CIRRUS LOGIC®

CL-CD1283

Preliminary Data Book

FEATURES

Parallel Port

High-speed, bidirectional, multi-protocol parallel port:

- **Hardware implementation of all modes of the IEEE STD (Standard) 1284 specification (including automatic negotiation):**
  - — Centronics®-compatible mode
  - — Reverse-Byte mode
  - — Reverse-Nibble mode
  - — ECP (extended capabilities port) mode with run-length encoding/decoding
  - — EPP (enhanced parallel port) mode
  - — Up to 2-Mbyte/second transfer rate in ECP and EPP modes
- **64-byte parallel FIFO with DMA interface**
- **Supports peripheral-side operation**
- **Data and control input/output pads support IEEE STD1284 level-2 interface specification**
- **Host bus interface**
  - — High-speed slave DMA handshake interface
  - — DMA transfers are three clocks per word
  - — On-the-fly data compression using RLE (run-length encoded) encoding and decoding
  - — 8/16-bit data interface
  - — Byteswap input provides easy interface to both Big- and Little-Endian systems
  - — Vectored interrupt simplifies interrupt service routines

General

- **System clock up to 25 MHz**
- **CMOS technology enables high speed and low power**
- **Available in a 100-pin PQFP package**

IEEE 1284-Compatible Parallel Interface Controller

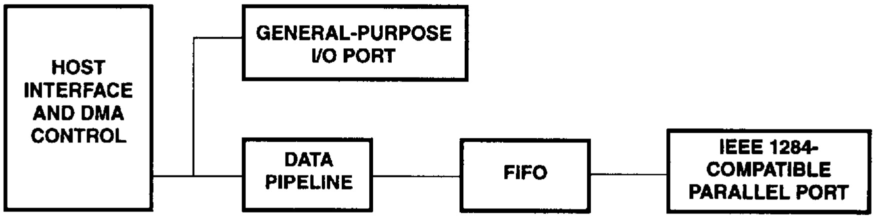
OVERVIEW

The CL-CD1283 is a multi-function interface controller for printer, scanner, and tape-drive applications that implement a high-speed, multi-protocol parallel port. In addition, the CL-CD1283 has both programmed-I/O and DMA (direct-memory access) operation, providing flexibility in host interface design and high-speed data transfers between the device and host memory.

The parallel port implements all modes of the IEEE STD 1284 standard signaling method for bidirectional parallel peripheral interface specification including: ECP, EPP, reverse byte, reverse nibble, and compatible. Data transfer rates of up to 2 Mbytes/second are achievable on the parallel port when the device is operated at its maximum rated clock of 25 MHz. This data rate is supported on the host side via a 64-byte FIFO and DMA interface. The parallel-port data and control signals implement the IEEE STD 1284-defined, level-2 electrical interface specification in symmetrical drive type, current capability (±12 mA), slew rate (0.4 V/ns) and hysteresis (0.8 V) (–2.0 to +7.0 V protection is not implemented).

(cont. next page)

Functional Block Diagram


HOST INTERFACE AND DMA CONTROL
GENERAL-PURPOSE I/O PORT
DATA PIPELINE
FIFO
IEEE 1284-COMPATIBLE PARALLEL PORT

Also included is a general-purpose port that provides eight bits of individual direction programmable I/O that can be used for status and control of external functions.

THEORY OF OPERATION

The CL-CD1283 is an efficient, high-performance communications controller using dedicated logic to implement the parallel channel. The controller offloads much of the sending and receiving of high-speed data from the local host CPU. Built-in features, such as automatic encoding and decoding of RLE compressed data, simplify local CPU routines and increases overall system performance.

The parallel channel has a single 64-byte FIFO to support the high speeds obtainable on the parallel-data port. The FIFO has a programmable threshold in both the receive and transmit directions. The 64-byte-deep FIFO reduces both the number of interrupt requests made of the CPU and the time required to service them. The time required to service the requests is reduced by having the ability to provide unique vectors for internal interrupt conditions; thus the system spends less time determining the source of the interrupt.

The parallel channel in the CL-CD1283 implements all protocols defined for the peripheral side by the IEEE STD 1284 titled *Standard Signaling Method for a Bidirectional Parallel Peripheral Interface for Personal Computers.* This specification defines four bidirectional protocols that allow a peripheral device to communicate with a host system (IBM® PC type or equivalent) via the parallel printer channel. The protocols, or modes include Reverse Nibble, Reverse Byte (IBM PS/2® style), ECP, and EPP (as implemented on the Intel® 80386SL processor). ECP and EPP both operate at data rates as high as 2 Mbytes/second.

The IEEE STD 1284-compatible port is implemented as two functional blocks: a data pipeline, which includes the 64-byte FIFO and the DMA interface (a high-speed state-machine), which controls the parallel port and implements the IEEE STD 1284 protocols.

While in ECP mode, the CL-CD1283 provides RLE data compression in both directions, as defined in the IEEE specification. If enabled, this data compression is performed automatically and is capable of compressing long strings (up to 128 bytes) of identical data into a 2-byte sequence (command/count and data). In printer applications, this can greatly reduce data transmission time since bit patterns have large amounts of identical data.

The EPP mode defines a means of sending address and data over the parallel channel (much like a processor address and data interface). This is used in LAN and SCSI interface adapters in laptop computers.

ADVANTAGES

Unique Features	Benefits
Parallel Port —	
■ Supports IEEE STD 1284 specification	❒ Multi-protocol bidirectional port for a wide range of applications. — Up to 2 Mbytes/second speed — Provides future connectivity with new host systems
■ 64-byte FIFO	❒ High throughput with reduced host load.
■ Parallel port pads provide level-2 drive characteristics	❒ Direct connection to printer cable, and reduced chip count.
■ DMA channel	❒ Reduced host interface overhead. High-speed data movement between memory and parallel port.
General-Purpose I/O —	
■ Byte-wide, direction-programmable port	❒ Added control and status provides flexible system design.

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

Table of Contents

Table of Contents *(cont.)*

List of Figures

List of Tables

CONVENTIONS

This section lists conventions used in this data book.

Abbreviations

Units of measure	Symbol
degree Celsius	°C
hertz (cycle per second)	Hz
kilobyte (1,024 bytes)	Kbyte
kilohertz	kHz
kilohm	kΩ
megabyte (1,048,576 bytes)	Mbyte
megahertz (1,000 kilohertz)	MHz
microfarad	µF
microsecond (1,000 nanoseconds)	µs
milliampere	mA
millisecond (1,000 microseconds)	ms
nanosecond	ns
picovolt	pV
volt	V
watt	W

The use of 'tbd' indicates values that are 'to be determined', 'n/a' designates 'not available', and 'n/c' indicates a pin that is a 'no connect'.

Acronyms

The following table lists acronyms used in this data book.

Acronym	Definition
AC	alternating current
BIOS	basic input/output system
CMOS	complementary metal-oxide semiconductor
DC	direct current
DMA	direct-memory access
DRAM	dynamic random-access memory
ECP	extended capabilities port
EEPROM	electrically erasable/programmable read-only memory
EISA	extended industry standard architecture
EPP	enhanced parallel port

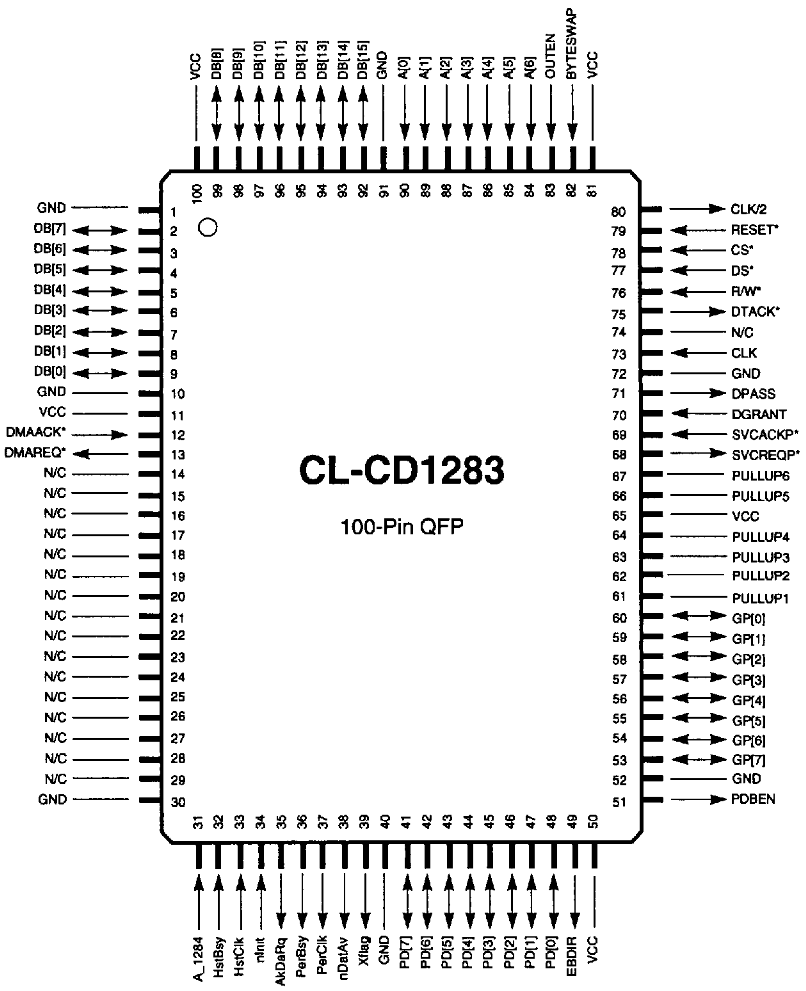
Acronym	Definition *(cont.)*
EPROM	electrically programmable read-only memory
FIFO	first in/first out
HSYNC/VSYNC	horizontal/vertical synchronization
ISA	industry standard architecture
LSB	least-significant bit
MSB	most-significant bit
PCI	peripheral component interconnect
PQFP	plastic quad-flat pack
RAM	random-access memory
RAS	row address strobe
RGB	red, green, blue
RLE	run-length encoded
R/W	read/write
TTL	transistor-transistor logic
VRAM	video random-access memory

Numeric Naming

Hexadecimal numbers are represented with all letters in uppercase and a lowercase 'h' is appended to them (for example, '14h', '3A7h', and 'C000h' are hexadecimal numbers). Numbers not indicated by an 'h' are decimal.

1. PIN INFORMATION

1.1 Pin Diagram



NOTE: (*) Denotes active-low signal. N/C indicates 'no connection'; make no connection to these pins.

2136639 0009019 614

1.2 Pin List

The following naming conventions are used in the pin-assignment tables:

- (*) after a name indicates that the signal is active-low
- 'I' indicates the pin is input-only
- 'O' indicates the pin is output-only
- 'I/O' indicates the pin is bidirectional
- 'OD' indicates an open-drain output that the user must tie to V_{CC} through a pull-up resistor (usually about 1 kΩ)
- 'PU' indicates pull-up, which must also be tied to V_{CC} through a 1-kΩ resistor (note that all six PU pins can be wire-OR'ed through the same pull-up resistor)
- 'AR' indicates active release (pin drives high and releases to OD)
- a '–' indicates ascending pin numbers
- a ':' indicates descending pin numbers

Pin name	Type	Number of pins	Pin numbers
VCC	I	5	11, 50, 65, 81, 100
GND	I	7	1, 10, 30, 40, 52, 72, 91
RESET*	I	1	79
OUTEN	I	1	83
CLK	I	1	73
CLK/2	O	1	80
DB[15:0]	I/O	16	92–99, 2–9
A[6:0]	I	7	84–90
R/W*	I	1	76

Pin name	Type	Number of pins	Pin numbers
CS*	I	1	78
DS*	I	1	77
BYTESWAP	I	1	82
DTACK*	AR	1	75
DMAREQ*	O	1	13
DMAACK*	I	1	12
PULLUP1	PU	1	61
PULLUP2	PU	1	62
PULLUP3	PU	1	63
PULLUP4	PU	1	64
SVCREQP*	OD	1	68
SVCACKP*	I	1	69
PULLUP5	PU	1	66
PULLUP6	PU	1	67
DGRANT	I	1	70
DPASS	O	1	71
PD[7:0]	I/O	8	41–48
GP[7:0]	I/O	8	53–60
A_1284	I	1	31
HstBsy	I	1	32
HstClk	I	1	33
nInit	I	1	34
AkDaRq	O	1	35
PerBsy	O	1	36
PerClk	O	1	37
nDatAv	O	1	38
Xflag	O	1	39
EBDIR	O	1	49
PDBEN	O	1	51

1.3 Pin Description

Symbol	Pin No.	Type	Description
A[6:0]	84–90	I	**ADDRESS BUS:** Together with CS* or one of the SVCACK* inputs and DS*, this input selects an on-chip register for a read or write operation or an acknowledgment to a service request.
BYTESWAP	82	I	**BYTESWAP:** This input determines the byte order for 2-byte DMA transfers and for writes to the DMA Buffer register. If BYTESWAP is set to a '1', Data Bus bits [15:8] are driven with the byte transferred first on the parallel port bus. Data Bus bits [7:0] are driven with the byte transferred second on the parallel port bus. If BYTESWAP is set to a '0', the data order is reversed, bits [7:0] are driven with the byte transferred first, and bits [15:8] are driven with the byte transferred second.
CLK	73	I	**SYSTEM CLOCK:** This input has a 25 MHz maximum; 16 MHz is the recommended minimum for satisfactory device performance.
CLK/2	80	O	**SYSTEM CLOCK DIVIDED BY TWO OUTPUT:** This signal is equivalent to the internal operating clock of the device.
CS*	78	I	**ACTIVE-LOW CHIP SELECT:** When active, the input CS* in conjunction with DS*, initiates a I/O cycle with the CL-CD1283. CS* must be set to a '1' during DMA read/write operations.
DB[15:0]	92–99, 2–9	I/O	**BIDIRECTIONAL DATA BUS:** Only DMA transfers and writes to the DMA Buffer register are true 16-bit operations. During all register writes other than to the DMA Buffer register, only bits [7:0] are written to the addressed register. Register reads duplicate the register contents on both the lower byte [7:0] and upper byte [15:8].
DS*	77	I	**ACTIVE-LOW DATA STROBE:** During an active I/O cycle, the input DS* strobes data into on-chip registers on write cycles or enables data onto the data bus during read cycles. DS* is ignored during DMA operations.
DTACK*	75	AR	**ACTIVE-LOW DATA TRANSFER ACKNOWLEDGE:** This output indicates: when the device has completed the requested I/O operation, and when the cycle may finish. This signal can be used to implement wait-state insertion for the local CPU. It is an Active Release output, driving to logic '1' then releasing to OD. DTACK* must be tied to external V_{CC} via a pull-up resistor.

Symbol	Pin No.	Type	Description
OUTEN	83	I	**OUTPUT ENABLE:** This pin must be set to a '1' to enable output pin functions. When OUTEN is set to a '0', if forces all pins that can act as outputs to remain in a three-state condition. This is used as a test input and is not normally used in an end application. User designs should tie this pin to V_{CC} through a pull-up resistor.
RESET*	79	I	**ACTIVE-LOW RESET INPUT:** Initializes the device to the default condition. All internal registers are set to their reset condition and all transfer operations are set to the default state.
R/W*	76	I	**READ/WRITE*:** Must be set to a '1' for a register read operation, must be set to a '0' for a register write. This input is ignored for DMA operations.
DMAREQ*	13	O	**ACTIVE-LOW DMA REQUEST:** When the internal control bit DMAen is set, the output DMAREQ* is asserted whenever internal FIFO conditions warrant a DMA transfer. DMAREQ* is deasserted on the falling edge of DMAACK* when DMA transfers must not continue past the current transfer.
DMAACK*	12	I	**ACTIVE-LOW DMA ACKNOWLEDGE:** This signal must never be asserted unless in response to a DMAREQ* from the chip. DMAACK* is the only bus handshake signal recognized during a DMA transfer. (CS* must be high whenever DMAACK* is asserted). The direction of DMA transfer is determined by internal control bit DMAdir.
SVCREQP*	68	OD	**ACTIVE-LOW SERVICE REQUEST PARALLEL:** This is an open-drain output and must be tied to external V_{CC} via a pull-up resistor.
SVCACKP*	69	I	**ACTIVE-LOW SERVICE ACKNOWLEDGE PARALLEL:** This input must not be driven active except in response to a modem service request presented by the device.
PD[7:0]	41–48	I/O	**PARALLEL PORT DATA LINES [7:0]:** Bidirectional, depending on the protocol being used, these signals are used to transfer data over the interface between the master and slave.
GP[7:0]	53–60	I/O	**GENERAL-PURPOSE I/O [7:0]:** General-purpose input/output port data lines. These signals are individually direction-programmable, acting as inputs or outputs. The direction of each signal is controlled by the corresponding bit in the GPDIR register. Control/status of the actual signals is provided via the GPIO register.
A_1284	31	I	**ACTIVE-HIGH 1284 ACTIVE INPUT:** (SLCTIN* in Compatibility mode).
nInit	34	I	**ACTIVE-LOW INIT SIGNAL:** (INIT* in Compatibility mode).

Symbol	Pin No.	Type	Description
HstBsy	32	I	**ACTIVE-HIGH HOST BUSY SIGNAL:** (AUTOFD* in Compatibility mode).
HstClk	33	I	**ACTIVE-LOW HOST CLOCK SIGNAL:** (STROBE* in Compatibility mode).

NOTE: The above four parallel handshake signals are driven by the master in an IEEE STD 1284 interface, and as such are inputs to the CL-CD1283. Their functions depend on the transfer protocol selected. Refer to the IEEE STD 1284 document for protocol functions.

Symbol	Pin No.	Type	Description
PerClk	37	O	**ACTIVE-LOW PERIPHERAL CLOCK:** (ACK* in Compatibility mode).
PerBsy	36	O	**ACTIVE-HIGH PERIPHERAL BUSY:** (BUSY in Compatibility mode).
AkDaRq	35	O	**ACKNOWLEDGE DATA REQUEST:** (PERROR* in Compatibility mode).
Xflag	39	O	**EXTENSIBILITY FLAG:** (SELECT in Compatibility mode).
nDatAv	38	O	**ACTIVE-LOW DATA AVAILABLE SIGNAL:** (FAULT* in Compatibility mode).

NOTE: The above five parallel handshake signals are driven by the slave in an IEEE STD 1284 interface, and as such are outputs from the CL-CD1283. Their functions depend on the transfer protocol selected. Refer to the IEEE STD 1284 document for protocol functions.

Symbol	Pin No.	Type	Description
EBDIR	49	O	**EXTERNAL BUFFER DIRECTION:** This signal is controlled by the internal parallel port control state-machine and can be used to control the direction of an external buffer connected to the Parallel Port data bus. An external buffer might be desirable in applications that require higher drive capacity than that provided by the CL-CD1283. EBDIR may be used in conjunction with PDBEN to control this buffer. EBDIR is a logic '0' when the parallel data bus is in an output mode and a logic '1' when in an input mode. It may be connected directly to the direction control input of a 74245-type device.
PDBEN	51	O	**PARALLEL DATA BUS ENABLE:** This signal can be used to control a buffer on the parallel port data lines in applications requiring more signal drive capability then that provided by the CL-CD1283. The signal is controlled by the internal parallel port control state-machine. When low, the parallel port data bus is off (not driving); when high, the port is in an output mode and is actively driving. The signal will toggle between the on and off states during output modes and will only be active (high) while the data bus pins are in the active driving state. This signal can be logically connected to the enable control of 74245 (or equivalent) bidirectional buffers (see Section 5.9 and Figure 5-8).

2. REGISTERS

Local CPU communication with the CL-CD1283 occurs via a register set. Within this register set, there are four types of registers:

- common to all functions of the device (global)
- specific to the parallel pipeline
- specific to the parallel port
- accessible only within the context of a service acknowledge

Global registers are always available to the CPU and their addresses are not affected by the contents of the Access Enable Register (AER). *This register is provided to maintain compatibility with the CL-CD1284. Binary code that worked with the CL-CD1284 loaded the Channel Access Register (CAR) with 0x00 to access the parallel port.* Each section of the parallel channel has a private set of registers that control functions and operation of that portion only. These local registers are accessible only by first loading the AER with the value 0x00.

The tables on the following pages define the register symbols, names, read and write access modes, internal address offsets, and bit definitions. A detailed description of each register, its contents and functions can be found in Section 3. The address offset defined is the binary value that should be applied to the address inputs (A[6:0]) during I/O cycles.

Note that the addresses are shown relative to the CL-CD1283 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-CD1283 appears at every other address; for example, the CL-CD1283 A[0] input is connected to CPU A[1]. In 32-bit systems, the CL-CD1283 appears at every fourth address (CL-CD1283 A[0] is connected to CPU A[2]). In either of these cases, the address used by the programmer will be different than what is shown in the tables. For instance, in a 16-bit Motorola® 68000-based system, the CL-CD1283 is placed on data lines D[7:0], which are at odd addresses in the Motorola manner of addressing. The CL-CD1283 A[0] input is connected to A[1] of the 68000, etc. Thus, CL-CD1283 address 0x40 becomes 0x81 to the programmer. It is left-shifted one bit and A[0] must be a '1' for low-byte (D[7:0]) accesses.

2.1 Register Map

2.1.1 Global Registers

Symbol	Register Name	R/W	A[6:0]	Hex	Page
GFRCR	Global Firmware Revision Code	R/W	100 1111	4F	page 19
AER	Access Enable	R/W	110 1000	68	page 19
SVRR	Service Request	R	110 0111	67	page 19
PIR	Parallel Interrupt	R/W	110 0001	61	page 20
PPR	Prescaler Period	R/W	111 1110	7E	page 20
GPIO	General-Purpose I/O Data	R/W	111 0000	70	page 20
GPDIR	General-Purpose I/O Direction	R/W	111 0001	71	page 20

2.1.2 Virtual Registers

Symbol	Register Name	R/W	A[6:0]	Hex	Page
PIVR	Parallel Interrupt Vector	R	100 0000	40	page 21
EOSRR	End-of-Service Request	W	110 0000	60	page 21

2.1.3 Parallel Pipeline Registers

Symbol	Register Name	R/W	A[6:0]	Hex	Page
DMABUF	DMA Buffer Data	R/W	011 0000	30	page 22
PFCR	Parallel FIFO Control	R/W	011 0001	31	page 22
PFSR	Parallel FIFO Status	R	011 0010	32	page 24
DER	Data Error	R	011 0011	33	page 25
HRSR	Holding Register Status	R	011 0100	34	page 26
LIVR	Local Interrupt Vector	R/W	001 1000	18	page 27
PFHR1	Parallel FIFO Holding Register 1	R/W	011 0101	35	page 27
PFHR2	Parallel FIFO Holding Register 2	R/W	011 0110	36	page 28
RLCR	Run Length Count	R/W	011 0111	37	page 28
PFFP	Parallel FIFO Fill Pointer	R/W	011 1000	38	page 29
PFEP	Parallel FIFO Empty Pointer	R/W	011 1001	39	page 29
PFQR	Parallel FIFO Quantity	R/W	011 1010	3A	page 29
PFTR	Parallel FIFO Threshold	R/W	011 1011	3B	page 29
SDTPR	Stale Data Timer Period	R/W	011 1100	3C	page 30
SDTCR	Stale Data Timer Count	R/W	011 1101	3D	page 30
PACR	Parallel Auxiliary Control	R/W	011 1111	3F	page 31
PCRR	Parallel Channel Reset	R/W	110 1100	6C	page 32

2.1.4 Parallel Port Registers

Symbol	Register Name	R/W	A[6:0]	Hex	Page
PCR	Parallel Configuration	R/W	010 0000	20	page 32
MDR	Manual Data	R/W	010 0001	21	page 33
PCIER	Parallel Channel Interrupt Enable	R/W	010 0010	22	page 33
PCISR	Parallel Channel Interrupt Status	R/W	010 0011	23	page 33
EAR	EPP Address	R/W	010 0101	25	page 34
SPR	Short Pulse	R/W	010 0110	26	page 34
NER	Negotiation Enable	R/W	010 1000	28	page 34
NSR	Negotiation Status	R/W	010 1001	29	page 35
SCR	Special Command	R/W	010 1010	2A	page 36
OVR	Output Value	W	010 1011	2B	page 36
IVR	Input Value	R	010 1110	2E	page 37
ZDR	Zeroes Detect	R/W	010 1100	2C	page 37
ODR	Ones Detect	R/W	010 1101	2D	page 37
SSR	Signal Status	R/W	010 1111	2F	page 37

2.1.5 Special Register

Symbol	Register Name	R/W	A[6:0]	Hex	Page
RCR	Reset Command	R/W	000 0101	05	page 38

2.2 Register Definitions

2.2.1 Global Registers

Register Name	Symbol	(Hex)	R/W	Page
Global Firmware Revision Code Register	**(GFRCR)**	**4F**	**Read/Write**	**page 19**

Firmware Revision Code

Register Name	Symbol	(Hex)	R/W	Page
Access Enable Register	**(AER)**	**68**	**Read/Write**	**page 19**

poll	poll	poll	poll	poll	0	0	0

Register Name	Symbol	(Hex)	R/W	Page
Service Request Register	**(SVRR)**	**67**	**Read Only**	**page 19**

DMAREQ	n/u	n/u	n/u	SRP	n/u	n/u	n/u

2.2.1 Global Registers *(cont.)*

Register Name	Symbol	(Hex)	R/W	Page
Parallel Interrupt Register	**(PIR)**	**61**	**Read/Write**	**page 20**

PpIreq	PPort	Pipeline	0	0	0	0	0

Register Name	Symbol	(Hex)	R/W	Page
Prescaler Period Register	**(PPR)**	**7E**	**Read/Write**	**page 20**

Binary Value

Register Name	Symbol	(Hex)	R/W	Page
General Purpose I/O Register	**(GPIO)**	**70**	**Read/Write**	**page 20**

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

Register Name	Symbol	(Hex)	R/W	Page
General Purpose I/O Direction Register	**(GPDIR)**	**71**	**Read/Write**	**page 20**

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

2.2.2 Virtual Registers

Register Name	Symbol	(Hex)	R/W	Page
Parallel Interrupt Vector Register	**(PIVR)**	**40**	**Read Only**	**page 21**

x	x	x	x	x	IT2	IT1	IT0

Register Name	Symbol	(Hex)	R/W	Page
End-of-Service Request Register	**(EOSRR)**	**60**	**Write Only**	**page 21**

x	x	x	x	x	x	x	x

2.2.3 Parallel Pipeline Registers

Register Name	Symbol	(Hex)	R/W	Page
Local Interrupt Vector Register	**(LIVR)**	**18**	**Read/Write**	**page 27**

User Defined Bits	IT2	IT1	IT0

Register Name	Symbol	(Hex)	R/W	Page
DMA Buffer Data Register, Low	**(DMABUF)**	**30**	**Read/Write**	**page 22**

7	6	5	4	3	2	1	0

Register Name	Symbol	(Hex)	R/W	Page
DMA Buffer Data Register, High	**(DMABUF)**	**30**	**Read/Write**	**page 22**

15	14	13	12	11	10	9	8

2.2.3 Parallel Pipeline Registers *(cont.)*

Register Name	Symbol	(Hex)	R/W	Page
Parallel FIFO Control Register	**(PFCR)**	**31**	**Read/Write**	**page 22**

FIFOres	DMAen	DMAdir	IntEn	RLEen	setTAG	ErrEn	DMAbufWe

Parallel FIFO Status Register	**(PFSR)**	**32**	**Read Only**	**page 24**

FFfull	FFempty	Timeout	HRtag	HRdata	Stale	OneChar	DataErr

Data Error Register	**(DER)**	**33**	**Read/Write**	**page 25**

DMAwrerr	DMArderr	Bufwrerr	Bufrderr	HR1wrerr	HR1rderr	Hr2wrerr	Hr2rderr

Holding Register Status Register	**(HRSR)**	**34**	**Read Only**	**page 26**

HR1full	HR1tag	HR2full	HR2tag	DMAfull	DMAmpty	DMAact	Ctnot0

Parallel FIFO Holding Register 1	**(PFHR1)**	**35**	**Read/Write**	**page 27**

8-bit Character Data

Parallel FIFO Holding Register 2	**(PFHR2)**	**36**	**Read/Write**	**page 28**

8-bit Character Data

Run Length Count Register	**(RLCR)**	**37**	**Read/Write**	**page 28**

0	7-bit Unsigned Binary Count

Parallel FIFO Fill Pointer	**(PFFP)**	**38**	**Read/Write**	**page 29**

0	0	6-bit binary FIFO Pointer Value

Parallel FIFO Empty Pointer	**(PFEP)**	**39**	**Read/Write**	**page 29**

0	0	6-bit binary FIFO Pointer Value

Parallel FIFO Quantity Register	**(PFQR)**	**3A**	**Read/Write**	**page 29**

Data or Space Available in FIFO — Max x40

Parallel FIFO Threshold Register	**(PFTR)**	**3B**	**Read/Write**	**page 29**

0	DMA Transfer Threshold

Stale Data Timer Period Register	**(SDTPR)**	**3C**	**Read/Write**	**page 30**

8-bit Stale Data Time-out Value

2.2.3 Parallel Pipeline Registers *(cont.)*

Register Name	Symbol	(Hex)	R/W	Page
Stale Data Timer Count Register	(SDTCR)	3D	Read/Write	page 30

8-bit Stale Data Timer Count

Register Name	Symbol	(Hex)	R/W	Page
Parallel Auxiliary Control Register	(PACR)	3F	Read/Write	page 31

ShrtTen	ShrtStal	StaleOff	FIFOlock	ClearTO	0	AsyncDMA	Unfair

Register Name	Symbol	(Hex)	R/W	Page
Parallel Channel Reset Register	(PCRR)	6C	Read/Write	page 32

0	0	0	0	0	0	0	PChReset

2.2.4 Parallel Port Registers

Register Name	Symbol	(Hex)	R/W	Page
Parallel Configuration Register	(PCR)	20	Read/Write	page 32

ManMd	E1284	ETxfr	0	0	0	MMDir	ManOE

Register Name	Symbol	(Hex)	R/W	Page
Manual Data Register	(MDR)	21	Read/Write	page 33

8-bit Binary Data

Register Name	Symbol	(Hex)	R/W	Page
Parallel Channel Interrupt Enable Register	(PCIER)	22	Read/Write	page 33

0	0	NegCh	SigCh	EPPAW	DirCh	IDReq	nINIT

Register Name	Symbol	(Hex)	R/W	Page
Parallel Channel Interrupt Status Register	(PCISR)	23	Read/Write	page 33

0	0	NegCh	SigCh	EPPAW	DirCh	IDReq	nINIT

Register Name	Symbol	(Hex)	R/W	Page
EPP Address Register	(EAR)	25	Read/Write	page 34

8-bit Binary Value

Register Name	Symbol	(Hex)	R/W	Page
Short Pulse Register	(SPR)	26	Read/Write	page 34

8-bit Binary Value

Register Name	Symbol	(Hex)	R/W	Page
Negotiation Enable Register	(NER)	28	Read/Write	page 35

0	RID	0	EPP	RLE	ECP	RVB	RVN

2.2.4 Parallel Port Registers *(cont.)*

Register Name	Symbol	(Hex)	R/W	Page
Negotiation Status Register	**(NSR)**	**29**	**Read/Write**	**page 35**

NegOK	NegFl	0	Invalid	4-bit Negotiation Result Code			

Register Name	Symbol	(Hex)	R/W	Page
Special Command Register	**(SCR)**	**2A**	**Read/Write**	**page 36**

0	0	0	0	ClrPs	SetPs	EPIrq	RevReq

Register Name	Symbol	(Hex)	R/W	Page
Output Value Register	**(OVR)**	**2B**	**Write Only**	**page 36**

PerBsy	PerClk	AkDaRq	xFlag	nDatAv	0	0	0

Register Name	Symbol	(Hex)	R/W	Page
Zeroes Detect Register	**(ZDR)**	**2C**	**Read/Write**	**page 37**

0	0	0	0	A1284	nInit	HstBsy	HstClk

Register Name	Symbol	(Hex)	R/W	Page
Ones Detect Register	**(ODR)**	**2D**	**Read/Write**	**page 37**

0	0	0	0	A1284	nInit	HstBsy	HstClk

Register Name	Symbol	(Hex)	R/W	Page
Input Value Register	**(IVR)**	**2E**	**Read Only**	**page 37**

0	0	0	0	A1284	nInit	HstBsy	HstClk

Register Name	Symbol	(Hex)	R/W	Page
Signal Status Register	**(SSR)**	**2F**	**Read/Write**	**page 37**

0	0	0	0	A1284	nInit	HstBsy	HstClk

2.2.5 Special Register

Register Name	Symbol	(Hex)	R/W	Page
Reset Command Register	**(RCR)**	**05**	**Read/Write**	

1	0	0	0	0	0	0	1

3. DETAILED REGISTER DESCRIPTIONS

This section presents a 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 divide the register into its component parts and describe the bits individually. The registers are presented in the same order as in Section 2.

3.1 Global Registers

Global Firmware Revision Code Register (GFRCR) 4F Read/Write

Firmware Revision Code

The GFRCR serves two purposes in the CL-CD1283. First, it displays the revision number of the firmware in the chip. When a revision to the CL-CD1283 is required, the revision number of the firmware is incremented by one. The revision code is 23 (hex) for the Revision C device, and 24 (hex) for the Revision D device.

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-CD1283 is ready to accept host accesses, the register is loaded with the revision code.

Access Enable Register (AER) 68 Read/Write

poll	poll	poll	poll	poll	0	0	0

The AER provides binary compatibility with the CL-CD1284. Users must program this register with the least-significant bits set to a '0' to access the parallel channel.

Service Request Register (SVRR) 67 Read Only

DMAREQ	n/u	n/u	n/u	SRP	n/u	n/u	n/u

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 7	DMA Request Status; a '1' indicates request pending
Bits 6:4	These bits are not used.
Bit 3	Service Request Parallel; a '1' indicates request pending
Bits 2:0	These bits are not used.

Parallel Interrupt Register **(PIR)** **61** **Read/Write**

ppireq	PPort	Pipeline	0	0	0	0	0

The Parallel Interrupt Register indicates the source of service requests being presented by the parallel channel, either from the parallel port or from the pipeline.

ppireq:

Internal logic sets this bit to generate the external service request output. It is a direct reflection of the inverse state of the SVCACKP* pin; it is the active-high output of the latch that drives the SVCACKP* pin. This bit can be scanned by the host to detect an active service request. The bit is cleared by the internal logic at the beginning of the hardware service-acknowledge cycle; the host clears it to indicate the completion of a software-activated acknowledge cycle.

PPort:

Pipeline:

These two bits indicate which of the two functional blocks of the parallel port are requesting service. PPort set indicates that the parallel channel control state-machine is the cause of the request; Pipeline set indicates that the data pipeline is requesting service. Both bits set indicates that both blocks are requesting service simultaneously.

The remainder of the bits in the PIR will always return zeroes when read by the host and cannot be modified.

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-CD1283 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. For best device operation, the value loaded into the PPR should not be less than x'30.

General-Purpose I/O Register **(GPIO)** **70** **Read/Write**

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

General-Purpose I/O Direction Register **(GPDIR)** **71** **Read/Write**

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

This pair of registers enables access and control of the General-Purpose I/O port. The General-Purpose I/O port provides a byte-wide general purpose set of signals that are individually direction programmable.

The GPIO register accesses the data port on pins 53:60 (G[7:0]) with Data 0 accessing GP[0], etc. The corresponding bit in the GPDIR register controls the direction of the associated signal; a '1' programs the signal as output and a '0' programs it as input. When writing to the GPIO register, '1's and '0's are reflected in their true states on the pins that are programmed as outputs. When reading from the GPIO register, bits programmed as inputs will reflect the true state of the signal condition on those bits; bits programmed as output will reflect the previously set state.

3.2 Virtual Registers

The CL-CD1283 has two operational contexts, a normal context that 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-CD1283 to maintain all channel-specific information. The host need not make any changes to chip registers to access the registers pertinent to the parallel channel.

The service-acknowledge context is entered into in one of two ways; either via activation of the SVCACKP* input pin (hardware activated), or via host software when the contents of any one of PIR is copied into the AER by host software during a Poll-mode acknowledge cycle (software-activated). See Section 5 for a discussion of the differences between these two modes.

Parallel Interrupt Vector Register (PIVR) 40 Read Only

x	x	x	x	x	IT2	IT1	IT0

This register provides information about the service request that is being acknowledged.

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-CD1283 when the register is read. The vector provided will be as indicated for the particular interrupt. The following table shows the encoding of IT2–IT0.

IT2	IT1	IT0	Encoding
0	0	0	Not used
0	0	1	Not used
0	1	0	Not used
0	1	1	Not used
1	0	0	The parallel channel state-machine requests service.
1	0	1	The parallel channel data pipeline request service.
1	1	0	Both the parallel channel state-machine and pipeline request service.
1	1	1	Not used

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-activated service-acknowledge procedure, activated via the SVCACKP* pin. The data pattern written is a 'don't care' value. Writing this location causes the CL-CD1283 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.

3.3 Parallel Pipeline Registers

DMA Buffer Data Register, Low (DMABUF) 30 Read/Write

7	6	5	4	3	2	1	0

DMA Buffer Data Register, High (DMABUF) 30 Read/Write

15	14	13	12	11	10	9	8

This 16-bit data register is used to buffer DMA data transfers to and from the CL-CD1283. Under normal operating conditions, this register will only be accessed during a DMA data transfer cycle. If the DMAbufWe bit of the PFCR is set to a '1' and DMAdir is set to a '1', data may be transferred from the host to the FIFO by writing to the DMABUF directly. The data will automatically move forward into the FIFO via the Data Pipeline Holding registers. The user must guarantee that the FIFO has sufficient free space to accept the data before writing into the DMABUF.

The BYTESWAP pin determines the order of byte transfer from this register into the data pipeline. If BYTESWAP is set to a '1', data transferred on DB[15:8] is the first byte transferred into the data pipeline and DB[7:0] is transferred second; BYTESWAP is set to a '0' reverses this sequence. The same applies during data read during DMA transfers; if BYTESWAP is set to a '1', data from the data pipeline will be moved to the upper byte of DMABUF and the next byte will go into the lower byte. Again, BYTESWAP is set to a '0' reverses the sequence.

Parallel FIFO Control Register (PFCR) 31 Read/Write

FIFOres	DMAen	DMAdir	IntEn	RLEen	setTAG	ErrEn	DMAbufWe

This register is used to control overall function of the parallel FIFO. These include resetting (flushing) the FIFO, enabling DMA transfers, enabling host interrupts, run-length encoding, etc. The host sets these bits according to the mode of operation desired.

After hard reset (either via the RESET* input pin or by setting bit 0 in the Parallel Channel Reset register, this register is cleared to all zeroes.

Bit 7 FIFOres — FIFO Reset

FIFOres must be set together with the correct value of DMAdir to properly initialize the data pipeline and FIFO registers for data transfer or when a new data transfer direction is desired. Data remaining in the FIFO, is discarded.

Bit 6 DMAen — DMA Enable

DMAen must be set in order for DMA requests to move data to or from the FIFO to be made. When DMAen is set to a '1', The PFQR quantity value is compared with the PFTR user programmed threshold value. In Receive mode, if the threshold is equalled or exceeded, DMAREQ* is asserted to cause DMA data transfers of whole (2-byte) words from the FIFO via the data pipeline. In Transmit mode, if the amount of data in the FIFO is equal or less than the threshold, DMAREQ* is asserted to cause DMA data transfers of whole (2-byte) words to the FIFO via the data pipeline.

Bit 5	DMAdir — DMA Direction The DMAdir bit sets the direction of transfer between the parallel FIFO and system memory. If DMAdir is set to a '1', the direction is transmit (system memory to the parallel FIFO), if it is a '0' the direction is receive. The desired DMAdir value must be set together with FIFOres when initializing the FIFO logic for data transfer. Once a DMAdir value is set and the FIFOres has been completed, that DMAdir selection must be maintained during any other changes to the control bits of the PFCR.
Bit 4	IntEn — Interrupt Enable This is the master interrupt enable for the parallel channel. This bit must be set for any interrupts to be generated by the data pipeline, parallel port, or error status. In Poll-mode operation, host software toggles this bit to signal the completion of the service-acknowledge cycle.
Bit 3	RLEen — RLE Enable RLEen enables RLE encoding/decoding for direction defined by DMAdir. The RLEen bit affects the flow of data through the data pipeline in the transmit direction. Data flow into the FIFO is managed in such a way that PFHR1 and PFHR2 are kept filled to permit evaluation of data sequences for possible compression. The effect is that following any data transfer while RLEen is set, the final 2 bytes written to the DMA Buffer register are kept in PFHR1 and PFHR2. To allow these bytes to be moved into the FIFO or to make room in PFHR1 for a tagged data transfer RLEen must be a '0' and both DMAen and DMAbufWe must be a '0'.
Bit 2	setTAG — Set TAG The setTAG bit specifies that the next character written to the parallel channel via the PFHR1 register is to be tagged as an ECP or EPP special character. The setTAG bit is cleared by the write to PFHR1 thus this bit must be set each time a tagged character is to be written.
Bit 1	ErrEn — Error Interrupt Enable Enables a non-zero DataErr status to cause an interrupt if IntEn is also set.
Bit 0	DMAbufWe — DMA Buffer Write Enable This bit must be set to enable host writes to the DMA Buffer register. It enables the FIFO data pipeline to empty the DMA Buffer register when it has been written to by the host system. In this case, the system will write to the DMA buffer, and not use DMA transfers, providing a low performance alternative to DMA transfers.

Parallel FIFO Status Register (PFSR) 32 Read Only

FFfull	FFempty	Timeout	HRtag	HRdata	Stale	OneChar	DataErr

The PFSR is a read-only register that provides current FIFO and data pipeline status. Host software should examine these bits in response to pipeline interrupts or for polling operations.

This register is not directly cleared by reset, but the individual bits will reflect the status of other registers.

Bit 7	FFfull — Parallel FIFO is full This bit, if set, indicates the parallel FIFO is full.
Bit 6	FFempty — Parallel FIFO is empty If this bit is set, the parallel FIFO is empty.
Bit 5	Timeout This bit is set when stale goes from false to true. In the receive direction, Timeout is delayed until the FIFO is empty and all DMA cycles are complete. Timeout is a pipeline interrupt condition, and must be cleared manually by the CPU by toggling ClrTo in PACR or by a FIFO reset in PFCR.
Bit 4	HRtag — Holding Register Tag The HRtag bit indicates that a tagged character is in either PFHR1 or PFHR2 or both. This bit being set will cause a host interrupt to be generated if so enabled. The host should examine the HRSR to determine the exact cause(s) of this bit being set.
Bit 3	HRdata — Holding Register Data This bit, if set, indicates that either PFHR1 or PFHR2 or both contain data.
Bit 2	Stale If a single byte remains in the data pipeline when stale is true due to the stale data timer expiring, a host interrupt is generated. No host interrupt is generated if the FIFO and data pipeline are empty when stale becomes true and StaleOff is not set (see description of SDT-PR).
Bit 1	OneChar — One Character In the receive direction, this bit being set indicates that the FIFO is empty and stale and one character remains in PFHR2. This condition will occur if an odd number of bytes is transferred via the parallel interface. Since DMA cycles will only move even numbers of bytes (words), and odd transfer will leave one byte remaining. Host software must remove this character outside of DMA transfer cycles.
Bit 0	DataErr — Data Error This bit, if set, indicates that one or more of the bits in the Data Error register (DER) are set.

Data Error Register **(DER)** **33** **Read Only**

DMAwrerr	DMArderr	Bufwrerr	Bufrderr	HR1wrerr	HR1rderr	Hr2wrerr	Hr2rderr

The bits in this read-only register indicate read/write errors involving the DMA Buffer register and the Data Pipeline registers. The DataErr bit in PFSR is the logical OR of these eight Error Status bits.

Reading this register has no effect on the error status. Writing to this register has the effect of clearing all the bits; they are not individually writable by the user. Host software should clear this register (write x'00) after completing an error service-acknowledge procedure. This bit is provided primarily as an aid to driver software development. Data errors should never occur under normal circumstances.

This register is cleared during device reset.

Bit 7 DMAwrerr — DMA Write Error

This bit is set if the DMA control logic has written to the DMA Buffer when it already contains data. It indicates that an invalid DMA transfer cycle occurred (a DMAACK* without a corresponding DMAREQ*).

Bit 6 DMArderr — DMA Read Error

As with bit 7, this bit indicates that DMA logic has performed a read from the DMA Buffer when there was no data in it. It indicates that an invalid DMA transfer cycle occurred.

Bit 5 Bufwrerr — Buffer Write Error

This bit indicates that a system write to the DMA Buffer occurred while it still contained data.

Bit 4 Bufrderr — Buffer Read Error

This bit indicates that a system read from the DMA Buffer occurred while it was empty.

Bit 3 HR1wrerr — Holding Register 1 Write Error

This bit indicates that a system write to Parallel FIFO Holding Register 1 (PFHR1) occurred while it still contained data.

Bit 2 HR1rderr — Holding Register 1 Read Error

This bit indicates that a system read from Parallel FIFO Holding Register 1 (PFHR1) occurred while it was empty.

Bit 1 Hr2wrerr — Holding Register 2 Write Error

This bit indicates that a system write to Parallel FIFO Holding Register 1 (PFHR2) occurred while it still contained data.

Bit 0 Hr2rderr — Holding Register 2 Read Error

This bit indicates that a system read from Parallel FIFO Holding Register 1 (PFHR2) occurred while it was empty.

Holding Register Status Register (HRSR) 34 Read Only

HR1full	HR1tag	HR2full	HR2tag	DMAfull	DMAmpty	DMAact	Ctnot0

The Holding Register Status register is a read-only register that indicates current data pipeline status.

This register is not directly set to any particular value by device reset, but it reflects the current state of bits in other registers.

Bit 7 HR1full — Holding Register 1 Full

Bit 6 HR1tag — Holding Register 1 Tagged

These two bits indicate status of Parallel FIFO Holding Register 1 (PFHR1). Bit 7 indicates that the register contains data and bit 6 indicates that the data is tagged. Both bits 7 and 6 can be set simultaneously.

Bit 5 HR2full — Holding Register 2 FullPFHR2

Bit 4 HR2tag — Holding Register 2 Tagged

These two bits indicate status of Parallel FIFO Holding Register 2 (PFHR2). Bit 5 indicates that the register contains data and bit 4 indicates that the data is tagged. Both bits 5 and 4 can be set simultaneously.

Bit 3 DMAfull — DMA Buffer is full

Bit 2 DMAmpty — DMA Buffer is empty

These two bits indicate status of the DMA transfer buffer (DMA Buffer). Bit 3 indicates that the register contains data and bit 2 indicates that it is empty.

Bit 1 DMAact — DMA Active

This bit being set indicates that the DMA handshake is active and that DMA service has been requested, but not yet completed (DMAREQ* active, waiting for DMAACK*).

Bit 0 Ctnot0 — Count Not Zero

This bit indicates that the Run-Length Encoding counter is not a '0', thus run-length encoding/decoding is in progress.

Local Interrupt Vector Register (LIVR) 18 Read/Write

User-Defined Bits	IT2	IT1	IT0

This read/write register is used during service-acknowledge cycles to indicate the type of service needed. Host software places any value desired in the upper five bits; the least-significant three bits are supplied by the device during the SVCACK* cycle and overlay data the host has loaded in them. During this cycle (a special-case read cycle), the CL-CD1283 drives the byte value on the data bus as a vector.

The user-definable bits are cleared by device reset; the vector-type bits are cleared by IntEn being cleared.

Bits 7:3 User-defined Interrupt Vector Bits

Host software can use these bits for any purpose appropriate to the application. In some cases, these bits might define the rest of a complete interrupt response vector (Motorola-type systems). In other applications, they might define information about the parallel channel, or in the case of daisy-chain systems including multiple CL-CD1283s, these bits would be used to define the device number in the chain.

Bits 2:0 IT[2:0] — Interrupt Vector Type Code

These bits are supplied during an interrupt acknowledge cycle to form a request specific vector:

IT2	IT1	IT0	Description
0	0	0	No interrupt source is active.
1	0	0	The parallel port state machine requests service.
1	0	1	The parallel data pipeline requests service.
1	1	0	Both the parallel data pipeline and the parallel port state machine request service.

Parallel FIFO Holding Register 1 (PFHR1) 35 Read/Write

8-bit Character Data

Parallel FIFO Holding Register 2 (PFHR2) 36 Read/Write

8-bit Character Data

These two 1-byte registers provide a data pipeline between the FIFO and DMA Buffer. Data always flows first into PFHR1, then to PFHR2 and finally, either to the FIFO or the DMA Buffer register. The flow is to the FIFO if DMAdir is a '1' and from the FIFO if DMAdir is a '0'. The pipeline and the holding registers support 'tagged' data for complete support of the ECP Parallel Port mode. Tagged data is either an address code or a run-length code.

In the receive direction, if RLEen is set in PFCR, run-length codes are captured in the RLCR for decompression of received data. ECP address codes are recognized and pass into the PFHR1–PFHR2 pipeline. The presence of an ECP address will interrupt DMA flow and cause an interrupt to the host so it can remove the tagged data from the pipeline by reading either PFHR2 or PFHR1.

In the transmit direction, the host may introduce ECP address (tagged) data or run-length codes for precompressed data by setting the setTAG bit in PFCR and writing the byte to be tagged to PFHR1. The setTAG bit must be set prior to writing to PFHR1 for each tagged data transfer. To perform a tagged data transfer, the automatic DMA function must be disabled prior to the transfer (set DMAen to a '0'). This may be done at the same time that setTAG is set to a '1'.

These registers are cleared by device or FIFO reset and marked as empty in HRSR. Any tagged status is also cleared.

Run Length Count Register (RLCR) 37 Read/Write

0	7-bit Unsigned Binary Count

This register works with the Holding registers (PFHR1, PFHR2) to perform run-length encoding and decoding when RLEen is set in PFCR (the parallel port must be in ECP mode; in other modes, run-length encoding will not occur).

In the transmit direction, strings of three or more identical characters are recognized and compressed. The running count of identical characters is kept in the RLCR. Once the sequence is broken by a different character or the end of the transmit burst transfer, the count and a single copy of the duplicated character are put in the FIFO.

In the receive direction, run-length codes may be received from the remote device. These codes are recognized 'on the fly' as data flows from the FIFO through the Holding register pipeline. A run-length code is diverted to the RLCR. The subsequent character from the FIFO is duplicated (held in PFHR1) while the RLCR is decremented. Once the RLCR reaches a '0,' normal pipeline data movement is resumed. If run-length codes are being received by the parallel port, but RLEen is not set, the codes will enter PFHR1 and PFHR2 as tagged data and cause interrupts to the host. The host must read the tagged Holding register directly to remove the character from the pipeline and clear the tag.

This register is cleared by device or FIFO reset.

Parallel FIFO Fill Pointer **(PFFP)** **38** **Read/Write**

0	0	6-bit binary FIFO Pointer Value

This register holds the internal fill location pointer of the FIFO. It identifies the location in the FIFO to receive the next data byte from the pipeline.

The PFFP is cleared by device or FIFO reset.

Parallel FIFO Empty Pointer **(PFEP)** **39** **Read/Write**

0	0	6-bit binary FIFO Pointer Value

This register holds the internal empty location pointer of the FIFO. It identifies the location in the FIFO from which the next byte of data will transfer from the FIFO.

The PFEP is cleared by device or FIFO reset.

Parallel FIFO Quantity Register **(PFQR)** **3A** **Read/Write**

Data or Space Available in FIFO — Max x'40

This register maintains the quantity (or count) of either data bytes or space available in the parallel FIFO. In the receive direction (DMAdir is set to a '0'), PFQR counts data characters in the FIFO. In the transmit direction (DMAdir is set to a '1'), PFQR counts space available in the FIFO for additional characters to transmit. FIFOres together with the value of DMAdir initialize PFQR to either x'00 (receive) or x'40 (transmit).

In either case, the PFQR indicates only the quantity of data or space available in the FIFO, and does not include the data pipeline registers.

Parallel FIFO Threshold Register **(PFTR)** **3B** **Read/Write**

0	DMA Transfer Threshold

This register sets the FIFO threshold for initiating DMA requests for data transfer. The value is expressed in bytes. Whenever DMAen is true, regular comparisons are made between the Quantity register (PFQR) and the PFTR. If the value in the PFQR is greater than or equal to the threshold, the DMA request logic becomes active and remains active until the FIFO is essentially filled or emptied. An odd character or space in the FIFO may remain.

In the receive direction, the Holding register pipeline consisting of PFHR1 and PFHR2 are kept filled, so that tagged data (for example, ECP-mode addresses) may be detected and passed to the host via an interrupt. If the FIFO and data pipeline are initialized for receive and, for example, 40 hex bytes are placed into the FIFO from the parallel port, the first two of those bytes will automatically be placed in the Pipeline registers. If the PFTR were programmed to x'40 bytes, x'42 bytes must arrive to trigger a DMA transfer.

PFTR is cleared by device reset; it is not cleared by FIFOres.

Stale Data Timer Period Register **(SDTPR)** **3C** **Read/Write**

8-bit Stale Data Time-out Value

This register provides a user-defined period value for use as the time-out value of the Stale Data Timer (see SDTCR below).

With a 25-MHz CLK input to the chip, the resolution of this timer is 0.1 ms, its maximum value is 25.5 ms. The 25-MHz clock is divided by 250 to produce a 10-µs intermediate clock for this timer. A fixed, divide-by-ten prescaler produces 0.1-ms 'ticks' to the stale data timer. The prescaler is reset each time the stale data timer is reloaded to ensure accuracy for small time-out values. (A user selection of 0.1-ms time-out would result in a time delay of between 0.09 and 0.1 ms.)

The SDTPR is cleared by device reset.

Stale Data Timer Count Register **(SDTCR)** **3D** **Read/Write**

8-bit Stale Data Timer Count

The Stale Data Timer Counter register determines the period that will be used to signal stale data in the FIFO. The timer is used only in the receive direction. Each time a new character is placed in the FIFO from the parallel port, the SDTCR is reloaded from the SDTPR and down-counting begins at the 'tick' rate. If the counter reaches a '0', the 'Stale' bit in PFSR is set. If the amount of data available is greater than or equal to one word, a DMA request is made to move all remaining whole words to the host by DMA transfer. Once the DMA transfer is complete, a single remaining character will cause an interrupt to the host to remove the character by reading PFHR1.

This register is cleared by device or FIFO reset; clearing causes the Stale bit in PFSR to become true.

Parallel Auxiliary Control Register (PACR) 3F Read/Write

ShrtTen	ShrtStal	StaleOff	FIFOlock	ClearTO	0	AsyncDMA	0

This register provides some special functions for the parallel data path and interrupt-generation circuitry.

The upper two bits are used to change the basic timing of the timers associated with the data pipeline. Bit 5 can be used to disable the stale data time.

Bit 7 ShrtTen

This function shortens the Prescaler count cycle that generates the internal 10-µs (based on a 25-MHz system clock) clock for the stale data counter. This bit is cleared by RESET*. If set, the 10-µs 'ticks' of the counter will be generated every two CLKs; the normal period is one 'tick' every 250 CLKs.

Bit 6 ShrtStal

This function shortens the period of the stale data timer. The stale data timer includes a divide-by-ten prescaler; setting this bit bypasses the prescaler function, causing the stale data timer to count on each 10-µs clock 'tick'.

If both ShrtTen and ShrtStal are set, the stale data timer counts on every other CLK.

Bit 5 StaleOff

This bit, if set, masks off the Stale Status bit. The inverse of this bit is AND'ed with the stale state condition of the parallel channel to produce the stale status and has the effect of disabling OneChar and Stale as interrupt sources. StaleOff is provided primarily for test and development purposes, when slow movement of data into the parallel port might cause Stale and OneChar to always appear true.

Bit 4 FIFOlock

The FIFOlock bit causes the FIFO to stop accepting data from the parallel channel state-machine. This action has the effect of making the FIFO appear full to the parallel port, thus causing it to enter the 'busy' state. This function is primarily intended for use in system testing to cause a time-out on the 1284 bus.

Bit 3 ClearTO

Clear Time-out is a reset bit for the time-out status latch logic. It is toggled by the peripheral host to reset a Time-out status; it may be left set to disable the Time-out status function. Note that if this bit is left set, the OneChar interrupt condition will never become true since there will be no FIFO time-out activity.

Bit 2 Reserved, must be a '0.'

Bit 1 AsyncDMA

AsyncDMA causes the device to synchronize the DMAACK* signal to the internal clock (rising clock edge). This capability provides an 'asynchronous' DMA interface for systems that cannot meet the setup times required by the synchronous DMA logic.

Refer to the Asynchronous Timing on page 67 for specific timing relationships between CLK and DMAACK* when AsyncDMA is enabled.

Bit 0 Not used.

Parallel Channel Reset Register (PCRR) 6C Read/Write

0	0	0	0	0	0	0	PChReset

This register may be used to issue a hardware reset to the parallel channel.

Bits 7:1	Not used; must be a '0'
Bit 0	PChReset

Setting this bit asserts the equivalent of a hardware power-on reset to the parallel channel, channel 0. If set by the host, it must be cleared to resume normal parallel channel operation. This hardware reset affects *only* the parallel channel and has no effect on other functions of the device.

3.4 Parallel Port Registers

Parallel Configuration Register (PCR) 20 Read/Write

ManMd	E1283	ETxfr	0	0	0	MMDir	ManOE

This register controls the overall configuration of the parallel port, each of which is described in IEEE 1284 format below.

Bits 7:5 Mode control

These three bits control the type of transfer desired and whether or not it is enabled to do so.

The ManMd bit selects Manual mode, which allows the user direct control over all parallel data and parallel port control signals. MmDir controls the direction of the Manual Data register (MDR) and ManOE is the output enable when MmDir selects the output mode.

E1283 allows the parallel port to engage in IEEE 1284 negotiations. ETXFR enables data transfers. This enable is only used for data transfers. EPP address read and write functions do not require that the ETXFR bit be set.

ManMd	E1283	Etxfr	Mode
0	0	0	Compatibility mode, transfers disabled
0	0	1	Compatibility mode, transfers enabled
0	1	0	1283 negotiation, transfers disabled
0	1	1	1283 negotiation, transfers enabled
1	x	x	Manual mode

Bits 4:2	Not used, must be a '0'
Bits 1:0	Manual mode control

2136639 0009044 87T

These two bits provide direction and output enable manual control over the parallel port.

MmDir	ManOE	Mode
0	0	Input
0	1	Input
1	0	Output, disabled
1	1	Output, enabled

Manual Data Register (MDR) 21 Read/Write

8-bit Binary Data

This read/write register can be used to read the state of the PD[7:0] signals in any mode. If the ManMD bit is set along with the MMDir and ManOE bits in the Parallel Configuration register (PCR), the value written into this register will be driven onto the PD[7:0] signals.

Parallel Channel Interrupt Enable Register (PCIER) 22 Read/Write

0	0	NegCh	SigCh	EPPAW	DirCh	IDReq	nINIT

Parallel Channel Interrupt Status Register (PCISR) 23 Read/Write

0	0	NegCh	SigCh	EPPAW	DirCh	IDReq	nINIT

The Parallel Channel Interrupt Enable register (PCIER) and Parallel Channel Interrupt Status register (PCISR) provide control and status of interrupts generated by the parallel channel control state-machine. They both have the same bit definitions. Each bit in the PCIER enables the interrupt of the same type in the PCISR. Writing any value to the PCISR in response to an interrupt request causes it to clear and the interrupt request to be removed.

Bits 7:6	Not used, must be a '0'
Bit 5	NegCh The NegCh bit indicates that a change has occurred in the negotiation status of the port. The Negotiation Status register (NSR) will indicate the new status of the parallel port.
Bit 4	SigCh The SigCh enable instructs the parallel port to generate an interrupt when any of the signals specified by the Zero Detect register (ZDR) or Ones Detect register (ODR) change state as programmed.
Bit 3	EPPAW EPPAW indicates that the remote master has written an EPP address to the CL-CD1283. The new EPP address value has been placed in the EPP Address register (EAR).
Bit 2	DirCh DIRCH indicates that the remote master has changed the direction of the port. Generally, this will be in response to a request made by the CL-CD1283 via a direction-change command in the PFCR. This interrupt indicates that the direction has been reversed via the defined protocol and the CL-CD1283 can now send data to the master.

Bit 1	IDReq The IDREQ indicates that the host has requested that the CL-CD1283 send its ID data string. The peripheral host should send the appropriate ID string (this will be application-dependent).
Bit 0	nINIT This interrupt will be generated when an nINIT pulse is received while in Compatibility mode. The interrupt occurs on the leading edge of the nINIT pulse.

EPP Address Register **(EAR)** **25** **Read/Write**

8-bit Binary Value

This register is only used during EPP mode.

The CL-CD1283 deposits the value obtained during an EPP address write command in this register. The CL-CD1283 provides this value in response to an EPP address read command.

Short Pulse Register **(SPR)** **26** **Read/Write**

8-bit Binary Value

This register performs two functions: it sets the duration of the short pulse used by the IEEE 1284 protocols for all modes other than Compatibility; in Compatibility mode it sets the duration of the ACK* pulse. For non-compatible modes, it must be set to n – 2, where *n* is the number of CLKs in a 500-ns pulse. The peripheral host initializes this register with the appropriate value to generate a 500-ns pulse width based on the operating frequency of the device. In Compatibility mode, it should be set to the desired length of the ACK* pulse. This is provided to enable the device to interface to slow masters that require an ACK* pulse longer than the maximum specified in the IEEE 1284 specification. The table below shows examples of the necessary binary value for various system clock frequencies to set the 500-ns pulse width:

Clock	SPR Value	Resultant Pulse Width
16 MHz	8	500 ns
20 MHz	10	500 ns
25 MHz	13	520 ns

Negotiation Enable Register **(NER)** **28** **Read/Write**

0	RID	0	EPP	RLE	ECP	RVB	RVN

Each bit set allows the CL-CD1283 to engage in IEEE STD 1284 negotiations and move into the corresponding protocol. It is assumed that the peripheral host software will respond to a request for slave ID and be able to send an ID string in any protocol that it supports; the CL-CD1283 does not provide a means of storing and automatically sending an ID string in response to an ID request. Note that the EPP protocol does not have provision for slave ID requests.

Bit 7	Not used, must be a ‘0’
Bit 6	Request Slave ID
Bit 5	Reserved — must be a ‘0’
Bit 4	Enable EPP mode
Bit 3	Enable Run-length Encoding in ECP mode
Bit 2	Enable ECP mode
Bit 1	Enable Reverse Byte mode
Bit 0	Enable Reverse Nibble mode

Negotiation Status Register **(NSR)** **29** **Read/Write**

NegOK	NegFl	0	Invalid	4-bit Negotiation Result Code

The results of negotiation attempts are stored in this register. NegOK indicates that the negotiation was successful, while NegFl shows that it failed. Invalid indicates that the state machine has been placed in an invalid state due to the last negotiation sequence and has reentered Compatibility mode. The lower four bits contain a result code, which show the current mode.

Any change of the mode of the parallel port will be reported to the peripheral host by interrupt if the NegCh bit is set in the PCIER; host software would then read the NSR to determine the current status and condition. Once the host has read the NSR status resulting from the current negotiation, it should clear the register in preparation for additional negotiation cycles. The NSR can be cleared by writing any value.

The table below shows the encoding of the result code (bits 3:0):

0 0 0 0	Compatibility mode — no negotiation
0 0 0 1	Reserved
0 0 1 0	Compatibility mode — termination of a 1284 mode
0 0 1 1	Reserved
0 1 0 0	Reserved
0 1 0 1	Reserved
0 1 1 0	EPP mode
0 1 1 1	Reserved
1 0 0 0	Reverse Nibble mode
1 0 0 1	Reverse Nibble mode — ID request
1 0 1 0	Reverse Byte mode
1 0 1 1	Reverse Byte mode — ID request
1 1 0 0	ECP mode without RLE
1 1 0 1	ECP mode without RLE — ID request
1 1 1 0	ECP mode with RLE
1 1 1 1	ECP mode with RLE — ID request

Special Command Register **(SCR)** **2A** **Read/Write**

0	0	0	0	ClrPs	SetPs	EPIrq	RevRq

This register provides the peripheral host processor to issue special commands to the channel control state-machine. In response, the state-machine will perform the indicated IEEE STD 1284-defined handshake on the parallel interface.

Bits 7:4	Must be a '0'
Bit 3	ClrPs — Clear Pause
Bit 2	SetPs — Set Pause

The Set and Clear Pause commands are used to implement an error pause in Compatibility mode. In common practice, errors are presented to the master by the peripheral slave during the active BUSY period of a data transfer. SetPs will remain set until the ClrPs is set, at which time both will clear.

In most cases, the slave host will also set RevRq at the same time when SetPs is set to, first, lockup Compatibility mode with BUSY high and, second, request a reverse transfer, if the master requests that an additional status be sent in the reverse direction.

Bit 1 EPIrq — EPP Interrupt Request

This command causes the state-machine to generate the EPP interrupt sequence. The EPIrq bit clears on the initiation of the Intr (PerClk) pulse on the parallel port interface.

Bit 0 RevRq — Reverse Request

This command will initiate the defined interface reversal handshake as defined by the IEEE STD 1284 specification. The command bit will clear to indicate completion after the command has been executed on the interface. In the case of Reverse Nibble and Reverse Byte modes, this will occur after negotiation is complete; in ECP mode, it will occur after the Reverse Request signal on the parallel port interface goes low.

The parallel port has five outputs and four inputs. The pin assignments are the same as those defined in the IEEE STD 1284 specification. The definition of the pins depends on the current negotiated mode; these are detailed in the following descriptions.

Output Value Register **(OVR)** **2B** **Write Only**

PerBsy	PerClk	AkDaRq	XFlag	nDatAv	0	0	0

This register is used to control output signals. In Manual mode, all signals can be controlled. In Compatible and EPP modes, the PerClk and PerBsy signals are under state machine control, so only those auxiliary status signals indicated can be set by the user.

Bit 7	PerBsy (peripheral busy)
Bit 6	PerClk (peripheral clock)
Bit 5	AkDaRq (acknowledge data request)

In Compatibility mode, this signal is the PError (peripheral error) signal.
In EPP mode, this signal is auxiliary and is a user-defined signal (USER 1).

Bit 4	Xflag In Compatibility mode, this signal is the SELCT (select) signal. In EPP mode, this signal is auxiliary and is a user-defined signal (USER 2).
Bit 3	nDatAv (negative-true data available) In Compatibility mode, this signal is the nFault (negative-true fault) signal. In EPP mode, this signal is auxiliary and is a user-defined signal (USER 3).
Bits 2:0	Reserved, must be written as zeroes.

Input Value Register **(IVR)** **2E** **Read Only**

0	0	0	0	A1283	nInit	HstBsy	HstClk

This register always shows the current state of the external handshake pins.

Bits 7:4	Not used, will return a '0' when read.
Bit 3	A1283
Bit 2	nInit (low active Init input)
Bit 1	HstBsy (Host Busy)
Bit 0	HstClk (Host Clock)

Zeroes Detect Register **(ZDR)** **2C** **Read/Write**

0	0	0	0	A1283	nInit	HstBsy	HstClk

Setting the bits in this register enables the CL-CD1283 to generate an interrupt (if the SIGCH bit in PCIER is set) when the selected signal changes from high to low (falling edge). Bits 7:4 are reserved and must be written as zeroes; they will return a '0' when read. This register is only enabled during Manual mode.

Ones Detect Register **(ODR)** **2D** **Read/Write**

0	0	0	0	A1283	nInit	HstBsy	HstClk

Setting the bits in this register enables the CL-CD1283 to generate an interrupt (if the SIGCH bit in PCIER is set) when the selected signal changes from low to high (rising edge). Bits 7:4 are reserved and must be written as zeroes; they will return a zero when read. This register is only enabled during Manual mode.

Signal Status Register **(SSR)** **2F** **Read/Write**

0	0	0	0	A1283	nInit	HstBsy	HstClk

The bits is this register show the results of changes specified in the ODR and ZDR registers. Normally, the host will read this register in response to a signal-change interrupt generated by the CL-CD1283. This register is active and valid only in Manual mode. Bits 7:4 will return zeroes when read. Writing any value to the register clears it.

3.5 Special Register

Reset Command Register **(RCR)** **05** **Read/Write**

1	0	0	0	0	0	0	1

This is a special-purpose register that allows the local CPU to issue a hard reset of the device via software. The RCR performs the same function as the CCR in the CL-CD1283 except that the only command available is the reset command. To maintain binary compatibility with the CL-CD1284, the CL-CD1283 places the RCR in the address space of either of the two serial channels in the CL-CD1284. Thus, before issuing the reset command, the local CPU must select either channel 2 or 3 via the AER. Once that has been done, writing a hex 81 to the RCR will initiate thee internal reset. The local CPU must wait for the GFRCR to become valid before beginning operation with the device.

4. PROGRAMMING

4.1 Overview

As shown in earlier sections, the CL-CD1283 local CPU interface is made up of a large array of registers. These registers control aspects of chip behavior. Although there are a large number of registers, most of the registers will only be set once, during initialization, and only rarely modified during normal operation. The purpose of this chapter is to discuss these aspects, as well as the methods of interacting with the CL-CD1283 for parallel channel service needs.

4.2 Initialization

To properly power-up a CL-CD1283, several procedures must be completed. These include chip initialization, programming global functions and setting port 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.

4.2.1 Device Reset

The procedures that perform chip reset will normally be executed after a power-up, system-wide reset and, therefore, the CL-CD1283 will have performed its own internal initialization, caused by the hardware reset control signal, RESET*. If desired, however, driver software can issue a software full-chip reset before chip initialization begins. The following steps can be used to accomplish this:

1. **Wait for RCR to contain 0x00**

   The contents of the Reset Command register (RCR) must be zero before the reset command is issued. This is required to be sure that the device is ready to accept the new command. Since this is probably the first command being written to the CL-CD1283 after power-on initialization, the RCR is likely to be a zero, but it is recommended to always check it before writing a new command into it.

2. **Set the Access Enable register (AER) 0x02.**

   This is the only time during normal operation that the AER would be set to any value other than 0x00. Again, this is required to maintain binary compatibility with the CL-CD1284.

3. **Write hexadecimal 81 (x'81) to the Reset Command register (RCR).**

   This command causes the CL-CD1283 to perform a global reset. Its effect is to cause the internal logic to enter its power-up reset location and the results are the same as if the RESET* input had been activated. All internal interface registers are cleared, the FIFO is flushed, and all channel operations are disabled.

4. **Wait for the firmware revision code to be written into the GFRCR.**

   This operation is used to flag completion of the reset procedure. After the reset is issued, the GFRCR is one of the first registers to be cleared and is the last register set before normal operation begins. The local CPU initialization routine *must* wait for this register to become non-zero before beginning any other programming of CL-CD1283 registers. If the CPU is sufficiently fast, it may begin testing the GFRCR before the logic clears it; thus, the assumption would be made that the CL-CD1283 has completed its internal initialization when, in fact, it has not even started. To avoid this error, the CPU should wait for the GFRCR to change to a zero and then to the current revision code. Alternatively, the CPU can clear the GFRCR just prior to issuing the global reset command and then poll for the correct revision code. This is useful in slow systems that cannot guarantee that the CPU will be able to check the register after it has been cleared, and before it is loaded with the revision code.

The following program listing shows a typical initialization sequence that prepares the parallel channel for Compatibility mode data reception and enables negotiation into all other modes (except EPP).

/* Initialization of the parallel channel consists of setting the SPR, selecting modes that will be supported during negotiation, stale data timeout value, initalizing the FIFO, the source for interrupts that will be accepted and other operational functions. */

```
par_init()
{

    /* First, issue chip reset command */

    outportb(GFRCR, 0x00);              /* Clear the GFRCR */
    outportb(AER, 0x02);                /* Set 0x02 in AER */
    while (inportb(RCR) == 0x00)
            ;                           /* Wait for RCR to clear */
    outportb(RCR, 0x81);
    while (inportb(GFRCR) == 0x00)
            ;                           /* Wait for GFRCR to be set */

    /* Start by initializing the parallel channel */

    outportb(AER, 0x00);                /* Set the Access Enable Register */
    outportb(SPR, 0x0d);                /* Assume 25MHz clock, set short pulse value */
    outportb(NER, 0x4f);                /* Support all modes except EPP */
    outportb(OVR, 0x18);                /* Start in Compatibility mode, set status signals: */
                                        /*             PError = 0 */
                                                       /* SELCT = 1 */
                                                       /* nFault = 1 */
    outportb(PCIER, 0x37);              /* Enable all interrupts except EPP Address Write */
    outportb(PCR, 0x60);                /* Enable 1284 negotiations and transfers */

    /* Next, set up the pipeline control registers */

    outportb(LIVR, 0x00);               /* Clear the LIVR, set device ID to 0 */
    outportb(PFCR, 0xd8);               /* Enable pipeline DMA, set the direction to input, */
                                        /* enable interrupts (but not error ints) and reset*/
                                        /* the FIFO. At reset, it is assumed that the starting */
                                        /* direction will be input. */
    outportb(PFTR, 0x20);               /* Set the DMA threshold for receive (burst = 32) */
    outportb(SDTPR, 0x64);              /* Set the stale data timeout period to 10ms */
    outportb(PACR, 0x02);               /* Set asynchronous DMA mode */
}
```

4.3 Parallel Channel Service Routines

In most respects, the parallel channel functions in the same way as the two serial channels. However, the Poll-mode operation is somewhat different and can be performed in a couple of ways. The MPU is only involved with the parallel channel in performing interrupt generation services; all other channel operations are completely separate. Since it is involved in the interrupt structure, some of its behavior must be taken into account.

4.3.1 Software-Activated Service Examples (Poll)

Software activation of the context switch is performed in the same manner, however, termination of the service can be done in two ways. The first method is similar to the serial channel method; the second method may work well in some systems but requires extra steps.

The first method follows the same basic procedure as the serial channels; however, the termination sequence requires only that the upper bit (ppireq) of the PIR be cleared by the CPU. Since Fair Share is not implemented on the parallel channel, there is no 'unfair' bit in the PIR; the 'busy' status is maintained by the MPU differently and, as such, is not in the PIR register.

The routine below shows one way of implementing the Poll-mode-service activation using the first method.

```
par_intr( )
{
    char    vector;

    vector = inportb(SVCACKP) & 0x08;       /* get the vector by activating SVCACKP * input */
    switch (vector) {
       case 4:                              /* just the parallel channel state-machine request is active */
           service_par_chan();
           break;
       case 5:                              /* just the data path pipeline request is active */
           service_pipeline();
           break;
       case 6:                              /* both requests are active */
           service_par_chan();
           service_pipeline();
           break;
       default:
           break;
    }
    outportb(PFCR, inportb(PFCR & 0xEF);    /* clear IntEn (first step of 'toggle' operation */
    outportb(PFCR, inportb(PFCR I 0x10);    /* set IntEn (second step of 'toggle' operation */
    outportb(EOSRR, 0x00);                  /* restore original CAR (if desired) */
    return(0);

}
```

2136639 0009053 882

```
service_par( )
{
    char    save_pir, save_car, livr_val;

    if (inportb(SVRR & 0x08)) {                  /* check for active service request */
       outportb(CAR, 0x00);                      /* switch CL-CD1283 to service ack. context */
       livr_val = inportb(LIVR) & 0x07;
       switch (livr_val) {
           case 4:                               /* just the parallel channel state-machine request is active */
                service_par_chan();
                break;
           case 5:                               /* just the data path pipeline request is active */
                service_pipeline();
                break;
           case 6:                               /* both requests are active */
                service_par_chan();
                service_pipeline();
                break;
           default:
                break;
       }
       outportb(PFCR, inportb(PFCR & 0xEF);     /* clear IntEn (first step of 'toggle' operation */
       outportb(PFCR, inportb(PFCR I 0x10);     /* set IntEn (second step of 'toggle' operation */
       outportb(PIR, save_pir & 0x00);          /* terminate service ack. sequence by clearing bit 7 */
       outportb(CAR, save_car);                 /* restore original CAR*/
       return(0);
    }
}
```

4.3.2 Hardware-Activated Service Examples

Hardware-activated context switching is performed by a service acknowledge cycle. During the service acknowledge cycle, the SVCACKP* input is active and the CL-CD1283 drives the parallel channel vector on the data bus. At the same time, the MPU pushes the current state of the device on the context stack and sets the context for the parallel channel. The vector comes from the PIVR, which is a reflection of the LIVR. The vector supplied will indicate the source of the request in the IT2-IT0 bits. Once the context switch has been occurred, the CPU may proceed to service the source of the request. The CPU must decode the ITx bits to determine which of the blocks need service. Each section of the parallel channel has its own interrupt status register to indicate what conditions, if any, in that block require service. These are the PFSR register in the data path and the PCISR in the channel control state-machine. At the end of the service routine, the CPU must perform two operations. First, it must toggle the IntEn bit in the PFCR, as in the software-activated service acknowledge, and it must execute a dummy write operation to the EOSRR register to inform the device that the parallel service has been completed. The write operation to the EOSRR generates a high-priority interrupt to the internal logic that causes it to pop the context stack and restore the device environment to what it was at the start of the interrupt service.

The example below shows a typical service routine for a hardware-activated acknowledge. The acknowledge occurs during the read from the SVCACKP location.

4.4 ASCII Code Tables

4.4.1 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

4.4.2 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

2136639 0009055 655

5. FUNCTIONAL DESCRIPTION

5.1 Device Architecture

The CL-CD1283 consists of dedicated logic tailored to the function of sending and receiving parallel data. It implements an IEEE 1284-compliant parallel port with a specialized data pipeline designed for high-speed transfers.

To maintain binary compatibility with the CL-CD1284, much of the architectural layout has been duplicated. Thus, access to the register set of the parallel channel is possible only after loading an access enable register with the address that the parallel port in the CL-CD1284 occupied, namely channel 0. This register is named the AER (Access Enable register). For all channel-specific accesses, the CPU first loads the AER with a pointer to channel zero. Thereafter, all read and write operations will occur with the parallel channel.

The parallel channel is comprised of a FIFO and DMA data interface as well as a high-speed state-machine that handles all of the modes defined in the IEEE STD 1284 specification. The parallel port performs the slave, or peripheral, function of the IEEE STD 1284 interface and can be commanded to accept negotiations into any or all of the IEEE defined modes.

5.2 CPU Interface

The CPU interface to the CL-CD1283 comprises an 8-bit bidirectional data bus, a 7-bit address bus, a 16-bit DMA port and various strobes that identify the type of I/O cycle that is occurring. In most system designs, the I/O cycles will be merely normal 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-CD1283 easily fits into any CPU environment.

The interface is completely compatible with the CL-CD1284. As such, a CL-CD1283 can be inserted into a system in place of a CL-CD1284 and functionality and binary compatibility as well as pin compatibility will be maintained. All signal timing has been duplicated on this device.

In most cases, when the CPU reads or writes an internal CL-CD1283 location, it actually accesses a location in a RAM array that serves as a bank of

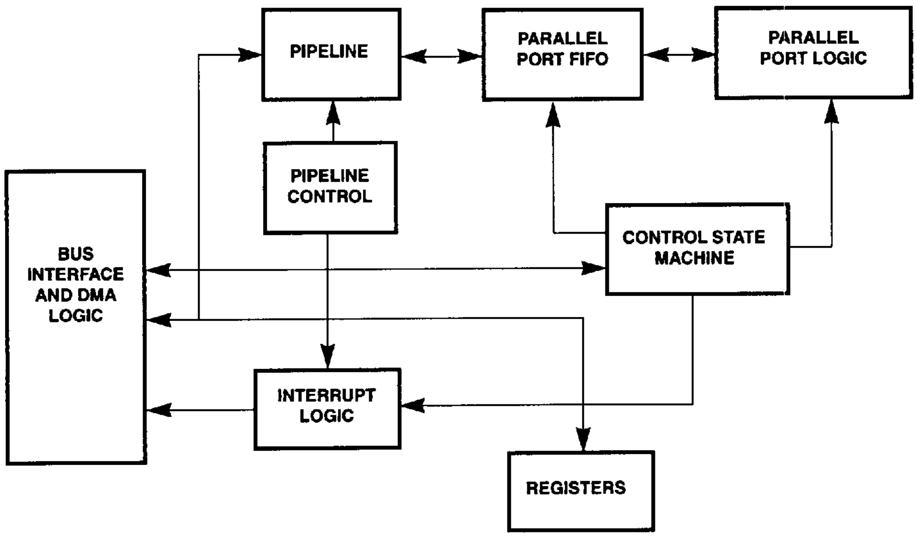


Figure 5-1. Functional Block Diagram

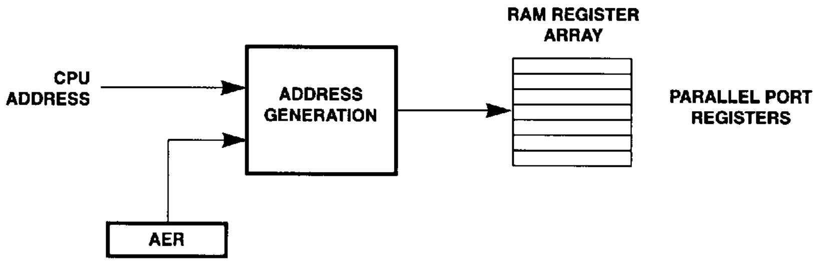


Figure 5-2. Internal Address Generation

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-CD1283 is by design a synchronous device. All internal operations occur 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 CPU performs an I/O cycle with the CL-CD1283, its strobes, address and data are sampled on rising edges of the internal clock. As can be seen in the timing diagrams in Section 6, 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-CD1283's clock, with events (address setup, write data setup and read data available) occurring at predictable times.

It is not necessary to design a synchronous interface to the CL-CD1283. In an asynchronous design, the Data Transfer Acknowledge (DTACK*) signal is used as an indication that the CL-CD1283 has completed the requested data transfer for all I/O cycles except DMA. Thus DTACK* can be an input to wait-state generation logic that will hold the 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-CD1283 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.

5.2.1 Read Cycles

Read cycles are initiated when the CL-CD1283 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 6. 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-CD1283 signals 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 CPU removes CS* and DS*.

5.2.2 Write Cycles

Write-cycle timing and strobe activity is nearly identical to that of 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 6.

Again, the DTACK* signal is used to indicate that the cycle is complete and the CL-CD1283 has taken the data. Removing both CS* and DS* terminates the cycle.

5.2.3 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 SVCACKP* input is activated instead of the CS* input (a slightly longer setup time is required on the SVCACKP* input than on the CS* input). The data that the CL-CD1283 provides during the read cycle is the contents of the Parallel Interrupt Vector register (PIVR). As with read and write cycles, DTACK* will indicate the end of the cycle and removing DS* and SVCACK* terminates the cycle.

It is important to note the following with regard to timing and service-acknowledge cycles: when the CPU 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 logic to complete activities associated with the switch out of the service-acknowledge context. These activities are primarily interrupt logic 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 CPU. If the situation occurs that the CPU attempts an access before the internal procedures are complete, the CL-CD1283 will hold off the cycle until it is ready. In system designs that 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. Failure to observe the delay requirement may cause device malfunction.

5.2.4 DMA Cycles

The CL-CD1283 provides a bidirectional 16-bit DMA interface to the parallel port. This is the only direct data interface to the port; other 8-bit register accesses make use of the normal CPU interface, as described above.

The handshake between the CL-CD1283 and the DMA circuitry makes use of two signals: the DMA Request (DMAREQ*) and the DMA Acknowledge (DMAACK*). The address bus is ignored during DMA transfers. When internal conditions warrant a DMA transfer (FIFO falls below the programmed threshold in Output mode or rises above the threshold in Input mode) and DMA transfers are enabled via the PFCR, the device will request DMA service by driving the DMAREQ* signal low. DMAREQ* will remain active until the FIFO has been filled (Output mode) or emptied (Input mode). In Output mode, the DMA controller logic responds by placing data on the 16-bit data bus and driving the DMAACK* input low. This cycle is repeated until the FIFO is full (or there is no more data to send). In Input mode, the CL-CD1283 responds to the active DMAACK* signal by driving the contents of the DMA Buffer register on to the data bus.

Odd-byte transfers in the receive direction are handled on an interrupt basis; if the number of bytes in the FIFO is odd, all bytes except the last are transferred via a number of 16-bit DMA cycles (two bytes per cycle). The odd byte remaining will be held in PFHR1 and an interrupt will be generated when the stale data timer expires; status indicating that PFHR1 has data will be indicated in the PFSR. The CPU interrupt service routine must manually remove the remaining byte from the interface. In the transmit direction, an odd remaining byte can be directly written to PFHR1 once the last DMA cycle has completed.

One additional input signal determines the 'Endian' format (whether the least-significant byte is on data bits 7:0 or 15:8) of the 16-bit DMA Buffer. BYTESWAP selects whether the lower or upper byte of the DMA Buffer moves into the FIFO data pipeline first in Output mode or from the FIFO data pipeline to the DMA Buffer first in Input mode. If BYTESWAP is low, then the least-significant byte (DB[7:0]) moves into or out of the data pipeline first; BYTESWAP high performs in the opposite manner (DB[15:8] move into or out of the pipeline first).

The effective duration of the DMA transfer block (burst) is determined by the setting of DMAdir in the Parallel FIFO Threshold register (PFTR). In whichever direction the port is moving data, when this threshold is reached (exceeded in receive; less than in transmit), a DMA cycle will begin and will remain active until the FIFO is empty (receive) or full (transmit).

The SVRR register provides a way of determining if a DMA cycle is being requested. Bit 7 of this register will be true if a DMA cycle is currently being requested. This status indication is provided as a general system status.

Refer to Section 6 for detailed information on DMA cycle options and timing values.

5.2.5 Interrupt

The term 'interrupt' is used as a generalized description of the method by which the CL-CD1283 gains CPU service. It is used interchangeably with a 'service request' because the two are the same function. 'Interrupt' is often used to describe an unconditional response on the part of the CPU. Whether or not this is the case, the source is still the same — a service request from the CL-CD1283. The hardware signal generated by the CL-CD1283 (SVCREQP) can be connected to the 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-CD1283s request by starting an acknowledge sequence.

Multiple CL-CD1283s can be connected together in a daisy-chained configuration, forming systems with multiple parallel ports. The device provides all the signals necessary for this configuration, with only minimal external logic being required (see Figure 5-1).

When the CPU acknowledges the request, both CL-CD1283s 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 the first device does not have a request pending, it will pass the DGRANT* input to the second CL-CD1283 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 CPU loaded into them during its initialization of the CL-CD1283s. These bits must be used as a unique chip identification number so the CPU can determine which CL-CD1283 responded to the service acknowledge. These five bits could be set to binary '00000' in the LIVR registers of the first CL-CD1283, and to binary '00001' in those of the second. The CPU can easily test the bit to determine which device responded. Some examples of service-acknowledge software routines that show one way of performing this task are provided in Section 4.

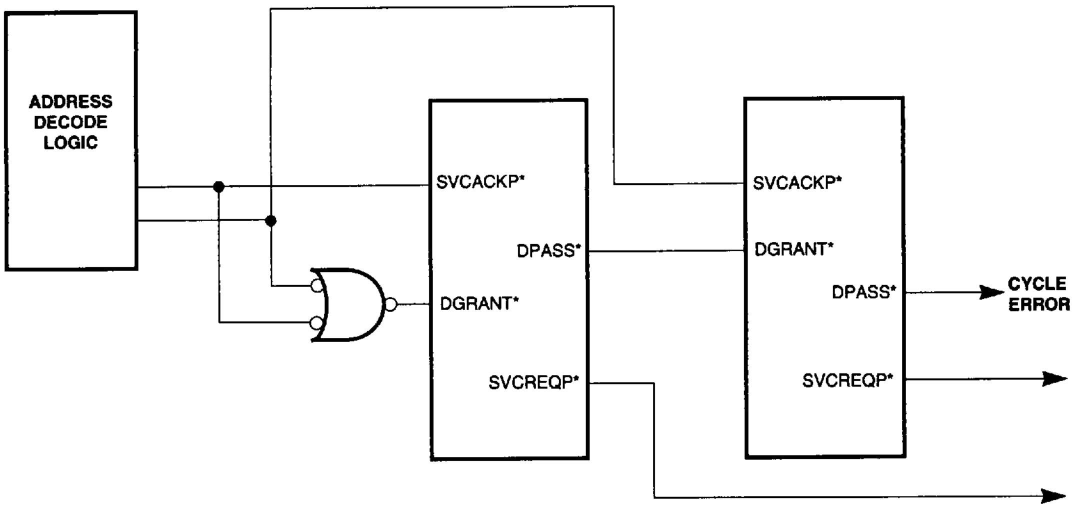


Figure 5-3. Daisy-Chaining a CL-CD1283

CAUTION: If no CL-CD1283 in the chain has a pending request, the daisy-grant will be passed by the last and none will respond, thus causing the bus cycle to fail (no DTACK* is generated). The only time this could happen would be due to an error condition outside the CL-CD1283s that caused the CPU to respond to a request that was not made. A mechanism should be provided to terminate or abort the bus cycle if this error should occur. This can be accomplished with time-out circuitry or the DPASS* output of the last CL-CD1283 can activate an abort condition. Other devices, such as the CL-CD1400, may share the daisy-chain mechanism and could be connected to the DPASS* output of the last CL-CD1283 in the chain. The actual implementation is system-dependent, but it is important to provide some way for the CPU to determine that the cycle did not complete normally if no device responds to the acknowledge cycle.

5.3 Parallel Port Service Requests

The parallel channel a service-request control state-machine (part of the data pipeline logic) handles the request needs of the parallel channel. The source of service requests from the parallel channel can be either (or both) of two logical blocks: the data pipeline and the port-control state-machine. Both of these blocks post their requests to the request state-machine, which then sets up the parallel port interrupt type bits (PPort and Pipeline) in the PIR register. At this point, the interrupt control logic responds to the setting of these two bits; when it detects that one or both of the bits is set, it will setup the PIVR register based on the contents of the LIVR plus the proper vector for the interrupt conditions (see below). Once this has been completed, the SVCREQP* output is activated by setting bit 7 (ppireq) of the PIR. Additionally, the request is reflected in the bit 3 of the SVRR. For polled systems, the SVRR can be scanned for this bit being active to determine that the parallel channel requires service.

Both potential sources of interrupt requests within the parallel channel have their own enable functions. Interrupts from the data pipeline are enabled via the Parallel FIFO Control register (PFCR); interrupts from the channel control state-machine are enabled via the Parallel Channel Interrupt Enable register (PCIER).

The PFCR has two enable bits, one for normal interrupts, such as tagged data being received, and one for data errors, such as a CPU write to a holding register that already has data in it. The first type of interrupt is enabled via the IntEn bit in the PFCR; the second type is enabled via the ErrEn bit. Note that IntEn must be set for ErrEn to generate an interrupt; however the CPU need not enable error interrupts if it does not require notification of these types of errors. The error interrupt will be generated if the DataErr bit in the Parallel FIFO Status register (PFSR) is non-zero; in this case, the Data Error register (DER) will indicate the causes of the error interrupt.

The channel-control state machine can generate six types of interrupts. Each of these has its own enable bit in the PCIER: NEGCH for negotiation changes, SIGCH for signal changes on the port status inputs (Manual mode only), EPPAW for EPP protocol address writes, DIRCH for direction changes on the parallel channel, IDREQ for slave ID requests from the remote master. Any or all of these may be set, based on the mode of operation.

The NEGCH interrupt will be issued whenever the remote master has performed a protocol change, such as moving from Compatibility mode to ECP; the CPU examines the Negotiation Status register (NSR) to determine the new state of the parallel interface. Signal changes can be identified by reading the Signal Status register (SSR). In response to the EPPAW interrupt, the CPU would read the EPP Address register (EAR) to retrieve the value that was written during the EPP address write cycle.

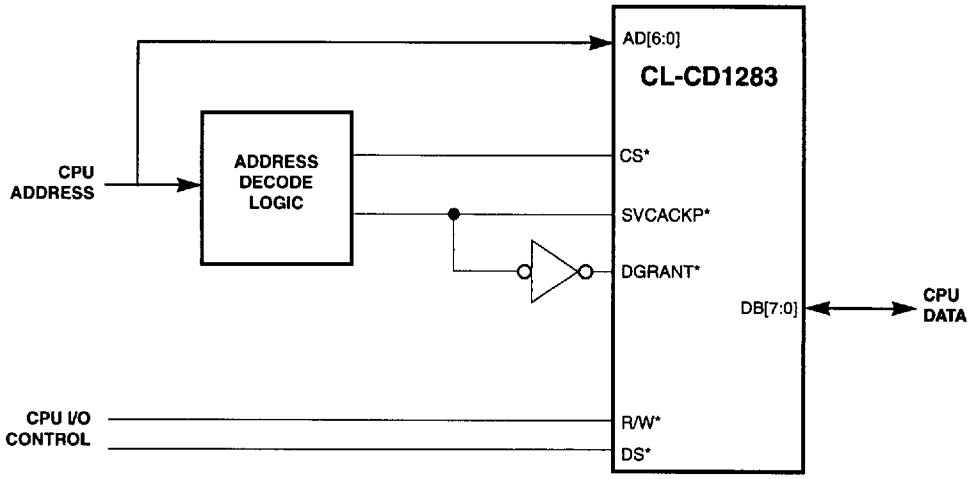


Figure 5-4. Control Signal Generation

2136639 0009061 959

Table 5-1. **Request-Type Bit Assignments**

Bit 2	Bit 1	Bit 0	Request Type
0	0	0	Not used
0	0	1	Not used
0	1	0	Not used
0	1	1	Not used
1	0	0	Parallel port state-machine requests service (refer to Section 5.3)
1	0	1	Parallel port data pipeline request service (refer to Section 5.3)
1	1	0	Both the parallel port state-machine and data pipeline request service (refer to Section 5.3)
1	1	1	Not used

A direction change (DIRCH) interrupt will occur when the remote master has reversed the channel in response to a data available status from the local CPU; this will occur when the direction of the data pipeline has been reversed via the DMAdir bit being set in the PFCR. The IDREQ interrupt is generated when the remote master issues an ID Request command during IEEE 1284 negotiations. The normal response by the local CPU is to send its ID string after requesting a channel reversal.

In an interrupt-driven system the SVCREQP* output would normally be connected to one of the local CPU's interrupt control inputs.

The parallel channel has a Local Interrupt Vector register (LIVR) that is used to set the base vector supplied by the device during a service-acknowledge cycle. This register must be set by the local CPU during device initialization; the upper five bits are defined by the local CPU and may be any value appropriate to the system design. The lower three bits should be initialized to a zero during the programming of the LIVR, however, they are 'don't care' and will be masked in the PIVR to provide the vector indicating the source and type of request from the parallel channel. Access to the parallel channel LIVR is made by first setting the AER to x'00, making the Channel Zero register set accessible. Since the LIVR is a read/write register, the local CPU may read it at any time. When read during a normal read cycle, the upper five bits always read back whatever was loaded into them by the local CPU. The lower three bits will always read back as the current service-request status of the parallel port if an interrupt is in progress; otherwise they read back as a '0.' The encoding of the three least-significant bits indicates which of the functional blocks in the parallel channel is requesting service and is as follows:

IT2 (Bit 2)	IT1 (Bit 1)	IT0 (Bit 0)	Requestor
1	0	0	Channel control state-machine
1	0	1	Data pipeline
1	1	0	Both

5.3.1 Hardware-Activated Context Switch

When conditions within the parallel channel require attention, a request will be made via the SVCREQP* output. If the system is interrupt driven, this output would be connected to the CPU interrupt-generation circuitry. In a hardware-activated service-acknowledge system, the CPU responds to the request by activating the SVCACKP* input, along with DGRANT* and DS*; the CS* input is not used and must remain inactive (high). The CL-CD1283 will respond to the

SVCACKP* cycle by driving the contents of the PIVR onto the data bus with IT2–IT0 encoded as shown above. The SVCACK cycle also places the device in the correct context to service the parallel channel request.

The vector supplied by the PIVR will indicate which block of the parallel channel has requested service; the cause of the request will be indicated in the status request registers of each: the PCISR in the channel control state-machine block and/or the PFSR in the data pipeline block. Refer to Section 3 for detailed descriptions of the various status bits in these registers.

The I/O cycle that activates the SVCACKP* input also removes the active SVCREQP* output. The request output will stay inactive until after the CPU terminates the acknowledge routine by writing to the EOSRR. This is a dummy operation and the data written is 'don't care'; the purpose of the write is to clear the internal logic of the current request context and allow it to generate another request when required. Until this write occurs, no further service requests can be made from the parallel channel. When the MPU detects the write to the EOSRR, it will zero-out the PIVR bits to a '0' in preparation for the next service-request cycle.

5.3.2 Software-Activated Context Switch

During a normal read cycle, the LIVR always reads back with the lower three bits, indicating the current service-request status of the device. Thus, in a Poll-mode system, this register can be used in conjunction with the SVRR to determine if service request needs are pending and, if so, which of the two possible sources is active. If SRP (bit 3) is set in the SVRR, then at least one of the request conditions is true and a subsequent read of the LIVR register will indicate the source. This is the most efficient method of performing a poll of the device; however, the system may also poll just the LIVR of the parallel channel to determine if any requests are pending. This requires more I/O operations since the LIVR is not a global register and, therefore, the AER must be loaded prior to the read operation so that the parallel channel is being addressed. Scanning just the SVRR allows the polling routine to perform only one read cycle to determine whether a parallel request is pending. If the SVRR indicates an active parallel channel service request, the CPU can proceed to switch the device context to that of the parallel channel and initiate the service acknowledge. Alternatively, it can read and decode the PIR to determine the current interrupt request status. The two type bits (PPort and Pipeline) will indicate which block is requesting service.

The start of a Poll-mode acknowledge of the parallel channel begins when the CPU copies the contents of the PIR into the AER (after first saving the current contents of the AER) to set the device context. The CPU can read either the LIVR or PIVR (or it can read status from the two status registers in the Parallel Port register set) to determine which of the parallel channel blocks is requesting service, copy the PIR into the AER (or just load it with x'00) to set the context and then proceed to service that request. Once the CPU has satisfied the request needs of the parallel channel, it must toggle the Int-En bit in the PFCR. This will clear the two PPort and Pipeline bits and the ppireq bit in the PIR. This action will also inform the MPU to clear the PIVR and remove the external request. The CPU would then restore the AER to its previous contents and exit the service routine.

The ppireq bit (bit 7) of the PIR can be cleared at any time by the CPU once it enters the service routine. If the system design requires that the request be removed quickly, this procedure can be performed at the beginning of the polled service routine. If the CPU waits until the end of the service routine, it can clear the bit itself or it can terminate the service in the manner described and simply let the MPU do it.

5.4 Parallel Port FIFO and Data Pipeline Overview

The parallel port within the CL-CD1283 implements all modes defined for the 'slave' (peripheral) side of the *IEEE STD 1284 Standard Signaling Method for a Bidirectional Parallel Peripheral Interface for Personal Computers*. This specification defines four methods of performing bidirectional data transfers between a computer system and a peripheral device, in addition to the generally accepted unidirectional Centronics®-'compatible' mode. These modes include Compatibility mode, Reverse-Nibble mode, Reverse-Byte mode, Extended Capabilities Port (ECP) with and without Run Length Encoding (RLE), and the Enhanced Parallel Port (EPP).

The IEEE 1284-compliant parallel port consists of two major functional blocks: a data pipeline, which moves data between the parallel port and the CPU and includes a FIFO, holding registers, DMA control, interrupt control logic, and a channel control state-machine that performs all control and handshake generation on the parallel port interface side of the device.

5.4.1 IEEE STD 1284 Protocols

The following sections discuss specifics of data movement within the pipeline for the various IEEE STD 1284 operating modes. For a complete description of these modes, refer to the IEEE STD 1284 specification; it is beyond the scope of this data book to give complete information on the specification. A copy of the IEEE STD 1284 standard can be obtained from:

IEEE Standards Department
445 Hoes Lane
P.O. Box 1331
Piscataway, NJ 08855-1331
USA

5.4.2 Bus Interface

DMA (direct-memory access) request and acknowledge handshake signals support transfers to or from the 16-bit-wide DMA Buffer register. The direction of transfer is determined by the DMAdir bit of the PFCR. DMA transfers are the preferred means of transferring data to or from the FIFO.

It is also possible to transfer data to or from the data pipeline that links the DMA Buffer register and the FIFO by reading or writing the holding registers directly. In the transmit direction, with DMAbufWe set, the CPU may write two bytes at a time directly to the DMA Buffer register. The CPU must avoid writing to these registers when they are already full, or reading from them if they are empty. Status bits in the HRSR will indicate whether or not the holding registers and the DMA buffer are full or empty. When writing a block of data to the CL-CD1283 (with DMAbufWe set to a '1'), the CPU may determine how much data the FIFO can accommodate by reading the PFQR.

Should data become 'trapped' in the DMA Buffer register in the receive direction because of a failure of the external DMA controller, or because the external buffer area has been filled, it may either be left there until DMA transfer can be resumed, or the CPU may read the data from the DMA buffer directly.

Once a DMA request is initiated by the CL-CD1283, it is maintained until the last data transfer the FIFO can accommodate occurs, or the CPU either clears DMAen or clears the FIFO and data-transfer logic by setting FIFOres. In the transmit direction, the DMA request is removed by the CL-CD1283 when it determines that the FIFO is nearly full (If RLEen is set, the pipeline will not fully drain into the FIFO, but the logic does not factor that into the decision to conclude the DMA transfer).

In the receive direction, the DMA request is removed when there are not at least two more bytes available to transfer, or a tagged byte has moved into the data pipeline. In the latter case, an interrupt is generated to the CPU (IntEn must be true) to remove the tagged data from the pipeline.

If RLEen is true, and the parallel port is in ECP mode and compressed data is being transferred, the quantity of data transferred within a single DMA request may significantly exceed the capacity of the FIFO. This is because the FIFO always stores the data in compressed form. The CPU sets RLEen only when the parallel port state machine is in ECP mode. In all other modes, in the transmit direction, if RLEen is true, data compression will occur as the data is transferred to the FIFO, but the Parallel Port mode does not support the transfer of tagged data. The tags will be lost, and the transferred data will be corrupted.

5.4.3 Parallel Port FIFO

The CL-CD1283 has a dedicated 64-byte FIFO with counters for fill/empty pointer addresses and logic to manage data transfer, automatic DMA handshake and status interrupts to the CPU. A simple register interface provides control over setting the direction of DMA/parallel port transfer, initializing/resetting the DMA pointers, possible options for DMA threshold, etc. The FIFO-management logic responds to data-transfer requests from the dedicated IEEE 1284 parallel port state machine.

Byte-alignment issues on transfers to/from the FIFO are avoided by having the FIFO be byte-oriented with 2-byte word packing/unpacking occurring between the DMA Data Buffer register and the Parallel FIFO Holding registers. The order of byte transfers to/from the DMA buffer is controlled by the BYTESWAP input. If BYTESWAP is high, the upper byte (bits 15:8) is transferred first. If BYTESWAP is low, the lower byte (bits 7:0) is transferred first.

Data transfers to/from the CPU are initiated by DMA request whenever the quantity of data or space in the FIFO equals or exceeds the threshold value stored in PFTR. The DMA request is deasserted during the DMA cycle that the logic determines must be the last because of filling/emptying the FIFO, or the presence of tagged data in the receive pipeline.

5.4.4 Receive Direction

In the receive direction (DMAdir is set to a '0'), the first 2 bytes of data placed into the FIFO by the parallel port are immediately moved into the data pipeline, PFHR1 and PFHR2 (see Figure 5-7). This is done in part to make the tagged status of the data visible to the pipeline-control logic. If RLEen is a '0,' any tagged data from the FIFO must move through the pipeline. However, tagged data cannot be transferred to the CPU by DMA transfer from the DMA Buffer register. Therefore, the presence of tagged data in the pipeline will cause an interrupt to the CPU. The CPU must examine the HRSR to determine the pipeline status. If there is tagged data in one of the holding registers, the CPU must read that register to empty it and clear the tag. If more data is available in the FIFO, data will immediately move forward to fill the pipeline. If the FIFO is empty, the pipeline will not 'move', so if the CPU emptied PFHR2, and PFHR1 is full, the data in PFHR1 will move forward to PFHR2 only if the FIFO is not empty.

The pipeline logic keeps the pipeline full in the receive direction. The transfer threshold is tested against the quantity of data in the FIFO. Therefore, a number of characters equal to the PFTR threshold value plus two must arrive before a DMA request is made to the CPU to transfer data.

5.4.5 Receiving Compressed Data

RLE (run-length-encoded) compressed data sequences consist of a tagged RLE count followed by the compressed data character are stored in the FIFO in compressed form. As data is moved from the FIFO into the data pipeline, the tag bit is inspected. If the tagged data is a RLE count (bit 7 must be a '0') and control bit RLEen is true, the RLE count is loaded into the RLCR instead of PFHR1. The next data character is loaded into PFHR1. Decompression occurs by holding the compressed character in PFHR1 as copies of the character are shifted forward into PFHR2. As each copy of the character is shifted, the RLCR value is decremented. When the RLCR has reached a '0', the hold on PFHR1 is released and it may shift forward in the pipeline as ordinary data.

Tagged data from the FIFO with bit 7 set is recognized to be an ECP-mode address, and is shifted into the pipeline where it will cause an interrupt to the CPU to remove the tagged data from the pipeline. If RLEen is a '0', all tagged data from the FIFO is shifted into the pipeline and will produce CPU interrupts.

5.4.6 Stale Data (Stale, OneChar, and Timeout Status Bits)

Data transfer to the CPU may also be initiated by the 'stale' data timer. This timer is reloaded with the value in the SDTPR and restarted each time data is placed into the FIFO from the parallel port. When the timer reaches a '0', the status indication stale (visible in the PFSR) is set true unless StaleOff in PTEST is true. StaleOff keeps the stale status false, even though the SDTCR counter value is a '0'. Should the stale status become true with at least two characters of data available, a DMA request will be made to transfer the data. If the stale is true and there is exactly one character available, the OneChar status bit is set in PFSR and an interrupt is generated to the CPU to transfer the single residual character. The Parallel FIFO Status register will indicate the 'Stale' and 'OneChar' conditions and 'FFmpty'. The Holding Register Status register will show that holding register PFHR2 has the final character. An odd number of bytes may not be transferred by DMA. If a DMA transfer completes with 1 byte of data left, the data will be held pending arrival of additional data or the expiration of the stale data timer.

The OneChar status is latched true when the FIFO and DMA buffer are empty, and there is one character in the pipeline in PFHR2. While the OneChar status is true, further pipeline operations are inhibited. This means that if additional data arrives in the FIFO, it will remain there until the CPU: 1) services the interrupt caused by the OneChar status, and 2) reads the data character from PFHR2. If new data has arrived since the OneChar status was latched, the FFmpty bit will be false. When the CPU reads the single character from PFHR2, any newly arrived data in the FIFO will immediately be moved forward into the pipeline and a DMA transfer may be begun if conditions warrant.

Another latched status condition associated with the stale data timer is the Timeout status bit in PFSR. The Timeout status bit is reset by FIFOres in PFCR and by ClrTo in PACR. When set, the Timeout status bit is a pipeline interrupt condition along with OneChar and DataErr. In the receive direction, the Timeout condition is armed when Stale is a '0' and ClrTo and FIFOres are also a '0.' When Stale becomes a '1,' the time-out is triggered, but not set until any DMA transfer is completed, the FIFO is empty and there is no more than one character left in the pipeline. To clear the time-out condition, set the ClrTo bit in PACR. To re-enable the time-out function, clear the ClrTo bit.

The CPU may arm the time-out by writing directly to the Stale Data Timer Count register (SDTCR). If the timer expires before any data arrives, an interrupt will be given for the time-out condition. If data does arrive before the timer expires, the interrupt is delayed until data becomes stale.

5.4.7 Transmit Direction

NOTE: In the transmit direction, the pipeline will behave in one of two ways, depending on the RLEen Control bit. RLEen should be set *only* by the CPU after the parallel port is in ECP mode, otherwise compression of data will occur but cannot be supported in data transfers on the parallel port. If RLEen is a '0,' data written to the DMAbuffer register by DMA (DMAen true) or CPU write (DMAbufWe true) will be moved through PFHR1 to PFHR2 and immediately transferred into the FIFO (if there is space available in the FIFO).

If RLEen is a '1,' run-length encoding is enabled and comparators among the pipeline stages come into play to recognize repeated strings of characters and compress them (see Figure 5-8). To allow the comparator-based logic to work, the pipeline registers PFHR1 and PFHR2 must be kept full.

One comparator determines if the characters in PFHR1 and PFHR2 are identical. Another comparator determines if the next character coming from the DMAbuffer register and the character in PFHR1 are identical. Compression is begun when the pipeline is full (immediately after a DMA or CPU write to the DMA buffer) and both comparators show identical characters in their pipeline stages. This starts the compression process and the character in PFHR1 and the character in the DMA buffer are shifted forward. The (same) character in PFHR2 is not loaded into the FIFO, but rather the RLCR is incremented to a '1.' As additional characters are transferred from the DMA buffer, the RLCR is incremented if as the characters in PFHR1 and PFHR2 are the same. When the repeated sequence is finally broken, or the RLCR count reaches 127, the RLCR value is transferred into the FIFO, the RLCR is zeroed, and the character in PFHR2 is transferred into the FIFO. Compression resumes when both comparators again indicate the presence of a string of at least three identical characters. During intervals between DMA transfers, the last two data characters are held in PFHR1 and PFHR2.

After the entire block transfer is completed, the CPU must either zero RLEen or ensure that both DMAen and DMAbufWe are zeroes. When either of these conditions is true, the pipeline is released and data held in PFHR1 and PFHR2 are transferred into the FIFO.

The time-out interrupt may be used as a general timer interrupt in the transmit direction. Unlike the receive case, when DMAdir is true, the time-out status is set immediately when the time-out is triggered by a '0'-to-'1' transition of Stale. To use the time-out interrupt, the CPU must load the desired time delay directly into the Stale Data Timer Count register (SDTCR). When the timer expires, Stale becomes true and the time-out interrupt is given.

5.5 Parallel Port Overview

5.5.1 Terminology

This document uses the terms 'master' and 'slave' for the IEEE-1284-specification terms 'host' and 'peripheral', which are used to describe the two sides of a parallel-port interface.

5.5.2 Signal Names

The IEEE-1284 specification uses different names for the nine control signals, depending on the current mode of operation (see Table 5-2). The CL-CD1283 uses fixed names for each of its pins. The names were chosen to represent the most commonly used names amongst the various protocols. The CL-CD1283 device operates as a slave only. There are four input-control signals driven by the master-side device and five output-control signals driven by the slave-side device. The Parallel Data bus PD[7:0] is bidirectional.

5.5.3 State Machine

The parallel port is controlled by a large synchronous state machine. The state machine is based on the IEEE STD 1284 specification (D2.00) and conforms to all the functional modes (except extensibility link options, which are not currently defined, as of the print date of this document).

5.5.4 Configuration

At power-up, the interface begins in Compatibility mode (Centronics mode), ready to accept data from the master. Only the ETXFR (Enable Transfer) bit in the PCR (Parallel Configuration register) is required to allow transfers in Compatibility mode (parallel port only, datapath section is separate). PCR bits are used to enable negotiations, transfers and Manual mode.

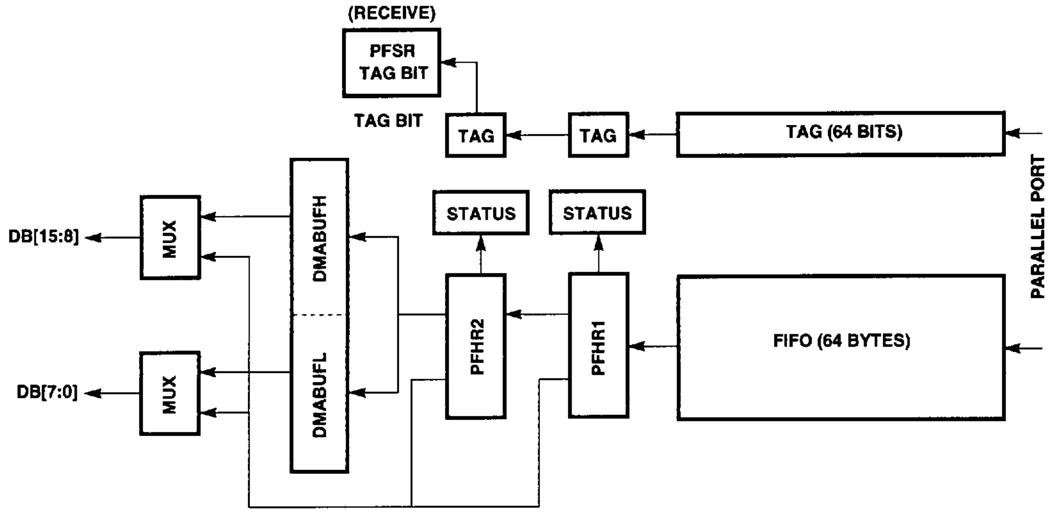


Figure 5-5. FIFO Data Path Functional Diagram, Receive

5.5.5 Interrupts

Interrupts are enabled in the Parallel Channel Interrupt Enable register (PCIER). Interrupt status can be read in the Parallel Channel Interrupt Status register (PCISR). These two registers have the same format.

5.5.6 Manual Mode

Manual mode allows direct control of the five output control signals and the PD bus. It is not intended for data transfers, but rather for advanced diagnostics. Enter Manual mode by setting the MANMD bit.

MMDIR sets the direction of the PD bus — '0' is the input, and '1' is the output. When MMDIR is set to a '1' data for the PD bus comes from the Manual Data register (MDR). MANOE controls the three-state buffer on the PD bus — a '0' is equal to floating, and a '1' is equal to driving. When MMDIR is a '0', MANOE is ignored, the PD lines are inputs and the data may be read in MDR.

5.5.7 Control Signals

Output signals are controlled by the Output Value register (OVR). The degree of control depends on the current mode. In Manual mode, all five signals are under user control. In Compatible and EPP modes, only three signals are available and the others are set by the state machine.

The Input Value register (IVR), Zero Detect register ZDR), Ones Detect register (ODR) and Signal Status register (SSR) are used to monitor the four input signals. These four registers all have a common format. The IVR always shows the values of the four input pins. The ZDR and ODR registers allow the user to force interrupts on specific signal transitions. Bits set in ZDR will cause an interrupt if the specified signal changes from a '1' to a '0'. Similarly, bits set in ODR will cause an interrupt if the specified signal changes from a '0' to a '1'. Setting both bits will cause interrupts on either transition. The SSR shows the status of signal changes

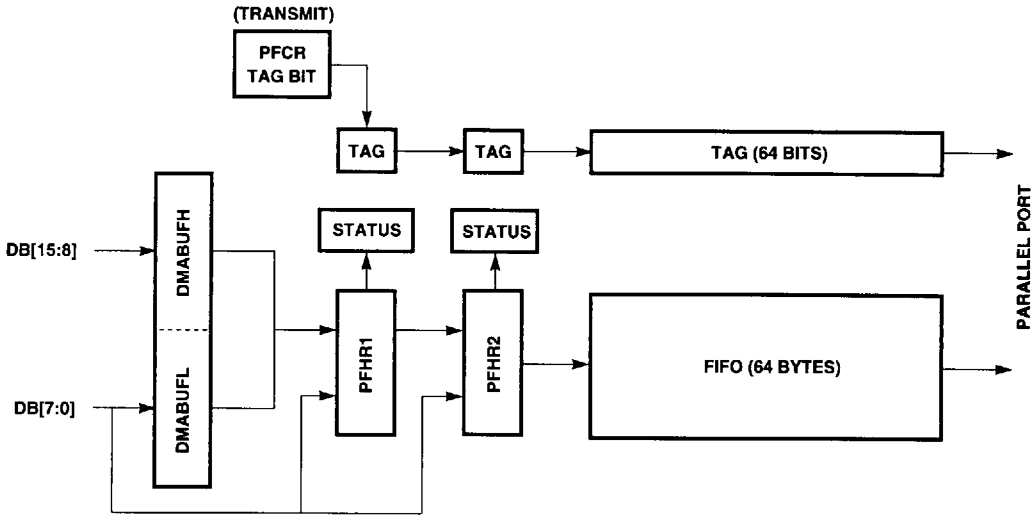


Figure 5-6. FIFO Data Path Functional Diagram,Transmit

according to ZDR/ODR. SSR shows which signal changed in the specified direction. (It is necessary for the user to read IVR to determine how the signal changed.) The signal change interrupt is enabled with the SIGCH bit in PCIER.

5.5.8 Parallel Port Interface to the FIFO

The DMAdir indicates the current direction ('0' is equal to in, '1' is equal to out) of transfers between the FIFO and DMA logic; due to a recent negotiation, this may differ from the parallel-port determination of direction. For example, the CPU must set DMA registers after it receives an interrupt showing a direction change. The FIFOlock bit of the Parallel Auxiliary Control register (PACR) stops the DMA pipeline, which may be useful in diagnostics. FIFOlock is also used in ECP and EPP modes to stop data transfers in the forward direction (input).

5.5.9 IEEE 1284-Protocol Negotiations

All IEEE 1284-protocol negotiations are initiated by the master side. The CL-CD1283 role is simply to accept or reject the attempted negotiation. The CL-CD1283 Negotiation Enable register (NER) contains bits which individually enable specific IEEE-1284 modes.

The various IEEE-1284 modes require negotiations on the parallel interface before they can be entered. Until a successful negotiation sequence has completed, the interface remains in Compatibility mode. These negotiations occur in two stages; both stages occur automatically after the device has been commanded to begin the negotiation procedure to a particular mode. The result of the requested negotiation will appear in the Negotiation Status register (NSR).

The first stage determines whether the slave is IEEE 1284-compatible. Once this has been determined, the interface continues the process to determine if the mode being requested is supported.

The slave must enable the E1284 bit in PCR in order for negotiations to occur. Data transfers may be enabled after the CPU receives the interrupt for Negotiation Change (NEGCH). Data transfers require that the ETXFR bit in the PCR be set, negotiations may occur without data transfer enabled.

Negotiation Status Register

After any IEEE-1284 negotiation or termination, the current protocol status can be read in the Negotiation Status register (NSR). NEGOK and NEGFL indicate successful and failed attempts. INVAL shows that the mode terminated from an invalid state. Termination from valid states are reported as successful with NEGOK. A 4-bit code is displayed in the lower portion of the NSR to indicate the results of successful negotiation.

Special Command Register

The bits in the Special Command register (SCR) cause actions on the parallel port. Set Pause (SetPS) and Clear Pause (ClrPs) provide a means of controlling data movement into the CL-CD1283 from the remote master. This function is used in Compatibility mode to provide a method of posting error status to the remote. Errors can only be presented to the master by the slave during the active BUSY period. SetPs will cause the CL-CD1283 to stop transfers in the BUSY state on reception of the first character following the setting of the bit. When the error status has been delivered, ClrPs will restore the parallel interface to the normal running state.

EPIRQ is used to send an interrupt pulse in EPP mode. The REVRQ bit is used to indicate that data is available for reverse transfer in either Compatible or ECP modes. These operations are described in more detail below, in the relevant protocol sections.

5.5.10 Data Transfers

In Compatibility mode, incoming HstClk (STROBE*) pulses activate PerBsy (BUSY), and the data on the PD lines is held in latches. PerBsy protects the data latches by signalling to the master that it is not ready for more transfers. After the HstClk pulse has ended, a pulse is sent on PerClk (ACK*) to acknowledge the receipt of the data into the holding latches. After the data has been moved from the latches to the FIFO, PerBsy is lowered to signal readiness for the next character.

All other data transfer modes require IEEE-1284 negotiations.

5.5.11 Compatibility Mode Status

The IEEE-1284 specification requires that the three Compatibility mode status lines (SELECT, nFault, and PError) must not be asserted unless PerBsy (BUSY) is high. PerBsy can only be activated in response to a received character, and PerBsy must remain high until the status condition (for example, paper out) is changed.

To send these status signals to the master device, the Set Pause (SetPs) bit of the SCR should be set along with the appropriate bit in OVR for each of the status signals. The next arriving character will activate PerBsy, which will stay active until Clear the Pause (ClrPs) bit of the SCR is set. No data is lost in this operation.

Table 5-2. Signal Names

Names	Compatibility	Rev. NB	Rev. BT	ECP	EPP
Inputs					
A_1284	SLCTIN*	A_1284	A_1284	A_1284	nAStrb
HstBsy	AUTOFD*	HstBsy	HstBsy	HstAck	nDStrb
HstClk	STROBE*	HstClk	HstClk	HstClk	nWrite
nInit	INIT*	nInit	nInit	nRevReq	nInit
Outputs					
AkDaRq	PError	AkDaRq	AkDaRq	nAkRev	USER1
PerBsy	BUSY	PerBsy	PerBsy	PerAck	nWait
PerClk	ACK*	PerClk	PerClk	PerClk	Intr
nDatAv	FAULT*	nDatAv	nDatAv	nPerReq	USER2
Xflag	SELECT	Xflag	Xflag	Xflag	USER3

5.6 IEEE 1284 Parallel Protocol Support

5.6.1 Reverse-Nibble and Reverse-Byte Modes

These modes support reverse transfers only, from slave to master. Reverse-Nibble mode is enabled with NER bit 0, Reverse-Byte mode is enabled with NER bit 1. Reverse Nibble mode sends four bits at a time over four of the peripheral status lines. The advantage of this scheme is that with software drivers any unidirectional PC parallel port may be used for bidirectional data transfers. Reverse-Byte mode requires bidirectional buffers on the PC hardware, but allows substantially faster transfers, because it moves a byte at a time.

There is no mechanism in Compatibility mode for the slave to show that data is available for reverse transfer. The master must 'poll' the slave by negotiating into a reverse mode and examining the NDatAv signal. The REVRQ bit in SCR is used to instruct the CL-CD1283 to advertise the availability of data to the master via the nDatAv signal during negotiation.

5.6.2 ID Request

ID request is enabled with a combination of NER bit 6, and one of four other transfer mode bits. ID requests can be made in conjunction with ECP, ECP/RLE, Reverse-Byte and Reverse-Nibble modes; there is no ID Request function defined for EPP mode. The CL-CD1283 can accept an ID request in any mode that is enabled to handle transfers. The IDREQ interrupt bit is set when an ID request is received in any enabled mode.

5.6.3 ECP Mode

ECP mode allows bidirectional transfers and supports a RLE (run-length encoding) compression scheme. The ability to expand RLE-coded data is required of all IEEE-1284 ECP-compliant devices, but the ability to compress data is optional. The CL-CD1283 handles both expansion and compression in the datapath section. The parallel port simply passes the inverse of the command signal to and from the FIFO on the ninth 'tag' bit. ECP mode is enabled by NER bit 2. RLE mode requires both bits 2 and 3.

The handshake is identical for both ECP and RLE modes. The control signals HstBsy and PerBsy are used (in the forward and reverse directions, respectively) to indicate command and address options. If HstBsy/PerBsy is low, the upper bit of the byte is examined — a '0' means that the lower 7 bits should be interpreted as an address, while a '1' means they should be used as an RLE repeat count. This count shows the number of times that the next data character is to be consecutively repeated in the data stream.

The master device is responsible for determining the direction of transfer. The slave can request a direction change, but the master actually changes the direction. ECP mode always begins in the forward direction, from master to slave. The CPU sets REVRQ (bit 0 of SCR) to request reverse transfers. Once the master changes direction, the REVRQ bit will automatically clear, and the Direction Change (DIRCH) interrupt status will appear in PCISR (if enabled in the PCIER).

The master device will switch the direction of the interface for forward transfers when the slave indicates that no more data is available.

5.6.4 EPP Mode

Data transfers use the DMA pipeline and FIFO. Address transfers are handled out of band, not in the FIFO stream. When the slave receives address write command, it will deposit address into the EPP Address register (EAR), and assert an EPPAW interrupt request. When the slave receives a read address command, the contents of EAR will be returned.

5.7 Protocol Timing

The IEEE-1284 specification timing parameter T_P specifies the minimum pulse width and the minimum setup time as 500 ns. The Short Pulse register (SPR) must be loaded with the number of system clock ticks equivalent to 500 ns.

CLK Freq.	Time/Tick	SPR Value	T_P Width
16 MHz	62.5 ns	8	500
20 MHz	50 ns	10	500
25 MHz	40 ns	13	520

5.8 General-Purpose I/O Port

The CL-CD1283 provides an 8-bit general-purpose port (GP[7:0]) that can be used for control or status of external functions. Each of the eight signals is individually programmable for direction, so the port can be comprised of any number of inputs and outputs. Each port signal is implemented with a standard, bidirectional HCMOS pad and is fully TTL-compatible. The port is controlled via two internal registers, the GP Direction Control register (GPDIR) and the GP Data register (GPIO).

Each bit in the GPDIR sets the direction of the corresponding bit in the GPIO; a '1' sets the signal as output and a '0' sets it as input. When writing to the GPIO, only the bits programmed as outputs are affected by the contents of the data bus; when reading the GPIO, bits programmed as inputs will reflect the true state of the condition of the external pin; bits programmed as outputs will reflect the state of the last value written to the register and thus the current state of the output pins.

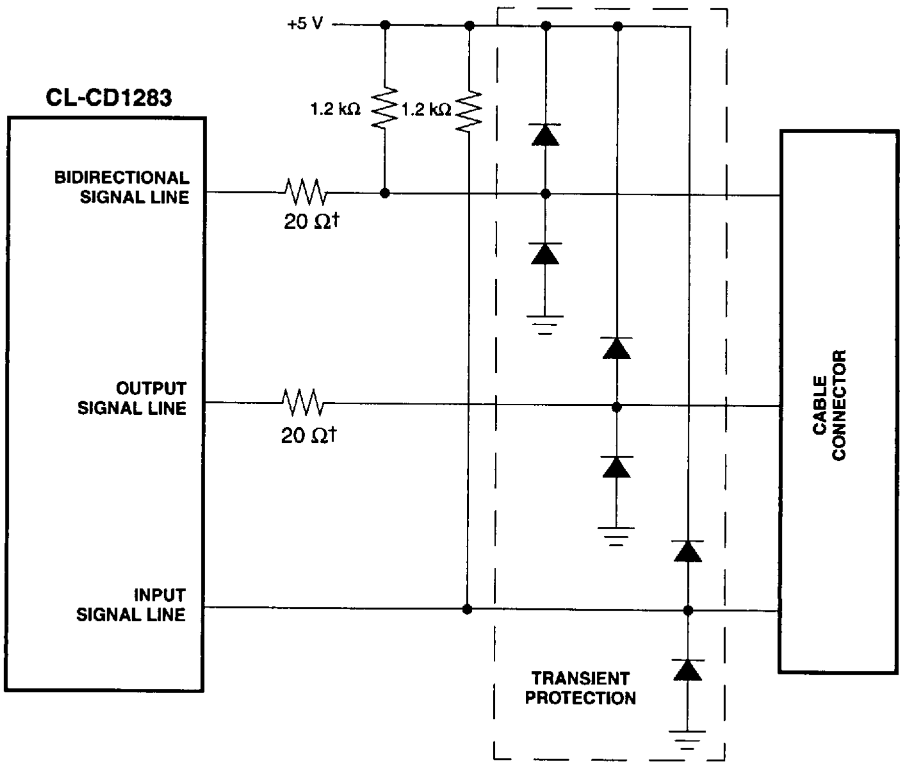
At reset, all bits in the GPIO are cleared and the signals are programmed as inputs.

Note that interrupts are not generated on signal changes within the General-Purpose I/O port; the CPU must periodically poll this register to detect changes in external conditions. Therefore, it is recommended that the port be used with signals that change with low duty cycles if it is necessary to detect changes.

5.9 Parallel Port Interface

The CL-CD1283 parallel port signals are implemented with Level-2 characteristics, as defined in the IEEE STD 1284 specification with the exception of transient protection. As such, the port can be connected directly to the interface cable with the addition of a few external components. The components consist of passive pull-up resistors, series impedance matching resistors, and clipping diodes. Additional noise filtering may be required in an end system. Figure 5-7 shows a typical interface with the components listed above.

Some system designs may require buffers between the CL-CD1283 and the cable. Such systems might drive cables longer than the specified maximum of 10 meters, and thus need buffers, or provide protection for the CL-CD1283 by using inexpensive buffers between it and the cable. The device provides two signal outputs, PDBEN and EBDIR, that can be useful in connecting and controlling buffers, such as 74AS245 or equivalent. These signals do not allow direct control of the buffer; however the addition of an exclusive-NOR gate will provide both an enable control signal and a signal to select the direction of the buffer. PDBEN and EBDIR are outputs from the control state machine that indicate its current state (see Figure 5-8).



† This value is preliminary; when the output impedance of the driver pads is known, this may need to be adjusted to correctly match the impedance of the cable.

Figure 5-7. Cable Connection

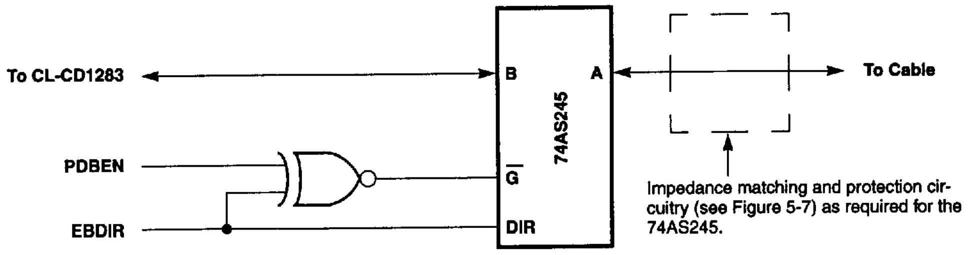

To CL-CD1283
PDBEN
EBDIR
B
A
G
DIR
74AS245
To Cable
Impedance matching and protection circuitry (see Figure 5-7) as required for the 74AS245.

Figure 5-8. External Buffer Control

5.10 Hardware Configurations

The simplicity of the CPU interface to the CL-CD1283 allows the device to be designed into systems that employ popular microprocessors such as the Intel 80X86 family (8086, 80286, 80386, etc.) and the Motorola® family (68000, 68010, 68020, etc.).

An example of CL-CD1283 configuration for a laser-printer application is shown in Figure 5-9. This example provides a parallel interface as well as general-purpose I/O for static control/status.

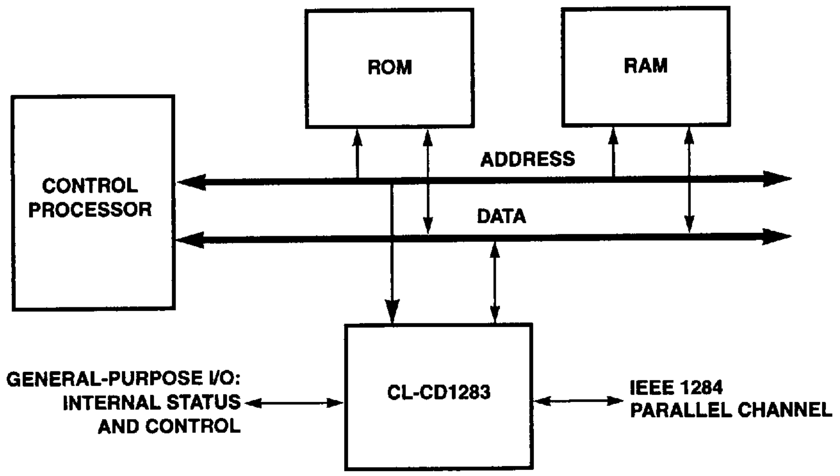

ROM
RAM
CONTROL PROCESSOR
ADDRESS
DATA
GENERAL-PURPOSE I/O: INTERNAL STATUS AND CONTROL
CL-CD1283
IEEE 1284 PARALLEL CHANNEL

Figure 5-9. Sample System Block Diagram

5.10.1 Interfacing to an Intel® Microprocessor-Based System

With very little additional logic, the CL-CD1283 can be interfaced to any system based on a processor in the Intel 80X86 family. Figure 5-10 shows a generalized view of an I/O-mapped interface with an 80286-based system. 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-CD1283 has completed the I/O request.

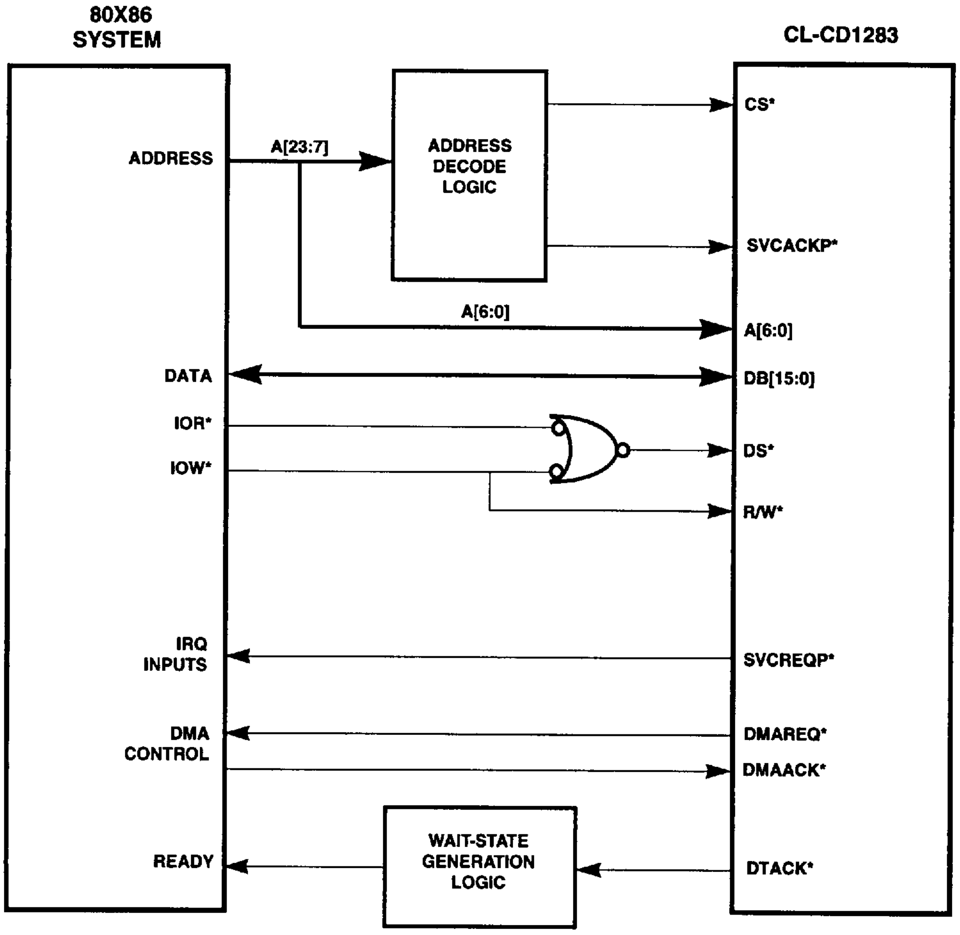


Figure 5-10. Intel® 80X86 Family Interface

5.10.2 Interfacing to a Motorola® Microprocessor-Based System

Interfacing to a 68000 family device is relatively simple. 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 in Figure 5-11 shows a generalized interface to a 68020 device.

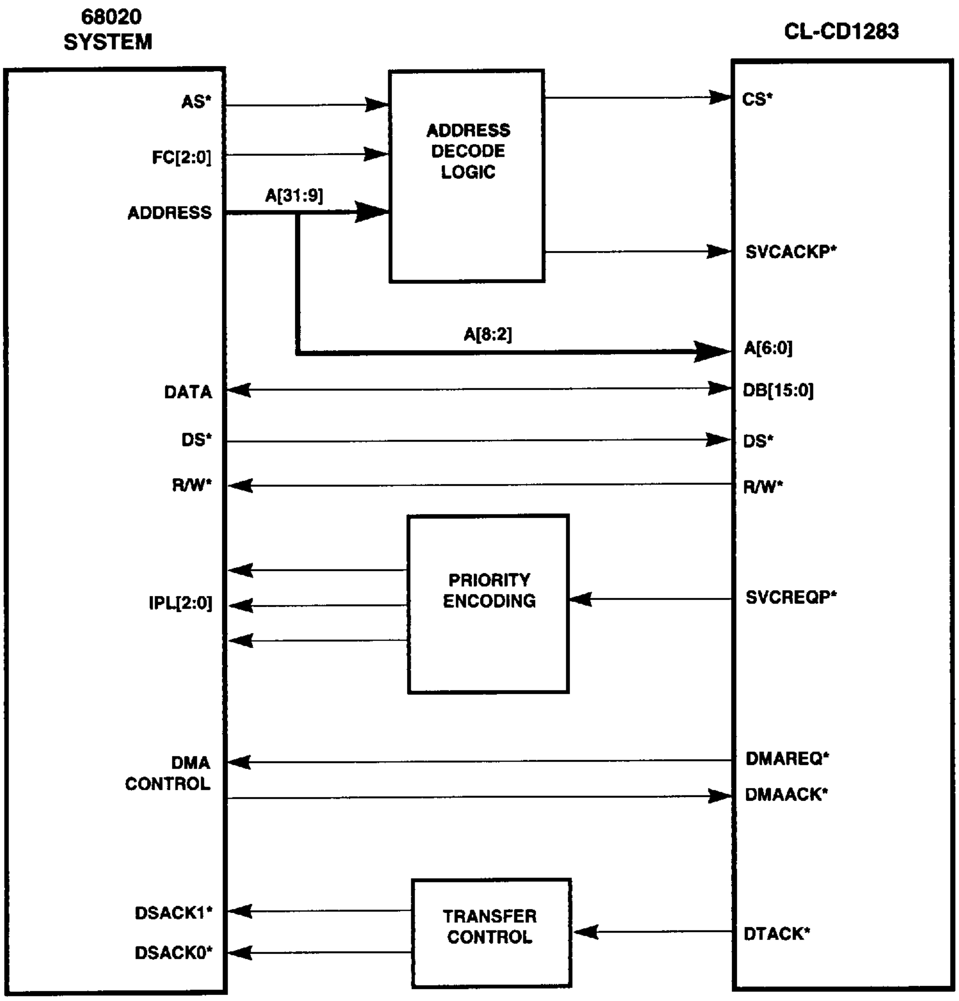


Figure 5-11. Motorola® 68020 Interface

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

6. ELECTRICAL SPECIFICATIONS

6.1 Absolute Maximum Ratings

Supply voltage (V_{CC})	+7.0 V (volts)
Input voltages, with respect to ground	–0.5 V to V_{CC} +0.5 V
Operating temperature (T_A)	0°C to 70°C
Storage temperature	–65°C to 150°C
Power dissipation	0.25 W (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 recommended operating conditions is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

6.2 Recommended Operating Conditions

Supply voltage (V_{CC})	5 V ± 5%
Operating free air ambient temperature	0°C < T_A < 70°C
System clock	25 MHz

6.3 DC Characteristics

(@ V_{CC} = 5 V ± 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 Note 1)
V_{OL}	Output low voltage		0.4	V	I_{OL} = 2.4 mA (see Note 2)
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 tristate 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		50	mA	CLK = 25 MHz
C_{IN}	Input capacitance		10	pF	
C_{OUT}	Output capacitance		10	pF	

2136639 0009077 216

The signals specific to the parallel port meet all requirements of the IEEE STD 1284 specification except for input signal protection (–2.0 to + 7.0 V); external circuitry is required to meet this specification:

- Symmetrical input/output drive: ±12 mA
- Controlled voltage slew rate: 0.4 V/µs
- Input hysteresis: 0.8 V

NOTES:

1) V_{IH} is 2.7 V minimum on RESET* and CLK.

2) V_{OL} for open drain signals is 0.5 V @ 10 mA sinking.

3) While the CL-CD1283 is a highly dependable device, there are a few guidelines which will help to ensure 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-CD1283 are strongly recommended. Noise typically occurs when either the CL-CD1283'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-CD1283, we have specified the input-high voltage (V_{IH}) of the CLOCK and RESET pins is specified at 2.7 V, instead of the TTL-standard 2.0 V. This eliminates any sensitivity to ground bounce, even in very noisy systems.

   Although 2.7 V is higher than the industry-standard 2.4-V 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 V 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 ensure fast, clean edges. Note that the RESET pin may, if desired, be pulled up passively with a 1-kΩ (or less) resistor.

6.4 AC Characteristics

6.4.1 Asynchronous Timing

Refer to the Figures 6-1 through 6-7 for the reference numbers in the following table.

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

Ref. #	Figure	Parameter	MIN	MAX	Unit
t_1	6-1	RESET* low pulse width	10		T_{CLK}
t_2	6-3	Address setup time to CS* or DS*		−20	ns
t_3	6-3	R/W* setup time to CS* or DS*		−10	ns
t_4	6-3	Address hold time after CS*	0		ns
t_5	6-3	R/W* hold time after CS*	0		ns
t_6	6-3	DTACK* low to read data valid		10	ns
t_7	6-3	DTACK* low from CS* or DS[2]	2 T_{CLK}	4 T_{CLK} + 40	ns
t_8	6-3	Data Bus tristate after CS* or DS* high	0	30	ns
t_9	6-3	CS* or DGRANT* high from DTACK* low	0		ns
t_{10}	6-3	DTACK* inactive from CS* or DGRANT* and DS* high		40	ns
t_{11}	6-3	DS* high pulse width	10		ns
t_{12}	6-4	Write data valid from CS* and DS* low		1T_{CLK}	ns
t_{13}	6-4	Write data hold time after DS* high	0		ns
t_{14}	6-2	Clock period (T_{CLK})[1, 3]	40.0	1000	ns
t_{15}	6-2	Clock low time[1]	0.3 T_{CLK}	0.7 T_{CLK}	ns
t_{16}	6-2	Clock high time[1]	0.3 T_{CLK}	0.7 T_{CLK}	ns
t_{17}	6-5	Setup time, SVCACK* to DS* and DGRANT*	10		ns
t_{18}	6-6	Setup time, DMAACK* to rising edge of CLK	10		ns
t_{19}	6-6	Hold time, read data after rising edge of DMAACK	20		ns
t_{20}	6-7	Setup time, write data to rising edge of CLK		−10	ns
t_{21}	6-3	DTACK* active pull-up time[4]			

NOTES:

1) Timing numbers for RESET* and CLK in this table are valid for both asynchronous and synchronous specifications. The device will operate on any clock with a 40–60 duty cycle or better.

2) On host-I/O cycles immediately following SVCACK* cycles and writes to EOSRR, DTACK* will be delayed by 20 CLKs (1 µs @ 20 MHz, 800 ns @ 25 MHz). 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-CD1283 will not be accessed until after this time period.

3) As T_{CLK} increases, device performance decreases. A minimum clock frequency of 25 MHz is required to guarantee performance as specified. The recommended maximum TCLK is 1000 ns.

4) DTACK* sources current (drives 'high') until the voltage on the DTACK* line is approximately 1.5 V; then DTACK* goes to the 'open-drain' (high-impedance) state.

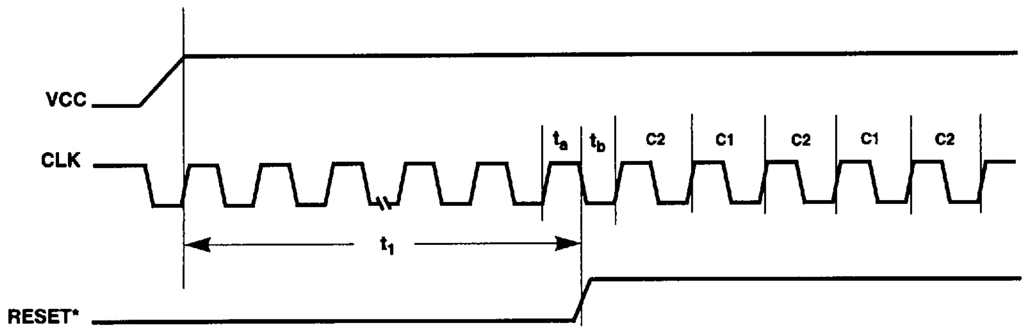


Figure 6-1. Reset Timing

NOTE: For synchronous systems, it is necessary to determine 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 rising edge of the clock; t_b is defined as 5 ns, minimum, before the next rising edge of the clock. If these conditions are met, the cycle starting after the second rising edge will be C1. See the synchronous timing diagrams for additional information. Clock numbers are not important in asynchronous systems.

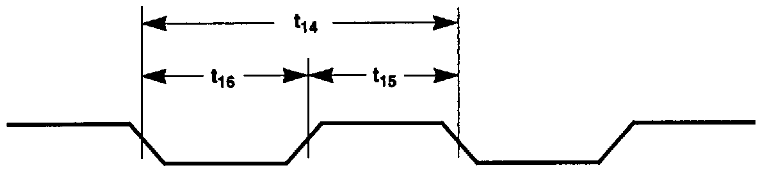


Figure 6-2. Clock Timing

2136639 0009080 800

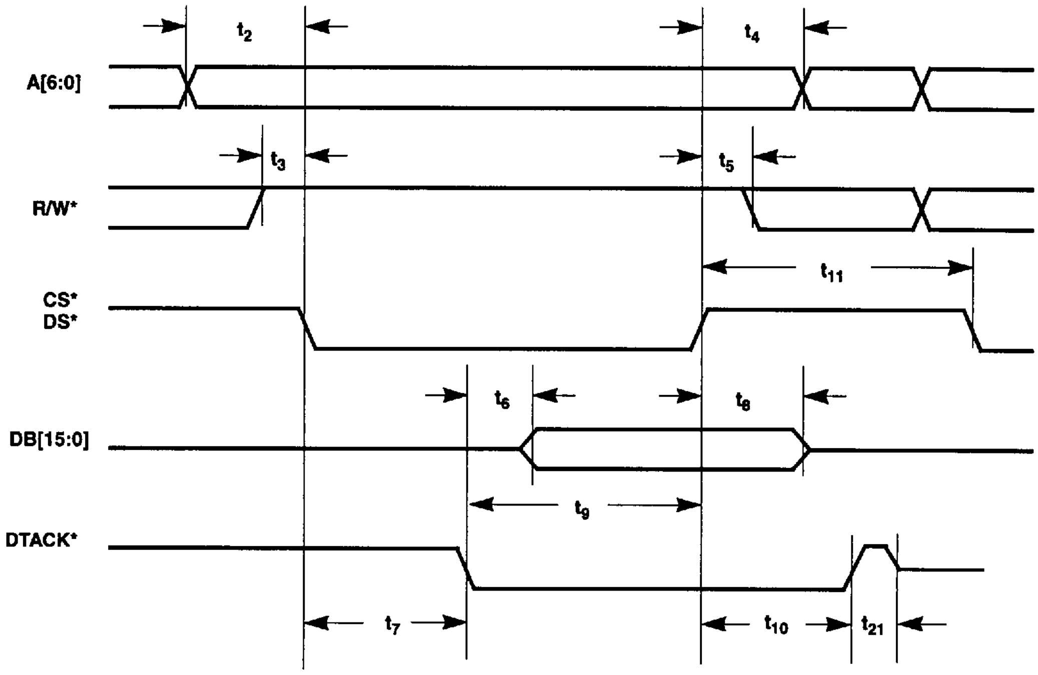


Figure 6-3. Asynchronous Read Cycle Timing

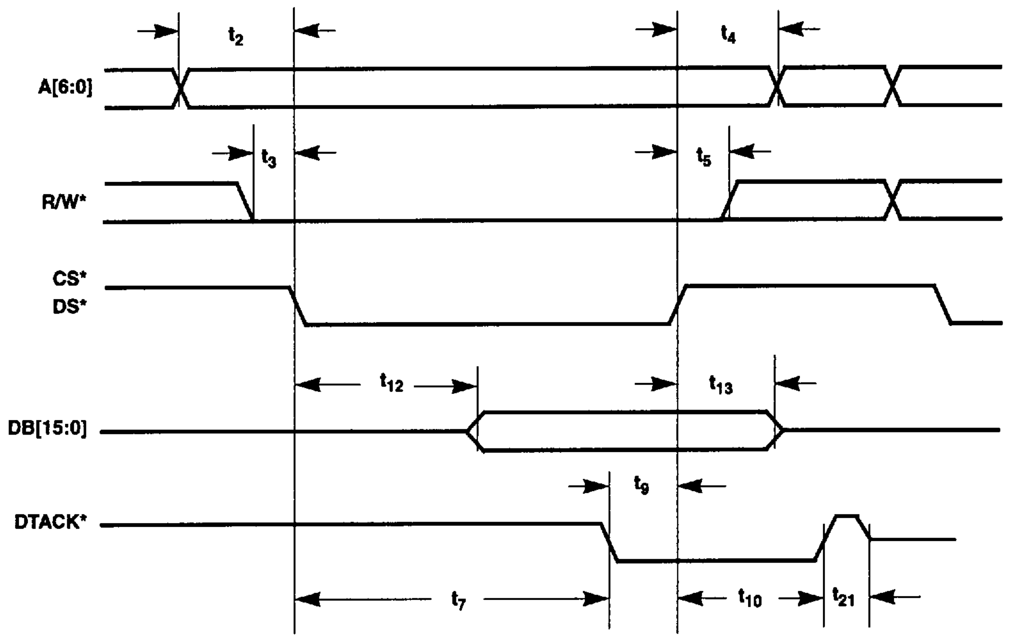


Figure 6-4. Asynchronous Write Cycle Timing

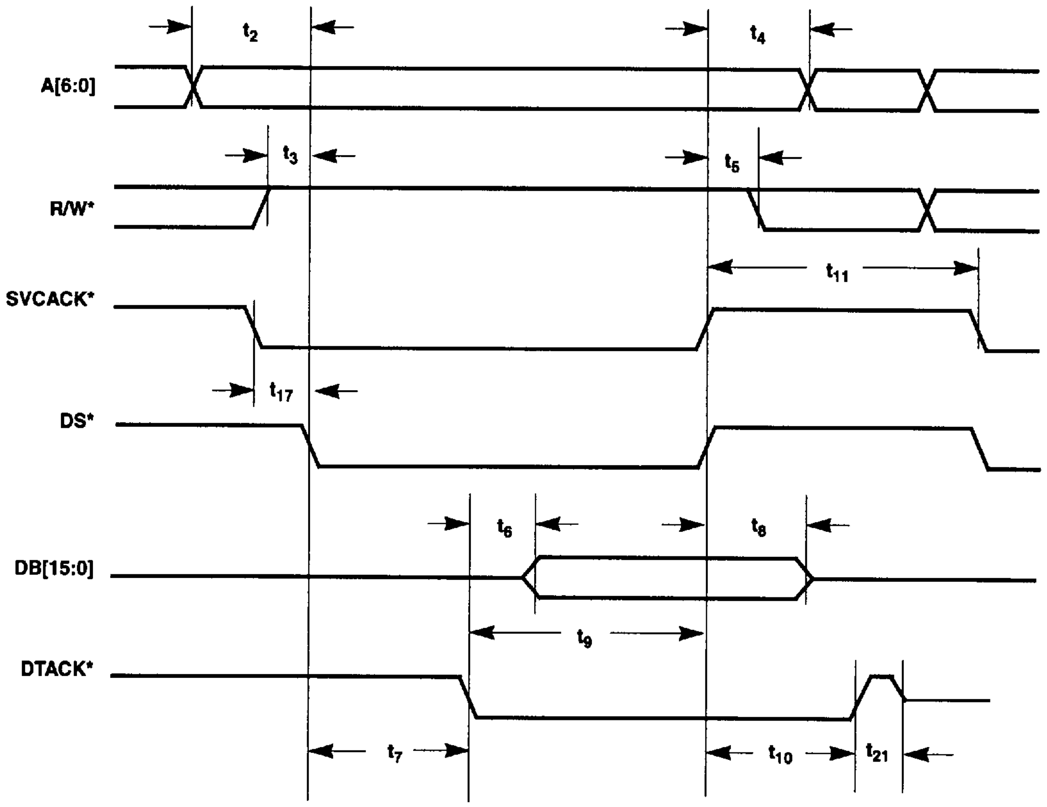
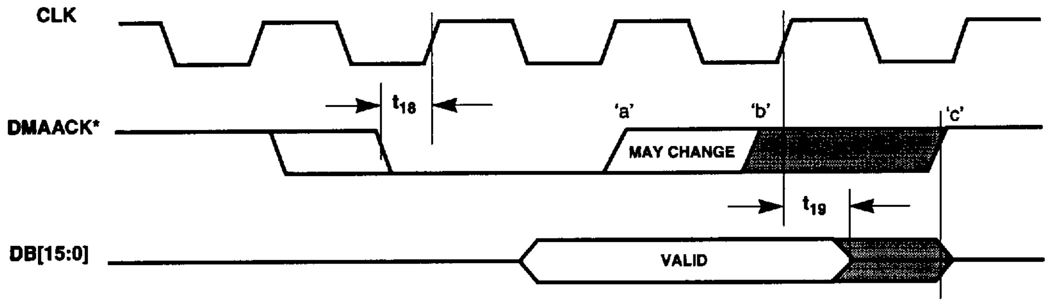


Figure 6-5. Asynchronous Service Acknowledge Cycle Timing



NOTES:

1) The DMA handshake operates in asynchronous mode only if the AsyncDMA bit is set in PACR.
2) If DMAACK* is released after point 'a,' but before point 'b' (before the rising edge of the third full CLK cycle), DB[15:0] will be released at t_{HX} following the rising edge of CLK. If DMAACK* is held past this edge, it controls the release of DB[15:0]; the data bus will remain active until DMAACK* becomes inactive (point 'c').

Figure 6-6. Asynchronous DMA Read Cycle Timing

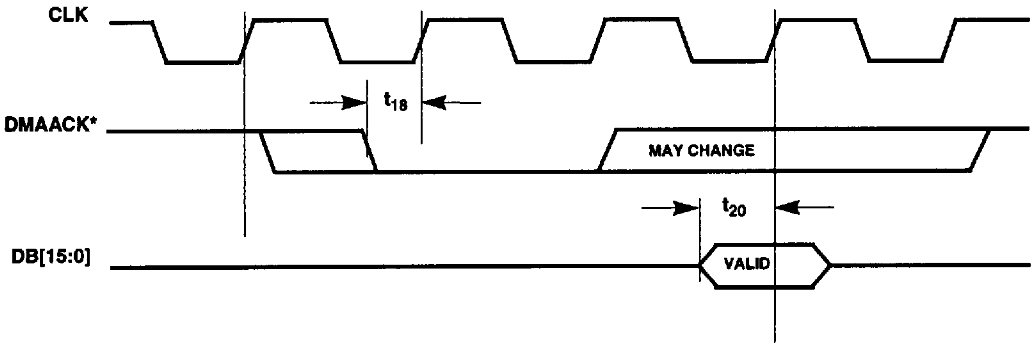


Figure 6-7. Asynchronous DMA Write Cycle Timing

2136639 0009084 456

6.4.2 Synchronous Timing

Use the following table as a reference to timing parameters of figures in this section.

Ref. #	Figure	Parameter	MIN	MAX	Unit
t_1	6-8	Setup time, CS* and DS* to C1 rising edge	15		ns
t_2	6-8	Setup time, R/W* to C1 rising edge	15		ns
t_3	6-8	Setup time, address valid to C1 rising edge	20		ns
t_4	6-8	C3 rising edge to data valid		60	ns
t_5	6-8	DTACK* low from C3 rising edge		30	ns
t_6	6-8	CS* and DS* trailing edge to data bus high-impedance		30	ns
t_7	6-8	CS* and DS* inactive between host accesses	10		ns
t_8	6-8	Hold time, R/W* after C3 rising edge	20		ns
t_9	6-8	Hold time, address valid after C3 rising edge	0		ns
t_{10}	6-9	Setup time, write data valid to C3 rising edge	0		ns
t_{11}	6-10	Setup time, DS* and DGRANT* to C1 rising edge	20		ns
t_{12}	6-10	Setup time, SVCACK* to DS* and DGRANT*	10		ns
t_{13}	6-9	Hold time, write data valid after C3 rising edge	0		ns
t_{14}	6-11	Rising edge CLK to falling edge DMAREQ*		20	ns
t_{15}	6-11	Hold time, DMAREQ* after DMAACK* falling edge, last DMA cycle	15		ns
t_{16}	6-12	Setup time, write data to rising edge C3		−10	ns
t_{17}	6-12	Setup time, falling edge DMAACK* to C1 falling edge	10		
t_{18}	6-12	Data valid before falling edge of C2 (read cycle)		20	
t_{19}	6-12	Data hold time, after rising edge DMAACK*		20	
t_{20}	6-8	DTACK* active pull-up time (see Note 2)			

NOTES:

1) On host I/O cycles immediately following SVCACK* cycles and writes to EOSRR, DTACK* will be delayed by 20 CLKs (1 µs @ 20 MHz, 800 ns @ 25 MHz). 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-CD1283 will not be accessed until after this time period.
2) DTACK* sources current (drives 'high') until the voltage on the DTACK* line is approximately 1.5 V; then DTACK* enters the 'open-drain' (high-impedance) state.

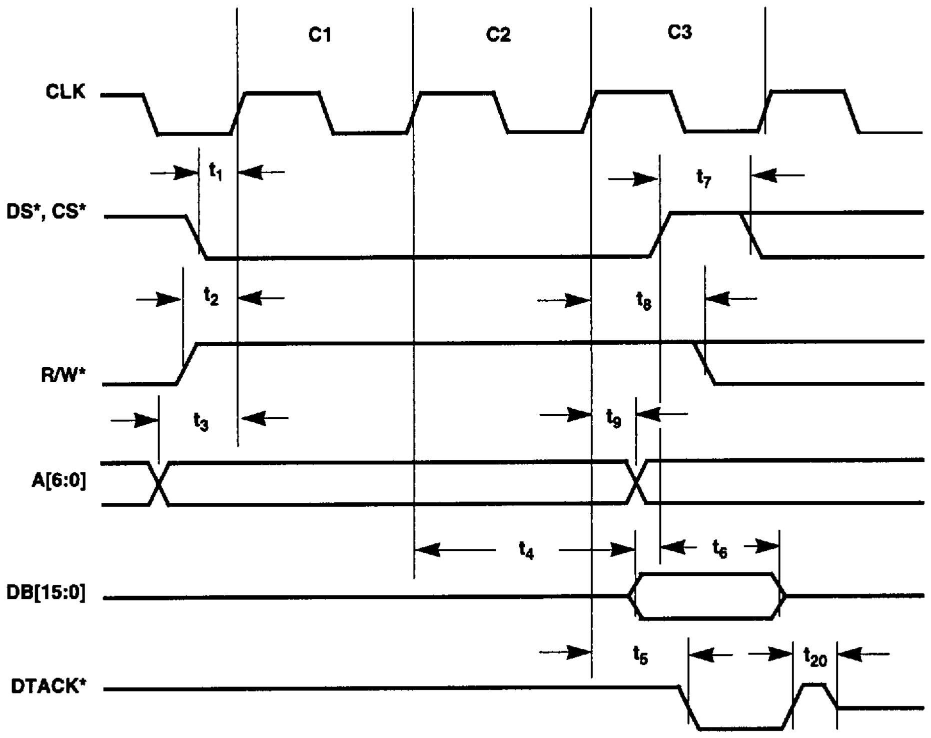


Figure 6-8. Synchronous Read Cycle Timing

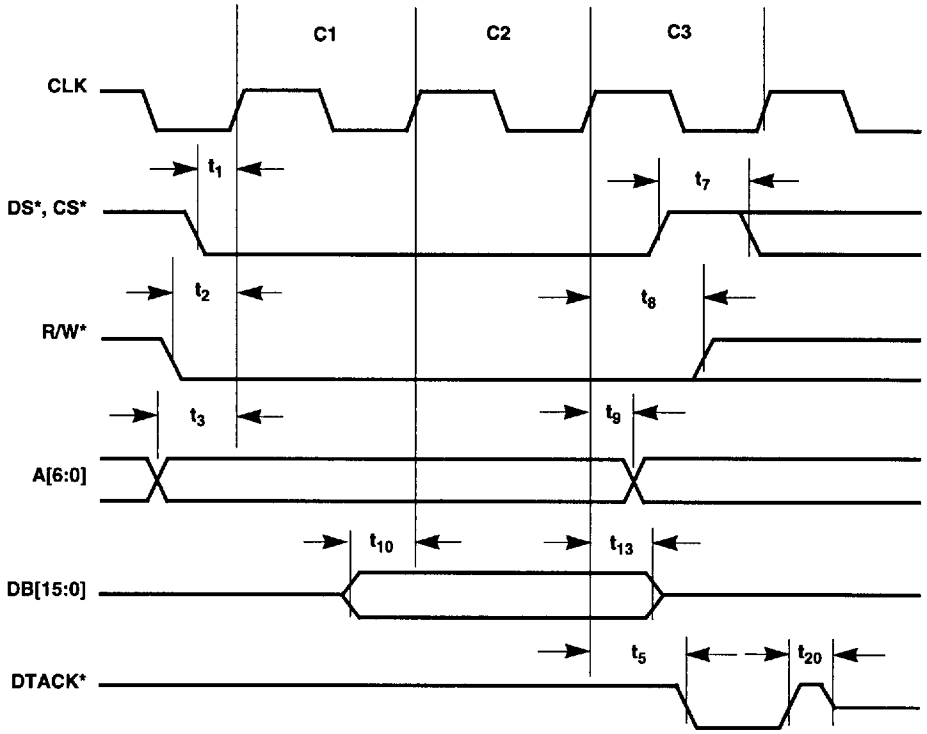


Figure 6-9. Synchronous Write Cycle Timing

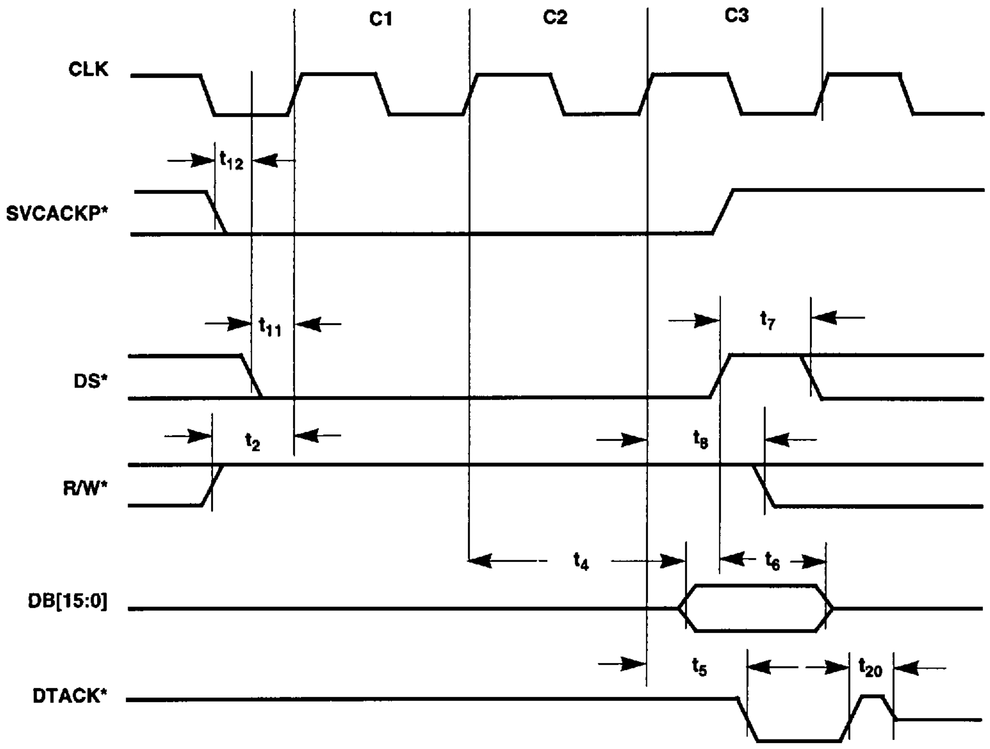


Figure 6-10. Synchronous Service Acknowledge Cycle Timing

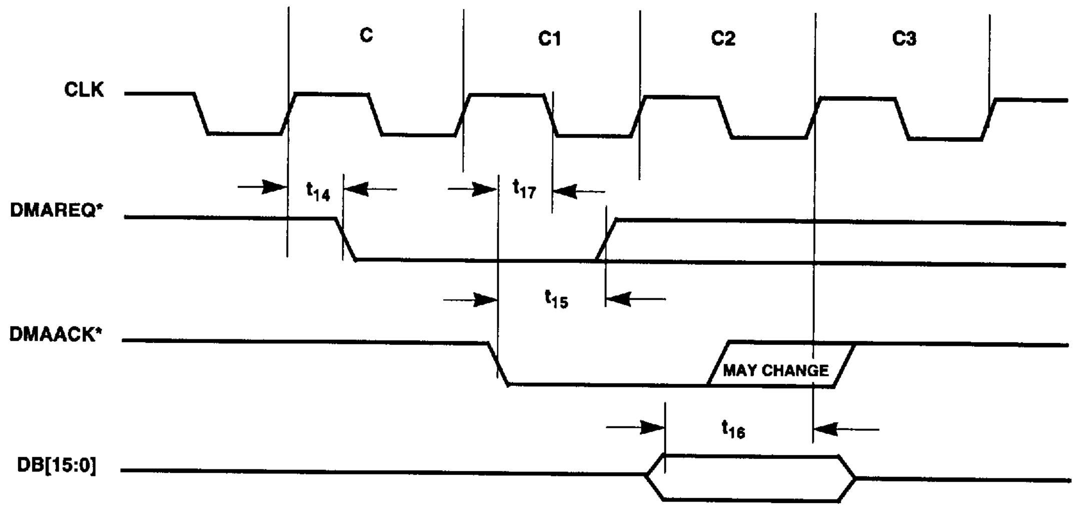


Figure 6-11. Synchronous DMA Write Cycle Timing

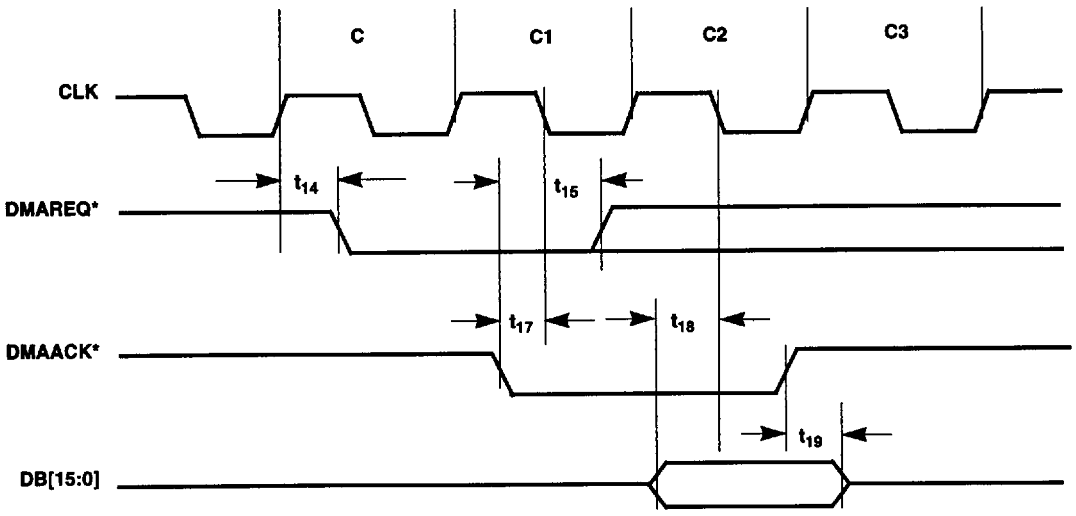
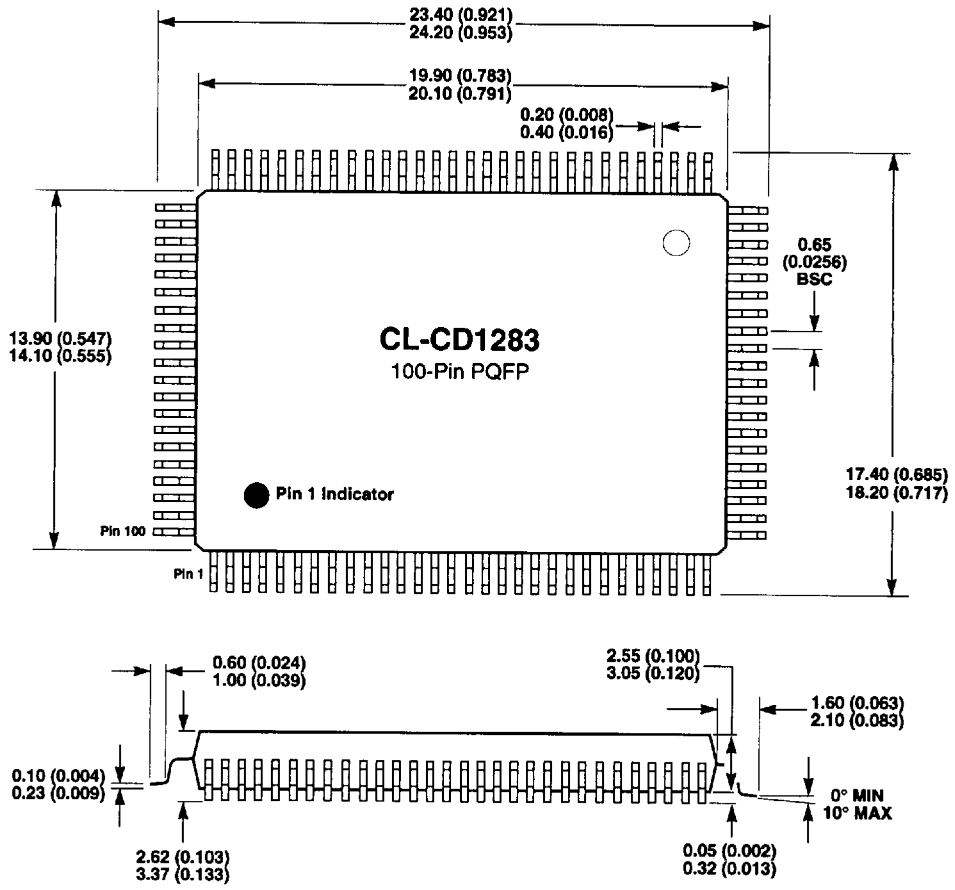


Figure 6-12. Synchronous DMA Read Cycle Timing

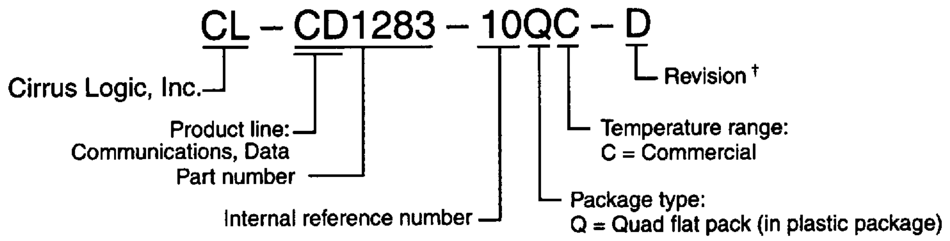
7. PACKAGE DIMENSIONS



NOTES:
1) Dimensions are in millimeters (inches), and controlling dimension is millimeter.
2) Before beginning any new design with this device, please contact Cirrus Logic for the latest package information.

8. ORDERING INFORMATION

The order number for the CL-CD1283 is:


CL – CD1283 – 10QC – D
Cirrus Logic, Inc.
Product line: Communications, Data
Part number
Internal reference number
Revision †
Temperature range: C = Commercial
Package type: Q = Quad flat pack (in plastic package)

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

INDEX

2136639 0009093 469