1 Introduction

1.1 Overview

The CL480 is an MPEG-1 audio/video decoder chip. Designed for consumer electronics products and multimedia PCs, the CL480 reduces system cost by requiring only four Mbits of DRAM, to provide CD-ROM decoding and 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 suitable for video CD players (microcode version CL480VCD) because it provides glueless interfaces to the CD-DSP, audio DAC, and NTSC/PAL encoder. It is also suitable for multimedia PC applications (microcode version CL480PC) because it fully supports the OM-1 (Open MPEG) standard.

The CL480 processes the following data types:

- CD-ROM data
- CD-DA data
- MPEG-1 system streams in real time, containing:
  - Constrained-parameters MPEG-1 video bitstreams (typically SIF-resolution)
  - MPEG-1 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-1 audio and video decoding, video timing generation, and CD-ROM decoding functions
- Fully supports VideoCD 2.0 and 1.1, KaraokeCD 1.0, OM-1 and CD-I Green Book[1] standards
- 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 I or II audio streams
  - Resolutions include 352 x 240 at 30 Hz, 352 x 288 at 25 Hz, and 384 x 240 at 24 Hz
  - Audio sample rates include 32, 44.1, and 48 kHz
- Decodes and displays still images with coded resolutions up to 704 x 576 while decoding MPEG audio streams
- Requires only 4 Mbits of 80-ns DRAM, even for PAL systems
- Provides 13 Kbytes of free DRAM space for playback control sectors and video overlays, even while displaying 704 x 576 high-resolution stills with only 4 Mbits of DRAM
- Accepts compressed data from a 4-pin CD interface or from an 8-bit host interface
- Accepts a constant input data rate of up to 2.8 Mbits per second
- Accepts burst input data rates of up to 20 Mbits per second to the host interface and 6.6 Mbits per second to the CD interface
- Provides glueless interfaces to the 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 the display rate (typically 50 or 60 Hz) is independent of the coded frame rate (typically 24, 25 or 29.97 Hz)

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.

- Performs horizontal interpolation of decoded video using a seven-tap filter
- Provides RGB or YCbCr video output using on-chip color space conversion
- Supports $\overline{\text{HSYNC}}$ and $\overline{\text{VSYNC}}$ signals as inputs or outputs
- Supports VCK input (typically 27 MHz) or generates VCK output (13.5 MHz) by dividing GCK (40.5 MHz) by three
- Optionally generates NTSC and PAL composite synchronization

1.2.2 Antialiased Video Overlays

- Can be either SIF or CCIR 601 resolution
- Can be smoothly faded with decompressed MPEG video

1.2.3 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 to reduce power consumption when decoding MPEG audio only or when in CD-DA pass-through mode without video streams
- Packaged in a 128-pin small-outline PQFP (18mm x 18mm body)

1.2.4 Powerful, Easy-to-Use Microcode

- Performs audio/video synchronization without host intervention
- Provides high-quality error concealment for bitstream errors
- Performs error correction of CD-ROM data at up to 2.8 Mbits per second when not decoding MPEG
- Can initialize itself from the ROM connected to the DRAM interface
- Provides high-level macro commands, allowing the host to monitor and control the inputting, decoding, and outputting 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
  - DisplayBorderColor - Sets active display region to border color
  - Pause - Freezes displaying and decoding process
  - Freeze - Freezes displaying but continues decoding
  - DumpData - Performs error correction on CD-ROM data
  - SetStreams - Selects which streams to decode
  - DisplayDigest - Displays digest feature
  - DisplayGraphics - Displays graphics images
  - SetVideoFormat - Changes output video format
  - InquireBufferEmptiness - Measures data in bitstream buffer
  - Abort - Sets the processing state to IDLE
  - Replay - Restarts the previous Play command
  - Reset - Initializes the CL480 and its microcode
- Provides 15 interrupts, giving the host feedback on bitstream transitioning, displaying, 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.

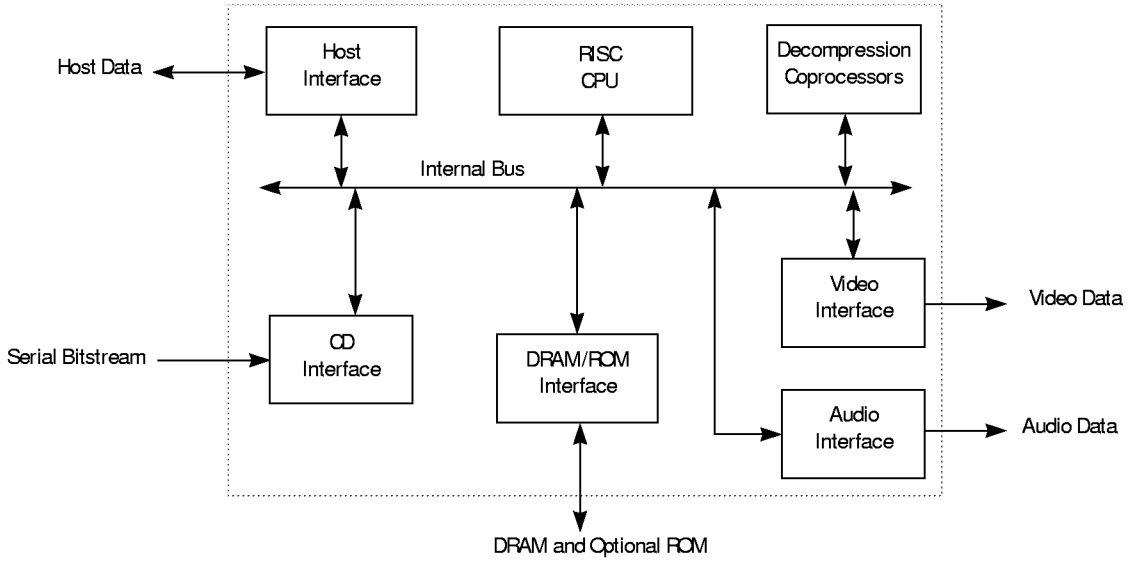


Figure 1-1 Block Diagram of the CL480

Each of the CL480's 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. This host interface is based on an eight-bit bus tailored for low-cost applications; it minimizes pins while allowing the host to access DRAM/ROM, the code FIFO (CFIFO), 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 all 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 256K x 16 DRAM or four 256K x 4 DRAMs. It can also be connected directly to an eight-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 64 Kbytes. The CL480 supports up to two Mbytes of ROM for uses such as logo bitstreams and graphics.

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 these modes:

- 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).

The CL480 provides a video overlay that can be used to display text and graphics on top of decompressed MPEG video. This overlay can be antialiased to reduce flicker and jagged edges.

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 Typical Applications

The CL480 is designed for low-cost applications that include consumer electronics products (CL480VCD) and multimedia PC applications (CL480PC).

1.4.1 Consumer Electronics Applications (CL480VCD)

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 four-pin CD interface. This CD data stream could contain an MPEG-1 system stream, CD-DA data, or CD-ROM data.

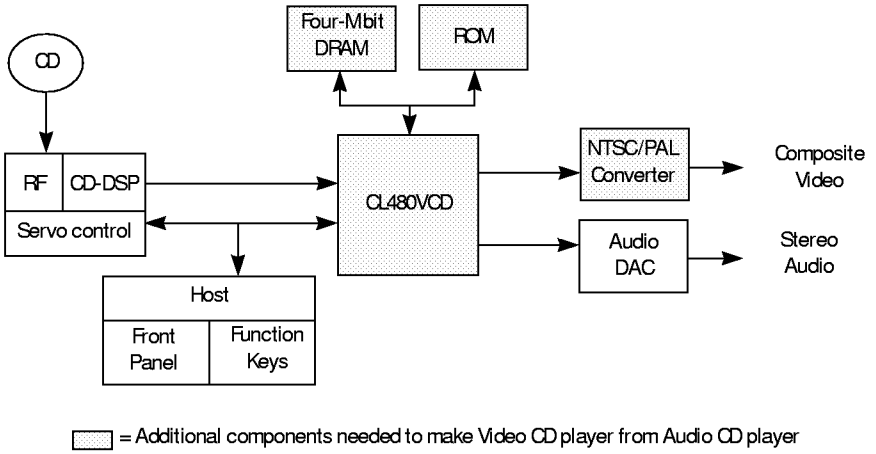


Figure 1-2 CL480 in Consumer Electronics Configuration (CL480VCD)

1.4.2 Multimedia PC Application (CL480PC)

In multimedia PCs, the CL480 typically receives an MPEG-1 system stream from its host interface. A typical configuration is shown in Figure 1-3.

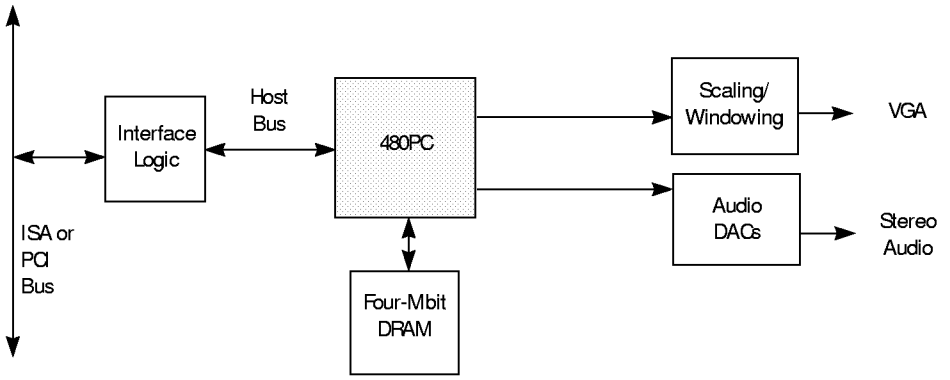


Figure 1-3 CL480 in Multimedia PC Configuration (CL480PC)

The host controls the CL480 by sending high-level commands to a command FIFO in the CL480's DRAM. It then writes system, video, or audio frames to the CFIFO. After being given the Play command, the CL480 can decode MPEG audio and video without host intervention.

Note: The CL480PC has the advantage of not requiring a ROM for the CL480 microcode.

1.4.3 Portable Applications

Future versions of the CL480 will apply additional power management techniques such as ultra-low power "sleep" modes. These techniques, combined with the CL480's small footprint and thin packaging, will enable a wide variety of mobile applications such as Notebook Multimedia PCs and portable VideoCD players.

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 variable bit-rate coding or a higher 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 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 Sections 2.2 and 2.3 for more information about video compression.)

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

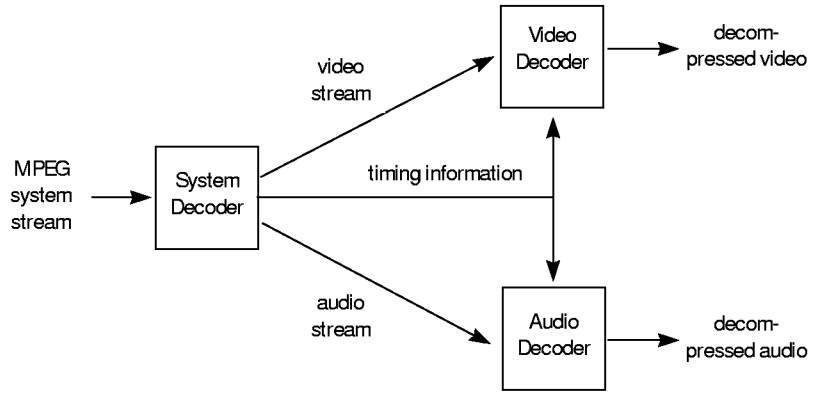


Figure 2-1 General MPEG Decoding System

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.

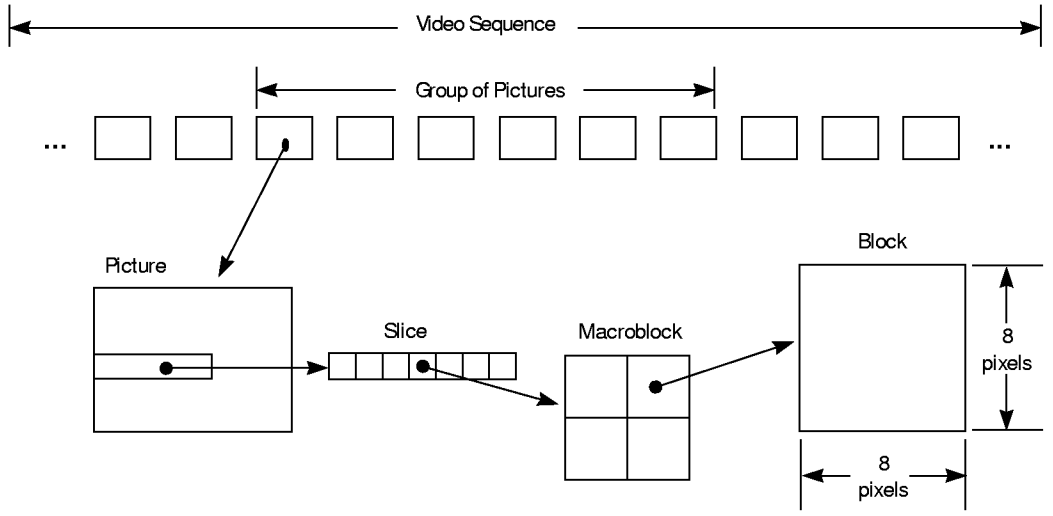


Figure 2-2 MPEG Data Hierarchy

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 (GOP)

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.)

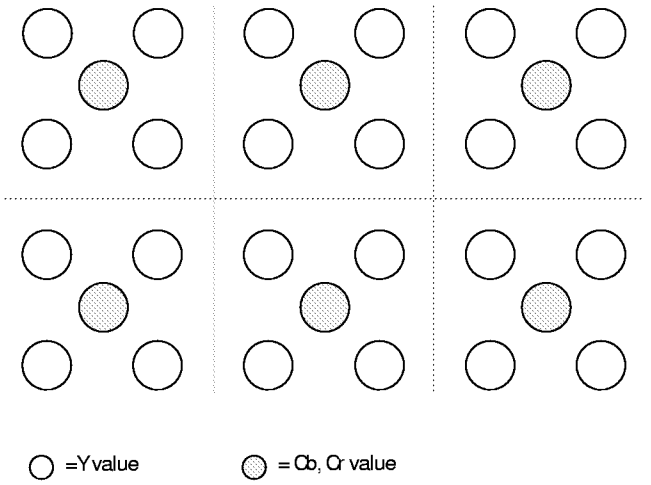


Figure 2-3 Location of Luminance and Chrominance Values

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.

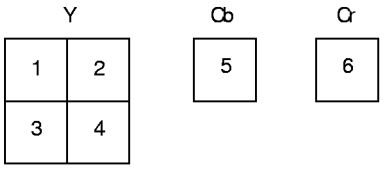


Figure 2-4 Macroblock Composition

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.

2.1.4 Audio Stream Data Hierarchy

The MPEG standard defines a hierarchy of data structures that accept, decode and produce digital audio output. The MPEG audio stream, like the MPEG video stream, consists of a series of packets. Each audio packet contains an audio packet header and one or more audio frames as shown in Figure 2-5.

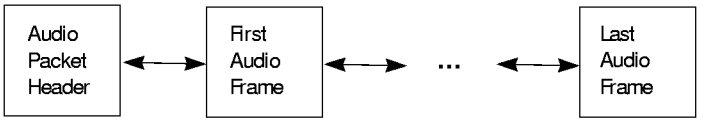


Figure 2-5 Audio Stream Structure

Each audio packet header contains the following information:

- Packet start code - Identifies the packet as being an audio packet
- Packet length - Indicates the number of bytes in the audio packet.

An audio frame contains the following information:

- Audio frame header - Contains synchronization, ID, bit rate, and sampling frequency information
- Error-checking code - Contains error-checking information
- Audio data - Contains information used t o reconstruct the sampled audio data.
- Ancillary data - Contains user-defined data.

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) 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 2-6.

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.

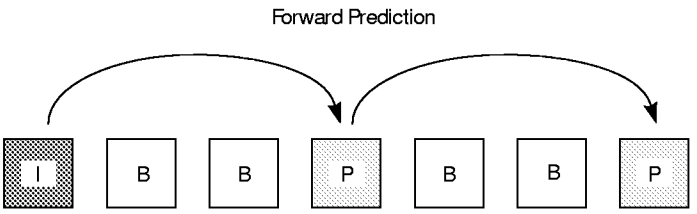


Figure 2-6 Forward Prediction

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 2-7. 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.

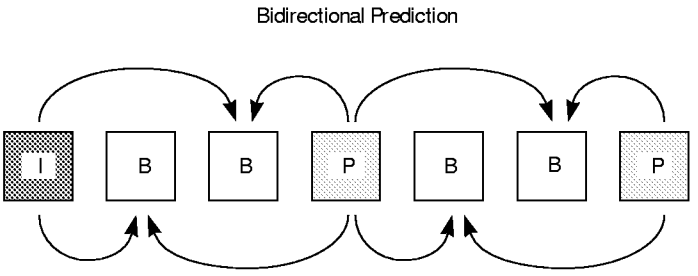


Figure 2-7 Bidirectional Prediction

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 2-8 in the order in which they are displayed.

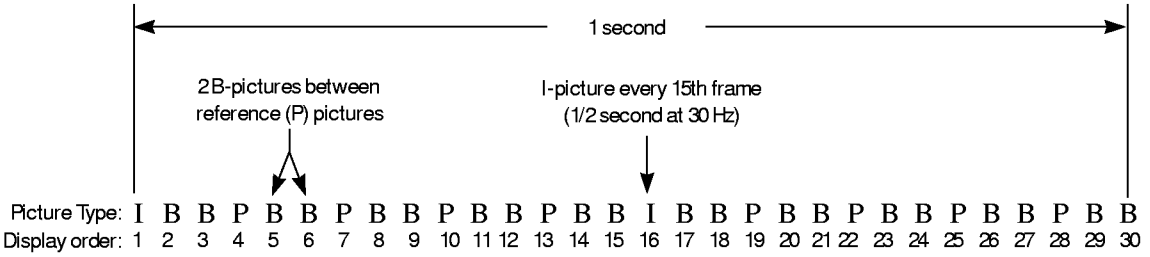


Figure 2-8 Typical Display Order of Picture Types

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 2-9 demonstrates this ordering for the first section of the example shown above.

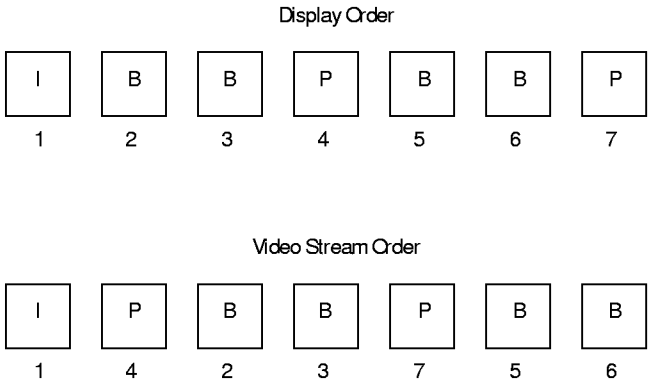


Figure 2-9 Video Stream versus Display Ordering

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 com-

pensation—it is coded in the same way as a macroblock in an I-picture using transform coding techniques (see Section 2.3).

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 2-10). 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.

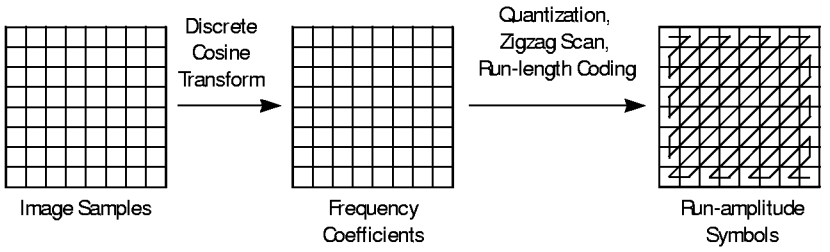


Figure 2-10 Transform Coding Operations

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 timestamp (PTS)*.

The MPEG-specified "system clock" runs at 90 kHz. System clock reference and presentation timestamp 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 2-11. During decoding, these values are used to update the system clock counter in the CL480.

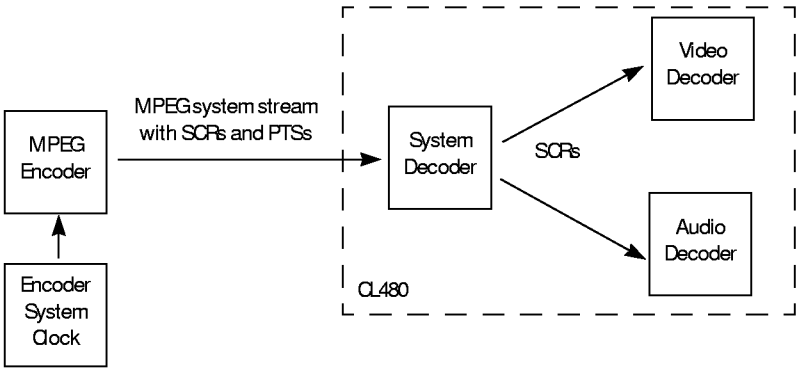


Figure 2-11 SCR Flow in MPEG System

2.4.2 Presentation Timestamps

Presentation timestamps 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.

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, 8 and 9.

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

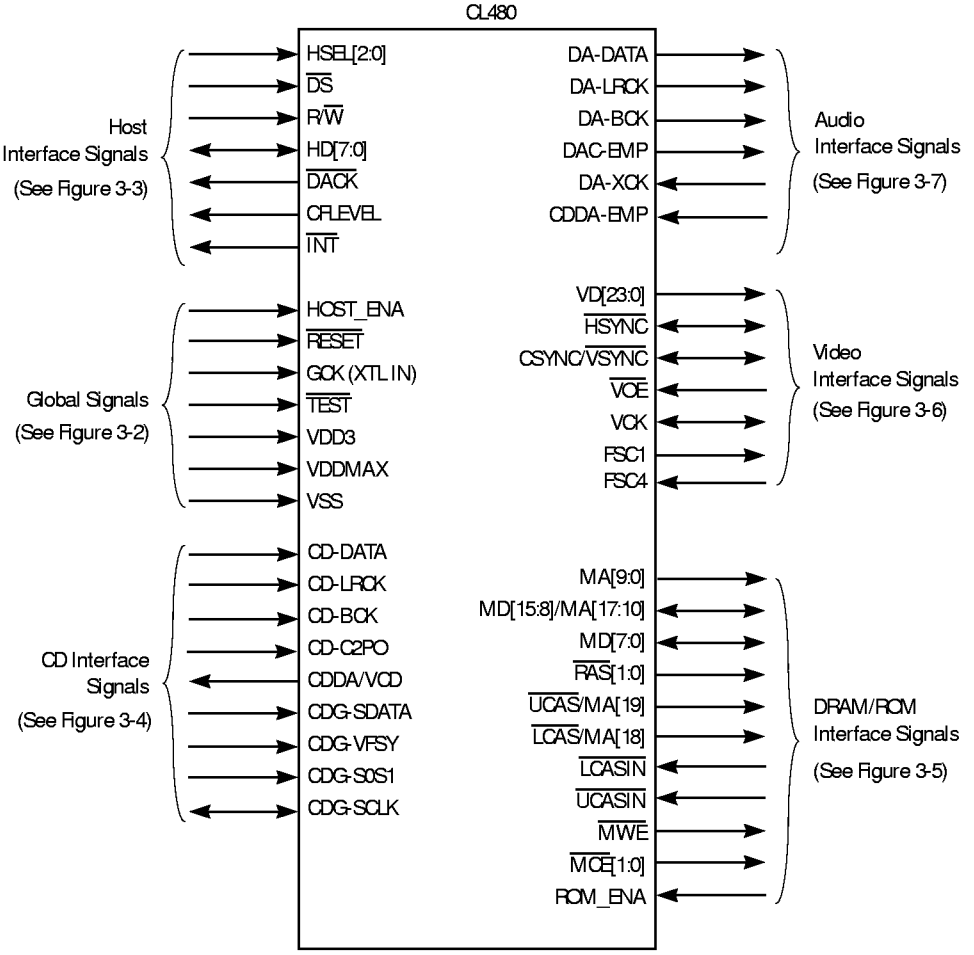


Figure 3-1 Bus Connection Diagram

3.1 Global Interface Signals

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

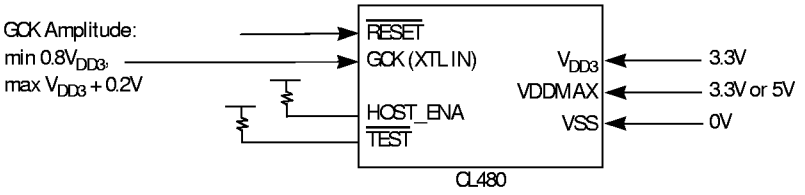


Figure 3-2 Global Interface Signals

$\overline{\text{RESET}}$ — 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.

GCK (XTL IN) — Global Clock **Input**

The CL480 uses GCK (XTL IN and XTL OUT) to clock the internal processor. This oscillator is typically 40 MHz or 40.5 MHz. If a clock driver (i.e., oscillator) is used for GCK, it should be connected to pin 19, XTL IN, with a minimum amplitude of $0.8V_{DD3}$ and a maximum amplitude of V_{DD3} + 0.2V.

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

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

VDD3 — Power (supply voltage) **Input**

This pin supplies 2.7 to 3.6 volts of internal power and output high voltage.

VSS — Power (ground) **Input**

Ground.

VDDMAX — Power (maximum) **Input**

This pin is the maximum input voltage connected to any CL480 pin. The CL480 operates at 3.3 volts, but can accept 5-volt inputs if VDDMAX is set to 5 volts. The voltage of VDDMAX should be greater than or equal to the voltage driven on any input, I/O, or open-drain output.

3.2 Host Interface Signals

The host interface signals 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.

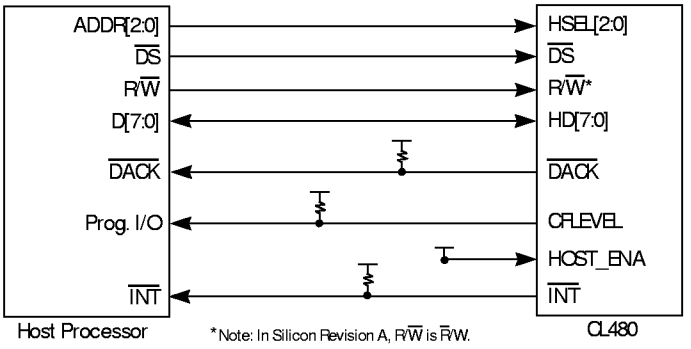


Figure 3-3 Host Interface Signals

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.

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

The host processor asserts $\overline{DS}$ 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 eight-bit bidirectional host data bus. The host processor uses HD[7:0] to write data to the CL480's code FIFO (CFIFO), 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/$\overline{W}$ ($\overline{R}$/W in Silicon Revision A)[1].

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

The host processor drives R/$\overline{W}$ LOW to initiate a write operation, and drives R/$\overline{W}$ HIGH to initiate a CL480 read operation to the host data bus. (R/$\overline{W}$ polarity is reversed in Silicon Revision A.)

1. Silicon Revision A refers to the silicon version previously denoted as ES1.

$\overline{\text{DACK}}$ — Host Data Acknowledge Open-Drain Output

The CL480 asserts $\overline{\text{DACK}}$ (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 $\overline{\text{DACK}}$ deasserted (HIGH) until the requested data is ready. When the CL480 responds to a write request, it asserts $\overline{\text{DACK}}$ when it has received and latched the write data.

DACK 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 — CFIFO Level Status Output

When CFLEVEL is zero, the CL480 CFIFO has room for at least 44 bytes of compressed data. CFLEVEL 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.

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

The CL480 asserts $\overline{\text{INT}}$ to request an interrupt from the host processor. Interrupt events are determined from the INT_STATUS word in the DRAM Status Area (see Table 15-1). $\overline{\text{INT}}$ is an open-drain signal. A 1.5K pull-up should be added on this pin.

HOST_ENA—Host Enable Input

Enables or disables the host interface. For normal operations, HOST_ENA must be held HIGH (asserted).

3.3 CD Interface Signals

The CL480's CD interface is dedicated to receiving the serial-bit output of a CD DSP. A four-wire serial bus connects the CD DSP directly with the CL480 as Figure 3-4 shows. A four-wire serial bus also connects the CD+G signals.

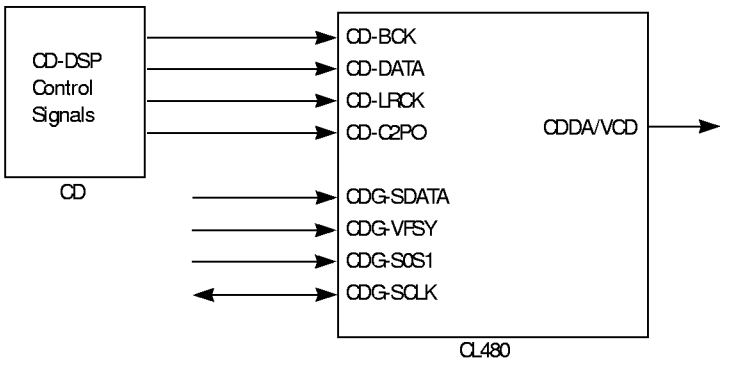


Figure 3-4 CD-Decoder Interface Signals

CD-BCK — CD Bit Clock **Input**

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

CD-DATA — CD Data **Input**

CD-DATA receives the serial data input from CD-DSP.

CD-LRCK — CD 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-C2PO — CD-ROM Data **Input**

CD-C2PO signals a corrupted byte (error byte flag MSB or LSB first). It is used when receiving CD-ROM data, but is ignored in CDDA pass-through mode. This signal is HIGH when an error occurs.

CDDA/VCD— CDDA/V-CD Enable **Output**

CDDA/VCD outputs whether incoming data is CDDA versus V-CD: CDDA = 1, V-CD = 0.

Note: Pin assignments have been made to allow future versions of the CL480 to support the subcode interface to the CD-DSP, such as the CD-G functions described next.

CDG-SDATA — CD+G (Subcode) Data **Input**
Indicates serial subcode data input.

CDG-VFSY — CD+G (Subcode) Frame Sync **Input**
Indicates frame-start or composite synchronization input.

CDG-S0S1 — CD+G (Subcode) Block Sync **Input**
Indicates block-start synchronization input.

CDG-SCLK — CD+G (Subcode) Clock **Input/Output**
Indicates subcode data clock input or output.

3.4 DRAM Interface Signals

Figure 3-5 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.

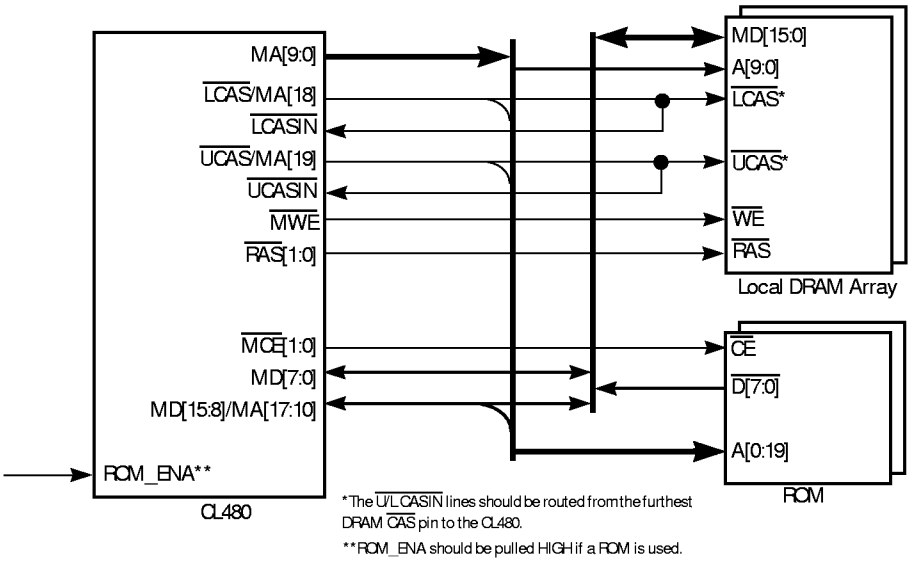


Figure 3-5 DRAM/ROM Interface Signals

MA[9:0] — Memory Address Bus **Outputs**

The CL480 multiplexes the row and column addresses on the 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 determined by the state of $\overline{\text{MWE}}$.

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

$\overline{\text{RAS}}$[1:0] — Row Address Strobe **Outputs**

The CL480 asserts these signals to latch the row address into the DRAM array. $\overline{\text{RAS1}}$ (active LOW) latches the row address for bank 1, and $\overline{\text{RAS0}}$ (active 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].

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 **Input**

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

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 **Outputs**

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**

ROM_ENA informs the CL480 that a boot ROM is attached. If this signal is active when the CL480 receives a $\overline{\text{RESET}}$ signal, the first 2048 bytes of ROM (ROM bank 0) are loaded into the CL480's processor instruction memory. After the instruction memory is loaded, instruction execution begins at address zero.

3.5 Video Interface Signals

The CL480's video interface outputs pixel data to the video display subsystem in RGB or YCbCr format. Figure 3-6 shows two possible configurations of the signals in the CL480's video interface: $\overline{\text{VSYNC}}$ (or CSYNC)/$\overline{\text{HSYNC}}$/VCK as outputs at 13.5 MHz, and $\overline{\text{VSYNC}}$/$\overline{\text{HSYNC}}$/VCK as inputs at 27 MHz.

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.

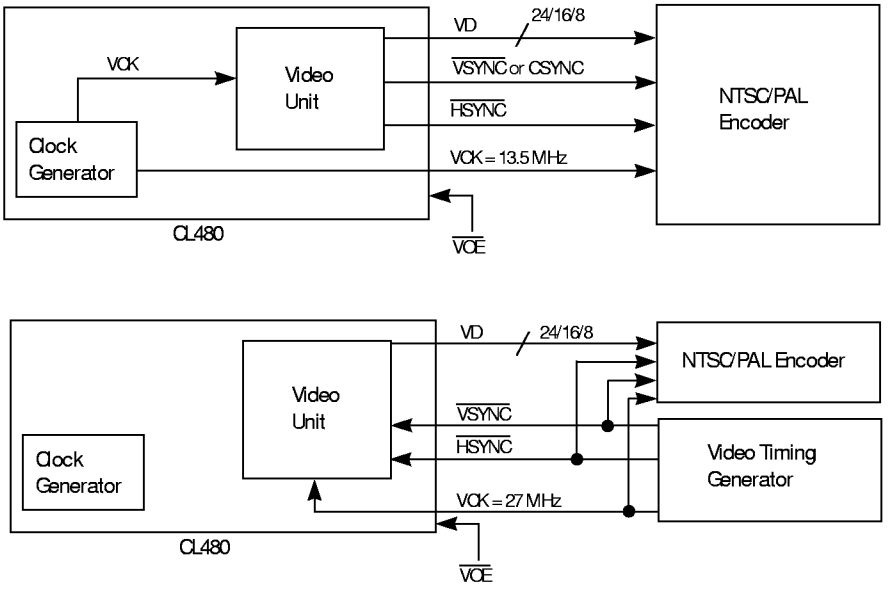


Figure 3-6 Video Interface Signals

VD[23:0] — Video Data Bus **Outputs**

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

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

The CL480 begins outputting pixel data for a new horizontal line after the falling (active) edge of $\overline{\text{HSYNC}}$. When used as an input, $\overline{\text{HSYNC}}$ must be synchronous to VCK as Chapter 7 shows.

$\overline{\text{VSYNC}}$ — Vertical Synchronization **Bidirectional**
(or CSYNC) — Composite Synchronization **Output**

The CL480 begins outputting the top border of a new field on the first $\overline{\text{HSYNC}}$ after the falling edge of $\overline{\text{VSYNC}}$. $\overline{\text{VSYNC}}$ can be asynchronous with respect to VCK. Alternately, CSYNC can be chosen on the same pin but can only be an output, never an input. See Chapter 7 for more information on the relationship of $\overline{\text{VSYNC}}$ (or CSYNC) 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 Silicon Revision A.

VCK — Video Clock **Bidirectional**

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

FSC1— Color Subcarrier Signal **Output**
FSC4— Color Subcarrier Signal **Input**

These pins can optionally be used to generate the color subcarrier for NTSC and PAL. Typical inputs on FSC4 (pin 126) are 14.318 MHz for NTSC and 17.73 MHz for PAL. The source clock divided by four is output on FSC1 (pin 124).

3.6 Audio Interface Signals

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 two external sources, DA-XCK and CDDA-EMP, as input and then outputs DAC-EMP. Figure 3-6 shows the CL480's audio interface signals.

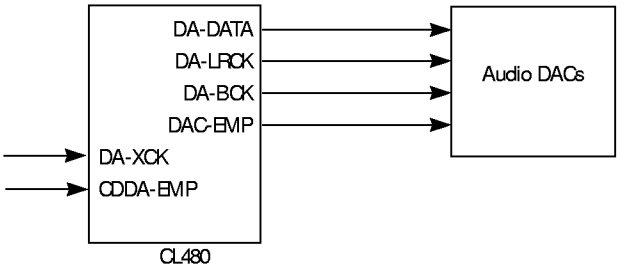


Figure 3-7 Audio Interface Signals

DA-DATA — Audio Data Bus **Output**

DA-DATA outputs bit serial audio samples relative to the DA-BCK clock.

DA-LRCK — Audio Left-Right Clock **Output**

DA-LRCK identifies the channel for each audio sample.

DA-BCK — Audio 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 — Audio Output Emphasis Flag **Output**

DAC-EMP is used to control the de-emphasis circuitry of the audio output DACs. When in CD-DA mode, this pin follows the state of the CDDA-EMP pin; when in VCD mode, this pin follows the state of the least-significant bit of the emphasis field of the MPEG-1 audio header.

DA-XCK — Audio External 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 — Audio Input Emphasis Flag **Input**

In CD-DA mode, the input is directly routed to the DAC-EMP output pin. It is ignored when receiving MPEG-1 bitstreams.

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

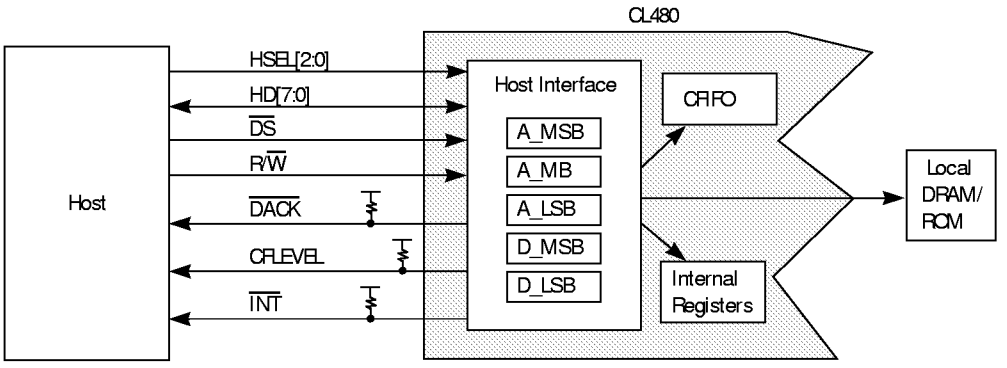


Figure 4-1 CL480 Host Interface

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

The host processor accesses CL480 resources by writing to address registers, and reading or writing to data registers. It uses a combination of five host interface registers:

- 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 code FIFO (CFIFO), local DRAM/ROM 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, CFIFO, and GBUS

A_MSB[7:6]	Data From/To
00_2	CFIFO
01_2	DRAM/ROM (No autoincrement)
10_2	GBUS
11_2	DRAM/ROM (Autoincrement)

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

- Write to CFIFO
- Read/write to GBUS (internal registers)
- Read/write to DRAM and read from ROM (external memory)
- Read/write to DRAM and read from ROM (external memory) with auto-increment addressing

4.2.2 Host Interface Data Registers

The two 8-bit host 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 CFIFO, DRAM and internal registers, only the address bytes that differ from earlier transaction(s) need to be rewritten 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_2	A_LSB	Address, byte 0 (LSB)
010_2	A_MB	Address, byte 1
011_2	A_MSB	Address, byte 2 (MSB)
100_2	D_LSB	Data, byte 0 (LSB)
101_2	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

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. This 21-bit memory address is a 16-bit word address, not a byte address.

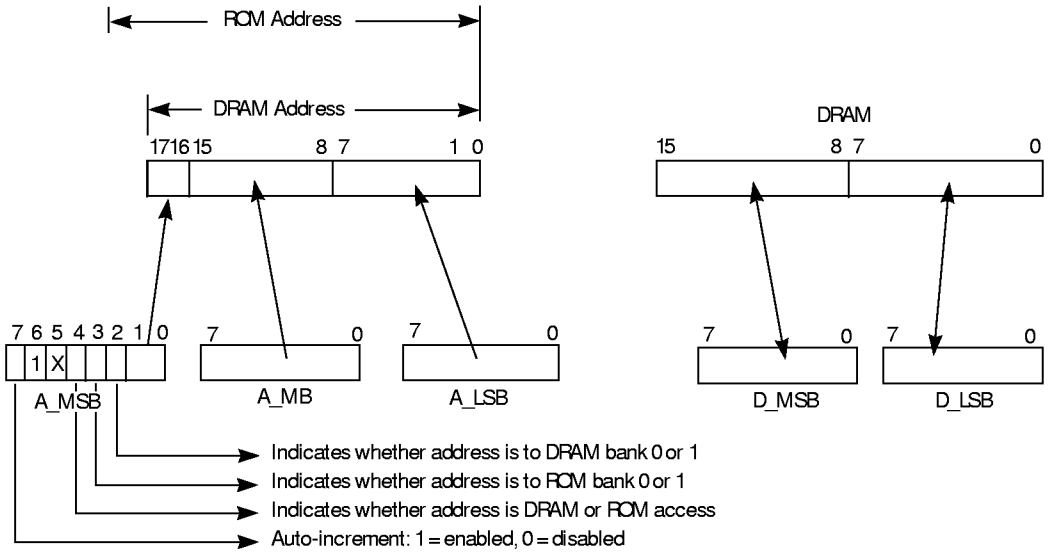


Figure 4-2 DRAM/ROM Address Formed by Host Address Registers

Bit 4 of A_MSB determines whether DRAM (set to 0) versus ROM (set to 1) is selected. Bit 3 of A_MSB selects the ROM banks, while bit 2 of A_MSB selects the DRAM banks as shown in Table 4-3.

Table 4-3 Memory Address Map for A_MSB

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

For DRAM accesses, bit 7 of the 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*.

4.2.5

CFIFO Access

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.

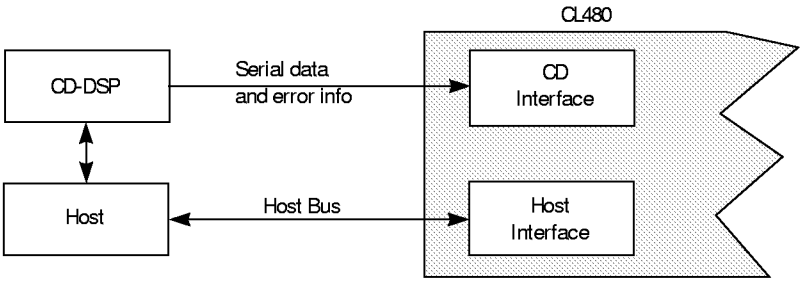


Figure 4-3 Sending Coded Data on CD Interface

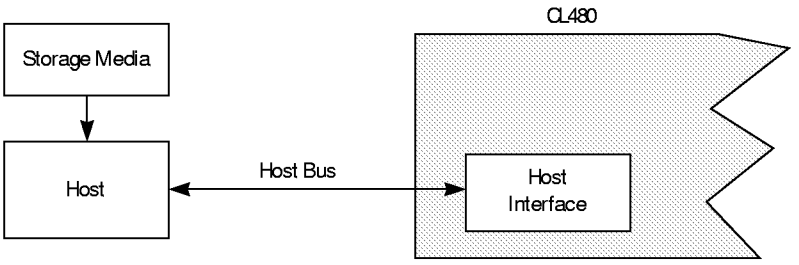


Figure 4-4 Sending Coded Data on Host Interface

In CFIFO 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 CFIFO.

MPEG bitstreams are input to the CL480 with the first bit of the stream placed in the MSb of the D_LSB register, and the eighth bit of the stream placed in the LSb of that same register. The ninth bit of the stream is placed in the MSb of the D_MSB register, and the sixteenth bit of the stream is placed in the LSb of that same register. This ordering begins again with the 17th bit of the stream.

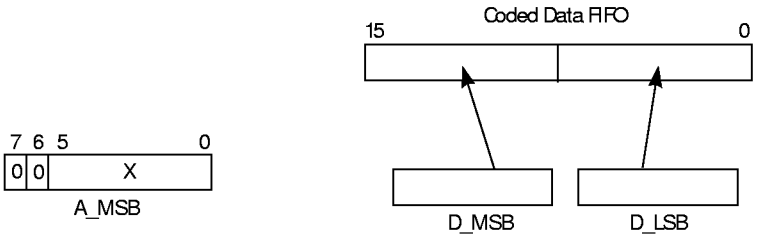


Figure 4-5 CFIFO Register Address Formed by A_MSB

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 CFIFO holds 32 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.

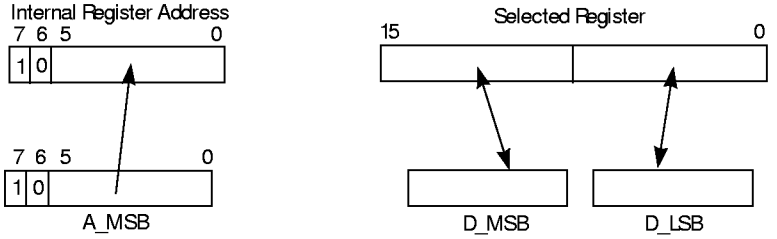


Figure 4-6 Internal Register Address Formed by A_MSB

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

Table 4-4 Internal Registers Decoded by A_MSB

A_MSB[5:0]	Register Name	R/W	Description
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 interrupt register

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 9.2.2 for the detailed timing parameters for this operation.) The circled numbers in the figure refer to the steps below it.

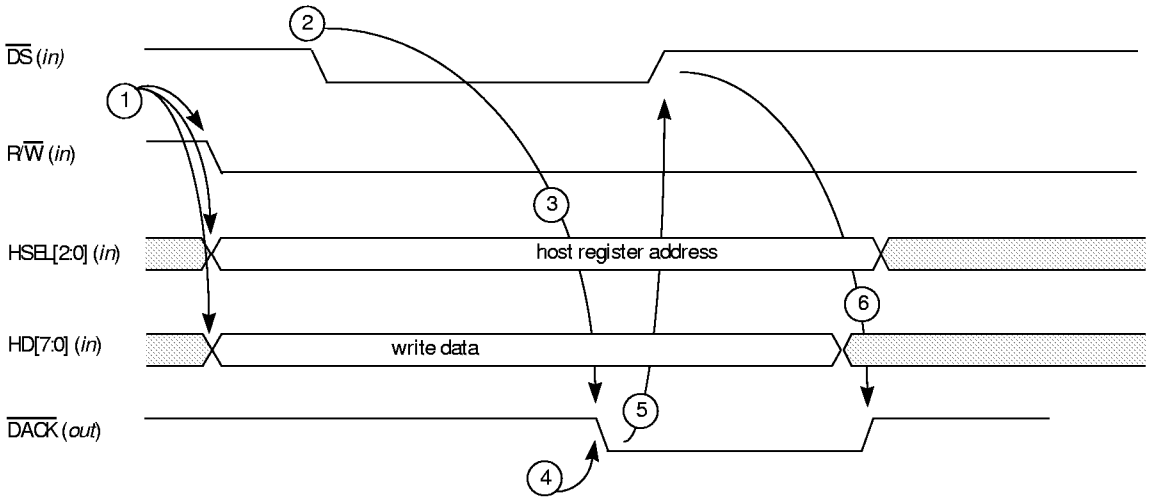


Figure 4-7 Host Interface Local Register, Write Operation

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 9.2.2 for detailed timing parameters for this operation.) The circled numbers in the figure refer to the steps below it.

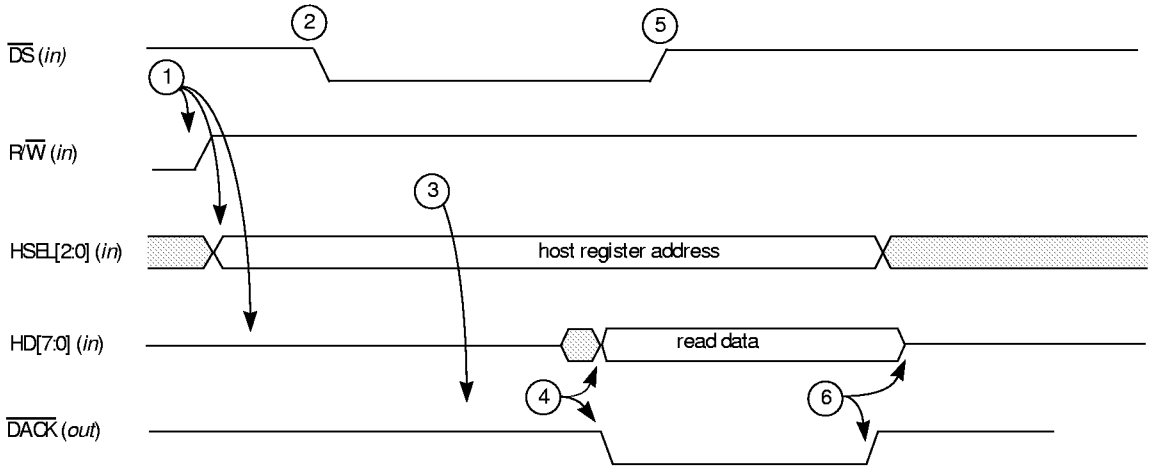


Figure 4-8 Host Interface Local Register, Read Operation

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 HD[7:0].

5 DRAM/ROM Interface

This chapter describes the local DRAM/ROM interface bus. It details all of the signals necessary to connect the CL480 to a DRAM array and optional ROM.

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 approximately 2.5 Mbytes per second. The CL480 can be booted directly from ROM or the host can load microcode into DRAM.

Figure 5-1 illustrates the DRAM interface bus and how it is used.

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 (512K bytes) of DRAM are needed to read, decode, and output Video CD, MPEG-1 system and*

CD-DA bitstreams. The CL480 can address a maximum of eight Mbits or one Mbyte of DRAM. Any unused memory in the first four Mbits of memory and any additional memory provided above the four-Mbit minimum is available to the user by the host processor for on-screen graphics display data, Dumpdata() buffers, or any other use.

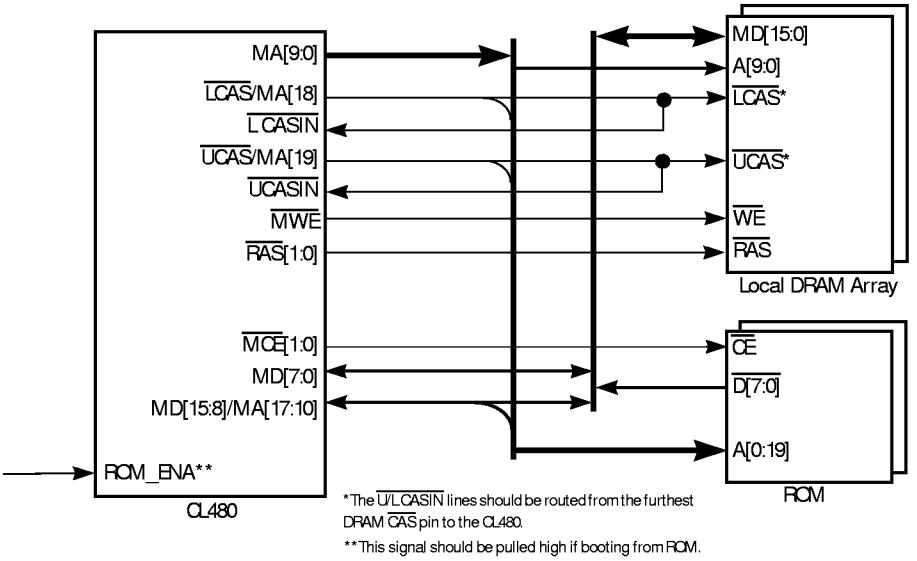


Figure 5-1 DRAM Interface

The DRAM array is organized as either one 256K x 16 DRAM or four 256K x 4 DRAMs.

The DRAM array is used to store the following:

- 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

For a further description of actual DRAM addresses and their access, see Section 11.2.1 and Chapters 12 and 15.

5.1.2 ROM: Amount and Organization

The CL480 can support up to one Mbyte of ROM: specifically, two ROMs, each which can be up to 512K x 8 bits.

The required ROM size depends on the customer and what needs to be stored in ROM, for example, 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 3 to 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 12.3.*

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:

- 256K x 16 DRAMs: 10-bit row address, 8-bit column address
- 256K x 16 DRAMs: 9-bit row address, 9-bit column address

Table 5-1 shows the address mapping for these two modes. The 9/9 DRAMs use 20-30 percent 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
256K x 16, 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
256K x 16, 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 256K x 16 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 $\overline{RAS}$[0] or $\overline{RAS}$[1] signals, respectively. Host address signals MA[18:1] map directly into the multiplexed row and column DRAM addresses. The host bus interface signal R/$\overline{W}$ maps directly to the DRAM interface $\overline{MWE}$ signal.

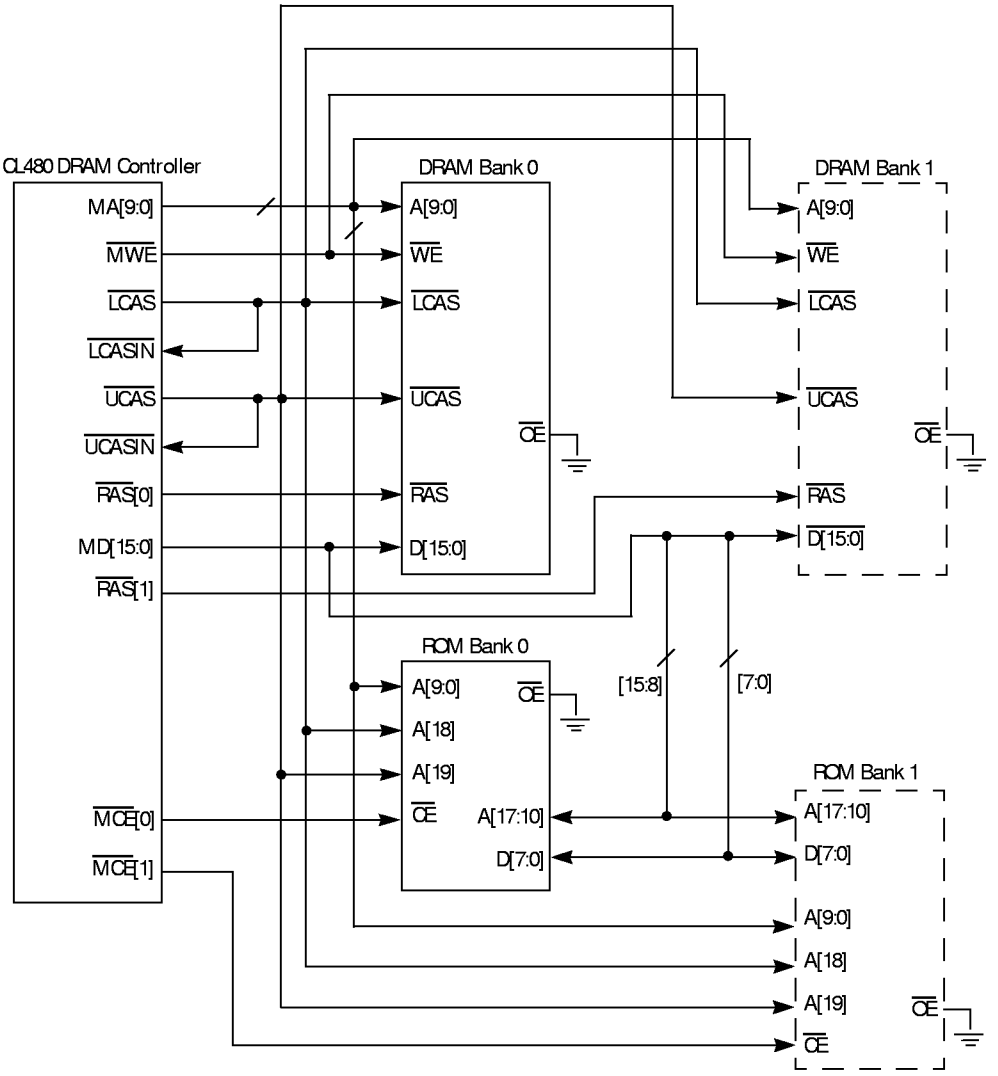


Figure 5-2 Local DRAM/ROM Implementation with 256K x 16 DRAMs

5.3 ROM Memory Bus Interface and Timing

Figure 5-3 illustrates the 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]. $\overline{\text{LCAS}}$ and $\overline{\text{UCAS}}$ are used for ROM address bits A[18] and A[19], respectively. A programmable number of wait cycles (3 to 34 GCKs) can be introduced between cycles 8 and 9.

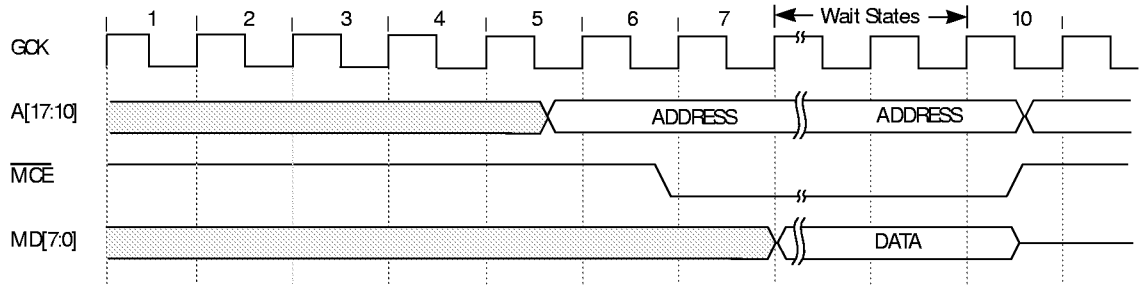


Figure 5-3 ROM Access Timing

5.3.1 ROM_CONFIG Parameter

The ROM_CONFIG parameter is provided (in the CL480VCD microcode only) as the means to program the ROM access time.

Note: This parameter is initialized at microcode start and is also described in Table 15-7.

Specifically, ROM_CONFIG specifies the number of GCKs to use when accessing the attached ROM. Bits 5 and 6 of this parameter must be set to 1, with bits [4:0] defined as the ROM_ACCESS time, or *RTA*.

Field	Bits	R/W	Description
	[6:5]	R/W	These bits should always be 1.
RTA	[4:0]	R/W	ROM Access time (in GBUS cycles). A "0" in *RTA* can be used for ROM accesses (T_{OE}) of less than 3 cycles (75 ns) at a 40-MHz GCK.

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 of from 2 to 33 GCK periods (from 50 to 825 ns for a clock frequency of 40 MHz). The formula for defining access time is as follows:

$$ROM\_ACCESS\ time = (RTA + 2)\ x\ \text{GCK}\ Period$$

Since ROM access mode does not use a signal equivalent to $\overline{\text{CASIN}}$ 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 ex-

ample, if the ROM access time is 200 ns, the recommended *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 $\overline{RAS}$ line, followed by several column addresses strobed-in by $\overline{CAS}$. Releasing the $\overline{RAS}$ line completes the set of cycles.

Refreshing is accomplished by asserting the $\overline{CAS}$ signal before the $\overline{RAS}$ signal. Refreshing is performed at regular intervals, determined by the value programmed into the REFRESH_CNT DRAM parameter (see page 50). Typical DRAM specifications require that all pages be refreshed every eight 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 that follow.

Note: All DRAM accesses are synchronized to the rising edge of GCK.

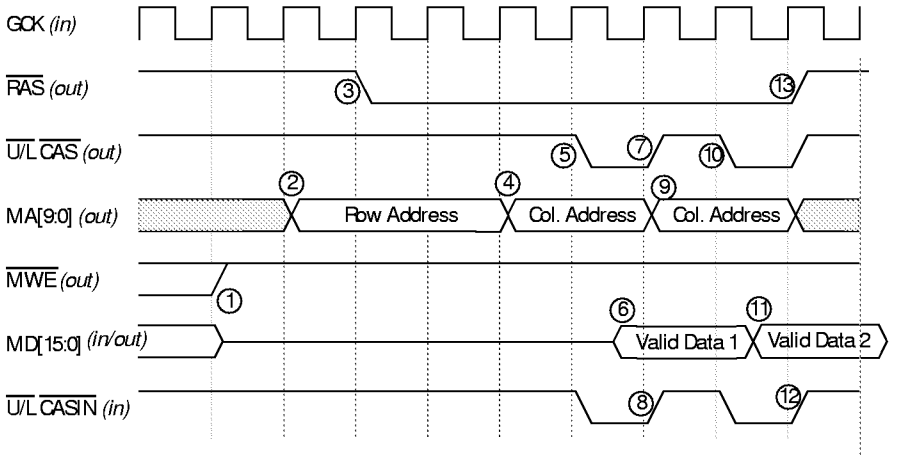


Figure 5-4 DRAM Page-Mode Read Timing

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 that follow.

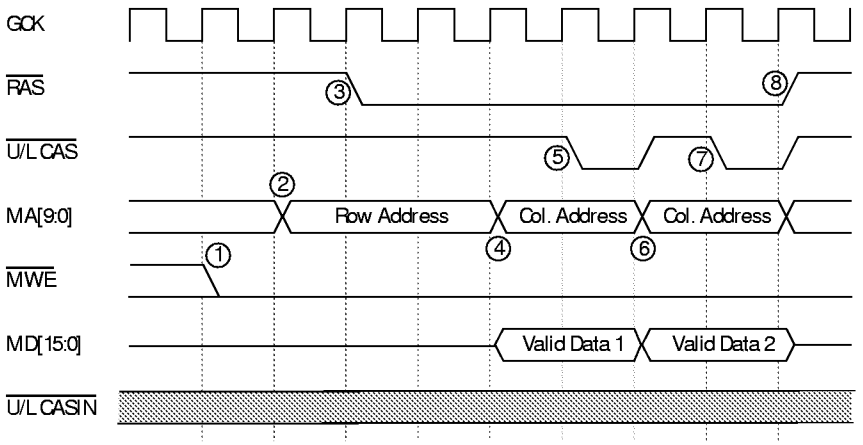


Figure 5-5 DRAM Page-Mode Write Timing

1. The CL480 starts a write cycle by asserting the write enable ($\overline{MWE}$) line LOW.
2. The CL480 then drives the row address on the memory address (MA) bus.
3. The CL480 asserts the $\overline{RAS}$ 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 $\overline{CAS}$ 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 $\overline{RAS}$ and $\overline{CAS}$ 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 $\overline{CAS}$ 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 $\overline{RAS}$ and $\overline{CAS}$ 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 that follow.

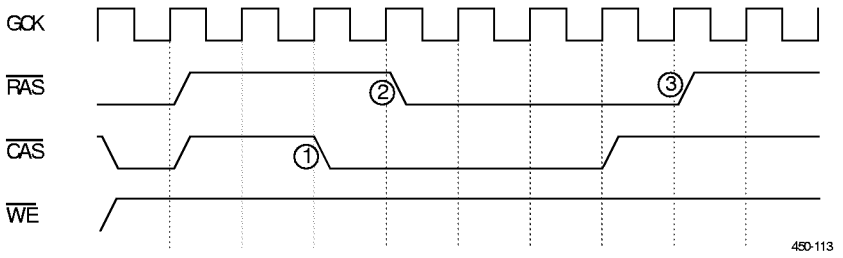


Figure 5-6 $\overline{\text{CAS}}$-before-$\overline{\text{RAS}}$ Refresh Cycle

1. The CL480 initiates a DRAM refresh cycle by asserting the $\overline{\text{CAS}}$ line while the $\overline{\text{RAS}}$ line is inactive (HIGH).
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_REFRESH Parameter

The DRAM_REFRESH parameter specifies the amount of time between DRAM refreshes in the number of GCK cycles. There is one field within this DRAM location which specifies the time between refreshes: COUNT (0x1fff). This field specifies the number of GCKs to count between refreshes.

Note: This parameter is initialized at the microcode start and is described in more detail in Table 15-7.

5.5 Memory Design Guidelines

This section contains guidelines for the memory interface of the CL480:

- 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. To minimize the length of signal traces, the CL480 and DRAM should be on the same side of the board, and pin 63 of the CL480 should be next to pin 22 of a SOJ 256 x 16 DRAM (pin 24 of a TSOP). The ROM can be on either side of the board. The ROM should be positioned so the address side of the ROM is near pin 63 of the CL480 and the data side of the ROM is extended out past pin 33 of the CL480.
- It is not necessary to use any termination if there is only one 256K x 16 DRAM connected to the CL480 and the DRAM signal traces are short. If four 256K x 4 DRAMs are used, series resistors (about 33 ohms) should be placed between the CL480 and the DRAM address and control signals.
- When using 256K x 16 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 256K x 16 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.

  Note: It is not necessary to program any of the internal registers in the CL480 when switching between these two kinds of DRAMs.

- 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 13.5 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 specification for an 80-ns DRAM. At the time this manual goes to press, a 70-ns 256Kx16 costs about $1.00 more than a 80-ns 256K x 16. To avoid this $1.00 penalty, it is best to either use a 40-MHz GCK or to test every system before shipping. This way, in the extremely unlikely event that a DRAM is too slow, this condition 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 requires special attention. 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.
- The CL480 can use either regular fast-page mode DRAMs or EDO (extended data-out) DRAMs.
- It is acceptable to have a 5-volt DRAM with a 3-volt ROM, or a 3-volt DRAM with a 5-volt ROM if the VDDMAX pins are wired to 5 volts.
- To reduce the problem of not being able to get a 256K x 16 DRAM in a particular package, it is prudent make the board layout accept either an SOJ, TSOP or ZIP 256K x 16 DRAM.
- To make a board that can use either two-$\overline{\text{CAS}}$/one-$\overline{\text{WE}}$ DRAMs or one-$\overline{\text{CAS}}$/two-$\overline{\text{WE}}$ DRAMs, the board layout should wire the $\overline{\text{MWE}}$ output of the CL480 to both of the pins that are used for $\overline{\text{WE}}$ on a dual-$\overline{\text{WE}}$ DRAM (these are pins 12 and 13 of a SOJ package, pins 14 and 15 of a TSOP, and pins 22 and 23 of a ZIP). The two $\overline{\text{CAS}}$ outputs of the CL480 should be wired to the two pins used for $\overline{\text{CAS}}$ inputs on a dual-$\overline{\text{CAS}}$ DRAM (these are pins 28 and 29 on a SOJ, pins 30 and 31 on a TSOP, and pins 38 and 39 on a ZIP). The only disadvantage of using a dual-$\overline{\text{WE}}$ DRAM with the CL480 is that the byte-write feature required for CD-I would not be supported. This byte-write feature is not used in videoCD players. In videoCD players, the $\overline{\text{LCAS}}$ output is always the same level as the $\overline{\text{UCAS}}$ output.

6
CD Interface

6.1 General Description

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

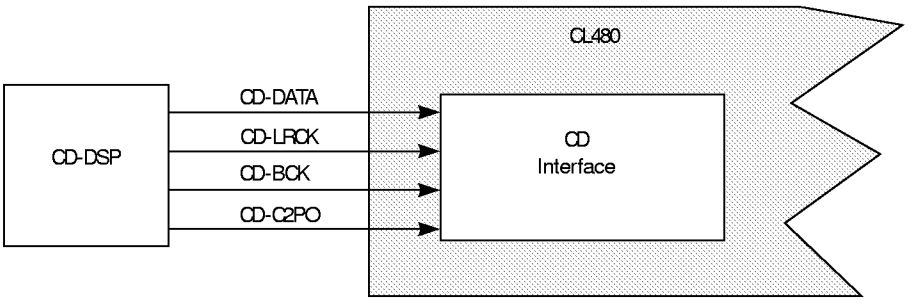


Figure 6-1 CD-Decoder Interface

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 tracks and Mode 2 Form 1 sectors (including error correction when not decoding MPEG)
  - Real-time processing and skipping of Mode 0 sectors
- In CDDA mode:
  - Direct output (pass-through) to the audio interface (DA-signals) of the CD input
  - Ignores data error pin, CD-C2PO

The CL480 stores the subheader 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 disk information, 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 a P-error check.

6.1.2 Mode Selection

The CL480VCD microcode can auto-detect the input bitstream type when the input bitstream is either a CDDA or CD-ROM bitstream as Figure 6-2 shows.

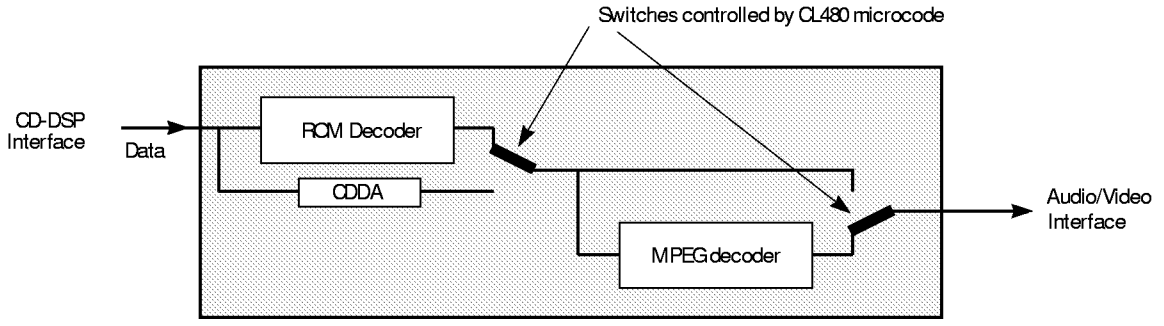


Figure 6-2 System Block Diagram Showing Bypass of ROM Decoder

When the PLAY_MODE parameter is set to CD_ROM_AUTO, auto detecting is enabled (see Table 15-7).

The CL480VCD microcode uses the CD-ROM synchronization characters in CD-ROM bitstreams to determine whether the input bitstream is a CD-ROM or CDDA bitstream. Before decoding begins, the CL480 microcode searches for sync characters for two seconds; then continues to examine the incoming bitstream for the presence of CD-ROM sync characters, switching between the two bitstream types accordingly.

The CL480VCD microcode can also be configured to expect bitstreams of one type only by setting the PLAY_MODE variable as follows:

- When set to CD_ROM_MPEG, the incoming bitstream is expected to contain CD_ROM sector information.
- When set to CD_ROM_CDDA, the incoming bitstream is expected to contain CDDA data.

Setting the PLAY_MODE variable to any of the above values disables the auto-detecting algorithm described above.

> *Note: Since the CL480PC microcode does not support CD-ROM or CDDA bitstreams, it does not examine the PLAY_MODE parameter and, therefore, operates as if this parameter was set to HOST_MPEG_SYSTEM.*

6.2 Signal Interface

The CL480 CD interface accepts the following signals as input. Their default settings are controlled by microcode at initialization and may be modified using the Configuration Area's CD_CONFIG parameter as Table 15-7 describes.

- CD-DATA - The serial data input signal, which receives the data stream in a format programmed to be either MSB or LSB first.
- CD-C2PO - 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 it is ignored in CDDA pass-through mode.
- CD-LRCK - This pin provides 16-bit word synchronization. Its polarity (left or right channel set 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

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

Figure 6-3 illustrates the input signal formats of the CD interface's six different serial modes.

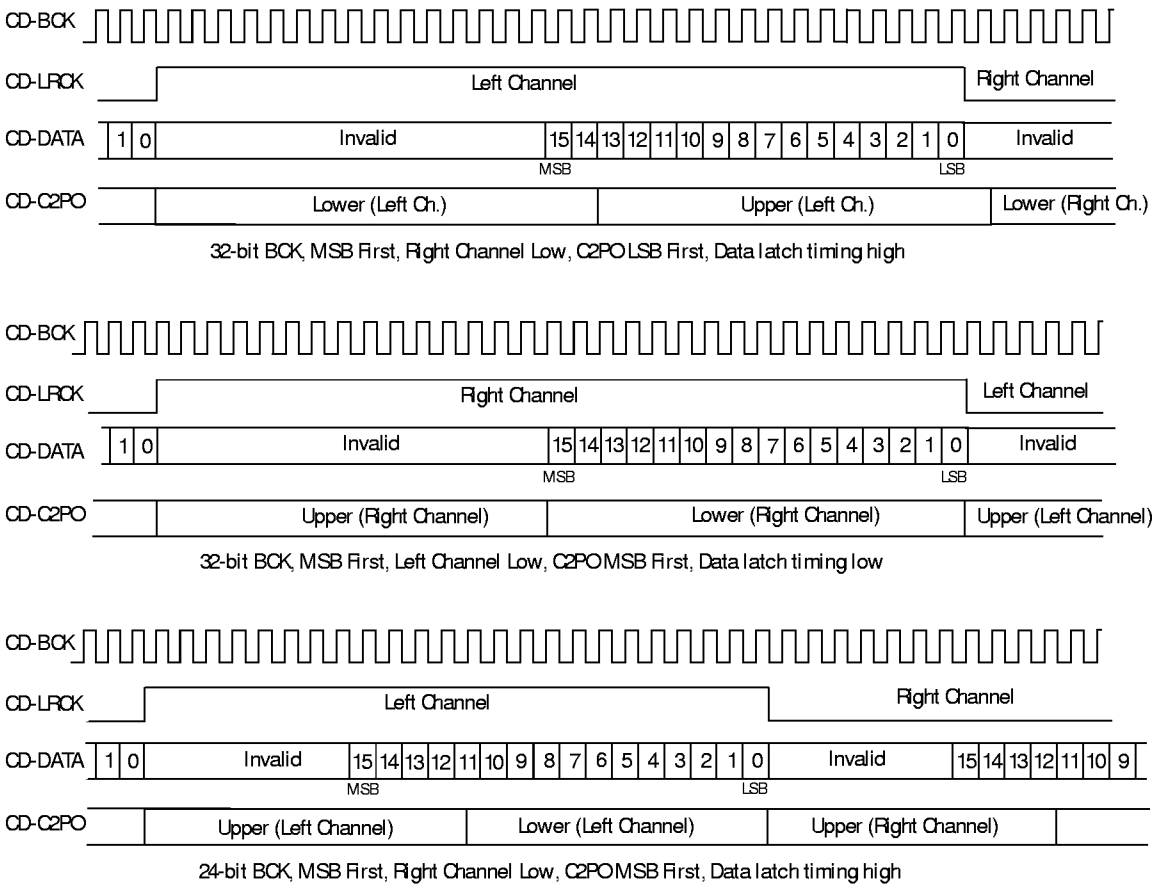


Figure 6-3 CD Interface Input Signal Formats

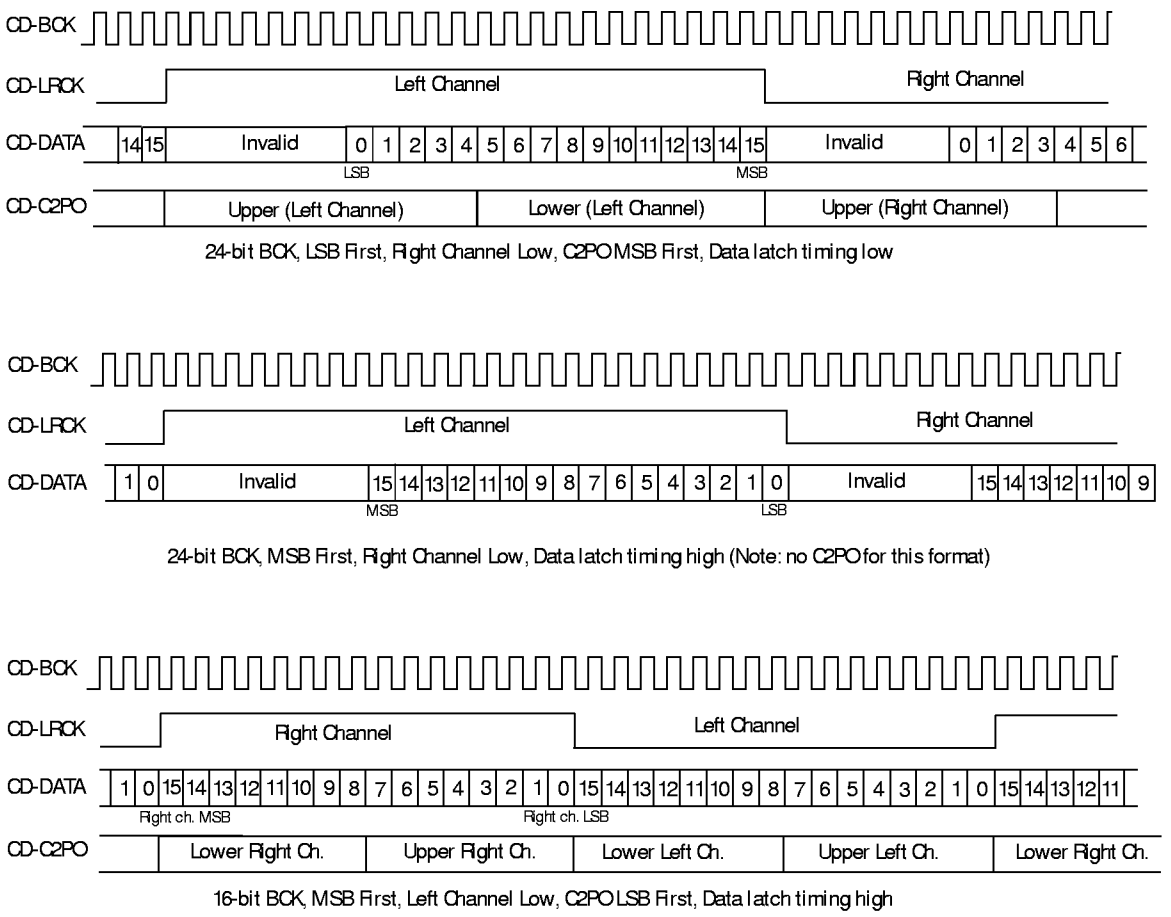


Figure 6-3 CD Interface Input Signal Formats (Continued)

6.3.2 Compact Disc Sector Format

Figure 6-4 illustrates the serial data CD formats.

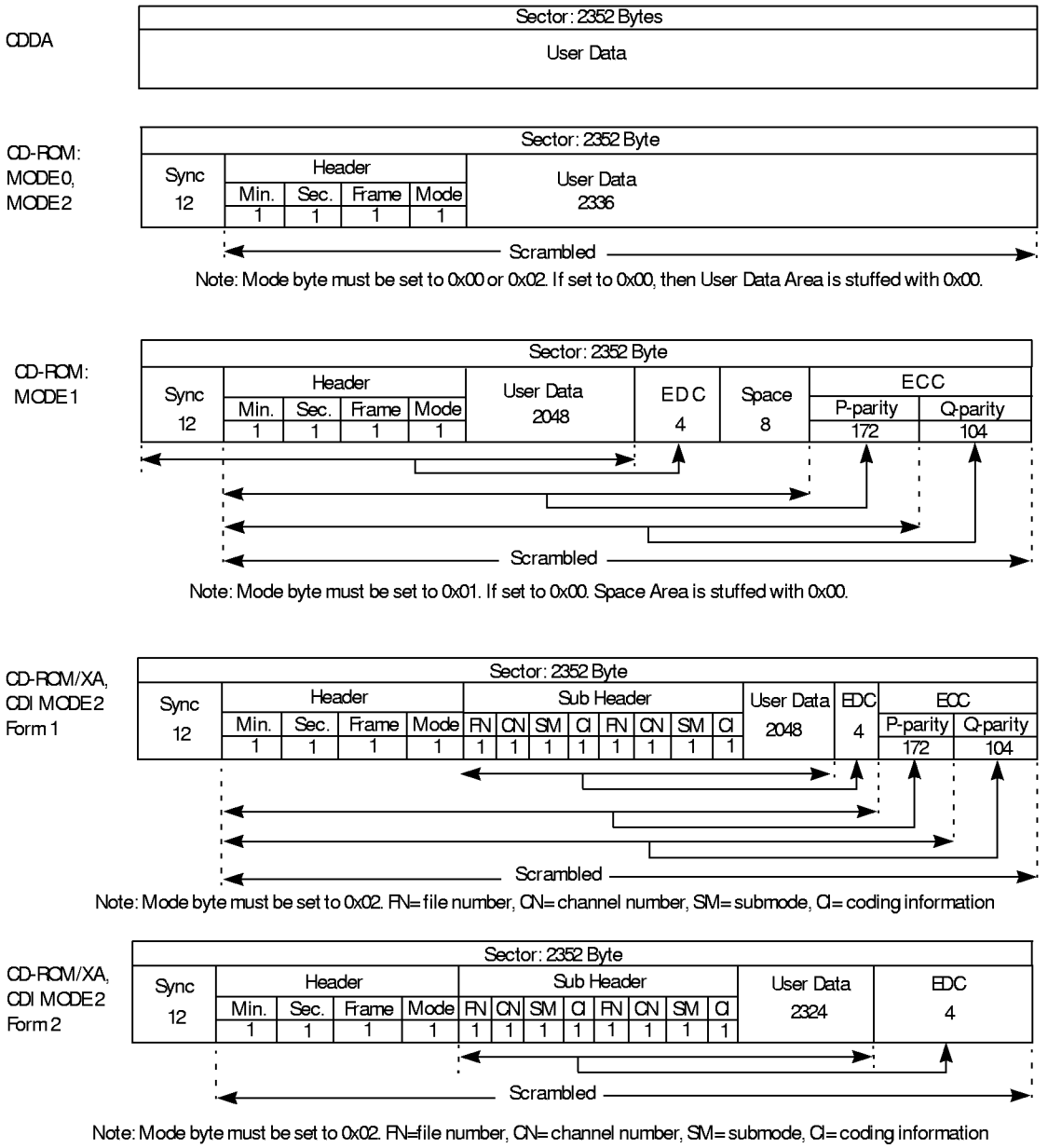
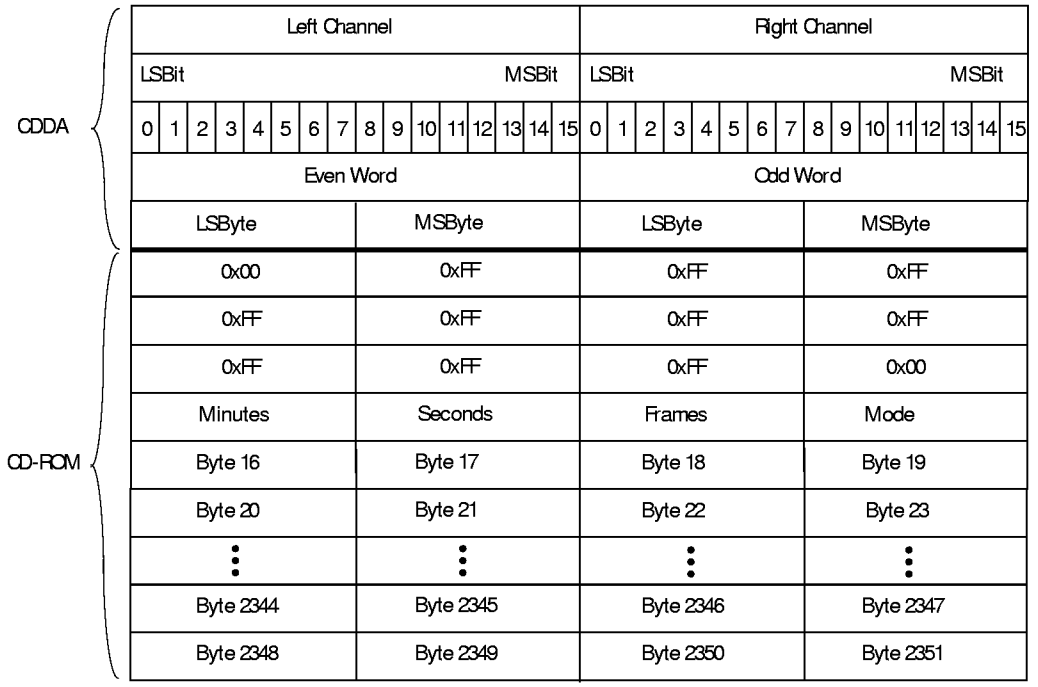


Figure 6-4 Serial Data CD Formats

6.3.3 CD-ROM Data Alignment



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

Figure 6-5 CD-ROM Data Alignment

6.4 Design and Operation Considerations

This section provides CD operation and design guidelines.

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, making the microcode 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. Audio pitch control can be managed 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 CL480, 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 one-sixth the GCK frequency. With a GCK of 40 MHz, a burst rate of up to 6.6 Mbits per second could be input using a constant BCK period.

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

7
Video Display Interface

This chapter of the user's manual discusses 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 either be 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 [0:255] (for computer displays) or [16:235] (for television). The video clock(VCK) can either be input or generated by dividing GCK by three. A composite sync (CSYNC) can be output in place of a $\overline{\text{VSYNC}}$ signal output.

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 synchronization pulse, followed by an interval of 22.5 lines without 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 without 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 output using one of two data formats:

- *YCbCr data*: Each pixel consists of a Y value for the luminance, or brightness, and two color values, Cb and Cr, for the chrominance. (This format is used internally by the MPEG standard.)
- *RGB (red, green, blue)*: This data output has a digital value that corresponds to the amplitude of each of the red, green, and blue signals.

7.2 Video Display Unit

In compliance 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 (configured by data in the Configuration Area of DRAM; see Table 15-7) will produce RGB in the range of either [16:235] or [0:255], where 16-16-16

and 0-0-0 RGB indicates the level of black, while 235-235-235 and 255-255-255 RGB indicate 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 video clock signal (VCK) and synchronized to external devices using the Horizontal Sync ($\overline{\text{HSYNC}}$) and Vertical Sync ($\overline{\text{VSYNC}}$) signals.

7.3 Timing Generation

The CL480 outputs pixels in either YCbCr (4:2:2 mode) or RGB (4:4:4 mode) using a 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 Table 15-7.)

The CL480 supports the following VCK modes:

- *24-bit and 15-bit RGB*: In this mode, the B component is presented as the MSB and the R component as the LSB.
- *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 are 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 (B Silicon)

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[1]	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[2]	C4	C3	C2	C1	C0			Y7	Y1	Y0	C7	C6	C5				Y6	Y5	Y4	Y3	Y2			
8-bit YCbCr[3]								YC7	YC1	YC0							YC6	YC5	YC4	YC3	YC2			

1. Note: 15-bit RGB is a dithered version of 24-bit RGB, so the pin assignments are the same.
2. Note: In silicon revision A, 16-bit YCbCr data, Y[7:0] is on VD[15:8] and C[7:0] is on VD[7:0].
3. Note: In silicon revision A, 8-bit YCbCr data, YC[7:0] is on VD[15:8].

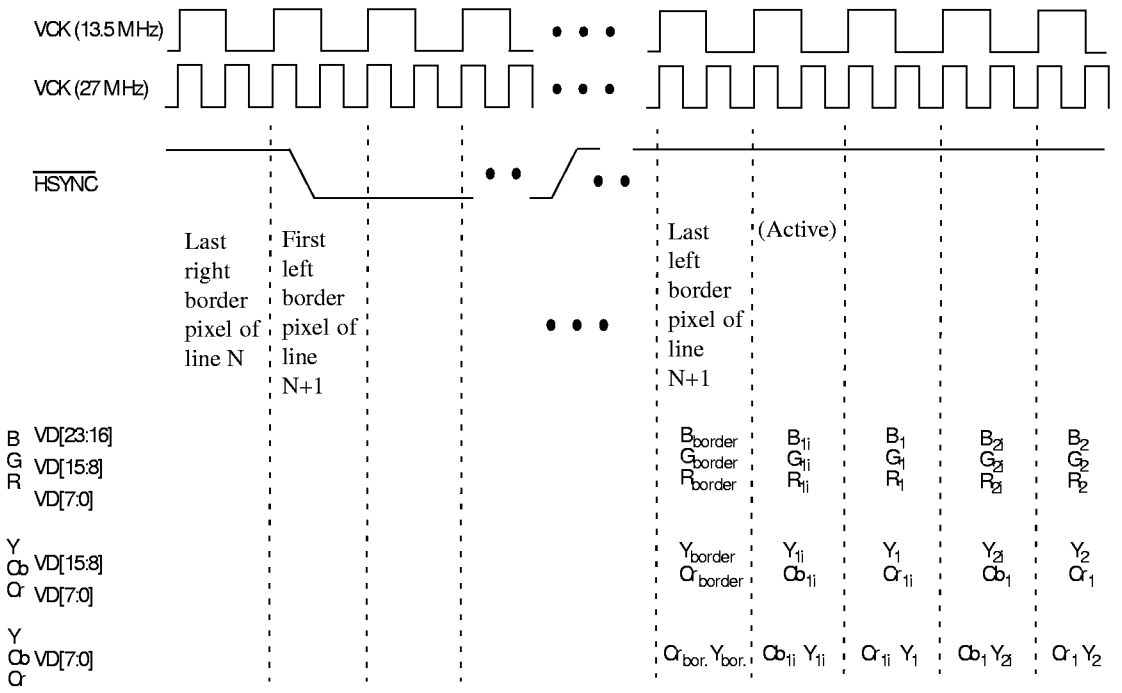


Figure 7-1 VCK versus $\overline{\text{HSYNC}}$

Data changes in the VCK period where $\overline{\text{HSYNC}}$ falls and on every other rising VCK edge after that. (For a 13.5 MHz clock, changes in the VCK period would occur every rising edge instead of every other rising edge.) The VCK-to-VD delay allows VD to be latched with the rising edge of VCK from either or both VCK periods.

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

- $\overline{\text{VSYNC}}$ (or CSYNC)/ $\overline{\text{HSYNC}}$ / VCK Out
- $\overline{\text{VSYNC}}$/ $\overline{\text{HSYNC}}$ / VCK In
- $\overline{\text{VSYNC}}$ (or CSYNC)/ $\overline{\text{HSYNC}}$ Out and VCK In
- $\overline{\text{VSYNC}}$/ $\overline{\text{HSYNC}}$ In and VCK Out

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

Note: CSYNC (composite SYNC) may be selected (as an output only) instead of VSYNC and is provided on the VSYNC pin. This feature is selectable by enabling the VIDEO_MODE2 DRAM parameter described in Table 15-7.

7.3.1 (VSYNC or CSYNC)/HSYNC/VCK Out

In this mode, as shown in Figure 7-2, VSYNC (or CSYNC), HSYNC and VCK are generated internally and sent to the NTSC/PAL encoder. In this configuration, VCK is fixed to 13.5 MHz and is obtained by dividing GCK (40.5 MHz) by three.

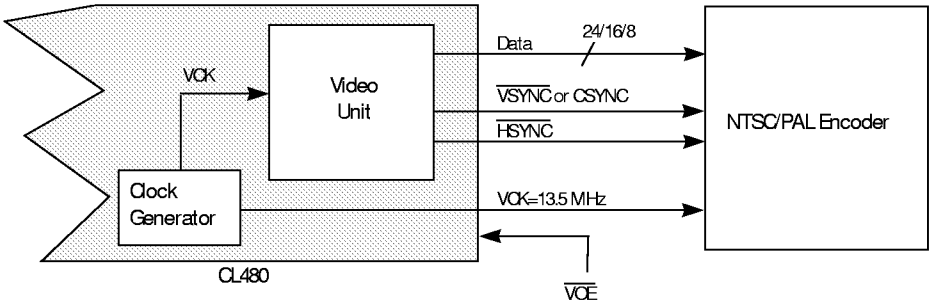


Figure 7-2 VSYNC/HSYNC/VCK Out Mode

7.3.2 VSYNC/HSYNC/VCK In

In this mode, as shown in Figure 7-3, the video timing signals— VCK, VSYNC and HSYNC —are generated by the external signal generator. VCK and GCK can be asynchronous.

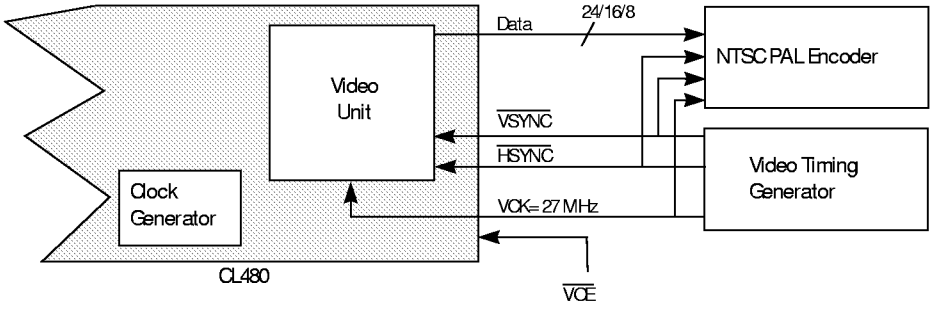


Figure 7-3 VSYNC/HSYNC/VCK In

7.4 Output Format Conversion

If the output video frame rate doesn't match the frame rate of the video bitstream, the CL480 will automatically convert the frame rate of the video bitstream to match the display frame rate.

7.5 Video Overlays

The CL480 microcode provides for the display of simple, user-definable graphics called video overlays. For example, an overlay can be used to indicate that the user has paused the video, or it can show a bar graph to help the user adjust audio volume. In these applications, either the host processor or the CL480 CPU would write the overlay information into the CL480 DRAM.

7.6 Selecting Video Features

Video features may be selected by accessing the Configuration Area DRAM locations as described in the following sections. See Chapter 15 for a more detailed description of the DRAM locations described.

7.6.1 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 the DRAM locations (see Table 15-7) as described below:

- *24-bit RGB*: This mode is selected by the default setting (set to 1) of the RGB_MODE field of the VIDEO_MODE parameter.
- *15-bit RGB:* This mode is a dithered version of the 24-bit RGB mode, with five bits each for R, G and B. It is set by writing a 1 to the RGB_15 field of the VIDEO_MODE2 parameter (assuming RGB mode is already enabled as described above). The 15-bit RGB mode is intended for systems such as PC add-in cards and game boxes that write the CL480 output into a 16-bit wide memory before outputting RGB.
- *16- and 8-bit YCbCr:* These modes are chosen by first setting the RGB_MODE field of the VIDEO_MODE parameter to 0, and then setting the VBUS_MODE of the VIDEO_MODE parameter to either 1 (16 bits) or 0 (8 bits).

7.6.2 Video Timing Signals: Type and Direction

The CL480 can be configured to accept or output video timing signals as follows:

- *Selecting CSYNC versus $\overline{VSYNC}$:* The CSYNC_VSYNC field of the VIDEO_MODE2 configuration parameter selects whether the synchronization signals are CSYNC (composite) or separate ($\overline{VSYNC}$).
- *Selecting the direction of VCK and $\overline{HSYNC}$/$\overline{VSYNC}$:* The VCLK_OUT and SYNC_OUT fields of the VIDEO_MODE configuration parameter select the direction of VCK (the video clock) and the direction of $\overline{HSYNC}$ and $\overline{VSYNC}$ (the sync signals), respectively.

  Note: The synchronization signals $\overline{HSYNC}$ and $\overline{VSYNC}$ are both either inputs or outputs; they can't be configured separately.

7.6.3 $\overline{HSYNC}$ and $\overline{VSYNC}$ Output Parameters

The following parameters are selectable when the CL480 is generating both $\overline{HSYNC}$ and $\overline{VSYNC}$ signals for either NTSC or PAL video output:

Note: The units of all the parameters are in VCK periods.

- *$\overline{HSYNC}$ LOW Time:* NTSC_HSYNC_LO and PAL_HSYNC_LO parameters specify the time $\overline{HSYNC}$ is LOW for both NTSC and PAL, respectively (see Figure 7-5).
- *$\overline{HSYNC}$ HIGH Time:* NTSC_HSYNC_HI and PAL_HSYNC_HI parameters specify when $\overline{HSYNC}$ is HIGH for both NTSC and PAL, respectively (see Figure 7-5).
- *$\overline{VSYNC}$ LOW Relative to $\overline{HSYNC}$ LOW for Bottom Field Only:* VSYNC_EVEN_TP and VSYNC_ODD_TP specify the position relative to $\overline{HSYNC}$ going LOW (for both even and odd fields, respectively) that $\overline{VSYNC}$ will go LOW for the bottom field (see Figure 7-5).

  Note: The VSYNC_EVEN_TP and VSYNC_ODD_TP parameters are used for both NTSC and PAL.

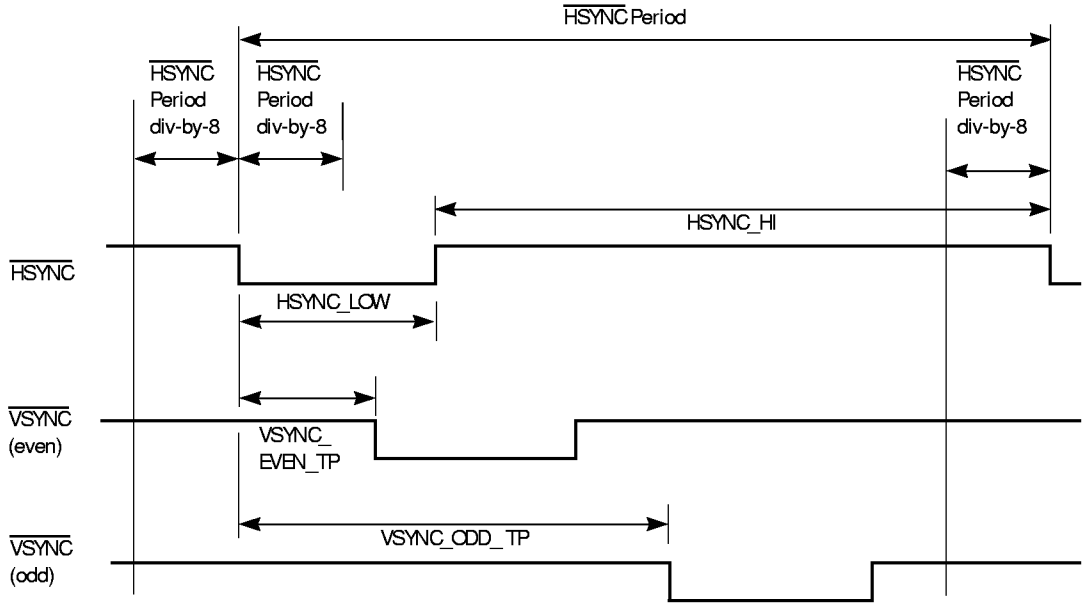


Figure 7-5 HSYNC and VSYNC Timing

7.6.4 HSYNC and VSYNC Input Parameters

If the HSYNC and VSYNC signals are supplied externally, the FIRST_FIELD configuration parameter can be used to determine which field, odd or even, is displayed when the leading edge of VSYNC is within the video window.

The position of the leading edge of the VSYNC signal is examined relative to a window whose width is approximately one-eighth the HSYNC period centered about the leading edge of the HSYNC signal, as shown in the top portion of Figure 7-5.

The relationship between the field to be displayed, the FIRST_FIELD configuration parameter, and the VSYNC leading-edge position is defined in Table 7-2.

Table 7-2 Setting Field Display when HSYNC and VSYNC are Inputs

FIRST_FIELD Parameter	Position of VSYNC Leading Edge	Field Displayed
Non-zero	Inside window	Even
Non-zero	Outside window	Odd
Zero	Inside window	Odd
Zero	Outside window	Even

7.6.5 Video Formats

The CL480 supports one of the following four video output formats using the SetVideoFormat() command or the VIDEO_FORMAT DRAM parameter:

- PAL
- NTSC
- Multisync (PAL or NTSC)
- No field repeat

Use the parameters listed in Section 7.6.3 to specify $\overline{\text{HSYNC}}$ and $\overline{\text{VSYNC}}$ timing.

> *Note: If the output video frame rate doesn't match the frame rate of the incoming video bitstream, the CL480 will automatically convert the frame rate of the incoming video bitstream to match the display frame rate.*

7.6.6 Windowing

The CL480 allows you to specify the size and positioning of the video output window and what portion of the decoded picture to display within this window. Windowing functionality differs, depending upon which CL480 chip version is being used.

The B-series or greater silicon chip supports more windowing functionality than the A3 chip. The A3 chip only supports centering of the frame vertically in the output display area (TOP_BORDER) and positioning of the display horizontally (LEFT_BORDER); it does not support centering of the video image using the LEFT_BORDER DRAM parameter. The B-series, however, supports all of the windowing functionality described below and shown in Figure 7-6:

- *LEFT_BORDER and TOP_BORDER:* These parameters specify where the upper-leftmost corner of the display is to start.
- *WIDTH and HEIGHT:* These parameters specify the width and height of the output display starting at the point specified by the TOP_BORDER and LEFT_BORDER parameters.
- *XOFFSET and YOFFSET:* These parameters specify the upper-leftmost corner of the video frame to display at the point indicated by LEFT_BORDER and TOP_BORDER.

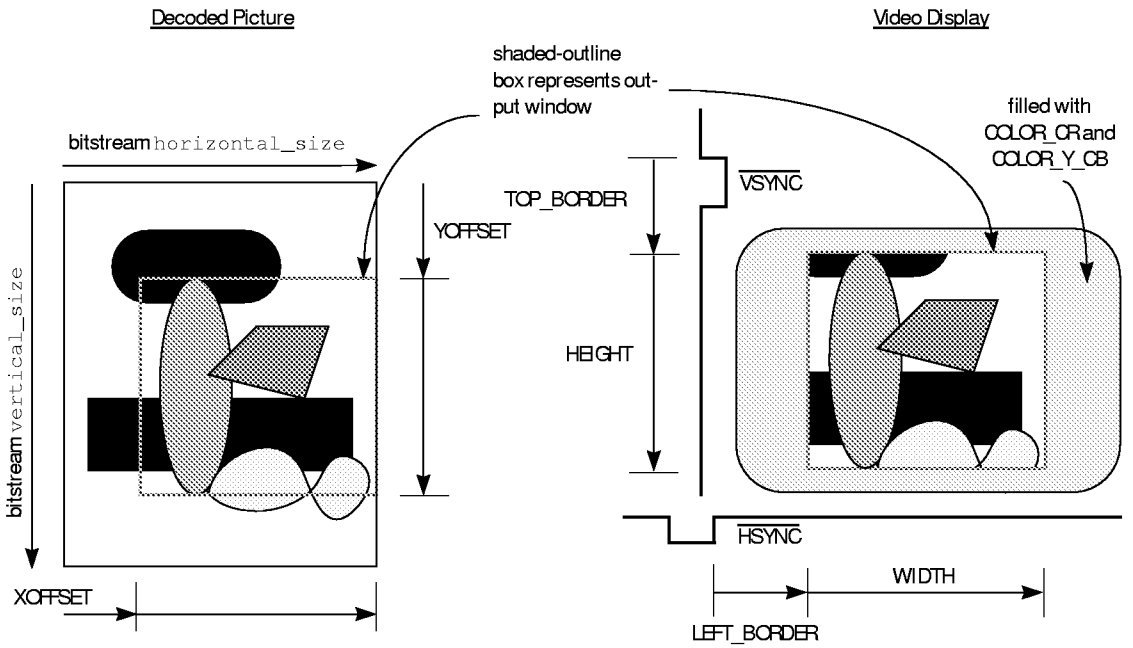


Figure 7-6 Video Display Configuration

Setting either LEFT_BORDER or TOP_BORDER to zero will cause the display to be centered in that dimension, while setting either WIDTH or HEIGHT to zero will cause the picture width and height values from the bitstream to be used. If the XOFFSET or YOFFSET parameters are zero, the video frame will be centered in the corresponding dimension of the video display area.

7.6.7 Border Color

The display border color is controlled by the COLOR_CR and the COLOR_Y_CB Configuration Area DRAM parameters shown in Figure 7-6 and described in Table 15-7.

7.6.8 Interpolation

The CL480 performs both horizontal and vertical interpolation using the EVEN_INTERPCOEF and ODD_INTERPCOEF DRAM Configuration Area parameters (see Table 15-7).

Horizontal Interpolation

Horizontal interpolation is used to generate the extra pixels when converting from SIF input to CCIR 601 output. The EVEN_INTERPCOEF and ODD_INTERPCOEF parameters are used to set horizontal interpolation for the even and odd video fields, respectively. Bits 10 and 11 of these parameters specify the chroma interpolation values, while bits 8 and 9 specify the luma interpolation coefficients. The values for each of these two-bit arguments are 0 = 0, 1 = 1/8, 2 = 1/4, and 3 = 1/2. At the left border, the video unit copies the first pixel twice; at the right border, the video unit copies the last pixel once.

Vertical Interpolation

Vertical interpolation is used to generate two fields from the encoded SIF frame. Vertical interpolation is controlled by using the EVEN_INTERPCOEF and ODD_INTERPCOEF configuration parameters. It is implemented by performing a weighted average of the SIF input lines to obtain the CCIR 601 lines, where the value of each of these two-bit arguments is 0 = 0, 1 = 1/4, 2 = 1/2 or 3 = 3/4. Interpolation is performed separately for each field output, and interpolation coefficients are provided for each field. (See Table 15-7.)

7.6.9 Video Overlay (Graphics Display)

The DisplayGraphics() command is used to display graphic images loaded into DRAM. The graphic image must be loaded into an unused area of DRAM. The DisplayGraphics() command specifies the first and last line of the graphic image displayed, along with the amount of fade to apply to the image. See Chapter 13 for further information on the DisplayGraphics() command.

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.

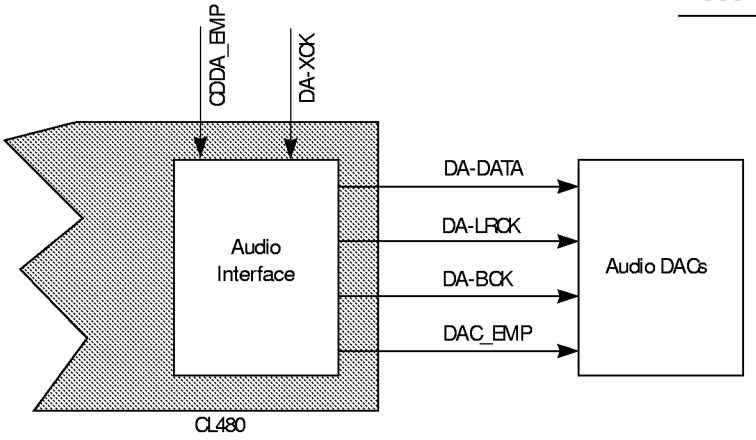


Figure 8-1 Audio Interface Block Diagram

The audio decoding process supports decoding of MPEG-1 layer I and II audio bitstreams. The audio header information is parsed, processed, and stored in DRAM. The resultant audio information is then converted to audio samples by the special processing units that write the audio samples to the audio output FIFO. Audio decoding is only performed when a complete audio frame is contained in the audio bitstream buffer and there is enough room in the audio output buffer for the resultant audio samples.

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 AUDIO_CONFIG parameter.
DAC_EMP	Out	In both modes, this output 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	In CDDA pass-through mode, this input signal is routed to DAC_EMP.

8.2.1 Configuring Data Output

The audio interface supports several serial data output modes that may be selected from the AUDIO_CONFIG parameter described in Table 15-7. The following parameters may be set: LEFT_LRCK_HI, RIGHT_LRCK_LO, OUTPUT_24_BITS, and LSB_FIRST.

Table 8-2 shows which audio channel is output when the DA_LRCK signal is HIGH and LOW. The relationship between the audio channel output and the DA_LRCK signal depends upon the state of LEFT_LRCK_HI, RIGHT_LRCK_LO, and OUTPUT_24_BITS.

Table 8-2 Output Audio Channel when DA_LRCK is HIGH and LOW

OUTPUT_24_BITS	RIGHT_LRCK_LO	LEFT_LRCK_HI	DA_LRCK (HIGH)	DA_LRCK (LOW)
0	0	0	Left	Right
0	0	1	Left	Left
0	1	0	Right	Right
0	1	1	Right	Left
1	0	0	Right	Left
1	0	1	Right	Right
1	1	0	Left	Left
1	1	1	Left	Right

8.2.2 Controlling the Audio Output Level

Part of the audio information processing performed by the special processing units is the attenuation of the audio data. The attenuation of the audio data is controlled by the AUDIO_MUTE configuration parameter described in Table 15-7. Three attenuation settings are possible: no attenuation, -12 db attenuation, and no audio output.

8.2.3 Setting the DAC_EMP Signal

In CDDA pass-through mode, CDDA_EMP is routed to the emphasis output, DAC_EMP. In MPEG audio decoding (including still), DAC_EMP is driven with the emphasis indicated in the MPEG audio stream (the least significant bit of the emphasis field of the MPEG audio frame header).

Table 8-3 lists the value of the DAC_EMP output signal corresponding to the values of the CDDA_EMP input signal, the CDDA_EMPHASIS DRAM configuration parameter, and the CL480 silicon revision.

Table 8-3 DAC_EMP Output Determination

Silicon Revision	CDDA_EMPHASIS Parameter	CDDA_EMP Input Signal	DAC_EMP Output Signal
A3	X0	X	0
A3	X1	X	1
B0 or greater	00	X	0
B0 or greater	01	X	1
B0 or greater	1X	0	0
B0 or greater	1X	1	1

8.2.4 Low-Power Operation

The CL480 operates in two automatic power reduction modes: when decoding stereo MPEG audio only (no video), and when in CDDA pass-through mode.

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 the DA-BCK and DA-LRCK signals. However, nearly all MPEG-1 titles are encoded with 44.1 kHz audio sample rates.

The external reference clock, DA-XCK, can be either 384 or 256 times the sampling frequency. The bit clock signal, 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 the AUDIO_CONFIG parameter (see Table 15-7). Table 8-4 shows the relationship between DA-XCK, DA-BCK, and DA-LRCK signals for a sampling frequency (f_s) of 44.1 kHz for the two modes of operation: 384 x f_s and 256 x f_s.

Table 8-4 DA-XCK, DA-BCK, and DA-LRCK for 44.1 KHz

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

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

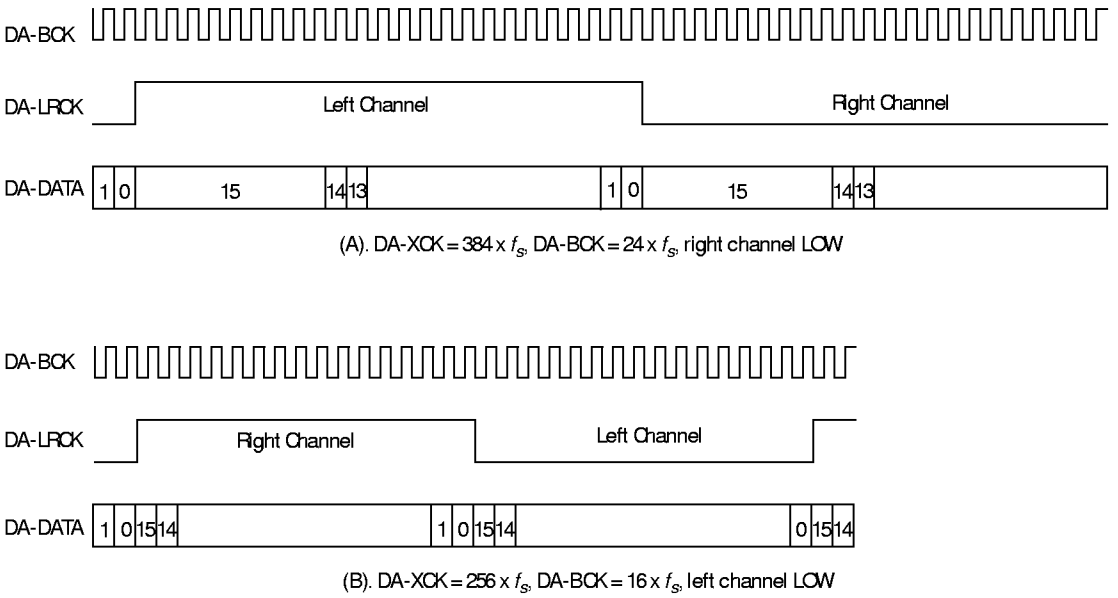


Figure 8-2 Audio Interface Clocking Modes

8.4 Operation and Design Guidelines

This section provides audio operation and design guidelines.

8.4.1 DA-XCK and Cycle Jitter

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 one-bit audio DAC. C-Cube's current microcode expects DA-XCK and CD-BCK to 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 future microcode releases.

9 Electrical and Physical Specifications

This chapter describes the CL480's electrical and mechanical characteristics.

9.1 Operating Conditions

Tables 9-1 through 9-3 specify the CL480's electrical characteristics.

Table 9-1 Operating Conditions

Parameters		Commercial Min	Commercial Max	Unit
V_{DD3}	Supply Voltage	2.7[1]	3.6	V
V_{DDMAX}	VDDMAX pin (max input voltage)	V_{DD3}	5.25	V

1. Silicon revision A3 is 3.0V.

Table 9-2 Absolute Maximum Ratings[1]

Parameter	Value
Supply voltage (V_{DD3})	-0.3 to 4.5 V
Input voltage[2]	-0.3 to V_{DDMAX} + 0.25V
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.
2. 5.5V is the maximum for all input voltages except XTLIN (pin 19), which should be 4.0V.

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
$V_{IH(OSC)}$ [2]	High-level OSC input V.	V_{DD3} = MAX	$0.8V_{DD3}$		V_{DD3}+0.2V	V
$V_{IL(OSC)}$ [2]	Low-level OSC input V.		0		$0.2V_{DD3}$	V
V_{OH27}	High-level output voltage	V_{DD} = 2.7V, I_{OH} = -2 mA	2.1			V
V_{OH30}	High-level output voltage	V_{DD} = 3.0V, 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 [3] @V_{DD3} = 2.7V	GCK = 40MHz V_{IN} = 0 or 2.7V, C_L=50pF, Ta=70°C		150	200	mA
I_{DD33}	Supply Current @V_{DD3} = 3.3V	GCK = 40MHz V_{IN} = 0 or 3.3V, C_L=50pF, Ta=70°C		180	240	mA
I_{DD36}	Supply Current @V_{DD3} = 3.6V	GCK = 40MHz V_{IN} = 0 or 3.6V, C_L=50pF, Ta=70°C		200	270	mA
C_{IN}	Input Capacitance [1]			10		pF
C_{OUT}	Output Capacitance [1]			12		pF
C I/O	Output Capacitance [1]			12		pF

1. Not 100% tested, guaranteed by design characterization
2. This value refers to XTL IN, pin 19.
3. 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, 21, 22, 29, 49, 65, 77, 81, 98, 110, and 112 should be connected to 2.7 to 3.6 volts. 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 5-volt DRAM and a 3-volt host are being used, connect both these pins to 5 volts.

9.1.3 Power-Up and Power-Down Constraints

During power-up and power-down, the 5-volt supply voltage (VDDMAX) should always be at a higher voltage than the 3.3-volt supply. The 5-volt 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 3.3 volts. If the 3.3-volt supply is more than about .6 volts above the 5-volt supply, the diode between the P-diffusion and the N-well conducts current, which could latch up the chip or destroy the bond wires for the 5-volt supply.

9.2 AC Timing Characteristics

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

- GCK, $\overline{\text{RESET}}$ and CFLEVEL Timing:
  - Figure 9-1, Timing Diagram - GCK (XTLIN)
  - Figure 9-2, Timing Diagram - $\overline{\text{RESET}}$
  - Figure 9-3, Timing Diagram - CFLEVEL
  - Table 9-4, Timing Characteristics, GCK, $\overline{\text{RESET}}$ and CFLEVEL
- Host Bus Interface Timing:
  - Figure 9-4, Host Interface Local Register, Write Operation
  - Table 9-5, Timing Characteristics, Local Register Write
  - Figure 9-4, Host Interface Local Register, Read Operation
  - Table 9-6, Timing Characteristics - Local Register Read
- DRAM/ROM Bus Timing:
  - Figure 9-6, Local DRAM Bus Timing
  - Figure 9-7, Local DRAM $\overline{\text{CAS}}$-before-$\overline{\text{RAS}}$ Refresh
  - Table 9-7, Timing Characteristics - Local DRAM Bus
- CD Interface Timing:
  - Figure 9-8, CD Input: Data Latch Timing High
  - Figure 9-9, CD Input: Data Latch Timing Low
  - Table 9-8, CD Input Timing
- CD Subcode (CD+G) Interface Timing:
  - Figure 9-10, Clocking CD Subcode Data
  - Figure 9-11, Separate CD Subcode Control Signals
  - Figure 9-12, Composite CD Subcode Control Signals
  - Table 9-9, Timing Characteristics: CD+G (Preliminary)
- Video Bus Timing
  - Figure 9-13, VCK (In and Out)
  - Table 9-10, Timing Characteristics: VCK

- Figure 9-14, $\overline{\text{VOE}}$ and VD
- Table 9-11, Timing Characteristics: $\overline{\text{VOE}}$ and VD
- Figure 9-15, VCK and VD (pixel data)
- Table 9-12, Timing Characteristics: VCK and VD
- Figure 9-16, $\overline{\text{VSYNC}}$ and $\overline{\text{HSYNC}}$ Out
- Figure 9-17, $\overline{\text{VSYNC}}$ and $\overline{\text{HSYNC}}$ In
- Figure 9-18, VCK Out to $\overline{\text{HSYNC}}$ Out
- Figure 9-19, $\overline{\text{HSYNC}}$ In to VCK In
- Figure 9-20, VCK In to $\overline{\text{HSYNC}}$ Out
- Figure 9-21, $\overline{\text{HSYNC}}$ In to VCK Out
- Table 9-13, Timing Characteristics: VCK and $\overline{\text{HSYNC}}$
- Figure 9-22, VCK Out to CSYNC/$\overline{\text{VSYNC}}$ Out
- Figure 9-23, VCK In to CSYNC/$\overline{\text{VSYNC}}$ Out
- Table 9-14, Timing Characteristics: VCK and CSYNC/ $\overline{\text{VSYNC}}$

- Audio Bus Timing
  - Figure 9-24, Audio Interface Timing Modes
  - Table 9-15, Timing Characteristics: Audio

9.2.1 GCK (XTLIN), $\overline{\text{RESET}}$ and CFLEVEL Timing

This section describes CL480 GCK, $\overline{\text{RESET}}$ and CFLEVEL timing.

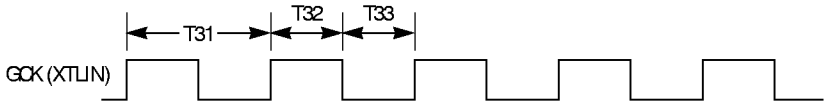


Figure 9-1 Timing Diagram - GCK (XTLIN)

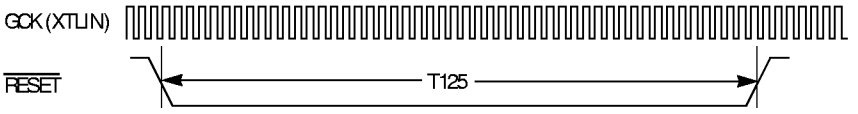


Figure 9-2 Timing Diagram - $\overline{\text{RESET}}$

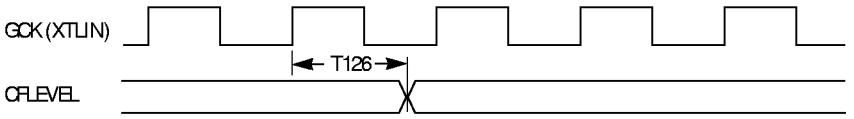


Figure 9-3 Timing Diagram - CFLEVEL

Table 9-4 Timing Characteristics - GCK, $\overline{\text{RESET}}$, and CFLEVEL

Time	Description	Min	Max	Units
T31	GCK (XTLIN) period	24.5	25.6	ns
T32	GCK (XTLIN) HIGH pulse width	10		ns
T33	GCK (XTLIN) LOW pulse width	10		ns
T125	$\overline{\text{RESET}}$ pulse width	10 (T31)		
T126	CFLEVEL access time from GCK HIGH	38		ns

9.2.2 Host Bus Interface Timing

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

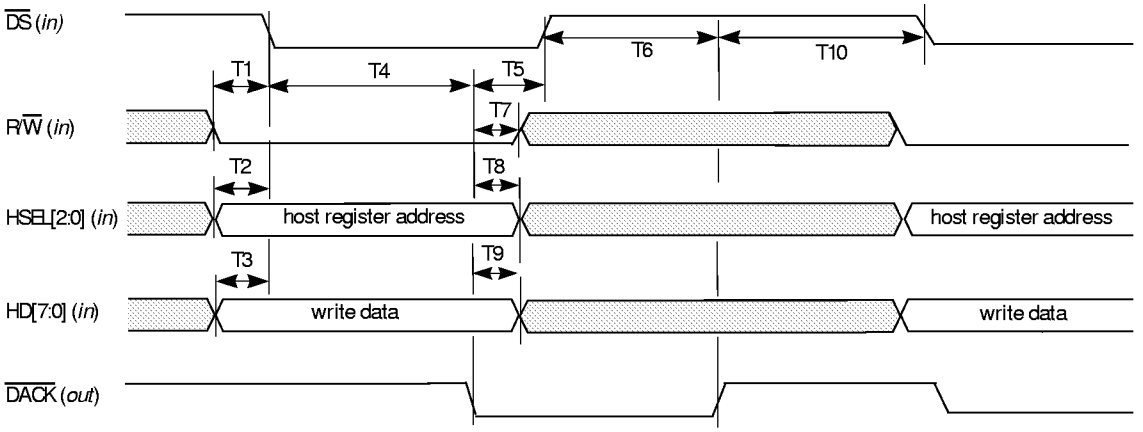


Figure 9-4 Host Interface Local Register, Write Operation

Table 9-5 Timing Characteristics - Local Register Write

Time	Description	Min	Max	Units
T1	R/$\overline{\text{W}}$ valid to $\overline{\text{DS}}$ LOW	0		ns
T2	HSEL[2:0] valid to $\overline{\text{DS}}$ LOW	0		ns
T3	HD[7:0] valid to $\overline{\text{DS}}$ LOW	0		ns
T4	$\overline{\text{DS}}$ LOW to $\overline{\text{DACK}}$ LOW	(note)[1]	(note)[2]	
T5	$\overline{\text{DACK}}$ LOW to $\overline{\text{DS}}$ HIGH	3		ns
T6	$\overline{\text{DS}}$ HIGH to $\overline{\text{DACK}}$ HIGH	1 GCK	2 GCK	
T7	R/$\overline{\text{W}}$ holds after $\overline{\text{DACK}}$ LOW	0		ns
T8	HSEL[2:0] holds after $\overline{\text{DACK}}$ LOW	0		ns
T9	HD[7:0] holds after $\overline{\text{DACK}}$ LOW	0		ns
T10	$\overline{\text{DACK}}$ HIGH to $\overline{\text{DS}}$ LOW (next operation)	5		ns

1. (GCK * 2) when A_MSB, A_MB, A_LSB and D_LSB write operation; (GCK * 21) when CFIFO D_MSB write; (GCK * 6) when D_MSB write.
2. (GCK * 3) when A_MSB, A_MB, A_LSB and D_LSB write operation; (GCK * 22) when CFIFO D_MSB write; (GCK * 6 + GBUS waiting time) when D_MSB write. Note: GBUS waiting time can be from 0 to over 500 GCKs.

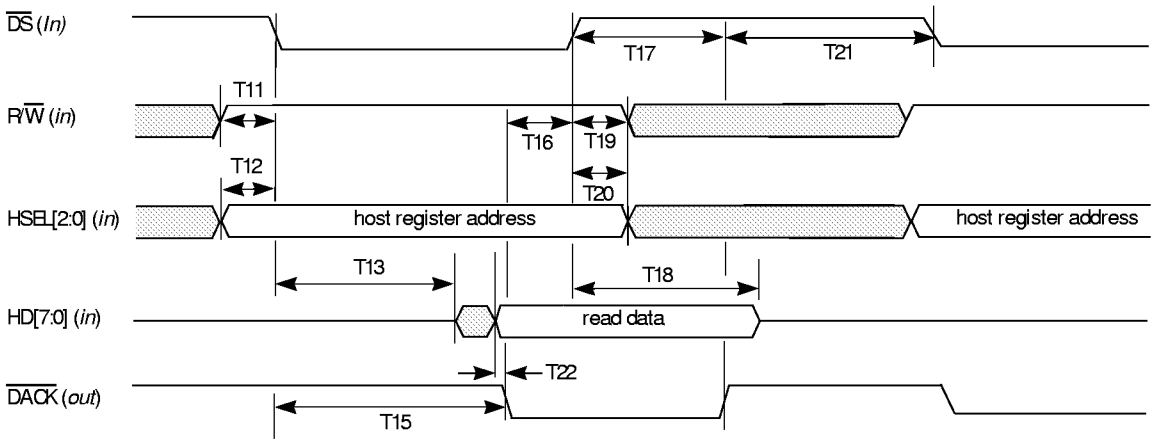


Figure 9-5 Host Interface Local Register, Read Operation

Table 9-6 Timing Characteristics - Local Register Read

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

1. Not 100% tested, guaranteed by design.
2. (GCK * 2) when A_MSB and D_LSB read; (GCK * 6) when D_MSB read
3. GCK * 3) when A_MSB and D_LSB read; (GCK * 6) + GBUS waiting time when D_MSB read
4. Note: A3 silicon is -10 ns.

9.2.3 DRAM/ROM Bus Timing

Local DRAM/ROM Bus Timing is shown next.

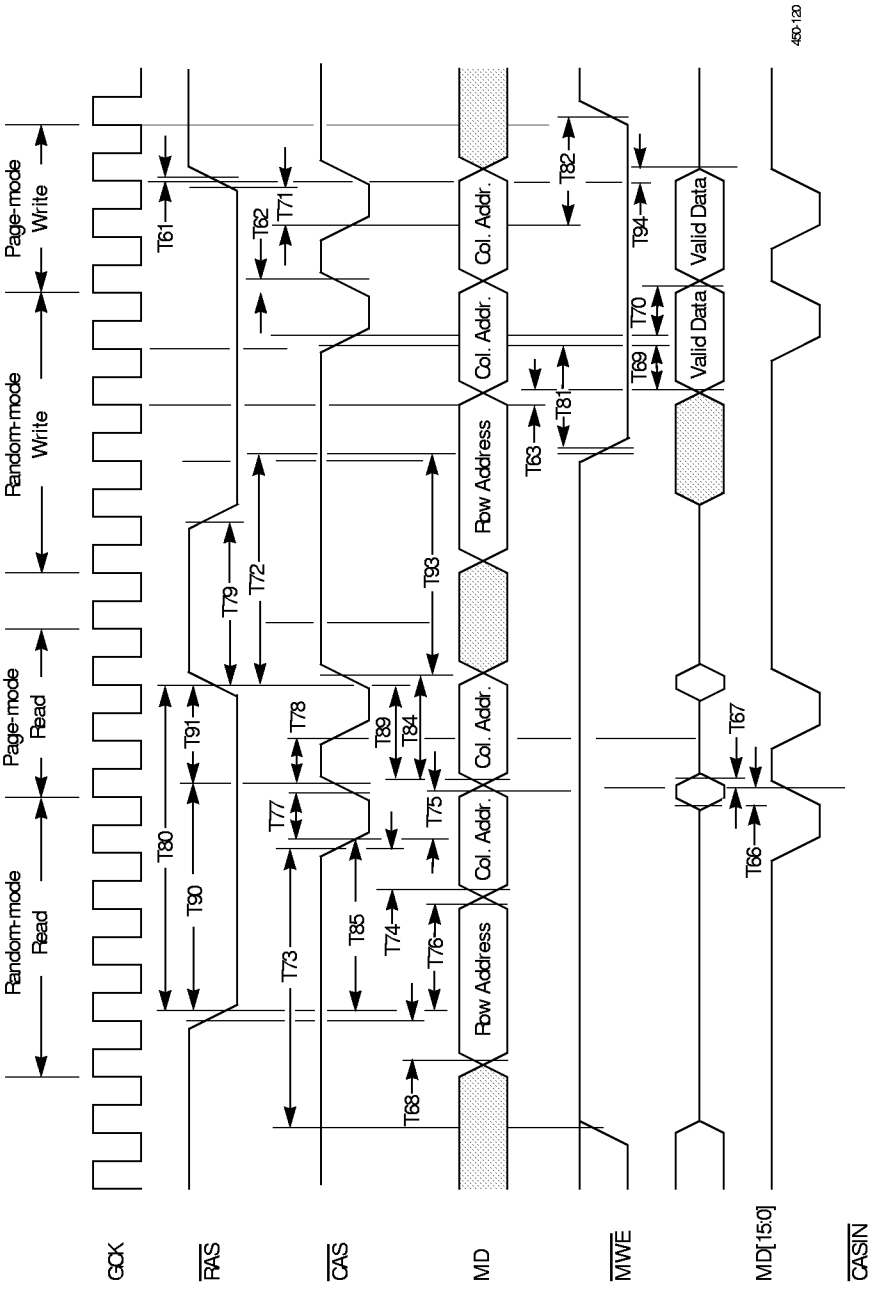


Figure 9-6 Local DRAM Bus Timing

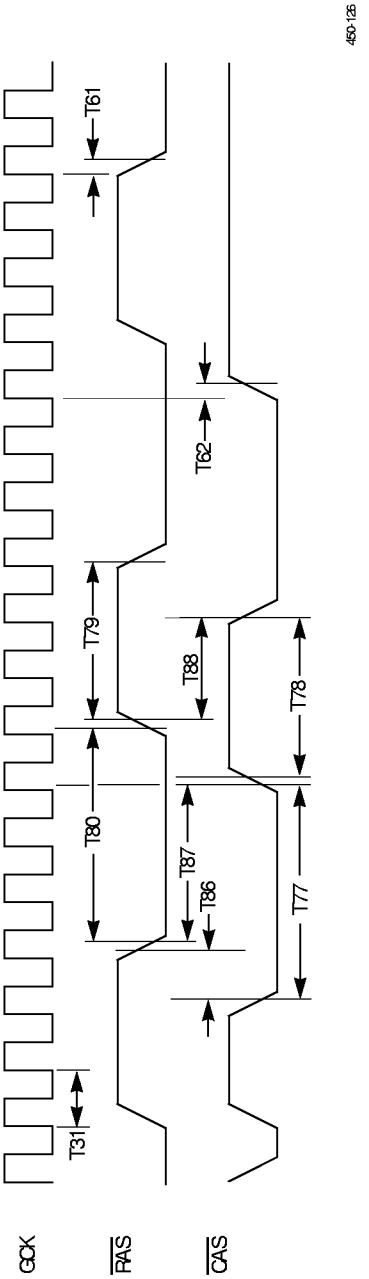


Figure 9-7 Local DRAM CAS-before-RAS Refresh

Table 9-7 Timing Characteristics - Local DRAM Bus[1, 2]

Time	Description	Parameter	Min	Max	Units
T66	Read Data Setup Time before $\overline{CAS}$ IN HIGH		5		ns
T67	Read Data Hold Time after $\overline{CAS}$ IN HIGH		5		ns
T68	Row Address Setup Time[2]	t_{ASR}	5		ns
T69	Write Data Setup Time before $\overline{CAS}$ LOW[2]	t_{DS}	5		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 $\overline{RAS}$[2]	t_{RRH}	2		ns
T73	Read Command Setup Time to $\overline{CAS}$ LOW[2]	t_{RCS}	15		ns
T74	Column Address Setup Time to $\overline{CAS}$ LOW[2]	t_{ASC}	5		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}	21		ns
T78	$\overline{CAS}$ HIGH Time[2]	t_{CP}	10		ns
T79	$\overline{RAS}$ HIGH Time[2]	t_{RP}	62		ns
T80	$\overline{RAS}$ LOW Time[2]	t_{RAS}	80	5000	ns
T81	Write Setup time to $\overline{CAS}$		10		ns
T82	Write Command Hold Time from $\overline{CAS}$ LOW[2]	t_{WCH}	15		ns
T84	Column Address to $\overline{CAS}$ HIGH[2]	t_{AA} + T66	45		ns
T85	$\overline{RAS}$ to $\overline{CAS}$ delay[2]	t_{RCD}	50		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}	47		ns
T93	Read Command Hold Time from $\overline{CAS}$	t_{RRH}+t_{RCH}	0		ns
T94	GCK to MD three-state (write)		15	40	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.

9.2.4 CD Interface Timing

Figures 9-8 and 9-9 and Table 9-8 show the timing for each of the six different CD signal input formats.

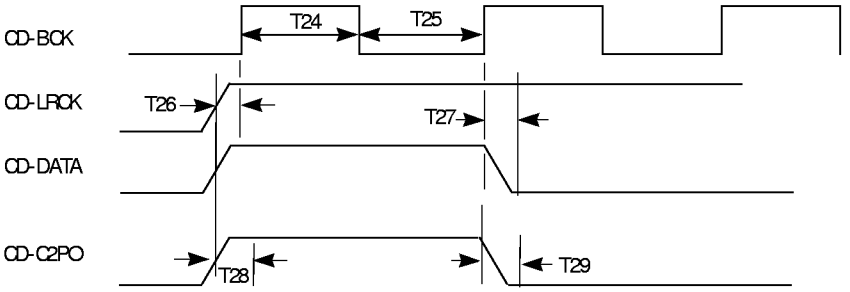


Figure 9-8 CD Input: Data Latch Timing High

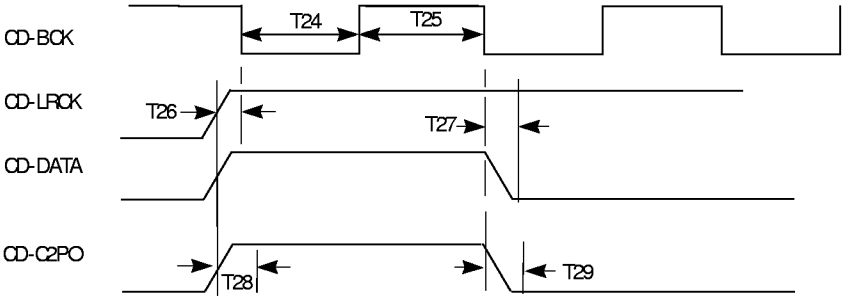


Figure 9-9 CD Input: Data Latch Timing Low

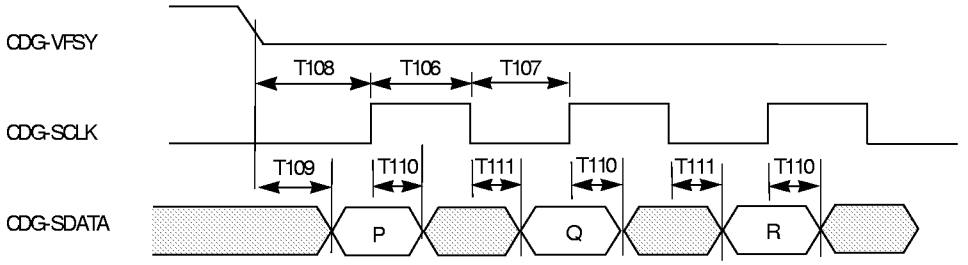
Table 9-8 CD Input Timing[1]

Time	Description	Min	Max	Units
T24	CD-BCK High Pulse Width	(3 GCKs)		
T25	CD-BCK Low Pulse Width	(3 GCKs)		
T26	CD-DATA, CD-LRCK setup	10		ns
T27	CD-DATA, CD-LRCK hold	10		ns
T28	CD-C2PO setup	10		ns
T29	CD-C2PO hold	10		ns

1. Not 100% tested, guaranteed by design.

9.2.5 CD Subcode (CD+G) Interface Timing

Timing for the four CD+G signal pins is shown in Figures 9-10 to 9-12 and Table 9-9. CD+G functions are available only in Silicon Revision C and later silicon. In all cases, new bits of subcode are clocked in on each rising edge of CDG-SCLK as Figure 9-10 shows.



Note: CDG-SCLK must remain LOW between clocking in the "W" subcode bit and the next falling CDG-VFSY edge.

Figure 9-10 Clocking CD Subcode Data

The CL480 supports two different subcode control signal formats. One format uses separate block and frame synchronization signals (see Figure 9-11), and the other format uses a composite frame/block signal as Figure 9-12 shows.

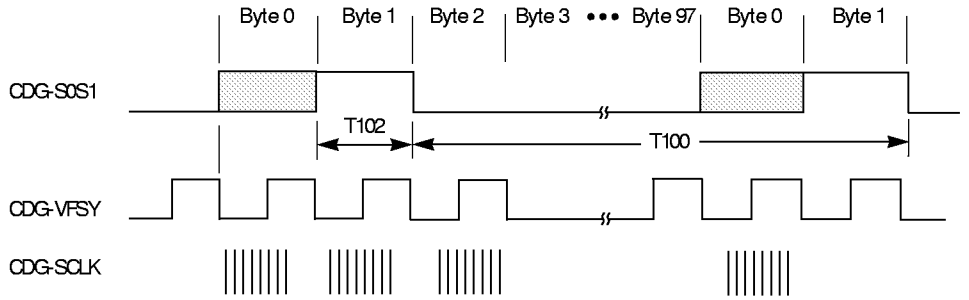


Figure 9-11 Separate CD Subcode Control Signals

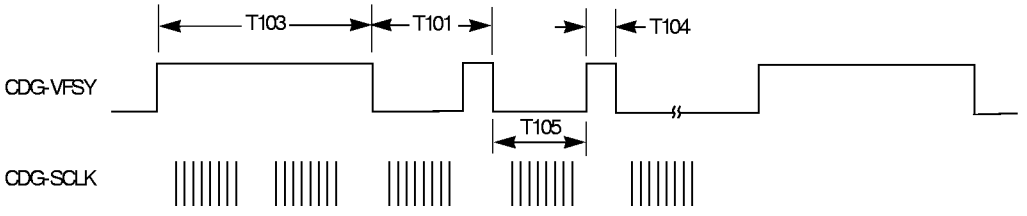


Figure 9-12 Composite CD Subcode Control Signals

Table 9-9 Timing Characteristics: CD+G (Preliminary)

Time[1]	Description	Min	Max	Units
T100	Block period	12.0	14.7	ms
T101	Frame period	122	150	µs
T102	CDG-S0S1 HIGH pulse width	122	800	µs
T103	Composite VFSY HIGH pulse width at start of block	244	800	µs
T104	CDG-VFSY HIGH pulse width	4		µs
T105	CDG-VFSY LOW pulse width	1.5		µs
T106	CDG-SCLK (input) HIGH pulse width	2	6	µs
T107	CDG-SCLK (input) LOW pulse width	2	6	µs
T108	Delay time from CDG-VFSY LOW to CDG-SCLK HIGH edge for "P" subcode bit (CDG-SCLK input)	10	30	µs
T109	Subcode "P" bit access time from CDG-VFSY LOW edge		10	µs
T110	CDG-SDATA hold time from CDG-SCLK HIGH	0		ns
T111	CDG-SDATA access time from CDG-SCLK LOW		80	ns

1. Not 100% tested, guaranteed by design.

9.2.6 Video Bus Timing

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

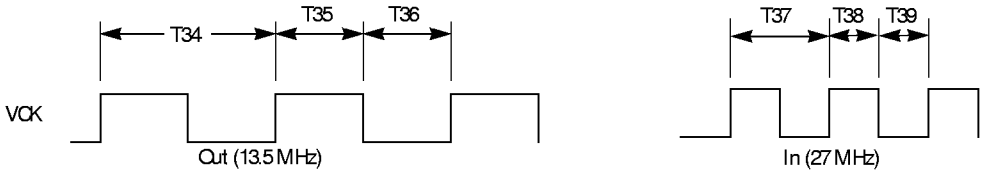


Figure 9-13 Timing - VCK (In and Out)

Table 9-10 Timing Characteristics: VCK

Time[1]	Description	Min	Max	Units
T34	VCK (Out) Frequency (13.5 MHz)	3 GCK	3 GCK	
T35	VCK (Out) HIGH	33	42	ns
T36	VCK (Out) LOW	33	42	ns
T37	VCK (In) Frequency (27 MHz)	26.8 GCK	27.2 GCK	MHz
T38	VCK (In) HIGH	16		ns
T39	VCK (In) LOW	16		ns

1. Not 100% tested, guaranteed by design.

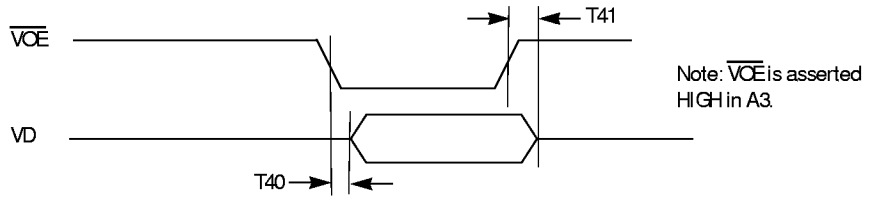


Figure 9-14 Timing - $\overline{VOE}$ and VD

Table 9-11 Timing Characteristics: $\overline{VOE}$ and VD

Time[1]	Description	Min	Max	Units
T40	$\overline{VOE}$ LOW to VD non-tristate	3	25	ns
T41	$\overline{VOE}$ HIGH to VD tristate	3	25	ns

1. Not 100% tested, guaranteed by design.

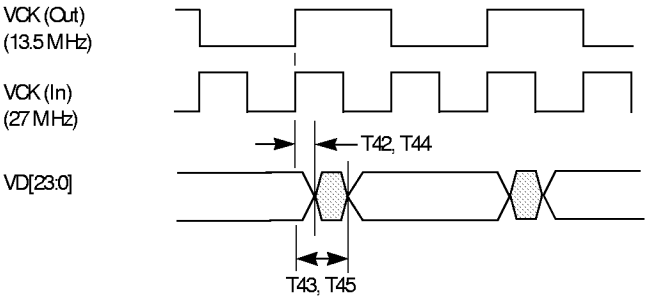


Figure 9-15 Timing - VCK and VD (pixel data)

Table 9-12 Timing Characteristics: VCK and VD

Time[1]	Description	Min	Max	Units
T42	601: VCK (In) HIGH to VD invalid (VD hold after VCK)	3		ns
T43	601: VCK (In) HIGH to VD valid	7	32	ns
T44	601: VCK (Out) HIGH to VD invalid (VD hold after VCK)	-10		ns
T45	601: VCK (Out) HIGH to VD valid	-5	15	ns

1. Not 100% tested, guaranteed by design.

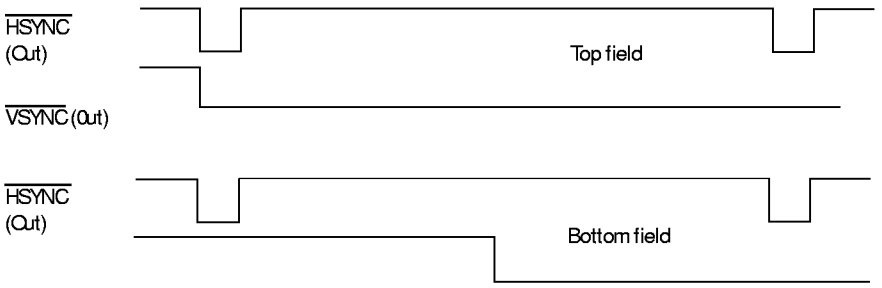
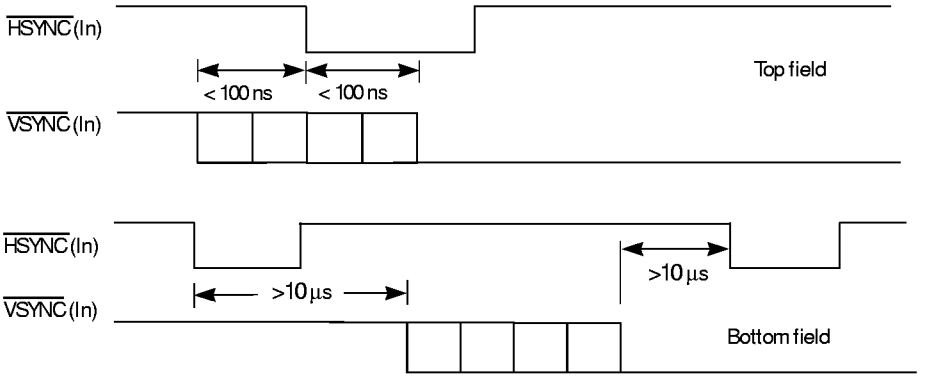


Figure 9-16 VSYNC and HSYNC Out

Note: When HSYNC and VSYNC are outputs (Figure 9-16), the position of the VSYNC edges in the HSYNC period is completely programmable by writing to DRAM locations.



Note: *Top field* means the field that starts with the first line from the top of the screen, while *bottom field* begins with the second line from the top of the screen.

Figure 9-17 VSYNC and HSYNC In

Note: HSYNC must always be synchronous to VCK in HSYNC in mode and is produced synchronously for HSYNC out.

For VSYNC/HSYNC In, the minimum HSYNC LOW time is one VCK, and the minimum VSYNC LOW time is two VCKs.

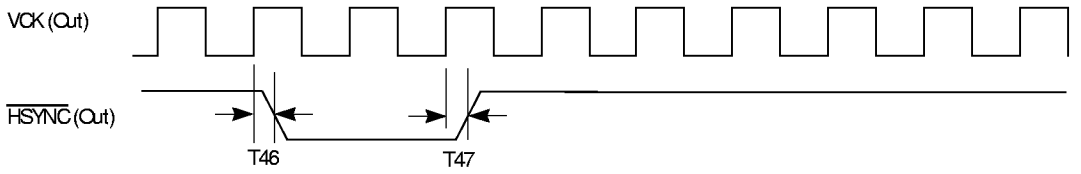


Figure 9-18 VCK Out to HSYNC Out

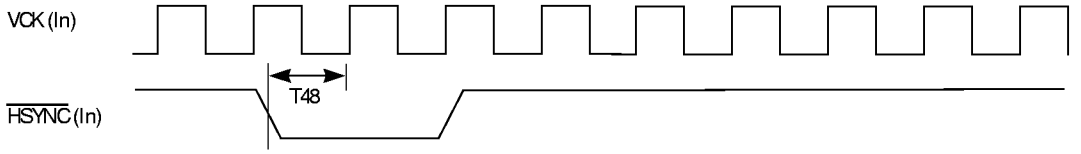


Figure 9-19 HSYNC In to VCK In

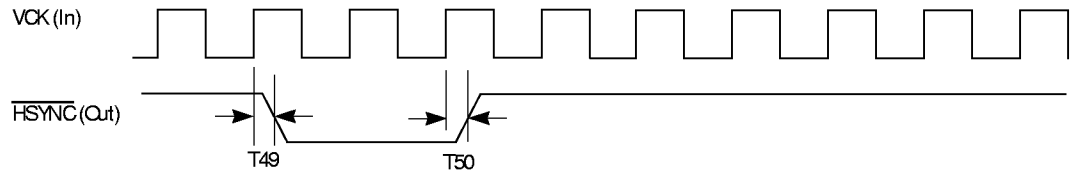


Figure 9-20 VCK In to HSYNC Out

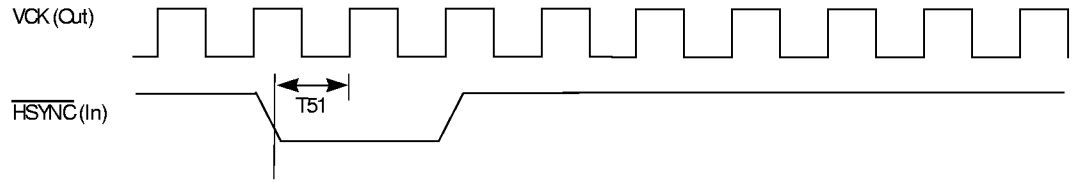


Figure 9-21 HSYNC In to VCK Out

Table 9-13 Timing Characteristics: VCK and $\overline{\text{HSYNC}}$

Time[1]	Description	Min	Max	Units
T46	VCK (Out) HIGH to $\overline{\text{HSYNC}}$ (Out) LOW	-5	10	ns
T47	VCK (Out) HIGH to $\overline{\text{HSYNC}}$ (Out) HIGH	-5	10	ns
T48	$\overline{\text{HSYNC}}$ (In) LOW to VCK (In) HIGH	15		ns
T49	VCK (In) HIGH to $\overline{\text{HSYNC}}$ (Out) LOW	3	30	ns
T50	VCK (In) HIGH to $\overline{\text{HSYNC}}$ (Out) HIGH	3	30	ns
T51	$\overline{\text{HSYNC}}$ (In) LOW to VCK (Out) HIGH	22		ns

1. Not 100% tested, guaranteed by design.

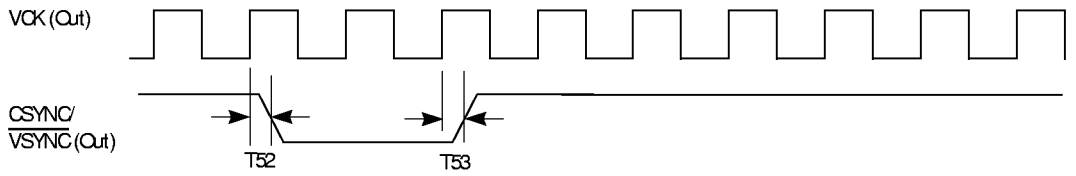


Figure 9-22 VCK Out to CSYNC/$\overline{\text{VSYNC}}$ Out

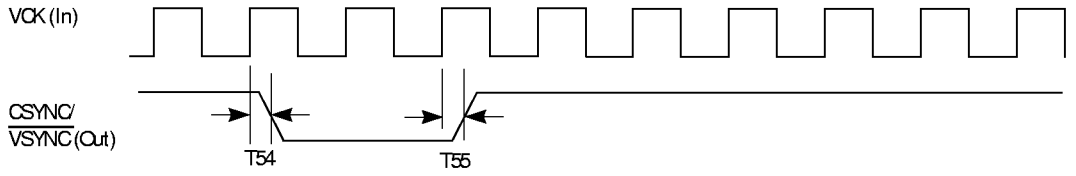


Figure 9-23 VCK In to CSYNC/$\overline{\text{VSYNC}}$ Out

Table 9-14 Timing Characteristics: VCK and CSYNC/$\overline{\text{VSYNC}}$

Time[1]	Description	Min	Max	Units
T52	VCK (Out) HIGH to CSYNC/$\overline{\text{VSYNC}}$ (Out) LOW	-5	10	ns
T53	VCK (Out) HIGH to CSYNC/$\overline{\text{VSYNC}}$ (Out) HIGH	-5	10	ns
T54	VCK (In) HIGH to CSYNC/$\overline{\text{VSYNC}}$ (Out) LOW	3	30	ns
T55	VCK (In) HIGH to CSYNC/$\overline{\text{VSYNC}}$ (Out) HIGH	3	30	ns

1. Not 100% tested, guaranteed by design.

9.2.7 Audio Bus Timing

Figure 9-24 shows the timing for the audio interface of the CL480.

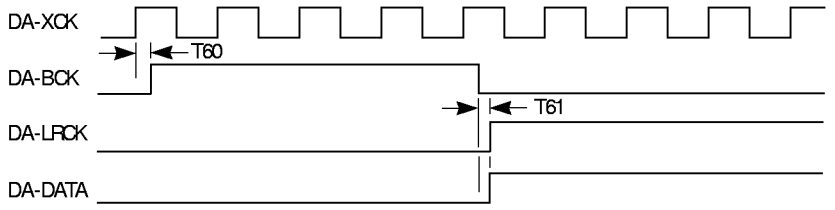


Figure 9-24 Audio Interface Timing Modes

The DA-BCK output pin of the audio bus is synchronized to the rising edge of the input signal DA-XCK, and it has a delay of T60 as Table 9-15 shows. The DA-LRCK and DA-DATA output pins are synchronized to the falling edge of the DA-BCK output signal and have a delay of T61, which is also shown in Table 9-15. The signal DA-XCK is independent of GCK, so it does not have a specified set and hold time.

Table 9-15 Timing Characteristics: Audio

Time[1]	Description	Min.	Max	Units
T60	Delay from DA-XCK to DA-BCK,	3	30	ns
T61	Delay from DA-BCK to DA-LRCK and DA-DATA	3	30	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 information on the following:

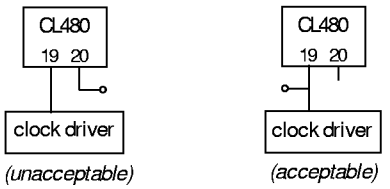
- The CL480 Pinout Diagram
- Tables of CL480 Pin Connections
  - Host Bus Interface Pins
  - CD Interface Pins
  - Global Interface Pins
  - DRAM/ROM Bus Interface Pins
  - Audio Bus Interface Pins
  - Video Bus Interface Pins
  - Power and Miscellaneous Pins
- Package Physical Dimensions

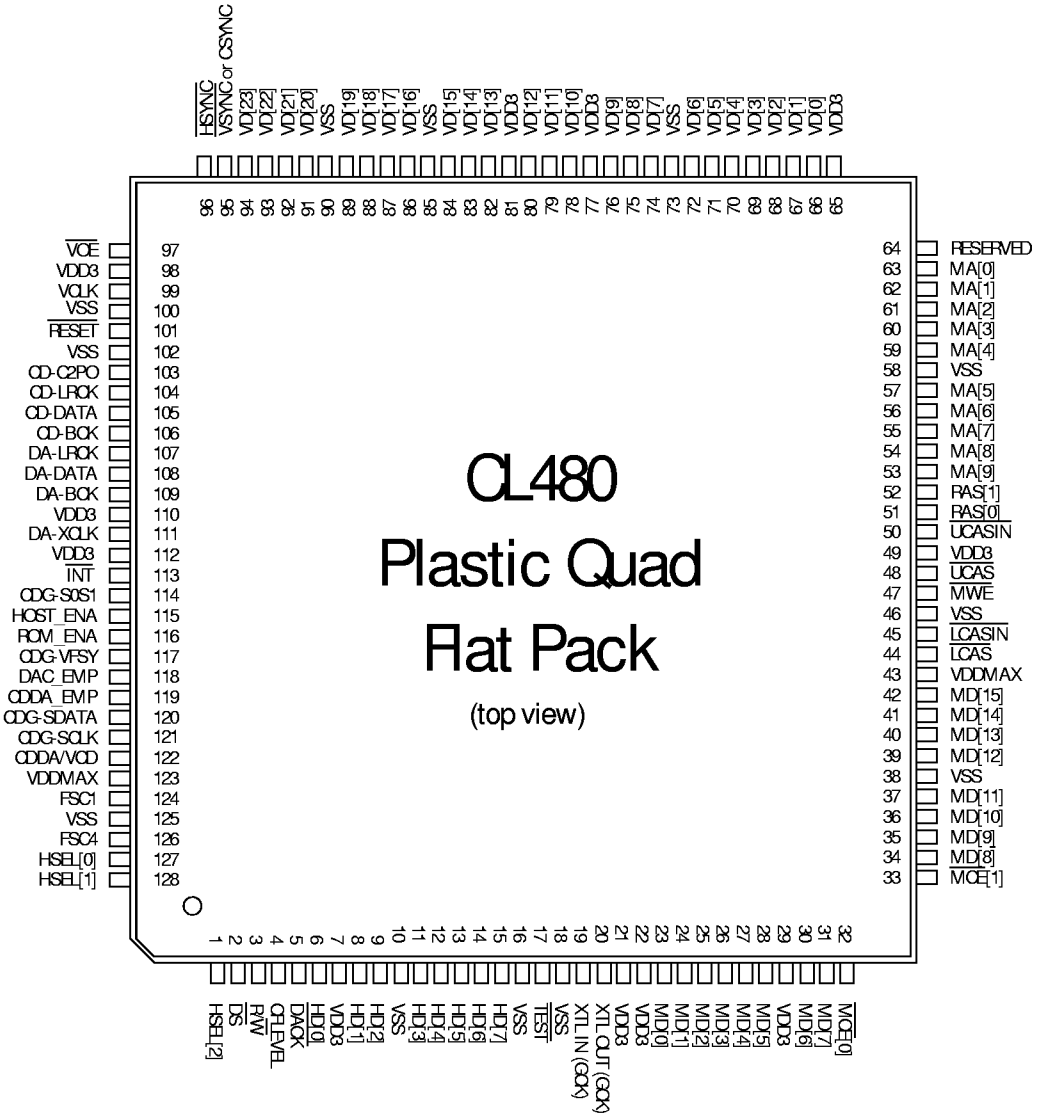
The CL480 is shipped in a drypack with desiccant and a humidity monitor. Do not use the parts if the humidity indicator indicates that the humidity is more than 30 percent at the initial opening of the drypack. To avoid cracking the plastic during soldering, the parts should be soldered within three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

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

The 40-MHz (or 40.5-MHz, if VCK is an output of the CL480) clock can be derived from either a crystal connected across XTL IN (pin 19) and XTL OUT (pin 20), or from a clock driver connected to XTL IN (pin 19).

If a clock driver is used, the signal amplitude at pin 19 must be in the range 0.8(V_{DD3}) to V_{DD3} + 0.2V, and pin 20 should not be connected. That is, pin 20 should be connected to a pad on the board, but there should be no extra traces coming from that pad. If a test point is desired, it should come from XTL IN (pin 19) as shown in the diagram below:





Note: All VDD3 pins should be 2.7 volts to 3.6 volts.

Note: If an oscillator is used, it should be connected to pin 19 (XTL IN) with a minimum amplitude of $0.8V_{DD3}$ and a maximum amplitude of $V_{DD3} + 0.2V$.

Figure 9-25 CL480 PQFP Pinout Diagram

9.3.1 Pin List

Table 9-16 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
CRLEVEL[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	$\overline{DACK}$[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 Silicon A3.

Table 9-17 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	CDG-SDATA	CD	I	120
CDG-S0S1	CD	I	114	CDG-SCLK	CD	I/O	121
CDG-VFSY	CD	I	117				

Table 9-18 Global Interface Pins

Function	Group	I/O	Pin #	Function	Group	I/O	Pin #
XTL IN (GCK)	Global	I	19	XTL OUT (GCK)	Global	O	20
$\overline{RESET}$	Global	I	101	$\overline{TEST}$[1]	Global	I	17

1. Requires a pullup resistor of 1.5K ohms.

Table 9-19 DRAM/ROM Bus Interface Pins

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

Table 9-20 Audio Bus Interface Pins

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

Table 9-21 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/O	96
VD12	Video	O	80	$\overline{\text{VCE}}$[1]	Video	I	97
VD11	Video	O	79	VCK	Video	I/O	99
VD10	Video	O	78	$\overline{\text{VSYNC}}$/ CSYNC	Video	I/O	95
FSC1	Color	O	124	FSC4	Color	I	126

1. Note: In Silicon A3, this signal is asserted HIGH (i.e., VCE, not $\overline{\text{VCE}}$).

Table 9-22 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	XTLOUT	Global	O	20
XTLIN[2]	Global	I	19				

1. VDDMAX is the maximum of the DRAM and host interface input voltages.
2. The maximum input voltage for XTLIN is 4.0 V. This signal has a minimum amplitude of 0.8VDD3 and a maximum amplitude of VDD3 + 0.2V.

9.3.2 Package Drawings

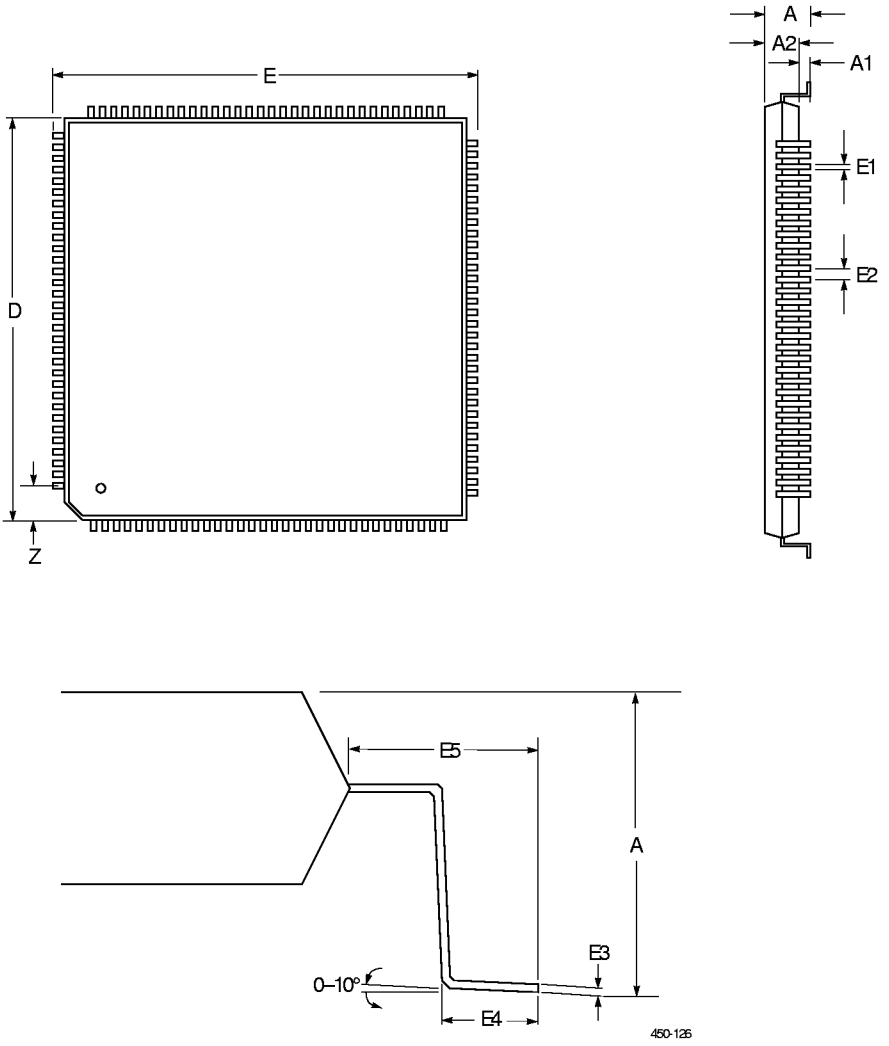


Figure 9-26 Plastic Quad Flat Pack Physical Dimensions

Table 9-23 Plastic Quad Flat Pack Physical Dimensions

SYMBOL	DIMENSIONS MM
A	3.7 MAX.[1]
A1	0 MIN.
A2	3.5 MAX.[2]
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.

1. With the new, thinner package design, this parameter has been reduced to 1.7 mm.
2. With the new, thinner package design, this parameter has been reduced to 1.4 mm.

10
Registers

This chapter describes the CL480 registers. You should be familiar with the relationship of these registers to the CL480's external signals as Chapter 3 describes. Also, be aware that *the functions described by these registers are initialized by microcode as outlined in Chapter 12.*

The CL480 registers listed below are used routinely to configure and communicate with the CL480 and its microcode, although not all applications use all of these registers:

- HOST_int
- CPU_cntl
- CPU_pc
- CPU_iaddr
- CPU_imem

These registers can be accessed using a single pair of accesses to D_MSB and D_LSB. 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] used to select the register.

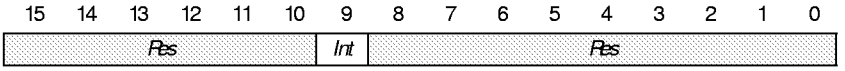
The sections that follow describe the registers contained in each CL480 interface module. In the detailed definitions given for each register, note that bits marked *Res* are reserved. They will return a 0 or 1 when read, but should always be written to 0 (except for HOST_int).

10.1 Internal Host 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.

(normal) **0x0F**

HOST_int Register



Int **Interrupt (bit 9)** **R/W**

This bit controls the external interrupt signal, $\overline{\text{INT}}$. When *Int* is 1, the CL480 will assert $\overline{\text{INT}}$, driving it LOW. When *Int* is 0, the CL480 does not drive $\overline{\text{INT}}$, which will be pulled HIGH (inactive) by an external pullup resistor.

Note: If you want to change the Int bit: (1) read out the entire register, (2) AND the value that you want to write with 0xFDFF, and then (3) write back this register. (This procedure is for the HOST_int Register only.)

10.2 Internal CPU Registers

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

10.2.1 CPU Execution Registers

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

(normal) **0x33** **CPU_cntl Register**

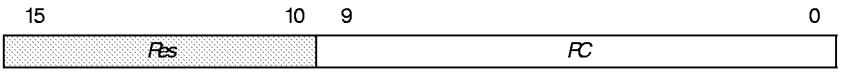
LS **Local State (bits 15, 14)** **R/W**

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.

(normal) **0x3A** **CPU_pc Register**



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.

10.2.2 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 10-1. The circled numbers in the figure refer to the steps that follow.

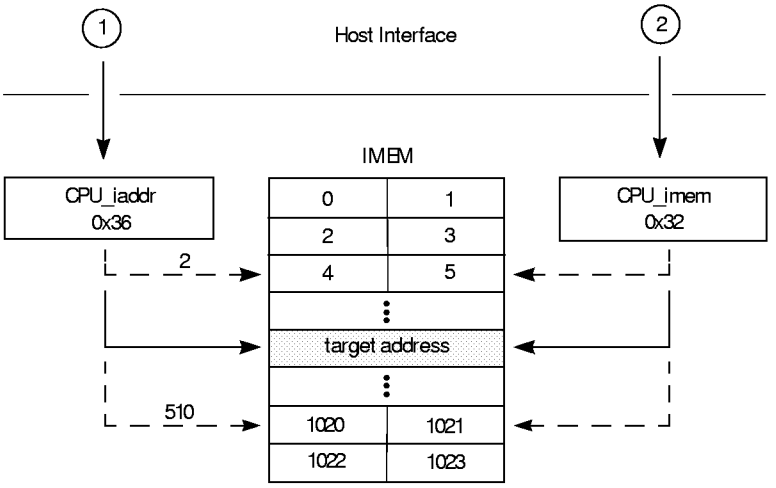


Figure 10-1 IMEM Write Data Flow

The sequence of events for writing to the IMEM is as follows:

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**

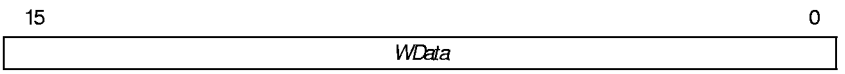
15	10	9	8	1	0
Res			*WAdd*		

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 (that is, after two 16-bit writes).

CPU_imem Register

(normal) **0x32**



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.

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 in one of two forms: microcode which boots from ROM (CL480VCD), and microcode which boots through the host interface and does not require ROM (CL480PC).

Both types of microcode operate on both the Silicon Revision A3 CL480 chip and the B0, or greater, CL480 chips. Additional features are available when using the B0, or greater, CL480 chips. If a feature not available to the A3 chip is requested, the CL480 microcode will ignore the request.

The CL480VCD microcode is intended to be loaded from ROM, while the CL480PC microcode is loaded from the boot disk. The CL480VCD microcode doesn't support use of the host interface for inputting bitstreams, while the CD-ROM interface is disabled in the CL480PC microcode. Additional information on loading and executing this code is contained in this chapter, the next chapter, and on the distribution disk that is provided with the CL480.

11.1 Microcode Process States

The operation of the CL480 microcode and the host processor can be described as a set of "microcode process states"—each a named group of software processes, some executing within the host processor and some executing within the CL480's microcode. Six different process states occur at different times in a CL480 system. These process states are Microcode-Load, Initialization, Command, Bitstream Input, Decode, and Output.

11.1.1 Microcode-Load Process

The CL480 microcode must be loaded at startup in all systems. In CL480VCD systems, the microcode is loaded from ROM automatically following a pulse on the $\overline{\text{RESET}}$ pin. In CL480PC systems, the host must explicitly load the executables into the CL480 IMEM and DRAM (see Appendix B).

11.1.2 Initialization Process

During initialization, the CL480 initializes all hardware interface areas according to the values stored in the microcode executable and, subsequently, DRAM. (See Chapter 12 for an extended explanation of initialization.) The method by which the microcode receives direction is described next.

11.1.3 Command Process

Following initialization of the microcode, CL480 operations are controlled by the host via the following two methods:

- Changing configuration parameters by writing new values into the Configuration Area of the DRAM
- Issuing macro commands that consist of a function code and parameter values (see Table 11-1)

Command and operating state information can be read from the Status Area of DRAM.

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 Area in DRAM (see Table 15-2).

Table 11-1 Macro Command Summary

Priority	Name	Description	Cmd. ID	Page
	DisplayBorderColor()	Sets active display region to border color	001D	142
	DisplayDigest()	Decodes a single I-picture digest display	021B	143
	DisplayDigest(start)[1]	Same as above, but specifies starting point	041B	144
	DisplayGraphics()[1]	Controls graphics overlay on screen	0417	146
	DisplayStill()[2]	Decodes/displays multiple still pictures w/audio	000C	148
	DisplayStill(start)[1]	Same as above but specifies starting point	020C	150
	DisplayStill(start,stop)[1]	Decodes/displays multiple still pictures w/audio	040C	152
	DisplayStill(romAddress)[1]	Same as DisplayStill(), but specifies ROM address	030C	151
	DumpData()[1]	Reads and performs ECC on CD-ROM data	0414	154
Low	Freeze()	Freezes display, but continues audio decoding	0010	156
	Pause()	Freezes display and decoding process	000E	158
	Play()	Decodes and displays at normal rate	000D	159
	Play(start, stop)[1]	Same as above, but specifies start/stop	040D	161
	Replay()	Restarts the previous Play() command	001C	163
	Scan()	Decodes and displays next single I-picture	000A	165
	SetStreams()	Selects which system streams to decode	0213	169
	SetVideoFormat()	Sets format to NTSC, PAL, or progressive	0305	170
	SingleStep()	Decodes and stores next single picture	000B	171
	SlowMotion()	Decodes and displays at slower rate	0109	172
	Abort()	Sets processing state to IDLE	801A	141
High	InquireBufferEmptiness()	Measures data in bitstream buffer	8003	157
	Reset()	Reinitializes CL480 and its microcode	8000	164

1. For CL480VCD microcode version only
2. Allows display of a high-resolution still image with double horizontal and vertical resolution as specified in VideoCD 2.0.

Note: If a macro command is issued and the Command ID 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 CL480PC accepts MPEG-1 system streams from the host interface, while the CL480VCD accepts data streams from the CD interface containing:

- CD-ROM data (CDROM Mode 1 or CDROM Mode 2 Form 1)
- CD-DA data

Besides obtaining input from either the host or serial interface (depending on the microcode version) and auto-determining the type of data, the bitstream input process also includes demultiplexing of the bitstream into the appropriate rate buffer and generating the appropriate interrupts to the host processor.

Host Bitstream Input

Data can be transferred from the host to the CL480 using direct host writes to the CL480's CFIFO, a FIFO with 31, 16-bit locations. The host must check the CFLEVEL pin before writing to ensure that the CFIFO does not overflow.

CD Bitstream Input

Serial data can also be transferred from a CD-DSP to the CL480. For CD-ROM decoding, the CL480 stores the sector header and sub-header information into a dedicated DRAM location called Sector Header and Sector Subheader (see Chapter 15) which is readable by the host. CD bitstream data is then deserialized and written to the CFIFO.

CL480 Internal Bitstream Handling

Once inside the CL480's CFIFO, the data is automatically transferred as a burst to the *bitstream buffers* located in the local DRAM—one each for independent audio and video bitstream buffers. (The bitstream buffer is called the *rate buffer* in the MPEG standard.)

If the stream identifier within the data corresponds with the value of the video stream ID in the VA_STREAM_ID configuration parameter (see Table 15-7), the packet is expected to contain video data. Likewise, if the stream identifier corresponds with the value of the audio stream ID in the VA_STREAM_ID configuration parameter, the packet is expected to contain audio data.

Any valid audio or video stream identifier can be used to route the bitstream to the associated buffer. Any packets having a stream identifier not matching the value contained in the two configuration parameters mentioned above are discarded.

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 in their respective buffers in the CL480's local DRAM.

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 audio DAC.

11.2 Memory Usage

The CL480 microcode requires a minimum of four Mbits or 512K bytes of DRAM to read, decode, and output VideoCD, MPEG-1 system and CD-DA bitstreams.

11.2.1 DRAM Memory Map

The DRAM memory map of the CL480 microcode is shown in Figure 11-1. The CL480 can address a maximum of one Mbyte of DRAM.

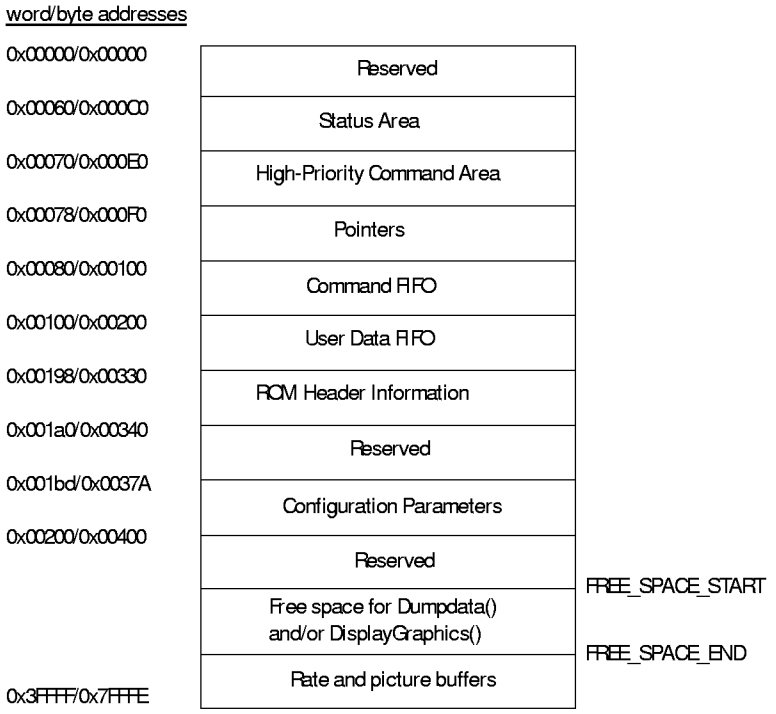


Figure 11-1 DRAM Memory Map

Note: FREE_SPACE_START and FREE_SPACE_END are defined in Table 15-1.

The areas from 0x00060 to 0x00200 are collectively called the DRAM Configuration Reference (where most user-definable parameters are stored), and are defined in greater detail in Chapter 15. The pointers FREE_SPACE_START and FREE_SPACE_END are also provided in the Configuration Area, indicating what portion of the address space from 0x00200 through 0x3FFFF is unused by the CL480 microcode.

11.3 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. (See the ERROR_LEVEL parameter in Table 15-7.)

11.4 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 it wishes to be informed of by using the INT_MASK configuration parameter to enable the generation of any or all of the 15 logical interrupts (see Table 15-7).

When an interrupt occurs, the event that caused the interrupt is signalled in the INT_STATUS word in the DRAM Status Area (see Table 15-1), and $\overline{\text{INT}}$ is asserted (active LOW). For more information on interrupts, see Chapter 14.

11.5 A/V Synchronization

The synchronization process is divided into two parts:

- Updating the STC (system time clock) from timing information provided in the incoming bitstream
- Outputting the video frames according to the presentation timestamp (PTS) included in the video bitstream

The system time clock must be kept in sync with the incoming bitstream to ensure that the PTS values used to present video frames are synchronized with the original, un-encoded video and audio data. Without proper video and audio synchronization, the audio and video information will not be synchronized and the bitstream buffers may overflow or underflow.

The system time clock is initialized by the SCR value found in the first pack header after entering the PLAY or DISPLAYSTILL state. Afterwards, there are two methods of updating the system time clock:

- Updating the STC with SCR values found in subsequent pack headers.
- Updating the STC with the PTS value found in the audio pack header. (The audio PTS value is adjusted based upon the amount of data remaining in the audio output buffer before being written to the STC.)

The method desired is chosen by setting the AV_SYNC_MODE configuration parameter (for CL480PC microcode only) as explained below and in Table 15-7:

- If AV_SYNC_MODE is set to NO_SYNC, updating of the STC is disabled and the PTS values of decoded picture are ignored.

  Note: This mode should only be used for debugging; decoding of bitstreams with synchronization disabled is not recommended.

- If AV_SYNC_MODE is set to SYNC_AUDIO_PTS, the STC is kept in synchronization with the audio PTS values found in the audio packet header.

  Note: This mode is normally used in designs utilizing the host interface for transfer of compressed data (i.e., PC-based designs).

- When AV_SYNC_MODE is set to SYNC_SCR (CL480VCD microcode is always SYNC_SCR), the STC is updated with the SCR value found in every pack header received. If AV_SYNC_MODE contains a value greater than SYNC_SCR, the STC is loaded with the SCR value found in the pack header every AV_SYNC_MODE pack header.

The presentation timestamps are used by video decoding process for decoding and displaying of each frame. The video presentation timestamp is compared to the current system time clock value. If the presentation timestamp differs from the system time clock by more than the allowed tolerance, the display rate is adjusted. If the display is behind, B-pictures will be skipped until it is caught up. If the display is ahead, decoding will

be held and pictures will be repeated until the current timestamp match occurs. The conditions for adjustment are:

Presentation Timestamp + INTERVAL < System Time Clock

Presentation Timestamp - INTERVAL > System Time Clock

If the first condition is false, pictures will be repeated; if the second condition is false, pictures will be skipped. The INTERVAL parameter is the number of 90-kHz clocks per video frame and is defined in Table 15-1.

11.6 Detecting Input Bitstream Type

When the input bitstream is either a CD-DA or CD-ROM bitstream, the CL480VCD microcode can enabled to either auto-detect the input bitstream or expect bitstreams of one type only.

11.6.1 Auto-Detecting the Input Bitstream

This type of bitstream detection is achieved by setting the PLAY_MODE configuration parameter equal to CD_ROM_AUTO. The CL480VCD microcode then uses the presence or absence of CD-ROM synchronization characters to determine whether the input bitstream is either a CD-ROM or CDDA bitstream, sync characters being present in CD-ROM bitstreams.

Before decoding begins and during execution of the Play() command, the CL480 microcode searches for synchronization characters for two seconds. The CL480VCD microcode continues to examine the incoming bitstream for the presence of CD-ROM sector synchronization characters, and switches between the two bitstream types accordingly. If sector synchronization characters are not detected, the CL480 switches to processing CD-DA data; otherwise, if sector synchronization characters are detected, the CL480 switches to processing CD-ROM data.

11.6.2 Setting Detection to One Bitstream Type Only

The CL480VCD microcode is configured to expect bitstreams of one type only by setting the PLAY_MODE variable to one of two field locations:

- CD_ROM_MPEG: The incoming bitstream is expected to contain CD-ROM sector information.
- CDDA: The incoming bitstream is expected to contain CDDA data.

Setting the PLAY_MODE variable to any of the above values disables the auto-detecting algorithm described in Section 11.6.1.

Since the CL480PC microcode does not support CD-ROM or CDDA bitstreams, the PLAY_MODE parameter is not examined, and the microcode operates as if the PLAY_MODE parameter was set to HOST_MPEG_SYSTEM.

11.7 Switching Between CD-DA and CD-ROM Bitstreams

Normally, the CL480VCD microcode will only accept CD-DA bitstreams when the PLAY_MODE configuration parameter is set to CD-DA. However, the CL480VCD microcode can also read and output CD-DA bitstreams while in the PLAY state when the PLAY_MODE configuration parameter is set to CD_ROM_AUTO.

When the PLAY_MODE configuration parameter is set to CD_ROM_AUTO, the auto-bitstream detecting procedure described in Section 11.6.1 is executed when the Play() command is executed.

Switching from an MPEG-1 bitstream to a CD-DA bitstream is performed when all of the following conditions are met in the following order:

1. The CL480VCD microcode is in the PLAY state.
2. The PLAY_MODE configuration parameter is set to CD_ROM_AUTO.
3. CD-ROM sector synchronization characters have not been detected for two seconds.

When the CL480VCD microcode switches from processing a CD-ROM bitstream to a CD-DA bitstream, the picture displayed at the time of the switch is displayed while the CD-DA bitstream is processed. Any video overlay graphic displayed at the time of the switch will continue to be displayed.

Switching from a CD-DA bitstream to a CD-ROM bitstream requires the same conditions to be met as for switching from CD-ROM to CD-DA, except when synchronization characters are detected. The CL480VCD microcode switches to processing the CD-ROM bitstream when two sector synchronization characters are detected, indicating the start of two consecutive CD-ROM sectors. The video and audio bitstream buffers and the audio output FIFO are flushed when switching to the CD-ROM bitstream. The decoding process is also initialized.

11.8 Dump Data Feature

Using the DumpData feature, CD-ROM sector information can be read, error-corrected and copied into a user specified buffer. The DumpData feature is accessed using the DumpData() command described in Chapter 13. CD-ROM Mode 1 and CD-ROM/XA Mode 2 Form 1 data sectors are the only sectors types supported. The raw sector data is copied into the buffer specified. Once a sector is copied, error detection and correction is performed, and any uncorrected sectors are indicated in SE_STATUS (see Chapter 15). The error correction information is removed, leaving 2048 bytes of data. If more than one sector is read, the data for subsequent sector information follows immediately after the previous sector.

Four error conditions may occur while reading the sector information. The conditions causing these errors are as follows:

- The buffer address argument of the DumpData() command specifies an address which will overwrite reserved areas of DRAM.
- The number of sectors to read, correct, and copy is zero or greater than 16.
- While searching for the start address, an address greater than the start address was detected. The AOR interrupt is not generated in this case.
- A sector type other than Mode 1 or Mode 2 Form 1 is specified as an argument to the DumpData() command or was detected while reading the number of sectors requested. The "data" bit of the submode field of a Mode 2 Form 1 sector was not set.

Any of the above errors cause the DumpData() command to terminate with the MRC_STATUS field indicating the reason for the abnormal termination (see Chapter 15 for further details). The MRC_STATUS field should be examined when the DumpData() command completes. If the MRC_STATUS field equals DONE_STATUS, the DumpData() command finished successfully. In all cases, and if enabled, the END-D interrupt is generated upon completion of the DumpData() command. The MRC_STATUS field equals WORKING_STATUS while sector data is processed.

> *Note: The buffer used to hold the sector data must be twice as large as a raw sector to hold the raw data and error information. The resultant data size is 2048 times the number of sectors processed.*

11.9 Auto-Pause

The CL480VCD microcode uses the trigger bit of the CD-ROM/XA submode field, (bit 4), to implement an auto-pause feature. When this bit in the submode field is set, the CL480VCD microcode will automatically enter the PAUSE state when the following conditions are met:

- The CL480VCD microcode's processing state is SINGLESTEP, SLOWMOTION, DISPLAYSTILL, or PLAY.
- Auto-Pause is enabled using the AUTO_PAUSE configuration parameter (see Table 15-7).
- A CD-ROM/XA sector with the trigger bit set is detected.
- The last video picture copied, or the current video picture being copied into the video bitstream buffer when detection of the trigger bit being set occurs, is displayed.

Detection of the trigger bit being set can also cause an interrupt to occur. The A/E/E interrupt will be generated when the trigger bit is set and if the A/E/E interrupt is enabled. The interrupt is generated when the CL480VCD microcode reads the sector address from the CD-ROM Decoder. The interrupt is generated in all states except IDLE.

The occurrence of the Auto-Pause sector will be reset when entering the following states or when an error occurs in the input or decoding process: IDLE, PLAY, DISPLAYSTILL, SCAN, FREEZE or DISPLAYDIGEST.

11.10 Bitstream Buffer Underflow or 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. Buffer underflow is signalled to the host through an interrupt.

11.10.1 Underflow

If video data underflow occurs during the decoding of a B-picture, a "tear" may occur on the display. Audio data underflow will cause audible artifacts. Further, video data underflow may cause corruption of the audio output; while audio underflow may stall the video data, leaving an incomplete picture on the display. If underflow occurs, processing of low-priority macro commands will be suspended until more data becomes available. However, depending on the duration, not all underflows will cause the UND interrupt.

11.10.2 Overflow

Bitstream overflow is a condition that is prevented from occurring in the CL480 as described below.

For V-CD Microcode

In a Video-CD environment, the CL480 performs a/v synchronization from the system SCRs, which are delivered to the CL480 at a fixed rate. By performing synchronization from the SCRs, the CL480 discards pictures if the SCR gets ahead of the video PTSs, thus preventing bitstream overflow.

For PC Microcode

In a PC environment, the CL480 uses the audio PTSs as the time master. The CL480 performs video synchronization by comparing incoming video PTSs with the internal SCR counter loaded from audio PTSs. There is no possibility of overflow, since the host must wait for CFLEVEL to be LOW before sending additional data to the bitstream buffer.

11.11 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 in VideoCD 2.0 may have 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 a bitstream 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 one-half or one-quarter second) using the Play() command before going to the next intra picture.

12
Initialization

The host processor executes an initialization sequence for the CL480 by interacting with the microcode (either macro commands or the Configuration Area). For initialization to occur, the CL480 microcode must be loaded in either of the two provided systems:

- ROM-based (CL480VCD) systems, where the microcode is loaded from ROM automatically, following a pulse on the $\overline{\text{RESET}}$ pin.
- Host-based (CL480PC) systems, where the host must explicitly load the microcode into the CL480 internal IMEM (instruction memory), DRAM, and registers.

12.1 Distributed Files

The distribution disk (MS-DOS format 3.5") contains a copy of both the CL480PC host-based microcode in its permanent binary file format (480P.UX) and the 480VCD microcode (480V.HEX). The CL480PC microcode is logically contained in a series of "segments," each a block of numeric constants (constant data or microcode) as further described in Appendix B. The 480VCD microcode is in "Intel MCS-86 Hexadecimal Object File Format" as found in the "*DATA I/O Unisite Programmer, Code 88.*"

Both distribution microcode executables contain default values that determine the protocol used for each of the CL480's interfaces after they are loaded into the CL480's registers at initialization time. These default values are located in DRAM in an area known as the Configuration Area. Thus, the Configuration Area needs to be altered to match your setup requirements.

12.2 CD-ROM-Based (CL480VCD)

The CL480VCD microcode is distributed in Intel HEX file format for loading into a ROM. The minimum size for this ROM is 64K bytes. All configuration parameters are changeable in ROM before the code is loaded.

When configured for ROM operation (ROM_ENA pin pulled HIGH when the CL480 is reset), the CL480 hardware loads the first 1024 instructions from ROM address zero into the CPU's instruction memory and starts execution at address zero of the instruction memory. The CL480 microcode copies that portion of the ROM necessary to provide the basic functionality of the CL480. The CL480VCD microcode may copy additional information from ROM as necessary for the operation being requested.

12.2.1 Modifying the Configuration in ROM

The configuration in ROM should be initialized using the following sequence.

1. Load the ROM programmer with the 480 microcode file provided. (This file is in Intel hex format.)
2. Edit the memory in the ROM programmer. For each memory location whose default value doesn't match your configuration, change the appropriate bits to match your system. (Each of the important memory locations and their default values are defined in Table 12-1 and, more extensively, in Table 15-7.) For instance, the default for the v1.3 and v2.0 microcode is for the CL480 to output CSYNC rather than $\overline{\text{VSYNC}}$. If your system expects $\overline{\text{VSYNC}}$ to be output, then you will want to change the default in the ROM memory.
3. Change all of the appropriate locations in the ROM programmer's memory; then burn the ROM.

While there are many locations in the Configuration Parameter area that can be changed, only the locations described in Table 12-1 will likely require your attention.

Table 12-1 Likely Configuration Changes for CL480VCD Microcode

Parameter Name	Word/byte Addr	Default Value	Description
AUDIO_CONFIG	0x1cb/0x396	0xd130	This location controls the output data format on the audio interface. As described in Table 15-7, field location 0x02 controls either 24 or 16 bits per channel, and location 0x01 controls MSb or LSb first on the data.
CD_CONFIG	0x1cc/0x398	0x0216	This parameter controls the input data format on the CD interface. Field location 0x30 controls the number of bits per channel; location 0x08 sets LSb or MSb first for the data; location 0x04 sets LSb or MSb first on CD-C2PO; location 0x02 selects right versus left channel CD-DATA input; and location 0x01 controls whether the data is latched on the rising or falling CD-BCK edge.
VIDEO_MODE2	0x1cd/0x39A	0x000c	VIDEO_MODE2 field location 0x04 controls whether the CL480 outputs $\overline{\text{VSYNC}}$ or CSYNC.
VIDEO_FORMAT	0x1eb/0x3d6	0x0004	VIDEO_FORMAT field values 0x03 and 0x04 determine PAL versus NTSC output, respectively.
VIDEO_MODE	0x1d7/0x3AE	0x0b24	The VIDEO_MODE field locations 0x0080, 0x0040, 0x0004 and 0x0001, respectively, select the VCK direction, control the video bus width, specify RGB versus YUV mode, and control the $\overline{\text{HSYNC}}$/$\overline{\text{VSYNC}}$ direction.
ROM_CONFIG	0x1ce/0x39C	0x0007	This location controls the access time for the ROM.

12.2.2 Start-Up Process

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

1. Toggle the $\overline{\text{RESET}}$ pin.
2. Poll DRAM location 0x70 (word address) for a zero value. When a zero value occurs, microcode initialization is complete.
3. Modify Configuration Area DRAM locations, if desired (see Table 15-8).
4. Issue commands (Play(), etc.).

12.3 Host-Based (CL480PC)

For the CL480PC host-based microcode, the host system must explicitly load the executable file into either the CL480 internal IMEM (instruction memory) or external DRAM. The CL480PC microcode will reside within 64K bytes. A description of the contents of the CL480PC.UX file is contained in Appendix B. A sample loader is provided on the microcode distribution disk.

12.3.1 Loading .UX Microcode

The CL480PC microcode is logically contained in a series of "sections," each of which is a block of numeric constants (constant data or microcode) that the host must load into the internal IMEM (instruction memory), the external DRAM, or both.

All sections of the CL480PC microcode are distributed in a single microcode executable file. In addition to microcode sections, the executable file also contains an information header, which includes the initial program counter value for the CL480's CPU.

12.3.2 Start-Up Process

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 logic).
2. Run the C-Cube provided loader.
3. Modify DRAM configuration locations.
4. Write 0 to CPU_cntl.
5. Poll DRAM location word address 0x70 for a zero value. When this occurs, microcode initialization is complete.
6. Issue commands (Play(), etc.).

When the microcode is loaded, the DRAM parameter DATE_VERSION (see Table 15-7) may be read for information describing the microcode version. This information is duplicated in the microcode executable file header, but it is loaded into DRAM to ensure that the microcode version present in a running system can be determined.

> *Note: For additional information on the CL480PC microcode, see Appendix B.*

12.4 DRAM Parameters (Alphabetized)

CL480 DRAM parameters are listed alphabetically in Table 12-2.

Table 12-2 CL480 DRAM Parameters

Parameter	Word /byte Addr	Default Value	Change Effective
ABS_EADR	0x1c3/0x386	0x0054	Decode
ABS_SADR	0x1c2/0x384	0x0040	Decode
AEE_FM	0x6b/0xd6		
AEE_MS	0x6a/0xd4		
ARGUMENT1	0x71/0xE2		
ARGUMENT2	0x72/0xE4		
ARGUMENT3	0x73/0xE6		
ARGUMENT4	0x74/0xE8		
ARGUMENT5	0x75/0xEA		
ARGUMENT6	0x76/0xEC		
ARGUMENT7	0x77/0xEE		
AUDIO_AVERAGE	0x1d0/0x3A0	0x0000	Immediate
AUDIO_CONFIG	0x1cb/0x396	0xd130	Immediate
AUDIO_EADR	0x1c5/0x38A	0x006e	Decode
AUDIO_EMPTINESS	0x65/0xCA		
AUDIO_HEADER1	0x1bb/0x376		
AUDIO_HEADER2	0x1bc/0x378		
AUDIO_MUTE	0x1e7/0x3CE	0x0000	Immediate
AUDIO_SADR	0x1c4/0x388	0x0054	Decode
AUTO_PAUSE	0x1e2/0x3C4	0x0000	Immediate
AV_SYNC_MODE	0x1e4/0x3C8	0x0001	Immediate
BIT_RATE_H	0x1a4/0x348		
BIT_RATE__L	0x1a5/0x34A		
BROKEN_LINK	0x1ad/0x35A		
BWD_CODE	0x1b4/0x368		
CD_CONFIG	0x1cc/0x398	0x0216	Decode
CDDA_EMPHASIS	0x1e3/0x3C6	0x0000	Immediate
CLOSED_GOP	0x1ac/0x358		
CMDF_EADDR	0x79/0xF2		
CMDF_READ	0x7c/0xF8		
CMDF_SADDR	0x78/0xF0		
CMDF_WRITE	0x7d/0xFA		
CMD_ID	0x70/0xE0		
CCLCR_CR	0x1da/0x3b4	0x0080	Video Initialization
CCLCR_Y_CB	0x1db/0x3b6	0x1080	Video Initialization
CONS_FLAG	0x1a7/0x34E		
DATE_VERSION	0x1fa/0x3F4		
DRAM_REFRESH	0x1cf/0x39E	0x200	Startup
ERROR_LEVEL	0x1e5/0x3CA	0x0001	Immediate
EVEN_INTERPCOEF	0x1e9/0x3d2	0x0025	Immediate
F_MODE	0x199/0x332		

Table 12-2 CL480 DRAM Parameters (Continued)

Parameter	Word /byte Addr	Default Value	Change Effective
FC_NUM	0x19a/0x334		
FIRST_FIELD	0x1c6/0x38C		
FREE_SPACE_START	0x6c/0xd8		
FREE_SPACE_END	0x6d/0xdA		
FULL_PEL_BCK	0x1b3/0x366		
FULL_PEL_FOR	0x1b1/0x362		
FWD_CODE	0x1b2/0x364		
H_SIZE	0x1a0/0x340		
HEIGHT	0x0f2/0x1E4	0x0000	
HORIZONTAL_SIZE	0x1f3/0x3E6	0x0160	Immediate
INPUT_FORMAT	0x6e/0xdC	0x0000	
INT_MASK	0x1e6/0x3CC	0x0000	Immediate
INT_SOURCE	0x6f/0xdE		
INT_STATUS	0x63/0xC6	0x000	
INTERVAL	0x1ba/0x374		
LIQ_MATRIX	0x1a8/0x350		
LNIQ_MATRIX	0x1a9/0x352		
LSA_FM	0x69/0xd2		
LSA_MS	0x68/0xd0		
M_SEC	0x198/0x330		
MRC_ID	0x61/0xC2		
MRC_STATUS	0x62/0xC4		
NTSC_HSYNC_HI	0x1dd/0x3bA	0x0359	Video Initialization
NTSC_HSYNC_LO	0x1dc/0x3b8	0x004F	Video Initialization
NUM_DECODED	0x1b7/0x36E		
NUM_REPEATED	0x1b9/0x372		
NUM_SKIPPED	0x1b8/0x370		
ODD_INTERPCOEF	0x1eA/0x3d4	0x008F	Immediate
PA_RATIO	0x1a2/0x344		
PAL_HSYNC_HI	0x1df/0x3bE	0x035F	Video Initialization
PAL_HSYNC_LO	0x1de/0x3bC	0x004F	Video Initialization
PIC_RATE	0x1a3/0x346		
PIC_TYPE	0x1af/0x35E		
PICTURE_RATE	0x1f5/0x3EA	0x0004	Immediate
PLAY_MODE	0x1e0/0x3C0	0x0000 (for VCD)	Decode
PROC_STATE	0x60/0xC0		
ROM_CONFIG	0x1ce/0x39C	0x0007	Startup
SE_STATUS	0x67/0xCE		
S_HORIZONTAL_SIZE	0x1f6/0x3EC	0x02c0	Immediate

Table 12-2 CL480 DRAM Parameters (Continued)

Parameter	Word /byte Addr	Default Value	Change Effective
SMC_INFO	0x19b/0x336		
S_VERTICAL_SIZE	0x1f7/0x3EE	0x01e0	Immediate
STILL_MODE	0x1e1/0x3C2	0x0001	Decode
TEMP_REF	0x1ae/0x35C		
TIME_CODE_H	0x1aa/0x354		
TIME_CODE_L	0x1ab/0x356		
TOP_BORDER	0x1ee/0x3dC	0x0000	Video Initialization
UDF_EADDR	0x7b/0xF6		
UDF_READ	0x7e/0xFC		
UDF_SADDR	0x7a/0xF4		
UDF_WRITE	0x7f/0xFE		
UNUSED_IO	0x1ca/0x394		Startup
VA_STREAM_ID	0x1e8/0x3d0	0xe0c0	Startup
VBS_EADR	0x1c1/0x382	0x00ce	Decode
VBS_SADR	0x1c0/0x380	0x006e	Decode
VBV_DELAY	0x1b0/0x360		
VBV_BSIZE	0x1a6/0x34C		
VERTICAL_SIZE	0x1f4/0x3E8	0x00f0	Immediate
VIDEO_MODE	0x1d7/0x3AE	0x0b24	Startup
VIDEO_MODE2	0x1cd/0x39A	0x000c	Startup
VIDEO_EMPTINESS	0x64/0xC8		
VIDEO_FIELD	0x66/0xCC		
VIDEO_FORMAT	0x1eb/0x3d6	0x0004	Startup
VSYNC_EVEN_TP	0x1d9/0x3b2	0x0000	
VSYNC_ODD_TP	0x1d8/0x3b0	0x0180	
V_SIZE	0x1a1/0x342		
WEIGHTK0	0x1d1/0x3A2	0x0198	Video Initialization
WEIGHTK1	0x1d2/0x3A4	0x0730	Video Initialization
WEIGHTK2	0x1d3/0x3A6	0x079c	Video Initialization
WEIGHTK3	0x1d4/0x3A8	0x0204	Video Initialization
WEIGHTK4	0x1d5/0x3AA	0x0129	Video Initialization
WIDTH	0x0f1/0x1E2	0x0000	
XOFFSET	0x1ef/0x3dE	0x0000	Video Initialization
YOFFSET	0x1f0/0x3E0	0x0000	Video Initialization

13
Macro Commands

Note: The macro commands SetMute() and SetInterruptMask(), listed in the previous edition, have been replaced by the DRAM locations AUDIO_MUTE and INT_MASK, respectively. SetBorderColor() has been replaced by DisplayBorderColor() and the DRAM parameters COLOR_CR and COLOR_Y_CB. Also, InquireBufferFullness() has become InquireBufferEmptiness().

The host software uses CL480 macro commands as its primary method of communication. Macro commands are command IDs and argument values written into local DRAM by the host (as described in Section 4.2.4). Each command has a separate command ID and may have zero or more arguments.

Once the command IDs and arguments 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 must 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 (though more than one command can be written at a time).

13.1 High-Priority Commands

High-priority commands have the priority bit set in their command number. The high-priority commands are Reset(), InquireBufferEmptiness(), and Abort(). The high-priority commands are not stored in 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 the first word of this location 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.

The command is complete when the command ID field goes to zero.

13.2 Low-Priority Commands

Most CL480 commands, including all Play-type and Set-type commands, are low-priority. Low-priority commands are initiated by the host by writing the command and any command arguments into the command FIFO in DRAM. (Only low-priority commands should be placed in the command FIFO.) Low-priority commands are then 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, and thus, cannot be guaranteed to occur within a set period of time. 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. Otherwise, if the input bitstream buffer underflows while decoding a picture, the command FIFO will not be checked until enough data is input to complete the picture decoding. If picture decoding is not occurring—as in the PAUSE state, for example—the command FIFO is continually checked. Thus, the command FIFO checking frequency will vary from being continuous to very infrequent, depending on the state and availability of input data.

When a Play-type command— Play(), Pause(), Freeze(), DisplayDigest(), SetStreams(), SingleStep(), SlowMotion(), Scan(), DisplayStill(), DisplayBorderColor(), or DumpData()—is encountered in the command FIFO, it is extracted and executed immediately. If the com-

mand was not Pause(), DisplayBorderColor() 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 low-priority Set-type command—either SetVideoFormat(), 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 write a fixed number of bytes for each command; only write as many 16-bit words as the command needs.*

13.2.1 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 of the input bitstream.

The command FIFO has read and write pointers that 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 later write back the new value of the write pointer into DRAM. The CL480 microcode maintains the read pointer. (See Table 15-4 for further details on the command FIFO pointers.)

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 Figure 13-1.

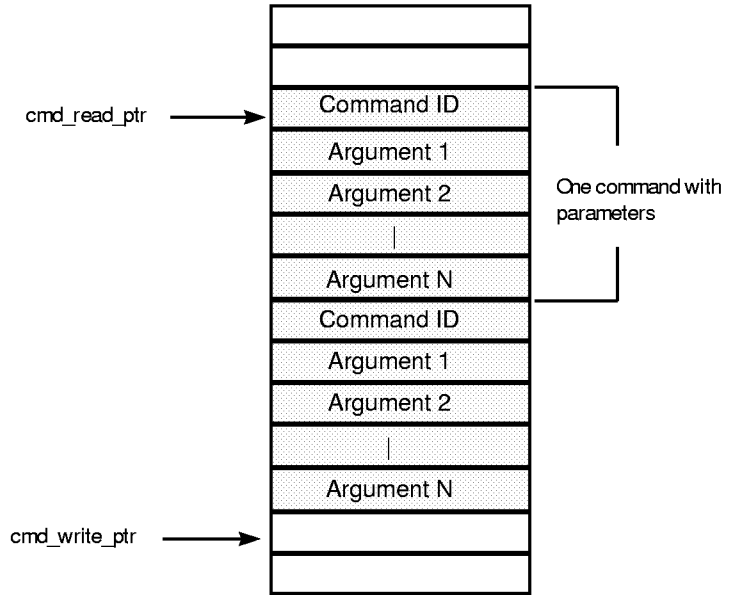


Figure 13-1 Command FIFO Structure

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.

13.2.2 User Data FIFO

The CL480 user data FIFO is where the CL480 microcode writes user data found in the user_data portion of the MPEG-1 picture header. The user data FIFO functions almost exactly the same as the command FIFO just described. Is is maintained as a circular buffer, with UDF_SADDR and UDF_EADDR specifying where the circular buffer resides in memory, and UDF_READ and UDF_WRITE specifying the read and write pointers, respectively. The CL480 microcode maintains the write pointer (any user_data encountered is written here), and the host processor maintains the read pointer. The host compares the read and write pointers to determine if and where any data exists in the buffers.

13.3 Status Area

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 CL480 microcode updates the Status Area upon executing low-priority commands. (See Table 15-1 for the contents and layout of the Status Area.) The Status Area contains the following parameters that are updated:

- *PROC_STATE (processing state):* Contains the new command state upon command completion.
- *MRC_ID (most recent command identifier):* Contains the identifier of the last command executed. The MRC_ID field is changed when the CL480 microcode starts executing the command read from the command FIFO.
- *MRC_STATUS (most recent command status):* Contains the completion status of the last command executed. This parameter is associated with the command indicated by the MRC_ID parameter. Thus, when the MRC_STATUS parameter indicates the WORKING_STATE field, it is also indicating the command ID for the MRC_ID field.
- *INT_STATUS (interrupt status):* Indicates which of the interrupt sources caused the interrupt to the host processor.
- *VIDEO_EMPTINESS:* Indicates the emptiness of the video rate buffer in 16-word blocks.
- *AUDIO_EMPTINESS:* Indicates the emptiness of the audio rate buffer in 16-bit blocks.
- *SE_STATUS (sector error status):* Provided only in the CL480VCD version, this field is used by the DumpData() command to indicate which sectors read were incorrectable.
- *LSA_MS* and *LSA_FM (last sector address minutes/seconds and frame/mode):* Provided only in the CL480VCD microcode, these fields hold the sector address of the of the last sector whose data was copied into bitstream buffer.
- *AEE_MS* and *AEE_FM:* Provided only in the CL480VCD microcode, these fields hold the sector address of the last sector received with auto-pause, end-of-record, or the end-of-file bit set.
- *NUM_DECODED:* Provides the count of the number of decoded pictures.
- *NUM_SKIPPED:* Provides the count of the number of pictures skipped due to audio/video synchronization.

- *NUM_REPEATED:* Provides the count of the number of pictures repeated due to audio/video synchronization.
- *FREE_SPACE_START* and *FREE_SPACE_END:* Used by the CL480VCD microcode only, these parameters specify the memory below four Mbits that is usable by the host processor.
- *INPUT_FORMAT:* Used by the CL480VCD microcode only, this parameter specifies the type of bitstream being received.
- *INT_SOURCE:* Indicates which source caused the END-P, ERR, AOR, and A/E/E interrupts.

13.4 Writing DRAM Locations

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 described in Chapter 15.

13.5 Command States

The CL480 operates in one of twelve possible internal command states. Command states determine how future commands are interpreted and how the decoding and displaying processes operate.

Several command states correspond to a macro command that can cause a state transition. The macro commands which can explicitly change the command state are the Reset() and Abort() commands and all of the Play-type commands. In addition to macro commands, several internal operations can also cause a state transition.

The command states and their relationships are shown in Figures 13-2 and 13-3. Within these figures, the twelve command states are inscribed within ovals. The shaded arrows indicate state transitions which occur due to the completion of internal processing, while solid arrows indicate transitions caused by the execution of a macro command.

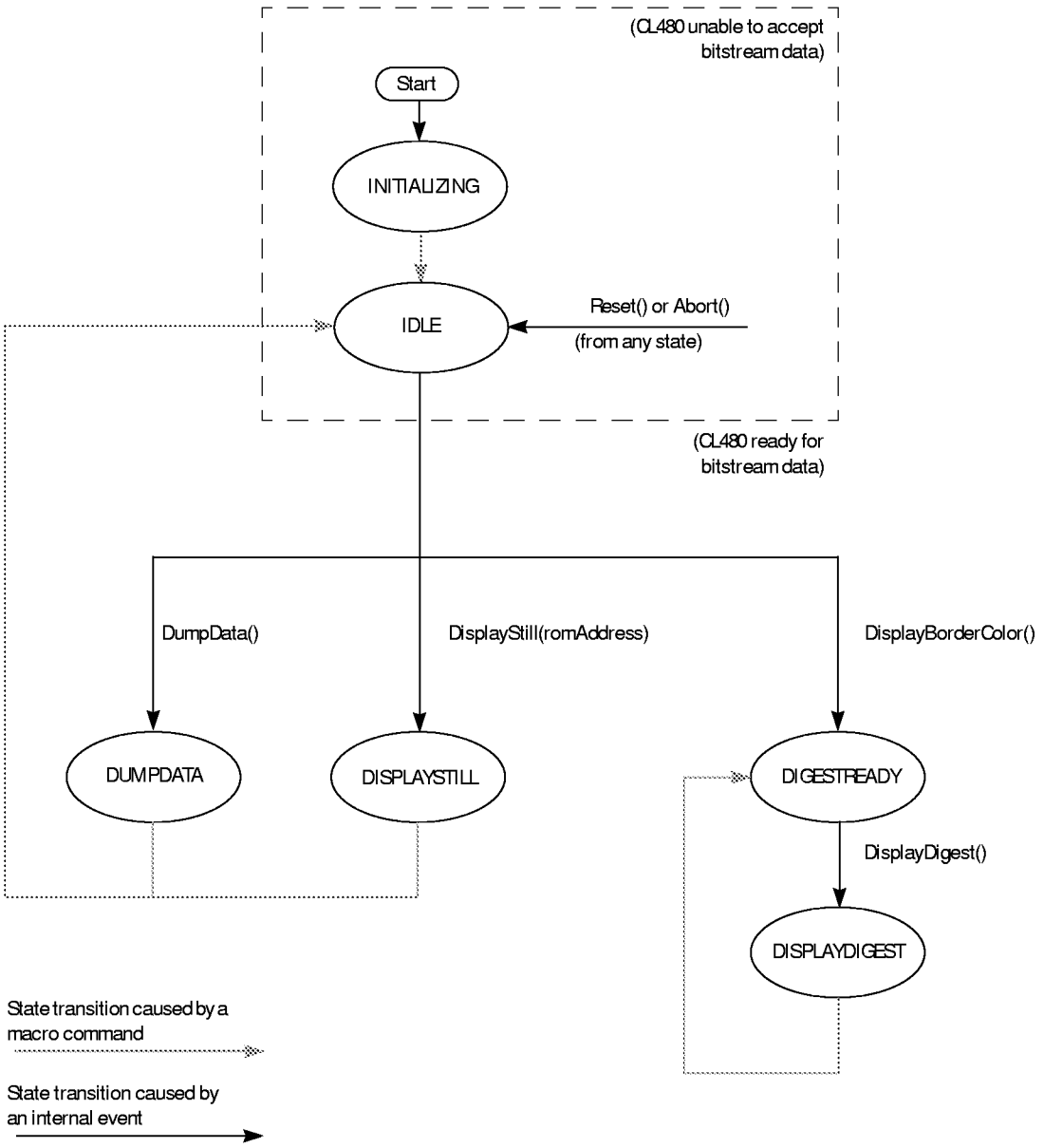
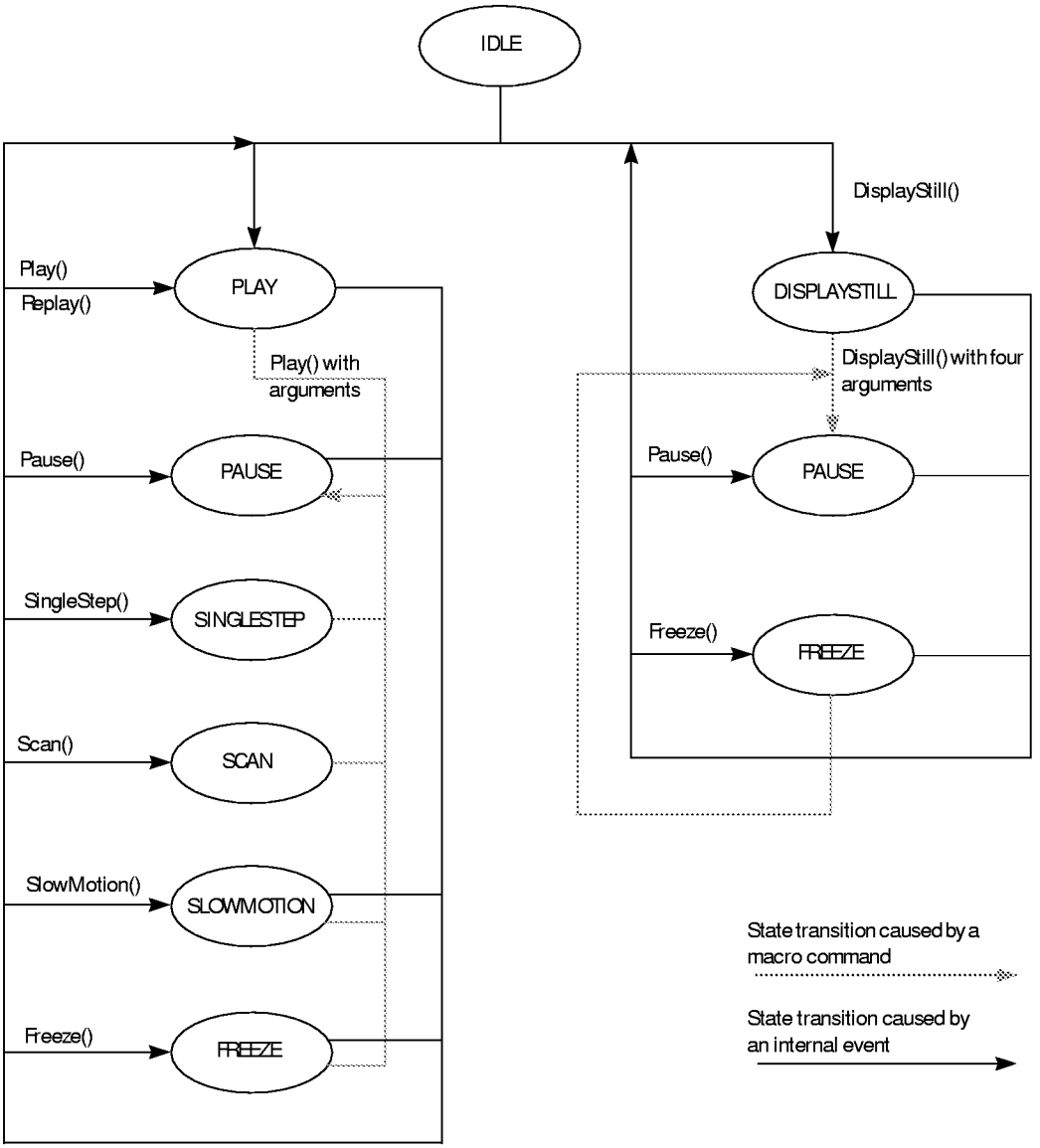


Figure 13-2 Command State Transition Diagram: DumpData(), DisplayStill() and DisplayDigest()



Note: The automatic state transition from FREEZE and SLOWMOTION to PAUSE occurs when the most recent Play() or DisplayStill() command was a Play() or DisplayStill() with four arguments and the stop sector address was detected.

Figure 13-3 Command State Transition Diagram: Play() and DisplayStill()

13.6 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 and in Table 13-1.

13.6.1 Set-Type Commands

The CL480 has two Set-type macro commands, DisplayGraphics() and SetVideoFormat(), both of which:

- Never affect the command state
- May be issued regardless of the current command state
- Have no effect on the decoding process

13.6.2 Play-Type Commands

The CL480 has 14 Play-type macro commands, each of which causes the current command state to change.

Play-type commands—except the Pause(), DumpData() and DisplayBorderColor() commands—cause the CL480 to read, decode and output data from the host or serial interface. The Pause() command suspends bitstream decoding and outputting.

The DumpData() command is also different from other Play-type commands because it reads the data from the serial interface, performs error correction, and writes the data to DRAM for the host processor to read.

13.6.3 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 13-1 Macro Command Summary

Category	Priority	Effect on Command State	Name	Description	Cmd. ID	Page
Set-Type	Low		DisplayGraphics()[1]	Decodes and displays on-screen graphic overlay	0417	146
			SetVideoFormat()	Sets format to NTSC, PAL or progressive	0305	170
Play-Type	Low	Yes	DisplayBorderColor()	Sets active display region to border color	001D	142
			DisplayDigest()	Decodes a single I-picture digest display	031B	143
			DisplayDigest(start)[1]	Same as above but specifies starting point	041B	144
			DisplayStill()[2]	Decodes/displays single still picture w/ audio	000C	148
			DisplayStill(start)[1,2]	Same as above but specifies starting point	020C	150
			DisplayStill(RomAddress)[1,2]	Same as DisplayStill() but specifies ROM address	030C	151
			DisplayStill(start, stop)[1,2]	Same as DisplayStill() but for CL480VCD only	040C	152
			DumpData()[1]	Reads and performs ECC on CD-ROM data	0414	154
			Freeze()	Freezes display but continues decoding	0010	156
			Pause()	Freezes display and decoding process	000E	158
			Play()	Decodes and displays at normal rate	000D	159
			Play(start, stop)	Same as above but specifies start/stop	040D	161
			Replay()	Restarts the previous Play() command	001C	163
			Scan()	Decodes and displays next single I-picture	000A	165
			SetStreams()	Selects video and audio stream identifiers	0213	169
			SingleStep()	Decodes and stores next single picture	000B	171
			SlowMotion()	Decodes and displays at slower rate	0109	172
Control	High	Yes	Abort()	Sets processing state to IDLE	801A	141
			Reset()	Reinitializes CL480 and its microcode	8000	164
			InquireBufferEmptiness()	Measures data in bitstream buffer	8003	157

1. For CL480VCD microcode version only
2. Allows display of a high-resolution still image with double horizontal and vertical resolution as specified in VideoCD 2.0.

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

13.7 Macro Command Reference

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

Note: The macro commands SetMute() and SetInterruptMask(), listed in the previous edition, have been replaced by the DRAM locations AUDIO_MUTE and INT_MASK, respectively. SetBorderColor() has been replaced by DisplayBorderColor().

Abort()

Format: Abort()
Priority: High
Category: Control

Command ID: 801A

This high-priority command causes the CL480 to transition to the IDLE state. Any commands currently executing are terminated. No other functions are performed.

This command is used to instruct the CL480 microcode to return to examine the contents of the command FIFO. Since the content of the frame buffers is indeterminate when this command is executed, MPEG-1 decoding must begin by decoding an I-picture.

This command can be executed from any state. Afterwards, the CL480 microcode waits for a new command to be issued. The screen is not blanked by this command.

The Abort() command is complete after the high-priority command ID location goes to zero.

DisplayBorderColor()

Format:	DisplayBorderColor()
Priority:	Low
Category:	Play-Type
Command ID	001D

This low-priority command can only be executed from the IDLE state. It causes the current border color to be displayed in the active region of the video output device. (COLOR_CR and COLOR_Y_CB configuration parameters contain the current border color.)

The CL480 transitions to the DIGESTREADY state upon completion of this command.

The Abort() command can be used to transition back to IDLE.

DisplayDigest()

Format:	DisplayDigest(xOffset, yOffset, decimation)
Priority:	Low
Category:	Play-Type
Command ID	031B

This low-priority command causes the CL480 to search for the first I-picture, decode it and display the picture at the specified location (**xOffset** and **yOffset**) from the upper left-hand corner of the high resolution display area. (The decoded picture is not interpolated before display.)

The **decimation** argument specifies the amount to reduce the decoded picture in both the horizontal and vertical dimensions.

This command causes the CL480 to transition to the DISPLAYDIGEST state while the I-picture is being decoded. When complete, the CL480 transitions to the DIGESTREADY state.

This command can be executed from the DIGESTREADY state.

DisplayDigest(**xOffset**, **yOffset**) arguments are defined as follows:

```
unsigned XOffset;       /* X offset to display picture (units: 601 pixels)*/
unsigned YOffset;       /* Y offset to display picture (units: 601 lines)*/
unsigned Decimation;    /* Amount of decimation to apply to the decoded*/
                        /* picture (0 = 2:1 and 1 = 4:1) */
```

The following restrictions apply to this command:

- It must be executed after the DisplayBorderColor() command, not before, since the DisplayDigest() command causes the CL480 to transition to the DISPLAYDIGEST state.
- It ignores the XOFFSET, YOFFSET, WIDTH and HEIGHT configuration parameters.
- It does not permit displayed pictures to extend beyond the edge of the display area.

DisplayDigest(start)

Format:	DisplayDigest(xOffset, yOffset, start1, start2, decimation)
Priority:	Low
Category:	Play-Type
Command ID	051B

This low-priority command causes the CL480 to (1) search for the first I-picture after detecting the start address specified by **start1** and **start2**, (2) decode this I-picture, and then (3) display this picture at the specified location (**xOffset** and **yOffset**) from the upper left-hand corner of the high-resolution display area.

The **decimation** argument specifies the amount to reduce the decoded picture in both the horizontal and vertical dimensions.

This command causes the CL480 to transition to the DISPLAYDIGEST state while waiting for the **start** address to be detected and while decoding the first I-picture after the **start** address is detected. When complete, the CL480 transitions to the DIGESTREADY state. It can also be executed from the DIGESTREADY state.

Address-out-of-range checking is performed while searching for the start sector address. If an address greater than the start address is detected while searching for the start address, the CL480 will generate an AOR interrupt, if enabled. The CL480 will then enter the DIGESTREADY state with the MRC_STATUS field set to ERROR_STATUS.

The following restrictions apply to this command:

- It must be executed after the DisplayBorderColor() command, not before, since the DisplayDigest() command causes the CL480 to transition to the DISPLAYDIGEST state.
- It ignores the XOFFSET, YOFFSET, WIDTH and HEIGHT configuration parameters.
- It does not permit displayed pictures to extend beyond the edge of the display area.

DisplayDigest(start)

DisplayDigest(**xOffset**, **yOffset, start1, start2, decimation**) arguments are defined as follows:

```
unsigned XOffset;          /* X offset to display picture (units: 601 pixels)*/
unsigned YOffset;          /* Y offset to display picture (units: 601 lines)*/
unsigned Start1, Start2;   /* Location to start decoding*/
                           /* Start1 is comprised of bytes: Minutes, Seconds*/
                           /* Start2 is comprised of bytes: Frame, Mode*/
unsigned Decimation;       /* Amount of decimation to apply to the decoded*/
                           /* picture (0 = 2:1 and 1 = 4:1) */
```

Note: All sector addresses specifying Minutes, Seconds, Frame and Mode are stored in two words: ***start1*** *and* ***start2*** *or* ***stop1*** *and* ***stop2****. The most significant byte of* ***start1*** *or* ***stop1*** *specifies the Minutes, and the least significant byte specifies the Seconds. The most significant byte of* ***start2*** *or* ***stop2*** *specifies the Frame, and the least significant byte specifies the Mode.*

DisplayGraphics()

Format: DisplayGraphics(startline, stopline, address, fade)
Priority: Low
Category: Control

Command ID: 0417

This low-priority command is available in the CL480VCD microcode version only. It instructs the CL480 to display a graphic overlay loaded in a user area of DRAM. (See the description of the FREE_SPACE_START and FREE_SPACE_END parameters in Table 15-1 for how to determine usable memory.) The graphic overlay is displayed starting with the first field following execution of the DisplayGraphics() command. The graphics overlay is defined to start at the first line of the active region of the display defined by the TOP_BORDER configuration parameter.

DisplayGraphics() arguments are defined as follows:

- **startline** - Used to position the graphic vertically on the display from the line specified by the TOP_BORDER configuration parameter. The sum of the graphic overlay's `numblank` parameter and the **startline** argument specify the vertical starting position of the graphic on the display. If TOP_BORDER is zero (centering enabled), the graphic image will be displayed relative to the first display line.
- **stopline** - The line number used to stop displaying the graphics overlay. That is, **stopline** is the number of the line immediately after the last line of the graphics overlay displayed.

  *Note: Both **startline** and **stopline** are in the units of lines counted down from the top margin of each field.*

- **address** - The starting address of the buffer in CL480 local DRAM which holds the graphics overlay. The starting address for the graphic overlay must be on a 256-word page boundary.
- **fade** - Used to control the fade factor used to display the graphic. The **fade** argument is a number from 0 to 31 in bits 14 through 10. The larger the number for the fade factor, the more of the background will show through the graphic.

DisplayGraphics()

The graphics overlay is disabled by issuing the DisplayGraphics() command with a value of **stopline** equal to zero, or issuing the Reset() command. This command does not cause a state transition and can be executed while in any state, including IDLE.

The DisplayGraphics() command does not wait for the graphic image to be displayed before checking the command FIFO for other commands. If another DisplayGraphics() command is executed from the command FIFO before the previous graphic image was displayed, the previous graphic image will not be displayed. Instead, the new graphic image will be displayed. (Graphic image data is parsed and displayed by the video output process.)

DisplayGraphics() arguments are defined as follows:

```
short int StartLine;    /* Video line number to start displaying*/
                        /* graphics overlay (601 lines / 2).*/
short int StopLine;     /* Video line number to stop displaying*/
                        /* graphics overlay. A value of zero will*/
                        /* disable the graphics overlay (601 lines / 2)*/
short int Addr          /* Address in CL480 local DRAM of the start of */
                        /* the graphics overlay buffer (16 bit 256 word*/
                        /* page address).*/
short int Fade          /* Amount of fade to apply to the graphic. 0-31 in*/
                        /* bits 14 through 10; 0 = no fade, 31 = max. fade*/
```

DisplayStill()

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

This low-priority command causes the CL480 microcode to read, decode and output normal- to high-resolution still pictures and audio samples from the incoming bitstream.

The CL480 microcode initializes the hardware and software for processing incoming bitstreams before decoding the incoming bitstream. All bitstream buffers and the audio output buffer are flushed. The hardware components initialized include the ROM decoder and the video output unit.

This command continues decoding and outputting video and audio data until a Freeze() or Pause() command is executed. The new CL480 processing state will be the DISPLAYSTILL state. The DisplayStill() command can be issued from the IDLE, PAUSE, FREEZE, PLAY, SLOWMOTION or DISPLAYSTILL states.

> *Note: The STILL_MODE configuration variable (described in Table 15-7) can be used to control the blanking of the display between high resolution still pictures. If the STILL_MODE variable is non-zero, the screen will be blanked before the new picture is posted.*

The CL480 can decode and display both high- and low-resolution still pictures as defined by the *Video CD Specification Version 2.0*. Which still pictures are decoded and displayed depends upon which video stream is selected and the resolution specified in the sequence header for the stream selected.

For Video CD video streams, the stream identifier for normal-resolution still pictures is defined to be 0xE1 (352 x 240 pixels or 352 x 288). For high-resolution still pictures, the stream identifier is defined to be 0xE2 (704 x 480 or 704 x 576).

Any valid MPEG-1 video stream identifier can be used as long as the video stream selected meets the requirements of the specification mentioned above. The stream identifiers can be set using the SetStreams() command (see Chapter 13).

> *Note: The DisplayStill() command does not terminate at the end of the first picture; instead, it plays slide shows. To stop decoding at the end of the first picture, a Pause() command should be issued after issuing the DisplayStill() command.*

For normal resolution pictures, the previous picture is displayed until a new picture has been decoded and is ready to be displayed. When the picture is ready to be displayed, the display is switched to the newly decoded still picture.

For high resolution pictures, the picture is displayed as it is decoded with the newly decoded picture filling the display from top-to-bottom.

If the currently displayed picture is of normal resolution and the new picture being decoded is high resolution, the display is blanked to the background color while the picture is decoded, and then the picture fills the display from top-to-bottom as the picture is decoded.

DisplayStill(start)

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

Command ID: 020C

This low-priority command is available in the CL480VCD microcode version only. This command is similar to the DisplayStill() command on the previous page, except that it always expects the incoming bitstream to be a CD-ROM bitstream.

This command searches the incoming bitstream for the sector address indicated by the arguments **start1** and **start2** (**start2** must be zero for Video CD). The first still picture in the selected video stream is decoded and displayed.

> *Note: The STILL_MODE configuration variable (described in Table 15-7) can be used to control the blanking of the display between high resolution still pictures. If the STILL_MODE variable is non-zero, the screen will be blanked before the new picture is posted.*

If while searching for the start sector address an address greater than the start address is detected, the CL480 microcode will generate an AOR interrupt, if requested, and enter the PAUSE state. This DisplayStill() command can be issued from the DISPLAYSTILL, IDLE, PAUSE, FREEZE, SLOWMOTION or PLAY states.

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

```
short int Start1, Start2;      /* Location to start decoding (Minutes,*/
                               /* Seconds, Frame and Mode) – Start1 is*/
                               /* comprised of bytes: Minutes and Seconds, while*/
                               /* Start2 is comprised of bytes: Frame and Mode*/
```

> *Note: All sector addresses specifying Minutes, Seconds, Frame and Mode are stored in two words: **start1** and **start2** or **stop1** and **stop2**. The most significant byte of **start1** or **stop1** specifies the Minutes, and the least significant byte specifies the Seconds. The most significant byte of **start2** or **stop2** specifies the Frame, and the least significant byte specifies the Mode.*

DisplayStill(romAddress)

Format: DisplayStill(romAddress1, romAddress2, length)
Priority: Low
Category: Play-Type

Command ID: 030C

This low-priority command is available in the CL480VCD microcode version only. This command causes the CL480 to read **length** words of the Video CD formatted still picture bitstream at the specified ROM address (**romAddress1** and **romAddress2**). The bitstream must contain demultiplexed video data only and must fit in the video bitstream buffer specified by the VBS_SADR and VBS_EADR configuration parameters.

Audio will not be played.

The CL480 transitions to the DISPLAYSTILL state while reading, decoding and displaying the video frame. Once the still picture is displayed, the CL480 transitions to the IDLE state.

DisplayStill(**romAddress, romAddress2, length**) arguments are defined as follows:

```
short int   RomAddress1,      /* Location of Video CD formatted still picture.*/
            RomAddressd2;     /* RomAddress1 is most significant word of the address*/
                              /* and RomAddress2 is least significant word of the*/
                              /* address (The address used is a word address.)*/
short int   Length            /* Length of Video CD formatted still picture (in words)*/
                              /* bitstream (the still picture must fit in the video*/
                              /* bitstream buffer).*/
```

DisplayStill(start, stop)

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

Command ID: 040C

This low-priority command is available in the CL480VCD microcode version only and is similar to the DisplayStill(**start1**, **start2**) command. Once decoding has begun, decoding and outputting continues until the specified stop sector address is detected (i.e., a slide show). After detecting the sector stop address, the CL480 microcode enters the FREEZE state.

> *Note: The STILL_MODE configuration variable (described in Table 15-7) can be used to control the blanking of the display between high resolution still pictures. If the STILL_MODE variable is non-zero, the screen will be blanked before the new picture is posted.*

If while decoding still pictures an address less than the start address is detected, the CL480 microcode will generate an AOR interrupt, if requested, and enter the PAUSE state. This DisplayStill() command can be issued from the IDLE, PAUSE, FREEZE, PLAY, SLOWMOTION or DISPLAYSTILL states.

For normal resolution pictures, the previous picture is displayed until a new picture has been decoded and is ready to be displayed. When the picture is ready to be displayed, the display is switched to the newly decoded still picture.

For high resolution pictures, the picture is displayed as it is decoded with the newly decoded picture filling the display from top-to-bottom.

If the currently displayed picture is of normal resolution and the new picture being decoded is high resolution, the display is blanked to the background color while the picture is decoded, and then the picture fills the display from top-to-bottom as the picture is decoded.

DisplayStill(start, stop)

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

```
short int Start1, Start2;     /* Location to start playing (Minutes,*/
                              /* Seconds, Frame and Mode) Start1 is comprised*/
                              /* of bytes: Minutes, Seconds while Start2 is*/
                              /* comprised of bytes: Frame, Mode*/
short int Stop1, Stop2;       /* Location to stop playing (Minutes,*/
                              /* Seconds, Frame and Mode) Stop1 is comprised*/
                              /* of bytes: Minutes, Seconds while Stop2 is*/
                              /* comprised of bytes: Frame, Mode*/
```

DumpData()

Format: DumpData (start1, start2, length, address)
Priority: Low
Category: Play-Type

Command ID: 0414

This low-priority command is available in the CL480VCD microcode version only. This command causes the CL480 microcode to read Mode 1 or Mode 2 Form 1 CD-ROM data from the specified starting point and for the specified **length** sectors (up to a maximum of 16 sectors). The CL480 microcode reads up to 16 sectors performing ECC error correction, as necessary, to the data into a buffer in DRAM.

The starting address of this DRAM buffer is specified by the **address** parameter. The buffer must start on a 256 word boundary. The address for DumpData() can be any address greater than that specified by the FREE_SPACE_START pointer. If data is written to an address equal to or greater than that specified by FREE_SPACE_END, it will be overwritten by execution of a Play-type command.

If any of the arguments for this command are invalid, MRC_STATUS indicates which parameter caused the error. MRC_STATUS also indicates if an error occurred while reading sectors, but before error correction is performed. Table 13-2 lists the error values and the condition causing the error. Refer to Table 15-1 for numeric values stored in MRC_STATUS.

Table 13-2 DumpData() Error Condition

MRC_STATUS	Error Condition
INVALID_SECTOR_COUNT	Length parameter not within valid range
INVALID_BUF_ADDRESS	Address parameter that specifies a dump to a reserved location.

Table 13-2 DumpData() Error Condition

MRC_STATUS	Error Condition
SECTOR_OUT_OF_RANGE	Sector address greater than start1 and start2 detected while searching for address specified by start1 and start2. (The Frame field of start2 is checked not the Mode field.)
SECTOR_TYPE_ERROR	Mode set to any value other than Mode 1 or Mode 2. A sector was detected while reading sectors with a Mode other than specified in the start argument. The Data bit of the Sub-Mode field is not set for a Mode 2 Form 1 sector.

At the conclusion of this command, the last sector address read and the sector error status are stored into the Status Area's LSA_MS and SE_STATUS parameters, respectively (See Table 15-1).

SE_STATUS, the sector error status, is a word which reflects which sectors the ECC error correction procedure failed to correct. Each bit of the sector error status word 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. After updating the Status Area, the END-D interrupt is generated, if enabled.

This command causes the CL480 microcode state to change to DUMPDATA while the data is being read and corrected; afterwards, the state changes to IDLE. The DumpData() command can be executed from the IDLE state. If the CL480 is in a command state other than IDLE, the Abort() command can be used to move the CL480 to the IDLE state.

DumpData() arguments are defined as follows:

```
short int Start1, Start2;   /* Location to start decoding (Minutes,*/
                            /* Seconds, Frame and Mode) Start1 is comprised*/
                            /* of bytes: Minutes, Seconds while Start2 is*/
                            /* comprised of bytes: Frame, Mode*/
short int Length;           /* Length of sequence to read (1 ≤ Length ≤ 16) */
                            /* 16 sectors) */
short int Address;          /* Address in CL480 DRAM to store the*/
                            /* data (16 bit 256 word page address). The*/
                            /* size of this buffer should be Length * 2048*/
                            /* bytes. The entire corrected data buffer must*/
                            /* reside within the rate buffer space as*/
                            /* designated in the configuration area*/
```

Freeze()

Format: Freeze()
Priority: Low
Category: Control

Command ID: 0010

This low-priority command causes the CL480 microcode to stop outputting video frames. Audio continues to be decoded and outputted

If a video frame is being displayed at the time the Freeze() command is executed, the video frame will be displayed while in the FREEZE state.

If the previous Play() command was a Play() command with arguments, the PAUSE state will be entered when the stop sector address is detected. The Freeze() command can be issued from any state except the IDLE and PAUSE states.

If the FREEZE processing state is terminated by receiving a Play() or Replay() command, video decoding begins with the first I-picture detected following the execution of the Play() or Replay() command.

InquireBufferEmptiness()

Format: InquireBufferEmptiness()
Priority: High
Category: Control

Command ID: 8003

Note: This command was previously called InquireBuffer-Fullness().

This high-priority command causes the CL480 microcode to calculate the number of unused 16-word blocks in the video and audio bitstream buffers. This state of emptiness is indicated by the VIDEO_EMPTINESS and AUDIO_EMPTINESS DRAM parameters described in Table 15-1.

Buffer emptiness is calculated when the command is executed, and the END-C interrupt can be used to determine when the calculation is complete.

Values computed by the microcode are based on an internal snapshot that may be slightly out of date before the buffer emptiness computation is complete.

The emptiness of the video and audio rate buffers is indicated by the VIDEO_EMPTINESS and AUDIO_EMPTINESS DRAM parameters defined in Table 15-1.

Pause()

Format:	Pause()
Priority:	Low
Category:	Play-Type
Command ID:	000E

This low-priority command causes the CL480 microcode to stop accepting input bitstreams and to stop decoding and outputting audio data and video frames.

The CL480VCD microcode will read any data input after entering the PAUSE state and write the data into the video or audio bitstream buffer, allowing the buffer to overflow.

The CL480PC microcode will continue to read data until one of the bitstream buffers fills and more data is to be written.

If a video frame is being displayed at the time the Pause() command is executed, the video frame will be displayed while in the PAUSE state. The new state after processing this command will be the PAUSE state. This command can be issued from any state except the IDLE state.

Play()

Format: Play()
Priority: Low
Category: Play-Type

Command ID: 000D

This low-priority command causes the CL480 microcode to start decoding and outputting the incoming bitstream. Before the decoding begins, the CL480 microcode initializes the hardware and software. The two hardware components initialized are described below:

- *The ROM decoder* - Initialized according to the value of the PLAY_MODE configuration variable (see Table 15-7). If the PLAY_MODE variable is set to CD_ROM_AUTO, the ROM decoder is initialized to search for sector sync words indicating bitstreams from a VideoCD disk (see Table 15-6 for numerical values for the PLAY_MODE variable). For PLAY_MODE set to CD_ROM_CDDA, the ROM decoder is initialized to not search for sector sync words.
- *The Video Output unit* - Initialized according to the configuration parameters stored in the Configuration Area (see Table 15-7).

The software initialization includes flushing all bitstream buffers and the audio output buffer, and informing the decoding process to start decoding MPEG-1 bitstreams with the next I-picture.

When initialization is complete, the CL480 microcode starts the bitstream auto determination process explained in Chapter 12. If at any time the bitstream changes, the CL480 microcode will try to determine the type of the new bitstream and change its processing to handle the new bitstream type.

Once a valid bitstream is detected, the input, decoding and output processes are performed according to the corresponding sections above. The incoming bitstream is parsed and placed in the appropriate bitstream buffer or output buffer in the case of a CD-DA bitstream. For MPEG-1 bitstreams, the audio and video bitstreams are decoded and the

decoded audio and video pictures are output. The decoded audio and video pictures are synchronized according to the synchronization procedure selected by the AV_SYNC configuration variable (see Table 15-7).

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

This command causes a state transition to the PLAY state and can be issued from the IDLE, PAUSE, FREEZE, SLOWMOTION or DISPLAY-STILL states. The CL480 will remain in the PLAY state until another command is executed causing the state to change. The new state will be the state indicated by the command executed.

Since the command FIFO is checked once per video frame decode and the input and demultiplex process performs the address search, the next command in the command FIFO will not be executed until the start address is detected. The Abort() command can be used to cause the CL480 microcode to examine the command FIFO.

Play(start, stop)

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

Command ID: 040D

This low-priority command is available in the CL480VCD microcode version only. This command is similar to the Play() command mentioned previously. The difference in the two commands is that this command allows the host to specify the start and stop sector addresses to decode and output. This command can be executed from the IDLE, SLOWMOTION, FREEZE, PAUSE or PLAY states and will cause the CL480 to transition to the PLAY state.

Execution of this command starts by searching the incoming bitstreams for the sector addresses provided as arguments. If the start sector address is detected before the stop address is detected, the CL480 begins decoding and outputting the incoming bitstream until the stop address is detected. When the stop address is detected, the CL480 enters the PAUSE state.

If while searching for the start sector address a sector address is received which is greater than or equal to the stop address, the CL480 microcode will generate an AOR interrupt, if requested, and enter the PAUSE state. If a sector address received is less than the start sector address once decoding begins, the CL480 microcode will again generate the AOR interrupt, if requested, and transition to the PAUSE state.

If the PLAY_MODE configuration parameter is set to CD_ROM_AUTO and a CD-DA bitstream is input, the CL480 microcode will transition to decoding and outputting the CD-DA data and suspend the above address search until a CD_ROM bitstream is input.

Since the command FIFO is checked once per video frame decode and the input and demultiplex process performs the address search, the next command in the command FIFO will not be executed until the start address is detected. The Abort() command can be used to cause the CL480 microcode to examine the command FIFO.

Play(start, stop)

When a Play() command with arguments starts the decoding and output processes, the detection of the stop address will terminate the decode and output processes from any state. The Replay() command, explained next, behaves slightly different than this.

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 while Start2 is*/
                               /* comprised of bytes: Frame, Mode(=2)*/
short int Stop1, Stop2;        /* Location to stop playing (Minutes,*/
                               /* Seconds, Frame) Stop1 is comprised of*/
                               /* bytes: Minutes, Seconds while Stop2 is*/
                               /* comprised of bytes: Frame, Mode(=2)*/
```

Replay()

Format: Replay()
Priority: Low
Category: Play-Type

Command ID: 001C

This low-priority command restarts the previous Play() command. No hardware and software initialization is performed. The CL480 microcode does not flush the bitstream buffers; instead, it starts to decode with the next word in the bitstream buffers. This command can be used to continue the previous Play() command.

The Replay() command can be used to restart a Play() command with arguments. If the Replay() command is executed before the stop address is detected, decoding and outputting will continue until the stop address or an address less than the start address is detected, similar to the Play() command with arguments, mentioned previously. If the stop sector address has been detected, the Replay() command will restart decoding and outputting as if the previous command was a Play() command without arguments.

This command will cause the processing state to transition to PLAY. If the previous Play() command was a Play() command with arguments, the PAUSE state will be entered when the stop sector address is detected.

The Replay() command can be issued from the PLAY, PAUSE, FREEZE or SLOWMOTION states. The previous command must have been a Play() command for the Replay() command to have effect while in the SLOWMOTION state.

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 output window is blanked (screen is filled with current border color).
- The CL480 enters the IDLE state and is ready to accept the next command.

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, the Pause() command should be used. Unlike the Reset() command, the Pause() command allows the CL480 to continue to receive bitstream data and to continue to display the last picture decoded.

The Reset() command is done when the high-priority command ID becomes zero.

Scan()

Format: Scan()
Priority: Low
Category: Play-Type

Command ID 000A

The Scan() command can be issued from the PLAY, PAUSE, FREEZE, or SLOWMOTION states. The Scan() command searches for the first I-picture in the buffer, decodes it, and displays the picture. If the next I-picture includes user data, the user data will be processed as described in Table 15-4.

After an I-picture is displayed, the CL480 generates an END-P, if requested, and enters the PAUSE state.

Audio is muted while in the SCAN state.

SetBorderColor()

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

Command ID: 0207

The functions previously performed by the SetBorderColor() macro command are now performed by the DisplayBorderColor() macro command and the DRAM locations COLOR_CR and COLOR_Y_CB described in Table 15-7.

SetInterruptMask()

Format: SetInterruptMask(IntEventEna)
Priority: Low
Category: Set-Type

Command ID: 0104

The functions previously performed by the SetInterruptMask() macro command are now performed by the DRAM location INT_MASK, described in Table 15-7.

SetMute()

Format: SetMute(muteSetting)
Priority: High
Category: Set-Type

Command ID: 8118

The functions previously performed by the SetMute() macro command are now performed by the DRAM location AUDIO_MUTE, described in Table 15-7.

SetStreams()

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

Command ID: 0213

This low-priority command instructs the CL480 microcode to change the stream identifiers used to select the system packets containing either audio or video data. The stream identifiers specified are used for any of the Play-type commands: Play(), Replay(), Freeze(), SingleStep(), Scan(), SlowMotion(), Digest() and DisplayStill() commands.

The host processor can use this command to disable audio or video output by specifying an audio or video stream identifier not found in the incoming bitstream. Table 15-7 gives the default setting for the stream identifiers.

This command can be executed from any state. This command causes the CL480 to transition to the PAUSE state if the current state is not IDLE. No state transition occurs if the current state is IDLE.

The SetStreams() arguments are defined as follows:

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

SetVideoFormat()

Format: SetVideoFormat (format, 0, 0)
Priority: Low
Category: Set-Type

Command ID: 0305

This low-priority command instructs the CL480 microcode to modify the video output format according to the arguments given. If the video format specified in the input bitstream doesn't match the video output format, picture rate conversion is performed as specified in Section 7.4. This command can be issued any time after the microcode has been loaded and initialization is complete. No processing state transition occurs.

SetVideoFormat() arguments are defined as follows:

```
short int Format;          /* New Video Output Format: 0 = Use argument from last*/
                           /* SetVideoFormat() command or the VIDEO_FORMAT*/
                           /* DRAM parameter if no previous SetVideoFormat() command*/
                           /* was issued.*/
                           /* 1 = no field repeat, 2 = Output video format follows input */
                           /* bitstream video format, 3 = PAL output, 4 = NTSC output*/

                           /* Note: Parameters 2 and 3 need to be zero.*/
```

SingleStep()

Format: SingleStep()
Priority: Low
Category: Play-Type

Command ID: 000B

This low-priority command instructs the CL480 microcode to decode and display the next picture and then enter the PAUSE state. Audio output is muted while decoding and displaying the picture.

If the next picture is an I-picture with user data, the user data will be processed as described in Table 15-4. An END-P interrupt will be generated once the CL480 transitions to the Pause state, if enabled.

This command can be executed from the PLAY, FREEZE, SLOWMOTION or PAUSE states and will cause the CL480 to transition to the PAUSE state once the next frame is displayed.

The CL480VCD microcode will issue the END-P and RDY-S interrupts as required (see Chapter 14 for interrupt listings).

The CL480PC microcode uses the CFLEVEL pin on the CL480 to control data input during the SingleStep() command.

SlowMotion()

Format: SlowMotion(N)
Priority: Low
Category: Play-Type

Command ID: 0109

This low-priority command instructs the CL480 microcode to accept incoming data and decode pictures, displaying each picture for N-frame times. Audio decoding and output operates at the normal rate. Audio and video synchronization is suspended.

This command causes the SLOWMOTION state to be entered. This command can be issued from the PLAY, PAUSE, FREEZE or SLOWMOTION states.

The audio output can be muted using the AUDIO_MUTE configuration variable. I-pictures with user data will be processed as described in Table 15-4.

The CL480VCD microcode will issue the END-P and RDY-S interrupts as required (see Chapter 14 for interrupt listings).

The CL480PC microcode uses the CFLEVEL pin on the CL480 to control data input during the SlowMotion() command.

The SlowMotion() argument is defined as follows:

```
short int N;        /* factor to reduce speed by is 1 / N where*/
                    /* N <= 16*/
```

14
Interrupts

The 15 logical host interrupts which the CL480 microcode generates are produced within the microcode by monitoring internal conditions while the decoding and display processes execute. Because of this, the times at which interrupts can be produced (the domain for interrupts) is restricted. For example, the CL480 microcode will *not* produce new interrupts in any of the following circumstances:

- The microcode is not executing (CPU_cntl[0] == 0).
- The microcode is in the IDLE state.

  Note: Although new interrupts ($\overline{INT}$ pin transitioning to low) cannot be generated during any of these conditions, it is possible for the $\overline{INT}$ pin to remain active during these conditions. Once the $\overline{INT}$ pin is active, it will remain active until forced inactive by the host (see Section 14.2).

The events that can cause interrupts are generated by events which are detected during bitstream processing. The INT_MASK interrupt configuration parameter specifies the enabled interrupts.

> *Note: The INT_MASK DRAM configuration parameter replaces the SetInterruptMask() function described in the previous edition of this manual.*

When an interrupt occurs, the event which caused the interrupt is indicated by the contents of the INT_STATUS (interrupt status) field in the DRAM Status Area and the assertion of the $\overline{\text{INT}}$ pin (active low). The INT_STATUS field may indicate more than one interrupt source. If the interrupt can be caused by more than one source, the INT_SOURCE (interrupt source) status field will indicate the actual source of the interrupt.

The host processor must service all interrupts indicated in the INT_STATUS field before clearing the CL480 interrupt. To respond to an interrupt, the host should do the following:

1. Read the DRAM INT_STATUS location in the Status Area.
2. Handle the interrupt.
3. Write 0 to the *Int* bit of register HOST_int using a read-modify-write.
4. Write 0 to the INT_SOURCE and INT_STATUS locations (in the given order) in the DRAM Status Area.

Figures 14-1 and 14-2 show the contents of the INT_STATUS and INT_MASK fields according to the V-CD and PC versions of microcode.

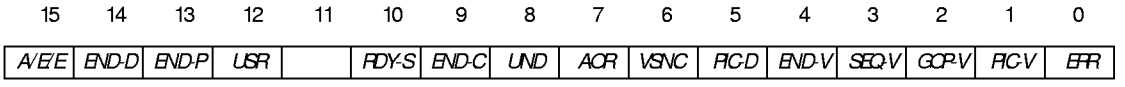


Figure 14-1 Mask Bit Allocation for V-CD Microcode

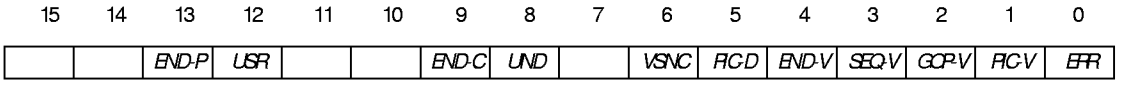


Figure 14-2 Mask Bit Allocation for PC Microcode

Each interrupt is referenced in Table 14-1 by name and the event which causes it.

Table 14-1 CL480 Interrupt Summary

Category	Interrupt Name	Event	Mask Bit
Decode-time	ACR[1]	Address out of range	7
	A/E/E[1]	Submode is auto pause, end of record/file	15
	END-C	High-priority command execution complete	9
	END-D[1]	End of DumpData() command	14
	END-P[2]	SingleStep(), SlowMotion() or End of Pause()	13
	ERR	Bitstream data error	0
	PIC-D	Picture decode complete	5
	UND	Bitstream buffer underflow error	8
	RDY-S[1]	Ready for data during SlowMotion()	10
	USR	User Data Ready	12
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
	VSYNC-V	$\overline{\text{VSYNC}}$ pulse occurred	6

1. Used only by the V-CD microcode.
2. The generation of the END-P interrupt in the SINGLESTEP or SLOWMOTION states during a buffer-full condition is available in the V-CD microcode only.

14.1 Interrupt Types

Each of the 15 logical interrupts produced by the CL480 microcode belongs to one of two separate classes based on when they are reported to the host:

- *Display-Time Interrupts*: These interrupts are denoted by a *-V* extension because they are issued following the active (falling) edge of $\overline{\text{VSYNC}}$.
- *Decode-Time Interrupts*: These interrupts, signified by not containing a *-V* extension, are issued every time the appropriate bitstream element is decoded, *if* the interrupt is enabled. Thus, decode-time interrupts are synchronous with picture decoding (which occurs in advance of picture display).

14.2 Interrupt Handshaking

All interrupts cause the $\overline{INT}$ pin to be asserted. To determine which interrupt(s) has occurred, the DRAM INT_STATUS location is written by the CL480 at the same time the $\overline{INT}$ pin is asserted.

Each bit in INT_STATUS corresponds to a different logical interrupt source as given in Table 14-1.

The host may read the INT_STATUS location at any time to determine which interrupts, if any, are pending. The host may also write to this location to clear pending interrupts by writing a zero into the *Int* field of the CL480 HOST_int register and then writing a zero into the INT_STATUS word.

The same handshaking protocol is used for all CL480 interrupts, regardless of the logical interrupt source. When the CL480 issues an interrupt to the host, it makes the $\overline{INT}$ pin active (LOW) and sets one or more bits in the INT_STATUS location of DRAM.

Because there is no semaphore to protect INT_STATUS from simultaneous access by both the host and the microcode, the protocol for issuing interrupts is structured such that simultaneous access cannot occur for the following two reasons:

- The microcode only writes INT_STATUS when it is 0.
- The host is only allowed to write INT_STATUS when it is not 0.

Once the host writes a 0 to INT_STATUS, the host cannot write to INT_STATUS again (even a 0) until the microcode sets one or more bits. If the host violates this protocol, then interrupts may be lost or spuriously created.

Excessive latency by the host in clearing the active bit(s) within INT_STATUS could also prevent the microcode from issuing subsequent interrupts as distinct events.

14.3 Interrupt Listing

This section gives a detailed description of the CL480 display-time and decode-time interrupts, listed together in alphabetical order.

Event: Subcode is auto-pause, end-of-record/file
Category: Decode-time
Mask Bit: 15

A/E/E

The A/E/E interrupt (CL480VCD microcode only) is generated as the CL480 microcode reads the CD sector information being input. If the subcode field of the CD-ROM/XA subheader indicates the sector is an auto-pause, end-of-record or end-of-file, the A/E/E interrupt will be generated. The bit description of the subcode field is as follows:

- Bit 0 of the subcode field indicates an end-of-record sector.
- Bit 4 of the subcode indicates an auto-pause sector.
- Bit 7 of the subcode indicates an end-of-file sector.

Each sector type bit is active when the bit position contains a one.

This interrupt is only valid while reading CD-ROM/XA sectors (CD-I or Video CD disks).

The INT_SOURCE status field indicates whether the sector causing the interrupt had set the end-of-record, auto-pause or end-of-file bit. This interrupt will be generated during the input bitstream and demultiplex process when the subcode field is read.

> *Note: This interrupt notifies the user that the auto-pause bit is set in the subcode header. If you wish to have the CL480 execute a Pause() command upon receiving the auto-pause bit, you must set the AUTO_PAUSE configuration DRAM parameter. If the END-P interrupt is enabled, the CL480 will issue an END-P interrupt when the PAUSE state is entered.*

Event: Address out of Range
Category: Decode-time
Mask Bit: 7

AOR

The AOR interrupt (CL480VCD microcode only) is generated when the CD sector address is out of range for the Play() or DisplayStill() commands (with arguments). The interrupt is generated when the CL480 microcode reads and processes the sector addresses.

While searching for the start sector address, the AOR interrupt indicates an address has been detected which is greater than or equal to the stop address. Once the start address is detected, the AOR interrupt indicates an address has been detected which is less than or equal to the start address.

The INT_SOURCE status field in DRAM indicates which of the above conditions generated the interrupt. Address checking is suspended when a CD-DA bitstream is input; thus, this interrupt will not be generated during CD-DA play.

END-C

Event: High Priority Command Execution Complete
Category: Decode-time
Mask Bit: 9

The END-C interrupt informs the host of the completion of the last high-priority command issued. This interrupt is most useful in determining when the CL480 microcode has updated the buffer emptiness fields in the Status Area upon completion of the InquireBufferEmptiness() command.

END-D

Event: End of DumpData()
Category: Decode-time
Mask Bit: 5

The END-D interrupt (CL480VCD microcode only) indicates that the CL480 microcode has completed the previous DumpData() command and that the read and corrected data is stored in DRAM at the host-specified area.

END-P

Event: PAUSE state entered, or buffer full with CL480VCD code
Category: Decode-time
Mask Bit: 13

Two conditions can cause the END-P interrupt to be generated:

- The CL480 transitioning to the PAUSE state.
- The CL480 is in the SLOWMOTION or SINGLESTEP state and the CL480VCD microcode is used. When in these states, the END-P interrupt indicates that the video bitstream buffer is at or above the high threshold level. The END-P interrupt is generated when the next CD-ROM sector is read after the bitstream buffer reaches the high threshold.

If the AUTO_PAUSE DRAM parameter is enabled, then this interrupt occurs after the CL480 transitions to the PAUSE state.

END-V

Event: Last picture displayed before `sequence_end_code`
Category: Display-time
Mask Bit: 4

The END-V interrupt event occurs at the rising edge of VSYNC prior to the first display field of the last picture (in display order) decoded prior to the detection of a `sequence_end_code`. Each time an END-V event occurs, the microcode either issues an END-V interrupt or places an END-V interrupt in the pending interrupt queue.

An END-V interrupt event occurs if all of the following are true:

- A `sequence_end_code` has been decoded.
- The picture which will be displayed in the following field time is the last picture (in display order) in the sequence terminated by the `sequence_end_code`.
- The END-V interrupt is enabled (See the INT_MASK DRAM parameter).

Name: Bitstream Data Error
Category: Decode-time
Mask Bit: 0

ERR

The ERR (bitstream data error) interrupt detects errors at the CD sector, MPEG-1 system stream, or MPEG-1 audio and video decoding layer. The INT_SOURCE field value will indicate which error occurred (see Table 15-1). The ERR interrupt event occurs each time the decoding process detects one of the following errors in the bitstream:

- An MPEG `sequence_error_code`
- Invalid variable-length codes
- `marker_bits` with 0 values
- Header fields with illegal values
- CD-ROM sector errors

The decoding process does not detect dropped or incorrect bits unless they cause one of these detectable conditions, in which case this interrupt may be used by the host to take the appropriate action (possibly applying system-level error concealment techniques), if any.

An ERR interrupt event occurs if both of the following are true:

- An error is detected in the coded data stream while decoding is being performed.
- The ERR interrupt is enabled (See the INT_MASK DRAM parameter).

Not all data errors can be detected, and the portion of the bitstream being decoded when an error is detected will not necessarily be the portion containing the error. For example, a dropped bit may cause an erroneous VLC (variable-length code) to be detected considerably later in the bitstream.

Typically, the microcode's internal error concealment operates by continuing to display a correctly-decoded reference frame until it finds a portion of the bitstream that it can decode. When this is occurring, undecodable portions of the bitstream (typically B-pictures and sometimes P-pictures) are discarded at a rapid rate. Because of this, the bitstream buffer can underflow temporarily, which may cause an additional interrupt.

Event: First I-picture display after `group_start_code`
Category: Display-time
Mask Bit: 2

GOP-V

The GOP-V interrupt event occurs at the rising edge of VSYNC before the first display field of the first I-picture decoded following the detection of a `group_start_code`. Each time a GOP event occurs, the microcode either issues a GOP interrupt or places a GOP interrupt in the pending interrupt queue.

The microcode uses an internal flag to produce the GOP-V interrupt. This flag is cleared each time the microcode reaches the IDLE state, the FlushBitstream() command is executed, or a GOP-V interrupt event occurs. The flag is set each time a `group_start_code` is decoded.

A GOP interrupt event occurs if all of the following are true:

- The timing for display-time interrupts is satisfied.
- The picture to be displayed in the following field time is an I-picture which has not been previously displayed.
- The GOP-V interrupt flag is set.
- The GOP-V interrupt is enabled (See the INT_MASK DRAM parameter).

A group of pictures can be decoded and displayed without a GOP-V interrupt event occurring (if it contains only P- and B-pictures), but GOP interrupt events cannot occur more often than once per group_of_pictures start code.

Event: New picture decoded
Category: Decode-time
Mask Bit: 5

PIC-D

The PIC-D (new picture decoded) interrupt event occurs each time all of the following are true:

- A coded picture is detected in the coded data stream and completely decoded without error.
- The PIC-D interrupt is enabled (See the INT_MASK DRAM parameter).

Each time a PIC-D event occurs, the microcode either issues a PIC-D interrupt or places a PIC-D interrupt in the pending interrupt queue.

The PIC-D interrupt is closely related to the PIC-V interrupt because both are produced by the processing of newly coded pictures. However, while the PIC-D and other Decode-time interrupts are issued to the host immediately after the *entire* picture is decoded, the PIC-V interrupt is produced by the detection of a `picture_start_code` and is not issued until the VSYNC prior to the display of the picture found in the bitstream after the `picture_start_code`.

Therefore, the order in which PIC-D and PIC-V interrupts are issued depends on the type of picture being decoded. If an I- or P-picture is decoded, then the entire picture is completely decoded before display starts, and PIC-D precedes PIC-V. If a B-picture is being decoded, then display will begin after only half of the picture is decoded, and PIC-V precedes PIC-D by approximately one field time. In addition, bitstream errors may cause PIC-V interrupts to be produced without a corresponding PIC-D (if an error occurs while decoding a B-picture) or vice versa (if an error occurs which causes a completely decoded reference frame never to be displayed).

Event: New Picture Displayed
Category: Decode-time
Mask Bit: 1

PIC-V

The PIC-V (new picture displayed) interrupt event occurs at the rising edge of VSYNC before the first display field of a newly-decoded picture. Each time a PIC-V event occurs, the microcode either issues a PIC-V interrupt or places a PIC-V interrupt in the pending interrupt queue.

A PIC-V interrupt event occurs if all of the following are true:

- The timing for VSYNC is satisfied.
- The picture to be displayed in the following field time has not been previously displayed.
- The PIC-V interrupt is enabled (See the INT_MASK DRAM parameter).

Event: Ready for data during SlowMotion() or SingleStep()
Category: Display-time
Mask Bit: 10

RDY-S

The RDY-S interrupt event—ready for data during SlowMotion() or SingleStep()—(CL480VCD microcode version only) occurs when the amount of data in the video bitstream rate buffer in DRAM has fallen below the low threshold value. It is used in conjunction with the END-P interrupt to control data flow during the SlowMotion() command.

It can be used by the host:

- To indicate that more data needs to be sent to the CL480.
- With a larger threshold, as a warning that bitstream buffer underflow is about to occur.

Each time a RDY-S event occurs, the microcode either issues a RDY-S interrupt or places a RDY-S interrupt in the pending interrupt queue.

A RDY interrupt event occurs when the following conditions are true:

- The timing for the decode-time interrupt is satisfied.
- The amount of data in the video bitstream rate buffer has fallen below the low threshold.
- The RDY interrupt is enabled (See the INT_MASK DRAM parameter).

SEQ-V

Event: First I-picture display after `sequence_start_code`
Category: Display-time
Mask Bit: 3

The SEQ-V interrupt event occurs at the rising edge of VSYNC prior to the first display field of the first I-picture decoded following the detection of a `sequence_start_code`. Each time a SEQ-V event occurs, the microcode either issues an SEQ-V interrupt or places an SEQ-V interrupt in the pending interrupt queue.

An SEQ-V interrupt event occurs if all of the following are true:

- The timing for display-time interrupts is satisfied.
- The picture to be displayed in the following field time is an I-picture that has not been previously displayed.
- The SEQ-V interrupt is enabled (See the INT_MASK DRAM parameter).

Event: Bitstream Buffer Underflow Error
Category: Decode-time
Mask Bit: 8

UND

The UND (bitstream buffer underflow) interrupt event occurs when the number of valid bytes in the bitstream buffer in DRAM is 0 while the CL480 microcode expects the bitstream buffer to contain data. Both audio and video bitstream buffers are checked while in the PLAY, DISPLAYSTILL or SCAN states. The video buffer is checked in the SLOWMOTION state, and the audio buffer is checked in the FREEZE state.

> *Note: This interrupt is generated within the CL480's internal timer interrupt routine and, thus, some underflows may go undetected due to the sampling interval of the timer interrupt. No underflow checks are performed during CD-DA play.*

This interrupt can be used by the host to detect a bitstream buffer underflow error condition. Each time a UND event occurs, the microcode either issues a UND interrupt or places a UND interrupt in the pending interrupt queue.

A UND interrupt event occurs if all of the following are true:

- The number of valid words in the bitstream buffer is 0, and a new word is needed by the decoding process.
- The UND interrupt is enabled (See the INT_MASK DRAM parameter).

  Note: The UND interrupt is sensitive to the buffer's emptiness being 0, not becoming 0. This means that, once a UND interrupt is produced, new UND interrupts will be produced continuously until the host takes steps to make the bitstream buffer non-empty.

USR

Event: User data ready
Category: Decode-time
Mask Bit: 12

The USR interrupt is generated after copying any user data from the MPEG-1 picture header into the user data FIFO. The USR interrupt is generated when the picture header is decoded.

MPEG-1 picture header user data is copied to the user data FIFO even if the USR interrupt is disabled.

No checks are performed to prevent the user data FIFO from overflowing. When an overflow occurs, all data in the user data FIFO is lost, including the word which caused the overflow. No indication of an overflow is available.

VSYNC-V

Name: VSYNC pulse occurred
Category: Display-time
Mask Bit: 6

The VSYNC-V interrupt indicates a video $\overline{\text{VSYNC}}$ pulse was generated in master mode or was input in slave mode.

The timing of the VSYNC-V interrupt relative to either edge of the $\overline{\text{VSYNC}}$ signal cannot be guaranteed, except that it will occur sometime after the leading edge of $\overline{\text{VSYNC}}$ and before the active region of the output display.

All VSYNC-V related interrupts are generated at the same time.

15 DRAM Configuration Reference

15.1 Modifying DRAM

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.

15.2 Contents of DRAM

Locations for the CL480 DRAM are shown below.

Note: Both word and byte addresses are given: The word address is used when accessing locations in the CL480's DRAM; the byte address is used when accessing locations in the CL480's ROM.

15.2.1 Status Area

The CL480 Status Area (described in Table 15-1) contains identifiers which include the following:

- Processing state (PROC_STATE)
- Most recent command identifier (MRC_ID)
- Most recent command status (MRC_STATUS)
- Interrupt status (INT_STATUS)

Table 15-1 Status Area

Parameter (word/byte addr)	Field Name	Field Value	Field Location	Description
PROC_STATE[1]				Indicates command state:
(0x60/0xC0)		0xFF		INITIALIZING
		0x00		IDLE
		0x09		SLOWMOTION
		0x0a		SCAN
		0x0b		SINGLESTEP
		0x0c		DISPLAYSTILL
		0x0d		PLAY
		0x0e		PAUSE
		0x10		FREEZE
		0x14		DUMPDATA
		0x1b		DISPLAYDIGEST
		0x1d		DIGESTREADY
MRC_ID[1]				Indicates last valid command:
(0x61/0xC2)		0x0000		(No previous command)
		0x0005		SetVideoFormat()
		0x0109		SlowMotion()
		0x000a		Scan()
		0x000b		Singlestep()
		0x000c		DisplayStill()
		0x020c		DisplayStill(start1, start2)
		0x030c		DisplayStill(romAddress1, romAddress2, length)
		0x040c		DisplayStill(start1, start2, stop1, stop2)
		0x000d		Play()
		0x040d		Play(start1, start2, stop1, stop2)
		0x000e		Pause()
		0x0010		Freeze()
		0x0213		SetStreams()
		0x0314		DumpData()
		0x0417		DisplayGraphics()
		0x021b		DisplayDigest(xOffset, yOffset, decimation)
		0x041b		DisplayDigest(xOffset, yOffset, start1, start2, decimation)
		0x001c		Replay()
		0x001d		DisplayBorder()
MRC_STATUS[1]				Indicates status of most recent command:
(0x62/0xC4)	WORKING_STATUS		0x00	Command identifier is valid and being processed.
	DONE_STATUS		0x01	Parsing and execution of command is complete.
	ERROR_STATUS		0x02	Error occurred in command execution.
	ADDR_OUT_OF_RANGE		0x00	Used only by DumpData() command. See DumpData().
	INVALID_BUF_ADDR		0x04	Used only by DumpData() command. See DumpData().
	INVALID_SECTOR_COUNT		0x08	Used only by DumpData() command. See DumpData().
	SECTOR_TYPE_ERROR		0x10	Used only by DumpData() command. See DumpData().

1. Access: read-only by host (CL480 writes)

Table 15-1: Status Area (Continued)

Parameter (word/byte addr)	Field Name	Field Location	Description
INT_STATUS[2] (0x63/0xC6)			Indicates what caused the interrupt
	ERR_INT	0x0001	Bitstream data error
	PICV_INT	0x0002	New picture display
	GOPV_INT	0x0004	First I-picture display after `group_start_code`
	SEQV_INT	0x0008	First I-picture display after `sequence_header_code`
	ENDV_INT	0x0010	Last picture display before `sequence_end_code`
	PICD_INT	0x0020	Picture decode complete
	VSYNC_INT	0x0040	VSYNC pulse occurred
	ACR_INT	0x0080	Address out of range
	UND_INT	0x0100	Bitstream buffer underflow error
	ENDC_INT	0x0200	High-priority command execution complete
	RDYS_INT	0x0400	Ready for data during SlowMotion()
	USR_INT	0x1000	User data ready
	ENDP_INT	0x2000	End of Pause(), or SingleStep() / SlowMotion() during buffer-full condition
	ENDD_INT	0x4000	End of DumpData() command
	AEE_INT	0x8000	Submode is auto-pause, end-of-record, or end-of-file
VIDEO_EMPTINESS[1] (0x64/0xC8) AUDIO_EMPTINESS[1] (0x65/0xCA)			Indicates the emptiness of the video and audio rate buffers, respectively, in 16-word blocks. These values are updated when the InquireBufferEmptiness() command is executed. If the END-C interrupt is enabled, the host will be notified of the update when it receives the END-C interrupt indicating the completion of the high-priority command.
VIDEO_FIELD (0x66/0xCC)			This field, valid only when the VSYNC-V interrupt occurs, indicates the field currently being output: zero equals the top field, and one equals the bottom field.
SE_STATUS[1] (0x67/0xCE)			This field, sector error status (CL480VCD version only), is used by the DumpData() command to indicate which sectors read were uncorrectable. Each bit position indicates a different sector: Bit 15 contains the status of the first sector read, bit 14 contains the status of the second sector read, etc. A zero in the bit position indicates the sector was not in error or the error was corrected, while a one indicates an error was detected and was uncorrectable.
LSA_MS[1] (0x68/0xd0)		0xff00 0x00ff	These fields (*last sector address_minutes/seconds* and *_frame/mode,* for CL480VCD microcode only) hold the sector address of the last sector whose data was copied into a bitstream buffer. The *Minutes* and *Frame* fields hold the upper byte of the word address, while the *Seconds* and *Mode* fields hold the lower byte. Since these fields are continually updated while receiving CD-ROM data, their contents are guaranteed to be valid when the CL480 is in the PAUSE state. These fields are not updated while executing the DumpData() command or while reading a CDDA bitstream, but they are updated while in the SLOWMOTION, SINGLESTEP and FREEZE states.
LSA_FM[1] (0x69/0xd2)		0xff00 0x00ff	
AEE_MS[1] (0x6a/0xd4)		0xff00 0x00ff	These fields (CL480VCD only) hold the sector address of the sector with the CD-ROM/XA trigger (auto-pause) bit set. These fields are valid when the A/E/E interrupt (if enabled) is generated and when the CL480 enters the PAUSE state (if enabled with the AUTO_PAUSE configuration parameter). If another auto-pause sector is received before the interrupt is processed or before the CL480 enters the PAUSE state, these fields will contain the address of the last sector received with the auto-pause bit set.
AEE_FM[1] (0x6b/0xd6)		0xff00 0x00ff	
FREE_SPACE_START[1] (0x6c/0xd8) FREE_SPACE_END[1] (0x6d/0xdA)			These parameters (CL480VCD microcode only) define the memory below four Mbits that is usable by the host processor. FREE_SPACE_START contains the address of the first unused DRAM word, and FREE_SPACE_END contains the address one beyond the last unused DRAM word. Each address found in these parameters is the actual DRAM address divided by 256; for example, the DRAM address 0x3c00 is stored as 0x3c.

1. Access: read-only by host (CL480 writes)
2. Access: read and write by both CL480 and host

Table 15-1: Status Area (Continued)

Parameter (word/byte addr)	Field Name	Field Value	Field Location	Description
NUM_DECODED[2] (0x1b7/0x36E)				This field is the count of the number of decoded pictures. The value is incremented at completion of picture decoding at the same time an END-D interrupt is generated if it is enabled. The counter wraps to zero if the count is 65535 and one more picture is decoded. The counter is incremented during Play(), DisplayStill(), Replay(), Slow-Motion(), SingleStep() and Scan() command execution. The count will be set to zero by a Reset() command.
NUM_SKIPPED[2] (0x1b8/0x370)				This field is the count of the number of pictures *skipped* due to audio/video synchronization. The count is incremented each time a picture is skipped. The count wraps to 0 when the count is equal to 65535 and one more picture is skipped. The counter is incremented while in the PLAY processing state, and is set to zero by a Reset() command.
NUM_REPEATED[2] (0x1b9/0x372)				This parameter is the count of the number of pictures repeated due to audio/video synchronization. The count is incremented each time a picture is repeated. If a single picture is repeated multiple times, the count will be incremented each time the picture is repeated. The count wraps to 0 when the count is equal to 65535 and one more picture is repeated. The counter is incremented while in the PLAY processing state, and is set to zero by a Reset() command.
INPUT_FORMAT[1,3] (0x6e/0xdC)	NOT_KNOWN CD_ROM_MPEG CDDA	0x00 0x02 0x04		Used with CL480VCD microcode only, this parameter specifies the type of bitstream being received. The contents of this parameter is updated when the configuration parameter PLAY_MODE is set to CD_ROM_AUTO. The contents will also be updated when a different bitstream input has been detected.
INT_SOURCE[2] (0x6f/0xdE)				This parameter indicates which source caused the END-P, EFR, ACR and A/E/E interrupts. Its contents are updated at the same time INT_STATUS is updated and should be cleared by the host at the same time INT_STATUS is cleared.
	ROM_ERROR[3]		0x0001	An error was detected while decoding the CD-ROM sector.
	MPEG_SYS_ERROR		0x0002	
	AUDIO_ERROR		0x0004	An error was detected in the audio bitstream.
	VIDEO_ERROR		0x0008	An error was detected in the video bitstream.
	ADDRESS_LOW[3]		0x0100	While executing a Play() or DisplayStill() command with arguments, a sector address was detected below the start address specified after detecting the start address
	ADDRESS_HI[3]		0x0200	While executing a Play() or DisplayStill() command with arguments, a sector address was detected greater than the stop address specified after detecting the start address.
	ENDP_PAUSE		0x0400	Generates END-P interrupt because of CL480 entering PAUSE state
	ENDP_BUFFER_FULL[3]		0x0800	Generates END-P interrupt because CL480 can't accept more data
	AUTO_PAUSE_EVENT[3]		0x2000	Interrupt caused when a AUTO-PAUSE CD sector was detected.
	EOR_EVENT[3]		0x4000	Interrupt caused when a END-OF-RECORD CD sector was detected.
	EOF_EVENT[3]		0x8000	Interrupt caused when a END-OF-FILE CD sector was detected.

1. Access: read-only by host (CL480 writes)
2. Access: read and write by both CL480 and host
3. For CL480VCD microcode only

Table 15-1: Status Area (Continued)

Parameter (word address)	Default Value	When Effective	Description
INTERVAL[1] (0x1ba/0x374)			This field is the time interval between successive picture frames. The units are the number of 90K system time clocks between successive picture frames.
VBS_SADR[1] (0x1c0/0x380)	0x006e	At Decode	These parameters hold the starting and ending 256-word page addresses for the video bitstream buffer. The buffer is implemented as a circular buffer with the VBS_SADR parameter being the low page address and the VBS_EADR parameter being one beyond the highest page address of the circular buffer.
VBS_EADR[1] (0x1c1/0x382)	0x00ce	At Decode	
ABS_SADR[1] (0x1c2/0x384)	0x0040	At Decode	These parameters hold the starting and ending 256-word page addresses for the audio bitstream buffer. The buffer is implemented as a circular buffer with the ABS_SADR parameter being the low page address and the ABS_EADR parameter being one beyond the highest page address of the circular buffer.
ABS_EADR[1] (0x1c3/0x386)	0x0054	At Decode	
AUDIO_SADR[1] (0x1c4/0x388)	0x0054	At Decode	These parameters hold the starting and ending 256-word page addresses for the audio output buffer. The buffer is implemented as circular buffer with the AUDIO_SADR parameter being the low page address and the AUDIO_EADR parameter being one beyond the highest page address of the circular buffer.
AUDIO_EADR[1] (0x1c5/0x38A)	0x006e	At Decode	

1. Access: read-only by host (CL480 writes).

15.2.2 High-Priority Command Area

The high-priority command area is the memory area where high-priority commands are written. To write a high-priority command into this memory area, first check that the CMD_ID field is zero, indicating the previous high-priority command has been processed. Once the CMD_ID field is zero, write the arguments into the memory area first. Afterwards, write the command ID to the CMD_ID field. The high-priority command processor uses the CMD_ID field to determine if a command is in the high-priority command area. Currently, there aren't any high-priority commands that require arguments.

Table 15-2 High-Priority Command Area

Parameter	Access	Word/byte Addr
CMD_ID	Read and write by host and CL480	0x70/0xE0
ARGUMENT1	Read-only by CL480, and read and write by host	0x71/0xE2
ARGUMENT2	Read-only by CL480, and read and write by host	0x72/0xE4
ARGUMENT3	Read-only by CL480, and read and write by host	0x73/0xE6
ARGUMENT4	Read-only by CL480, and read and write by host	0x74/0xE8
ARGUMENT5	Read-only by CL480, and read and write by host	0x75/0xEA
ARGUMENT6	Read-only by CL480, and read and write by host	0x76/0xEC
ARGUMENT7	Read-only by CL480, and read and write by host	0x77/0xEE

15.2.3 FIFO Pointers

The CL480 FIFO pointers contain field locations for the following:

- The command FIFO (see Table 15-4):
  - Start and end addresses: CMDF_SADDR and CMDF_EADDR
  - Read and write pointers: CMDF_READ and CMDF_WRITE
- The user data FIFO (see Table 15-4):
  - Start and end addresses: UDF_SADDR and UDF_EADDR
  - Read and write pointers: UDF_READ and UDF_WRITE

Table 15-3 Command FIFO Pointers

Parameter Type	Parameter	Word/byte Address	Definition
Command FIFO			This memory area is where the host processor writes low-priority commands for CL480 execution. See Section 13.2.1 for further details on how to use the command FIFO. See Section 13.1 for discussion of low-priority commands.
	CMDF_SADDR[1]	0x78/0xF0	These two locations contain the starting and ending addresses of the low-priority macro command FIFO. The starting address is less than the ending address. The starting address points to the first word of the command FIFO, while the ending address points to the location one *beyond* the last word of the command FIFO. The command FIFO is implemented as a circular buffer with the CMDF_READ and CMDF_WRITE fields being the read and write pointers, respectively.
	CMDF_EADDR[1]	0x79/0xF2	

Table 15-3 Command FIFO Pointers

Parameter Type	Parameter	Word/byte Address	Definition

	CMDF_READ[1]
	CMDF_WRITE[2]
		0x7c/0xF8
		0x7d/0xFA
			These two locations contain the read and write pointers for the command FIFO. The command FIFO is implemented as a circular buffer with CMDF_SADDR and CMDF_EADDR specifying its start and end. The read pointer is used by the CL480 to fetch commands from the command FIFO, while the host uses the write pointer to place commands into the command FIFO. The host processor should not modify the contents of the read pointer. The host processor should update the write pointer only *after* writing all words necessary for the command into the command FIFO. The value of the write pointer should always be greater than or equal to CMD_SADDR and should always be less than CMD_EADDR. The value written to the write pointer should point one *beyond* the last word of the command written. See the example below:
			
			`char *write_pointer`
			`read CMDF_WRITE to write pointer`
			`loop for each word of command`
			`write next command word to command fifo`
			`increment write pointer`
			`if write pointer equal to CMDF_EADDR`
			`set write pointer to CMDF_SADDR`
			`end if`
			`end loop`
			`write write pointer to CMDF_WRITE`
			
			The above procedure assumes there is enough room in the command FIFO for the command before the command is written to the FIFO. The host processor needs to ensure that enough room is present in the command FIFO before writing the command and its arguments.
			
			The command FIFO is empty when the read and write pointers are equal to each other. The command FIFO is full when the write pointer is *one less* than the read pointer taking into account buffer wrap around.

1. Access: Read-only by host
2. Access: Read and write by host

Table 15-4 User Data FIFO Pointers

Parameter Type	Parameter	Word/byte Address	Definition
User Data FIFO			This memory area is where the CL480 microcode writes user_data found in MPEG-1 picture headers. See Section 13.2.2 for further details on how to use the user data FIFO.
	UDF_SADDR[1] UDF_EADDR[1]	0x7a/0xF4 0x7b/0xF6	These two locations contain the starting and ending addresses of the user data FIFO. The starting address is less than the ending address. The starting address points to the first word of the user data FIFO while the ending address points to the location one *beyond* the last word of the user data FIFO. The user data FIFO is implemented as a circular buffer with the UDF_READ and UDF_WRITE fields being the read and write pointers, respectively.
	UDF_READ,[1] UDF_WRITE[2]	0x7e/0xFC 0x7f/0xFE	These two locations contain the read and write pointers for the user data FIFO. The user data FIFO is implemented as a circular buffer with UDF_SADDR and UDF_EADDR specifying where the circular buffer resides in memory. The write pointer is used by the CL480 to write data into the user data FIFO, while the host processor uses the read pointer to read data from the user data FIFO. The host processor should not modify the contents of the write pointer. The FIFO is empty if the read pointer *equals* the write pointer and is full if the write pointer is *one less* than the read pointer considering FIFO wrap around. If the host doesn't read the data out of the FIFO before it overflows, the FIFO contents will be flushed. Filling of the FIFO will begin with the next byte following the byte which caused the overflow. The user data FIFO is large enough to hold four Video CD `user_data` packets. Therefore, in this application, the host processor has 120 ms to read the contents of the user data FIFO before overflow occurs. This time is for 30 pictures per second. For picture rates slower than 30 per second, more time is available for reading the user data FIFO contents.

1. Access: Read-only by host
2. Access: Read and write by host

15.2.4 ROM Header and Subheader

ROM header and subheader parameters, shown in Table 15-5, are present only in the CL480VCD microcode version.

Table 15-5 ROM Header and Subheader

Parameter (word/byte addr)	Field Name	Field Location	Access	Definition
M_SEC[1]	Minutes	0xff00	Read-only by host	These fields contain the CD-ROM header and sub-
(0x198/0x330)	Seconds	0x00ff	Read-only by host	header information for the last sector read by the CL480 input process. These values are updated be-
F_MODE[1]	Frame	0xff00	Read-only by host	fore any checks are made for the proper sector type.
(0x199/0x332	Mode	0x00ff	Read-only by host	Since these values are continuously updated while receiving CD-ROM data, the contents are guaran-
FC_NUM[1]	File_number	0xff00	Read-only by host	teed to be valid when the CL480 is in the PAUSE
(0x19a/0x334)	Channel_number	0x00ff	Read-only by host	state. These fields are not updated while executing the DumpData() command or while reading a CDDA
SMC_INFO[1]	Submode_info	0xff00	Read-only by host	bitstream, but they are updated while in the SLOW-
(0x19b/0x336)	Coding_info	0x00ff	Read-only by host	MOTION, SINGLESTEP and FREEZE states.

1. Access: Read-only by host

15.2.5 MPEG-1 Header Parameters

MPEG-1 header parameters and fields (contained in Table 15-6) are provided for monitoring the MPEG-1 decoding process. All of these fields are copied from or derived from the corresponding MPEG-1 header information and are not to be modified by the host. Changes to these fields will not have any effect on CL480 operation and will be overwritten when new values for the fields are decoded from the incoming bitstream.

Table 15-6 MPEG-1 Header Parameters

Parameter Type	Parameter	Word/byte Address	Definition
	H_SIZE[1]	0x1a0/0x340	horizontal_size
	V_SIZE[1]	0x1a1/0x342	vertical_size
	PA_RATIO[1]	0x1a2/0x344	pel_aspect _ratio
	PIC_RATE[1]	0x1a3/0x346	picture_rate
Sequence header[2]	BIT_RATE_H[1]	0x1a4/0x348	bit_rate (upper 15 bits)
	BIT_RATE_L[1]	0x1a5/0x34A	bit_rate (lower 2 bits)
	VBV_BSIZE[1]	0x1a6/0x34C	vbv_buffer_size
	CONS_FLAG[1]	0x1a7/0x34E	constrained_parameter_flag
	LIQ_MATRIX[1]	0x1a8/0x350	load_intra_quantizer_matrix
	LNIQ_MATRIX[1]	0x1a9/0x352	load_non_intra_quantizer_matrix

1. Access: read-only by host
2. These parameters are loaded from the corresponding fields of the MPEG-1 sequence header when it is decoded.

Table 15-6 MPEG-1 Header Parameters (Continued)

Parameter Type	Parameter	Word/Byte Address	Field Value	Definition
GOP header[2]	TIME_CODE_H[1]	0x1aa/0x354		time_code (upper 12b + 1b marker)
	TIME_CODE_L[1]	0x1ab/0x356		time_code (lower 12 bits)
	CLOSED_GOP[1]	0x1ac/0x358		closed_gop
	BROKEN_LINK[1]	0x1ad/0x35A		broken_link
Picture header[3]	TEMP_REF[1]	0x1ae/0x35C		temporal_reference
	PIC_TYPE[1]	0x1af/0x35E		picture_coding_type
	VBV_DELAY[1]	0x1b0/0x360		vbv_delay
	FULL_PEL_FOR[1]	0x1b1/0x362		full_pel_forward_vector
	FWD_CODE[1]	0x1b2/0x364		forward_f_code
	FULL_PEL_BCK[1]	0x1b3/0x366		full_pel_backward_vector
	BWD_CODE[1]	0x1b4/0x368		backward_f_code
Audio header[4]				The first and second fields contain the first and second 16 bits of the MPEG-1 audio header, respectively, as shown below:
	AUDIO_HEADER1[1]	0x1bb/0x376	0xfff	syncword
			0x1000	ID
			0x6000	layer
			0x8000	protection_bit
	AUDIO_HEADER2[1]	0x1bc/0x378	0x000f	bitrate_index
			0x0030	sampling_frequency
			0x0040	padding_bit
			0x0080	private_bit
			0x0300	mode
			0x0c00	mode_extension
			0x1000	copyright
			0x2000	original/home
			0xc000	emphasis

1. Access: Read-only by host
2. These parameters are loaded from the corresponding fields of the MPEG-1 group-of-pictures header when it is decoded.
3. These parameters are loaded from the corresponding fields of the MPEG-1 picture header when it is decoded.
4. These parameters are loaded from the corresponding fields of the MPEG-1 audio header when it is decoded.

15.2.6 Configuration Parameters

CL480 Configuration Area parameters are shown in Table 15-7. These DRAM locations contain parameters which can be modified to change the behavior of the CL480. Changes to different configuration parameters take effect at the four different times shown below:

- *Start*: The value is examined after the microcode is loaded and before the processing state first transitions from INITIALIZATION to IDLE.
- *Decode:* When MPEG-1 decoding is initiated by the CL480 microcode executing a Play() or DisplayStill() command, and the microcode transitions from IDLE to any state other than DUMPDATA.
- *Video Initialization*: When the video unit is initialized, when MPEG-1 decoding begins, or when a SetVideoFormat() command is executed. Thus *video initialization* is equivalent to *start*, *decode* and SetVideoFormat() execution.
- *Immed:* Immediately. Parameters specified as being *Immed.* will take effect the next time the CL480 performs the related operations.

  Note: All parameters in Table 15-7 are readable and writable by the host but are read-only by the CL480.

Table 15-7 Configuration Parameters

Parameter (word/byte addr) (default value)	When Effective	Field Name	Field Location	Definition
FIRST_FIELD (0x1c6/0x38C)	Video initialization			In the field-type determination algorithm, this parameter indicates which $\overline{\text{VSYNC}}$ and $\overline{\text{HSYNC}}$ timing relationship is the even or odd field. This parameter is only used when hardware external to the CL480 is supplying the $\overline{\text{HSYNC}}$ and $\overline{\text{VSYNC}}$ signals. See Section 7.6.4 for more information.
UNUSED_IO (0x1ca/0x394)	Startup			This parameter specifies whether the unused programmable I/O pins will be inputs or outputs. If this parameter is set to one, the unused I/O pins will be configured as outputs and driven to a high logic level; if set to zero, the ususued I/O pins will be configured as inputs. The programmable I/O pins are as follows: pin 126 (FSC4), pin 124, (FS1), pin 120 (CDG-SDATA), pin 117 (CDG-VFSY), and pin 114 (CDG-S0S1).

Table 15-7 Configuration Area Parameters (Continued)

Parameter (word/byte addr) (default value)	When Effective	Field Name	Field Location	Definition
AUDIO_CONFIG (0x1cb/0x396) (0xd130)	Immediate			This parameter configures how the CL480 hardware outputs audio data and is examined whenever a new audio header is decoded.
		LEFT_LRCK_HI	0x08	When set, left channel (channel 0 in bitstream) is HIGH.
		RIGHT_LRCK_LO	0x04	When set, right channel (channel 1 in bitstream) is HIGH.
		OUTPUT_24_BITS	0x02	When set, 24 DA-BCK's are used to output each 16-bit audio sample. The audio sample is present on the last 16 DA-BCK's of the 24 generated. Either the most or least significant bit of the sample is present for the first eight clocks of the 24 generated. Which bit is outputted first depends upon which bit of the sample is output first. If the OUTPUT_24_BITS field is zero, 16 DA-BCK's are used to output each sixteen bit audio sample. (See Table 8-2.)
		LSB_FIRST	0x01	When set, the least significant bit of the audio sample is output first, otherwise the most significant bit is output first. This bit also determines which bit is output during the first eight DA-BCK's when the OUTPUT_24_BIT field is set. When the LSB_FIRST field is set, the least significant bit is output during the eight DA-BCK's, otherwise the most significant bit is output.
CD_CONFIG (0x1cc/0x398) (0x0216)	At Decode			This parameter is provided in the CL480VCD microcode only and is used to configure how the CL480 hardware reads the input CD data.
		LENGTH	0x30	Specifies the number of CD-BCKs per 16 bits of CD data input using four values as listed below: ■ 00 = 32 CD-BCK's for every 16-bit data word input (data is on the last 16 CD-BCK's) ■ 01 = 16 CD-BCK's for every 16-bit data word input (one CD-BCK per bit of CD data input) ■ 10 = 24 CD-BCK's for every 16-bit data word input (data is on the last 16 CD-BCKs) ■ 11 = 24 CD-BCK's for every 16-bit data word input (data is on the first 16 CD-BCKs)
		DATA_LSB_FIRST	0x08	When 1, the LSB of the CD data is input first; otherwise, the MSB is input first. This bit is set to 0 after receiving the RESET signal.
		C2PO_LSB_FIRST	0x04	When 1, the LSB of CD C2PO is input first; otherwise, the MSB is input first.
		LRCK_RCH	0x02	When 1, *right* channel CD-DATA is input when LRCK is HIGH; otherwise, the left channel CD-DATA is input.
		BCK_FALLING	0x01	When 1, the CD-DATA input is sampled with the *falling* edge of CD-BCK; otherwise, the rising edge is used to sample the data.

Table 15-7 Configuration Area Parameters (Continued)

Parameter (word address) (default value)	When Effective	Field Name	Field Location	Definition
VIDEO_MODE2 (0x1cd/0x39A) (0x000c)	At Start			This parameter contains the initial value for the video unit's VSYNC/CSYNC setting.
		RGB_15		Only valid in the Silicon Revision B chips and later versions. When 1, the video output uses five bits each for red, blue, and green color signals.
		INTERLACE	0x08	This field is only valid when the CSYNC_VSYNC bit is 1; it specifies whether or not the video output is to be interlaced.
		CSYNC_VSYNC	0x04	This field is used to determine whether the CSYNC or VSYNC signal is output to the VSYNC pin. When 1, CSYNC is output. CSYNC can only be an output, never an input.
ROM_CONFIG (0x1ce/0x39C) (0x0007)	At Start			This parameter is provided in the CL480 VCD microcode only. It specifies the number of GCKs to use when accessing the attached ROM.
		ROM_ACCESS	0x1f	Specifies ROM access time in GCK cycles: This time = (ROM_ACCESS + 2) x GCK period. Thus, CL480 will support ROM access times of from 2 to 33 GCK periods (50 ns to 825 ns for a clock frequency of 40 MHz).
DRAM_REFRESH (0x1cf/0x39E)				Specifies the number of GCKs between successive refresh cycles.
		COUNT	0x1fff	This is the number of GCK's between refreshes. This value is used to initialize the refresh counter when the CL480 hardware is reset and whenever the refresh timer decrements to zero. Whenever the refresh counter decrements to zero, a DRAM refresh cycle is performed. The default value of 512 causes a DRAM page to be refreshed every 12.8 us (512/40,000,000).
AUDIO_AVERAGE (0x1d0/0x3A0) (0x0000)	Immed.			This parameter controls audio averaging and is examined whenever an audio frame is decoded, but is not valid for CD-DA bitstreams.
		AVERAGE	0x01	When the AVERAGE field is non-zero, the left and right audio output samples are averaged. The result is output to both the left and right audio channels.

Table 15-7 Configuration Area Parameters (Continued)

Parameter (word/byte addr) (default value)	When Effective	Field Name	Field Location	Description
WEIGHTK0 (0x1d1/0x3A2) (0x0198) WEIGHTK1 (0x1d2/0x3A4) (0x0730) WEIGHTK2 (0x1d3/0x3A6) (0x079c) WEIGHTK3 (0x1d4/0x3A8) (0x0204) WEIGHTK4 (0x1d5/0x3AA) (0x0129)	Video Initialization			These parameters are used in the conversion of YUV video encoding to RGB video encoding using the following equation: R = K4 * (Y - N) + K0 * Cr G = K4 * (Y - N) - K1 * Cr - K2 * Cb B = K4 * (Y - N) + K3 * Cb where K0 is WEIGHTK0, K1 is WEIGHTK1, K2 is WEIGHTK2, K3 is WEIGHTK3, and K4 is WEIGHTK4. WEIGHTK0 through WEIGHTK3 are encoded in 11 bits as follows: Bit 10 -> sign bit Bits 9-8 -> integer part Bits 7-0 -> fractional part WEIGHTK4 is encoded in eight bits as follows: Bit 8 -> integer part Bits 7-1 -> fractional part
VIDEO_MODE (0x1d7/0x3AE) (0x0b24)	At Start			This parameter selects the video mode operation. When modifying this parameter, bits 11 and 9 must contain a one, while bits 15 through 12 and bit 1 must contain a zero.
		VRANGE	0x0100	Specifies whether 16 is subtracted from the luminance in converting from YUV to RGB: 1 = subtract 16; 0 = no-adjust. See WEIGHT0-4 descriptions on previous page.
		VCLK_OUT	0x0080	Specifies the direction of the VCK signal: 1 = the internally generated signal is output on the VCK pin; 0 = the VCK signal is an input supplied from an external source using the VCK pin.
		VBUS_MODE	0x0040	Only valid in YUV operation. Specifies the video bus width: 1 = the video bus width is 16 bits; 0 = 8 bits.
		H_INTERP	0x0010	When set to 1, horizontal interpolation is disabled. If set to 0, horizontal interpolation by two is enabled.
		RGB_MODE	0x0004	Specifies whether the video unit should be in RGB or YUV mode: 1 = the video unit outputs RGB mode; 0 = YCbCr.
		SYNC_OUT	0x0001	Specifies the direction of the $\overline{\text{VSYNC}}$ and $\overline{\text{HSYNC}}$ signals: 1 = output (both are generated by the video unit); 0 = input.

Table 15-7 Configuration Area Parameters (Continued)

Parameter (word/byte addr) (default value)	When Effective	Field Name	Field Value	Description
CCLCR_CR[1] (0x1da/0x3b4) (0x0080)	Video Initialization			This parameter specifies the Cr chrominance value of the YCbCr color of the border color using the least significant eight bits.
CCLCR_Y_CB[1] (0x1db/0x3b6) (0x1080)	Video Initialization			This parameter specifies the luminance, Y, and the chrominance, Cb, values of the YCbCr color used for the border. The Y value is the MSB, while the Cb value is the LSB.
NTSC_HSYNC_LO[1] (0x1dc/0x3b8) (0x004f)	Video Initialization			In NTSC mode, this parameter specifies the time (in VCK periods) that HSYNC is LOW when the CL480 is generating HSYNC and VSYNC.
NTSC_HSYNC_HI[1] (0x1dd/0x3bA) (0x0359)	Video Initialization			In NTSC mode, this parameter specifies the time (in VCK periods) that HSYNC is HIGH when the CL480 is generating HSYNC and VSYNC.
PAL_HSYNC_LO[1] (0x1de/0x3bC) (0x004f)	Video Initialization			In PAL mode, this parameter specifies the time (in VCK periods) that HSYNC is LOW when the CL480 is generating HSYNC and VSYNC.
PAL_HSYNC_HI[1] (0x1df/0x3bE) ((0x035f)	Video Initialization			In PAL mode, this parameter specifies the time (in VCK periods) that HSYNC is HIGH when the CL480 is generating HSYNC and VSYNC.
VSYNC_ODD_TP[1] (0x1d8/0x3b0) (0x0180)	Video Initialization			This parameter specifies the position relative to HSYNC going LOW that VSYNC will go LOW for odd fields only of the bottom field. This parameter is used for both PAL and NTSC.
VSYNC_EVEN_TP[1] (0x1d9/0x3b2) (0x0000)	Video Initialization			This parameter specifies the position relative to HSYNC going LOW that VSYNC will go LOW for even fields only of the bottom field. This parameter is used for both PAL and NTSC.
PLAY_MODE[1, 2] (0x1e0/0x3C0) (0x0000)	At Decode	CD_ROM_AUTO[2]	0x00	Automatically switches between CD-ROM video and CD-DA.
		HOST_MPEG_SYSTEM	0x01	All data is interpreted as an MPEG-1 constrained parameter bitstream.
		CD_ROM_MPEG[2]	0x02	All data is interpreted as CD-ROM video bitstreams.
		CD_ROM_CDDA[2]	0x04	All data is interpreted as CDDA bitstreams.
STILL_MODE[1] (0x1e1/0x3C2) (0x0001)	At Decode			Causes the video to be blanked before displaying the next still picture, and it is only valid for still pictures. If reset, video blanking is performed.
AUTO_PAUSE[1] (0x1e2/0x3C4) (0x0000)	Immed.			Specifies if auto-pause feature is enabled or disabled. If 1, auto-pause is enabled.

1. Access: Write-only (CL480 will write only during initialization).
2. For CL480VCD microcode only.

Table 15-7 Configuration Area Parameters (Continued)

Parameter (word/byte addr) (default value)	When Effective	Field Name	Field Value	Field Location	Description
CDDA_EMPHASIS[1] (0x1e3/0x3C6) (0x0000)	Immed.				This parameter is present in the CL480VCD microcode only and is examined when a CDDA bitstream is input (at the same time the CDDA_EMP signal is examined). See Table 8-3 for more info.
		EMP_FOLLOWS_INPUT		0x0002	When set on B0 or greater silicon, the DAC_EMP signal will follow the CDDA_EMP signal.
		SET_EMP_ON		0x0001	Indicates value of DAC_EMP output signal for A3 or B0 or greater silicon when the EMP_FOLLOWS_INPUT field in 0.
AV_SYNC_MODE[3] (0x1e4/0x3C8) (0x0001)	Immed.	NOSYNC	0x00		No synchronization is performed. All packets will be displayed or sounded as they are received and decoded.
		SYNC_AUDIO_PTS	0x01		When set, synchronization is to the PTS of the audio packets.
		SYNC_SCR	0x02		When set, the system time clock is loaded with the SCR value of every pack header. When AV_SYNC_MODE is a value greater than SYNC_SCR, the system time clock is loaded with the SCR value from the pack header every AV_SYNC_MODE pack header.
ERROR_LEVEL[1] (0x1e5/0x3CA) (0x0001)	Immed.			0x0001	This parameter is examined when an error occurs while in the DISPLAYSTILL processing state. When this parameter is zero and an error is detected, the CL480 microcode transitions to the PAUSE processing state and the picture containing the error is not displayed. When this parameter is non-zero and an error is detected, the picture or audio data is presented using error concealment, and decoding continues. (See description of error handling in Section 11.3.)
INT_MASK[1] (0x1e6/0x3CC) (0x0000)	Immed.	ERR_INT		0x0001	Specifies the interrupts to be generated by the CL480 to the host processor. A non-zero value in any bit position enables generation of the corresponding interrupt. See Chapter 14 for descriptions of each interrupt.
		PICV_INT		0x0002	
		GOPV_INT		0x0004	
		SEQV_INT		0x0008	
		ENDV_INT		0x0010	
		PICD_INT		0x0020	
		VSYNC_INT		0x0040	
		ACR_INT		0x0080	
		UND_INT		0x0100	
		ENDC_INT		0x0200	
		RDYS_INT		0x0400	
		USR_INT		0x1000	
		ENDP_INT		0x2000	
		ENDD_INT		0x4000	
		AEE_INT		0x8000	

1. Access: Write-only (CL480 will write only during initialization).
3. For CL480PC microcode only.

Table 15-7 Configuration Area Parameters (Continued)

Parameter (word/byte addr) (default value)	When Effective	Field Name	Field Value	Field Location	Description
AUDIO_MUTE (0x1e7/0x3CE) (0x0000)	Immed.	NO_AUDIO_MUTE MUTE_AUDIO ATTENUATE_AUDIO	0x00 0x01 0x02		This parameter specifies the three attenuation settings used for audio output: no attenuation, -12db, and no audio output. The value of this parameter is examined each time an audio frame is decoded.
VA_STREAM_ID (0x1e8/0x3d0) (0xe0c0)	Startup	VIDEO_STREAM_ID AUDIO_STREAM_ID		0xff00 0x00ff	This parameter is examined each time a packet header is parsed and specifies the stream identifiers used to extract the video and audio bitstreams from the input system bitstream. All packets whose stream identifier doesn't match either the video and audio stream identifiers specified by this parameter are ignored.
EVEN_INTERPCOEF (0x1e9/0x3d2) (0x0025)	Immed.	VLUM_BOT_LINE VLUM_TOP_LINE VCHR_BOT_LINE VCHR_TOP_LINE HLUM_BOT_LINE HCHR_TOP_LINE		0x0003 0x000c 0x0030 0x00c0 0x0300 0x0c00	These parameters define the vertical and horizontal interpolation coefficients for the video output process used for the *even* video field.
ODD_INTERPCOEF (0x1ea/0x3d4) (0x008f)	Immed.	VLUM_BOT_LINE VLUM_TOP_LINE VCHR_BOT_LINE VCHR_TOP_LINE HLUM_BOT_LINE HCHR_TOP_LINE		0x0003 0x000c 0x0030 0x00c0 0x0300 0x0c00	These parameters define the vertical and horizontal interpolation coefficients for the video output process used for the *odd* video field. Each of the vertical fields (prefixed by VLUM_ or VCHR_) can take any of the following values: ■ 0 = INTERP_ZERO ■ 1 = INTERP_ONE_QUARTER ■ 2 = INTERP_ONE_HALF ■ 3 = INTERP_THREE_QUARTERS Each of the horizontal fields (prefixed by HLUM_ or HCHR_) can take any of the following values: ■ 0 = INTERP_ZERO ■ 1 = INTERP_ONE_EIGHTH ■ 2 = INTERP_ONE_QUARTER ■ 3 = INTERP_ONE_HALF The new values will take effect when the next VSYNC pulse occurs. See Section 7.6.7 for more details on how interpolation is performed.
VIDEO_FORMAT (0x1eb/0x3d6) (0x0004)	Startup	NO_REPEAT MULTI_SYNC PAL_FORMAT NTSC_FORMAT	0x00 0x01 0x03 0x04		This parameter specifies the initial video output mode, with four valid values as shown. This parameter is examined at video initialization and is not examined again.

Table 15-7 Configuration Area Parameters (Continued)

Parameter (word/byte addr) (default value)	When Effective	Description
LEFT_BORDER (0x1ec/0x3dA) (0x0000)		This parameter only has effect when using the Silicon Revision B chip. This parameter specifies the number of VCK's from the start of the HSYNC pulse and the start of video output. If zero, the display is centered in the horizontal direction based upon the number of VCK's between HSYNC pulses.
TOP_BORDER (0x1ee/3dC) (0x0000)		This parameter specifies the number of blank lines between the start of the VSYNC pulse and the first line of the display. If zero, the display is centered vertically based upon the number of lines for each video output format.
XOFFSET (0x1ef/0x3dE) (0x0000) YOFFSET (0x1f0/0x3E0) (0x0000)		These parameters only have effect when using the Silicon Revision B chip. These parameters specify which pixel of the picture to be shown is to be displayed at the point specified by the LEFT_BORDER and TOP_BORDER parameters. By default, the upper left-hand corner of the picture is displayed at the point specified by LEFT_BORDER and TOP_BORDER.
WIDTH (0x0f1/0x1E2) (0x0000) HEIGHT (0x0f2/0x1E4) (0x0000)		These parameters only have effect when using the Silicon Revision B chip. These parameters specify the width and height, respectively, of the active video output area. The upper-left corner of this area is specified by the TOP_BORDER and LEFT_BORDER parameters. By default, the width and height parameters are read from the incoming bitstream (both parameters set to zero).
HORIZONTAL_SIZE (0x1f3/0x3E6) (0x0160) VERTICAL_SIZE (0x1f4/0x3E8) (0x00f0) PICTURE_RATE (0x1f5/0x3EA) (0x0004) S_HORIZONTAL_SIZE (0x1f6/0x3EC) (0x02c0) S_VERTICAL_SIZE (0x1f7/0x3EE) (0x01e0)	Immed.	These parameters specify the values to use for the corresponding fields of the MPEG-1 sequence header if the sequence header contains an error. These parameters are examined whenever an error occurs in the sequence header. These fields are as defined in the MPEG-1 specification, and they are in the same units.
DATE_VERSION (0x1fa/0x3F4)		This field contains the ASCII representation of when this version of microcode was released, including the version number. The field is formatted as ddMmmyy/vvv. For example, version 1.10 released on Jan. 1, 1993 would appear in ASCII as '01Jan93/1.10'.

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. Two design examples are shown below.

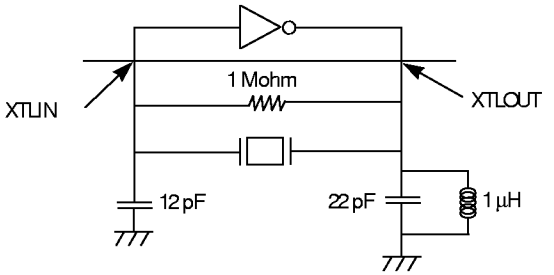


Figure A-1 Crystal Design Example Number 1

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

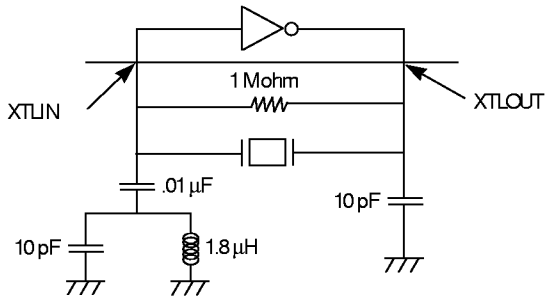


Figure A-2 Crystal Design Example Number 2

DAISHINKU (KDS AMERICA) CORPORATION
17151 Newhope Street,
Suite 210,
Fountain Valley, California USA 92
Phone: 714-557-7833
FAX: 714-557-4315

STANDARD CRYSTAL, INC.
Phone: 1-800-423-4578

Appendix B
480PC Microcode File Format

This appendix describes the C-Cube Microsystems host-based binary "microcode executable" file format (*.UX) in tabular (see Table B-1) and C-structure (see Section B.2) format, with CL480-specific file items listed in Table B-2. All specified multi-byte values are stored in the file as "little-endian," with the least-significant eight bits of each value stored in the first byte present in the file.

> *Note: Addresses specified in the file format have a product-specific format. That is, they may refer to 16-bit, 32-bit or some other size quantities that are native to the microcode linker and loader. Data to be loaded onto the chip is also product-specific in size and endian.*

B.1 File Element Description

Table B-1 describes each of the .UX file parameters.

Table B-1 CL480PC Binary File Format

File Element	Definition
Initial header	This header contains the minimum information necessary to load the microcode.
■ Magic number (32 bits)	This is the 32-bit value 0x58553343, which is equivalent to the ASCII string "C3UX" (without null termination).
■ Number of sections (32 bits long)	This is the total number of "Sections" (defined on the next page) contained in the file.
■ Length of initial portion (32 bits long)	The length of the initial portion is the length of that part of the file which is always distributed to customers. This is specified as a number of bytes. Optional information may be present at the end of the file beyond this length, but it is not needed for normal operation.

Table B-1 CL480PC Binary File Format (Continued)

File Element	Definition
Section	Each Section contains part of the microcode that is to be loaded into a contiguous Section of the CL480's memories. All Sections must be contiguous within the file and occur between the two headers. Data from each Section *must* be loaded to the CL480 in the same order they occur in the file.
■ Data type (1 byte)	This is the memory space in which this Section loads. The predetermined values are: 0 IMEM 1 DRAM 2 GBUS (registers)
■ Flags (1 byte)	Bit 0: ignore checksum. If bit 0 is 0, the checksum (see below) for this Section is valid. If 1, it is invalid and should be ignored by the microcode loader. This bit is set by the microcode linker for only those Sections intended for customer modification. It should not be modified by customers. Bit 1: Section format. If bit 1 is 0, the Section data consists of a block of data loaded at the Section start address. If it is 1, the Section data consists of address and data pairs, where the address is a 32-bit little-endian quantity, and the data format varies based on the Data type. The loader determines how many bits of each quantity are valid. Bits 2-7: unused (will be 0).
■ Unused flags (2 bytes)	Will be 0.
■ Section length (32 bits long)	This is the length of this Section in bytes.
■ Section start (32 bits long)	This is the product-specific address within the selected memory space for the first data word in the segment. Other data words must be loaded sequentially at increasing addresses. If Flags[1] is 1, then this value is unused and will be 0 (see above).
■ Section checksum (32 bits long)	This is a simple arithmetic checksum of the Section. It is ignored by the loader if Flags[0] is 1 (see above). It is computed as the sum of each *byte* in the Section, including all fields of the header except the checksum itself.
■ Section data	This variable-length field contains the data values actually written to the selected memory space (Data type). The contents of this field are product-dependent, and may contain 32-bit address values if Flag[1] is 1.

Table B-1 CL480PC Binary File Format (Continued)

File Element	Definition
Extended header	This header is distributed to customers, but is not needed for production loads of microcode. All strings are ASCII characters and are padded with zeroes.
■ File format revision (2 bytes)	The first byte contains the minor revision code for the file format. Changes to the minor revision indicate changes only to the C-Cube private portions of a file (debugging data, below) or changes which don't affect the basic load mechanism. The second byte is the major revision for the format. The description in this appendix matches a major revision value of 1.
■ Microcode version number (4 bytes)	ASCII string (no null termination) indicating the version number of the microcode contained in this file.
■ Microcode product name (32 bytes)	Text identifying the microcode product contained in the file, such as "480VCD" or "CLM4500."
■ Customer specific name (32 bytes)	Text describing customer-specific variant, or all zeroes if none.
■ Silicon product name (16 bytes)	Name on string such as "CL480" or"CL4000."
■ Minimum silicon revision (1 byte) ■ Maximum silicon revision (1 byte)	This is the minimum silicon mask revision needed to run this microcode, and the maximum revision (if any) past which this microcode will not work. If there is no maximum revision field, the maximum revision is 0. Silicon revisions are encoded as four bits for the major revision (all layer change) and four bits for the minor revision (metal mask only). 0x10 represents an "A0" mask revision.
■ Minimum extended feature revision (1 byte) ■ Maximum extended feature revision (1 byte)	These are the minimum silicon revisions needed for any conditional or extended features supported by this microcode, and the maximum silicon revision needed to run all extended features supported by this microcode. The microcode will have to examine the revision itself to decide which features can be enabled.
■ Copyright string (64 bytes)	This field contains the microcode's copyright notice.
■ Header checksum (32 bits long)	This is a simple arithmetic checksum of the contents of the Initial header and the other fields of the Extended header.
Debug data (optional)	The data following this is not counted by the length in the initial header, and is used exclusively by C-Cube development tools. It is not meant that this part of the file will be routinely shipped to customers.

B.2 C-Structure Description

The following is a C-structure description of the file items contained in Table B-1.

```
typedef unsigned long ulong; /* 32 bits */
typedef unsigned char uchar; /* 8 bits */

/*This structure describes the Initial header of the file.*/
struct init_hdr {
  char magic_num[4];
  ulong num_sections;
  ulong init_length;
}
/*This structure describes the Sections contained in the file.*/
struct Section {
  uchar data_type;
  uchar section_flags; /*No bitfields because of portability concerns */
  uchar unused[2];
  ulong section_length;
  ulong section_addr;
  ulong section_checksum;
  ulong section_data[];
}
/*The Section flags could be defined with the following structure: */
struct section_flags {
  ignore_cksum : 1;
  section_format : 1;
  unused_bits : 6;
}

/*This structure describes the Extended header of the file.*/
struct extend_hdr {
  uchar file_minor_rev;
  uchar file_major_rev;
  char ucode_rev[4]; /*no null termination*/
  char ucode_name[32];
  char cust_name[32];
  char chip_name[16];
  uchar min_revision;
  uchar max_revision;
  uchar min_ext_revision;
  uchar max_ext_revision;
  char copyright[64];
  ulong header_checksum;
}

/*In MPEG format:*/
microcode_file () {
   init_hdr
   for (i = 0; i < num_segs; i++) {
      Section
   }
   extend_hdr
   private_data /*C-Cube private_data may be released for debug purposes.*/
}
```

B.3 CL480-Specific File Items

File items that are specific to the CL480 are shown in Table B-2.

Table B-2 CL480-Specific File Items

Memory Space	Data Word Size	How stored in File	Max Address
IMEM	32-bit	Big-Endian	0x3ff
DRAM	16-bit	Big-Endian	0x7fff
GBUS	16-bit	Little-Endian	0x3f

CL480-Specific File Items