# CD1283

## *IEEE 1284-Compatible Parallel Interface*

**Datasheet**

# Product Features

### Parallel Port (Peripheral-side)

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-Mbytes/sec. transfer rate in ECP and EPP modes
- 64-byte parallel FIFO with DMA interface
  - —64-byte FIFO can accommodate up to 4 Kbytes of compressed data with RLE (run-length encoded) compression enabled
- Supports peripheral-side operation
- Data and control input/output pads support IEEE STD1284 level-2 interface specification
- CPU bus interface
  - —High-speed slave DMA handshake interface
  - —Three clocks per word DMA transfers
  - —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 interrupts simplify interrupt service routines

### General

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

As of May 2001, this document replaces the Basis Communications Corp. document *CL-CD1283 — IEEE 1284-Compatible Parallel Interface*.

May 2001



**Datasheet**

# Contents


**1.0 Overview** ..... 9
1.1 Advantages ..... 9
**2.0 Conventions** ..... 11
**3.0 Pin Information** ..... 13
3.1 Pin Diagram ..... 13
3.2 Pin List ..... 14
3.3 Pin Descriptions ..... 16
**4.0 Register Summary** ..... 19
4.1 Register Summary Tables ..... 19
**5.0 Functional Description** ..... 22
5.1 Device Architecture ..... 22
5.2 CPU Interface ..... 22
5.2.1 Read Cycles ..... 24
5.2.2 Write Cycles ..... 24
5.2.3 Service-Acknowledge Cycles ..... 24
5.2.4 DMA Cycles ..... 24
5.2.5 Interrupts ..... 25
5.2.6 DMAREQ* as Interrupt Source ..... 25
5.2.7 Daisy-Chain Configurations ..... 26
5.3 Parallel Port Service Requests ..... 27
5.3.1 Hardware-Activated Acknowledge ..... 32
5.3.2 Software-Activated Acknowledge ..... 32
5.4 Parallel Port FIFO and Data Pipeline ..... 32
5.4.1 IEEE Standard 1284 Protocols ..... 33
5.4.2 Bus Interface ..... 33
5.4.3 Parallel Port FIFO ..... 34
5.4.4 Receive Direction ..... 34
5.4.5 Receiving Compressed Data ..... 35
5.4.6 Stale Data (Stale, OneChar, and Timeout Status Bits) ..... 35
5.4.7 Transmit Direction ..... 36
5.5 Parallel Port Overview ..... 36
5.5.1 Terminology ..... 36
5.5.2 Signal Names ..... 37
5.5.3 State Machine ..... 37
5.5.4 Configuration ..... 37
5.5.5 Interrupts ..... 37
5.5.6 Manual Mode ..... 38
5.5.7 Control Signals ..... 38
5.5.8 Parallel Port Interface to the FIFO ..... 39
5.5.9 IEEE 1284-Protocol Negotiations ..... 39
5.5.10 Data Transfers ..... 40
5.5.11 Compatibility Mode Status ..... 40
5.6 IEEE 1284 Parallel Protocol Support ..... 40

5.6.1 Compatibility Mode ..... 40
5.6.2 Reverse-Nibble and Reverse-Byte Modes ..... 40
5.6.3 ID Request ..... 41
5.6.4 ECP Mode ..... 41
5.6.5 EPP Mode ..... 41
5.7 Protocol Timing ..... 41
5.8 General-Purpose I/O Port ..... 42
5.9 Parallel Port Interface ..... 42
5.10 Hardware Configurations ..... 44
5.10.1 Interfacing to an Intel, Microprocessor-Based System ..... 45
5.10.2 Interfacing to a Motorola, Microprocessor-Based System ..... 46

**6.0 Programming** ..... 47

6.1 Overview ..... 47
6.2 Initialization ..... 47
6.2.1 Device Reset ..... 47
6.2.2 Service Acknowledge Handling ..... 49
6.3 ASCII Code Tables ..... 51

**7.0 Detailed Register Descriptions** ..... 53

7.1 Global Registers ..... 53
7.1.1 Access Enable Register ..... 53
7.1.2 Global Firmware Revision Code Register ..... 53
7.1.3 General-Purpose I/O Direction Register ..... 54
7.1.4 General-Purpose I/O Register ..... 54
7.1.5 Parallel Interrupt Register ..... 54
7.1.6 Prescaler Period Register ..... 55
7.1.7 Service Request Register ..... 55
7.2 Virtual Registers ..... 56
7.2.1 End-of-Service Request Register ..... 56
7.2.2 Parallel Interrupt Vector Register ..... 56
7.3 Parallel Pipeline Registers ..... 57
7.3.1 Data Error Register ..... 57
7.3.2 DMA Buffer Data Register ..... 58
7.3.3 Holding Register Status Register ..... 58
7.3.4 Host Timeout Value Register ..... 59
7.3.5 Local Interrupt Vector Register ..... 60
7.3.6 Parallel Auxiliary Control Register ..... 61
7.3.7 Parallel Channel Reset Register ..... 61
7.3.8 Parallel FIFO Control Register ..... 62
7.3.9 Parallel FIFO Empty Pointer Register ..... 63
7.3.10 Parallel FIFO Fill Pointer Register ..... 63
7.3.11 Parallel FIFO Holding Registers ..... 63
7.3.12 Parallel FIFO Quantity Register ..... 64
7.3.13 Parallel FIFO Status Register ..... 64
7.3.14 Parallel FIFO Threshold Register ..... 65
7.3.15 Run-Length Count Register ..... 66
7.3.16 Stale Data Timer Count Register ..... 66
7.3.17 Stale Data Timer Period Register ..... 67
7.4 Parallel Port Registers ..... 67

7.4.1 EPP Address Register ..... 67
7.4.2 Input Value Register ..... 67
7.4.3 Manual Data Register ..... 68
7.4.4 Negotiation Enable Register ..... 68
7.4.5 Negotiation Status Register ..... 69
7.4.6 Ones Detect Register ..... 70
7.4.7 Output Value Register ..... 70
7.4.8 Parallel Channel Interrupt Enable Register ..... 71
7.4.9 Parallel Channel Interrupt Status Register ..... 71
7.4.10 Parallel Configuration Register ..... 71
7.4.11 Special Command Register ..... 72
7.4.12 Short Pulse Register ..... 73
7.4.13 Signal Status Register ..... 74
7.4.14 Zeros Detect Register ..... 74
7.5 Special Register ..... 74
7.5.1 Reset Command Register ..... 74

**8.0 Electrical Specifications** ..... 76

8.1 Absolute Maximum Ratings ..... 76
8.2 Recommended Operating Conditions ..... 76
8.3 AC Characteristics ..... 78
8.3.1 Asynchronous Timing ..... 78
8.3.2 Synchronous Timing ..... 84

**9.0 Package Dimensions** ..... 90

**10.0 Ordering Information** ..... 91

**Index** ..... 93

# Figures


1 Functional Block Diagram ..... 10
2 Functional Block Diagram ..... 23
3 Internal Address Generation ..... 23
4 CD1283 Daisy-Chain Configuration ..... 26
5 Interrupt Generation Logic ..... 28
6 Control Signal Generation ..... 31
7 FIFO Data Path Functional Diagram – Receive ..... 37
8 FIFO Data Path Functional Diagram: Transmit ..... 38
9 Supported Compatibility Mode Timing ..... 40
10 Cable Connection ..... 43
11 External Buffer Control ..... 44
12 Sample System Block Diagram ..... 44
13 Intel, 80x86 Family Interface ..... 45
14 Motorola, 68020 Interface ..... 46
15 Flow Diagram of the CD1283 Master Initialization Sequence ..... 48
16 Polling Flow Chart ..... 51
17 Reset Timing ..... 79
18 Clock Timing ..... 80
19 Asynchronous Read Cycle Timing ..... 80
20 Asynchronous Write Cycle Timing ..... 81
21 Asynchronous Service Acknowledge Cycle Timing ..... 82
22 Asynchronous DMA Read Cycle Timing ..... 83
23 Asynchronous DMA Read Cycle Timing (Two Back-to-Back DMA Reads) ..... 83
24 Asynchronous DMA Write Cycle Timing ..... 84
25 Asynchronous DMA Write Cycle Timing ..... 84
26 Synchronous Read Cycle Timing ..... 86
27 Synchronous Write Cycle Timing ..... 87
28 Synchronous Service Acknowledge Cycle Timing ..... 88
29 Synchronous DMA Write Cycle Timing (Two Back-to-Back 3-Cycle DMA Writes) ..... 89
30 Synchronous DMA Read Cycle Timing (Two Back-to-Back 3-Cycle DMA Reads) ..... 89


# Tables


1 Global Registers ..... 19
2 Virtual Registers ..... 20
3 Parallel Pipeline Registers ..... 20
4 Parallel Port Registers ..... 20
5 Special Register ..... 21
6 LIVR[2:0] Encoding ..... 31
7 System Clock Setup ..... 42
8 Hexadecimal — Character ..... 51
9 Decimal — Character ..... 52
10 PIVR[2:0] Encoding ..... 56
11 SPR Binary Values to Set 500-ns Pulse Widths ..... 74
12 Asynchronous Timing Reference Parameters ..... 78
13 Synchronous Timing Reference Parameters ..... 85

# Revision History

| Revision | Date | Description |
|---|---|---|
| 1.0 | 4/01 | Initial release. |

# 1.0 Overview

The CD1283 is a multi-function interface controller for printers, scanners, tape-drives, set-top boxes, data acquisition, and other applications that require high-speed, bidirectional, parallel communication with a host computer. All modes of the *IEEE STD 1284 Standard Signaling Method for Bidirectional Parallel Peripheral Interface for Personal Computers* specification are supported, including ECP, EPP, Reverse Byte, Reverse Nibble, and Compatible. With full support of this standard, the CD1283 provides compatibility with all types of host parallel ports, including older Centronics®, IBM® PS/2® bidirectional, and the latest IEEE 1284-compliant ports.

The dedicated state-machine design provides the fastest possible response times to all host signal changes, with 100% guaranteed compliance to all IEEE 1284 timing, protocol, and signaling requirements. The CD1283 device, operating at 25 MHz, has signal response times to support 2 Mbytes/sec. transfers, provided that a comparably fast host parallel port is used. This performance headroom guarantees that the faster data rates of future host parallel port implementations will be supported by peripheral applications using the CD1283.

In addition to the dedicated state machine, the CD1283 provides slave DMA support, and a 64-byte FIFO to allow maximum total throughput performance. Interrupts are generated based on status changes of the parallel port. Note, however that interrupts are not generated by FIFO threshold, or FIFO full/empty conditions. The DMA request signal can be used to generate interrupts as long as hardware and software implementation is handled correctly. If maximum performance is not a requirement, the device can be monitored and controlled by polling its detailed status registers.

Another unique feature of the CD128X series of devices is the dedicated hardware for RLE compression/decompression in ECP mode. Special logic is used to perform the ECP-RLE compression/decompression ‘on-the-fly’ while data is moved to and from the FIFO. All of these capabilities above and beyond the requirements of the IEEE 1284 specification permit the use of a less expensive microprocessor by reducing the required CPU bandwidth needed for the parallel port.

To aid in the development of hardware and software, an evaluation kit — complete with application notes and programmer’s guide — is provided along with software examples and evaluation board schematics. The ISA add-in card is designed to demonstrate the capabilities of the CD128X family of devices, and enables software developers to begin testing code while the system hardware is still in development.

## 1.1 Advantages

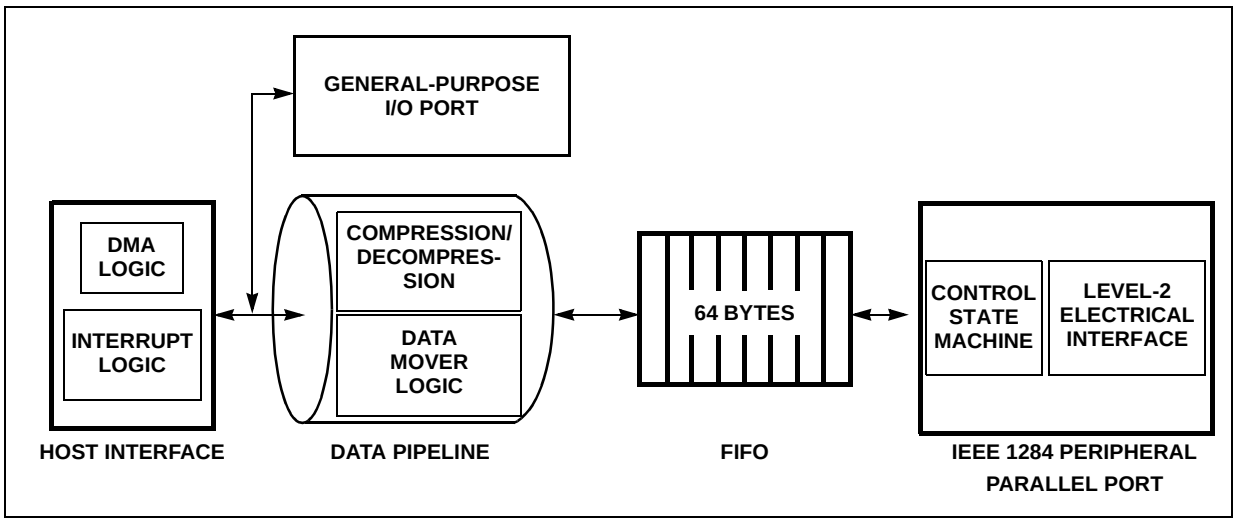
### Unique Features

- Supports IEEE STD 1284 specification
- Hardware support of IEEE STD 1284 timings
- 64-byte FIFO
- Parallel port signals provide level-2 drive characteristics
- DMA channel
- ECP compression/decompression in hardware

## Benefits

- Multi-protocol bidirectional port for a wide range of applications.
  - — Up to 2 Mbytes/sec. transfer rate
  - — Provides future connectivity with new host systems
- Reduces software complexity and guarantees specification compliance.
- High throughput with reduced load on host CPU.
- Direct connection to printer cable; reduces chip count.
- Reduces host interface overhead. High-speed data movement between memory and parallel port.
- Reduces software complexity and increases throughput of compressed-data tranfers.

**Figure 1. Functional Block Diagram**



# 2.0 Conventions

## Abbreviations

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

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’.

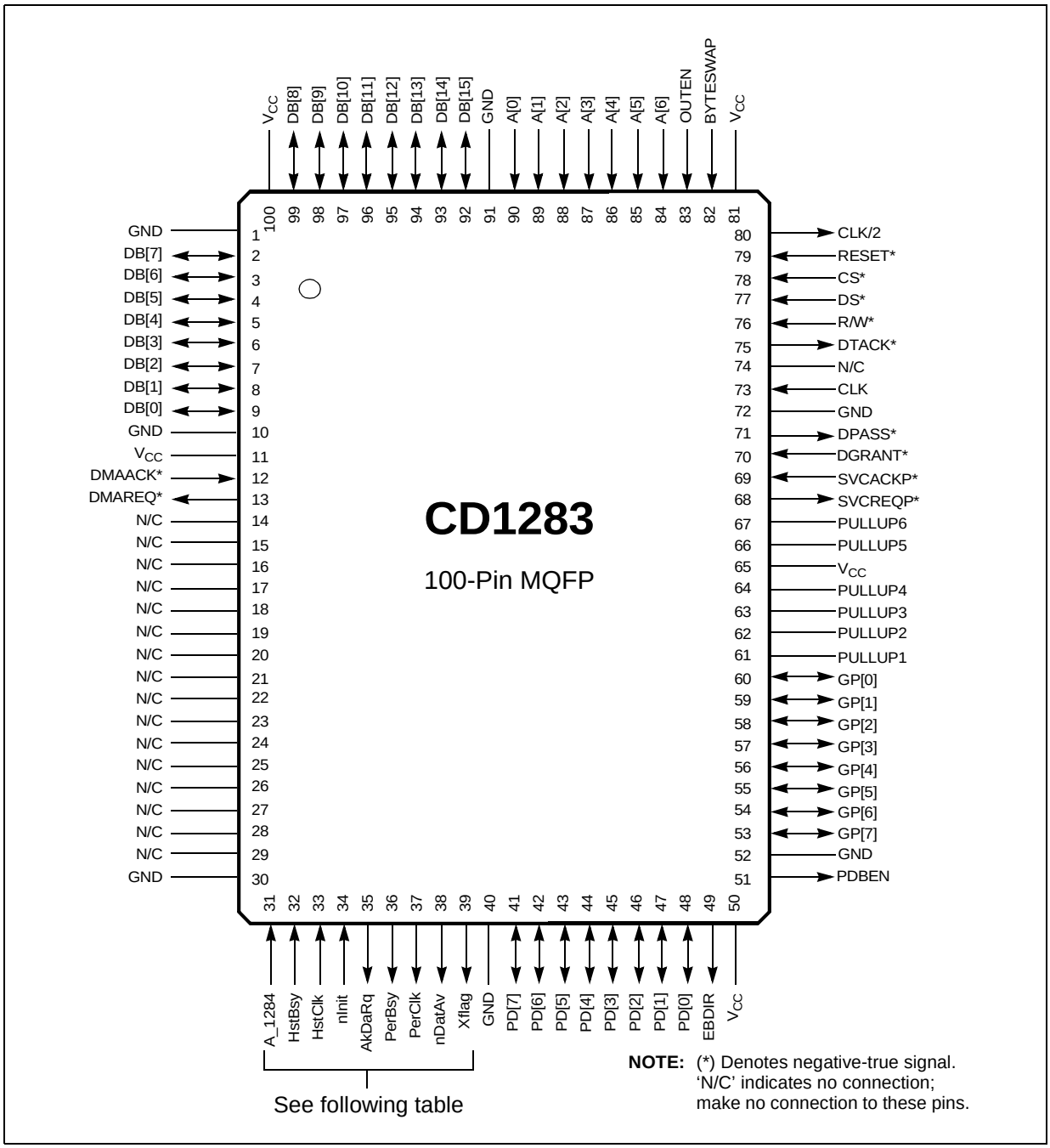
## Acronyms

| Acronym | Definition |
|---|---|
| AC | alternating current |
| BIOS | basic input/output system |
| CISC | complex instruction set computer |
| CMOS | complementary metal-oxide semiconductor |
| DC | direct current |
| DMA | direct-memory access |
| DRAM | dynamic random-access memory |
| ECP | extended capibilities port |
| EPP | enhanced parallel port |
| FIFO | first in/first out |
| GPIO | general-purpose IO |
| HCMOS | high-performance complementary metal-oxide semiconductor |
| HDLC | high-level data link control |

| **Acronym** | **Definition** (Continued) |
|---|---|
| IC | integrated circuit |
| IDC | instruction and data cache |
| ISA | industry standard architecture |
| LSB | least-significant bit |
| MPU | microprocessing unit |
| MSB | most-significant bit |
| PIO | programmed I/O |
| PPP | point-to-point protocol |
| MQFP | metric quad-flat pack |
| RAM | random-access memory |
| RLE | run-length encoded |
| R/W | read/write |
| SDLC | synchronous data link control |
| SRAM | static random-access memory |
| SWI | software interrupt instruction |
| TLB | translation look-aside buffer |
| TTB | translation table base |
| TTL | transitor-transitor logic |
| VRAM | video random-access memory |
| WB | write buffer |

# 3.0 Pin Information

## 3.1 Pin Diagram



| Pin Names | Compati-bility | Reverse Nibble Mode | Reverse Byte Mode | ECP Mode | EPP Mode |
|---|---|---|---|---|---|
| **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 |

## 3.2 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)
- ‘TS’ indicates tristate
- ‘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 | Reset State |
|---|---|---|---|---|
| A[6:0] | I | 7 | 84–90 | |
| BYTESWAP | I | 1 | 82 | |
| CLK | I | 1 | 73 | |
| CLK/2 | O | 1 | 80 | n/a |
| CS* | I | 1 | 78 | |

| Pin Name | Type | Number of Pins | Pin Numbers | Reset State |
|---|---|---|---|---|
| DB[15:0] | I/O | 16 | 92–99, 2–9 | TS |
| DS* | I | 1 | 77 | |
| DTACK* | AR | 1 | 75 | TS |
| OUTEN | I | 1 | 83 | |
| RESET* | I | 1 | 79 | |
| R/W* | I | 1 | 76 | |
| DMAREQ* | O | 1 | 13 | High |
| DMAACK* | I | 1 | 12 | |
| SVCREQP* | OD | 1 | 68 | High |
| SVCACKP* | I | 1 | 69 | |
| DGRANT* | I | 1 | 70 | |
| DPASS* | O | 1 | 71 | High |
| PD[7:0] | I/O | 8 | 41–48 | TS |
| GP[7:0] | I/O | 8 | 53–60 | TS |
| A_1284 | I | 1 | 31 | |
| nInit | I | 1 | 34 | |
| HstBsy | I | 1 | 32 | |
| HstClk | I | 1 | 33 | |
| PerBsy | O | 1 | 36 | Low |
| PerClk | O | 1 | 37 | High |
| AkDaRq | O | 1 | 35 | Low |
| Xflag | O | 1 | 39 | Low |
| nDatAv | O | 1 | 38 | High |
| EBDIR | O | 1 | 49 | High |
| PDBEN | O | 1 | 51 | Low |
| GND | I | 7 | 1, 10, 30, 40, 52, 72, 91 | |
| PULLUP1 | PU | 1 | 61 | |
| PULLUP2 | PU | 1 | 62 | |
| PULLUP3 | PU | 1 | 63 | |
| PULLUP4 | PU | 1 | 64 | |
| PULLUP5 | PU | 1 | 66 | |
| PULLUP6 | PU | 1 | 67 | |
| $V_{CC}$ | I | 5 | 11, 50, 65, 81, 100 | |

## 3.3 Pin Descriptions

| Symbol | Pin No. | Type | Description (Sheet 1 of 3) |
|---|---|---|---|
| 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 DMABUF register.<br>If BYTESWAP is set to '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 '0', the data order is reversed, bits 7:0 are driven with the first byte transferred, and bits 15:8 are driven with the second byte transferred. |
| 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 CD1283. CS* must be set to '1' during DMA read/write operations. |
| DB[15:0] | 92–99, 2–9 | I/O | **BIDIRECTIONAL DATA BUS [15:0]:** 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, bits 7:0, and upper byte, bits 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 $V_{CC}$ with a pull-up resistor. DTACK* is not activated on DMA cycles. |
| OUTEN | 83 | I | **OUTPUT ENABLE:** This pin must be set to '1' to enable output pin functions. When OUTEN is set to '0', it forces all pins that can act as outputs to remain in a tristate condition.<br>OUTEN 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*:** This pin must be set to '1' for a register read operation and set to '0' for a register write operation. 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 device. DMAACK* is the only bus handshake signal recognized during a DMA transfer. (CS* must be high whenever DMAACK* is asserted.) The direction of the DMA transfer is determined by the internal control bit DMAdir. |

| Symbol | Pin No. | Type | Description (Sheet 2 of 3) |
|---|---|---|---|
| SVCREQP* | 68 | OD | **ACTIVE-LOW SERVICE REQUEST PARALLEL:** This is an open-drain output and must be tied to external $V_{CC}$ through a pull-up resistor. Note that this output is only activated by certain conditions on the parallel port (such as, negotiation changes, direction changes, etc.). SVCREQP* is not activated by FIFO threshold, or FIFO full/empty conditions (refer to Chapter 5.0 for information on how to use DMAREQ* to implement a fully interrupt-driven system). |
| SVCACKP* | 69 | I | **ACTIVE-LOW SERVICE ACKNOWLEDGE PARALLEL:** This input must not be driven active except in response to a parallel service request presented by the device. |
| DGRANT* | 70 | I | **ACTIVE-LOW DAISY GRANT:** This input is driven active during service acknowledge cycles to enable the daisy-chain function. This input, when qualified with DS* and a valid service acknowledge (SVCACKP*), activates the service acknowledge cycle. |
| DPASS* | 71 | O | **ACTIVE-LOW DAISY PASS:** This output is driven active during service acknowledge cycles to enable the next device in the daisy-chain. It is driven active when no valid service request exists and the service acknowledge input is active. In multiple CD1283 designs, this signal is normally connected to the DGRANT* input of the next device in the chain. |
| 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 through 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). |
| 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 CD1283. Their functions depend on the transfer protocol selected. Refer to the IEEE Std 1284-1994 document for protocol functions. (See Chapter 10.0 for ordering information.) | | | |
| 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) |

| Symbol | Pin No. | Type | Description (Sheet 3 of 3) |
|---|---|---|---|
| **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 CD1283. Their functions depend on the transfer protocol selected. Refer to the IEEE Std 1284-1994 document for protocol functions. (See Section 5.4.1 on page 33 for ordering information.) | | | |
| 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 CD1283. EBDIR can 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 can be connected directly to the direction control input of a 74245-type device. |
| PDBEN | 51 | O | **PARALLEL DATA BUS ENABLE:** This signal can control a buffer on the Parallel Port data lines in applications requiring more signal-drive capability than provided by the CD1283. PDBEN is controlled by the internal Parallel Port control state machine. When low, the parallel port data bus is not driving; when high, the port is in output mode and is actively driving. PDBEN will toggle between the on and off states during output modes and is only active (high) while the data bus pins are in the active driving state. PDBEN can be logically connected to the enable control of 74245 (or equivalent) bidirectional buffers (see Section 5.9 and Figure 11). |

# 4.0 Register Summary

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

- Global, common to all functions of the device
- Parallel pipeline
- Parallel port
- Service-acknowledge accessible

Global registers are always available to the CPU and their addresses are not affected by the contents of the AER (this register is provided to maintain compatibility with the CD1284).

***Note:*** AER must be set to '00h' and must not be changed (except to access RCR), or access to many registers will not work properly!

The following tables define the register 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 Chapter 7.0 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 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 CD1283 appears at every other address (for example, the CD1283 A[0] input is connected to CPU A[1]). In 32-bit systems, the CD1283 appears at every fourth address (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 CD1283 is placed on data lines D[7:0], which are at odd addresses in the Motorola scheme of addressing. The CD1283 A[0] input is connected with A[1] of the 68000, A[1] with A[2], and so on. Thus, the CD1283 address 0x40 becomes 0x81 to the programmer. It is left-shifted one bit and A[0] must be '1' for low-byte (D[7:0]) accesses.

## 4.1 Register Summary Tables

**Table 1. Global Registers**

| Name | Hex | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Page |
|---|---|---|---|---|---|---|---|---|---|---|
| AER | 68 | Poll | Poll | Poll | Poll | Poll | 0 | 0 | 0 | 53 |
| GFRCR | 4F | Firmware Revision Code | | | | | | | | 53 |
| GPDIR | 71 | Dir 7 | Dir 6 | Dir 5 | Dir 4 | Dir 3 | Dir 2 | Dir 1 | Dir 0 | 54 |
| GPIO | 70 | Data 7 | Data 6 | Data 5 | Data 4 | Data 3 | Data 2 | Data 1 | Data 0 | 54 |
| PIR | 61 | PPIreq | PPort | Pipeline | 0 | 0 | 0 | 0 | 0 | 54 |
| PPR | 7E | Binary Value | | | | | | | | 55 |
| SVRR | 67 | DMAREQ | n/u | n/u | n/u | SRP | n/u | n/u | n/u | 55 |

### Table 2. Virtual Registers

| Name | Hex | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Page |
|---|---|---|---|---|---|---|---|---|---|---|
| EOSRR | 60 | X † | X | X | X | X | X | X | X | 56 |
| PIVR | 40 | X | X | X | X | X | IT2 | IT1 | IT0 | 56 |

† ‘X’ indicates ‘don’t care’.

### Table 3. Parallel Pipeline Registers

| Name | Hex | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Page |
|---|---|---|---|---|---|---|---|---|---|---|
| DER | 33 | DMAwrerr | DMArderr | Bufwrerr | Bufrderr | HR1wrerr | HR1rderr | HR2wrerr | HR2rderr | 57 |
| DMABUF (High byte) | 30 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 58 |
| DMABUF (Low byte) | 30 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | 58 |
| HRSR | 34 | HR1full | HR1tag | HR2full | HR2tag | DMAfull | DMAempty | DMAact | Ctnot0 | 58 |
| HTVR | 24 | HTVR[7] | HTVR[6] | HTVR[5] | HTVR[4] | HTVR[3] | HTVR[2] | HTVR[1] | HTVR[0] | 59 |
| LIVR | 18 | User-Defined Bits | | | | | IT2 | IT1 | IT0 | 60 |
| PACR | 3F | ShrtTen | ShrtStal | StaleOff | FIFOlock | ClearTO | 0 | AsyncDMA | 0 | 61 |
| PCRR | 6C | 0 | 0 | 0 | 0 | 0 | 0 | 0 | PChReset | 61 |
| PFCR | 31 | FIFOres | DMAen | DMAdir | IntEn | RLEen | setTAG | ErrEn | DMAbufWe | 62 |
| PFEP | 39 | 0 | 0 | 6-Bit Binary FIFO Pointer Value | | | | | | 63 |
| PFFP | 38 | 0 | 0 | 6-Bit Binary FIFO Pointer Value | | | | | | 63 |
| PFHR1 | 35 | 8-Bit Character Data | | | | | | | | 63 |
| PFHR2 | 36 | 8-Bit Character Data | | | | | | | | 63 |
| PFQR | 3A | Data or Space Available in FIFO — Max 0x40 | | | | | | | | 64 |
| PFSR | 32 | FFfull | FFempty | Timeout | HRtag | HRdata | Stale | OneChar | DataErr | 64 |
| PFTR | 3B | 0 | DMA Transfer Threshold | | | | | | | 65 |
| RLCR | 37 | 0 | 7-Bit Unsigned Binary Count | | | | | | | 66 |
| SDTCR | 3D | 8-Bit Stale Data Timer Count | | | | | | | | 66 |
| SDTPR | 3C | 8-Bit Stale Data Timeout Value | | | | | | | | 67 |

### Table 4. Parallel Port Registers (Sheet 1 of 2)

| Name | Hex | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Page |
|---|---|---|---|---|---|---|---|---|---|---|
| EAR | 25 | 8-Bit Binary Value | | | | | | | | 67 |
| IVR | 2E | 0 | 0 | 0 | 0 | A1284 | nInit | HstBsy | HstClk | 67 |
| MDR | 21 | 8-Bit Binary Data | | | | | | | | 68 |
| NER | 28 | 0 | RID | 0 | EPP | RLE | ECP | RVB | RVN | 68 |
| NSR | 29 | NegOK | NegFI | HostTO | Invalid | 4-Bit Negotiation Result Code | | | | 69 |

**Table 4. Parallel Port Registers** (Sheet 2 of 2)

| Name | Hex | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Page |
|---|---|---|---|---|---|---|---|---|---|---|
| ODR | 2D | 0 | 0 | 0 | 0 | A1284 | nInit | HstBsy | HstClk | 70 |
| OVR | 2B | PerBsy | PerClk | AkDaRq | xFlag | nDatAv | 0 | 0 | 0 | 70 |
| PCIER | 22 | 0 | 0 | NegCh | SigCh | EPPAW | DirCh | IDReq | nINIT | 71 |
| PCISR | 23 | 0 | 0 | NegCh | SigCh | EPPAW | DirCh | IDReq | nINIT | 71 |
| PCR | 20 | ManMd | E1284 | ETxfr | Ig_SEL | HTmrTst[1] | HTmrTst[0] | MMDir | ManOE | 71 |
| SCR | 2A | 0 | 0 | 0 | 0 | ClrPs | SetPs | EPIrq | RevRq | 72 |
| SPR | 26 | 8-Bit Binary Value | | | | | | | | 73 |
| SSR | 2F | 0 | 0 | 0 | 0 | A1284 | nInit | HstBsy | HstClk | 74 |
| ZDR | 2C | 0 | 0 | 0 | 0 | A1284 | nInit | HstBsy | HstClk | 74 |

**Table 5. Special Register**

| Name | Hex | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Page |
|---|---|---|---|---|---|---|---|---|---|---|
| RCR | 05 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 74 |

# 5.0 Functional Description

## 5.1 Device Architecture

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

To maintain binary compatibility with the CD1284, much of the architectural layout has been duplicated. Therefore, access to the register set of the parallel channel is only possible after loading the AER with the CD1284 occupied parallel port address — namely channel 0. For all channel-specific accesses, the CPU first loads the AER with a pointer to channel 0. Thereafter, all read and write operations occur through the parallel channel.

The parallel channel is comprised of a FIFO and DMA data interface, as well as a high-speed state machine to manage all modes defined in the IEEE Std 1284-1994 specification, *Standard Signaling Method for a Bidirectional Parallel Peripheral Interface for Personal Computers*. The parallel port performs the slave or peripheral function of the IEEE Std 1284 interface, and can accept negotiations into any or all of the IEEE defined modes.

## 5.2 CPU Interface

The CPU interface is comprised of a 7-bit address bus, 8-bit bidirectional data bus, 16-bit DMA port, and control inputs to identify the type of I/O cycle occurring. The signaling and basic timing match that of the Motorola® 68000 family. With the addition of minimal glue logic, the interface will work with nearly any CPU. A special input is provided to swap the bytes on the data bus to reduce the necessary logic needed with Intel®-style CPUs.

The interface is completely compatible with the CD1284. Therefore, a CD1283 can be inserted into a system in instead of a CD1284 and the parallel port operates without any modifications to the CPU interface, parallel port hardware or software.

In most cases, when the CPU reads or writes an internal CD1283 location, it accesses a location in a RAM array to serve as a bank of registers. However, some locations 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 a read of the actual state of an input is necessary (such as a parallel port handshake signal in the IVR).

**Figure 2. Functional Block Diagram**

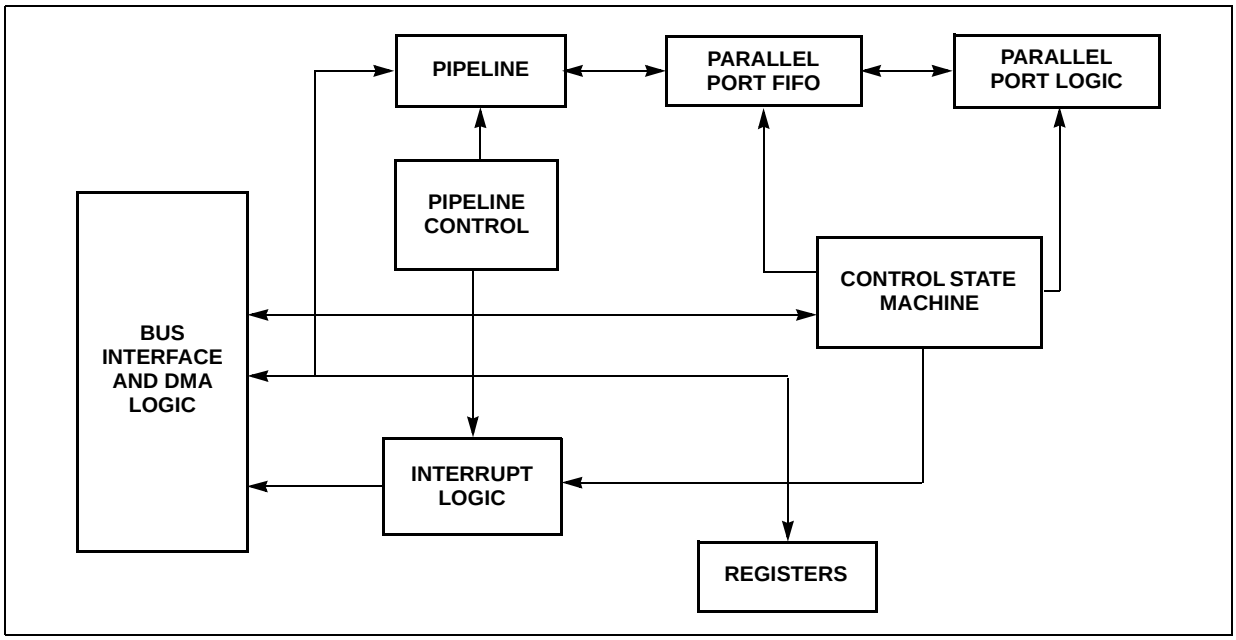
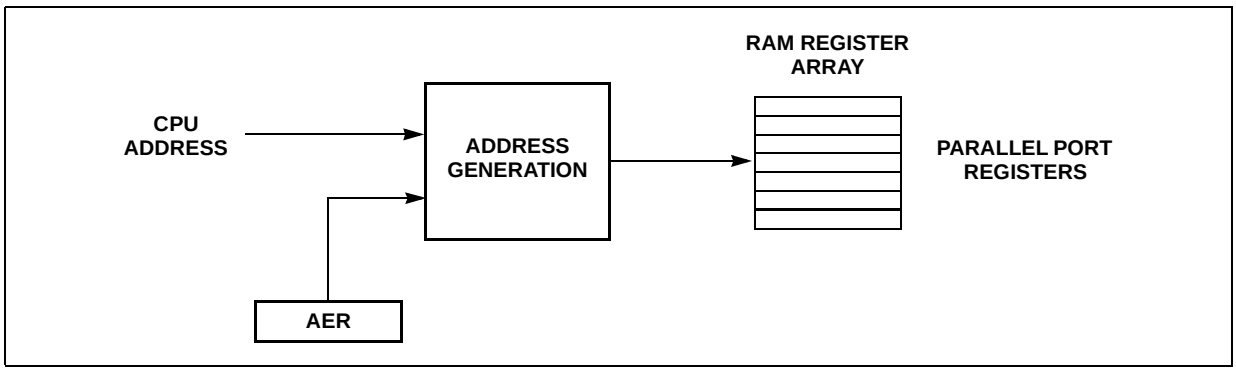



**Figure 3. Internal Address Generation**




The CD1283 is a synchronous device. All internal operations occur on edges and levels (phases) of the internal clock. The internal clock is generated by dividing the external (system) clock by two. When the CPU performs an I/O cycle with the CD1283, it strobes; address and data are sampled on the rising edges of the internal clock. As illustrated in Chapter 8.0, external control signals must meet setup times with respect to system clock edges. Once a cycle starts, the sequence of events is locked to the CD1283 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 CD1283. In an asynchronous design, the DTACK* (Data Transfer Acknowledge) signal indicates that the CD1283 has completed the requested data transfer for all I/O cycles except DMA. DTACK* can be an input to wait-state generation logic that pauses the CPU until the operation is complete. If the CS* and DS* strobes (Chip Select and Data Strobe) do not meet the minimum setup time with respect to the system clock edge, the CD1283 does not detect the I/O request, and the cycle delays for two full-system clock cycles, meeting the setup time. The I/O cycle commences and follows the predictable timing with DTACK* signaling the end.

### 5.2.1 Read Cycles

Read cycles are initiated when both the CS* and DS* inputs are activated and the R/W* (read/write) input is high. All strobes and address inputs must meet the setup times as specified in Chapter 8.0. Both the CS* and DS* signals must be valid for a cycle to start. Cycle times are measured from whichever of the two signals goes active last. The CD1283 signals the completion of the read cycle (placing the data from the addressed register on the data bus pins) by activating DTACK*. The read cycle terminates when the CPU removes CS* and DS*.

### 5.2.2 Write Cycles

Write cycle timing and strobe activity is nearly identical to read cycles except that the R/W* signal must be held low. Write data, strobes, and address inputs must meet setup and hold times as specified in Chapter 8.0. DTACK* indicates that the cycle is complete and the CD1283 has accepted 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, but 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 CD1283 provides during the read cycle is the contents of the PIVR. As with read and write cycles, DTACK* indicates the end of the cycle and removing DS* and SVCACK* terminates the cycle.

***Note:*** With regard to timing and service-acknowledge cycles, when the CPU completes the service routine and writes to the EOSRR, a subsequent I/O cycle, if started immediately, is delayed approximately 1 µs by delaying DTACK*. This is due to the time required by the internal processor to complete activities associated with the service-acknowledge cycle.

These activities are primarily interrupt-logic updates and restoration of the environment prior to the service-request/service-acknowledge procedure. These must be completed before any internal registers are modified by the CPU.

If the CPU attempts an access before the internal procedures are complete, the CD1283 will hold off the cycle until it is ready. In system designs that monitor DTACK*, this is not a problem; the cycle is extended until DTACK* becomes active, and the delay is automatically met. If a system design does not monitor DTACK*, a mechanism must be provided to introduce the required delay.

***Warning:*** Failure to observe the above delay requirement can cause device malfunction.

### 5.2.4 DMA Cycles

The 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 previously described.

The handshake between the CD1283 and the DMA circuitry uses two signals: DMAREQ* and DMAACK*. The address bus is ignored during DMA transfers. When internal conditions warrant a DMA transfer (as when the FIFO falls below the programmed threshold in the forward direction or rises above the threshold in the reverse direction) and DMA transfers are enabled through the PFCR, the device requests DMA service by driving DMAREQ* low. DMAREQ* remains active until the FIFO has less than two empty byte locations remaining (forward direction) or until the

FIFO has less than 2 bytes remaining (reverse direction). In the forward direction, 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 has less than two empty byte locations remaining or there is no more data to send. In the reverse direction, the CD1283 responds to the active DMAACK* signal by driving the contents of the DMABUF register onto the data bus.

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

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 the forward direction, or from the FIFO data pipeline to the DMA buffer first in the reverse direction. If BYTESWAP is low, then the least-significant byte (DB[7:0]) immediately moves into or out of the data pipeline. If BYTESWAP is high, the opposite occurs (DB[15:8] move into or out of the pipeline first).

The effective duration of the DMA transfer block (burst) is determined by the threshold value in the PFTR. Regardless of where the port is moving data when this threshold is reached (exceeded in receive; less than in transmit), a DMA cycle begins and remains active until the FIFO has less than 2 bytes remaining (receive) or has less than two empty byte locations remaining (transmit).

The SVRR provides can determine if a DMA cycle is being requested. SVRR[7] is true if a DMA cycle is currently being requested. This status indication is provided as a general system status.

Refer to Chapter 8.0 for detailed information on DMA cycle options and timing values.

### 5.2.5 Interrupts

The term interrupt is a generalized description of the method where the CD1283 gains the attention of the CPU. Interrupt is used interchangeably with ‘service request’ as the two are the same function. Interrupt often describes 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 CD1283. The hardware signal generated by the CD1283 (SVCREQP*) can be connected to the CPU interrupt input to start an interrupt service routine. The service routine can then begin servicing the request from the CD1283 by starting an acknowledge sequence.

### 5.2.6 DMAREQ* as Interrupt Source

Interrupts are not generated by FIFO threshold conditions; therefore, if the system design requires data to move through interrupts, connect DMAREQ* directly to a CPU interrupt input or logically OR it into the same CPU interrupt input as SVCREQP*. If DMAREQ* is used to generate interrupts, the following are required:

1. A 16-bit data interface must be implemented to support 16-bit reads of the DMABUF register.
2. The DMA threshold value in the PFTR must be initialized.
3. The DMAREQ* remains active until the FIFO is nearly empty (Rx) or nearly full (Tx), followed by the toggling of DMAen if data is moved to/from the FIFO through PIO (refer to

Section 5.2.4). However, the software can easily change this by clearing the DMAen bit (PFCR[6]) at the start of the interrupt service routine and setting it again at the end.

4. If SVCREQP* and DMAREQ* are logically OR'ed together, the service routine must start by checking the SVRR to determine which signal is active.
5. SVCACKP* must not be activated in response to DMAREQ*; likewise, DMAACK* must not be activated in response to SVCREQ*.
6. The DMAdir bit (PFCR[5]) can determine whether to write or read to/from the DMABUF register.
7. The PFQR can determine how many reads of the 16-bit DMABUF register are necessary to empty the pipeline. Note however, four must be added to the PFQR value, then that number then must be divided by two and truncated to the nearest integer (this accounts for the extra four bytes in the two holding registers and the 16-bit DMABUF register, as well as 16-bit instead of 8-bit reads).

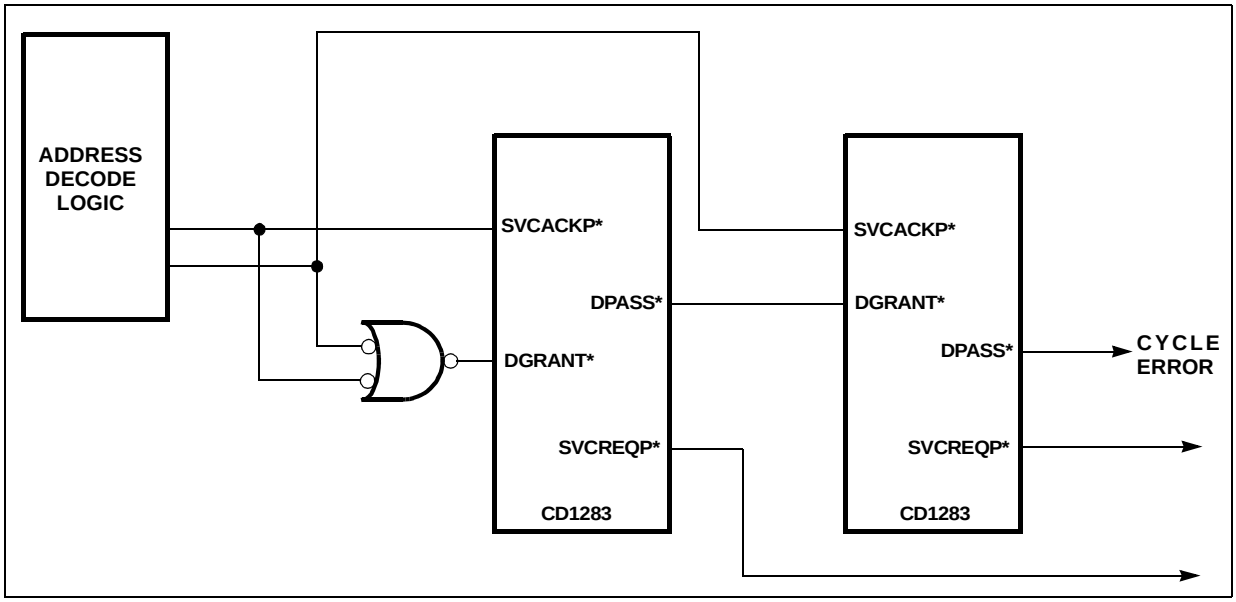
## 5.2.7 Daisy-Chain Configurations

Multiple CD1283s can be connected in a daisy-chain configuration, forming systems with multiple parallel ports. The device provides all signals necessary for this configuration, with only minimal external logic being (Figure 4).

When the CPU acknowledges the request, both CD1283s receive the acknowledge through SVCACK*. However, only the device receives DGRANT*. If it has an active request of this type pending, it takes the acknowledge and drives the vector register (RIVR, TIVR, MIVR) onto the data bus.

If the first device does not have a request pending, it passes DGRANT* to the second CD1283 through DPASS*. Assuming that the second CD1283 has an active request pending, it then takes the acknowledge and drives its vector register onto the data bus.

**Figure 4. CD1283 Daisy-Chain Configuration**



As previously mentioned, the upper 5 bits of the LIVR reflect what the CPU loaded into them during initialization of the CD1283s. These bits are used as a unique chip identification number. Now the CPU can determine which CD1283 responded to the service acknowledge. These five bits could be set to binary '0' in the LIVRs of the first CD1283, and to binary '00001' in those of the second. The CPU is able to test the bits to determine which device responded. Some examples of service-acknowledge software routines that show one way of performing this task are provided in Chapter 6.0.

***Caution:*** If no CD1283 in the chain has a pending request, DGRANT* is passed by the last CD1283 and none respond. This causes the bus cycle to fail (no DTACK* is generated). The only time this happens is when an error condition outside the CD1283s cause the CPU to respond to a request that is not made. Provide a mechanism to terminate or abort the bus cycle if this error occurs. This can be accomplished with timeout circuitry, or the DPASS* output of the last CD1283 can activate an abort condition. Other devices, such as the CD1400, can share the daisy-chain mechanism and be connected to the DPASS* output of the last 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

Service requests can derive from two internal sources: the data pipeline or the parallel port state machine (Figure 5 on page 28). If the data pipeline internal service request becomes active, the Pipeline bit (PIR[5]) is set; likewise, if the parallel port state machine internal service request becomes active, the PPort bit (PIR[6]) is set. Internal service requests from these sources are monitored through the Pipeline and PPort bits by microcode running in the internal MPU. When either (or both) of these bits are detected active, the microcode sets the PPireq bit (PIR[7]). The PPireq bit is also mirrored by the SRP bit (SVRR[3]). The SVRR is useful in polled systems because it allows the detection of DMA service requests, as well as parallel port service requests with a single register read operation.

***Note:*** For specific register definitions and default settings, refer to Chapter 7.0.

Both internal sources of service requests within the parallel channel have their own enable functions. Interrupts from the data pipeline are enabled through the PFCR; interrupts from the parallel port state machine are enabled through the 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 holds data). The first type of interrupt is enabled through the IntEn bit (PFCR[4]). The second type of interrupt is enabled through the ErrEn bit (PFCR[1]). 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 is generated if the DataErr bit (PFSR[0]) is a non-zero. In this case, the DER indicates the cause of the error interrupt.

The parallel channel-control state machine can generate six types of interrupts. Each of these has its own enable bit in the PCIER:

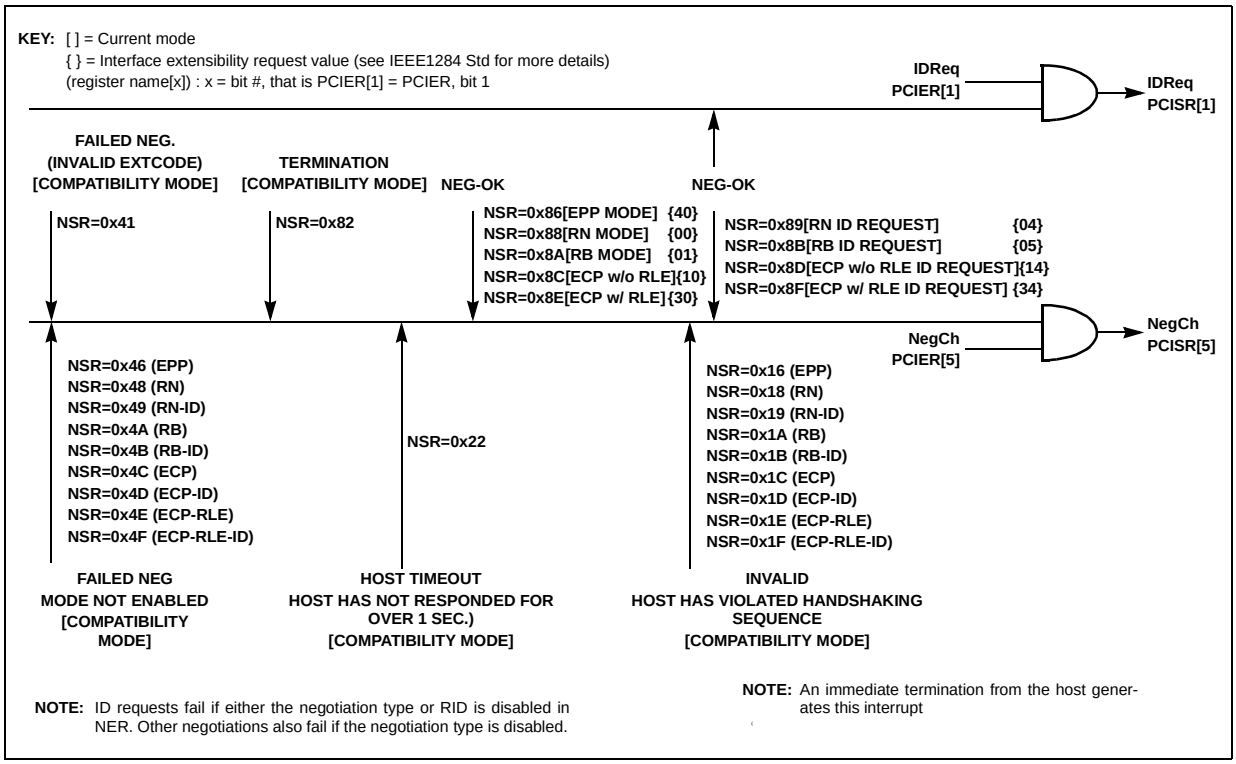
- NegCh for negotiation changes
- SigCh for signal changes on the port status inputs (Manual mode only)
- EPPAW for EPP protocol address writes
- DirCh for direction changes on the parallel channel

- IDReq for slave ID requests from the remote master.
- nINIT for initialization pulses from the master (Compatibility mode only)

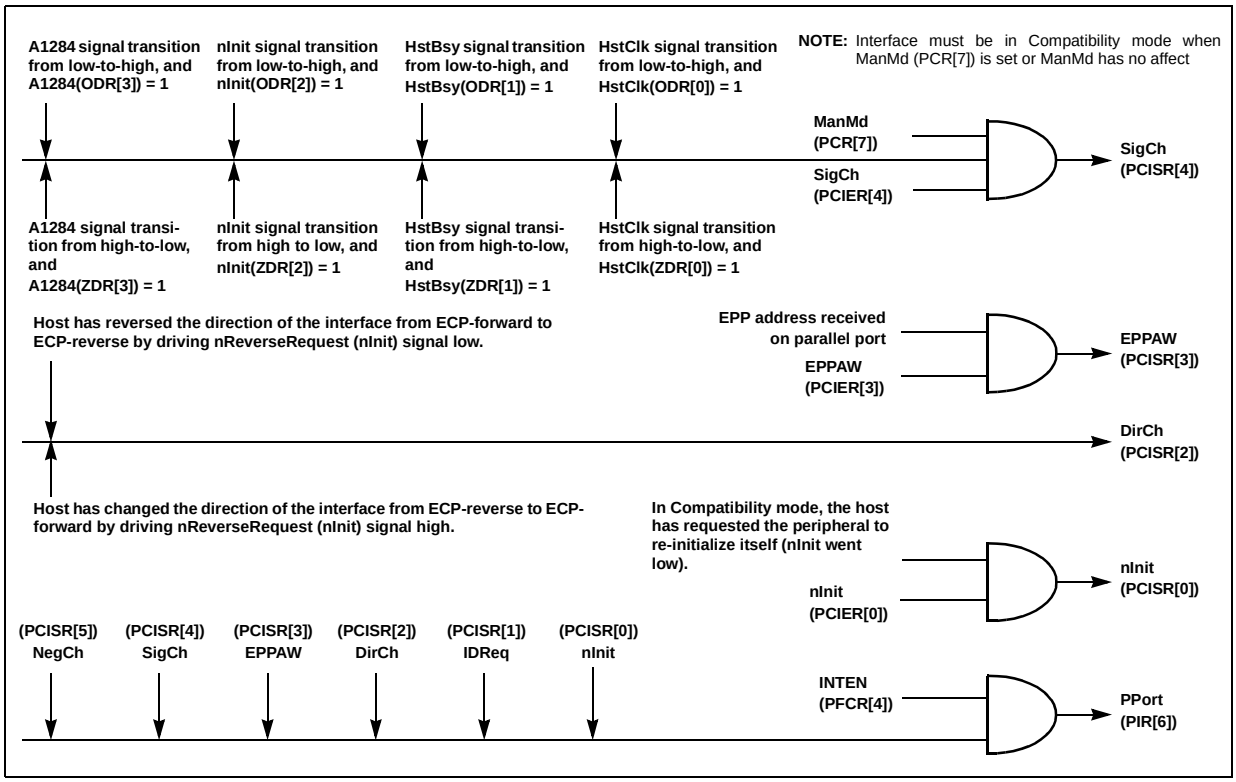
Any or all of these bits may be set, based on the mode of operation.

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

**Figure 5. Interrupt Generation Logic**



## Figure 5. Interrupt Generation Logic (Continued)



## Figure 5. Interrupt Generation Logic (Continued)

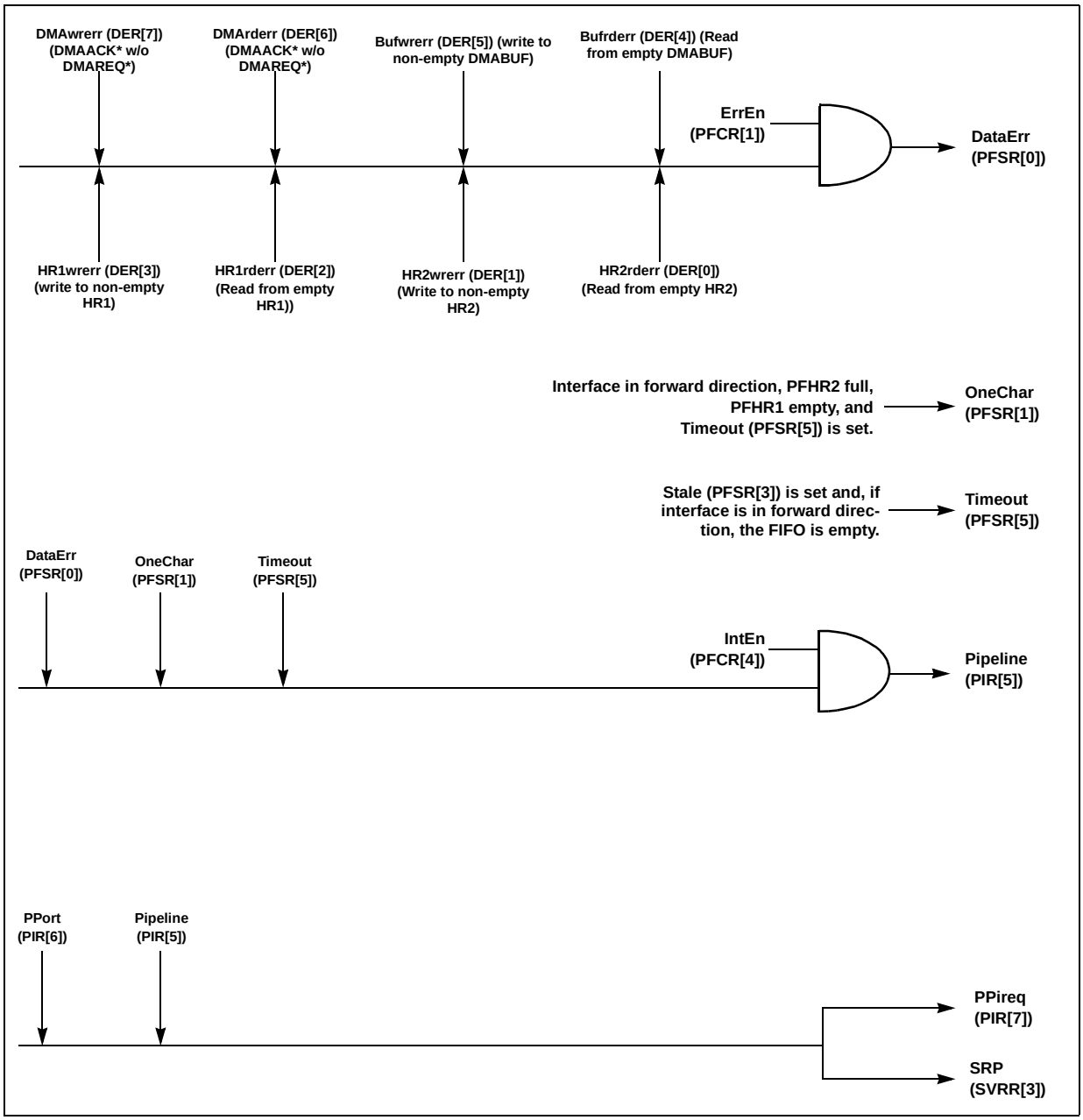
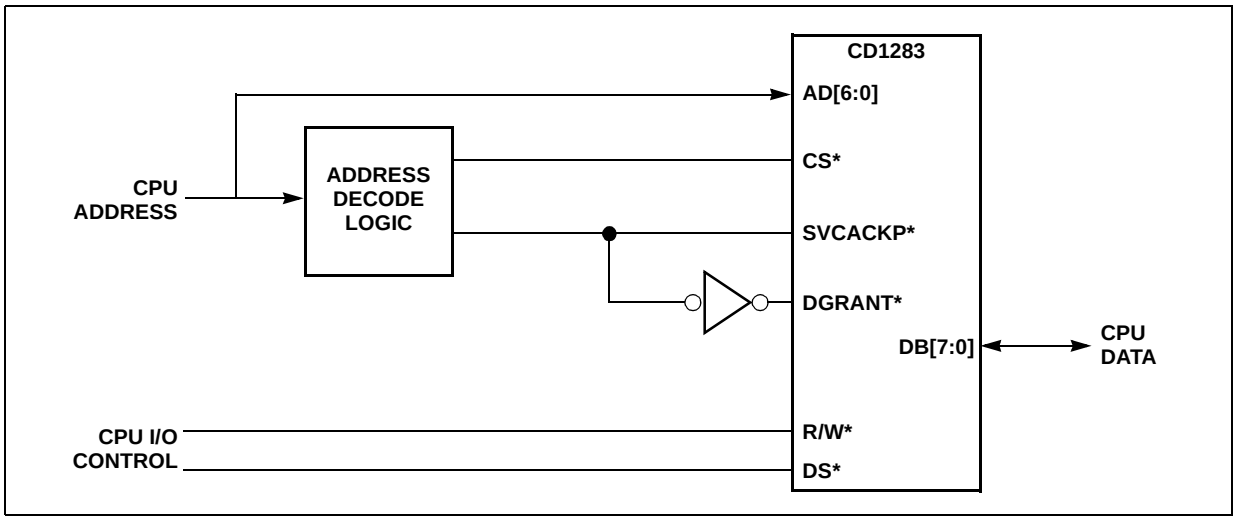


**Figure 6. Control Signal Generation**




A direction change (DirCh) interrupt occurs when the remote master has reversed the interface from ECP forward to ECP reverse or ECP reverse to ECP forward. 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 reversing the direction of the data pipeline by setting the DMAdir bit to ‘1’.

If vectored interrupts are required by the system, then the LIVR must be initialized by the local CPU. The upper five bits are defined by the local CPU and can be any value appropriate to the system design. The lower three bits should be initialized to zero during the programming of the LIVR, however they are ‘don’t cares’ and 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 can read it at any time. When read during a normal read cycle, the upper 5 bits return the original value loaded by the CPU.

The three least-significant bits always ready back as the current service-request status of the parallel port if an interrupt is in progress; otherwise they read back as ‘0’. The encoding of the three least-significant bits of LIVR during a service acknowledge cycle indicates which of the functional blocks in the parallel channel is requesting service as shown in the following table.

**Table 6. LIVR[2:0] Encoding**

| IT2 | IT1 | IT0 | Requestor |
|---|---|---|---|
| 1 | 0 | 0 | Channel control state machine |
| 1 | 0 | 1 | Data pipeline |
| 1 | 1 | 0 | Both |

### 5.3.1 Hardware-Activated Acknowledge

When conditions within the parallel channel require attention, a request is made through the SVCREQP* output. If the system is interrupt driven, this output is 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 CD1283 responds to the SVCACKP* cycle by driving the contents of the PIVR onto the data bus with IT2–IT0 encoded as shown in Table 6. The SVCACKP* cycle also places the device in the correct context to service the parallel channel request.

The vector supplied by the PIVR indicates which block of the parallel channel requested service; the cause of the request is 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.

The I/O cycle that activates the SVCACKP* input also removes the active SVCREQP* output. The request output remains inactive until after the CPU terminates the acknowledge routine by a write 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 clears the PIVR bits to ‘0’ in preparation for the next service-request cycle.

### 5.3.2 Software-Activated Acknowledge

During a normal read cycle, the LIVR always reads back with the lower 3 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 the SRP bit is set (SVRR[3]), at least one of the request conditions is true and a subsequent read of the LIVR indicates the source. A scan of just the SVRR allows the polling routine to perform only one read cycle to determine if a parallel request is pending. If the SVRR indicates an active parallel channel service request, the software can initiate the appropriate service routine that reads PIR to determine the source of the parallel port request. The PPort and Pipeline bits indicate the block requesting service.

Once the CPU satisfies the request needs of the parallel channel, it must toggle the IntEn bit (PFCR[4]) or clear the PIR. Toggling IntEn clears the PPireq, PPort, and Pipeline bits (PIR[7:5]). This action also informs the MPU to clear the PIVR and remove the external request.

The PPireq bit 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. After the interrupt source is determined, the CPU can clear PIR or toggle IntEn, then the PIR is automatically cleared.

## 5.4 Parallel Port FIFO and Data Pipeline

The parallel port within the 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, ECP (Extended Capabilities port) with and without RLE (run-length encoding, and the EPP (Enhanced Parallel port).

The IEEE 1284-compliant parallel port consists of two major functional blocks:

- A data pipeline that moves data between the parallel port and the CPU and includes a FIFO, holding registers, DMA control, and interrupt control logic.
- A channel control state machine to performs all control and handshake generation on the parallel port interface side of the device.

### 5.4.1 IEEE Standard 1284 Protocols

The following sections discuss 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 relate complete information on the specification. A copy of the IEEE Std 1284-1994 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 transfers are the preferred means of transferring data to/from the FIFO. However, it is also possible to transfer data to/from the data pipeline by reading and writing the holding registers directly through PIO. DMA request and acknowledge handshake signals support transfers to/from the 16-bit-wide DMABUF register. The direction of transfer is determined by the DMAdir bit (PFCR[5]).

In the transmit direction, with DMAbufWe set (PFCR[0]), the CPU can write 2 bytes at a time directly to the DMABUF register. However, most applications are not concerned with speed on the parallel port in the reverse direction and do not require 16-bit writes to the FIFO. The CPU must avoid writing to these registers when they are already full or reading from them if they are empty. The status bits in the HRSR indicate if the holding registers and the DMA buffer are full or empty. When writing a block of data to the CD1283 (with DMAbufWe set to ‘1’), the CPU can determine how much data the FIFO can accommodate by reading the PFQR.

Should data become ‘trapped’ in the DMABUF register in the receive direction because of a failure of the external DMA controller or because the external buffer area is full, it can either remain until the DMA transfer can be resumed or the CPU can read the data directly from the DMA buffer.

***Note:*** The DMA buffer can only be read when DMAREQ* is active because data is not moved into the DMABUF register until DMAREQ* is activated by the threshold logic or a timeout condition.

Once a DMA request is initiated by the 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 CD1283 when it determines that the FIFO is nearly full. (If RLEen is set, the pipeline does 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.

The quantity of data transferred within a single DMA request can significantly exceed the capacity of the FIFO if RLEen is set, the parallel port is in ECP mode, and compressed data is being transferred. This is because the FIFO always stores the data in compressed form. Since other modes do not support RLE compression, the CPU should only set RLEen when the parallel port interface is in ECP mode.

### 5.4.3 Parallel Port FIFO

The CD1283 has a dedicated 64-byte FIFO with counters to maintain the fill/empty pointer addresses, logic to manage data transfers, automatic DMA handshake, and status interrupts to the CPU. A simple register interface provides control over setting the direction of the pipeline, initializing/resetting the DMA pointers, setting the DMA threshold, and so on. 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 byte-oriented with 2-byte word packing/unpacking occurring between the DMABUF register and PFHR1 and PFHR2. 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) transfers first. If BYTESWAP is low, the lower byte (bits 7:0) transfers first.

Data transfers to/from the CPU are initiated by a DMA request whenever the quantity of data or space in the FIFO equals or exceeds the threshold value stored in the PFTR. The DMA request is deasserted during the DMA cycle determined by the logic to 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 = 0), the first two bytes of data placed into the FIFO by the parallel port are immediately moved into the data pipeline, PFHR1 and PFHR2 (Figure 7 on page 37). This is done in part to make the tagged status of the data visible to the pipeline control logic. If RLEen is ‘0’, any tagged data from the FIFO must move through the pipeline. However, tagged data cannot be transferred to the CPU by a DMA transfer from the DMABUF register. Therefore, the presence of tagged data in the pipeline causes an interrupt to the CPU. The CPU must then 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 immediately moves forward to fill the pipeline. If the FIFO is empty, the pipeline does not move so, if the CPU emptied PFHR2 and PFHR1 is full, the data in PFHR1 moves forward to PFHR2 only if the FIFO is *not* empty.

The pipeline logic keeps the pipeline full in the receive direction. The value in the threshold register 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 remove the data.

### 5.4.5 Receiving Compressed Data

RLE compressed data sequences that 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 an RLE count (HostAck signal is high) and 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 decrements. When the RLCR has reached zero, the hold on PFHR1 is released and it can shift forward in the pipeline as ordinary data.

Tagged data from the FIFO is recognized to be an ECP mode address and shifts into the pipeline where it causes an interrupt to the CPU to remove the tagged data from the pipeline. If RLEen is '0', all tagged data from the FIFO is shifted into the pipeline and produces CPU interrupts.

If an immediate termination occurs between the reception of the RLE count and the corresponding data, then the RLE count is stored in RLCR and the next data byte received in ECP mode is uncompressed into the FIFO (based on the values in RLCR provided and if RLEen is still set). If the next byte received in ECP mode is a new RLE count, then that value overwrites the old value in the RLCR.

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

Data transfers to the CPU can also be initiated by the stale data timer. This timer is reloaded with the value in the SDTPR and restarts each time data is placed into the FIFO from the parallel port. When the timer reaches zero, the status indication stale bit (PFSR[2]) is set true unless StaleOff (PACR[5]) is true.

StaleOff keeps the stale status bit false, even though the SDTCR counter value is zero. Should the stale status become true with at least two characters of data available, a DMA request is made to transfer the data. If the stale status is true and there is exactly one character available, the OneChar status bit (PFSR[1]) is set and an interrupt generated to the CPU to transfer the single residual character.

The PFSR indicates the Stale, OneChar, and FFempty conditions. The HRSR shows that PFHR2 contains the final character. An odd number of bytes cannot be transferred by DMA. If a DMA transfer completes with one byte of data remaining, the data is held pending arrival of additional data or the expiration of the stale data timer.

The OneChar status is latched true when the FIFO and the 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. If additional data arrives in the FIFO, it remains 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 bit was latched, the FFempty bit will be false. When the CPU reads the single character from PFHR2, any newly arrived data in the FIFO immediately moves forward into the pipeline and a DMA transfer can begin if conditions warrant.

Another latched status condition associated with the stale data timer is the Timeout status bit (PFSR[5]). Timeout is reset by the FIFOres bit (PFCR[7]) and the ClrTO bit (PACR[3]). Timeout, OneChar, and DataErr are pipeline interrupt conditions and, if enabled, generate an interrupt. In the receive direction, the Timeout condition is armed when Stale is '0' and ClrTO and FIFOres are also

‘0’. When Stale becomes ‘1’, the timeout is triggered, but not set until any DMA transfer is complete, the FIFO is empty, and there is no more than one character left in the pipeline. To clear the timeout condition, set the ClrTO bit. To reenable the timeout function, clear ClrTO.

The CPU can arm the timeout by a write of ‘01h’ directly to the SDTCR. If the timer expires before any data arrives, an interrupt is generated for the timeout condition. If data arrives before the timer expires, the interrupt delays until the data becomes stale.

### 5.4.7 Transmit Direction

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

If RLEen is ‘1,’ run-length encoding is enabled and comparators among the pipeline stages recognize repeated strings of characters and compress them (Figure 8 on page 38). 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 DMABUF register and the character in PFHR1 are identical. Compression begins 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 increments to ‘1.’ As long as identical additional characters are loaded into the DMA buffer, the RLCR value continues to increment and the data in PFHR2 is not moved into the FIFO. When the repeated sequence is finally broken or the RLCR count reaches 127, the RLCR value transfers into the FIFO, the RLCR zeroes, and the character in PFHR2 transfers 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 complete, the CPU must either force RLEen to ‘0’ or ensure that both DMAen and DMAbufWe are ‘0’. When either of these conditions is true, the pipeline is released and the data held in PFHR1 and PFHR2 transfers into the FIFO.

The timeout interrupt can be used as a general timer interrupt in the transmit direction. Unlike the receive scenario, when DMAdir is true, the Timeout status bit is immediately set when the timeout is triggered by a ‘0’-to-‘1’ transition of Stale. To use the timeout interrupt, the CPU must load the desired time delay directly into the SDTCR. When the timer expires, Stale becomes true and the timeout interrupt is activated.

## 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 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 (Table 6 on page 31). The CD1283 uses fixed names for each of the pins. The names were selected to represent the most commonly used names amongst the various protocols. The 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.

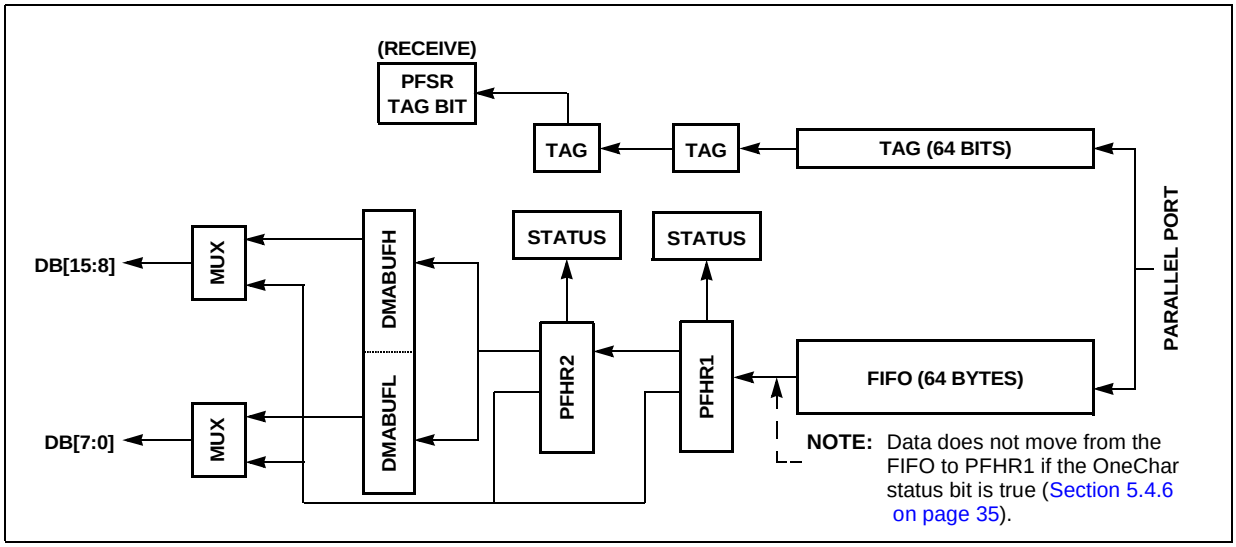
### 5.5.3 State Machine

The parallel port is controlled by a large synchronous state machine. The state machine is based on the IEEE Std 1284-1994 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 bit (PCR[5]) is required to allow transfers in Compatibility mode. PCR[7:5] enable transfers, negotiations, and Manual mode.

**Figure 7. FIFO Data Path Functional Diagram – Receive**




### 5.5.5 Interrupts

Interrupts are enabled in the PCIER and interrupt status can be read in the 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 (PCR[7]) when the interface is in Compatibility mode.

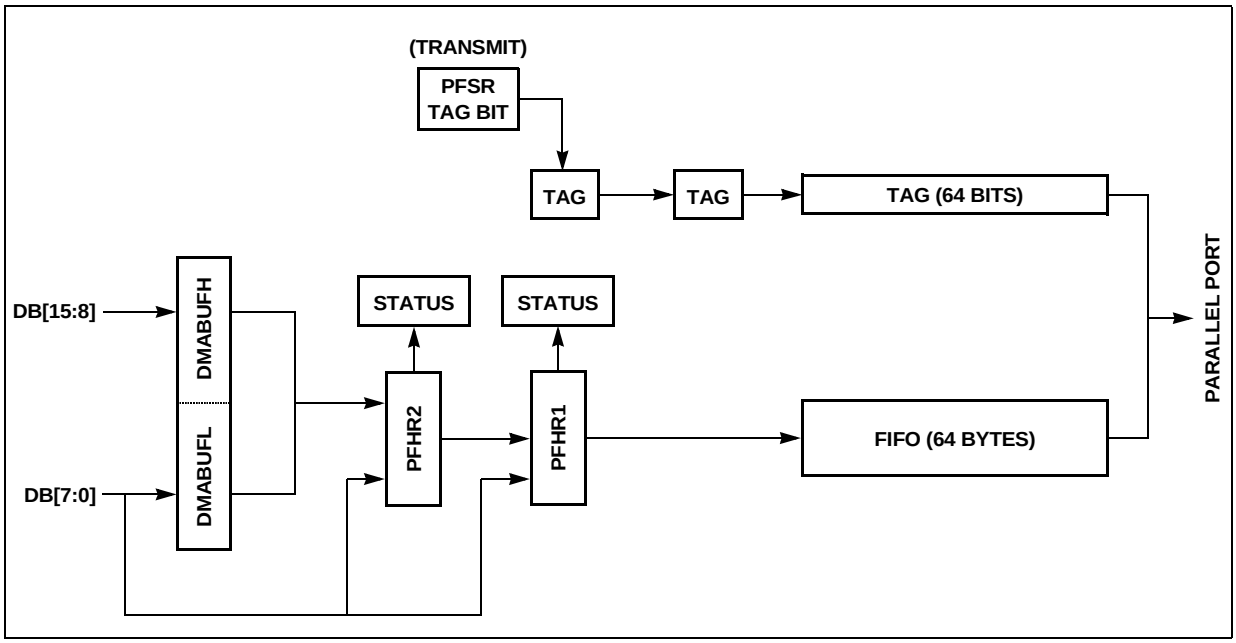
The MMDir bit (PCR[1]) sets the direction of the PD bus: 0 = input; 1 = output. When the MMDir bit is set to ‘1’, data for the PD bus comes from the MDR. The ManOE bit controls the tristate buffer on the PD bus: 0 = floating; 1 = driving. When MMDir is ‘0’, ManOE is ignored, PD[7:0] are inputs, and the data can be read in the MDR.

### 5.5.7 Control Signals

Output signals are controlled by the 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, the others are set by the state machine.

IVR, ZDR, ODR, and SSR monitor the four input signals. These four registers have a common format. The IVR always shows the values of the four input pins. ZDR and ODR allow the user to force interrupts on specific signal transitions. Bits set in the ZDR generate an interrupt, if the specified signal changes from ‘1’ to ‘0’. Similarly, bits set in the ODR generate an interrupt if the specified signal changes from ‘0’ to ‘1’. When both bits are set, interrupts are generated on either transition. The SSR shows the status of signal changes according to ZDR and ODR. SSR shows which signal changed. (It is necessary for the user to read the IVR to determine how the signal changed.) The signal change interrupt is enabled with the SigCh bit (PCIER[4]).

**Figure 8. FIFO Data Path Functional Diagram: Transmit**



### 5.5.8 Parallel Port Interface to the FIFO

The DMAdir bit indicates the current direction (0 = in; 1 = out) of transfers between the FIFO and the DMA logic. Due to a recent negotiation, this can differ from the current parallel-port interface direction. The CPU must change the direction after it receives an interrupt showing a direction change.

The FIFOlock bit (PACR[4]) stops the DMA pipeline. This can be useful in diagnostics. FIFOlock is also used in ECP and EPP modes to stop data transfers in the forward direction.

### 5.5.9 IEEE 1284-Protocol Negotiations

All IEEE 1284 protocol negotiations are initiated by the master side. The role of the CD1283 is to accept or reject the attempted negotiation. The NER contains bits to 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 is complete, the interface remains in Compatibility mode. These negotiations occur in two stages; both stages occur automatically after the device is commanded to begin the negotiation procedure to a particular mode. The first stage determines if the slave is IEEE 1284-compatible. Once determined, the interface continues the process to determine if the mode requested is supported. The result of the requested negotiation appears in the NSR.

For negotiations to occur, the slave must enable the E1284 bit (PCR[6]). Data transfers require that the ETxfr bit (PCR[5]) be set; negotiations can occur without data transfer enabled.

#### Negotiation Status Register

After any IEEE-1284 negotiation or termination, the current protocol status can be read in the NSR. NegOK and NegFl (bits 7:6) indicate successful and failed attempts. Invalid (bit 4) indicates 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. The 4-bit code in NSR also indicates the mode that the interface was in when an invalid termination was detected, as well as a failed negotiation. Interrupts indicating a successful negotiation into a reverse mode should prompt the CPU to load reverse data into the FIFO.

#### Special Command Register

The bits in the SCR cause actions on the parallel port. SetPs and ClrPs (bits 3:2) control data movement into the CD1283 from the remote master. In Compatibility mode, this function posts error status to the remote. Errors can only be presented to the master by the slave during the active BUSY period. SetPs causes the CD1283 to stop transfers by asynchronously asserting the BUSY signal. To protect against the possibility of data loss, one more byte can be strobed into the CD1283 after BUSY goes active due to the setting of SetPs. When the error status is delivered, ClrPs restores the parallel interface to the normal running state.

EPIrq sends an interrupt pulse in EPP mode. The RevRq bit indicates that data is available for reverse transfer in either Compatibility or ECP mode. These operations are further described in the relevant protocol sections.

### 5.5.10 Data Transfers

In Compatibility mode, incoming HstClk (STROBE*) pulses activate PerBsy (BUSY), and the data on PD[7:0] is held in latches. PerBsy protects the data latches by signalling the master it is not ready for more transfers. After the HstClk pulse ends, a pulse is sent on PerClk (ACK*) to acknowledge the receipt of the data into the holding latches. After the data moves from the latches to the FIFO, PerBsy goes low 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, FAULT*, and PError) must not be asserted unless PerBsy (BUSY) is high. PerBsy can only be activated in response to a received character, and must remain high until the status condition (for example, paper out) changes.

To send these status signals to the master device, set the SetPs bit (SCR[2]) and the appropriate bit in the OVR for each of the status signals. The SetPs bit activates PerBsy, which remains active until ClrPs (SCR[3]) is set. No data is lost in this operation.

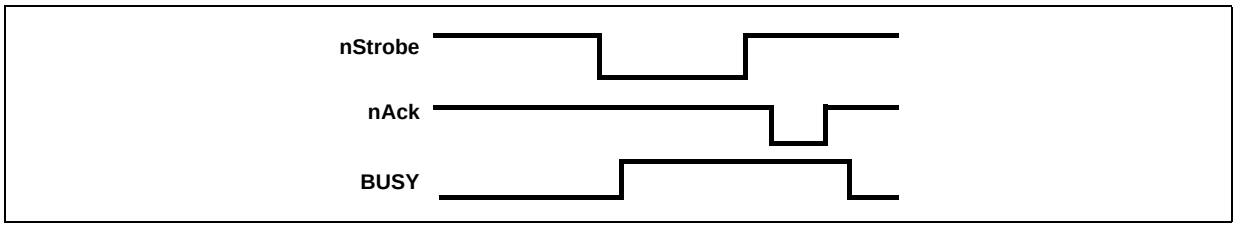
## 5.6 IEEE 1284 Parallel Protocol Support

### 5.6.1 Compatibility Mode

Compatibility mode provides backward compatibility with Centronics and PC-compatible printer interfaces. When the host parallel port is in Compatibility mode (with no data transfer in progress), the host can initiate data transfers in Compatibility mode or initiate negotiations to a new operating mode.

Only Busy-while-Strobe and Ack-in-Busy timing is supported in Compatibility mode. Busy-after-Strobe, Ack-after-Busy, and Ack-while-Busy timings are not supported.

**Figure 9. Supported Compatibility Mode Timing**




### 5.6.2 Reverse-Nibble and Reverse-Byte Modes

These modes support reverse transfers only, from slave to master. Reverse-Nibble mode is enabled with NER[0]; Reverse-Byte mode is enabled with NER[1]. Reverse-Nibble mode sends 4 bits at a time over four of the peripheral status lines. With software drivers the advantage of this scheme is that any unidirectional PC parallel port can be used for bidirectional data transfers. Reverse-Byte mode requires bidirectional buffers on the PC hardware, but allows substantially faster transfers because it moves one byte at a time.

There is no mechanism in Compatibility mode for the slave to indicate that data is available for reverse transfers. The master must poll the slave by negotiating into a reverse mode and examining the nDatAv signal. The RevRq (SCR[0]) instructs the CD1283 to post the availability of data to the master through the nDatAv signal.

### 5.6.3 ID Request

ID request is enabled with a combination of NER[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 CD1283 can accept an ID request in any mode enabled to manage transfers. IDReq is set when an ID request is received in any enabled mode.

### 5.6.4 ECP Mode

ECP mode allows bidirectional transfers and supports the RLE-compression scheme. The ability to expand RLE data is required of all IEEE-1284, ECP-compliant devices, but the ability to compress data is optional. The CD1283 handles both expansion and compression in the data path section. The parallel port simply passes the inverse of the command signal to/from the FIFO on the ninth tag bit in the FIFO. ECP mode is enabled by NER[2]. RLE mode enabling requires both NER bits 2 and 3.

The handshake is identical for both ECP and RLE modes. The control signals, HstBsy and PerBsy (in the forward and reverse directions, respectively), indicate command and address options. If HstBsy/PerBsy is low, the upper bit of the byte is examined: ‘0’ indicates to interpret the lower 7 bits as an address; ‘1’ indicates to use the lower 7 bits as an RLE repeat count. This count shows the number of times to consecutively repeat that the next data character in the datastream.

The master device is responsible for determining the direction of the 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 the RevRq bit (SCR[0]) to request reverse transfers. Once the master changes direction, RevRq is automatically cleared and the DirCh interrupt status appears in PCISR (if enabled in the PCIER).

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

### 5.6.5 EPP Mode

Data transfers use the DMA pipeline and the FIFO. Address transfers are handled out-of-band, not in the FIFO stream. When the slave receives an address write command, it deposits the address into the EAR and asserts an EPPAW interrupt request. When the slave receives a read address command, the contents of the EAR are 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 SPR must be loaded with the number of system clock ticks equivalent to 500 ns, as shown in Table 7.

**Table 7. System Clock Setup**

| CLK Freq. (MHz) | Time/Tic (ns) | SPR Value | $T_P$ Width |
|---|---|---|---|
| 16 | 62.5 | 8 | 500 |
| 20 | 50 | 10 | 500 |
| 25 | 40 | 13 | 520 |

## 5.8 General-Purpose I/O Port

The CD1283 provides an 8-bit general-purpose port (GP[7:0]) used to control or give status of external functions. Each of the eight signals are 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 through two internal registers — GPDIR and GPIO.

Each bit in the GPDIR sets the direction of the corresponding bit in the GPIO; ‘1’ sets the signal as output, and ‘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 reflect the true state of the condition of the external pin; bits programmed as outputs reflect the state of the last value written to the register and the current state of the output pins.

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

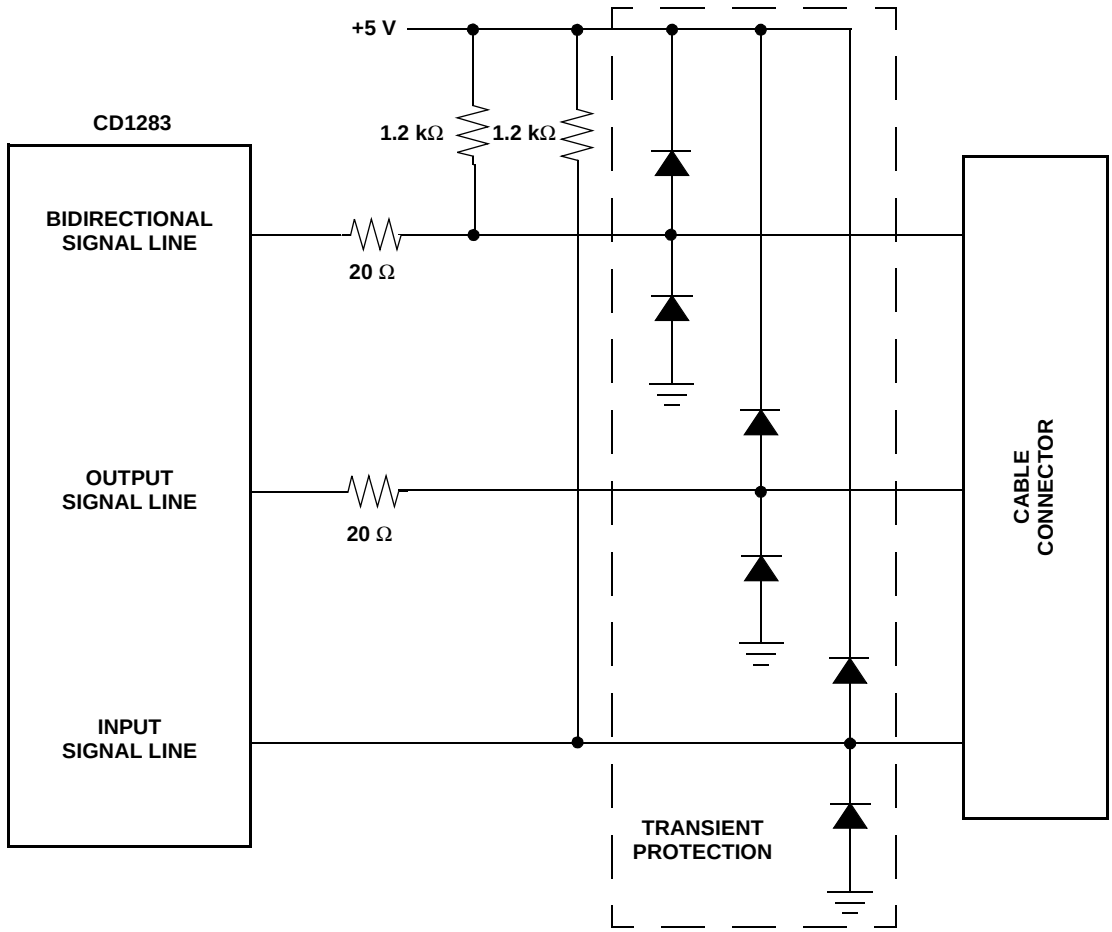
***Note:*** Interrupts are not generated on signal changes within the General-Purpose I/O port; the CPU must periodically poll GPIO to detect changes in external conditions. Therefore, if it is necessary to detect changes, use the port with signals that change with low-duty cycles.

## 5.9 Parallel Port Interface

The CD1283 parallel port signals are implemented with Level-2 characteristics, as defined in the IEEE Std 1284-1994 specification with the exception of transient protection. As such, the port can be directly connected 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 10 illustrates a typical interface with the components listed above.

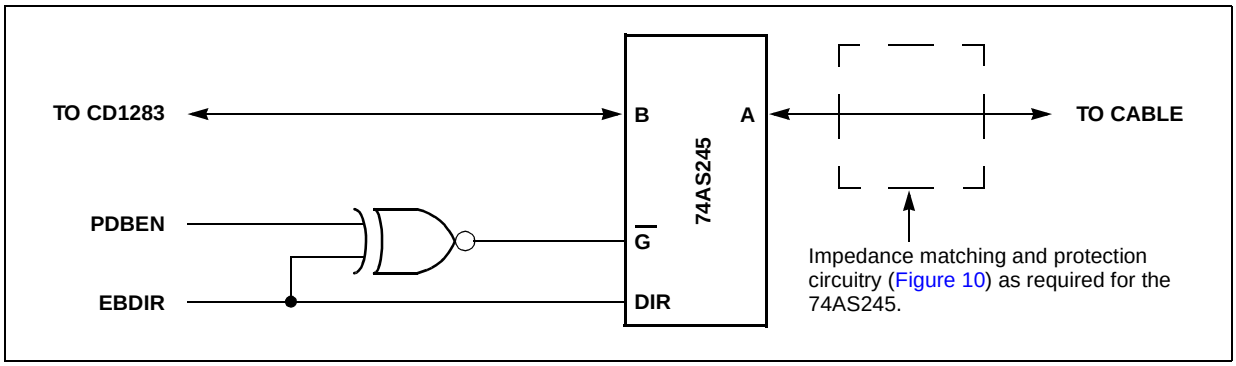
Some system designs may require buffers between the CD1283 and the cable. Systems that require drive cables longer than the specified maximum of 10 m or those that need to protect the CD1283 require inexpensive buffers between it and the cable. The device provides two signal outputs, PDBEN and EBDIR, for connecting and controlling buffers (such as, 74AS245 or equivalent). These signals do not allow direct control of the buffer. However, the addition of an XNOR gate provides 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 11 on page 44).

**Figure 10. Cable Connection**




***Caution:*** Transient protection is not implemented inside the CD1283 device, therefore transient voltages may cause damage. Laboratory testing has shown that this type of protection is not necessary under normal conditions. However, damage may occur under harsh conditions or when subjected to unusual abuse. Also note, the protection circuit shown here may cause a powered-up host to supply power to the +5 V ($V_{cc}$) of the peripheral if it is not powered up. If this is a concern, then another protection circuit must be designed.

Figure 11. External Buffer Control

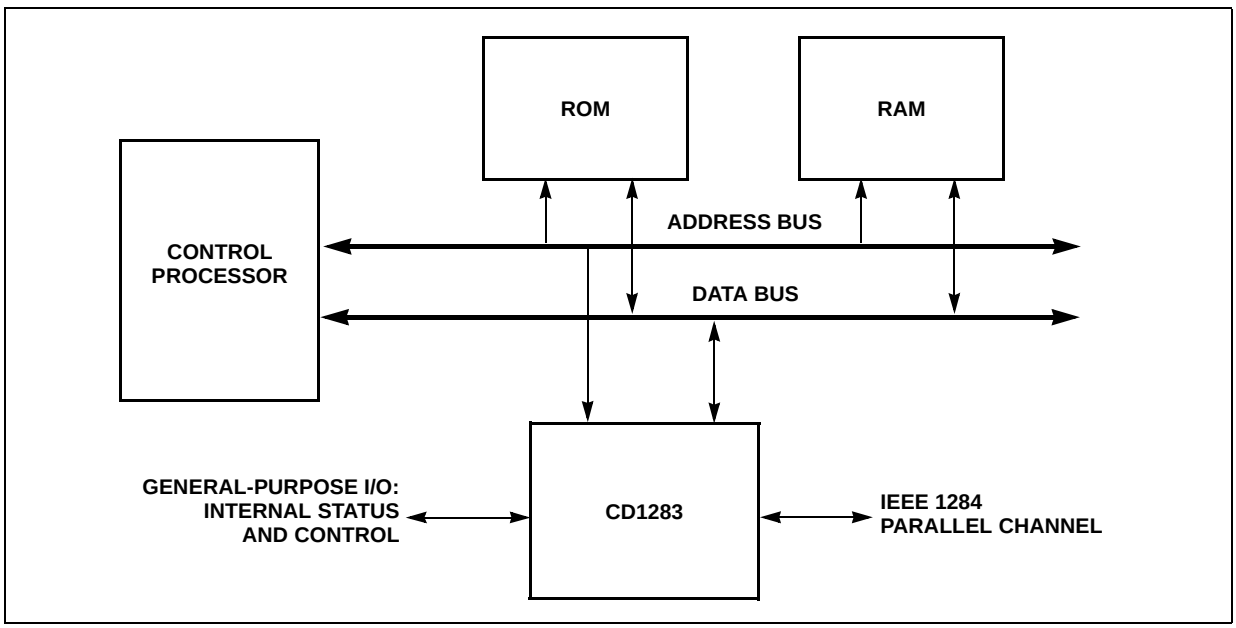



## 5.10 Hardware Configurations

The simplicity of the CPU interface to the CD1283 allows the device to be designed into systems that employ popular microprocessors such as the Intel 80x86 family (8086, 80286, 80386, and so on) and the Motorola® family (68000, 68010, 68020, and so on).

An example of CD1283 configuration for a laser-printer application is shown in Figure 12. This example provides a parallel interface, as well as general-purpose I/O for static control/status.

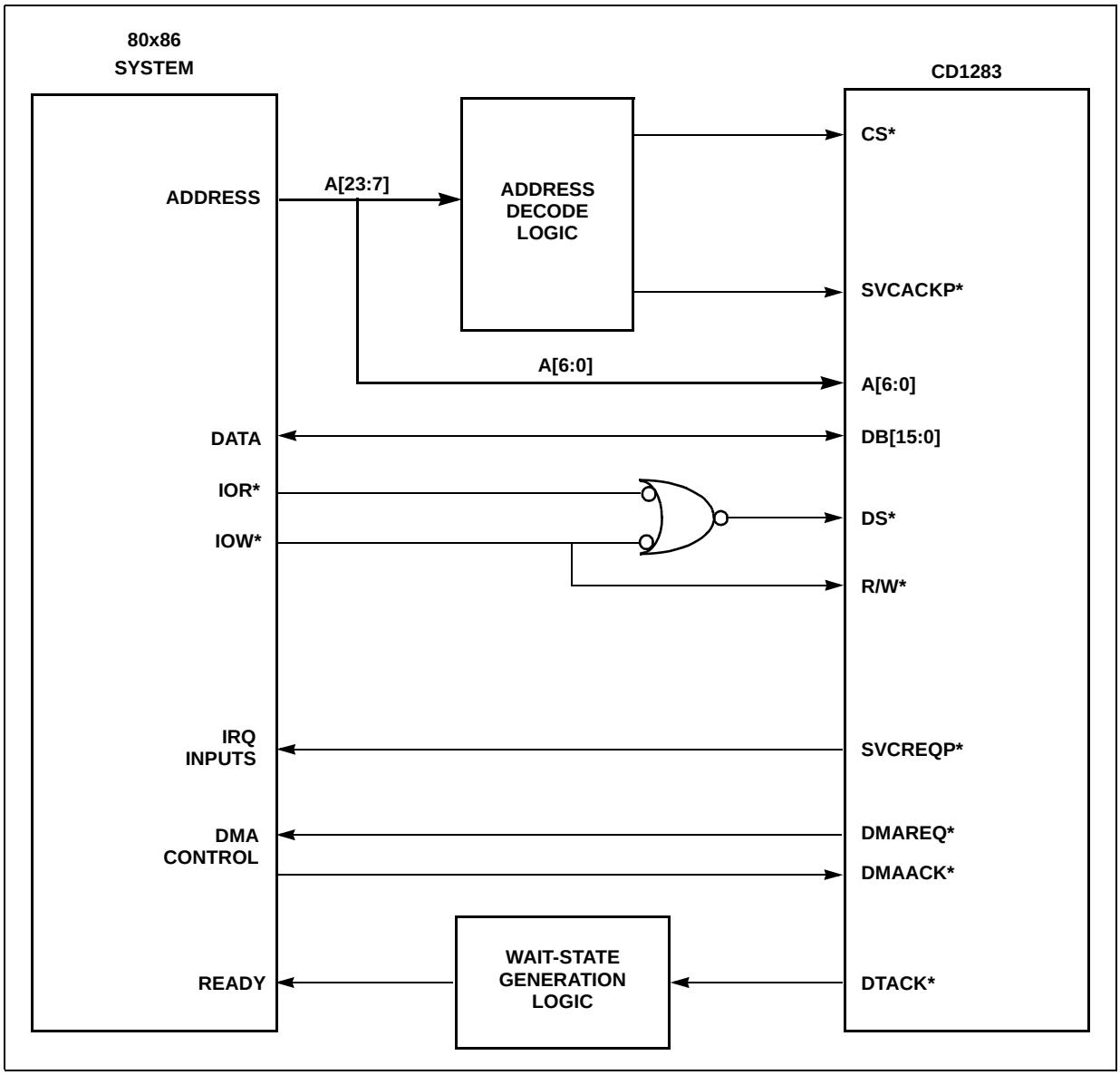
Figure 12. Sample System Block Diagram



### 5.10.1 Interfacing to an Intel® Microprocessor-Based System

With very little additional logic, the CD1283 can interface to any system based on a processor in the Intel 80x86 family. Figure 13 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 synthesize the DS* and R/W* signals. DTACK* is an input to wait-state-generation logic that holds the processor (if necessary) until the CD1283 completes the I/O request.

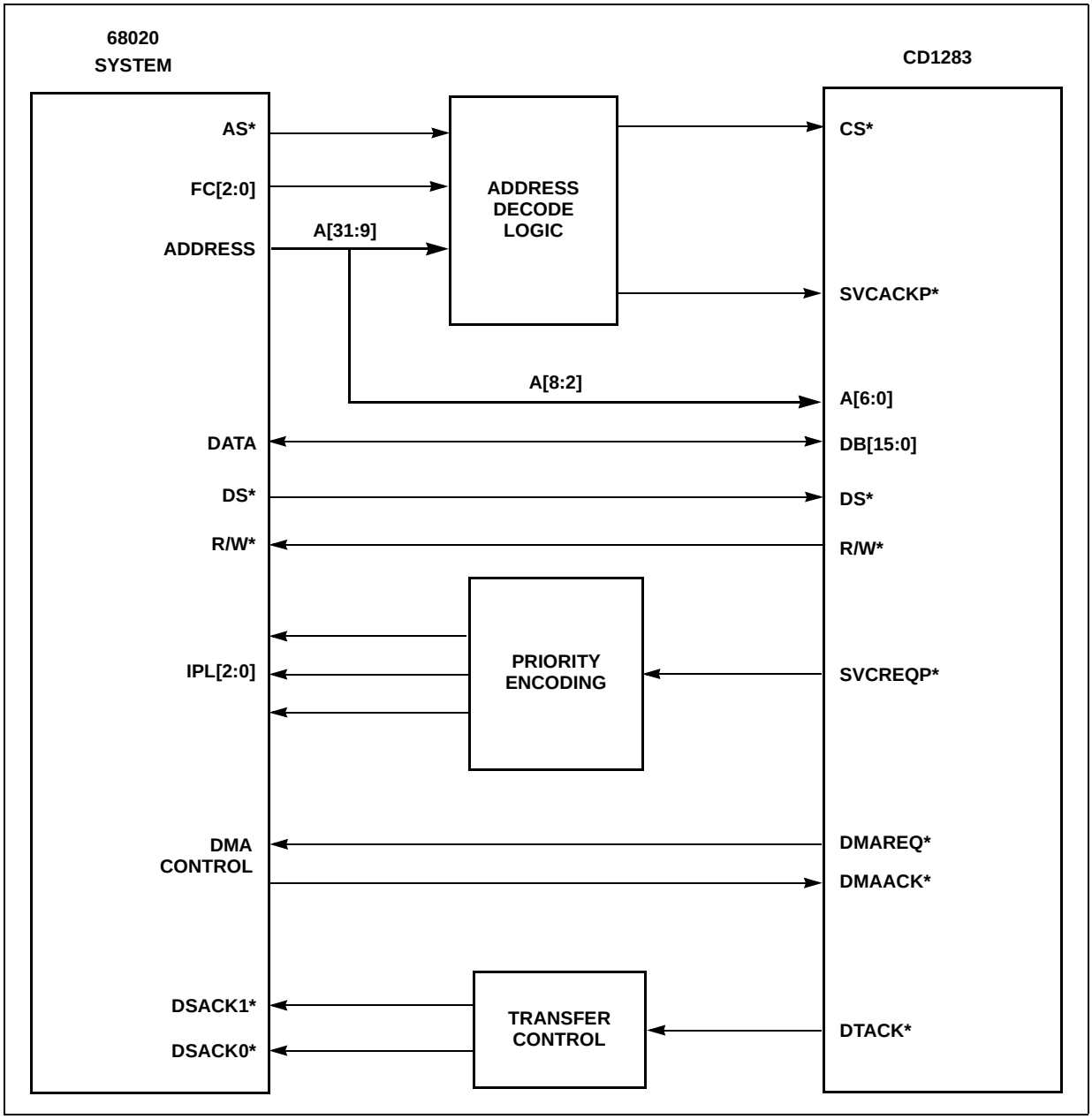
**Figure 13. Intel® 80x86 Family Interface**



## 5.10.2 Interfacing to a Motorola® Microprocessor-Based System

Interfacing to a Motorola 68000 family device is relatively simple. Bus timing and interface signal definitions closely match those of the 68000 microprocessor, which allows a direct connection. With later versions (68020, 68030), some additional logic is required to generate the DSACK0* and DSACK1* functions that replace the DTACK* on earlier devices. Figure 14 shows a generalized interface to a 68020 device.

**Figure 14. Motorola® 68020 Interface**



# 6.0 Programming

## 6.1 Overview

As shown in the register summary tables in Chapter 4.0, the CD1283 local CPU interface consists of a large array of registers. These registers control all aspects of device behavior. Most registers are only modified once, during initialization, and rarely modified during normal operation. This chapter discusses these aspects, as well as the methods of interacting with the CD1283 for parallel-channel service requirements.

## 6.2 Initialization

To properly power-up a CD1283, several procedures must be completed. These include device initialization, programming global functions, and setting port parameters. In most cases, initialization routines are only executed once – during overall system boot-up. Section 6.2.1 details these steps (Figure 15 on page 48 for a flow-chart step outline).

### 6.2.1 Device Reset

The procedures that perform chip reset are normally executed after a power-up, system-wide reset. The hardware reset control signal, RESET* causes the CD1284 to perform its own internal initialization. If desired, the driver software can issue a full chip reset before chip initialization begins. To accomplish this, perform the following steps.

1. Wait for RCR to contain ‘0x00’.

   The contents of the RCR must be ‘0’ before the reset command is issued. This is required to ensure that the device is ready to accept the new command. Since this is probably the first command written to the CD1283 after power-on initialization, the RCR is likely to be ‘0’, but it is recommended to always check the RCR before writing a new command.

2. Set the AER ‘0x02’.

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

3. Write hexadecimal 81 (x’81) to the RCR.

   This command causes the CD1283 to perform a global reset. It causes the internal RISC processor to begin execution from its power-up reset location. The results are the same as if the RESET* input is activated. All internal interface registers are cleared, the FIFO is flushed, and all channel operations are disabled.

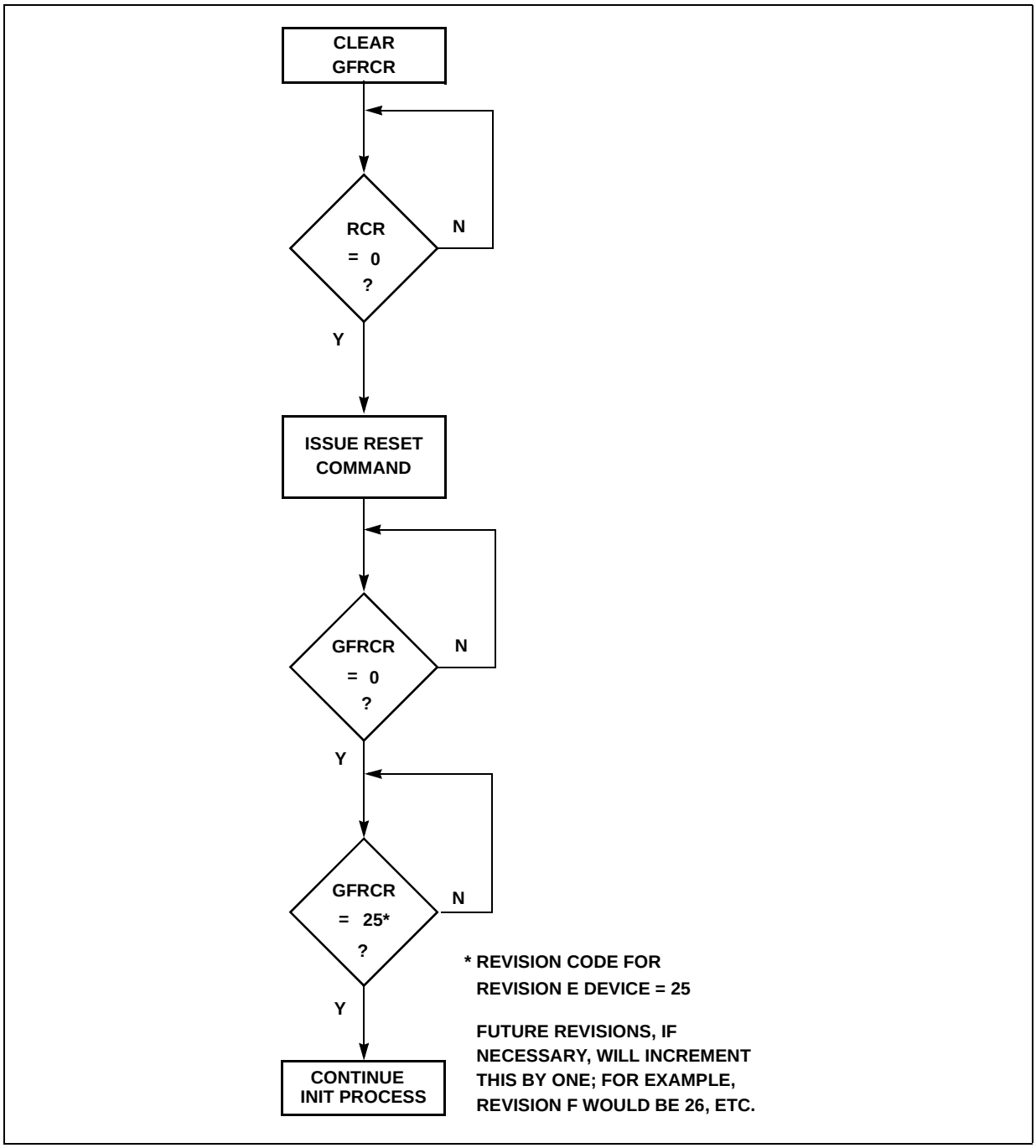
4. Wait for the firmware revision code to be written into the GFRCR.

   Internal firmware uses this operation to flag completion of the reset procedure. After the reset is issued, the GFRCR is one of the first registers cleared and it is the last one set before normal runtime code execution begins. The initialization routine *must* wait for this register to become non-zero before it begins any other programming of CD1284 registers. If the CPU is sufficiently fast, it could begin testing the GFRCR before the MPU clears it. The assumption could be made that the CD1284 has completed internal initialization when, in fact, it has not even started. To avoid this error, the CPU should look for the GFRCR to change to ‘0’. It should then look 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 can check the register after it is cleared or before it is loaded with the revision code.

This procedure is also used as part of a diagnostic test suite. The device completes internal initialization within 500 µsec. A timer (software or hardware) detects when the operation is not completed within that time and cues if the device is functional.

**Figure 15. Flow Diagram of the CD1283 Master Initialization Sequence**



The following section of programming code shows a typical initialization sequence preparing the parallel channel for Compatibility mode data reception and enabling negotiation into all modes, except EPP. This procedure can also be used as part of a diagnostic test suite. The device will complete internal initialization within 500 µsec. Therefore, a timer (software or hardware) can be used to detect that the operation does not complete within this time and that the device may not be functional.

```
/* 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);        /* AER must equal 02h or o3h to access RCR*/
       while (inportb(RCR) != 0x00)
              ;                    /* Wait for RCR to clear */
       outportb(RCR, 0x81);
       while (inportb(GFRCR) != 0x00);/* Wait for GFRCR to be cleaared */
       while (inportb(GFRCR) != 0x25);/* 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);       /* Initialize the interrupt vector 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(PFCR, 0x58);       /* Remove Reset */
       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 */
}
```

### 6.2.2 Service Acknowledge Handling

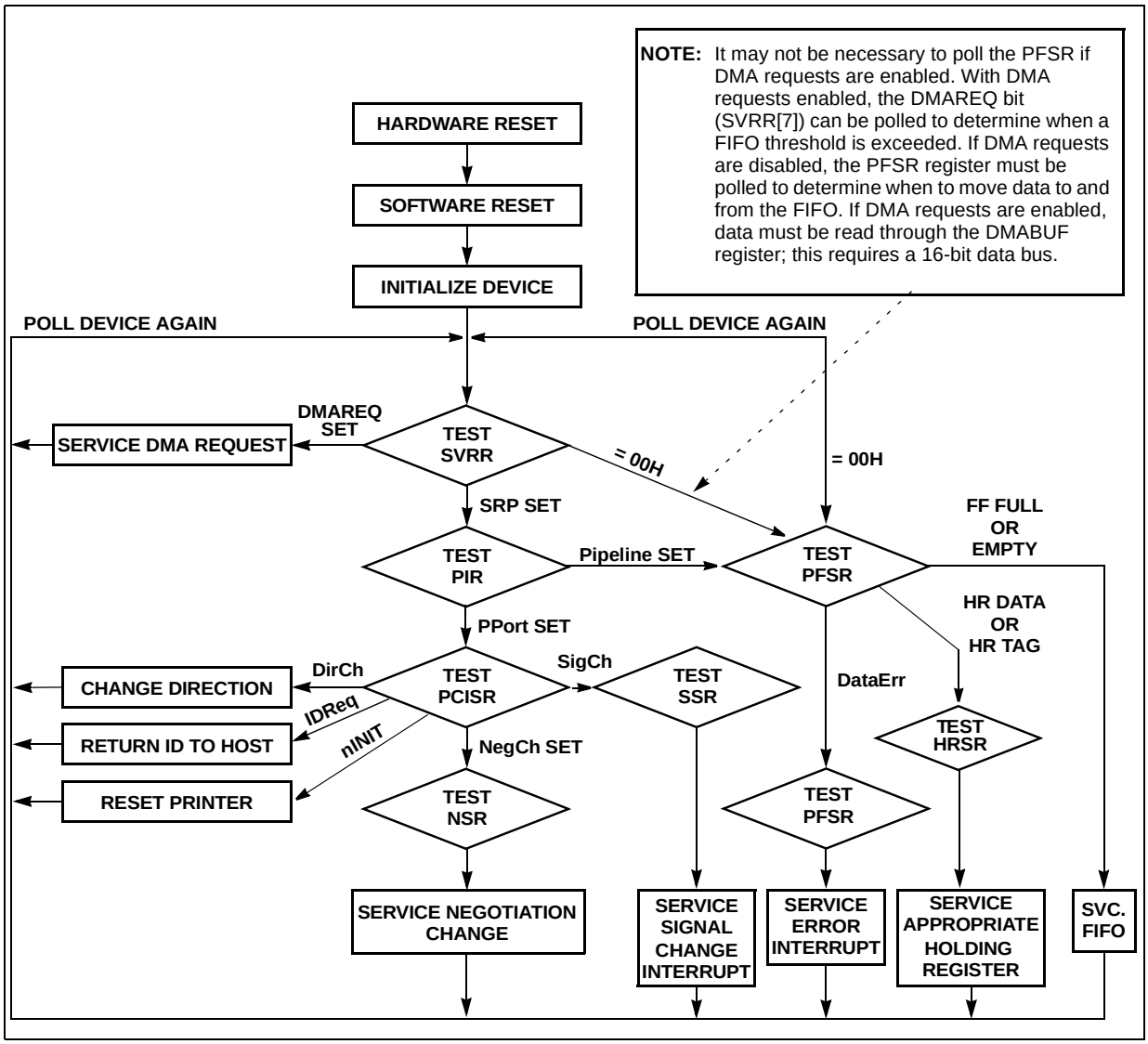
Service request and acknowledge processing, as well as DMA request and acknowledge processing, is performed by the internal MPU. It is important to take the behavior of the MPU into account if interrupts are used. There are two different variations where service requests can be serviced. One variation uses the SVCACKP*, the other does not. If the SVCACKP* signal is activated through an input instruction then the device will return the value of the LIVR on the data bus. This can be used as a vector to the service routine or used in a switch instruction to jump to the correct routine.

When the SVCACK* is activated, the SVCREQP* is deactivated. If the SVCACKP* signal is not activated, then the service request must be removed by clearing PpIreq (PIR[7]), and the source of the interrupt must be determined by reading the LIVR, PIVR, or PIR. Regardless of the variation performed, IntEn (PFCR[4]) must be toggled at the end of the service routine to inform the device that the service routine has terminated.

```
service_par( )
{
        char        livr_val;

        if (inportb(SVRR & 0x08)) { /* check for active service request */
              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);/* terminate service ack. sequence
by */
              outportb(PFCR, inportb(PFCR | 0x10);/* toggling IntEn bit in PFCR */
              return(0);
        }
}
```

**Figure 16. Polling Flow Chart**




## 6.3 ASCII Code Tables

**Table 8. Hexadecimal — Character** (Sheet 1 of 2)

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 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 | ‘ |

**Table 8. Hexadecimal — Character** (Sheet 2 of 2)

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 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 |

**Table 9. Decimal — Character**

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | NUL | 1 | SOH | 2 | STX | 3 | ETX | 4 | EOT | 5 | ENQ | 6 | ACK | 7 | BEL |
| 8 | BS | 9 | HT | 10 | NL | 11 | VT | 12 | 13 | 13 | CR | 14 | SO | 15 | SI |
| 16 | DLE | 17 | DC1 | 18 | DC2 | 19 | DC3 | 20 | DC4 | 21 | NAK | 22 | SYN | 23 | ETB |
| 24 | CAN | 25 | EM | 26 | SUB | 27 | ESC | 28 | FS | 29 | GS | 30 | RS | 31 | US |
| 32 | SP | 33 | ! | 34 | “ | 35 | # | 36 | $ | 37 | % | 38 | & | 39 | ‘ |
| 40 | ( | 41 | ) | 42 | * | 43 | + | 44 | , | 45 | - | 46 | . | 47 | / |
| 48 | 0 | 49 | 1 | 50 | 2 | 51 | 3 | 52 | 4 | 53 | 5 | 54 | 6 | 55 | 7 |
| 56 | 8 | 57 | 9 | 58 | : | 59 | ; | 60 | < | 61 | = | 62 | > | 63 | ? |
| 64 | @ | 65 | A | 66 | B | 67 | C | 68 | D | 69 | E | 70 | F | 71 | G |
| 72 | H | 73 | I | 74 | J | 75 | K | 76 | L | 77 | M | 78 | N | 79 | O |
| 80 | P | 81 | Q | 82 | R | 83 | S | 84 | T | 85 | U | 86 | V | 87 | W |
| 88 | X | 89 | Y | 90 | Z | 91 | [ | 92 | \ | 93 | ] | 94 | ^ | 95 | _ |
| 96 | ~ | 97 | a | 98 | b | 99 | c | 100 | d | 101 | e | 102 | f | 103 | g |
| 104 | h | 105 | i | 106 | j | 107 | k | 108 | l | 109 | m | 110 | n | 111 | o |
| 112 | p | 113 | q | 114 | r | 115 | s | 116 | t | 117 | u | 118 | v | 119 | w |
| 120 | x | 121 | y | 122 | z | 123 | { | 124 | \| | 125 | } | 126 | _ | 127 | DEL |

# 7.0 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 outlined in Chapter 4.0. Bits defined as '0' should not be modified and, if values other than '0' are read, program execution should not be affected or software compatibility with future revisions will be uncertain.

## 7.1 Global Registers

### 7.1.1 Access Enable Register

| Register Name: AER | | | | | | 8-Bit Hex Address: 68 | |
|---|---|---|---|---|---|---|---|
| Register Description: Access Enable | | | | | | Default Value: XX | |
| Access: R/W | | | | | | | |
| *Bit 7* | *Bit 6* | *Bit 5* | *Bit 4* | *Bit 3* | *Bit 2* | *Bit 1* | *Bit 0* |
| X | X | X | X | X | 0 | 0 | 0 |

The AER provides binary compatibility with the CD1284. Users must program this register with the least-significant bits set to '0' to access the parallel channel; however, to perform a device reset through the RCR, AER must = 02h. The contents of the upper 5 bits should be ignored when read.

### 7.1.2 Global Firmware Revision Code Register

| Register Name: GFRCR | | | | | | 8-Bit Hex Address: 4F | |
|---|---|---|---|---|---|---|---|
| Register Description: Global Firmware Revision Code | | | | | | Default Value: 25 | |
| Access: R/W | | | | | | | |
| *Bit 7* | *Bit 6* | *Bit 5* | *Bit 4* | *Bit 3* | *Bit 2* | *Bit 1* | *Bit 0* |
| Firmware Revision Code | | | | | | | |

The GFRCR serves two purposes in the CD1283. First, it displays the revision number of the firmware in the device. When a revision to the CD1283 is required, the revision number of the firmware is increments by one. The revision code is 24 (hex) for the Revision D device, and 25 (hex) for the Revision E 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 (through the RESET* input) or a software global reset (through the reset command in the CCR). Immediately after the reset operation begins, the internal CPU clears the register. When complete, and the CD1283 is ready to accept host accesses, the register is loaded with the revision code.

### 7.1.3 General-Purpose I/O Direction Register

| Register Name: GPDIR<br>Register Description: General-Purpose I/O Direction<br>Access: R/W | | | | | | 8-Bit Hex Address: 71<br>Default Value: 00 | |
|---|---|---|---|---|---|---|---|
| *Bit 7* | *Bit 6* | *Bit 5* | *Bit 4* | *Bit 3* | *Bit 2* | *Bit 1* | *Bit 0* |
| Dir7 | Dir6 | Dir5 | Dir4 | Dir3 | Dir2 | Dir1 | Dir0 |

### 7.1.4 General-Purpose I/O Register

| Register Name: GPIO<br>Register Description: General-Purpose I/O<br>Access: R/W | | | | | | 8-Bit Hex Address: 70<br>Default Value: 00 | |
|---|---|---|---|---|---|---|---|
| *Bit 7* | *Bit 6* | *Bit 5* | *Bit 4* | *Bit 3* | *Bit 2* | *Bit 1* | *Bit 0* |
| Data7 | Data6 | Data5 | Data4 | Data3 | Data2 | Data1 | Data0 |

The GPDIR and GPIO registers enable 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 Data0 accessing GP[0], and so on. The corresponding bit in the GPDIR controls the direction of the associated signal; a logic ‘1’ programs the signal as output, and a logic ‘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 reflect the true state of the signal condition on those bits; bits programmed as output will reflect the previously set state.

### 7.1.5 Parallel Interrupt Register

| Register Name: PIR<br>Register Description: Parallel Interrupt<br>Access: R/W | | | | | | 8-Bit Hex Address: 61<br>Default Value: 00 | |
|---|---|---|---|---|---|---|---|
| *Bit 7* | *Bit 6* | *Bit 5* | *Bit 4* | *Bit 3* | *Bit 2* | *Bit 1* | *Bit 0* |
| PPireq | PPort | Pipeline | 0 | 0 | 0 | 0 | 0 |

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

| Bit | Description |
|---|---|
| 7 | **PPireq:** Internal logic sets this bit to generate the external service request output. It is a direct reflection of the inverse state of the SVCREQP* pin; it is the active-high output of the latch that drives the SVCREQP* pin. This bit can be scanned by the host to detect an active service request. This bit is cleared by the internal logic at the beginning of the hardware service-acknowledge cycle or by toggling InTen (PFCR[4]). Clearing PIR automatically deactivates the SVCREQP* output and clears the SRP bit (SVRR[3]). |
| 6:5 | **PPort and Pipeline:** These two bits indicate which of the two functional blocks of the parallel port are requesting service. When PPort is set, it indicates that the parallel channel control state machine is the cause of the request; when Pipeline is set, it indicates that the data pipeline is requesting service. If both bits are set, it indicates that both blocks are requesting service simultaneously. |
| 4:0 | **Reserved:** The remainder of the bits in the PIR always return '0' when read by the host and should not be modified. |

### 7.1.6 Prescaler Period Register

Register Name: PPR — 8-Bit Hex Address: 7E
Register Description: Prescale Period — Default Value: FF
Access: R/W

| *Bit 7* | *Bit 6* | *Bit 5* | *Bit 4* | *Bit 3* | *Bit 2* | *Bit 1* | *Bit 0* |
|---|---|---|---|---|---|---|---|
| Binary Value | | | | | | | |

The PPR sets the divisor used to generate the time period for 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.

### 7.1.7 Service Request Register

Register Name: SVRR — 8-Bit Hex Address: 67
Register Description: Service Request — Default Value: 00
Access: Read only

| *Bit 7* | *Bit 6* | *Bit 5* | *Bit 4* | *Bit 3* | *Bit 2* | *Bit 1* | *Bit 0* |
|---|---|---|---|---|---|---|---|
| DMAREQ | X | X | X | SRP | X | X | X |

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

| Bit | Description |
|---|---|
| 7 | **DMA Request Status:** When this bit is set to '1', it indicates that a request is pending. |
| 6:4 | These bits are not used and are don't cares. |
| 3 | **Service Request Parallel:** When this bit is set to '1', it indicates that a request is pending. |
| 2:0 | These bits are not used and are don't cares. |

## 7.2 Virtual Registers

The 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 are 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 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 in one of two ways: either through activation of the SVCACKP* input pin (hardware activated), or through 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 Chapter 5.0 for a discussion of the differences between these two modes.

### 7.2.1 End-of-Service Request Register

Register Name: EOSRR — 8-Bit Hex Address: 60
Register Description: End-of-Service Request — Default Value: XX
Access: Write only

| *Bit 7* | *Bit 6* | *Bit 5* | *Bit 4* | *Bit 3* | *Bit 2* | *Bit 1* | *Bit 0* |
|---|---|---|---|---|---|---|---|
| X | X | X | X | X | X | X | X |

The EOSRR is a ‘dummy’ location and is used to signal the end of a hardware-activated service-acknowledge procedure, invoked by the activation of SVCACKP*. The data pattern written is a ‘don’t care’ value. Writing this location causes the CD1283 to perform its internal switch out of the service-acknowledge context. This register is used only during a hardware-activated service acknowledge and must not be written during Poll-mode operation.

### 7.2.2 Parallel Interrupt Vector Register

Register Name: PIVR — 8-Bit Hex Address: 40
Register Description: Parallel Interrupt Vector — Default Value: 00
Access: Read only

| *Bit 7* | *Bit 6* | *Bit 5* | *Bit 4* | *Bit 3* | *Bit 2* | *Bit 1* | *Bit 0* |
|---|---|---|---|---|---|---|---|
| User-Defined – Upper 5 Bits of LIVR | | | | | IT2 | IT1 | IT0 |

The value in this register is placed on the data bus, DB[7:0], when SVCACKP* is activated in response to an active SVCREQP*. See Section 7.3.5 on page 60 for more details on the LIVR.

**Table 10. PIVR[2:0] Encoding**

| IT2 | IT1 | IT0 | Description |
|---|---|---|---|
| 0 | 0 | 0 | No active interrupt. |

**Table 10. PIVR[2:0] Encoding**

| IT2 | IT1 | IT0 | Description |
|---|---|---|---|
| 0 | 0 | 1 | Invalid. |
| • | • | • | |
| 0 | 1 | 1 | |
| 1 | 0 | 0 | The parallel channel state machine requests service. |
| 1 | 0 | 1 | The parallel channel data pipeline requests service. |
| 1 | 1 | 0 | Both the parallel port state machine and the parallel port data pipeline request service. |
| 1 | 1 | 1 | Invalid. |

## 7.3 Parallel Pipeline Registers

### 7.3.1 Data Error Register

| Register Name: DER<br>Register Description: Data Error<br>Access: Read only | | | | | | 8-Bit Hex Address: 33<br>Default Value: 00 | |
|---|---|---|---|---|---|---|---|
| *Bit 7* | *Bit 6* | *Bit 5* | *Bit 4* | *Bit 3* | *Bit 2* | *Bit 1* | *Bit 0* |
| 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.

A read of this register has no effect on the error status. A write to this register clears all 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. Under normal circumstances, data errors should never occur. This register is cleared during device reset.

| Bit | Description |
|---|---|
| 7 | **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*). |
| 6 | **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. |
| 5 | **Buffer Write Error:** This bit indicates that a system write to the DMA buffer occurred while it still contained data. |
| 4 | **Buffer Read Error:** This bit indicates that a system read from the DMA buffer occurred while it was empty. |
| 3 | **Holding Register 1 Write Error:** This bit indicates that a system write to PFHR1 (Parallel FIFO Holding register 1) occurred while it still contained data. |
| 2 | **Holding Register 1 Read Error:** This bit indicates that a system read from PFHR1 occurred while it was empty. |
| 1 | **Holding Register 2 Write Error:** This bit indicates that a system write to PFHR2 (Parallel FIFO Holding register 2) occurred while it still contained data. |
| 0 | **Holding Register 2 Read Error:** This bit indicates that a system read from PFHR2 occurred while it was empty. |

## 7.3.2 DMA Buffer Data Register

| Register Name: DMABUF<br>Register Description: DMA Buffer Data high<br>Access: R/W | | | | | | | 8-Bit Hex Address: 30<br>Default Value: 00 |
|---|---|---|---|---|---|---|---|
| *Bit 15* | *Bit 14* | *Bit 13* | *Bit 12* | *Bit 11* | *Bit 10* | *Bit 9* | *Bit 8* |
| DMA Buffer Data High Byte | | | | | | | |

| Register Name: DMABUF<br>Register Description: DMA Buffer Data low<br>Access: R/W | | | | | | | 8-Bit Hex Address: 30<br>Default Value: 00 |
|---|---|---|---|---|---|---|---|
| *Bit 7* | *Bit 6* | *Bit 5* | *Bit 4* | *Bit 3* | *Bit 2* | *Bit 1* | *Bit 0* |
| DMA Buffer Data Low Byte | | | | | | | |

This 16-bit data register is used to buffer DMA data transfers to and from the CD1283. Under normal operating conditions, this register is only accessed during a DMA data transfer cycle. If DMAbufWe (PFCR0) is set to ‘1’ and DMAdir (PFCR[5]) is set to ‘1’, 16-bit data can be transferred from the host to the FIFO by directly writing to the DMABUF. The data automatically moves forward into the FIFO through the Data Pipeline Holding registers. The user must ensure 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 ‘1’, data transferred on DB[15:8] is the first byte transferred into the data pipeline and DB[7:0] is transferred second. If BYTESWAP is set to ‘0’ this sequence is reversed. The same applies during data read during DMA transfers: if BYTESWAP is set to ‘1’, data from the data pipeline moves to the upper byte of DMABUF, the next byte moves into the lower byte. Again, if BYTESWAP is set to ‘0’, this sequence is reversed.

These resisters can be read through DMA acknowledge or PIO cycles. However, the DMABUF registers can only be read when the DMAREQ* signal is active. If DMAREQ* is inactive, the DMABUF registers will be empty. DMAfull (HRSR[3]) indicates if the DMABUF register is empty when DMAREQ* is inactive.

## 7.3.3 Holding Register Status Register

| Register Name: HRSR<br>Register Description: Holding Status<br>Access: Read only | | | | | | | 8-Bit Hex Address: 34<br>Default Value: 04 |
|---|---|---|---|---|---|---|---|
| *Bit 7* | *Bit 6* | *Bit 5* | *Bit 4* | *Bit 3* | *Bit 2* | *Bit 1* | *Bit 0* |
| HR1full | HR1tag | HR2full | HR2tag | DMAfull | DMAmpty | DMAact | Ctnot0 |

The HRSR is read-only and indicates current data pipeline status This register is not directly set to any particular value at device reset, but reflects the current state of bits in other registers

| Bit | Description |
|---|---|
| 7:6 | **HR1full and HR1tag:**These two bits indicate status of PFHR1. Bit 7 indicates that the register contains data and bit 6 indicates that the data is tagged. Both bits can be set simultaneously. |
| 5:4 | **HR2full and HR2tag:**These two bits indicate status of PFHR2. Bit 5 indicates that the register contains data and bit 4 indicates that the data is tagged. Both bits can be set simultaneously. |
| 3:2 | **DMAfull and DMAmpty:** 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. |
| 1 | **DMAact:** This bit when set, indicates that the DMA handshake is active and that DMA service is requested, but not yet complete (DMAREQ* active, waiting for DMAACK*). |
| 0 | **Ctnot0:** This bit indicates that the RLE counter is not zero, thus run-length encoding/decoding is in progress. |

### 7.3.4 Host Timeout Value Register

Register Name: HTVR
Register Description: Host Timeout Value
Access: Read/Write
8-Bit Hex Address: 24
Default Value: FF

| *Bit 7* | *Bit 6* | *Bit 5* | *Bit 4* | *Bit 3* | *Bit 2* | *Bit 1* | *Bit 0* |
|---|---|---|---|---|---|---|---|
| Host Timeout Period | | | | | | | |

HVTR holds the 8-bit value used to set the Host Timeout period. The HTVR is an unsigned, binary value. The reset state of this register is ‘0xFF’.

A function missing in the revision ‘C’ and earlier devices is an on-chip timer to indicate that the remote host has not responded in a specified time period. The Host timeout is defined in the *IEEE STD 1284 Specification* as a period of one second.

The revision ‘D’ device adds a user-programmable timer that provides a timeout if the remote host does not respond to specific parallel port transactions. The timer is started by the parallel port state machine each time it starts a sequence requiring a host response. Activation of the timer is automatic and an interrupt is generated to the local host CPU if the timer expires before the remote host responds.

***Note:*** Users familiar with the IEEE specification note that the events that start the timer cause the peripheral device to move to a state where it waits for a remote host-generated event. For example, during the negotiation sequence after event 2, the peripheral waits for event 3, a host-generated event. If the host does not respond and moves the negotiation sequence to event 4 within one second, the peripheral enters the ‘host Timeout’ condition.

The timer is a 14-bit counter clocked by the system clock (CLK) prescaled (divided) by 2048. Program the 8-bit Host timeout Value register (HTVR, address offset 0x’24), which is then compared with the most-significant 8 bits of the 14-bit counter. Each time the parallel port executes an event requiring a host response, the 14-bit counter is started (from 0x’00). It counts up until either the expected event occurs or the count matches the value in HTVR. If a match occurs, a timeout condition exists. The HTVR need only be loaded once, typically during device initialization.

The value placed in HTVR yields an approximate one second count time, based on the value of the input CLK. For example, if the system clock driving the device is 25 MHz, the HTVR should be loaded with 0xC0. The following equation provides an example.

$$\frac{25MHz}{2048} = 12207_{10} = 2FAF_{16}$$

The computed value is rounded up to the next largest whole hex value, in this case ‘0x3000’. Load HTVR with the most-significant 8 bits of this value, left-shifted two places since HTVR is a 14-bit counter. This results in a value of ‘0xC0’. For 20 MHz, the value is computed to be ‘0x9C’ and for 16 MHz, the value is ‘0x7C’; values for other clocks can be easily computed in the same manner. At reset, the HTVR defaults to a value of ‘0xFF’; this prevents the extremely short Timeouts that occur if the register is cleared at device reset and not initialized.

A timeout causes a negotiation status change interrupt. This status is displayed as 0x22 in the Negotiation Status Register (Host Timeout (bit 5), and the code for return to Compatibility mode (0010) in the result code field). When Compatibility mode is reentered, the port control state machine waits in a locked state until signals on the parallel port return to normal Compatibility mode conditions.

For debug purposes, disable the host Timeout timer by setting PCR bits 2 and 3 (Host Timer Test [1:0]). In this case, no Timeouts occur and the link can hang indefinitely while waiting for a host-generated event.

### 7.3.5 Local Interrupt Vector Register

| Register Name: LIVR | | | | | | 8-Bit Hex Address: 18 | |
|---|---|---|---|---|---|---|---|
| Register Description: Local Interrupt Vector | | | | | | Default Value: 00 | |
| Access: R/W | | | | | | | |
| *Bit 7* | *Bit 6* | *Bit 5* | *Bit 4* | *Bit 3* | *Bit 2* | *Bit 1* | *Bit 0* |
| User-Defined Bits | | | | | X | X | X |

This read/write register can be initialized to any desired value and, when read in the normal context (that is, not a service acknowledge context), the same value is returned. The upper 5 bits are copied into the appropriate vector register, PIVR when the SVCACKP* signal is activated *and* SVCREQP* is active. During this hardware-activated service acknowledge read cycle, the PIVR is driven onto the data bus, DB[7:0]. Bits 7:3 come from LIVR and bits 2:0 are supplied by the CD1283 (Section 7.2.2 on page 56 for details). This value can be used as a vector into the appropriate service routine (typical in Motorola-type systems) or as a device identifier for systems with multiple, daisy-chained CD1283s. Bits 2:0 are ignored. Initialization of this register is only necessary if vectored interrupts are used.

| Bit | Description |
|---|---|
| 7:3 | **User-Defined Interrupt Vector:** Host software can use these five 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 the case of daisy-chain systems made up of multiple CD1283s, these bits are used to define the device number in the chain. |
| 2:0 | These bits are ‘don’t cares’. |

### 7.3.6 Parallel Auxiliary Control Register

| Register Name: PACR | | | | | | | 8-Bit Hex Address: 3F |
|---|---|---|---|---|---|---|---|
| Register Description: Parallel Auxiliary Control | | | | | | | Default Value: 00 |
| Access: R/W | | | | | | | |
| *Bit 7* | *Bit 6* | *Bit 5* | *Bit 4* | *Bit 3* | *Bit 2* | *Bit 1* | *Bit 0* |
| ShrtTen | ShrtStal | StaleOff | FIFOlock | ClearTO | 0 | AsyncDMA | 0 |

The PACR provides some special functions for the parallel data path and interrupt-generation circuitry.

The upper 2 bits are used to change the basic timing of the timers associated with the data pipeline. Bit 5 can disable the stale data time.

| Bit | Description |
|---|---|
| 7 | **ShrtTen:** This function shortens the Prescaler count cycle that generates the internal 10-µs clock (based on a 25-MHz system clock) for the stale data counter. This bit is cleared by RESET*. If set, 10-µs 'ticks' of the counter will be generated every two CLKs; the normal period is one 'tick' every 250 CLKs. |
| 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'.<br>If both ShrtTen and ShrtStal are set, the stale data timer counts on every other CLK. |
| 5 | **StaleOff:** When this bit is set, it 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. |
| 4 | **FIFOlock:** This bit causes the FIFO to stop accepting data from the parallel channel state machine. This action makes 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 timeout on the 1284 bus.<br>Setting this bit in ECP Forward mode may cause a stall condition event 35 because event 36 will not occur until FIFOlock is cleared. The ECP mode host transfer recovery handshake sequence (from event 35 stall) is supported and the byte transit discarded as required by the specification. This bit does not provide an effective means to flow control the host. |
| 3 | **ClearTO:** The Clear Timeout bit is a reset bit for the timeout status latch logic. When toggled by software, the timeout status in the PFSR is cleared; it may be left set to disable the Timeout status function. Note that if this bit is left set, the OneChar interrupt condition will never become true since there will be no FIFO timeout activity. |
| 2 | **Reserved:** This read-only bit is always '0'. |
| 1 | **AsyncDMA:** This bit 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.<br>Refer to the Section 8.3.1 on page 78 for specific timing relationships between CLK and DMAACK* when AsyncDMA is enabled. |
| 0 | **Reserved:** This read-only bit is always '0'. |

### 7.3.7 Parallel Channel Reset Register

| Register Name: PCRR | | | | | | | 8-Bit Hex Address: 6C |
|---|---|---|---|---|---|---|---|
| Register Description: Parallel Channel Reset | | | | | | | Default Value: 00 |
| Access: R/W | | | | | | | |
| *Bit 7* | *Bit 6* | *Bit 5* | *Bit 4* | *Bit 3* | *Bit 2* | *Bit 1* | *Bit 0* |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | PChReset |

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

| Bit | Description |
|---|---|
| 7:1 | These bits are not used and must always be ‘0’. |
| 0 | **PChReset:** When this bit is set, it asserts the equivalent of a hardware power-on reset to the parallel channel, Channel 0. If set by the host, PChReset 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. |

## 7.3.8 Parallel FIFO Control Register

Register Name: PFCR
Register Description: Parallel FIFO Control
Access: R/W
8-Bit Hex Address: 31
Default Value: 00

| *Bit 7* | *Bit 6* | *Bit 5* | *Bit 4* | *Bit 3* | *Bit 2* | *Bit 1* | *Bit 0* |
|---|---|---|---|---|---|---|---|
| FIFOres | DMAen | DMAdir | IntEn | RLEen | setTAG | ErrEn | DMAbufWe |

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

After hard reset (either through the RESET* input pin or by setting bit 0 in the PCRR), this register is cleared to all zeroes.

| Bit | Description (Sheet 1 of 2) |
|---|---|
| 7 | **FIFO Reset:** This bit 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. |
| 6 | **DMA Enable:** This bit must be set for DMA requests to move data to/from the FIFO. When DMAen is set to ‘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 through 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 through the data pipeline. |
| 5 | **DMA Direction:** This bit sets the direction of transfer between the parallel FIFO and system memory. If DMAdir is set to ‘1’, the direction is transmit (system memory to the parallel FIFO); if it is ‘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 is complete, that DMAdir selection must be maintained during any other changes to the control bits of the PFCR. |
| 4 | **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 may toggle this bit to signal the completion of the service-acknowledge cycle and clear the current status in the PIR, SVRR, and LIVR. Toggling this bit updates the state of SVCREQP* and the PIR according to the current state of PCISR, DERR and PFSR. For this reason, PCISR, DERR, and PFSR should be read and cleared at the beginning of the service routine. These registers should be checked again at the end of the service routine to ensure that no requests were skipped because an edge-sensitive interrupt controller may not detect a request that is already active when the program returns from the service routine. |
| 3 | **RLE Enable:** This bit enables run-length 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 so that the PFHR1 and PFHR2 are kept full 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 DMABUF 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 ‘0’ and both DMAen and DMAbufWe must be ‘0’. |

| Bit | Description (Continued) (Sheet 2 of 2) |
|---|---|
| 2 | **Set Tag:** This bit specifies that the next character written to the parallel channel through the PFHR1 is to be tagged as an ECP or EPP special character. This bit is cleared by the write to PFHR1, thus this bit must be set each time a tagged character is to be written. |
| 1 | **Error Interrupt Enable:** This bit enables a non-zero DataErr status to cause an interrupt (if IntEn is also set). |
| 0 | **DMA Buffer Write Enable:** This bit must be set to enable host writes to the DMABUF register. It also enables the FIFO data pipeline to empty DMABUF 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. |

### 7.3.9 Parallel FIFO Empty Pointer Register

Register Name: PFEP — 8-Bit Hex Address: 39
Register Description: Parallel FIFO Empty Pointer — Default Value: 00
Access: R/W

| *Bit 7* | *Bit 6* | *Bit 5* | *Bit 4* | *Bit 3* | *Bit 2* | *Bit 1* | *Bit 0* |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 6-Bit Binary FIFO Pointer Value | | | | | |

This register contains 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 register is cleared by a device or FIFO reset.

### 7.3.10 Parallel FIFO Fill Pointer Register

Register Name: PFFP — 8-Bit Hex Address: 38
Register Description: Parallel FIFO Fill Pointer — Default Value: 00
Access: R/W

| *Bit 7* | *Bit 6* | *Bit 5* | *Bit 4* | *Bit 3* | *Bit 2* | *Bit 1* | *Bit 0* |
|---|---|---|---|---|---|---|---|
| 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 register is cleared by a device or FIFO reset.

### 7.3.11 Parallel FIFO Holding Registers

Register Name: PFHR1 — 8-Bit Hex Address: 35
Register Description: Parallel FIFO Holding Register 1 — Default Value: 00
Access: R/W

| *Bit 7* | *Bit 6* | *Bit 5* | *Bit 4* | *Bit 3* | *Bit 2* | *Bit 1* | *Bit 0* |
|---|---|---|---|---|---|---|---|
| 8-Bit Character Data | | | | | | | |

| Register Name: PFHR2 | | | | | | | 8-Bit Hex Address: 36 |
|---|---|---|---|---|---|---|---|
| Register Description: Parallel FIFO Holding Register 2 | | | | | | | Default Value: 00 |
| Access: R/W | | | | | | | |
| *Bit 7* | *Bit 6* | *Bit 5* | *Bit 4* | *Bit 3* | *Bit 2* | *Bit 1* | *Bit 0* |
| 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 DMABUF register. The flow is to the FIFO if DMAdir is a ‘1’, and from the FIFO if DMAdir is ‘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 the 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 ‘0’). This can be done at the same time that setTAG is set to ‘1’.

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

### 7.3.12 Parallel FIFO Quantity Register

| Register Name: PFQR | | | | | | | 8-Bit Hex Address: 3A |
|---|---|---|---|---|---|---|---|
| Register Description: Parallel FIFO Quantity | | | | | | | Default Value: 00 |
| Access: R/W | | | | | | | |
| *Bit 7* | *Bit 6* | *Bit 5* | *Bit 4* | *Bit 3* | *Bit 2* | *Bit 1* | *Bit 0* |
| 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 ‘0’), PFQR counts data characters in the FIFO. In the transmit direction (DMAdir is set to ‘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.

### 7.3.13 Parallel FIFO Status Register

| Register Name: PFSR | | | | | | | 8-Bit Hex Address: 32 |
|---|---|---|---|---|---|---|---|
| Register Description: Parallel FIFO Status | | | | | | | Default Value: 40 |
| Access: Read only | | | | | | | |
| *Bit 7* | *Bit 6* | *Bit 5* | *Bit 4* | *Bit 3* | *Bit 2* | *Bit 1* | *Bit 0* |
| FFfull | FFempty | Timeout | HRtag | HRdata | Stale | OneChar | DataErr |

The PFSR is read-only and 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. This register is cleared by device or FIFO reset.

| Bit | Description |
|---|---|
| 7 | **Parallel FIFO Full:** If this bit is set, it indicates the parallel FIFO is full. |
| 6 | **Parallel FIFO Empty:** If this bit is set, the parallel FIFO is empty. |
| 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 ClearTo in the PACR or by a FIFO reset in the PFCR. |
| 4 | **Holding Register Tag:** This bit indicates that a tagged character is in either the PFHR1, PFHR2, or both. This bit being set will cause a host interrupt to be generated (if enabled). The host should examine the HRSR to determine the exact cause(s) of this bit being set. |
| 3 | **Holding Register Data:** If this bit is set, it indicates that either the PFHR1, PFHR2, or both contain data. |
| 2 | **Stale:** This bit is set when the stale data timer expires (see description of SDTPR). If a single byte remains in the data pipeline when this bit is set, a host interrupt is generated, the OneChar bit is set, and new data entering the FIFO will not move into PFHR1 until PFHR2 is emptied. If two or more bytes remain in the pipeline when this bit is set, a host interrupt is not generated, however, a DMA request will be generated if enabled. |
| 1 | **One Character:** In the receive direction, when this bit is set it indicates that the FIFO is empty and stale, and one character remains in the PFHR2. This condition occurs if an odd number of bytes is transferred through the parallel interface. Since DMA cycles only moves an even numbers of bytes (words), an odd transfer leaves one byte remaining. Host software must remove this character outside of DMA transfer cycles. |
| 0 | **Data Error:** When this bit is set, it indicates that one or more of the bits in the DER (Data Error register) is set. |

## 7.3.14 Parallel FIFO Threshold Register

Register Name: PFTR — 8-Bit Hex Address: 3B
Register Description: Parallel FIFO Threshold — Default Value: 00
Access: R/W

| *Bit 7* | *Bit 6* | *Bit 5* | *Bit 4* | *Bit 3* | *Bit 2* | *Bit 1* | *Bit 0* |
|---|---|---|---|---|---|---|---|
| 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 PFQR (Parallel FIFO Quantity register) 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) and DMABUF (if DMA is enabled) are kept filled so that tagged data (for example, ECP-mode addresses) can 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 is automatically placed in the Pipeline registers. If the PFTR were programmed to x’40 bytes, x’44 bytes must arrive to trigger a DMA transfer.

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

### 7.3.15 Run-Length Count Register

| Register Name: RLCR<br>Register Description: Run-Length Count<br>Access: R/W | | | | | | | 8-Bit Hex Address: 37<br>Default Value: 00 |
|---|---|---|---|---|---|---|---|
| *Bit 7* | *Bit 6* | *Bit 5* | *Bit 4* | *Bit 3* | *Bit 2* | *Bit 1* | *Bit 0* |
| 0 | 7-Bit Unsigned Binary Count | | | | | | |

This register works with PFHR1 and PFHR2 to perform run-length encoding/decoding when the RLEen bit (PFCR[3]) is set (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 can 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 zero, 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 directly read the tagged Holding register to remove the character from the pipeline and clear the tag.

### 7.3.16 Stale Data Timer Count Register

| Register Name: SDTCR<br>Register Description: Stale Data Timer Count<br>Access: R/W | | | | | | | 8-Bit Hex Address: 3D<br>Default Value: 00 |
|---|---|---|---|---|---|---|---|
| *Bit 7* | *Bit 6* | *Bit 5* | *Bit 4* | *Bit 3* | *Bit 2* | *Bit 1* | *Bit 0* |
| 8-Bit Stale Data Timer Count | | | | | | | |

This register determines the period 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 zero, the Stale bit (PFSR[2]) 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 causes an interrupt to the host to remove the character by reading PFHR2.

This register is cleared by device or FIFO reset. Clearing it manually causes the Stale bit to be true.

### 7.3.17 Stale Data Timer Period Register

| Register Name: SDTPR | | | | | | | 8-Bit Hex Address: 3C |
|---|---|---|---|---|---|---|---|
| Register Description: Stale Data Timer Period | | | | | | | Default Value: 00 |
| Access: R/W | | | | | | | |
| *Bit 7* | *Bit 6* | *Bit 5* | *Bit 4* | *Bit 3* | *Bit 2* | *Bit 1* | *Bit 0* |
| 8-Bit Stale Data Timeout Value | | | | | | | |

This register provides a user-defined period value for use as the timeout value of the stale data timer (see SDTCR).

With a 25-MHz CLK input to the device, 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 a 0.1-ms timeout would result in a time delay of between 0.09 and 0.1 ms.)

The SDTPR is cleared by device reset.

## 7.4 Parallel Port Registers

### 7.4.1 EPP Address Register

| Register Name: EAR | | | | | | | 8-Bit Hex Address: 25 |
|---|---|---|---|---|---|---|---|
| Register Description: EPP Address | | | | | | | Default Value: 00 |
| Access: R/W | | | | | | | |
| *Bit 7* | *Bit 6* | *Bit 5* | *Bit 4* | *Bit 3* | *Bit 2* | *Bit 1* | *Bit 0* |
| 8-Bit Binary Value | | | | | | | |

This register is only used during EPP mode. The CD1283 deposits the value obtained during an EPP address write command in this register. The CD1283 provides this value in response to an EPP address read command.

### 7.4.2 Input Value Register

| Register Name: IVR | | | | | | | 8-Bit Hex Address: 2E |
|---|---|---|---|---|---|---|---|
| Register Description: Input Value | | | | | | | Default Value: XX |
| Access: Read only | | | | | | | |
| *Bit 7* | *Bit 6* | *Bit 5* | *Bit 4* | *Bit 3* | *Bit 2* | *Bit 1* | *Bit 0* |
| 0 | 0 | 0 | 0 | A1284 | nInit | HstBsy | HstClk |

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

| Bit | Description |
|---|---|
| 7:4 | These read-only bits are always ‘0’. |
| 3 | **A1284** |
| 2 | **nInit:** (active-low Init input) |
| 1 | **HstBsy:** (Host Busy) |
| 0 | **HstClk:** (Host Clock) |

### 7.4.3 Manual Data Register

Register Name: MDR
Register Description: Manual Data
Access: R/W
8-Bit Hex Address: 21
Default Value: 00

| *Bit 7* | *Bit 6* | *Bit 5* | *Bit 4* | *Bit 3* | *Bit 2* | *Bit 1* | *Bit 0* |
|---|---|---|---|---|---|---|---|
| 8-Bit Binary Data | | | | | | | |

This read/write register can read the state of the PD[7:0] signals in any mode. If the ManMd bit (PCR[7])is set along with the MMDir and ManOE bits (PCR[1:0]), then the value written into this register is driven onto the PD[7:0] signals.

### 7.4.4 Negotiation Enable Register

Register Name: NER
Register Description: Negotiation Enable
Access: R/W
8-Bit Hex Address: 28
Default Value: 00

| *Bit 7* | *Bit 6* | *Bit 5* | *Bit 4* | *Bit 3* | *Bit 2* | *Bit 1* | *Bit 0* |
|---|---|---|---|---|---|---|---|
| 0 | RID | 0 | EPP | RLE | ECP | RVB | RVN |

Each bit set along with E1284 (PCR[6]) allows the CD1284 to engage in IEEE STD 1284 negotiations and move into the corresponding protocol. It is assumed that the peripheral host software responds to a request for slave ID and is able to send an ID string in any protocol that it supports. In response to an ID request, the CD1284 does not provide a method of storing and automatically sending an ID string. Note that the EPP protocol does not have provision for slave ID requests.

| Bit | Description |
|---|---|
| 7 | **Reserved:** This read-only bit is always ‘0’. |
| 6 | **Request Slave ID** |
| 5 | **Reserved:** This bit must always be ‘0’. |
| 4 | **EPP Mode Enable** |
| 3 | **Run Length Encoding in ECP Mode Enable** |
| 2 | **ECP Mode Enable** |
| 1 | **Reverse Byte Mode Enable** |
| 0 | **Reverse Nibble Mode Enable** |

## 7.4.5 Negotiation Status Register

| NSR<br>Negotiation Status<br>R/W | | | | 29<br>00 | | | |
|---|---|---|---|---|---|---|---|
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| NegOK | NegFl | HostTO | Invalid | 4-bit negotiation result code | | | |

The results of negotiation attempts are stored in this register.

| Bit | Description |
|---|---|
| 7 | **NegOK:** The state of this bit indicates that the negotiation was successful. |
| 6 | **Negotiation Failed:** The state of this bit indicates that the negotiation failed. The result code indicates which mode attempted. |
| 5 | **Host Timeout:** This bit indicates that a host time-out occurred on the parallel channel. The accompanying 4-bit result code indicates that the link has returned to Compatibility mode (x02). See description of HTVR on page 59. |
| 4 | **Invalid:** The state of this bit indicates that the state machine is in an invalid state due to the last negotiation sequence and has reentered Compatibility mode |
| 3:0 | The lower 4 bits contain a result code, which shows the current mode. |

| Bits | | | | Description |
|---|---|---|---|---|
| 3 | 2 | 1 | 0 | |
| 0 | 0 | 0 | 0 | Compatible mode — no negotiation. |
| 0 | 0 | 0 | 1 | Failed negotiation. |
| 0 | 0 | 1 | 0 | Compatible mode — termination of a 1284 mode. |
| 0 | 0 | 1 | 1 | Reserved. |
| 0 | 1 | 0 | 0 | |
| 0 | 1 | 0 | 1 | |
| 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 |

Any change in the mode of the parallel port is reported to the peripheral host by interrupt if the NegCh bit is set in the PCIER; host software then reads 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.

## 7.4.6 Ones Detect Register

Register Name: ODR — 8-Bit Hex Address: 2D
Register Description: Ones Detect — Default Value: 00
Access: R/W

| *Bit 7* | *Bit 6* | *Bit 5* | *Bit 4* | *Bit 3* | *Bit 2* | *Bit 1* | *Bit 0* |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | A1284 | nInit | HstBsy | HstClk |

Setting the bits in this register enables the CD1284 to generate an interrupt – if SigCh (PCIER[4]) is set – when the selected signal changes from low to high (rising edge). Bits 7:4 are reserved and must be written as zeros; they return zero when read. The settings in this register have no effect (that is, SigCh interrupt is not generated) unless the device is in Manual mode.

## 7.4.7 Output Value Register

Register Name: OVR — 8-Bit Hex Address: 2B
Register Description: Output Value — Default Value: 48
Access: Write only

| *Bit 7* | *Bit 6* | *Bit 5* | *Bit 4* | *Bit 3* | *Bit 2* | *Bit 1* | *Bit 0* |
|---|---|---|---|---|---|---|---|
| PerBsy | PerClk | AkDaRq | XFlag | nDatAv | 0 | 0 | 0 |

This register controls output signals. In Manual mode, all signals are controlled by these register settings. In Compatibility and EPP modes, PerBsy and PerClk are controlled by the internal parallel port state machine while AkDaRq, xFlag, and nDataAv are controlled by this register. In ECP mode, the settings in this register have no effect.

| Bit | Description |
|---|---|
| 7 | **Peripheral Busy:** User-controlled in Manual mode only. |
| 6 | **Peripheral Clock:** User-controlled in Manual mode only. |
| 5 | **Acknowledge Data Request:** In Compatible mode, this signal is the PError (Peripheral Error) signal.<br>In EPP mode, this signal is auxiliary and is a user-defined signal (USER 1). |
| 4 | **XFlag:** In Compatible mode, this signal is the SELCT (Select) signal.<br>In EPP mode, this signal is auxiliary and is a user-defined signal (USER 2). |
| 3 | **Negative-true Data Available:** In Compatible mode, this signal is the nFault (negative-true fault) signal.<br>In EPP mode, this signal is auxiliary and is a user-defined signal (USER 3). |
| 2:0 | **Reserved:** These bits must be written as ‘0’. |

### 7.4.8 Parallel Channel Interrupt Enable Register

| Register Name: PCIER | | | | | | 8-Bit Hex Address: 22 | |
|---|---|---|---|---|---|---|---|
| Register Description: Parallel Channel Interrupt Enable | | | | | | Default Value: 00 | |
| Access: R/W | | | | | | | |
| *Bit 7* | *Bit 6* | *Bit 5* | *Bit 4* | *Bit 3* | *Bit 2* | *Bit 1* | *Bit 0* |
| 0 | TimEn | NegCh | SigCh | EPPAW | DirCh | IDReq | nINIT |

### 7.4.9 Parallel Channel Interrupt Status Register

| Register Name: PCISR | | | | | | 8-Bit Hex Address: 23 | |
|---|---|---|---|---|---|---|---|
| Register Description: Parallel Channel Interrupt Status | | | | | | Default Value: 00 | |
| Access: R/W | | | | | | | |
| *Bit 7* | *Bit 6* | *Bit 5* | *Bit 4* | *Bit 3* | *Bit 2* | *Bit 1* | *Bit 0* |
| 0 | TimOvr | NegCh | SigCh | EPPAW | DirCh | IDReq | nINIT |

The PCIER and PCISR provide control and status of interrupts generated by the parallel channel control state machine. They have the same bit definitions. Each bit in the PCIER enables the interrupt of the same type in the PCISR. A write of any value to the PCISR in response to an interrupt request, causes it to clear and the interrupt request to be removed.

| Bit | Description |
|---|---|
| 7 | **Reserved:** This read-only bit is always ‘0’. |
| 6 | **TimEn/TimOvr:** Used for factory test purposes only. |
| 5 | **NegCh:** The state of this bit indicates that a change occurred in the negotiation status of the port. The NSR indicates the new status of the parallel port. |
| 4 | **SigCh:** This bit instructs the parallel port to generate an interrupt when any of the signals specified by the ZDR or ODR change state as programmed. |
| 3 | **EPPAW:** The state of this bit indicates that the remote master has written an EPP address to the CD1283. The new EPP address value is placed in the EAR. |
| 2 | **DirCh:** This bit indicates that the host-side parallel port changed the direction of the interface. Generally, this is in response to a request made by the CD1283 through the RevRq bit in the SCR (bit 0). DirCh indicates that the direction was reversed through the defined protocol and the CD1283 can now send data to the master. This bit is only valid in ECP and EPP (bidirectional) modes. |
| 1 | **IDReq:** The state of this bit indicates that the host has requested that the CD1283 send its ID data string. The peripheral host should send the appropriate ID string (this is application-dependent). |
| 0 | **nINIT:** This interrupt is generated when an nINIT pulse is received in Compatibility mode. The interrupt occurs on the leading edge of the nINIT pulse. |

### 7.4.10 Parallel Configuration Register

| Register Name: PCR | | | | | | 8-Bit Hex Address: 20 | |
|---|---|---|---|---|---|---|---|
| Register Description: Parallel Configuration | | | | | | Default Value: 00 | |
| Access: R/W | | | | | | | |
| *Bit 7* | *Bit 6* | *Bit 5* | *Bit 4* | *Bit 3* | *Bit 2* | *Bit 1* | *Bit 0* |
| ManMd | E1284 | ETxfr | Ig_SEL | HTmrTst[1] | HTmrTst[0] | MMDir | ManOE |

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

<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>7:5</td><td>Mode Control: These three bits control the type of transfer desired and whether or not it is enabled to do so.<br>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 MDR (Manual Data register) and ManOE is the output enable when MMDir = 1 (output mode).<br>E1284 allows the parallel port to engage in IEEE 1284 negotiations. ETxfr enables data transfers. ETxfr enable is only used for data transfers. EPP address read and write functions do not require that the ETxfr bit be set
<table>
<tr><th>ManMd</th><th>E1284</th><th>Etxfr</th><th>Mode</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>Compatibility mode; transfers disabled.</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>Compatibility mode; transfers enabled.</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>IEEE 1284 negotiation; transfers disabled.</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>IEEE 1284 negotiation; transfers enabled.</td></tr>
<tr><td>1</td><td>X</td><td>X</td><td>Manual mode.</td></tr>
</table></td></tr>
<tr><td>4</td><td>Ig_SEL: This bit prevents the CD1284 from considering the state of the nSLCTIN input when deciding whether or not to accept Compatibility mode forward data transfers.<br>When Ig_SEL is reset, nSLCTIN must be active (low) to receive data on the parallel port in response to an nStrobe input. If Ig_SEL is set, nSLCTIN is not considered and data is accepted regardless of its state. The Ig_SEL bit should be set/reset together with the E1284 bit.</td></tr>
<tr><td>3:2</td><td>Host Timer Test Control: These two bits control the clock rate of the host timeout timer and are intended primarily for manufacturing test purposes. As such, normal user-level programming should leave these bits cleared (‘0’). When these bits are set to ‘1’, the timer is completely disabled, this is useful for factory debug purposes.</td></tr>
<tr><td>1:0</td><td>Manual Mode Control: These two bits provide direction and output enable manual control over the parallel port.
<table>
<tr><th>MMDir</th><th>ManOE</th><th>Mode</th></tr>
<tr><td>0</td><td>0</td><td>Reverse direction.</td></tr>
<tr><td>0</td><td>1</td><td>Reverse direction.</td></tr>
<tr><td>1</td><td>0</td><td>Forward direction disabled.</td></tr>
<tr><td>1</td><td>1</td><td>Forward direction enabled.</td></tr>
</table></td></tr>
</table>

## 7.4.11 Special Command Register

Register Name: SCR
Register Description: Special Command
Access: R/W
8-Bit Hex Address: 2A
Default Value: 00

| *Bit 7* | *Bit 6* | *Bit 5* | *Bit 4* | *Bit 3* | *Bit 2* | *Bit 1* | *Bit 0* |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | TestMux | ClrPs | SetPs | EPIrq | RevRq |

This register allows 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.

| Bit | Description |
|---|---|
| 7:5 | These read-only bits are always ‘0’. |
| 4 | **TestMux:** When this bit is set, the state of the state machine is multiplexed onto the GPIO pins for debugging purposes.<br>GPIO is not possible when this bit is set. |
| 3:2 | **Clear Pause and Set Pause:** The Set and Clear Pause commands implement an error pause in Compatibility mode. Usually, errors are presented to the host parallel port by the peripheral during the active BUSY period of a data transfer. SetPs remains set until ClrPs is set, at which time both will clear.<br>In most cases, the slave host also sets RevRq at the same time when SetPs is set to:<br>1) lockup Compatibility mode with BUSY high, and<br>2) request a reverse transfer if the master requests that an additional status be sent in the reverse direction. |
| 1 | **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. |
| 0 | **Reverse Request:** This command requests that the host parallel port initiate the defined interface reversal handshake as defined by the IEEE Std 1284 specification. The command bit clears to indicate completion after the command has been executed on the interface. For Reverse Nibble and Reverse Byte modes, this occurs after negotiation is complete; in ECP mode, it occurs after the Reverse Request signal on the parallel port interface goes low.<br>In ECP mode, nPeriphRequest (nFault) is driven low to request that the host-side parallel port reverse the direction of the interface.<br>When this bit is set upon termination to Compatibility mode, the CD1283 can indicate that reverse data is available (through the nDataAv signal) immediately upon recognition of a Reverse-Nibble or Reverse-Byte negotiation. To obtain this behavior, this bit should be initialized to ‘1’, and set to ‘1’ upon termination to Compatibility mode. |

## 7.4.12 Short Pulse Register

| Register Name: SPR<br>Register Description: Short Pulse<br>Access: R/W | | | | | | 8-Bit Hex Address: 26<br>Default Value: 00 | |
|---|---|---|---|---|---|---|---|
| *Bit 7* | *Bit 6* | *Bit 5* | *Bit 4* | *Bit 3* | *Bit 2* | *Bit 1* | *Bit 0* |
| 8-Bit Binary Value | | | | | | | |

This register performs two functions:

1. SPR sets the duration of the short pulse used by the IEEE 1284 protocols for all modes other than Compatibility.
2. In Compatibility mode, SPR sets the duration of the ACK* pulse.

For Non-compatibility modes, SPR must be set to n – 2, where *n* is the number of CLKs in a 500-ns pulse. The peripheral host initializes SPR with the appropriate value to generate a 500-ns pulse width based on the operating frequency of the device.

In Compatibility mode, SPR 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. Table 11 shows examples of the necessary binary value for various system clock frequencies to set the 500-ns pulse width.

**Table 11. SPR Binary Values to Set 500-ns Pulse Widths**

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

### 7.4.13 Signal Status Register

Register Name: SSR — 8-Bit Hex Address: 2F
Register Description: Signal Status — Default Value: 00
Access: R/W

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | A1284 | nInit | HstBsy | HstClk |

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

### 7.4.14 Zeros Detect Register

Register Name: ZDR — 8-Bit Hex Address: 2C
Register Description: Zero Detect — Default Value: 00
Access: R/W

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | A1284 | nInit | HstBsy | HstClk |

When the bits 3:0 in ZDR are set, it enables the 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 ‘0’; these bits return ‘0’ when read. This register is enabled only during Manual mode.

## 7.5 Special Register

### 7.5.1 Reset Command Register

Register Name: RCR — 8-Bit Hex Address: 05
Register Description: Reset Command — Default Value: 00
Access: R/W

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

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

# 8.0 Electrical Specifications

***Caution:*** Before beginning any new design with this device, please contact Intel for the latest errata information. This datasheet is in reference to Revision E or newer devices.

## 8.1 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.

## 8.2 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

| | | |
|---|---|---|
| ESD<br>(Human body model) | 100 pF, 1.5 kΩ, ± 2 kV | Mil-Std-883D<br>Method 3015.7 |
| ESD<br>(Machine model) | 200 pF, 0 Ω, ± 200 V | EIAJ IC-121 |
| Latch-up | I/O ± 100 mA, $V_{CC}$ = 5 V<br>Temperature = 25°C and 70°C | JEDEC number 17 |
| | $V_{CC}$ ramp 5 V to 9 V<br>Temperature = 25°C and 70°C | JEDEC number 17 |
| Hysteresis | 200 mV | |

| 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 | [1] |
| $V_{OL}$ | Output low voltage | | 0.4 | V | $I_{OL}$ = 2.4 mA [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 | |

**NOTES:**
1. $V_{IH}$ is 2.7 V minimum on RESET* and CLK.
2. $V_{OL}$ for open-drain signals is 0.5 V @ 8 mA sinking because these signals can be wire-OR'ed in some systems and can have multiple pull-up resistors that increase the load on the output.

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: ±14 mA |
| Controlled voltage slew rate: 0.4 V/µs |
| Input hysteresis: 0.8 V |

***Note:*** While the CD1283 is a highly dependable device, there are a few guidelines to ensure that the maximum possible level of overall system reliability is achieved. First, design the PC board 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 CD1283 are strongly recommended. Noise typically occurs when either the CD1283 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 CD1283, Intel has specified the input-high voltage ($V_{IH}$) of the CLK and RESET* pins at 2.7 V, instead of the TTL-standard 2.0 V. This eliminates any sensitivity to ground bounce, even in extremely 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:

1. Use any of the available advanced-CMOS logic families (FACT, ACL, and so on). These CMOS output buffers will pull-up close to $V_{CC}$ when not heavily loaded. In addition, AS and ALS TTL can be used if the output of the TTL device is only driving one or two CMOS loads.
2. 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. Intel recommends the use of one of these two options for the CLK input to ensure fast, clean edges.
   Note that RESET* can, if desired, be pulled up passively with ≤ 1-kΩ resistor.

## 8.3 AC Characteristics

### 8.3.1 Asynchronous Timing

Refer to Figure 17 through Figure 25 for the reference numbers in Table 12.

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

**Table 12. Asynchronous Timing Reference Parameters** (Sheet 1 of 2)

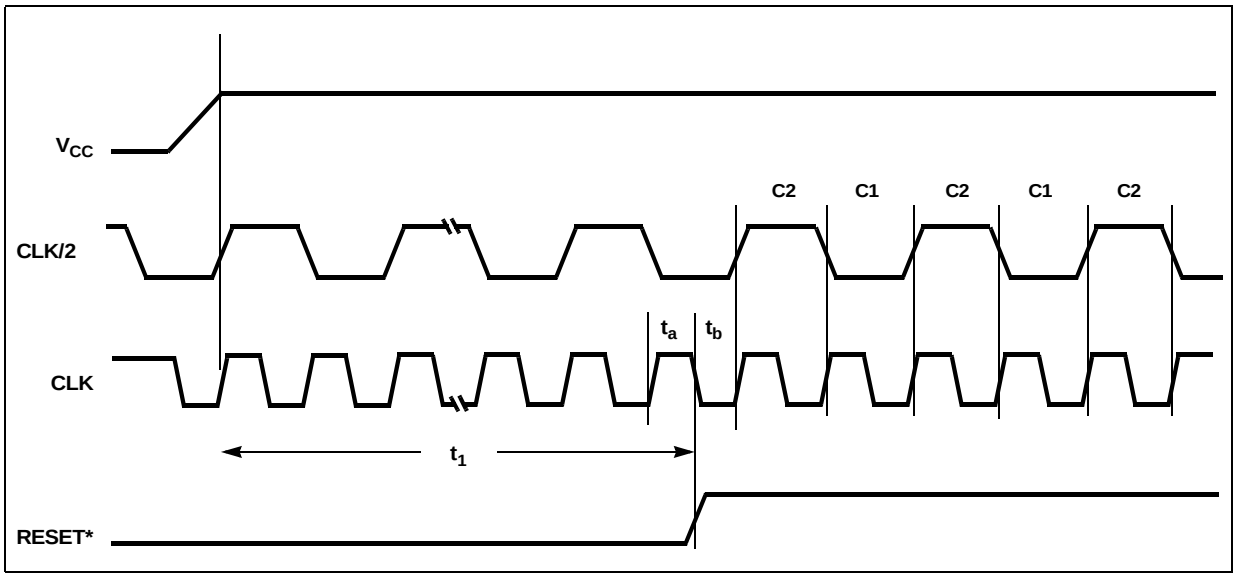
| Timing No. | Figure | Parameter | MIN | MAX | Unit |
|---|---|---|---|---|---|
| $t_1$ | 17 | RESET*[1] low pulse width | 10 | | $T_{CLK}$ |
| $t_2$ | 19 | Address setup time to CS* or DS* | 10 | | ns |
| $t_3$ | 19 | R/W* setup time to CS* or DS* | 10 | | ns |
| $t_4$ | 19 | Address hold time after CS* | 0 | | ns |
| $t_5$ | 19 | R/W* hold time after CS* | 0 | | ns |
| $t_6$ | 19 | DTACK* low to read data valid | | 10 | ns |
| $t_7$ | 19 | DTACK* low from CS* or DS [2] | 2 $T_{CLK}$ | 4 $T_{CLK}$ + 30 | ns |
| $t_8$ | 19 | Data bus tristate after CS* or DS* high | 0 | 30 | ns |
| $t_9$ | 19 | CS* or DGRANT* high from DTACK* low | 0 | | ns |
| $t_{10}$ | 19 | DTACK* inactive from CS* or DGRANT* and DS* high | | 40 | ns |
| $t_{11}$ | 19 | DS* high pulse width | 10 | | ns |
| $t_{12}$ | 20 | Write data valid from CS* and DS* low | | 1$T_{CLK}$ | ns |
| $t_{13}$ | 20 | Write data hold time after DS* high | 0 | | ns |
| $t_{14}$ | 18 | Clock period ($T_{CLK}$) [1, 3] | 40.0 | 1000 | ns |
| $t_{15}$ | 18 | Clock low time [1] | 0.3 $T_{CLK}$ | 0.7 $T_{CLK}$ | ns |
| $t_{16}$ | 18 | Clock high time [1] | 0.3 $T_{CLK}$ | 0.7 $T_{CLK}$ | ns |
| $t_{17}$ | 21 | Propagation delay, DGRANT* and DS* to DPASS* | | 35 | ns |
| $t_{18}$ | 21 | Setup time, SVCACK* to DS* and DGRANT* | 10 | | ns |
| $t_{19}$ | 22 | Setup time, DMAACK* to rising edge of CLK | 10 | | ns |
| $t_{20}$ | 22 | Hold time, read data after rising edge of CLK | 10 | 30 | ns |
| $t_{21}$ | 24 | Setup time, write data to rising edge of CLK | 0 | | ns |
| $t_{22}$ | 19 | DTACK* active pull-up time[4] | | | ns |
| $t_{23}$ | 22 | Data valid after falling edge of CLK (DMA read) | | 25 | ns |
| $t_{24}$ | 22<br>24 | Hold time, DMAREQ* after DMAACK* falling edge, last DMA cycle | 10 | 1 CLK + 15 | ns |

## Table 12. Asynchronous Timing Reference Parameters (Sheet 2 of 2)

| Timing No. | Figure | Parameter | MIN | MAX | Unit |
|---|---|---|---|---|---|
| The following timing numbers are for the back-to-back asynchronous DMA timing diagrams. | | | | | |
| $t_{25}$ | 23 | Hold time, DMAACK* active (DMA read/write) | 3 CLK | | |
| $t_{26}$ | 23 | Delay, data valid after falling edge DMAACK* (DMA read) | 0.5 CLK + 20 | 1.5 CLK + 25 | ns |
| $t_{27}$ | 23 | Hold time, data valid after rising edge DMAACK* (DMA read) | 10 | 30 | ns |
| $t_{28}$ | 23<br>25 | Inactive time, DMAACK* (DMA read/write) | 10 | | ns |
| $t_{29}$ | 23<br>25 | Hold time, DMAREQ* rising edge after DMAACK* falling edge (DMA read/write) | 10 | 1 CLK + 15 | ns |
| $t_{30}$ | 25 | Hold time, DMAACK* active (DMA write) | 2.5 CLK | | |
| $t_{31}$ | 25 | Delay, data valid after falling edge DMAACK* (DMA write) | | 1.5 CLK | |

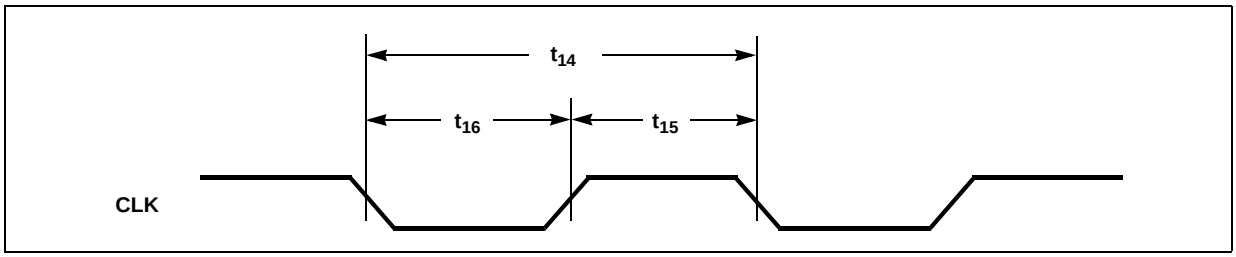
**NOTES:**
1. Timing numbers for RESET* and CLK are valid for both asynchronous and synchronous specifications. The device will operate on any clock with a 40–60 or better duty cycle.
2. On host-I/O cycles, immediately following SVCACK* cycles and writes to EOSRR, DTACK* will be delayed by 20 CLKs (1 ms @ 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 CD1283 is not accessed until after this time period.
3. As TCLK increases, device performance decreases. A minimum clock frequency of 25 MHz is required to guarantee specified performance. 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 17. 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 18. Clock Timing




### Figure 19. Asynchronous Read Cycle Timing

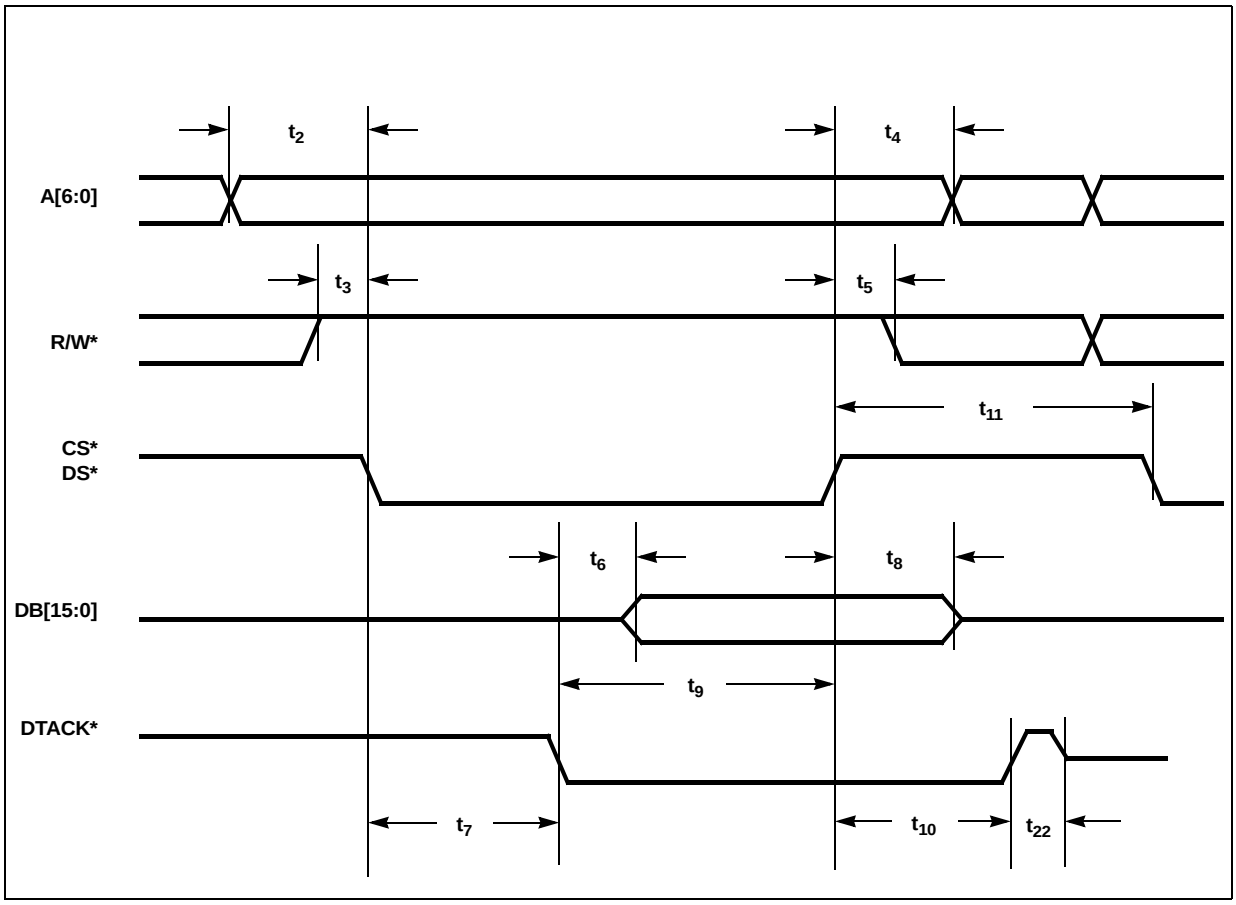


**Figure 20. Asynchronous Write Cycle Timing**

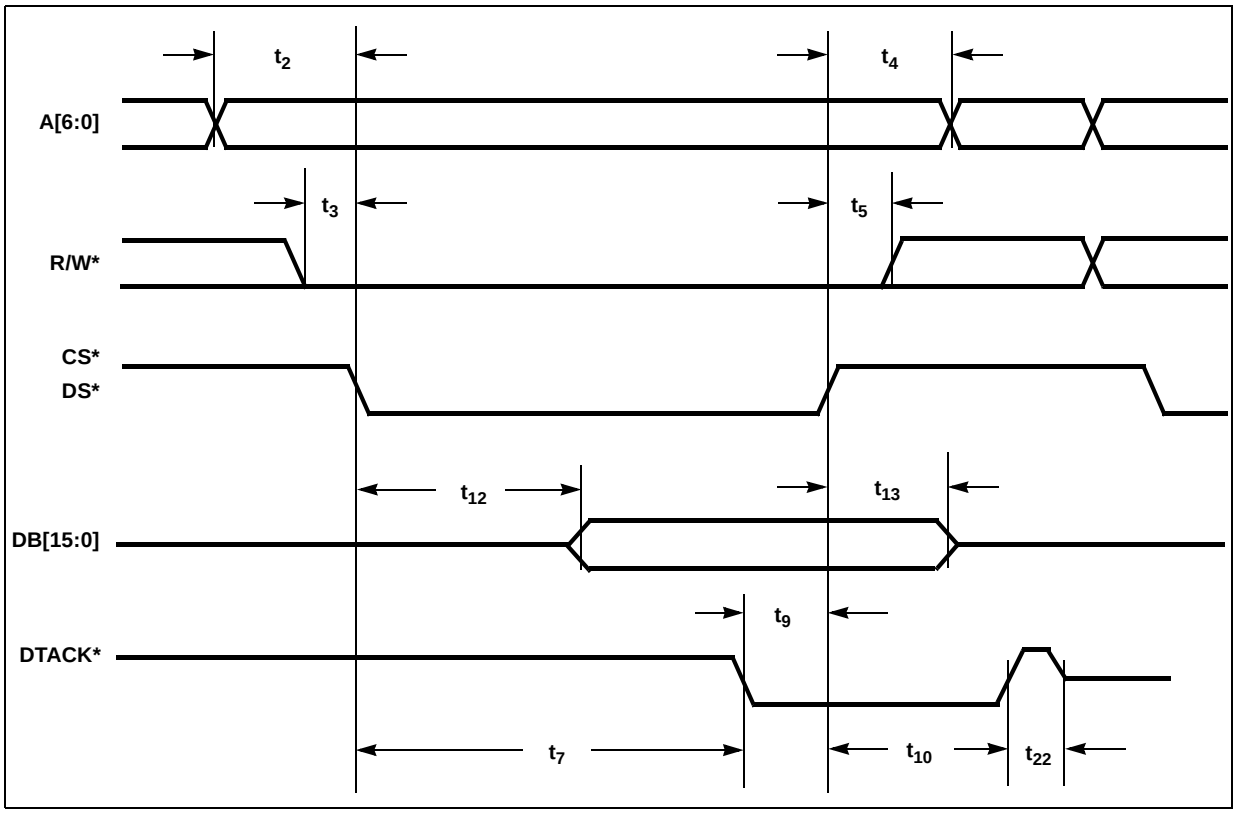
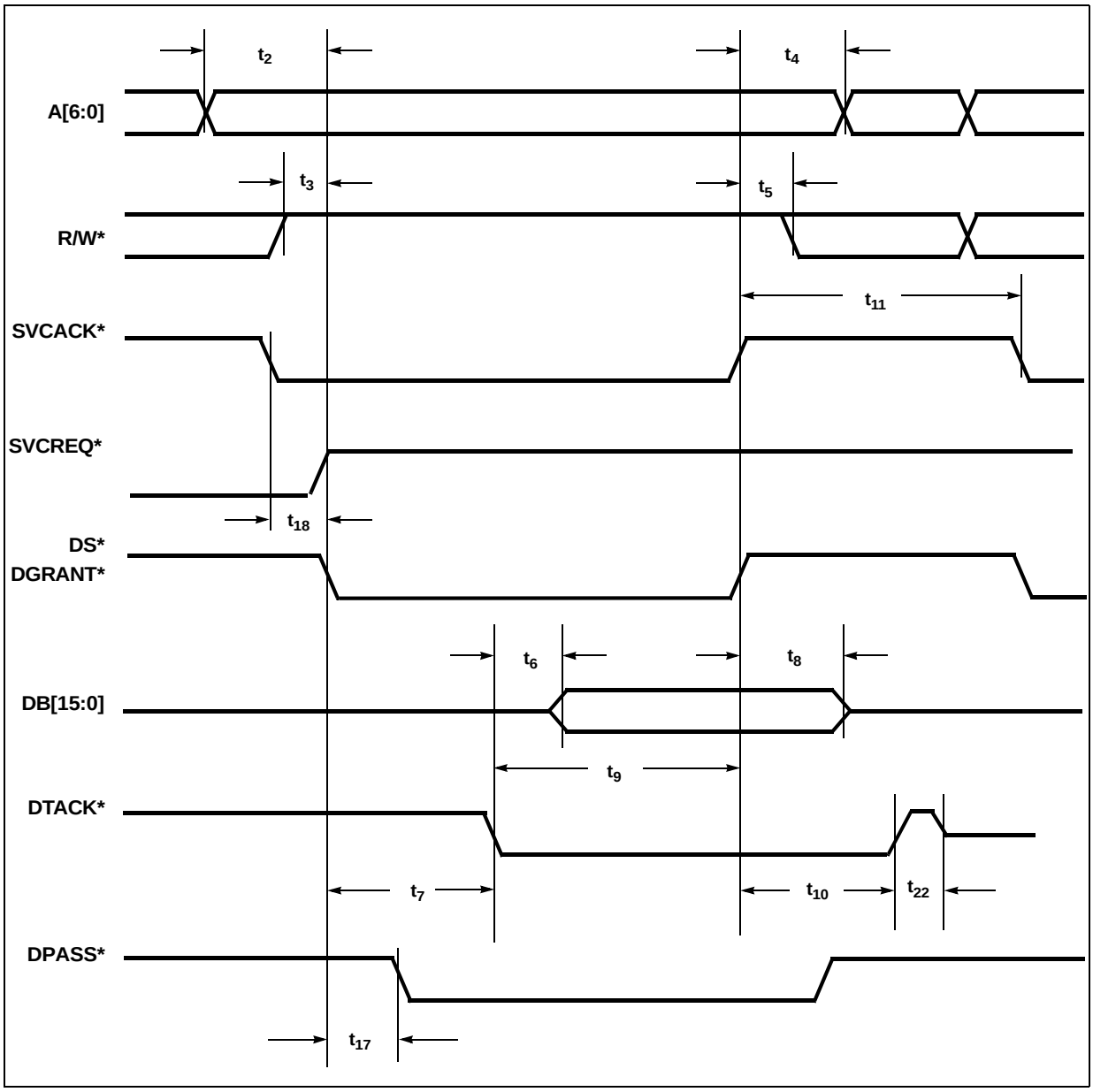
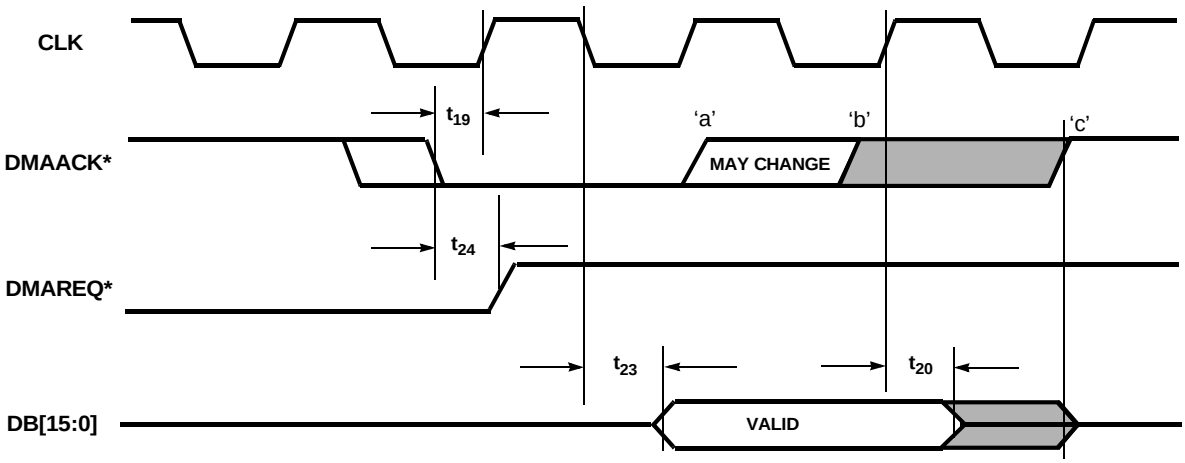


**Figure 21. Asynchronous Service Acknowledge Cycle Timing**

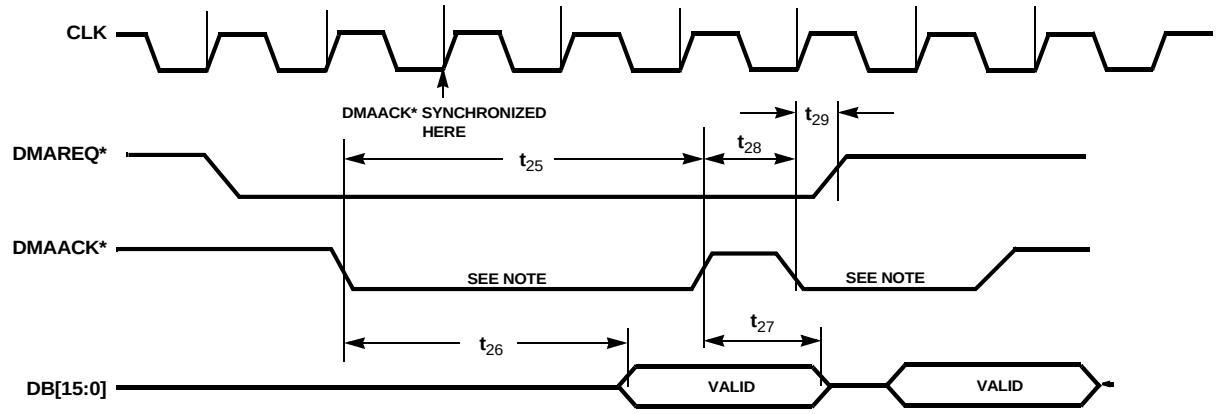


## Figure 22. Asynchronous DMA Read Cycle Timing



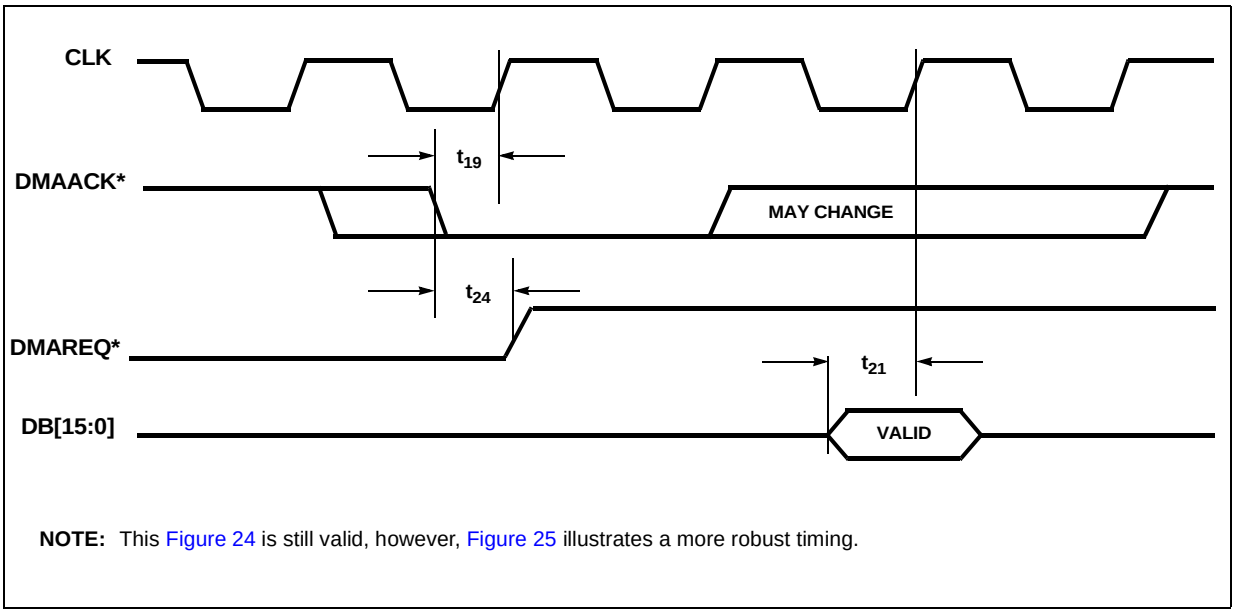

**NOTES:**
1. The DMA handshake operates in asynchronous mode only if the AsyncDMA bit is set in PACR.
2. If DMAACK* is released after point 'a,' but before point 'b' (two rising CLK edges after the falling edge of DMAACK*), DB[15:0] is released at $t_{20}$ following the rising edge of CLK. If DMAACK* is held past this edge, it controls the release of DB[15:0]; the data bus remains active until DMAACK* becomes inactive (point 'c').
3. This Figure 22 is still valid, however, Figure 23 illustrates a more robust timing.

## Figure 23. Asynchronous DMA Read Cycle Timing (Two Back-to-Back DMA Reads)



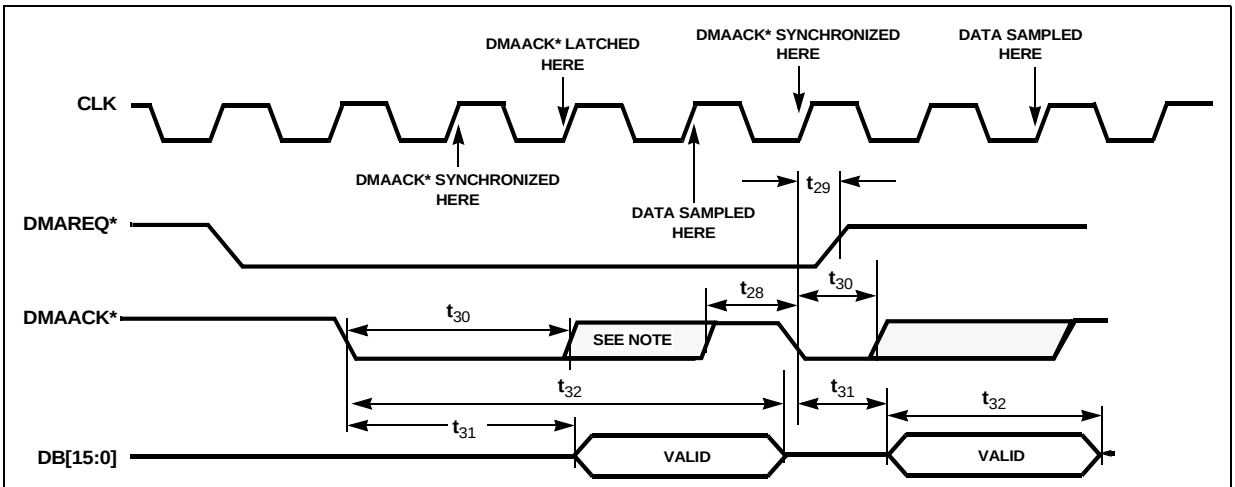

**NOTE:** The falling edge of DMAACK* is synchronized internally with the rising edge of the clock when asynchronous timing is selected by PACR[1]. The data valid time can vary by as much as one full CLK cycle depending on when DMAACK* falling edge occurs in relation to the CLK rising edge. The minimum DMAACK* active time must be met to ensure that the data has become valid before the rising edge of DMAACK*. The DMAACK* can be extended to any length, which extends the data valid hold time accordingly. If $t_{25}$ is not met and DMAACK* is deasserted in less than $t_{25}$ (MIN), then the data bus tristates $t_{27}$ after the third rising clock edge following the assertion of DMAACK*.

**Figure 24. Asynchronous DMA Write Cycle Timing**




**NOTE:** This Figure 24 is still valid, however, Figure 25 illustrates a more robust timing.

**Figure 25. Asynchronous DMA Write Cycle Timing**




**NOTE:** The data is sampled on the third rising edge of CLK following the assertion of DMAACK*. If DMAACK* is held active for more than three CLK cycles then the next DMA write cycle will simply be delayed, but the data will still be sampled on the third rising CLK edge following the assertion of DMAACK*. If DMAACK* is active for < 3 CLKs, the n the data is still sampled on the third rising CLK edge following the assertion of DMAACK* (provided that DMAACK* is active long enough for the device to lastch it. Due to this somewhat synchronous behavior, care must be taken to guarantee that the data is valid at this CLK edge. Do not assume that the data will be sampled on the deassertion of DMAACK*.

## 8.3.2 Synchronous Timing

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

### Table 13. Synchronous Timing Reference Parameters

| Timing Number | Figure | Parameter | MIN | MAX | Unit |
|---|---|---|---|---|---|
| $t_1$ | 26 | Setup time, CS* and DS* to C1 rising edge | 15 | | ns |
| $t_2$ | 26 | Setup time, R/W* to C1 rising edge | 15 | | ns |
| $t_3$ | 26 | Setup time, address valid to C1 rising edge | 20 | | ns |
| $t_4$ | 26 | C2 rising edge to data valid | | 60 | ns |
| $t_5$ | 26 | DTACK* low from C3 rising edge[1] | | 30 | ns |
| $t_6$ | 26 | CS* and DS* trailing edge to data bus high-impedance | | 30 | ns |
| $t_7$ | 26 | CS* and DS* inactive between host accesses | 10 | | ns |
| $t_8$ | 26 | Hold time, R/W* after C3 rising edge | 20 | | ns |
| $t_9$ | 26 | Hold time, address valid after C3 rising edge | 0 | | ns |
| $t_{10}$ | 27 | Setup time, write data valid to C2 rising edge | 0 | | ns |
| $t_{11}$ | 28 | Setup time, DS* and DGRANT* to C1 rising edge | 30 | | ns |
| $t_{12}$ | 28 | Setup time, SVCACK* to DS* and DGRANT* | 10 | | ns |
| $t_{13}$ | 27 | Hold time, write data valid after C3 rising edge | 0 | | ns |
| $t_{14}$ | 29 | Propagation delay, DS* and DGRANT* to DPASS* | | 35 | ns |
| $t_{15}$ | 29<br>30 | Falling edge DMAREQ* after rising edge CLK (DMA write/read) | | 25 | ns |
| $t_{16}$ | 29<br>30 | Hold time, rising edge DMAREQ* after falling edge DMAACK* (DMA write/read) | | 20 | ns |
| $t_{17}$ | 29 | Setup time, data valid before rising edge C3 (DMA write) | 5 | | ns |
| $t_{18}$ | 29<br>30 | Setup time, falling edge DMAACK* to falling edge C1 (DMA write/read) | 10 | | ns |
| $t_{21}$ | 26 | DTACK* active pull-up time[2] | | | |
| $t_{22}$ | 30 | Hold time, data valid after rising edge C3 (DMA write) | 5 | | |
| $t_{23}$ | 30 | Hold time, data valid after rising edge C1 (DMA read) | 10 | 30 | |
| $t_{24}$ | 30 | Data valid after falling edge C1 (DMA read) | | 25 | |
| $t_{25}$ | 30 | Inactive time, DMAACK* (DMA read) | 10 | | |

**NOTES:**
1. On host I/O cycles immediately following SVCACK* cycles and writes to the EOSRR, DTACK* will be delayed by 20 CLKs (1 ms @ 20 MHz, 800 ns @ 25 MHz). On systems that do not use DTACK* to signal the end of the I/O cycle, use wait states or some other form of delay generation to assure that the CD1283 is not 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 26. Synchronous Read Cycle Timing**

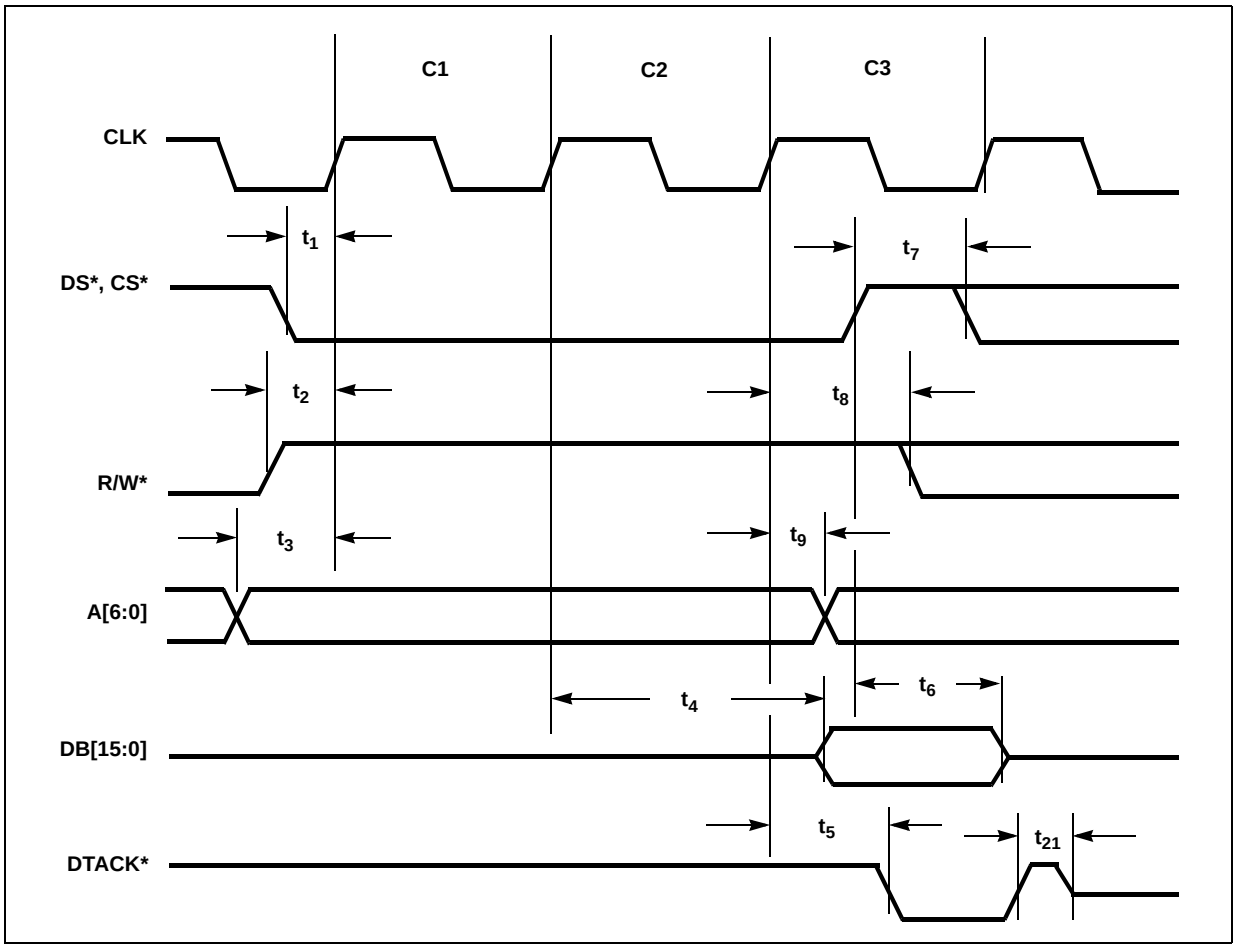


**Figure 27. Synchronous Write Cycle Timing**

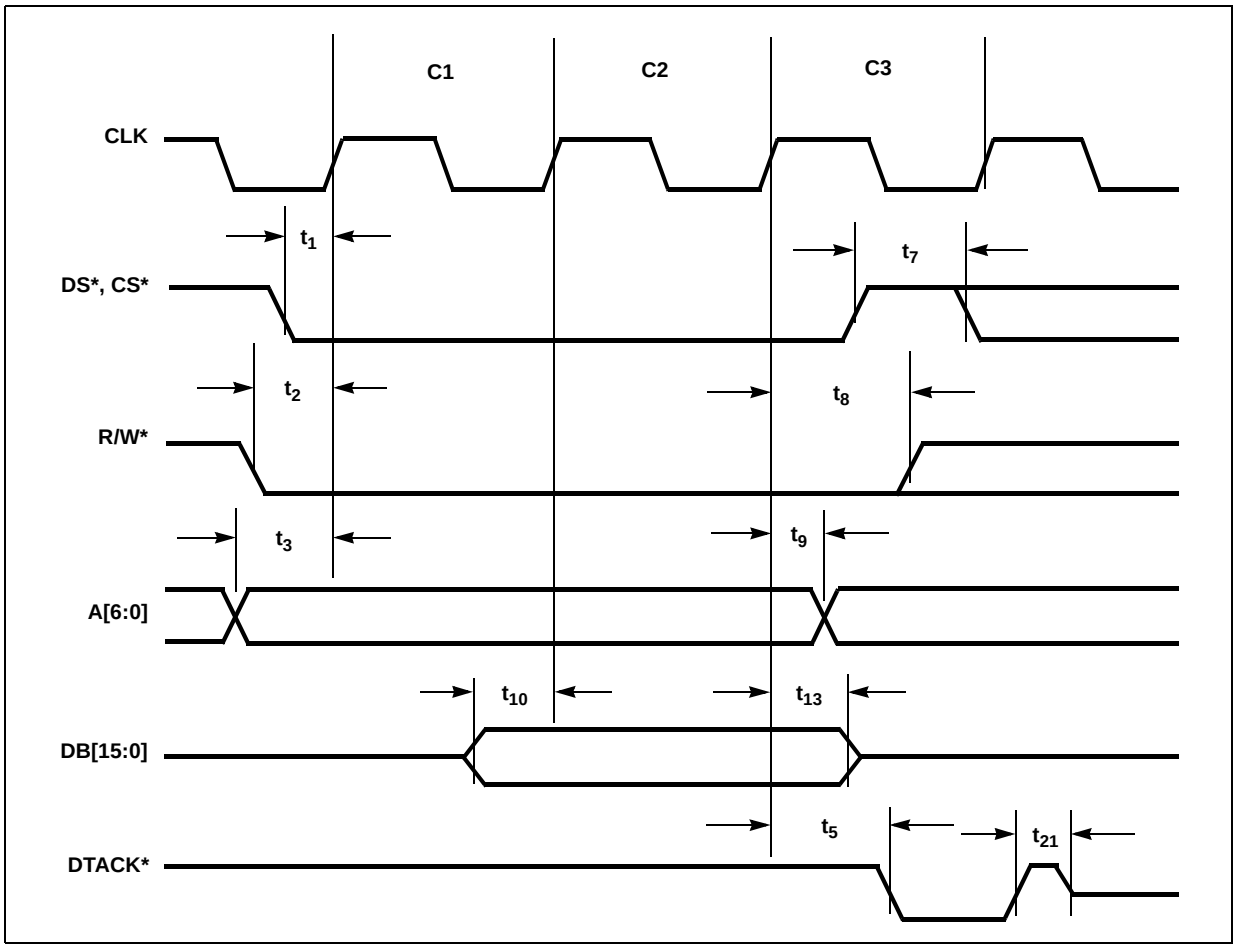


**Figure 28. Synchronous Service Acknowledge Cycle Timing**

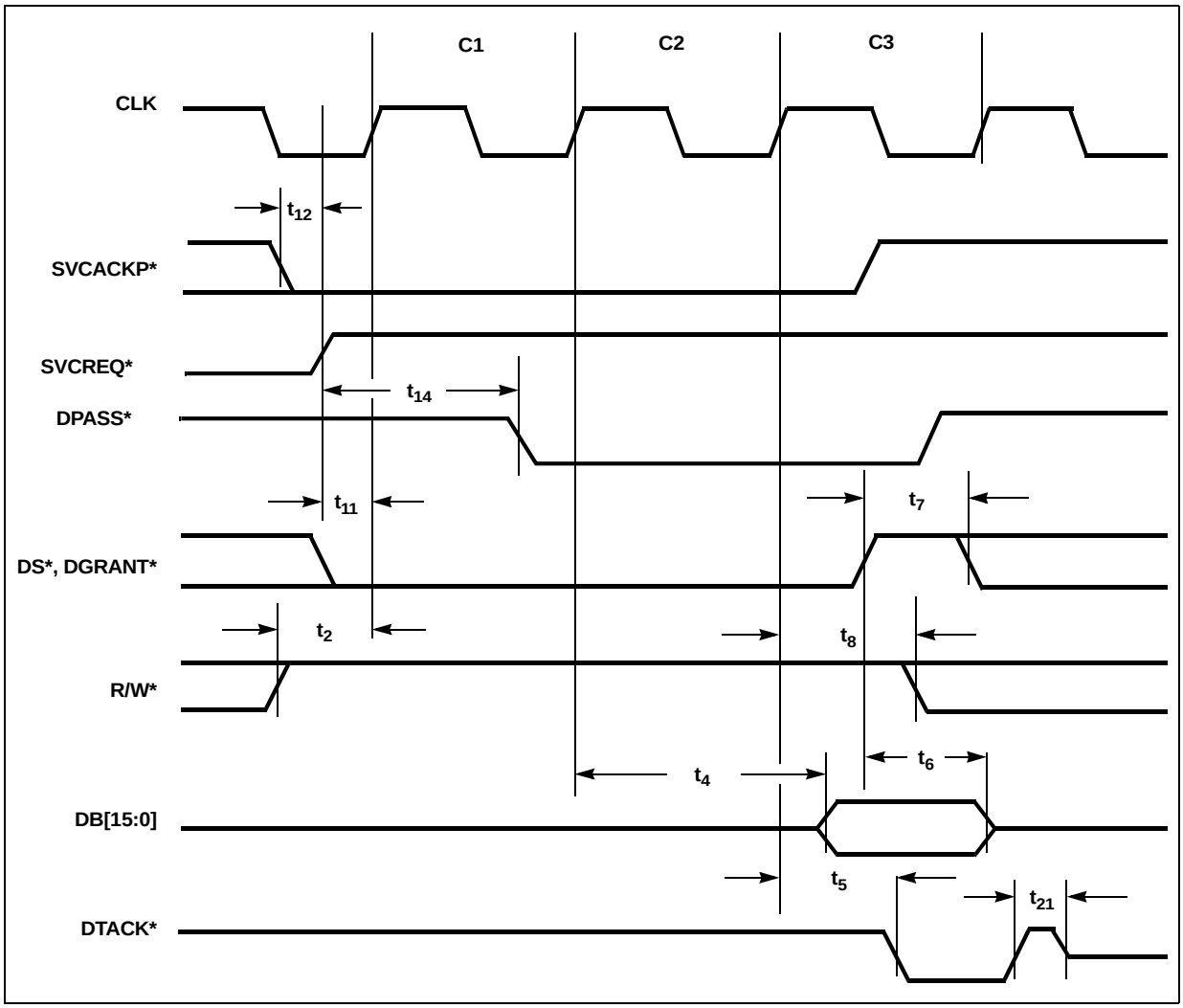
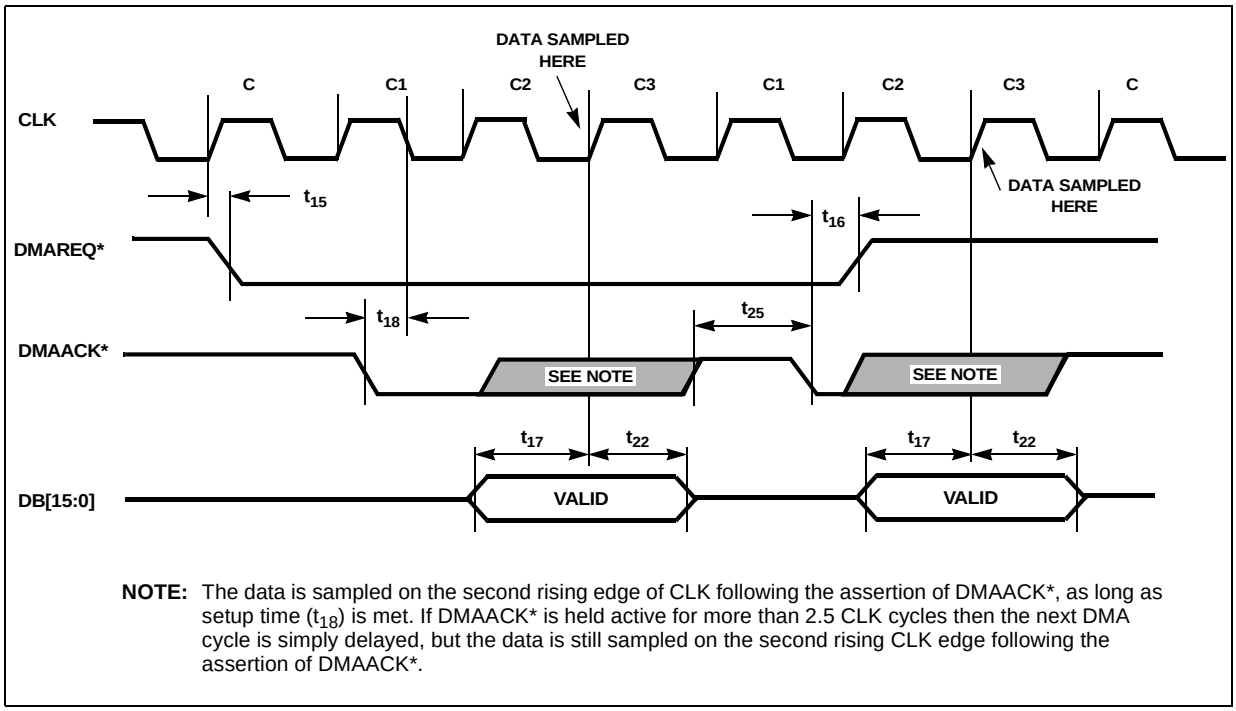
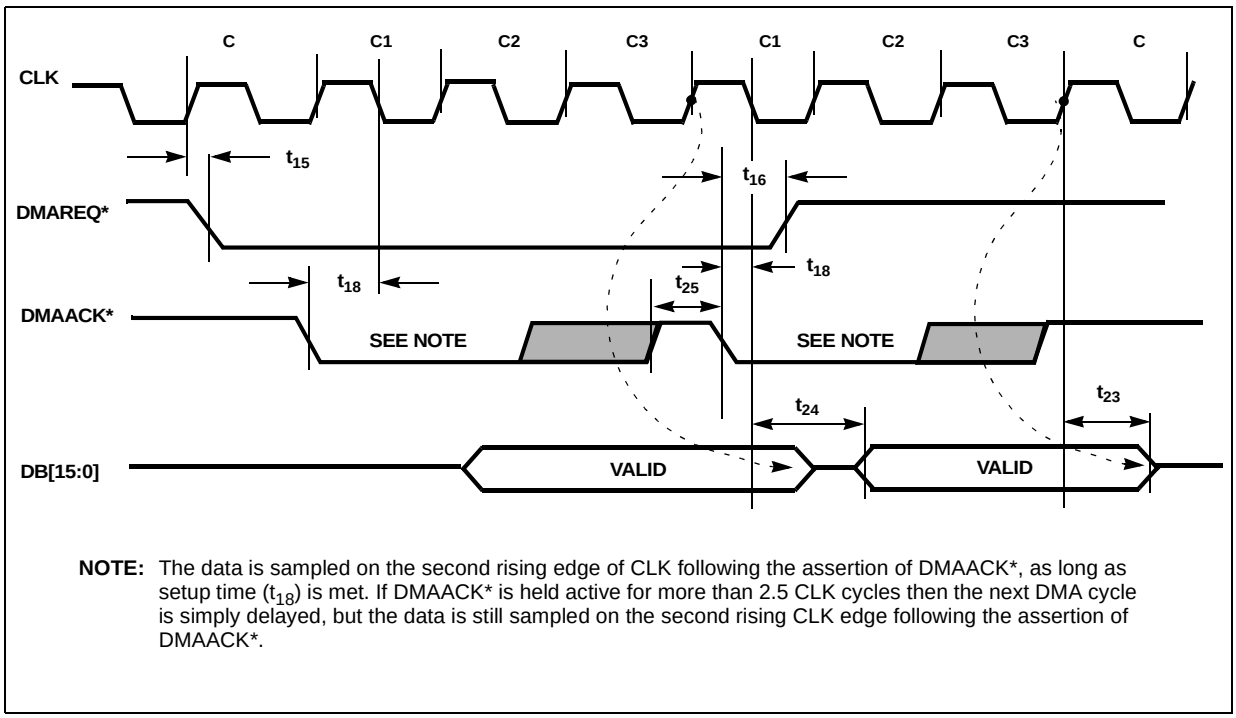


**Figure 29. Synchronous DMA Write Cycle Timing (Two Back-to-Back 3-Cycle DMA Writes)**



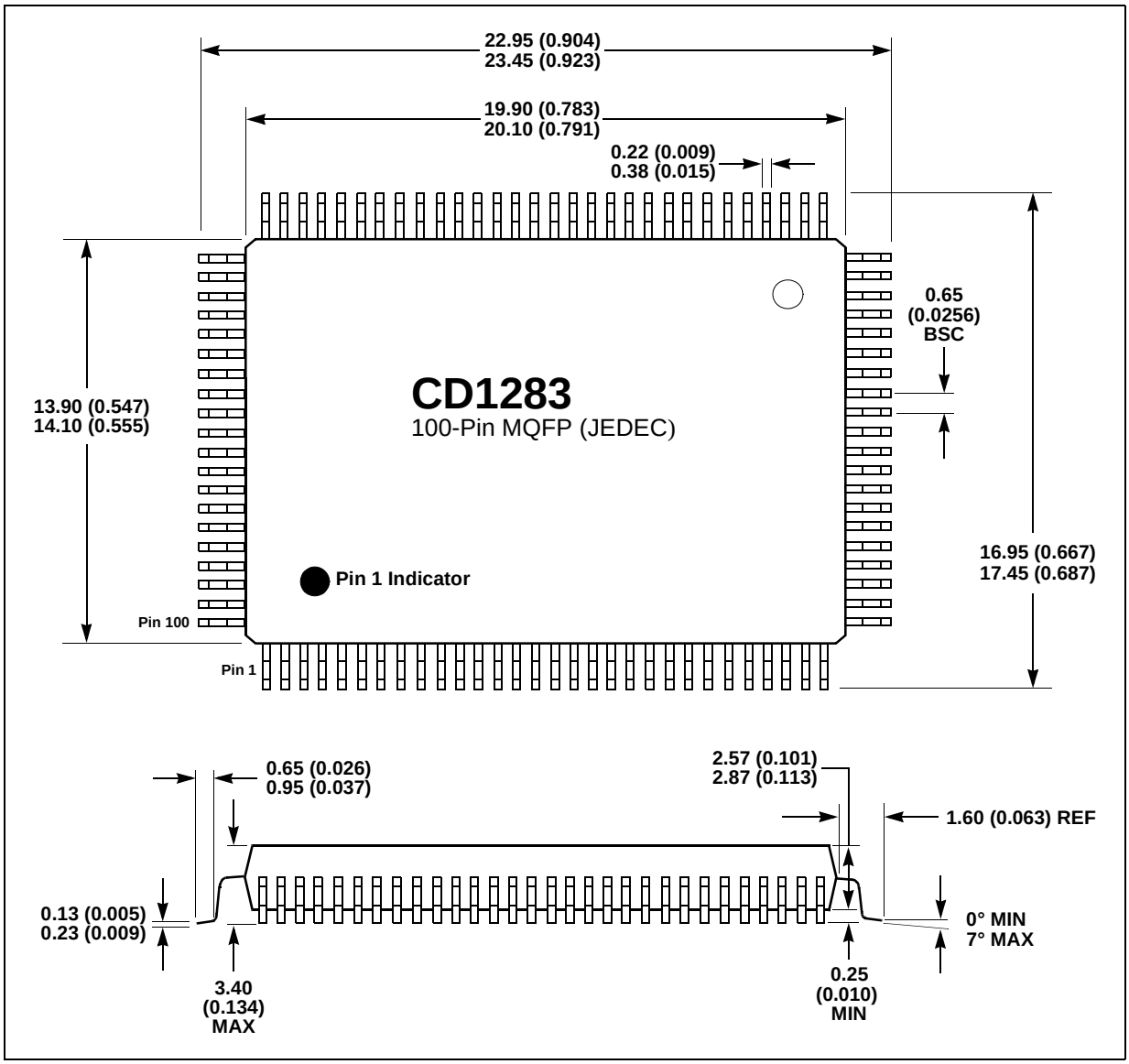

**NOTE:** The data is sampled on the second rising edge of CLK following the assertion of DMAACK*, as long as setup time ($t_{18}$) is met. If DMAACK* is held active for more than 2.5 CLK cycles then the next DMA cycle is simply delayed, but the data is still sampled on the second rising CLK edge following the assertion of DMAACK*.

**Figure 30. Synchronous DMA Read Cycle Timing (Two Back-to-Back 3-Cycle DMA Reads)**




**NOTE:** The data is sampled on the second rising edge of CLK following the assertion of DMAACK*, as long as setup time ($t_{18}$) is met. If DMAACK* is held active for more than 2.5 CLK cycles then the next DMA cycle is simply delayed, but the data is still sampled on the second rising CLK edge following the assertion of DMAACK*.

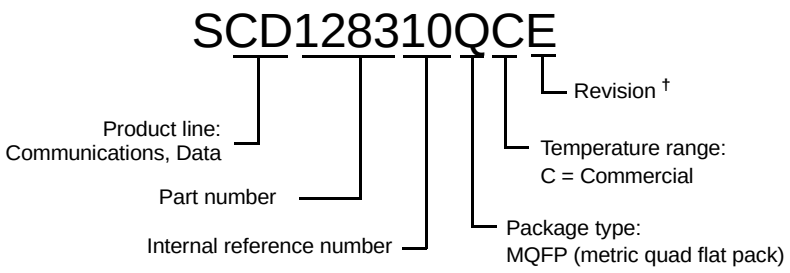
# 9.0 Package Dimensions




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

# 10.0 Ordering Information

The order number for the CD1283 is:




† Contact Intel Corporation for up-to-date information on revisions.

# Index


## A

abbreviations 11
absolute maximum ratings 76
AC characteristics 78
acronyms 11
ASCII code tables
    decimal 52
    hexadecimal 51

## B

bus interface 33
BYTESWAP 34

## C

cable connection 43
Compatibility mode status 40
context 32
context switch
    hardware-activated 32
control signal generation 31
control signals 38
conventions 11
CPU interface 22, 44

## D

daisy-chaining 26
data pipeline 32
data transfers 40
decimal code tables 52
definitions
    host 36
    master 36
    peripheral 36
    slave 36
device architecture 22
device reset 47

## E

ECP mode 34
electrical specifications 76
endian format 25
EPP mode 38
external buffer control 44

## F

failed negotiation 39
FIFO data path
    receive 37
    transmit 38
functional block diagram 23
functional description 22

## G

general-purpose I/O port 42

## H

hardware configurations
    interfacing to a Motorola microprocessor-based system 46
    interfacing to an Intel microprocessor-based system 45
hardware-activated context switch. *See* context switch 32
hexadecimal code tables 51

## I

ID request 41
IEEE Standards Department 33
IEEE STD 1284 33, 36
IEEE STD 1284 protocol negotiations 39
initialization, CD1283 47
interface 34
    parallel port 39

internal address generation 23
interrupts 25, 37
    DirCh 31
    EPPAW 28
    IDReq 31
    NegCh 28
invalid termination 39
IVR 38

## M

modes
    ECP 34
    EPP 38
    Manual 38
    Reverse Byte 40
    Reverse Nibble 40

## O

odd-byte transfers 25
ODR 38
operating conditions 76
ordering information 91

## P

package dimensions 90
parallel port 34, 36
    configuration 37
    FIFO 32, 34
    overview 36
    service requests 27
    signal names 37
    state machine 37
    terminology 36
parallel protocol support 40
pin information
    pin diagram 13
    pin list 14
protocol negotiations 39
protocol timing 41

## R

read cycles 24
receive direction 34
receiving compressed data 35
register map 19
register summary 19
registers
    Channel — Parallel
        HTVR 20
    Global
        AER 53
        CAR 19
        GFRCR 19, 53
        GPDIR 19, 54
        GPIO 19, 54
        PIR 19, 54
        PPR 19, 55
        SVRR 19, 55
    Parallel Pipeline
        DER 20
        DMABUF 20, 58
        HRSR 20, 58
        LIVR 20, 60
        PACR 20, 61
        PCRR 20, 61
        PFCR 20, 61
        PFEP 20, 63
        PFFP 20, 63
        PFHR1 20, 63
        PFHR2 20, 63
        PFQR 20, 64
        PFSR 20, 64
        PFTR 20, 65
        RLCR 20, 66
        SDTCR 20, 66
        SDTPR 20, 67
    Parallel Port
        EAR 20
        IVR 20, 67
        MDR 20, 68
        NER 20, 68
        NSR 20, 39, 69
        ODR 21, 70

OVR 21, 70
PCIER 21, 71
PCISR 21, 71
PCR 21, 71
SCR 21, 39, 72
SPR 21, 73
SSR 21, 74
ZDR 21, 74
Virtual
EOSRR 20
PIVR 20, 56
reset, device 47
RLE (run-length-encoding) 35

## S

sample system block diagram 44
signal names 37
SSR 38
stale data timer 35
state machine 37
SVCREQP* 56
synchronous timing reference parameters 85

## T

timing
asynchronous 78
clock 80
DMA read cycle 83
read cycle 80
reset 79
service acknowledge cycle 82
write cycle 81
synchronous 84
read cycle 86
service acknowledge cycle 88
write cycle 87
transmit direction 36

## U

units of measure used 11

## W

write cycles 24

## Z

ZDR 38