.

Modem Change Option Registers

The CL-CD1400 has two registers that control its response to changes on the modem input pins. It can be programmed to respond to the low-to-high transition, the high-to-low transition or both. In addition, the threshold at which the DTR signal will be negated can be set by the DTRth3-DTRth0 bits in MCOR1.

Modem Change Option Register 1 **(MCOR1)** **15** **Read/Write**

DSRzd	CTSzd	Rlzd	CDzd	DTRth3	DTRth2	DTRth1	DTRth0	Serial

PBUSYzd	PSLCTzd	PPEzd	PERRORzd	0	0	0	0	Parallel

Channel 0 has two formats for MCOR1: one applies when the channel is in serial mode and the other applies in parallel mode, as set by the GCR. Channels 3-1 have only the serial mode format.

Modem Change Option Register 1 – Serial

Bit 7 DSRzd

Bit 6 CTSzd

Bit 5 RIzd

Bit 4 CDzd

Each of these bits controls its corresponding input pin. If the bit is set, the function is enabled and transitions from one-to-zero will generate an SVCREQM* service request.

Bits 3-0 DTRth3-DTRth0

These bits form a binary value that determines when the DTR output will be negated, based on the number of characters in the receive FIFO. When the FIFO holds more characters than this value, DTR will be negated, informing the remote that it should stop transmission. This value must be set to a value numerically larger than the value set for the receive FIFO threshold in COR3.

DTRth3	DTRth2	DTRth1	DTRth0	Number of characters in FIFO
0	0	0	0	Automatic DTR mode disabled
0	0	0	1	1 Character
0	0	1	0	2 Characters
•	•	•	•	•
•	•	•	•	•
•	•	•	•	•
1	0	1	1	11 Characters
1	1	0	0	12 Characters
1	1	0	1	Not used
1	1	1	0	Not used
1	1	1	1	Not used

Modem Change Option Register 1 – Parallel

Bit 7 PBUSYzd

Bit 6 PSLCTzd

Bit 5 PPEzd

Bit 4 PERRORzd

Effectively, these four bits are identical to the equivalent bits in the serial format above. The only difference is in the signal names. These inputs are renamed for convenience in working with the register when channel 0 is programmed to be a parallel port. Setting any of these bits will enable the detection of a one-to-zero transition on the corresponding input.

Bits 3-0 Not used; must be zero.

Modem Change Option Register 2 (MCOR2) 16 Read/Write

DSRod	CTSod	Rlod	CDod	0	0	0	0	Serial
PBUSYod	PSLCTod	PPEod	PERRORod	0	0	0	0	Parallel

Channel 0 has two formats for MCOR1: one applies when the channel is in serial mode and the other applies in parallel mode, as set by the GCR. Channels 3-1 have only the serial mode format.

Modem Change Option Register 2 – Serial

Bit 7 DSRod

Bit 6 CTSod

Bit 5 Rlod

Bit 4 CDod

Each of these bits controls its corresponding input pin. If the bit is set, the function is enabled and transitions from zero-to-one will generate an SVCREQM* service request.

Bits 3-0 These bits are not used and must be programmed to zero.

Modem Change Option Register 2 – Parallel

Bit 7 PBUSYod

Bit 6 PSLCTod

Bit 5 PPEod

Bit 4 PERRORod

Effectively, these four bits are identical to the equivalent bits in the serial format. The only difference is in the signal names. As with the parallel format of MCOR1, these inputs are renamed for convenience in working with the register when channel 0 is programmed to be a parallel port. Setting any of these bits will enable the detection of a zero-to-one transition on the corresponding input.

Bits 3-0 Not used, must be programmed to zero.

Receive Time-out Period Register (RTPR) 21 Read/Write

Binary Count Value

The RTPR determines the time period that will be used for the No New Data Time-out (NNDT) and the "no new data" time-out. The time-out counter is loaded from this register whenever a new character is placed in, or the last character is removed from, the receive FIFO. The counter is decremented on each "tick" of the prescaler counter (PPR). A service request will be generated if the count reaches zero; either an NNDT if the FIFO is empty and the NNDT is enabled, or a good data service request if there is data in the FIFO, but the time-out period has expired before the FIFO reaches the programmed threshold.

Modem Signal Value Register 1 (MSVR1) 6C Read/Write

DSR	CTS	RI	CD	PSTROBE*	0	0	RTS

Modem Signal Value Register 2 (MSVR2) 6D Read/Write

DSR	CTS	RI	CD	PSTROBE*	0	DTR	n/u

The MSVR1 and MSVR2 registers provide information regarding the state of the modem input pins (DSR*, CTS*, RI* and CD*), the current state of Printer Strobe output pin (PSTROBE*) and allows control of the modem output pins (DTR* and RTS*). The PSTROBE* bit is only valid for channel 0; on all other channels it is not used. Writing to any of the input bits has no effect. With the exception of the least significant two bits, the registers reflect identical data. The two are provided as a convenience for control of the modem output pins. Host software need not keep a copy of the current state of either when controlling the other. The actual signal level on the output is the inverse of the value placed in this register: setting the DTR bit, for example, will cause the DTR output to become active low. The state of the modem input pins also the inverse of the value in the corresponding bit in the registers.

Printer Signal Value Register **(PSVR)** **6F** **Read/Write**

PBUSY*	PSLCT*	PPE*	PERROR	PACK*	PAUTOFD	PINIT	PSLIN

This register groups all of the modem signals for channel 0 into one register. The PSVR is only valid for channel 0. The most significant five bits reflect the current signal value on the input pins PBUSY*, PSLCT*, PPE*, PERROR and PACK*. Note that PERROR is unique in that it reflects the same signal level as the actual input pin; the remainder of the bits reflect the inverse of the input signal values. The least significant three bits can be used by the host software to control the modem output pins PAUTOFD, PINIT and PSLIN. These bits directly control the outputs, and the values are not inverted.

Bit 7 Printer Busy – the current state of the printer busy input.
Bit 6 Printer Select – the current state of the printer select input.
Bit 5 Printer Paper Empty – the current state of the printer paper empty input.
Bit 4 Printer Error – the current state of the printer error input.
Bit 3 Printer Acknowledge – the current state of the printer acknowledge input.
Bit 2 Printer Autofeed – the current state of the printer autofeed output.
Bit 1 Printer Initialize – the current state of the printer initialize output.
Bit 0 Printer Selection – the current state of the printer selectin output.

Receive Baud Rate Period Register **(RBPR)** **78** **Read/Write**

Binary Divisor Value

This register holds the baud rate divisor for the receiver. It is used in conjunction with the Receive Clock Option Register (RCOR), which provides the clock that will be divided by this value. The time period produced must equal the value for one bit time of the receive data.

Receive Clock Option Register **(RCOR)** **7C** **Read/Write**

n/u	n/u	n/u	n/u	n/u	ClkSel2	ClkSel1	ClkSel0

The RCOR selects the clock source which will drive the baud rate period register (RBPR). The value in ClkSel2-ClkSel0 selects one of five possible clocks generated from the master clock (CLK).

ClkSel2	ClkSel1	ClkSel0	Clock Selected
0	0	0	clk0 (CLK divided by 8)
0	0	1	clk1 (CLK divided by 32)
0	1	0	clk2 (CLK divided by 128)
0	1	1	clk3 (CLK divided by 512)
1	0	0	clk4 (CLK divided by 2048)
1	0	1	Not used
1	1	0	Not used
1	1	1	Not used

Transmit Baud Rate Period Register (TBPR) 72 Read/Write

Binary Divisor Value

This register holds the baud rate divisor for the transmitter. It is used in conjunction with the Transmit Clock Option Register (TCOR), which provides the clock that will be divided by this value. The time period produced must equal the value for one bit time of the transmit data.

Transmit Clock Option Register (TCOR) 76 Read/Write

n/u	n/u	n/u	n/u	n/u	ClkSel2	ClkSel1	ClkSel0

The TCOR selects the clock source which will drive the baud rate period register (TBPR). The value in ClkSel2-ClkSel0 selects one of five possible clocks generated from the master clock (CLK).

Transmit Clock Option Register *(cont.)*

ClkSel2	ClkSel1	ClkSel0	Clock Selected
0	0	0	clk0 (CLK divided by 8)
0	0	1	clk1 (CLK divided by 32)
0	1	0	clk2 (CLK divided by 128)
0	1	1	clk3 (CLK divided by 512)
1	0	0	clk4 (CLK divided by 2048)
1	0	1	Not used
1	1	0	Not used
1	1	1	Not used

When channel 0 is programmed to be a parallel port, the TBPR/TCOR pair determine the pulse width of the PSTROBE* output. The resulting pulse width must not be shorter than the one-bit time of 100K baud.

6. CL-CD1400 PROGRAMMING

6.1 Overview

As shown in earlier sections, the CL-CD1400 host interface is made up of a large array of registers. These registers control aspects of chip behavior; some affect overall chip operations, and some affect only one channel. At first glance, this can appear to be a bewildering number of registers that need to be manipulated. However, most of the registers will only be set up once, during initialization, and only rarely modified during normal operation. The purpose of this section is to discuss these aspects, as well as the methods of interacting with the CL-CD1400 for channel service needs.

6.2 Initialization

In order to properly bring up a CL-CD1400, several procedures must be completed. These include chip initialization, programming global functions and setting channel-specific parameters. In most cases, initialization routines will only be executed once, during overall system boot-up. The following sections discuss these steps in detail. The flow chart on the next page presents this information in a visual format.

6.2.1 Chip Initialization

The procedures that perform chip reset will normally be executed after a power-up, system-wide reset and, therefore, the CL-CD1400 will have performed its own internal initialization, caused by the hardware reset control signal, RESET*. It is good practice, however, to issue a software chip reset anyway to make sure it has been completed before chip initialization begins. The following steps can be followed to accomplish this (following the text description, there is a flow chart version of the same steps):

1) Wait for CCR to contain 0x00

   The contents of the channel command register (CCR) must be zero before a command is issued. This is required so that any currently executing command has completed before the new one is started. Since this is probably the first command being written to the CL-CD1400 after power-on initialization, the CCR is likely to be zero, but it is good practice to always check the CCR before writing a new command into it.

2) Write hexadecimal 81 (x'81) to the Channel Command Register (CCR).

   This command causes the CL-CD1400 to perform an all-channel and global reset. Its effect is to cause the internal RISC processor to begin execution from its power-up reset location. All internal host interface registers are cleared, the FIFOs are flushed, and all channels are disabled.

   The all-channel reset command is a special case CCR operation. Normally, commands issued to the CCR affect only the channel selected by the CAR. In this case, the setting of the CAR is not significant.

3) Wait for the firmware revision code to be written into the GFRCR.

   This operation is used by the internal firmware to flag completion of the reset procedure. After reset, the GFRCR is one of the first registers to be cleared and is the last register set before normal run-time code execution begins. The initialization routine *must* wait for this register to become non-zero before beginning any other programming of CL-CD1400 registers. However, if the host code is sufficiently fast, it may begin testing the GFRCR before the MPU clears it; thus, the assumption made would be that the CL-CD1400 has completed its internal initialization when, in fact, it has not. In order to avoid this error, the host software should look for the GFRCR to change to a zero and then to the current revision code. Alternatively, the host can clear the GFRCR just prior to issuing the global reset and then poll for the correct revision code. This would be useful in slow systems that cannot guarantee that the host will be able to check the register after it has been cleared, and before it is loaded with the revision code.

This procedure can also be used as part of a diagnostic test suite. The device will complete internal initialization within 500 µsec. Therefore, a timer (software or hardware) can be used to detect that the operation does not complete within this time and that the chip may not be functional.

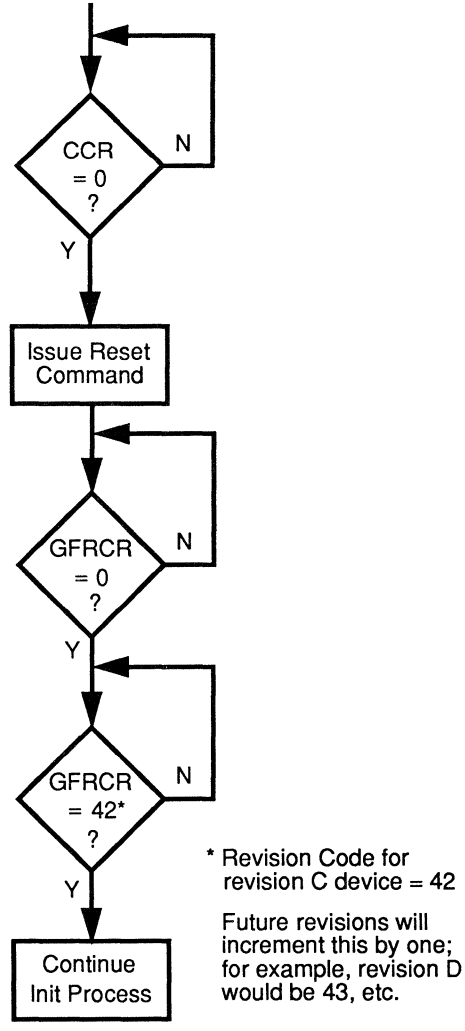


Figure 6–1. Flow Diagram of CL-CD1400 Master Initialization Sequence

6.2.2 Global Function Initialization

Once chip reset has been completed, the next step is to set the global operating mode and timer prescale. All other initialization will take place at the channel level.

1) Set the Global Configuration Register (GCR).

   The GCR setting determines the mode of operation of channel 0. After reset, this register is set to all '0's. This sets channel 0 to be a serial port. If this is the intended mode for channel 0, nothing further needs to be done with this register. If channel 0 will be used as a parallel port, bit 7 must be set to a '1'.

2) Set the Prescaler Period Register (PPR).

   The PPR sets the master time 'tick' for the CL-CD1400. It is a binary value that sets the constant by which the system clock is divided (after a fixed prescale of 512) to produce the internal clock for the on-chip timers (*not* the baud rate generators, however). This clock is used for receiver FIFO time-out generation and delay timing for the insert delay command in the embedded transmit command set. For example, to generate a timer clock of 1 ms, the value is computed as:

$$\left(\frac{20\,MHz}{512}\right) \times 1\,ms = 39.0625$$

The value 39 would be loaded into the PPR. This value, in effect, selects an approximate 1 KHz clock as the source for the Receiver Time-out Period Registers of each channel. Those registers would, in turn, be loaded with an appropriate value divisor to generate the desired character time-out periods. This value, 39, is the recommended minimum value that should be placed in the PPR. Values that generate a time period of less than 1 ms adversely affect the performance of the MPU and, thus, overall serial data performance.

6.2.3 Individual Channel Initialization

At this point, the basic operation of the CL-CD1400 has been set up. The internal register states have been cleared, the mode for channel 0 is set, and basic timer operations initialized. The next step is to program the operating modes of each channel. This includes setting the values for the interrupt vectors, the receive and transmit baud rates, number of bits per character, number of stop bits, parity, special characters, if any, etc. Each channel can have a completely unique set of operating characteristics or they can all be the same. It is application dependent; the operating modes of one channel have no effect on the operation of any other (operating channel 0 as a parallel port does affect the available input/output signals associated with channels 1-3 (CD and RI are borrowed) but not operating characteristics).

The following shows a typical initialization sequence to set up a single serial channel. In this example, channel 1 is set up as:

9600 Baud, send and receive
8 bits per character, 1 stop bit1.0 pt
No parity
Automatic In-Band (Xon/Xoff) flow control
Transparent flow control
Special character detect enabled
Eight character receive FIFO threshold
Receiver and transmitter enabled for interrupt operation
Enable ISTRIP on incoming characters

The clearest way to show this initialization sequence is via a 'C' program fragment; the code shown is compatible with Borland Turbo C™:

```
/* Init channel. Channel number is included in call. Register names and addresses are defined in
 * the header file (not shown).
 */

init_channel(chan)
char    chan
{
        outportb(CAR, chan);        /* set channel number in CAR */
        outportb(TCOR, 0x01);       /* constants for 20.2752-MHz clock – clock option*/
        outportb(TBPR, 0x42);       /*                                 – baud rate period */
        outportb(RCOR, 0x01);       /* constants for 20.2752-MHz clock – clock option*/
        outportb(RBPR, 0x42);       /*                                 – baud rate period */
        outportb(COR1, 0x03);       /* no parity, 1 stop bit, 8 bit chars */
        outportb(COR2, 0x40);       /* auto. in-band flow control */
        outportb(COR3, 0x38);       /* transp. flow-control, special char 1 & 2 detect, fifo thresh = 8 */
        while (inportb(CCR) != 0)   /* make sure that CCR is zero before issuing commands */
                ;
        outportb(CCR, 0x4E);        /* issue COR changed command for COR1, 2, 3 */
        outportb(COR5, 0x80);       /* enable ISTRIP */
        outportb(SRER, 0x14);       /* enable receive and transmit interrupts */
        while (inportb(CCR) != 0)   /* make sure that CCR is zero before issuing commands */
                ;
        outportb(CCR, 0x1A);        /* issue receiver and transmitter enable command to CCR */
}
```

6.3 Poll Mode Examples

The CL-CD1400 provides a set of seven registers that are dedicated to poll mode operation, as described in Section 4. This section shows one of many ways in which these registers can be used to detect and service requests from any of the channels receiver, transmitter or modem signal change functions.

The primary registers involved in polling are the SVRR, RIR, TIR, MIR and CAR; supplementary registers are the RIVR, TIVR and MIVR. Of the latter three, only the RIVR is actually used; it provides the status about whether the service request is for "good" data or exception data. The TIVR and MIVR provide redundant information and are rarely used. Other registers related to service requests (TDR, RDSR, MISR, etc.) perform the same functions as they would in a hardware acknowledged service request.

Once again, "C" code fragments will be used to describe the functions. As with other coding examples, it is assumed that register addresses are defined elsewhere, such as in a header file, and are not shown here. Also, the routines cannot be considered complete. Some pieces will be dependent on the system software design so liberties are taken in the examples. They do, however, show methods that can be used to implement the poll mode service request/service acknowledge sequence.

6.3.1 Polling Routine Examples

6.3.1.1 Scanning Loop

```
/* Poll mode code fragment. This routine simply checks for any servicing requests and
 * branches to the appropriate service routine. The code prioritizes service requests as
 * receive, transmit and modem, in that order.
 */

poll( )
{
        char    status;
        char    rx_stat = tx_stat = md_stat = 0;

        if (status = inportb(SVRR)) {
                switch (status) {
                        case 1:                         /* all values that include a receive request */
                        case 3:
                        case 5:
                        case 7:
                                rx_stat = service_rec( );
                                return(rx_stat);
                                break;
                        case 2:                         /* all values that include transmit but not receive */
                        case 6:
                                tx_stat = service_txm( );
                                return(tx_stat);
                                break;
                        case 4:                         /* modem service request alone */
                                md_stat = service_mdm( );
                                return(md_stat);
                                break;
                        default:                        /* can't happen :-) */
                                break;
                }
        }
}
```

Once the code above finds an active request posted in the SVRR, it calls the appropriate subroutine to service the request. The service routines follow.

6.3.1.2 Receive Service

```
/* The receive service acknowledge cycle begins by reading the RIR. This register contains the
 * necessary information to switch the CL-CD1400 into the correct service acknowledge context. The
 * RIR is saved for use at the end of the routine and then copied into the CAR. The act of copying the
 * RIR into the CAR forces the context switch. The channel number requesting service is extracted from
 * the RIR. The RIVR register indicates whether the request is for good data or exception data
 * and is used to correctly handle the request. At the end of the service, the upper two bits in the
 * RIR are cleared causing the switch out of the service acknowledge context.
 */

service_rec( )
{
        char    serv_type, save_rir, save_car, channel, status, char;
        int     char_count, i;

        save_rir = inportb(RIR);                    /* retrieve and save receive interrupt value */
        channel = save_rir & 0x03;                  /* extract channel number from the RIR*/
        save_car = inportb(CAR);                    /* save CAR for restore */
        outportb(CAR, save_rir);                    /* switch CL-CD1400 to service ack. context */
        serv_type = inportb(RIVR) & 0x07;           /* read vector register; get type (good/exception)*/
        switch (serv_type) {
                case 3:                             /* good data service */
                        char_count = inportb(RDCR);         /* get number of characters in FIFO */
                        for ( i = 1; i <= char_count; i++) {    /* - read that number of chars */
                                char = inportb(RDSR);       /* read char from FIFO */

                        /* Code here would put the character in a buffer of some sort for each
                         * channel. That code would be dependent on system software design
                         * so it won't be shown here. */

                        }
                        outportb(RIR, save_rir & 0x3f);     /* terminate service ack. sequence */
                        outportb(CAR, save_car);            /* restore original CAR* /
                        return(0);
                        break;
                case 7:                             /* exception data service request */
                        status = inportb(RDSR);     /* by definition, only one char; get status */
                        outportb(RIR, save_rir & 0x3f);     /* terminate service ack. sequence */
                        outportb(CAR, save_car);            /* restore original CAR */
                        return(status);             /* just return the error type */
                        break;
        }
}
```

6.3.1.3 Transmit Service

```
/* The transmit service acknowledge routine follows very nearly the same steps that the receive
 * service routine follows. This time, the TIR is used to force the switch to a transmit service for
 * the requesting channel.
 */

service_txm( )
{
        char    save_tir, save_car, channel;
        int     char_count, i;

        save_tir = inportb(TIR);                    /* retrieve and save transmit interrupt value */
        channel = save_tir & 0x03;                  /* extract channel number from the TIR*/
        save_car = inportb(CAR);                    /* save CAR for restore */
        outportb(CAR, save_tir);                    /* switch CL-CD1400 to service ack. context */

                /* Buffer management code would set-up pointers to the next 12
                 * characters (maximum) to be sent on this channel. Again, buffer
                 * layout is system design dependent and won't be shown here.
                */

        for ( i = 0; i < char_count; i++) {         /* transmit FIFO can take 12 characters */
                outportb(TDR, *next_char++);

                /* it is assumed that char_count and next_char is set up by buffer code */

        }

        outportb(TIR, save_tir & 0x3f);     /* terminate service ack. sequence */
        outportb(CAR, save_car);            /* restore original CAR; may not be necessary*/
        return(0);
}
```

6.3.1.4 Modem Service

```
/* Code to handle modem signal change service request can be simple or complex depending
 * on whether port control is handled directly in the service routine or simply noted with status
 * returned. The following routine services the request and returns the status of which signals
 * changed with the channel number OR'ed into the least significant two bits; the main driver
 * software must perform the necessary functions. As with the receive and transmit routines,
 * the interrupt register, this time the MIR, is used to force the CL-CD1400 into the service context.
 */

service_mdm( )
{
        char    save_mir, channel, save_car, mdm_status;

        save_mir = inportb(MIR);                /* retrieve and save modem interrupt value */
        channel = save_mir & 0x03;              /* extract channel number from the MIR*/
        save_car = inportb(CAR);                /* save CAR for restore */
        outportb(CAR, save_mir);                /* switch CL-CD1400 to service ack. context */
        mdm_status = inportb(MISR);             /* get status of which modem signals changed */
        outportb(MIR, save_mir & 0x3f)          /* terminate the service ack. sequence */
        outportb(CAR, save_car);                /* restore CAR */
        return(mdm_status | channel);
}
```

6.4 Hardware-Activated Service Examples

In nearly all respects, the way in which the hosts interacts with the CL-CD1400 during hardware-activated service acknowledge is the same as for the software-activated methods. The main difference is that the SVCACK* input signals perform the context switch automatically, thus relieving that duty from the host. The result is the same; the CAR is set to point to the correct channel, and the chip is placed in the proper internal mode to service the request. When the host activates the SVCACK* input, a read cycle is performed. The CL-CD1400 places the contents of the appropriate interrupt vector reigster (RIVR, TIVR, MIVR) of the channel requesting service on the data bus. The host uses the information provided to determine the type of service and the ID number of the device being accessed in the case of multiple CL-CD1400s daisy-chained together. At the end of the service routine, the host writes a dummy value to the EOSRR register. This causes the switch out of the service acknowledge context and restores the environment to what it was before the service began.

The following code fragments show the differences between this type of service acknowledge and those shown above for the software-activated context switch. Only the beginning and ending steps are shown; the code in between would be very similar to the previous examples. These routines could be executed as the result of a hardware interrupt or via software polling, as in the previous examples. For purposes of this discussion, the method of arriving at the proper service routine is not important.

6.4.1 Receive Service

```
/* The receive service acknowledge cycle begins by executing a service acknowledge cycle that
 * activates the SVCACKR* input. The data obtained as a result of this "read" cycle is the content
 * of the LIVR register of the channel making the service request. The service routine decodes the
 * vector in the least significant three bits to determine if the data is "good" or "bad" (exception).
 * The context switch was done automatically when the SVCACKR* signal was activated so the
 * CAR does not need to be loaded. The routine reads the RICR to determine the requesting
 * channel number. If this were a multiple-CL-CD1400 system using daisy-chaining, the routine would
 * extract the chip ID from the upper five bits of the RIVR.
 */

service_rec( )
{
        char    serv_type, vector, channel, status, char;
        int     char_count, i;

        vector = inportb(SVCACKR);              /* gen. ack and get vector (read LIVR) */
        channel = inportb(RICR) >> 2;           /* extract channel number from the RICR*/
        serv_type = vector & 0x07;              /* mask RIVR to get type (good/exception)*/
        switch (serv_type) {
                case 3:                         /* good data service */
                        char_count = inportb(RDCR);     /* get number of characters in FIFO */
                        for ( i = 1; i <= char_count; i++) {    /* - read that number of chars */
                                char = inportb(RDSR);   /* read char from FIFO */

                        /* Code here would put the character in a buffer of some sort for each
                         * channel. That code would be dependent on system software design
                         * so it won't be shown here; this code just shows how to manipulate the
                         * CL-CD1400 registers to implement the poll mode service acknowledge. */

                        }
                        break;
                case 7:                         /* exception data service request */
                        status = inportb(RDSR); /* by definition, only one char; get status */
                        break;
        }
        outportb(EOSRR, 0x00);                  /* write dummy value to EOSRR to terminate */
}
```

6.4.2 Transmit Service

```
/* The transmit service acknowledge routine follows very nearly the same steps that the receive
 * service routine follows. The SVCACKT* input is activated to start the service cycle and the TICR
 * is read to get the channel number.
 */

service_txm( )
{
        char    vector, channel;
        int     char_count, i;

        vector = inportb(SVCACKT);              /* retrieve and save transmit interrupt value */
        channel = inportb(TICR) >> 2;           /* extract channel number from the RICR*/

                /* Buffer management code would set-up pointers to the next 12
                 * characters (maximum) to be sent on this channel. Again, buffer
                 * layout is system design dependent and won't be shown here.
                */

        for ( i = 0; i < char_count; i++) {     /* transmit FIFO can take 12 characters */
                outportb(TDR, *next_char++);

                /* it is assumed that char_count and next_char is set up by buffer code */

        }
        outportb(EOSRR, 0x00);                  /* write dummy value to EOSRR to terminate */
}
```

6.4.3 Modem Service

```
/* The following routine services the modem change service request. Context switch is set up by
 * activating the SVCACKM* input, reading the MIVR. Channel status is an externally defined variable
 * that this routine updates.
 */

service_mdm( )
{
        char    vector, channel;

        vector = inportb(SVCACKM);              /* retrieve and save transmit interrupt value */
        channel = inportb(MICR) >> 2;           /* extract channel number from the RICR*/
        mdm_status[channel] = inportb(MISR);    /* get status of which modem signals changed */
        outportb(EOSRR, 0x00);                  /* write dummy value to EOSRR to terminate */
}
```

6.5 Baud Rate Tables

The tables on the following three pages show the values that need to be loaded into the RCOR/RBPR and TCOR/TBPR registers to set the designated baud rate when using three standard frequency crystals. The first one uses a 20.2752-MHz frequency which yields near-perfect bit rates. The second table uses a 20-MHz frequency and shows error rates that are a little larger although still well within the limits set by the various standards covering asynchronous communications. The third table also uses another standard communications base frequency (18.432 MHz) that yields divisors with nearly zero errors overall. However, since this frequency is below 20 MHz, performance at the higher baud rates (76.8K and 115.2K) may be slightly lower. It is, of course, not necessary that both the receiver and transmitter of a channel be programmed to the same baud rate; the CL-CD1400 can send and receive at different rates on the same channel.

Baud Rate Constants, CLK = 20.2752 MHz

Baud Rate	RCOR/TCOR	RBPR/TBPR (Hex)	Error
110	4	5A	0.00%
150	4	42	0.00%
300	3	84	0.00%
600	3	42	0.00%
1200	2	84	0.00%
2400	2	42	0.00%
4800	1	84	0.00%
9600	1	42	0.00%
19200	0	84	0.00%
38400	0	42	0.00%
56000	0	2D	0.57%
57600	0	2C	0.00%
64000	0	28	1.00%
76800	0	21	0.00%
115200	0	16	0.00%

Baud Rate Constants, CLK = 20.00 MHz

Baud Rate	RCOR/TCOR	RBPR/TBPR (Hex)	Error
110	4	59	0.25%
150	4	41	0.16%
300	3	82	0.16%
600	3	41	0.16%
1200	2	82	0.16%
2400	2	41	0.16%
4800	1	82	0.16%
9600	1	41	0.16%
19200	0	82	0.16%
38400	0	41	0.16%
56000	0	2D	0.79%
57600	0	2B	0.94%
64000	0	27	0.16%
76800	0	21	1.36%
115200	0	16	1.36%

Baud Rate Constants, CLK = 18.432 MHz

Baud Rate	RCOR/TCOR	RBPR/TBPR (Hex)	Error
110	4	52	0.22%
150	3	F0	0.00%
300	3	78	0.00%
600	2	F0	0.00%
1200	2	78	0.00%
1800	2	50	0.00%
2000	2	48	0.00%
2400	1	F0	0.00%
4800	1	78	0.00%
9600	0	F0	0.00%
19200	0	78	0.00%
38400	0	3C	0.00%
56000	0	29	0.35%
57600	0	28	0.00%
64000	0	24	0.00%
76800	0	1E	0.00%
115200	0	14	0.00%

6.6 ASCII Code Table

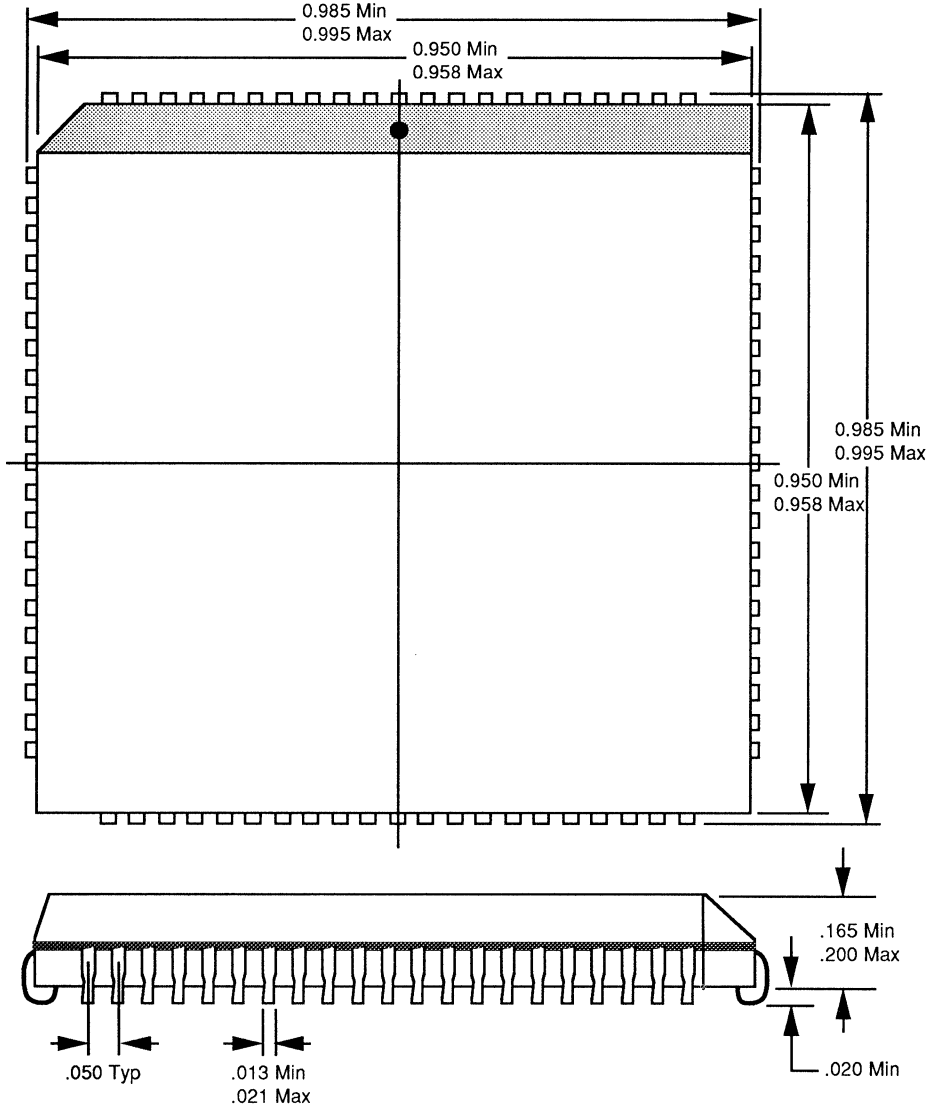
6.6.1 Hexadecimal - Character

00	NUL	01	SOH	02	STX	03	ETX	04	EOT	05	ENQ	06	ACK	07	BEL
08	BS	09	HT	0A	NL	0B	VT	0C	NP	0D	CR	0E	SO	0F	SI
10	DLE	11	DC1	12	DC2	13	DC3	14	DC4	15	NAK	16	SYN	17	ETB
18	CAN	19	EM	1A	SUB	1B	ESC	1C	FS	1D	GS	1E	RS	1F	US
20	SP	21	!	22	"	23	#	24	$	25	%	26	&	27	'
28	(	29	)	2A	*	2B	+	2C	,	2D	-	2E	.	2F	/
30	0	31	1	32	2	33	3	34	4	35	5	36	6	37	7
38	8	39	9	3A	:	3B	;	3C	<	3D	=	3E	>	3F	?
40	@	41	A	42	B	43	C	44	D	45	E	46	F	47	G
48	H	49	I	4A	J	4B	K	4C	L	4D	M	4E	N	4F	O
50	P	51	Q	52	R	53	S	54	T	55	U	56	V	57	W
58	X	59	Y	5A	Z	5B	[	5C	\	5D	]	5E	^	5F	_
60	~	61	a	62	b	63	c	64	d	65	e	66	f	67	g
68	h	69	i	6A	j	6B	k	6C	l	6D	m	6E	n	6F	o
70	p	71	q	72	r	73	s	74	t	75	u	76	v	77	w
78	x	79	y	7A	z	7B	{	7C	|	7D	}	7E	_	7F	DEL

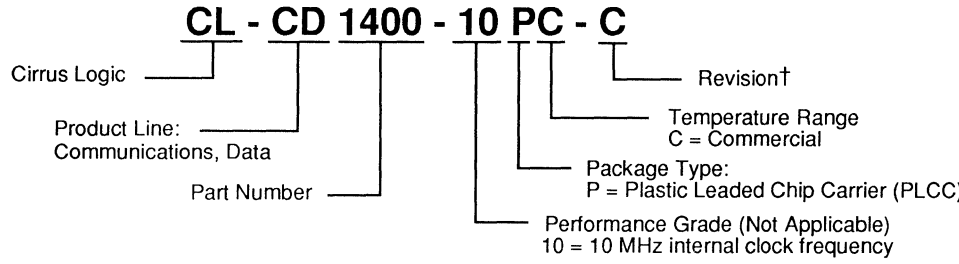
6.6.2 Decimal - Character

0	NUL	1	SOH	2	STX	3	ETX	4	EOT	5	ENQ	6	ACK	7	BEL
8	BS	9	HT	10	NL	11	VT	12	13	13	CR	14	SO	15	SI
16	DLE	17	DC1	18	DC2	19	DC3	20	DC4	21	NAK	22	SYN	23	ETB
24	CAN	25	EM	26	SUB	27	ESC	28	FS	29	GS	30	RS	31	US
32	SP	33	!	34	"	35	#	36	$	37	%	38	&	39	'
40	(	41	)	42	*	43	+	44	,	45	-	46	.	47	/
48	0	49	1	50	2	51	3	52	4	53	5	54	6	55	7
56	8	57	9	58	:	59	;	60	<	61	=	62	>	63	?
64	@	65	A	66	B	67	C	68	D	69	E	70	F	71	G
72	H	73	I	74	J	75	K	76	L	77	M	78	N	79	O
80	P	81	Q	82	R	83	S	84	T	85	U	86	V	87	W
88	X	89	Y	90	Z	91	[	92	\	93	]	94	^	95	_
96	~	97	a	98	b	99	c	100	d	101	e	102	f	103	g
104	h	105	i	106	j	107	k	108	l	109	m	110	n	111	o
112	p	113	q	114	r	115	s	116	t	117	u	118	v	119	w
120	x	121	y	122	z	123	{	124	|	125	}	126	_	127	DEL

7. SAMPLE PACKAGE – 68-pin PLCC



8. ORDERING INFORMATION



† Contact CIRRUS LOGIC for up-to-date information on revisions

Notes

CIRRUS LOGIC

Notes

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
Boulder, CO
TEL: 303/939-9739
FAX: 303/442-6388

NORTH CENTRAL AREA
Westchester, IL
TEL: 708/449-7715
FAX: 708/449-7804

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

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

Philadelphia, PA
TEL: 215/251-6881
FAX: 215/651-0147

SOUTH EASTERN AREA
Boca Raton, FL
TEL: 407/994-9883
FAX: 407/994-9887

Atlanta, GA
TEL: 404/263-7601
FAX: 404/729-6942

International

GERMANY
Herrsching
TEL: 49/8152-2030
FAX: 49/8152-6211

JAPAN
Kanagawa-Ken
TEL: 81/462-76-0601
FAX: 81/462-76-0291

SINGAPORE
TEL: 65/3532122
FAX: 65/3532166

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

UNITED KINGDOM
Berkshire, England
TEL: 44/344-780-782
FAX: 44/344-761-429

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 proprietary S/LA™ † IC design automation with system design expertise. The S/LA design system is a proven tool for developing high-performance logic circuits in half the time of most semiconductor companies. The results are better VLSI products, on-time, that 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.

† U.S. Patent No. 4,293,783

© Copyright, Cirrus Logic, Inc., 1991

Preliminary product information describes products which are in production, but for which full characterization data is not yet available. Cirrus Logic, Inc. believes the information contained in this document is accurate and reliable. However, it is marked *Preliminary* and is subject to change without notice. No responsibility is assumed by Cirrus Logic, Inc. for its use, nor for infringements of patents or other rights of third parties. This document implies no license under patents or copyrights. Trademarks in this document belong to their respective companies. Cirrus Logic, Inc. products are covered under one or more of the following U.S. patents: 4,293,783; Re. 31,287; 4,763,332; 4,777,635; 4,839,896; 4,931,946; 4,979,173.

CIRRUS LOGIC, Inc., 3100 West Warren Ave. Fremont, CA 94538
TEL: 415/623-8300 FAX: 415/226-2160