PlayCD™

CL480 MPEG SYSTEM DECODER USER'S MANUAL

C-Cube
Microsystems

PlayCD™

CL480 MPEG System Decoder User's Manual

C-Cube
Microsystems

C-Cube Microsystems reserves the right to change any products described herein at any time and without notice. C-Cube Microsystems assumes no responsibility or liability arising from the use of the products described herein, except as expressly agreed to in writing by C-Cube Microsystems. The use and purchase of this product does not convey a licence under any patent rights, copyrights, trademark rights, or any other intellectual property rights of C-Cube Microsystems.

Trademark Acknowledgment:

PlayCD is a trademark of C-Cube Microsystems.

C-Cube and the corporate logo are registered trademarks of C-Cube Microsystems.

C-Cube Microsystems
1778 McCarthy Boulevard
Milpitas, CA 95035
Telephone (408) 944-6300
Fax (408) 944-6314

Customer Comments and Feedback:

If you have any comments about this document, send them to the C-Cube Technical Publications Department at the address listed above, or send e-mail to:

techpubs@c-cube.com

C-Cube Part # 92-0480-101

Preface

This manual is the primary source of technical information for the operation and programming of the C-Cube CL480 MPEG System Decoder.

Audience

This manual is intended for:

- □ System designers and managers who are evaluating the CL480 for possible use in a system.
- □ Design, software and system engineers developing a decoding system using the CL480 for whom a comprehensive programming background as well as a detailed understanding of MPEG compression is assumed.

Related Publications

In particular, readers should understand the MPEG standard: *Coded Representation of Picture, Audio and Multimedia/Hypermedia Information*, ISO 11172. This document is available from the American National Strandards Institute (ANSI), 11 West 42nd St., New York, NY 10036, (212) 642-4900.

Organization

This manual is divided into three main sections:

- □ Section I, General Information, presents an introduction to the CL480 and the MPEG standard that it implements.
- □ Section II, Hardware, describes signals, operational information for the external interfaces of the CL480, registers, and detailed electrical and mechanical specifications.

□ Section III, Software, describes the CL480's microcode, and includes an alphabetical listing of all macro commands available to the programmer. Note: the microcode information in this manual applies to CL480 version 2.0.

Conventions

Please note the following notation examples and conventions used in this manual:

Notation Examples	Explanation
0x1c3	"0x" prefix indicates a hexadecimal number.
11011_2	"2"subscript indicates a binary number.
IMEM	Four-letter mnemonics indicate on-chip memories, starting with a letter to indicate function of memory, and ending with "MEM." For example, IMEM is the CL480's instruction memory.
CPU_cntl CD_cntl	This format indicates a register name. The first part of a register name is a group specifier, given in all upper-case. The second part (separated from the first by an underscore and given in all lower-case) is a register specifier, indicating the function performed by the register within the group. In the examples shown, CPU_cntl and CD_cntl are separate registers, even though both have the same register specifier, "cntl."
VIE, Res	Italicized acronyms or abbreviations (initial or all caps) indicate bit field names within registers or data words.
IMEM[3] CPU_cntl[0] IMEM[0][15] IMEM[3][2:0]	Square bracket notation similar to C language array subscripting indicates words within memories, and bits within words and registers. IMEM[3], for example, is the 16-bit word at address 3 within IMEM, while CPU_cntl[0] is bit 0 of the CPU_cntl register. Similarly, IMEM[0][15] is the most significant bit of the word at IMEM address 0.
	Ranges of bits are indicated by numbers separated by a colon such as the three bits HMEM[3][2:0]. Ranges of words within a memory are indicated by numbers separated by a dash such as the eight words HMEM[0-7].
RESERVED or *Res*	Indicates bit fields within registers which are not defined. RESERVED bit fields may return any value when read and must be written with 0 (or 1 if so specified). Writing the incorrect value to a RESERVED CL480 register bit will cause indeterminate behavior. In addition, all CL480 registers which are not explicitly given names are also RESERVED, and accessing these registers may cause indeterminate results on current or future CL480 implementations.
leftBorder	Bold-face type represents macro command arguments.
`return();`	C-style syntax presented in courier typeface represents program pseudocode and equations.
`vbv_delay`	Names presented in courier represent names of items within the MPEG bitstream taken from the MPEG standard.

Contents

Section I. General Information

Section II. Hardware

8 Audio Interface

9 Electrical and Physical Specifications

10 Registers

Section III. Software

11 Microcode Overview

12 Macro Commands

Figures

Tables

Chapter 1
Introduction

1.1 PlayCD Family

The CL480 is an MPEG-1 audio/video decoder and the first member of C-Cube's PlayCD™ family. Designed for consumer electronics products and multimedia PCs, the CL480 reduces system cost by requiring only 4Mbits of DRAM, providing CD-ROM decoding, and providing a serial CD interface. The CL480 reduces development time and effort by performing system stream processing and audio/video synchronization. The CL480 provides high-quality output through its advanced video post-processing, sophisticated error concealment and support for high-resolution still pictures. The chip is particularly well suited for video CD players because it provides glueless interfaces to the CD-DSP, audio DAC and NTSC/PAL encoder.

The CL480 processes:

- CD-ROM data, CD-DA data, and constrained-parameters MPEG-1 video bitstreams (typically SIF-resolution) in real time.
- MPEG audio (Layer I or II) bitstreams.

The CL480 also features an extensive microcode set supplied with the hardware.

1.2 CL480 Features

The CL480 has these key features:

- Integrates on one chip: MPEG audio and video decoding, video timing generation, and CD-ROM decoding functions
- Fully supports Video CD 2.0 and 1.1, Karaoke CD 1.0 and CD-I Green Book[1]
- Accepts MPEG-1 system streams or CD data streams with no external parsing required
- Decodes and synchronizes SIF-resolution MPEG video and two channels of MPEG Layer 1 or 2 audio
  - Resolutions include 352x240 at 30 Hz, 352x288 at 25 Hz, and 384x240 at 24 Hz
  - Audio sample rates include 32, 44.1 and 48 kHz
- Decodes still images with coded resolutions up to 704x576 while decoding MPEG audio
- Requires only 4 Mbits of DRAM, even for PAL systems
- Accepts compressed data from 4-pin CD interface or from 8-bit host interface
- Glueless interfaces to CD-DSP, DRAM, ROM, audio DACs, and NTSC/PAL encoder

1.2.1 Flexible Video Interface with High-Quality Video Output

- Interpolates two different fields from each decoded frame to reduce flicker and improve image quality on interlaced displays
- Performs frame-rate conversion so that display rate (typically 50 or 60 Hz) is independent of coded frame rate (typically 24, 25 or 30 Hz)
- Performs horizontal interpolation of decoded video using a seven-tap filter
- Provides RGB or YCbCr video output using on-chip color space converter
- Supports $\overline{\text{HSYNC}}$ and $\overline{\text{VSYNC}}$ as inputs or outputs

[1] The CL480 can implement CD-I but does not have all of the required microcode. To work in the current type of CD-I systems, the CL480 requires a glue chip between its host interface and a 68000.

- □ Supports VCK input or generates VCK output by dividing GCK (40.5 MHz) by 3/2
- □ Optionally generates NTSC and PAL composite sync

1.2.2 Low Voltage, Low Power Operation in Small Package

- □ Operates with a supply voltage of 2.7 to 3.6 volts
- □ Can accept 5-volt inputs
- □ Consumes less than 1 watt when decoding audio and video
- □ Automatically divides GCK by 4, 8, or 16 to reduce power when decoding MPEG audio only or when in CD-DA pass-through mode
- □ Packaged in a 128-pin small-outline PQFP (18mm x 18mm body)

1.2.3 Powerful, Easy-to-Use Microcode

- □ Performs audio/video synchronization
- □ Provides high-quality error concealment for bitstream errors
- □ Performs error correction of CD-ROM data at 2.8 Mbps when not decoding MPEG
- □ Supports low-cost systems with no host processor
- □ Can initialize itself from ROM connected to the DRAM interface
- □ Provides high-level macro commands, allowing the host to monitor and control the input, decode and output processes:
  - □ Play - Decodes and displays at normal rate
  - □ SlowMotion - Decodes and displays at slower rate
  - □ SingleStep - Decodes and displays next picture
  - □ Scan - Decodes and displays next I-picture
  - □ DisplayStill - Decodes and displays high-resolution still picture with audio
  - □ Pause - Freezes display and decoding process
  - □ Freeze - Freezes display but continues decoding
  - □ DumpData - Performs error correction on CD-ROM data
  - □ SetStreams - Selects which streams to decode
  - □ SetWindow - Sets video window size and location
  - □ SetBorderColor - Sets border color
  - □ SetErrorLevel - Sets error level for DisplayStill

  - □ SetMute - Sets audio mute and attenuation
  - □ SetVideoFormat - Sets format to NTSC, PAL or progressive
  - □ SetInterruptMask - Enables/disables interrupts to host
  - □ FlushBitstream - Discards contents of bitstream buffer
  - □ InquireBufferFullness - Measures data in bitstream buffers
  - □ Reset - Initializes the CL480 and its microcode
- □ Provides 13 interrupts, providing the host feedback on bitstream transition, display, and decoding processes

1.3 Functional Description

The CL480 decodes audio and video using a combination of hardwired coprocessors and a programmable RISC CPU, as shown in Figure 1-1. The coprocessors contain special-purpose logic to efficiently perform operations such as Huffman decoding. The RISC CPU is a general-purpose processor that controls the coprocessors and assists in the decoding process.

The microcode for the CPU is stored in DRAM and is loaded into on-chip memory as needed. During power-up, the microcode can either be loaded into DRAM by a host processor or can be read from a ROM connected to the CL480.

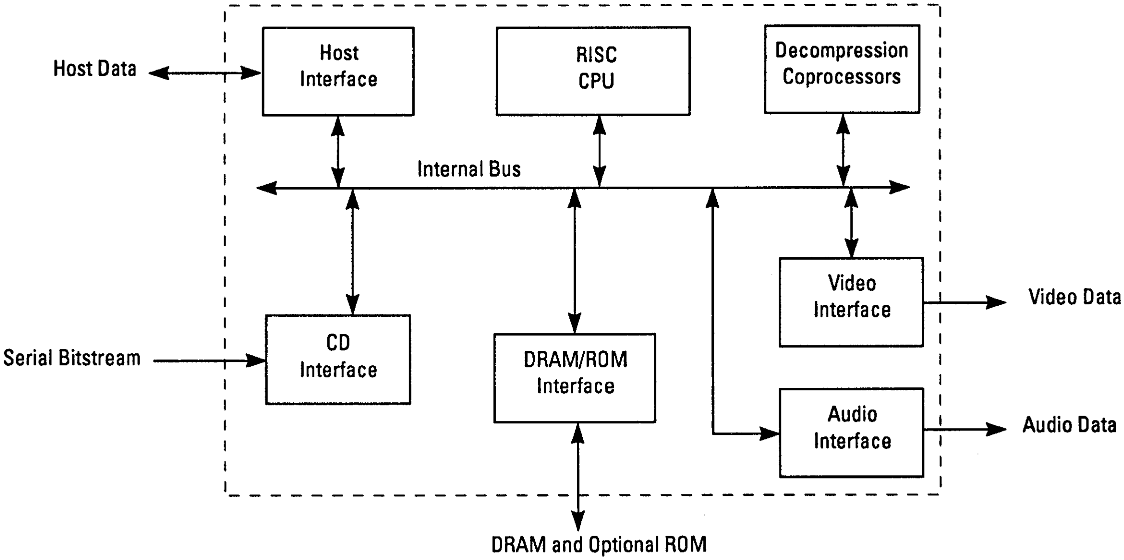


Figure 1-1 Block Diagram of the CL480

Each of the interface modules is described next.

1.3.1 Host Interface

The CL480's host interface is used to initialize the chip, supply compressed data, report status, and control operation. The CL480's host interface is based on an 8-bit bus tailored for low-cost applications; it minimizes pins while allowing the host to access DRAM, the code FIFO, and on-chip registers. These accesses are performed by writing the selected address to a series of three one-byte host address registers. Data is then read from or written to two one-byte data registers.

1.3.2 CD Interface

This interface is designed to receive serial compressed data from a CD-DSP. The CL480 supports six different input formats so that many of the popular CD-DSP chips can be used.

1.3.3 DRAM/ROM Interface

The CL480's DRAM/ROM interface provides a glueless interface between the CL480 and external memory. It connects directly to fast page-mode DRAM, typically either one 256Kx16 DRAM or four 256Kx4 DRAMs. It can also be connected directly to an 8-bit ROM. The ROM access time can be programmed to be 3 to 34 GCKs. The ROM size required for storing the CL480's MPEG microcode is less than 45 Kbytes.

1.3.4 Video Interface

The video interface performs horizontal and vertical interpolation of decoded video. It accepts decompressed video data from the local DRAM and outputs it in:

- □ 24- or 15-bit RGB mode (formatted as BGR-BGR)
- □ 16-bit YCbCr mode (formatted as CbY-CrY)
- □ 8-bit YCbCr mode (formatted as Cb-Y-Cr-Y)

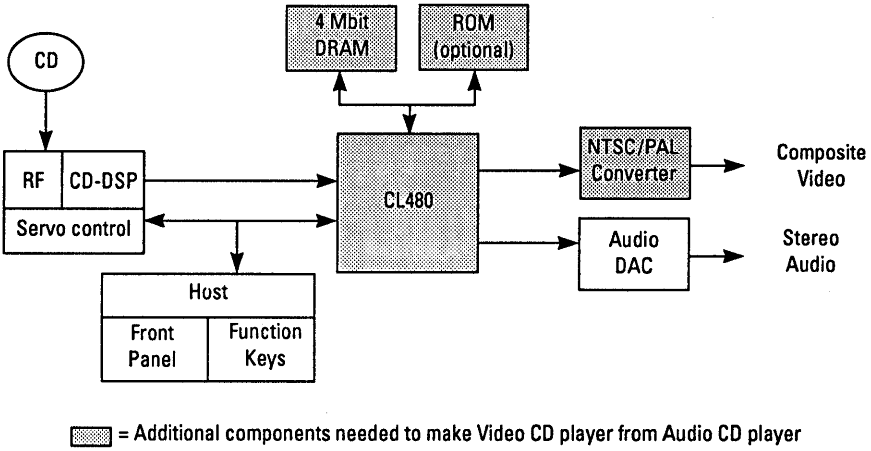
In YCbCr mode, pixels are output in 4:2:2 format, meaning luminance (Y) has twice the horizontal resolution of chrominance (Cb and Cr).

1.3.5 Audio Interface

The CL480 audio interface outputs decoded audio samples in bit-serial format and is able to control their attenuation. The CL480 also has a left/right signal to indicate which channel is being output. The polarity of the left/right signal and the order and number of bits per audio sample are programmable so that any of the popular audio DACs can be used.

1.4 Consumer Electronics Application

The CL480 is designed for low-cost applications that include the consumer electronics market. In consumer electronics products, such as the example shown in Figure 1-2, the CL480 typically receives a CD data stream from a CD-DSP chip via its 4-pin CD interface. This CD data stream could contain an MPEG-1 system stream, CD-DA data or CD-ROM data.



▨ = Additional components needed to make Video CD player from Audio CD player

Figure 1-2 CL480 in Consumer Electronics Configuration

Chapter 2
MPEG Overview

This chapter presents an overview of the Moving Picture Experts Group (MPEG) standard that is implemented by the CL480. The standard is officially known as ISO/IEC Standard, *Coded Representation of Picture, Audio and Multimedia/hypermedia Information*, ISO 11172. It is more commonly referred to as the *MPEG-1 standard*.[1]

MPEG addresses the compression, decompression and synchronization of video and audio signals. The MPEG video algorithm can compress video signals to an average of about 1/2 to 1 bit per coded pixel. At a compressed data rate of 1.2 Mbits per second, a coded resolution of 352 x 240 at 30 Hz is often used, and the resulting video quality is comparable to VHS. Image quality can be significantly improved by using a more highly-compressed data rate (for example, 2 Mbits per second) without changing the coded resolution.

[1] For documentation, contact the American National Standards Institute (ANSI), 11 West 42nd St., New York, NY 10036, (212) 642-4900.

2.1 MPEG Stream Structure

This section explains the structure of an MPEG system stream and introduces some concepts used in the rest of the chapter.

2.1.1 MPEG System Stream Structure

In its most general form, an *MPEG system stream* is made up of two layers:

- □ The *system layer* contains timing and other information needed to demultiplex the audio and video streams and to synchronize audio and video during playback.
- □ The *compression layer* includes the audio and video streams.

2.1.2 General Decoding Process

Figure 2-1 shows a generalized decoding system for the audio and video streams.

The *system decoder* extracts the timing information from the MPEG system stream and sends it to the other system components. (Section 2.4, Synchronization, has more information about the use of timing information for audio and video synchronization.) The system decoder also demultiplexes the video and audio streams from the system stream, and sends each to the appropriate decoder.

The *video decoder* decompresses the video stream as specified in Part 2 of the MPEG standard. (See Section 2.2, Inter-picture Coding, and Section 2.3, Intra-picture Coding, for more information about video compression.)

The *audio decoder* decompresses the audio stream as specified in Part 3 of the MPEG standard.

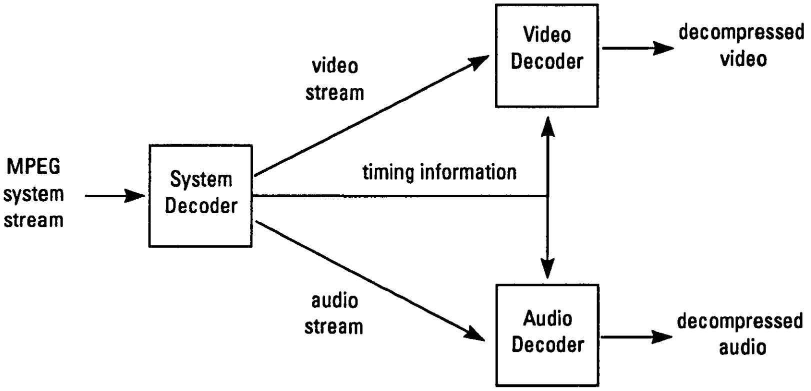


Figure 2-1 General MPEG Decoding System

2.1.3 Video Stream Data Hierarchy

The MPEG standard defines a hierarchy of data structures in the video stream as shown schematically in Figure 2-2.

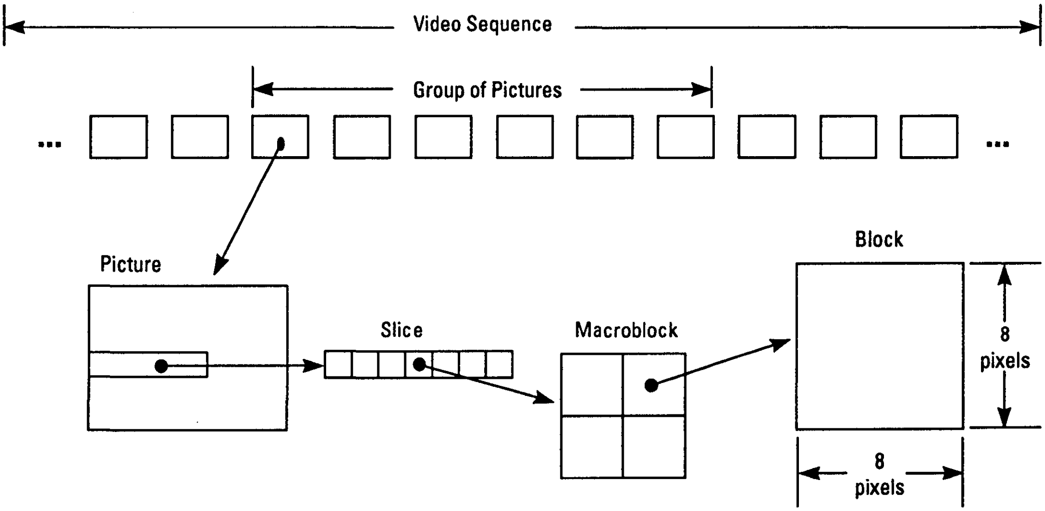


Figure 2-2 MPEG Data Hierarchy

Video Sequence

Begins with a sequence header (may contain additional sequence headers), includes one or more groups of pictures, and ends with an end-of-sequence code.

Group of Pictures

A header and a series of one or more pictures intended to allow random access into the sequence.

Picture

The primary coding unit of a video sequence. A picture consists of three rectangular matrices representing luminance (Y) and two chrominance (Cb and Cr) values. The Y matrix has an even number of rows and columns. The Cb and Cr matrices are one-half the size of the Y matrix in each direction (horizontal and vertical).

Figure 2-3 shows the relative x-y locations of the luminance and chrominance components. Note that for every four luminance values, there are two associated chrominance values: one Cb value and one Cr value. (The location of the Cb and Cr values is the same, so only one circle is shown in the figure.)

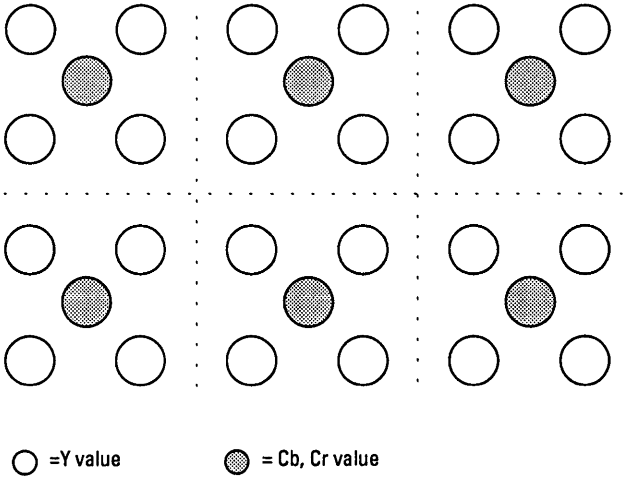


Figure 2-3 Location of Luminance and Chrominance Values

Slice

One or more "contiguous" macroblocks. The order of the macroblocks within a slice is from left to right and top to bottom.

Slices are important in the handling of errors. If the bitstream contains an error, the decoder can skip to the start of the next slice. Having more slices in the bitstream allows better error concealment but uses bits that could otherwise be used to improve picture quality.

Macroblock

A 16-pixel by 16-line section of luminance components and the corresponding 8-pixel by 8-line section of the two chrominance components. See Figure 2-3 for the spatial location of luminance and chrominance components. A macroblock contains four Y blocks, one Cb block and one Cr block as shown in Figure 2-4. The numbers correspond to the ordering of the blocks in the data stream, with block 1 first.

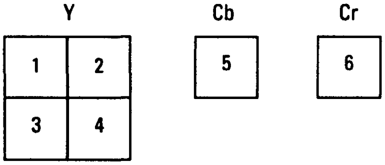


Figure 2-4 Macroblock Composition

Block

A block is an 8-pixel by 8-line set of values of a luminance or a chrominance component. Note that a luminance block corresponds to one-fourth as large a portion of the displayed image as does a chrominance block.

2.2 Inter-picture Coding

Much of the information in a picture within a video sequence is similar to information in a previous or subsequent picture. The MPEG standard takes advantage of this temporal redundancy by representing some pictures in terms of their differences from other (reference) pictures, or what is known as *inter-picture coding*. This section describes the types of coded pictures and explains the techniques used in this process.

2.2.1 Picture Types

The MPEG standard specifically defines three types of pictures: intra, predicted, and bidirectional.

Intra Pictures

Intra pictures, or I-pictures, are coded using only information present in the picture itself. I-pictures provide potential random access points into the compressed video data. I-pictures use only transform coding (as explained in section 2.3 on page 10) and provide moderate compression. I-pictures typically use about two bits per coded pixel.

Predicted Pictures

Predicted pictures, or P-pictures, are coded with respect to the nearest previous I- or P-picture. This technique is called *forward prediction* and is illustrated in Figure 2-5.

Like I-pictures, P-pictures serve as a prediction reference for B-pictures and future P-pictures. However, P-pictures use *motion compensation* (see section 2.2.3) to provide more compression than is possible with I-pictures. Unlike I-pictures, P-pictures can propagate coding errors because P-pictures are predicted from previous reference (I- or P-) pictures.

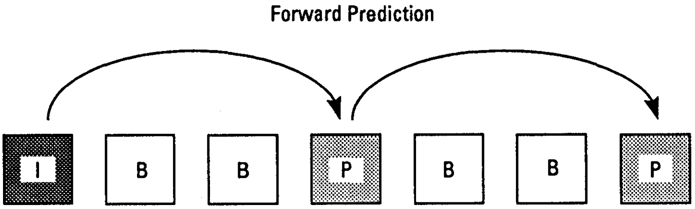


Figure 2-5 Forward Prediction

Bidirectional Pictures

Bidirectional pictures, or B-pictures, are pictures that use both a past and future picture as a reference. This technique is called *bidirectional prediction* and is illustrated in Figure 2-6. B-pictures provide the most compression and do not propagate errors because they are never used as a reference. Bidirectional prediction also decreases the effect of noise by averaging two pictures.

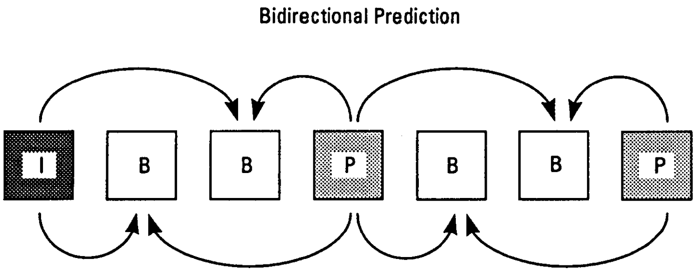


Figure 2-6 Bidirectional Prediction

2.2.2 Video Stream Composition

The MPEG algorithm allows the encoder to choose the frequency and location of I-pictures. This choice is based on the application's need for random accessibility and the location of scene cuts in the video sequence. In applications where random access is important, I-pictures are typically used two times a second.

The encoder also chooses the number of B-pictures between any pair of reference (I- or P-) pictures. This choice is based on factors such as the amount of memory in the encoder and the characteristics of the material being coded. For example, a large class of scenes have two bidirectional pictures separating successive reference pictures. A typical arrangement of I-, P-, and B-pictures is shown in Figure 2-7 in the order in which they are displayed.

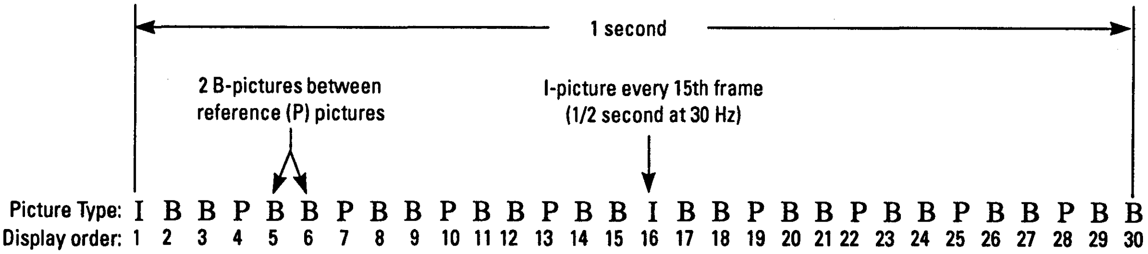

1 second
2 B-pictures between reference (P) pictures
I-picture every 15th frame (1/2 second at 30 Hz)
Picture Type: I B B P B B P B B P B B P B B I B B P B B P B B P B B P B B
Display order: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30

Figure 2-7 Typical Display Order of Picture Types

The MPEG encoder reorders pictures in the video stream to present the pictures to the decoder in the most efficient sequence. In particular, the reference pictures needed to reconstruct B-pictures are sent *before* the associated B-pictures. Figure 2-8 demonstrates this ordering for the first section of the example shown above.

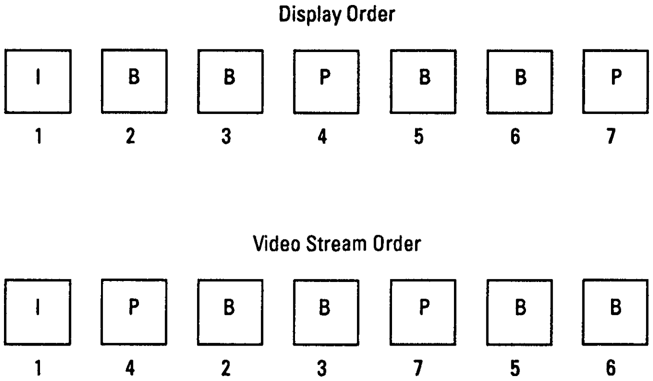


Figure 2-8 Video Stream versus Display Ordering

2.2.3 Motion Compensation

Motion compensation is a technique for enhancing the compression of P- and B-pictures by eliminating temporal redundancy. Motion compensation typically improves compression by about a factor of three compared to intra-picture coding. Motion compensation algorithms work at the macroblock level.

When a macroblock is compressed by motion compensation, the compressed file contains this information:

- □ The spatial vector between the reference macroblock(s) and the macroblock being coded (*motion vectors*)
- □ The content differences between the reference macroblock(s) and the macroblock being coded (*error terms*)

Not all information in a picture can be predicted from a previous picture. Consider a scene in which a door opens: The visual details of the room behind the door cannot be predicted from a previous frame in which the door was closed. When a case such as this arises—i.e., a macroblock in a P-picture cannot be efficiently represented by motion compensation—it is coded in the same way as a macroblock in an I-picture using transform coding techniques (see Section 2.3, Intra-picture Coding).

The difference between B- and P-picture motion compensation is that macroblocks in a P-picture use the previous reference (I- or P-picture) only, while macroblocks in a B-picture are coded using any combination of a previous or future reference picture.

Four codings are therefore possible for each macroblock in a B-picture:

- □ Intra coding: no motion compensation
- □ Forward prediction: the previous reference picture is used as a reference
- □ Backward prediction: the next picture is used as a reference
- □ Bidirectional prediction: two reference pictures are used, the previous reference picture and the next reference picture

Backward prediction can be used to predict uncovered areas that do not appear in previous pictures.

2.3 Intra-picture (Transform) Coding

The MPEG transform coding algorithm includes these steps:

- □ Discrete cosine transform (DCT)
- □ Quantization
- □ Run-length encoding

Both image blocks and prediction-error blocks have high spatial redundancy. To reduce this redundancy, the MPEG algorithm transforms 8 x 8 blocks of pixels or 8 x 8 blocks of error terms from the spatial domain to the frequency domain with the Discrete Cosine Transform (DCT).

Next, the algorithm quantizes the frequency coefficients. Quantization is the process of approximating each frequency coefficient as one of a limited number of allowed values. The encoder chooses a quantization matrix that determines how each frequency coefficient in the 8 x 8 block is quantized. Human perception of quantization error is lower for high spatial frequencies, so high frequencies are typically quantized more coarsely (i.e., with fewer allowed values) than low frequencies.

The combination of DCT and quantization results in many of the frequency coefficients being zero, especially the coefficients for high spatial frequencies. To take maximum advantage of this, the coefficients are organized in a zigzag order to produce long runs of zeros (see Figure 2-9). The coefficients are then converted to a series of run-amplitude pairs, each pair indicating a number of zero coefficients and the amplitude of a non-zero coefficient. These run-amplitude pairs are then coded with a variable-length code, which uses shorter codes for commonly occurring pairs and longer codes for less common pairs.

Some blocks of pixels need to be coded more accurately than others. For example, blocks with smooth intensity gradients need accurate coding to avoid visible block boundaries. To deal with this inequality between blocks, the MPEG algorithm allows the amount of quantization to be modified for each macroblock of pixels. This mechanism can also be used to provide smooth adaptation to a particular bit rate.

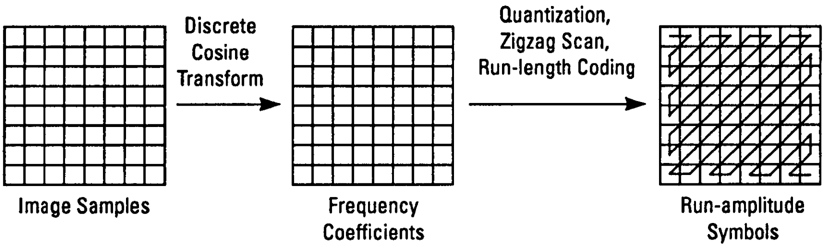


Figure 2-9 Transform Coding Operations

2.4 Synchronization

The MPEG standard provides a timing mechanism that ensures synchronization of audio and video. The standard includes two parameters: the *system clock reference (SCR)* and the *presentation time stamp (PTS)*.

The MPEG-specified "system clock" runs at 90 kHz. System clock reference and presentation time stamp values are coded in MPEG bitstreams using 33 bits, which can represent any clock cycle in a 24-hour period.

2.4.1 System Clock References

A system clock reference is a snapshot of the encoder system clock which is placed into the system layer of the bitstream, as shown in Figure 2-10. During decoding, these values are used to update the system clock counter in the CL480.

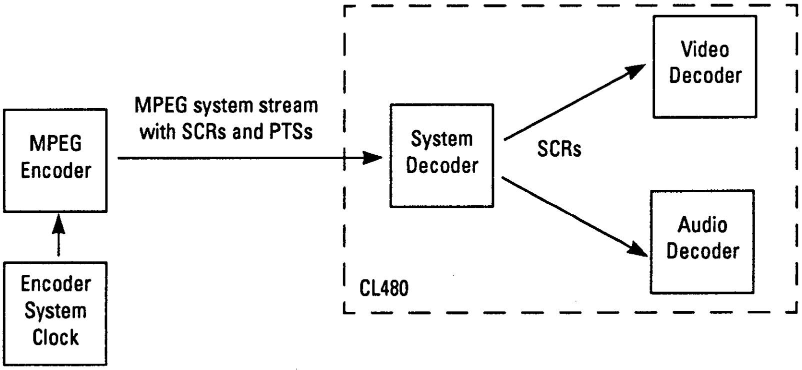


Figure 2-10 SCR Flow in MPEG System

2.4.2 Presentation Time Stamps

Presentation time stamps are samples of the encoder system clock that are associated with video or audio *presentation units*. A presentation unit is a decoded video picture or a decoded audio time sequence. The PTS represents the time at which the video picture is to be displayed or the starting playback time for the audio time sequence.

The decoder either skips or repeats picture displays to ensure that the PTS is within one picture's worth of 90 kHz clock tics of the SCR when a picture is displayed. If the PTS is earlier (has a smaller value) than the current SCR, the decoder discards the picture. If the PTS is later (has a larger value) than the current SCR, the decoder repeats the display of the picture.

Chapter 3
Signal Descriptions

This chapter describes the signals that comprise the external physical interface to the CL480. The information presented for each signal includes the signal mnemonic and name, type (input, output, or bidirectional), and description. (Note: The overbar symbol denotes active low polarity.) For information about the functional operation of the CL480, including functional and timing waveforms, see Chapters 4, 5, 6, 7 and 8.

This chapter is divided into the following seven sections that correspond to the components that interface to the CL480:

- 3.1: Global Interface
- 3.2: Host Interface
- 3.3: CD Interface
- 3.4: DRAM Interface
- 3.5: Video Interface
- 3.6: Audio Interface
- 3.7: Programmable I/Os

Figure 3-1 shows a diagram of the CL480 with all signals grouped together.

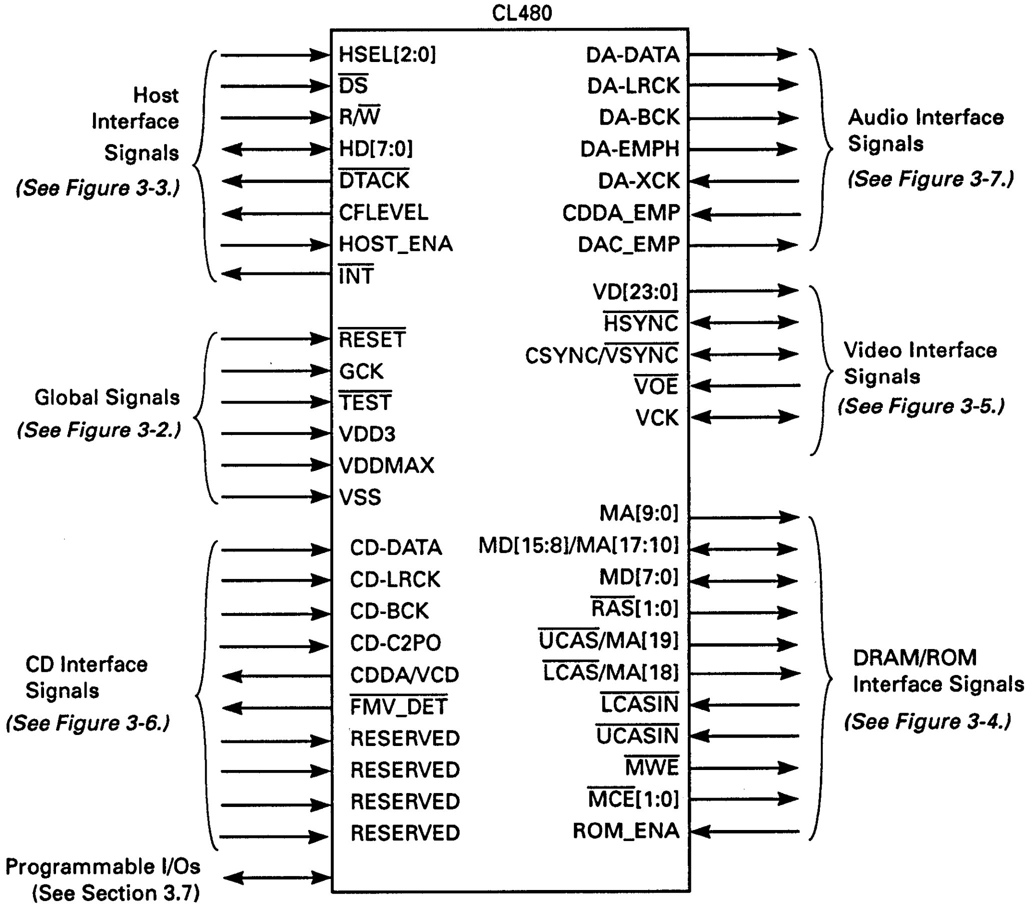


Figure 3-1 Bus Connection Diagram

3.1 Global Interface

The CL480 global signals are composed of the RESET, GCK, TEST, and power signals as shown in Figure 3-2.

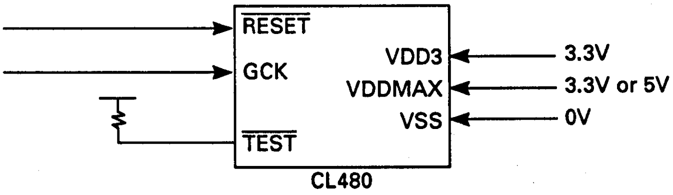


Figure 3-2 Global Signals

$\overline{\text{RESET}}$ — Hardware Reset **Input**

An external device asserts $\overline{\text{RESET}}$ (active low) to force the CL480 to execute a hardware reset. To be recognized, $\overline{\text{RESET}}$ must be asserted for at least ten complete GCK cycles. After a reset, the CL480 registers are in an indeterminate state, and the microcode should be reloaded as described in Chapter 11.

GCK — Global Clock **Input**

The CL480 uses GCK to clock the internal processor. This oscillator is typically 40 MHz or 40.5 MHz. The host can write a register to internally divide the clock by 16 (2.5 MHz) to provide a low-power sleep mode.

$\overline{\text{TEST}}$ — Test **Input**

This pin is used for chip testing. For normal operation, $\overline{\text{TEST}}$ must be held HIGH (deasserted).

VDD3 — Power **Input**

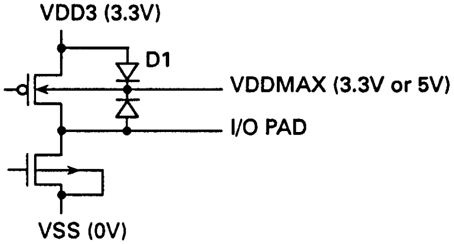
2.7 to 3.6V internal power and output high voltage.

VSS — Power **Input**

Ground.

VDDMAX — Power **Input**

Maximum input voltage connected to any CL480 pin. The VDDMAX pin is connected to the N-well of the P-channel output transistor on I/O pads as shown in the figure below. The voltage of VDDMAX should be greater than or equal to the voltage driven on any input.



3.2 Host Interface

These signals are used to communicate between the CL480 and the host processor. Figure 3-3 shows how the data transfer signals of the CL480 connect to the host processor. The various modes of transferring data are discussed in Sections 4.2 and 4.3.

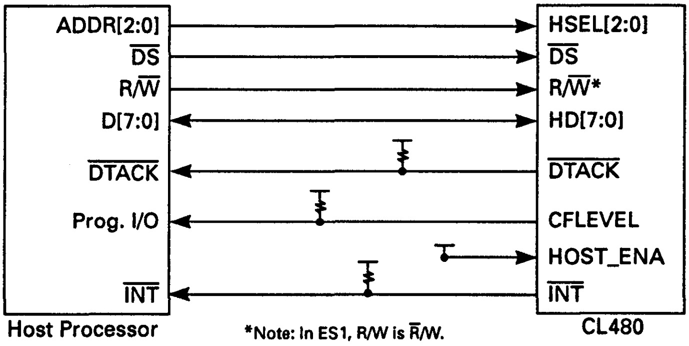


Figure 3-3 Host Interface Signals

HSEL[2:0] — Host Address Bus **Inputs**

This three-bit address bus selects one of five host interface registers from which other resources within the CL480 may be accessed.

$\overline{\mathrm{DS}}$ —Data Strobe **Input**

The host processor asserts $\overline{\mathrm{DS}}$ to select the CL480 for a read or write operation. The falling edge of this signal triggers a new cycle.

HD[7:0] — Host Data Bus **Bidirectionals**

HD[7:0] comprises the 8-bit bidirectional host data bus. The host processor uses HD[7:0] to write data to the CL480's Code FIFO, internal registers, and local DRAM. The CL480 uses HD[7:0] to send requested data to the host processor. The direction is determined by R/$\overline{\mathrm{W}}$ ($\overline{\mathrm{R}}$/W in ES1).

R/$\overline{\mathrm{W}}$ — Read/Write **Input**

The host processor drives R/$\overline{\mathrm{W}}$ low to initiate a write operation, and drives R/$\overline{\mathrm{W}}$ high to initiate a CL480 read operation to the host data bus. (R/W polarity is reversed in ES1.)

$\overline{\mathrm{DTACK}}$ — Host Data Acknowledge Open-Drain Output

The CL480 asserts $\overline{\mathrm{DTACK}}$ (active low) when it is ready to receive or output data on HD[7:0]. When the CL480 responds to a read request, it holds $\overline{\mathrm{DTACK}}$ deasserted (high) until the requested data is ready. When the CL480 responds to a write request, it asserts $\overline{\mathrm{DTACK}}$ when it has received and latched the write data.

$\overline{\text{DTACK}}$ is an open-drain signal, which allows it to be wire-ORed with other components on the host bus. It requires a pullup resistor of at least 1.5K ohms.

CFLEVEL — Coded Data FIFO Level Status Output

When CFLEVEL is zero, the CL480 Coded Data FIFO has room for at least 44 bytes of compressed data. CFLEVEL is an open-drain signal. This allows it to be wire-ORed with other components on the host bus. It requires a pullup resistor of at least 1.5K ohms.

$\overline{\text{INT}}$ — Interrupt Request Output

The CL480 asserts $\overline{\text{INT}}$ to request an interrupt from the host processor. Interrupt events are determined from the SetInterruptMask() macro command. A 1.5K pull-up should be added on this pin.

HOST_ENA—Host Enable Input

Enables the host interface. By disabling HOST_ENA (HOST_ENA=0), host interface pins may be used for control functions as described in Section 3.7 and 11.6.

3.3 CD Interface

The CL480's CD interface is dedicated to receiving the serial-bit output of a CD DSP. A 4-wire serial bus connects the CD DSP directly with the CL480 as shown in Figure 3-5.

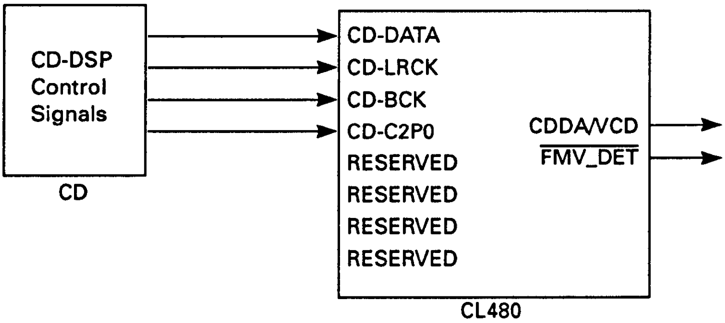


Figure 3-4 CD-Decoder Interface Signals

CD-BCK — Bit Clock **Input**

BCK is the CD-Decoder bit clock. The CL480 can accept multiple BCK rates as detailed in Table 6-1 of Chapter 6.

CD-DATA — Data **Input**

The serial data input from the CD-DSP.

CD-LRCK — Left-Right Clock **Input**

CD-LRCK provides 16-bit word synchronization to the CL480 and has programmable polarity (either left or right channel high).

CD-C2P0 — CD-ROM Data **Input**

This signals a corrupted byte (error byte flag MSB or LSB first). It is used when receiving CD-ROM data but is ignored in CD-DA pass-through mode. This signal is high when an error occurs.

CDDA/VCD— CDDA/V-CD Enable **Output**

Outputs whether incoming data is CDDA versus V-CD: CDDA = 1, V-CD = 0.

$\overline{\text{FMV\_DET}}$— FMV Detected **Output**

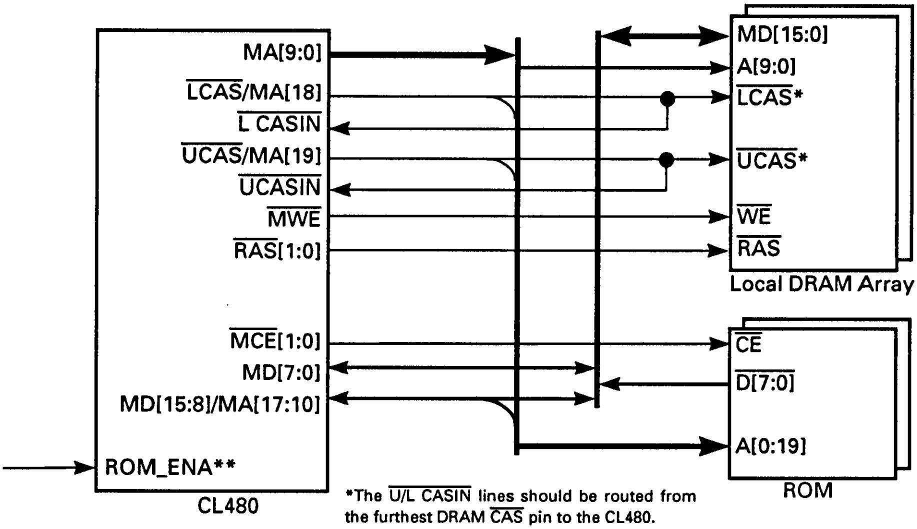
Indicates a CD-I bitstream. Outputs zero if FMV is detected.

RESERVED — RESERVED **Output**

These four pins are reserved.

3.4 DRAM Interface

Figure 3-4 shows the signals that comprise the CL480's DRAM interface. The signal descriptions are presented following the figure. See Chapter 5 for more information about the DRAM memory architecture.



*The U/L CASIN lines should be routed from the furthest DRAM CAS pin to the CL480.

**This signal should be pulled high if a ROM is used.

Figure 3-5 DRAM Interface Signals

MA[9:0] — Memory Address Bus Outputs

The CL480 multiplexes the row and column addresses on these signals to address up to one Mbyte of DRAM and uses address bits 9:0 for ROM addresses. See Chapter 5 for details of addressing various DRAM components and DRAM array sizes.

MD[15:0] — Memory Data Bus Bidirectionals

These signals comprise the memory data bus by which data is transferred between the CL480 and the local DRAM/ROM array. All 16 bits are used in DRAM accesses, the direction of which is determined by the state of MWE.

For ROM accesses, MDATA[7:0] is the ROM data bus, and MDATA[15:8] becomes ROM address bits [17:10].

RAS[1:0] — Row Address Strobe Outputs

The CL480 asserts these signals to latch the row address into the DRAM array. RAS1 (active low) latches the row address for bank 1, and RAS0 (low) latches the row address for bank 0.

$\overline{\text{UCAS}}$— Upper Column Address Strobe **Output**

$\overline{\text{LCAS}}$— Lower Column Address Strobe **Output**

The CL480 asserts these signals to latch the column address into the DRAM array. $\overline{\text{UCAS}}$ (active low) latches the column address for the upper memory data byte, MD[15:8], and $\overline{\text{LCAS}}$ (active low) latches the address for the lower byte, MD[7:0]. $\overline{\text{UCAS}}$ is generated in response to the $\overline{\text{UDS}}$ input, and $\overline{\text{LCAS}}$ is generated in response to the $\overline{\text{LDS}}$ input.

Note: The $\overline{\text{LCAS}}$ and $\overline{\text{UCAS}}$ bits are also used as bits A[18] and A[19], respectively, in ROM accesses.

$\overline{\text{UCASIN}}$ — Upper Data Latch Enable **Inputs**

$\overline{\text{LCASIN}}$ — Lower Data Latch Enable **Inputs**

When the CL480 reads data from the local DRAM array, the data on MD[15:0] is latched into the CL480 on the rising edge of the two $\overline{\text{CASIN}}$ signals. $\overline{\text{UCASIN}}$ latches data coming from the high data byte, MD[15:8], and $\overline{\text{LCASIN}}$ latches data coming from the low data byte, MD[7:0]. Typically, these are connected to the $\overline{\text{UCAS}}$ and $\overline{\text{LCAS}}$ pins, respectively.

$\overline{\text{MWE}}$ — Write Enable **Output**

The CL480 asserts $\overline{\text{MWE}}$ (active low) during write operations (data transfer from CL480 to DRAM). The CL480 leaves $\overline{\text{MWE}}$ high during a read operation (DRAM to CL480).

$\overline{\text{MCE}}$[1:0] —Chip Enable **Output**

The CL480 asserts $\overline{\text{MCE}}$ (one for each ROM) during a read operation from ROM to the CL480.

ROM_ENA—Boot ROM Enable **Input**

Enables boot ROM. When this pin is high, the CL480 will initialize itself with the data in ROM bank 0.

3.5 Video Interface

The CL480's video interface outputs pixel data to the video display subsystem in RGB or YCbCr format. Figure 3-5 shows two possible configurations of the signals in the CL480's video interface: $\overline{\text{VSYNC}}$/$\overline{\text{HSYNC}}$/VCK as outputs, and $\overline{\text{VSYNC}}$/$\overline{\text{HSYNC}}$/VCK as inputs.

Note: depending on the video mode selected—RGB-24, YCbCr-16, and YCbCr-8—the pin count of the video interface varies, respectively, from 28, to 20, to 12 pins.

Operation of the video interface is discussed in Chapter 7.

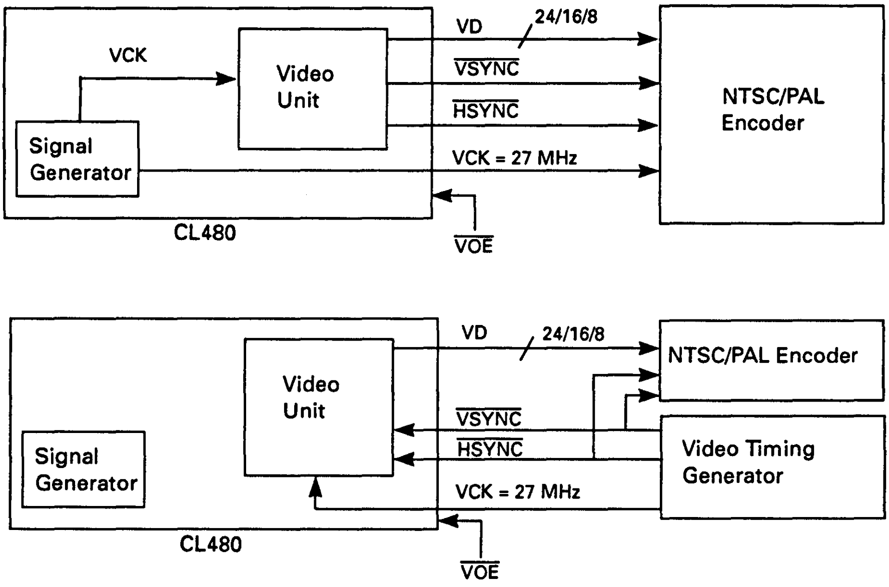


Figure 3-6 Video Interface Signals

VD[23:0] — Pixel Data Bus Outputs

The CL480 transmits pixel data to the video display subsystem using these signals. The definition of the signal lines differs for RGB and YCbCr formats as explained in Chapter 7.

$\overline{\text{HSYNC}}$ — Horizontal Synchronization Bidirectional

The CL480 begins outputting pixel data for a new horizontal line after the rising (inactive) edge of $\overline{\text{HSYNC}}$. $\overline{\text{HSYNC}}$ must be synchronous to VCK as shown in Chapter 7.

$\overline{\text{VSYNC}}$ — Vertical Synchronization Bidirectional

The CL480 begins outputting the top border of a new field on the first $\overline{\text{HSYNC}}$ after the rising edge of $\overline{\text{VSYNC}}$. $\overline{\text{VSYNC}}$ can be asynchronous with respect to VCK. See Chapter 7 for more information on the relationship of $\overline{\text{VSYNC}}$ to $\overline{\text{HSYNC}}$.

$\overline{\text{VOE}}$ — Video Output Enable **Input**

$\overline{\text{VOE}}$ must be asserted (active low) to enable the CL480 to drive the pixel bus, PD[23:0]. When $\overline{\text{VOE}}$ is deasserted, the CL480 puts the pixel bus in a high-impedance state (Note: This signal is active high in ES1).

VCK — Video Clock **Bidirectional**

VCK is derived from GCK and can be either an input or an output signal (which is independent of the direction of $\overline{\text{VSYNC}}$ and $\overline{\text{HSYNC}}$). VCK does not need to be synchronous with GCK.

3.6 Audio Interface

The CL480's audio interface provides the reconstructed audio samples to the audio DACs. This bus complies with the usual three-wire (DA-BCK, DA-DATA, DA-LRCK) audio output. It also takes as input two external sources, DA-XCK and CDDA_EMP, and outputs DAC_EMP.

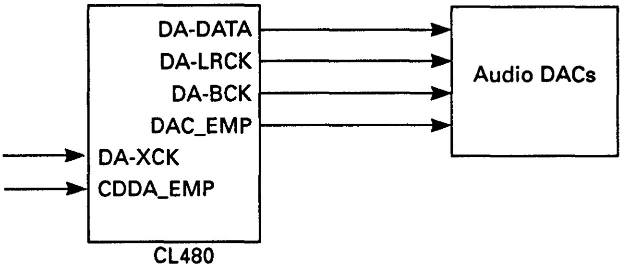


Figure 3-7 **Audio Interface Signals**

DA-DATA — Data **Output**

This signal outputs bit serial audio samples based on the DA-BCK clock.

DA-LRCK — Left-Right Clock **Output**

This signal identifies the channel for each audio sample.

DA-BCK — Bit Clock **Output**

DA-BCK is the audio bit clock. Divided by 8 from DA-XCK, it can be either 48 or 32 times the sampling clock.

DAC_EMP - Output Emphasis Flag **Output**

In CD-DA pass-through mode, this signal is routed to the audio DACs.

DA-XCK — External Audio Frequency Clock **Input**

Used to generate DA-BCK and DA-LRCK, DA-XCK can be either 384 or 256 times the sampling frequency.

CDDA_EMP — Input Emphasis Flag **Input**

In CD-DA pass-through mode, this imput is directly routed to the DAC_EMP output pin.

3.7 Programmable I/Os

The CL480 provides programmable I/Os that may be used for control. These I/Os become available by disabling the host interface (HOST_ENA=0) so that host interface pins may be used for control functions as shown in the following tables.

Pin Name	Name/Function	I/O
HSEL[0]	STOP/PLAY: 1 = Pause, 0 = Play	I
HSEL[1]	SCAN: 1 = Scan	I

Note: In the table below, HD[0] and HD[1] perform the same functions normally set by SetMute().

HD[0] (MUTE0)	HD[1] (MUTE1)	Attenuation on the PCM Samples
0	0	0 dB
1	0	-12 dB
0	1	-∞ (20 ms soft mute)
1	1	-12 dB

4
Host Interface Functional Description

4.1 Functional Overview

The CL480's host interface, shown in Figure 4-1, is intended to provide a simple interface to an eight-bit microcontroller. The host interface provides three distinct functions:

- □ Coded data input (coded data may be sent through the host interface if the CD interface is not used for this purpose)
- □ Local DRAM/ROM access
- □ Internal register access

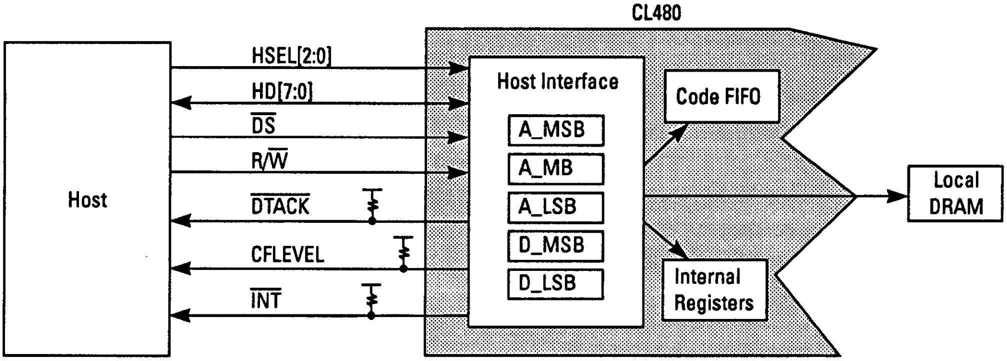


Figure 4-1 CL480 Host Interface

The host communication functions also include device initialization and microcode loading (if the ROM is not used). Host accesses to the CL480 can be asynchronous to GCK.

4.2 Host Interface Registers

As shown in Figure 4-1, the host processor accesses CL480 resources by writing to address registers and reading or writing to data registers. A combination of five host interface registers are used:

- Three 8-bit address registers: A_MSB, A_MB, A_LSB
- Two 8-bit data registers: D_LSB and D_MSB

4.2.1 Host Interface Address Registers

The CL480's address registers allow the host to access the CL480's C-FIFO, local DRAM array or internal GBUS registers by setting A_MSB[7:6], as shown in Table 4-1.

Table 4-1 Register Address Bits Used to Access DRAM/ROM, C-FIFO, and GBUS

A_MSB[7:6]	Data From/To
00	C-FIFO
01	DRAM/ROM (No autoincrement)
10	GBUS
11	DRAM/ROM (Autoincrement)

Thus the address register specifies which of four types of operations the CL480 can perform:

- Write to Code FIFO
- Read/write to GBUS (internal registers)
- Read/write to DRAM (external memory)
- Read/write to DRAM (external memory) with auto-increment addressing

4.2.2 Host Interface Data Registers

The two host 8-bit data registers, D_LSB and D_MSB, are buffers between the host bus and internal modules.

Transfers from the data registers to the destination specified by the address registers are triggered on the write operation to D_MSB and the read operation from D_MSB. Therefore, the write and read sequences are as follows:

- □ Write:
  - □ Set address registers
  - □ Write D_LSB register
  - □ Write D_MSB register
- □ Read:
  - □ Set address registers
  - □ Read D_MSB register
  - □ Read D_LSB register

Note: For back-to-back transactions to the C-FIFO, DRAM and internal registers, only the address bytes that differ from earlier transaction(s) need to be re-written a second time.

4.2.3 Accessing Host Interface Registers

The host accesses each of the five host interface registers by issuing a bus access with HSEL[2:0] set appropriately, as shown in Table 4-2.

Table 4-2 Summary of Host Interface Local Registers

HSEL[2:0][1]	Register Name	Contents
001	A_LSB	Address, byte 0 (LSB)
010	A_MB	Address, byte 1
011	A_MSB	Address, byte 2(MSB)
100	D_LSB	Data, byte 0 (LSB)
101	D_MSB	Data, byte 1 (MSB)

[1] Note: 000, 110 and 111 are illegal combinations which should not be used.

4.2.4 DRAM/ROM Access

Bit 4 of A_LSB determines whether DRAM (set to 0) versus ROM (set to 1) is selected.

DRAM Access

When bit 6 of A_MSB is set to 1, the host sets A_MSB, A_MB, and A_LSB to form a 21-bit memory address as shown in Figure 4-2.

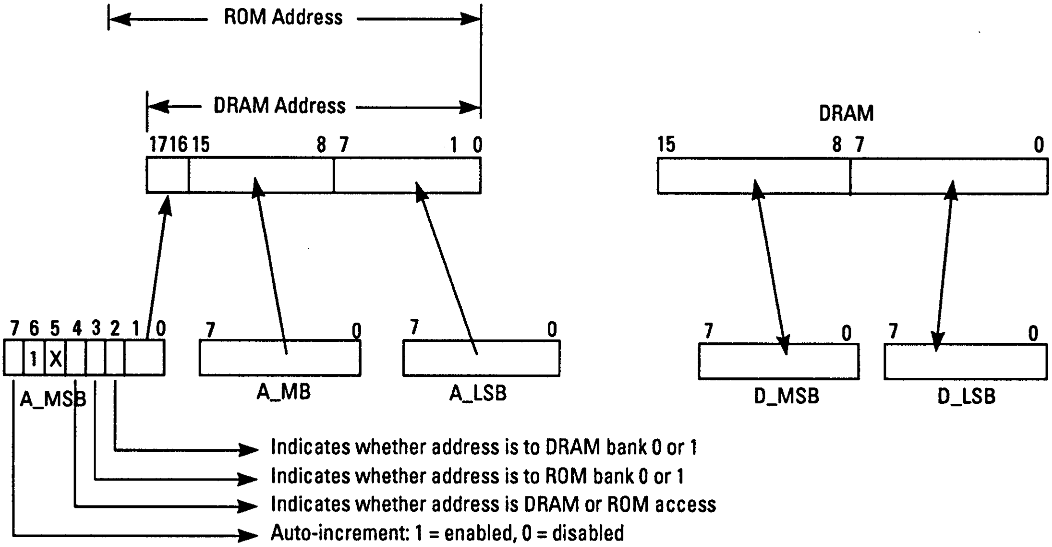

ROM Address
DRAM Address
17 16 15 8 7 1 0
DRAM
15 8 7 0
7 6 5 4 3 2 1 0
1
X
A_MSB
7 0
A_MB
7 0
A_LSB
7 0
D_MSB
7 0
D_LSB
Indicates whether address is to DRAM bank 0 or 1
Indicates whether address is to ROM bank 0 or 1
Indicates whether address is DRAM or ROM access
Auto-increment: 1 = enabled, 0 = disabled

Figure 4-2 DRAM/ROM Address Formed by Host Address Registers

This 21-bit memory address is a 16-bit word address, not a byte address. The address map is shown in Table 4-3.

Table 4-3 Memory Address Map

A_MSB[4]	A_MSB[3]	A_MSB[2]	Memory Accessed
1	1	1	ROM Bank 1
1	1	0	ROM Bank 1
1	0	1	ROM Bank 0
1	0	0	ROM Bank 0
0	1	1	unused
0	1	0	unused
0	0	1	DRAM Bank 1
0	0	0	DRAM Bank 0

For DRAM accesses, bit 7 of Host address register A_MSB controls autoincrementing. When autoincrementing is enabled (bit 7 = 1), each 16-bit data transfer between the host and the CL480 causes the DRAM address to be incremented to the next *word.*

ROM Access

For access of ROM bank 0 versus 1, see Table 4-3.

4.2.5 Code FIFO Access

Host bus coded data may be sent to the CL480 in one of two basic modes:

- □ Serial mode (through the CD interface), as shown in Figure 4-3 and described further in Chapter 8.
- □ Parallel mode (through the host interface), as shown in Figure 4-4 and described in this section.

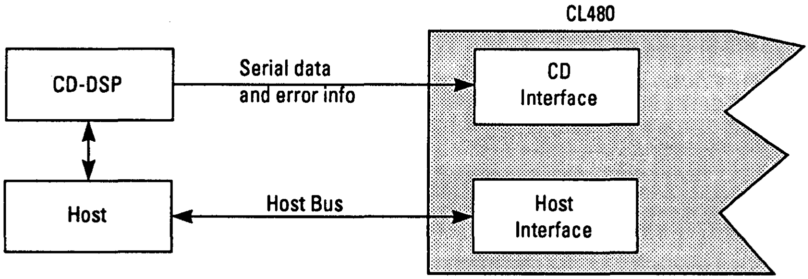


Figure 4-3 Sending Coded Data on CD Interface

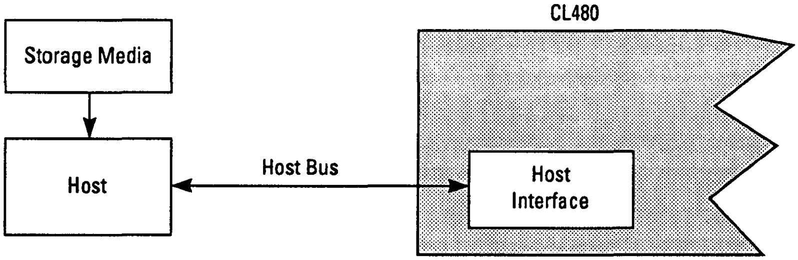


Figure 4-4 Sending Coded Data on Host Interface

In code FIFO accesses through the Host interface, the Host specifies A_MSB[7:6]=00 as shown in Figure 4-6. Then the words written to D_MSB and D_LSB will transfer to the Code FIFO.

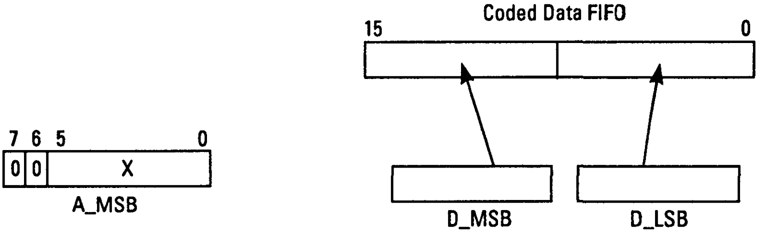


Figure 4-5 C-FIFO Register Address Formed by A_MSB

The CL480 can sustain a coded data input transfer rate of up to 2.5 Mbytes per second through the D_MSB and D_LSB registers. The C-FIFO holds 31 words, each of which is 16-bits wide.

4.2.6 Internal Register Access

If the Host specifies A_MSB[7:6]=10, then the access is an internal register access, as shown in Figure 4-6.

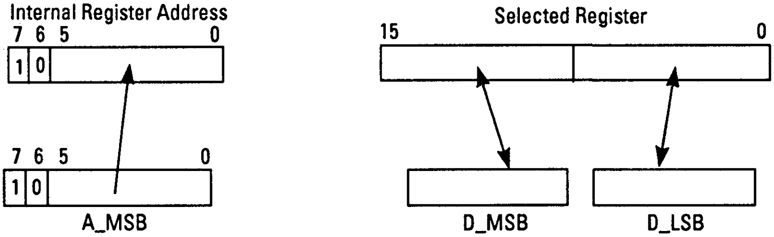


Figure 4-6 Internal Register Address Formed by A_MSB

Internal registers that are accessible by the host are listed in Table 4-3.

Note: Registers are defaulted to the correct value through microcode initialization unless otherwise noted in the Configuration Area description given in Section 12.3.

Table 4-4 Internal Registers Decoded by A_MSB

Access	A_MSB[5:0]	Register Name	R/W	Description
Direct	0x33	CPU_cntl	R/W	Selects type of data sent to the CL480
	0x3A	CPU_pc	R/W	CPU program counter
	0x36	CPU_iaddr	R/W	IMEM write address
	0x32	CPU_imem	R/W	Data for instruction memory (IMEM)
	0x0F	HOST_int	R/W	Host nterrupt register
	0x07	HOST_pio	R/W	Direction and data for programmable I/Os
	0x20	DRAM_iaddr	R/W	Indirect address for DRAM controller
	0x21	DRAM_data	W	Data for DRAM controller registers
	0x12	CD_cntl	R/W	Selects type of data sent to the CL480
	0x10	CD_cnfg	R/W	Selects data format for CD interface
	0x1B	AUD_mode	R/W	Data port to udio mode register
	0x03	VID_iaddr	R/W	Indirect address for video register access
	0x04	VID_data	W	Data for video registers
	0x0B	VID_csync	R/W	Composite sync control
Indirect[1]	0x32	DRAM_ROMTA	W	ROM access time
	0x33	DRAM_REF	W	DRAM refresh time
	0xh2	VID_WEIGHT0	W	W0 coefficient of YUV to RGB conversion
	0xh3	VID_WEIGHT1	W	W1 coefficient of YUV to RGB conversion
	0xh4	VID_WEIGHT2	W	W2 coefficient of YUV to RGB conversion
	0xh5	VID_WEIGHT3	W	W3 coefficient of YUV to RGB conversion
	0xh6	VID_WEIGHT4	W	W4 coefficient of YUV to RGB conversion
	0xh7	VID_MODE	W	Video modes
	0xhA	VID_Cr	W	Border color: Cr
	0xhB	VID_YCb	W	Border color: Y, Cb
	0xhC	VID_HSYNC_LO	W	Width of HSYNC low
	0xhD	VID_HSYNC_CYC	W	Width of HSYNC high
	0xhE	VID_VSYNCTOG	W	VSYNC toggle point
	0xhF	VID_HINT	W	Horizontal interrupt position

[1] Indirect register addresses are written to either DRAM_iaddr or VID_iaddr.

4.3 CL480 Read/write Operations

CL480 read and write operations using the host interface registers are described in the next two sections.

4.3.1 Host Write

Figure 4-7 shows typical waveforms for a host write to a host interface register. (See Section 9.2.4 for detailed timing parameters for this operation.) The sequence of events is:

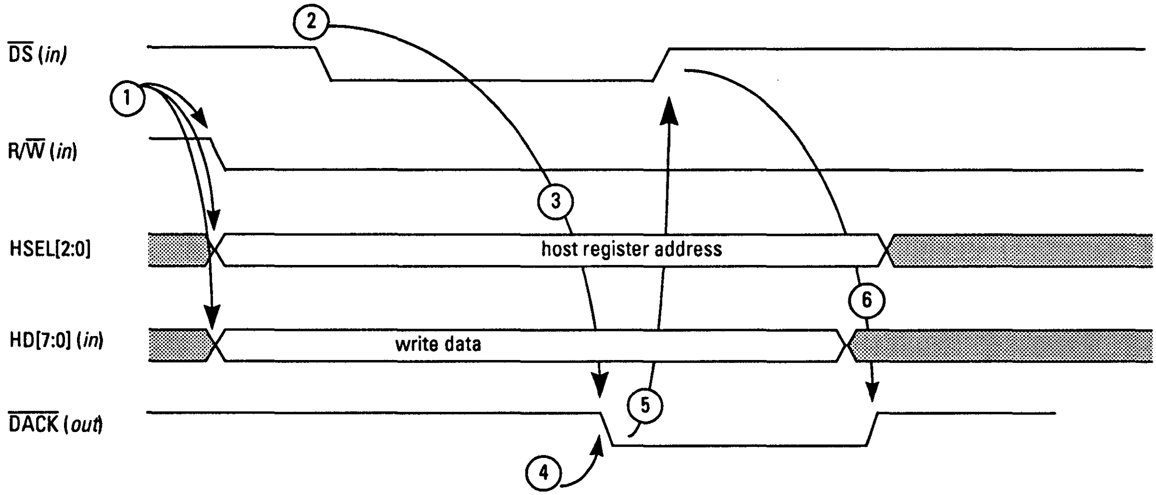


Figure 4-7 Host Interface Local Register, Write Operation

1. The host drives R/$\overline{W}$ low to indicate a write operation. It also drives the host register address onto HSEL[2:0] and the write data onto HD[7:0].
2. When the HD[7:0], R/$\overline{W}$ and HSEL[2:0] lines have settled, the host processor asserts $\overline{DS}$.
3. In response to the assertion of $\overline{DS}$, the CL480 begins the write sequence and sometime later generates a $\overline{DACK}$ signal.
4. The CL480 latches the data when $\overline{DACK}$ goes low.
5. In response to the assertion of $\overline{DACK}$, the host deasserts $\overline{DS}$.
6. The CL480 responds by releasing $\overline{DACK}$. This completes the write cycle.

4.3.2 Host Read

Figure 4-7 shows typical waveforms for a host read from a host interface register. (See Section 4.4 for detailed timing parameters for this operation.) The sequence of events is:

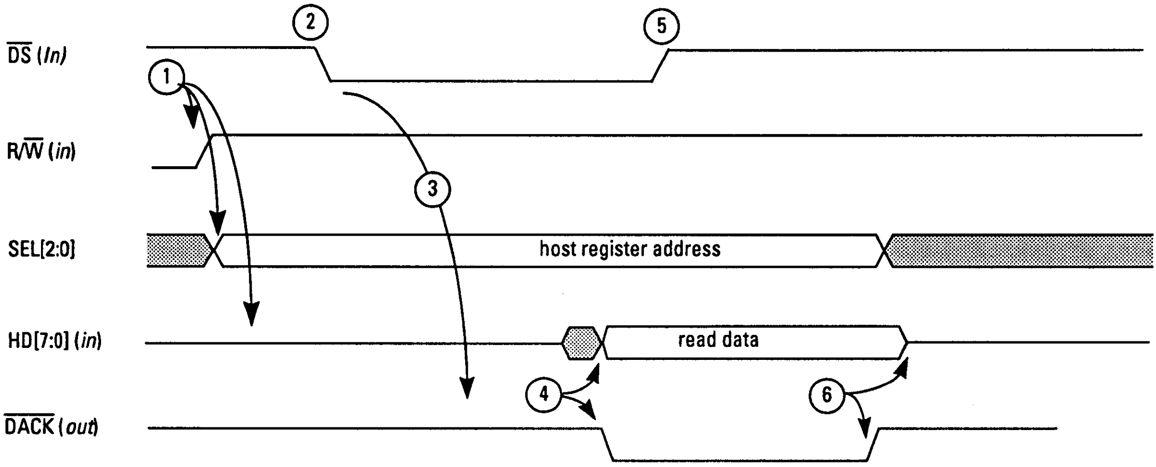


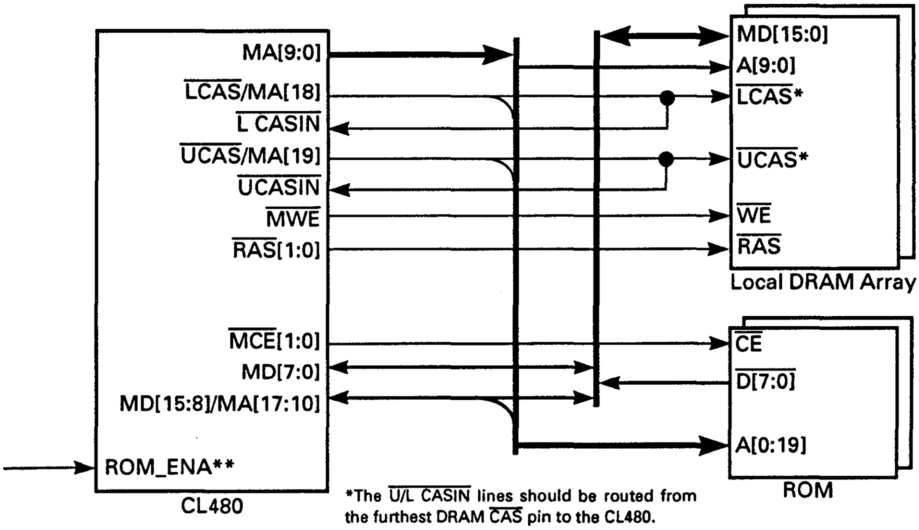
Figure 4-8 Host Interface Local Register, Read Operation

1. The host drives R/$\overline{W}$ high to indicate a read operation and drives the host register address onto HSEL[2:0].
2. When the R/$\overline{W}$ and HSEL[2:0] lines have settled, the host processor asserts $\overline{DS}$ to indicate to the CL480 that a host bus read access is in progress.
3. In response to the assertion of $\overline{DS}$, the CL480 holds $\overline{DACK}$ deasserted (high) until it can respond to the read request. Because the CL480 is performing arbitration of its internal data buses during register access, $\overline{DACK}$ assertion is delayed at least one GCK cycle.
4. When the data is available, the CL480 asserts $\overline{DACK}$ and drives the data.
5. The host deasserts $\overline{DS}$ when the data has been read.
6. In response to the deassertion of $\overline{DS}$, the CL480 releases $\overline{DACK}$ and D[7:0].

Chapter 5 DRAM/ROM Interface

This chapter describes the local DRAM/ROM Interface bus, shown in Figure 5-1, and how it is used. It details all of the signals necessary to connect the CL480 to a DRAM array and optional ROM.

The sections in this chapter are:



*The U/L CASIN lines should be routed from the furthest DRAM CAS pin to the CL480.

**This signal should be pulled high if booting from ROM.

Figure 5-1 DRAM Interface

5.1 General Description

The memory controller in the CL480 interacts directly with external memory and generates the control signals required to access external DRAMs and ROMs. The maximum transfer rate via the host is 5 Mbytes per second. The CL480 can be booted directly from ROM or the host can load microcode into DRAM.

5.1.1 DRAM: Amount and Organization

The CL480 DRAM controller can support up to one Mbyte of local Dynamic RAM; however, *only four Mbits of DRAM are needed to decode MPEG-1 bitstreams.* The DRAM array is organized as either one 256Kx16 DRAM or four 256Kx4 DRAMs.

The DRAM array is used to store:

- □ Compressed audio and video data waiting to be decoded
- □ The decoded audio and video frame that is currently being displayed
- □ Future and past decoded reference frames

- □ CL480 microcode instructions
- □ Bitstream header parameters
- □ The command FIFO

5.1.2 ROM: Amount and Organization

The CL480 can support up to two Mbytes of ROM: specifically, two ROMs, each of which can be up to 1Mx8 bits.

The required ROM size depends on the customer and what needs to be stored in ROM, some examples of which are MPEG microcode, application-specific microcode, background still images and sound data. The ROM size required for storing the CL480's MPEG microcode is less than 45 Kbytes.

The ROM access time can be programmed to be 3to 34 GCKs.

> *Note: If a ROM is connected to the CL480, the chip can initialize itself automatically after powerup. If there is no ROM, the host must write the microcode into DRAM and follow the initialization procedure described in Section 11.1.1.*

5.2 DRAM Memory Bus Interface

The DRAM memory bus on the CL480 consists of a 10-bit multiplexed address bus, a 16-bit data bus, and the control lines necessary for reading and writing the DRAM banks: Write Enable ($\overline{\text{MWE}}$), Column and Row Address Strobes ($\overline{\text{LCAS}}$, $\overline{\text{UCAS}}$ and $\overline{\text{RAS}}$), and the data latch enable signals ($\overline{\text{LCASIN}}$ and $\overline{\text{UCASIN}}$). The high output drive of the CL480 allows it to directly drive the DRAMs without external buffers or logic.

5.2.1 DRAM Address Mapping

The CL480's DRAM interface supports:

- 256Kx16 DRAMs: 10-bit row address, 8-bit column address
- 256Kx16 DRAMs: 9-bit row address, 9-bit column address

The address mapping for these two modes is shown in Table 5-1.The 9/9 DRAMs use 20-30% more power than the 10/8 DRAMs, but they are usually less expensive.

Table 5-1 DRAM Address Mapping

Configuration	MA Bits	9	8	7	6	5	4	3	2	1	0
256Kx16 8-bit Column Address	Row Address	8	17	16	15	14	13	12	11	10	9
	Col. Address	-	-	7	6	5	4	3	2	1	0
256Kx16 9-Bit Column Address	Row Address	-	17	16	15	14	13	12	11	10	9
	Col. Address	-	8	7	6	5	4	3	2	1	0

5.2.2 DRAM Interface Connections

Figure 5-2 shows how the memory bus interface is oriented when using 256Kx16 DRAMs to connect the maximum amount of external memory.

The host processor addresses the local DRAM whenever bit 4 of A_MSB is *LOW* (see Figure 4-2). MA[19] is used to select Bank 0 or Bank 1 through assertion of the $\overline{\text{RAS0}}$ or $\overline{\text{RAS1}}$ signals, respectively. Host address signals MA[18:1] map directly into the multiplexed row and column DRAM addresses. The host bus interface signal R/$\overline{\text{W}}$ maps directly to the DRAM interface $\overline{\text{MWE}}$ signal.

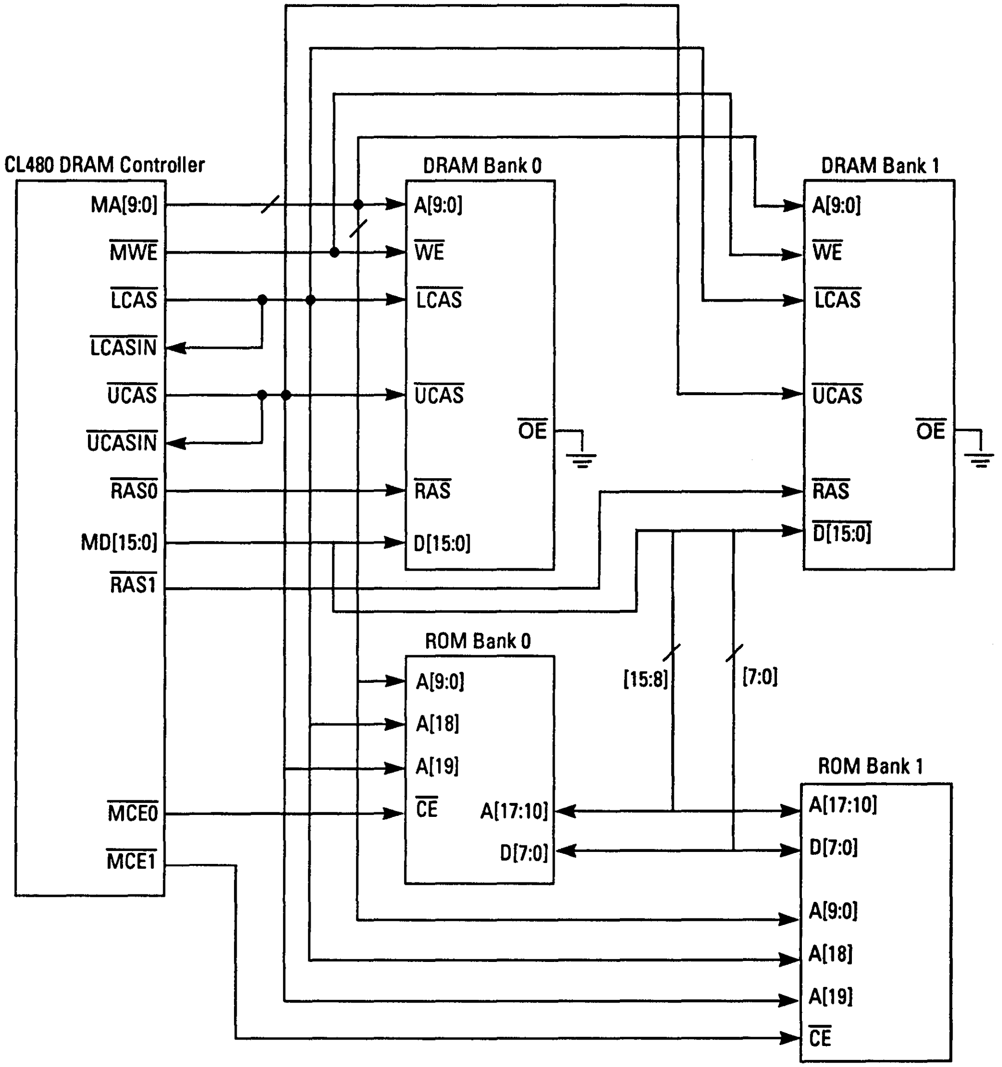


Figure 5-2 Local DRAM/ROM Implementation with 256K x 16 DRAMs

5.3 ROM Memory Bus Interface and Timing

Figure 5-3 illustrates ROM accesses. Two $\overline{\text{MCE}}$ signals are used for ROM chip enable. MD[7:0] is the ROM data bus, and MD[15:8] becomes the ROM address bits A[17:10]. A programmable number of wait cycles (3 to 34 GCKs) can be introduced between cycles 8 and 9.

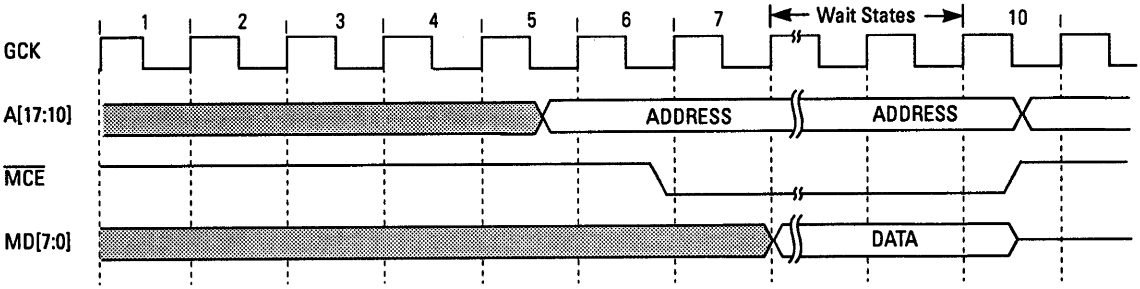


Figure 5-3 ROM Access Timing

5.3.1 DRAM_ROMTA Register

The DRAM_ROMTA register, shown below, is used to program the ROM access time. Table 5-2 shows the bit assignments.

> *Note: This register is initialized by the microcode as described in the Configuration Area of Section 12.3.*

(internal) **0x32**

15 7	6	5	4 0
Res	1	1	RTA

Field	Bits	R/W	Description
Res	15:7	R/W	These bits are reserved.
	6:5	R/W	These bits should always be 1.
RTA	4:0	R/W	ROM Access time (in number of GBUS cycles). A "0" in RTA can be used for ROM accesses (T_{CE}) of less than 3 cycles (75ns).

The *RTA* bit field defines ROM access time. Figure 5-3, which shows a four-cycle $\overline{\text{MCE}}$ low time, corresponds to *RTA*=1. A five-bit *RTA* can actually support ROMs with an access time up to 850 ns (for 40.5 MHz parts), or three cycles. The formula for defining access time is as follows:

$$ROM\ Access\ Time = (RTA + 3) * GCK\ Period$$

Since ROM access mode does not use a $\overline{\text{CASIN}}$ signal equivalent for DRAM access, the *RTA* field should specify a value which takes into consideration chip I/O delay. A safe way to do this is to define *RTA* to be one clock period more than the ROM specification required. For example, if the ROM access time is 200 ns, the *RTA* value would be 6.

5.4 DRAM Memory Bus Timing

The DRAM interface on the CL480 is designed to work with fast page-mode DRAMs. Fast page-mode DRAMs allow shorter read and write cycle periods within a DRAM row. During a set of fast page-mode accesses, a row address is strobed in using the $\overline{\text{RAS}}$ line, followed by several column addresses strobed in by $\overline{\text{CAS}}$. Releasing the $\overline{\text{RAS}}$ and $\overline{\text{CAS}}$ lines simultaneously completes the set of cycles.

Refresh is accomplished by asserting the $\overline{\text{CAS}}$ signal before the $\overline{\text{RAS}}$ signal. Refresh is performed at regular intervals determined by the value programmed into the DRAM_REF register (see page 5-12). Typical DRAM specifications require that all pages be refreshed each 8 ms.

5.4.1 DRAM Page-Mode Read Timing

Figure 5-4 shows the timing for a page-mode read. The circled numbers in the figure refer to the steps below.

Note: All DRAM accesses are synchronized to GCK.

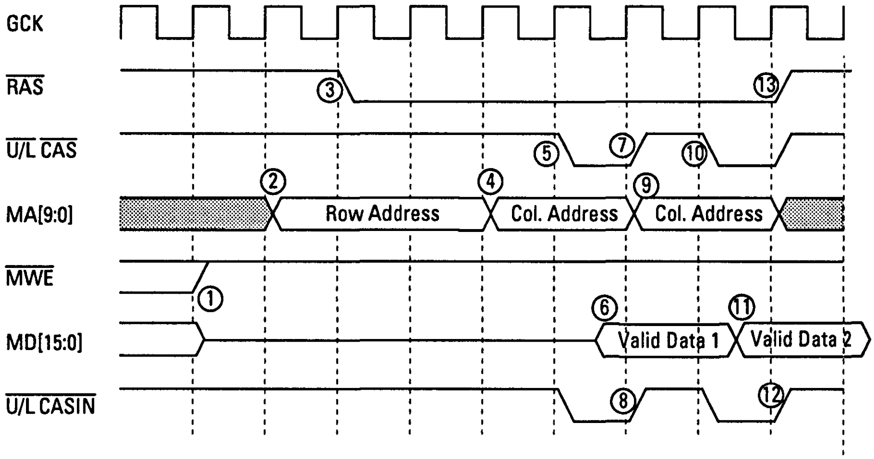


Figure 5-4 DRAM Page-Mode Read Timing

1. The CL480 starts a read cycle by three-stating the memory data

1. The CL480 starts a read cycle by three-stating the memory data bus, MD[15:0], and setting the write enable ($\overline{\text{MWE}}$) line *HIGH*.
2. The CL480 then drives the row address onto the memory address (MA) bus.
3. The CL480 asserts the $\overline{\text{RAS}}$ line *(LOW)* to latch the row address into the DRAM.
4. The CL480 then outputs the column address of the first word to be read on MA.
5. The CL480 latches the column address into the DRAM by asserting the $\overline{\text{CAS}}$ line(s) (*LOW*).
6. The DRAMs output the data from the selected address.
7. The CL480 deasserts $\overline{\text{CAS}}$.
8. The data is loaded into the input latch of the CL480 on the rising edge of $\overline{\text{CASIN}}$. For a single-location read, the CL480 ends the cycle by deasserting $\overline{\text{RAS}}$ at this time.
9. For a page-mode read cycle, the CL480 outputs the column address of the next word to be read on MA[9:0].
10. The CL480 latches the address into the DRAM by asserting the $\overline{\text{CAS}}$ line.
11. The DRAMs output the data from the selected address.
12. The data is loaded into the input latch of the CL480 on the rising edge of $\overline{\text{CASIN}}$.
13. The CL480 deasserts $\overline{\text{RAS}}$ and $\overline{\text{CAS}}$ to complete the cycle.

5.4.2 DRAM Page-Mode Write Timing

Figure 5-5 shows the timing for a page-mode write. The circled numbers in the figure refer to the steps below.

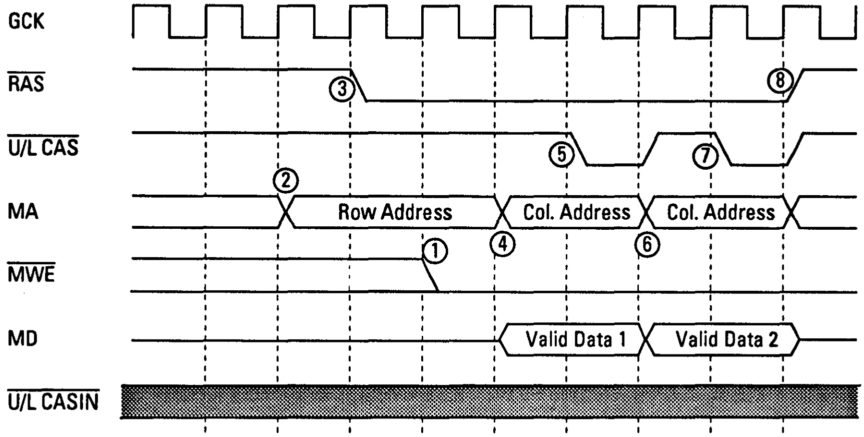


Figure 5-5 DRAM Page-Mode Write Timing

1. The CL480 starts a write cycle by asserting the write enable ($\overline{\text{MWE}}$) line *LOW*.
2. The CL480 then drives the row address on the memory address (MA) bus.
3. The CL480 asserts the $\overline{\text{RAS}}$ line (*LOW*) to latch the row address into the DRAM.
4. The CL480 then outputs the column address of the first word to be written on the MA bus, and outputs the data to be written into the first address on the MD bus.
5. The CL480 asserts the $\overline{\text{CAS}}$ line (*LOW*) to write the data into the selected location of the DRAM.
6. For a single-location write, the write operation is terminated by driving the $\overline{\text{RAS}}$ and $\overline{\text{CAS}}$ lines inactive (*HIGH*) at this point. For a page-mode write, the address and data for the next word to be written are output on the MA and MD buses, respectively.
7. The CL480 asserts the $\overline{\text{CAS}}$ line to write the data into the selected location of the DRAM.
8. When the last word is written, the write operation is terminated by driving the $\overline{\text{RAS}}$ and $\overline{\text{CAS}}$ lines HIGH.

5.4.3 DRAM Refresh Timing

Figure 5-6 shows the timing for a memory refresh cycle. The circled numbers in the figure refer to the steps below.

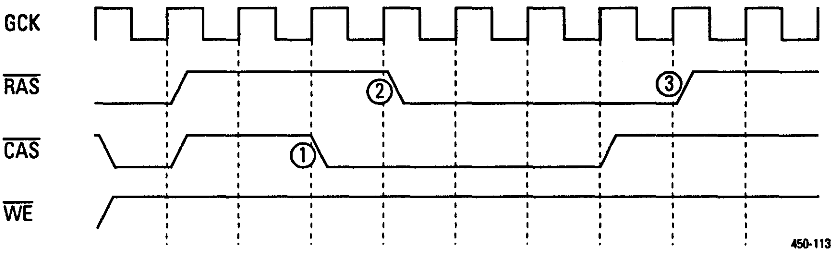


Figure 5-6 $\overline{\text{CAS}}$-before-$\overline{\text{RAS}}$ Refresh Cycle

1. The CL480 initiates a DRAM refresh cycle by asserting the $\overline{\text{CAS}}$ line while the $\overline{\text{RAS}}$ line is inactive.
2. The CL480 asserts the $\overline{\text{RAS}}$ line to enable the refresh at the address pointed to by the DRAM's internal address counter.
3. The CL480 completes the refresh cycle by releasing $\overline{\text{RAS}}$ and $\overline{\text{CAS}}$.

5.4.4 DRAM_REF Register

The DRAM_REF register, shown below, defines the time between refresh cycles.

(internal) **0x33**

15	14	13	12	0
REF	*Res*		*REF_CLK*	

Field	Bits	W	Description
REF	15	W	When asserted, this signal disables refresh.
Res	14:13	W	Reserved. 00=default value.
REF_CLK	12:0	W	Defines number of GCKs between successive refreshes.

5.5 Memory Design Guidelines

When designing DRAM arrays for use with the CL480, the following guidelines should be followed:

- During layout, it is important to keep the DRAM very close to the CL480. If possible, do not pass any other signals under the DRAM signal traces, especially the DRAM data bus.
- It is not necessary to use any termination if there is only one 256Kx16 DRAM connected to the CL480 and the DRAM signal traces are short. If four 256Kx4 DRAMs are used, series resistors (about 33 ohms) should be placed between the CL480 and the DRAM address and control signals.
- When using 256Kx16 DRAMs, connect all ten address lines to the DRAMs. This allows DRAMs with either ten row and eight column address lines or nine row and nine column address lines to be used. The tenth address bit on 256Kx16 DRAMs is located on pin 15 of SOJ packages, pin 17 of TSOP packages and pin 25 of ZIP packages. This pin has no effect on DRAMs with nine row and nine column addresses.
- When VCK is an input, GCK is typically 40 MHz, and 80-ns DRAMs can be used. When VCK is an output, GCK is typically 40.5 MHz, so that VCK will be 27 MHz. With GCK at 40.5 MHz, the two clock page-mode cycle time will be 49.4 ns, which is .6 ns outside the spec for an 80-ns DRAM. At the time this manual goes to press, a 70-ns 256Kx16 costs about $1 more than a 80-ns 256K x16. To avoid this $1 penalty, it is best to either use a 40 MHz GCK or test the systems before shipping them so that, in the extreme unlikely event that a DRAM is too slow, it will be caught before the system is shipped to the customer.
- The CL480's DRAM controller logic uses a version of the $\overline{\text{CAS}}$ signal that is routed back from the DRAM into the chip to latch data on DRAM reads. The timing and thus the routing of these signals is critical and special attention should be paid. Specifically, the $\overline{\text{CAS}}$ signals should be routed out to the DRAM farthest from the CL480 and then back to the $\overline{\text{CASIN}}$ pins. It is important NOT to route $\overline{\text{CAS}}$ directly from the $\overline{\text{CAS}}$ pin to the $\overline{\text{CASIN}}$ pin. The path followed by $\overline{\text{CASIN}}$ from the DRAMs to the CL480 should match the path followed by the data lines as closely as possible.

□ It is acceptable to have a 5V DRAM with a 3V ROM or a 3V DRAM with a 5V ROM if the VDDMAX pins are wired to 5V.

6
CD Interface

6.1 General Description

This interface, shown in Figure 6-1, is designed to receive directly the output of the CD-DSP in bit-serial mode. Serial data from the CD-DSP may be in any of the following formats: CD-DA, CD-ROM, CD-ROM/XA, CD-I, MPEG system streams, MPEG video streams, or MPEG audio streams.

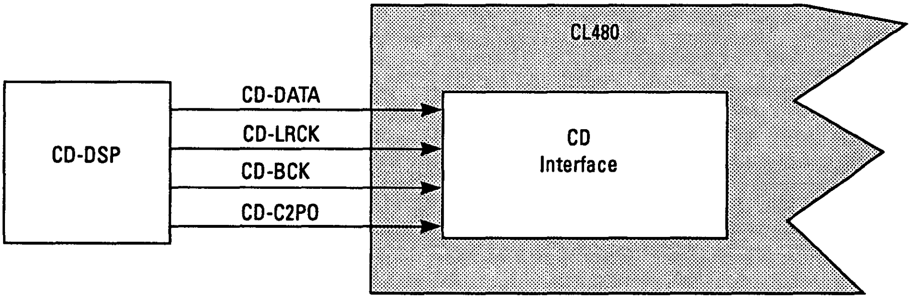


Figure 6-1 CD-Decoder Interface

6.1.1 Mode Functions

The CL480 processes data in the two basic modes described below:

- □ In ROM-Decoder mode:
  - – Real-time parsing for Mode 2 form 2 sectors
  - – Real-time parsing for Mode 1 and Mode 2 form 1 sectors (including error correction when not decoding MPEG)
  - – Real-time processing of Mode 0 and skipping to next sector
- □ In CD-DA mode:
  - – Direct output to the audio interface of the CD-DA input
  - – Ignores data error flag C2PO

The CL480 stores the sub-header and other system information related to ROM decoding into dedicated DRAM locations which are accessible to the Host.

> *Note: In a system with a host processor, the host would use the DumpData() command to read the disc info, play sequence descriptors, list ID offset tables, and other items.*

In CD-ROM decoding, the CL480 does Q-erasure correction, then P-erasure correction, followed by P-error check.

6.1.2 Mode Selection

The CL480 selects the mode in which it operates by searching for sector syncs. If none are found, the CL480 automatically switches to CD-DA mode as shown in Figure 6-2.

> *Note: For debug purposes, CD interface mode and data type can also be selected by the Host via the CD_cntl register described in Chapter 10.*

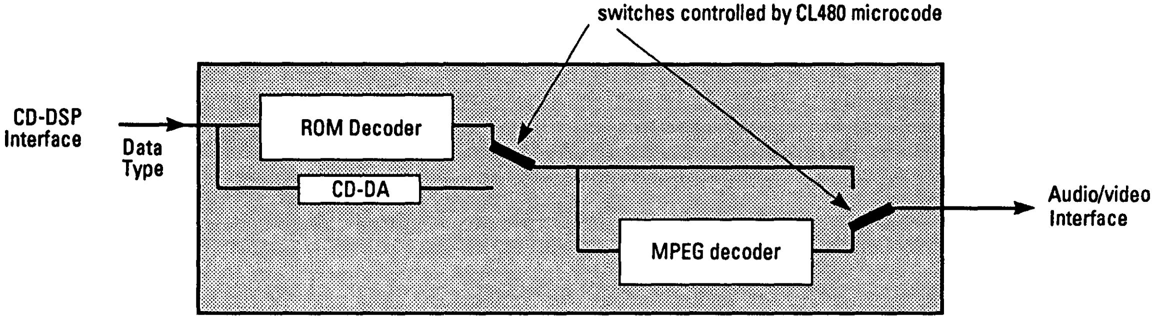


Figure 6-2 System Block Diagram Showing Bypass of ROM Decoder

6.2 Signal Interface

The CL480 CD interface takes as input the following signals, whose default settings are controlled by microcode at initialization and may be modified in the Configuration Area of the CL480 as described in Section 12.3:

- CD-DATA - The serial data input which receives the data stream in a format programmed to be either MSB or LSB first.
- CD-C2PO - This is a pin that the CD-DSP uses to tell the CL480 that a byte contains an error (error byte flag MSB or LSB first). This signal is used when receiving CD-ROM data but is ignored in CD-DA pass-through mode.
- CD-LRCK - Driven by CD-BCK, CD-LRCK provides 16-bit word synchronization. Its polarity (left/right channel high) is programmable.
- CD-BCK - The bit clock, which may be set to variable speeds according to Table 6-1.

Table 6-1 BCK/Data Rate/LRCK Comparison for the CD-Decoder

BCK	CD Speed	Data Rate	BCK per LRCK
1.42 MHz	Normal	1.42 Mbits/s	32
2.13 MHz	Normal	1.42 Mbits/s	48
2.84 MHz	Normal	1.42 Mbits/s	64
2.84 MHz	Double	2.84 Mbits/s	32
4.26 MHz	Double	2.84 Mbits/s	48
5.84 MHz	Double	2.84 Mbits/s	64

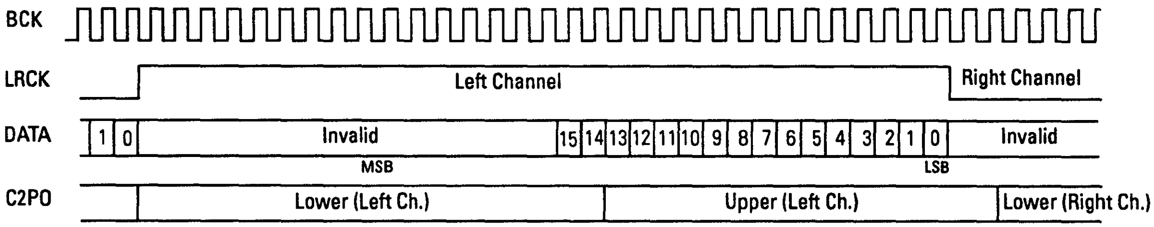
Note: For debug purposes, CD interface signals can also be regulated by the Host via the CD_cntl register described in Chapter 10.

6.3 Disc and Signal Formats

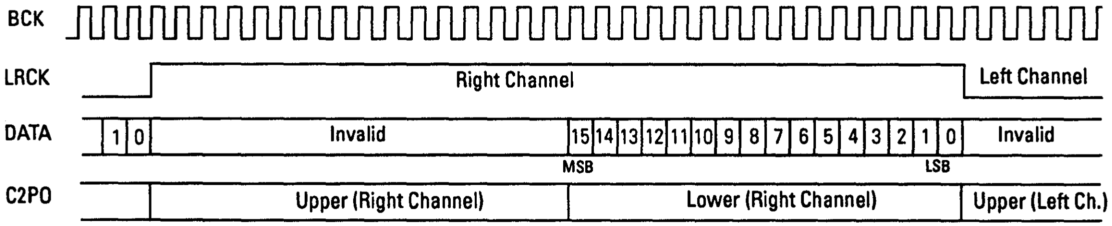
The next three sections summarize CL480 input data and disc sector formats.

6.3.1 CD Interface Input Signal Format

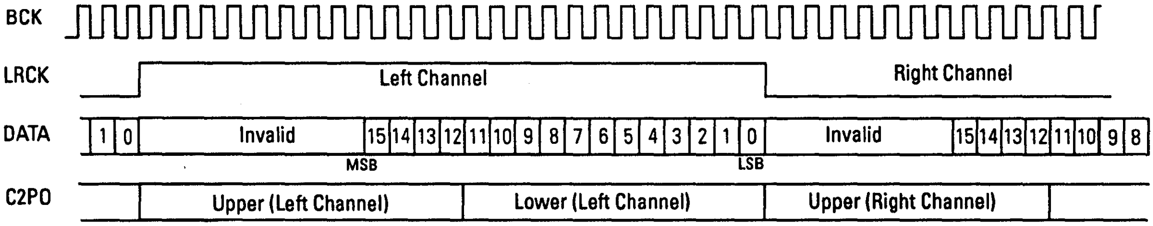
The input signal formats of the six different serial modes of the CD interface are shown below and on the following page.



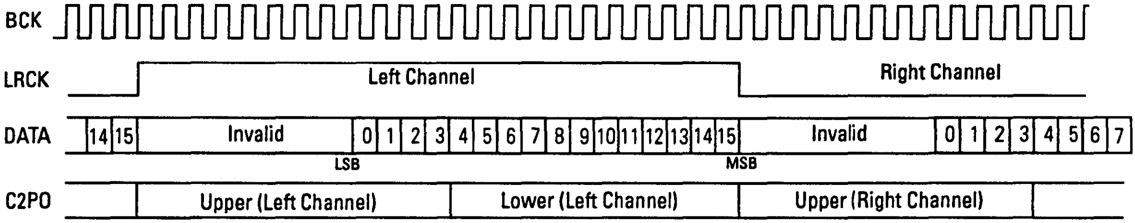
32-bit BCK, MSB First, Right Channel Low, C2PO LSB First, Data latch timing high



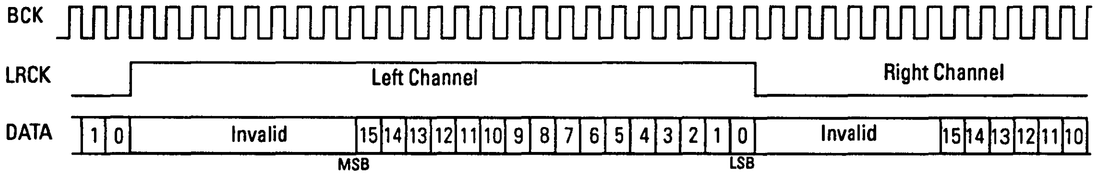
32-bit BCK, MSB First, Left Channel Low, C2PO MSB First, Data latch timing low



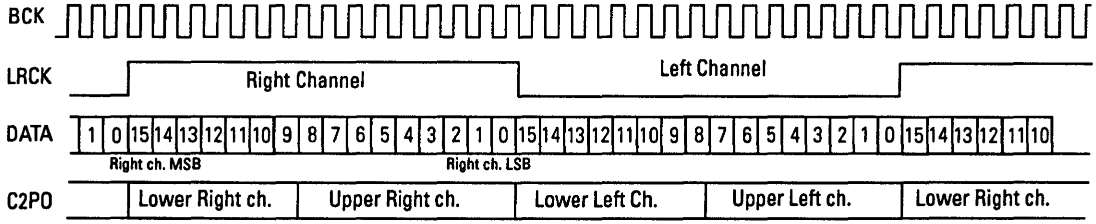
24-bit BCK, MSB First, Right Channel Low, C2PO MSB First, Data latch timing high



24-bit BCK, LSB First, Right Channel Low, C2PO MSB First, Data latch timing low



24-bit BCK, MSB First, Right Channel Low, Data latch timing high (Note: no C2PO for this format)

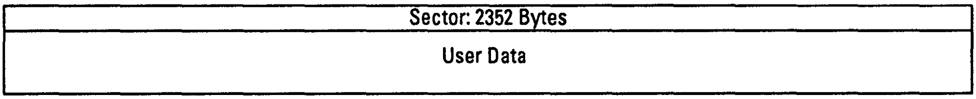


16-bit BCK, MSB First, Left Channel Low, C2PO LSB First, Data latch timing high

6.3.2 Compact Disc Sector Format

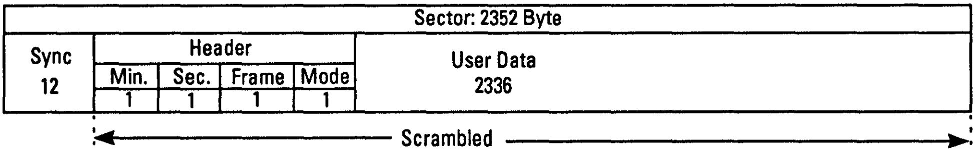
Serial data CD formats are listed below:

CD-DA



Sector: 2352 Bytes
User Data

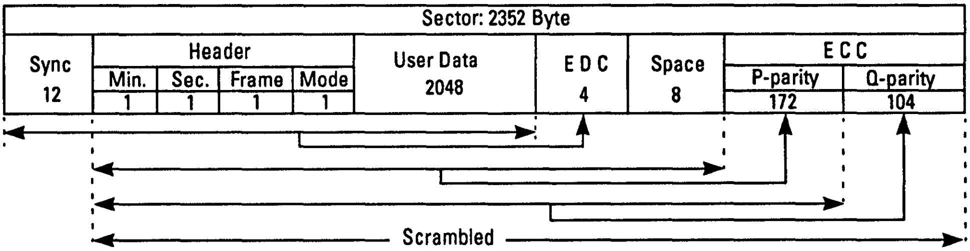
CD-ROM: MODE0, MODE2



Sector: 2352 Byte					
Sync	Header				User Data
	Min.	Sec.	Frame	Mode	
12	1	1	1	1	2336

Note: Mode byte must be set to 0x00 or 0x02. If set to 0x00, then user data area is stuffed with 0x00.

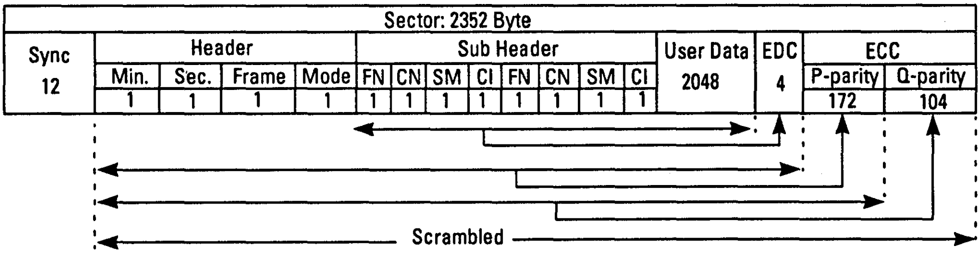
CD-ROM: MODE1



Sector: 2352 Byte									
Sync	Header				User Data	E D C	Space	E C C	
	Min.	Sec.	Frame	Mode				P-parity	Q-parity
12	1	1	1	1	2048	4	8	172	104

Note: Mode byte must be set to 0x01. If set to 0x00. Space area is stuffed with 0x00.

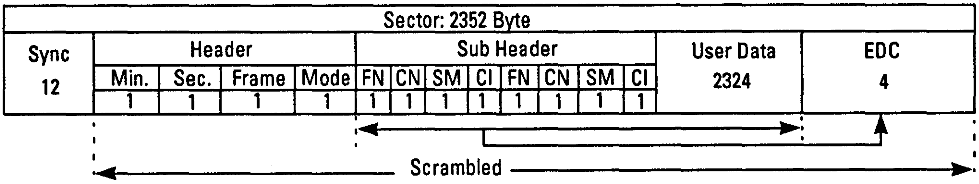
CD-ROM/XA, CDI MODE2-Form1



Sector: 2352 Byte																
Sync	Header				Sub Header								User Data	EDC	ECC	
	Min.	Sec.	Frame	Mode	FN	CN	SM	CI	FN	CN	SM	CI			P-parity	Q-parity
12	1	1	1	1	1	1	1	1	1	1	1	1	2048	4	172	104

Note: Mode byte must be set to 0x02. FN= file number, CN= channel number, SM= submode, CI= coding information

CD-ROM/XA, CDI MODE2-Form2



Sector: 2352 Byte														
Sync	Header				Sub Header								User Data	EDC
	Min.	Sec.	Frame	Mode	FN	CN	SM	CI	FN	CN	SM	CI		
12	1	1	1	1	1	1	1	1	1	1	1	1	2324	4

Note: Mode byte must be set to 0x02. FN=file number, CN= channel number, SM= submode, CI= coding information

6.3.3 Data Alignment for CD-ROM

<table>
<tr><td rowspan="5">CD-DA</td><td colspan="16">Left Channel</td><td colspan="16">Right Channel</td></tr>
<tr><td colspan="8">LSBit</td><td colspan="8">MSBit</td><td colspan="8">LSBit</td><td colspan="8">MSBit</td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td></tr>
<tr><td colspan="16">Even Word</td><td colspan="16">Odd Word</td></tr>
<tr><td colspan="8">LSByte</td><td colspan="8">MSByte</td><td colspan="8">LSByte</td><td colspan="8">MSByte</td></tr>
<tr><td rowspan="9">CD-ROM</td><td colspan="8">0x00</td><td colspan="8">0xFF</td><td colspan="8">0xFF</td><td colspan="8">0xFF</td></tr>
<tr><td colspan="8">0xFF</td><td colspan="8">0xFF</td><td colspan="8">0xFF</td><td colspan="8">0xFF</td></tr>
<tr><td colspan="8">0xFF</td><td colspan="8">0xFF</td><td colspan="8">0xFF</td><td colspan="8">0x00</td></tr>
<tr><td colspan="8">Minutes</td><td colspan="8">Seconds</td><td colspan="8">Frames</td><td colspan="8">Mode</td></tr>
<tr><td colspan="8">Byte 16</td><td colspan="8">Byte 17</td><td colspan="8">Byte 18</td><td colspan="8">Byte 19</td></tr>
<tr><td colspan="8">Byte 20</td><td colspan="8">Byte 21</td><td colspan="8">Byte 22</td><td colspan="8">Byte 23</td></tr>
<tr><td colspan="8">⋮</td><td colspan="8">⋮</td><td colspan="8">⋮</td><td colspan="8">⋮</td></tr>
<tr><td colspan="8">Byte 2344</td><td colspan="8">Byte 2345</td><td colspan="8">Byte 2346</td><td colspan="8">Byte 2347</td></tr>
<tr><td colspan="8">Byte 2348</td><td colspan="8">Byte 2349</td><td colspan="8">Byte 2350</td><td colspan="8">Byte 2351</td></tr>
</table>

6.4 Design and Operation Considerations

CD operation and design guidelines are provided in this section.

6.4.1 Effects of Varying CD-DSP Signal Speed

The CL480 uses DA-XCK to clock the system clock reference (SCR) timer. If the DA-XCK frequency is increased and the disc spins faster, the SCR timer will run faster so the microcode will skip pictures to maintain audio/video synchronization and prevent bitstream buffer overflow. Similarly, if the DA-XCK frequency is decreased, the SCR timer will run slower so the microcode will repeat pictures to maintain audio/video synchronization and prevent bitstream buffer underflow. Pitch control can be done by changing the DA-XCK frequency and changing the speed that the disc spins. The CL480 will automatically skip or repeat pictures to maintain audio/video synchronization.

6.4.2 Maximum BCK Frequency

The maximum instantaneous BCK frequency on the 480, assuming the sustained rate doesn't exceed the double speed CD rates, is as follows: If the BCK period is a constant, the BCK frequency can be up to 1/6th the GCK frequency. With a GCK of 40 MHz, a burst rate of up to 6.6 Mbits/sec could be input using a constant BCK period. If you are willing to make a BCK period that is not fixed, BCK can run at the GCK frequency until the end of the word, then it needs to be high for 3 GCKs and low for 3 GCKs. This second method would give a burst input rate of 30.4 Mbits/sec. These burst rates can be used until the CFIFO is full.

6.4.3 Maximum Rate for Error Correction

The maximum data rate that the CL480 can error correct is 2.8 Mbits/sec, which is the double-speed CD rate. This error correction can only be done when not decoding MPEG.

6.4.4 ROM Acceses during MPEG Decode and Display

The ROM/DRAM accesses by the host during MPEG decoding should be a small percent of the total memory clocks. For example, if a 200ns ROM is used, the ROM access time would be programmed to 9 GCKs per 8-bits.

Chapter 7
Video Display Interface

This section of the user's manual covers the operation of the video display unit.The purpose of the video display unit is to output the decompressed video data in a form that can be either displayed on a television or video monitor, or mixed with computer-generated graphics to provide a video signal within a graphics window.

The CL480 video display unit horizontally and vertically interpolates decoded video and optionally converts pixel data from YCbCr to RGB. Two different fields are interpolated from each decoded frame. The RGB converter can make the range of each color component either 0 to 255 (for computer displays) or 16 to 235 (for television). The video clock (VCK) can either be input or can be generated by dividing GCK by 3/2. A composite sync (CSYNC) can be output in place of a $\overline{\text{VSYNC}}$ output.

The sections in this chapter are:

- □ 7.1 Digital Video Standards
- □ 7.2 Video Display Unit
- □ 7.5 Timing Generation
- □ 7.4 Operation and Design Guidelines

7.1 Digital Video Standards

There are two primary standards for video transmission used in the world today:

- □ NTSC (National Television Systems Committee) standard - used by North America, Japan, Korea, and Taiwan.
- □ PAL (Phase Alternating Line) standard - used by Europe, China, India, Australia, and most of Africa.

The NTSC video standard specifies a scanning system of 525 horizontal lines per frame. Each frame consists of two interlaced fields of 262.5 horizontal lines. The field rate is 59.94 Hz.

The start of a field is synchronized with a vertical sync pulse, followed by an interval of 22.5 lines with no video, called *vertical blanking*. This leaves 240 lines in a field for picture information (480 in a frame).

The horizontal lines are synchronized with a horizontal synchronization pulse followed by an interval with no video, called *horizontal blanking*. Horizontal synchronization pulses occur at a rate of 15,734 Hz.

PAL video operates in much the same way, with these differences:

- □ The field rate is 50 Hz.
- □ There are 625 lines in a frame and 312.5 lines in a field.
- □ The horizontal sync pulses occur at a rate of 15,626 Hz.
- □ There are 576 lines of active video in a PAL frame and 288 lines in a field.

The SIF (Source Input Format) specification reflects these standards by defining two digital video formats: 352 pixels x 240 lines x 30 Hz for compatibility with NTSC, and 352 pixels x 288 lines x 25 Hz for compatibility with PAL.

Video information is represented using two data formats:

- □ YCbCr data - consists of a Y value for the luminance, or brightness, of a pixel and two color values, Cb and Cr, for the chrominance.
- □ RGB (red, green, blue) - the data output that has a digital value that corresponds to the amplitude of the red, green, and blue signals.

7.2 Video Display Unit

In accordance with the MPEG standard, CL480 compressed video data is always stored in YCbCr format. If RGB-encoded output video is needed, the CL480's programmable color-space converter (set in the Configuration Area of DRAM; see Section 12.1) will produce RGB in the range of either 16-235 or 0-255, in which 16-16-16 and 0-0-0 RGB indicates the level of black, while 235-235-235 and 255-255-255 RGB indicates the level of white.

The CL480's video display unit accepts decompressed YCbCr data from the Local DRAM and outputs it as horizontally and vertically interpolated YCbCr or RGB pixels. The pixels are clocked out by the VCK (video clock) signal, and are synchronized to external devices using the $\overline{\mathrm{HSYNC}}$ (Horizontal Sync) and $\overline{\mathrm{VSYNC}}$ (Vertical Sync) signals.

7.3 Timing Generation

The CL480 outputs pixels either in YCbCr (4:2:2 mode) or RGB (4:4:4 mode) using a video clock (VCK) that may be programmed to be generated either externally or internally. (VCK is programmed by setting values in the Configuration Area of DRAM; see Section 12.3.)

The CL480 supports the following VCK modes:

- □ *24-bit RGB*: In this mode, the B component is presented as the MSB byte and the R component as the LSB byte.
- □ *16-bit YCbCr in 4:2:2*: In this mode, the Y component is presented alternately with the Cb and Cr components in the repeating pattern of CbY-CrY.
- □ *8-bit YCbCr in 4:2:2*: In this mode, the Cb, Y and Cr components is presented in successive order in the repeating pattern of Cb-Y-Cr-Y.

VD[23:0] pin assignments are given in Table 7-1 and shown with their respective timing parameters in Figure 7-1.

Table 7-1 VD[23:0] Pin Assignments for RGB and YCbCr Data

VD[23:0]	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
24-bit RGB	B7	B6	B5	B4	B3	B2	B1	B0	G7	G6	G5	G4	G3	G2	G1	G0	R7	R6	R5	R4	R3	R2	R1	R0
16-bit YCbCr[1]	C4	C3	C2	C1	C0			Y7	Y1	Y0	C7	C6	C5				Y6	Y5	Y4	Y3	Y2			
8-bit YCbCr[2]								YC7	YC1	YC0							YC6	YC5	YC4	YC3	YC2			

[1] Note: In ES1, 16-bit YCbCr data, Y[7:0] is on VD[15:8] and C[7:0] is on VD[7:0].

[2] Note: In ES1, 8-bit YCbCr data, YC[7:0] is on VD[15:8]. In ES2, it is as shown above.

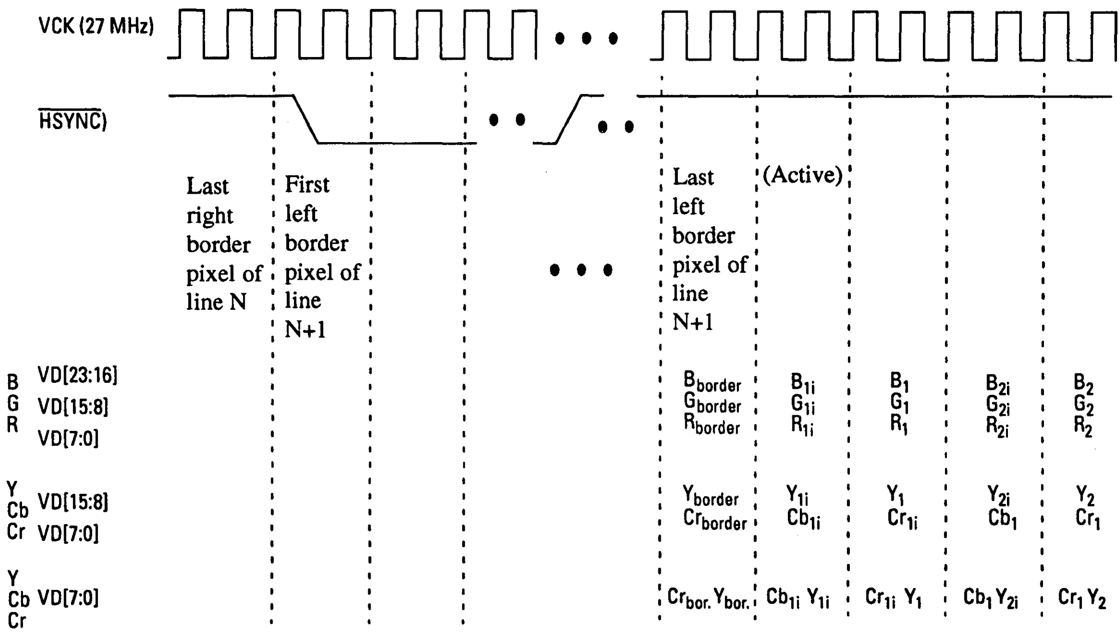


Figure 7-1 VCK versus HSYNC

Data changes in the VCK period where HSYNC falls and every other VCK after that. The VCK-to-VD delay allows VD to be latched with the rising edge of VCK from either or both VCK periods.

The HSYNC/VSYNC direction (in or out) and the VCK direction (in or out) are independently selectable. HSYNC and VSYNC are always in the same direction. This makes possible the following four video signal configurations:

- □ (VSYNC or CSYNC)/HSYNC/VCK Out
- □ VSYNC/HSYNC/VCK In
- □ (VSYNC or CSYNC)/HSYNC Out and VCK In
- □ VSYNC/HSYNC In and VCK Out

Two examples of these configurations are shown in the next sections.

> *Note: CSYNC (composite SYNC) may be selected instead of VSYNC and is provided on the VSYNC pin. This feature is microcode-selectable or may be enabled by the VID_csync register.*

7.3.1 ($\overline{\text{VSYNC}}$ or CSYNC)/$\overline{\text{HSYNC}}$/VCK Out

In this mode, as shown in Figure 7-1, ($\overline{\text{VSYNC}}$ or CSYNC), $\overline{\text{HSYNC}}$ and VCK are generated internally and sent to the NTSC/PAL encoder. In this configuration, VCK is fixed to 27 MHz and is obtained by dividing GCK (40.5 MHz) by 3/2.

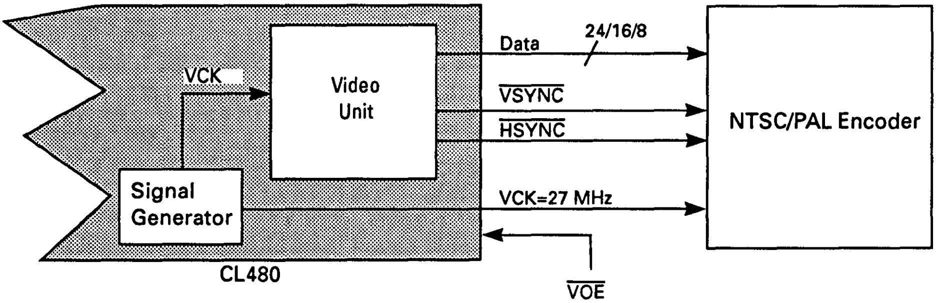


Figure 7-2 $\overline{\text{VSYNC}}$/$\overline{\text{HSYNC}}$/VCK Out Mode

7.3.2 $\overline{\text{VSYNC}}$/$\overline{\text{HSYNC}}$/VCK In

In this mode, as shown in Figure 7-2, the video timing signals—VCK, $\overline{\text{VSYNC}}$ and $\overline{\text{HSYNC}}$—are generated by the external signal generator. VCK and GCK can be asynchronous.

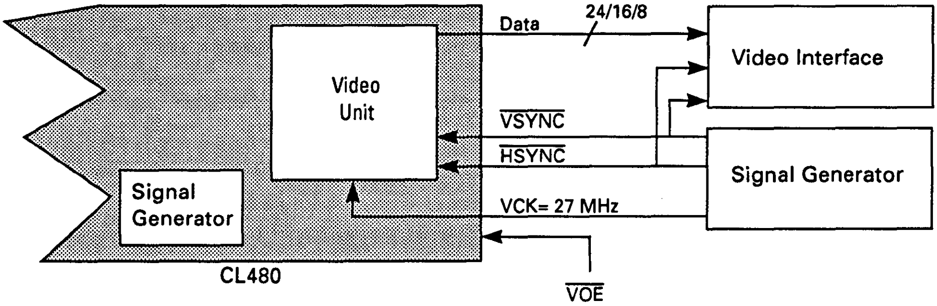


Figure 7-3 $\overline{\text{VSYNC}}$/$\overline{\text{HSYNC}}$/VCK In

7.4 Operation and Design Guidelines

Video operation and design guidelines are provided in this section.

7.4.1 How to Set Video Output Mode

The selection between 24-bit RGB, 15-bit RGB, 16-bit YCbCr and 8-bit YCbCr mode is done by microcode that reads a DRAM location. The selection between VCK in or VCK out is also done by microcode. C-Cube currently has two sets of microcode: one for $\overline{\text{VSYNC/HSYNC}}$ in and one for $\overline{\text{VSYNC/HSYNC}}$ out. Also, the 15-bit RGB mode is a dithered version of the 24-bit RGB mode with 5 bits each for R, G and B. The 15-bit RGB mode is intended for systems that write the CL480 output into a 16-bit wide memory before outputing RGB. Examples of systems like this are PC add-in cards and game boxes.

7.4.2 Aspect Ratio Limitations

The CL480 does not change the aspect ratio of the decoded pixels other than interpolating horizontally by 2. To make a disc that looks acceptable on both NTSC and PAL TVs, the coded frames can use an aspect ratio halfway between the aspect ratio for NTSC and PAL.

7.4.3 CD-I Pixel Output

For 384-pixel-wide CD-I format, the CL480 can output 384 SIF pixels/line or 768 interpolated pixels/line. It can also display a window of 352 SIF pixels/line or 704 interpolated pixels/line.

7.4.4 Transferring Video Data Via Host Bus

The CL480 does not have an internal bus between the output of the video unit and the host interface. When performance is not a concern, such as during diagnostics, it is possible to read decoded frames out of DRAM from the host interface. Decoded frames read from DRAM will be in 4:2:0 YCbCr format. DRAM transfers from the host interface do not use page mode cycles so it is not possible for the current chip to decode SIF resolution MPEG in real-time while transferring the results out the host interface.

7.4.5 Vertical Interpolation and Full-Screen Playback Mode

Progressive scan mode (vertical interpolation enabled) can support 480 or 576 line playback, but it will only output 480 lines 30 times per second or 576 lines 25 times per second.

8
Audio Interface

8.1 General Description

This interface is designed to output Pulse Code Modulation (PCM) samples in bit-serial format directly from the CL480 to the audio DACs, as shown in Figure 8-1.

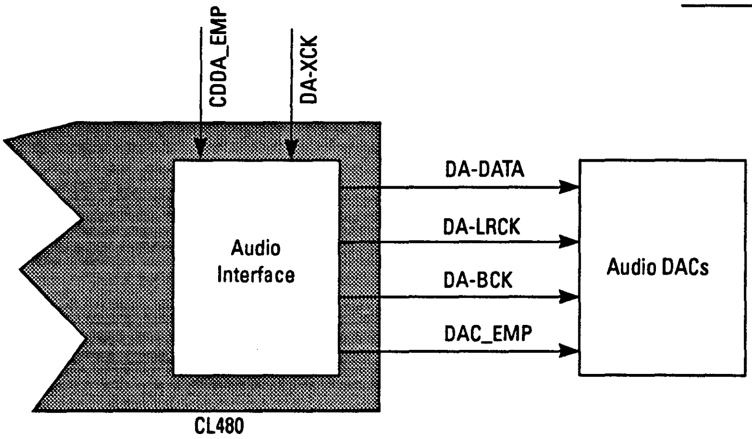


Figure 8-1 Audio Interface Block Diagram

The audio interface supports several serial data output modes that may be selected in the microcode Configuration Area or by writing to the AUD_mode register. The following parameters may be set: Left_sel, Right_sel, Out_en, Sel_24, and LSB_first.

8.2 Signal Interface

Table 8-1 summarizes the functions of the CL480 audio signals.

Table 8-1 Audio Interface Signals

Signal	Direction	Description
DA-DATA	Out	Outputs audio samples based on the DA-BCK clock.
DA-BCK	Out	The audio bit clock. Divided by 8 from DA-XCK, it can be either 48 or 32 times the sampling clock.
DA-LRCK	Out	The clock that identifies the channel for each audio sample. The polarity of this signal is programmable by microcode or by setting bits in the AUD_mode register.
DAC_EMP	Out	IN CD-DA pass-through mode, this signal is routed to the audio DACs.
DA-XCK	In	External audio frequency clock used to generate DA-BCK and DA-LRCK, DA-XCK can be programmed to be either 384 or 256 times the sampling frequency.
CDDA_EMP	In	Emphasis flag input used in CD-DA mode.

8.2.1 Controlling the Audio Output Level

In addition to its standard clock and data signals, the audio interface uses the SetMute() macro command to control the attenuation of the PCM samples. (See SetMute() description on page 12-28.)

The same attenuation function performed by SetMute() may be achieved by disabling the host interface (HOST_ENA=0), whereupon pins HD[0] and HD[1] become MUTE0 and MUTE1, respectively, as shown in Table 8-2.

Table 8-2 Audio Attenuation Based on SetMute() or HOST_ENA=0

MUTE0	MUTE1	Attenuation on the PCM Samples
0	0	0 dB
1	0	-12 dB
0	1	-∞ (20 ms soft mute)
1	1	-12 dB

8.2.2 Using Emphasis Flags to Identify the Data Stream

In CD-DA pass-through mode, the CD-DA emphasis signal is routed to the emphasis output. In MPEG audio decoding, the signal is driven with the emphasis indicated in the MPEG audio stream.

8.2.3 Low-Power Operation

The CL480 operates in two automatic power reduction modes. When decoding stereo MPEG audio only (no video), the GCK is divided by 8 and the power consumption on the CL480 is about 0.1W. When in CD-DA pass-through mode, GCK is divided by 16 and the power consumption is about 0.05W.

8.3 Clocking Modes

The Audio Interface Unit can support sampling frequencies of 32 kHz, 44.1 kHz and 48 kHz and takes as input an external source, DA-XCK, that is used to generate DA-BCK and DA-LRCK.

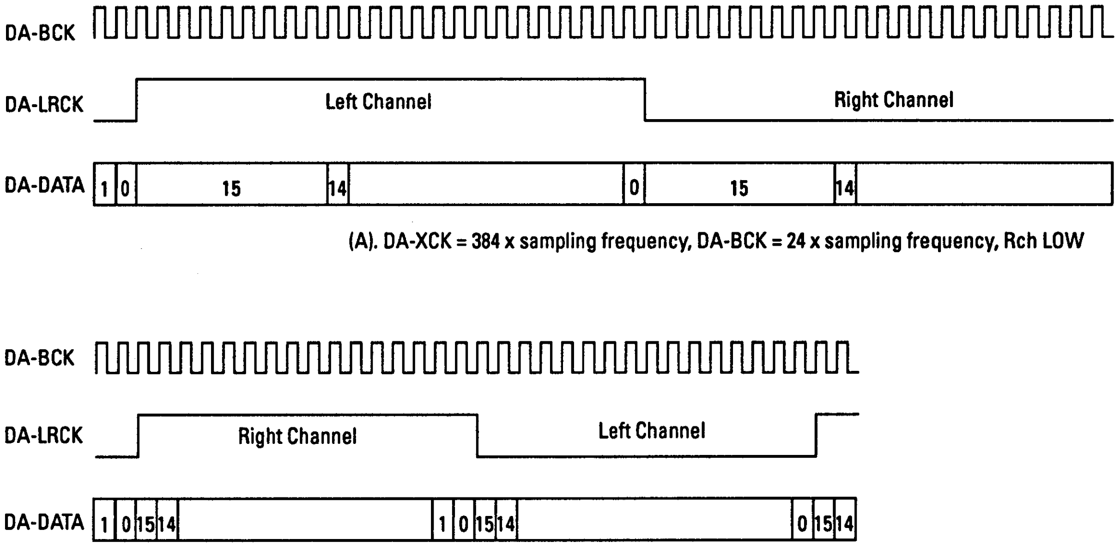
The external reference clock, DA-XCK, can be either 384 or 256 times the sampling frequency. The bit clock, DA-BCK, is derived from the external reference clock, DA-XCK, by dividing by 8. This gives a bit clock value of either 48 or 32 times the sampling clock.

The left-right channel clock, DA-LRCK, identifies the channel for each audio sample. The polarity of this signal is programmable using microcode or the AUD_mode register. Table 8-3 shows the relationship between XCK, BCK and LRCK for a sampling frequency (SF) of 44.1 KHz for the two modes of operation: 384 x SF and 256 x SF.

Table 8-3 Relationship between XCK, BCK and LRCK for 44.1 KHz

XCK	BCK	LRCK
384 x SF=16.93 MHz	XCK/8 = 2.1168 MHz	BCK/48=44.1 KHz
256 x SF=11.29 MHz	XCK/8=1.4112 MHz	BCK/32=44.1 KHz

In 384 x SF mode, as shown in diagram A of Figure 8-2, the interface timing for MSB is extended for eight additional DA-BCK clocks with DA-LRCK HIGH denoting the left channel and DA-LRCK LOW denoting the right channel. In 256 x SF mode (diagram B), DA-LRCK LOW denotes the left channel and DA-LRCK HIGH denotes the right channel.



(A). DA-XCK = 384 x sampling frequency, DA-BCK = 24 x sampling frequency, Rch LOW

(B). DA-XCK = 256 x sampling frequency, DA-BCK = 16 x sampling frequency, Lch LOW

Figure 8-2 Audio Interface Clocking Modes

8.4 Operation and Design Guidelines

Audio operation and design guidelines are provided in this section.

8.4.1 Audio Buffer Underflow

When the audio output buffer underflows, as it would if there is no audio bitstream, the audio output stays at the value of the last audio sample until the audio interface is reset.

8.4.2 DA-XCK and Cycle Jitters

If a frequency multiplier is used to create DA-XCK from CD-BCK, the jitter will not cause any problem for the CL480. However, the jitter might cause a problem for a 1-bit audio DAC. C-Cube's current microcode expects DA-XCK, CD-BCK and CD-LRCK to all be derived from the same clock so that the frequency of these signals drifts together. The bitstream buffers could overflow or underflow if these signals are not derived from the same crystal. C-Cube plans to remove this requirement in a future microcode release.

9 Electrical and Physical Specifications

This chapter describes the CL480's electrical and mechanical characteristics. The chapter is divided into three sections:

- 9.1: Operating Conditions
- 9.2: AC Timing Characteristics
- 9.3: Package Specifications

9.1 Operating Conditions

This section specifies the electrical characteristics of the CL480.

Table 9-1 Absolute Maximum Ratings[1]

Parameter	Value
Supply Voltage (V_{DD})	-0.3 to 4.5 V
Input Voltage	-0.3 to 5.5 V
Output Voltage	-0.3 to (V_{DD3} +0.5)
Storage temperature range	-55°C to 150°C
Operating temperature range (ambient)	-10°C to 70°C
Reflow Soldering temperature	240°C for 5 Seconds Maximum

[1] Exposure to stresses beyond those listed in this table may result in device unreliability, permanent damage, or both.

Table 9-2 Operating Conditions

Parameters		Commercial		Unit
		Min	Max	
V_{DD3}	Supply Voltage	2.7	3.6	V
V_{DDMAX}	VDDMAX pin (max input voltage)	V_{DD3}	5.25	V

Table 9-3 DC Characteristics

Parameters		Test Conditions	Commercial			Unit
			Min	Typ	Max	
V_{IH}	High-level input voltage[1]	V_{DD} = MAX	2.4			V
V_{IL}	Low-level input voltage[1]	V_{DD} = MIN			0.8	V
V_{OH}	High-level output voltage	V_{DD} = MIN, I_{OH} = -2 mA	2.4			V
V_{OL}	Low-level output voltage	V_{DD} = MIN, I_{OL} = 2 mA			0.5	V
I_{IH}	High-level input current	V_{DD} = MAX, HostV$_{DD}$ = MAX DRAMV$_{DD}$ = MAX			10	µA
I_{IL}	Low-level input current	V_{DD} = MAX, V_{IN} = 0 V	-10			µA
I_{OZ}	Output leakage current	Hi-Z output driven to 0V and 5.25 V	-10		+10	µA
I_{DD27}	Supply Current[2] @ V_{DD3} = 2.7V	GCK = 40MHz V_{IN} = 0 or 2.7V, C_L=50pF, Ta=70°C		150		mA
I_{DD33}	Supply Current @ V_{DD3} = 3.3V	GCK = 40MHz V_{IN} = 0 or 3.3V, C_L=50pF, Ta=25°C		180		mA
I_{DD36}	Supply Current[2] @ V_{DD3} = 3.6V	GCK = 40MHz V_{IN} = 0 or 3.6V, C_L=50pF, Ta=70°C		200		mA
C_{IN}	Input Capacitance[1]			10		pF
C_{OUT}	Output Capacitance[1]			12		pF
CI/O	Output Capacitance[1]			12		pF

[1] Not 100% tested, guaranteed by design characterization

[2] Supply current is a few percent lower at -10°C than 70°C.

9.1.1 Duty Cycle

The CL480 was designed for GCK and VCK duty cycles of 55/45.

9.1.2 Power Pin Supply Voltages

Pins 7, 29, 49, 65, 77, 81, 98, 110, 112, 21, 22 should be connected to 2.7 to 3.6V. The maximum voltage output by the CL480 is the voltage on these pins. Pins 43 and 123 should be connected to the maximum voltage that any of the inputs or I/Os will receive. For example, if a 5V DRAM and a 3V host are being used, connect both these pins to 5V.

9.1.3 Power-up and Power-down Constraints

During power-up and power-down, the 5V supply voltage should always be at a higher voltage than the 3.3V supply voltage. The 5V supply is connected to the N well of the P channel output transistor in the I/O pads. The source terminal of this transistor is connected to 3.3V. If the 3.3V supply is more than about .6V above the 5V supply, the diode between the P diffusion and the N well will conduct current, latching up the chip or destroying the bond wires for the 5V supply.

9.2 AC Timing Characteristics

This section describes the AC timing characteristics of the CL480. The timing characteristics are divided into related groups.

- □ Host Bus Interface Timing
  - □ Figure 9-1, Host Interface Local Register, Write Operation
  - □ Table 9-4, Timing Characteristics, Local Register Write
  - □ Figure 9-2, Host Interface Local Register, Read Operation
  - □ Table 9-5, Timing Characteristics - Local Register Read
- □ DRAM/ROM Bus Timing
  - □ Figure 9-3, Local DRAM Bus Timing
  - □ Figure 9-4, Local DRAM $\overline{\text{CAS}}$-before-$\overline{\text{RAS}}$ Refresh
  - □ Table 9-6, Timing Characteristics - Local DRAM Bus
- □ CD Interface Timing
  - □ Figure 9-5, CD Input: 32-bit CD-BCK, Data Latch Timing High
  - □ Figure 9-6, CD Input: 32-bit CD-BCK, Data Latch Timing Low
  - □ Figure 9-7, CD Input: 24-bit CD-BCK, Data Latch Timing High
  - □ Figure 9-8, CD Input: 24-bit CD-BCK, Data Latch Timing Low
  - □ Figure 9-9, CD Input: 16-bit CD-BCK, Data Latch Timing High
  - □ Table 9-7, CD Input Timing

9.2.1 Host Bus Interface Timing

This section describes the host-related AC timing characteristics of the CL480.

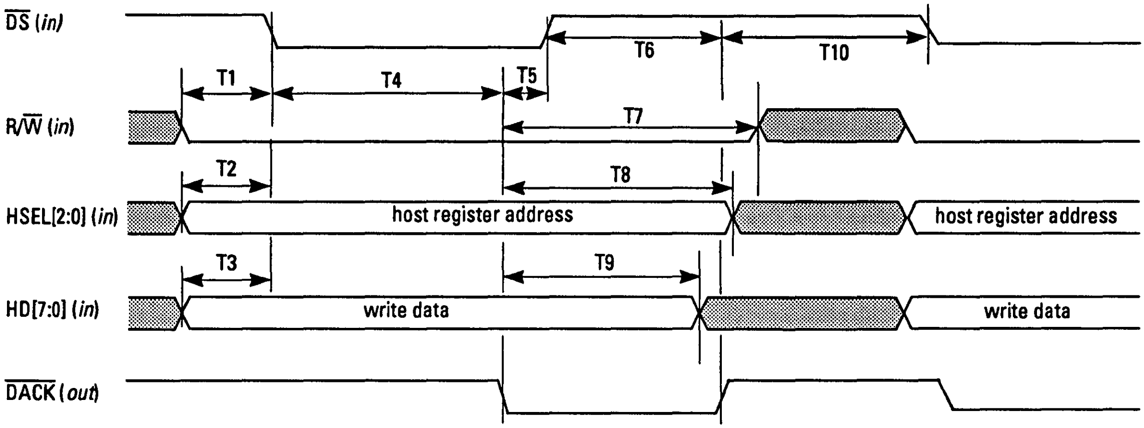


Figure 9-1 Host Interface Local Register, Write Operation

Table 9-4 Timing Characteristics - Local Register Write

Time[1]	Description	Min	Max	Units
T1	R/W valid to DS LOW	5		ns
T2	HSEL[2:0] valid to DS LOW	5		ns
T3	HD[7:0] valid to DS LOW	5		ns
T4	DS LOW to DACK LOW	(below)[2]	(below)[3]	
T5	DACK LOW to DS HIGH	0		ns
T6	DS HIGH to DACK HIGH	1 GCK	2 GCK	
T7	R/W hold after DACK LOW	0		ns
T8	HSEL[2:0] hold after DACK LOW	0		ns
T9	HD[7:0] hold after DACK LOW	0		ns
T10	DS HIGH hold after DACK HIGH	5		ns

[1] Not 100% tested, guaranteed by design.

[2] (GCK * 2) when A_MSB, A_MB, A_LSB and D_LSB write operation; (GCK * 21) when C-FIFO D_MSB write; (GCK * 6) when D_MSB write.

[3] (GCK * 3) when A_MSB, A_MB, A_LSB and D_LSB write operation; (GCK * 22) when C_FIFO D_MSB write; (GCK * 6 + GBUS waiting time) when D_MSB write. Note: GBUS waiting time can be from 0 to over 500 GCKs.

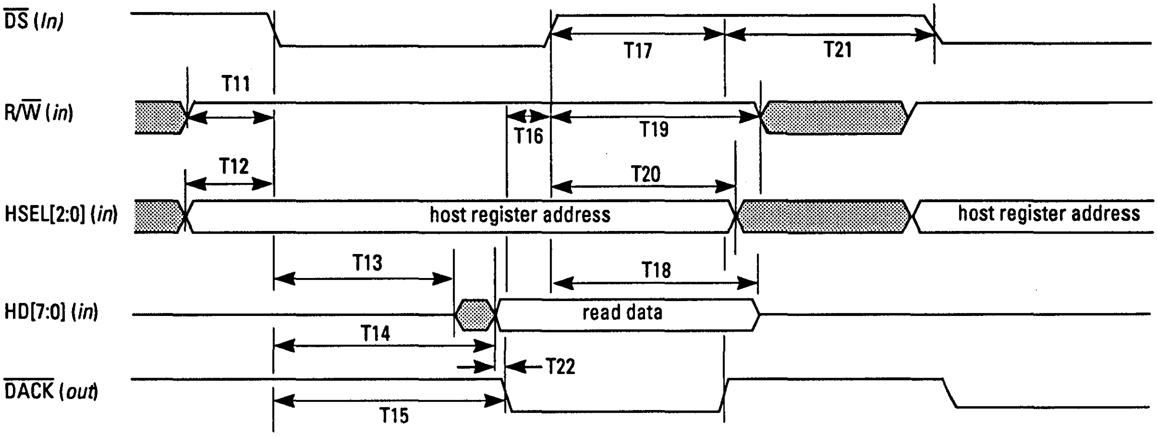


Figure 9-2 Host Interface Local Register, Read Operation

Table 9-5 Timing Characteristics - Local Register Read

Time[1]	Description	Min	Max	Units
T11	R/W̅ valid to D̅S̅ LOW	5		ns
T12	HSEL[2:0] valid to D̅S̅ LOW	5		ns
T13	D̅S̅ LOW to HD[7:0] (non-tristate)	20		ns
T14	D̅S̅ LOW to HD[7:0] valid		Note[2]	ns
T15	D̅S̅ LOW to D̅A̅C̅K̅ LOW	Note[3]	Note[4]	ns
T16	D̅A̅C̅K̅ LOW to D̅S̅ HIGH	25		ns
T17	D̅S̅ HIGH to D̅A̅C̅K̅ HIGH	1 GCK	3 GCK	
T18	D̅S̅ HIGH to HD[7:0] tristate	1 GCK	3 GCK	
T19	R/W̅ hold after D̅S̅ HIGH	2 GCK		
T20	HSEL[2:0] hold after D̅S̅ HIGH	2 GCK		
T21	D̅A̅C̅K̅ HIGH to D̅S̅ LOW (next operation)	5		ns
T22	HD[7:0] valid to D̅A̅C̅K̅ LOW	-10		ns

1. Not 100% tested, guaranteed by design.
2. (3ns + GCK * 2) when A_MSB and D_LSB read; (3ns + GCK * 5 + GBUS waiting time) when D_MSB read
3. (GCK * 2) when A_MSB and D_LSB read; (GCK * 6) when D_MSB read
4. GCK * 3) when A_MSB and D_LSB read; (GCK * 6) + GBUS waiting time when D_MSB read

9.2.2 DRAM/ROM Bus Timing

Local DRAM/ROM Bus Timing is shown next.

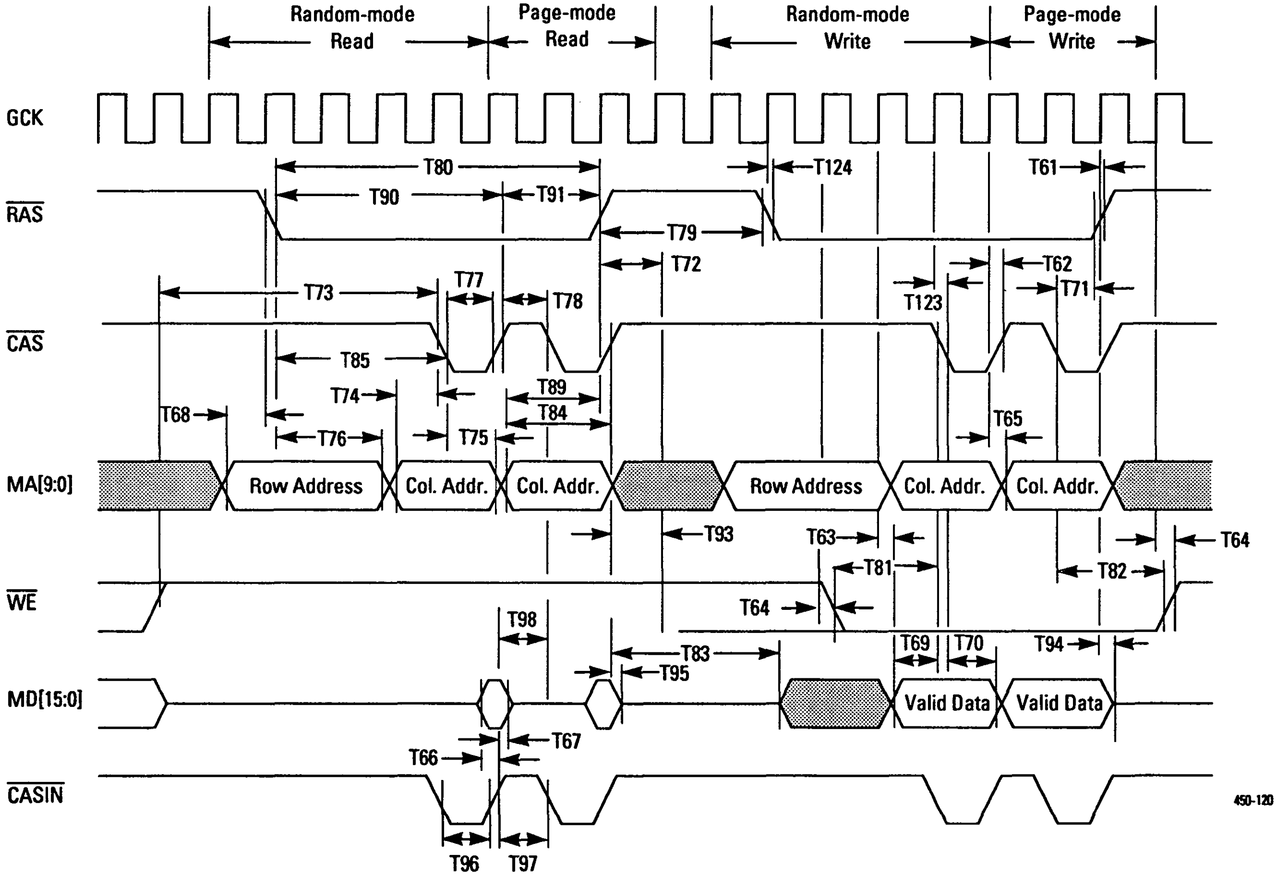


Figure 9-3 Local DRAM Bus Timing

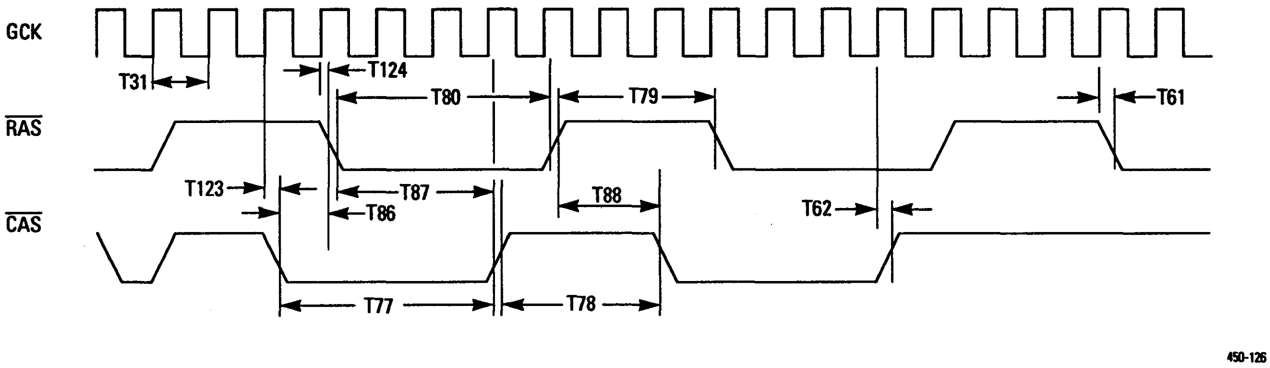


Figure 9-4 Local DRAM $\overline{\text{CAS}}$-before-$\overline{\text{RAS}}$ Refresh

Table 9-6 Timing Characteristics - Local DRAM Bus[1,2]

Time	Description	Parameter	Min	Max	Units
T31	GCK Period (tCYC)		TBD		ns
T61	GCK HIGH to $\overline{RAS}$ HIGH [2]		TBD	TBD	ns
T62	GCK HIGH to $\overline{CAS}$ HIGH [2]		TBD	TBD	ns
T63	GCK HIGH to Memory Data HIGH or LOW (DRAM write) [2]		TBD	TBD	ns
T64	GCK HIGH to $\overline{WE}$ HIGH or LOW [2]		TBD	TBD	ns
T65	GCK HIGH to Memory Address HIGH or LOW [2]		TBD	TBD	ns
T66	Read Data Setup Time before $\overline{CASIN}$ HIGH				ns
T67	Read Data Hold Time after $\overline{CASIN}$ HIGH				ns
T68	Row Address Setup Time [2]	t_{ASR}	0		ns
T69	Write Data Setup Time before $\overline{CAS}$ LOW [2]	t_{DS}	0		ns
T70	Write Data Hold Time after $\overline{CAS}$ LOW [2]	t_{DH}	15		ns
T71	$\overline{RAS}$ Hold Time after $\overline{CAS}$ LOW [2]	t_{RSH}	20		ns
T72	Read Command Hold Time from $\overline{CAS}$ [2]	t_{RRH}	10		ns
T73	Read Command Setup Time to $\overline{CAS}$ LOW [2]	t_{RCS}	0		ns
T74	Column Address Setup Time to $\overline{CAS}$ LOW [2]	t_{ASC}	0		ns
T75	Column Address Hold Time from $\overline{CAS}$ LOW [2]	t_{CAH}	15		ns
T76	Row Address Hold Time from $\overline{RAS}$ LOW [2]	t_{RAH}	10		ns
T77	$\overline{CAS}$ LOW Time [2]	t_{CAS}	20		ns
T78	$\overline{CAS}$ HIGH Time [2]	t_{CP}	10		ns
T79	$\overline{RAS}$ HIGH Time [2]	t_{RP}	70		ns
T80	$\overline{RAS}$ LOW Time [2]	t_{RAS}	80	TBD	ns
T81	Write Command Setup Time to $\overline{CAS}$ LOW [2]	t_{WCS}	0		ns
T82	Write Command Hold Time from $\overline{CAS}$ LOW [2]	t_{WCH}	15		ns
T83	$\overline{CAS}$ HIGH to Memory Data Driven [2,3]				ns
T84	Column Address to $\overline{CAS}$ HIGH [2]	t_{AA} + T66	TBD		ns
T85	$\overline{RAS}$ to $\overline{CAS}$ delay [2]	t_{RCD}	60		ns
T86	$\overline{CAS}$ Setup Time to $\overline{RAS}$ (Memory Refresh Cycle) [2]	t_{CSR}	10		ns
T87	$\overline{CAS}$ Hold Time from $\overline{RAS}$ (Memory Refresh Cycle) [2]	t_{CHR}	15		ns
T88	$\overline{RAS}$ HIGH to $\overline{CAS}$ LOW delay (Memory Refresh Cycle) [2]	t_{RPC}	0		ns
T89	Column Address to $\overline{RAS}$ HIGH [2]	t_{RAL}	40		ns
T90	$\overline{RAS}$ LOW to $\overline{CAS}$ HIGH [2]	t_{RAC} + T66	85		ns
T91	$\overline{RAS}$ Hold Time from $\overline{CAS}$ Precharge [2]	t_{RHCP}	TBD		ns
T93	Read Command Hold Time from $\overline{RAS}$	t_{RRH},t_{RCH}	0		ns
T94	GCK to MD three-state (write)		TBD	TBD	ns

1. Inputs switch between 0.0V and 3.5V at 1V/ns and measurements are made at 0.8V and 2.4V. Output load capacitance = 50 pF.
2. Not 100% tested, guaranteed by design.
3. MD[15:0] driven only when next cycle will be a write.

Table 9-6 Timing Characteristics - Local DRAM Bus (cont.)

Time	Description	Parameter	Min	Max	Units
T95	$\overline{\text{CAS}}$ HIGH to MD three-state (read)		TBD	TBD	ns
T96	$\overline{\text{CASIN}}$ LOW Time		TBD		ns
T97	$\overline{\text{CASIN}}$ HIGH TIME		TBD		ns
T98	$\overline{\text{CASIN}}$ HIGH to $\overline{\text{CAS}}$ LOW		TBD		ns
T123	GCK HIGH to $\overline{\text{CAS}}$ LOW		TBD	TBD	ns
T124	GCK HIGH to $\overline{\text{RAS}}$ LOW		TBD	TBD	ns

9.2.3 CD Interface Timing

Timing for each of the six different CD signal input formats is shown below:

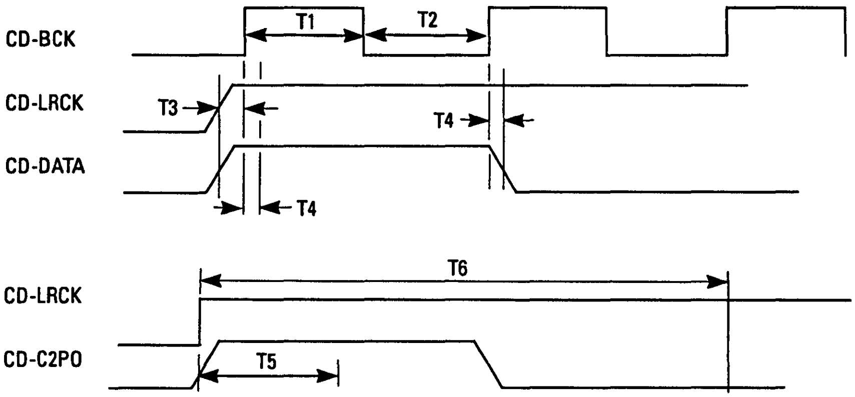


Figure 9-5 CD Input: 32-bit CD-BCK, Data Latch Timing High

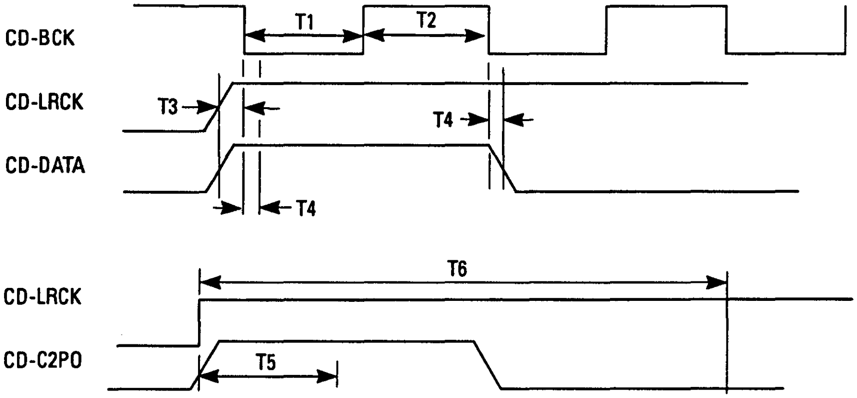


Figure 9-6 CD Input: 32-bit CD-BCK, Data Latch Timing Low

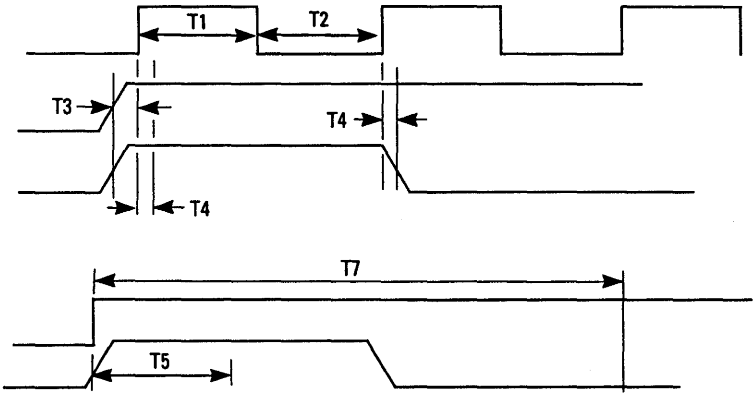


Figure 9-7 CD Input: 24-bit CD-BCK, Data Latch Timing High

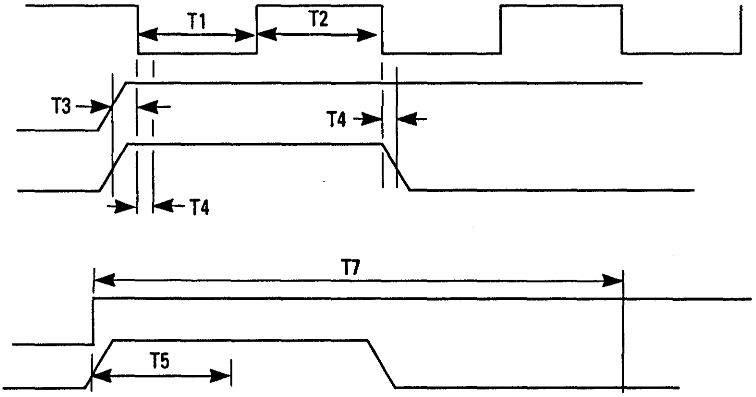


Figure 9-8 CD Input: 24-bit CD-BCK, Data Latch Timing Low

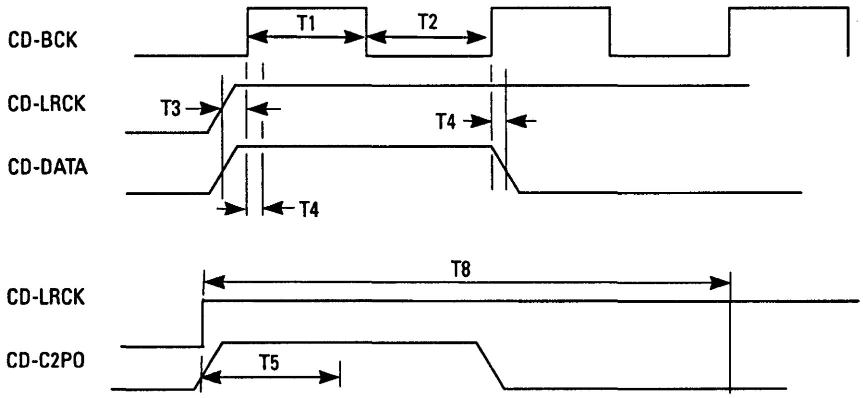


Figure 9-9 CD Input: 16-bit CD-BCK, Data Latch Timing High

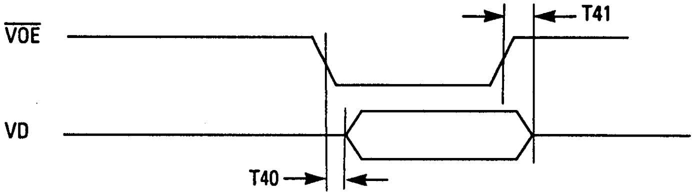
Table 9-7 CD Input Timing[1]

Time	Description	Min	Max	Units
T1	CD-BCK High Pulse Width	(3 GCKs)		
T2	CD-BCK Low Pulse Width	(3 GCKs)		
T3	CD-DATA, CD-LRCK setup	5		ns
T4	CD-DATA, CD-LRCK hold	10		ns
T5	CD-C2PO first byte latch	5 (CD-BCKs)		
T6	CD-C2PO second byte latch	27 (CD-BCKs)		
T7	CD-C2PO second byte latch	19 (CD-BCKs)		
T8	CD-C2PO second byte latch	11 (CD-BCKs)		

[1] Not 100% tested, guaranteed by design.

9.2.4 Video Bus Timing

Timing diagrams for video output are shown in the following pages.



Note: $\overline{\text{VOE}}$ is asserted high in ES1.

Figure 9-10 Timing - $\overline{\text{VOE}}$ and VD

Table 9-8 Timing Characteristics: $\overline{\text{VOE}}$ and VD

Time[1]	Description	Min	Max	Units
T40	$\overline{\text{VOE}}$ LOW to VD non-tristate	3	15	ns
T41	$\overline{\text{VOE}}$ HIGH to VD tristate	3	15	ns

[1] Not 100% tested, guaranteed by design.

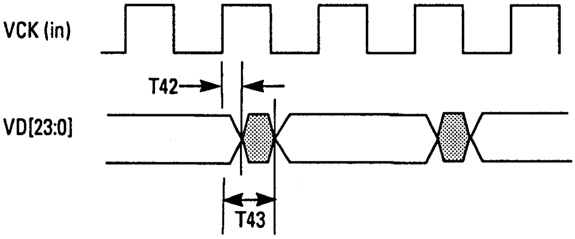


Figure 9-11 Timing - VCK in at 27 MHz and VD (pixel data)

Table 9-9 Timing Characteristics: VCK and VD

Time[1]	Description	Min	Max	Units
T42	VCK (In) HIGH to VD invalid (VD hold after VCK)	__		ns
T43	VCK (In) HIGH to VD valid	7	30	ns

[1] Not 100% tested, guaranteed by design.

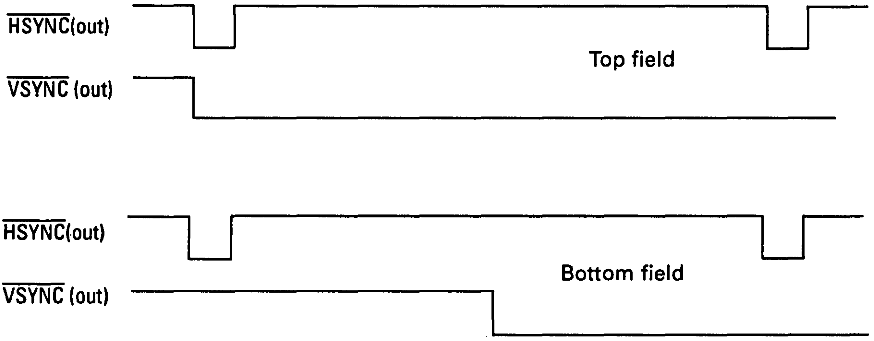

$\overline{\text{HSYNC}}$(out)
$\overline{\text{VSYNC}}$ (out)
Top field
$\overline{\text{HSYNC}}$(out)
$\overline{\text{VSYNC}}$ (out)
Bottom field

Figure 9-12 $\overline{\text{VSYNC}}$ Out versus $\overline{\text{HSYNC}}$ Out

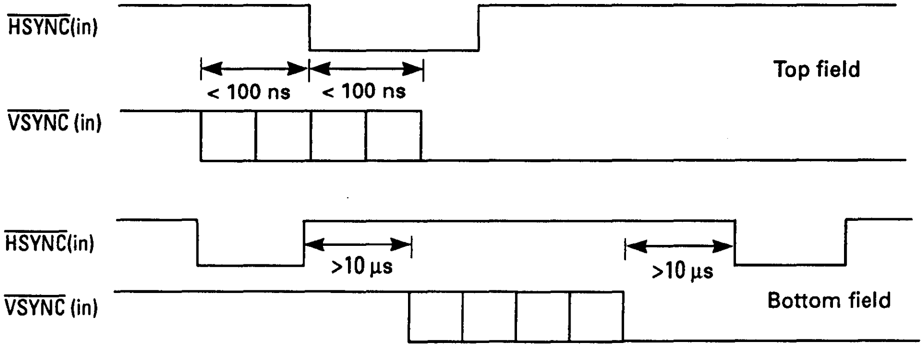

$\overline{\text{HSYNC}}$(in)
< 100 ns
< 100 ns
Top field
$\overline{\text{VSYNC}}$ (in)
$\overline{\text{HSYNC}}$(in)
>10 µs
>10 µs
$\overline{\text{VSYNC}}$ (in)
Bottom field

Figure 9-13 $\overline{\text{VSYNC}}$ In versus $\overline{\text{HSYNC}}$ In

Note: There are no unusual timing requirements on $\overline{\text{HSYNC}}$, $\overline{\text{VSYNC}}$ or CSYNC.

When $\overline{\text{HSYNC}}$ and $\overline{\text{VSYNC}}$ are outputs (Figure 9-12), the position of the $\overline{\text{VSYNC}}$ edges in the HSYNC period is completely programmable by writing DRAM locations.

Note: $\overline{\text{HSYNC}}$ must always be synchronous to VCK in both $\overline{\text{HSYNC}}$ in and $\overline{\text{HSYNC}}$ out modes.

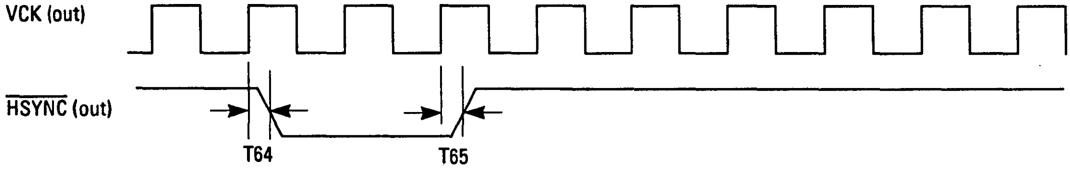


Figure 9-14 VCK Out versus $\overline{\text{HSYNC}}$ Out

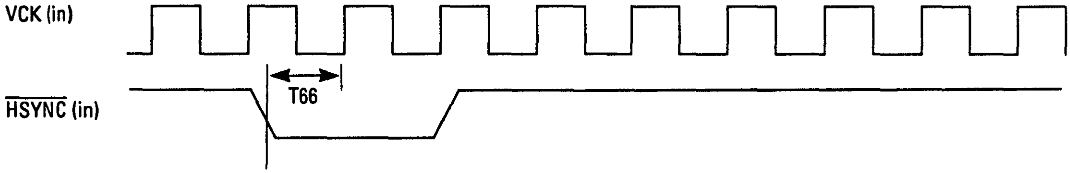


Figure 9-15 VCK In versus $\overline{\text{HSYNC}}$ In

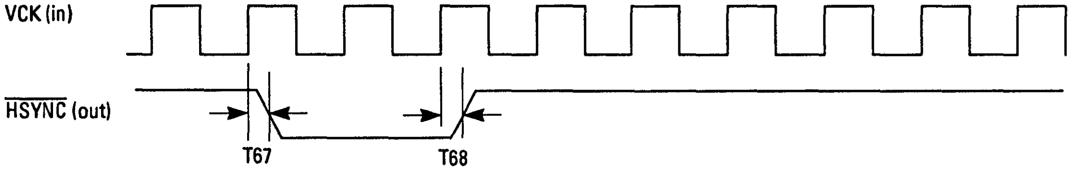


Figure 9-16 VCK In versus $\overline{\text{HSYNC}}$ Out

Table 9-10 Timing Characteristics: VCK, $\overline{\text{HSYNC}}$, and CSYNC/$\overline{\text{VSYNC}}$

Time[1]	Description	Min	Max	Units
T64	VCK (out) HIGH to $\overline{\text{HSYNC}}$ (out) LOW	0	15	ns
T65	VCK (out) HIGH to $\overline{\text{HSYNC}}$ HIGH	0	15	ns
T66	$\overline{\text{HSYNC}}$ (in) LOW to VCK (in) HIGH	10		ns
T67	VCK (in) HIGH to $\overline{\text{HSYNC}}$ (out) LOW	3	30	ns
T68	VCK (in) HIGH to $\overline{\text{HSYNC}}$ (out) HIGH	3	30	ns

[1] Not 100% tested, guaranteed by design.

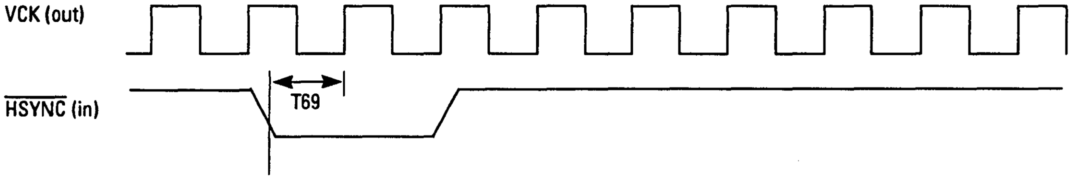


Figure 9-17 VCK Out versus $\overline{\text{HSYNC}}$ In

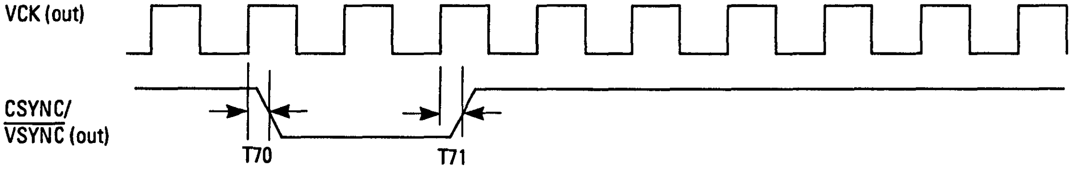


Figure 9-18 VCK Out versus CSYNC/$\overline{\text{VSYNC}}$ Out

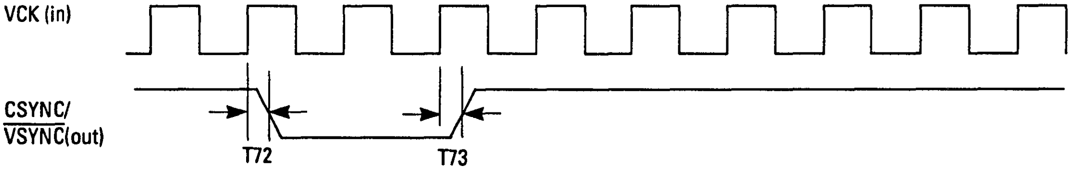


Figure 9-19 VCK In versus CSYNC/$\overline{\text{VSYNC}}$ Out

Table 9-11 Timing Characteristics: VCK, $\overline{\text{HSYNC}}$, and CSYNC/$\overline{\text{VSYNC}}$

Time[1]	Description	Min	Max	Units
T69	$\overline{\text{HSYNC}}$ (in) LOW to VCK (out) HIGH	20		ns
T70	VCK (out) HIGH to CSYNC/$\overline{\text{VSYNC}}$ (out) LOW	0	8	ns
T71	VCK (out) HIGH to CSYNC/$\overline{\text{VSYNC}}$ (out) HIGH	0	8	ns
T72	VCK (in) HIGH to CSYNC/$\overline{\text{VSYNC}}$ (out) LOW	3	13	ns
T73	VCK (in) HIGH to CSYNC/$\overline{\text{VSYNC}}$ (out) HIGH	3	13	ns

[1] Not 100% tested, guaranteed by design.

9.2.5 Audio Bus Timing

Timing for the audio interface of the CL480 is shown in Figure 9-20.

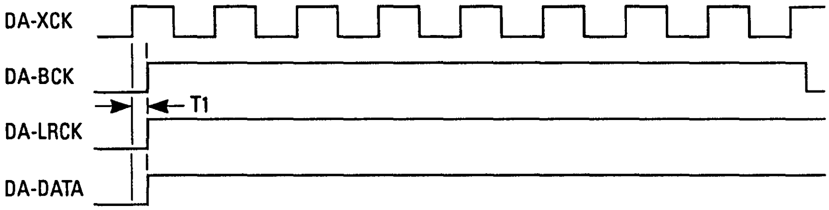


Figure 9-20 Audio Interface Timing Modes

The three output pins of the audio bus—DA-BCK, DA-LRCK, and DA-DATA—are synchronized to the rising edge of the input signal DA-XCK. The delay time between the rising edge of DA-XCK and these three output pins is the same and is shown below in Table 9-12. The signal DA-XCK is independent of GCK, so it does not have a specified set and hold time.

Table 9-12 Timing Characteristics: Audio

Time[1]	Description	Min.	Max	Units
T1	Delay from DA-XCK to DA-BCK, DA-LRCK, and DA-DATA	3.3	30.0	ns

[1] Not 100% tested, guaranteed by design.

Note: DA-DATA is driven out of the CL480 on the falling edge of DA-BCK and is typically latched in the audio DAC with the rising edge DA-BCK.

9.3 Package Specifications

The CL480 is packaged in a 128-pin small outline plastic quad flat pack (PQFP). This section includes:

- □ The CL480 Pinout Diagram
- □ Tables of CL480 pin connections
  - □ Host Bus Interface Pins
  - □ CD Interface Pins
  - □ Global Interface Pins
  - □ DRAM/ROM Bus Interface Pins
  - □ Audio Bus Interface Pins
  - □ Video Bus Interface Pins
  - □ Power and Miscellaneous Pins
- □ Package Physical Dimensions

The CL480 is shipped in a drypack with desiccant and a humidity monitor. Do not use the parts if the humidity indicator indicates that the humidity is greater than 30% at the initial opening of the drypack. To avoid cracking the plastic during soldering, the parts should be soldered within three days after breaking the drypack seal.

Note that "RESERVED" pins are reserved for future use. They should be pulled either HIGH or LOW.

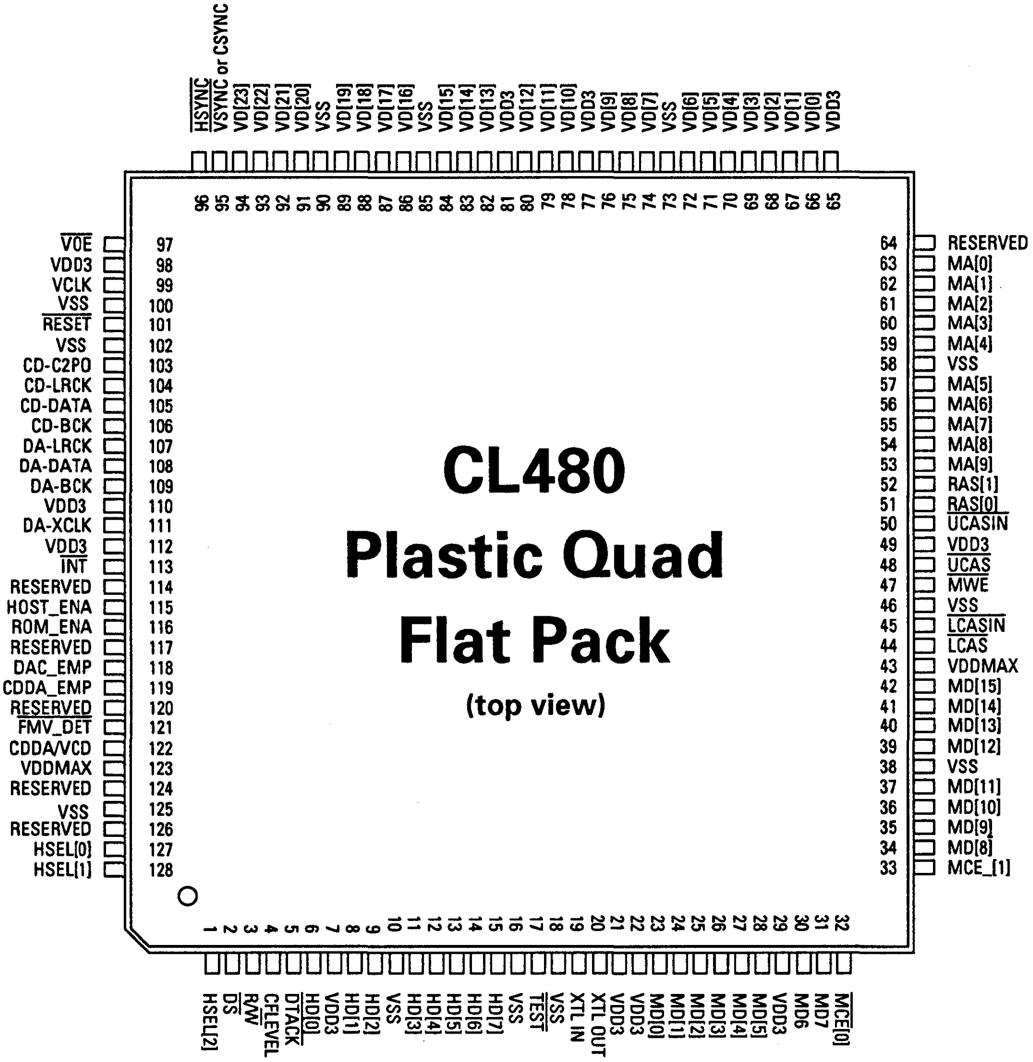


Figure 9-21 CL480 PQFP Pinout Diagram

9.3.1 Pin List

Table 9-13 Host Bus Interface Pins

Function	Group	I/O	Pin #	Function	Group	I/O	Pin #
HSEL2	Host	I	1	HD[3]	Host	I/O	11
HSEL1	Host	I	128	HD[2]	Host	I/O	9
HSEL0	Host	I	127	HD[1]	Host	I/O	8
CFLEVEL[1]	Host	O	4	HD[0]	Host	I/O	6
$\overline{DS}$	Host	I	2	R/$\overline{W}$[2]	Host	I	3
HD[7]	Host	I/O	15	$\overline{DTACK}$[1]	Host	O	5
HD[6]	Host	I/O	14	$\overline{INT}$[1]	Host	I/O	113
HD[5]	Host	I/O	13	HOST_ENA	Host	I	115
HD[4]	Host	I/O	12				

[1] Open-drain output, requires a pullup resistor of 1.5K ohms.

[2] Note: The signal is asserted $\overline{R}$/W in ES1.

Table 9-14 CD Interface Pins

Function	Group	I/O	Pin #	Function	Group	I/O	Pin #
CD-LRCK	CD	I	104	CD-BCK	CD	I	106
CD-DATA	CD	I	105	CD-C2PO	CD	I	103
CDDA/VCD	CD	O	122	$\overline{FMV\_DET}$	CD	O	121
Reserved	CD		119	Reserved	CD		124
Reserved	CD		120	Reserved	CD		126

Table 9-15 Global Interface Pins

Function	Group	I/O	Pin #	Function	Group	I/O	Pin #
$\overline{RESET}$	Global	I	101	$\overline{TEST}$[1]	Global	I	17

[1] Requires a pullup resistor of 1.5K ohms.

Table 9-16 DRAM/ROM Bus Interface Pins

Function	Group	I/O	Pin #	Function	Group	I/O	Pin #
MA9	DR/RM	0	53	MD8	DR/RM	I/O	34
MA8	DR/RM	0	54	MD7	DR/RM	I/O	31
MA7	DR/RM	0	55	MD6	DR/RM	I/O	30
MA6	DR/RM	0	56	MD5	DR/RM	I/O	28
MA5	DR/RM	0	57	MD4	DR/RM	I/O	27
MA4	DR/RM	0	59	MD3	DR/RM	I/O	26
MA3	DR/RM	0	60	MD2	DR/RM	I/O	25
MA2	DR/RM	0	61	MD1	DR/RM	I/O	24
MA1	DR/RM	0	62	MD0	DR/RM	I/O	23
MA0	DR/RM	0	63	$\overline{\text{RAS}}$[0]	DR/RM	0	51
MD15	DR/RM	I/O	42	$\overline{\text{RAS}}$[1]	DR/RM	0	52
MD14	DR/RM	I/O	41	$\overline{\text{LCAS}}$	DR/RM	0	44
MD13	DR/RM	I/O	40	$\overline{\text{UCAS}}$	DR/RM	0	48
MD12	DR/RM	I/O	39	$\overline{\text{LCASIN}}$	DR/RM	I	45
MD11	DR/RM	I/O	37	$\overline{\text{UCASIN}}$	DR/RM	I	50
MD10	DR/RM	I/O	36	$\overline{\text{WE}}$	DRAM	0	47
MD9	DR/RM	I/O	35	$\overline{\text{CE}}$[1]	ROM	0	33
$\overline{\text{CE}}$[0]	ROM	0	32	RESERVED	ROM	0	64
ROM_ENA	ROM	I	116				

Table 9-17 Audio Bus Interface Pins

Function	Group	I/O	Pin #	Function	Group	I/O	Pin #
DA-LRCK	Audio	0	107	DA-BCK	Audio	0	109
DA-DATA	Audio	0	108	DA-XCLK	Audio	I	111
DAC_EMP	Audio	0	118	CDDA_EMP	Audio	I	119

Table 9-18 Video Bus Interface Pins

Function	Group	I/O	Pin #	Function	Group	I/O	Pin #
VD23	Video	O	94	VD9	Video	O	76
VD22	Video	O	93	VD8	Video	O	75
VD21	Video	O	92	VD7	Video	O	74
VD20	Video	O	91	VD6	Video	O	72
VD19	Video	O	89	VD5	Video	O	71
VD18	Video	O	88	VD4	Video	O	70
VD17	Video	O	87	VD3	Video	O	69
VD16	Video	O	86	VD2	Video	O	68
VD15	Video	O	84	VD1	Video	O	67
VD14	Video	O	83	VD0	Video	O	66
VD13	Video	O	82	$\overline{\text{HSYNC}}$	Video	I	96
VD12	Video	O	80	$\overline{\text{VOE}}$[1]	Video	I	97
VD11	Video	O	79	VCLK	Video	I	99
VD10	Video	O	78	$\overline{\text{VSYNC}}$	Video	I	95

[1] Note: In ES1, this signal is asserted high (VOE, not $\overline{\text{VOE}}$).

Table 9-19 Power and Miscellaneous Pins

Function	Group	I/O	Pin #	Function	Group	I/O	Pin #
VDD3	Power	PWR	7	VSS	Power	GND	10
VDD3	Power	PWR	29	VSS	Power	GND	16
VDDMAX[1]	Power	PWR	43	VSS	Power	GND	18
VDD3	Power	PWR	49	VSS	Power	GND	38
VDD3	Power	PWR	65	VSS	Power	GND	46
VDD3	Power	PWR	77	VSS	Power	GND	58
VDD3	Power	PWR	81	VSS	Power	GND	73
VDD3	Power	PWR	98	VSS	Power	GND	85
VDD3	Power	PWR	110	VSS	Power	GND	90
VDD3	Power	PWR	112	VSS	Power	GND	100
VDDMAX[1]	Power	PWR	123	VSS	Power	GND	102
VDD3	Power	PWR	21	VSS	Power	GND	125
VDD3	Power	PWR	22	VSS	Power	GND	18
XTL IN			19	XTL OUT			20

[1] VDDMAX is the maximum of the DRAM and Host interface input voltages.

9.3.2 Package Drawings

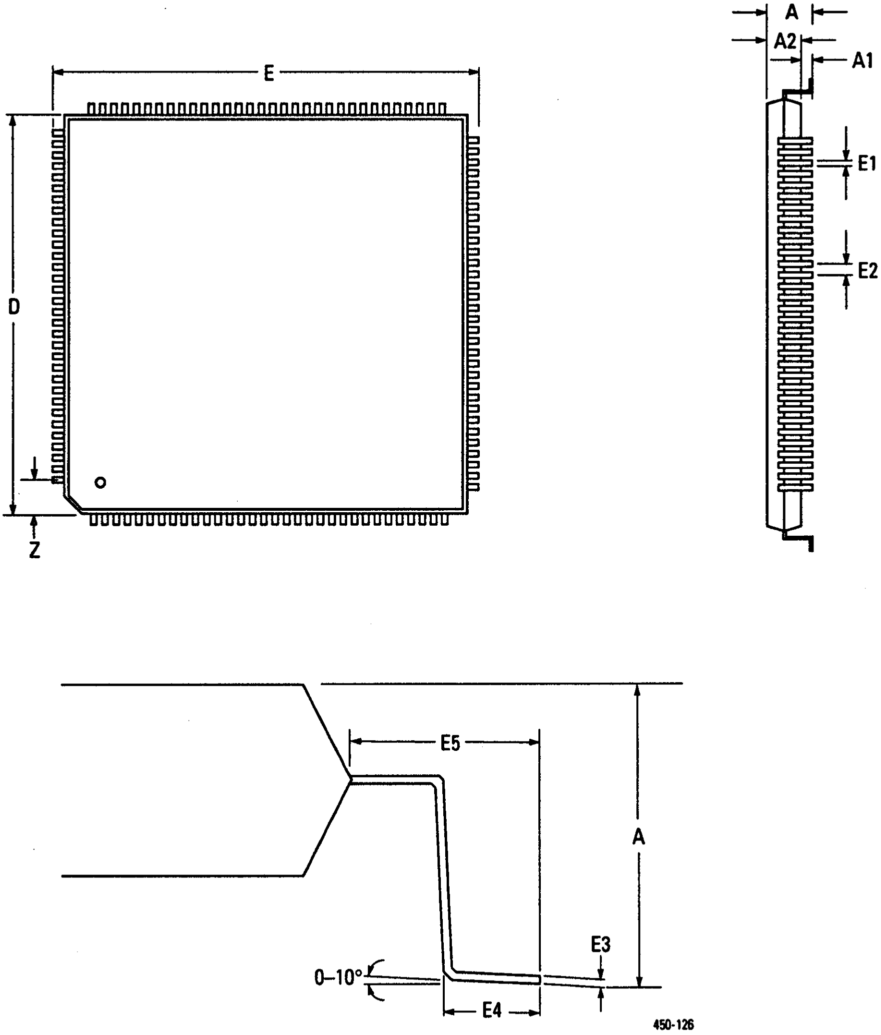


Figure 9-22 Plastic Quad Flat Pack Physical Dimensions

Table 9-20 Plastic Quad Flat Pack Physical Dimensions

SYMBOL	DIMENSIONS	
	INCHES	MM
A		3.7 MAX.
A1		0 MIN.
A2		3.5 MAX.
D		18.0 ±0.20
E		20.0 ±0.20
E1		0.2 TYP.
E2		0.5 TYP.
E3		0.15 TYP.
E4		0.50 ±0.20
E5		1.0 ±0.20
Z		1.25 TYP.

10 Registers

This chapter describes the CL480 registers. You should be familiar with the relation of these registers to the CL480's external signals as described in Chapter 3, Signal Descriptions. You should also be aware that *the functions described by these registers are initialized by microcode as outlined in Section 12.3.*

The sections in this chapter are:

10.1 CL480 Register Categories

The CL480 registers perform low-level operations or act as part of the protocol for communicating with the microcode executing on the CL480's internal CPU. For purposes of this reference, they are divided into three categories as described below and shown in Table 10-1.

□ *Normal*: These registers are used routinely to configure/communicate with the CL480 and its microcode, although not all applications use all of these registers.

□ *Initialization*: These registers are initialized by microcode and should only be accessed by the host prior to the beginning of microcode execution as part of the chip initialization procedure.

□ *Diagnostic*: These registers are used *only* when debugging a problem. Because the host's use of these registers may conflict with CL480 operations, diagnostic register accesses should not be performed during normal operation.

Table 10-1 CL480 Registers (listed by category)

Category	Register Name	Address	Category	Register Name	Address
Normal	CPU_cntl	0x33	Diagnostic	AUD_mode	0x1B
	CPU_pc	0x3A		VID_WEIGHT0	0xh2
	CPU_iaddr	0x36		VID_WEIGHT1	0xh3
	CPU_imem	0x32		VID_WEIGHT2	0xh4
	HOST_int	0x0F		VID_WEIGHT3	0xh5
Initialization	DRAM_ROMTA	0x32		VID_WEIGHT4	0xh6
	DRAM_REF	0x33		VID_MODE	0xh7
	DRAM_iaddr	0x20		VID_Cr	0xhA
	DRAM_data	0x21		VID_YCb	0xhB
	VID_iaddr	0x03		VID_HSYNC_LO	0xhC
	VID_data	0x04		VID_HSYNC_CYC	0xhD
Diagnostic	VID_csync	0x0B		VID_VSYNCTOG	0xhE
	CD_cntl	0x12		VID_HINT	0xhF
	CD_cnfg	0x10		HOST_pio	0x07

10.1.1 Direct-access Registers

Registers that can be accessed using a single transfer across the CL480's host interface are listed in Table 10-2. To access these registers, perform a host read or write with address bits A_MSB[7:6] equal to 10 and address bits A_MSB[5:0] set to the register address.

Table 10-2 Direct-access Registers

Register	Address	Used For:	R/W	Page #
CD_cntl	0x12	Selects type of data sent to the CL480	R/W	10-8
CD_cnfg	0x10	Selects data format for CD interface	R/W	10-9
CPU_cntl	0x33	Sets CPU Run enable bit	R/W	10-6
CPU_pc	0x3A	CPU program counter	R/W	10-6
CPU_iaddr	0x36	Write address for instruction memory (IMEM)	R/W	10-7
CPU_imem	0x32	Data for instruction memory (IMEM)	R/W	10-8
DRAM_iaddr	0x20	Indirect address for DRAM controller	R/W	10-10
DRAM_data	0x21	Data for DRAM controller registers	W	10-11
VID_iaddr	0x03	Indirect address for video register access	R/W	10-13
VID_data	0x04	Data for video registers	W	10-14
VID_csync	0x0B	Composite sync control	R/W	10-13
AUD_mode	0x1B	Data port to audio mode register (normal)	R/W	10-12
HOST_int	0x0F	Host interrupt register	R/W	10-5
HOST_pio	0x07	Direction and data for programmable I/Os	R/W	10-5

10.1.2 Indirect-access Registers

The first two registers listed in Table 10-3 are accessed indirectly by specifying the desired register number in bits 5:0 of the DRAM_iaddr register (the *ID* field), and then reading or writing the DRAM_data register. The remainder of registers in Table 10-3 are addresed by writing bits 8:5 of the VID_iaddr register. Note that the "*ID* " value of Table 10-3 replaces the "Address" column of Table 10-2 but that the tables are otherwise identical.

> *Note: All register addresses and values not shown in Table 10-2 and Table 10-3 are RESERVED. Attempting to read or write these register addresses—or to read a write-only register, or to write a read-only register—causes indeterminate results.*

Table 10-3 Indirect-access Registers (described)

Register	ID	Used For:	R/W	Page #
DRAM_ROMTA	0x32	ROM access time	W	10-11
DRAM_REF	0x33	Refresh time	W	10-11
VID_WEIGHT0	0xh2	W0 coefficient of YUV to RGB conversion	W	10-16
VID_WEIGHT1	0xh3	W1 coefficient of YUV to RGB conversion	W	10-16
VID_WEIGHT2	0xh4	W2 coefficient of YUV to RGB conversion	W	10-16
VID_WEIGHT3	0xh5	W3 coefficient of YUV to RGB conversion	W	10-16
VID_WEIGHT4	0xh6	W4 coefficient of YUV to RGB conversion	W	10-16
VID_MODE	0xh7	Video modes	W	10-17
VID_Cr	0xhA	Border color: Cr	W	10-14
VID_YCb	0xhB	Border color: Y, Cb	W	10-14
VID_HSYNC_LO	0xhC	Width of $\overline{\text{HSYNC}}$ low	W	10-15
VID_HSYNC_CYC	0xhD	Sets number of pixel times in active window	W	10-15
VID_VSYNCTOG	0xhE	$\overline{\text{VSYNC}}$ toggle point	W	10-15
VID_HINT	0xhF	Horizontal interrupt position	W	10-15

The sections that follow describe the registers contained in each CL480 interface module. In the detailed definitions given for each register, note that bits marked *Res* are reserved. They will return a 0 or 1 when read, but should always be written to 0.

10.2 Host Interface Register

This section describes the interrupt control register in the host interface. The $\overline{\text{INT}}$ pin is deasserted by writing a 0 to the HOST_int register.

HOST_int Register

(normal) **0x0F**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res						*Int*	*Res*								

Field	Bit	Default	Description
Int	9	output	Interrupt request

If you want to change the Int bit: (1) read out the entire register, 0xFEFF, (2) AND the value that you want to write with 0xFEFF, and then (3) write back this register.

HOST_pio Register

(diagnostic) **0x07 [direct]**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	*DAC*	*0*	*CDe*	*1*	*Res*	*1*	*FMV*	*I/O*	*CD*	*1*	*Res*	*1*	*Res*	*1*	*Res*

DAC **(bit 14)** **W**

DAC emphasis flag output.

CDe **(bit 12)** **R**

CD-DA emphasis input.

FMV **(bit 8)** **W**

FMV bitstream (CD-I) detected.

CD **(bit 6)** **W**

CD-DA bitstream detected.

10.3 Internal CPU Registers

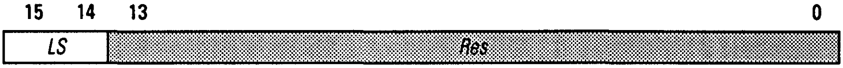
This next section describes the registers that allow application programs to communicate with the CL480's internal CPU.

10.3.1 CPU Execution Registers

These registers are used to enable the CPU and to monitor the execution of internal microcode.

CPU_cntl Register

(normal) **0x33**



LS **Local State (bits 15, 14)** **R/W**

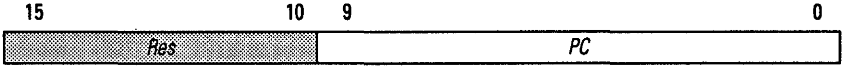
The host processor sets this bit as follows to determine the local state:

00: run
01: single step
10: halt
11: resets CPU module

After initialization, the host needs to set these register bits to 00.

CPU_pc Register

(normal) **0x3A**



PC **Program Counter (bits 9:0)** **R/W**

PC is a ten-bit field that holds the current contents of the program counter used by the internal CPU. When a new value is written to *PC*, the internal CPU executes the instruction at that address on the next instruction cycle.

10.3.2 IMEM Access Registers

The IMEM access registers, CPU_iaddr and CPU_imem, provide a mechanism by which the host processor can write 16-bit microcode words into the internal instruction memory (IMEM) to initialize the CL480 if a boot ROM is not used. IMEM can hold up to 1024 16-bit microcode words. The procedure for writing to IMEM is shown in Figure 10-4. The circled numbers refer to the steps that follow.

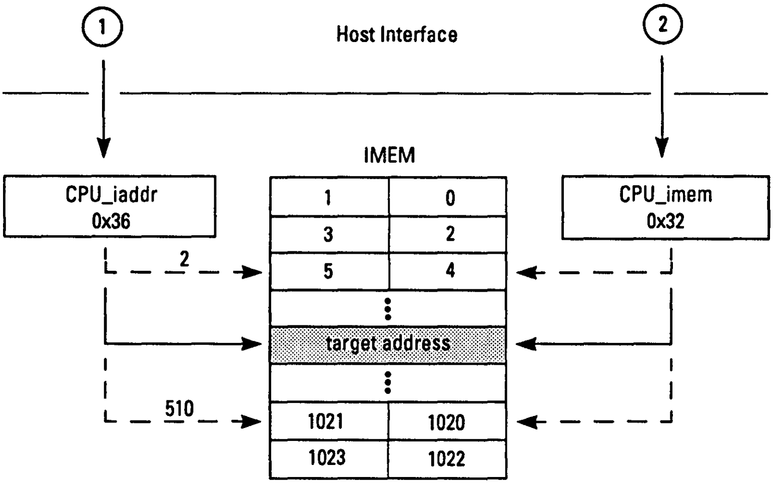


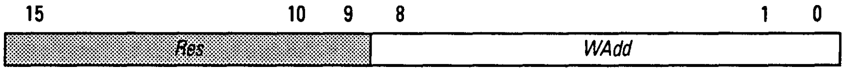
Figure 10-1 IMEM Write Data Flow

The sequence of events for writing to the IMEM is:

1. The host processor loads the IMEM address to be written into CPU_iaddr bits 8:0. The address points to a 32-bit location, and thus the 16-bit word address should be shifted right one bit.
2. The host processor writes successive pairs of 16-bit words to the CPU_imem register. (Each pair of words makes up a double 16-bit instruction.) The CL480 automatically increments CPU_iaddr with every other write to CPU_imem.

(normal) 0x36

CPU_iaddr Register



WAdd **Write Address (bits 8:0)** **R/W**

This nine-bit field contains the address in the IMEM to which the write operation is to be performed. *WAdd* is automatically post-incremented by the CL480 when 32-bit data is written to CPU_imem (two 16-bit writes).

CPU_imem Register

(normal) **0x32**

15	0
WData	

WData **Write Data (bits 15:0)** **R/W**

The host processor writes the IMEM data to this 16-bit field. The data written to *WData* is transferred internally to the IMEM address pointed to by CPU_iaddr.

Note: Writes to this register must be done in pairs, with the first word written containing the most significant 16 bits of each instruction. If writes are not performed in pairs, the auto-increment feature of the CPU_iaddr register and the contents of IMEM will be left in an unknown state.

10.4 CD Interface Registers

This next section describes the two CD interface control registers.

CD_cntl Register

The CD_cntl register, shown below, selects the CD interface data format and controls ROM decode.

(diagnostic) **0x12**

15	8	7	6	5	4	3	2	1	0
Res		*Swe*	*Se*	*Dwe*	*Dtp*			*Res*	

Table 10-4 CD_cntl Register Bit Selection

Field	Bits	R/W[1]	Description
Res	15:8		Reserved.
Swe	7	W	When this bit is high, bit *Se* is enabled to write.
Se	6	R/W	If this bit is high, search sector sync is active even if *Dtp* in this register is CD-DA.
Dwe	5	W	When this bit is set high by the host, *Dtp* is enabled to be written by the host.
Dtp	4:2	R/W	This value is used to control the mode and data type of the ROM decode function: 3'b000 - MPEG video/audio 3'b001 - MPEG system 3'b010 - CD-DA 3'b011 - CD-ROM (MODE 0 or MODE 2) 3'b100 - CD-ROM (MODE 1) 3'b101 - MODE2 - Form 1 3'b110 - MODE2 - Form2

[1] Note: Bits listed as write-only are returned with indeterminate values upon read operations.

CD_cnfg Register

The CD_cnfg register selects the data format of the CD interface as shown below.

(diagnostic) **0x10**

15		5	4	3	2	1	0
Res		*Length*		*LSBF*	*C2LF*	*RCH*	*BCKF*

Table 10-5 CD_cnfg Register Bit Selection

Field	Bits	R/W	Description
Res	[15:6]		Reserved.
Length	[5:4]	R/W	These bits set the number of BCKs per 16-bit word: 00: 32 BCKs, Data valid last 16 BCKs 01: 16 BCKs, Data valid every BCK. 10: 24 BCKs, Data valid last 16 BCKs. 11: 24 BCKs, Data valid first 16 BCKs (I^{2}S)
LSBF	3	R/W	This bit must be set equal to one for input data sent with LSB first. This bit is set to zero after receiving the reset signal.
C2LF	2	R/W	This bit must be set equal to one for input data sent with LSB first.
RCH	1	R/W	This bit must be set to one if high on LRCK indicates the right channel. Bits 5:0 of CD_cnfg are set to zero by the reset signal.
BCKF	0	R/W	0 : Input data is latched on the rising edge of BCK. 1 : Input data is latched on the falling edge of BCK.

10.5 DRAM Interface Registers

This section describes two registers internal to the DRAM controller—DRAM_ROMTA and DRAM_REF—and the two registers provided to indirectly address them: DRAM_iaddr and DRAM_data.

DRAM_iaddr Register

The DRAM_iaddr register, shown below, contains the address of the indirectly addressed registers.

(initialization) **0x20**

15 8	7	6	5 0
Res	DN	R/W	ID

DN **Access (bit 7)** **R**

This bit is only readable and specifies whether the previous indirect register access has completed. When the DRAM_iaddr is written, this bit is cleared and remains clear until the request access has completed, when this bit is set. So for proper operation, this bit must be checked. For a read operation, after writing the address to this register, the host should continuously monitor this bit until it is set. This guarantees the data requested is in the DRAM_data register. For a write operation, this bit is set after the write is completed, so subsequent indirect writes need to check this bit before another indirect access is issued.

R/W **Read/write (bit 6)** **R/W**

Indirect access read/write. 1 = read, 0 = write

ID **Register ID (bits 5:0)** **R/W**

This six-bit field points to one of the indirect registers. To access one of the indirect registers, the host processor loads the register ID into *ID*, then performs a read from or a write to the direct register DRAM_data. The register IDs are listed in numerical order in Table 10-6. *ID*s not listed are reserved and should not be used.

Table 10-6 DRAM_iaddr Register IDs

ID	Register	*ID*	Register
0x32	DRAM_ROMTA	0x33	DRAM_REF

DRAM_data Register

The DRAM_data register contains the data read from or written to the indirect registers pointer to by *ID* in DRAM_iaddr as shown below.

(initialization) **0x21**

15	0
DATA	

DRAM_ROMTA Register

The DRAM_ROMTA register, shown below, is used to program ROM access time.

(initialization **0x32**

15	7	6	5	4	0
Res		*1*	*1*	*RTA*	

RTA **ROM Access Time** **R**

These bits are used to set the ROM access time (in GCK cycles). ROM access time = (RTA+3)* GCK period. Bits 5 and 6 of DRAM_ROMTA must both be set to 1.

DRAM_REF Register

The DRAM_REF register, shown below, defines the time between refresh cycles.

(initialization) **0x33**

15	14	13	12	0
NOR	*Res*		*REF_CK*	

NOR **Refresh Disable** **W**

When one, this bit disables refresh.

REF_CK **Refresh Period** **W**

Defines the number of GCKs between DRAM refresh cycles.

10.6 Audio Interface Register

This section describes the register in the Audio interface module.

AUD_mode Register

The AUD_mode register selects the parameters which define one of several serial output modes supported by the audio interface unit.

(diagnostic) **0x1B**

15–6	5	4	3	2	1	0
Res	*AE*	*OE*	*RCH*	*LCH*	*S24*	*LSB*

AE **Audio Mode Write Enable** **W**

Bits 4:0 of AUD_mode are only written when AE is one.

OE **Output Enable** **R/W**

If this bit is high, the audio output is enabled. If this bit is low, the audio unit is reset.

LCH **Left Output (DA-LRCK = S24)** **R/W**

0: right data (channel 1 in bitstream)
1: left data (channel 0 in bitstream)

RCH **Right Output (DA-LRCK =$\overline{\text{S24}}$)** **R/W**

0: left data (channel 0 in bitstream)
1: right data (channel 1 in bitstream)

S24 **Select 24 bits** **R/W**

If this bit is one, 24 DA-BCKs are used for each 16-bit audio sample. The last 16 of 24 DA-BCKs contain the audio sample. The first eight DA-BCKs output on the ninth DA-BCK. The bit output on the ninth DA-BCK is either the LSB or the MSB, depending on bit 0 of AUD_mode.

Note: The polarity of DA-LRCK is controlled by S24. When S24 = 1, the left channel is output when DA-LRCK=1.

LSB **LSB First** **R/W**

If this bit is one, the LSB is shifted out first.

10.7 Video Interface Registers

This section describes the registers in the video interface module. Note that the first three—VID_csync, VID_iaddr, and VID_data—are addressed directly, while all the others are addressed indirectly by writing the appropriate register ID (*VRID*) to the VID_iaddr register.

VID_csync Register

The bit assignments for the VID_csync register are shown below.

(diagnostic) **0x0B**

15	7	6	5	4	3	2	1	0
Res		*Dwe*	*DI*	*WE*	*IL*	*CE*	*CDI*	*PAL*

Dwe **Write enable for dither bit (bit 6)** **W**

When one, the DI bit is written.

DI **Dither enable (bit 5)** **W**

When one, dithering of 24-bit RGB to 15-bit RGB is enabled. Note: dithering is only available on ES2 and ES3.

WE **Write enable for csync modes (bit 4)** **W**

When one, bits 3:0 of VID_csync can be written.

IL **Interlace (bit 3)** **W**

When one (default), the composite sync signal will be interlaced (0 = non-interlaced).

CE **CSYNC Enable (bit 2)** **W**

When one, CSYNC (composite sync) is output.When zero (default), $\overline{\text{VSYNC}}$ is output.

CDI **CD-I (bit 1)** **W**

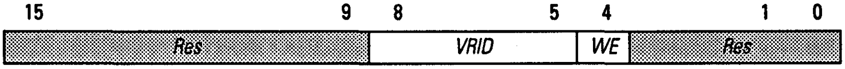
When one, the composite sync will be in CD-I mode.

PAL **PAL (bit 0)** **W**

When one, the composite sync will be in PAL mode.

VID_iaddr Register

(initialization) **0x03**



VRID **Video Register ID (bits 8:5)** **R/W**

This four-bit field points to one of the indirect video registers. To access one of the indirect video registers, the host processor loads the video register ID into *VRID*, then performs a read from or a write to the direct register VID_data. The video register IDs are listed in numerical order in Table 8-8. *VRIDs* not listed are reserved and should not be used.

Table 10-7 Video Register IDs

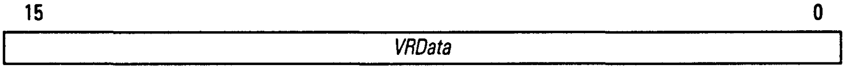
VRID	Video Register	*VRID*	Video Register
A	VID_Cr	2	VID_WEIGHT0
B	VID_YCb	3	VID_WEIGHT1
C	VID_HSYNC_LO	4	VID_WEIGHT2
D	VID_HSYNC_CYC	5	VID_WEIGHT3
E	VID_VSYNCTOG	6	VID_WEIGHT4
F	VID_HINT	7	VID_selmode

WE **Register ID Write Enable (bit 4)** **R/W**

When one, this bit field allows VRID data to be written.

VID_data Register

(initialization) **0x04**

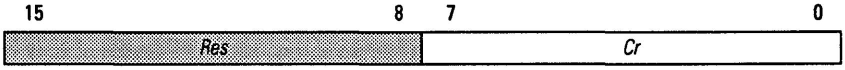


VRData **Video Register Data (bits 15:0)** **W**

The host processor uses this 16-bit field to read from and write to the indirect video register pointed to by *VRID* in VID_iaddr.

VID_Cr Register

(diagnostic) ***VRID* = A**

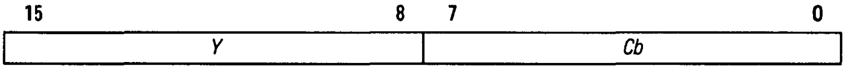


Cr **Border Cr Data (bits 7:0)** **W**

Diagnostic programs may use this 8-bit field to write Cr chrominance border color.

VID_YCb Register

(diagnostic) *VRID* **=B**



Y **Border Y Data (bits 15:8)** **W**

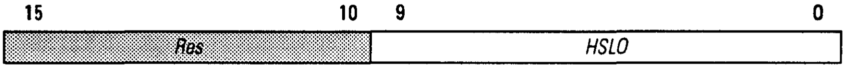
Diagnostic programs may use this 8-bit field to write Y luminance border color.

Cb **Border Cb Data (bits 7:0)** **W**

Diagnostic programs may use this 8-bit field to write Cb chrominance border color.

VID_HSYNC_LO Register

(diagnostic) *VRID* **= C**

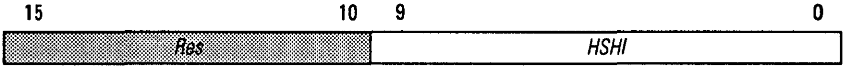


HSLO **(bits 9:0)** **W**

Diagnostic programs may use this 10-bit field to set the width of $\overline{\text{HSYNC}}$ low in units of pixel times.

VID_HSYNC_CYC Register

(diagnostic) *VRID* **= D**

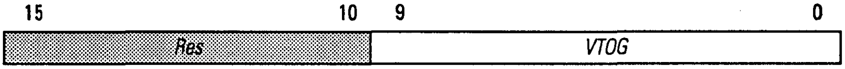


HSHI **(bits 9:0)** **W**

Diagnostic programs may use this 10-bit field to set the number of pixel times in the active video window plus horizontal blanking.

VID_VSYNCTOG Register

(diagnostic) *VRID* **= E**



VTOG **(bits 9:0)** **W**

Diagnostic programs may use this 10-bit field to set the $\overline{\text{VSYNC}}$ toggle point for the bottom field. VTOG is the number of pixel times from the falling edge of $\overline{\text{HSYNC}}$.

VID_HINT Register

(diagnostic) ***VRID*** **= F**

15 10	9 0
Res	*HINT*

HINT **(bits 9:0)** **W**

Diagnostic programs may use this 10-bit field to set the horizontal interrupt position. HINT is the number of VCKs from the falling edge of $\overline{\text{HSYNC}}$ where the video interrupt is requested. This is typically set to the right edge of active video.

VID_WEIGHT0 Register

(diagnostic) ***VRID*** **= 2**

15 11	10 0
Res	*W0*

VID_WEIGHT1 Register

(diagnostic) ***VRID*** **= 3**

15 11	10 0
Res	*W1*

VID_WEIGHT2 Register

(diagnostic) ***VRID*** **=4**

15 11	10 0
Res	*W2*

VID_WEIGHT3 Register

(diagnostic) ***VRID*** **= 5**

15 11	10 0
Res	*W3*

VID_WEIGHT4 Register

(diagnostic) ***VRID*** **= 6**

15 9	8 1	0
Res	*W4*	*0*

K4 – K0 ***RGB*** **Conversion Coefficients** **W**

These five weights are the coefficients for the YCbCr-to-RGB conversion performed by the internal color-space converter. The equations for the color-space conversion are:

$$Red = W4\,(Y - YOFF) + W0 \times Cr$$

$$Blue = W4\,(Y - YOFF) + W3 \times Cb$$

$$Green = W4\,(Y - YOFF) + W1 \times Cr + W2 \times Cb$$

Figure 10-2 shows the location of the binary points for W0 to W4. YOFF is either 0 or 16 depending on the YA bit in the VID_mode register.

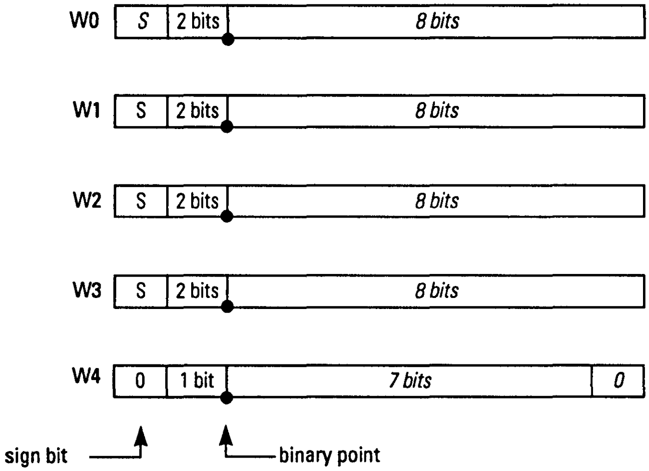


Figure 10-2 Binary Representations of Conversion Coefficients

To get an RGB color range of 16 to 235, as is often desired in consumer applications, the values of the coefficients should be set as shown in the left column. To get an RGB color range of 0 to 255, as is often desired in computer applications, the values of the coefficients should be set as shown in the right column.

16 to 235	0 to 255
W2 = 0x7A8	W2 = 0x79C
W3 = 0x1C6	W3 = 0x202
W1 = 0x749	W1 = 0x731
W0 = 0x167	W0 = 0x197
W4 = 0x100	W4 = 0x12A

VID_MODE Register

(diagnostic) ***VRID* = 7**

<table>
<tr><th>15</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="2">Res</td><td>H_E</td><td>Res</td><td>Ven</td><td>YA</td><td>VCK</td><td>V16</td><td colspan="2">Res</td><td>RGB</td><td>Res</td><td>Syn</td></tr>
</table>

H_E **$\overline{\text{HSYNC}}$ Mode Select (bit 11)** **W**

This bit controls the output mode of $\overline{\text{HSYNC}}$: 1 = $\overline{\text{HSYNC}}$ out enable, 0 = $\overline{\text{HSYNC}}$ forced to 0.

Ven **VCK Enable(bit 9)** **W**

This bit controls the output mode of VCK: 1 = enable, 0 = disable. This bit is valid in either VCK output or input mode.

YA **Y Offset (bit 8)** **W**

This bit controls theY offset: 1 = subtract 16, 0 = no adjust. To get a RGB color range of 0 to 255, YA should be set to one.

VCK **VCK Direction (bit 7)** **W**

This bit controls VCK input/output direction: 1 = output, 0 = input.

V16 **Video Bus Width (bit 6)** **W**

This bit controls the Vbus width in YCbCr mode: 1 = 16-bit, 0 = 8-bit.

RGB **RGB versus YCbCr (bit 2)** **W**

This bit controls video output mode: 1 = RGB, 0 = YCbCr.

Syn **Video Range (bit 0)** **W**

This bit controls $\overline{\text{VSYNC}}$ and $\overline{\text{HSYNC}}$ input/output direction: 1 = output, 0 = input.

11 Microcode Overview

The CL480 performs its higher-level functions by executing microcode on its internal CPU. The microcode is supplied by C-Cube Microsystems with the hardware.

The microcode must be loaded into the CL480 by a boot ROM or through the CL480 host interface; it needs to be loaded in specific sections of the local DRAM and on-chip IMEM. Additional information on loading and executing this code is contained in this chapter and on the distribution disk that is provided with the CL480 .

11.1 Process Configurations

The operation of the CL480 microcode and the host processor can be described by a set of "process configurations"—each a named group of software processes, some executing within the host processor and some executing within the CL480's microcode. Five different process configurations occur at different times in a CL480 system. These process configurations are shown in Figure 11-1 and Figure 11-2.

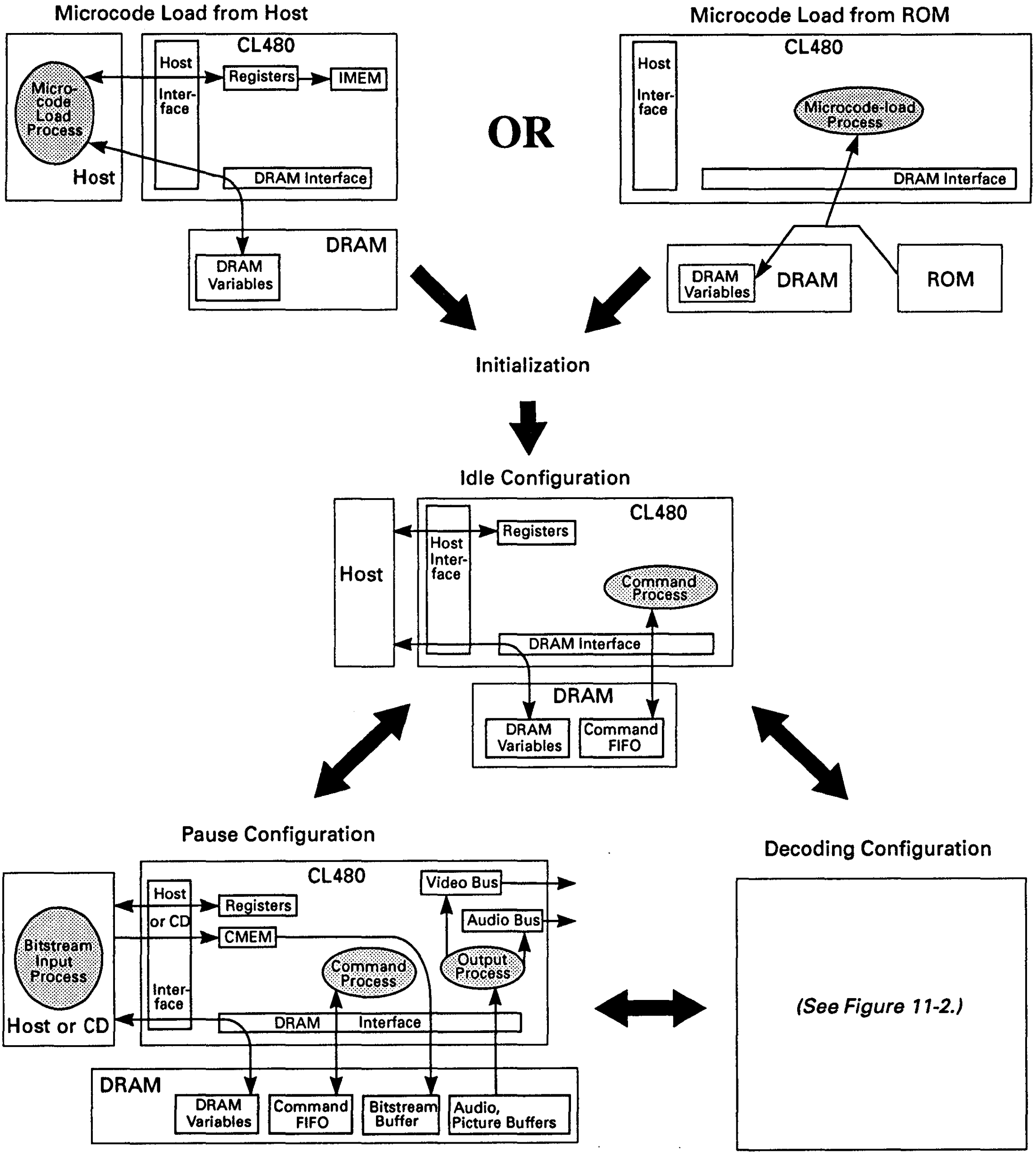


Figure 11-1 CL480 Microcode Process Configurations

Within each process configuration shown in Figure 11-1 and Figure 11-2, shaded ovals indicate individual but concurrently-running software processes executed by the host or the CL480, rectangles indicate hardware resources, and arrows indicate logical data transfers. (Physical buses used for data transfers are not indicated.) The large arrows pointing to separate process configurations in Figure 11-1 indicate possible microcode progress over time in a CL480-based system.

A total of six separate software processes are shown (shaded ovals) in the five process configurations of Figure 11-1 and Figure 11-2. They are Microcode-load, Initialization, Command, Bitstream Input, Decode, and Output. The first two of these processes are described in Chapter 12 (Initialization). The remainder of these processes execute simultaneously when the system is in the Decoding process configuration shown in Figure 11-2 and are described in more detail following this figure.

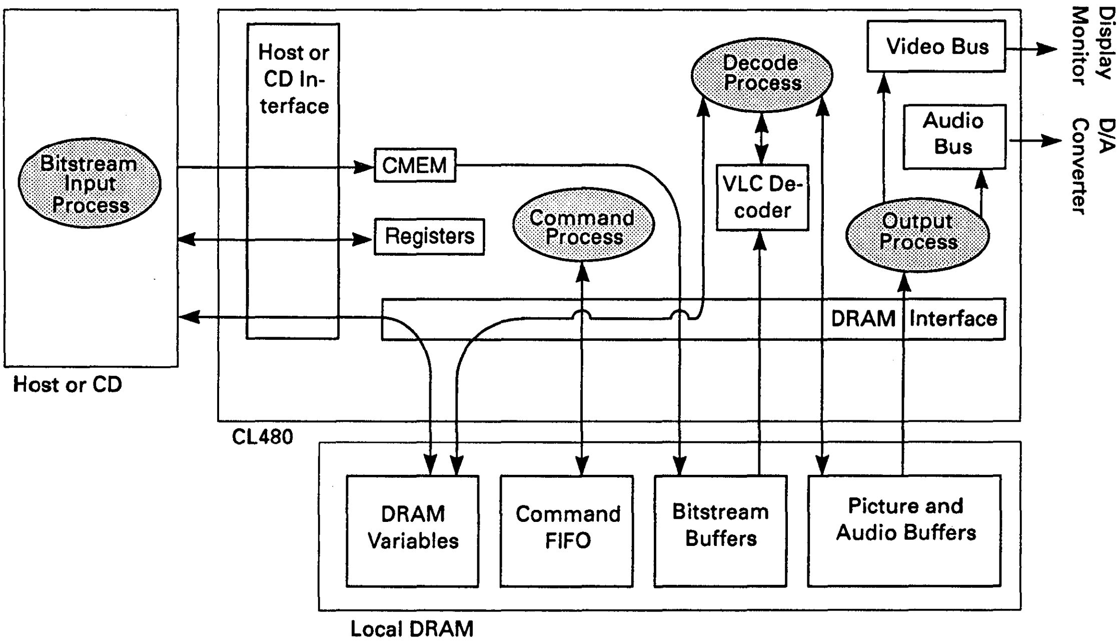


Figure 11-2 CL480 Decoding Process Configuration

11.1.1 Microcode-Load

The CL480 microcode-load procedure for systems both with and without a boot ROM is described below.

With Boot ROM

When the boot ROM is enabled (the ROM_ENA pin is pulled high and a ROM containing microcode is installed), the initialization procedure is:

1. Toggle the $\overline{\text{RESET}}$ pin.
2. Write a non-zero value to DRAM location 0xE0 (High Priority Command area Command ID described on page 12-2).
3. Poll DRAM location 0x0E for a zero value. When this occurs, microcode initialization is complete.
4. Issue commands (Play(), etc).

Without Boot ROM

When the boot ROM is disabled (ROM_ENA pulled low), the initialization procedure is as follows:

1. Toggle the $\overline{\text{RESET}}$ pin on the CL480 (which causes initialization of the on-chip DRAM controller).
2. Write microcode into CL480 DRAM as described on the distribution disk.
3. Write a non-zero value to DRAM location 0xE0 (High Priority Command area Command ID described on page 12-2).
4. Write startup microcode starting at address 0 of CL480 IMEM.
5. Set PC to the initial value (CPU_pc = 0).
6. Start the CPU by setting the *LS* bit field in CPU_cntl to 00.
7. Poll DRAM location 0x0E for a zero value. When this occurs, microcode initialization is complete.
8. Issue commands (Play(), etc).

11.1.2 Initialization

During initialization, The CL480 initializes all hardware interface areas according to the values stored in the Configuration Area of DRAM before setting the High-Priority Command ID to zero. Then the microcode enters the Command process, described next.

11.1.3 Command Process

The primary method by which the microcode executing on the CL480's CPU receives direction from the host software is by means of *macro commands*, which are function codes and parameter values written into DRAM by the host (see Table 11-1).

Macro commands are either low or high priority. Low priority commands are initiated by the host writing into the command FIFO in DRAM. High priority commands are written into the High Priority Command field in DRAM (see Section 12.3).

Table 11-1 Macro Command Summary

Category	Priority	Effect on Command State	Name	Description	Func. Code	Page
Set-type	Low	Yes	SetBorderColor()	Sets border color	0207	12-25
			SetErrorLevel()	Sets error level for DisplayStill()	0119	12-26
			SetInterruptMask()	Enables/disables interrupts to host	0104	12-27
			SetStreams()	Selects which streams to decode	0213	12-30
			SetVideoFormat()	Sets format to NTSC, PAL or progressive	0305	12-31
			SetWindow()	Sets video window size and location	0606	12-32
Play-type	Low	Yes	DisplayStill()[1]	Decodes/displays single still picture w/ audio	000C	12-11
			DisplayStill(start)	Same as above but specifies starting point	020C	12-12
			DisplayStill(start,stop)	Decodes/displays multiple still pictures w/ audio	040C	12-13
			DumpData()	Performs ECC on CD-ROM data	0314	12-14
			Freeze()	Freezes display but continues decoding	0010	12-17
			Pause()	Freezes display and decoding process	000E	12-19
			Play()	Decodes and displays at normal rate	000D	12-20
			Play(start, stop)	Same as above but specifies start/stop	040D	12-21
			Scan()	Decodes and displays next single I-picture	000A	12-24
			SingleStep()	Decodes and stores next single picture	000B	12-33
			SlowMotion()	Decodes and displays at slower rate	0109	12-34
Set-type	High	No	SetMute()	Sets audio mute and attenuation	8118	12-29
Control			FlushBitstream()[2]	Discards contents of bitstream buffer	8002	12-16
			InquireBufferFullness()	Measures data in bitstream buffer	8001	12-18
		Yes	Reset()	Reinitializes CL480 and its microcode	8000	12-23

[1] Allows display of a high-resolution still image with double horizontal and vertical resolution as specified in VideoCD 2.0.

[2] Not included in the alpha release of microcode.

Note: If a macro command is issued and the function code contains a value other than one of the values listed in the column above, indeterminate behavior will occur.

11.1.4 Bitstream Input Process

The CL480 accepts MPEG system streams or CD data streams which can contain:

- □ CD-ROM and CD-DA data
- □ MPEG-1 video streams.
- □ MPEG audio streams (Layer 1 or Layer 2)

Host Bitstream Transfer

Data can be transferred from the host to the CL480 using direct host writes to CL480's CMEM, a FIFO with thirty-one 16-bit locations. The host must check the CFLEVEL pin before writing to ensure that CMEM does not overflow.

CD Bitstream Transfer

As with host bitstream transfer, serial data can also be transferred from a CD-DSP to the CL480. For CD decoding, the CL480 stores the ROM header and sub-header information into a dedicated DRAM location called ROM Header and Subheader (see Section 12.3) which is accessible to the Host.

CL480 Bitstream Transfer

Once inside the CL480's CMEM, the data is transferred as a burst to the *bitstream buffers* located in the local DRAM. (The bitstream buffer is called the *rate buffer* in the MPEG standard.)

When the DRAM bitstream buffer is full, the input process will be disabled. Optionally, the decoder can be configured to handle a near buffer-full condition automatically and thus prevent buffer overflow.

11.1.5 Decode Process

The decode process is the process by which the CL480 decompresses the input bitstream using the MPEG decoding algorithm and places the decompressed audio and video frames (as the case may be) in their respective buffers in the CL480's local DRAM.

The decoding process pauses if the bitstream buffer is empty, the audio output buffer is full, or if no space is available for writing the decoded frame. Such a condition could occur if the display of the previous frame was not completed before the decoding process was ready to begin writing to its frame buffer.

> *Note: The decode process is interlocked with the Display process and the Input process and continues only if data is available in the input bitstream buffer and space is available in the output frame buffer.*

11.1.6 Output Process

The output process in the CL480's microcode handles the transfer of decoded video and audio data to the video and audio buses, respectively, from where data is passed to the display monitor and DAC converter.

11.2 Error Handling

The CL480 has the ability to detect errors in the MPEG bitstream or to react to error start codes inserted in the stream by invoking error concealment microcode, which minimizes the artifacts introduced as a result of the error. Each stream type—video, audio, system layer—uses different error concealment procedures. Error concealment is always turned on, so no setting is required.

11.3 Interrupts

The CL480 provides an interrupt output pin, $\overline{\text{INT}}$, to the host processor which allows the microcode to alert the host when certain events occur.

The host selects the interrupt events of which it wishes to be informed by using the SetInterruptMask() macro command (see page 12-27) to assign mask bits to one of 13 logical interrupts as shown in Figure 11-3.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A/E/E	*END-D*	*END-P*	*USR*	*SCN*	*RDY-S*		*UND*	*AOR*			*END-V*	*SEQ-V*	*GOP-V*	*PIC-V*	*ERR*

Figure 11-3 Mask Bit Allocation

When an interrupt occurs, the event which caused the interrupt is signalled in the INT_STATUS word in the DRAM Status Area, and $\overline{\text{INT}}$ is asserted (active low).

To respond to an interrupt, the host should:

1. Read the DRAM INT_STATUS location in the Status Area.
2. Handle the interrupt.
3. Write 0 to the *Int* bit of register INT_cntl using a read-modify-write.
4. Write 0 to the INT_STATUS location in the DRAM Status Area.

Each of the 13 logical interrupts produced by the CL480 microcode belongs to one of two categories (based on when they are reported to the host) and is referenced by name and the event which causes it in Table 11-2. The interrupts are described in more detail in the SetInterrupt-Mask() macro command description given in Chapter 12, Interrupts.

Table 11-2 CL480 Interrupt Summary

Category	Interrupt Name	Event	Mask Bit
Decode-time	AOR	Address out of range	7
	A/E/E	Submode is auto pause, end of record/file	15
	END-D	End of DumpData() command	14
	END-P	End of Pause(), SingleStep(), or SlowMotion()	13
	ERR	Bitstream data error	0
	SCN	Picture decode complete in Scan()	11
	UND	Bitstream buffer underflow error	8
	RDY-S	Ready for data during SlowMotion()	10
	USR	User Data Ready	12

Category	Interrupt Name	Event	Mask Bit
Display-time	END-V	Last picture display before sequence_end_code	4
	GOP-V	First I-picture display after group_start_code	2
	PIC-V	New picture display	1
	SEQ-V	First I-picture display after sequence_header_code	3

11.4 Starting Operation

Full-rate MPEG decoding and display operation is started by the host by storing a Play() command into the command FIFO in CL480's DRAM. When the CL480 recognizes the Play() command, it will update the command status in the status area and start to execute the command.

The Play() command will cause decoding to begin at the first packet start code received. The CL480 automatically handles synchronization.

11.5 Bitstream Buffer Under- /Overflow

Bitstream buffer underflow and overflow are exception events which should not occur in properly operating systems. For video processing, the decoder responds to bitstream buffer underflow by implementing a repeat picture operation until the bitstream buffer fullness is sufficient to decode the next frame in the sequence. In the case of audio, the audio output is muted. Buffer underflow is signalled to the host through an interrupt.

The decoder responds to bitstream buffer overflow by not accepting input data.

11.6 Programmable I/Os

The CL480 provides programmable I/Os that may be used for control information. These I/Os become available by disabling the host interface (pull pin HOST_ENA low), which enables host interface pins to be used for control functions as shown in the following tables.

Pin Name	Name/Function	I/O
HSEL[0]	STOP/PLAY: 1 = Pause or Freeze, 0 = Play	I
HSEL[1]	SCAN: 1 = Scan	I

Note: In the table below, HD[0] and HD[1] perform the same functions normally set by SetMute().

HD[0] (MUTE0)	HD[1] (MUTE1)	Attenuation on the PCM Samples
0	0	0 dB
1	0	-12 dB
0	1	-∞ (20 ms soft mute)
1	1	-12 dB

11.7 Fast-Forward and Fast-Reverse

The CL480 has a macro command called Scan() that searches for the next intra picture and decodes and displays it. The user data of the MPEG video bitstream inVideo CD 2.0 has an optional pointer to the next two future and two previous sectors that contain the beginning of an intra picture. The CL480 stores the user data in DRAM when it parses the video bitstream. The host can read the user data and tell the CD-DSP to seek to the appropriate sector. If the disc does not contain pointers to the sectors with intra pictures, the host must guess where the desired intra picture starts. The quality of fast forward and fast reverse may be better if the host plays the video for a short time (maybe 1/2 or 1/4 second) using the Play() command before going to the next intraframe.

12 Macro Commands

The host software and the CL480's microcode use macro commands as their primary method of communication. Macro commands are function codes and parameter values written into the local DRAM by the host. Each command has a separate function code and may have 0 to 4 parameters.

Once the function code and parameters are written, the microcode acts on them according to their priority:

- □ *High-priority*: These commands start execution as soon as the CL480's microcode detects their presence and complete execution *before* another high-priority command may be issued.
- □ *Low-priority*: These commands are stored by the CL480 in the Command FIFO and executed by the CL480 in the order that they were received.

12.1 High Priority Commands

High-priority commands have the priority bit set in their command number. The high-priority commands are Reset(), FlushBitstream(), InquireBufferFullness(), and SetMute(). The high priority commands are not stored into the command FIFO; instead, they are written into a separate command buffer, the High Priority Command Area. The sequence for writing High-priority commands is as follows:

1. The host processor polls the High-Priority command ID location and waits until it is zero before giving a command to the CL480.
2. The host writes the arguments for the new command.
3. The host writes the command ID.

Note: The host can write the command ID field to a non-zero value during initialization; it will become zero when initialization is complete.

12.2 Low Priority Commands

Low-priority commands are extracted from the DRAM Command FIFO and executed by a low-priority microcode task. The frequency at which the command FIFO gets processed depends on the operation being performed by the CL480. During normal play operation, the command FIFO will be serviced at intervals up to 50 ms in duration. When the CL480 microcode determines there are commands present in the command FIFO, then the commands are extracted and executed one by one until the command FIFO is empty.

When a Play-type command— Play(), Pause(), Freeze(), SingleStep(), SlowMotion(), Scan(), DisplayStill(), or DumpData()—is encountered in the command FIFO, it is extracted and executed immediately. If the command was not Pause() or DumpData(), the CL480 microcode will then decode the next MPEG video picture before checking the command FIFO again. For Pause() and DumpData(), the CL480 will check the command FIFO immediately after completing execution of the command.

When a Set-type (low-priority) command—SetStreams(), SetWindow(), SetBorderColor(), SetVideoFormat(), SetInterruptMask(), or DisplayGraphics()—is encountered in the command FIFO, it is extracted and executed immediately. The CL480 checks the command FIFO immediately after completing execution of the Set-type command and, if the command FIFO is not empty, executes the next command.

Note: When writing macro commands to the Command FIFO, you do not need to fill the Command FIFO with a fixed number of bytes; only write as many 16-bit words as the command needs.

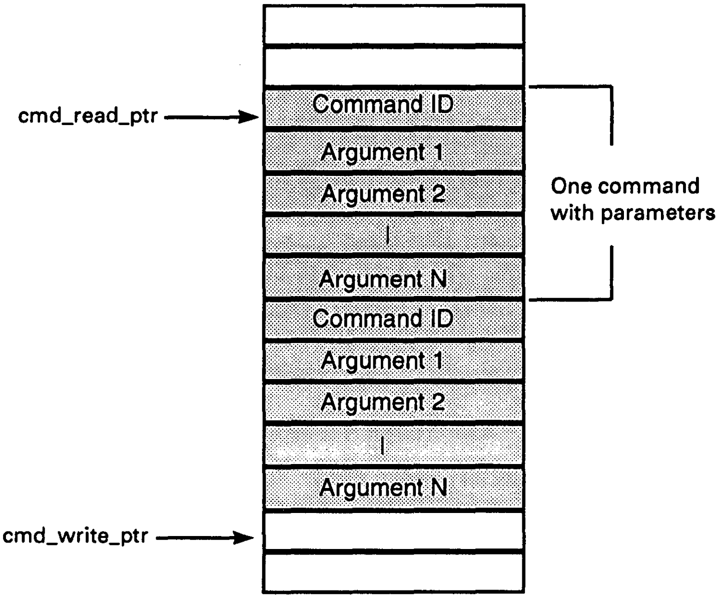
12.2.1 Command FIFO

The CL480 maintains a Command FIFO in DRAM through which the host initiates most microcode operations. For example, the Play() command must be written into the command FIFO to start normal MPEG decode and display.

The Command FIFO has read and write pointers which are also located in DRAM. The host processor maintains the write pointer; it must read the write pointer from DRAM, write the command(s) and arguments (if any) into the Command FIFO, and write back the new value of the write pointer into DRAM.

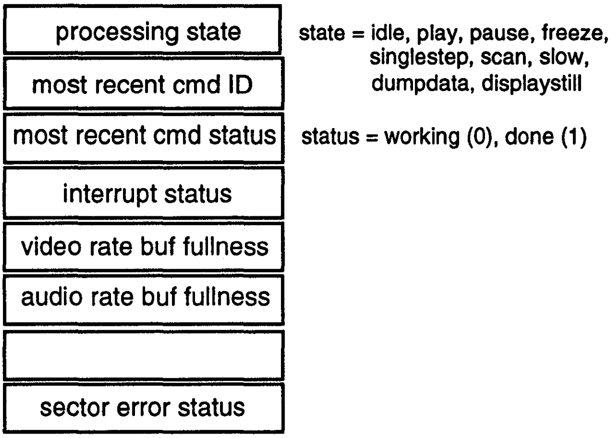
Since the Command FIFO is maintained as a circular buffer, the host processor must reset the write pointer back to the start of the Command FIFO when the last location has been reached. It is therefore possible for a single command to be stored as two non-contiguous pieces: the first part at the end of the Command FIFO, and the second part at the beginning.

The Command FIFO has the structure shown in the figure below.



When the read pointer equals the write pointer, the Command FIFO is defined to be empty. This definition requires that the Command FIFO must at all times have at least one unused location.

A status area is maintained in DRAM by the CL480 which gives the status of the currently executing or just completed command and other information. The structure of the status area is shown below and on the next page.



12.2.2 User Data FIFO

The CL480 user data FIFO is almost exactly the same as the Command FIFO just described. Is is maintained as a circular buffer, with the CL480 microcode maintaining the write pointer (any user data encountered is written here), and the host processor maintaining the read pointer. The host compares the read and write buffer to determine if any data exists in the read or write pointers.

12.3 Writing Macro Commands

Macro commands are written into local DRAM by the host as described in Section 4.2.4, DRAM/ROM Access.

When local DRAM_data is read, the contents of the currently selected local DRAM location are returned to the host. When DRAM_data is written by the host, the currently selected DRAM location is written with the same data.

CL480 DRAM memory locations are shown below:

	Status Area
0xC0	PROC_STATE: Processing State (includes Play, Freeze, Pause, DisplayStill, Scan, SingleStep, SlowMotion)
0xC2	MRC_ID: Most recent command ID (either currently or most-recently executed command)
0xC4	MRC_STATUS: Most recent command status (0 = in progress; 1 = done)
0xC6	INT_STATUS: Interrupt status (INT - follows order of Figure 11-3)
0xC8	VRB_FULLNESS: Video rate buffer fullness (in multiples of 16 bytes)
0xCA	ARB_FULLNESS: Audio rate buffer fullness (in multiples of 16 bytes)
0xCE	SE_STATUS: Sector error status (See DumpData() command)
	High Priority Command
0xE0	Command ID (See Section 12.1)
0xE2	ARGUMENT1: Argument 1
0xE4	ARGUMENT2: Argument 2
0xE6	ARGUMENT3: Argument 3
0xE8	ARGUMENT4: Argument 4
0xEA	ARGUMENT5: Argument 5
0xEC	ARGUMENT6: Argument 6
0xEE	ARGUMENT7: Argument 7
	Pointers
0xF0	CMFD_SADDR: Command FIFO start address (See Section 12.2.1, Command FIFO)
0xF2	CMFD EADDR: Command FIFO end address
0xF4	UDF_SADDR: User data FIFO start address (See Section 12.2.2, User Data FIFO)
0xF6	UDF_EADDR: User data FIFO end address
0xF8	CMFD_READ: Command FIFO read pointer
0xFA	CMFD_WRITE: Command FIFO write pointer
0xFC	UDF_READ: User data FIFO read pointer
0xFE	UD_WRITE: User data FIFO write pointer
	⋮

	⋮	
0x100	**Command FIFO**	
0x200	**User Data FIFO**	This FIFO contains application-specific data from the MPEG video bitstream. The user data FIFO contains the last user data read from the video bitstream.
0x300	**Video Parameters**	
	ROM Header and Subheader	
	Only valid in Pause or Idle	
	ROM_HEADER_BASE:	
0x330	Minutes, Seconds (in separate bytes; packed BCD format)	
0x332	Frame, Mode (in separate bytes; packed BCD format)	
0x334	File Number, Channel Number	
0x336	SubMode, Coding Information	
0x340	**MPEG `sequence_header` Parameters**	
	horizontal_size	
	vertical_size	
	pel_aspect_ratio	
	picture_rate	
	bit_rate upper 15 bits	
	bit_rate lower 3 bits	
	vbv_buffer_size	
	constrained_parameter_flag	
	load_intra_quantizer_matrix	
	load_non_intra_quantizer_matrix	
	⋮	

	⋮
0x354	**GOP Header Parameters** time_code upper 12b+1b marker time_code lower 12 bits closed_gop broken_link
0x35C	**Picture Header Parameters** temporal_reference picture_coding_type vbv_delay full_pel_forward_vector forward_f_code full_pel_backward_vector backward_f_code
0x380	**Configuration Area*** Video bitstream buffer start (in multiples of 512 bytes) Video bitstream buffer end (in multiples of 512 bytes) Audio bitstream buffer start (in multiples of 512 bytes) Audio bitstream buffer end (in multiples of 512 bytes) Audio output buffer start (in multiples of 512 bytes) Audio output buffer end (in multiples of 512 bytes) reserved reserved reserved reserved Initial value of Prog_I/O 1 register (misc. functions) Initial value of Audio_mode register (audio config) Initial value of CD_cnfg register (CD-ROM config) Initial value of VID_csync register (interlaced,vsync pin,CD/I...) Initial value of ROM_mode register (Sets Rom acces time... Initial value of DRAM_ref register (sets refresh counter) Initial value of SPU_audave register (enables R+L mode) Initial value of VID_interpcoeff register (hor,vert interpolation coeffs) Initial value of VID_weightk0 register (K0 coef. of YUV to RGB conversion) Initial value of VID_weightk1 register (K1 coef. of YUV to RGB conversion) Initial value of VID_weightk2 register (K2 coef. of YUV to RGB conversion) Initial value of VID_weightk3 register (K3 coef. of YUV to RGB conversion) Initial value of VID_weightk4 register (K4 coef. of YUV to RGB conversion)
	⋮

*Stored as part of microcode file and downloaded to DRAM with microcode.

•
•
•

Configuration Area (cont.)

Initial value of VID_selmode register (Misc video: hsync i/o, vclk, conv. offset,...)
Initial value of VID_active register (Active region config)
Initial value of VID_leftbor register (left border)
Initial value of VID_colorr register (border color: Cr)
Initial value of VID_colorYCb register (border color: Y, Cb)
Initial value of VID_hsync_low register (width of hsync low)
Initial value of VID_hsync_hi register (width of hsync hi)
Initial value of VID_vstp register (vsync toggle point)
Initial value of VID_hintp register (horizontal interrupt position)

Default Starting layer for Play() command
(layer to begin search for sync/start codes,
0=CD-ROM, 1= MPEG system, 4=CD-DA)
Default DisplayStill() hi-res still transition mode
(0=blank screen first, 1=overwrite previous still)
Default value of DumpData() Length parameter
Default value of DumpData() Address parameter (word address)
Base address for DumpData() Address parameter (16-byte address)
Default value of SetErrorLevel() ErrorLevel parameter
Default value of SetInterruptMask() IntEventEna parameter
Default value of SetMute() MuteSetting parameter
Default value of SetStreams() VideoStreamID parameter
Default value of SetStreams() AudioStreamID parameter
Reserved
Default value of SetVideoFormat() Format parameter
Default value of SetVideoFormat() Interlaced parameter
Default value of SetWindow() LeftBorder parameter
Default value of SetWindow() TopBorder parameter
Default value of SetWindow() XOffset parameter
Default value of SetWindow() YOffset parameter
Default value of SetWindow() Width parameter
Default value of SetWindow() Height parameter

Default values for MPEG sequence header parameters
(during execution of all Play-type commands except DisplayStill):
Horizontal size
Vertical size
Picture Rate

Default values for MPEG sequence header parameters
(during execution of DisplayStill() command):
Horizontal size
Vertical size

0x400 **Microcode (31K bytes)**

0x8000 **Bitstream Buffer**
(During DumpData command,data buffer)

12.4 Macro Command Groups

CL480 macro commands are divided into three functional categories:

- □ Set-type
- □ Play-type
- □ Control

Each command group has distinct properties, which are described below, in Table 12-1.

12.4.1 Set-type Commands

The CL480 has seven Set-type macro commands, all of which:

- □ Never affect the command state
- □ May be issued regardless of the current command state
- □ Have no effect on the decoding process, except for SetVideoFormat(), which helps determine if video sequences are transcoded

12.4.2 Play-type Commands

The CL480 has eleven Play-type macro commands, each of which causes the:

- □ Current command state to change
- □ Macro command processing to be suspended
- □ Microcode to transition into the PLAY-SETUP state and the CL480 to be configured to accept bitstream data

12.4.3 Control Commands

The CL480 has three Control macro commands, each of which:

- □ Can be issued regardless of the current command state
- □ Is high priority

Table 12-1 Macro Command Summary

Category	Priority	Effect on Command State	Name	Description	Func. Code	Page
Set-type	Low	Yes	SetBorderColor()	Sets border color	0207	12-25
			SetErrorLevel()	Sets error level for DisplayStill()	0119	12-26
			SetInterruptMask()	Enables/disables interrupts to host	0104	12-27
			SetStreams()	Selects which streams to decode	0213	12-30
			SetVideoFormat()	Sets format to NTSC, PAL or progressive	0305	12-31
			SetWindow()	Sets video window size and location	0606	12-32
Play-type	Low	Yes	DisplayStill()[1]	Decodes/displays single still picture w/ audio	000C	12-11
			DisplayStill(start)	Same as above but specifies starting point	020C	12-12
			DisplayStill(start,stop)	Decodes/displays multiple still pictures w/ audio	040C	12-13
			DumpData()	Performs ECC on CD-ROM data	0314	12-14
			Freeze()	Freezes display but continues decoding	0010	12-17
			Pause()	Freezes display and decoding process	000E	12-19
			Play()	Decodes and displays at normal rate	000D	12-20
			Play(start, stop)	Same as above but specifies start/stop	040D	12-21
			Scan()	Decodes and displays next single I-picture	000A	12-24
			SingleStep()	Decodes and stores next single picture	000B	12-33
			SlowMotion()	Decodes and displays at slower rate	0109	12-34
Set-type	High	No	SetMute()	Sets audio mute and attenuation	8118	12-29
Control			FlushBitstream()[2]	Discards contents of bitstream buffer	8002	12-16
			InquireBufferFullness()	Measures data in bitstream buffer	8001	12-18
		Yes	Reset()	Reinitializes CL480 and its microcode	8000	12-23

[1] Allows display of a high-resolution still image with double horizontal and vertical resolution as specified in VideoCD 2.0.

[2] Not included in the alpha release of microcode.

Note: If a macro command is issued and a function code contains a value other than one of the values listed in the Function Code column above, indeterminate behavior occurs.

12.5 Macro Command Reference

All CL480 macro commands are listed alphabetically in the pages that follow.

DisplayStill()

Format:	DisplayStill()
Priority:	Low
Category:	Play-type
Command ID	000C

The DisplayStill() command decodes and displays a single picture. It flushes the bitstream buffer, decodes the next intra-coded still picture, and displays it. At the conclusion of this command, the CL480 transitions to the "freeze" state. Therefore, audio is enabled and continues to be decoded and output after the still picture is displayed.

The resolution of the still picture is determined by the dimensions coded in the video sequence header. For Video CD, this command should be preceded by a SetStreams() command which selects the VideoStreamID to be either 0xE1 (normal resolution, 352x240 or 352x288) or 0xE2 (high resolution, 704x480 or 704x576).

For normal resolution (352x240 or 352x288), the video display is frozen while decoding takes place. When decoding is complete, display is switched to the newly decoded still picture.

For high resolution (704x480 or 704x576), there are two picture transition modes: overwrite and blanking. For overwrite mode, the picture is displayed as it is decoded with the newly decoded still picture filling the display from top to bottom. For blanking mode, the display is immediately blanked to the background color at the start of decoding a new picture. Regardless of the mode, if the previous command was displaying normal resolution video, the display is immediately blanked.

DisplayStill (start1, start2)

Format: DisplayStill(**start1, start2**)
Priority: Low
Category: Play-type

Command ID: 020C

This command functions the same as DisplayStill() but starts decoding the still picture from specified starting point (**start2** must be zero for Video CD). This command terminates after the first still picture is decoded.

DisplayStill(**start1**, **start2**) arguments are defined as follows:

```
short int Start1,Start2; /* Location to start decoding (Minutes, Seconds, Frame)  */
                          /* Start1 is comprised of bytes: Minutes, Seconds (in
                            packed BCD format)*/
                          /* Start2 is comprised of bytes:  Frame (in packed BCD for
                           mat), Mode (=2) */
```

DisplayStill(start1, start2, stop1, stop2)

Format: DisplayStill(**start1, start2, stop1, stop2**)
Priority: Low
Category: Play-type

Command ID: 040C

This command functions the same as DisplayStill() but starts decoding the still picture from a specified starting point (**start2** must be zero for Video CD). This command will decode and display a sequence of still pictures until the stop point is reached (i.e., a "slide show"). Audio decoding begins when the first audio access unit is detected.

DisplayStill(**start1, start2, stop1, stop2**) arguments are defined as follows:

```
short int Start1,Start2;   /* Location to start decoding (Minutes, Seconds, Frame)  */
                           /* Start1 is comprised of bytes: Minutes, Seconds (in
                              packed BCD format*/
                           /* Start2 is comprised of bytes:  Frame (in packed BCD
                              format), Mode (=2)      */
short int Stop1, Stop2;    /* Location to stop displaying (Minutes, Seconds,Frame)  */
                           /*    Stop1 is comprised of bytes:  Minutes, Seconds (in
                                packed BCD format*/
                           /*   Stop2 is comprised of bytes:  Frame(in packed
                                BCD format), Mode (=2)    */
```

DumpData()

Format:	DumpData **(start1, start2, length, address)**
Priority:	Low
Category:	Play-type
Syntax:	0314

The DumpData command performs error correction on CD-ROM data. It reads Mode 1 or Mode 2 Form 1 data from disk from a specified starting point and for the specified length (up to a maximum of 16 sectors). DumpData() corrects data using ECC information, if necessary, and reads up to 16 sectors (32 KBytes) into a buffer in DRAM. The starting address of this DRAM buffer is specified by the Address parameter (16-bit word address).

At the conclusion of this command, the last sector address read and the sector error status are stored into the Status Area. (Note: Only the first subheader is saved to be read by the host.) The sector error status is a word which reflects which sectors were read error-free or were corrected using ECC. Each bit of the sector error status designates the error status for a particular sector: bit 15 corresponds to the first sector read and bit 0 to the 16th sector read. A "1" in a particular bit position indicates that the corresponding sector was uncorrectable. If the length is less than 16, then unused bits of the sector error status are cleared. After updating the Status Area and if the END-D bit has been set by the SetInterruptMask() command, a host interrupt is generated.

> *Note: If the data read from the specified location is not Mode1Form1 or Mode2Form1, then its validity is not assured.*

DumpData() arguments are defined as follows:

```
short int Start1,Start2; /* Location to start reading (Minutes,Seconds,Frame) */
                        /* Start1 is comprised of bytes: Minutes,Seconds (in
                            packed BCD format*/
                        /* Start2 is comprised of bytes: Frame(in
                            packed BCD format*/,Mode (=2) */
short int Length;        /* Length of sequence to read (maximum of 16 sectors)     */
short int Address;       /* Address in CL480 local DRAM to store the corrected   */
                         /* data (16-bit word address).It is recommended that*/
                         /*the address parameter point to a buffer within   */
                         /*the bitstream buffer area.The size of this buffer is */
                         /* Length*2048 bytes. The entire corrected data buffer */
                         /* must reside within ratebuffer space as designated     */
                         /* in the configuration area. */
```

FlushBitstream()

Format:	FlushBitstream
Priority:	High
Category:	Control
Command ID:	8002

The FlushBitstream() macro command causes the CL480 to delete the unprocessed contents of all video and audio bitstream buffers.

The CL480 continues to display the most-recently decoded picture after FlushBitstream() has been issued.

Freeze()

Format: Freeze()
Priority: Low
Category: Control

Command ID: 0010

The Freeze() command causes the display to freeze but allows the CL480 to continue decoding. In the freeze state, all decoded data is discarded.

Freeze applies to video only. Audio output continues normally.

When the host interface is disabled (HOST_ENA = 0), the Freeze() command may be initiated by resetting the host interface pin, HD[1] (MUTE1), to zero and setting the host interface pin, HSEL[0] (STOP/PLAY), to one.

InquireBufferFullness()

Format: InquireBufferFullness()
Priority: High
Category: Control

Command ID: 8001

The InquireBufferFullness() command returns the amount of data currently used in the video and audio bitstream buffers. The fullness of each buffer is represented as a multiple of 16 bytes and is written into the Status Area in DRAM.

Pause()

Format:	Pause()
Priority:	Low
Category:	Play-type
Syntax:	000E

The Pause() macro command freezes the display and holds the decoding processes. Audio output is disabled.

When the host interface is disabled (HOST_ENA = 0), the Pause() command may be initiated by setting the host interface pin, HD[1] (MUTE1), to one and setting the host interface pin, HSEL[0] (STOP/PLAY), to one.

Play()

Format:	Play()
Priority:	Low
Category:	Play-type
Command ID:	000D

Compressed data input is enabled and decoding begins at the first packet start code. Synchronization of the video and audio streams takes place by comparing presentation time stamps passed in the data stream with the system time clock.

After the Play() command is issued, the CL480 begins a hierarchical search through the CD-ROM and MPEG layers. The CL480 will first look for sector sync. In the event that no sector sync is found in the first 8K bytes, the CL480 searches for MPEG system layer start codes (pack, packet, or system header). If, after an appropriate period, no system layer start code is found, the CL480 will switch to CD-DA mode. Subsequently, if a sector sync is found and the format is Mode 2 Form 2, the CL480 will switch back to normal playback with CD-ROM decoding.

The CL480 will by default begin this hierarchical search at the CD-ROM layer. The starting layer for this search can be changed by setting the starting_layer variable in the configuration area (0=CD-ROM, 1= MPEG system, 4=CD-DA).

When the host interface is disabled (HOST_ENA = 0), the Play() command is executed only while the host interface pin, HSEL[0], is zero. If HD[0] (MUTE0) is set to one, then either the Pause() or Freeze() commands are executed.

> *Note: If the CD-ROM decoder encounters the Auto Pause in sector data while the Play() command is executing, the CL480 does NOT pause automatically.*

Play(start1, start2, stop1, stop2)

Format: Play(**start1, start2, stop1, stop2**)
Priority: Low
Category: Play-type

Command ID: 040D

This command functions the same as the Play() command described on the previous page, but only plays sequences from a specified starting point on disk to the specified stopping point. Video decoding begins when the first "I" picture is detected. Audio decoding begins with the first audio access unit is detected. Video decoding and display continues after the stop point is reached for all complete compressed pictures stored in the video rate buffer. At this command's conclusion, the CL480 transitions to the "pause" state.

Immediately after this command is issued, the CL480 begins looking for an address greater than or equal to the start point and less than the stop point. During this time, if an address is detected which is greater than or equal to the stop point, the CL480 will terminate the command immediately and, if the AOR bit has been set by the SetInterruptMask() command, cause a host interrupt to be generated. When the CL480 finds an address which is greater than or equal to the start point and less than the stop point, it initiates full rate CD-ROM and MPEG decoding and display. During this time, if an address is detected which is less than the start point or greater than or equal to the stop point, the CL480 will terminate the command and, if the AOR bit has been set by the SetInterruptMask() command, cause a host interrupt to be generated.

The Play(**start1, start2, stop1, stop2**) arguments are defined as follows:

```
short int Start1,Start2;  /* Location to start playing (Minutes, Seconds, Frame)  */
                          /* Start1 is comprised of bytes: Minutes, Seconds (in
                             packed BCD format*/
                        /* Start2 is comprised of bytes:  Frame (in
                             packed BCD format*/, Mode (=2)      */
short int Stop1, Stop2;   /* Location to stop playing (Minutes, Seconds,Frame)  */
                        /*    Stop1 is comprised of bytes:  Minutes, Seconds (in
                             packed BCD format*/
                        /*    Stop2 is comprised of bytes:  Frame (in
                             packed BCD format*/, Mode (=2)    */
```

If this command is issued with all four arguments set equal to zero after a Pause() command has been executed, the CL480 will start at the first sector after where it had paused. Thus, this command is useful for restarting after a Pause() as in the sequence of Play(), Pause(), wait for END-P interrupt, seek the CD, Play(=0, =0, =0, =0).

Reset()

Format: Reset()
Priority: High
Category: Control

Command ID: 8000

Reset() is a high-priority macro command which is used to halt the execution of the current command and re-initialize the CL480 and its microcode. When this command is executed:

- □ The contents of the bitstream buffer, the Command FIFO, and the picture buffers are lost.
- □ The video display process is re-initialized and the output window blanked (screen is filled with current border color).
- □ The CL480 enters the "idle" state and is ready to accept the next command.

Note: This is the only high-priority macro command which has an effect on the command processing state.

The Reset() macro command is typically used only to recover from error conditions. When suspending and resuming decode operations, or when changing from decoding one bitstream to another, some combination of the FlushBitstream() and Pause() commands should be used. Unlike the Reset() command, these commands allow the CL480 to continue to receive bitstream data and to continue to display the last picture decoded.

The Reset() command is done when the processing state is set to done.

Scan()

Format:	Scan()
Priority:	Low
Category:	Play-type
Command ID	000A

The Scan() command searches for the first “I” picture in the buffer, decodes it, and displays the picture. If the "I" picture includes user data, the CL480 will extract the user data, store it into DRAM, and interrupt the host if the USR bit has been set by the SetInterruptMask() command. After decoding is complete, the CL480 flushes the bitstream buffer, interrupts the host if the SCN bit has been set by the SetInterruptMask() command, and waits for the next command.

The CL480 will decode and play back any audio sectors it finds while searching for the "I" picture. Audio output can optionally be disabled through the SetMute() command.

When the host interface is disabled (HOST_ENA = 0), the Scan() command may be initiated by setting the host interface pin, HSEL[1] (SCAN), to one. At the completion of the command, the CL480 sets the $\overline{\text{INT}}$ pin to one.

The decoded I-picture is stored in one of the picture buffers in the CL480’s DRAM and is immediately posted for display; it remains in DRAM and is displayed until the next Play-type macro command is executed.

SetBorderColor()

Format: SetBorderColor(**Cr-Border, YCb-Border**)
Priority: Low
Category: Set-type

Command ID: 0207

The SetBorderColor() command sets the border colors. The new border color parameters will take effect after the next vertical blanking period.

SetBorderColor() arguments are defined as follows:

```
short int Cr-Border;      /* Cr (bits 7:0) component of border color */
short int YCb-Border;     /* Y (bits 15:8) and Cb (bits 7:0) of border color*/
```

SetErrorLevel()

Format: SetErrorLevel(**ErrorLevel**)
Priority: Low
Category: Set-type

Command ID: 0119

The SetErrorLevel() command sets the error level for use by the DisplayStill() command. The error level determines the action taken by the CL480 after reading video or audio data which has been corrupted.

The SetBorderLevel() argument is defined as follows:

```
short int ErrorLevel;  /* Error processing level */
                         /* 0 = Abort your error and do not display*/
                         /* 1 = Use concealment and continue decoding
```

SetInterruptMask()

Format:	SetInterruptMask(**IntEventEna**)
Priority:	Low
Category:	Set-type
Command ID:	0104

The SetInterruptMask() macro command is used to enable interrupt events and select the interrupt lines which are logically connected to each event. A "1" in a particular bit position of **IntEventEna** will enable an interrupt for the event corresponding to that bit. When one or more interrupt events occur, the Interrupt Status Word in the Status Area will be updated to reflect which interrupt events occurred and the interrupt pin, $\overline{\text{INT}}$, will be set to 1.

The SetInterruptMask() command argument is defined as follows:

```
short int IntEventEna;     /* Interrupt event enable (1= event enabled)      */
```

The assignment of **IntEventEna** bits is shown on the next page.

Table 12-2 **Mask Bit Assignments**

Mask Bit	Interrupt Name	Description
0	ERR	Bitstream data error - The CL480 has detected an erro code or other data format error in the data stream.
1	PIC-V	New picture display - The CL480 has detected a picture start code and the new picture is about to be displayed for the first time.
3	SEQ-V	Sequence header display - The CL480 has detected a Sequence header start code and the first picture of the sequence is about to be displayed for the first time.
7	AOR	Address Out of Range - During execution of the play(start1, start2, stop1, stop2) command, a sector header address has been detected which is out of range.
8	UND	Buffer underflow - The bitstream buffer is empty and the CL480 is waiting for more data.
10	RDY-S	Ready for data during Slow Motion - The CL480's video rate buffer fullness has fallen below a threshold and the CL480 is requesting more data.
12	USR	User Data ready - The CL480 has decoded the User Data of the picture header and has transferred it to DRAM.
13	END-P	End of Pause, SingleStep, or SlowMotion - The CL480 has finished execution of the Pause(), SingleStep(), or SlowMotion() commands and has updated the status area with the last sector address. This bit is ignored during execution of other commands.
14	END-D	End of DumpData command - The CL480 has completed the DumpData command and the corrected data is stored in DRAM.
15	A/E/E	Submode is Auto Pause, End of Record, of End of File - The CL480 has read the ROM Subheader and the submode is Auto Pause (Trigger), End of Record, or End of File.

SetMute()

Format: SetMute(**muteSetting**)
Priority: High
Category: Set-type

Command ID: 8118

The SetMute() macro command is used to change the audio attenuation settings as shown in Table 12-3. The video output is not affected.

Table 12-3 Audio Attenuation Based on SetMute()

MUTE0	MUTE1	Attenuation on the PCM
0	0	0 dB
1	0	-12 dB
0	1	-∞ (20 ms soft mute)
1	1	-12 dB

The SetMute() argument is defined as follows:

```
short int MuteSetting;    /* Bits [0:1] are the audio mute & attenuation settings  */
```

> *Note: When the host interface is disabled (HOST_ENA=0), host interface pins HD[1] and HD[0] become MUTE1 and MUTE0, respectively.*

SetStreams()

Format: SetStreams(**VideoStreamID, AudioStreamID**)
Priority: Low
Category: Set-type

Command ID: 0213

The SetStreams() macro command is used to specify which of the elementary video and audio streams are demultiplexed, decoded, and output. This command is used for selecting which elementary streams to output for any of the Play-type commands: Play(), Pause(), Freeze(), SingleStep(), Scan(), SlowMotion(), and DisplayStill(). The host processor can also disable output of video and audio by specifying a stream ID which is invalid for the respective stream type.

The SetStreams() arguments are defined as follows:

```
short int VideoStreamID;   /* Stream ID for video stream to decode and output  */
short int AudioStreamID;   /* Stream ID for audio stream to decode and output  */
```

SetVideoFormat()

Format:	SetVideoFormat (**format, HIntplCoeff, Interlaced**)
Priority:	Low
Category:	Set-type
Command ID:	0305

The SetVideoFormat() macro command is used to inform the CL480 of the video format so that picture rate conversion can be performed by the CL480 (if necessary). This command also specifies the coefficient to be used for horizontal interpolation. Vertical interpolation is specified by a location in the Configuration area.

SetVideoFormat() arguments are defined as follows:

```
short int Format;        /* Video Format:                                        */
                         /*      0 = No field repeat, 3 = PAL, 4 = NTSC */
short int HIntplCoeff;   /* Horizontal interpolation coefficient                 */
                         /*     0 = No interpolation                             */
short int Interlaced;    /* Interlaced or non-interlaced output                  */
                         /*      0 = Non-interlaced                             */
```

SetWindow()

Format: SetWindow (**LeftBorder, TopBorder, xOffset, yOffset, width, height**)
Priority: Low
Category: Set-type

Syntax: 0606

The SetWindow() macro command is used to set video window parameters. The **LeftBorder** and **TopBorder** parameters specify the width of the left and top borders, respectively. Setting either of these parameters to zero will cause the video window to be centered in that dimension. The source picture offset parameters, **xOffset** and **yOffset**, specify the pixel location of the source picture which will be displayed in the top-left corner of the video window. The **width** and **height** parameters specify the width and height of the video window. Setting these parameters to zero will cause the width and/or height of the window to be initialized to the values coded in the bitstream. The new window parameters will take effect after the next vertical blanking period.

SetWindow() arguments are defined as follows:

```
short int LeftBorder;     /* left border width in units of 601 pixels */
short int TopBorder;      /* top border width in units of SIF lines */
short int XOffset;        /* x offset into picture in units of 601 pixels  */
short int YOffset;        /* y offset into picture in units of SIF lines */
short int Width;          /* width of active region in units of 601 pixels */
short int Height;         /* height of active region in units of SIF lines */
```

SingleStep()

Format:	SingleStep()
Priority:	Low
Category:	Play-type
Command ID:	000B

Decodes and displays next picture, then pauses. Audio output is disabled.

SlowMotion()

Format:	SlowMotion(**N**)
Priority:	Low
Category:	Play-type
Command ID:	0109

When the SlowMotion() macro command is executed, the CL480 decodes pictures and displays, repeating the display of each picture for **N** frame times. This command applies to video only. No synchronization takes place.

During execution of the SlowMotion() command, the CL480 will accept compressed data until one of its bitstream buffers is full. If the END-P bit has been set by the SetInterruptMask() command, a host interrupt is generated.

When the fullness of the CL480's buffer falls below a threshold and if the RDY-S bit has been set by the SetInteruptMask() command, a host interrupt is generated, and the CL480 will re-enable bitstream input.

The SlowMotion() argument is defined as follows:

```
short int N;            /* factor to reduce speed by is 1/N where N<=8 *
```

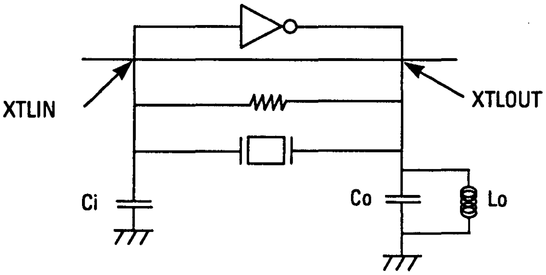
Appendix A CL480 Design Guidelines

This appendix contains design guidelines for the CL480.

Note: Before starting a design layout, please contact C-Cube technical support for the latest microcode and hardware errata information.

A.1 Choosing a Microcontroller and Crystal

In general, use a third-order harmonics frequency crystal. A design example is shown below.



Note: The example shown is just one of many possible examples. For specific details, consult a crystal manufacturer such as one of the companies listed on the following page.

DAISHINKU (AMERICA) CORPORATION
17151 Newhope Street,
Site 210,
Fountain Valley, California USA 92708
Phone 714-557-7833
FAX 714-557-4315

NDK America Inc.
47671 Westinghouse Dr.,
Fremont, CA USA 94539,
Phone 510-623-6500
FAX 510-623-6590

A.2 Initialization and ROM

If a ROM is connected to the CL480, the chip can initialize itself automatically after power-up. If there is no ROM, the host must write the microcode into DRAM, write a code loader into the instruction memory (IMEM) and turn on the CPU run enable. (See Section 11.1.1 for additional information.)

A.3 Pull-up/Pull-down

The signals CFLEVEL, $\overline{\text{DTACK}}$, $\overline{\text{INT}}$ and $\overline{\text{TEST}}$ require a pullup resistor of 1.5K ohms. ROM_ENA should be pulled high if using an external ROM. HOST_ENA should be pulled high unless the host is disabled. All pullups should be pulled to VDD3.

Unused inputs should be pulled either high or low—that is, if the host interface is being used (as in most designs), use a pull-up; if not, use a pull-down.

Special consideration should also to be given when writing DRAM signals as described in Section 5.6.

Customer Feedback

C-Cube Microsystems is always working to improve the quality of our documentation. If you have comments or suggestions about this document, please send us a marked-up copy of the page or pages or send us an e-mail message. We will acknowledge all comments received. Our address is:

Technical Publications Department
C-Cube Microsystems
1778 McCarthy Boulevard
Milpitas, CA 95035

phone: (408) 944-6300
fax: (408) 944-6314

e-mail: techpubs@c-cube.com

North American Representatives

Alabama

M^{2}I
1910 Sparkman Avenue
Huntsville, AL 35816
phone: 205-830-0498
fax: 205-837-7049

Arkansas

TL Marketing
14850 Quorum Dr., Suite 100
Dallas, TX 75240
phone: 214-490-9300
fax: 214-960-6075

California

Bager Electronics
519 Encinitas Blvd.
Encinitas, CA 92024
phone: 619-632-8816
fax: 619-632-8810

Bager Electronics
17220 Newhope Street, Suite 209
Fountain Valley, CA 92708
phone: 714-957-3367
fax: 714-546-2654

Bager Electronics
6324 Variel, Suite 314
Woodland Hills, CA 91367
phone: 818-712-0011
fax: 818-712-0160

Norcomp
8880 Wagon Way
Granite Bay, CA 95746
phone: 916-791-7776
fax: 916-791-2223

Norcomp
1267 Oakmead Parkway
Sunnyvale, CA 94086
phone: 408-733-7707
fax: 408-774-1947

Colorado

Promotech Sales, Inc.
2901 S. Colorado Blvd.
Denver, CO 80222
phone: 303-692-8484
fax: 303-692-8416

Florida

M^{2}I
402 S. North Lake Blvd
Suite 1016
Altamonte Springs, FL 32701
phone: 407-260-6422
fax: 407-260-6460

Georgia

M^{2}I
3000 Northwoods Parkway #110
Norcross, GA 30071
phone: 404-447-6124
fax: 404-447-0422

Hawaii

Bager Electronics
17220 Newhope Street, Suite 209
Fountain Valley, CA 92708
phone: 714-957-3367
fax: 714-546-2654

Illinois

Beta Technology Sales, Inc.
1009 Hawthorn Drive
Itasca, IL 60143
phone: 708-250-9586
fax: 708-250-9592

Iowa

Cahill, Schmitz & Howe
226 Sussex Drive. N.E.
Cedar Rapids, IA 52402
phone: 319-377-8219
fax 319-377-0958

Louisiana

TL Marketing
14850 Quorum Dr., Suite 100
Dallas, TX 75240
phone: 214-490-9300
fax: 214-960-6075

Massachusetts

Advanced Technical Services
348 Park Street, Suite 102
North Reading, MA 01864
phone: 508-664-0888
fax: 508-664-5503

Minnesota

Cahill, Schmitz & Cahill, Inc.
315 N. Pierce Street
St. Paul, MN 55104
phone: 612-646-7217
fax: 612-646-4484

Mississippi

M^{2}I
1910 Sparkman Avenue
Huntsville, AL 35816
phone: 205-830-0498
fax: 205-837-7049

Montana

Promotech Sales, Inc.
2901 S. Colorado Blvd.
Denver, CO 80222
phone: 303-692-8484
fax: 303-692-8416

Nevada

Clark County Only
Bager Electronics
6324 Variel, Suite 314
Woodland Hills, CA 91367
phone: 818-712-0011
fax: 818-712-0160

Other
Norcomp
2140 Professional Drive, #200
Roseville, CA 95661
phone: 916-782-8070
fax: 916-782-8073

New Jersey

Parallax
734 Walt Whitman Road
Melville, NY 11747
phone: 516-351-1000
fax: 516-351-1606

North American Representatives (cont.)

New York

Empire Technical Associates
29 Fennell Street, Suite A
Skaneateles, NY 13152
phone: 315-685-5703
fax: 315-685-5979

Empire Technical Associates
349 W. Commercial Street
Suite 2920
East Rochester, NY 14445
phone: 716-381-8500
fax: 716-381-0911

Parallax
734 Walt Whitman Road
Melville, NY 11747
phone: 516-351-1000
fax: 516-351-1606

North Carolina

M^2I
1200 Trinity Road
Raleigh, NC 27607
phone: 919-851-0010
fax: 919-851-6620

North Dakota

Cahill, Schmitz & Cahill, Inc.
315 N. Pierce Street
St. Paul, MN 55104
phone: 612-646-7217
fax: 612-646-4484

Oklahoma

TL Marketing
14850 Quorum Dr., Suite 100
Dallas, TX 75240
phone: 214-490-9300
fax: 214-960-6075

South Carolina

M^2I
1200 Trinity Road
Raleigh, NC 27607
phone: 919-851-0010
fax: 919-851-6620

South Dakota

Cahill, Schmitz & Cahill, Inc.
315 N. Pierce Street
St. Paul, MN 55104
phone: 612-646-7217
fax: 612-646-4484

Tennessee

M^2I
3000 Northwoods Parkway #110
Norcross, GA 30071
phone: 404-447-6124
fax: 404-447-0422

Texas

TL Marketing
14850 Quorum Dr., Suite 100
Dallas, TX 75240
phone: 214-490-9300
fax: 214-960-6075

TL Marketing
8100 Shoal Creek, Suite 250
Austin, TX 78758
phone: 512-371-7272
fax: 512-371-0727

TL Marketing
14343 Tory Chase Blvd. Suite I
Houston, TX 77014
phone: 713-587-8100
fax: 713-580-7517

Wisconsin

Western
Cahill, Schmitz & Cahill, Inc.
315 N. Pierce Street
St. Paul, MN 55104
phone: 612-646-7217
fax: 612-646-4484

Eastern
Beta Technology Sales, Inc.
9401 W. Beloit Road, Suite 409
Milwaukee, WI 53227
phone: 414-543-6609
fax: 414-543-9288

Canada

Electrosource
6875 Royal Oak
Burnaby, BC
Canada V5J 4J3
phone: 604-435-2533
fax: 604-435-2538

Electrosource
230 Galaxy Blvd.
Rexdale, ONT
Canada M9W 5R8
phone: 416-675-4490
fax: 416-675-6871

Electrosource
300 March Road, Suite 203
Kanata, ONT
Canada
K2K 2E2
phone: 613-592-3214
fax: 613-592-4256

Electrosource
6600 Trans Canada Highway,
Suite 420
Pointe Claire, Quebec
H9R 4S2
phone: 514-630-7486
fax: 514-630-7421

Others (Not Listed)

Contact nearest office of
C-Cube Microsystems

International Representatives and Distributors

France
NEWTEK (Rep/Dist.)
8, rue de l'Esterel
SILIC 583
94663 Rungis Cedex
phone: (33) 1-46.87.22.00
fax: (33) 1-46.87.80.49

United Kingdom
Kudos Thame Ltd. (Rep/Dist.)
55 Suttons Park, London Rd.
Reading, BERKS RG6 1AZ
phone: (44) 734-351010
fax: (44) 734-351030

Germany
Metronik GmbH (Rep/Dist.)
Leonhardsweg 2
8025 Unterhaching
phone: (49) 89-61108-0
fax: (49) 89-6116858

Italy
Newtek Italia
Via G da Procida 10
20149 Milano
phone: (39) 02-33-10-53-08
fax: (39) 02-33-10-36-94

Scandinavia
Magnus Granfelt (Rep.)
MEMEC Scandinavia AB
Kvarnholmsvagen 52
131 31 Nacka
phone: (46) 8-6434190
fax: (46) 8-6431195

Australia
ZATEK Components Pty Ltd.
(Rep/Dist.)
Suite 8, 1059 Victoria Rd.
West Ryde 2114
Sydney
phone: (61) 2-874-0122
fax: (61) 2-874-6171

Hong Kong
MEMEC Asia Pacific (Rep/Dist.)
Unit No 2308-2319, Tower 1
Metroplaza
Hing Fong Road
Kwai Fong, N.T.,
phone: (852) 410-2780
fax: (852) 418-1600

Japan
Kubota C-Cube Inc.
Fuso Building, 7F, 2-12-8
Shin-Yokohama
Kohoku-Ku
Yokohama, Kanagawa 222
phone: (81) 45-474-7571
fax: (81) 45-474-7570

Korea
MEMEC Asia Pacific (Rep/Dist.)
4th Floor, Jae Woong Bldg
176-11 Nonhyun-Dong
Kangnam-ku, Seoul
phone: (82) 2-518-8181
fax: (82) 2-518-9419

Singapore
Serial System Marketing
(Rep/Dist.)
11 Jalan Mesin
Standard Industrial Bldg, #06-00
Singapore 1336
phone: (65) 280-0200
fax: (65) 286-6723

Republic of China
MEMEC Asia Pacific(Rep)
14F-1, 171 Section 58
Min Sheng East Road
Hai Hwa Building
Taipei, Taiwan
phone: (886) 2 760-2028
fax: (886) 2 765-1488

ALLY, Inc. (Dist.)
7F, 18, Alley 1 Lane 768, Sec. 4
Pa Teh Rd.,
Taipei, Taiwan
phone: (886) 2 788-6270
fax: (886) 2 786-3550

C-Cube Microsystems Sales Offices

Home Office
C-Cube Microsystems
1778 McCarthy Boulevard
Milpitas, CA 95035
phone: 408-944-6300
fax: 408-944-6314

Eastern Area Office
C-Cube Microsystems
One Kendall Square, Suite 220
Cambridge, MA 02139
phone: 617-621-7180
fax: 617-621-7179

Southwestern Area Office
C-Cube Microsystems
453 Bristol Avenue
Cardiff, CA 92007
phone: 619-632-0864
fax: 619-632-0864

European Office
C-Cube Microsystems
44 Dartford Road
Sevenoaks, Kent
UK TN 133 TQ
phone: (44) 732 743 256
fax: (44) 732 450 151

Japan Office
Kubota C-Cube Inc
Fuso Building 7F
2-12-8 Shin-Yokohama
Kohoku-ku
Yokohama, Kanagawa 222
phone: 81-45-474-7571
fax: 81-45-474-7570

C-Cube
Microsystems
1778 McCarthy Blvd.
Milpitas, CA 95035
Tel: (408) 944-6300
Fax: (408) 944-6314