INTEGRATED JPEG CODEC

FEATURES

- Single-chip JPEG processor that integrates all the modules needed for JPEG encoding and decoding:
  - Raster-to-block and block-to-raster converter
  - Strip buffer
  - JPEG codec
- Motion video compression and expansion capability:
  - Up to 25 frames/sec, square pixel and CCIR PAL
  - Up to 30 frames/sec, square pixel and CCIR NTSC
- Three modes of Bit Rate Control (BRC):
  - Auto Two Pass: for still image compression, produces tightly controlled compressed code volume
  - Single pass: for motion video compression, keeps the file size approximately fixed
  - No BRC: uses fixed quantization tables
- Glueless interface to common video decoders (e.g., Philips, Brooktree, Samsung, ITT, Harris)
- Supports 8- and 16-bit YUV video interfaces
- Supports master and slave modes of video synchronization
- Interfaces to a variety of host controllers, ranging from the dedicated high-performance ZR36057 PCI controller to generic low-cost microcontrollers
- Two clock speed grades available:
  - ZR36060-27, 27 MHz, for CCIR video format
  - ZR36060-29.5, 29.5 MHz, for PAL and NTSC square pixel video formats
- Flexible compressed data interface:
  - 8-bit master mode, supporting up to 29.5 Mbytes/sec*
  - 16-bit slave mode, supporting up to 16.8 Mbytes/sec*
  - 8-bit slave mode, supporting up to 9.8 Mbytes/sec*

  (* peak data rates, with 29.5 MHz clock)
- On-chip video processing, including:
  - Mixing of two video sources
  - Horizontal (1:2 and 1:4) and vertical (1:2) up and down scaling
  - Cropping in compression and programmable background color in decompression
- 3.3V power supply with 5V-tolerant I/O
- Low power consumption:
  - 675 mW (typical) at 27 MHz operating frequency
  - 725 mW (typical) at 29.5 MHz operating frequency
  - Power down mode for power saving
- 100-pin PQFP package

APPLICATIONS

- Desktop video editing subsystems
- PCMCIA video capture cards
- Digital still cameras
- Digital video recording
- JPEG-based video conferencing systems

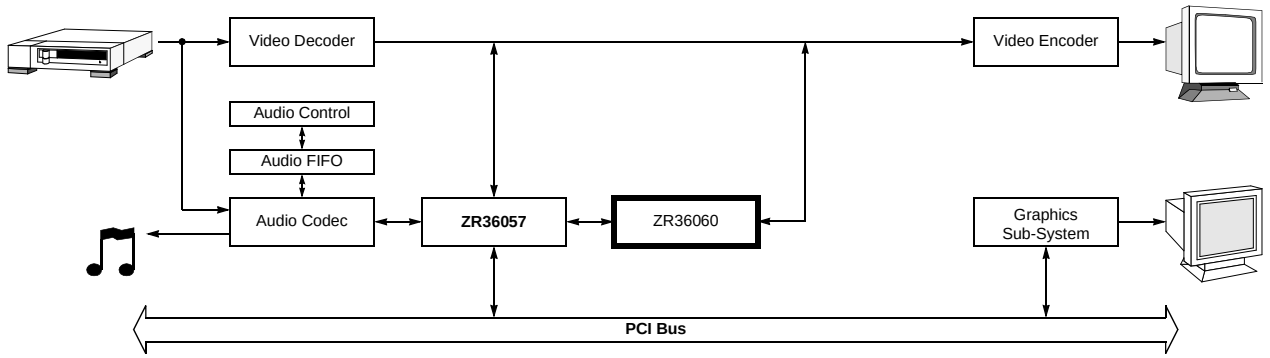


Figure 1. JPEG-Based Video Editing Subsystem For PCI Systems

Contents

1.0 INTRODUCTION

1.1 The ZR36060

The ZR36060 is an integrated JPEG codec targeted to video capture and editing applications in desktop and laptop computers. Figure 1 shows an example of a typical application, a video editing subsystem for PCI bus computers.

The ZR36060 integrates the functionality of a JPEG codec such as the ZR36050, a raster-to-block converter such as the ZR36015, as well as the strip buffer SRAM for the raster-to-block converter and additional functions. It is based on the field proven, fully compliant Zoran JPEG device technology, and incorporates Zoran's patented bit rate control mechanism.

In compression, the ZR36060 accepts YUV 4:2:2 digital video, performs optional cropping and decimation, and encodes it into a JPEG baseline compressed bitstream, which it outputs to a host controller. In decompression, it receives the bitstream from the host controller, decodes it back to YUV 4:2:2 format digital video, up-scales it if required, and outputs the video to a composite video encoder or other destination.

The ZR36060 incorporates hardware support for multiplexing two video sources (in rectangular windows) in compression, or the reconstructed video with another source in decompression. It can operate as a video sync master or slave, with 8-bit or 16-bit video bus widths. A pixel flow control mechanism is provided for convenient implementation of non-real-time video rates, such as for still picture compression.

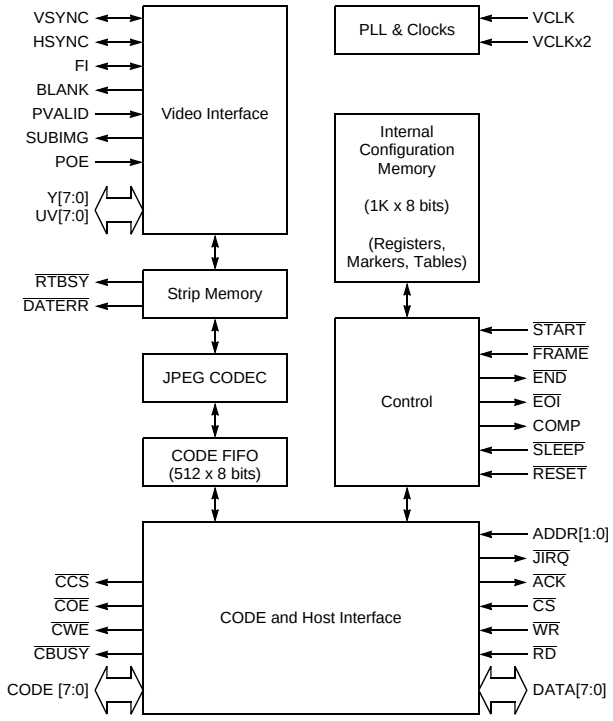


Figure 2. ZR36060 Block Diagram

The code interface of the ZR36060 can operate in 8-bit master, 8-bit slave or 16-bit slave modes. In slave mode, code transfer shares the host interface, which is generic enough to be able to interface gluelessly with a variety of host controllers, ranging from the dedicated, high performance ZR36057 to common microcontrollers.

The ZR36060 is a CMOS device, requiring a 3.3 Volt power supply. Its inputs and outputs are 5 Volt tolerant. A power-down ("sleep") mode reduces current consumption to a very low level, while preserving the logic state of the device.

A block diagram of the ZR36060 is shown in Figure 2.

1.2 The ZR36060 and the JPEG Standard

The JPEG standard, ISO/IEC 10918-1, defines a whole range of options for compressing continuous-tone images - a baseline lossy compression process, extended lossy processes, lossless compression, and hierarchical compression methods. The ZR36060 implements the baseline process.

Even the baseline method is defined by the JPEG standard to provide maximal flexibility in choosing the color space in which an image is compressed - an image can have an almost unlimited number of color components, and these can be compressed in a single scan, or in multiple scans. Because its main targeted application is motion color video compression and decompression, the architecture of the ZR36060 supports one particular subset: Since the ZR36060 supports only the YUV 4:2:2 pixel format, it supports three color components, in a single interleaved scan.

1.2.1 JPEG baseline overview

The JPEG baseline compression method is based on the discrete cosine transform or DCT. The DCT is performed on 8x8 blocks of samples, of each color component, resulting in a set of 64 DCT coefficients for each block. Thus, in order for a normal raster-scanned image to be compressed, it must first be converted to block format This requires that an 8-line strip of the image (for YUV 4:2:2, containing 8 lines of each color component) be stored in a strip buffer, so that the samples can be re-ordered (see Figure 2).

For subsequent stages of the compression, the 64 DCT coefficients of each block are further re-ordered by scanning the block in a zig-zag sequence. Each of the 64 coefficients is quantized using the appropriate value from a 64-entry quantization table. In the ZR36060, it is possible to define three different quantization tables, one per color component; generally, however, two tables are sufficient, one for the luminance component and one for the chrominance components.

The quantized DCT coefficients are passed to a Huffman encoder, for the final stage of the process. The Huffman coding is performed separately for the DC coefficient of each block (the

first coefficient of the block), and the remaining 63 AC coefficients. The encoding methods used for DC and AC coefficients differ in their details, and this requires two Huffman tables to be specified, one for DC and one for AC. And since the statistics of the luminance and chrominance components are generally quite different, separate Huffman tables are required for luminance and chrominance, for a total of four tables, two DC and two AC. The ZR36060 supports this configuration.

Baseline decompression essentially consists of the inverses of each of the stages used in compression, in reverse order: Huffman decoding, dequantization, inverse DCT, and conversion of the blocks back to raster order using the strip buffer.

1.2.1.1 The Minimum Coded Unit

If the compressed image data is interleaved, as is the case in the ZR36060, the compression is performed in units of a Minimum Coded Unit, or MCU, which contains one or more blocks of each color component. For the 4:2:2 pixel format used by the ZR36060, where the chrominance (U and V) components are decimated by 2:1 horizontally relative to the luminance (Y), the MCU consists of 2 blocks of Y followed by one block each of U and V.

1.2.1.2 Restart Intervals

The ZR36060 supports compression and decompression of JPEG data that includes restart intervals. A restart interval is defined as an integral number of MCUs, which are processed as an “independent sequence”, meaning that it is possible to identify and decode a restart interval within a JPEG data sequence, without the need to decode whatever data precedes it. In the context of baseline compression, this has significance because the DC coefficients of the DCT are differentially encoded. Note that the use of restarts is optional; it is acceptable (and very common) to use no restart markers and encode the whole image as a single sequence.

1.2.2 JPEG markers

JPEG defines three data formats for the compressed bitstream, all of which are supported by the ZR36060:

- The *interchange* format, which contains the specifications of all the tables required to decode the image.
- the *abbreviated* format for compressed data, which can contain some or none of the tables, under the assumption that the remaining tables are known to the decoder and are already loaded in the decoder or can be loaded. This is commonly used for motion video, in order to save the time otherwise required to decode the tables from their specifications.
- the abbreviated *tables-only* format, which contains no compressed data but only tables. It is one means by which it is possible to load tables into the decoder; in the ZR36060 the other means is by specifying the tables to the device and issuing an explicit Load command.

In all three of the formats, the table specifications and the parameters required for decoding the image and/or the tables are contained in *marker segments*, which are sequences of bytes that start with special two-byte codes called *markers* or *marker codes*. The two bytes that follow the marker specify the length of the marker segment in bytes, including the two length bytes but not including the marker code itself. There are three special stand-alone markers that are not associated with marker segments, to mark the start-of-image (SOI), restart (RST), and end-of-image (EOI). The code values are 0xFFD8 for SOI, 0xFFD0-0xFFD7 for RST and 0xFFD9 for EOI.

The first byte of every marker is 0xFF. A marker may be prefixed by an arbitrary number of 0xFF bytes which are discarded by the decoder. The second byte of a marker has defined values, except for 0x00, which is used as follows. In order to permit a decoder to identify the restart markers, if they exist, and the EOI marker, the encoder stuffs a 0x00 byte after every 0xFF byte that results from the Huffman encoding. Note that this “byte stuffing” is an essential part of the JPEG standard, and there is no definition in the standard of a bitstream that does not include the byte stuffing. The ZR36060 always produces image bitstreams with byte stuffing, and requires the byte stuffing to be present in order to decode a JPEG bitstream.

The JPEG standard also does not define any sort of “markerless” bitstream data format. Certain markers and marker segments are defined in the standard to be “required”, and others, such as the restart markers and the table marker segments, are optional. The ZR36060 always includes the required markers when it produces a compressed bitstream, and can be programmed to include certain optional markers. To be decompressed by the ZR36060, an image bitstream *must* include the required markers. All markers included in the bitstream, required and optional, are handled automatically, without host intervention, by the ZR36060 in decompression.

1.2.2.1 Required markers and marker segments

The required markers for baseline JPEG are:

- Start-of-image, SOI (0xFFD8). This is the first marker in a JPEG image bitstream.
- Start-of-frame marker segment, SOF0 (0xFFC0), followed by a variable number of bytes depending on the number of color components. For the ZR36060, there are always three components and the segment has a length of 17 bytes. The SOF segment is used to specify which quantization table to use for each color component, and the number of blocks of each color component in the MCU.
- Start-of-scan marker segment, SOS (0xFFDA), followed by a variable number of bytes depending on the number of color components. The Huffman coded data follows immediately after the last byte of the SOS segment. In the case of the ZR36060, the length of the SOS segment is always 12 bytes. The SOS segment is used to specify which Huffman table to use for each color component.

- End-of-image, EOI (0xFFD9). This marker follows the last byte of the compressed data.

1.2.2.2 Optional markers and marker segments

The ZR36060 supports the following optional markers and segments:

- Application specific, APPn (0xFFE0-0xFFEF). The standard allows up to 16 different APP markers in a single image bitstream. The ZR36060 can insert one APP marker in compression. A ZR36060 APP marker can have a segment length of up to 62 bytes. In decompression, if the image bitstream contains a single APP marker with a segment length of 62 bytes or fewer, the host can retrieve it from the internal memory after the ZR36060 has finished decompressing the image; if the segment is longer, the data is lost. If there are multiple APP segments, only the last one can be retrieved.
- Comment, COM (0xFFFE). The restriction on the length (62 bytes) is the same as for the APP marker.
- Define restart interval, DRI (0xFFDD). Specifies that restarts are to be used, and the size in MCUs of the restart interval.
- Define quantization tables, DQT (0xFFDB). Specifies the quantization tables used to compress the image.
- Define Huffman tables, DHT (0xFFC4). Specifies the Huffman tables used to compress the image.
- Restart, RSTm (0xFFD0-0xFFD7). Marks the beginning of a restart interval in the compressed data. This marker is inserted by the ZR36060 only if the restart mechanism is enabled (see the description of the RST bit of the Markers Enable Register). There is no RSTm marker before the first restart interval.

Note that when quantization and Huffman tables are loaded into the ZR36060 by the host controller, they are specified in exactly the same format as is used in the marker segments.

In compression, the ZR36060 inserts optional marker segments, if programmed to do so, into the compressed data bitstream in a fixed order: APP, COM, DRI, DQT, DHT. These appear immediately after SOI, before SOF. In decompression, they can appear in any order or position allowed by the JPEG standard; multiple marker segments of the same type are supported to the extent allowed by the standard.

1.2.3 Motion JPEG

The JPEG standard defines a method for compression of a single (“still”) image. It does not have any provision for motion video, and the term “motion JPEG” simply means that each field of a video sequence is compressed as a separate JPEG image bitstream, starting with SOI and ending with EOI. The ZR36060 includes features that make this procedure straightforward.

1.3 Notational Conventions

The following notational conventions are used in this data sheet:

External signals: capital letters (e.g., COMP)

Active-low mark: overbar (e.g., $\overline{\text{RESET}}$)

Buses: XXmsb_index:lsb_index (e.g., UV7:0)

Register fields: XXmsb_index:lsb_index (e.g., Count27:16)

Register types:

- R - read only
- W - write only
- RW - read-write (data written can be read back)

Numbers: numbers with no prefix or suffix are decimal (e.g., 365, 23.19). Hexadecimal numbers are indicated with a ‘0x’ prefix (e.g., 0xB000, 0x3). Binary numbers are indicated with a ‘b’ suffix (e.g., 010b, 0000110100011b).

Control register bit fields are *italicized* when mentioned in the text.

2.0 SIGNAL DESCRIPTION

Table 1: Signal Description

Symbol	Type	Description
Code/Host Port (26 pins)		
CODE7:0	I/O	Code bus. In Code Master mode, this 8-bit bidirectional bus is used to read (or write) the compressed data from (or to) an external code FIFO. In 16-bit Code Slave mode, this is used as an extension (the MSB) of the DATA bus. During and after $\overline{\text{RESET}}$ this bus is floating, with internal pull-ups.
$\overline{\text{CCS}}$	O	Code Chip Select, used only in Code Master mode. This active-low output signal acts as a chip select from the ZR36060 to the external code FIFO. $\overline{\text{CCS}}$ goes active at the start of a read or write cycle and remains active throughout the cycle. $\overline{\text{CCS}}$ remains active continuously in back-to-back read or write cycles. During and after $\overline{\text{RESET}}$ this pin is at logic high.
$\overline{\text{COE}}$	O	Code Read (output enable), used only in Code Master mode. This active-low output signal acts as a read strobe signal from the ZR36060 to the external code FIFO. $\overline{\text{COE}}$ goes active 0.5 VCLKx2 cycles after start of a read cycle. The CODE bus input is latched on the rising edge of $\overline{\text{COE}}$. During and after $\overline{\text{RESET}}$ this pin is at logic high.
$\overline{\text{CWE}}$	O	Code Write, used only in Code Master mode. This active-low output signal acts as a write strobe from the ZR36060 to the external code FIFO. $\overline{\text{CWE}}$ goes active 0.5 VCLKx2 cycles after start of a write cycle. CODE bus data is valid throughout the strobe pulse and permits the external code FIFO to latch the data on the rising edge of $\overline{\text{CWE}}$. During and after $\overline{\text{RESET}}$ this pin is at logic high.
$\overline{\text{CBUSY}}$	I/O	Code FIFO Busy. When the ZR36060 is the master of the code bus $\overline{\text{CBUSY}}$ is an active-low input, used by the external code FIFO controller to temporarily halt the transfer of compressed data. When the ZR36060 is the slave of the code bus $\overline{\text{CBUSY}}$ is an active-low output. It is asserted (low) by the ZR36060 to indicate the internal code FIFO cannot be accessed, due to an empty/full condition (for compression/decompression modes respectively). On deassertion, $\overline{\text{CBUSY}}$ is driven high for one internal clock and then released to a floating condition (needs external pull-up). When the ZR36060 is connected to the ZR36057, $\overline{\text{CBUSY}}$ is connected to the $\overline{\text{CBUSY}}$ input of the latter. During and after $\overline{\text{RESET}}$ this pin is floating (input mode).
DATA7:0	I/O	Data bus. This 8-bit bidirectional bus is used to access the internal memory of the ZR36060. In Code Slave mode, it is also used to transfer the compressed data. In 16-bit Code Slave mode, the CODE bus is used as an extension of the DATA bus. During and after $\overline{\text{RESET}}$ this bus is floating with internal pullup.
ADDR1:0	I	Address bus. This 2-bit bus is used by the host to access the code register (in Code Slave mode), or the indirect address/data register which maps the 1Kbyte internal memory array of the ZR36060.
$\overline{\text{CS}}$	I	Chip Select. This active-low input signal acts as a chip select from the host to the ZR36060.
$\overline{\text{WR}}$	I	Write strobe. This active-low input signal acts as a write pulse from the host to the ZR36060. The DATA bus, with CODE bus extension in 16-bit Code Slave mode, is latched on the rising edge of $\overline{\text{WR}}$.
$\overline{\text{RD}}$	I	Read strobe. This active-low input signal acts as a read pulse from the host to the ZR36060. The DATA bus, with CODE bus extension in 16-bit Code Slave mode, is enabled as an output during the $\overline{\text{RD}}$ pulse so the host can latch the ZR36060 data on the rising edge of $\overline{\text{RD}}$.
$\overline{\text{ACK}}$	O	Acknowledge. Used by the ZR36060 to notify the host that the current read or write strobe pulse can be completed. During code access (Code Slave mode), the ZR36060 will not issue an $\overline{\text{ACK}}$ if the internal code FIFO is empty/full (in compression/decompression respectively). On deassertion, $\overline{\text{ACK}}$ is driven high for 1 VCLKx2 cycle and then released to a floating condition (needs external pull-up). During and after $\overline{\text{RESET}}$ this pin is floating (at logic high with pullup).
Video Port (25 pins)		
Y7:0	I/O	In 16-bit video mode, these are the video luminance pins. In 8-bit mode, these are luminance/chrominance pins, multiplexed in time according to the CCIR656 component order. In compression these pins are inputs, while in decompression they are outputs. During and after $\overline{\text{RESET}}$ these pins are floating with internal pullups.
UV7:0	I/O	In 16-bit video mode, these are the video chrominance pins. In compression these pins are inputs, while in decompression they are outputs. In 8-bit video mode, these pins are not used: in compression they are ignored (inputs), and in decompression they are floating. During and after $\overline{\text{RESET}}$ these pins are floating with internal pull-ups.
VCLKx2	I	Main Video Clock input. The video interface of the ZR36060 is synchronized by this clock.

Table 1: Signal Description (Continued)

Symbol	Type	Description
VCLK	I	Digital video bus clock enable. Used as a qualifier of the video bus data. Must be synchronized and toggling at half the frequency of VCLKx2, in both 8 and 16-bit video bus width modes.
HSYNC	I/O	Horizontal sync. When the ZR36060 is a sync slave, HSYNC is input, and when it is the sync master, HSYNC is an output. During and after $\overline{\text{RESET}}$ this pin is floating (input mode).
VSYNC	I/O	Vertical sync. When the ZR36060 is a sync slave, VSYNC is input, and when it is the sync master, VSYNC is an output. During and after $\overline{\text{RESET}}$ this pin is floating (input mode).
FI	I/O	Digital video bus field indicator (odd/even). When the ZR36060 is the sync master FI is an output, otherwise it is an input. The polarity of FI, as input or output, is programmable. During and after $\overline{\text{RESET}}$ this pin is floating (input mode).
BLANK	O	Digital video bus composite blank output. Active only when the ZR36060 is the sync master of the video bus, otherwise the pin is floating. The horizontal and vertical blanking areas are programmable. During and after $\overline{\text{RESET}}$ this pin is floating with internal pullup.
PVALID	I	When the ZR36060 is in compression mode, this input is used as an additional qualifier (other than VCLK) of the video data signals and the sync signals. An active level sampled on this signal at the time when a pixel is sampled, indicates that this is a valid pixel. When the ZR36060 is in decompression mode, this input is used by the recipient of the video to stall the video stream of the ZR36060. A non-active level sampled on this signal will cause the ZR36060 to continue to output the current pixel instead of proceeding to the next one. Once PVALID is sampled active again the normal pixel sequence resumes. If the ZR36060 is the video sync master, then PVALID not active will freeze the internal sync generator. The polarity of PVALID is programmable. When the ZR36060 is connected to the ZR36057, PVALID is connected to the $\overline{\text{PXEN}}$ output of the latter.
SUBIMG	O	This output dynamically indicates the boundaries of a sub-image rectangle within the main input or output field size. When the pixels within the programmable rectangle are output/input, SUBIMG is active. For a sub-line of consecutive pixels within the rectangle, SUBIMG is continuously active. The polarity of SUBIMG is programmable. SUBIMG may be connected to the FEIN input of the SAA7110/11, or the read-enable input of a line buffer, FIFO, etc., to permit pixel-by-pixel video mixing during compression and decompression. During and after $\overline{\text{RESET}}$ this pin is logic high.
POE	I	Pixel Output Enable. Used to disable the video bus during decompression, to permit pixel-by-pixel video mixing of the ZR36060 video output with another source. It can be directly connected to the SUBIMG output, or to another suitable control signal.
Control & Status (10 pins)		
$\overline{\text{RESET}}$	I	Reset. When this input is asserted the ZR36060 goes into its RESET state. When it is deasserted all state machines are in the IDLE state and registers contain their default values. $\overline{\text{RESET}}$ must be active for at least 8 VCLKx2 cycles.
$\overline{\text{SLEEP}}$	I	Power-down mode. When this input is active (low), the ZR36060 goes into its SLEEP (power-down) state, discontinuing all chip operation and consuming minimal supply current. This pin also initiates coarse locking of the internal PLL to the VCLKx2 frequency. It must be toggled at least once after the initial $\overline{\text{RESET}}$ applied after power-up. $\overline{\text{SLEEP}}$ must remain low for at least 8 VCLKx2 cycles.
$\overline{\text{END}}$	O	End of process indication. This active-low output signal indicates completion of a field compression/decompression process. During and after $\overline{\text{RESET}}$ this pin is logic low.
$\overline{\text{EOI}}$	O	End-of-image marker indication. In Code Slave mode, this active-low output signal indicates that the End-Of-Image code word FFD9 (16-bit code bus), or the second byte of this word (8-bit code bus), is being output or input. $\overline{\text{EOI}}$ is deasserted together with the deassertion (rising edge) of $\overline{\text{END}}$ at the beginning of the next field process. During and after $\overline{\text{RESET}}$ this pin is logic low.
$\overline{\text{START}}$	I	Start compression/decompression command input. When the ZR36060 is in the IDLE state, it waits for an active low level on this input in order to start compression or decompression. Once the active level is sampled, the ZR36060 will start compression or decompression with the next VSYNC or with the next odd VSYNC (depending on the $\overline{\text{FRAME}}$ input). To be detected correctly, $\overline{\text{START}}$ must remain low for at least 2 VCLK cycles.
$\overline{\text{FRAME}}$	I	This input is sampled by the ZR36060 together with the $\overline{\text{START}}$ input. When $\overline{\text{START}}$ is sampled active, then if $\overline{\text{FRAME}}$ is also active the ZR36060 will start compressing/decompressing at the next odd field. Otherwise it will start with the next field.
$\overline{\text{DATERR}}$	O	This output is asserted when there is a data corruption event. It is deasserted together with the deassertion (rising edge) of $\overline{\text{END}}$ upon beginning of the next field process. On deassertion, $\overline{\text{DATERR}}$ is floating (needs external pull-up). During and after $\overline{\text{RESET}}$ this pin is floating (logic high with pullup).

Table 1: Signal Description (Continued)

Symbol	Type	Description
$\overline{\text{RTBSY}}$	O	In compression this output signal indicates a “nearly full” condition in the internal raster-to-block memory (strip buffer). This condition occurs when the strip buffer is 16 (or fewer) pixels away from an overflow condition. In decompression $\overline{\text{RTBSY}}$ indicates that the strip buffer is near an underflow condition. In the IDLE state $\overline{\text{RTBSY}}$ is not asserted. If while $\overline{\text{RTBSY}}$ is asserted a data corruption event occurs (overflow or underflow), $\overline{\text{RTBSY}}$ continues to be asserted together with $\overline{\text{DATERR}}$ until the beginning of the next field process (deassertion of $\overline{\text{END}}$). If no data corruption occurs, $\overline{\text{RTBSY}}$ is deasserted as soon as the almost-overflow/underflow condition is no longer true. If the ZR36060 is used with a ZR36057/67, $\overline{\text{RTBSY}}$ should be connected to the $\overline{\text{RTBSY}}$ input of the ZR36057/67. During and after $\overline{\text{RESET}}$ this pin is at logic high.
$\overline{\text{JIRQ}}$	O	Interrupt request (active low). This output signal requests an interrupt from the host controller, if an interrupt request is enabled and the event associated with interrupt request occurs. It is deasserted if the host responds to the interrupt by reading the interrupt status register, or if the host disables the interrupt, or upon a reset to the ZR36060. On deassertion $\overline{\text{JIRQ}}$ is floating (needs external pull-up). When $\overline{\text{JIRQ}}$ is active, the $\overline{\text{START}}$ signal is disregarded. During and after $\overline{\text{RESET}}$ this pin is floating (logic high with the pullup).
COMP	O	Compression/Decompression. This output signal provides an indication of the current operating mode of the ZR36060. When it is high, the ZR36060 is in the compression mode; when it is low, the ZR36060 is in the decompression mode. During and after $\overline{\text{RESET}}$ this pin is at logic high.
Power Signals		
GND		Ground
VDD		Power supply (3.3V)
VDDPLL		Power pin of the PLL (3.3V). See page 41.

3.0 VIDEO INTERFACE

The video interface of the ZR36060 is highly configurable, to facilitate a glueless connection to most video decoders, encoders, MPEG decoders, frame memory controllers, graphics accelerators, etc.

3.1 Video Syncs - Master and Slave Modes

The ZR36060 supports two video sync source modes:

- Sync Master - the ZR36060 internally generates all the video timing signals.
- Sync Slave - the ZR36060 synchronizes itself with an external video source.

The 1-bit *SyncMstr* parameter selects the mode. Normally, in compression the ZR36060 would be slaved to the output of a video decoder, but not necessarily; for example, the ZR36060 could control a frame memory in Sync Master mode.

3.1.1 Master mode

When configured as a sync Master, the ZR36060 drives the following signals:

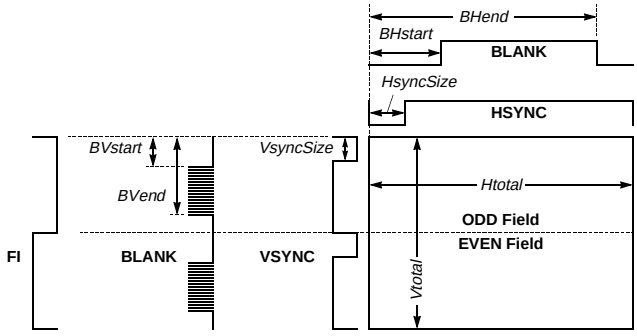
- HSYNC - Horizontal sync
- VSYNC - Vertical sync
- FI - Even/Odd field indication
- BLANK - Composite blanking

The parameters that configure the sync generator when the ZR36060 is a sync master are (see Figure 3):

- *Vtotal* - Total number of lines per frame (e.g.- for NTSC, 525 video lines)
- *Htotal* - Total number of VCLKs (pixels) per line (e.g.- for CCIR NTSC, 858 pixels)
- *VsyncSize* - Length of the VSYNC pulse measured in lines
- *HsyncSize* - Length of HSYNC pulse measured in VCLKs
- *BVstart* - Length (in lines) from VSYNC to first non-BLANK line.
- *BVend* - Length (in lines) from VSYNC to last non-BLANK line.
- *BHStart* - Length (in pixels) from HSYNC to first non-BLANK pixel.
- *BHend* - Length (in pixels) from HSYNC to last non-BLANK pixel.
- *VSPol* - Polarity of the VSYNC signal
- *HSPol* - Polarity of the HSYNC signal
- *FIPol* - Polarity of the FI signal
- *BlPol* - Polarity of the BLANK signal
- *FIVedge* - Defines at which VSYNC edge the FI signal changes state (leading or trailing edge). This is also the reset point for the vertical counters, indicating the end of the previous field and the beginning of a new field.

After the parameters are properly initialized and loaded (using the Load command), the sync generator is free running, and is not affected by the state of the JPEG codec. The *SyncRst*

register bit resets the sync generator counters and the PVALID signal can temporarily freeze the counting and sync signals.



Note: In this example *VSPol* = *HSPol* = *FIPol* = *BlPol* = *FIVedge* = 0.

Figure 3. Video Sync Generation

3.1.2 Slave mode

When configured as a sync Slave, the ZR36060 samples the following signals:

- HSYNC - Horizontal sync
- VSYNC - Vertical sync
- FI - Even/Odd field indication

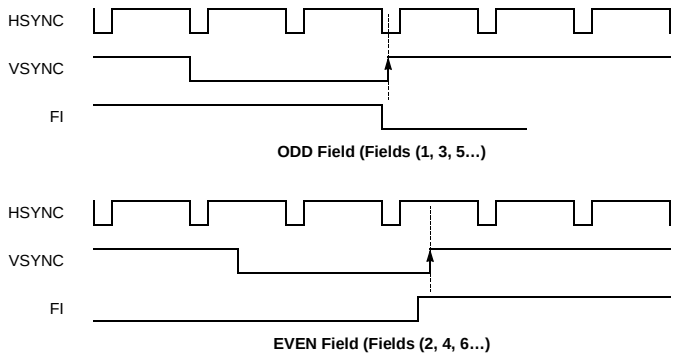
The parameters *Vtotal*, *Htotal*, *VsyncSize*, *HsyncSize*, *BVstart*, *BVend*, *BHstart*, *BHend*, *BlPol*, *FIPol* are not used in Slave mode. *VSPol*, *HSPol*, *FIDet* and *FIVedge* are used as follows:

- *VSPol* - Polarity of the VSYNC input signal.
- *HSPol* - Polarity of the HSYNC input signal.
- *FIDet* - Exchange the even/odd field interpretation after detection. (Detection can be accomplished in two ways according to the *FIExt* parameter, see below.)
- *FIVedge* - Defines the reset point for the vertical counters indicating the end of the previous field and the beginning of a new field. When *FIExt* = 0 it also defines the proper VSYNC edge used to latch HSYNC to internally detect the even/odd field.

The field detection can be accomplished in two ways depending on the *FIExt* parameter (see Figure 4):

- External indication by means of the FI signal (*FIExt* = 1), toggling every VSYNC, indicating whether the current field is even or odd. The polarity of FI is programmable, using the *FIPol* parameter, while the even/odd interpretation can be exchanged using the *FIDet* parameter.
- Internal detection (*FIExt* = 0), derived from latching the state of HSYNC at each VSYNC. This is useful when using the ZR36060 with video sources that do not provide a dedicated field indication signal. Odd fields are those where the VSYNC edge latches the HSYNC during its short sync period, while on even fields the VSYNC edge latches the HSYNC in the middle of the line (see Figure 4). The VSYNC edge (leading or trailing) used to latch the HSYNC signal can be programmed by means of the *FIVedge* parameter. Changing *FIDet* will change the even/odd interpretation.

Note: the HSYNC edge must precede the latching VSYNC edge by at least 2 VCLKs for reliable latching.



Note: In this figure, *VSPol* = *HSPol* = *FIPol* = *FIDet* = 0, *FIVedge* = 1

Figure 4. Field Detection Signal Relationships

3.2 Data Formats

When the ZR36060 is configured for 16-bit video bus width (*Video8*=0), the luminance signal is on Y7:0, and the chrominance signals are multiplexed on the UV7:0 lines (see Figure 5). When operating in 8-bit video bus mode (*Video8*=1), both the luminance and the chrominance signals are on Y7:0, multiplexed in time according to the CCIR656 recommendation (U=Cb, V=Cr):

U0,Y0,V0,Y1,U2,Y2,V2,Y3,....

For 16-bit video, the pixels are sampled on every other rising edge of VCLKx2, which is enabled by VCLK, the video clock qualifier. The polarity of the VCLK qualifier is programmable via the VCLKPol parameter. 8-bit video is sampled using all rising edges of VCLKx2, at twice the pixel rate.

Note that the duration of a pixel is always 1 VCLK period, with both 16-bit and 8-bit video widths. All internal counters and video events are based on VCLK, that must always be present (at half the frequency of VCLKx2) even when the video interface is configured for 8-bit width.

In decompression, the output pixel levels are CCIR-601 compliant, with values in the range [16,235]. It is possible to override this with the *Range* parameter bit and let the ZR36060 output the full 256-level scale.

Note that in both 16-bit and 8-bit modes, the ZR36060 does not output, nor expect to receive, control codes indicating timing information, on its YUV video bus.

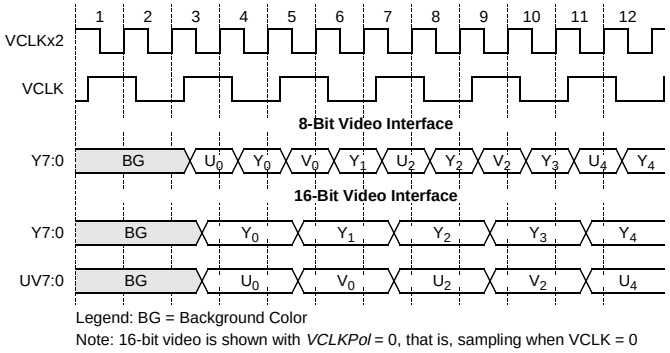


Figure 5. Video Data Formats, 8 and 16 bit

3.3 Video stream sampling and cropping

Only pixels within a rectangular active area are sampled and compressed (in compression) or output (in decompression), as shown in Figure 6. The VSYNC signal indicates the beginning of a new field (the VSYNC edge and polarity are configured by *FIVedge* and *VSPol*). The *Vstart* and *Vend* parameters determine the first and last lines to be sampled in a field. The leading edge of HSYNC indicates the beginning of a horizontal line (with HSYNC polarity according to *HSPol*). The *Hstart* and *Hend* parameters determine the first and last pixels to be sampled in each line. Further processing such as formatting, scaling and compression is done only to pixels within the active rectangle. In decompression, the video bus outputs a background color, specified by the *BackY*, *BackU*, and *BackV* parameters, outside the processed active area rectangle.

Figure 7 and Figure 8, and the associated notes, describe how the active area is aligned relative to VSYNC and HSYNC.

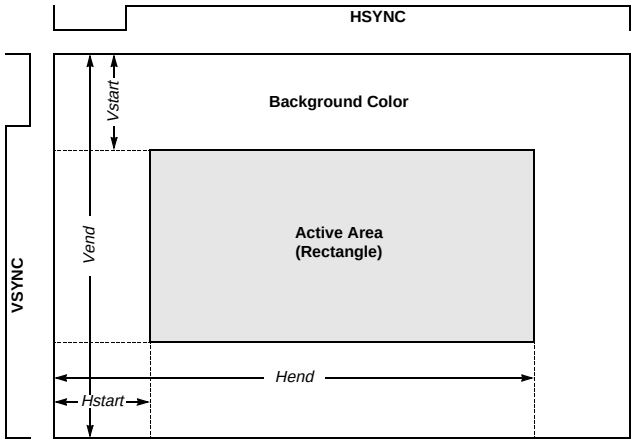
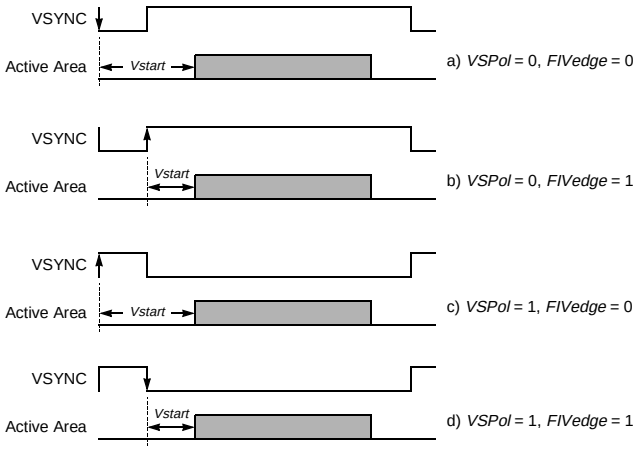
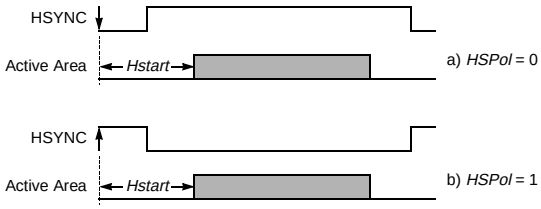


Figure 6. Active Area of the Video



Notes: The active video area must not overlap the VSYNC pulse. In other words, the active area must always be contained between the trailing edge of VSYNC and the next leading edge.
There are restrictions on the allowed values of *VStart* and *VEnd*. See page 32.

Figure 7. Relationship of VSYNC and Active Video Area



Notes: The line counting (for *Vstart*, *Vend*) always uses the leading edge of the HSYNC pulse. *Hstart* is specified from the leading edge of HSYNC.
The active video area is allowed to partially overlap the HSYNC pulse. In other words, *Hstart* could be before or after the trailing edge of HSYNC.
There are restrictions on the allowed values of *HStart* and *HEnd*. See page 32.

Figure 8. Relationship of HSYNC and Active Video Area

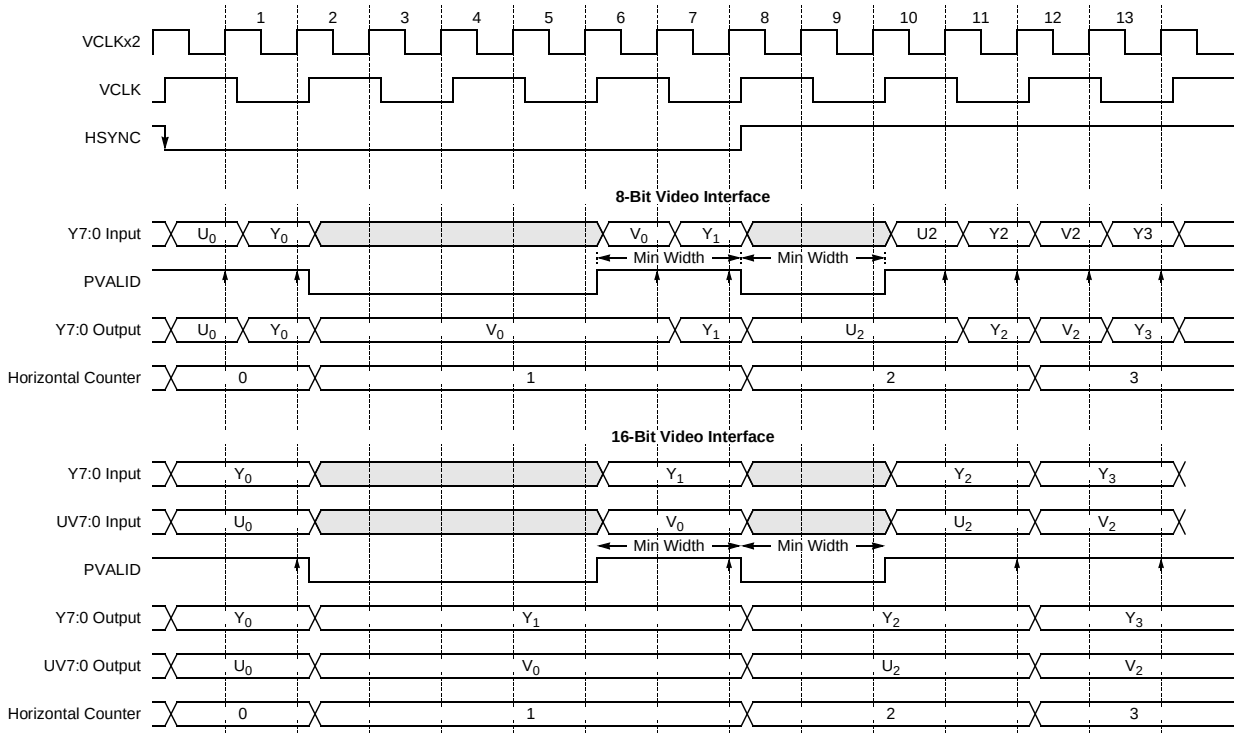
3.3.1 The PVALID control signal

The continuous video stream is usually used by encoders and decoders for ‘real-time’ video capture and playback. However, sometimes it may be desirable to ‘hold’ or not sample the video pixels intermittently, especially when connected to a slow peripheral (such as a host interface compressing a still image, or a memory controller) that cannot cope with the real-time pixel rate. The PVALID signal can be used for this purpose.

PVALID acts as a pixel qualifier indicating the presence of ‘valid’ pixels on the bus (analogous to the action of VCLK in 16-bit mode). It can interrupt the video stream (in and out) for any period of time with a resolution of VCLK, as shown in Figure 9. VCLK and PVALID differ in that VCLK must always toggle at half the rate of VCLKx2, while PVALID can maintain a continuous level. Only pixels qualified by PVALID that are within the rectangular active area are sampled. PVALID also acts as a ‘count enable’ to the horizontal and vertical counters that implement *Hstart*, *Hend*, *Vstart*, and *Vend*. For example, after the leading edge of HSYNC the ZR36060 counts *Hstart* pixels qualified by

PVALID, and then samples pixels qualified by PVALID until *Hend*.

The polarity of PVALID is programmable by means of the *PValPol* parameter.



Notes: 1. In this illustration, *HSPol* = 0; *VCLKPol* = 0 (sample when VCLK = 0); *PValPol* = 1 (valid when PVALID = 1).
2. "Horizontal counter" represents an internal counter used to identify the active area.
3. PVALID granularity is one VCLK, in both 8- and 16-bit video interface modes.
4. PVALID may toggle only on a pixel boundary, in this figure when VCLK=0.

Figure 9. PVALID Operation, 8- and 16-bit Video Widths

3.4 Video Scaling

The ZR36060 incorporates a scaler, that can scale the video in the active area, horizontally and vertically, by simple ratios. It can down-scale the video before it is compressed, and up-scale it after it is decompressed. The result is to permit straightforward implementation of "half screen" and "quarter screen" compression.

The horizontal down- and up-scaling are accompanied by filtering. Note that this filtering can not be disabled. Vertical down- and up-scaling employ line dropping and line replication respectively.

3.4.1 Horizontal down-scaling in compression

This is specified by the 2-bit *HScale* parameter. There are three possible configurations:

HScale = 00b or 11b: No down-scaling

HScale = 01b: 2:1 decimation, with a 3-tap filter. The filter coefficients are (0.25, 0.5, 0.25).

HScale = 10b: 4:1 decimation, with a 5-tap filter. The filter coefficients are (0.0625, 0.25, 0.375, 0.25, 0.0625).

3.4.2 Vertical down-scaling in compression

This is specified by the 1-bit *VScale* parameter:

VScale = 0b: No down-scaling

VScale = 1b: 2:1 decimation, by line dropping

In the case of 2:1 vertical scaling, the second, fourth,...etc. lines of the active area of the video field are dropped before the video is compressed.

3.4.3 Horizontal up-scaling in decompression

As in compression, this is specified by *HScale*:

HScale = 00b or 11b: No up-scaling

HScale = 01b: 2:1 interpolation

HScale = 10b: 4:1 interpolation

The interpolated samples are created by averaging of the two neighboring samples, with weighting proportional to the distance of the interpolated point from the two samples used.

3.4.4 Vertical up-scaling in decompression

As in compression, this is specified by *VScale*:

VScale = 0b: No up-scaling

VScale = 1b: 2:1 interpolation, by line replication

3.5 Active Area Size Restrictions

The maximum allowed size for the active area rectangle is 768 pixels x 64K lines.

The ZR36060 JPEG codec always processes an image with dimensions of 2*8*x and 8*y pixels (x, y integers ≥ 2). This is because of the YUV 4:2:2 format, where the MCU is 2 blocks of Y, 1 block of U and 1 block of V. The active area of the video interface must be configured to reflect the dimensions before down-scaling in compression, and after up-scaling in decompression. Tables 2 and 3 show the resulting restrictions imposed on the dimensions of the active area.

Table 2: Active Area, Horizontal Dimension (*HEnd - HStart*)

Horizontal scaling	Restriction
No scaling (1:1)	Multiple of 16
2:1	Multiple of 32
4:1	Multiple of 64

Table 3: Active Area, Vertical Dimension (*VEnd - VStart*)

Vertical Scaling	Restriction
No scaling (1:1)	Multiple of 8
2:1	Multiple of 16

In the internal sampling scheme, the first chrominance sample is always assumed to be a U (Cb) sample. This is directly controlled by the *Hstart* parameter. In compression, *Hstart* must be programmed to an appropriate value (even or odd) in order for the ZR36060 to sample first the U (Cb) pixel, otherwise U-V inversion occurs.

3.6 Spatial Mix of Video Streams

The ZR36060 is capable of spatially mixing (multiplexing) two video sources for compression, and also of multiplexing the ZR36060 output video with another video source during decompression. The latter is a useful feature for video editing, e.g. to superimpose titles or subtitles onto the images.

The SUBIMG output signal creates a sub-image rectangle defined by the *SHstart*, *SHend*, *SVstart*, *SVend* parameters, where one image is outside and the other one is inside the rectangle (see Figure 10). In compression, this signal can be connected, for example, to two synchronized sources of live video to multiplex their outputs. Some digital video sources have a video bus which can be placed in a floating state (for example, the Philips SAA7110 video digitizer/decoder); SUBIMG can be used to float this bus while enabling a second video source. Some possible options are to multiplex two video decoders, one decoder and one field memory, one video decoder and one MPEG decoder, etc.

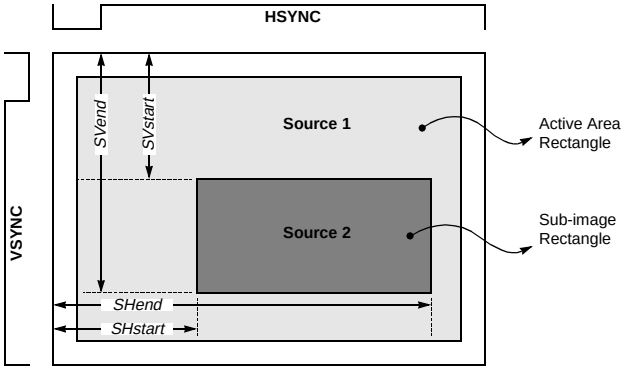


Figure 10. Sub-image Parameters In Compression

There are several inherent problems in mixing video that the system designer must bear in mind:

- The two video sources must be synchronized. This means that pixel clocks, horizontal and vertical timing must come from only one source which is the sync master.
- Both video sources must work in the same mode (16-bit or 8-bit width).
- The *SHstart* and *SHend* parameters must be chosen so that the boundaries of the sub-image rectangle (where SUBIMG changes state) coincide with the boundaries between independent YUV 4:2:2 units (units of two VCLKs, containing related U and V samples) for both sources.

To permit video mixing during decompression (playback), the SUBIMG output can be externally connected to the POE input. This way, the ZR36060 places its video data only within (or outside) the rectangle defined by SUBIMG, floating the output video bus outside (or inside) the boundaries (see Figure 11). The polarity of the SUBIMG signal is defined by the *SImgPol* parameter, and the polarity of the POE signal (to place ZR36060 data inside or outside the rectangle during playback) is defined by the *PoePol* parameter. In Figure 11, the polarity of SUBIMG has been chosen so as to float the video bus outside the sub-image rectangle.

Note that SUBIMG and POE operate independently of each other, so they can also be used separately.

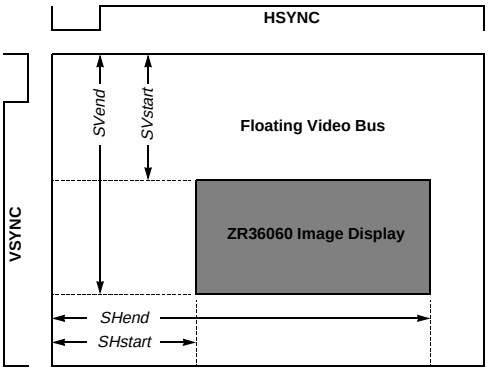


Figure 11. Sub-image Parameters (with SUBIMG Wired to POE) In Decompression

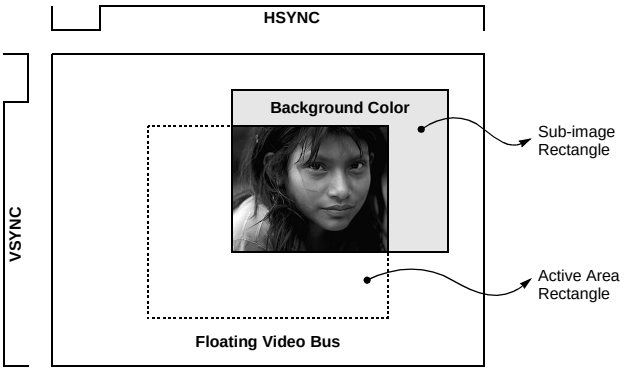


Figure 12. Video Output from the ZR36060 in Decompression, Using SUBIMG with POE

During decompression, the SUBIMG rectangle is overlaid on the active rectangle. In other words, the video bus will be floating or active in all the areas indicated in Figure 11 regardless of whether the underlying pixels are active or background color (see also Figure 12).

The example in Figure 13 shows the result of the spatial mixing of the decompressed video with another video source, as seen by the destination (such as a video encoder).

Figure 14 shows the timing of the transitions at the sub-image boundaries, for the same typical example in which SUBIMG is used to control POE. The timing of the 16-bit external video source in the example is that of the Philips SAA7110.

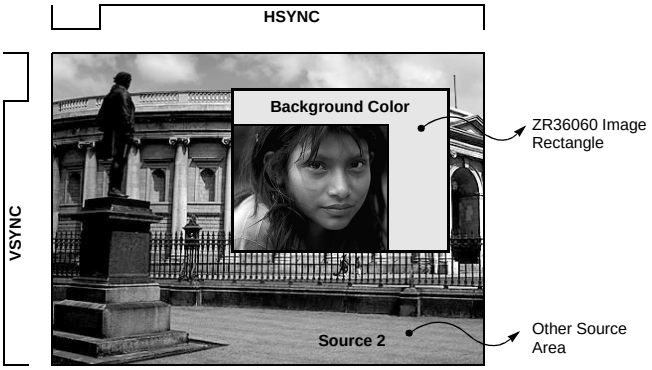
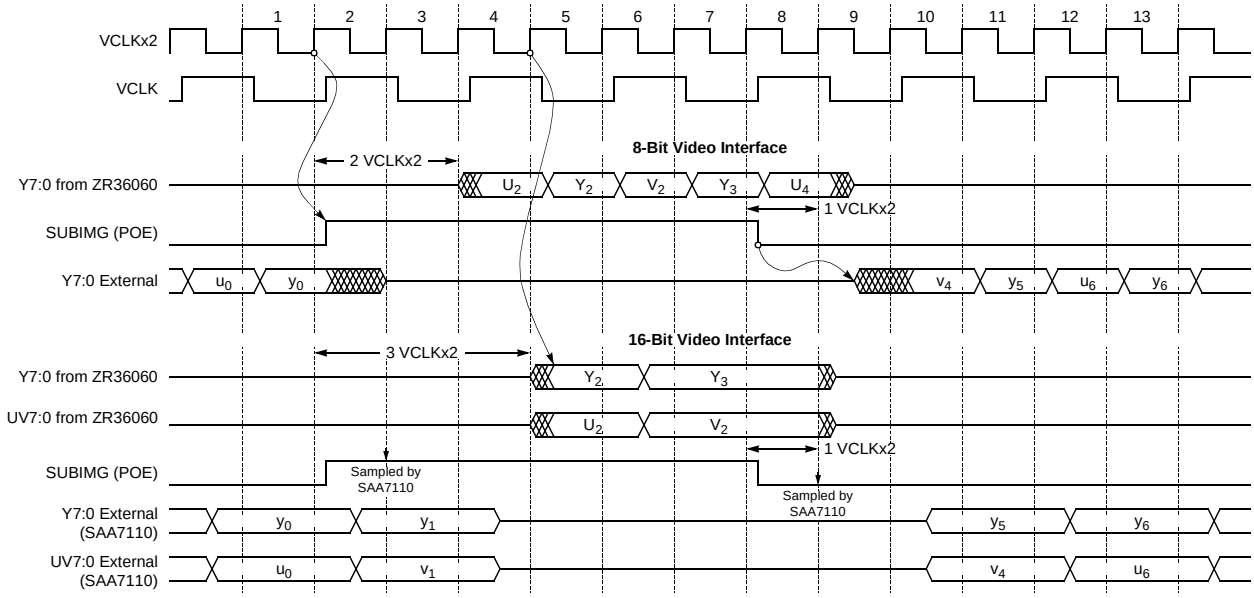


Figure 13. ZR36060 Output Image Multiplexed with Another Source, as Seen by a Video Encoder



Notes:
1. In this example *SImgPol* = *VCLKPol* = 0.
2. In this example SUBIMG is connected to POE to float the ZR36060 video bus.
3. SUBIMG changes state with resolution of one VCLK and at the rising edge of VCLKx2, in both 8-bit and 16-bit mode.
4. In 8-bit mode, the first pixel is enabled 2 VCLKx2 after SUBIMG changes state, and the last pixel is disabled 1 VCLKx2 after SUBIMG changes.
5. In 16-bit mode, the first pixel is enabled 3 VCLKx2 after SUBIMG changes state (this causes the first pixel from the ZR36060 to appear on the bus for 0.5 VCLK instead of a complete VCLK period). The last pixel is disabled 1 VCLKx2 after SUBIMG changes, to avoid contention. This timing was chosen to match the characteristics of the SAA7110. SUBIMG is also connected to the SAA7110's output enable, FEIN.

Figure 14. SUBIMG and POE timing during decompression, shown for 8- and 16-bit video widths

4.0 HOST INTERFACE

The host interface is a generic interface with an 8-bit bidirectional data bus, 2-bit address bus (that indirectly maps a 1Kbyte internal memory space), $\overline{RD}$, $\overline{WR}$, $\overline{CS}$, and $\overline{ACK}$ pins. It supports glueless or low-glue interface to many microprocessors and microcontrollers, and buses like the ISA.

When the Code interface is configured in Slave mode (see section 5.0 "Code Interface") some of the ZR36060 Host interface pins have dual functions, as can be seen in Figure 15.

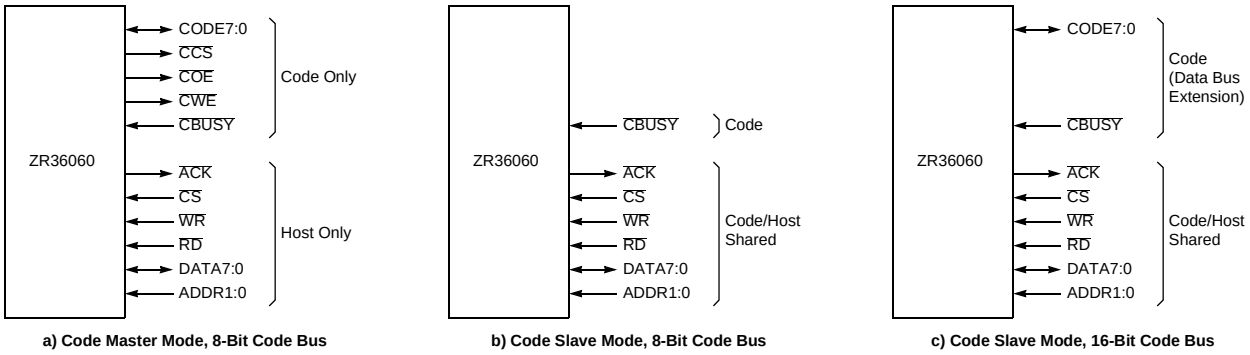


Figure 15. The Various Code Interface Modes and the Host Interface

The ADDR1:0 address pins map 4 direct access registers (Figure 16 and Figure 17):

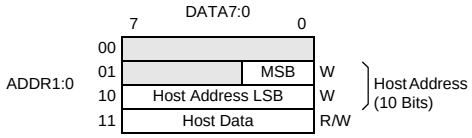
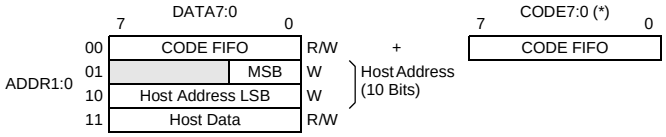


Figure 16. Address Space of ZR36060 in Code Master Mode



(*) The Code FIFO register can be 8 or 16 bits wide, depending on the *Code16* parameter. When in 16-bit Code Slave mode, the CODE7:0 bus is an extension of the DATA7:0 bus.

Figure 17. Address Space of ZR36060 in Code Slave Mode

To access the ZR36060's internal Code FIFO (in Code Slave mode only), read and write operations are directed to address 00b. For more information on the Code FIFO access, refer to section 5.0 "Code Interface".

To access the ZR36060's internal registers and markers array, the host must first write the 10-bit host address, followed by a read or write to the 8-bit host data register. Note that the host address is not self-incrementing. However, it does not need to be re-written every data access, only if a different register or memory location is to be accessed. The host address of the location to be accessed can be changed by writing the LSB register, the MSB register, or both, as required.

The host interface is an asynchronous interface (see Figure 18). Internally, however, all the interface I/O is synchronized to an internal PLL clock (at twice the frequency of VCLKx2), so VCLKx2 must exist and be stable, and the PLL locked, before any host access can take place.

A Host-ZR36060 handshake is performed using the $\overline{WR}$ or $\overline{RD}$ strobe pulses, and the $\overline{ACK}$ signal. Some time after $\overline{WR}$ or $\overline{RD}$ goes low, $\overline{ACK}$ is activated by the ZR36060 to acknowledge that it is ready to input or output host or code data. Only after this event is the host allowed to release the strobe. The ZR36060 acknowledges the end of the access cycle by releasing $\overline{ACK}$.

A slow host may extend the strobe pulse beyond the activation of the $\overline{ACK}$ signal by the ZR36060. In a read cycle, data from the ZR36060 stays on the bus until after the $\overline{RD}$ signal is deasserted. In a write cycle, the data is strobed in on the rising edge of $\overline{WR}$.

Note that, if it is guaranteed that the minimum $\overline{WR}$ or $\overline{RD}$ strobe width is always larger than the minimum specified in the Timing Characteristics, the $\overline{ACK}$ signal can be ignored.

When accessing the Code FIFO (address 00b) in Code Slave mode, the $\overline{ACK}$ signal reflects also the $\overline{CBUSY}$ status (see section 5.0 "Code Interface" for more details). (But $\overline{CBUSY}$ is only to be used by the host in Code Slave mode and must be ignored in all other host accesses.)

For a complete description of the internal memory and register mapping, refer to section 8.0 "Register and Memory Description".

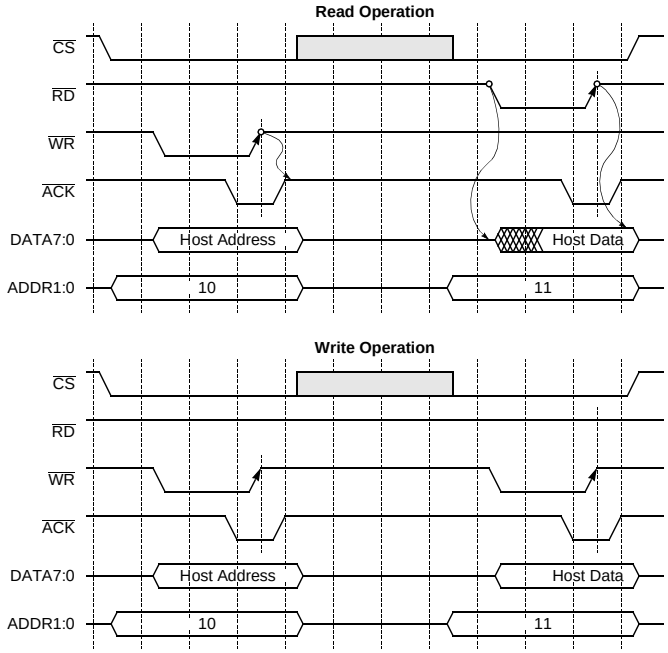


Figure 18. Asynchronous Operation of the Host Interface

4.1 Interrupt Request and Associated Registers

The ZR36060 is capable of requesting an interrupt from the host controller by means of the $\overline{JIRQ}$ output signal. This section describes the protocol and registers associated with the interrupt request.

An interrupt request can occur as a consequence of one or more of the following events:

- Assertion of the $\overline{DATERR}$ output (a data corruption event).
- Assertion of the $\overline{END}$ output.
- Assertion of the $\overline{EOI}$ (end-of-image marker detection) output during decompression.
- End of the active rectangle of the video field (*EOAV*) which is being processed by the ZR36060.

Each one of the events has a dedicated bit (*DATERR*, *END*, *EOI*, and *EOAV*, respectively) in the Interrupt Mask Register that enables or disables it as an interrupt requesting event.

Each of the events also has a status bit in the Interrupt Status Register.

The *DATERR* and *END* bits exactly reflect the levels of the $\overline{\text{DATERR}}$ and $\overline{\text{END}}$ output pins, respectively, but with positive logic (as opposed to the negative logic of the output pins).

The *EOI* bit should only be used when decompressing in Code Slave mode; in Code Master mode it is meaningless. It exactly reflects the state of the $\overline{\text{EOI}}$ signal (but with positive logic).

The *EOAV* bit indicates that the last line of the active area (as defined by the active area parameters), has been sampled (or displayed) by the ZR36060. Note that in Auto Two-Pass Compression mode, the *EOAV* bit is asserted only in the first pass.

The *DATERR*, *END*, *EOI*, and *EOAV* Interrupt Status bits are set when the respective event occurs, and cleared together with their respective pins (with the exception of *EOAV*, which does not have a pin associated with it) at the beginning of the next process, i.e.- at the next $\overline{\text{START}}$.

Note that the Interrupt Status Register bits always reflect valid status information without regard to whether their corresponding interrupts are enabled in the Interrupt Mask Register.

When an interrupt-enabled event occurs, $\overline{\text{JIRQ}}$ is asserted, and once the ZR36060 asserts $\overline{\text{END}}$ (completion of the field process, i.e. compression or decompression of the current field) it moves to the WAIT-ISR state (see the state diagram in Figure 28). $\overline{\text{JIRQ}}$ remains asserted until the host reads the Interrupt Status Register (see Figure 19). When this happens $\overline{\text{JIRQ}}$ is deasserted and the ZR36060 completes the process and returns to the IDLE state, ready to sample $\overline{\text{START}}$ for the next field process.

The Interrupt Status Request register includes another pair of bits, *ProCount1:0*, that are not related to interrupt requests, but are located in this register for convenience. *ProCount1:0* is the output of a modulo-4 cyclic counter that advances with every start of a process (every rising edge of $\overline{\text{END}}$). It is reset only by $\overline{\text{RESET}}$ which initializes the counter to 01b. It may be used by host controllers as an indication of a field dropped by the ZR36060 (e.g., when the ZR36060 outputs $\overline{\text{END}}$ of one field after the next one already started).

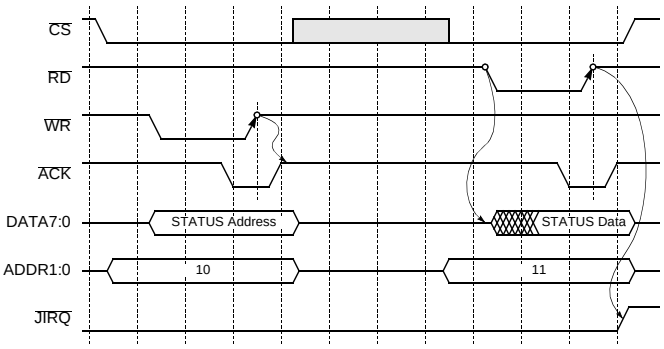


Figure 19. Interrupt Acknowledgment by Reading the Interrupt Status Register

5.0 CODE INTERFACE

The code interface has two modes of operation:

- Code Master mode
- Code Slave mode

The operating mode of the code port is selected through the *CodeMstr* register bit. After $\overline{\text{RESET}}$ the ZR36060 defaults to Code Master mode. With 29.5 MHz VCLKx2, the peak code throughput is 29.5 MByte/sec in Master mode, 16.8 MByte/sec in 16-bit Slave mode and 9.8 MByte/sec in 8-bit Slave mode. With 27 MHz VCLKx2, the peak code throughput is 27 MByte/sec in Master mode, 15.4 MByte/sec in 16-bit Slave mode and 9 MByte/sec in 8-bit Slave mode. The master mode is almost identical to the master mode of the ZR36050. It is compatible with the ZR36057 PCI JPEG controller and with the ZR36055 ISA JPEG controller. The slave mode is compatible with common microprocessors or microcontrollers.

5.1 Master Mode

In this mode the compressed data is transferred on the 8-bit CODE bus, using the $\overline{\text{CCS}}$, $\overline{\text{COE}}$, and $\overline{\text{CWE}}$ outputs to inform the system when a code bus cycle is taking place, and the $\overline{\text{CBUSY}}$ input to stall further accesses until the system is available again. Master mode differs from the ZR36050's master mode in two minor ways:

- The *CFIS* parameter, that determines the bus cycle time in this mode, is limited to the values 0b (one VCLKx2 per cycle) and 1b (2 VCLKx2 per cycle)
- The CAEN signal of the ZR36050 does not exist in the ZR36060.

A Master mode cycle starts with the activation of $\overline{\text{CCS}}$, on the rising edge of VCLKx2. $\overline{\text{CCS}}$ remains active throughout the bus cycle and remains active continuously in back-to-back cycles. In a read cycle, executed during decompression, $\overline{\text{COE}}$ goes active 0.5 VCLKx2 period after the beginning of the cycle and remains active until the end of the cycle. Data is strobed in on the trailing edge of $\overline{\text{COE}}$. Similarly, in a write cycle, executed during compression, $\overline{\text{CWE}}$ goes active 0.5 VCLKx2 period after the beginning of the cycle and remains active until the end of the cycle. Examples are shown in Figure 20 and Figure 21.

$\overline{\text{CBUSY}}$ is sampled one VCLKx2 before the beginning of each bus cycle and if active, inhibits the bus cycle. If a bus cycle started at the same time $\overline{\text{CBUSY}}$ was sampled active it completes normally.

Note: the *CBUSY* and *EOI* status bits are not valid in Code Master mode.

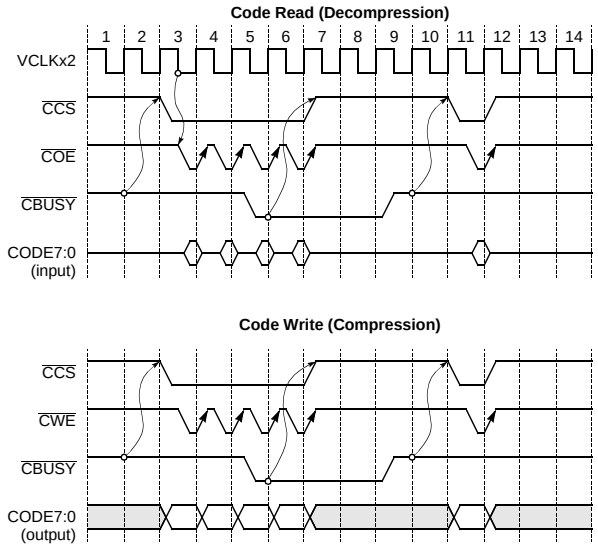


Figure 20. Master Mode Operation of the Code Bus, with *CFIS*=0b (1 VCLKx2 Per Cycle)

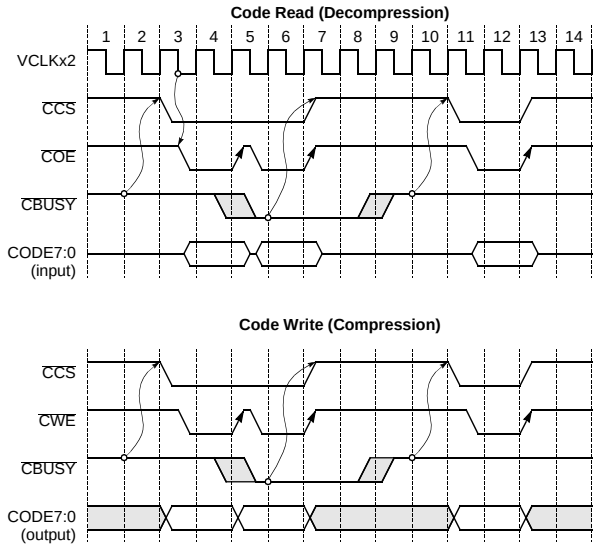


Figure 21. Master Mode Operation of the Code Bus, with *CFIS*=1b (2 VCLKx2 Per Cycle)

5.2 Slave Mode

In Slave mode, access to the internal code FIFO is accomplished using the host interface, by reading or writing (depending on compression or decompression mode respectively) direct access address zero (ADDR1:0 = 00b). The data bus width can be 8 or 16 bits, depending on the *Code16* parameter:

- 8-bit width (*Code16* = 0). Only DATA7:0 is used for the code transfer.
- 16-bit width (*Code16* = 1). CODE7:0 is an extension of DATA7:0]to transfer 16-bit words. The byte ordering can be exchanged using the *Endian* parameter.

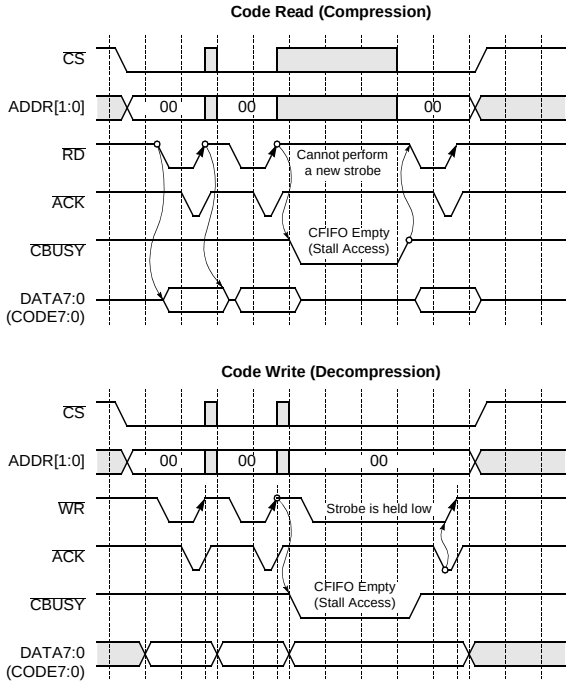
Code Slave mode accesses are asynchronous (Figure 22). The $\overline{CS}$ (Code Chip-Select) and ADDR1:0 inputs can be deasserted after every $\overline{RD}$ or $\overline{WR}$ cycle, or, in order to achieve the best performance, left asserted for a burst of code transfer cycles.

The host must select one of three different methods to handshake with the ZR36060 throughout the compression or decompression process:

- use of the $\overline{CBUSY}$ signal, or
- use of the $\overline{ACK}$ signal, or
- by polling the CFIFO level bits.

$\overline{CBUSY}$ is used as an indication of the empty/full status of the internal code FIFO of the ZR36060. When $\overline{CBUSY}$ is active (low), it means that the code FIFO is empty (during compression) or full (during decompression). When the host uses this signal, it must sample $\overline{CBUSY}$ prior to each code access, and hold off the assertion of $\overline{RD}$ or $\overline{WR}$ until $\overline{CBUSY}$ is deasserted.

The ZR36060's $\overline{ACK}$ signal indicates permission to a complete the current cycle. Assertion of $\overline{ACK}$ indicates that the internal code FIFO is not empty (during compression) or not full (during



Notes:
1. $\overline{CS}$ can be pulsed, or maintained active for burst of read or write pulses.
2. $\overline{ACK}$ is not granted when $\overline{CBUSY}$ is active, in both compression and decompression.
3. The top example (compression) shows a system using $\overline{CBUSY}$ to decide when to perform the next $\overline{RD}$ strobe.
4. The bottom example (decompression) shows a system using $\overline{ACK}$ to decide when to terminate the current strobe. Note the extension of the $\overline{WR}$ cycle when the FIFO is full.

Figure 22. Slave Mode Operation of the Code Bus

decompression), and therefore the transfer can be successfully completed. A host using this signal must not terminate the $\overline{RD}$ or $\overline{WR}$ strobe before the ZR36060 acknowledges the cycle by asserting $\overline{ACK}$. This can actually stall the host in the middle of a compressed data stream, or between compressed data fields in the case of continuous operation, if the host attempts to read (or write) the FIFO while it is completely empty (or full).

The system must be designed to use $\overline{CBUSY}$ (or its equivalents, see below) or $\overline{ACK}$ output signal as a handshake, but not both. Maximum code transfer rate performance is achieved when using $\overline{CBUSY}$ handshake and 16-bit code bus width.

The host also has access to a status register to interrogate the full/empty status of the internal code FIFO via the *CBUSY* bit (positive logic as opposed to the $\overline{CBUSY}$ pin state), and to a pair of read-only bits, *CFIFO1:0,* which indicate the fullness of the code FIFO as follows:

- *CFIFO1:0* = 00b: less than 1/4 of the FIFO is occupied.
- *CFIFO1:0* = 01b: 1/4 or more, but less than 1/2, of the FIFO is occupied.
- *CFIFO1:0* = 10b: 1/2 or more, but less than 3/4, of the FIFO is occupied.
- *CFIFO1:0* = 11b: 3/4 or more of the FIFO is occupied.

Using this register, the host can poll the fifo status directly (instead of using the $\overline{CBUSY}$ or $\overline{ACK}$ signals), to determine when it should temporarily stop the code transfer. However, the system must guarantee minimum $\overline{RD}$ and $\overline{WR}$ strobe widths, since $\overline{ACK}$ is not being used.

After the last code word is input (in decompression) or output (in compression), the ZR36060 asserts the $\overline{EOI}$ signal, indicating the end of the code stream with the End-Of-Image marker.

The *CFIFO1:0* status bits are not valid after the end of code stream signal ($\overline{EOI}$) and corresponding status bit have been asserted.

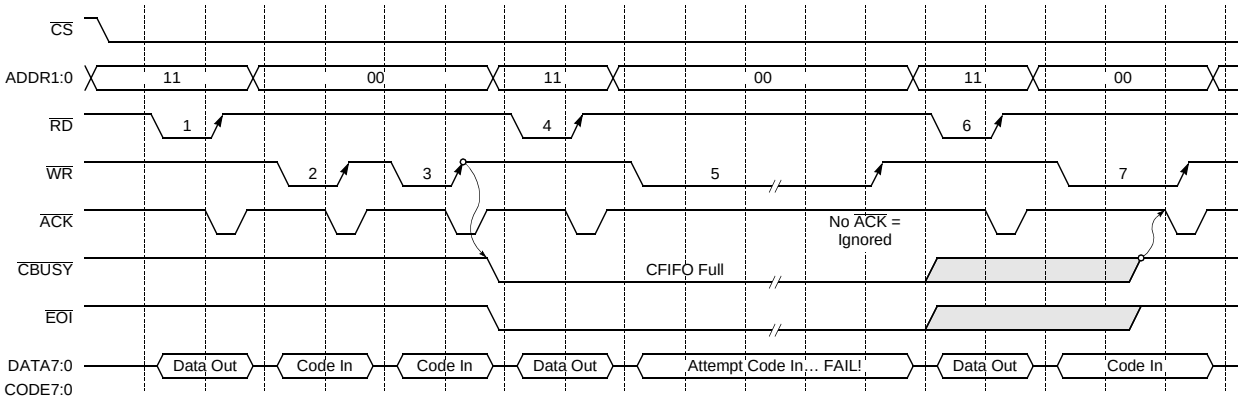
5.2.1 Host abort of a code read or write cycle

During the compression or decompression process the host must obey the handshake rules to create a valid code file. The host can safely abort (e.g.- due to a timeout) a $\overline{RD}$ or $\overline{WR}$ cycle only after the $\overline{EOI}$ signal is asserted. If a code transfer cycle is aborted by the host before this, the behavior of the ZR36060 will be unpredictable, and it must be reset to resume normal operation.

Figure 23 shows an example of a code transfer abort after $\overline{EOI}$ is asserted.

5.2.2 Data alignment in Code Slave mode

In compression, in code slave mode, the ZR36060 always completes the compressed data file for each field so that it is 32-bit (double-word) aligned. This is true for both 8-bit and 16-bit interface modes. A variable number of padding bytes of value 0xFF is appended by the ZR36060 after the EOI marker, to complete the last double word. In decompression, such padding bytes constitute a legal preamble for the SOI marker code of the next field, and thus are ignored by the ZR36060.



This example shows status register reads interleaved with code FIFO write transactions. After the end-of-image code, instead of waiting too long (for the $\overline{ACK}$ signal) until the next field, the host can decide to abort the cycle.

Notes:
1. The host reads the status register, receives $\overline{ACK}$ normally.
2, 3. The host writes some code bytes (two in this example), and the ZR36060 asserts $\overline{ACK}$ normally. After the second code write cycle, the ZR36060 senses the end-of-image code indicating the last code byte of the current field and asserts $\overline{CBUSY}$.
4. In this example the host reads a status register, and $\overline{ACK}$ is asserted, independently of $\overline{CBUSY}$.
5. This is an access to the code FIFO while $\overline{CBUSY}$ is asserted. No $\overline{ACK}$ can be issued by the ZR36060, until after the beginning of the next field. The host is stalled for a while, then decides to abort the cycle, i.e.- release the $\overline{WR}$ strobe without $\overline{ACK}$ from the ZR36060. The ZR36060 senses this situation as an abort of the cycle and ignores the strobe. Again note, that this is only allowed while $\overline{EOI}$ is asserted, and not in the middle of the process.
6. The host now decides to read a status register. This operation is completed normally.
7. The host writes code into the ZR36060. The ZR36060 has already started decompressing a new field, so $\overline{CBUSY}$ and $\overline{EOI}$ are released and $\overline{ACK}$ can be issued.

Figure 23. Example of ZR36060 Interleaved Code/Data Accesses and Abort of Access

In 8-bit interface mode, the number of appended bytes can be 1, 2, 3 or 4.

In 16-bit interface mode, the number of appended bytes can be 2, 3, 4 or 5.

Note that in both cases, if the number of bytes from the first byte of SOI up to and including the second byte of EOI is an exact multiple of 4, the ZR36060 actually appends 4 more bytes. Note also that in 16-bit mode, if one byte is required to make the total a multiple of 4, the ZR36060 actually appends 5 bytes.

5.2.3 Transition between fields in compression

For compression, Figure 24 shows code transfer with 8-bit interface and Figure 25 with 16-bit interface, at the transition between consecutive fields showing the behavior of the $\overline{EOI}$, $\overline{END}$ and $\overline{CBUSY}$ signals.

$\overline{CBUSY}$ is asserted after the last padding byte has been read out. It remains asserted continuously until the first code byte of the next field is available. A code read is shown in the figures while $\overline{CBUSY}$ is asserted, with the host access stalled.

$\overline{EOI}$ is asserted as soon as the read cycle of last byte of the EOI marker code (0xD9) is complete.

$\overline{END}$ is asserted only after the all the compressed data, including the padding bytes, was read out, and the post-compression calculations are completed and their results stored in the host-accessible registers. At this time the ZR36060 returns to the IDLE state. Compression of the next field is started when the ZR36060 senses the $\overline{START}$ signal active.

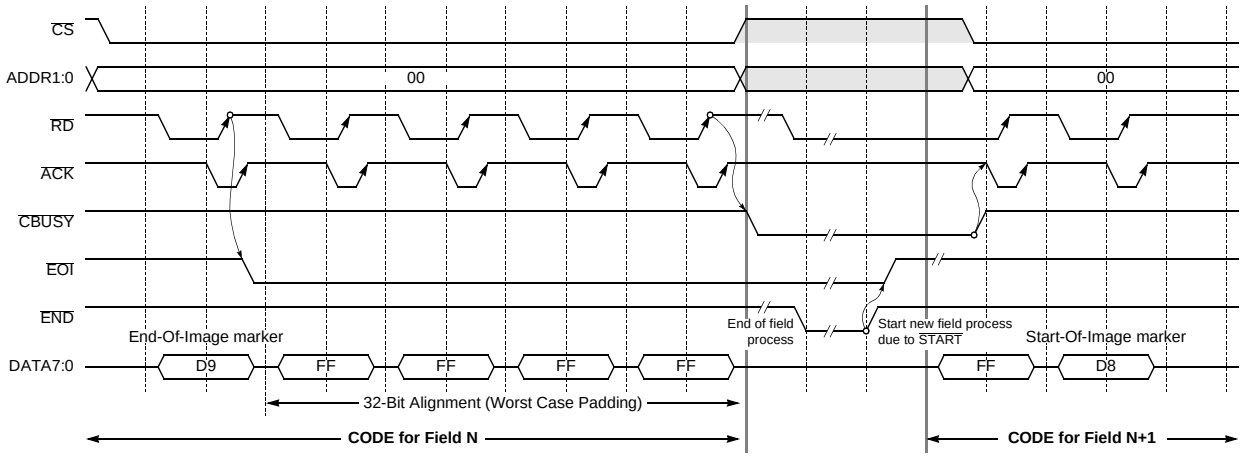


Figure 24. 8-Bit Code Slave Mode Compression, Transition Between Consecutive Fields

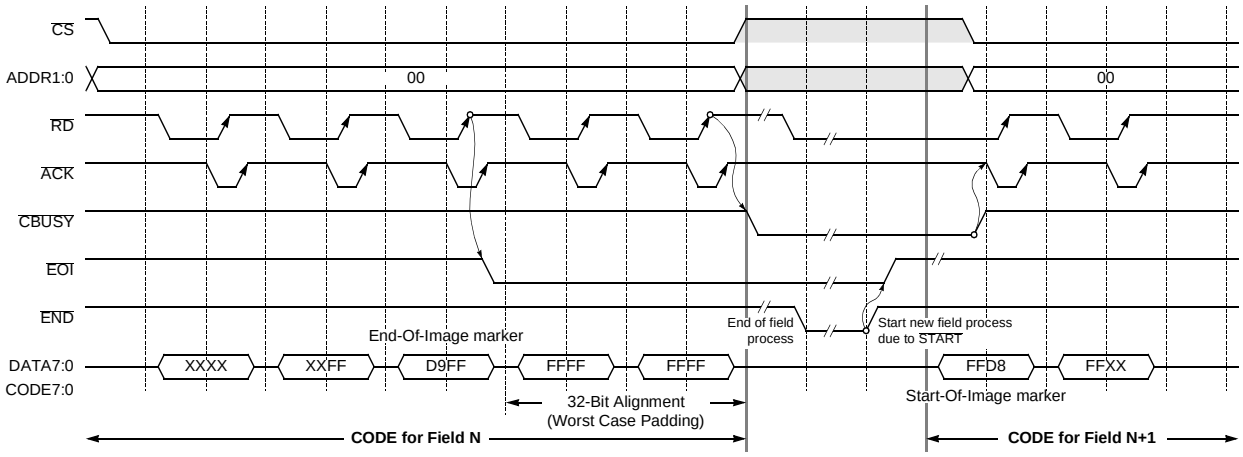


Figure 25. 16-Bit Code Slave Mode Compression, Transition Between Consecutive Fields

5.2.4 Transition between fields in decompression

For decompression, Figure 26 shows code transfer with 8-bit interface and Figure 27 with 16-bit interface, at the transition between consecutive fields, showing the behavior of the $\overline{EOI}$, $\overline{END}$ and $\overline{CBUSY}$ signals.

JPEG compressed files input to the ZR36060 can be any size, not required to be 32-bit (double-word) aligned. The ZR36060 detects the EOI marker (0xFFD9), asserting $\overline{EOI}$ and $\overline{CBUSY}$ at the next write strobe. Note that the host is allowed to write one additional byte after the EOI marker code in 8-bit mode, or one additional word after the word containing the second byte of the EOI marker code in 16-bit interface mode. This byte or word is discarded by the ZR36060. If this discarded code byte or word contained one or both bytes of the SOI marker of the next field, the ZR36060 automatically reconstructs the marker when it starts decompressing the next field.

Note that $\overline{EOI}$ and $\overline{CBUSY}$ may remain unasserted until $\overline{END}$ is asserted, if the host has other means to detect the EOI marker code and therefore does not issue the additional write strobe.

Attempts to access the code FIFO while $\overline{CBUSY}$ is asserted will be held off until the FIFO is available again, using the $\overline{ACK}$ signal. In this example, a $\overline{WR}$ pulse is shown being extended until the next field begins, when $\overline{CBUSY}$ is deasserted.

$\overline{END}$ is asserted only after the whole decompressed field has been output from the video interface. At this time the ZR36060 sets the *EOAV* status bit and returns to the IDLE state. Decompression is started again when the ZR36060 senses the START signal active.

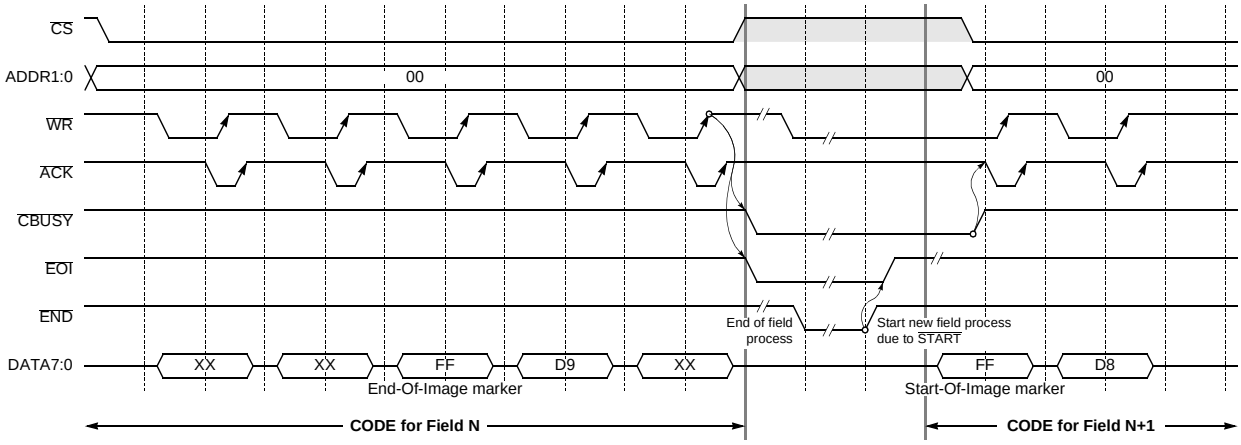


Figure 26. 8-Bit Code Slave Mode Decompression, Transition Between Consecutive Fields

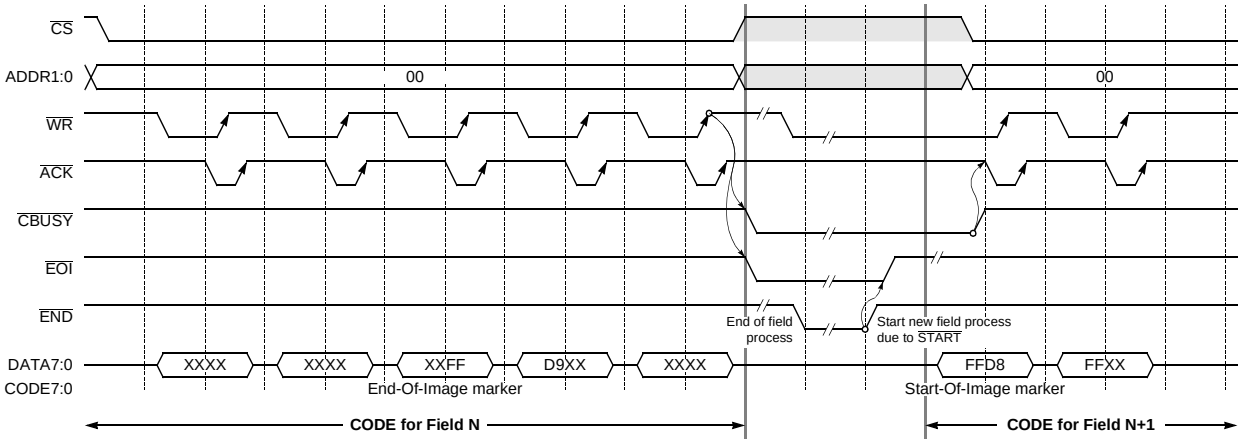


Figure 27. 16-Bit Code Slave Mode Decompression, Transition Between Consecutive Fields

6.0 OPERATION

6.1 ZR36060 Functional States

For the purposes of this description, the ZR36060 can be viewed as having 7 states:

- RESET - In this state the $\overline{\text{RESET}}$ input is held active.
- SLEEP - Power-down. The $\overline{\text{SLEEP}}$ input is held active in this state.
- IDLE - $\overline{\text{END}}$ is asserted and the ZR36060 is waiting for $\overline{\text{START}}$.
- WAIT-ACTIVE - After the ZR36060 sensed $\overline{\text{START}}$ asserted, it waits for the beginning of the active area of the next field to be processed (this depends on the state of $\overline{\text{FRAME}}$ when $\overline{\text{START}}$ was sampled active). $\overline{\text{END}}$ is deasserted.
- CMP - Compression of the active area. The video bus is input, and the code data bus is output. $\overline{\text{END}}$ is deasserted.
- EXP - Decompression (expansion) of the active area. The video bus is output, and the code data bus is input. $\overline{\text{END}}$ is deasserted.
- WAIT-ISR - After the ZR36060 finished the compression or decompression and asserted $\overline{\text{END}}$ while $\overline{\text{JIRQ}}$ is active (due to a non-masked interrupt), the ZR36060 waits in this state for the host to read the Interrupt Status Register.

6.2 State Transitions

Figure 28 depicts the states and their transitions.

6.3 The SLEEP State

In this state, all the pins remain in the logic states they were in immediately before the transition to SLEEP. No host, video or code interface operation is allowed in the SLEEP state.

When the ZR36060 leaves the SLEEP state it returns to IDLE, ready for the next compression or decompression operation.

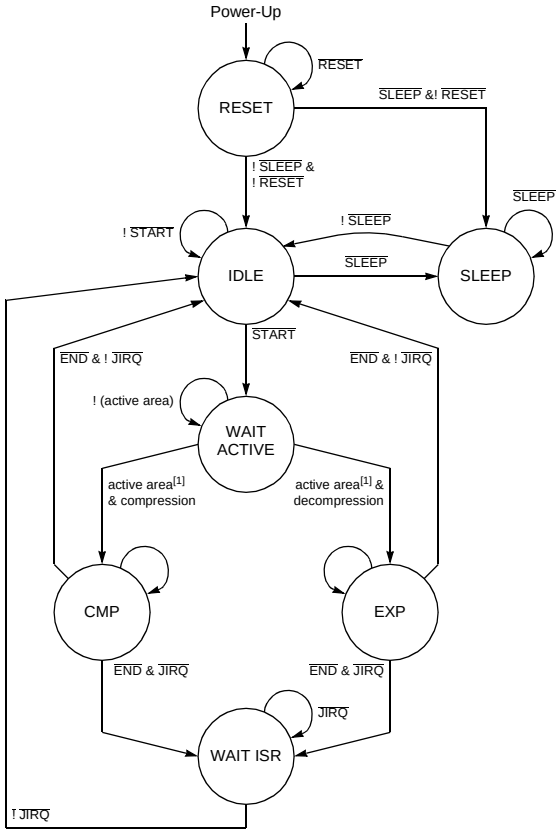
This state is also used to lock the internal PLL to the frequency of VCLKx2, so it is mandatory to go through the SLEEP state at least once after power-up and before operating the device (see section 7.0 “Power Management and Power-up”).

6.4 Loading Parameters and Tables

Prior to a compression or decompression process the host must load the appropriate parameters and tables into the ZR36060. The parameters affect the compression/decompression mode, the video interface, and the operation of the code port.

All parameters and tables may be loaded only when the ZR36060 is in the IDLE state.

First, the host controller writes (via the host interface) the desired parameters and/or tables in their correct locations in the 1Kbyte internal memory (see section 8.0 "Register and Memory Description" for details).



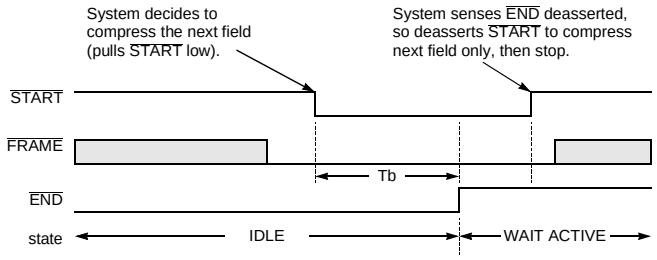
1. Active area of the correct field, depending on the state of $\overline{\text{FRAME}}$ when $\overline{\text{START}}$ was sampled active.

Figure 28. ZR36060 Functional States

Then, the host sets the ZR36060’s (write-only) *Load* bit. This commands the ZR36060 to initialize or “Load” all internal hardware blocks with the parameters in its internal memory, and also to decode the Huffman and Quantization tables into the internal form required for compression or decompression. While the ZR36060 is performing this Load operation, the (read-only) *Busy* bit is set to ‘1’. The host must poll for the completion of the loading, i.e.- wait for the *Busy* bit to be reset to ‘0’, before starting the compression or decompression process.

The $\overline{\text{START}}$ signal is ignored during the execution of the Load, i.e.- the ZR36060 remains in the IDLE state (with $\overline{\text{END}}$ asserted). Only after the Load is complete (*Busy* is reset), and the new parameter values become effective, is the ZR36060 ready to sample $\overline{\text{START}}$ and to move to the WAIT-ACTIVE state to start the compression or decompression process (see Figure 29).

Parameters and status registers can be read in any ZR36060 state with the exception of RESET and SLEEP.



Tb = Delay while the ZR36060 completes execution of the Load command.

Figure 29. IDLE to WAIT-ACTIVE state transition

6.5 Data Flow Overview

This section provides an overview of the data flow in the ZR36060 during compression and decompression. For this purpose it is useful to view the ZR36060 roughly as a JPEG engine with one dual port data buffer on each side: the code FIFO buffer on one side and the video buffer (strip buffer) on the other side (see Figure 30 through Figure 33).

6.5.1 Data flow in compression

The video input, after being processed in the video interface, is written to the strip buffer in raster format. The JPEG engine reads out the data in block format, and writes the JPEG code into the code FIFO on the other side. From the FIFO the data is transferred out either by the ZR36060 itself, if it is the code bus master, or by the host controller, if the ZR36060 is in code slave mode.

When the ZR36060 is the code bus master (see Figure 30) it writes out the code as long as its $\overline{\text{CBUSY}}$ input is not asserted. When the code FIFO is empty the ZR36060 does not perform code write cycles. If the host controller is too slow and asserts $\overline{\text{CBUSY}}$ for too long, the code FIFO may fill up, and in order to prevent overflow, the ZR36060 stops reading data from the strip buffer. If this situation continues long enough, the strip buffer overflows, because video keeps flowing in. A strip buffer overflow is a data corruption event. At the system level this event may be prevented by two means: First, the host should be able to accept the code at the same rate it is generated by the ZR36060. Second, some system configurations may have the capability to halt the video input stream when the strip buffer is close to overflow (16 pixels, or less, from overflow). The ZR36060 indicates this “nearly full” condition with its $\overline{\text{RTBSY}}$ output. One configuration for implementing this is if the ZR36060 is the master of the video syncs, and the system stops the video using the PVALID input. This may be useful, for example, when compressing still pictures.

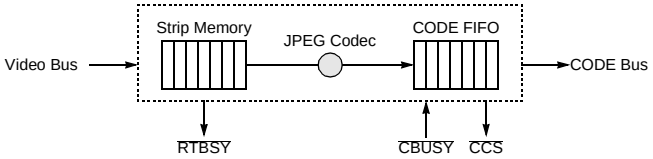


Figure 30. Data Flow in Compression, Code Master Mode

When the code interface operates in slave mode (see Figure 31) the scenario is almost identical. The main difference is that $\overline{\text{CBUSY}}$ is now an output of the ZR36060, used to indicate that the code FIFO is “nearly empty”, thus the host must stop reading code until $\overline{\text{CBUSY}}$ indicates that the FIFO occupancy is above its threshold.

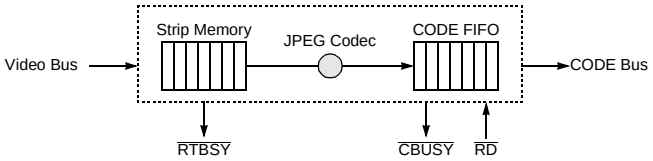


Figure 31. Data Flow in Compression, Code Slave Mode

6.5.2 Data flow in decompression

The JPEG code is transferred on the code bus into the code FIFO, either by the ZR36060, in Code Master mode, or by the host, in Code Slave mode. The JPEG engine writes the decoded video into the strip buffer in block format. The video interface reads the video out in raster format, executes the post-processing operations and outputs the video on the digital video bus.

When the ZR36060 is the code bus master (see Figure 32) it reads in the code as long as its $\overline{\text{CBUSY}}$ input is not asserted. Whenever the code FIFO is full the ZR36060 stops reading code. If the host controller is too slow and asserts $\overline{\text{CBUSY}}$ for too long, the code FIFO may become empty. In order to prevent underflow the ZR36060 stops writing data into the strip buffer. If this situation continues long enough, the strip buffer underflows, because the video unit keeps reading out video from the strip buffer, to keep up with the timing of the digital video bus. A strip buffer underflow is a data corruption event. At the system level this event may be prevented by two means: First, the system should be able to provide the code at the rate it is required by the ZR36060; second, some system configurations may have the capability to stop the video output stream when the strip buffer is close to underflow (16 pixels, or less, away from underflow). The ZR36060 indicates this “nearly empty” condition with its $\overline{\text{RTBSY}}$ output. One configuration for implementing this is if the ZR36060 is the master of the video syncs, and the system stops the video

using the PVALID input. This may be useful, for example, when decompressing still pictures.

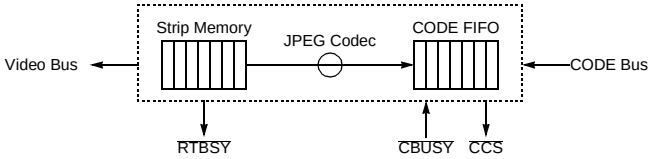


Figure 32. Data Flow in Decompression, Code Master Mode

When the code interface operates in slave mode (see Figure 33) the scenario is almost identical. The main difference is that CBUSY is now an output of the ZR36060, and it is used to indicate that the code FIFO is “nearly full”, thus the host must temporarily stop writing code until CBUSY indicates that the FIFO occupancy is below its threshold.

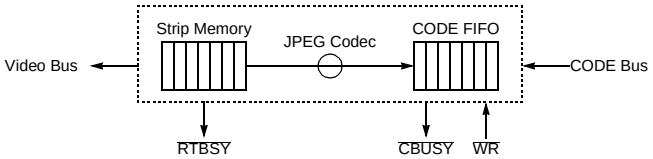


Figure 33. Data Flow in Decompression, Code Slave Mode

6.6 Compression and Decompression Modes

The ZR36060 has one decompression mode (called simply decompression), and four compression modes:

- Compression Pass
- Statistical Compression Pass
- Auto Two-Pass Compression
- Tables-Only Compression Pass

The following sections describe these modes.

6.7 Compression Pass

When the ZR36060 is in the IDLE state, and after the correct initialization has been done by the host (loading parameters and/or tables), it waits for the command to start compressing (assertion of START). RTBSY is not asserted at this time. Once the ZR36060 senses an active (low) START, it checks the level of FRAME, and then, if FRAME is active (Figure 34), it starts compressing the next odd field (i.e., at the next odd VSYNC).

Note: If FRAME is maintained active, and consequently will be detected active every time START is sampled active, the ZR36060 will compress only the odd fields. This is a convenient method for implementing field decimation by 2.

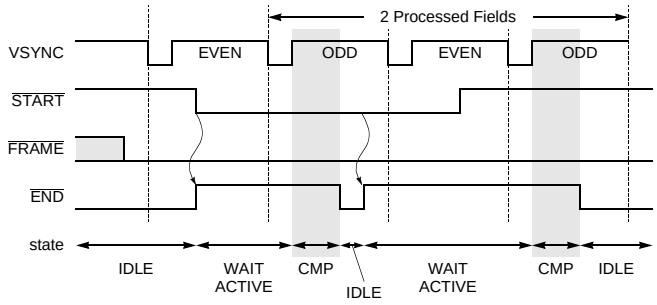


Figure 34. Compression With START and FRAME Continuously Asserted

If FRAME is not active when START is sampled (Figure 35), the ZR36060 always starts compressing the next field (i.e., at the next VSYNC).

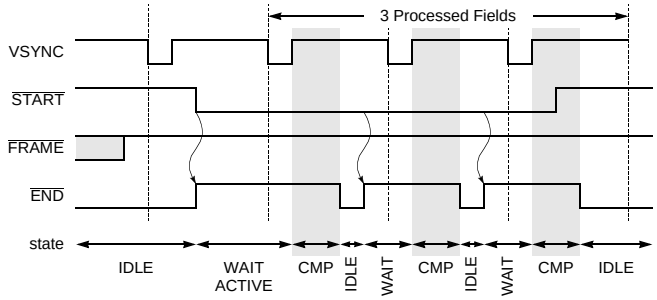


Figure 35. Compression With X START Continuously Asserted and FRAME Deasserted

The VSYNC edge used by the ZR36060 to make the decision on the ‘next’ field is configurable with the *FIVedge* parameter (leading or trailing edge).

During IDLE, CBUSY is active in code slave mode, preventing the host controller from reading code, since the code FIFO buffer is empty. Once the compression process starts, CBUSY is released and the host controller is expected to read the code. In code master mode, the ZR36060 drives out the code only after the compression process starts and the active area begins.

When the compression of the field is not stopped because of an interrupt request targeted to the host (IRQ assertion), then upon completion the ZR36060 asserts the END signal and returns to the IDLE state, looking again at START and FRAME. Note that it does not matter if the host controller continuously asserts START (and/or FRAME) or if it only asserts START (and/or FRAME) after the ZR36060 asserts END. The reason is that the ZR36060 interrogates START only when it is in the IDLE state. See Figure 36.

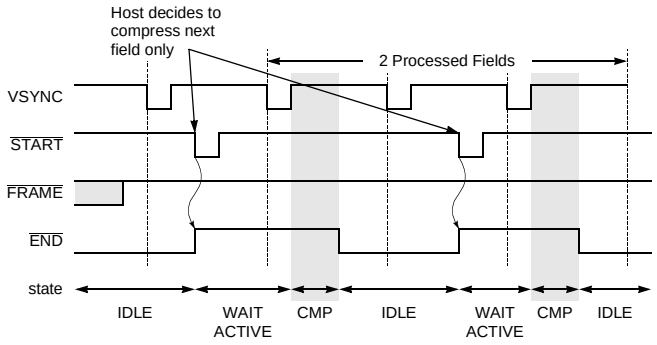


Figure 36. Compression with Pulsed START

The compression pass is always executed with Bit Rate Control (BRC). At the end of every compression process the ZR36060 writes the accumulated code volume (*ACV*) into its register. It also calculates a new iteration of the scale factor, based on the target and accumulated code volumes, and, if the *FSF* (Fixed Scale Factor) bit equals zero, it writes the new scale factor in the Scale Factor register. It uses this new value in the register as the scale factor for compressing the next image field. The host only has to program the initial scale factor to be used for compressing the first field. Otherwise, if FSF=1, the ZR36060 uses a fixed scale factor for all fields.

In a compression pass the ZR36060 does not update the Allocation Factor (AF). During encoding of each block the ZR36060 calculates a measure of the spatial "activity" in this block, denoted BACT. If the Block Accumulated Code Volume (BACV) exceeds the specified allocation (given by BACT*AF), or if it exceeds the Maximum Block Code Volume (MBCV), the code for the block is truncated accordingly. Note that both means of bit rate controlling (namely truncation due to the Allocation Factor, or truncation due to the MBCV) cannot be turned off directly, but their effects can be reduced to insignificance by setting MBCV and/or AF to their maximum values.

Note: Use of the Allocation Factor is only valid if the compression pass follows a Statistical Pass. If only Compression Passes are executed, the mode normally used for motion JPEG compression, the Allocation Factor *must* be set to its maximum value, otherwise the image will not be compressed correctly.

6.7.1 Data corruption during compression

If during the compression of a field the ZR36060 senses a data corruption event, it immediately asserts the DATERR output. However, the ZR36060 continues the process until it finishes compressing the field. At this time it asserts END and enters its IDLE state.

1) If *DATERR* is not enabled as an interrupt requesting event (i.e., it is cleared in the Interrupt Mask register), then the ZR36060 samples START again, and if START is asserted a new process begins.

2) If *DATERR* is enabled as an interrupt requesting event, JIRQ is asserted together with the assertion of the DATERR output, and when the ZR36060 completes the current process it enters the WAIT-ISR state and remains there, ignoring START, until the host reads the Interrupt Status register. At this time the ZR36060 goes back to its IDLE state and is again ready to start a new process, as soon as it samples START active.

In both cases, DATERR is deasserted at the beginning of the next process, i.e. simultaneously with the deassertion of END.

In compression the ZR36060 identifies a data corruption condition if:

a) the strip buffer overflows, or

b) the active edge of VSYNC (leading or trailing edge controlled by the *FIVedge* parameter) of the next video field arrives before END of the current field is asserted.

6.8 Statistical Compression Pass

In this mode the ZR36060 goes through all the calculations involved in encoding the input video, accumulating the code volume but without writing any code to the code FIFO. It does not assert CCS, COE, CWE as a code master, or CBUSY as a code slave.

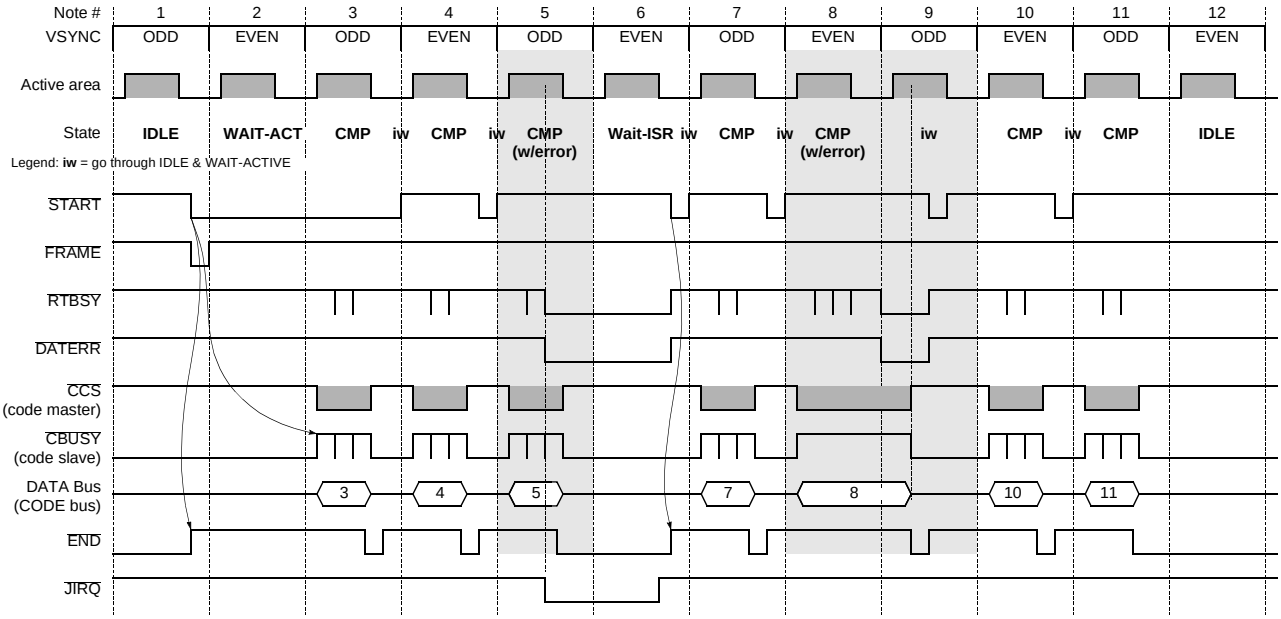
At the end of the process the ZR36060 calculates a new scale factor and allocation factor and writes them into their respective registers, and writes the accumulated *ACV* and *ACT* values into their respective registers. Then it asserts the END signal and returns to the IDLE state.

6.9 Auto Two-Pass Compression

In this mode the ZR36060 first executes a Statistical Compression Pass, then, without asserting END, it immediately starts a Compression Pass. Activation of START (and optionally FRAME) is required only for the first pass. The second pass follows immediately with the next VSYNC, without regard for START and FRAME. (Note that START and FRAME are only sampled when the ZR36060 is in IDLE, and in this mode the ZR36060 does not go into IDLE between the two passes). This mode only works correctly if exactly the same image data is fed to the ZR36060 in both passes.

6.10 Tables-Only Compression Pass

In this mode the ZR36060 produces only the "abbreviated format for table specification", that is, its code output contains no frame header, scan header, or Huffman-coded segments. The output includes the SOI marker, table marker segment(s), optional APP and/or COM marker segments, and the EOI marker. The process is activated by START (possibly with FRAME), and upon completion END is asserted and the ZR36060 returns to IDLE.



Notes:
1. In field #1, START & FRAME asserted indicate to begin on next ODD VSYNC
2. Wait for next ODD field; meanwhile START signal is ignored (END not asserted)
3. Field #3 is compressed, issue END and return to IDLE. At this point, START is sensed low and FRAME high, so compression starts at next VSYNC.
4. Field #4 is compressed. Again, after END the system asserts START but not FRAME, so the next field must be compressed too.
5. Begin field compression, but system is slow to read data out, so DATERR is asserted. END with DATERR informs the system of a corrupted field. Since interrupt request on data error is enabled, the ZR36060 asserts JIRQ and waits until the host acknowledges the JIRQ by reading the associated interrupt status register.
6. During this field, the host services the interrupt (ZR36060 de-asserts JIRQ) and asserts START to compress the next field. In this example, the host also disables interrupts after servicing this one.
7. Field #7 is compressed. After END, the host asserts START again.

8, 9. Field is compressed normally, but during the last lines of the active area, the system response is slow and the last part of the code is read out at a much slower rate. Next field (#9) begins (VSYNC is asserted) before the ZR36060 activates END. This is an error condition indicated by DATERR. No JIRQ here since interrupts are disabled. START is sensed low for the next field (the ZR36060 skips field #9).

10, 11. Fields are compressed normally. START is de-asserted during field #11, therefore after END the ZR36060 returns to IDLE and waits for next operation.

Figure 37. Examples of ZR36060 Compression (after the host loaded parameters and the Load command)

6.11 Decompression

Before starting decompression the ZR36060 is in the IDLE state. After the correct initialization has been done by the host (loading of parameters and/or tables), the ZR36060 is ready to receive a command to start decompression. The video bus already outputs the background color, and if the ZR36060 is a code slave, CBUSY is asserted, so the host cannot write compressed data to the ZR36060. After START is activated, the ZR36060 starts reading compressed data (if it is the code bus master), decoding it, and filling up the strip buffer with pixels. However, pixels will not start flowing out the video bus before the VSYNC that follows START (or the odd VSYNC, if FRAME was asserted together with START).

Host controllers that are capable of synchronizing the start command to VSYNC may do so, providing the start command as soon as possible after a VSYNC. This makes sure that once VSYNC arrives, the ZR36060 has enough pixels in the strip buffer to avoid a condition of strip buffer underflow. In systems where the video sync signals can be controlled by the host, this capability can be used to guarantee that START is asserted long enough before the next VSYNC.

Once START is asserted, CBUSY is deasserted (if it is configured as output, i.e., in code slave mode), and RTBSY is asserted indicating that the strip memory is initially (close to) empty (underflow). The host should now provide compressed data to the ZR36060 as quickly as possible, and fill the strip memory. Deassertion of RTBSY is an indication that sufficient data is available in the strip buffer to start video output. Every time that the ZR36060 senses that the strip buffer is close to an overflow (full), it asserts an internal flag that stops the transfer of pixels into the strip buffer, and eventually might result in assertion of CBUSY (in code slave mode) or in stopping of compressed data acquisition (in code master mode). When the ZR36060 senses the EOI marker, or when the active portion of the video field is over (whichever occurs later), it asserts the END signal and returns to the IDLE state, waiting for a new start command. (As in compression, the host controller may choose to leave START asserted continuously, rather than asserting it after every END).

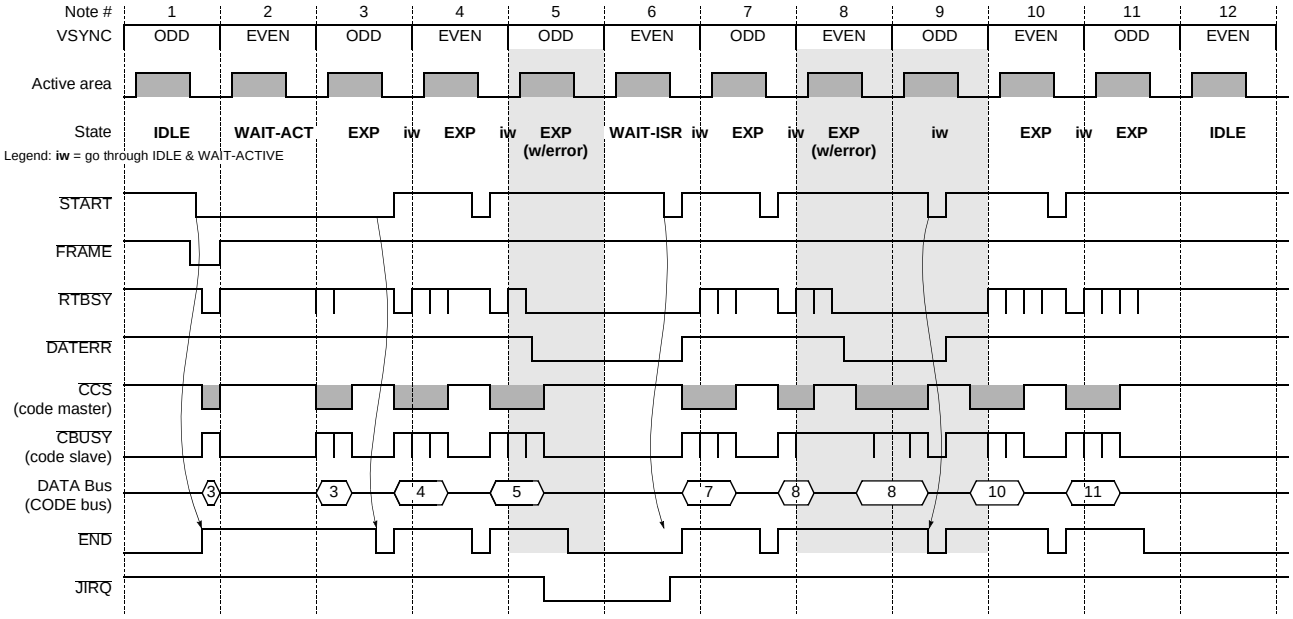
6.11.1 Data corruption during decompression

If during the decompression of a field the ZR36060 senses a data corruption event, it immediately asserts $\overline{\text{DATERR}}$, then continues the decompression process until it senses the EOI (End Of Image) marker or the end of the active video area (whichever occurs later). At this time it asserts $\overline{\text{END}}$ and enters the IDLE state. The deassertion of $\overline{\text{END}}$ (upon $\overline{\text{START}}$ for the next field) deasserts the $\overline{\text{DATERR}}$ signal.

1) If *DATERR* is not enabled as an interrupt requesting event (i.e., it is cleared in the Interrupt Mask register), then the ZR36060 samples $\overline{\text{START}}$ again, and if $\overline{\text{START}}$ is asserted a new process begins.

2) If *DATERR* is enabled as an interrupt requesting event, $\overline{\text{JIRQ}}$ is asserted together with the assertion of $\overline{\text{DATERR}}$, and when the ZR36060 completes the current process it enters the WAIT-ISR state and remains there, ignoring $\overline{\text{START}}$, until the host reads the Interrupt Status register. At this time the ZR36060 goes back to IDLE and is again ready to start a new process, waiting for $\overline{\text{START}}$.

In decompression the ZR36060 signals a data corruption condition if the strip buffer underflows.



Notes:
1, 2. In field #1, $\overline{\text{START}}$ & $\overline{\text{FRAME}}$ asserted instruct the ZR36060 to decompress on next ODD VSYNC (field #3). Code can be input immediately after $\overline{\text{START}}$ is asserted, to decode and fill with 8 lines of video the first strip buffer before the beginning of the active area. After the first strip is decompressed, code is no longer fetched, and the ZR36060 waits for the active area of the next ODD field.
3. Field #3 decompression continues and when completed, the ZR36060 issues $\overline{\text{END}}$ and returns to IDLE.
4. Field #4 is decompressed normally.
5. Begin field decompression, but the system is slow to feed new data, so $\overline{\text{DATERR}}$ is asserted. $\overline{\text{END}}$ with $\overline{\text{DATERR}}$ informs the system of a corrupted field. Since interrupt request on data error is enabled, the ZR36060 asserts $\overline{\text{JIRQ}}$ and waits until the host acknowledges the $\overline{\text{JIRQ}}$ by reading the associated interrupt status register.
6. During this field, the host services the interrupt, the ZR36060 de-asserts $\overline{\text{JIRQ}}$, and the host asserts $\overline{\text{START}}$ to decompress the next field. In this example, the host also disables interrupts after servicing this one.
7. Field #7 is decompressed. After $\overline{\text{END}}$, the system asserts $\overline{\text{START}}$ for the next field.
8, 9. In Field #8 the ZR36060 starts to input code, but somewhere during the active area, code is unavailable long enough that the ZR36060 issues a $\overline{\text{DATERR}}$ (no $\overline{\text{JIRQ}}$ here since interrupts are disabled). Nevertheless, the system continues to feed new code to the ZR36060 until $\overline{\text{END}}$ which occurs only after the VSYNC of the next field (#9). Now $\overline{\text{START}}$ is sensed low meaning that the next decompression will happen on field #10 (field #9 is skipped).
10, 11. Fields are decompressed normally. $\overline{\text{START}}$ is not asserted during field #11, therefore after $\overline{\text{END}}$ the ZR36060 returns to IDLE and waits for the next operation.

Figure 38. Examples of ZR36060 Decompression (after the host loaded parameters and the Load command)

7.0 POWER MANAGEMENT AND POWER-UP

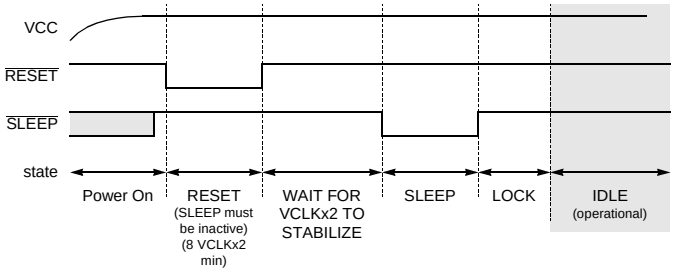
The ZR36060 has two power consumption modes: the normal mode, and a low-power mode, in the SLEEP state, entered by activating the $\overline{\text{SLEEP}}$ pin. This power saving is achieved by disabling the internal clocking to all flip-flops and gates, so this mode can be regarded as a “frozen” state of the ZR36060. All outputs retain their states, and bidirectional signals retain their directions.

Transitions to or from the SLEEP state must be done via the IDLE state. No host accesses are allowed in the SLEEP state, and during the IDLE - SLEEP transition. Otherwise the ZR36060 must be reset again.

After the $\overline{\text{SLEEP}}$ pin is de-activated, the ZR36060 is operational again, without the need for a reset, retaining all registers, markers and parameters previously loaded. But note: before $\overline{\text{START}}$ can be activated again for the next compression or decompression, the host must write the *Load* bit to reload the parameters and tables.

Deactivation of $\overline{\text{SLEEP}}$ also serves to initiate the coarse frequency lock phase of the internal PLL. For this reason, it is mandatory to pulse $\overline{\text{SLEEP}}$ at least once after power-up, when the system clock (VCLKx2) is stable (within 5% of its nominal frequency). The coarse lock must be re-initiated (by pulsing the $\overline{\text{SLEEP}}$ pin) each time the system changes the frequency of VCLKx2, for example if the video standard is changed. See Figure 39.

The coarse PLL frequency lock procedure takes 5000 VCLKx2 cycles, and is executed every low-to-high transition of $\overline{\text{SLEEP}}$. The ZR36060 remains in the SLEEP state during this time interval.



RESET Period: Minimum pulse width of 8 VCLKx2 cycles.
WAIT Period: wait for VCLKx2 to stabilize at the correct operating frequency.

SLEEP Pulse: Minimum pulse width of 8 VCLKx2 cycles.
LOCK Period: After $\overline{\text{SLEEP}}$ was deasserted, the system must wait 5,000 VCLKx2 cycles until the PLL is correctly locked to the clock frequency.
IDLE: The ZR36060 is ready for operation.

Figure 39. Power-Up Sequence and $\overline{\text{SLEEP}}$ Operation

8.0 REGISTER AND MEMORY DESCRIPTION

The ZR36060 internal memory space is implemented as 1024 bytes of RAM, accessible by the host controller through the host interface. The contents of this RAM may be loaded into the working storage registers and tables using the Load command (refer to 6.4 "Loading Parameters and Tables"). Figure 40 depicts the partitioning of the RAM.

A “default” is specified for each register bit. This is its value after reset.

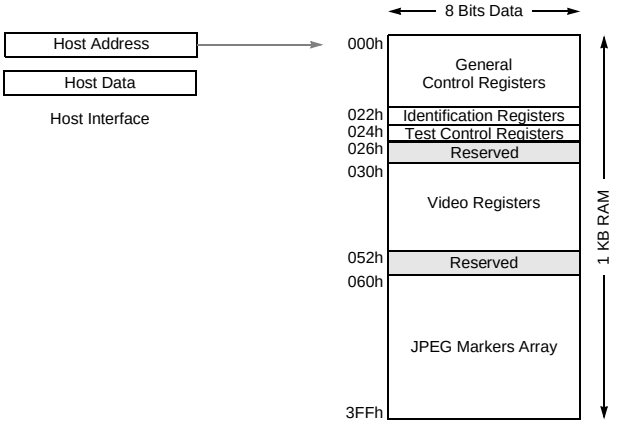


Figure 40. Internal Memory Map

8.1 General Control Registers

Load Parameters Register **Address 0x000**

	7	6	5	4	3	2	1	0
0x000	Load	0						SyncRst
type	RW							RW
default	0	X	X	X	X	X	X	0

SyncRst: Resets the video sync generator.

0 - No reset

1 - Resets the sync generator. Starts horizontal and vertical counting from 0. The sync generator is maintained in the reset state until the host writes 0 to this bit.

Load: Loads all the internal memory parameters (including tables) to the respective internal working stores. After writing *Load* = 1, the ZR36060 sets the *Busy* bit until the load is complete and the ZR36060 is ready to operate. The *Load* bit must be set before starting an operation every time a new parameter is written anywhere in the internal memory space.

0 - No effect

1 - Load parameters now

Code FIFO Status Register (Read only) **Address 0x001**

	7	6	5	4	3	2	1	0
0x001	Busy	X				CBUSY	CFIFO1	CFIFO0
type	R					R	R	R
default	0	X	X	X	X	1	0	0

CFIFO1:0 : Indicates the fullness of the Code FIFO:

00 - less than 1/4 full.

01 - 1/4 or more but less than 1/2 full.

10 - 1/2 or more but less than 3/4 full.

11 - 3/4 full or more.

These bits are valid only when $\overline{EOI}$ is not asserted.

CBUSY: Indicates the full/empty condition of the code FIFO. Identical to the state of the $\overline{CBUSY}$ pin, but with inverse logic.

0 - code FIFO not full / not empty

1 - code FIFO full / empty

This bit is valid only when $\overline{CBUSY}$ is an output; i.e.- in code slave mode.

Busy: Status of the Load operation. The host must poll this bit to determine when the ZR36060 is ready (not Busy) to load a new parameter set or start a compression/decompression process.

0 - ZR36060 is ready to operate, no Load operation in progress

1 - Load is in progress now, do not access any internal memory location while this bit is set

Code Interface Register **Address 0x002**

	7	6	5	4	3	2	1	0
0x002	Code16	Endian	0			CFIS	0	CodeMstr
type	RW	RW				RW		RW
default	X	X	X	X	X	X	X	1

CodeMstr: Selects Code Master or Slave mode.

0 - Code Slave mode

1 - Code Master mode

CFIS: Only in Master mode, defines the number of clocks for each code byte transfer. Must be 0 in Code Slave mode.

0 - one VCLKx2 cycle

1 - two VCLKx2 cycles

Endian: Defines the byte ordering when using 16-bit code slave interface. Must be 0 in Master mode or 8-bit Slave mode.

0 - first byte (‘FF’ of ‘FFD8’ SOI code) is on the DATA bus

1 - first byte (‘FF’ of ‘FFD8’ SOI code) is on the CODE bus

Code16: Defines the code bus width for slave mode only. Must be 0 in Master mode.

0 - 8-bit code bus

1 - 16-bit code bus

Codec Mode Register — Address 0x003

	7	6	5	4	3	2	1	0
0x003	COMP	ATP	PASS2	TLM	0	BRC	FSF	0
type	RW	RW	RW	RW		RW	RW	
default	1	X	X	X	X	X	X	X

11000100 - Auto Two-Pass Compression
10000100 - Statistical Compression Pass
10100100 - Compression Pass with Variable Scale Factor
10100110 - Compression Pass with Fixed Scale Factor
10110000 - Tables-Only Compression Pass
00000000 - Decompression Pass
All other combinations of the register bits are illegal

Reserved — Address 0x004

	7	6	5	4	3	2	1	0
0x004	0	0	0	0	0	0	0	0
type								
default	X	X	X	X	X	X	X	X

Must be programmed to 0x00 for correct operation.

Maximum Block Code Volume Register — Address 0x005

	7	6	5	4	3	2	1	0
0x005	MBCV							
type	RW							
default	X							

MBCV: Maximum Block Code Volume. In all compression modes, MBCV imposes an upper limit on the number of bits that are used to encode each 8x8 block of samples.

The number of bits is twice the value coded in this register. MBCV=01 represents two bits per block, and MBCV=FF represents 510 bits per block.

Markers Enable Register — Address 0x006

	7	6	5	4	3	2	1	0
0x006	APP	COM	DRI	DQT	DHT	0	0	0
type	RW	RW	RW	RW	RW			
default	X	X	X	X	X	X	X	X

In compression, this register specifies which of the optional marker segments to include in the compressed data. Not used in decompression.

If a bit is programmed to 0, the respective marker segment is not included. If a bit is programmed to 1, the ZR36060 performs the following action:

APP: Reads the Application segment from the Internal Memory and writes it to the compressed data during the Compression Pass. Also in Tables-only pass.

COM: Reads the Comment segment from the Internal Memory and writes it to the compressed data during the Compression Pass. Also in Tables-only pass.

DRI: Define Restart Interval. Enables the restart mechanism and writes the DRI marker segment to the compressed data during the Compression Pass. When the restart interval is zero, the restart function is disabled.

DQT: Define Quantization Tables. Reads the base Quantization Tables defined in the DQT segment in the Internal Memory, multiplies the quantization values by the Scale Factor (SF), rounds to eight bits and writes the results together with the DQT marker and parameters in the compressed data during the Compression Pass or the Tables-Only Pass. The number of Quantization Tables to be processed is inferred from the LEN (segment length) parameter of the DQT segment. Note: the identical scaled tables are used to compress the data.

DHT: Define Huffman Tables. Reads the Huffman Tables defined in the DHT segment of the Internal Memory, and writes the DHT segment in the compressed data during the Compression Pass or the Tables-only Pass.

Interrupt Mask Register — Address 0x007

	7	6	5	4	3	2	1	0
0x007	0	0	0	0	EOAV	EOI	END	DATERR
type					RW	RW	RW	RW
default	X	X	X	X	X	X	X	X

DATERR: Enable interrupt on assertion of $\overline{\text{DATERR}}$ during a process
0 - Interrupt disabled
1 - Interrupt enabled

END: Enable interrupt on assertion of $\overline{\text{END}}$ at the end of a process
0 - Interrupt disabled
1 - Interrupt enabled

EOI: Enable interrupt when $\overline{\text{EOI}}$ is asserted, i.e., when the EOI marker is being read or written at the code interface
0 - Interrupt disabled
1 - Interrupt enabled

EOAV: Enable interrupt on End-Of-Active-Video area during the process
0 - Interrupt disabled
1 - Interrupt enabled

Interrupt Status Register (Read Only) — Address 0x008

	7	6	5	4	3	2	1	0
0x008	ProCnt1	ProCnt0	X	X	EOAV	EOI	END	DATERR
type	R	R			R	R	R	R
default	0	0	X	X	0	0	1	0

DATERR: Status of the $\overline{\text{DATERR}}$ output pin
0 - $\overline{\text{DATERR}}$ is not asserted (normal operation)
1 - $\overline{\text{DATERR}}$ is asserted (data corruption)

END: Status of the $\overline{\text{END}}$ output pin
0 - $\overline{\text{END}}$ is not asserted (process in progress)
1 - $\overline{\text{END}}$ is asserted (process ended and in IDLE state)

EOI: Status of the $\overline{\text{EOI}}$ output pin
0 - an EOI marker event did not occur
1 - an EOI marker has been read or written by the host

EOAV: Latch an event upon End-Of-Active-Video area during the process
0 - an EOAV event did not occur (video is still being input or output)
1 - an EOAV event occurred (active area of video has finished)

ProCnt1:0 : 2-bit cyclic process (compression or decompression) counter. It is incremented on $\overline{\text{START}}$ of each field process. Note: ProCnt appears here only as a status, it is not an interrupt-generating event.

Target Net Code Volume Register — Address 0x009 - 0x00C

	7	6	5	4	3	2	1	0
0x009	TCV_NET31:24							
0x00A	TCV_NET23:16							
0x00B	TCV_NET15:8							
0x00C	TCV_NET7:0							
type	RW							
default	X							

TCV_NET31:0: Target Net Code Volume. Must be programmed by the host for a Compression Pass and Auto Two-Pass compression. TCV_NET is used by the ZR36060 in the calculation of the new Scale Factor (SF) at the end of the Compression Pass or the second pass of the Auto Two-Pass compression. It is the Target Code Volume in bits for the compressed data excluding the marker segments.

Target Data Code Volume Register — Address 0x00D - 0x010

	7	6	5	4	3	2	1	0
0x00D	TCV_DATA31:24							
0x00E	TCV_DATA23:16							
0x00F	TCV_DATA15:8							
0x010	TCV_DATA7:0							
type	RW							
default	X							

TCV_DATA31:0: Target Data Code Volume. Required by the ZR36060 only for Auto Two-Pass compression and for a Statistical Pass. TCV_DATA is used by the ZR36060 to calculate the new Scale Factor (SF) and Allocation Factor (AF) after the Statistical Pass. It is the Target Code Volume in bits for the compressed data excluding the marker segments, the EOB (end-of-block) Huffman codes, the byte stuffing, and the bit stuffing (which completes the last data byte). Note: byte stuffing typically represents about 1% of the code volume.

Scale Factor Register — Address 0x011 - 0x012

	7	6	5	4	3	2	1	0
0x011	SF15:8 (integer part of SF)							
0x012	SF7:0 (fractional part of SF)							
type	RW							
default	X							

SF15:0: Scale Factor. It is used for scaling the quantization table values. SF should be provided to the ZR36060 as a parameter at the beginning of every compression operation. If Variable SF option is used, this register is internally computed and updated at the end of every compression pass in order to converge to the desired TCV. SF is a 16-bit fixed point binary number, with 8 bits after the binary point.

Allocation Factor Register — Address 0x013 - 0x015

	7	6	5	4	3	2	1	0
0x013	AF23:16							
0x014	AF15:8							
0x015	AF7:0							
type	RW							
default	X							

AF23:0: Allocation Factor. AF is used in the bit rate control to compute the Allocated Code Volume for each block. The AF is computed by the ZR36060 and written to the AF register at the end of a Statistical Pass. This value can later be used in a Compression Pass; otherwise for a Compression Pass without a prior Statistical Pass, AF must be programmed to 0xFFFFFF. AF is a 24-bit fixed point binary number, with 19 bits after the binary point.

Accumulated Code Volume Register — Address 0x016 - 0x019

	7	6	5	4	3	2	1	0
0x016	ACV31:24							
0x017	ACV23:16							
0x018	ACV15:8							
0x019	ACV7:0							
type	R							
default	X							

ACV31:0: Accumulated Code Volume. A 32-bit binary integer.

ACV is written by the ZR36060, and can be read by the host at the end of a compression operation. It has two different interpretations, depending on the compression operation:

(1) ACV_DATA: the Code Volume in bits excluding marker segments, EOB codes and bit and byte stuffing at the completion of a Statistical Pass.

(2) ACV_NET: the Net Code Volume in bits excluding marker segments at the completion of every Compression Pass.

ACV_NET / 8 = number of bytes passing through the Code FIFO excluding marker segments. It does not include the padding bytes for double-word alignment.

Accumulated Total Activity Register — Address 0x01A - 0x01D

	7	6	5	4	3	2	1	0
0x01A	ACT31:24							
0x01B	ACT23:16							
0x01C	ACT15:8							
0x01D	ACT7:0							
type	R							
default	X							

ACT31:0: Accumulated Total Activity of the image, a measure used for internal bit rate control calculations. ACT is written here by the ZR36060 at the end of Statistical Pass and Auto Two Pass compression modes. It is a 32-bit binary integer.

Accumulated Truncated Bits Register — Address 0x01E - 0x021

	7	6	5	4	3	2	1	0
0x01E	ACV_TRUN31:24							
0x01F	ACV_TRUN23:16							
0x020	ACV_TRUN15:8							
0x021	ACV_TRUN7:0							
type	R							
default	X							

ACV_TRUN31:0: Total number of bits truncated as a result of block truncation, a measure used for internal bit rate control calculations in Compression Pass and Auto Two-Pass modes. ACV_TRUN is written here by the ZR36060 at the end of these modes. It is a 32-bit binary integer.

8.2 ID and Testing Registers

Identification Registers (Read Only) — Address 0x022 - 0x023

	7	6	5	4	3	2	1	0
0x022	DeviceID							
type	R							
default	0	0	1	1	0	0	1	1
0x023	Revision							
type	R							
default	0	0	0	0	0	0	0	1

DeviceID: Hardwired to the chip device ID number (0x33).

Revision: Hardwired to the current chip revision number (0x01).

Test Control Registers — Address 0x024 - 0x025

	7	6	5	4	3	2	1	0
0x024	0							
0x025	0							
type	–							
default	X							

Reserved: Reserved for test mode. Must be initialized to 0x00 for correct operation.

8.3 Video Registers

Video Control Register — Address 0x030

	7	6	5	4	3	2	1	0
0x030	Video8	Range	0	0	FIDet	FIVedge	FIExt	SyncMstr
type	RW	RW			RW	RW	RW	RW
default	X	X	X	X	X	X	X	0

SyncMstr: Configures the ZR36060 as a video sync Master or Slave.
0 - Video sync Slave
1 - Video sync Master

FIExt: Field detection by external pin or decoding from HSYNC and VSYNC.
0 - Even/odd field detection by latching HSYNC with VSYNC
1 - Even/odd field detection by means of the dedicated FI pin

FIVedge: Defines the start of a video field at the leading or trailing edge of VSYNC (affects the reset point for the vertical counters, the FI signal state change, the next field decision after $\overline{\text{START}}$, and $\overline{\text{DATERR}}$ assertion when VSYNC arrives before the end of a process).
0 - Leading edge of VSYNC
1 - Trailing edge of VSYNC

FIDet: Interpretation of detected field type (detection controlled by FIExt).
0 - ODD fields: FI is low, or VSYNC latches the HSYNC pulse
1 - ODD fields: FI is high, or VSYNC latches the middle of a line

Range: Defines the full-scale range of the video bus pixels in decompression. Has no effect in compression.
0 - Pixel values are full-scale with 256 levels.
1 - Pixel values limited to the range [16,235] (per CCIR 601.)

Video8: Defines the video bus width.
0 - 16-bit video bus
1 - 8-bit video bus

Video Polarity Register — Address 0x031

	7	6	5	4	3	2	1	0
0x031	VCLKPol	PValPol	PoePol	SImgPol	BLPol	FIPol	HSPol	VSPol
type	RW	RW	RW	RW	RW	RW	RW	RW
default	X	X	X	X	X	X	X	X

VSPol: Polarity of the VSYNC signal (note that this parameter is completely independent of the FIVedge parameter)
0 - Sync pulse is active low
1 - Sync pulse is active high

HSPol: Polarity of the HSYNC signal
0 - Sync pulse is active low
1 - Sync pulse is active high

FIPol: Polarity of the Field Identification signal
0 - FI is low during ODD fields
1 - FI is high during ODD fields

BLPol: Polarity of the BLANK signal
0 - BLANK area is active low
1 - BLANK area is active high

SImgPol: Polarity of the SUBIMG signal
0 - SUBIMG is low before *SVStart*, *SHStart* and after *SVEnd*, *SHEnd*
1 - SUBIMG is high before *SVStart*, *SHStart* and after *SVEnd*, *SHEnd*

PoePol: Polarity of the POE signal to permit floating (disabling) of the ZR36060 video bus during decompression:
0 - Disable bus when POE is low
1 - Disable bus when POE is high

PValPol: Polarity of the PVALID signal
0 - Pixels are valid when PVALID is low
1 - Pixels are valid when PVALID is high

VCLKPol: Polarity of the VCLK signal (relevant in 16-bit video width only)
0 - Pixels are valid when VCLK is low
1 - Pixels are valid when VCLK is high

Scaling Register — Address: 0x032

	7	6	5	4	3	2	1	0
0x032	0	0	0	0	0	VScale	HScale	
type						RW	RW	RW
default	X	X	X	X	X	X	X	X

HScale: Horizontal down or up scaling (in compression or decompression)
00b - No scaling
01b - 2:1 scaling ratio, with horizontal filtering
10b - 4:1 scaling ratio, with horizontal filtering
11b - Not used

VScale: Vertical down or up scaling (in compression or decompression)
0 - No scaling
1 - In compression, only even indexed lines (0,2,..) are processed. In decompression, lines are duplicated

Background Color Registers — Address: 0x033 - 0x035

	7	6	5	4	3	2	1	0
0x033	BackY7:0							
0x034	BackU7:0							
0x035	BackV7:0							
type	RW							
default	X							

BackX: Y, U, V components for the background color (used only in decompression)

Sync Generator Registers — Address: 0x036 - 0x041

	7	6	5	4	3	2	1	0
0x036	Vtotal 15:8							
0x037	Vtotal 7:0							
0x038	Htotal 9:8							
0x039	Htotal 7:0							
0x03A	VsyncSize 7:0							
0x03B	HsyncSize 7:0							
0x03C	BVstart 7:0							
0x03D	BHstart 7:0							
0x03E	BVend 15:8							
0x03F	BVend 7:0							
0x040	BHend 9:8							
0x041	BHend 7:0							
type	RW							
default	X							

Parameters used by the internal video sync generator when it is in Master mode (*SyncMstr*=1).

Horizontal measures are in number of VCLKs (1 VCLK = 1 pixel, regardless of the video bus width), from the leading edge of HSYNC.

Vertical measures are in number of HSYNCs (1 HSYNC = 1 line), from the leading or trailing edge of VSYNC as specified by the *FIVedge* parameter.

Vtotal 15:0: Number of lines per frame. Writing N indicates that the frame has N+1 lines. (e.g. - *Vtotal* = 524, for NTSC, 525 lines per frame)

Maximum permitted value is 65535.

Htotal 9:0: Number of pixel periods (VCLKs) per line. Writing N indicates that a line has N+1 VCLKs. (e.g. - Htotal = 857, for NTSC-CCIR, 858 pixels per line)

Maximum permitted value is 1023.

VsyncSize 7:0: Length of VSYNC pulse measured in lines. Writing N indicates that the sync pulse width is N+1 lines. (e.g. - *VsyncSize* = 5, for a 6-line vertical sync interval)

HsyncSize 7:0: Length of HSYNC pulse measured in VCLKs (pixels). Writing N indicates that the sync pulse width is N+1 pixels. (e.g. - *HsyncSize* = 31, for a 32-pixel horizontal sync interval)

BVstart 7:0: Length from active VSYNC edge to first non-BLANK line measured in lines. Writing N indicates that the first non-BLANK line is line N+1. (e.g. - *BVstart* = 11, to have the first non-BLANK line on line number 12)

BHstart 7:0: Length from HSYNC leading edge to first non-BLANK pixel measured in pixels (VCLKs). Writing N indicates that the first non-BLANK pixel is pixel number N+1. (e.g. - *BHstart* = 99, to have the first non-BLANK pixel on VCLK number 100)

BVend 15:0: Length from active VSYNC edge to last non-BLANK line measured in lines. Writing N indicates that the last non-BLANK line is line N. (e.g. - *BVend* = 241, to have the last non-BLANK line on line number 241)

BHend 9:0: Length from HSYNC leading edge to last non-BLANK pixel measured in pixels (VCLKs). Writing N indicates that the last non-BLANK pixel is pixel number N. (e.g. - *BHend* = 720, to have the last non-BLANK pixel on VCLK number 720)

Active Area Registers — Address: 0x042 - 0x049

	7	6	5	4	3	2	1	0
0x042	Vstart 15:8							
0x043	Vstart 7:0							
0x044	Vend 15:8							
0x045	Vend 7:0							
0x046	Hstart 9:8							
0x047	Hstart 7:0							
0x048	Hend 9:8							
0x049	Hend 7:0							
type	RW							
default	X							

Parameters that define the active area rectangle of the processed video.

Horizontal measures are in number of VCLKs (1 VCLK = 1 pixel, regardless of the video bus width), from the leading edge of HSYNC.

Vertical measures are in number of HSYNCs (1 HSYNC = 1 line), from the leading or trailing edge of VSYNC as specified by the *FIVedge* parameter.

Vstart 15:0: Length from VSYNC edge to first active (processed) line measured in lines. Writing N indicates that the first active line is line N+1. (e.g. - *Vstart* = 11, to have the first active line on line number 12).

Vstart must be ≥ 2.

Vend 15:0: Length from VSYNC edge to last active (processed) line measured in lines. Writing N indicates that the last line is line N. (e.g. - *Vend* = 241, to have the last line on line number 241).

Maximum permitted value for (*Vend* - *Vstart*) is 32768.

Vend must be ≤ (N_{TF} - 10), where N_{TF} is the total number of lines *per field.*

(N_{TF}=262 for NTSC, 312 for PAL)

Hstart 9:0: Length from HSYNC leading edge to first active (processed) pixel measured in pixels. Writing N indicates that the first active pixel is pixel number N+1. (e.g. - *Hstart* = 99, to have the first active pixel on VCLK number 100).

Hstart must be ≥ 8 in compression, and must be ≥ 12 in decompression.

Hend 9:0: Length from HSYNC leading edge to last active (processed) pixel measured in pixels. Writing N indicates that the last active pixel is pixel number N. (e.g. - Hend = 720, to have the last active pixel on VCLK number 720).

Maximum permitted value for (*Hend* - *Hstart*) is 768.

Note: The active video area must not overlap the VSYNC pulse. In other words, the active area must always be contained between the trailing edge of VSYNC and the next leading edge (see Figure 7).

Sub-image Window Registers **Address: 0x04A - 0x051**

	7	6	5	4	3	2	1	0
0x04A	SVstart 15:8							
0x04B	SVstart 7:0							
0x04C	SVend 15:8							
0x04D	SVend 7:0							
0x04E	SHstart 9:8							
0x04F	SHstart 7:0							
0x050	SHend 9:8							
0x051	SHend 7:0							
type	RW							
default	X							

Parameters that define the sub-image rectangle of the video.

Horizontal measures are in number of VCLKs (1 VCLK = 1 pixel, regardless of the video bus width), from the leading edge of HSYNC.

Vertical measures are in number of HSYNCs (1 HSYNC = 1 line), from the leading or trailing edge of VSYNC as specified by the *FIVedge* parameter.

SVstart 15:0: Length from active VSYNC edge to first sub-image line measured in lines. Writing N indicates that the first sub-image line is line N+1. (e.g. - *SVstart* = 11, to have the first sub-image line on line number 12).

SVend 15:0: Length from active VSYNC edge to last sub-image line measured in lines. Writing N indicates that the last line is line N. (e.g. - *SVend* = 241, to have the last line on line number 241).

Maximum permitted value for (*SVend* - *SVstart*) is 32768.

SHstart 9:0: Length from HSYNC leading edge to first sub-image pixel measured in pixels. Writing N indicates that the first sub-image pixel is pixel number N+1. (e.g. - *SHstart* = 99, to have the first sub-image pixel on VCLK number 100).

SHend 9:0: Length from HSYNC leading edge to last sub-image pixel measured in pixels. Writing N indicates that the last sub-image pixel is pixel number N. (e.g. - *SHend* = 720, to have the last sub-image pixel on VCLK number 720).

Maximum permitted value for (*SHend* - *SHstart*) is 1023.

8.4 JPEG Marker Segments

Table 4 shows the mapping of all JPEG markers in the ZR36060 internal memory. The Value column contains common values (hexadecimal) used for the YUV 4:2:2 format images supported by the ZR36060. There are no default values, therefore the user must pre-load the correct values before operating the device.

In compression, the SOF0 and SOS marker segments are always copied from the memory into the compressed image bitstream header. The other marker segments are optional, and their inclusion in the bitstream is controlled by the Markers Enable register.

In decompression, the marker segments contained in the compressed bitstream are automatically copied by the ZR36060 into the internal memory at the appropriate starting locations. At the end of a decompression, the host can read them if so desired. Note that if a compressed image contains more than one marker of the same type, only the last one will be contained in the memory at the end of the process. For example, if there are two DHT segments, the first defining the DC Huffman tables, and the second defining the AC Huffman tables, only the AC tables can be read from memory at the end of the process. Of course, all the tables will be used as required for the decompression.

The marker segments in the Internal Memory have exactly the same syntax as the marker segments specified in the JPEG standard. These segments are: SOF, SOS, DRI, DQT, DHT, APP, and COM. The SOI, EOI and RST markers are supported automatically by the ZR36060.

Before starting to compress a sequence of images, the host must program the appropriate marker segments in the internal memory, and issue a Load command to load them. Only the markers actually used need to be programmed: the required markers SOF and SOS, and whichever optional markers are specified in the Markers Enable register. Note that the starting location of each marker segment in the internal memory is fixed, but the length and content of a marker segment are variable.

The host does not have to program any of the marker segments in order to decompress an image bitstream that contains all the necessary tables. If the bitstream is in an abbreviated format and lacks one or more tables, the host must preload them, by programming the appropriate tables in the internal memory, and issuing a Load command before starting to decompress a sequence of images that uses the same tables for all the images.

Table 4: ZR36060 JPEG Marker Segments

Address	Content	Value	Description
060	SOF0	FF	Start Of Frame marker (FFC0). This segment contains 17 bytes that define the 3 components (Y,U,V) of the MCU, and the quantization table to be used by each component.
061	SOF0	C0	
062	LEN_H	00	Length of this segment (without the marker)
063	LEN_L	11	
064	P	8	Precision (8 bits)
065	Y_H		Number of lines in the active area (must always equal *Vend* - *Vstart*)
066	Y_L		
067	X_H		Number of pixels in the active area (must always equal *Hend* - *Hstart*)
068	X_L		
069	Nf	3	Number of Color Components (YUV = 3 components)
06A	CY	0	ID for the Y Component
06B	HY,VY	21	Number of appearances of Y in MCU, horizontally and vertically
06C	TqY	0	Quantization table ID for Y
06D	CU	1	ID for the U Component
06E	HU,VU	11	Number of appearances of U in MCU, horizontally and vertically
06F	TqU	1	Quantization table ID for U
070	CV	2	ID for the V Component
071	HV,VV	11	Number of appearances of V in MCU, horizontally and vertically
072	TqV	1	Quantization table ID for V
073-079	...		Unused
07A	SOS	FF	Start Of Scan marker (SOS). Specifies the Huffman table ID to use with each color component.
07B	SOS	DA	

Table 4: ZR36060 JPEG Marker Segments

Address	Content	Value	Description
07C	LEN_H	00	Length of this segment (without the marker)
07D	LEN_L	C	
07E	Ns	3	Number of Components in this scan
07F	CY	0	ID for the Y Component
080	TYd,TYa	00	Huffman DC, AC table selections for Y Component
081	CU	1	ID for the U Component
082	TUd,TUa	11	Huffman DC, AC table selections for U Component
083	CV	2	ID for the V Component
084	TVd,TVa	11	Huffman DC, AC table selections for V Component
085		00	Constant
086		3F	Constant
087		00	Constant
088-0BF	.....		Unused
0C0	DRI	FF	Define Restart Interval.
0C1	DRI	DD	
0C2	LEN	00	Length of this segment (without the marker)
0C3	LEN	04	
0C4	RI_H	0	Length of Restart Interval in MCU units.
0C5	RI_L	8	
0C6-0CB	.....		Unused
0CC	DQT	FF	Define Quantization Tables.
0CD	DQT	DB	
0CE	LEN	0	Typical DQT segment length, with two tables
0CF	LEN	84	
0D0...		Typical DQT segment data (2 tables): 00 10 0B 0C 0E 0C 0A 10 OE OD OE 12 11 10 13 18 28 1A 18 16 16 18 31 23 25 1D 28 3A 33 3D 3C 39 33 38 37 40 48 5C 4E 40 44 57 45 37 38 50 6D 51 57 5F 62 67 68 67 3E 4D 71 79 70 64 78 5C 65 67 63 01 11 12 12 18 15 18 2F 1A 1A 2F 63 42 38 42 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63	
....	....		65 more bytes for optional third table, the remainder unused
1D4	DHT	FF	Define Huffman Tables.
1D5	DHT	C4	
1D6	LEN	1	Typical DHT segment length, with 2 DC and 2 AC tables
1D7	LEN	A2	
1D8...		Typical DHT segment data (2 DC and 2 AC): 00 00 01 05 01 01 01 01 01 01 00 00 00 00 00 00 00 00 01 02 03 04 05 06 07 08 09 0A 0B 01 00 03 01 01 01 01 01 01 01 01 01 00 00 00 00 00 00 01 02 03 04 05 06 07 08 09 0A 0B 10 00 02 01 03 03 02 04 03 05 05 04 04 00 00 01 7D 01 02 03 00 04 11 05 12 21 31 41 06 13 51 61 07 22 71 14 32 81 91 A1 08 23 42 B1 C1 15 52 D1 F0 24 33 62 72 82 09 0A 16 17 18 19 1A 25 26 27 28 29 2A 34 35 36 37 38 39 3A 43 44 45 46 47 48 49 4A 53 54 55 56 57 58 59 5A 63 64 65 66 67 68 69 6A 73 74 75 76 77 78 79 7A 83 84 85 86 87 88 89 8A 92 93 94 95 96 97 98 99 9A A2 A3 A4 A5 A6 A7 A8 A9 AA B2 B3 B4 B5 B6 B7 B8 B9 BA C2 C3 C4 C5 C6 C7 C8 C9 CA D2 D3 D4 D5 D6 D7 D8 D9 DA E1 E2 E3 E4 E5 E6 E7 E8 E9 EA F1 F2 F3 F4 F5 F6 F7 F8 F9 FA 11 00 02 01 02 04 04 03 04 07 05 04 04 00 01 02 77 00 01 02 03 11 04 05 21 31 06 12 41 51 07 61 71 13 22 32 81 08 14 42 91 A1 B1 C1 09 23 33 52 F0 15 62 72 D1 0A 16 24 34 E1 25 F1 17 18 19 1A 26 27 28 29 2A 35 36 37 38 39 3A 43 44 45 46 47 48 49 4A 53 54 55 56 57 58 59 5A 63 64 65 66 67 68 69 6A 73 74 75 76 77 78 79 7A 82 83 84 85 86 87 88 89 8A 92 93 94 95 96 97 98 99 9A A2 A3 A4 A5 A6 A7 A8 A9 AA B2 B3 B4 B5 B6 B7 B8 B9 BA C2 C3 C4 C5 C6 C7 C8 C9 CA D2 D3 D4 D5 D6 D7 D8 D9 DA E2 E3 E4 E5 E6 E7 E8 E9 EA F2 F3 F4 F5 F6 F7 F8 F9 FA	
....	....		Unused
380	APP	FF	Application marker segment. Limited to 64 bytes maximum length including the marker
381	APP	E0	Can be En, n=0,...,F
382	LEN_H	00	Maximum length of 62 bytes
383	LEN_L	3E	
...	....		Contents of the APP segment
3C0	COM	FF	Comment marker segment. Limited to 64 bytes maximum length including the marker
3C1	COM	FE	In compression, if this byte is programmed to En instead of FE, it is possible to include a second APP segment in the bitstream instead of a COM segment
3C2	LEN_H	00	Maximum length of 62 bytes
3C3	LEN_L	3E	
...3FF	....		Contents of the COM segment
			(End of ZR36060 internal memory)

9.0 ELECTRICAL CHARACTERISTICS

ABSOLUTE MAXIMUM RATINGS

Storage Temperature .....-65°C to +150°C
Supply Voltage (V_{DD}).....-0.5 V to +5.0 V
DC Output Voltage ..... -0.5 V to V_{DD}+0.5
DC Input Voltage.....-0.5 V to +6.0 V

DC Output Current..... +-20 mA/output, Max. 200 mA
DC Input Current.....-30 mA to +5 mA

Note: Stresses above these values may cause permanent device failure. Functionality at or above these limits is not implied. Exposure to absolute maximum ratings for extended periods may affect device reliability.

OPERATING RANGE

Temperature.....0°C to +70°C

Supply Voltage ..... 3.3 V ±5%

DC CHARACTERISTICS

Table 5: Input Characteristics

Symbol	Parameter	Min	Max	Unit	Test Conditions
V_{IL}	Input Voltage Low	-0.3	0.8	V	
V_{IH}	Input Voltage High	2.0	5.5	V	
I_{LI}	Input leakage current		±10	µA	
C_{IN}	Input Capacitance		10	pF	
I_{CC}	Power Supply Current		230 250	mA mA	ZR36060-27 @ 27 MHz VCLKx2 ZR36060-29.5 @ 29.5 Mhz VCLKx2
I_{SC}	Stand-by (SLEEP) current		10	mA	

Table 6: Output Characteristics

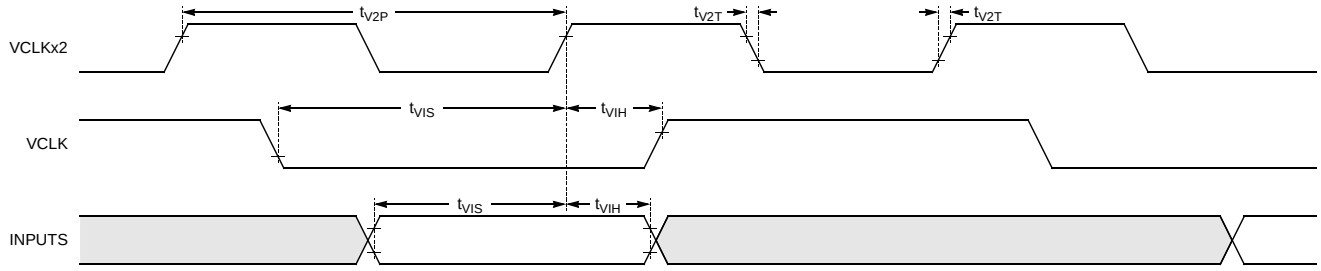
Symbol	Parameter	Min	Max	Unit	Test Conditions
V_{OL}	Output Voltage Low		0.4	V	I_{OL} = 2 mA
V_{OH}	Output Voltage High	2.4		V	I_{OH} = - 400 µA
I_{LO}	Output Leakage Current		±10	µA	
C_{OUT}	Output Capacitance		10	pF	

TIMING CHARACTERISTICS

Unless otherwise specified, the timing characteristics apply to both ZR36060-27 and ZR36060-29.5.

Table 7: Video Bus Input Timing

Symbol	Parameter	Min	Max	Unit	Comments
F_{VCLKx2}	VCLKx2 Frequency	22.2 22.2	27.0 29.5	MHz MHz	ZR36060-27 ZR36060-29.5 40% to 60% duty cycle
t_{V2P}	VCLKx2 Period	33.8 37.0	45 45	ns ns	ZR36060-27 ZR36060-29.5
t_{CLK}	Internal PLL Clock Period	$t_{V2P} = 2 * t_{CLK}$			The PLL multiplies the VCLKx2 frequency by 2. t_{CLK} is used as a reference variable for other timing parameters.
t_{V2T}	VCLKx2 Rise/Fall Transition		3	ns	
t_{VT}	VCLK Rise/Fall Transition		3	ns	
t_{VIS}	Video Bus Input Setup	6		ns	
t_{VIH}	Video Bus Input Hold	0		ns	



Note: 1. In this diagram *VCLKPol* = 0.
2. In 16-bit interface mode inputs are sampled on VCLKx2 rising edges qualified by VCLK.
3. In 8-bit interface mode inputs are sampled on all VCLKx2 rising edges.

Figure 41. Video Bus Input Timing

Table 8: Video Bus Output Timing

Symbol	Parameter	Min	Max	Unit	Comments
t_{VBO}	Video Bus Output Delay (16-bit)	t_{CLK} - 1	t_{CLK} + 10	ns	50pf load
t_{VBO8}	Video Bus Output Delay (8-bit)	$0.5t_{CLK}$ - 1	$0.5t_{CLK}$ + 10	ns	50pf load

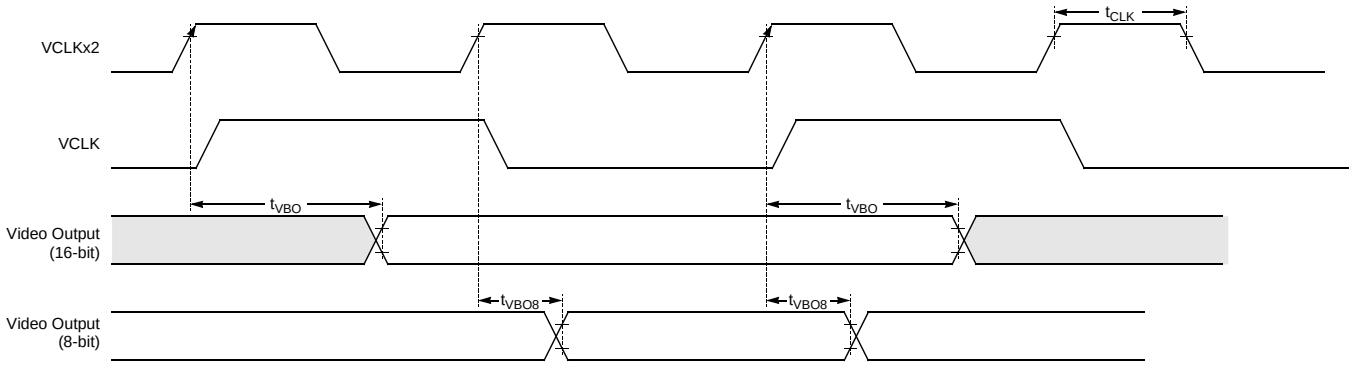


Figure 42. Video Bus Output Timing

Table 9: Host Interface Timing

Symbol	Parameter	Min	Max	Unit	Comment
t_{CAS}	$\overline{CS}$/ ADDR1:0 setup to $\overline{WR}$ or $\overline{RD}$ falling edge	5		ns	
t_{CAH}	$\overline{CS}$/ ADDR1:0 hold from $\overline{WR}$ or $\overline{RD}$ rising edge	5		ns	
t_{WDUR}	Minimum $\overline{WR}$ strobe pulse width	3 * t_{CLK}		ns	
t_{WACK}	$\overline{WR}$ low to $\overline{ACK}$ assertion (low)	3 * t_{CLK}	3 * t_{CLK} + 10	ns	50 pF load
t_{DS}	Input Data Setup to $\overline{WR}$ rising edge	3		ns	
t_{DH}	Input Data Hold from $\overline{WR}$ rising edge	2		ns	
t_{RDUR}	Minimum $\overline{RD}$ strobe pulse width	5 * t_{CLK}		ns	
t_{RACK}	$\overline{RD}$ low to $\overline{ACK}$ assertion (low)	5 * t_{CLK}	5 * t_{CLK} + 10	ns	50 pF load
t_{OPD}	$\overline{RD}$ low to output data valid		4 * t_{CLK} + 10	ns	
t_{OH}	Output Data Hold, $\overline{RD}$ rising edge to data float	t_{CLK}	2 * t_{CLK} + 10	ns	50 pF load
t_{REC}	$\overline{WR}$ or $\overline{RD}$ rising edge to next falling edge of $\overline{WR}$ or $\overline{RD}$	3 * t_{CLK}		ns	
t_{AH}	$\overline{WR}$ or $\overline{RD}$ rising edge to $\overline{ACK}$ rising edge		25	ns	

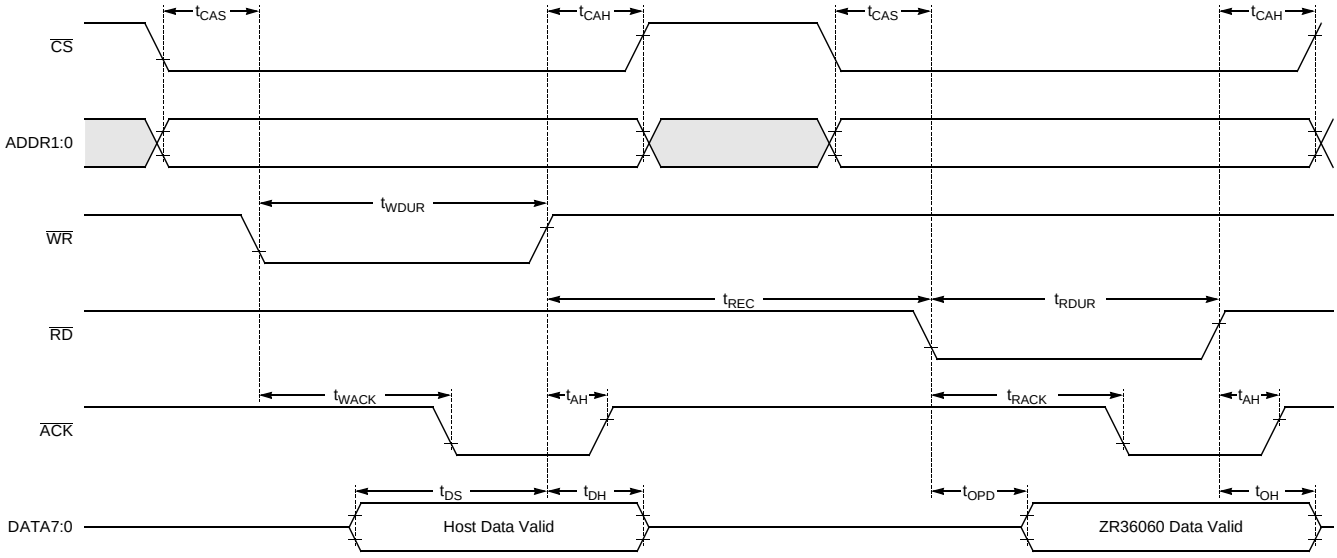


Figure 43. Host Interface Timing

Table 10: Code Slave Interface Timing

Symbol	Parameter		Min	Max	Unit	Comment
t_{CAS}	$\overline{CS}$ / ADDR1:0 setup to $\overline{WR}$ or $\overline{RD}$ falling edge		5		ns	
t_{CAH}	$\overline{CS}$ / ADDR1:0 hold from $\overline{WR}$ or $\overline{RD}$ rising edge		5		ns	
t_{DUR}	Minimum $\overline{WR}$ or $\overline{RD}$ strobe pulse width	8-bit mode:	3 * t_{CLK}		ns	
		16-bit mode:	4 * t_{CLK}		ns	
t_{ACK}	$\overline{WR}$ or $\overline{RD}$ low to $\overline{ACK}$ assertion (low)		3 * t_{CLK}	3 * t_{CLK} + 10	ns	50 pF load
t_{DS}	Input Data Setup to $\overline{WR}$ rising edge		5		ns	
t_{DH}	Input Data Hold from $\overline{WR}$ rising edge		2		ns	
t_{OPD}	$\overline{RD}$ low to output data valid			2 * t_{CLK} + 10	ns	50 pF load
t_{OH}	Output Data Hold $\overline{RD}$ rising edge to data float		t_{CLK}	2 * t_{CLK} + 10	ns	50 pF load
t_{REC}	$\overline{WR}$ or $\overline{RD}$ rising edge to next falling edge of $\overline{WR}$ or $\overline{RD}$		3 * t_{CLK}		ns	
t_{AH}	$\overline{WR}$ or $\overline{RD}$ rising edge to: - $\overline{ACK}$ rising edge - $\overline{CBUSY}$ falling edge - $\overline{EOI}$ falling edge			25	ns	

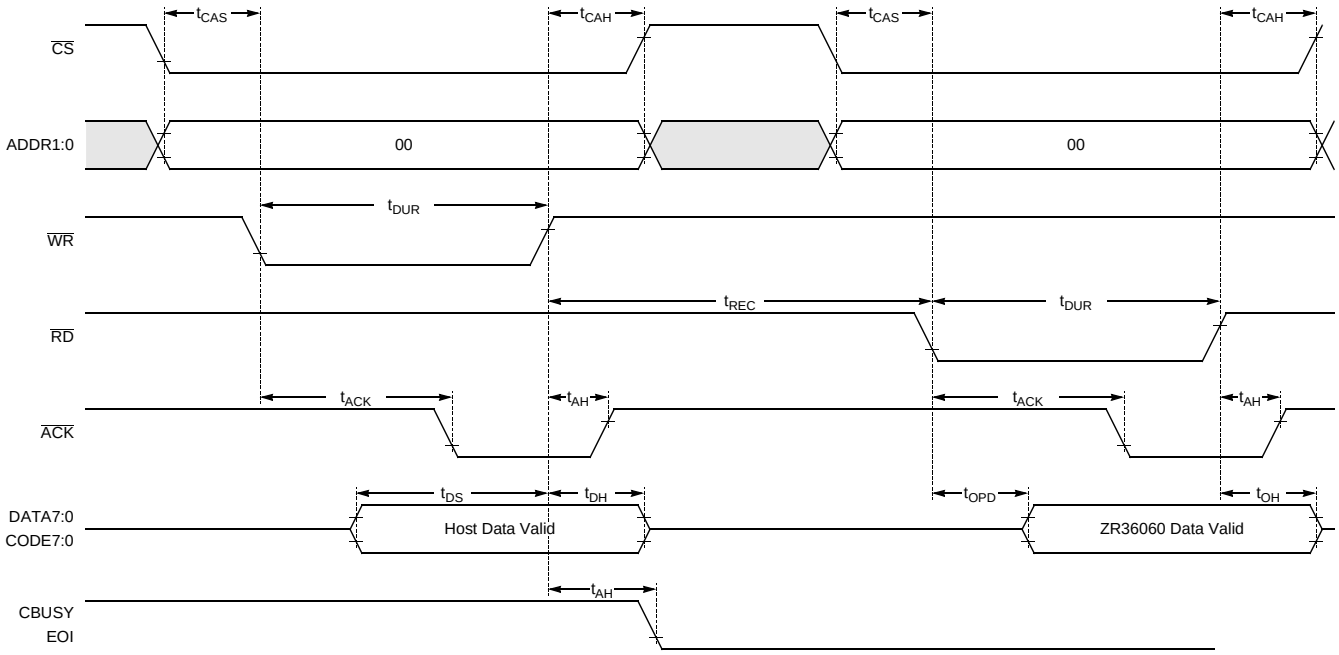


Figure 44. Code Slave Interface Timing

Table 11: Code Master Interface Timing - Compression

Symbol	Parameter	Min	Max	Unit	Comment
t_{CPD}	$\overline{CCS}$ Propagation Delay	0	7	ns	
t_{SPD}	$\overline{CWE}$ Propagation Delay	t_{CLK}	t_{CLK} + 7	ns	[1]
t_{SH}	$\overline{CWE}$ Hold Delay	0		ns	
t_{DPD}	Code Data Propagation Delay	t_{CLK}	t_{CLK} + 7	ns	[1]
t_{ZD}	Code Data Turn-on Delay (Float to Drive)	0	7	ns	
t_{DZ}	Code Data Hold Delay (Drive to Float)			ns	[2]
t_{BSU}	$\overline{CBUSY}$ Setup	6		ns	
t_{BH}	$\overline{CBUSY}$ Hold	0		ns	

1. These signals are triggered by an edge of the internal 2X PLL clock that falls half way between positive edges of VCLKx2. Thus the delay times relative to VCLKx2 are specified in terms of t_{CLK}, the period of the PLL clock.
2. At the termination of a CODE bus cycle, when a new bus cycle does not start immediately, the ZR36060 stops driving the CODE bus and the bus floats. In this case, the data hold time is determined by the decay time of the bus capacitance and any pullup resistors connected to it.

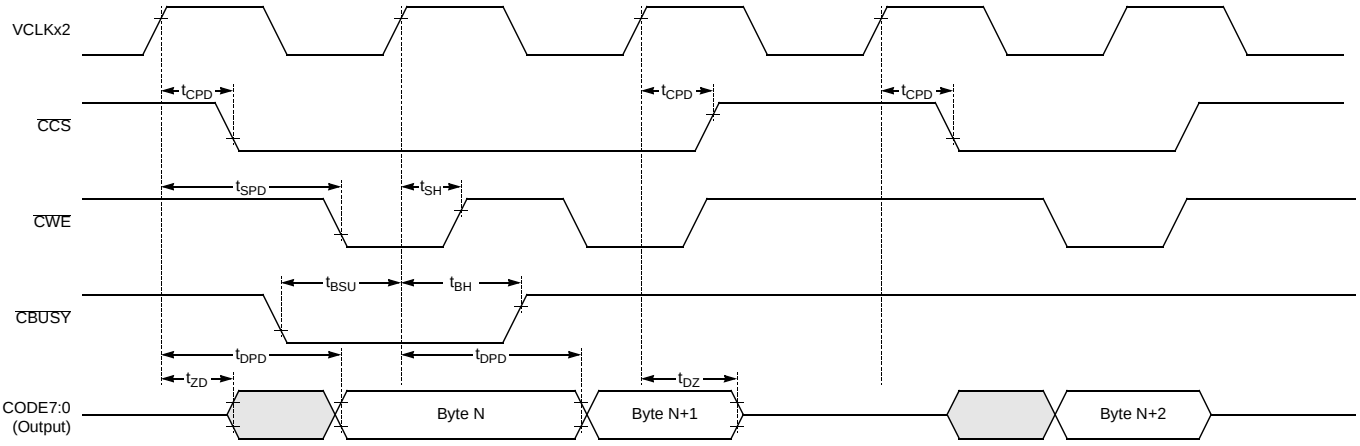


Figure 45. Code Master Interface Timing - Compression

Table 12: Code Master Interface Timing - Decompression

Symbol	Parameter	Min	Max	Unit	Comment
t_{CPD}	$\overline{CCS}$ Propagation Delay	0	7	ns	
t_{SPD}	$\overline{COE}$ Propagation Delay	t_{CLK}	t_{CLK}+7	ns	[1]
t_{SH}	$\overline{COE}$ Hold Delay	0	7	ns	
t_{DSU}	Code Data Input Setup	6		ns	
t_{DH}	Code Data Input Hold	0		ns	
t_{BSU}	$\overline{CBUSY}$ Setup	6		ns	
t_{BH}	$\overline{CBUSY}$ Hold	0		ns	

1. This signal is triggered by an edge of the internal 2X PLL clock that falls half way between positive edges of VCLKx2. Thus the delay time relative to VCLKx2 is specified in terms of t_{CLK}, the period of the PLL clock.

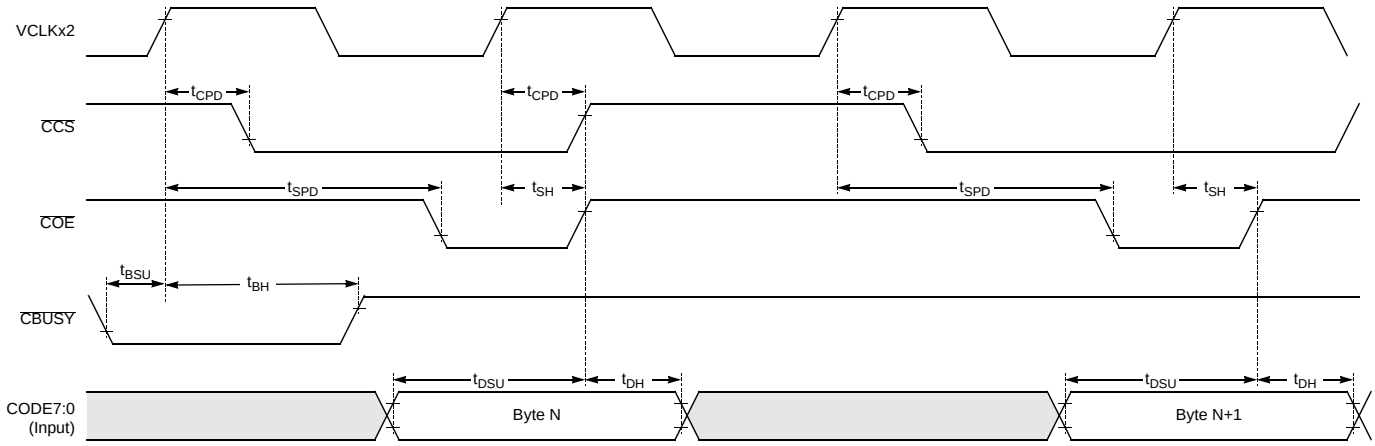


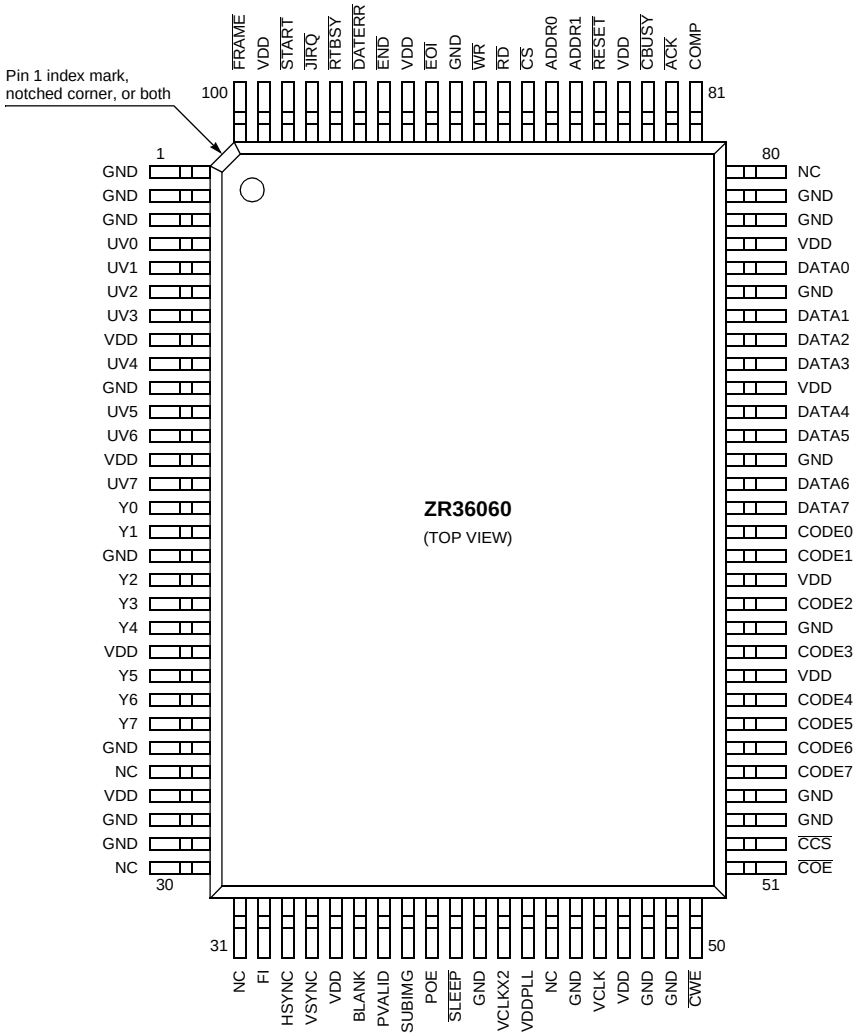
Figure 46. Code Master Interface Timing - Decompression

10.0 PINOUT

Table 13: 100-Pin Quad Flat Pack Pin Assignment

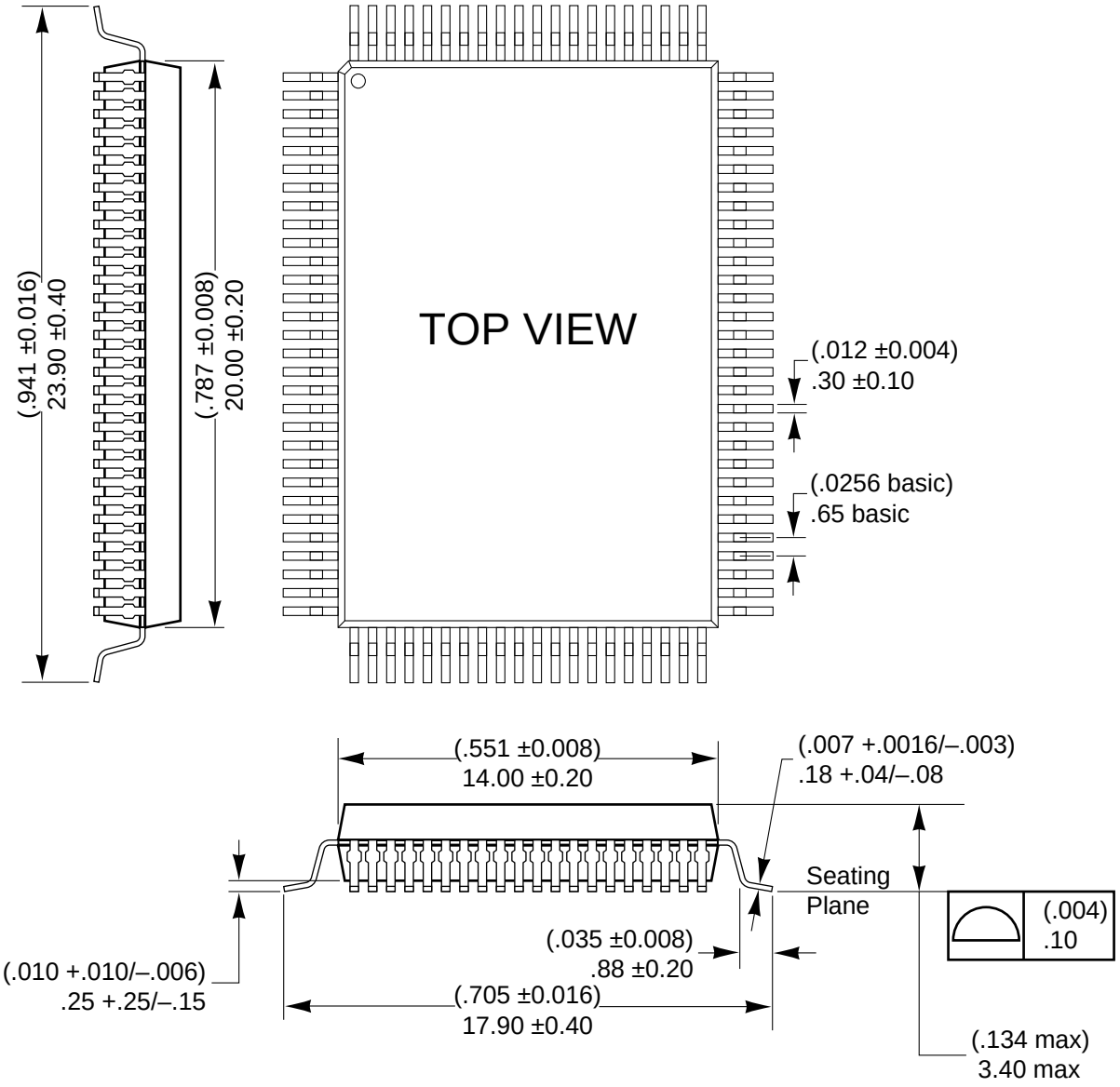
Pin No	Pin Name	Type [1]	Pin No	Pin Name	Type[1]	Pin No	Pin Name	Type[1]	Pin No	Pin Name	Type[1]
1	GND	P	26	NC [2]	–	51	$\overline{COE}$	O	76	DATA0	I/O
2	GND	P	27	VDD	P	52	$\overline{CCS}$	O	77	VDD	P
3	GND	P	28	GND	P	53	GND	P	78	GND	P
4	UV0	I/O	29	GND	P	54	GND	P	79	GND	P
5	UV1	I/O	30	NC[2]	–	55	CODE7	I/O	80	NC[2]	–
6	UV2	I/O	31	NC[2]	–	56	CODE6	I/O	81	COMP	O
7	UV3	I/O	32	FI	I/O	57	CODE5	I/O	82	$\overline{ACK}$	O
8	VDD	P	33	HSYNC	I/O	58	CODE4	I/O	83	$\overline{CBUSY}$	I/O
9	UV4	I/O	34	VSYNC	I/O	59	VDD	P	84	VDD	P
10	GND	P	35	VDD	P	60	CODE3	I/O	85	$\overline{RESET}$	I
11	UV5	I/O	36	BLANK	O	61	GND	P	86	ADDR1	I
12	UV6	I/O	37	PVALID	I	62	CODE2	I/O	87	ADDR0	I
13	VDD	P	38	SUBIMG	O	63	VDD	P	88	$\overline{CS}$	I
14	UV7	I/O	39	POE	I	64	CODE1	I/O	89	$\overline{RD}$	I
15	Y0	I/O	40	$\overline{SLEEP}$	I	65	CODE0	I/O	90	$\overline{WR}$	I
16	Y1	I/O	41	GND	P	66	DATA7	I/O	91	GND	P
17	GND	P	42	VCLKX2	I	67	DATA6	I/O	92	$\overline{EOI}$	O
18	Y2	I/O	43	VDDPLL [3]	P	68	GND	P	93	VDD	P
19	Y3	I/O	44	NC [4]	–	69	DATA5	I/O	94	$\overline{END}$	O
20	Y4	I/O	45	GND	P	70	DATA4	I/O	95	$\overline{DATERR}$	O
21	VDD	P	46	VCLK	I	71	VDD	P	96	$\overline{RTBSY}$	O
22	Y5	I/O	47	VDD	P	72	DATA3	I/O	97	$\overline{JIRQ}$	O
23	Y6	I/O	48	GND	P	73	DATA2	I/O	98	$\overline{START}$	I
24	Y7	I/O	49	GND	P	74	DATA1	I/O	99	VDD	P
25	GND	P	50	$\overline{CWE}$	O	75	GND	P	100	$\overline{FRAME}$	I

1. P=power, I=input, O=output.
2. Must be left unconnected.
3. VDDPLL must be connected to VDD. To minimize PLL clock jitter, the following power supply decoupling is recommended:
   a. Use a 0.1 microfarad capacitor as close as possible to each VDD pin.
   b. Use a 0.1 microfarad and a 0.001 microfarad capacitor on the VDDPLL pin.
   c. Use 47 microfarad (or larger) tantalum electrolytic on the VDD power plane.
4. This pin is the output of the PLL clock doubler. It must be left unconnected.


Pin 1 index mark,
notched corner, or both
100
81
1
80
30
51
31
50
FRAME
VDD
START
JIRQ
RTBSY
DATERR
END
VDD
EOI
GND
WR
RD
CS
ADDR0
ADDR1
RESET
VDD
CBUSY
ACK
COMP
GND
GND
GND
UV0
UV1
UV2
UV3
VDD
UV4
GND
UV5
UV6
VDD
UV7
Y0
Y1
GND
Y2
Y3
Y4
VDD
Y5
Y6
Y7
GND
NC
VDD
GND
GND
NC
ZR36060
(TOP VIEW)
NC
GND
GND
VDD
DATA0
GND
DATA1
DATA2
DATA3
VDD
DATA4
DATA5
GND
DATA6
DATA7
CODE0
CODE1
VDD
CODE2
GND
CODE3
VDD
CODE4
CODE5
CODE6
CODE7
GND
GND
CCS
COE
NC
FI
HSYNC
VSYNC
VDD
BLANK
PVALID
SUBIMG
POE
SLEEP
GND
VCLKX2
VDDPLL
NC
GND
VCLK
VDD
GND
GND
CWE

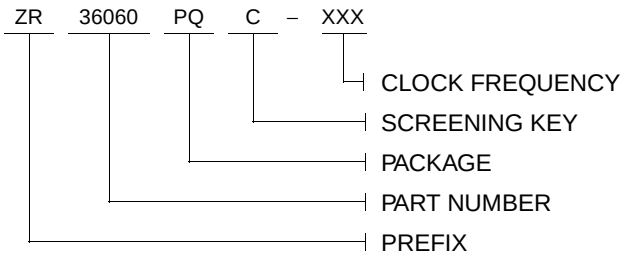
11.0 MECHANICAL DATA

Package Outline Drawing - ZORAN 100 lead Metric QFP



NOTE: Principal dimensions in millimeters, dimensions in brackets in inches.

ORDERING INFORMATION



PACKAGE
PQ - Plastic Quad Flat Pack (EIAJ)

SCREENING KEY
C - 0°C to +70°C (V_{CC} = 4.75V to 5.25V)

CLOCK FREQUENCY
27 - VCLKx2 Frequency 27 MHz Max
29.5 - VCLKx2 Frequency 29.5 MHz Max

SALES OFFICES

■ ***U.S. Headquarters***
Zoran Corporation
3112 Scott Blvd.
Santa Clara, CA 95054 USA
Telephone: 408-919-4111
FAX: 408-919-4122

■ ***Israel Design Center***
Zoran Microelectronics, Ltd.
Advanced Technology Center
P.O. Box 2495
Haifa, 31024 Israel
Telephone: 972-4-854-5777
FAX: 972-4-855-1550

■ ***Japan Office***
Zoran Japan
2-26-2 Sasazuka
Shibuya-ku,
Tokyo 151 Japan
Telephone: 03-5352-0971
FAX: 03-5352-0972

The material in this data sheet is for information only. Zoran Corporation assumes no responsibility for errors or omissions and reserves the right to change, without notice, product specifications, operating characteristics, packaging, etc. Zoran Corporation assumes no liability for damage resulting from the use of information contained in this document.