1 Introduction

1.1 Overview

The CL484 is the second generation of C-Cube's MPEG-1 audio/video decoder chip based on the CL480 architecture. In addition to incorporating all existing features of the CL480, the CL484 maintains pin compatibility with the CL480 while providing the following new features:[1]

- *DiscView^{TM}:* Allows quick scanning and selection of disc contents by providing a menu display of up to nine images from which different tracks or discs may be chosen
- *CD-G (CD plus graphics):* Allows simultaneous delivery of CD audio data and low-bandwidth graphics and lyrics as per the Red Book audio standard
- *Bitmapped OSD (on-screen display):* Allows full-motion video to be overlaid with text and graphics
- *Key Control:* Allows you to adjust the audio pitch of a karaoke machine up or down to match the key of a karaoke singer
- *Advanced Error Concealment:* Allows best possible playback of any disc with minimized interruptions
- *Vertical Scaling:* Allows full-scale output of NTSC source material on PAL display monitors

Note: Unless otherwise stated, the term CL48x will be used throughout this manual as a shorthand notation to describe features that are common to both the CL480 and the CL484 architectures.

[1] See Chapter 11 for a full discussion of each of these features.

Designed for consumer electronics products, the CL48x reduces system cost by performing CD-ROM decoding, providing a serial CD interface, and by requiring only four Mbits of DRAM. The CL48x reduces development time and effort by performing system stream processing and audio/video synchronization. The CL48x 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 because it provides glueless interfaces to the CD-DSP, audio DAC, and NTSC/PAL encoder.

The CL48x processes the following data types:

- CD-ROM data
- CD-DA data
- MPEG-1 system streams, containing:
  - Constrained-parameters MPEG-1 video bitstreams
  - MPEG-1 audio (Layer II) bitstreams

The CL48x also features an extensive microcode set supplied with the hardware.

1.2 CL48x Features

The CL48x has these key features:

- Integrates on one chip: MPEG-1 audio and video decoding, video timing generation, CD-ROM decoding functions, and a memory controller
- Fully supports VideoCD 2.0 and 1.1, KaraokeCD 1.0, OM-1, CD-I Green Book[1] standard, CD-ROM and CD-ROM/XA Yellow Book standards, and the CD-DA and CD-G Red Book standards
- Accepts MPEG-1 system streams or CD data streams with no external parsing required
- Decodes and synchronizes SIF-resolution MPEG-1 video and two channels of MPEG Layer 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 rate supported is 44.1 kHz (16-bit)

[1] The CL48x can play back CD-I full-motion video titles.

- Decodes and displays still images with coded resolutions up to 704 x 576 while decoding MPEG audio streams
- Provides 8K bytes 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
- Requires only 4 Mbits of 80-ns DRAM, even for PAL systems
- Accepts compressed data from a 4-pin 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)
- Performs horizontal interpolation of decoded video using a four-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 background 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 1.2 watts at 3.6V when decoding audio and video
- Packaged in a 128-pin small-outline PQFP (18mm x 18mm body) or a TQFP

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/displays still picture(s) (5 versions)
  - Pause - Freezes displaying and decoding process
  - Freeze - Freezes displaying but continues decoding
  - DumpData - Allows retrieval of CD-ROM data for error correction
  - SetStreams - Selects which streams to decode
  - DisplayGraphics - Displays graphics images
  - SetVideoFormat - Changes output video format
  - InquireBufferEmptiness - Measures data in bitstream buffer
  - Replay - Restarts the previous Play command
  - Reset - Initializes the CL48x and its microcode
  - DisplayDigest - Displays still pictures (4 versions)[1]
  - DigestReady - Prepares CL484 for DisplayDigest[1]
  - CDGSetChannel - Selects the display channel from disc[1]
  - CDGFreeze - Freezes or blanks the screen[1]
  - CDGSetBorderColor - Defines background color[1]
  - CDGTest - Tests bitstream for CD-G data[1]
  - CDGDisplayMode - Specifies screen to be CD-G or blank color[1]
  - ModuleLoad - Loads and runs new module[1]
  - ReadSequenceHeader - Decodes bitstream until sequence header
  - PitchShift - Controls the shift in pitch of the audio stream[1]
- Provides 15 interrupts, giving the host feedback on bitstream transitioning, displaying, and decoding processes

[1] CL484 microcode version only.

1.3 Hardware Description

The CL48x 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/ROM and is loaded into on-chip memory as needed. During power-up, the microcode is read from a ROM connected to the CL48x.

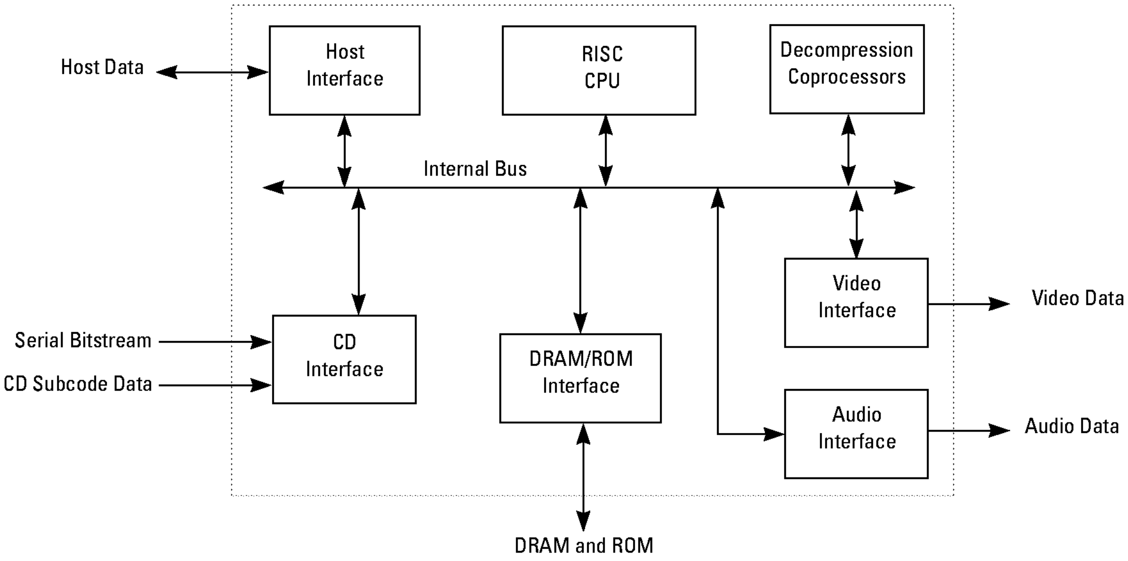


Figure 1-1 Block Diagram of the CL48x

Each of the CL48x's interface modules is described next.

1.3.1 Host Interface

The CL48x's host interface is used to initialize the chip, 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 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 receives serial compressed data from a CD-DSP. The CL48x supports seven different input formats so that all of the popular CD-DSP chips can be used.

1.3.3 DRAM/ROM Interface

The CL48x's DRAM/ROM interface provides a glueless interface between the CL48x 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 size required for storing the CL480's MPEG microcode is less than 64 Kbytes; for the CL484, it is greater than 64 Kbytes, requiring a 128 Kbyte ROM. The CL48x supports up to two Mbytes of ROM for uses such as still-image 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 CL48x 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 CL48x audio interface outputs decoded audio samples in bit-serial format and controls their attenuation. The CL48x also has a left/right signal to indicate which channel is being output. The polarity of the left/right signal, the order of bits and the number of clocks per audio sample are programmable so that any of the popular audio DACs can be used.

1.4 Microcode Description

The operation of the CL48x 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 CL48x's microcode. Six different process states occur at different times in a CL48x system. These process states are Microcode-Load, Initialization, Command, Bitstream Input, Decode, and Output.

1.4.1 Microcode-Load Process

The CL48x microcode must be loaded at startup in all systems. The microcode is loaded from ROM automatically following a pulse on the $\overline{\text{RESET}}$ pin.

1.4.2 Initialization Process

After hardware reset, the CL48x 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.)

1.4.3 Command Process

Following initialization of the microcode, CL48x 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 Command ID and parameter values

Command and operating state information are 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).

1.4.4 Bitstream Input Process

The CL48x accepts data streams from the CD interface containing:

- □ Compressed MPEG data (Mode 2 Form2)
- □ CD-ROM data (Mode 1 or Mode 2 Form 1)
- □ CD-DA data

Besides obtaining input from the serial interface 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.

1.4.5 Decode Process

The decode process is the process by which the CL48x decompresses the input bitstream using the MPEG decoding algorithm and places the decompressed audio and video frames in their respective buffers in the CL48x's local DRAM.

1.4.6 Output Process

The output process in the CL48x'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.

1.5 Typical Application

The CL48x is designed for low-cost applications such as consumer electronics products. In the example shown in Figure 1-2, the CL484 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.

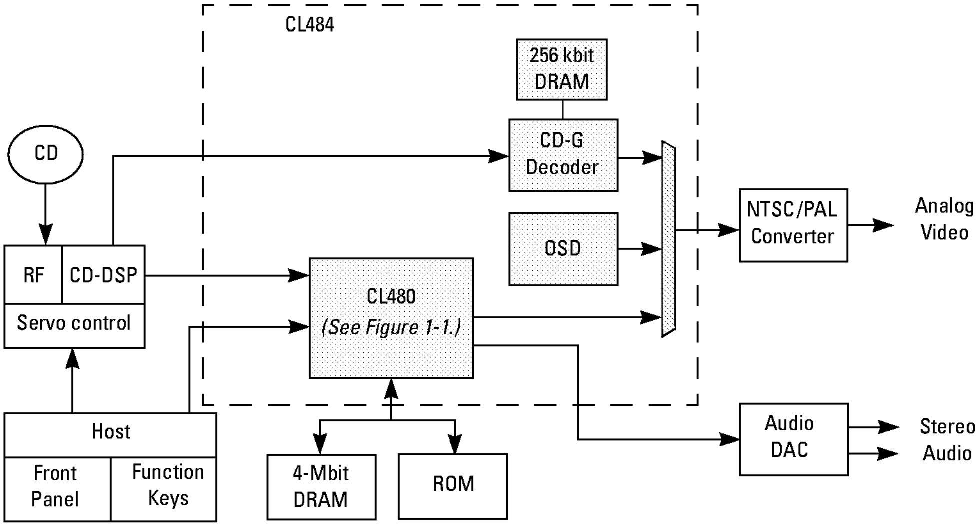


Figure 1-2 CL48x Typical Application

2
MPEG Overview

This chapter presents an overview of the Moving Picture Experts Group (MPEG) standards 11172 and 13818, commonly known as MPEG-1 and MPEG-2, respectively.[1] It first presents some of the history of the development of the MPEG standards. Later sections describe the structure of the two standards from the top down, beginning with system streams (MPEG-1), program streams, and transport streams (both MPEG-2). Finally, the basic compression algorithms common to both standards are described: motion compensation, discrete cosine transform (DCT), quantization, and run-length encoding.

2.1 Historical Background

When the MPEG committee began the task of specifying a syntax for compressed digital video, its goal was to deliver video on a compact disc, taking into account its low data transfer rate of 1.416 Mbits per

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

second. Aware that it was impossible to represent a CCIR 601-resolution image at such a low data rate, the committee specified a one-quarter resolution image (352x240 NTSC; 352x288 PAL) as the standard input format (SIF). As a result, the committee made MPEG-1 a frame-oriented syntax, rather than a field-oriented syntax. When decoded, the SIF-resolution video is expanded to fill a full television screen, resulting in an image quality similar to VHS tape.

Broadcast-television equipment makers immediately recognized the potential of MPEG technology to increase the channel efficiency of satellite transponders and cable networks, but the broadcast industry was not limited to compact disc bandwidths and unwilling to settle for VHS resolution. Consequently, the MPEG committee developed a second standard, called MPEG-2, specifically designed for broadcast applications. The MPEG-2 standard is designed to represent CCIR 601-resolution video (704x480 NTSC; 704x576 PAL) at data rates of 4.0 to 8.0 Mbits per second. In addition, MPEG-2 provides support for interlaced fields, 16:9 aspect ratio video, multiple video channels in a single-system stream and extensibility to HDTV. It is also important to note that MPEG-1 is a subset of MPEG-2, so any compliant MPEG-2 decoder will be able to decode MPEG-1 syntax video.

2.2 Content of the Standards

Each of the two standards, MPEG-1 and MPEG-2, is divided into three sections: systems, video, and audio.

The systems specification addresses the combination of separate audio and video streams into a single stream for storage or transmission. It also provides a mechanism for synchronizing audio and video by MPEG decoders. This chapter discusses both MPEG-1 and MPEG-2 systems specifications.

The video and audio specifications give the syntax and semantics of encoded video and audio bitstreams. This chapter describes the video specifications in some detail. Future editions of this document will include more detailed descriptions of the audio specifications.

2.3 MPEG Streams

This section explains the general structure of an MPEG-1 stream and introduces some basic concepts used in the rest of the chapter.

2.3.1 MPEG-1 Stream Structure

In its most general form, an MPEG-1 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 compressed audio and video streams.

Figure 2-1 shows a generalized MPEG decoding system.

The *system demux* extracts the timing information from the MPEG stream and sends it to the other system components. (Section 2.6 has more information about the use of timing information for audio and video synchronization.) The system demux also demultiplexes the video and audio streams 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 2.4 and 2.5 for more information about video compression.) C-Cube currently offers the CL450, CL480/484 and the CL9100 as MPEG decoders.

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

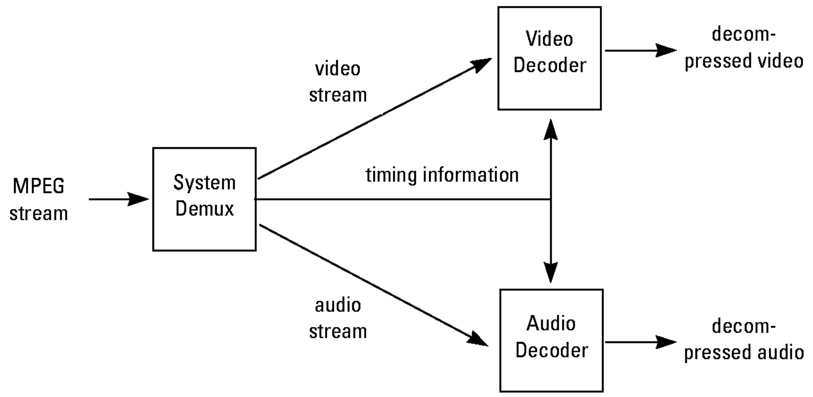


Figure 2-1 General MPEG Decoding System

2.3.2 MPEG-2 Streams

MPEG-2 defines two kind of system streams: program and transport. Both are multiplexed streams consisting of video and audio elementary streams, and both are subdivided into packets for transmission.

Program Stream

A single stream, resulting from the combination of one or more elementary streams, which share a common time base. The MPEG-2 program stream is similar to the MPEG-1 system stream.

The program stream is designed to be used in relatively error-free environments such as multimedia applications. The packets in a program stream can be of any length.

Transport Stream

A single stream, resulting from the combination of one or more programs, where a program is a collection of elementary streams with a common time base. The transport stream is designed for relatively error-prone environments such as broadcast applications. Transport stream packets are fixed at 188 bytes in length.

2.3.3 Video Stream Data Hierarchy

The MPEG-1 and MPEG-2 standards define a hierarchy of data structures in the video stream as Figure 2-2 shows schematically.

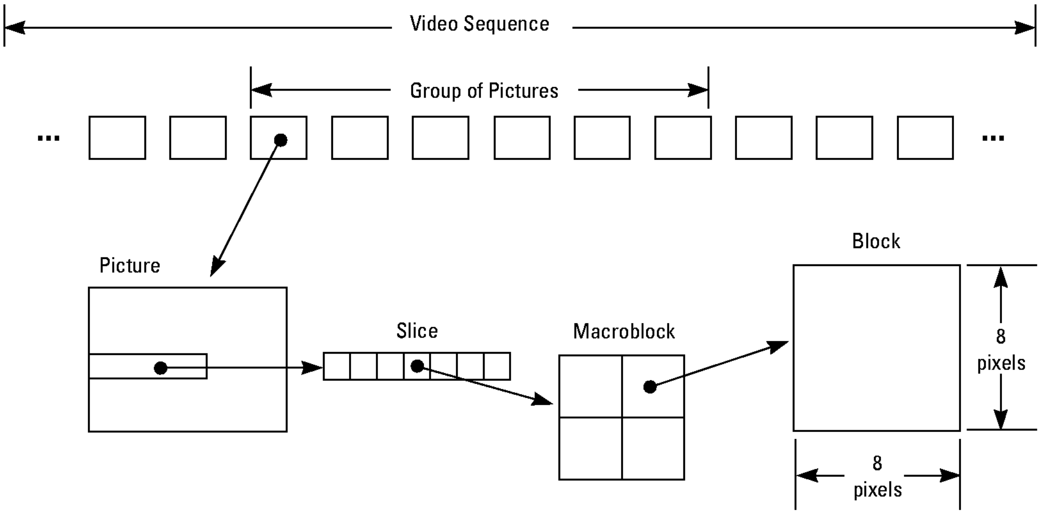


Figure 2-2 MPEG Data Hierarchy

Video Sequence

Consists of a sequence header, one or more groups of pictures, and an end-of-sequence code. The video sequence is another term for a video stream as defined above.

Group of Pictures

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 (CbCr) 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.)

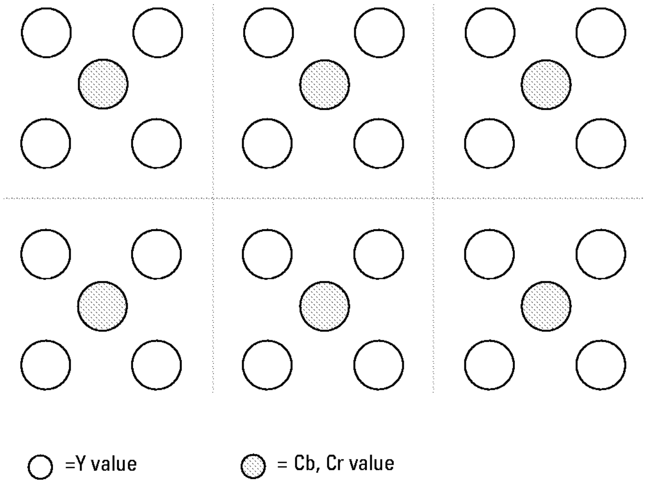


Figure 2-3 Location of Luminance and Chrominance Values

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 x 16-line section of luminance components and the corresponding 8-pixel x 8-line section of the 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 Figure 2-4 shows. The numbers correspond to the ordering of the blocks in the data stream, with block 1 first.

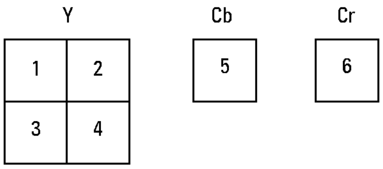


Figure 2-4 Macroblock Composition

Block

A block is an 8 x 8 set of values of a luminance or chrominance component. Note that a luminance block corresponds to one-fourth the size of the displayed image as compared to a chrominance block.

2.4 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-1 and MPEG-2 standards take advantage of this temporal redundancy to represent some pictures in terms of their differences from reference picture. This section describes the picture types and explains the techniques used in inter-picture coding.

2.4.1 Picture Types

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

Intra-Pictures

Intra, 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, therefore, provide moderate compression. I-pictures typically use about two bits per coded pixel.

Predicted Pictures

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

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

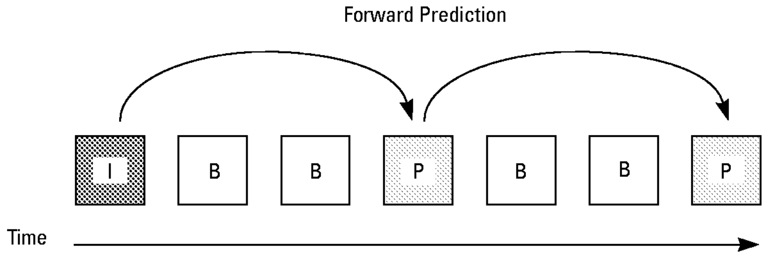


Figure 2-5 Forward Prediction (display order shown)

Bidirectional Pictures

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

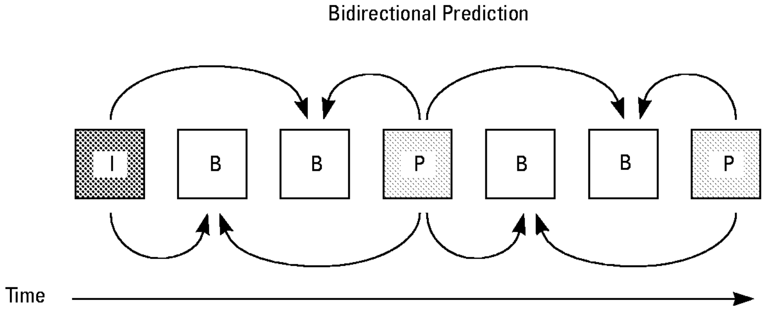
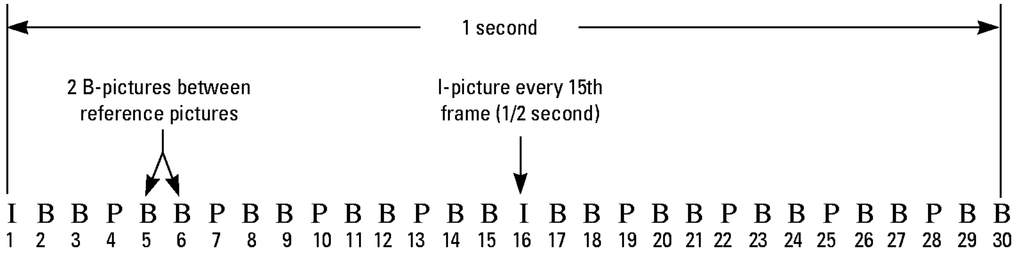


Figure 2-6 Bidirectional Prediction (display order shown)

2.4.2 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, intra pictures are typically used two times per second.

The encoder also chooses the number of bidirectional 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 a large class of scenes, a workable arrangement is to have two bidirectional pictures separating successive reference pictures. Figure 2-7 shows a typical arrangement of I-, P-, and B-pictures in the order that they are displayed.


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

Note: The I-picture at every 15th frame is for NTSC only; for PAL, it is at every 12th frame.

Figure 2-7 Typical Display Order of Picture Types

The MPEG encoder reorders pictures in the video stream to present the pictures to the decoder in the most efficient sequence. In particular, the reference pictures needed to reconstruct B-pictures are sent *before* the associated B-pictures. Figure 2-8 demonstrates *C-Cube's encoder ordering* for the first section of the example shown in Figure 2-7.

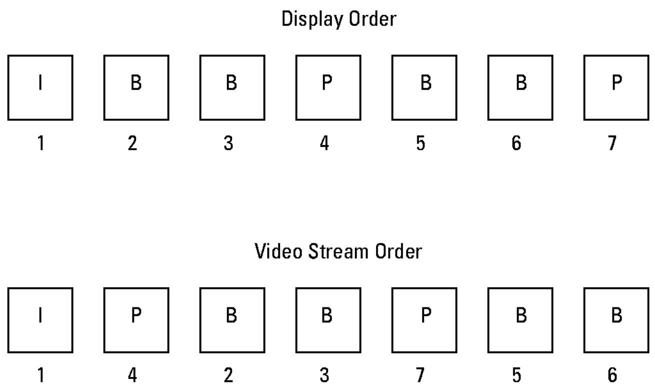


Figure 2-8 Video Stream versus Display Ordering

2.4.3 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 a factor of approximately three as 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 difference between the reference and the macroblock being coded (*motion vectors*)
- □ The content differences between the reference and the macroblock being coded (*error terms*)

Not all information in a picture can be predicted from a previous picture. Consider a scene in which a door opens: The visual details of the room behind the door cannot be predicted from a previous frame in which the door was closed. When a case such as this arises—i.e., a macroblock in a P-picture cannot be efficiently represented by motion compensation—it is coded in the same way as a macroblock in an I-picture using transform coding techniques (see Section 2.5).

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.

2.5 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 to the frequency domain with the DCT.

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

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

Some blocks of pixels must be coded more accurately than others. For example, blocks with smooth intensity gradients require 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 16 x 16 block of pixels. This mechanism can also be used to provide smooth adaptation to a particular bit rate.

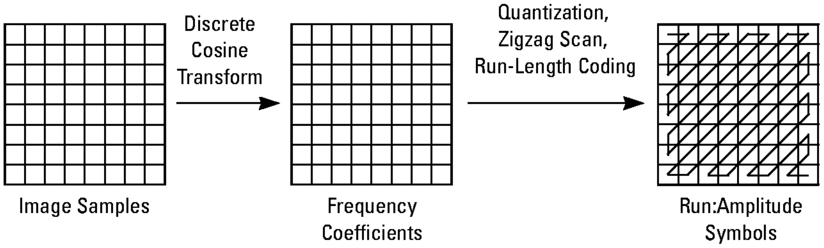


Figure 2-9 Transform Coding Operations

2.6 Synchronization

The MPEG standards provide a timing mechanism that ensures synchronization of audio and video. The MPEG-1 standard defines two parameters used by the decoder: the *system clock reference* (SCR) and the *presentation timestamp* (PTS). The MPEG-2 standard adds the *program clock reference* (PCR) and also provides for the SCR and PCR to have extensions with a resolution of 27 MHz.

The MPEG-specified "system clock" runs at 90 kHz and generates 7.8 x 10^9 clocks in a 24-hour day. SCRs and PTSs in MPEG-1 are 33-bit values, which can represent any clock cycle in a 24-hour period.

2.6.1 System Clock References

An SCR is a snapshot of the encoder system clock. The SCRs used by the audio and video decoder must have approximately the same value. To keep their values in agreement, SCRs are inserted into the MPEG stream at least every 0.7 seconds (minimum) by the MPEG encoder, then extracted by the system decoder and sent to the audio and video decoders as Figure 2-10 illustrates. The video and audio decoders update their internal clocks, using the SCR value sent by the system decoder.

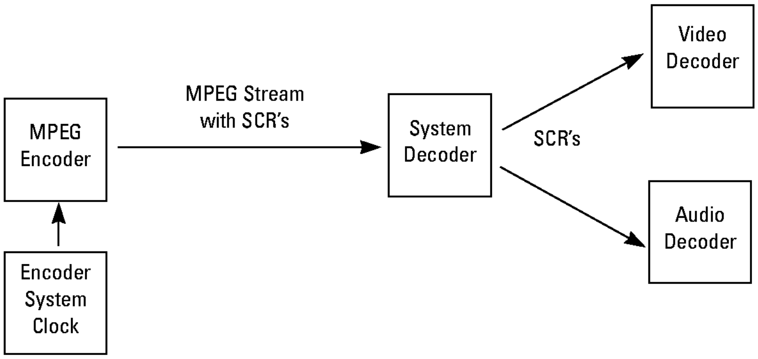


Figure 2-10 SCR Flow in MPEG System

2.6.2 Presentation Timestamps

PTS's are samples of the encoder system clock that are associated with some 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 encoder inserts PTS's into the MPEG stream at least every 0.7 seconds (minimum).

The video decoder either deletes or repeats pictures to ensure that the PTS matches the current value of the SCR when a picture with a PTS is displayed. If the PTS is earlier (has a smaller value) than the current SCR, the video decoder discards the picture. If the PTS is later (has a larger value) than the current SCR, the video decoder repeats the display of the picture.

2.6.3 Program Clock References

PCR's are used only in MPEG-2. The PCR is used in the transport stream in the same way that the SCR is used in an MPEG-1 system stream. Since each program can have its own time base, a transport stream containing multiple programs has a separate PCR field for each program.

3
Signal Descriptions

This chapter describes the signals that comprise the external physical interface to the CL48x. The information presented for each signal includes the signal mnemonic and name, type (input, output, or bidirectional), and description. (Note: The $\overline{\text{overbar}}$ symbol denotes active LOW polarity.)

For information about the functional operation of the CL48x, including functional and timing waveforms, see Chapters 4, 5, 6, 7, 8, and 9.

Figure 3-1 shows a diagram of the CL48x with all signals grouped together.

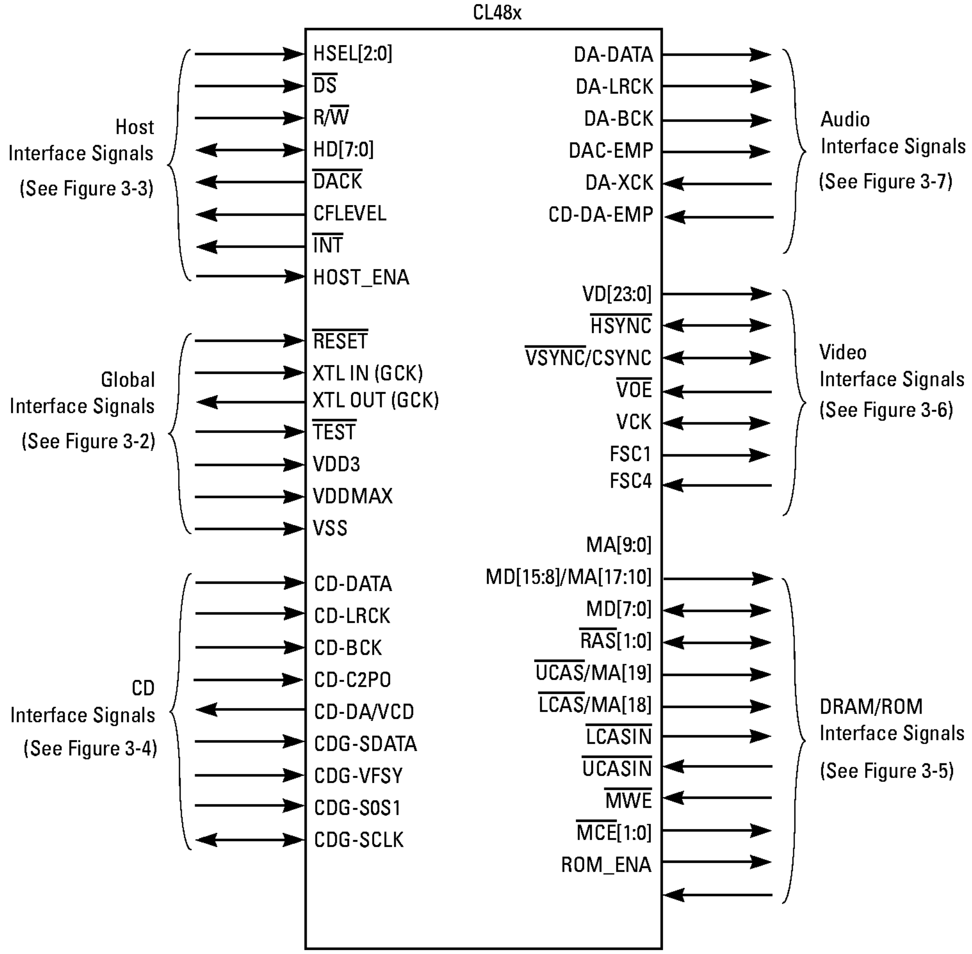


Figure 3-1 Bus Connection Diagram

3.1 Global Interface Signals

The CL48x global interface signals are composed of the $\overline{\text{RESET}}$, XTL IN, XTL OUT, $\overline{\text{TEST}}$, and power signals as shown in Figure 3-2.

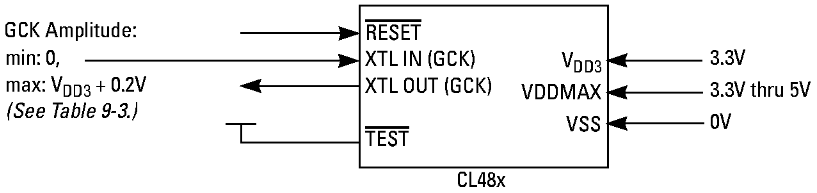


Figure 3-2 Global Interface Signals

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

An external device asserts $\overline{\text{RESET}}$ (active LOW) to force the CL48x to execute a hardware reset. To be recognized, $\overline{\text{RESET}}$ must be asserted for at least fifteen complete GCK cycles.

GCK (XTL IN) — Global Clock **Input**

The CL48x uses GCK (XTL IN and XTL OUT) to clock the internal processor. This oscillator is typically 40 MHz (VCK In mode) or 40.5 MHz (VCK Out mode). If a clock driver (i.e., oscillator) is used for GCK, it should be connected to pin 19, XTL IN, with a maximum low voltage of $0.8V_{DD3}$ and a maximum high voltage 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). When pulled LOW, the $\overline{\text{TEST}}$ pin will 3-state all the CL48x's 3-stateable outputs except the DRAM interface's data pins. The DRAM interface's data pins can be 3-stated, however, by conducting a DRAM read transaction immediately before the $\overline{\text{TEST}}$ pin is asserted. The read cycle causes the data pins to become input pads, thereby 3-stating the associated output buffers.

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

This pin supplies 2.7 to 3.6 volts to internal power. The output high voltage of the CL48x is, at most, VDD3.

VSS — Power (ground) **Input**

Ground.

VDDMAX — Power (maximum) **Input**

This pin is the maximum input voltage connected to any CL48x pin. The CL48x operates at 3.3 volts, but can accept up to 5-volt inputs (except on GCK) 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 CL48x and the host processor. Figure 3-3 shows how the data transfer signals of the CL48x connect to the host processor. The various modes of transferring data are discussed in Section 4.2.

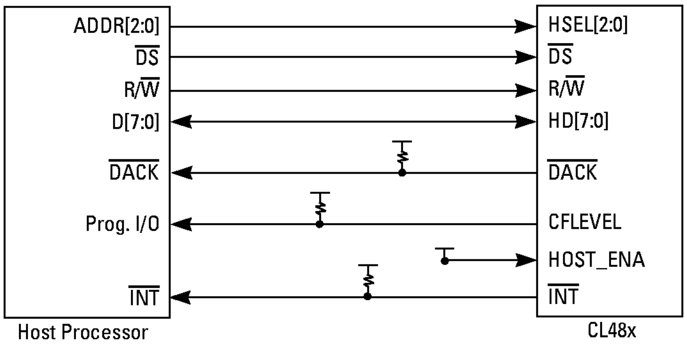


Figure 3-3 Host Interface Signals

HOST_ENA—Host Enable **Input**

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

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 CL48x may be accessed: A_MSB, A_MB, A_LSB, D_MSB, and D_LSB.

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

The host processor asserts $\overline{\text{DS}}$ to indicate that a valid host bus read or write operation is in progress. The falling edge of this signal triggers a new cycle.

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

HD[7:0] comprise the eight-bit bidirectional host data bus. The host processor uses HD[7:0] to write data to the CL48x's code FIFO (CFIFO), internal registers, and local DRAM. The CL48x uses HD[7:0] to send requested data to the host processor. The direction is determined by R/$\overline{\text{W}}$.

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

The host processor drives R/$\overline{\text{W}}$ LOW to initiate a write operation, and drives R/$\overline{\text{W}}$ HIGH to initiate a CL48x read operation to the host data bus.

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

The CL48x asserts $\overline{\text{DACK}}$ (active LOW) when it is ready to receive or output data on HD[7:0]. When the CL48x responds to a read request, it holds $\overline{\text{DACK}}$ deasserted (HIGH) until the requested data is ready. When the CL48x responds to a write request, it asserts $\overline{\text{DACK}}$ when it has received and latched the write data.

$\overline{\text{DACK}}$ is an open-drain signal, which allows it to be wire ORed with other components on the host bus. It requires a pullup resistor value that is dependent on the voltage supply as shown in Table 3-1.

Table 3-1 **$\overline{\text{DACK}}$, CFLEVEL, and $\overline{\text{INT}}$ Pullup Resistor Values**[1]

Pullup Supply Voltage	Minimum	Typical
3.3 V	439 ohms	500 ohms
5.0 V	680 ohms	1.0K ohms

[1] Product reliability may be affected if these values are not observed.

CFLEVEL — CFIFO Level **Open-Drain Output**

When CFLEVEL is zero, the CL48x 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 value as shown in Table 3-1. (Note: This signal is not used in VCD players.)

$\overline{\text{INT}}$ — Interrupt Request **Open-Drain Output**

The CL48x 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. It requires a pullup resistor value as shown in Table 3-1.

3.3 CD Interface Signals

The CL48x's CD interface is dedicated to receiving the output of a CD DSP. A four-wire serial bus connects the CD DSP directly with the CL48x as Figure 3-4 shows. A four-wire serial bus also connects the CD-G signals.

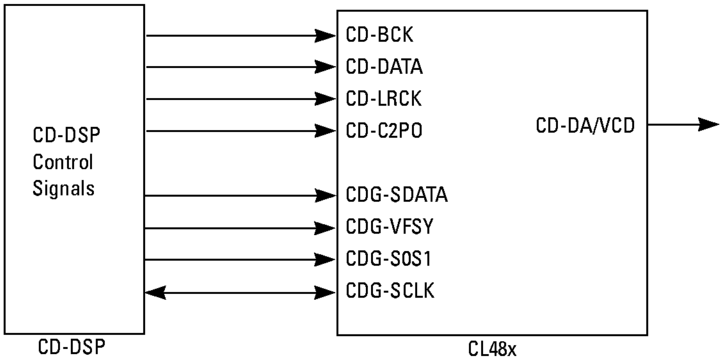


Figure 3-4 CD-Decoder Interface Signals

CD-BCK — CD Bit Clock Input

BCK is the CD-decoder bit clock. The CL48x can accept multiple BCK rates as detailed in Table 6-2.

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 CL48x and has programmable polarity (either left- or right-channel HIGH).

CD-C2PO — CD-ROM Data Input

CD-C2PO signals a corrupted byte by going HIGH when an error occurs. It is used when receiving CD-ROM data, but is ignored when in CD-DA pass-through mode.

CD-C2PO is asserted sample-aligned (see Figure 6-3 on pages 57 and 58) for one-half the duration of the sample for each corrupted byte. The control parameter C2PO_LSB_FIRST is used by the CL48x to determine whether CD-C2PO refers to the least-significant byte or the most-significant byte of the sample.

CD-DA/VCD— CD-DA/VCD Output

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

CDG-SDATA — CD-G (Subcode) Data **Input**

Indicates serial subcode data input. Subcode data is time, text or graphical information stored together with audio across eight channels on a CD.

CDG-VFSY — CD-G (Subcode) Frame Sync **Input**

Indicates frame-start or composite synchronization input.

CDG-S0S1 — CD-G (Subcode) Block Sync **Input**

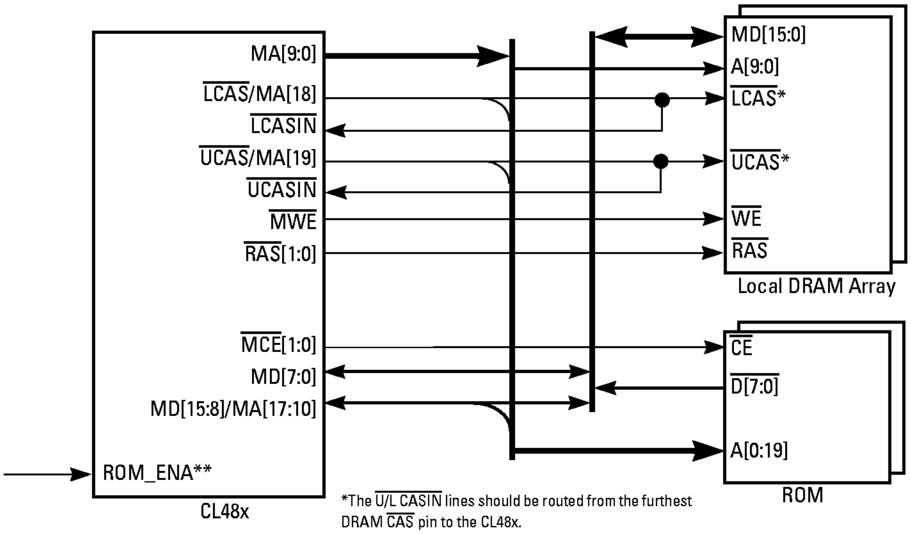
Indicates block-start synchronization input.

CDG-SCLK — CD-G (Subcode) Clock **Bidirectional**

Indicates subcode data clock input or output.

3.4 DRAM Interface Signals

Figure 3-5 shows the signals that comprise the CL48x's DRAM interface. The signal descriptions are presented following the figure. See Chapter 5 for more information about the DRAM memory architecture.



*The $\overline{U/L\ CASIN}$ lines should be routed from the furthest DRAM $\overline{CAS}$ pin to the CL48x.

**ROM_ENA should be pulled HIGH.

Figure 3-5 DRAM/ROM Interface Signals

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

The CL48x multiplexes the row and column addresses on these signals to address up to one Mbyte of DRAM, and uses address bits 9:0 for ROM addresses. See Chapter 5 for details of addressing various DRAM components and DRAM array sizes.

MD[15:0] — Memory Data Bus Bidirectionals

These signals comprise the memory data bus by which data is transferred between the CL48x 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 CL48x 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 CL48x 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 MA[18] and MA[19], respectively, in ROM accesses.

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

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

When the CL48x reads data from the local DRAM array, the data on MD[15:0] is latched into the CL48x 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, $\overline{\text{UCASIN}}$ and $\overline{\text{LCASIN}}$ connect to the $\overline{\text{UCAS}}$ and $\overline{\text{LCAS}}$ pins, respectively.

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

The CL48x asserts $\overline{\text{MWE}}$ (active LOW) during write operations (data transfer from CL48x to DRAM). The CL48x leaves $\overline{\text{MWE}}$ HIGH during a read operation (DRAM to CL48x).

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

The CL48x asserts $\overline{\text{MCE}}$ (one for each ROM) during a read operation from ROM to the CL48x.

ROM_ENA—Boot ROM Enable **Input**

ROM_ENA informs the CL48x that a boot ROM is attached. If this signal is active when the CL48x receives a $\overline{\text{RESET}}$ signal, ROM bank 0 is loaded into the CL48x's processor instruction memory. After the instruction memory is loaded, instruction execution begins at address zero. This pin should be wired to 1 during normal operation, and can be set to 0 when debugging the hardware.

3.5 Video Interface Signals

The CL48x'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 CL48x'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.

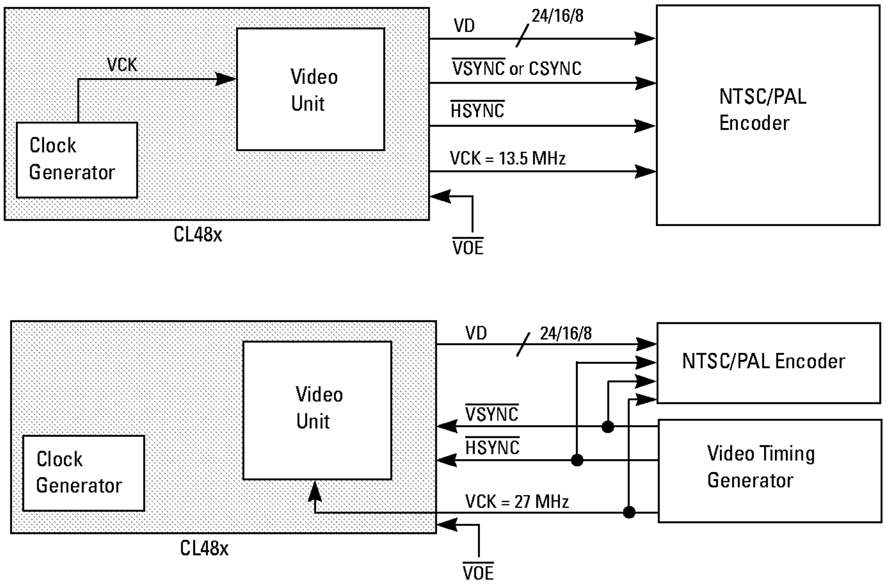


Figure 3-6 Video Interface Signals

Note: In revision C1 and E1 silicon, concurrent output of CSYNC and $\overline{\text{HSYNC}}$ is not supported.

VD[23:0] — Video Data Bus   Outputs

The CL48x 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 CL48x 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 CL48x 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 CL48x to drive the pixel bus, VD[23:0]. When $\overline{\text{VOE}}$ is deasserted, the CL48x puts the pixel bus in a high-impedance state

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 CL48x'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 signals, DA-XCK and CD-DA-EMP, as input and outputs DAC-EMP. Figure 3-6 shows the CL48x's audio interface signals.

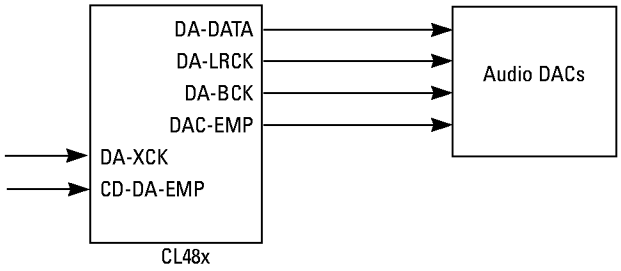


Figure 3-7 Audio Interface Signals

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 controls the de-emphasis circuitry of the audio output DACs. When in CD-DA mode, this pin reflects the state of the CD-DA-EMP pin; when in VCD mode, this pin reflects 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.

CD-DA-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 CL48x's host interface, shown in Figure 4-1, provides a simple interface to an eight-bit microcontroller. The host interface provides three 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

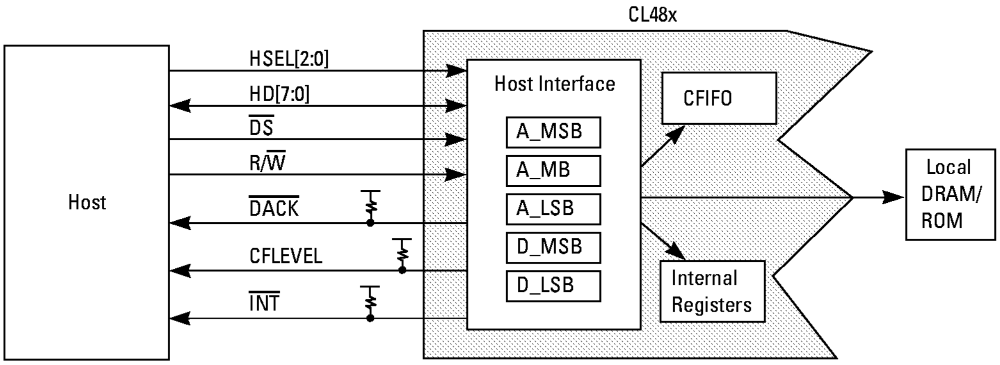


Figure 4-1 CL48x Host Interface

The host communication functions also include device initialization and microcode loading (if the ROM is not used). Host accesses to the CL48x can be asynchronous to GCK.

4.2 Host Interface Registers

The host processor accesses CL48x 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 CL48x's address registers allow the host to access the CL48x'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 CL48x 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.

CL48x read and write operations using the host interface registers are described in the next two sections.

Host Interface Register Write

Figure 4-2 shows typical waveforms for a host write to an 8-bit 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.

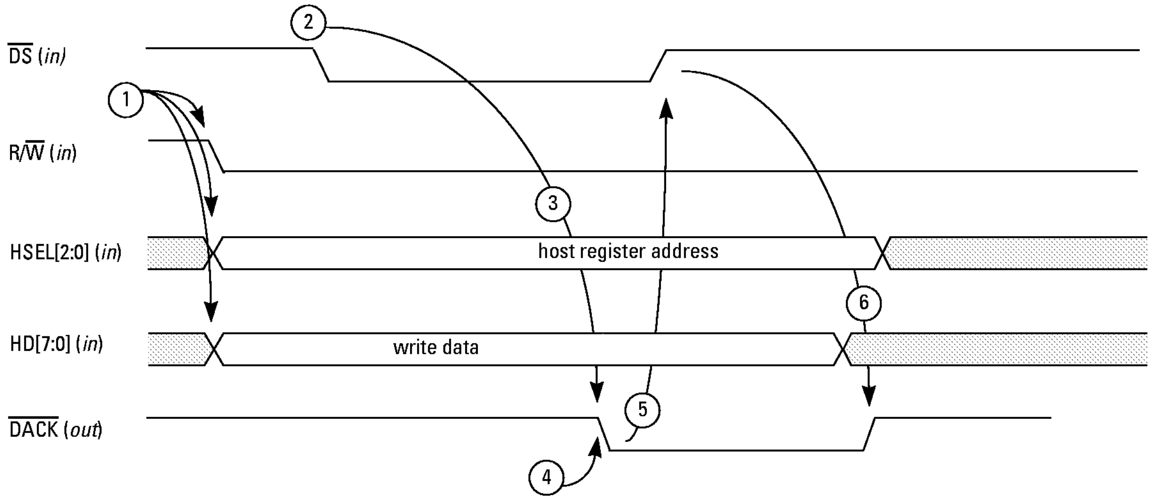

$\overline{\text{DS}}$ (*in*)
R/$\overline{\text{W}}$ (*in*)
HSEL[2:0] (*in*)
HD[7:0] (*in*)
$\overline{\text{DACK}}$ (*out*)
1
2
3
4
5
6
host register address
write data

Figure 4-2 Host Interface Local Register, Write Operation

1. The host drives R/$\overline{\text{W}}$ 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/$\overline{\text{W}}$, and HSEL[2:0] lines have settled, the host processor asserts $\overline{\text{DS}}$.
3. In response to the assertion of $\overline{\text{DS}}$, the CL48x begins the write sequence and sometime later asserts a $\overline{\text{DACK}}$ signal.
4. The CL48x latches the data when $\overline{\text{DACK}}$ goes LOW.
5. In response to the assertion of $\overline{\text{DACK}}$, the host deasserts $\overline{\text{DS}}$.
6. The CL48x responds by releasing $\overline{\text{DACK}}$. This completes the write cycle.

Host Interface Register Read

Figure 4-2 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.

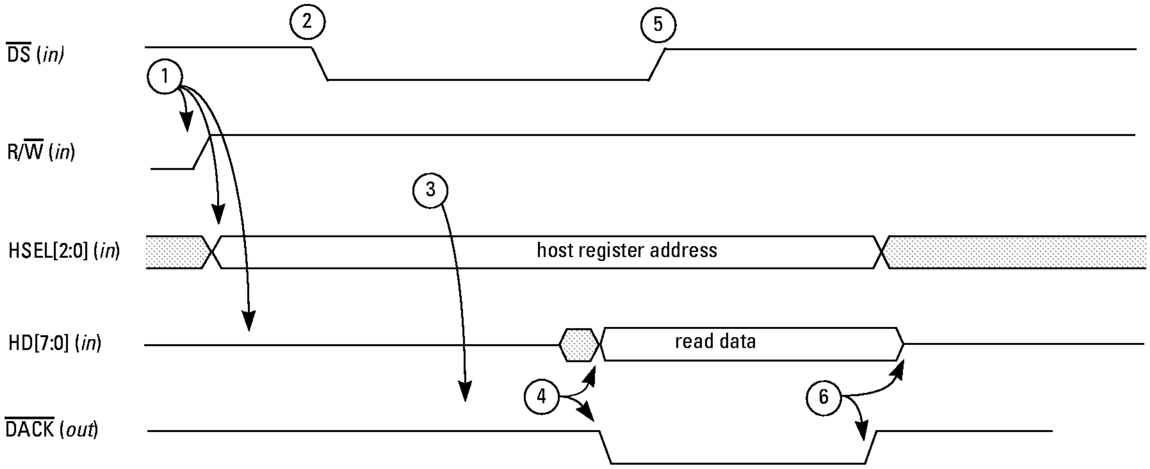


Figure 4-3 Host Interface Local Register, Read Operation

1. The host drives R/$\overline{\text{W}}$ HIGH to indicate a read operation and drives the host register address onto HSEL[2:0].
2. When the R/$\overline{\text{W}}$ and HSEL[2:0] lines have settled, the host processor asserts $\overline{\text{DS}}$ to indicate to the CL48x that a host bus read access is in progress.
3. In response to the assertion of $\overline{\text{DS}}$, the CL48x holds $\overline{\text{DACK}}$ deasserted (HIGH) until it can respond to the read request. Because the CL48x is performing arbitration of its internal data buses during register access, $\overline{\text{DACK}}$ assertion is delayed at least one GCK cycle.
4. When the data is available, the CL48x asserts $\overline{\text{DACK}}$ and drives the data.
5. The host deasserts $\overline{\text{DS}}$ when the data has been read.
6. In response to the deassertion of $\overline{\text{DS}}$, the CL48x releases $\overline{\text{DACK}}$ and HD[7:0].

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 4-4. This 21-bit memory address is a 16-bit word address, not a byte address.

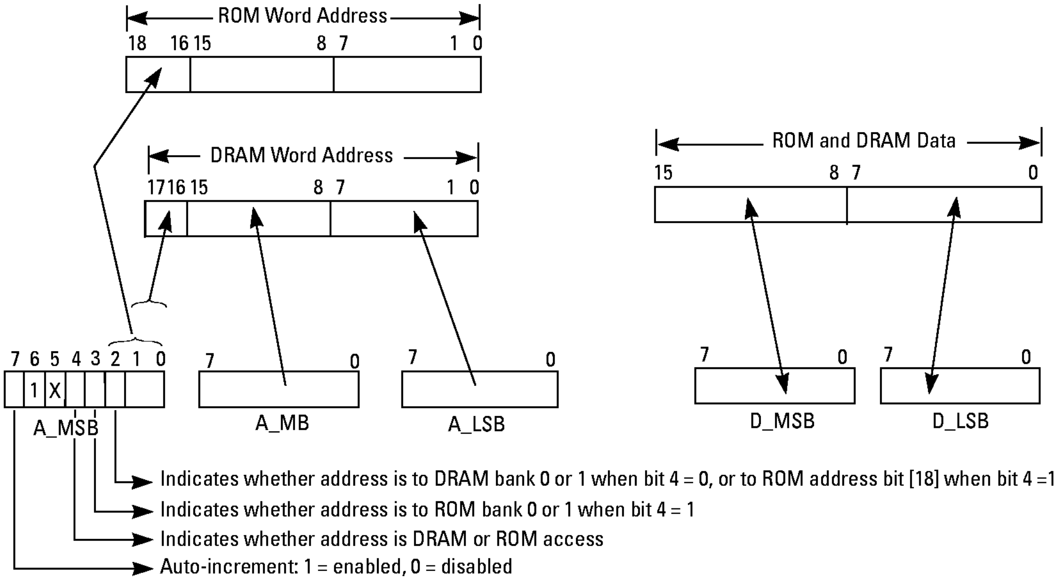

ROM Word Address
18 16 15 8 7 1 0
DRAM Word Address
17 16 15 8 7 1 0
ROM and DRAM Data
15 8 7 0
7 6 5 4 3 2 1 0
1 X
A_MSB
7 0
A_MB
7 0
A_LSB
7 0
D_MSB
7 0
D_LSB
Indicates whether address is to DRAM bank 0 or 1 when bit 4 = 0, or to ROM address bit [18] when bit 4 =1
Indicates whether address is to ROM bank 0 or 1 when bit 4 = 1
Indicates whether address is DRAM or ROM access
Auto-increment: 1 = enabled, 0 = disabled

Figure 4-4 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	1 or 0	ROM Bank 1
1	0	1 or 0	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 auto-incrementing. When auto-incrementing is enabled (bit 7 = 1), each 16-bit data transfer between the host and the CL48x causes the DRAM address to be incremented to the next *word.*

For ROM accesses, a single access from the host generates two cycles; 16 bits are created from two sequential reads of an 8-bit ROM address.

DRAM and ROM address ordering is shown in the memory map of Figure 4-5.

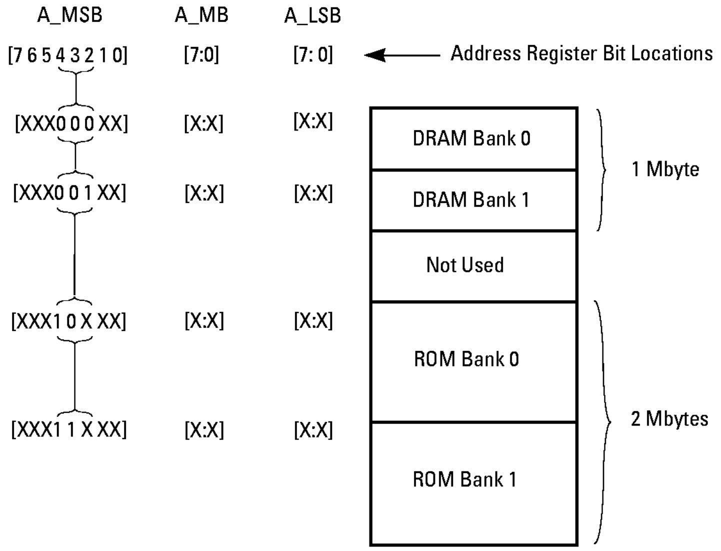


Figure 4-5 DRAM and ROM Address Memory Map

To initiate auto-incrementing to external memory (DRAM or ROM):

1. Establish the starting address by writing the registers:

   A_LSB

   A_MB

   A_MSB with bits [7:6] set (see Table 4-1)

2. Conduct normal read/write transactions from/to the DRAM via the D_LSB and D_MSB registers. The first pair of transactions (i.e., one access each to D_LSB and D_MSB registers) accesses the specified word address. Subsequent pairs of transactions to D_LSB and D_MSB will, first, automatically increment the word address by one, then access the adjacent word in external memory.

3. Do not access the address registers until you wish to disable or reset the auto-increment mode.

4.2.5 CFIFO Access

Coded data may be sent to the CL48x in one of two basic modes:

- Serial mode (through the CD interface) as shown in Figure 4-6 and described further in Chapter 8.
- Parallel mode (through the host interface) as shown in Figure 4-7 and described in this section.

Note: The CFIFO does not need to be accessed from the host in VCD players.

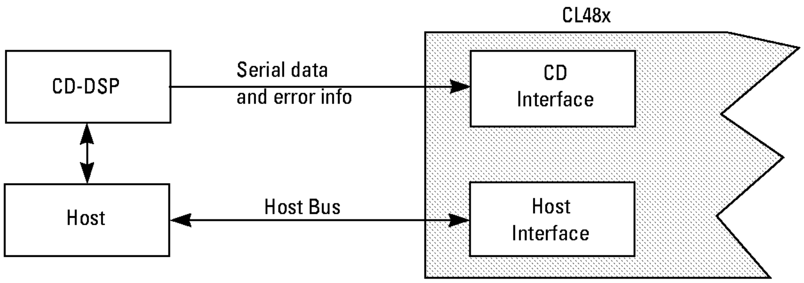


Figure 4-6 Sending Serial Coded Data on CD Interface

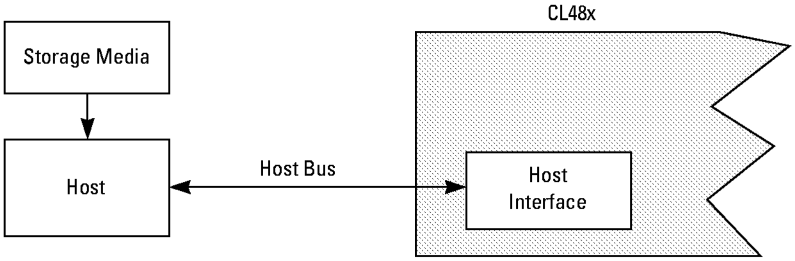


Figure 4-7 Sending Parallel Coded Data on Host Interface

In CFIFO accesses through the host interface, the host specifies A_MSB[7:6]=00 as shown in Figure 4-9, then writes to the D_LSB register, and finally writes to the D_MSB register. The words written to D_MSB and D_LSB will transfer to the CFIFO on the write operation to D_MSB.

MPEG bitstreams are input to the CL48x 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 16th bit of the stream is placed in the LSb of that same register. This ordering begins again with the 17th bit of the stream.

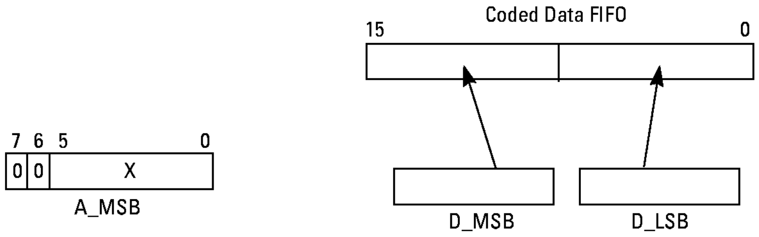


Figure 4-8 CFIFO Register Address Formed by A_MSB

The CL48x 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 31 words, each of which is 16-bits wide. The write will auto-increment, with the CL48x taking care of all of the pointers.

The CL48x will assert the CFLEVEL signal (active high) when the CL48x CFIFO has room for at least 44 bytes of compressed data.

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 4-9.

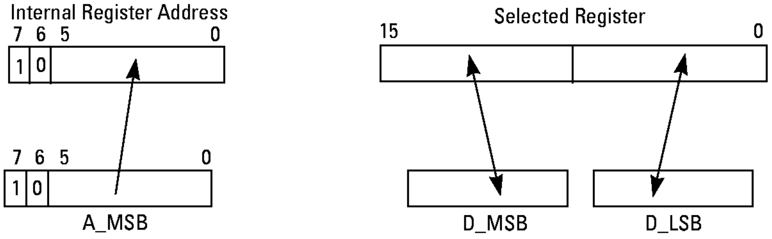


Figure 4-9 Internal Register Address Formed by A_MSB

Transfers from the host interface 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

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 CL48x
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

5 DRAM/ROM Interface

This chapter describes the local DRAM/ROM interface bus. It details all of the signals necessary to connect the CL48x to a DRAM array and optional ROM.

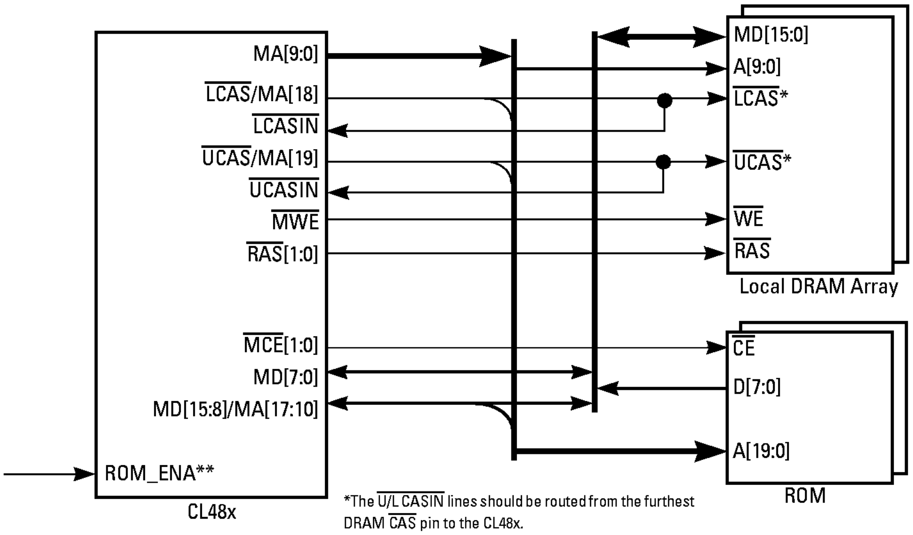
5.1 General Description

The memory controller in the CL48x interacts directly with external memory and generates the control signals required to access external DRAMs and ROMs. The maximum transfer rate between the host and DRAM is approximately 2.5 Mbytes per second. The CL48x 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 CL48x 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 VideoCD, MPEG-1 system and CD-DA bitstreams*. 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 for on-screen graphics display data, Dumpdata() buffers, or any other use.



*The U/L CASIN lines should be routed from the furthest DRAM CAS pin to the CL48x.

**This signal should be pulled high.

Figure 5-1 DRAM Interface

The DRAM array is typically one 256K x 16 DRAM.

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
- CL48x microcode instructions
- Bitstream header parameters and other variables
- The Command FIFO

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

5.1.2 ROM: Amount and Organization

The CL48x can support up to two Mbytes of ROM: specifically, two ROMs, each which can be up to 1M x 8 bits.

The required ROM size depends on the customer and what needs to be stored in ROM, such as 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 64 Kbytes; for the CL484, it is greater than 64 Kbytes, requiring a 128 Kbyte ROM.

> *Note: If a ROM is connected to the CL48x, 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 Chapter 12.*

5.2 DRAM Memory Bus Interface

The DRAM memory bus on the CL48x 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 CL48x allows it to directly drive the DRAMs without external buffers or logic.

5.2.1 DRAM Address Mapping

The CL48x'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 bits 3 and 4 of A_MSB are LOW (see Figure 4-4). A_MSB[2] is used to select Bank 0 or Bank 1 through assertion of the $\overline{\text{RAS}}$[0] or $\overline{\text{RAS}}$[1] signals, respectively. Host address signals—A_LSB[7:0], A_MB[7:0], and A_MSB[1:0]—map directly into the multiplexed row and column DRAM addresses.

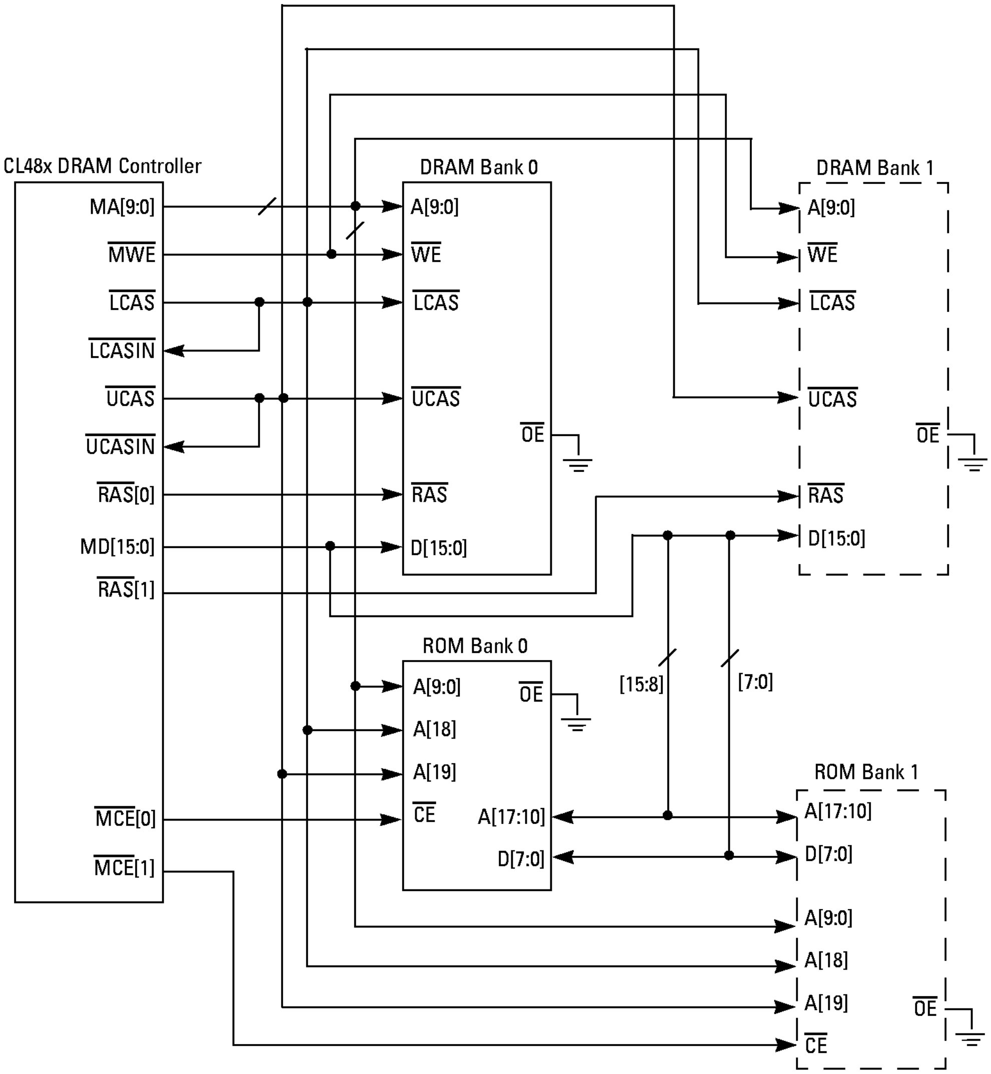


Figure 5-2 Local DRAM/ROM Implementation with 256K x 16 DRAMs

5.3 ROM Bus Interface and Timing

Figure 5-3 illustrates the ROM accesses. Two $\overline{\text{MCE}}$ signals are used as the ROM chip enables. 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.

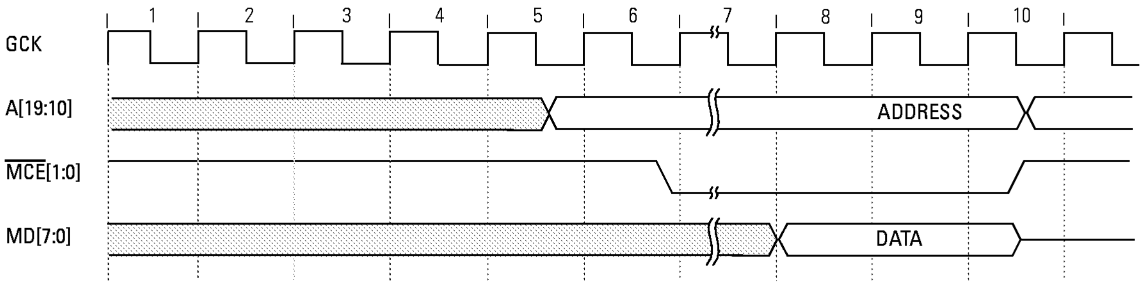


Figure 5-3 ROM Access Timing

5.4 DRAM Bus Timing

The DRAM interface on the CL48x is designed to work with fast page-mode DRAMs. Fast page-mode DRAMs allow shorter read and write cycle periods within a DRAM row. During a set of fast page-mode accesses, a row address is strobed in using the $\overline{\text{RAS}}$ line, followed by several column addresses strobed in by $\overline{\text{CAS}}$. Releasing the $\overline{\text{RAS}}$ line completes the set of cycles.

Refreshing is accomplished by asserting the $\overline{\text{CAS}}$ signal before asserting the $\overline{\text{RAS}}$ 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.

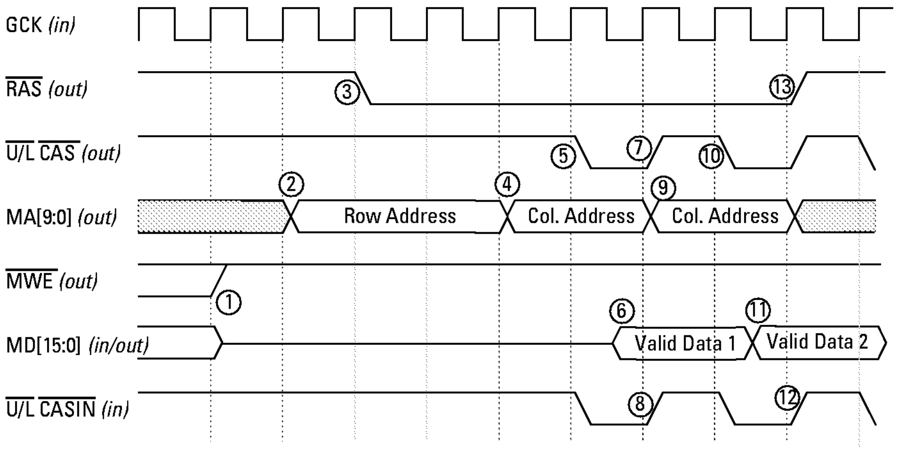


Figure 5-4 DRAM Page-Mode Read Timing

1. The CL48x starts a read cycle by three-stating the memory data bus, MD[15:0], and driving the write enable ($\overline{\text{MWE}}$) line HIGH.
2. The CL48x then drives the row address onto the memory address (MA) bus.
3. The CL48x asserts the $\overline{\text{RAS}}$ line (LOW) to latch the row address into the DRAM.
4. The CL48x then outputs the column address of the first word to be read on MA.
5. The CL48x 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 CL48x deasserts $\overline{\text{CAS}}$.
8. The data is loaded into the input latch of the CL48x on the rising edge of $\overline{\text{CASIN}}$. For a single-location read, the CL48x ends the cycle by deasserting $\overline{\text{RAS}}$ at this time.
9. For a page-mode read cycle, the CL48x outputs the column address of the next word to be read on MA[9:0].
10. The CL48x 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 CL48x on the rising edge of $\overline{\text{CASIN}}$.
13. The CL48x 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.

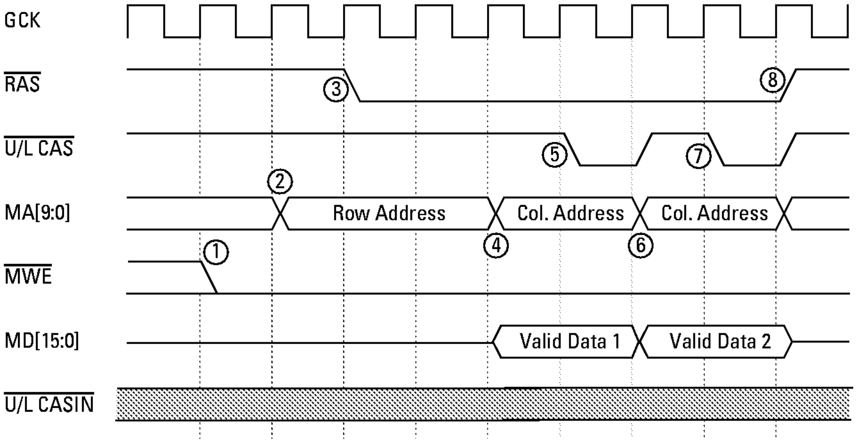


Figure 5-5 DRAM Page-Mode Write Timing

1. The CL48x starts a write cycle by asserting the write enable ($\overline{\text{MWE}}$) line LOW.
2. The CL48x then drives the row address on the memory address (MA) bus.
3. The CL48x asserts the $\overline{\text{RAS}}$ line (LOW) to latch the row address into the DRAM.
4. The CL48x 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 CL48x asserts the $\overline{\text{CAS}}$ line (LOW) to write the data into the selected location of the DRAM.
6. For a single-location write, the write operation is terminated by driving the $\overline{\text{RAS}}$ and $\overline{\text{CAS}}$ lines inactive (HIGH) at this point. For a page-mode write, the address and data for the next word to be written are output on the MA and MD buses, respectively.
7. The CL48x asserts the $\overline{\text{CAS}}$ line to write the data into the selected location of the DRAM.
8. When the last word is written, the write operation is terminated by driving the $\overline{\text{RAS}}$ and $\overline{\text{CAS}}$ lines HIGH.

5.4.3 DRAM Refresh Timing

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

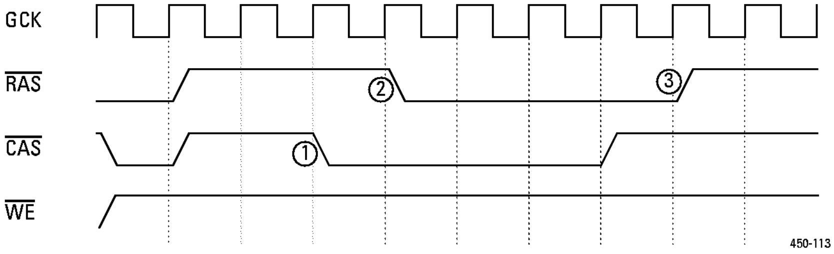


Figure 5-6 $\overline{\text{CAS}}$-before-$\overline{\text{RAS}}$ Refresh Cycle

1. The CL48x initiates a DRAM refresh cycle by asserting the $\overline{\text{CAS}}$ line while the $\overline{\text{RAS}}$ line is inactive (HIGH).
2. The CL48x asserts the $\overline{\text{RAS}}$ line to enable the refresh at the address pointed to by the DRAM's internal address counter.
3. The CL48x 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. The 13 LSb's of DRAM_REFRESH are used by the CL48x to determine the time between refreshes. This value is loaded into a counter after each page refresh so that, when the counter decrements to zero, the next page will be refreshed. The counter decrements by one (1) each GCK.

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

5.5 Memory Design Guidelines

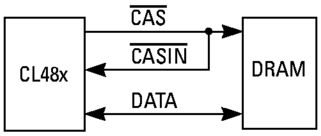
This section contains guidelines for the memory interface of the CL48x:

- It is important to place the DRAM very close to the CL48x. 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 CL48x 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 CL48x and the data side of the ROM is extended out past pin 33 of the CL48x.
- It is not necessary to use any termination if there is only one 256K x 16 DRAM connected to the CL48x 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 CL48x 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.

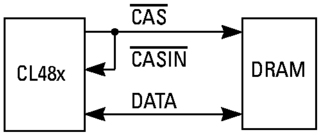
  Note: It is not necessary to program any of the internal registers in the CL48x when switching between these two kinds of DRAMs.

- 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 0.6 ns outside the specification for an 80-ns DRAM.
- The CL48x'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 routing of these signals 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, so that the return $\overline{\text{CAS}}$ line is connected as close to the

DRAM as possible (see below). $\overline{\text{CAS}}$ should not be routed 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 CL48x should match the path followed by the *data* lines as closely as possible.



CORRECT



INCORRECT

- The CL48x 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 CL48x 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 CL48x 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 CL48x 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 is designed to directly receive the output of the CD-DSP in bit-serial mode through the top four signal lines shown in Figure 6-1. Serial data from the CD-DSP may be in any of the following formats: CD-DA, CD-ROM, CD-ROM/XA, or CD-I.[1]

The bottom four signal lines provide an optional subchannel interface (defined in the Red Book) that allows the addition of low-bandwidth graphics information to the CD, commonly referred to as CD-G. CD-G is only provided in the CL484 microcode (see Section 11.12 for further details).

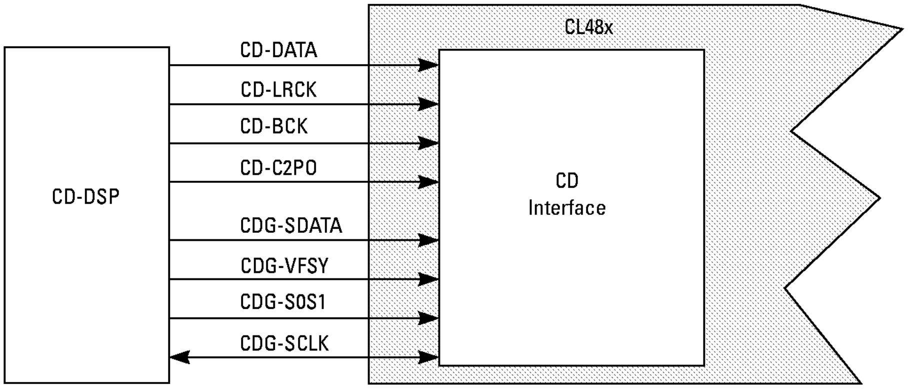


Figure 6-1 CD-Decoder Interface

[1] CD-I, as mentioned here, means full-motion video CD-I only; it does not apply to CD-I interactive applications. (VideoCD supports interactive applications.)

6.1.1 Mode Functions

The CL48x processes CD 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 CL48x error correction using the DumpData() command when not decoding MPEG)
  - Real-time processing and skipping of Mode 0 sectors
- In CD-DA mode:
  - Direct output (pass-through) to the audio interface (DA-signals) of the CD input
  - Ignores data error pin, CD-C2PO

CL48x CD standards, formats and error correction (EC) layers are shown in Table 6-1.

Table 6-1 CL48x-specific CD Standards and Formats

Standard	Mode	Form	EC Layer
	Mode 0	None	0
CD-ROM (Yellow Book)	Mode 1	None	3[1]
	Mode 2	Form 1	3[1]
CD-I (Green Book)	Mode 2	Form 1	3[1]
CD-ROM/XA (Yellow Book)			
Video CD (White Book)		Form 2	2
CD-DA and CD-G (Red Book)	None	None	2

[1] Note: The CL48x does the third layer of error correction only; the other two layers are always done in the CD-DSP.

The CL48x stores the header, subheader and other system information related to ROM decoding into dedicated DRAM locations which are accessible from the host.

> *Note: In a CL48x system with a host processor, the host would use the DumpData() command to extract error-corrected CD-ROM data from the CL48x's DRAM.*

In CD-ROM decoding, the CL48x does Q-erasure correction, then P-erasure correction, followed by a P-error check.

6.1.2 Mode Selection

The CL48x microcode can auto-detect the input bitstream type when the input bitstream is either a CD-DA or CD-ROM bitstream as Figure 6-2 shows.

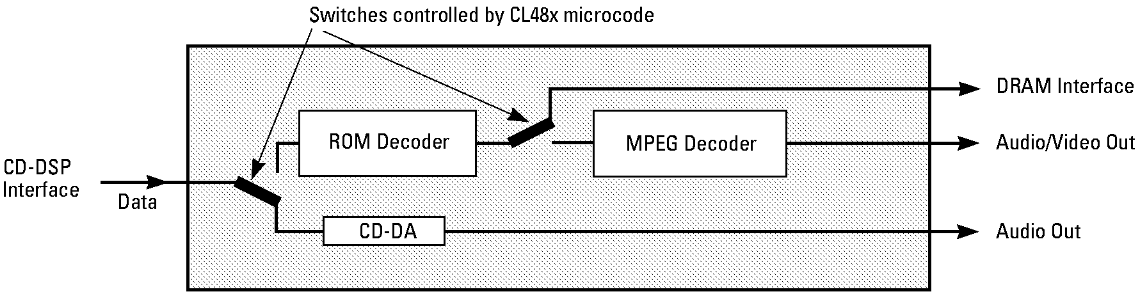


Figure 6-2 System Block Diagram Showing Bypass of ROM Decoder

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

The CL48x microcode uses the CD-ROM synchronization characters in CD-ROM bitstreams to determine whether the input bitstream is a CD-ROM or CD-DA bitstream. Before decoding begins, the CL48x 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 CL48x 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-DA, the incoming bitstream is expected to contain CD-DA data.

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

6.2 Signal Interface

The CL48x 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 CL48x that a byte contains an error. This signal is used when receiving CD-ROM data, but it is ignored in CD-DA pass-through mode.
- CD-LRCK - This pin provides 16-bit word synchronization. Its polarity (left/right channel HIGH) is programmable.
- CD-BCK - The bit clock, which may be set to variable speeds according to Table 6-2.

Table 6-2 BCK/Data Rate/LRCK Comparison for the CD-Decoder

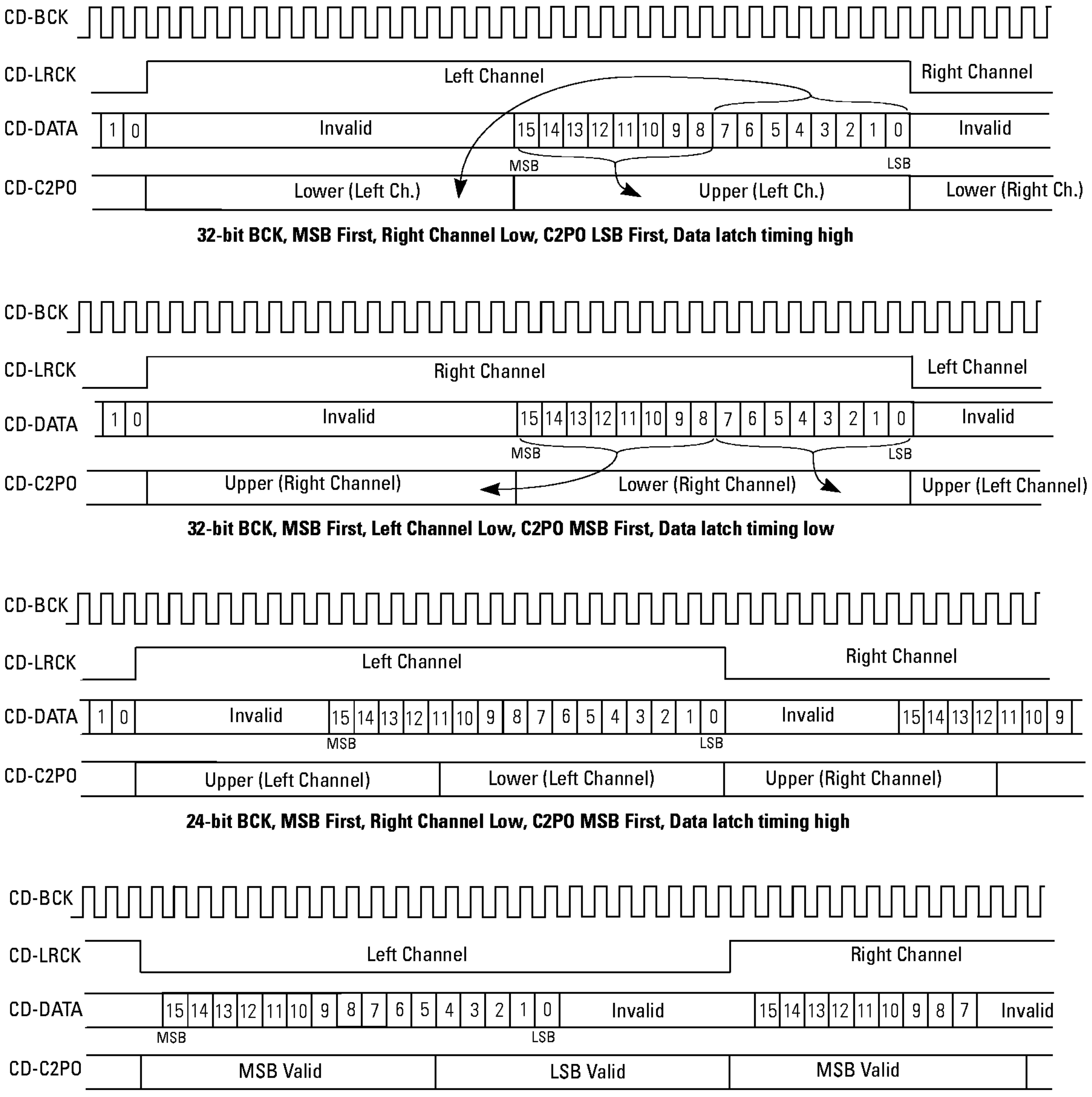
BCK	CD Speed	Data Rate	BCKs per LRCK
1.41 MHz	Normal	1.41 Mbits/s	32
2.12 MHz	Normal	1.41 Mbits/s	48
2.82 MHz	Normal	1.41 Mbits/s	64
2.82 MHz	Double	2.82 Mbits/s	32
4.23 MHz	Double	2.82 Mbits/s	48
5.64 MHz	Double	2.82 Mbits/s	64

6.3 Disc and Signal Formats

The next three subsections summarize CL48x 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 seven different serial modes.



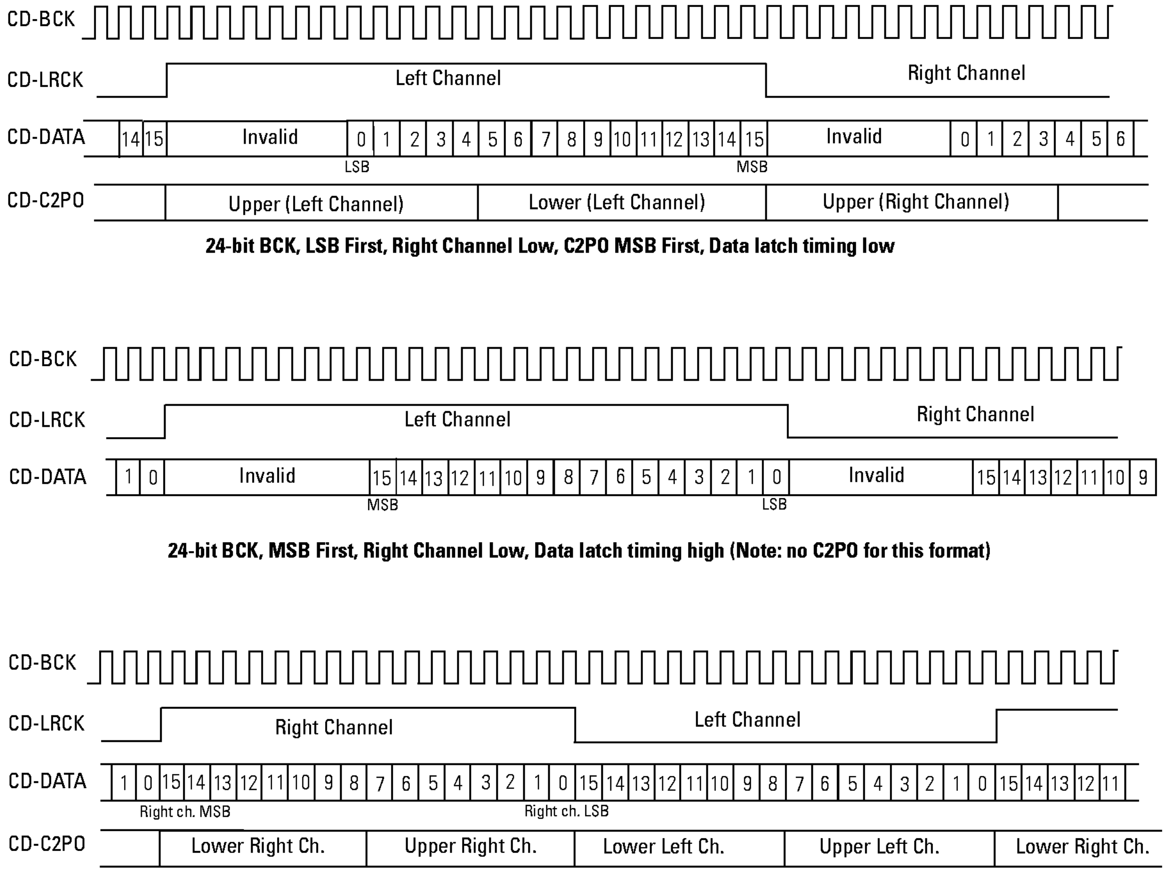
32-bit BCK, MSB First, Right Channel Low, C2PO LSB First, Data latch timing high

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

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

24-bit BCK, MSB First, Left Channel Low, C2PO MSB First, Data latch timing high (I^2S)

Figure 6-3 CD Interface Input Signal Formats



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

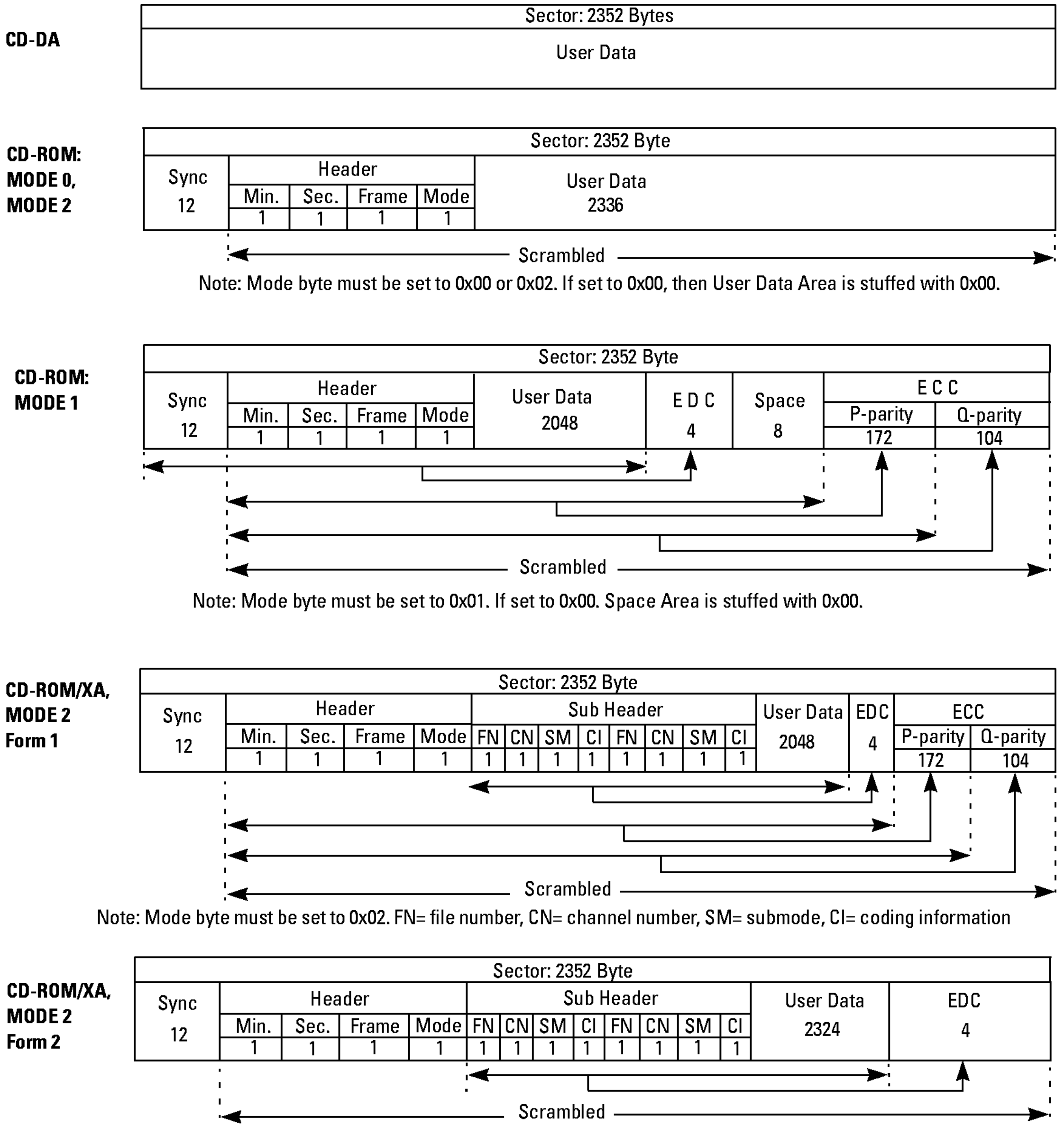
24-bit BCK, MSB First, Right Channel Low, Data latch timing high (Note: no C2PO for this format)

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

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.



CD-DA

Sector: 2352 Bytes
User Data

CD-ROM: MODE 0, MODE 2

Sector: 2352 Byte

Sync	Header				User Data
	Min.	Sec.	Frame	Mode	
12	1	1	1	1	2336

Note: Mode byte must be set to 0x00 or 0x02. If set to 0x00, then User Data Area is stuffed with 0x00.

CD-ROM: MODE 1

Sector: 2352 Byte

Sync	Header				User Data	EDC	Space	ECC	
	Min.	Sec.	Frame	Mode				P-parity	Q-parity
12	1	1	1	1	2048	4	8	172	104

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

CD-ROM/XA, MODE 2 Form 1

Sector: 2352 Byte

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

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

CD-ROM/XA, MODE 2 Form 2

Sector: 2352 Byte

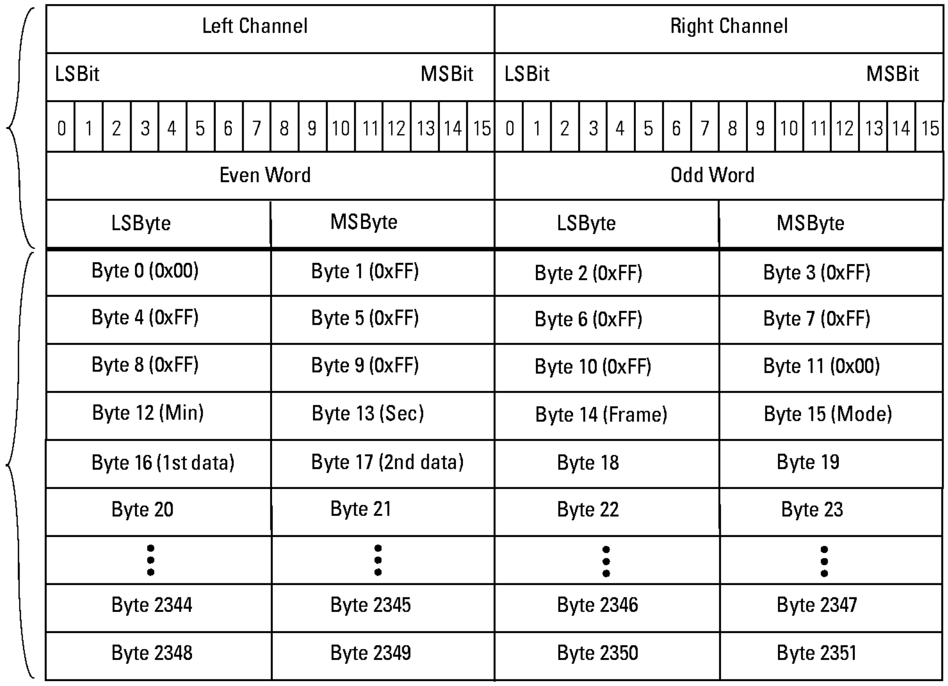
Sync	Header				Sub Header								User Data	EDC
	Min.	Sec.	Frame	Mode	FN	CN	SM	CI	FN	CN	SM	CI		
12	1	1	1	1	1	1	1	1	1	1	1	1	2324	4

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

Figure 6-4 Serial Data CD Formats

6.3.3 Data Alignment of 16-bit CD-DA and CD Data Bytes

Figure 6-5 shows the relationship between 16-bit CD-DA samples and how data bytes are aligned on a CD.



16-bit CD-DA	Left Channel		Right Channel	
	LSBit	MSBit	LSBit	MSBit
	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
	Even Word		Odd Word	
	LSByte	MSByte	LSByte	MSByte
CD Bytes	Byte 0 (0x00)	Byte 1 (0xFF)	Byte 2 (0xFF)	Byte 3 (0xFF)
	Byte 4 (0xFF)	Byte 5 (0xFF)	Byte 6 (0xFF)	Byte 7 (0xFF)
	Byte 8 (0xFF)	Byte 9 (0xFF)	Byte 10 (0xFF)	Byte 11 (0x00)
	Byte 12 (Min)	Byte 13 (Sec)	Byte 14 (Frame)	Byte 15 (Mode)
	Byte 16 (1st data)	Byte 17 (2nd data)	Byte 18	Byte 19
	Byte 20	Byte 21	Byte 22	Byte 23
	⋮	⋮	⋮	⋮
	Byte 2344	Byte 2345	Byte 2346	Byte 2347
	Byte 2348	Byte 2349	Byte 2350	Byte 2351

Figure 6-5 Data Alignment of 16-bit CD-DA and CD Data Bytes

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 CL48x 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 tempo control can be altered by changing the DA-XCK frequency and changing the speed that the disc spins. The CL48x will automatically skip or repeat pictures to maintain audio/video synchronization.

6.4.2 Error Correction and Data Rate

The maximum data rate that the CL48x can error correct is 2.8 Mbits per second, which is the double-speed CD rate.

The CL48x does third-layer error correction when not decoding MPEG. The first two layers of error correction are done in the CD-DSP.

7 Video Display Interface

This chapter 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 CL48x 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 ("slave mode") or generated by dividing GCK by three ("master mode"). 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 15,625 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, compressed video data is always stored in YCbCr format. If RGB-encoded output video is needed, the CL48x'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 CL48x'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 CL48x outputs pixels in either YCbCr (4:2:2 mode) or RGB (4:4:4 mode) using a VCK that may be generated either externally or internally. (VCK is programmed by setting values in the Configuration Area of DRAM; see Table 15-7.)

The CL48x supports the following video output modes:

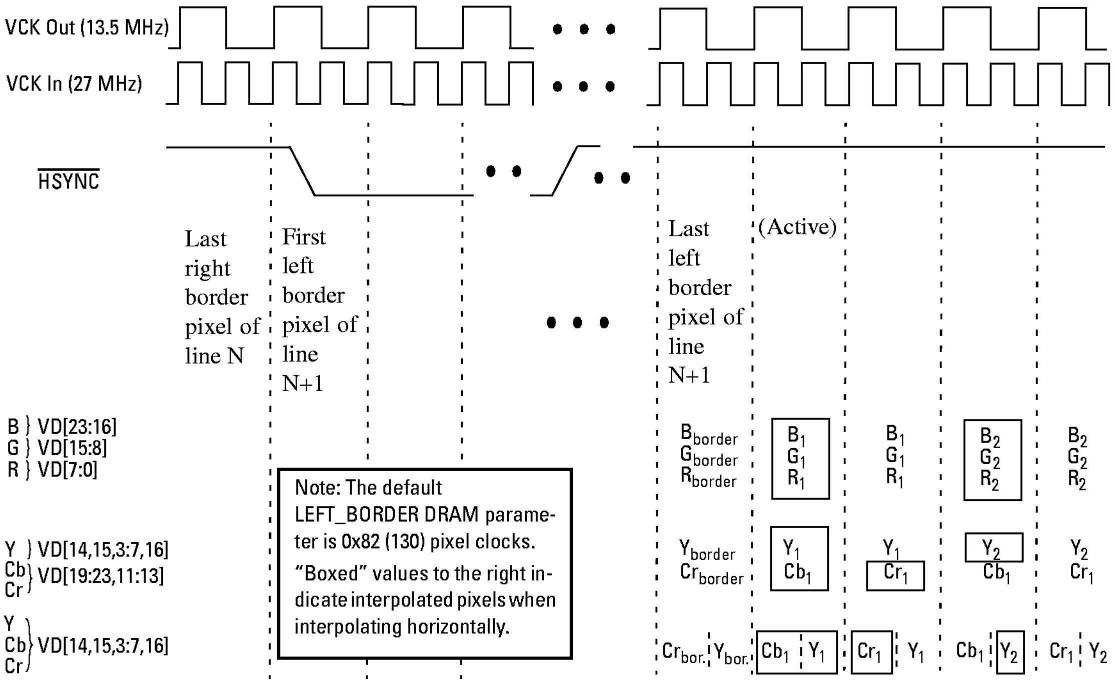
- *24-bit and 15-bit RGB*: In this mode, the B component is presented as the MSB and the R component as the LSB. The color components are dithered in 15-bit RGB mode.
- *16-bit YCbCr in 4:2:2*: In this mode, the Y component is interleaved with the Cb and Cr components in the repeating pattern of CbY-CrY.
- *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

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

[1] Note: In 15-bit RGB mode, only the upper five bits of R[7:0], G[7:0] and B[7:0] are used.


VCK Out (13.5 MHz)
VCK In (27 MHz)
$\overline{\text{HSYNC}}$
Last right border pixel of line N
First left border pixel of line N+1
Last left border pixel of line N+1
(Active)
B } VD[23:16]
G } VD[15:8]
R } VD[7:0]
Y } VD[14,15,3:7,16]
Cb, Cr } VD[19:23,11:13]
Y, Cb, Cr } VD[14,15,3:7,16]
B_{border} G_{border} R_{border}
B_1 G_1 R_1
B_1 G_1 R_1
B_2 G_2 R_2
B_2 G_2 R_2
Y_{border} Cr_{border}
Y_1 Cb_1
Y_1 Cr_1
Y_2 Cb_1
Y_2 Cr_1
$Cr_{bor.}$ $Y_{bor.}$ Cb_1 Y_1 Cr_1 Y_1 Cb_1 Y_2 Cr_1 Y_2

Note: The default LEFT_BORDER DRAM parameter is 0x82 (130) pixel clocks. "Boxed" values to the right indicate interpolated pixels when interpolating horizontally.

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

The first pixel in a line is output in the VCK period where $\overline{\text{HSYNC}}$ falls and on every other rising VCK edge after that. For a 13.5 MHz VCK, a new pixel is output every rising edge of VCK. For a 27 MHz VCK, a new pixel is output every other rising edge of VCK. The VCK-to-VD timing (Table 9-12) requires that VD be latched on the falling edge of VCK in VCK Out (master) mode.

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

Note: VCK is typically 27 MHz if the CL48x is operating in VCK In mode, and 13.5 MHz when the CL48x is operating in VCK Out mode.

Examples of the two possible video configurations are shown next.

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

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

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

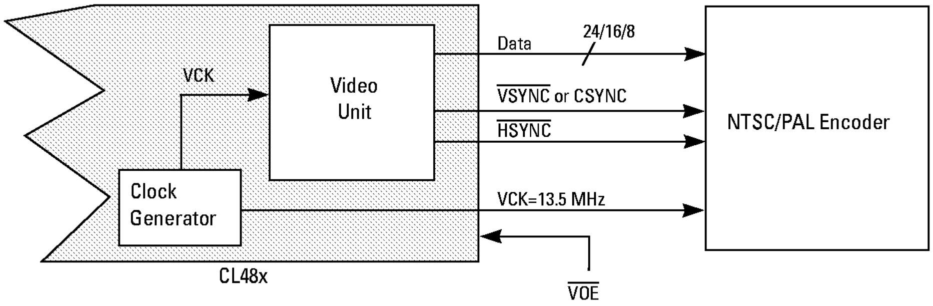


Figure 7-2 $\overline{\text{VSYNC}}$/$\overline{\text{HSYNC}}$/VCK Out Mode

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

In this mode, as shown in Figure 7-3, the video timing signals—VCK, $\overline{\text{VSYNC}}$ and $\overline{\text{HSYNC}}$—are generated by the external signal generator. VCK and GCK can be asynchronous, but VCK and $\overline{\text{HSYNC}}$ must be synchronous with each other. VCK is fixed to 27 MHz.

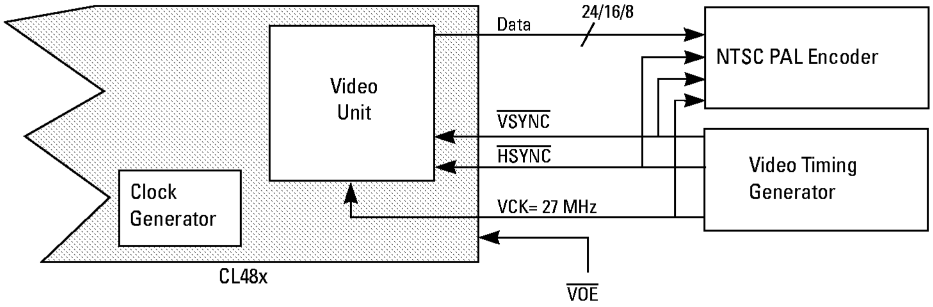


Figure 7-3 $\overline{\text{VSYNC}}$/$\overline{\text{HSYNC}}$/VCK In

7.4 On-screen Display (OSD)

CL48x microcode has the ability to display graphics images as overlays on top of normal video or on a blank screen. This feature is known as on-screen display (OSD). Graphics images are stored in the CL48x local DRAM and are displayed using the DisplayGraphics() command. Two formats are supported: run-length and bitmap.

See Section 11.13 and the description of the DisplayGraphics() command in Chapter 13 for more information.

See Appendix B for a detailed description of bitmap format.

7.5 Selecting Video Parameters

Video parameters 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 parameters described.

7.5.1 Video Color Space 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 via the RGB_MODE field of the VIDEO_MODE parameter. This is the default RGB mode.
- *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 CL48x 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).

7.5.2 Video Timing Signals: Type and Direction

The CL48x can be configured to accept or output video timing signals as follows:

Selecting CSYNC versus $\overline{\text{VSYNC}}$

In SYNC output mode, the CSYNC_VSYNC field of the VIDEO_MODE2 configuration parameter selects whether CL48x outputs CSYNC or VSYNC. In SYNC input mode, this bit must be set to 0 ($\overline{\text{VSYNC}}$ input only).

Selecting the Direction of VCK and $\overline{\text{HSYNC}}$/$\overline{\text{VSYNC}}$

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 $\overline{\text{HSYNC}}$ and $\overline{\text{VSYNC}}$ (the sync signals), respectively.

> *Note: The synchronization signals $\overline{\text{HSYNC}}$ and $\overline{\text{VSYNC}}$ are both either inputs or outputs; they cannot be configured separately.*

7.5.3 $\overline{\text{HSYNC}}$ and $\overline{\text{VSYNC}}$ as Output Parameters at 13.5 MHz (Master Mode)

The following parameters are selectable when the CL48x is generating both $\overline{\text{HSYNC}}$ and $\overline{\text{VSYNC}}$ signals for either NTSC or PAL video output; however, modifying these parameters from their default is generally not recommended:

> *Note: The units of all parameters are in pixel clocks.*

- *$\overline{\text{HSYNC}}$ LOW Time:* NTSC_HSYNC_LO and PAL_HSYNC_LO parameters specify the time $\overline{\text{HSYNC}}$ is LOW for both NTSC and PAL, respectively (see Figure 7-5).
- *$\overline{\text{HSYNC}}$ Period:* NTSC_HSYNC_PER and PAL_HSYNC_PER parameters specify the $\overline{\text{HSYNC}}$ period for both NTSC and PAL, respectively (see Figure 7-5).
- *$\overline{\text{VSYNC}}$ LOW Relative to $\overline{\text{HSYNC}}$ LOW:* VSYNC_TOP_TP and VSYNC_BOT_TP specify the delay time from $\overline{\text{HSYNC}}$ going LOW (for both top and bottom fields, respectively) to the start of $\overline{\text{VSYNC}}$ going LOW (see Figure 7-5). The default for VSYNC_TOP_TP is 0; the default for VSYNC_BOT_TP is 0x180 (384) pixel clocks.

> *Note: The VSYNC_TOP_TP and VSYNC_BOT_TP parameters are used for both NTSC and PAL.*

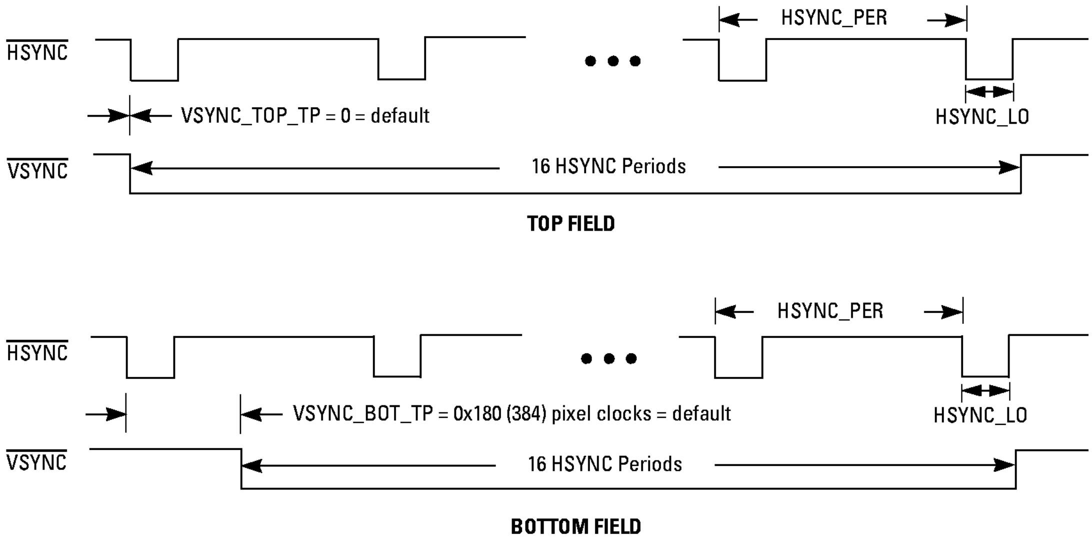

HSYNC_PER
$\overline{\text{HSYNC}}$
VSYNC_TOP_TP = 0 = default
HSYNC_LO
$\overline{\text{VSYNC}}$
16 HSYNC Periods
TOP FIELD
HSYNC_PER
$\overline{\text{HSYNC}}$
VSYNC_BOT_TP = 0x180 (384) pixel clocks = default
HSYNC_LO
$\overline{\text{VSYNC}}$
16 HSYNC Periods
BOTTOM FIELD

Figure 7-5 $\overline{\text{HSYNC}}$ **and** $\overline{\text{VSYNC}}$ **Output Timing (Master Mode)**

7.5.4 $\overline{\text{HSYNC}}$ and $\overline{\text{VSYNC}}$ as Input Parameters at 27 MHz (Slave Mode)

If the $\overline{\text{HSYNC}}$ and $\overline{\text{VSYNC}}$ signals are supplied externally, the CL48x, at startup or when the video format is changed, will enter field-detection mode. In this mode, the CL48x automatically compares the falling edge of $\overline{\text{VSYNC}}$ to the $\overline{\text{HSYNC}}$ pulse to determine which field, top or bottom, is displayed, as shown in Figure 7-6.

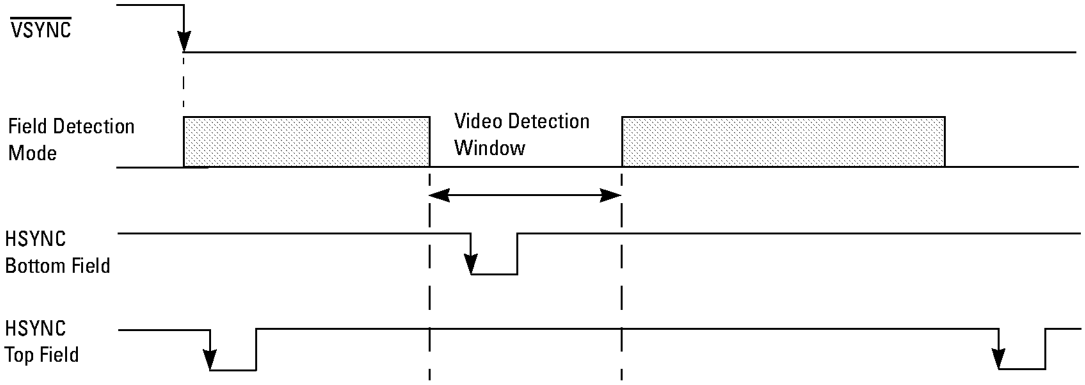


Figure 7-6 $\overline{\text{HSYNC}}$ **and** $\overline{\text{VSYNC}}$ **Input Timing (Slave Mode)**

If the falling edge of $\overline{\text{HSYNC}}$ is *inside* the video detection window generated by the falling edge of $\overline{\text{VSYNC}}$, as in the example shown in Figure 7-6, the top field is displayed. If the falling edge of $\overline{\text{HSYNC}}$ is *outside* the video window generated by the falling edge of $\overline{\text{VSYNC}}$, the bottom field is displayed. See Table 7-2.

Table 7-2 Field Display Modes when $\overline{\text{HSYNC}}$ and $\overline{\text{VSYNC}}$ are Inputs

Position of $\overline{\text{HSYNC}}$ Leading Edge Generated by Falling Edge of $\overline{\text{VSYNC}}$	Field Displayed
Inside video detection window	Bottom
Outside video detection window	Top

In either case of a top or bottom field display, the FIRST_FIELD configuration parameter can be used to invert the results. A 0 value in the FIRST_FIELD parameter will maintain the default value; a 1 value in the FIRST_FIELD parameter will invert the default value.

7.5.5 Video Formats

The CL48x supports one of the following four video output formats:

- PAL
- NTSC
- Multisync (PAL or NTSC)

For more information, see the SetVideoFormat() command in Chapter 13 or the VIDEO_FORMAT DRAM parameter in Chapter 15.

> *Note: If the output video frame rate doesn't match the frame rate of the incoming video bitstream, the CL48x will automatically convert the frame rate of the incoming video bitstream to match the display frame rate.*

7.5.6 LEFT_BORDER and TOP_BORDER

These parameters (described next and in Table 15-7) specify where the upper-leftmost corner of the display is to start. Setting either LEFT_BORDER or TOP_BORDER to zero will cause the display to be centered in that dimension.

> *Note: The SetVideoFormat() command must be issued after LEFT_BORDER and TOP_BORDER are changed in order for them to take effect.*

LEFT_BORDER

After the display is centered horizontally, LEFT_BORDER specifies the number of VCKs from the centered position to start video output. The following equation is used to calculate the horizontal video start position:

$$\text{HorizStartPosition} = \text{LEFT\_BORDER} + \frac{\text{VideoDisplayWidth} - \text{VideoSourceWidth}}{2}$$

LEFT_BORDER has a minimum value of 0x70, a maximum value of 0x98, and a default value of 0x82; it must be set to a positive value. The default value is 704 for both VideoDisplayWidth and VideoSourceWidth.

> *Note: For CD-G use, see the description of this parameter given in the SetVideoFormat(**format, 0, 0**) command in Chapter 13.*

TOP_BORDER

After the display is centered vertically, TOP_BORDER specifies the number of lines from the centered position to start video output. The following equation is used to calculate the vertical video start position:

$$\text{VertStartPosition} = \text{TOP\_BORDER} + \frac{\text{VideoDisplayHeight} - \text{VideoSourceHeight}}{2}$$

The TOP_BORDER parameter default and minimum value is 1. TOP_BORDER may be set to either positive or negative values. The default value is 240 for NTSC (288 for PAL) for both VideoDisplayHeight and VideoSourceHeight.

> *Note: For CD-G use, see the description of this parameter given in the SetVideoFormat(**format, 0, 0**) command in Chapter 13.*

7.5.7 Border Color

The display border color is controlled by the COLOR_CR and the COLOR_Y_CB Configuration Area DRAM parameters. Each of these parameters are described in Table 15-7.

7.5.8 Interpolation

The CL48x performs both horizontal and vertical interpolation using the TOP_INTERPCOEF and BOT_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 TOP_INTERPCOEF and BOT_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.

Vertical Interpolation

Vertical interpolation is used to generate two fields from the encoded SIF frame. Vertical interpolation is controlled by using the TOP_INTERPCOEF and BOT_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 the vertical fields is 0 = 0, 1 = 1/4, 2 = 1/2, and 3 = 3/4. Interpolation is performed separately for each field output, and interpolation coefficients are provided for each field. (See Table 15-7.)

7.5.9 Changing the Color Palette

The TRANSPARENT configuration parameter allows you to specify which color numbers in the graphics stream will be displayed as transparent (See Table 15-7), while the Y01, Y23, U01, U23, V01, and V23 configuration parameters allow you to define the Y, U, and V values for color palette locations 0 and 1, and 2 and 3, respectively (See Table 15-7).

The TRANSPARENT, Y01, Y23, U01, U23, V01 and V23 configuration parameters will only take effect when the UPDATE_GRAPHICS configuration parameter is set to one. The UPDATE_GRAPHICS configuration parameter will be set to zero after these parameters have been changed.

For a discussion of OSD (on-screen display), see Section 11.13. For a description of C-Cube bitmap format, see Appendix B. For a description of C-Cube run-length format, contact a C-Cube representative.

8
Audio Interface

8.1 General Description

This interface is designed to output audio samples in bit-serial format directly from the CL48x to the audio DACs, as shown in Figure 8-1.

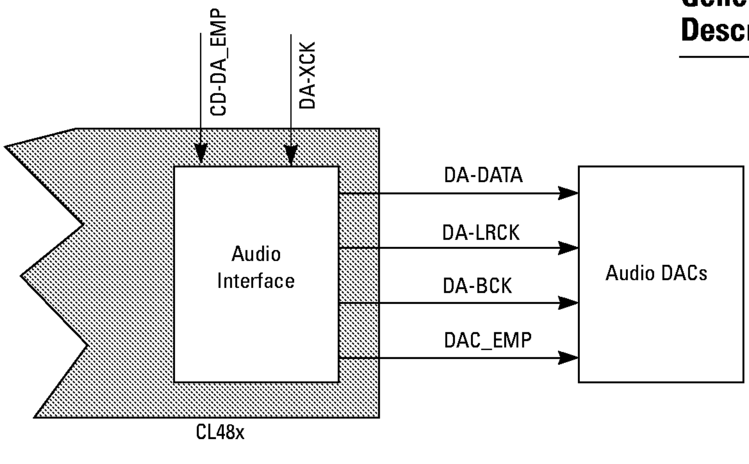


Figure 8-1 Audio Interface Block Diagram

8.2 Signal Interface

Table 8-1 summarizes the functions of the CL48x audio signals.

Table 8-1 Audio Interface Signals

Signal	Direction	Description
DA-DATA	Out	Outputs audio samples serially.
DA-BCK	Out	The audio bit clock. Divided by 8 from DA-XCK, it is either 48 or 32 times the sample clock.
DA-LRCK	Out	The clock that identifies the channel for each audio sample.
DAC_EMP	Out	Audio emphasis signal. Routed to the audio DACs for both CD-DA and MPEG audio.
DA-XCK	In	External clock used to generate DA-BCK and DA-LRCK, DA-XCK is either 384 or 256 times the sampling frequency. Use OUTPUT_24_BITS field of the AUDIO_CONFIG DRAM parameter to indicate which is used.
CD-DA_EMP	In	In CD-DA pass-through mode, this input signal is routed directly to DAC_EMP.

8.2.1 Configuring Data Output

Table 8-2 shows output options available with the CL48x. The CL48x can output either channel 0 (left) or channel 1 (right) data to both L and R channels if desired. The bits OUTPUT_24_BITS, RIGHT_LRCK_LO and LEFT_LRCK_HI may be selected from the AUDIO_CONFIG parameter described in Table 15-7.

Table 8-2 Audio Output 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 Setting the DAC_EMP Signal

In CD-DA pass-through mode, CD-DA_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 DAC_EMP pin connects directly to the EMP pin of an audio DAC. The output of the CL48x's DAC_EMP pin is determined by the microcode in MPEG audio mode, and may be programmed by the host in CD-DA mode as shown in Table 8-3.

Table 8-3 DAC_EMP Output Determination

CD-DA_EMPHASIS Parameter	CD-DA_EMP Input Signal	DAC_EMP Output Signal
00	X	0
01	X	1
1X	0	0
1X	1	1

8.2.3 Controlling the Audio Output Level

The amplitude of the audio data is controlled by the AUDIO_MUTE configuration parameter, through which sixty-four amplitude settings are possible. Value zero corresponds to no change in amplitude, each increment to the value of AUDIO_MUTE results in a 2dB attenuation in audio power (for example, a value of 3 results in half the normal audio amplitude), and value 63 results in zero audio output. Negative values, indicating amplification, should not be used because of the risk of numeric overflow in the audio decode calculations and the resultant corruption of the audio output. (See Table 15-7 for parameter settings.)

8.3 Clocking Modes

The audio interface unit supports a sampling frequency of 44.1 kHz. An external source, DA-XCK, is used to generate the DA-BCK and DA-LRCK signals.

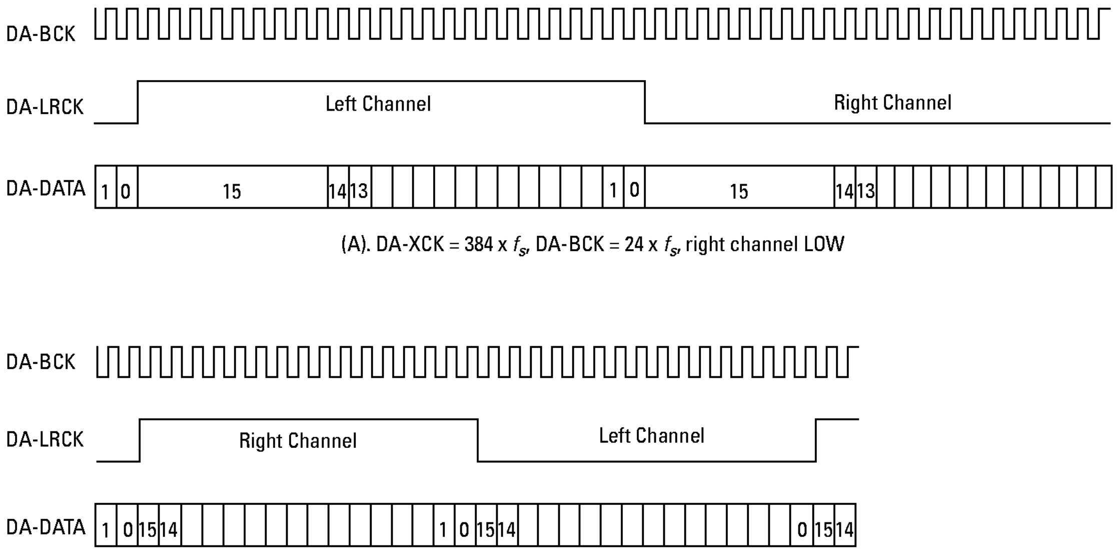
The external reference clock, DA-XCK, will be either 384 or 256 times the sampling frequency. The bit clock signal, DA-BCK, is derived from the external reference clock, DA-XCK, by dividing by 8. This gives a bit clock value of either 48 or 32 times the sampling clock.

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.



(A). DA-XCK = 384 x f_s, DA-BCK = 24 x f_s, right channel LOW

(B). DA-XCK = 256 x f_s, DA-BCK = 16 x f_s, left channel LOW

Figure 8-2 Audio Interface Clocking Modes

8.4 Operation and Design Guidelines

The CL48x 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.

9 Electrical and Physical Specifications

This chapter describes the CL48x's electrical and mechanical characteristics.

9.1 Operating Conditions

Tables 9-1 through 9-3 specify the CL48x's electrical characteristics.

Table 9-1 Operating Conditions

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

Table 9-2 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 XTL IN (pin 19), which should be 3.8V.

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		180	220	mA
I_{DD33}	Supply Current @ V_{DD3} = 3.3V	GCK = 40MHz V_{IN} = 0 or 3.3V, C_L=50pF, Ta=70°C		240	290	mA
I_{DD36}	Supply Current @ V_{DD3} = 3.6V	GCK = 40MHz V_{IN} = 0 or 3.6V, C_L=50pF, Ta=70°C		270	320	mA
C_{IN}	Input Capacitance [1]			10		pF
C_{OUT}	Output Capacitance [1]			12		pF
CI/O	Output Capacitance [1]			12		pF

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

[2] 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 CL48x was designed for GCK and VCK duty cycles of 55/45.

9.1.2 Power Pin Supply Voltages

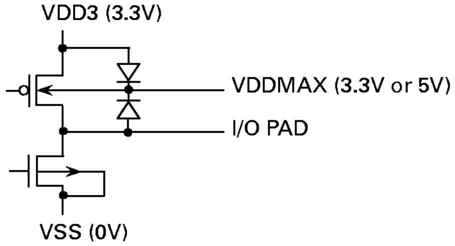
The VDD3 pins (7, 21, 22, 29, 49, 65, 77, 81, 98, 110, and 112) should be connected to 2.7 to 3.6 volts. The maximum voltage output by the CL48x is the voltage on these pins.

The VDDMAX 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 (VDDMAX) is connected to the N-well of the P-channel output transistor in the I/O pads.

The source terminal of this transistor (VDD3) is connected to 3.3 volts. If the 3.3-volt supply is more than about 0.6 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. (See Figure below.)



9.2 AC Timing Characteristics

This section describes the AC timing characteristics of the CL48x. The timing characteristics are divided into the following groups:

- GCK (XTLIN), $\overline{\text{RESET}}$, and CFLEVEL Timing
- Host Bus Interface Timing
- DRAM/ROM Bus Timing
- CD Interface Timing
- CD Subcode (CD-G) Interface Timing
- Video Bus Timing
- Audio Bus Timing

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

This section describes CL48x GCK, $\overline{\text{RESET}}$ and CFLEVEL timing.

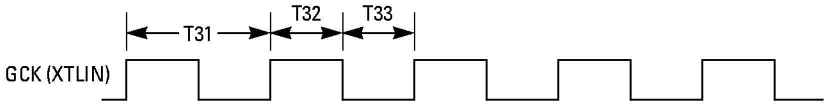


Figure 9-1 Timing Diagram - GCK (XTLIN)

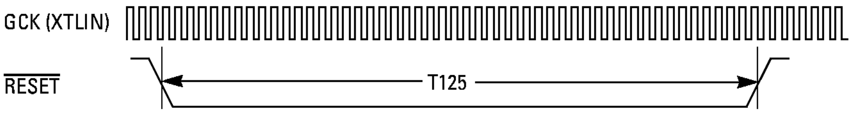


Figure 9-2 Timing Diagram - $\overline{\text{RESET}}$

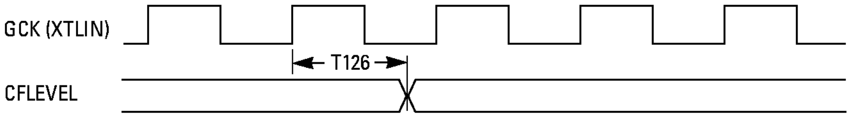


Figure 9-3 Timing Diagram - CFLEVEL

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	15 x 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 CL48x.

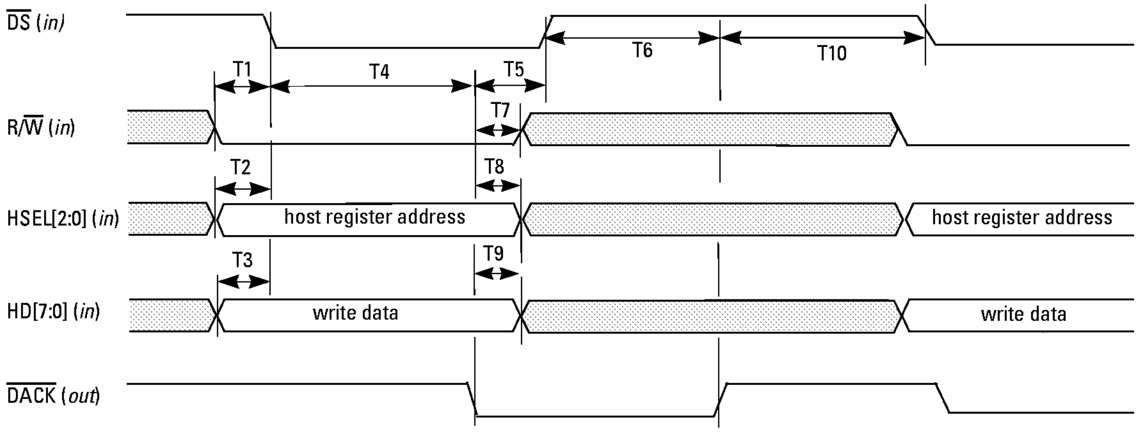


Figure 9-4 Host Interface Local Register, Write Operation

Table 9-5 Timing Characteristics: Local Register Write

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

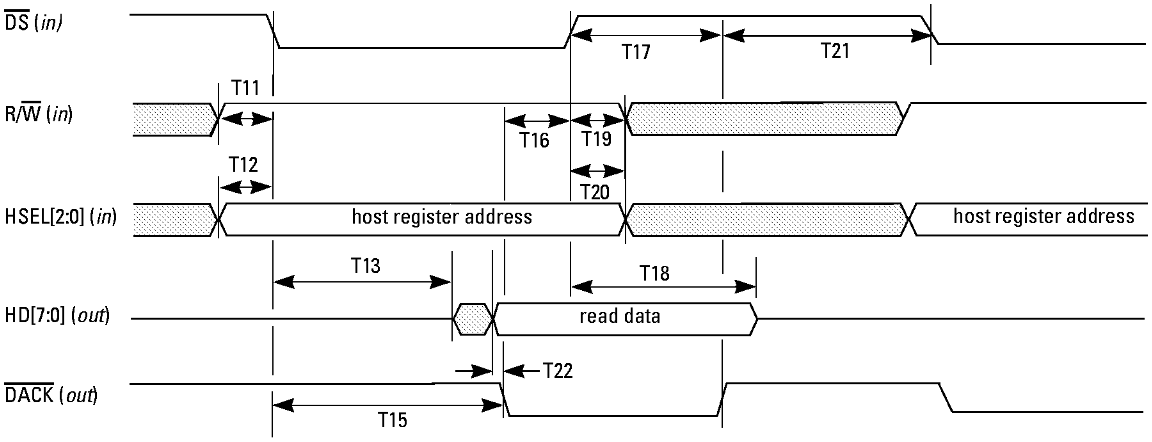


Figure 9-5 Host Interface Local Register, Read Operation

Table 9-6 Timing Characteristics: Local Register Read

Time[1]	Description	Min	Max	Units
T11	R/$\overline{W}$ valid to $\overline{DS}$ LOW	0		ns
T12	HSEL[2:0] valid to $\overline{DS}$ LOW	0		ns
T13	$\overline{DS}$ LOW to HD[7:0] (non-3-state)	20		ns
T15	$\overline{DS}$ LOW to $\overline{DACK}$ LOW	(note)[2]	(note)[3]	ns
T16	$\overline{DACK}$ LOW to $\overline{DS}$ HIGH	3		ns
T17	$\overline{DS}$ HIGH to $\overline{DACK}$ HIGH	1	3	GCK
T18	$\overline{DS}$ HIGH to HD[7:0] 3-state	1	3	GCK
T19	R/$\overline{W}$ hold after $\overline{DS}$ HIGH	0		ns
T20	HSEL[2:0] hold after $\overline{DS}$ HIGH	0		ns
T21	$\overline{DACK}$ HIGH to $\overline{DS}$ LOW (next operation)	5		ns
T22	HD[7:0] setup to $\overline{DACK}$ 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] A3 silicon is -10 ns.

9.2.3 DRAM/ROM Bus Timing

Local DRAM/ROM Bus Timing is shown in Figures 9-6 and 9-7.

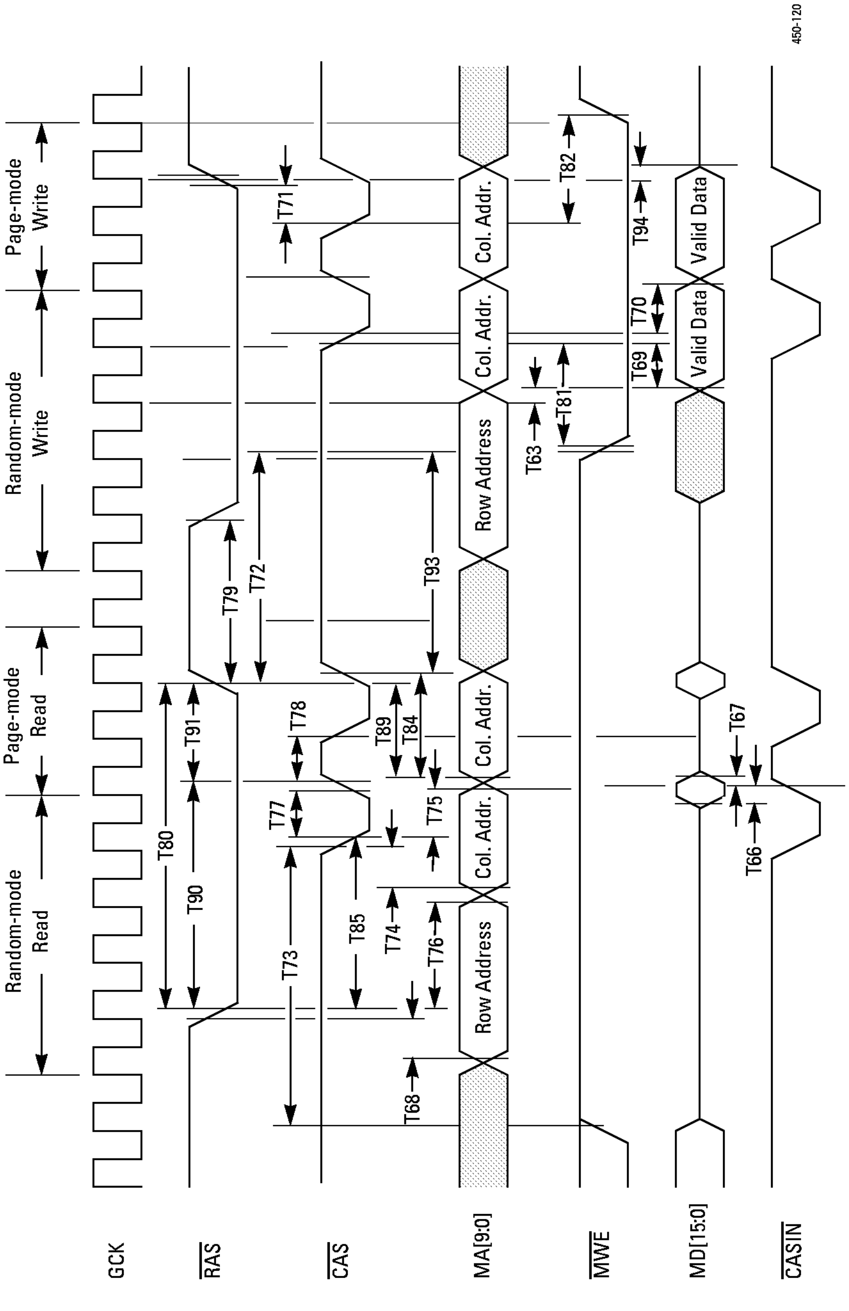


Figure 9-6 Local DRAM Bus Timing

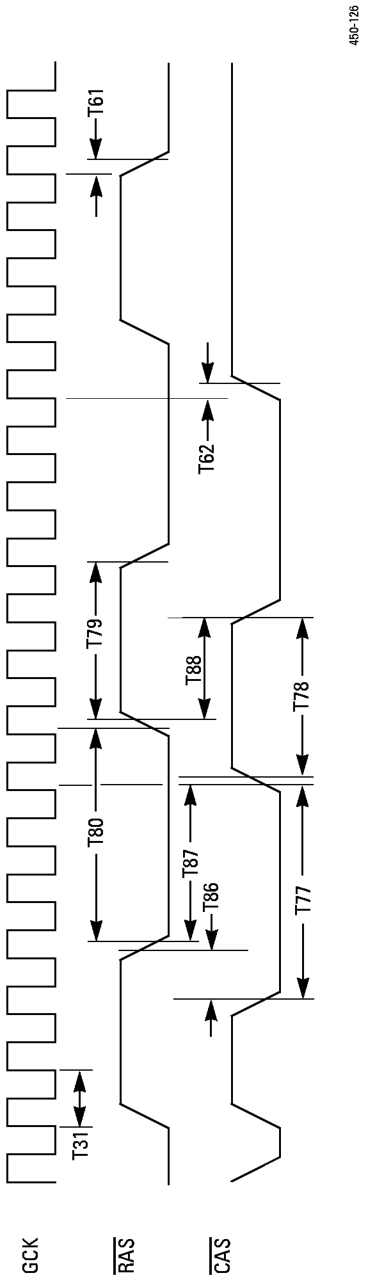


Figure 9-7 **Local DRAM $\overline{CAS}$-before-$\overline{RAS}$ Refresh**

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

Time	Description	Parameter	Min	Max	Units
T66	Read Data Setup Time before $\overline{\text{CASIN}}$ HIGH		5		ns
T67	Read Data Hold Time after $\overline{\text{CASIN}}$ HIGH		5		ns
T68	Row Address Setup Time [2]	t_{ASR}	5		ns
T69	Write Data Setup Time before $\overline{\text{CAS}}$ LOW [2]	t_{DS}	5		ns
T70	Write Data Hold Time after $\overline{\text{CAS}}$ LOW [2]	t_{DH}	15		ns
T71	$\overline{\text{RAS}}$ Hold Time after $\overline{\text{CAS}}$ LOW [2]	t_{RSH}	20		ns
T72	Read Command Hold Time from $\overline{\text{RAS}}$ [2]	t_{RRH}	2		ns
T73	Read Command Setup Time to $\overline{\text{CAS}}$ LOW [2]	t_{RCS}	15		ns
T74	Column Address Setup Time to $\overline{\text{CAS}}$ LOW [2]	t_{ASC}	5		ns
T75	Column Address Hold Time from $\overline{\text{CAS}}$ LOW [2]	t_{CAH}	15		ns
T76	Row Address Hold Time from $\overline{\text{RAS}}$ LOW [2]	t_{RAH}	10		ns
T77	$\overline{\text{CAS}}$ LOW Time [2]	t_{CAS}	21		ns
T78	$\overline{\text{CAS}}$ HIGH Time [2]	t_{CP}	10		ns
T79	$\overline{\text{RAS}}$ HIGH Time [2]	t_{RP}	62		ns
T80	$\overline{\text{RAS}}$ LOW Time [2]	t_{RAS}	80	5000	ns
T81	Write Setup time to $\overline{\text{CAS}}$		10		ns
T82	Write Command Hold Time from $\overline{\text{CAS}}$ LOW [2]	t_{WCH}	15		ns
T84	Column Address to $\overline{\text{CAS}}$ HIGH [2]	t_{AA} + T66	45		ns
T85	$\overline{\text{RAS}}$ to $\overline{\text{CAS}}$ delay [2]	t_{RCD}	50		ns
T86	$\overline{\text{CAS}}$ Setup Time to $\overline{\text{RAS}}$ (Memory Refresh Cycle) [2]	t_{CSR}	10		ns
T87	$\overline{\text{CAS}}$ Hold Time from $\overline{\text{RAS}}$ (Memory Refresh Cycle) [2]	t_{CHR}	15		ns
T88	$\overline{\text{RAS}}$ HIGH to $\overline{\text{CAS}}$ LOW delay (Memory Refresh Cycle) [2]	t_{RPC}	0		ns
T89	Column Address to $\overline{\text{RAS}}$ HIGH [2]	t_{RAL}	40		ns
T90	$\overline{\text{RAS}}$ LOW to $\overline{\text{CAS}}$ HIGH [2]	t_{RAC} + T66	85		ns
T91	$\overline{\text{RAS}}$ Hold Time from $\overline{\text{CAS}}$ Precharge [2]	t_{RHCP}	47		ns
T93	Read Command Hold Time from $\overline{\text{CAS}}$	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.

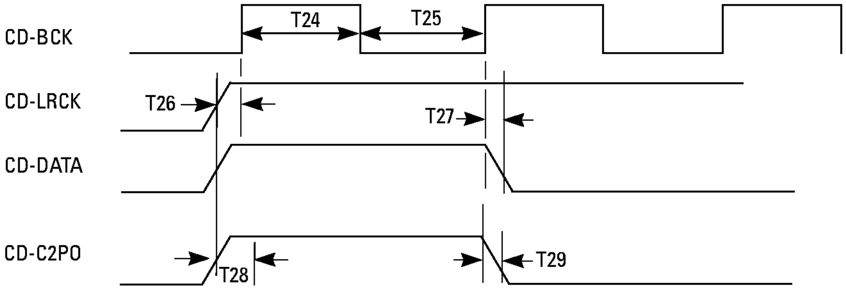


Figure 9-8 CD Input: Data Latch Timing High

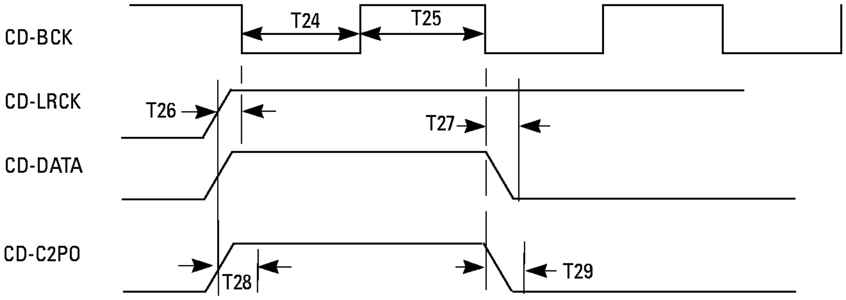


Figure 9-9 CD Input: Data Latch Timing Low

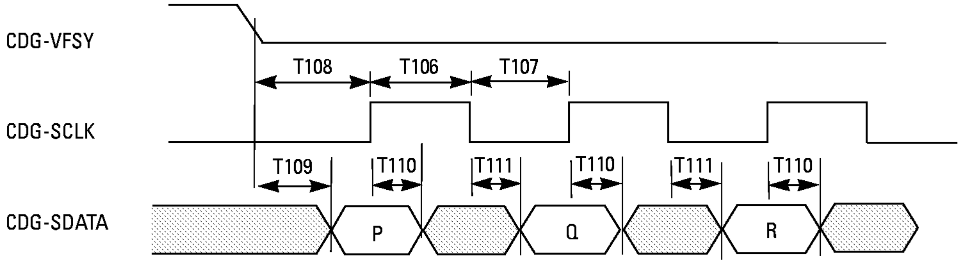
Table 9-8 Timing Characteristics: CD Input[1]

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 CL48x 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.

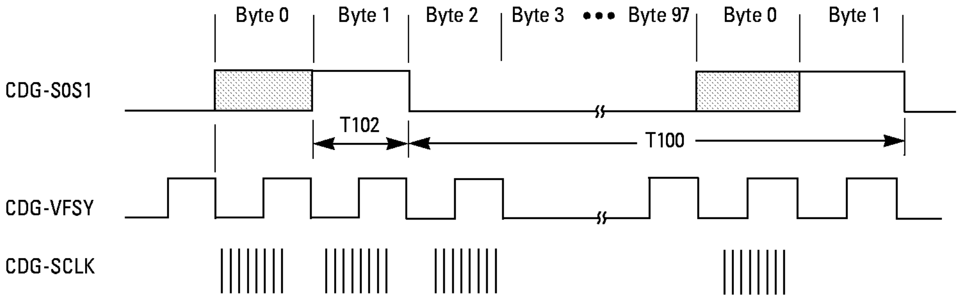


Figure 9-11 Separate CD Subcode Control Signals

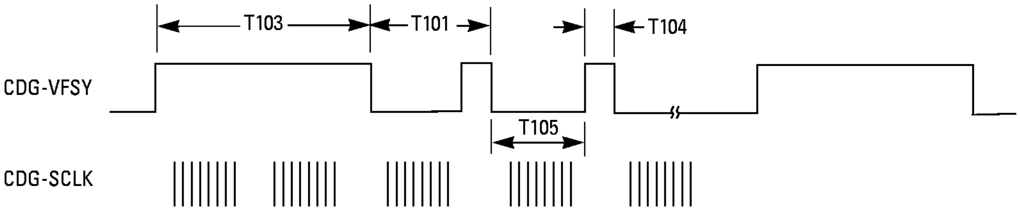


Figure 9-12 Composite CD Subcode Control Signals

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.

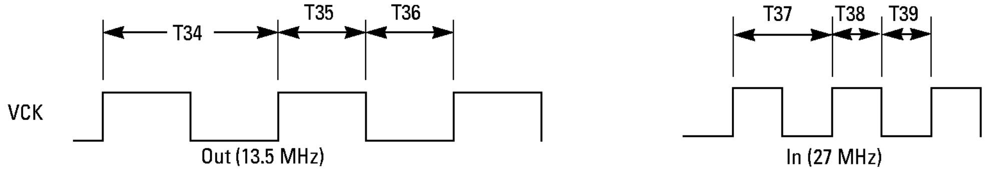


Figure 9-13 Timing - VCK (In and Out)

Table 9-10 Timing Characteristics: VCK

Time[1]	Description	Min	Max	Units
T34	VCK (Out) Frequency (13.5 MHz)	3	3	GCKs
T35	VCK (Out) HIGH[2]	33	42	ns
T36	VCK (Out) LOW[2]	33	42	ns
T37	VCK (In) Frequency (27 MHz)	26.8	27.2[3]	MHz
T38	VCK (In) HIGH	16		ns
T39	VCK (In) LOW	16		ns

1. Not 100% tested, guaranteed by design.
2. Assumes GCK frequency is 40.5 MHz.
3. CD-G requires a maximum VCK (In) of 28.6 MHz for correct pixel aspect ratio.

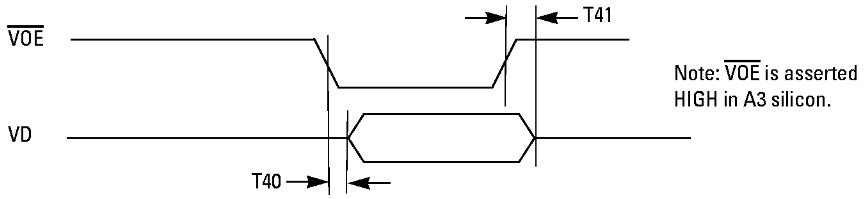


Figure 9-14 Timing - $\overline{\text{VOE}}$ and VD

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

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

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

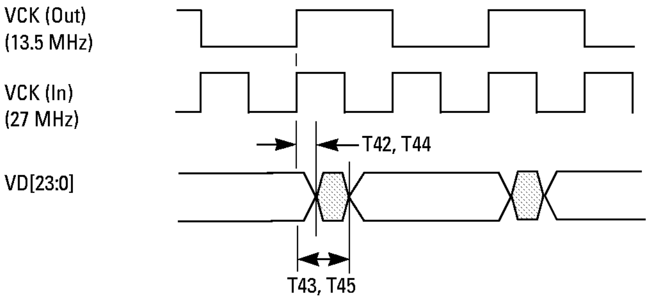
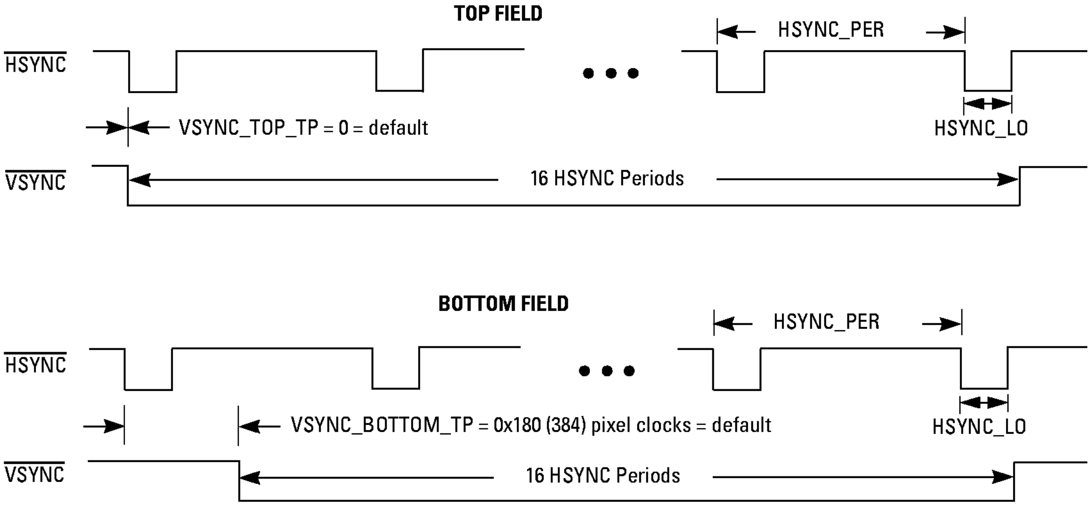


Figure 9-15 Timing - VCK and VD (pixel data)

Table 9-12 Timing Characteristics: VCK and VD

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

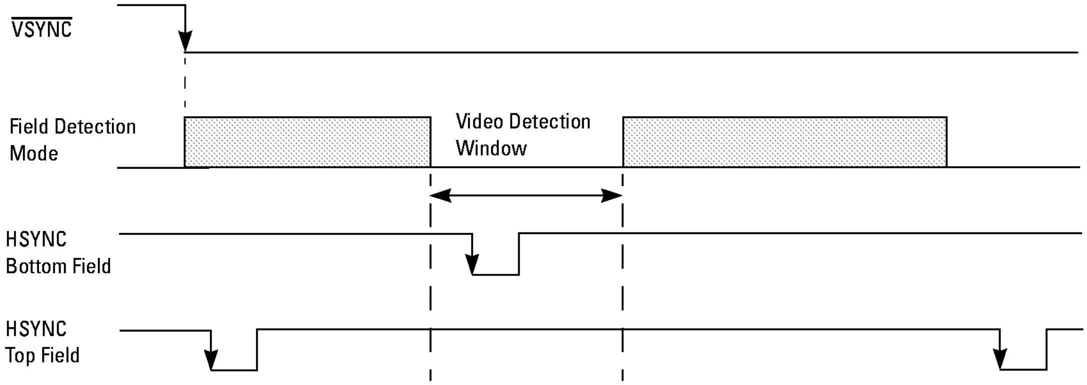
[1] Not 100% tested, guaranteed by design.


TOP FIELD
$\overline{\text{HSYNC}}$
HSYNC_PER
HSYNC_LO
VSYNC_TOP_TP = 0 = default
$\overline{\text{VSYNC}}$
16 HSYNC Periods
BOTTOM FIELD
$\overline{\text{HSYNC}}$
HSYNC_PER
HSYNC_LO
VSYNC_BOTTOM_TP = 0x180 (384) pixel clocks = default
$\overline{\text{VSYNC}}$
16 HSYNC Periods

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.

Figure 9-16 $\overline{\text{VSYNC}}$ and $\overline{\text{HSYNC}}$ Out

Note: When $\overline{\text{HSYNC}}$ and $\overline{\text{VSYNC}}$ are outputs (Figure 9-16), the position of the $\overline{\text{VSYNC}}$ 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.

Figure 9-17 $\overline{\text{VSYNC}}$ and $\overline{\text{HSYNC}}$ In

For $\overline{\text{VSYNC}}$/$\overline{\text{HSYNC}}$ In, the minimum $\overline{\text{HSYNC}}$ and $\overline{\text{VSYNC}}$ LOW times are both one VCK.

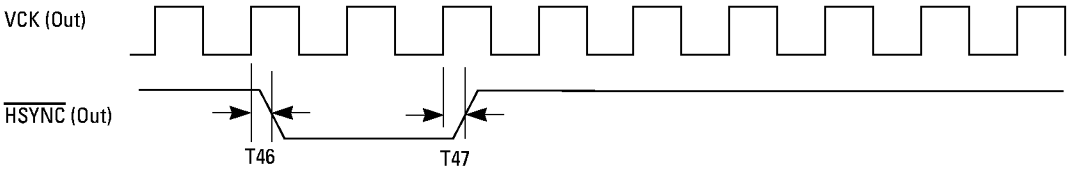


Figure 9-18 VCK Out to $\overline{\text{HSYNC}}$ Out

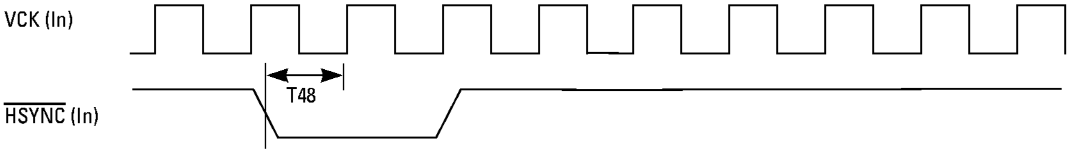


Figure 9-19 $\overline{\text{HSYNC}}$ In to VCK In

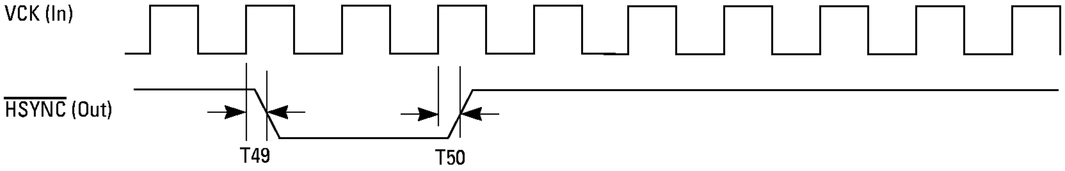


Figure 9-20 VCK In to $\overline{\text{HSYNC}}$ Out

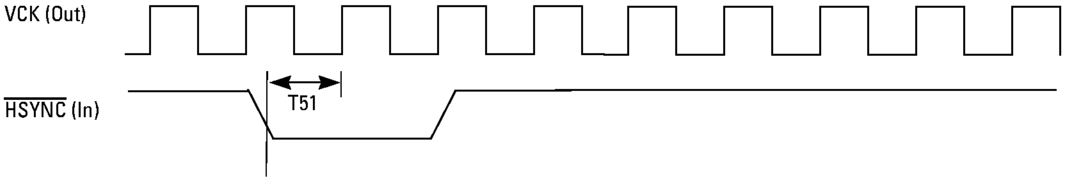


Figure 9-21 $\overline{\text{HSYNC}}$ In to VCK Out

Table 9-13 Timing Characteristics: VCK and HSYNC

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

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

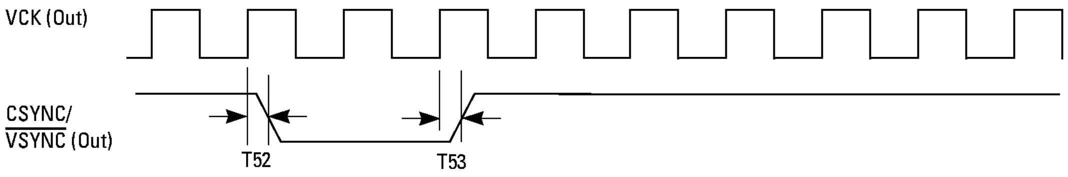


Figure 9-22 VCK Out to CSYNC/VSYNC Out

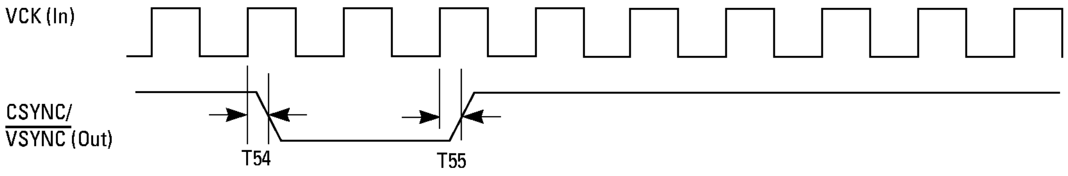


Figure 9-23 VCK In to CSYNC/VSYNC Out

Table 9-14 Timing Characteristics: VCK and CSYNC/VSYNC

Time[1]	Description	Min	Max	Units
T52	VCK (Out) HIGH to CSYNC/VSYNC (Out) LOW	-5	10	ns
T53	VCK (Out) HIGH to CSYNC/VSYNC (Out) HIGH	-5	10	ns
T54	VCK (In) HIGH to CSYNC/VSYNC (Out) LOW	3	30	ns
T55	VCK (In) HIGH to CSYNC/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 CL48x.

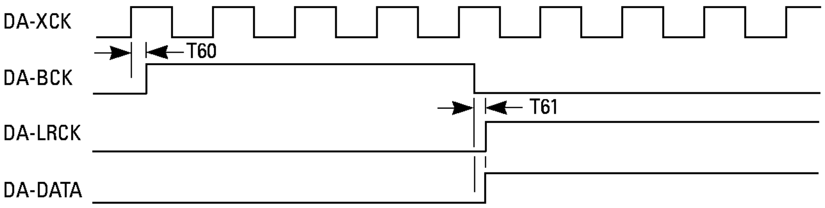


Figure 9-24 Audio Interface Timing Modes

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 CL48x 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 CL48x is packaged in a 128-pin, small outline plastic quad flat pack (PQFP) or a thin quad flat pack (TQFP). This section includes information on the following:

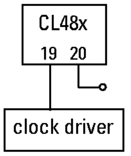
- The CL48x Pinout Diagram
- Tables of CL48x Pin Connections
  - Host Bus Interface Pins
  - CD Interface Pins
  - Global Interface Pins
  - DRAM/ROM Bus Interface Pins
  - Audio Bus Interface Pins
  - Video Bus Interface Pins
  - Power and Miscellaneous Pins
- Package Physical Dimensions

The CL48x 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.

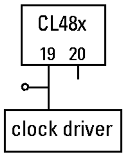
Note: "RESERVED" pins are reserved for future use. They should be pulled either HIGH or LOW.

The 40-MHz (or 40.5-MHz, if VCK is an output of the CL48x) 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).

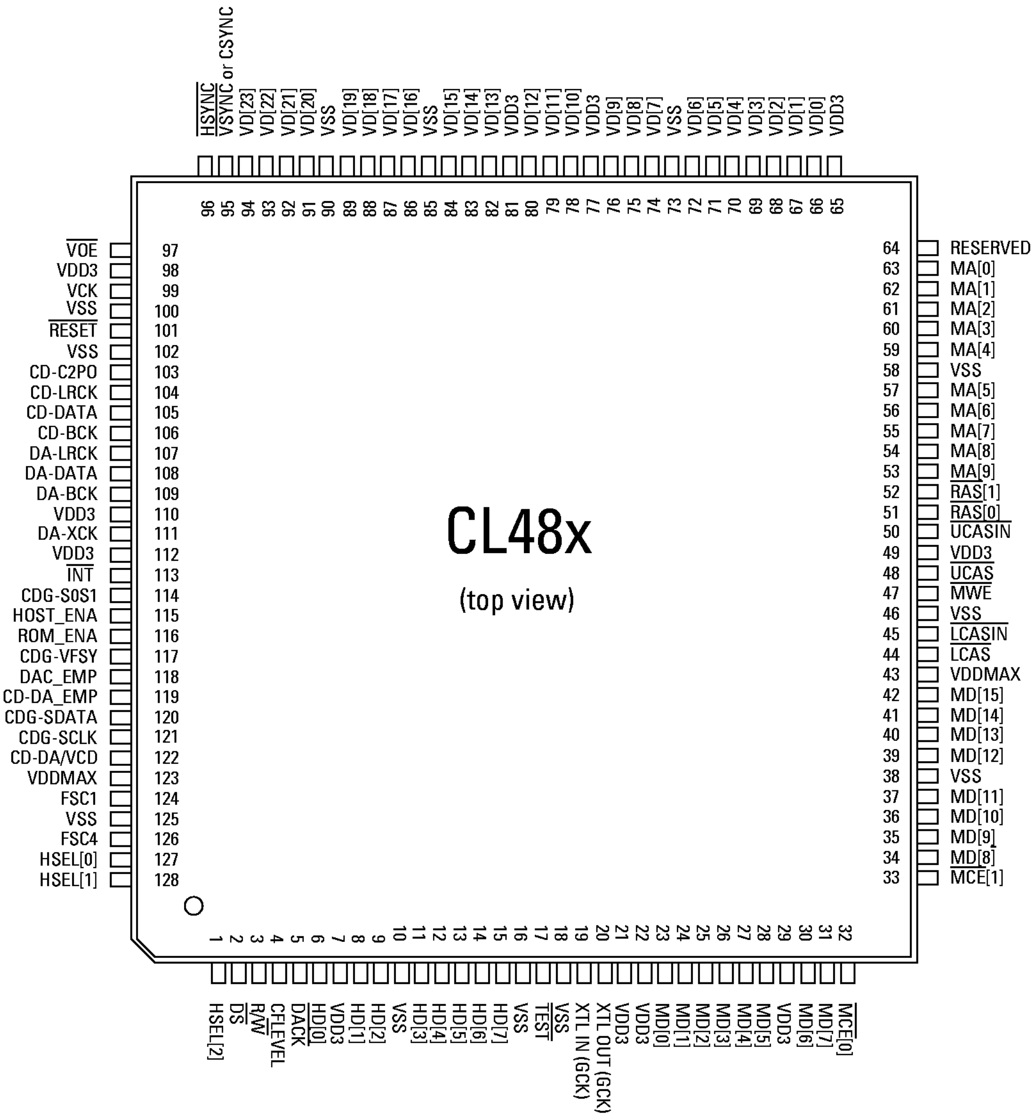
If a clock driver is used, the signal amplitude at pin 19 must be in the range 0.8(V_{DD3}) to V_{DD3} + 0.2V, and pin 20 should not be connected. That is, pin 20 should be connected to a pad on the board, but there should be no extra traces coming from that pad. If a test point is desired, it should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



(unacceptable)



(acceptable)



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.

Note: At 3.3V, DACK and INT should have a minimum pullup resistor value of 439 ohms and a typical pullup resistor value of 500 ohms. At 5.0V, DACK and INT should have a minimum pullup resistor value of 680 ohms and a typical resistor value of 1.0K ohms.

Note: Pins 64, 114, 117, 120, 121, 124, and 126 are programmable I/O pins that are defined by the UNUSED_IO DRAM parameter.

Figure 9-25 CL48x PQFP and TQFP 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
CFLEVEL[1]	Host	O	4	HD[0]	Host	I/O	6
$\overline{\text{DS}}$	Host	I	2	R/$\overline{\text{W}}$	Host	I	3
HD[7]	Host	I/O	15	$\overline{\text{DACK}}$ [1]	Host	O	5
HD[6]	Host	I/O	14	$\overline{\text{INT}}$[1]	Host	O	113
HD[5]	Host	I/O	13	HOST_ENA	Host	I	115
HD[4]	Host	I/O	12				

1. The $\overline{\text{DACK}}$ and $\overline{\text{INT}}$ signals require pullup resistor values which are dependent on the voltage supply: For 3.3V, the $\overline{\text{DACK}}$ and $\overline{\text{INT}}$ pullup resistor should have a minimum value of 439 ohms and a typical value of 500 ohms. For 5.0V, the $\overline{\text{DACK}}$ and $\overline{\text{INT}}$ pullup resistor should have a minimum value of 680 ohms and a typical value of 1.0K ohms.

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
CD-DA/VCD	CD	O	122	CDG-SDATA[1]	CD	I	120
CDG-SOS1[1]	CD	I	114	CDG-SCLK[1]	CD	I/O	121
CDG-VFSY[1]	CD	I	117				

1. These pins may be used as programmable I/O's as defined by the UNUSED_I/O DRAM parameter in Table 15-7.

Table 9-18 Global Interface Pins

Function	Group	I/O	Pin #	Function	Group	I/O	Pin #
XTL IN (GCK)[1]	Global	I	19	XTL OUT (GCK)	Global	O	20
$\overline{\text{RESET}}$	Global	I	101	$\overline{\text{TEST}}$[2,3]	Global	I	17

1. The maximum input voltage for XTL IN is 4.0 V. This signal has a minimum amplitude of 0.8VDD3 and a maximum amplitude of VDD3 + 0.2V.
2. Requires a pullup resistor if you wish to drive the pin LOW with a tester.
3. When pulled low, the $\overline{\text{TEST}}$ pin will 3-state all the CL48x's 3-statable outputs except the DRAM interface's data pins. The DRAM interface's data pins can be 3-stated, however, by conducting a DRAM read transaction immediately before the $\overline{\text{TEST}}$ pin is asserted. The DRAM read cycle causes the DRAM data pins to become input pads, thereby 3-stating the associated output buffers.

Table 9-19 DRAM/ROM Bus Interface Pins

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

[1] This pin may be used as a programmable I/O as defined by the UNUSED_I/O DRAM parameter in Table 15-7.

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	CD-DA_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	$\overline{\text{VOE}}$	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[1]	Color	O	124	FSC4[1]	Color	I	126

[1] These pins may be used as programmable I/O's as defined by the UNUSED_I/O DRAM parameter in Table 15-7.

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	XTL OUT	Global	O	20
XTL IN[2]	Global	I	19				

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

[2] The maximum input voltage for XTL IN 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

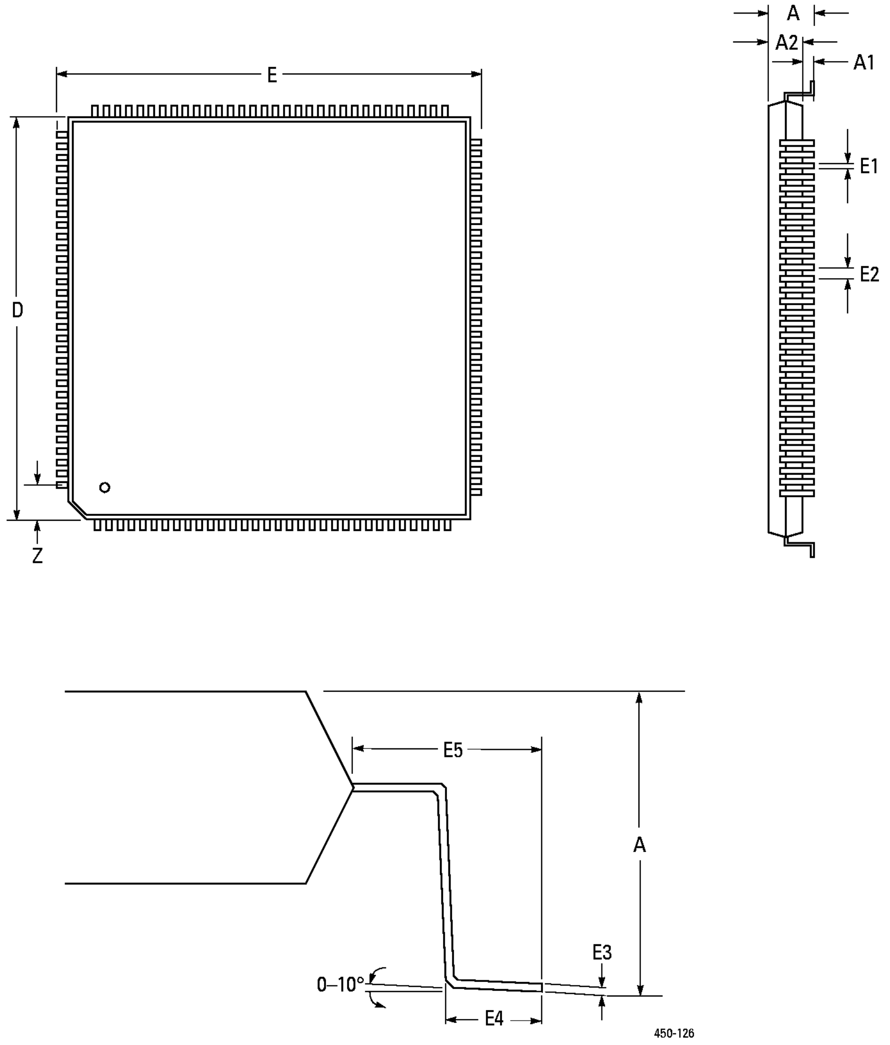


Figure 9-26 PQFP and TQFP Physical Dimensions

Table 9-23 PQFP and TQFP Physical Dimensions

Symbol	PQFP Dimensions mm (inches)	TQFP Dimensions mm (inches)
A	3.7 MAX. (0.145)	1.55 MAX. (0.061)
A1	0 MIN.	0 MIN.
A2	3.5 MAX. (0.138)	1.40 (0.055)
D	18.0 ±0.20 (0.709 ± 0.004)	18.0 ±0.20 (0.709 ± 0.004)
E	20.0 ±0.20 (0.789 ± 0.008)	20.0 ±0.20 (0.789 ± 0.008)
E1	0.2 TYP. (0.008)	0.13 TYP. (0.005)
E2	0.5 TYP. (0.020)	0.5 TYP. (0.020)
E3	0.15 TYP. (0.006)	0.15 TYP. (0.006)
E4	0.50 ±0.20 (0.020 ± 0.004)	0.50 ±0.20 (0.020 ± 0.004)
E5	1.0 ±0.20 (0.040 ± 0.004)	1.0 ±0.20 (0.40 ± 0.004)
Z	1.25 TYP. (0.049)	1.25 TYP. (0.049)

10 Registers

HOST_int is the single CL48x register used to configure and communicate with the CL48x and its microcode, although not all applications will use this register.

This register can be accessed using a single pair of accesses to D_MSB and D_LSB (see Section 4.2.6). To access this register, 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.

In the detailed definition given for the HOST_int register, note that bits marked *Res* are reserved. The reserved bits may be either 1 or 0 when read, and should not be changed.

10.1 Internal Host Register

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

0x0F

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

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

This bit controls the external interrupt signal, $\overline{\text{INT}}$. When *Int* is 1, the CL48x will assert $\overline{\text{INT}}$, driving it LOW. When *Int* is 0, the CL48x does not drive $\overline{\text{INT}}$, which will be pulled HIGH (inactive) by an external pullup resistor.

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

11 Microcode Features

The CL48x performs its higher-level functions by executing microcode on its internal CPU. A summary of the features of CL48x microcode is contained in this chapter.

11.1 Detecting Input Bitstream Type

When the input bitstream is either a CD-DA or CD-ROM bitstream, the CL48x microcode can be enabled to either auto-detect the input bitstream or expect bitstreams of one type only.

11.1.1 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 CL48x microcode then uses the presence or absence of CD-ROM synchronization characters to determine whether the input bitstream is either a CD-ROM or CD-DA bitstream.

During execution of the Play() command and before decoding begins, the CL48x microcode searches for synchronization characters with the proper spacing, or waits until two seconds elapse, whichever comes first. The CL48xVCD microcode continues to examine the incoming bitstream for the presence of CD-ROM sector synchronization characters, and switches between the two bitstream types accordingly.

Auto-detecting CD-DA

The CL48x switches to processing CD-DA data when all of the following conditions are met in the following order:

1. The CL48xVCD microcode is in the PLAY or REPLAY state.
2. The PLAY_MODE configuration parameter is set to CD_ROM_AUTO.
3. CD-ROM sector synchronization characters with the proper spacing have not been detected for two seconds.

When the CL48xVCD 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.

> *Note: CD-G is not auto-detected; instead, the host must issue the CDGTest() command to determine if the incoming bitstream contains CD-G data. However, the host will auto-detect the CD-DA portion of CD-G.*

Auto-detecting CD-ROM

The CL48x microcode switches to processing the CD-ROM data when the following conditions are met in the following order:

1. The CL48xVCD microcode is in the PLAY state.
2. The PLAY_MODE configuration parameter is set to CD_ROM_AUTO.
3. 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.

11.1.2 Setting Detection to One Bitstream Type Only

The CL48xVCD microcode is configured to expect bitstreams of one type only by setting the PLAY_MODE variable to one of two values:

- CD_ROM_MPEG: The incoming bitstream is expected to contain CD-ROM sector information (VideoCD or CD-I).
- CD-DA: The incoming bitstream is expected to contain CD-DA data.

Setting the PLAY_MODE variable to any of the above values disables the auto-detecting algorithm described in Section 11.1.1.

11.2 Bitstream Demultiplexing

For CD-ROM decoding, the CL48x stores the sector header and sub-header information into dedicated DRAM locations called Sector Header and Sector Subheader (See Table 15-6), which are readable by the host. CD bitstream data is then deserialized and written to the CL48x's bitstream buffers.

Once inside the CL48x's bitstream buffers, 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 local DRAM bitstream buffer is called the *rate buffer* in the MPEG standard.)

If the stream identifier within the data packets corresponds with the value of the video stream ID in the VA_STREAM_ID configuration parameter (see Table 15-7), the packet is assumed 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 assumed 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.

> *Note: CD-G data is read from subcode channels as specified in the SUBCODE_CHANNEL DRAM parameter (Chapter 15).*

11.3 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 synchronized 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, the system time clock is updated with SCR values found in subsequent pack headers.

The presentation time-stamps are used by the video decoding process for decoding and displaying each frame. The video presentation time-stamp is compared to the current system time clock value. If the presentation time-stamp differs from the system time clock by more than the allowed tolerance, pictures are skipped or repeated until the STC is within range. Specifically, if the value of the CL48x internal STC is within the following range, the next picture is decoded and displayed:

Presentation time-stamp - INTERVAL < STC < PTS + INTERVAL

If the value of the STC is less than PTS - INTERVAL, pictures are repeated until the value of the STC is within range. If the value of the STC is greater than PTS + INTERVAL, pictures are skipped until the value of the STC is within range.

The INTERVAL parameter is the number of 90-kHz clocks per video frame and is defined in Table 15-1.

11.4 Auto-Pause

The CL48x microcode uses the trigger bit of the CD-ROM/XA submode field (bit 4) to implement the auto-pause feature defined in the CD-I and VideoCD 2.0 specifications. When this bit in the submode field is set, the CL48x microcode will automatically enter the PAUSE state when the following conditions are met:

- The CL48x microcode's processing state is SINGLESTEP, SLOWMOTION, DISPLAYSTILL, PLAY, REPLAY or FREEZE.
- 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 CL48xVCD microcode reads the sector address from the CD-ROM Decoder. The interrupt is generated in all states except IDLE or PAUSE.

The Reset() command is the only way to clear a pending auto-pause condition (defined as the time between when an auto-pause sector is encountered and when the actual pause occurs).

11.5 Fast-Forward and Fast-Reverse

For fast-forward and fast-reverse with VideoCD 2.0, use the Scan() macro command, which searches for the next intra picture and decodes and displays it. Two cases are possible:

- *Scan data is present in the user data portion of the MPEG video bitstream:* In this case, the user may make use of the pointers to the next two future and two previous sectors that contain the beginning on an intra picture. The host can tell the CD-DSP to seek to the appropriate sector and then issue the Scan() command. Note: The CL48x stores the user data in DRAM when it parses the video bitstream.
- *Scan data is not present in the user data portion of the MPEG video bitstream:* In this case, the host must guess where the desired intra picture starts and tell the CD-DSP to seek to that sector before issuing the Scan() command.

11.6 Error Handling

The CL48x has the ability to detect errors in the MPEG bitstream or to react to error start codes inserted in the stream by invoking error concealment microcode, which minimizes the artifacts introduced as a result of the error. Each stream type—video, audio, system layer—uses different error concealment procedures. Error concealment is always turned on, so no setting is required.

11.7 Interrupts

The CL48x 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.

11.8 Bitstream Buffer Underflow or Overflow

Bitstream buffer underflow and overflow are exception events that should not occur in properly operating systems.

11.8.1 Underflow

For video and audio 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 the UND interrupt.

If video data underflow occurs during the decoding of a B-picture, a “tear” may occur on the display. Video data underflow may also cause corruption of the audio output.

If audio data underflow occurs, it will cause audible artifacts. Audio underflow may also stall the video data, leaving an incomplete picture on the display.

If either video or audio 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. Underflows shorter than 33 ms may not cause an underflow interrupt.

11.8.2 Overflow

Bitstream overflow is a condition that is prevented from occurring in the CL48x by the performance of audio/video synchronization from the system SCRs, which are delivered to the CL48x at a fixed rate. By performing synchronization from the SCRs, the CL48x discards pictures if the SCR gets ahead of the video PTSs, thus preventing bitstream overflow.

11.9 Dump Data Feature

Using the DumpData feature, CD-ROM sector information can be read, error-corrected and copied into a user specified buffer. The sector information can be used to determine the disc type, the file structure of the disc, the location of the items within the disc, and to analyze playback control information for VideoCD 2.0.

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 sector 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 that 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 (described in Chapter 14) 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 details). MRC_STATUS should be examined when the DumpData() command completes. If the MRC_STATUS field equals DONE_STATUS, the DumpData() command finished successfully. In all cases, the END-D interrupt, if enabled, 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 the sector read from the CD-ROM (2352 bytes x 2 x N, where N = the numbers of sectors dumped) to hold the raw data and error information. The resultant data size is 2048 times the number of sectors processed.*

11.10 DiscView™

Available only in the CL484, DiscView allows advanced menu-selection capability similar to VideoCD 2.0 while using any of the thousands of VideoCD 1.1 discs available today. It allows you to select different tracks or discs by choosing from among a menu display of up to nine simultaneous images from those tracks or discs (See Figure 11-1):

- *Inter-disc:* Displays one or more images from each disc in a multi-disc player
- *Intra-disc:* Displays images from each of nine tracks on a single disc, allowing quick scanning and selection of disc contents
- *Intra-track:* Displays images from nine scenes within a track—for example, every seven minutes of a movie—for "pseudo-chapter" access

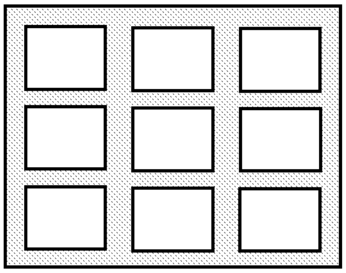


Figure 11-1 DiscView: Up to Nine Non-overlapping Images on One Screen

The DiscView background is a 560x464-pixel high-resolution color display area centered on the active display of the video screen, and the DiscView image or video window overlaid on the background, called Digest, is 176x120 pixels for NTSC and 176x144 pixels for PAL.

For special video effects, a virtually unlimited number of Digest windows may be overlapped on one screen. Digest windows are enabled by a series of four macro commands [variations of DisplayDigest()] as described in Chapter 13.

When the CL484 is decoding and displaying a Digest window, and a new Digest window is created, only the last Digest window to be displayed is capable of continuously updating decoded I-frames showing "slow motion" video effects. The older Digest windows will all be frozen, displaying the last I-frame decoded before the new DisplayDigest() command was invoked. I-frames will be updated in the last Digest window until a specified luma threshold value has been reached, at which time the last Digest window will also freeze.

The flowchart shown in Figure 11-2 describes what actions are taken by the CL484 when DisplayDigest() is executed:

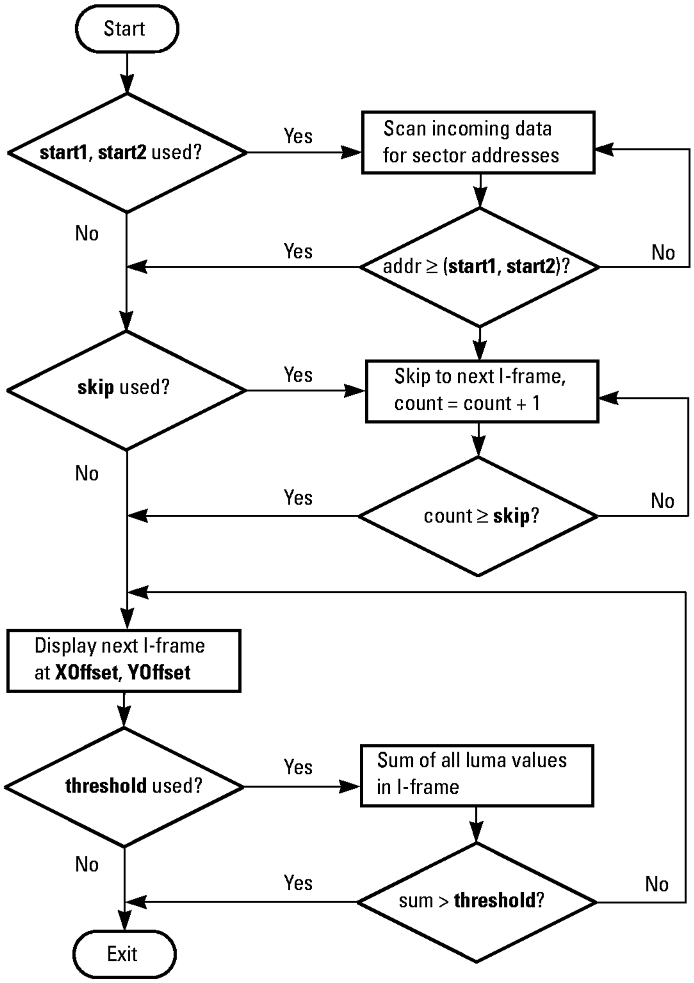

Start
start1, start2 used?
Yes
Scan incoming data for sector addresses
No
Yes
addr ≥ (start1, start2)?
No
skip used?
Yes
Skip to next I-frame, count = count + 1
No
Yes
count ≥ skip?
No
Display next I-frame at XOffset, YOffset
threshold used?
Yes
Sum of all luma values in I-frame
No
Yes
sum > threshold?
No
Exit

Figure 11-2 DisplayDigest Flowchart

11.11 CD-G Feature

Available only in the CL484, this feature provides on-chip support for the CD-G (CD plus graphics) format extension to the Red Book (Compact Disc Digital Audio) standard developed by Philips and Sony; it simultaneously delivers CD audio data and low-bandwidth graphics and lyrics using subcode channels R through W, and is the precursor to VideoCD for low-cost karaoke. CL484CDG allows you to:

- Play CD-DA samples and CD-G discs with video output

- Display OSD (on-screen) graphics
- Operate in master or slave video timing

In the current microcode, the CD-G code is resident (in ROM) at word address 0x5000 and above. The host software needs to manually load the CD-G code into DRAM by using the ModuleLoad() macro command described in Chapter 13. (See page 131 for a listing of other applicable macro commands.)

The CD-G microcode is written such that the configuration area addresses are exactly the same as the VCD microcode, with an offset of 0x5000—that is, the configuration area spans from 0x51C6 to 0x51FF. Therefore, when the configuration area is set, similar changes need to made in the CD-G microcode's configuration area as were made in the VCD microcode. For example, if the only change made to the VCD microcode was to set the VIDEO_MODE DRAM parameter's address 0x01D7 to 0x0B24, then address 0x51D7 of the CD-G microcode also needs to be set to 0x0B24.

SUBCODE_CNTL, at word address 0x1d1 and 0x51D1, is used to configure the CD-G subcode[1] interface.

CD-G is read from the subcode interface from the channels specified with the SUBCODE_CHANNEL configuration parameter. When CD-G data is being decoded, CD-DA data will be read and output, but the AUDIO_MUTE configuration parameter is not supported.

CD-G DRAM parameters are described below and in Chapter 15.

Table 11-1 CD-G DRAM Parameters

Parameter	Description
SUBCODE_CNTL	Used to set up the subcode interface
CD_TYPE	Determines CD data type by providing the results of the previous CDGTest() command
SUBCODE_CHANNEL	Specifies the subcode channels from which CD-G data is read

[1] Subcode data is time, text or graphical information stored together with audio across eight channels on a CD.

Additional CD-G control in the CL484 is provided by a series of four macro commands described below and in Chapter 13.

Table 11-2 CD-G Macro Commands

Macro Commands	Description
CDGSetChannel	Selects the display channel from disc
CDGFreeze	Freezes or blanks the screen
CDGSetBorderColor	Defines background color
CDGTest	Tests bitstream for CD-G data

In addition to the CD-G-specific commands, the following commands are supported by the CD-G module:

- Reset()
- SetVideoFormat()
- Play()
- DisplayGraphics()

11.12 On-screen Display (OSD)

CL48x microcode has the ability to display graphics images as overlays on top of normal video or on a blank screen. This feature is known as on-screen display (OSD). Graphics images are stored in the CL48x local DRAM and are displayed using the DisplayGraphics() command. Two formats are supported: run-length and bitmap. (See Appendix B for a detailed description of bitmap format.)

Bitmap OSD enables the host to easily modify a part of the graphics image. It can support more than one graphics image and up to four colors per graphics image (or an OSD block), whereas run-length encoded OSD supports only one graphics image for the entire screen and only two colors with eight different shades. Moreover, the run-length encoded OSD cannot be partially modified. However, bitmap OSD requires more memory space and also a higher data transfer rate if the user intends to transfer the OSD data from ROM to DRAM. Typical applications for bitmap OSD are for displaying small images such as icons for a user interface, buttons on the remote control, the current time, or the CD title.

11.12.1 Bitmap OSD

Available in the CL484, bitmap OSD uses two bits to represent each pixel within a bitmap. This results in a total of four colors per image. One or more of these colors can be used to specify transparency. The four colors can be specified for all graphics images on the screen by:

- Writing to the predefined DRAM locations UPDATE_GRAPHICS TRANSPARENCY, Y01, U01, V01, Y23, U23, and V23
- Separately specifying color palettes within the data structure for each graphics image (i.e., it is possible to have more than four colors for all graphics images on the entire screen, although each image is limited to four colors). Note that colors specified within the data structure will supersede those defined in DRAM, and that the colors written to DRAM will take effect only after you have set UPDATE_GRAPHICS (0x180) to 1.

  Note: The current microcode only supports OSD in which either all or none of the graphics images have color palettes within the data structure.

Table 11-3 shows some of the important maximum values for building the bitmap OSD. Note: the maximum number of graphics images is only limited by the amount of memory available.

Table 11-3 Bitmap OSD Maximum Values

Resolution	Horizontal	Vertical (NTSC/PAL)
SIF (low)	352	240/288
CCIR 601 (high)	704	480/576

*Note: For CD-G, the OSD cannot be in the active region of the CD-G image—i.e., it can only be displayed in either the top or bottom border. With bit 15 of **startline** clear, the OSD will be displayed in the top border, offset down from the top of the displayed screen by **startline**. With bit 15 set, the **startline** will be offset down from the bottom of the active CD-G screen.*

After the bitmap-encoded file has been created, follow the example procedure for displaying OSD shown below. Please refer to Chapter 13 for more details about the DisplayGraphics() command and its parameters.

1. Load the bitmap data structure into the DRAM location pointed to by the FREE_SPACE_START pointer (word address 0x6c in DRAM).
2. Issue the DisplayGraphics(**startline, stopline, address, fade**) command.

After issuing the command, the CL484 will overlay the graphic stored in DRAM at the selected address onto the screen. To remove the graphic from the screen, issue another DisplayGraphics() command with a **stopline** of 0. It is recommended to wait for two VSYNCs before issuing a new DisplayGraphics() command. This can be done by counting the number of VSYNC-V interrupts.

11.12.2 Run-length OSD

Run-length OSD is available in the CL48x. To generate a run-length format OSD display, you must first generate a bitmap of the desired graphic and then convert it to the CL48x's run-length format, as in the following example:[1]

1. Load the run-length encoded graphic into the CL48x's DRAM at the location stated by the FREE_SPACE_DRAM pointer.
2. Issue the DisplayGraphics() command as follows:
   - Command ID: 0417
   - **startline**: 0
   - **stopline**: 512
   - **address**: 0x2C00 (or FREE_SPACE_START)
   - **fade**: 0

After issuing the DisplayGraphics() command, the CL48x will overlay the graphic stored in DRAM at the selected address onto the screen. To remove the graphic from the screen, issue another DisplayGraphics() command with a **stopline** of 0.

11.13 Key Control

Available with CL484 microcode only, key control allows you to adjust the pitch (frequency spectrum) of the audio output of the karaoke machine to match the key of the karaoke singer. It provides:

- 12 semi-tone pitch control, whereas competing solutions usually provide a range of only 4 to 8 semi-tones
- A 3-button control panel (down, normal, and up)
- Substantial cost reduction over systems without built-in key control

There are 25 different levels of pitch. The difference between adjacent levels is equivalent to a semi-tone in music. Shift in pitch is accom-

[1] Contact a C-Cube representative for more information about the CL48x's run-length format.

plished by use of the PitchShift(**semitones, mode**) macro command described in Chapter 13.

Modes 1 and 2 of the PitchShift() **mode** argument are most useful in karaoke applications where one audio channel is encoded as mono *without* vocal, and the other channel is encoded as mono *with* vocal. The user can then select the mono *without* vocal channel, key-control it, and sing along.

Mode 3 of the PitchShift() **mode** argument can only be used in CD-G and CD-DA modes, in which the user can key-control both audio channels in stereo mode and watch the low-bandwidth graphics while singing along.

11.14 Vertical Scaling

Available with the CL484 only, this feature allows full-screen output of NTSC source material on PAL display monitors. This allows full-screen playback of the thousands of already available NTSC discs on PAL machines.

No action need be taken to invoke this feature. Vertical scaling is automatically effective at bootup [the VIDEO_FORMAT DRAM parameter (See Chapter 15) is set to a value of 3 (PAL_FORMAT)]. The output picture is automatically interpolated so that the entire PAL vertical display is utilized, with no black borders on the top or on the bottom of the screen.

If this feature is not required, you may disable it any time after the system is booted by issuing SetVideoFormat(5, 0, 0).

11.15 Advanced Error Concealment

Available with the CL484 only, this feature allows the best possible playback of discs which are either poorly manufactured or badly scratched.

No action need be taken to invoke this feature except for connecting the CD-C2PO signal.

If the CD-C2PO signal is asserted, the CL484 will keep the previously valid picture on-screen until the next valid picture is decoded.

12 Initialization

The host processor executes an initialization sequence for the CL48x by interacting with the microcode (either macro commands or the Configuration Area). For initialization to occur, the microcode must be loaded from ROM by issuing a pulse on the $\overline{\text{RESET}}$ pin.

12.1 Distributed Files

The C-Cube online bulletin board service (BBS) contains a copy of the CL48x microcode.[1] The CL48x microcode is in "Intel MCS-86 Hexadecimal Object File Format" as found in the "*DATA I/O Unisite Programmer, Code 99*."

The CL48x microcode executables contain default values that determine the protocol used for each of the CL48x's interfaces after they are loaded into the CL48x'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 system requirements. Note that the values of configuration locations that are only read at startup must be modified in the executable before the ROM is programmed.

The minimum size for the CL48x ROM is 128K bytes. All configuration parameters are changeable in ROM before the code is loaded. When the ROM_ENA pin is pulled HIGH and the CL48x is reset, the CL48x hardware loads the first 16 instructions from ROM address zero into the CPU's instruction memory and starts execution at address zero of the instruction memory. The CL48x microcode may copy additional informa-

[1] See the end of this manual for a complete list of C-Cube representatives and distributors from which such products may be obtained.

tion from ROM as necessary for the operation being requested. Any ROM memory locations unused by the CL48x microcode may be used by the host processor, but these locations are not guaranteed to remain unused between microcode versions.

12.1.1 Modifying the Configuration in ROM

The configuration in ROM should be initialized as follows.

1. Load the ROM programmer with the CL48x 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 does not match your configuration, change only the appropriate bits to match your system using a read-modify-write operation. (Each of the important memory locations and their default values are defined in Table 12-1 and, more extensively, in Table 15-7.)
3. Change all of the appropriate locations in the ROM programmer's memory; then burn the ROM.

The Configuration Area is located in the ROM image (executable) at the same locations to which it is copied in DRAM. (Note: For CD-G, the Configuration Area is at the same location as VCD but with an offset of 0x5000.) Table 12-1 lists some of the Configuration Parameters which will commonly be changed, depending on your application.

Table 12-1 Likely Configuration Changes for CL48xVCD Microcode

Parameter Name	Word/byte Addr	Default Value	Description[1]
AUDIO_CONFIG	0x1CB/0x396	0xd13E	This location controls the output data format on the audio interface.
CD_CONFIG	0x1CC/0x398	0x0220	This parameter controls the input data format on the CD interface, including the number of bits per channel; setting LSb or MSb first for the data, setting LSb or MSb first on CD-C2PO, selecting right versus left channel CD-DATA input; and controlling whether the data is latched on the rising or falling CD-BCK edge.
VIDEO_MODE2	0x1CD/0x39A	0x1008	VIDEO_MODE2 controls whether the CL48x outputs $\overline{\text{VSYNC}}$ or CSYNC, 15-bit RGB, and whether the output is interlaced.
VIDEO_FORMAT	0x1EB/0x3D6	0x0004	VIDEO_FORMAT determines PAL versus NTSC output, as well as no-repeat and multi-sync modes.
VIDEO_MODE	0x1D7/0x3AE	0x0B25	VIDEO_MODE selects the VCK direction, controls the video bus width, specifies RGB versus YUV mode, controls horizontal interpolation, and controls the $\overline{\text{HSYNC}}$/$\overline{\text{VSYNC}}$ direction.

[1] See Table 15-7 for actual bit values and locations.

12.1.2 Start-Up Process

The initialization procedure for the CL480VCD and the CL484VCD 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 15-7). Note that only some of the DRAM parameters may be changed after the CL48x is running.
4. 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.

12.2 Memory Usage

The CL48x microcode requires a minimum of four Mbits (or 512K bytes) of DRAM to read, decode, and output VideoCD, CD-G and CD-DA bitstreams.

The DRAM memory map of the CL48x microcode is shown in Figure 12-1. The CL48x can address a maximum of one Mbyte of DRAM and two Mbytes of ROM.

> *Note: Memory below four Mbits that is usable by the host processor is defined by the DRAM parameters FREE_SPACE_START and FREE_SPACE_END, which are defined in Table 15-1.*

The areas from 0x00060 to 0x00200 are collectively called the DRAM Configuration Area, 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 CL48x microcode.

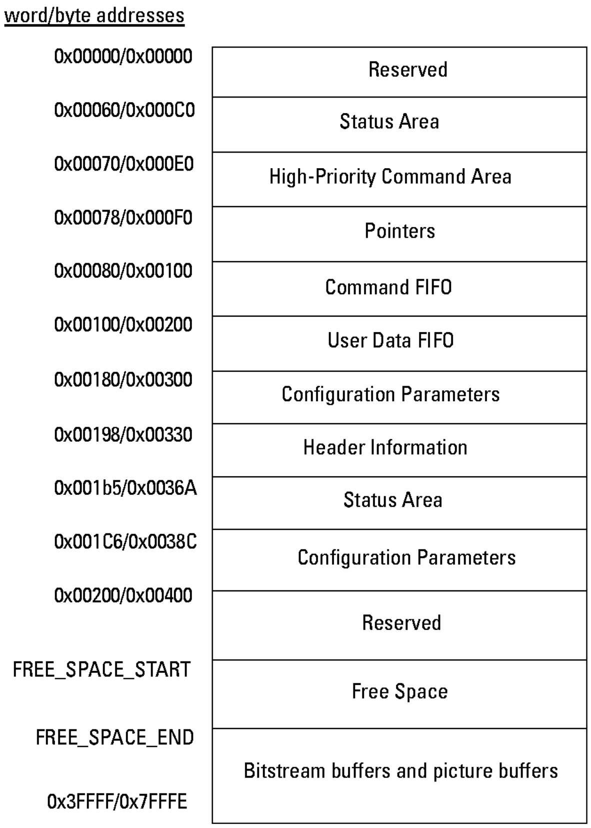


Figure 12-1 DRAM Memory Map

13 Macro Commands

The CL48x host software uses macro commands as its primary method of communication with the CL48x. 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 are checked at a regular interval (every 33 ms) and must complete execution *before* another high-priority command can be issued.
- *Low-priority*: These commands are stored by the CL48x in the Command FIFO and are executed by the CL48x in the order that they were received (more than one command can be written at a time).

13.1 High-Priority Commands

High-priority commands have the priority bit (MSB of the command ID) set in their command ID. 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 CMD_ID location (in local DRAM, Table 15-2) and waits until the first word of this location is zero before giving a macro command to the CL48x.

2. The host writes the arguments for the new macro command into local DRAM.
3. The host writes the new macro command ID into CMD_ID.

The command is complete when the CMD_ID field goes to zero. Also, the END-C interrupt may be used to check the completion of the last high-priority command issued.

13.2 Low-Priority Commands

Most CL48x commands are low-priority. Low priority commands are identified as low priority because of the frequency with which the low priority Command FIFO is checked.

13.2.1 Command FIFO

The CL48x 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 then write back the new value of the write pointer into DRAM. The CL48x 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.

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.

The Command FIFO has the structure shown in Figure 13-1.

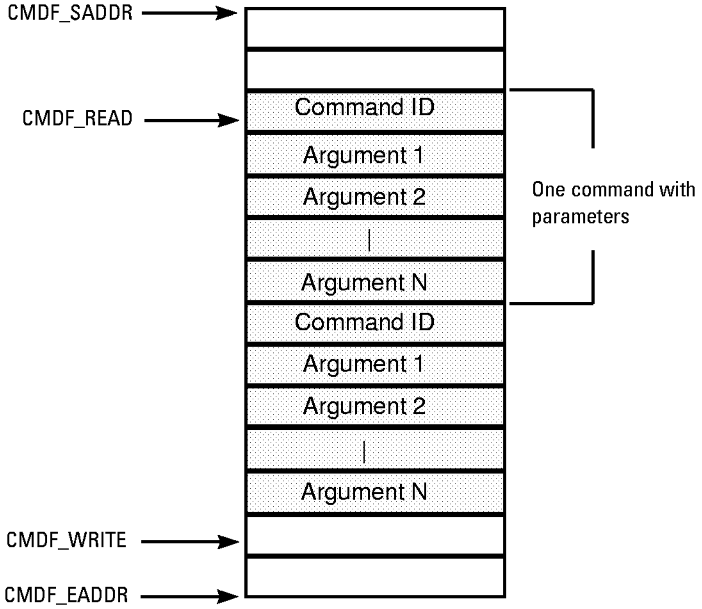


Figure 13-1 Command FIFO Structure

13.2.2 Command Writing Sequence

The sequence for writing low-priority commands is as follows:

1. Read the Command FIFO buffer pointers: CMDF_SADDR, CMDF_EADDR, CMDF_READ, and CMDF_WRITE.
2. Ensure there is enough room in the Command FIFO to store the desired command: Use CMDF_SADDR and CMDF_EADDR to calculate the size of the Command FIFO. Use the CMDF_READ and CMDF_WRITE pointers, along with the Command FIFO size, to determine the amount of free space (considering buffer wrap-around). Note: If the CMDF_READ value equals the CMDF _WRITE value, the Command FIFO is defined to be empty.
3. Write the command ID with arguments into the Command FIFO, incrementing a local write pointer (not CMDF_WRITE) after each write and setting the write pointer to the CMDF_SADDR value when the write pointer equals the CMDF_EADDR value.
4. Repeat steps 2 and 3 for all the commands to send.
5. Write the updated write pointer to CMDF_WRITE.

The command is complete when the CMDF_READ pointer changes to a DRAM address greater than the address at the end of the command.

When the CL48x microcode determines there are commands present in the Command FIFO, the commands are extracted and executed one by one until the Command FIFO is empty.

The frequency at which the Command FIFO gets processed depends on the operation being performed by the CL48x and, thus, cannot be guaranteed to occur within a set period of time.

If the command is Play(), Replay(), SingleStep(), Scan() or SlowMotion(), the CL48x microcode will decode the next MPEG video picture. If there's no video data (audio only), the Command FIFO is not checked until after the first two audio frames. (See Figure 13-2.)

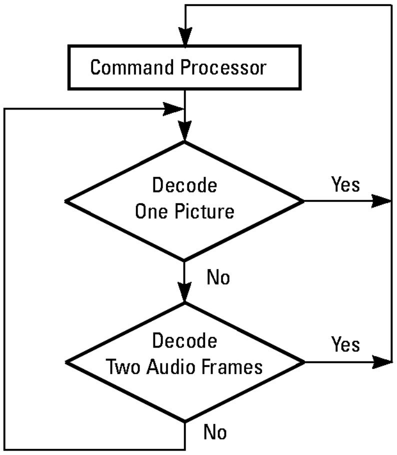


Figure 13-2 Command FIFO Processing

If the input bitstream buffer underflows while decoding, the Command FIFO will not be checked until enough data is input to complete the decoding process. The maximum delay before executing commands if underflow does *not* happen is one picture-decode time.

If 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.

> *Note: When writing macro commands to the Command FIFO, you do not need to write a fixed number of bytes for each command; write exactly as many 16-bit words as the command needs.*

13.3 Status Area

A Status Area is maintained in DRAM by the CL48x, which gives the status of the currently executing, or just completed, command and other information. The CL48x 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 relate to command execution:

- *PROC_STATE (processing state):* Contains the new processing state upon command execution.
- *MRC_ID (most recent command identifier):* Contains the identifier of the last command executed. MRC_ID is changed when the CL48x executes 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.

The Status Area remains valid only in the time between the end of one command execution and the beginning of the next.

13.4 Processing States

The CL48x operates in a multiple number of internal processing states. Processing states determine how future commands are interpreted and how the decoding and displaying processes operate. Some commands cause a processing state transition. In addition to macro commands, several internal operations can also cause a state transition.

The processing states and their relationships are shown in Figures 13-3, 13-4, and 13-5. Within these three figures, processing states are inscribed within ovals. The shaded arrows indicate state transitions which occur due to internal processing, while solid arrows indicate transitions caused by the execution of a host-supplied macro command.

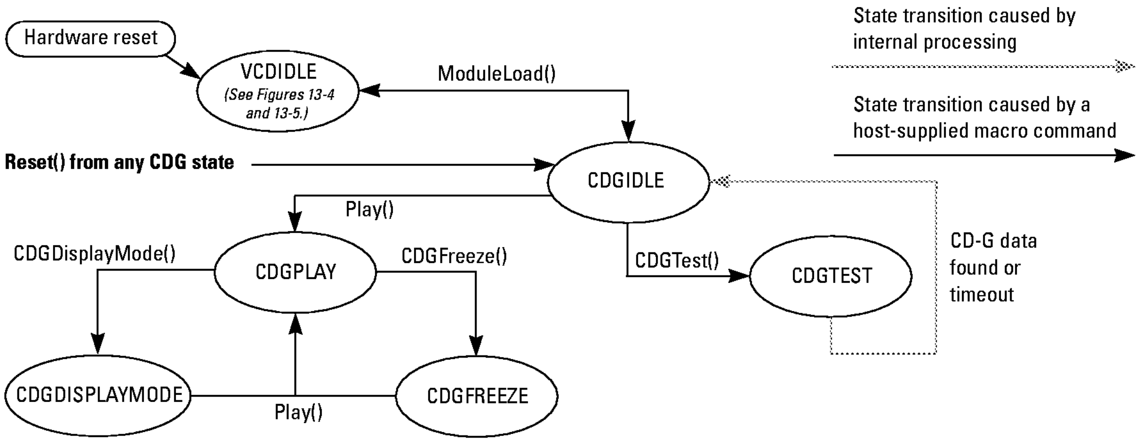

Hardware reset
VCDIDLE
(See Figures 13-4 and 13-5.)
ModuleLoad()
State transition caused by internal processing
State transition caused by a host-supplied macro command
Reset() from any CDG state
CDGIDLE
Play()
CDGDisplayMode()
CDGPLAY
CDGFreeze()
CDGTest()
CDGTEST
CD-G data found or timeout
CDGDISPLAYMODE
Play()
CDGFREEZE

Figure 13-3 CL484 CDG Processing States

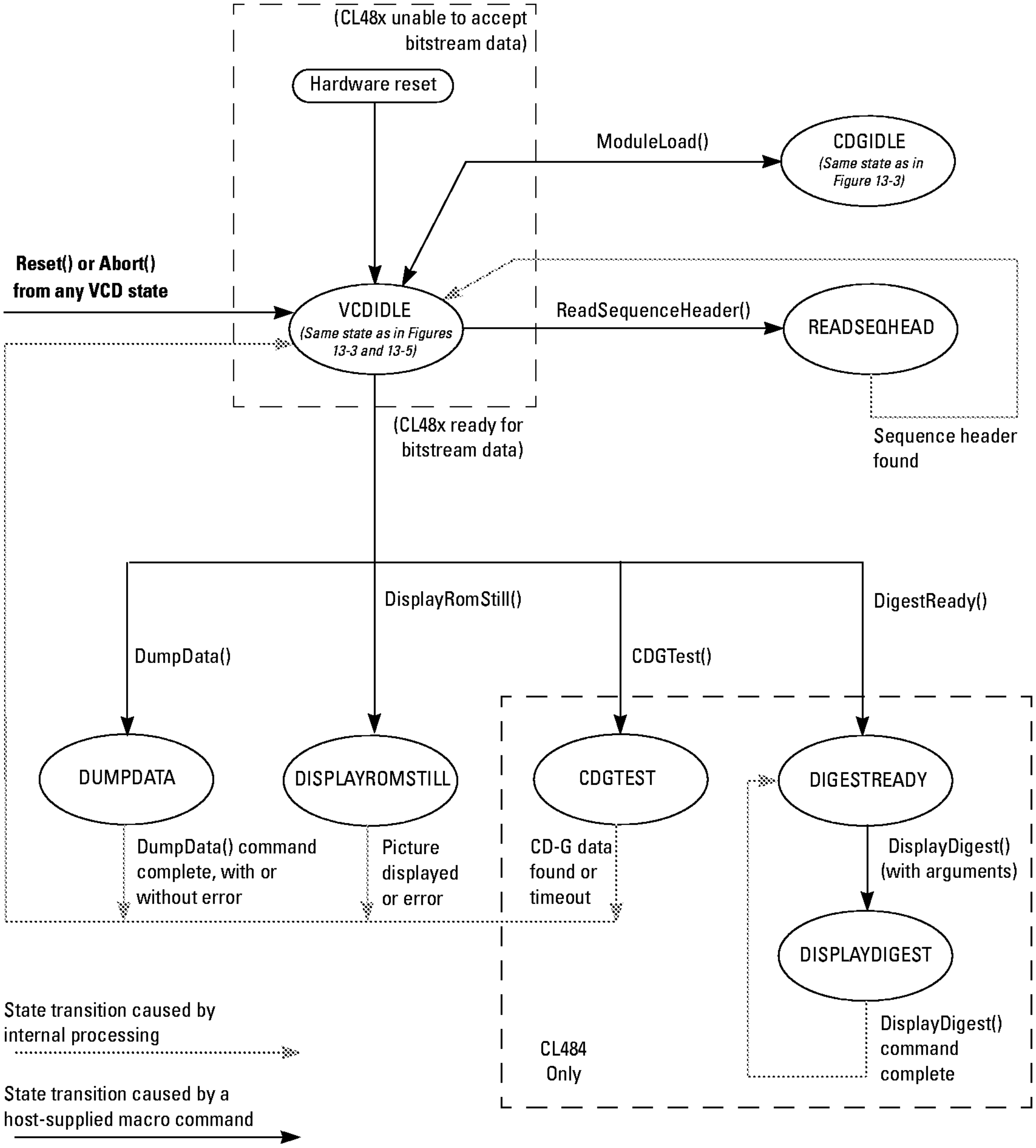

(CL48x unable to accept bitstream data)
Hardware reset
ModuleLoad()
CDGIDLE
(Same state as in Figure 13-3)
Reset() or Abort() from any VCD state
VCDIDLE
(Same state as in Figures 13-3 and 13-5)
ReadSequenceHeader()
READSEQHEAD
Sequence header found
(CL48x ready for bitstream data)
DisplayRomStill()
DigestReady()
DumpData()
CDGTest()
DUMPDATA
DISPLAYROMSTILL
CDGTEST
DIGESTREADY
DumpData() command complete, with or without error
Picture displayed or error
CD-G data found or timeout
DisplayDigest() (with arguments)
DISPLAYDIGEST
DisplayDigest() command complete
CL484 Only
State transition caused by internal processing
State transition caused by a host-supplied macro command

Figure 13-4 CL48x VCD Processing State Transition Diagram: DumpData(), DisplayRomStill(), CDGTest(), DisplayDigest(), and ReadSequenceHeader()

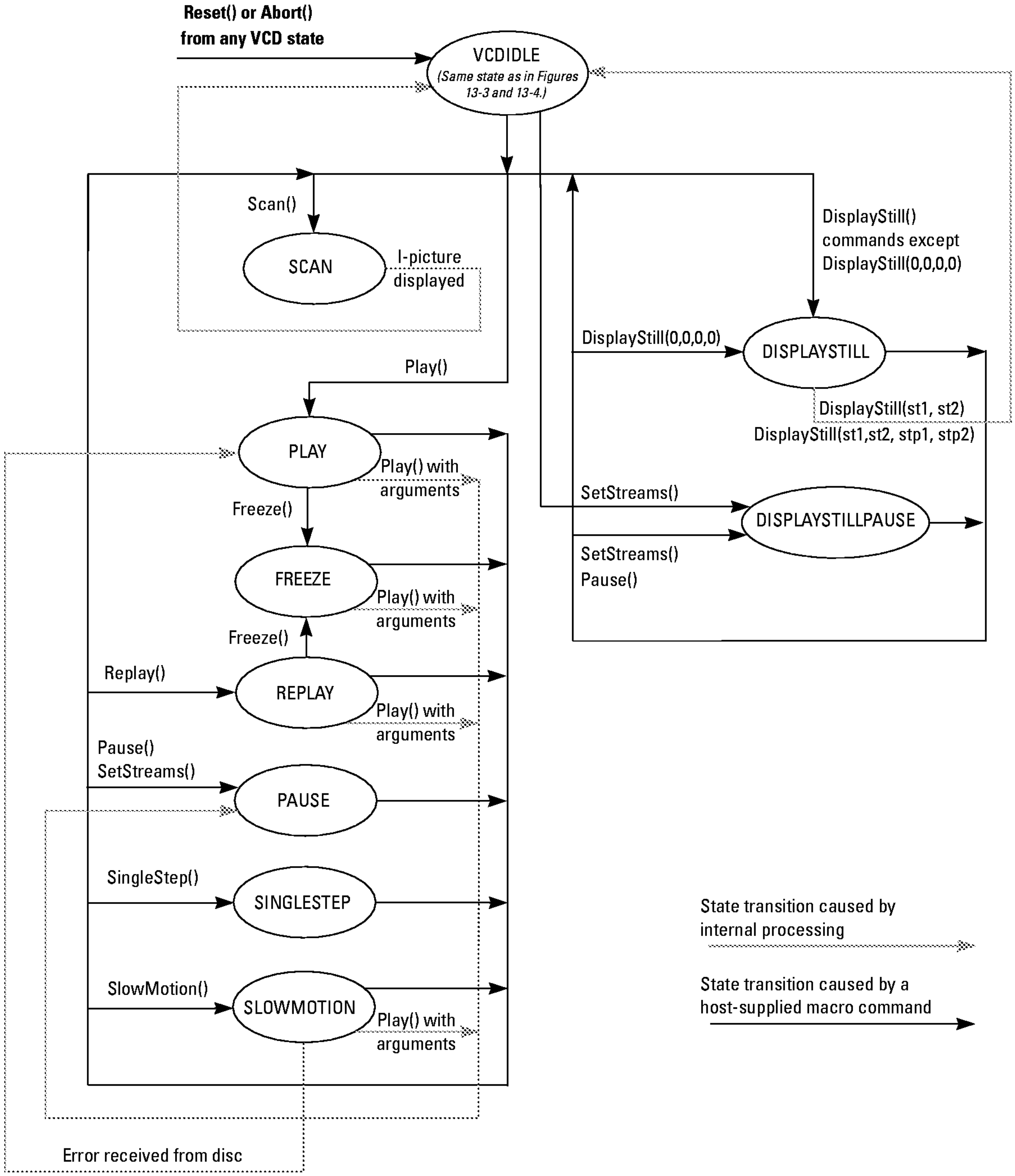


Figure 13-5 CL48x VCD Processing State Transition Diagram: Scan(), Play(), Freeze(), Replay(), Pause(), SetStreams(), SingleStep(), SlowMotion(), and DisplayStill()

13.5 Macro Command Groups

CL48x macro commands are divided into two functional categories: those that change the processing state upon completion, and those that do not.

Each command group has distinct properties, which are described below and in Table 13-1.

13.5.1 Non State Transition Commands

The CL48x has seven macro commands—DisplayGraphics(), SetVideoFormat(), SetVideoFormat(**format,0,0**), CDGSetBorderColor(), InquireBufferEmptiness(), CDGSetChannel() and PitchShift()—which never change the processing state. The first five of these commands have no effect on the decoding process, while the last two—CDGSetChannel() and PitchShift()—change CD-G data channels and audio pitch, respectively.

13.5.2 State Transition Commands

The CL48x has 27 state-transition commands, shown in Table 13-1, each of which can cause the current processing state to change. These commands—except the CDGTest(), Pause(), DumpData(), DigestReady(), and SetStreams() commands—cause the CL48x to read, decode, and output data from the serial interface.

The DumpData() command is different from other state-transition 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.

The Pause(), CDGTest() and SetStreams() commands suspend bitstream decoding and outputting.

> *Note: If a macro command is issued and a Command ID contains a value other than those listed in Table 13-1, indeterminate behavior occurs.*

Table 13-1 Macro Command Summary

Category	Priority	Name	Description	Cmd. ID	Page
Non State Transition Commands	Low	CDGSetBorderColor()[1]	Defines background (border) color	0x0207	135
	Low	CDGSetChannel()[1]	Selects the display channel from disc	0x010B	136
	Low	DisplayGraphics(stl, stpl, addr, fade)	Decodes and displays on-screen graphic overlay	0x0417	145
	Low	PitchShift(semitones, mode)[1, 2]	Controls pitch of the audio stream	0x0222	163
	Low	SetVideoFormat()	Restores format to the default value	0x0005	174
	Low	SetVideoFormat(format, 0, 0)	Sets format to NTSC, PAL or progressive	0x0305	174
	High	InquireBufferEmptiness()	Measures data in bitstream buffer	0x8003	160
State Transition Commands	Low	CDGDisplayMode(display)[1]	Specifies screen to be CD-G data or blank color	0x0109	133
	Low	CDGFreeze()[1]	Freezes or blanks the screen	0x010A	134
	Low	CDGTest()[1, 2]	Tests bitstream for CD-G data	0x011E	137
	Low	DigestReady()[2]	Sets border color to prepare for DisplayDigest()	0x001D	138
	Low	DisplayDigest(X, Y)[2]	Decodes a single I-picture at specific location	0x021B	139
	Low	DisplayDigest(X, Y, skip)[2]	Same as above but with skip argument	0x031B	140
	Low	DisplayDigest(X, Y, skip, thr)[2]	Same as above but with threshold argument	0x041B	141
	Low	DisplayDigest(X, Y, skip, thr, st1, st2)[2]	Same as above but specifies starting point	0x061B	143
	Low	DisplayRomStill(rA1, rA2, length)	Same as DisplayStill() but specifies ROM address	0x030C	148
	Low	DisplayStill()	Decodes/displays single still picture w/ audio	0x000C	150
	Low	DisplayStill(0, 0, 0, 0)	Restarts previous DisplayStill() command	0x040C	152
	Low	DisplayStill(start1, start2)	Same as DisplayStill() but specifies starting point	0x020C	153
	Low	DisplayStill(st1, st2, stp1, stp2)	Same as DisplayStill() but specifies stop point	0x040C	155
	Low	DumpData()	Reads and performs ECC on CD-ROM data	0x0414	157
	Low	Freeze()	Freezes display but continues decoding	0x0010	159
	Low	ModuleLoad(module)[1, 2]	Loads and runs new module	0x011F	161
	Low	Pause()	Freezes display and decoding process	0x000E	162
	Low	Play()	Decodes and displays at normal rate	0x000D	164
	Low	Play(st1, st2, stp1, stp2)	Same as above but specifies start/stop	0x040D	166
	Low	ReadSequenceHeader()	Decodes MPEG bitstream until sequence header	0x0023	168
	Low	Replay()	Restarts the previous Play() command	0x001C	170
	Low	Scan()	Decodes and displays next single I-picture	0x000A	172
	Low	SetStreams()	Selects video and audio stream identifiers	0x0213	173
	Low	SingleStep()	Decodes and stores next single picture	0x000B	176
	Low	SlowMotion()	Decodes and displays at slower rate	0x0109	177
	High	Abort()	Sets processing state to IDLE	0x801A	132
	High	Reset()	Re-initializes CL48x and its microcode	0x8000	171

[1] These commands are only available with the CL484 microcode when the CD-G module is loaded.

[2] These commands are only available with the CL484 microcode when the VCD module is loaded.

Note: All commands not referenced in footnotes 1 and 2 are available with both the CL480 and CL484 microcode when the VCD module is loaded. The Play( ), DisplayGraphics( ), and Reset( ) commands are available at all times.

13.6 Macro Command Reference

CL48x macro commands are listed alphabetically in the following pages. All arguments to all macro commands are 16-bit hexadecimal values (15=MSb, 0=LSb) with big-endian byte ordering, unless they contain CD-ROM sector addresses (Frame, Mode, Minutes, Seconds), in which case they are binary-coded decimal (BCD) notation. Note the following sector divisions: Mode = 2, Frame = 0-74, Minutes = 0-59, and Seconds = 0-59.

Abort()

Format:	Abort()
Priority:	High
Module Used:	CL48xVCD
Category:	State Transition Command
Command ID:	0x801A

This high-priority command (active only in VCD, not CDG) causes any currently-executing command to terminate. The CL48x microcode then examines the contents of the Command FIFO and executes the next command in the FIFO.

This command can be executed from any VCD state. If there are no commands in the FIFO, the CL48x will wait in the IDLE state. The screen is not blanked by this command.

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

CDGDisplayMode(display)

Format:	CDGDisplayMode(**display**)
Priority:	Low
Module Used:	CL484CDG
Category:	State Transition Command
Command ID:	0x0109

This low-priority command is only available in the CL484 microcode when the CD-G module is loaded. This command can only be executed in the CDGPLAY state, and transitions to the CDGDISPLAYMODE state.

The CDGDisplayMode() command specifies whether the screen should be blanked (display the border color) or should display CD-G graphics. Whether the screen is blanked or not, CD-DA data continues to play.

The screen can be *un*blanked by issuing CDGDisplayMode(0) or by issuing the Play() command.

The operation of this command is different from the CDGFreeze() command because, with CDGDisplayMode(), you can update the default color of the screen using CDGSetBorderColor() while the screen is blanked; with CDGFreeze(), you cannot.

CDGDisplayMode() arguments are defined as follows:

```
unsigned int display;        /* display specifies whether the display screen*/
                             /* should be set to the border color or should*/
                             /* display CD-G data.*/
                             /* 0 = display CD-G data*/
                             /* 1 = display border color*/
```

CDGFreeze()

Format:	CDGFreeze(**FreezeMode**)
Priority:	Low
Module Used:	CL484CDG
Category:	State Transition Command
Command ID:	0x010A

This low-priority command is only available in the CL484 microcode when the CD-G module is loaded. This command can only be executed in the CDGPLAY state.

The CDGFreeze() command enables or disables decoding of CD-G data according to the value of the command argument provided.

In all modes, CD-DA data continues to play.

The screen can be *un*frozen by issuing CDGFreeze(0) or by issuing the Play() command.

CDGFreeze() arguments are defined as follows:

```
unsigned int FreezeMode;    /* FreezeMode determines how the CD-G data*/
                            /* is suspended.*/
                            /* 0 = No freeze: CD-G data is decoded and*/
                            /* displayed.*/
                            /* 1 = Decoding of CD-G data is suspended and*/
                            /* the last video display is unchanged.*/
                            /* 3 = Decoding of CD-G data is suspended and*/
                            /* display screen is frozen to the border color.*/
                            /* (Issuing a CDGSetBorderColor(Cr,YCb) will not */
                            /* modify the screen color.)*/
```

CDGSetBorderColor()

Format:	CDGSetBorderColor(**Cr, YCb**)
Priority:	Low
Module Used:	CL484CDG
Category:	Non State Transition Command
Command ID:	0x0207

This low-priority command is only available in the CL484 microcode when the CD-G module is loaded. The CDGSetBorderColor() command sets the Cr, Y, and Cb values of the default screen color used by the CDGDisplayMode(), CDGFreeze(), and Reset() commands. This command can be executed at any time (if the CD-G module is installed).

CDGSetBorderColor() arguments are defined as follows

```
unsigned int    Cr, YCb;                          /* The Cr, Y and Cb values
                                                  for the new screen color.*/
```

Cr bits 7:0 are the Cr value of the required border color, **YCb** bits 15:8 are the Y value, and bits 7:0 are the Cb value.

To reset the default color to black, issue CDGSetBorderColor(0x80, 0x1080). The range of values for RGB is 16 to 235.

CDGSetChannel()

Format:	CDGSetChannel(**Channel**)
Priority:	Low
Module Used:	CL484CDG
Category:	Non State Transition Command
Command ID	0x010B

This low-priority command is only available in the CL484 microcode when the CD-G module is loaded, and can be executed at any time.

The CDGSetChannel() command is used to specify the channels from which to read and display CD-G data (multiple channels may be specified simultaneously).

The default setting of 3 implies channel numbers 0 and 1 will be displayed.

CDGSetChannel() arguments are defined as follows:

```
unsigned int Channel;        /* Bitmap indicating the channels from which to read
                             /* CD-G data. Bit 15 corresponds with channel 15,*/
                             /* bit 0 corresponds to channel 0 and the other bits*/
                             /* correspond to channels 1 through 14, respectively.*
                             /* A one enables data being read from the channel,*/
                             /* while a zero disables reading.*/
```

Note: This command's function is duplicated in the SUBCODE_CHANNEL DRAM parameter, which should be used, when possible, instead of CDGSetChannel( ).

CDGTest()

Format:	CDGTest(**NumberOfSyncs**)
Priority:	Low
Module Used:	CL484CDG or CL484VCD
Category:	State Transition Command
Command ID	0x011E

This low-priority command is available in both the CL484VCD and CL484CDG modules, and must be executed from the VCDIDLE or CDGIDLE states.

This command determines if the incoming bitstream contains CD-G data; it tests for subcode data, not subcode channels. If no CD-G data is received within the number of sync pulses specified by the command argument, the command will indicate that the disc contains no CD-G data.

The minimum number of subcode sync pulses received is five. The recommended maximum number is 75 (one second). If CD-G data is received before the number of sync pulses specified, the command indicates that the disc contains CD-G data by terminating immediately and transferring from the CDGTEST to the CDGIDLE state.

The CD_TYPE status location in DRAM contains the result of the CDGTest() command when the processing state transitions from the CDGTEST state to the IDLE state and the MRC_STATUS status variable indicates the command is done. The CD_TYPE parameter values are 0 = undetermined disc type, 0x06 = section of disc played contains VCD or CD-DA data, and 0x10 = section of disc played contains subcode data for CD-G. If the MRC_STATUS status variable indicates the command terminated with an error, invalid data was detected on the subcode interface.

> *Note: Because some CD-G discs do not always output CD-G data, it is recommended to issue this command at the start of the disc.*

CDGTest() arguments are defined as follows:

```
unsigned int   NumberOfSyncs  /* Maximum number of sync*/
                              /* pulses during which the CL484*/
                              /* tests for CD-G data.*/
                              /* Recommended value is between 5*/
                              /* and 75.*/
```

DigestReady()

Format:	DigestReady()
Priority:	Low
Module Used:	CL484VCD
Category:	State Transition Command
Command ID	0x001D

This low-priority command causes a state transition.

This low-priority command is only available in the CL484 microcode version when the VCD module is loaded; it is not available in the CD-G microcode. This command can only be executed from the VCDIDLE state or the DIGESTREADY state.

The DigestReady() command causes a 560 x 464 pixel area of the display to be set to the current border color (set by the COLOR_CR and COLOR_Y_CB configuration parameters) and to be centered both horizontally and vertically in the active display area of the video screen. This 560 x 464 display area is centered on the display screen by the CL484 microcode automatically. The new border color values will take effect every time the DigestReady() macro command is executed.

The CL484 transitions to the DIGESTREADY state upon completion of this command. The Abort() or Reset() command must be used to transition from the DIGESTREADY state to VCDIDLE.

> *Note: The DigestReady() command must be executed before any DisplayDigest() commands are executed, since DigestReady() causes the CL484 to transition to the DIGESTREADY state.*
>
> *Also, the upper left-hand corner of the 560 x 464 high-resolution display area is (0,0). For the DisplayDigest() series of commands,* ***XOffset*** *must be an even value, and* ***YOffset*** *must be a value which is a multiple of eight.*

DisplayDigest(X, Y)

Format:	DisplayDigest(**XOffset, YOffset**)
Priority:	Low
Module Used:	CL484VCD
Category:	State Transition Command
Command ID	0x021B

This low-priority command is only available in the CL484 microcode version when the VCD module is loaded, and can only be executed from the DIGESTREADY state.

The DisplayDigest() command causes the CL484 to decode, decimate (reduce) and display the next I-picture at the specified location (**XOffset** and **YOffset**) from the upper left-hand corner of the high-resolution display area. The picture is decimated by two both vertically and horizontally. The minimum value for **XOffset** and **YOffset** is 0. The maximum **XOffset** is 384; the maximum **YOffset** value is 344 (NTSC), 320 (PAL).

DisplayDigest() causes the CL484 to transition to the DISPLAYDIGEST state while the I-picture is being decoded and decimated. When complete, the CL484 will transition to the DIGESTREADY state. The following restrictions apply to this command:

- Its **XOffset** value will be truncated to an even value, and its **YOffset** value will be truncated to a value divisible by eight.
- It must be preceded by the DigestReady() command (causing the CL484 to transition to the DIGESTREADY state) or another DisplayDigest() command.
- It does not allow displayed pictures to extend beyond the edge of the colored display area.
- It ignores the XOFFSET, YOFFSET, WIDTH, and HEIGHT configuration parameters.

DisplayDigest(**XOffset**, **YOffset**) arguments are defined as follows:

```
unsigned int XOffset;       /* X offset to display picture (units: CCIR 601 pixels)*/
unsigned int YOffset;       /* Y offset to display picture (units: CCIR 601 lines)*/
```

Note: When the CL484 is in the DIGESTREADY state, only three types of commands are valid: DisplayDigest(), DisplayGraphics(), and Reset().

DisplayDigest(X, Y, skip)

Format:	DisplayDigest(**XOffset, YOffset, skip**)
Priority:	Low
Module Used:	CL484VCD
Category:	State Transition Command
Command ID	0x031B

This low priority command is only available in the CL484 microcode version when the VCD module is loaded. This command can only be executed from the DIGESTREADY state, to which it transitions upon completion.

The DisplayDigest() command with three arguments operates the same as the DisplayDigest() command with two arguments except for the addition of the **skip** argument. The **skip** argument specifies the number of I-pictures to skip before decoding, decimating, and displaying the next I-picture. **Skip** is an integer number, the minimum and maximum values of which are 0 and 65535, respectively. If attempting to display a title frame from a VideoCD track, a good value to use for **skip** is 5 (i.e., the 5th frame in a track frequently contains title information).

The following restrictions apply to this command:

- Its **XOffset** value will be truncated to an even value, and its **YOffset** value will be truncated to a value divisible by eight.
- It must be preceded by the DigestReady() command, causing the CL484 to transition to the DIGESTREADY state.
- It does not allow displayed pictures to extend beyond the edge of the display area.
- It ignores the XOFFSET, YOFFSET, WIDTH, and HEIGHT configuration parameters.

DisplayDigest() arguments are defined as follows:

```
unsigned int XOffset;          /* X offset to display picture (units: CCIR 601 pixels)*/
unsigned int YOffset;          /* Y offset to display picture (units: CCIR 601 lines)*/
unsigned int skip;             /* Number of I-pictures to skip before decoding*/
```

Note: The skip counter is decremented before the picture is decoded; therefore, all I-pictures encountered in the input bitstream, including those in error, are used to decrement the skip counter.

DisplayDigest(X, Y, skip, threshold)

Format:	DisplayDigest(**XOffset, YOffset, skip, threshold**)
Priority:	Low
Module Used:	CL484VCD
Category:	State Transition Command
Command ID	0x041B

This low-priority command is only available in the CL484 microcode version when the VCD module is loaded, and can only be executed from the DIGESTREADY state.

The DisplayDigest() with four arguments operates the same as the DisplayDigest() command with three arguments except for the addition of the **threshold** argument. The **threshold** argument defines an upper limit for the luma content of an I-picture, above which no more new pictures are generated. This feature prevents displaying a black frame in menu mode, and can be used to continually display all I-frames in the incoming bitstream (with audio).

When the **threshold** value is used, the CL484 adds all the luminance values in each I-frame. If the cumulative sum of the luminance values falls below the threshold, the CL484 will continue updating the display window with the next I-frame in the bitstream until an I-picture is found with the **threshold** value exceeded.

When the luma content of a displayed I-picture is above the value of the **threshold** argument (the likely case if **threshold** is *not* used or is a very low number), the display window will freeze after displaying the first I-frame, and the CL484 will complete execution of DisplayDigest() by entering the DIGESTREADY state. If **threshold** is set to a very large number (near the maximum), then the threshold will never be reached, and I-pictures will display indefinitely within the display window.

The minimum value for **threshold** is 0. The maximum value for the luma content is 0x1300 for PAL and 0x1000 for NTSC pictures. Setting **threshold** to around 0x500 prevents the CL484 from freezing the Digest display window on a blank or dark image. To continually update the display window with incoming I-frames, set **threshold** above the maximum luma content for a PAL or NTSC picture.

DisplayDigest(X, Y, skip, threshold)

The following restrictions apply to this command:

- Its **XOffset** value will be truncated to an even value, and its **YOffset** value will be truncated to a value divisible by eight.
- It must be preceded by the DigestReady() command, causing the CL484 to transition to the DIGESTREADY state.
- It does not allow displayed pictures to extend beyond the edge of the display area.
- It ignores the XOFFSET, YOFFSET, WIDTH, and HEIGHT configuration parameters.

DisplayDigest(**XOffset**, **YOffset, skip, threshold**) arguments are defined as follows:

```
unsigned int XOffset;        /* X offset to display picture (units: CCIR 601 pixels)*/
unsigned int YOffset;        /* Y offset to display picture (units: CCIR 601 lines)*/
unsigned int skip;           /* Number of I-pictures to skip before decoding*/
unsigned int threshold;      /* Luma content of last I-picture to display*/
```

DisplayDigest(X, Y, skip, threshold, st1, st2)

Format:	DisplayDigest(**XOffset, YOffset, skip, threshold, start1, start2**)
Priority:	Low
Module Used:	CL484VCD
Category:	State Transition Command
Command ID	0x061B

This low-priority command is only available in the CL484 microcode when the VCD module is loaded. It can only be executed from the DIGESTREADY state, which it will return to when complete.

The DisplayDigest() command with six arguments operates the same as the DisplayDigest() command with four arguments except for the addition of the **start1** and **start2** arguments. The **start1** and **start2** arguments specify the place on the disk from which to start reading data.

The minimum value for **threshold** is 0. The maximum values for the luma content is 0x1300 for PAL pictures and 0x1000 for NTSC pictures. Setting **threshold** to around 0x500 prevents the CL484 from freezing the Digest display window on a blank or dark image.

The following restrictions apply to this command:

- Its **XOffset** value will be truncated to an even value, and its **YOffset** value will be truncated to a value divisible by eight.
- It must be preceded by the DigestReady() command, causing the CL484 to transition to the DIGESTREADY state.
- It does not allow displayed pictures to extend beyond the edge of the display area.
- It ignores the XOFFSET, YOFFSET, WIDTH and HEIGHT configuration parameters.

DisplayDigest(**XOffset**, **YOffset, skip, threshold, start1, start2**) arguments are defined as follows:

```
unsigned int XOffset;          /* X offset to display picture (units: CCIR 601 pixels)*/
unsigned int YOffset;          /* Y offset to display picture (units: CCIR 601 lines)*/
unsigned int skip;             /* Number of I-pictures to skip before decoding*/
unsigned int threshold;        /* Luma content of last I-picture to display*/
unsigned int start1, start2;   /* Disc sector address (BCD values stored in two words) to */
                               /* start decoding: The MSB of start1 specifies the Minutes, and */
                               /* the LSB of start1 specifies the Seconds. The MSB of start2 */
                               /* specifies the Frame, and the LSB of start2 specifies the Mode */
                               /* (Mode must be 2).*/
```

DisplayDigest(X, Y, skip, threshold, st1, st2)

The order of execution for the arguments is as follows:

1. The CL484 scans incoming sector addresses and waits for a sector address greater than or equal to **start1** and **start2**.
2. After the start address is reached, I-pictures are skipped according to the **skip** argument
3. Each successive I-frame is displayed at **XOffset** and **YOffset** on the display window, and the luma content of each I-frame is summed and compared with the value of the **threshold** argument.
4. When the luma content of one of the I-frames is above the value of the **threshold** argument, the function exits, and the CL484 returns to the DIGESTREADY state.

An example DisplayDigest(**XOffset**, **YOffset, skip, threshold, start1, start2**) command with parameters loaded is as follows:

DisplayDigest(0, 320, 10, 0x1000, 0x1223, 0x3402)

For this command, the following should be written to the Command FIFO:

0x061B 0x0000 0x0140 0x000A 0x1000 0x1223 0x3402

The command then starts reading the CD disc at address 12 Minutes, 23 Seconds and 34 Frames with Mode 2. The CL48x skips the first 10 I-pictures after the above address. Afterwards, I-pictures are decimated and displayed at pixel location 0 column and 320 row until the luma content of an I-picture is equal to or greater than 0x1000. The I-picture that caused the decoding to stop is the last new picture to be displayed.

DisplayGraphics(stl, stpl, address, fade)

Format: DisplayGraphics(**startline, stopline, address, fade**)
Priority: Low
Module Used: CL48xVCD and CL484CDG
Category: No State Transition
Command ID: 0x0417

This low-priority command does not cause a state transition and can be executed while in any state, including CDGIDLE and VCDIDLE. It may be used by either the CL48xVCD or CL484CDG microcode.

DisplayGraphics() instructs the CL48x 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, and Section 11.12 for more information about on-screen display (OSD).]

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.

The graphics overlay is disabled by issuing the DisplayGraphics() command with a value of **stopline** equal to zero.

DisplayGraphics() arguments are defined as follows:

- **startline** - Used to position the graphic vertically on the display from the line specified by the TOP_BORDER DRAM 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 relative to TOP_BORDER.
- **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 first active line, for which any value given is legal. Also, these values are in field lines (240 total for NTSC) rather than frame lines (480 total for NTSC).*

DisplayGraphics(stl, stpl, address, fade)

- **address** - The starting word address of the buffer in CL48x local DRAM which holds the graphics overlay. The starting address for the graphic overlay must be on a 256-word page boundary—that is, it must be divided by 256 before use (e.g., word address 0x1000 is written as 0x10).
- **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 (bit 15 and bits 0 to 9 should be set to 0). The larger the number for the fade factor, the more background will show through the graphic.

The DisplayGraphics() command does not wait for the graphic image to be displayed before checking the command FIFO for new 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:

```
unsigned int startline;     /* Video line number to start displaying*/
                            /* graphics overlay (601 lines / 2).*/
unsigned int stopline;      /* Video line number to stop displaying*/
                            /* graphics overlay. A value of zero will*/
                            /* disable the graphics overlay (601 lines / 2)*/
unsigned int address;       /* Address in CL48x local DRAM of the start of */
                            /* the graphics overlay buffer (16-bit, 256-word*/
                            /* page address).*/
unsigned int fade;          /* Amount of fade to apply to the graphic. 0-31 in*/
                            /* bits 14 through 10; 0 = no fade, 31 = max. fade*/
```

*Note: DisplayGraphics(**startline, stopline, address, fade**) used in CD-G mode is the same as VCD except for the following changes: When using the CD-G module, you cannot display graphics in the CD-G active region, but you may display graphics in either the top or bottom border regions. To display graphics in the bottom border, set bit 15 of the **startline** address; for example, to display graphics one line below the active window, issue DisplayGraphics(0x8001, 0x20, 0x42, 0x0). To display graphics above the CD-G active region, clear bit 15 of the **startline** address. See Section 11.11 for more information on CD-G.*

DisplayGraphics(stl, stpl, address, fade)

An example DisplayGraphics(**startline**, **stopline, address, fade**) command with parameters loaded is as follows:

DisplayGraphics(4, 240, 0x2C, 0x4000)

For this command, the following should be written to the Command FIFO:

0x0417 0x0004 0x01A0 0x002C 0x4000

This command causes the graphic at DRAM word address 0x2C to be displayed starting at line 4 and ending at line 240 of each field with a fade factor of 16 (0x10 shifted left by 10).

DisplayRomStill(romA1, romA2, length)

Format:	DisplayRomStill(**romAddress1, romAddress2, length**)
Priority:	Low
Module Used:	CL48xVCD
Category:	State Transition Command
Command ID:	0x030C

This command can be executed from the VCDIDLE state. It causes the CL48x to read **length** 16-bit words of the VideoCD-formatted still picture at the specified ROM address (**romAddress1** and **romAddress2**).

The bitstream must contain valid VideoCD 2.0 demultiplexed video data and must fit in the video bitstream buffer specified by the VBS_SADR and VBS_EADR configuration parameters (a ROM still picture should be smaller than 36.5 KBytes).

The CL48x transitions to the DISPLAYROMSTILL state while reading, decoding and displaying the video frame. Once the still picture is displayed, the CL48x transitions to the VCDIDLE state.

While decoding and displaying the still picture, audio decoding and output are disabled. Any data in the video bitstream buffer is flushed before the data from ROM is copied.

If the error conditions described below are found, the error status bit will be set in the MRC_STATUS word in DRAM:

- Bitstream is too long for bitstream buffer
- ROM address is invalid
- **romAddress** and **length** are beyond the end of the buffer

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.

DisplayRomStill(romA1, romA2, length)

DisplayRomStill(**romAddress1, romAddress2, length**) arguments are defined as follows:

```
unsigned int    romAddress1,      /* Location of VideoCD-formatted still picture.*/
                romAddress2;      /* romAddress1 is most significant word of the address*/
                                  /* and romAddress2 is least significant word of the*/
                                  /* address (The address used is a word address.)*/
unsigned int    length;           /* length of VideoCD-formatted still picture (in words)*/
                                  /* bitstream (the still picture must fit in the video*/
                                  /* bitstream buffer).*/
```

An example DisplayRomStill(**romAddress1**, **romAddress2, length**) command with parameters loaded is as follows:

DisplayRomStill(0x0000, 0x8000, 2048)

For this command, the following should be written to the Command FIFO:

0x030C 0x0000 0x8000 0x0800

This command copies the VideoCD 2.0 still-picture elementary bitstream at ROM word address 0x0800 whose length is 2048 words into the video bitstream buffer. The still picture is then decoded and displayed.

DisplayStill()

Format:	DisplayStill()
Priority:	Low
Module Used:	CL48xVCD
Category:	State Transition Command
Command ID	0x000C

The DisplayStill() command can be issued from the IDLE, PAUSE, FREEZE, PLAY, REPLAY, SINGLESTEP, SLOWMOTION, DISPLAYSTILL, or DISPLAYSTILLPAUSE states. It causes the CL48x to enter the DISPLAYSTILL state.

This low-priority command causes the CL48x microcode to read, decode and output normal- or high-resolution still pictures and audio samples from the incoming bitstream. This command continues to decode and output video and audio data until a Pause(), Reset(), Abort() or SetStreams() command is executed. [SetStreams() should be used before sending the CL48x a command which causes a motion-video MPEG stream to be decoded.]

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

If a Pause() is executed while the CL48x is displaying a still picture, the still picture will be completed before the Pause() is executed. In the DISPLAYSTILL state, the CL48x checks the Command FIFO every 120ms.

The CL48x can decode and display both high- and low-resolution still pictures as defined by the *Video CD Specification Version 2.0*. The SetStreams() command must be used to select the type of stream (high or low resolution) to be displayed.

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).

DisplayStill()

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).

For normal resolution pictures, the previous picture is displayed until a new picture is decoded. When the new picture is ready, 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.

If the currently displayed picture is of high-resolution and the new picture being decoded is normal resolution, the display is blanked to the background color while the picture is decoded.

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 zero, the screen will be blanked before the new picture is displayed.

DisplayStill(0, 0, 0, 0)

Format:	DisplayStill(**0, 0, 0, 0**)
Priority:	Low
Module Used:	CL48xVCD
Category:	State Transition Command
Command ID:	0x040C

The DisplayStill(**0, 0, 0, 0**) command can be issued from the DISPLAYSTILL or DISPLAYSTILLPAUSE states. This low-priority command restarts the previous DisplayStill() command. The CL48x microcode does not flush the bitstream buffers; it starts to decode with the next start code in the bitstream buffers.

The DisplayStill(**0, 0, 0, 0**) command can be used to restart a previously issued DisplayStill() command with four arguments. If the DisplayStill(**0, 0, 0, 0**) 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. If the **stop** sector address has been detected, the DisplayStill(**0, 0, 0, 0**) command will terminate without decoding or outputting any audio or video data.

This command causes the processing state to transition to DISPLAYSTILL. If the previous DisplayStill() command was a DisplayStill() command with four arguments, the IDLE state will be entered when the **stop** sector address is detected.

For a discussion of normal-to-high resolution and high-to-normal resolution picture transitions, please see the DisplayStill() discussion on the previous page.

DisplayStill(start1, start2)

Format:	DisplayStill(**start1, start2**)
Priority:	Low
Module Used:	CL48xVCD
Category:	State Transition Command
Command ID:	0x020C

This command can be issued from the DISPLAYSTILL, IDLE, PAUSE, DISPLAYSTILLPAUSE, PLAY, REPLAY, FREEZE, SINGLESTEP and SLOWMOTION states. It displays the first still picture encountered after the **start1** and **start2** addresses. After DisplayStill() is executed, the CL48x goes into the IDLE state.

> *Note: This command does not decode audio. If audio decode is required, use DisplayStill(**start1, start2, stop1, stop2**).*

This command searches the incoming bitstream for the sector address indicated by the arguments **start1** and **start2**. Once the start sector address is detected, the CL48x decodes and displays a single still picture having the stream identifier specified by the VA_STREAM_ID configuration parameter or set using the SetStreams() command. If while searching for the start sector address an address greater than the start address is detected, the CL48x microcode will display the next picture found.

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.

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 CL48x microcode to examine the Command FIFO.

DisplayStill(start1, start2)

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

```
unsigned int start1, start2;    /* Disc sector address (BCD values stored in two words) to */
                                /* start decoding: The MSB of start1 specifies the Minutes, and */
                                /* the LSB of start1 specifies the Seconds. The MSB of start2 */
                                /* specifies the Frame, and the LSB of start2 specifies the Mode */
                                /* (Mode must be 2).*/
```

DisplayStill(start1, start2, stop1, stop2)

Format:	DisplayStill(**start1, start2, stop1, stop2**)
Priority:	Low
Module Used:	CL48xVCD
Category:	State Transition Command
Command ID:	0x040C

This command can be issued from the IDLE, PAUSE, PLAY, REPLAY, FREEZE, DISPLAYSTILLPAUSE, SINGLESTEP, SLOWMOTION, or DISPLAYSTILL states. Audio decoding is performed with this command.

Once decoding has begun (an address is found that is greater than the **start** address but less than the **stop** address), decoding and outputting continues until the specified **stop** sector address is detected. After detecting the sector **stop** address, the CL48x microcode enters the IDLE state.

If while searching for a **start** address, an address greater than the **stop** address is detected, the CL48x will generate an AOR interrupt, if enabled, and enter the IDLE state. The CL48x will begin decoding with the first picture whose address is greater than or equal to the **start** address but less than the **stop** address.

If while decoding still pictures, an address less than the **start** address is detected, the CL48x microcode will generate an AOR interrupt, if enabled, and enter the IDLE state.

For a discussion of normal-to-high resolution and high-to-normal resolution picture transitions, please see the DisplayStill() command description.

If a Pause() command is executed while new pictures are being decoded and displayed by the DisplayStill(**start1, start2, stop1, stop2**) command, the CL48x will enter the PAUSE state. To continue executing the DisplayStill(**start1, start2, stop1, stop2**) command from the current location, go back to the beginning previous sector (see LSA_MS and LSA_FM in Table 15-1) and issue a DisplayStill(**start1, start2, stop1, stop2**) command with **start1** = **start2** = **stop1** = **stop2** = 0. This will cause DisplayStill(**start1, start2, stop1, stop2**) to decode and display until the previous **stop** address is reached. (See DisplayStill(**0,0,0,0**) discussion.)

DisplayStill(start1, start2, stop1, stop2)

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

```
unsigned int start1, start2;    /* Disc sector address (BCD values stored in two words) to */
                                /* start decoding: The MSB of start1 specifies the Minutes, and */
                                /* the LSB of start1 specifies the Seconds. The MSB of start2 */
                                /* specifies the Frame, and the LSB of start2 specifies the Mode */
                                /* (Mode must be 2).*/

unsigned int stop1, stop2;      /* Disc sector address (BCD values stored in two words) to */
                                /* stop decoding: The MSB of stop1 specifies the Minutes, and */
                                /* the LSB of stop1 specifies the Seconds. The MSB of stop2 */
                                /* specifies the Frame, and the LSB of stop2 specifies the Mode */
                                /* (Mode must be 2).*/
```

Note: All four arguments need to be programmed.

An example DisplayStill(**start1**, **start2, stop1, stop2**) command with parameters loaded is as follows:

DisplayStill(0x0340, 0x3502, 0x1025, 0x1002)

For this command, the following should be written to the Command FIFO:

0x040C 0x0340 0x3502 0x1025 0x1002

This command will search for an address greater than 3 minutes, 40 seconds, frame 35, but less than 10 minutes, 25 seconds frame 10. Once this address is found, still pictures are decoded and displayed until an address greater than or equal to 10 minutes, 25 seconds, frame 10 is detected.

When decoding still pictures, an AOR interrupt is generated and decoding is stopped if an address is found which is:

- Less than or equal to the start address: 3 minutes, 40 seconds, frame 35
- Greater than or equal to the stop address: 10 minutes, 25 seconds, frame 10

While decoding still pictures, any audio frames read are decoded and output.

DumpData(st1, st2, length, address)

Format:	DumpData (**start1, start2, length, address**)
Priority:	Low
Module Used:	CL48xVCD
Category:	State Transition Command
Command ID:	0x0414

This low-priority command is available in the CL48x microcode only. This command causes the CL48x microcode to read Mode 1 or Mode 2 Form 1 CD-ROM data from the specified starting **address** and for the specified **length** sectors (up to a maximum of 16 sectors). The CL48x also performs ECC error correction, as necessary, and places the data into a buffer in DRAM.

> *Note: The ECC error-correction process requires both a storage buffer and a working buffer in DRAM. Therefore, if three sectors are to be read, six sectors worth of DRAM are required to execute this command. The resulting data, which consists of the error-corrected user data portion of the CD data (i.e., sector header information is removed), will be stored starting at the address given in the* ***address*** *argument.*

The starting address of this DRAM buffer is specified by the **address** parameter. The buffer must start on a 256-word boundary—i.e., the address parameter must be the DRAM word address shifted to the right by 8. For example, to write to 0x4000, the address parameter should be written with a 0x40. 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 another 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.

DumpData(st1, st2, length, address)

Table 13-2 DumpData() Error Condition

MRC_STATUS	Error Condition
INVALID_SECTOR_COUNT	**Length** parameter not within valid range (1 to 16)
INVALID_BUF_ADDRESS	**Address** parameter specifies a dump to a location less than FREE_SPACE_START.
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.) Note: Unlike DisplayStill(), the actual sector address indicated by **start1** and **start2** must exist in the input bitstream.
SECTOR_TYPE_ERROR	—Mode is 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 CL48x 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 CL48x is in a command state other than IDLE, the Abort() command can be used to move the CL48x to the IDLE state.

DumpData() arguments are defined as follows:

```
unsigned int start1, start2;   /* Disc sector address (BCD values stored in 2 words)*/
                               /* to start decoding: The MSB of start1 specifies the Minutes,*/
                               /* and the LSB of start1 specifies the Seconds. The MSB of*/
                               /* start2 specifies the Frame, and the LSB of start2 specifies*/
                               /* the Mode. (Mode must be 1 or 2.)*/
unsigned int length;           /* Length of sequence to read (1 ≤ Length ≤ 16) */
                               /* 16 sectors) (Hex coded)*/
unsigned int address;          /* Address in CL48x DRAM to store the*/
                               /* data (16 bit 256 word page address). The*/
                               /* size of this buffer should be (length * 2048) x 2*/
                               /* bytes (Hex coded). */
```

Freeze()

Format:	Freeze()
Priority:	Low
Module Used:	CL48xVCD
Category:	State Transition Command
Command ID:	0x0010

The Freeze() command should only be issued from the PLAY or REPLAY states.

This low-priority command causes the CL48x microcode to stop outputting new video frames. Audio decode and output continues.

If a video frame is being displayed at the time the Freeze() command is executed, the video frame will be repeatedly 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, and the last decoded video frame will continue to be displayed.

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
Module Used:	CL48xVCD
Category:	No State Transition
Command ID:	0x8003

This high-priority command causes the CL48x 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 outdated before the buffer emptiness computation is complete.

ModuleLoad(module)

Format:	ModuleLoad(**module**)
Priority:	Low
Module Used:	CL484VCD and CL484CDG
Category:	State Transition Command
Command ID:	0x011F

This low-priority command can be executed from any state and is available in the CL484 VCD and CD-G microcode modules. This command causes a new module to be loaded from ROM into DRAM.

The VCD module is the default module loaded when the CL484 detects a hardware reset signal on its RESET pin. The VCD module is capable of reading, decoding and outputting an MPEG-1 system bitstream from a Video-CD or CD-I disc. The VCD module is also capable of playing CD-DA disks and detecting the presence of CD-G data on the CD-DA disc.

The CD-G module can only be loaded using the ModuleLoad() command. The CD-G module is capable of playing CD-DA disks while also decoding and displaying any CD-G bitstream contained on the disk.

The CL484 will transition to the IDLE state upon completion of this command. All configuration parameters are re-initialized to the value set in the ROM. The ModuleLoad() command is completed when the location the command is written to is cleared (when the module is loaded) and when the high-priority command (0x70) is zeroed when the microcode initializes. In the CD-G code, the data transitions from 0x8000 to 0x0000; in the VCD code, the data transitions from 0xFFFF to 0x0000.

ModuleLoad() arguments are defined as follows:

```
unsigned int module;     /* Identifier of the CL484 module to load*/
                         /* 0 = load the VCD module*/
                         /* 1 = load the CD-G module*/
```

Note: Issuing this command is equivalent to a hardware reset. The screen is blanked to the default color and all data in the FREE_SPACE DRAM region is lost.

Pause()

Format:	Pause()
Priority:	Low
Module Used:	CL48xVCD
Category:	State Transition Command
Command ID:	0x000E

This low-priority command, not available in CD-G, can be issued from any of the following VCD states: PLAY, FREEZE, REPLAY, SINGLESTEP, SLOWMOTION or DISPLAYSTILL. It causes the CL48x microcode to stop decoding and outputting audio data and video frames.

The END-P interrupt will be asserted after the CL48x enters the PAUSE state if this interrupt is enabled.

The new state after processing this command will be either the PAUSE or DISPLAYSTILLPAUSE state.

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 or DISPLAYSTILLPAUSE state.

PitchShift(semitones, mode)

Format:	PitchShift(**semitones, mode**)
Priority:	Low
Module Used:	CL484VCD and CL484CDG
Category:	Non State Transition Command
Command ID:	0x0222

This low-priority command is available in both the CL484 VCD and CL484 CD-G microcode sets. For VCD, this command can be issued from the VCDIDLE, PLAY, PAUSE, FREEZE, REPLAY, SINGLESTEP, and SLOWMOTION states. For CDG, this command can be issued from the CDGIDLE, CDGPLAY, CDGFREEZE, and CDGDISPLAYMODE states. PitchShift(), in either mode, does not cause a state transition.

PitchShift() enables, disables, and controls the amount of shift in pitch of the audio stream (MPEG-1, CD-G, or CD-DA).

The amount of pitch is controlled by the **semitones** argument (there are 12 semitones per octave in Western music). Setting **semitones** to 0 performs a pitch shift of zero semitones.

The **mode** argument specifies the channel in which the pitch is shifted and output. Setting **mode** to 0 disables the pitch shift. Modes 1 and 2 are valid in the CD-DA, MPEG playback and CD-G modes. Mode 3 is valid in the CD-DA and CD-G modes, but not during MPEG playback.

> *Note: Key control must be turned off when still pictures are displayed and when DisplayDigest( ) is used.*

PitchShift() arguments are defined as follows:

```
signed int semitones;      /* The amount of pitch shift from the original pitch)*/
                           /* to perform in semitones.*/
                           /* Valid range is -12 to 12 (decimal values)*/
unsigned int mode;         /* Mode of operation:*/
                           /* 0: disable pitch shift*/
                           /* 1: mono right channel only to be output on both channels*/
                           /* 2: mono left channel only to be output on both channels*/
                           /* 3: stereo (can only be used in CDG and CD-DA)*/
```

See key control discussion in Section 11.13.

Play()

Format:	Play()
Priority:	Low
Module Used:	CL48xVCD and CL484CDG
Category:	State Transition Command
Command ID:	0x000D

This low-priority command is available in both the CL48x VCD and CL484 CD-G microcode sets. For VCD, this command can be issued from the IDLE, PAUSE, DISPLAYSTILLPAUSE, FREEZE, REPLAY, SINGLESTEP, SLOWMOTION, and DISPLAYSTILL states. For CD-G, this command can be issued from the CDG IDLE, CDGFREEZE, and CDGDISPLAYMODE states.

VCD Microcode

In the VCD microcode, Play() causes the CL48x microcode to start decoding the incoming bitstream and outputting decoded video and/or audio. Before the decoding begins, the CL48x 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 PLAY_MODE 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 and PLAY_MODE is set to CD_ROM_AUTO, the CL48x microcode starts the bitstream auto determination process explained in Chapter 1. If at any time the bitstream changes, the CL48x microcode will try to determine the type of the new bitstream and change its processing to handle the new bitstream type.

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 output.

This command causes a state transition to the PLAY state. The CL48x will remain in the PLAY state until the execution of another command which causes the state to change, or unless an auto-pause sector causes the CL48x to pause. The new state will be the state indicated by the command executed.

If the CL48x encounters an auto-pause sector during decoding, the CL48x will enter the PAUSE state if the AUTO_PAUSE configuration parameter (described in Table 15-7) is enabled. If this occurs, the CL48x will behave exactly as though a Pause() command was issued by the host controller.

CD-G Microcode

In the CD-G microcode, Play() causes the CL484 to start decoding the incoming bitstream and to output decoded graphics and audio. Before the decoding begins, the CL484 initializes the following components:

- *The subcode interface:* Initialized according to the SUBCODE_CNTL configuration parameter (see Table 15-7).
- *The Video Output unit:* Initialized according to the video configuration parameters (see Table 15-7).

The software initialization routine flushes all the picture and audio buffers and starts displaying graphics.

Play(start1, start2, stop1, stop2)

Format:	Play(**start1, start2, stop1, stop2**)
Priority:	Low
Module Used:	CL48xVCD
Category:	State Transition Command
Command ID:	0x040D

This command can be executed from the IDLE, SLOWMOTION, FREEZE, PAUSE, REPLAY, SINGLESTEP, DISPLAYSTILL, and DISPLAYSTILLPAUSE states, and will cause the CL48x to transition to the PLAY state. (This command is not applicable to CD-G.)

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 sector addresses at which to start and stop decoding.

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 CL48x begins decoding the incoming bitstream and displaying the resulting video and/or audio until the stop address is detected. When the stop address is detected, the CL48x 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 CL48x microcode will generate an AOR interrupt, if enabled, and enter the PAUSE state. If a sector address received is less than the start sector address once decoding has begun, the CL48x microcode will again generate the AOR interrupt, if enabled, 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 CL48x microcode will transition to decoding and outputting the CD-DA data and will 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 CL48x microcode to examine the Command FIFO.

Play(start1, start2, stop1, stop2)

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. In other words, if a Freeze(), Replay() or SlowMotion() command is executed prior to completion of Play(**start1**, **start2**, **stop1**, **stop2**), the Freeze(), Replay() or SlowMotion() command will also stop at the stop address specified by the argument to the Play() command.

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

```
unsigned int start1, start2;    /* Disc sector address (BCD values stored in two words) to */
                                /* start decoding: The MSB of start1 specifies the Minutes, and */
                                /* the LSB of start1 specifies the Seconds. The MSB of start2 */
                                /* specifies the Frame, and the LSB of start2 specifies the Mode */
                                /* (Mode must be 2).*/

unsigned int stop1, stop2;      /* Disc sector address (BCD values stored in two words) to */
                                /* stop decoding: The MSB of stop1 specifies the Minutes, and */
                                /* the LSB of stop1 specifies the Seconds. The MSB of stop2 */
                                /* specifies the Frame, and the LSB of stop2 specifies the Mode */
                                /* (Mode must be 2).*/
```

ReadSequenceHeader()

Format:	ReadSequenceHeader()
Priority:	Low
Module Used:	CL48xVCD
Category:	State Transition Command
Command ID:	0x0023

This low-priority command, which has no arguments, can only be issued in the VCDIDLE state, and is valid only when the stream identifier within the data packets corresponds with the value of the VIDEO_STREAM_ID for MPEG motion video. ReadSequenceHeader() is used to ensure that a sequence header is read before jumping to the middle of a track.

> *Note: This command, though intended for MPEG motion video streams, will allow still slideshows to return a valid sequence header. However, a CD-DA bitstream will never find a valid sequence header.*

ReadSequenceHeader() causes the CL48x microcode to decode the incoming MPEG stream and enter the READSEQHEAD state until a sequence header is read. When a sequence header is read, the CL48x transitions to the VCDIDLE state. No output of the incoming bitstream is performed during the execution of this command.

If a sequence header is not found, the CL48x will remain in the READSEQHEAD state (PROC_STATE = 0x23) until an Abort() or Reset() command is issued.

If this command is issued after a sequence header has already been found [i.e., not immediately after a Reset() or Abort() command] the following events will happen: If the current output is high-resolution still, it will be blanked immediately. Otherwise, the current video output will remain on the screen until a new sequence header is found.

If a new sequence header is found that is identical to the old sequence, the CL48x will enter the PLAY state and play video with no audio. If the new sequence header is different from the current sequence header, the screen will be blanked, and the CL48x will enter the VCDIDLE state.

ReadSequenceHeader()

Proper operation of this command is as follows:

1. Issue the Reset() command.
2. Back the CD head to the beginning of the track (where the sequence header is) and pause it.
3. Issue the ReadSequenceHeader() command.
4. Un-pause the CD head.

Sequence header parameters written to the MPEG-1 header area are described in Table 13-3.

Table 13-3 MPEG-1 Sequence Header Parameters

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
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.

Replay()

Format:	Replay()
Priority:	Low
Module Used:	CL48xVCD
Category:	State Transition Command
Command ID:	0x001C

This low-priority command can be issued from the PLAY, PAUSE, FREEZE, SINGLESTEP, REPLAY, or SLOWMOTION states. It restarts the previous Play() command. (This command is not applicable to CD-G.)

Replay() does not initialize hardware or software. The CL48x microcode does not flush the bitstream buffers; instead, it starts to decode with the next word in the bitstream buffers.

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 REPLAY. If the previous Play() command was a Play() command with arguments, the PAUSE state will be entered when the stop sector address is detected.

Reset()

Format:	Reset()
Priority:	High
Module Used:	CL48xVCD and CL484CDG
Category:	State Transition Command
Command ID:	0x8000

Reset() is a high-priority macro command which is used to halt the execution of the current command and re-initialize the CL48x 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 CL48x 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. Execution of the Reset() command is complete when the high-priority command ID becomes zero.

With the VCD module, when suspending and resuming decode operations, or when changing from decoding one bitstream to another, the Pause() command should be used. The Reset() command, unlike the Pause() command, allows the CL48x to continue to display the last picture decoded.

The Reset() command will not clear OSD in SYNC In mode but will clear OSD in SYNC Out mode.

Scan()

Format:	Scan()
Priority:	Low
Module Used:	CL48xVCD
Category:	State Transition Command
Command ID	0x000A

The Scan() command can be issued from the PLAY, REPLAY, PAUSE, FREEZE, DISPLAYSTILL, DISPLAYSTILLPAUSE, VCDIDLE, SINGLESTEP, or SLOWMOTION states.

The Scan() command searches for the first I-picture in the buffer, decodes it, and displays the picture. If the I-picture includes user data, the user data will be processed as described in Table 15-4.

After one I-picture is displayed, the CL48x generates an END-P, if enabled, and enters the VCDIDLE state.

> *Note: Audio is played during Scan(). Use AUDIO_MUTE to disable audio, if desired.*

SetStreams()

Format:	SetStreams(**VideoStreamID, AudioStreamID**)
Priority:	Low
Module Used:	CL48xVCD
Category:	State Transition Command
Command ID:	0x0213

This low-priority command can be executed from VCDIDLE, PLAY, FREEZE, REPLAY, PAUSE, SINGLESTEP, SLOWMOTION, DISPLAYSTILL, and DISPLAYSTILLPAUSE. If issued from VCDIDLE, Setstreams() can only transition to SCAN, PLAY and DISPLAYSTILL after first going through DISPLAYSTILLPAUSE.

Setstreams() instructs the CL48x 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 the following commands: Play(), Replay(), Freeze(), SingleStep(), Scan(), SlowMotion(), DisplayDigest(), and DisplayStill().

The host processor can use this command to disable audio or video output by specifying a valid audio or video stream identifier not found in the incoming bitstream. Table 15-7 gives the default setting for the stream identifiers. Valid identifiers are 0xE0-0xEF for video data, and 0xC0-0xDF for audio data. Invalid identifiers will cause this command to be ignored by the CL48x.

This command causes the CL48x to transition to the PAUSE or DISPLAYSTILLPAUSE states.

The SetStreams() arguments are defined as follows:

```
unsigned int VideoStreamID;     /* New video stream ID*/
unsigned int AudioStreamID;     /* New audio stream ID*/
```

SetVideoFormat()

Format:	SetVideoFormat()
Priority:	Low
Module Used:	CL48xVCD and CL484CDG
Category:	No state transition
Command ID:	0x0005

Format:	SetVideoFormat (**format, 0, 0**)
Priority:	Low
Module Used:	CL48xVCD and CL484CDG
Category:	No state transition
Command ID:	0x0305

This low-priority command has two variations: with arguments and without arguments. Both commands can be issued any time after the microcode has been loaded and initialization is complete. No processing state transition occurs.

When issued *without* arguments (command ID 0x0005), SetVideoFormat() sets the video output **format** to the default value from the Configuration Area. This command should always be executed after the video configuration parameters are modified in DRAM during decoding.

When issued *with* arguments (command ID 0x0305), SetVideoFormat(**format, 0,0**) instructs the CL48x microcode to modify the video output **format** according to the argument given and the video configuration parameters described in Table 15-7. If the video **format** specified in the input bitstream doesn't match the video output **format**, picture rate conversion is performed. SetVideoFormat() arguments (command ID 0305) are defined as follows:

```
unsigned int format;        /* New video output format: 2 = Multi-sync format (SYNC out only)*/
                            /* (output format is set to video format of input bitstream)*/
                            /* 3 = PAL_FORMAT (if input video is NTSC, picture will be */
                            /* interpolated [vertically scaled] to PAL picture size)*/
                            /* 4 = NTSC_FORMAT*/
                            /* 5 = PAL_FORMAT1 (Converts to PAL format but does not*/
                            /* interpolate NTSC video as does format value 3 above.*/
```

SetVideoFormat()

> *Note: For no field repeat, 704x240 (NTSC) or 704x288 (PAL) pixels will be output each VSYNC, and the EVEN_INTERPCOEF DRAM parameter will be used when outputting pictures. This will make luma and chroma the same for both fields.*

SetVideoFormat(**format, 0,0**) for the CDG microcode functions the same as VCD microcode except only PAL (3) and NTSC (4) settings are supported. A setting of 2 is not supported.

The following two configuration area parameters that are related to the SetVideoFormat(**format, 0, 0**) command can be used to center the active CD-G output both horizontally and vertically (See Table 15-7):

- TOP_BORDER is the number of lines displayed above the active CD-G screen. By increasing this, the number of lines available for displaying OSD in the top border will also be increased.
- LEFT_BORDER is the absolute number of pixels to the left of the active CD-G screen. The first 0x82 pixels (default value) will not be seen.

The SetVideoFormat() command must be issued after these values are changed in order for them to take effect.

SingleStep()

Format:	SingleStep()
Priority:	Low
Module Used:	CL48xVCD
Category:	State transition command
Command ID:	000B

This low-priority command can be executed from the PLAY, REPLAY, FREEZE, SINGLESTEP, SLOWMOTION or PAUSE states and will cause the CL48x to transition to the SINGLESTEP state.

This command instructs the CL48x microcode to decode and display the next picture in the video input buffer. Audio output is muted while decoding and displaying the picture.

If the picture has associated user data, the user data will be processed as described in Table 15-4. An END-P interrupt will be generated when the CL48x video bitstream buffer fills.

RDY-S is issued whenever the video bitstream buffer falls below a predefined level.

SlowMotion()

Format:	SlowMotion(**N**)
Priority:	Low
Module Used:	CL48xVCD
Category:	State transition command
Command ID:	0109

This low-priority command instructs the CL48x microcode to accept incoming data and decode pictures, displaying each picture for **N** frame times. Audio is muted, and audio and video synchronization is suspended.

This command causes the SLOWMOTION state to be entered, unless an error is received from the disc, in which case the PLAY state is entered. This command can be issued from the PLAY, REPLAY, PAUSE, FREEZE, SINGLESTEP or SLOWMOTION states.

If the picture has associated user data, the user data will be processed as described in Table 15-4. An END-P interrupt will be generated when the CL48x video bitstream buffer fills.

If the previous command was a Play() command with arguments, the PAUSE state will be entered when the stop sector address is detected, and the last decoded video frame will continue to be displayed.

RDY-S is issued whenever the video bitstream buffer falls below a predefined level.

The SlowMotion() argument is defined as follows:

```
unsigned int N;          /* 1 ÷ N is the factor by which decode speed*/
                         /* is reduced. N must be ≤ 16. For example,*/
                         /* SlowMotion(4) causes SlowMotion*/
                         /* playback at one-quarter normal speed.*/
```

14 Interrupts

The CL48x's 15 host interrupts are produced by the CL48x microcode, which monitors internal conditions while the decode and display processes execute. The host can enable each of the 15 interrupts independently through the INT_MASK configuration parameter.

When an interrupt occurs, the $\overline{\text{INT}}$ pin is asserted (active low), and the event which caused the interrupt is indicated by the contents of the INT_STATUS (interrupt status) field in the DRAM Status Area. 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. In addition, the INT_STATUS field may indicate that multiple interrupts have occurred simultaneously.

The host processor must service all interrupts indicated in the INT_STATUS field before clearing the CL48x interrupt. To respond to an interrupt, the host should do the following:

1. Read the DRAM INT_STATUS location in the Status Area. (INT_SOURCE may also be read at this time, if desired.)
2. Handle the interrupt.
3. Write 0 to the *Int* bit of register HOST_int (see Chapter 10) using a read-modify-write.

4. Write 0 to the INT_SOURCE and INT_STATUS locations (in this order) in the DRAM Status Area.

Figures 14-1 shows the contents of the INT_STATUS and INT_MASK fields when either the CL480 or CL484 VCD modules are loaded.

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

Figure 14-1 INT_STATUS and INT_MASK Bit Fields

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

Table 14-1 CL48x Interrupt Summary

Category	Interrupt Name	Event	Mask Bit
Decode-time	AOR	Address out of range	7
	A/E/E	Submode is auto pause, end of record/file	15
	END-C	High-priority command execution complete	9
	END-D	End of DumpData() command	14
	END-P	End of Pause(), or buffer full in SlowMotion() or SingleStep()	13
	ERR	Bitstream data error	0
	PIC-D	Picture decode complete	5
	UND	Bitstream buffer underflow error	8
	RDY-S	Ready for data during SlowMotion() or SingleStep()	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

14.1 Interrupt Types

Each of the 15 interrupts produced by the CL48x 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, if enabled, they are issued following the active (falling) edge of $\overline{\text{VSYNC}}$. The timing of the VSYNC-V interrupt relative to the $\overline{\text{VSYNC}}$ signal is as follows: it will occur sometime after the leading edge of $\overline{\text{VSYNC}}$ and before the active region of the output display. All Display-time interrupts are generated at the same time.
- *Decode-Time Interrupts*: These interrupts 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 $\overline{\text{INT}}$ pin to be asserted. To determine which interrupt(s) has occurred, the DRAM INT_STATUS location is written by the CL48x at the same time the $\overline{\text{INT}}$ pin is asserted.

Each bit in INT_STATUS corresponds to a different interrupt source as shown in Figure 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 CL48x HOST_int register and then writing a zero into the INT_STATUS word.

The same handshaking protocol is used for all CL48x interrupts, regardless of the logical interrupt source. When the CL48x issues an interrupt to the host, it makes the $\overline{\text{INT}}$ 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 already 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 CL48x microcode from issuing subsequent interrupts as distinct events.

14.3 Interrupt Queuing

When the CL48x is about to generate an interrupt, it first reads the INT_STATUS register. If the INT_STATUS register does not equal 0, the interrupt bit is set in an internal interrupt queue. In this manner, each type of interrupt may be queued once.

An internal timer causes the CL48x to periodically check the INT_STATUS register to see if it equals 0. If so, the interrupt queue register value is written into the INT_STATUS register, and $\overline{\text{INT}}$ is asserted.

Note that interrupts may not always occur exactly when expected if the host allows interrupts to queue due to slow interrupt handling.

14.4 Interrupt Listing

This section gives a detailed description of the CL48x display-time and decode-time interrupts, listed together in alphabetical order.

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

A/E/E

The A/E/E interrupt is generated as the CL48x microcode reads the CD sector information being input. If the submode 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 submode field is as follows:

- Bit 0 of the submode field indicates an end-of-record sector.
- Bit 4 of the submode indicates an auto-pause sector.
- Bit 7 of the submode 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 the end-of-record, auto-pause or end-of-file bit set. This interrupt will be generated during the input bitstream and demultiplex process when the submode field is read.

> *Note: This interrupt notifies the user that the auto-pause bit is set in the submode header. If you wish to have the CL48x 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 CL48x will issue an END-P interrupt when the PAUSE state is entered.*

The timing between the submode field of the subheader entering the CL48x and the A/E/E interrupt being produced cannot be guaranteed.

This interrupt is not available in CD-DA or CD-G modes.

AOR

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

The AOR interrupt is generated when the CD sector address is out of range for the Play() or DisplayStill() commands with four arguments. The interrupt is generated when the CL48x microcode reads and processes the sector address(es).

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.

The AOR interrupt is generated when the CD-ROM sector header is read from the incoming bitstream. The timing between the CD-ROM sector address entering the CL48x and the AOR interrupt being produced cannot be guaranteed.

This interrupt is not available in CD-DA or CD-G modes.

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

END-C

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 CL48x microcode has updated the buffer emptiness fields in the Status Area (VIDEO_EMPTINESS and AUDIO_EMPTINESS) upon completion of the InquireBufferEmptiness() command.

This interrupt is not available in CD-G mode.

END-D

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

The END-D interrupt indicates that the CL48x microcode has completed the previous DumpData() command. The data read by the DumpData() command is stored in DRAM at the host-specified address.

The SE_STATUS DRAM parameter (Table 15-1) indicates which sectors are valid and which contain errors.

> *Note: If an error occurs in the DumpData() command, the END-D interrupt will still occur, but the error-corrected data may not be stored in DRAM.*

The timing between the last sector entering the CL48x and the END-D interrupt being produced cannot be guaranteed.

This interrupt is not available in CD-G mode.

END-P

Event: PAUSE state entered, or video bitstream full
Category: Decode-time
Mask Bit: 13

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

- The CL48x transitioning to the PAUSE state after the Pause() command has been executed
- The CL48x reading an auto-pause sector after the AUTO_PAUSE DRAM parameter has been enabled
- After Scan() command has displayed a picture
- The video bitstream buffer has reached the high-threshold level while in the SLOWMOTION or SINGLESTEP states. VideoCD microcontrollers use END-P to determine when to stop sending data to the CL48x and where to position the CD heads when sending the next block of data to the CL48x.

This interrupt is not available in CD-DA or CD-G modes.

END-V

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

The END-V interrupt event occurs after the falling (active) edge of $\overline{\text{VSYNC}}$ when the picture which caused the interrupt to occur is first displayed.

The END-V interrupt indicates the last picture (in display order) decoded before a `sequence_end_code` (see MPEG-1 specification) is about to be displayed.

The `sequence_end_code` will have been decoded before this interrupt will be generated.

This interrupt is not available in CD-DA or CD-G modes.

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

ERR

This interrupt is generated in VCD mode, but not in CD-G mode.

The ERR (bitstream data error) interrupt indicates 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).

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

- The ERR interrupt is enabled (See the INT_MASK DRAM parameter).
- One of the following errors is detected in the coded data stream while decoding is being performed:
  - An MPEG `sequence_error_code`
  - Invalid variable-length codes (VLCs)
  - `marker_bits` with 0 values
  - Header fields with illegal values
  - CD-ROM sector errors
  - Picture size is larger than expected (stills only)
  - Still picture type not intra or first slice startcode missing
  - System or packet header size indicates pack or packet goes beyond current sector
  - First four bytes and second four bytes within sector don’t match
  - Sector mode field is not equal to two
  - No data after RDY-S within a two-second timeout

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.

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.

GOP-V

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

The GOP-V interrupt event occurs after the falling edge of $\overline{\text{VSYNC}}$ before the first display field of the first picture decoded following the detection of a group_start_code (see MPEG-1 specification).

A GOP interrupt event occurs if the picture to be displayed in the following field time has not been previously displayed.

This interrupt is not available in CD-DA or CD-G modes.

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

PIC-D

The PIC-D (new picture decoded) interrupt event occurs each time both 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)

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 not issued until the $\overline{\text{VSYNC}}$ prior to the display of the picture.

A PIC-D interrupt is not generated for a picture if an error is detected in the picture data during decoding.

This interrupt is not available in CD-DA or CD-G modes.

PIC-V

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

The PIC-V (new picture displayed) interrupt event occurs after the falling edge of $\overline{\text{VSYNC}}$ before the first display field of a newly-decoded picture.

A PIC-V interrupt event occurs if the picture to be displayed in the following field time has not been previously displayed.

If an error was detected in the I- and P-picture data by the CL48x during decoding, a PIC-V interrupt will *not* be generated.

This interrupt is not available in CD-DA or CD-G modes.

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

RDY-S

The RDY-S interrupt event—ready for data during SlowMotion() or SingleStep()—occurs when the amount of data in the video bitstream rate buffer in DRAM has fallen below the low threshold value.

RDY-S is used in conjunction with the END-P interrupt to control data flow during execution of the SlowMotion() and SingleStep() commands.

RDY-S is used by the host to determine when more data needs to be sent to the CL48x.

This interrupt is not available in CD-DA or CD-G modes.

SEQ-V

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

The SEQ-V interrupt event occurs after the falling edge of $\overline{\text{VSYNC}}$ prior to the first display field of the first picture decoded following the detection of a `sequence_start_code` (see MPEG-1 specification).

A SEQ-V interrupt event occurs if the picture to be displayed in the following field time has not been previously displayed.

This interrupt is not available in CD-DA or CD-G modes.

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

UND

This interrupt is not available in CD-DA or CD-G modes.

The UND (bitstream buffer underflow) interrupt event occurs when the number of valid bytes in the bitstream buffer in DRAM is 0 while the CL48x microcode expects the bitstream buffer to contain data. Both audio and video bitstream buffers are checked while in the PLAY, REPLAY, DISPLAYSTILL or SCAN states. The video bitstream buffer is checked in the SLOWMOTION state, and the audio bitstream buffer is checked in the FREEZE state.

> *Note: This interrupt is generated within the CL48x's internal timer interrupt routine and, thus, some underflows may go undetected due to the sampling interval of the timer interrupt.*

This interrupt can be used by the host to detect a bitstream buffer underflow error condition.

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

- The number of valid words in the relevant 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 an UND interrupt is produced, new UND interrupts will be produced continuously until the host takes steps to put data into the bitstream buffer.*

The host software should perform the following steps to stop the generation of this interrupt:

- Put data in the buffer
- Clear the interrupt *twice* (one may be queued)

USR

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

The USR interrupt is generated after the CL48x copies any user data from the MPEG-1 picture header into the User Data FIFO (see Table 15-4). MPEG-1 picture header user data is copied to the user data FIFO even if the USR interrupt is disabled.

> *Note: See the VideoCD 2.0 specification for information on the user data portion of the video stream.*

No checks are performed to prevent the User Data FIFO from overflowing. When an overflow occurs, older user data is overwritten by newer user data, and no indication of an overflow is available.

The size of the User Data FIFO may be determined by reading UDF_SADDR and UDF_EADDR in DRAM, as described in Table 15-4.

Use the read and write pointers (UDF_READ and UDF_WRITE) to determine the location and amount of data to read from the User Data FIFO.

The host cannot detect User Data FIFO overflow.

This interrupt is not available in CD-DA or CD-G modes.

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 to the CL48x in slave mode.

This interrupt is not available in CD-G mode.

15 DRAM Configuration Reference

This chapter describes the DRAM configuration reference, a collection of locations in local DRAM that the host may use to monitor and configure the operation of the CL48xVCD and CL484CD-G modules.

15.1 Modifying DRAM

Both word and byte addresses are given (words are 16 bits). The word address is used when accessing locations in the CL48x's DRAM or ROM through the host interface; the byte address is used when accessing locations in the CL48x's ROM by modifying the hex file before downloading to the PROM programmer or modifying the data on the PROM programmer itself. (See Chapter 4 for information on reading and writing DRAM contents.)

15.2 Contents of DRAM

Host-accessible parameter locations in VCD and CD-G DRAM are listed below according to module-type installed:

- CL480VCD: All are valid, except SUBCODE_CNTL and CD_TYPE.
- CL484VCD: All are valid.
- CL484CD-G: The following are valid: FREE_SPACE_START and FREE_SPACE_END, CD_TYPE, CMD_ID, the Command FIFO pointers (Table 15-3), and the Configuration Area parameters as so stated within Table 15-7 ("No CD-G" means does not support CD-G).

  Note: To find the ROM address of any of the parameters listed for CD-G, add 0x5000 to the address given for each parameter (for example, CMD_ID will be at ROM address 0x5000 plus 0x70, or 0x5070).

15.2.1 Status Area

The CL48x Status Area (Table 15-1) contains read-only locations.

Table 15-1 Status Area

Parameter (word/byte addr)	Field Name	Field Value	Bit Location	Description
PROC_STATE[1]				Indicates command state (includes CD-G):
(0x60/0xC0)		0xFF		INITIALIZING
		0x00		IDLE [also = value after initialization or Reset()]
		0x09		SLOWMOTION
		0x0a		SCAN
		0x0b		SINGLESTEP
		0x0c		DISPLAYSTILL and DISPLAYROMSTILL
		0x0d		PLAY
		0x0e		PAUSE and DISPLAYSTILLPAUSE
		0x10		FREEZE
		0x14		DUMPDATA
		0x1b		DISCVIEW
		0x1c		REPLAY
		0x1d		DISCVIEWREADY
		0x1e		CDGTEST
		0x23		READSEQHEAD
MRC_ID[1]				Indicates last valid command (no CD-G):
(0x61/0xC2)		0x0000		No previous command. Also = value after init. or Reset()
		0x0005		SetVideoFormat()
		0x0305		SetVideoFormat(format, 0, 0)
		0x0109		SlowMotion()
		0x000a		Scan()
		0x000b		SingleStep()
		0x000c		DisplayStill()
		0x020c		DisplayStill(start1, start2)
		0x030c		DisplayRomStill(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)
		0x031b		DisplayDigest(xOffset, yOffset, skip)
		0x041b		DisplayDigest(xOffset, yOffset, skip, threshold)
		0x061b		DisplayDigest(xOffset, yOffset, skip, threshold, st1, st2)
		0x001c		Replay()
		0x001d		DigestReady()
		0x0222		PitchShift()
		0x0023		ReadSequenceHeader()
MRC_STATUS[1]				Indicates status of most recent command (no CD-G):
(0x62/0xC4)	WORKING_STATUS[2]	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.
	INVALID_SECTOR_COUNT	0x04		Used only by DumpData() command. See DumpData().
	INVALID_BUF_ADDR	0x08		Used only by DumpData() command. See DumpData().
	SECTOR_TYPE_ERROR	0x10		Used only by DumpData() command. See DumpData().
	SECTOR_OUT_OF_RANGE	0x02		Used only by DumpData() command. See DumpData().

[1] Access: read-only by host (CL48x writes).

[2] When no bits are set (all bits are clear) WORKING_STATUS is valid.

Table 15-1: Status Area (Continued)

Parameter (word/byte addr)	Field Name	Bit Location	Description
INT_STATUS[2] (0x63/0xC6)			Indicates which of the named interrupt events caused the interrupt (no CD-G):
	ERR_INT	0	Bitstream data error
	PICV_INT	1	New picture display
	GOPV_INT	2	First I-picture display after `group_start_code`
	SEQV_INT	3	First I-picture display after `sequence_header_code`
	ENDV_INT	4	Last picture display before `sequence_end_code`
	PICD_INT	5	Picture decode complete
	VSYNC_INT	6	$\overline{\text{VSYNC}}$ pulse occurred
	AOR_INT	7	Address out of range
	UND_INT	8	Bitstream buffer underflow error
	ENDC_INT	9	High-priority command execution complete
	RDYS_INT	10	Ready for data during SlowMotion()
	USR_INT	12	User data ready
	ENDP_INT	13	End of Pause(), or SingleStep() / SlowMotion() during buffer-full condition
	ENDD_INT	14	End of DumpData() command
	AEE_INT	15	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 InquireBufferEmptiness(). (No CD-G.)
VIDEO_FIELD[1] (0x66/0xCC)			This field indicates the field currently being output: zero equals the top field, and one equals the bottom field. (No CD-G.)
SE_STATUS[1] (0x67/0xCE)			This field, sector error status, 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 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. (No CD-G.)
LSA_MS[1] (0x68/0xd0)	Minutes Seconds	15:8 7:0	These fields (*last sector address_minutes/seconds* and *_frame/mode*) hold the sector address of the last sector whose data was copied into a bitstream buffer. The *Minutes* and *Frame* fields are stored in the upper byte of the word address, while the *Seconds* and *Mode* fields are stored in the lower byte. Since these fields are continually updated while receiving CD-ROM data, their contents are guaranteed to be valid when the CL48x is in the PAUSE state. These fields are not updated while executing the DumpData() command or while reading a CD-DA bitstream, but they are updated while in the SLOWMOTION, SINGLESTEP, PLAY, REPLAY, DISPLAYSTILL and FREEZE states. (All bytes are BCD coded.) (No CD-G.)
LSA_FM[1] (0x69/0xd2)	Frame Mode	15:8 7:0	
AEE_MS[1] (0x6a/0xd4)	Minutes Seconds	15:8 7:0	These fields 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 CL48x enters the PAUSE state (if enabled with the AUTO_PAUSE parameter). If another auto-pause sector is received before the interrupt is processed or before the CL48x enters the PAUSE state, these fields will contain the address of the last sector received with the auto-pause bit set. (All bytes are BCD coded.) (No CD-G.)
AEE_FM[1] (0x6b/0xd6)	Frame Mode	15:8 7:0	
FREE_SPACE_START[1] (0x6c/0xd8) FREE_SPACE_END[1] (0x6d/0xdA)			These parameters 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 word 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. (Includes CD-G.)

[1] Access: read-only by host (CL48x writes).

[2] Access: read and write by both CL48x and host.

Table 15-1: Status Area (Continued)

Parameter (word/byte addr)	Field Name	Field Value	Bit Location	Description
STREAM_TYPE[1] (0x6e/0xdC)	NOT_KNOWN CD_ROM _MPEG CD-DA	0x00 0x02 0x04		This parameter specifies the type of bitstream being received when the configuration parameter PLAY_MODE is set to CD_ROM_AUTO. The contents will be updated when a different input bitstream has been detected. (No CD-G.)
INT_SOURCE[2] (0x6f/0xdE)				This parameter indicates which source caused the END-P, ERR, AOR 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. (No CD-G.)
	ROM_ERROR		0	An error was detected while decoding the CD-ROM sector.
	MPEG_SYS_ERROR		1	An error was detected in the MPEG-1 system stream.
	AUDIO_ERROR		2	An error was detected in the audio bitstream.
	VIDEO_ERROR		3	An error was detected in the video bitstream.
	ADDRESS_LOW		8	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		9	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		10	Generates END-P interrupt because of CL48x entering PAUSE state.
	ENDP_BUFFER_FULL		11	Generates END-P interrupt because CL48x can't accept more data.
	AUTO_PAUSE_EVENT		13	Interrupt caused when a AUTO-PAUSE CD sector was detected.
	EOR_EVENT		14	Interrupt caused when a END-OF-RECORD CD sector was detected.
	EOF_EVENT		15	Interrupt caused when a END-OF-FILE CD sector was detected.
NUM_DECODED[2] (0x1b7/0x36E)				This field is the count of the number of decoded pictures, and may be written from the host during the IDLE or PAUSE states.The value is incremented at completion of picture decoding at the same time an END-D interrupt is generated, if 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, SLOWMOTION, SINGLESTEP, and SCAN states. The count will be reset to zero by a Reset() command. (No CD-G.)
NUM_SKIPPED[2] (0x1b8/0x370)				This field is the count of the number of pictures skipped due to audio/video synchronization, and may be written from the host during the IDLE or PAUSE states. 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 and REPLAY processing states, and is set to zero by a Reset() command. (No CD-G.)
NUM_REPEATED[2] (0x1b9/0x372)				This parameter is the count of the number of pictures repeated due to A/V synchronization, and may be written from the host during the IDLE or PAUSE states. 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 and REPLAY states, and is set to 0 by a Reset() command. (No CD-G.)

[1] Access: read-only by host (CL48x writes).

[2] Access: read and write by both CL48x and host.

Table 15-1: Status Area (Continued)

Parameter (word/byte addr)	Field Name	Field Value	When Effective	Description
INTERVAL[1] (0x1ba/0x374)				This field is the time interval between successive picture frames, and is updated when the input bitstream's picture rate changes. The units are the number of 90K system time clocks between successive picture frames. (No CD-G.)
CD_TYPE[1] (0x1bd/0x37a)				Available in the CL484 microcode only, this configuration parameter contains the results of the previous CDGTest() command. (Includes CD-G.)
	INDETERMINATE	0x00		
	CD-DA_VIDEO	0x06		
	CD-G	0x10		
VBS_SADR[1] (0x1c0/0x380)			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. For example, if VBS_SADR contains 3C and VBS_EADR contains 6E, the first word of the buffer is at 3C00, and the last word is at 6DFF. (No CD-G.)
VBS_EADR[1] (0x1c1/0x382)			At Decode	
ABS_SADR[1] (0x1c2/0x384)			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. (No CD-G.)
ABS_EADR[1] (0x1c3/0x386)			At Decode	
AUDIO_SADR[1] (0x1c4/0x388)			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. (No CD-G.)
AUDIO_EADR[1] (0x1c5/0x38A)			At Decode	

[1] Access: read-only by host (CL48x writes)

15.2.2 High-Priority Command Area

The high-priority command area (see Table 15-2) is the memory area where high-priority commands are written (includes CD-G). To write a high-priority command into this memory area, first check that the CMD_ID location in DRAM is zero, indicating the previous high-priority command has been processed. Once the CMD_ID location in DRAM is zero, write the arguments first. Then, write the command ID to the CMD_ID location. The high-priority command processor uses the CMD_ID field to determine if a command is in the high-priority command area.

Table 15-2 High-Priority Command Area

Parameter	Access	Word/byte Addr
CMD_ID	Read and write by host and CL48x	0x70/0xE0
ARGUMENT1	Read-only by CL48x, and read and write by host	0x71/0xE2
ARGUMENT2	Read-only by CL48x, and read and write by host	0x72/0xE4
ARGUMENT3	Read-only by CL48x, and read and write by host	0x73/0xE6
ARGUMENT4	Read-only by CL48x, and read and write by host	0x74/0xE8
ARGUMENT5	Read-only by CL48x, and read and write by host	0x75/0xEA
ARGUMENT6	Read-only by CL48x, and read and write by host	0x76/0xEC
ARGUMENT7	Read-only by CL48x, and read and write by host	0x77/0xEE

15.2.3 FIFO Pointers

The CL48x FIFO pointers contain field locations for the following:

- The Command FIFO [see Table 15-3 (includes CD-G)]:
  - 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 (does not include CD-G)]:
  - 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 CL48x execution. See Section 13.2.1 for further details on how to use the Command FIFO. See Section 13.2.2 for a discussion of low-priority commands. (Includes CD-G.)
	CMDF_SADDR[1] CMDF_EADDR[1]	0x78/0xF0 0x79/0xF2	These two locations contain the starting and ending addresses of the low-priority 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_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 CL48x to fetch commands from the Command FIFO, while the host uses the write pointer to place commands into the Command FIFO. The host processor must not modify the contents of the read pointer.
			The host processor updates 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.
			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 CL48x microcode writes user_data found in MPEG-1 picture headers (see VideoCD 2.0 spec for more info). The user data FIFO functions almost exactly the same as the Command FIFO. It 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. Note: user data will be overwritten if the host doesn't read it out fast enough. See the USR interrupt in Chapter 14. (No CD-G.)
	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[2] UDF_WRITE[1]	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 updated by the CL48x when data is written into the user data FIFO, while the host processor updates the read pointer when data is read from the user data FIFO. The host processor must 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. The CL48x asserts the USR interrupt when it finishes writing a new picture's worth of user data into the FIFO. If the host doesn't read the data out of the FIFO before it overflows, the FIFO contents will be overwritten. 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 CD-ROM Header and Subheader

CD-ROM header and subheader parameters are shown in Table 15-5.

Table 15-5 CD-ROM Header and Subheader

Parameter (word/byte addr)	Field Name	Bit Location	Access	Definition
M_SEC[1] (0x198/0x330)	Minutes	15:8	Read-only by host	These fields contain the CD-ROM header and sub-header information for the last sector read by the CL48x input process. These values are updated before any checks are made for the proper sector type. Since these values are continuously updated while receiving CD-ROM data, the contents are guaranteed to be valid when the CL48x is in the PAUSE state. These fields are not updated while executing the DumpData() command or while reading a CD-DA bitstream, but they are updated while in the PLAY, REPLAY, DISPLAYSTILL, SLOW-MOTION, SINGLESTEP and FREEZE states. All bytes are BCD coded. (No CD-G.)
	Seconds	7:0	Read-only by host	
F_MODE[1] (0x199/0x332	Frame	15:8	Read-only by host	
	Mode	7:0	Read-only by host	
FC_NUM[1] (0x19a/0x334)	File_number	15:8	Read-only by host	
	Channel_number	7:0	Read-only by host	
SMC_INFO[1] (0x19b/0x336)	Submode_info	15:8	Read-only by host	
	Coding_info	7:0	Read-only by host	

[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. They are updated when read from the rate buffer.

Note: MPEG-1 header parameters do not apply to CD-G.

Table 15-6 MPEG-1 Header Parameters

Parameter Type	Parameter	Word/byte Address	Definition
Sequence header[2]	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
	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	Bit Location	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	15	protection_bit
			14:13	layer
			12	ID
			11:0	syncword
	AUDIO_HEADER2[1]	0x1bc/0x378	15:12	bitrate_index
			11:10	sampling _frequency
			9	padding_bit
			8	private_bit
			7:6	mode
			5:4	mode_extension
			3	copyright
			2	original/home
			1:0	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

CL48x Configuration Area parameters are shown in Table 15-7. These DRAM locations contain parameters which can be modified to change the behavior of the CL48x. Changes to different configuration parameters take effect at the five different times shown below:

- *Start*: The value is examined after the microcode is loaded and before the processing state first transitions from initialization to VCDIDLE. These types of parameters must be changed in ROM.
- *Decode:* The value is effective when MPEG-1 decoding is initiated by the CL48x microcode executing a Play() or DisplayStill() command.
- *Video Initialization*: The value is effective 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* and SetVideoFormat() execution.
- *Immed:* Immediately. Parameters specified as being *Immed.* will take effect the next time the CL48x performs the related operations.
- *Update_G:* The value is effective when the UPDATE_GRAPHICS parameter has been set.

Note: All parameters in Table 15-7 are readable and writable by the host but are read-only by the CL48x (except UPDATE_GRAPHICS). All modifications to these parameters should be read-modify-writes.

Table 15-7 Configuration Parameters

Parameter (word/byte addr) (default value)	When Effective	Field Name	Bit Number	Definition
UPDATE_GRAPHICS (0x180/0x300) (0x0000)	VSYNC			When set equal to 1, this configuration parameter instructs the microcode to examine the contents of the TRANSPARENT, Y01, U01, V01, Y23, U23 and V23 configuration parameters. This parameter is examined when the VSYNC signal is active. The value of UPDATE_GRAPHICS will be set to zero after the change of the listed parameters has been effected. (Includes CD-G.)

Table 15-7 Configuration Area Parameters (Continued)

Parameter (word/byte addr) (default value)	When Effective	Field Name	Bit Location	Definition
TRANSPARENT (0x181/0x302) (0x0000)	Update_G			This configuration parameter specifies which color numbers in the graphic bitstream will be displayed as transparent. The new values in this parameter take effect when UPDATE_GRAPHICS is set. Any combination of these bits may be set, but they only affect compressed graphics images. (Includes CD-G.)
		TRANSPARENT7	7	Indicates color number seven should be displayed as transparent
		TRANSPARENT6	6	Indicates color number six should be displayed as transparent
		TRANSPARENT5	5	Indicates color number five should be displayed as transparent
		TRANSPARENT4	4	Indicates color number four should be displayed as transparent
		TRANSPARENT3	3	Indicates color number three should be displayed as transparent
		TRANSPARENT2	2	Indicates color number two should be displayed as transparent
		TRANSPARENT1	1	Indicates color number one should be displayed as transparent
		TRANSPARENT0	0	Indicates color number zero should be displayed as transparent
Y01 (0x182/0x304) (0x0010)	Update_G	Y0 Y1	15:8 7:0	These configuration parameters define the Y, U and V values for color pallete locations 0 and 1. The color 0 parameter is in the most significant byte, and the color 1 parameter is in the least significant byte, of each parameter. Both colors 0 and 1 are available in bit-map graphics mode (see Chapter 11), while only color 1 is available in compressed graphics mode (see Chapter 11). The color pallete values are updated when the UPDATE_GRAPHICS configuration parameter is set. (Includes CD-G.)
U01 (0x183/0x306) (0x0010)	Update_G	U0 U1	15:8 7:0	
V01 (0x184/0x308) (0x0010)	Update_G	V0 V1	15:8 7:0	
Y23 (0x185/0x30a) (0x0010)	Update_G	Y2 Y3	15:8 7:0	These configuration parameters define the Y, U and V values for color pallette locations 2 and 3. The color 2 parameter is in the most significant byte, and the color 3 parameter is in the least significant byte, of each parameter listed above. Both colors 2 and 3 are available in bit-map graphics mode (see Chapter11), while only color 3 is available in compressed graphics mode (see Chapter 11). The color pallete values are updated when the UPDATE_GRAPHICS configuration parameter is set. (Includes CD-G.)
U23 (0x186/0x30c) (0x0010)	Update_G	U2 U3	15:8 7:0	
V23 (0x187/0x30e) (0x0010)	Update_G	V2 V3	15:8 7:0	
FIRST_FIELD (0x1c6/0x38C) (0x0000)	Video initialization		0	In the field-type determination algorithm (slave mode only), this parameter indicates from the VSYNC and HSYNC timing relationship whether the top or bottom field is displayed: a 0 will maintain the default value; a 1 will invert the default value. See Section 7.5.4 for more information. (Includes CD-G.)
UNUSED_IO (0x1ca/0x394) (0x0001)	Startup		0	This parameter specifies whether the unused programmable I/O pins will be inputs or outputs. If this parameter is set to 1, the unused I/O pins will be configured as outputs and driven to a high logic level; if set to 0, the usused I/O pins will be configured as inputs. The programmable I/O pins are as follows: pin 64 (RESERVED), pin 114 (CDG-S0S1), pin 117 (CDG-VFSY), pin 120 (CDG-SDATA), pin 124 (FSC1), and pin 126 (FSC4). Since the default configuration drives these pins high, a 4.7K pulldown resistor on each (except the output pin, 124) would be appropriate if they are not being used for another purpose. (No CD-G.)

Table 15-7 Configuration Area Parameters (Continued)

Parameter (word/byte addr) (default value)	When Effective	Field Name	Bit Location	Definition
AUDIO_CONFIG (0x1cb/0x396) (0xd13e)	Immediate			This parameter configures how the CL48x hardware outputs audio data, and is examined whenever a new audio header is decoded. (Includes CD-G.)
		STC_DIVISOR	15:4	The field must be set to 0xD100 when the XCK frequency is 16.93 MHz, and must be set to 0xE0C0 when the XCK frequency is 11.29 MHz.
		LEFT_LRCK_HI	3	Used to route channel 0 or 1 data to left or right output. See Table 8-2.
		RIGHT_LRCK_LO	2	Same as LEFT_LRCK_HI. See Table 8-2.
		OUTPUT_24_BITS	1	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 output 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	0	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) (0x0220)	At Decode			This parameter is used to configure how the CL48x hardware reads the input CD data. (Includes CD-G.)
		LENGTH	5:4	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) (I^{2}S)
		DATA_LSB_FIRST	3	When 1, the LSB of the CD data is input first; otherwise, the MSB is input first.
		C2PO_LSB_FIRST	2	When 1, the LSB of CD-C2PO is input first; otherwise, the MSB is input first.
		LRCK_RCH	1	When 1, *right* channel CD-DATA is input when LRCK is HIGH; otherwise, the left channel CD-DATA is input.
		BCK_FALLING	0	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/byte addr) (default value)	When Effective	Field Name	Bit Loc.	Definition
VIDEO_MODE2 (0x1cd/0x39A) (0x1008)	Video Init.			Contains the video unit's $\overline{\text{VSYNC}}$/CSYNC setting. (Includes CD-G.)
		NON_INT_CDG	13	When 1 (in CD-G mode when bit 3 = 0 only), outputs non-interlaced video with 263 lines (NTSC) and 313 lines (PAL) between successive VSYNCs. When 0, the values will be 262 and 312, respectively.
		HSYNC_CSYNC	12	When 1, outputs $\overline{\text{HSYNC}}$ and CSYNC.
		RGB_15_EN	6	When 1, bit 5 is enabled.
		RGB_15	5	When 1, the video output uses five bits each for red, blue, and green color signals.
		INTERLACE	3	When 1 (default), outputs interlaced video with 262.5 lines (NTSC) and 312.5 lines (PAL) between successive VSYNCs. When 0, outputs non-interlaced video with 262 lines (NTSC) and 312 lines (PAL).
		CSYNC_VSYNC	2	This field is used to determine whether the CSYNC or $\overline{\text{VSYNC}}$ signal is output to the $\overline{\text{VSYNC}}$ pin. When 1, CSYNC is output. CSYNC can only be an output, never an input.
ROM_CONFIG (0x1ce/0x39C) (0x0007)	At Start			This parameter specifies the number of GCKs to use when accessing the attached ROM. Bits 5 and 6 of this parameter must be set to 0. (No CD-G.)
		ROM_ACCESS	4:0	Specifies ROM access time in GCK cycles: This time = (ROM_ACCESS + 1) x GCK period. At hardware reset, the ROM access time is set to 800ns. The access time of the ROM being used in the system (typically 150ns) is read from this ROM location by the initialization microcode and written to the CL48x's memory controller.
DRAM_REFRESH (0x1cf/0x39E) (0x200)	At Start			Specifies the number of GCKs between successive refresh cycles. (No CD-G.)
		COUNT	12:0	This is the number of GCK's between refreshes. This value is used to initialize the refresh counter when the CL48x 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 with a 40 MHz GCK (512/40,000,000).
AUDIO_AVERAGE (0x1d0/0x3A0) (0x0000)	Immed.			This parameter controls audio averaging and is examined whenever an MPEG audio frame is decoded; it is not valid for CD-DA bitstreams. (No CD-G.)
		AVERAGE	0	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.
SUBCODE_CNTL (0x1d1/0x3a2) (0x001A)				This parameter (CL484 , both VCD and CD-G) sets up the subcode interface and is examined when a CDGPlay() or CDGTest() command is executed.
		SCLK_INPUT	4	Specifies direction of the SCLK signal. If non-zero, SCLK is output by the CL484; if zero, SCLK is input to the CL484.
		VFSY_SELECT	3	Selects timing on S0S1 and VFSY as composite or separate. If 0, the timing is composite, and any signal on the S0S1 pin is ignored. If non-zero, the timing is expected to be separate.
		SCLK_BCKS	2:0	Specifies the number of BCK clocks per SCLK when SCLK is output: ■ 1 = Sets the divisor of the BCK signal to 16 to generate SCLK ■ 2 = Sets the divisor of the BCK signal to 24 to generate SCLK ■ 4 = Sets the divisor of the BCK signal to 32 to generate SCLK

Table 15-7 Configuration Area Parameters (Continued)

<table>
<tr><th>Parameter
(word/byte addr)
(default value)</th><th>When Effective</th><th>Field Name</th><th>Bit Location</th><th>Description</th></tr>
<tr><td>WEIGHTK0
(0x1d2/0x3A2)
(0x0198)
WEIGHTK1
(0x1d3/0x3A4)
(0x0730)
WEIGHTK2
(0x1d4/0x3A6)
(0x079c)
WEIGHTK3
(0x1d5/0x3A8)
(0x0204)
WEIGHTK4
(0x1d6/0x3AA)
(0x012a)</td><td>Video Initialization</td><td></td><td></td><td>These parameters (available in CD-G) 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, K4 is WEIGHTK4, and N is 16 or 0, depending on the value of the VRANGE field of VIDEO_MODE:
<table>
<tr><th>VRANGE</th><th>Range</th><th>K0</th><th>K1</th><th>K2</th><th>K3</th><th>K4</th></tr>
<tr><td>1</td><td>16 - 235</td><td>0x0167</td><td>0x0749</td><td>0x07a8</td><td>0x01c6</td><td>0x0100</td></tr>
<tr><td>0</td><td>0 - 255</td><td>0x0198</td><td>0x0730</td><td>0x079c</td><td>0x0204</td><td>0x012a</td></tr>
</table>
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</td></tr>
<tr><td>VIDEO_MODE
(0x1d7/0x3AE)
(0x0b25)</td><td>At Start</td><td></td><td></td><td>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. (CD-G available.)</td></tr>
<tr><td></td><td></td><td>VRANGE</td><td>8</td><td>Specifies whether 16 is subtracted from the luminance in converting from YUV to RGB: 1 = subtract 16 (default); 0 = no-adjust (see WEIGHT0-4 descriptions).</td></tr>
<tr><td></td><td></td><td>VCLK_OUT</td><td>7</td><td>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 (default).</td></tr>
<tr><td></td><td></td><td>VBUS_MODE</td><td>6</td><td>Only valid in YUV operation. Specifies the video bus width: 1 = the video bus width is 16 bits; 0 = 8 bits (default).</td></tr>
<tr><td></td><td></td><td>H_INTERP</td><td>4</td><td>When set to 1, horizontal interpolation is disabled. If set to 0, horizontal interpolation is enabled (default).</td></tr>
<tr><td></td><td></td><td>RGB_MODE</td><td>2</td><td>Specifies whether the video unit should be in RGB or YUV mode: 1 = the video unit outputs RGB mode (default); 0 = YCbCr.</td></tr>
<tr><td></td><td></td><td>SYNC_OUT</td><td>0</td><td>Specifies the direction of the $\overline{\text{VSYNC}}$ and $\overline{\text{HSYNC}}$ signals: 1 = output (default), in which both signals are generated by the video unit; 0 = both signals are input from external source.</td></tr>
</table>

Table 15-7 Configuration Area Parameters (Continued)

Parameter (word/byte addr) (default value)	When Effective	Field Name	Bit Loc.	Description
VSYNC_BOT_TP[1] (0x1d8/0x3b0) (0x0180)	Video Initialization			This parameter specifies the position relative to HSYNC going LOW that VSYNC will go LOW for the bottom field. This parameter is used for both PAL and NTSC. Its default is 0x180 (384) pixel clocks. (Includes CD-G.)
VSYNC_TOP_TP[1] (0x1d9/0x3b2) (0x0000)	Video Initialization			This parameter specifies the position relative to HSYNC going LOW that VSYNC will go LOW for the top field. This parameter is used for both PAL and NTSC. (Includes CD-G.)
COLOR_CR[1] (0x1da/0x3b4) (0x0080)	Video Initialization	CR_VALUE	7:0	This parameter specifies the Cr chrominance value of the YCbCr color of the border color using the least significant eight bits. (Includes CD-G.)
COLOR_Y_CB[1] (0x1db/0x3b6) (0x1080)	Video Initialization	Y_VALUE CB_VALUE	15:8 7:0	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. (Includes CD-G.)
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 CL48x is generating HSYNC and VSYNC. (Includes CD-G.)
NTSC_HSYNC_PER[1] (0x1dd/0x3bA) (0x0359)	Video Initialization			In NTSC mode, this parameter specifies the time (in VCK periods) between HSYNC being active when the CL48x is generating HSYNC and VSYNC. (Includes CD-G.)
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 CL48x is generating HSYNC and VSYNC. (Includes CD-G.)
PAL_HSYNC_PER[1] (0x1df/0x3bE) ((0x035f)	Video Initialization			In PAL mode, this parameter specifies the time (in VCK periods) between HSYNC between active when the CL48x is generating HSYNC and VSYNC. (Includes CD-G.)
PLAY_MODE[1] (0x1e0/0x3C0) (0x0000)	At Decode			Specifies bitstream type to expect. (No CD-G.)
		CD_ROM_AUTO	0	Automatically switches between CD-ROM video and CD-DA.
		CD_ROM_MPEG	2	All data is interpreted as CD-ROM video bitstreams.
		CD_ROM_CDDA	4	All data is interpreted as CD-DA bitstreams.
STILL_MODE[1] (0x1e1/0x3C2) (0x0001)	At Decode			Causes the video to be blanked before displaying the next still picture, and is only valid for still pictures. If 0, video blanking is performed. (No CD-G.)
AUTO_PAUSE[1] (0x1e2/0x3C4) (0x0000)	Immed.			Specifies if auto-pause feature is enabled or disabled. If 1, auto-pause is enabled. (No CD-G.)

[1] Access: Write-only (CL48x will read only during initialization).

Table 15-7 Configuration Area Parameters (Continued)

Parameter (word/byte addr) (default value)	When Effective	Field Name	Field Value	Bit Location	Description
CDDA_EMPHASIS[1] (0x1e3/0x3C6) (0x0000)	Immed.				This parameter is examined when a CD-DA bitstream is input (at the same time the CD-DA_EMP signal is examined). See Table 8-3 for more info. (Includes CD-G.)
		EMP_FOLLOWS_INPUT		1	See Table 8-3.
		SET_EMP_ON		0	See Table 8-3.
ERROR_LEVEL[1] (0x1e5/0x3CA) (0x0001)	Immed.			0	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 CL48x 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 Chapter 11.) (No CD-G.)
INT_MASK [1] (0x1e6/0x3CC) (0x0000)	Immed.	ERR_INT PICV_INT GOPV_INT SEQV_INT ENDV_INT PICD_INT VSYNC_INT AOR_INT UND_INT ENDC_INT RDYS_INT USR_INT ENDP_INT ENDD_INT AEE_INT		0 1 2 3 4 5 6 7 8 9 10 12 13 14 15	Specifies the interrupts to be generated by the CL48x to the host processor. A non-zero value in any bit position enables generation of the corresponding interrupt. See Chapter 4 for descriptions of each interrupt. (No CD-G.)
AUDIO_MUTE (0x1e7/0x3CE) (0x0000)	Immed.				This parameter specifies the attenuation settings used for audio output: Its value is examined each time an audio frame is decoded. The range of allowable values is 0-63, with 0 indicating no attenuation. Note that the attenuation is exponential, not linear, with 2db of attenuation per increment. (No CD-G.)

[1] Access: Write-only (CL48x will write only during initialization).

Table 15-7 Configuration Area Parameters (Continued)

Parameter (word/byte addr) (default value)	When Effective	Field Name	Field Value	Bit Location	Description
VA_STREAM_ID (0x1e8/0x3d0) (0xe0c0)	Startup	VIDEO_STREAM_ID AUDIO_STREAM_ID		15:8 7:0	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. (No CD-G.)
TOP_INTERPCOEF (0x1e9/0x3d2) (0x0225)	Video Init.	VLUM_BOT_LINE VLUM_TOP_LINE VCHR_BOT_LINE VCHR_TOP_LINE HLUM_LINE HCHR_LINE		1:0 3:2 5:4 7:6 9:8 11:10	These parameters define the vertical and horizontal interpolation coefficients for the video output process used for the *top* video field. (No CD-G.)
BOT_INTERPCOEF (0x1ea/0x3d4) (0x022f)	Video Init.	VLUM_BOT_LINE VLUM_TOP_LINE VCHR_BOT_LINE VCHR_TOP_LINE HLUM_LINE HCHR_LINE		1:0 3:2 5:4 7:6 9:8 11:10	These parameters define the vertical and horizontal interpolation coefficients for the video output process used for the *bottom* video field. (No CD-G)) 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 7.5.8 for more details on how interpolation is performed.
VIDEO_FORMAT (0x1eb/0x3d6) (0x0004)	Video Init.	MULTI_SYNC[1] PAL_FORMAT NTSC_FORMAT PAL_FORMAT1	0x02 0x03 0x04 0x05		This parameter specifies the initial video output mode. MULTI_SYNC mode makes the output video format follow the input video format. PAL_FORMAT sets the output mode to PAL and also interpolates (vertically scales) any incoming NTSC pictures. NTSC_FORMAT sets the output mode to NTSC, allowing use of TOP_BORDER to determine what portion of a PAL picture is displayed. PAL_FORMAT1 sets the output mode to PAL, but does not interpolate NTSC. (Includes CD-G.)

[1] Note: MULTI_SYNC is not present in CD-G.

Table 15-7 Configuration Area Parameters (Continued)

Parameter (word/byte addr) (default value)	When Effective	Description
LEFT_BORDER (0x1ed/0x3dA) (0x0082)	Video Init.	After the display is centered horizontally, this parameter specifies the number of VCK's from the centered position to the start of video output. The following equation is used to calculate the horizontal video start position: $HorizStartPosition = LEFT\_BORDER + \frac{(VideoDisplayWidth - VideoSourceWidth)}{2}$ The HorizStartPosition is counted from the leading edge of the $\overline{\text{HSYNC}}$ pulse. LEFT_BORDER has a minimum value of 0x70, a maximum value of 0x98, and a default value of 0x82 (LEFT_BORDER must be set to a positive value). The default value is 704 for both VideoDisplayWidth and VideoSourceWidth (VideoDisplayWidth is set from the VIDEO_FORMAT parameter when the SetVideoFormat() command is executed). Note: For CD-G use, see the description of this parameter given in the SetVideoFormat(**format, 0, 0**) command in Chapter 13.
TOP_BORDER (0x1ee/3dC) (0x0001)	Video Init.	After the display is centered vertically, this parameter specifies the number of lines from the centered position to start video output. The following equation is used to calculate the vertical video start position: $VertStartPosition = TOP\_BORDER + \frac{VideoDisplayHeight - VideoSourceHeight}{2}$ The TOP_BORDER parameter default and minimum value is 1. The default value is 240 for NTSC (288 for PAL) for both VideoDisplayHeight and VideoSourceHeight. Note: For CD-G use, see the description of this parameter given in the SetVideoFormat(**format, 0, 0**) command in Chapter 13.
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. These parameters are readable and writable by the host but are read-only by the CL48x. (No CD-G.)
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/vvvv. For example, version 1.10 released on Jan. 1, 1993 would appear in ASCII as '01Jan93/1.10'. (Includes CD-G.)

Table 15-7 Configuration Area Parameters (Continued)

Parameter (word/byte addr) (default value)	When Effective	Field Name	Field Value	Bit Location	Description
SUBCODE_CHANNEL (0x1e8/0x3d0) (0x0003)	Immediate (CD-G)	CHANNEL15		15	This parameter (CD-G only) specifies the subcode channels from which CD-G data should be read. Each channel is assigned a bit within this parameter and the bits are defined as shown. Each channel, with its corresponding bit set, is read. All other channels are ignored. This parameter has the same function as the CDGSetChannel() macro command. Use this parameter instead of CDGSetChannel() when possible.
		CHANNEL14		14	
		CHANNEL13		13	
		CHANNEL12		12	
		CHANNEL11		11	
		CHANNEL10		10	
		CHANNEL9		9	
		CHANNEL8		8	
		CHANNEL7		7	
		CHANNEL6		6	
		CHANNEL5		5	
		CHANNEL4		4	
		CHANNEL3		3	
		CHANNEL2		2	
		CHANNEL1		1	
		CHANNEL0		0	

Appendix A CL48x Crystal Design Guidelines

This appendix contains guidelines for choosing a crystal for the CL48x.

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

In general, use a third-order harmonics frequency crystal. Two design examples are shown below.

> *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.*

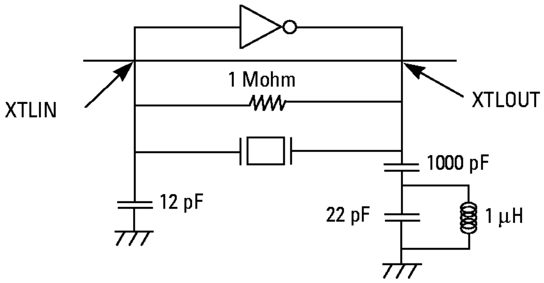


Figure A-1 Crystal Design Example Number 1

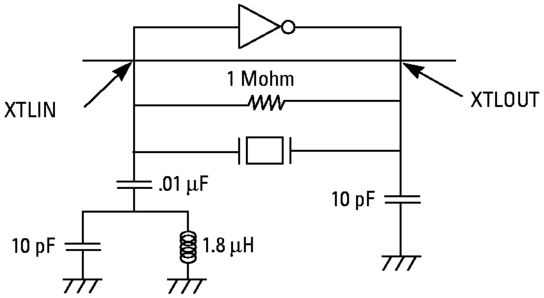


Figure A-2 Crystal Design Example Number 2

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
OSD Data Structure

This appendix describes the data structure of bitmap OSD. Users should generate all bitmap OSD files according to this structure.[1]

```
while (next bit is zero){
    zero bit (=0)                                                       1 bit
    numblank (# of transparent field lines above overlay)              15 bits
    numline1 (# of lines in field with common colors)                  16 bits
    sif resolution (indicates if overlay is SIF or CCIR 601)            1 bit
    screen start position (pixels in 601 resolution)                   10 bits
    no color palette                                                    1 bit
    zero bits (=0)                                                      3 bits
    one bit (=1)                                                        1 bit
    if (no color palette  == 0)
      { Y0 graphics color                                               8 bits
        Y1 graphics color                                               8 bits
        U0 graphics color                                               8 bits
        U1 graphics color                                               8 bits
        V0 graphics color                                               8 bits
        V1 graphics color                                               8 bits
```

(Continued)

[1] For additional information, refer to the application note entitled "Using CL484 Bitmap OSD Function," available from a C-Cube representative.

```
        Y2 graphics color                                   8 bits
        Y3 graphics color                                   8 bits
        U2 graphics color                                   8 bits
        U3 graphics color                                   8 bits
        V2 graphics color                                   8 bits
        V3 graphics color                                   8 bits
      }
    if (SIF resolution)
        number of lines = numline1;
    else
        number of lines = numline1 * 2;
    numwords (number of 16-bit words per line)             16 bits
    for (i=0; i<number of lines; i++)
        for (j=0; j<numwords; j++)
            for (k=0; k<8; k++)
                bitmap pixel;                               2 bits
}
0x8000                                                     16 bits
```

The text color may be modified by altering the values shown in Table B-1.

Table B-1 YCbCr Values

Color	Y	Cb	Cr
Black	10	80	80
Blue	29	F0	6E
Red	52	5A	F0
Violet	6A	CA	DE
Green	91	36	22
Cyan	A9	A6	10
Yellow	D2	10	92
White	EA	80	80