CEC Programming Interface (CPI)

Programmer Reference

SiI-PR-0041-B

June 2017

Copyright Notice

Copyright © 2009-2017 Lattice Semiconductor Corporation. All rights reserved. The contents of these materials contain proprietary and confidential information (including trade secrets, copyright, and other Intellectual Property interests) of Lattice Semiconductor Corporation and/or its affiliates. All rights are reserved. You are permitted to use this document and any information contained therein expressly and only for bona fide non-commercial evaluation of products and/or services from Lattice Semiconductor Corporation or its affiliates; and only in connection with your bona fide consideration of purchase or license of products or services from Lattice Semiconductor Corporation or its affiliates, and only in accordance with the terms and conditions stipulated. Contents, (in whole or in part) may not be reproduced, downloaded, disseminated, published, or transferred in any form or by any means, except with the prior written permission of Lattice Semiconductor Corporation and/or its affiliates. Copyright infringement is a violation of federal law subject to criminal and civil penalties. You have no right to copy, modify, create derivative works of, transfer, sublicense, publicly display, distribute or otherwise make these materials available, in whole or in part, to any third party. You are not permitted to reverse engineer, disassemble, or decompile any device or object code provided herewith. Lattice Semiconductor Corporation reserves the right to revoke these permissions and require the destruction or return of any and all Lattice Semiconductor Corporation proprietary materials and/or data.

Patents

The subject matter described herein may contain one or more inventions claimed in patents or patents pending owned by Lattice Semiconductor Corporation and/or its affiliates.

Trademark Acknowledgment

Lattice Semiconductor Corporation®, the Lattice Semiconductor logo, Silicon Image®, the Silicon Image logo, Instaport®, the Instaport logo, InstaPrevue®, Simplay®, Simplay HD®, the Simplay HD logo, Simplay Labs™, the Simplay Labs logo, the SiBEAM Snap™, the SiBEAM Snap logo, UltraGig™, the UltraGig logo are trademarks or registered trademarks of Lattice Semiconductor Corporation in the United States and/or other countries. HDMI® and the HDMI logo with High-Definition Multimedia Interface are trademarks or registered trademarks of, and are used under license from, HDMI Licensing, LLC. in the United States or other countries. MHL® and the MHL logo with Mobile High-Definition Link are trademarks or registered trademarks of, and are used under license from, MHL, LLC. in the United States and/or other countries. WirelessHD®, the WirelessHD logo, WiHD® and the WiHD logo are trademarks, registered trademarks or service marks of SiBeam, Inc. in the United States or other countries.

HDMI Licensing, LLC; MHL, LLC; Simplay Labs, LLC; and SiBeam, Inc. are wholly owned subsidiaries of Lattice Semiconductor Corporation.

All other trademarks and registered trademarks are the property of their respective owners in the United States or other countries. The absence of a trademark symbol does not constitute a waiver of Lattice Semiconductor's trademarks or other intellectual property rights with regard to a product name, logo or slogan.

Export Controlled Document

This document contains materials that are subject to the U.S. Export Administration Regulations and may also be subject to additional export control laws and regulations (collectively "Export Laws") and may be used only in compliance with such Export Laws. Unless otherwise authorized by an officer of Lattice Semiconductor Corporation in writing, this document and the information contained herein (a) may not be used in relation to nuclear, biological or chemical weapons, or missiles capable of delivering these weapons, and (b) may not be re-exported or otherwise transferred to a third party who is known or suspected to be involved in relation to nuclear, biological or chemical weapons, or missiles capable of delivering these weapons, or to any sanctioned persons or entities.

Further Information

To request other materials, documentation, and information, contact your local Lattice Semiconductor sales office or visit the Lattice Semiconductor web site at www.latticesemi.com.

Disclaimers

These materials are provided on an "AS IS" basis. Lattice Semiconductor Corporation and its affiliates disclaim all representations and warranties (express, implied, statutory or otherwise), including but not limited to: (i) all implied warranties of merchantability, fitness for a particular purpose, and/or non-infringement of third party rights; (ii) all warranties arising out of course-of-dealing, usage, and/or trade; and (iii) all warranties that the information or results provided in, or that may be obtained from use of, the materials are accurate, reliable, complete, up-to-date, or produce specific outcomes. Lattice Semiconductor Corporation and its affiliates assume no liability or responsibility for any errors or omissions in these materials, makes no commitment or warranty to correct any such errors or omissions or update or keep current the information contained in these materials, and expressly disclaims all direct, indirect, special, incidental, consequential, reliance and punitive damages, including WITHOUT LIMITATION any loss of profits arising out of your access to, use or interpretation of, or actions taken or not taken based on the content of these materials. Lattice Semiconductor Corporation and its affiliates reserve the right, without notice, to periodically modify the information in these materials, and to add to, delete, and/or change any of this information.

Products and Services

The products and services described in these materials, and any other information, services, designs, know-how and/or products provided by Lattice Semiconductor Corporation and/or its affiliates are provided on "AS IS" basis, except to the extent that Lattice Semiconductor Corporation and/or its affiliates provides an applicable written limited warranty in its standard form license agreements, standard Terms and Conditions of Sale and Service or its other applicable standard form agreements, in which case such limited warranty shall apply and shall govern in lieu of all other warranties (express, statutory, or implied). EXCEPT FOR SUCH LIMITED WARRANTY, LATTICE SEMICONDUCTOR CORPORATION AND ITS AFFILIATES DISCLAIM ALL REPRESENTATIONS AND WARRANTIES (EXPRESS, IMPLIED, STATUTORY OR OTHERWISE), REGARDING THE INFORMATION, SERVICES, DESIGNS, KNOW-HOW AND PRODUCTS PROVIDED BY LATTICE SEMICONDUCTOR CORPORATION AND/OR ITS AFFILIATES, INCLUDING BUT NOT LIMITED TO, ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND/OR NON-INFRINGEMENT OF THIRD PARTY RIGHTS. YOU ACKNOWLEDGE AND AGREE THAT SUCH INFORMATION, SERVICES, DESIGNS, KNOW-HOW AND PRODUCTS HAVE NOT BEEN DESIGNED, TESTED, OR MANUFACTURED FOR USE OR RESALE IN SYSTEMS WHERE THE FAILURE, MALFUNCTION, OR ANY INACCURACY OF THESE ITEMS CARRIES A RISK OF DEATH OR SERIOUS BODILY INJURY, INCLUDING, BUT NOT LIMITED TO, USE IN NUCLEAR FACILITIES, AIRCRAFT NAVIGATION OR COMMUNICATION, EMERGENCY SYSTEMS, OR OTHER SYSTEMS WITH A SIMILAR DEGREE OF POTENTIAL HAZARD. NO PERSON IS AUTHORIZED TO MAKE ANY OTHER WARRANTY OR REPRESENTATION CONCERNING THE PERFORMANCE OF THE INFORMATION, PRODUCTS, KNOW-HOW, DESIGNS OR SERVICES OTHER THAN AS PROVIDED IN THESE TERMS AND CONDITIONS.

Contents

Figures

Tables

1. Overview

The Lattice Semiconductor CEC application solution applies to both transmitter and receiver applications. It involves a low-level and a high-level component.

For the low-level component:

- The low-level CEC protocol is handled by a completely hardware-based solution. This logic manages timing, arbitration, and retries, and also implements a slave I^{2}C interface in hardware to handle communication with the host.
- For special applications, Lattice Semiconductor also offers microcontroller firmware that handles the low-level CEC protocol.
- For the high-level component:
- Lattice Semiconductor CEC software source code allows messages to be exchanged over the host interface as discussed above. For development, Lattice Semiconductor provides Windows-based software tools, including a kit that allows a PC to be used to generate I^{2}C commands over a serial port or a USB 1.1-capable port.

CEC Programming Interface (CPI). The I^{2}C register set used for this solution is referred to as CEC Programming Interface, or just CPI. This standard register set is used across all Lattice Semiconductor devices and applications.

Operation with Lattice Semiconductor Devices. The solution is designed to work with Lattice Semiconductor transmitters and receivers. Together with subsystems using the Transmitter Programming Interface (TPI), Receiver Programming Interface (RPI), or Repeater Programming Interface (RTPI) protocols, host code commands operations at a high level requiring minimal service overhead.

Figure 1.1 illustrates the CEC connectivity in a typical Rx application. The SiI9185 Switch with integrated CPI hardware implements:

- Hardware CEC port connection
- I^{2}C master interface to HDMI Rx chip
- Hardware I^{2}C slave interface from, and interrupt circuit to, TV Host Processor.

The TV Host processor manages both RPI and system-wide CEC operations through a single I^{2}C bus.

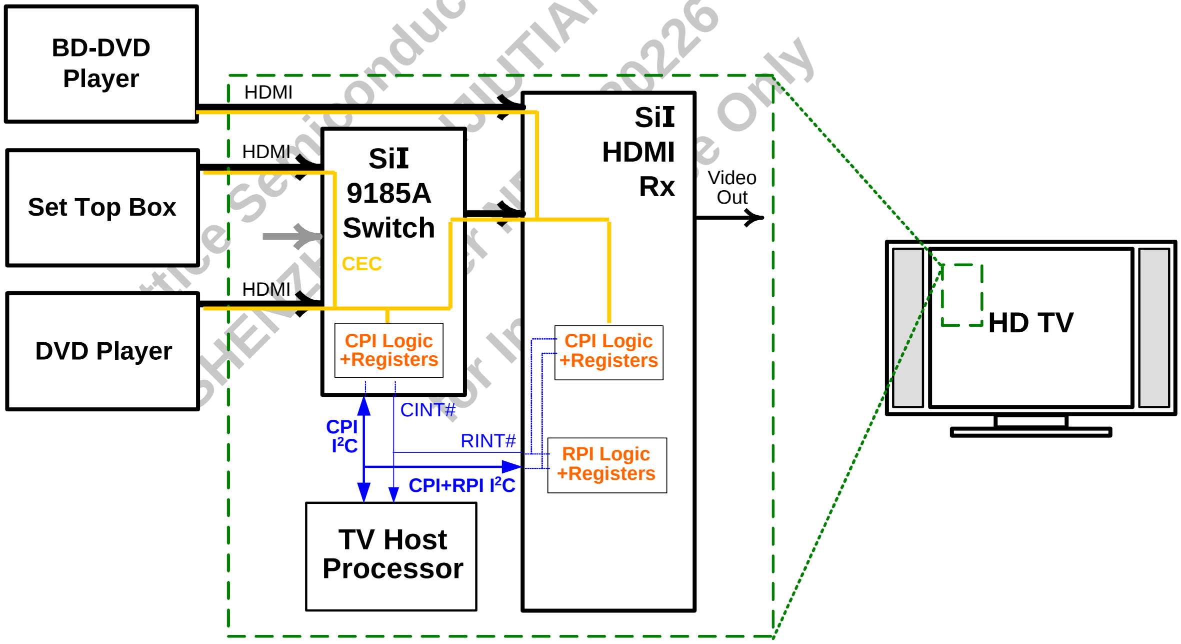

Figure 1.1. Typical CEC Application

1.1. Conventions

In this document, the following naming standards are used to describe operations related to CEC.

- **Host** refers to activities of the host processor implementing the high-level component of the solution.
- **CPI** refers to use of the I^{2}C interface to access the CPI registers implemented by logic, or by the microcontroller and firmware in an alternative software-based solution.
- **Opcode** refers to the single-byte code defined in the CEC spec to indicate the intent of the operation.
- **Operand** refers to one of the parameters following the opcode.
- **Message** refers to the Opcode plus any Operands.
- **Frame** refers to an exchange on the CEC bus itself, properly arbitrated and framed, and includes a Message along with initiator and destination information.
- **Command** is sometimes used to refer to bytes exchanged on the CPI level – that is, the I^{2}C interface. ‘Command’ is not part of the CEC specification.

Register addresses are shown as offsets from a base value of 0x00. Default register values, where meaningful, are provided in hexadecimal and are shown below the register offset in brackets [ ]. Where bits are undefined, programmers should use read-modify-write sequences to leave these bits undisturbed.

1.2. Host Processor Support

Lattice Semiconductor provides generic source code to support CPI across a variety of platforms. Contact your Lattice Semiconductor representative for additional information.

2. Register Set

The CPI register set provides access to data structures through the following register groups.

- Identification
- Message Transmission
- Message Reception
- Power State Management
- Interrupt Service

Figure 2.1 illustrates the interface of these register groups, as presented to the host in a typical receiver application.

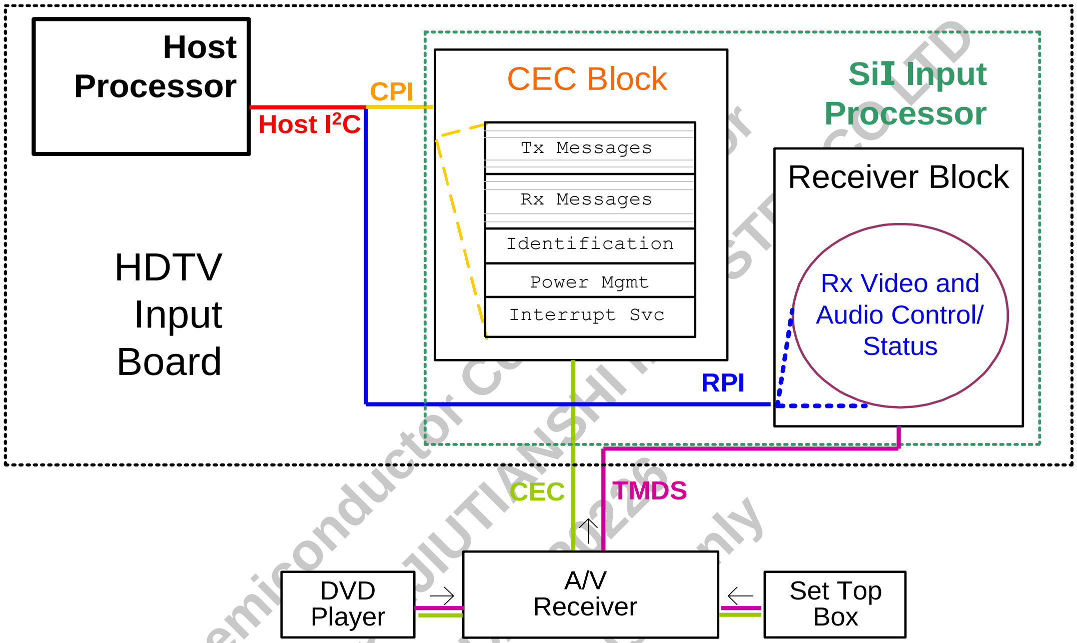

Figure 2.1. CPI Register Interface to Host

Message Queues. The CEC implementation includes queues for both transmission and reception of messages.

- Transmit Buffer: The host writes the CPI transmit register set first, and then triggers a load of this information into the internal Transmit Buffer. Once the opcode and operands are in the Transmit Buffer, the subsystem assembles a frame and starts to send it using the defined CEC wire protocol. All arbitration is handled automatically.

  During this time, one additional message can be written to the transmit register set through the CPI I²C interface. Host software waits for the Transmitter FIFO Empty interrupt (or polls) to determine when another message can be loaded.
- Receive FIFO: Three messages deep. The receive FIFO fills up as frames come across the CEC line. The Message Incoming interrupt tells the host that a frame destined for this target device is coming in. The Message Available interrupt tells the host that the FIFO contains at least one complete message and should be serviced soon.

Interrupt Service. The host can optionally set up Interrupt Service for the most efficient handling of CEC events. The CEC hardware interrupt is referred to as CINT# in this document, although it may take on different names when implemented as hardware in various Lattice Semiconductor products.

2.1. Host Requirements

The host processor must communicate with the CEC subsystem through a specific interface.

I^2C Interface. The hardware I^2C interface of the CEC device runs up to 400 kHz.

I^2C Access Address. The I^2C address used to access the CPI registers is 0xC0. This address may vary on future hardware implementations; refer to the device-specific programmer reference for details.

CEC Subsystem Reset and Interrupt. The host has software control of the CEC device Reset function, typically by means of a GPIO pin. It is also able to monitor the CEC subsystem CINT# interrupt signal using a GPIO pin or similar. When the I^2C interface cannot be accessed correctly, or if CINT# goes low and cannot be cleared, the host must generate Reset to the CEC subsystem. This is a "fail-safe" operation and serves to prevent system hang in the case of an unrecoverable event.

CEC Calibration. Timing requirements for CEC must be strictly controlled and monitored. The CPI hardware manages most of this task. For core revisions 1.2 and later, the calibration is automatic at reset; no setup is necessary. However, for prior core revisions, calibration must be performed under host control as described in Appendix A. When calibration is required, the host must control the calibration cycle using a GPIO pin.

2.2. Identification and Verification

2.2.1. Identification Registers

The ID registers return the device ID, CEC spec ID, and hardware (or firmware) revision ID. A bit is provided to differentiate between firmware-based and hardware-based implementations. The ID registers are listed in Table 2.1.

Access. These registers are accessed as single bytes.

Table 2.1. CPI Identification Registers (RO)

Offset	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x80	Device ID – 0xCC (CEC)							
0x81	CEC spec major release – 0x1				CEC spec minor release			
0x82	Subsytem[1] 1 = Hardware 0 = Firmware	RSVD	RSVD	RSVD	CEC spec suffix			
0x83	Major Revision ID – see Table 2.2				Minor Revision ID – see Table 2.2			

Note: This bit was documented incorrectly in revisions A02 and lower of this document. All chips return this bit as 1.

2.2.1.1. Revision ID

The CEC-CPI core and register set are used in a variety of Lattice Semiconductor products. Due to ongoing enhancements, feature changes occur as noted in Table 2.2.

Table 2.2. Revision/Feature Changes

Lattice Semiconductor Device	Register 0x83 (CPI Core revision)	Description
SiI9185 Switch	0x00	CEC support is limited – contact Lattice Semiconductor for information.
SiI9185A Switch with CEC	0x02	Supports CEC 1.2 fully through standard CPI register set. – Calibration control register is accessed through switch control registers.
SiI9022/9024 Tx	0x01	Supports CEC 1.3 fully through standard CPI register set. – Calibration control register is moved into CPI register group. – Adds feature to allow calibration control solely through register writes (no external GPIO control pin needed on CEC_D).
SiI9220/22/24/26 Tx SiI9290 Bridge	0x11	Same as above, and: – Eliminates re-calibration requirement: a single host-controlled calibration is required after hardware reset – Adds programmable Retry Limit at 0xA0 – Adds Feature Abort registers at 0xC0-0xDF.
SiI9155/65 Rx revision 2.1	0x12	Same as above, and: – Self-calibrates after reset, with no host involvement – Adds 0xA7 interrupt event summary bit at 0xA6[4] – Adds Automatic Discovery.
SiI9127 Rx SiI9223/33 Rx revision 1.1	0x13	
SiI9187/9287 SiI9187A/9287A Switch revision 0.x	0x14	Same as above, and: – Adds handling improvement for Destination/Initiator ID – Allows full readback of Transmit Buffer – Allows snoop of all devices – Allows self-clearing of Message Available interrupt – Adds “Do Not Receive Own Message” feature.
SiI9251/61 Rx revision 0	0x15	
SiI9022A/9024A Mobile Tx	0x17	
SiI9204/9206 Tx Phy	0x18	
SiI923x MHL Tx	0x19	
SiI9334 HDMI Tx revision 0	0x16	Same as above, and: – Adds CDC support for HDMI 1.4
SiI9385/9387/9389 Switch revision 0	0x1A	
SiI9187B/9285B/9287B Switch revision 1	0x1B	

2.3. Message Transmission Registers

The host writes CPI with a CEC message through the following read/write registers. CPI implements a transmit buffer that operates as described below.

2.3.1. CEC Initiator/Destination Block

The host typically sets the CEC Initiator ID just one time after system power-up, to preset the identifier of the host device for all future messages. For hosts capable of acting as multiple devices, this register must be set correctly before each message is written in the transmit registers. Thereafter, for each message to a specific target device, the host simply writes the number of operands and CEC Destination ID corresponding to that device.

Important Note: For core versions before rev. 1.4, all message transmissions must be allowed to complete, or the buffer must be flushed, before changing the Initiator ID or Destination ID. This ID information is attached to the CEC frame at the time of transmission, not at the time the messages are written to the CPI registers. For rev. 1.4 and later, the corresponding ID information is attached to each message as soon as the message is loaded into the buffer for transmission.

Send Polling Message. For discovery of other system devices, CEC defines a Polling message with no opcode and no operands, to which a destination device can respond simply by ACKing the message. The SEND_POLL bit used to generate this cycle is self-clearing. To perform a manual discovery sequence, the host simply writes this register repeatedly with incrementing destination addresses and with SEND_POLL =1, checking the returned ACK status each time. For later revisions of the CPI core, this process can be performed automatically (refer to Automatic Discovery section below).

When SEND_POLL is used to do a ping, transmission will go through the normal process: the subsystem will try sending and resending the message up to 5 times, until it gets an ACK. If after 5 tries there is no ACK, a Frame Retransmit Count Exceeded event occurs and the Tx FIFO must be flushed (refer to Sending / Verifying Transmission of a Message section).

Notes:

1. Transmit CEC commands (either auto-calc or manual) must **not** be issued when using SEND_POLL. Starting the transmission of any message at this time can corrupt the messages being sent by the Send Polling Message hardware.
2. SEND_POLL should not be used if Automatic Discovery is enabled (where available).
3. Disable Feature Abort before activating SEND_POLL.

Access. These registers are accessed as single bytes.

Table 2.3. CEC Initiator/Destination Block (R/W)

Offset	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x88	CEC_TX_INIT							
	RSVD				CEC_INIT_ID [3:0] CEC Initiator ID – needs to be written to identify this device in future transmissions. It must not be changed unless the transmit buffer is empty.			
0x89	CEC_TX_DEST							
	SEND_POLL Generate a polling message (no opcode or operands) 0 = No 1 = Yes (self-resetting bit)	RSVD	—	—	CEC_DEST_ID [3:0] CEC Destination ID – identifies the target device for the message. It must be written before writing the corresponding CEC opcode and operands. It must not be changed unless the transmit buffer is empty.			

INIT and DEST for transmission, and CAPTURE for reception, can be among these devices.

0000 = TV
0001 = Recording Device 1
0010 = Recording Device 2
0011 = Tuner 1
0100 = Playback Device 1
0101 = Audio System
0110 = Tuner 2
0111 = Tuner 3
1000 = Playback Device 2
1001 = Recording Device 3
1010 = Tuner 4
1011 = Playback Device 3
1100 = Reserved
1101 = Reserved
1110 = Free Use
1111 = Unregistered (as initiator address), Broadcast (as destination address)

2.3.2. CEC Transmit Data Block

The host writes the actual CEC opcode to be sent, along with up to 15 operands, to the CEC Data Transmit Block. CPI implements an internal Transmit Buffer, so that a second message can be written to the registers while the first is still being transmitted. If a previous message transmission is in progress, the loading and transmission of the new message will be delayed until completion of the previous one.

Two methods are allowed for triggering the load of the message bytes into the internal Transmit Buffer.

- If TX_AUTO_CALC is set to 1, writing the opcode followed by zero or more operands as an I^{2}C burst triggers the loading. The STOP at the end of the burst sets the TX_CNT length automatically.
- If TX_AUTO_CALC is set to 0, opcode and operands can be written in any order. Once they are all in place, a byte write to offset 0x9F with TRANSMIT_CEC_CMD set to 1 and the TX_CNT length set to the actual number of bytes triggers the loading.

> **Note:** For core revisions prior to rev. 1.4, if TX_AUTO_CALC is toggled from 0 to 1 or 1 to 0, the setting of the TRANSMIT_CEC_CMD bit is ignored. In other words, entering or exiting Auto Calc mode and issuing a manual CEC transmit command cannot take place with the same I^{2}C write cycle. Core rev. 1.4 and later remove this restriction.

Access. Any set of registers within the group can be accessed individually or in a burst. However, note that if TX_AUTO_CALC is set to 1, the registers should be accessed only as a burst starting from offset 0x89 and ending with the correct number of operands for that opcode.

Transmit Buffer Access. For debug purposes, the host can set the TX_BFR_ACCESS bit to 1 to allow it to read back the current buffer contents from 0x8F-0x9E. There is no interlock, so if the host has written a new message to the transmit data block and a previous frame is currently getting transmitted out of the buffer, it is possible that the readback values will not be valid (will instead be a mixture between the previous and current frame). This bit set =1 is not a normal operating condition, and the host must always clear this bit back to 0 before returning to normal operation.

Notes: For core revisions 1.4 and later, setting TX_BFR_ACCESS to 1 also allows readback of 0x88-89 to correspond to the message in the transmit buffer. That is:

- Core revisions 1.3 and earlier: CEC_INIT_ID and CEC_DEST_ID always return the last value written to them, regardless of the TX_BFR_ACCESS setting.
- Core revisions 1.4 and later: CEC_INIT_ID and CEC_DEST_ID return the value corresponding to the buffer being read. Therefore, if TX_BFR_ACCESS is set to 1, then CEC_INIT_ID and CEC_DEST_ID will return ID information associated with the message in the buffer.

This feature of the later core revision is convenient in case of error conditions: if a message fails to be acknowledged, the host can read back the source and intended target specific to the message that is stuck in the buffer as an aid in diagnosing the failure.

2.3.3. Sending / Verifying Transmission of a Message

Once a message has been written and loading to the internal Transmit Buffer is triggered by one of the methods described above, or a SEND_POLL request is generated, the host can check the status of the transmission. Two outcomes are possible, as indicated by bits in the Interrupt Status register.

Message Sent Event indicates that the transmission was received and ACKed by the destination device.

Frame Retransmit Count Exceeded Event indicates that after five tries (configurable on later core revisions), the transmission still was not acknowledged (was NACKed).

For a NACK event, the host must flush the Tx FIFO (bit 0x87[7]) before clearing this interrupt pending bit to avoid an automatic retry of the transmission.

Table 2.4. CEC Transmit Data Block (R/W)

Offset	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x8F	CEC_TX_OPCODE							
0x90	CEC_TX_OPERAND[0]							
0x91	CEC_TX_OPERAND[1]							
0x92	CEC_TX_OPERAND[2]							
0x93	CEC_TX_OPERAND[3]							
0x94	CEC_TX_OPERAND[4]							
0x95	CEC_TX_OPERAND[5]							
0x96	CEC_TX_OPERAND[6]							
0x97	CEC_TX_OPERAND[7]							
0x98	CEC_TX_OPERAND[8]							
0x99	CEC_TX_OPERAND[9]							
0x9A	CEC_TX_OPERAND[10]							
0x9B	CEC_TX_OPERAND[11]							
0x9C	CEC_TX_OPERAND[12]							
0x9D	CEC_TX_OPERAND[13]							
0x9E	CEC_TX_OPERAND[14]							
0x9F	CEC_TX_CONTROL_1							
	RSVD	TX_BFR_ACCESS Read back internal buffer contents from 0x8F-0x9E 0 = No 1 = Yes	TX_AUTO_CALC Auto-Calculate TX_CNT, and Send 0 = No 1 = Yes	TRANSMIT_CEC_CMD Send CEC Opcode and TX_CNT Operands 0 = No 1 = Yes Self-resetting bit	TX_CNT [3:0] Transmit Byte Count – selects the number of CEC_TX_OPERAND bytes to send with the opcode. 0000 = No operands 0001 = one operand .. 1111 = 15 operands			
0xA0 [04]	CEC_TX_CONTROL_2[1]							
	RSVD					TX_RETRY_LIMIT [2:0] Transmit Retry Limit – selects the number of retry attempts the transmitter will make. 000 = Try only once 001 = 1 retry 010 = 2 retries 011 = 3 retries 100 = 4 retries others RSVD		

Note: This register is implemented only on core revisions 1.1 (CPI 0x83=0x11) and later. The retry limit defaults to 0x04 for software backwards compatibility with those earlier core revisions.

2.3.4. Automatic Discovery

For core revisions 1.2 (CPI 0x83=0x12) and later, automatic device discovery is supported in hardware, eliminating the need to use the Send Polling Message function described previously. The feature works as follows.

1. Firmware starts automatic discovery by setting the START_DISC bit to 1.
2. The CPI core logic starts sending header-only packets with destination addresses from 1 to 14, recording ACK status for each packet and writing the status to the bitmap at CPI 0xE1-E2.
3. Once the process is completed, the logic sets the DISC_DONE bit to 1 and clears the START_DISC bit to 0.
4. After using the results, firmware clears the bitmap and DISC_DONE bit by setting the CLEAR_DISC bit to 1.

Note that the CEC receiver core is operational at this time, and the automatic Feature Abort message will be sent as appropriate after the discovery sequence is done.

If during the discovery process, the firmware clears the START_DISC bit before the process completes (the DISC_DONE bit still 0), the logic will stop the discovery sequence and write a partial status to the status register.

Note: Transmit CEC commands (either auto-calc or manual) must not be issued during the automatic device discovery process. Starting the transmission of any message at this time can corrupt the messages being sent by the automatic device discovery hardware.

Table 2.5. CEC Automatic Discovery Block

Offset	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0xE0 [00]	CEC_DISCOVERY_CTRL							
	DISC_DONE Automatic Discovery status 0 = Results not ready 1 = Discovery complete, in Discovery Status registers (read-only)	—	—	—	—	—	CLEAR_DISC Status Clear 0 = No effect 1 = Clear DISC_DONE and CEC_DISCOVERY_STATUS (write-only; always reads back 0)	START_DISC Start/Stop Automatic Discovery by Hardware 0 = Resume normal operation 1 = Start Automatic Discovery[1]
0xE1	CEC_DISCOVERY_STATUS (RO)							
	Device 7 0 = Not found 1 = Present	Device 6 0 = Not found 1 = Present	Device 5 0 = Not found 1 = Present	Device 4 0 = Not found 1 = Present	Device 3 0 = Not found 1 = Present	Device 2 0 = Not found 1 = Present	Device 1 0 = Not found 1 = Present	Device 0 0 = Not found 1 = Present
0xE2	RSVD	Device 14 0 = Not found 1 = Present	Device 13 0 = Not found 1 = Present	Device 12 0 = Not found 1 = Present	Device 11 0 = Not found 1 = Present	Device 10 0 = Not found 1 = Present	Device 9 0 = Not found 1 = Present	Device 8 0 = Not found 1 = Present

Note: Bit self-clears after process is complete, and can also be cleared manually mid-process.

2.4. Message Reception Registers

The host reads CEC messages destined for it through the CEC Receive Data Block registers. The CEC Capture ID register selects the device frames that will be captured in the receive FIFO.

2.4.1. CEC Capture ID Registers

The CEC Capture ID register is separate from the CEC Initiator ID. It selects the received frames that will be captured in the receive FIFO and acknowledged. If the message target destination matches one of the bits set in this register or is a Broadcast cycle, it will be captured.

Table 2.6. CEC Capture ID Registers

Offset	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
—	CEC_CAPTURE_ID [15:0]							
0xA2	Device 0111	Device 0110	Device 0101	Device 0100	Device 0011	Device 0010	Device 0001	Device 0000
0xA3	Device 1111	Device 1110	Device 1101	Device 1100	Device 1011	Device 1010	Device 1001	Device 1000

2.4.2. CEC Receive Data Block

The host reads a single opcode byte, along with up to 15 operands, from the CEC Data Receive Block. CPI currently implements a three-level receive FIFO, so that multiple messages can be received and stored before the host is required to service the subsystem.

FIFO operation is controlled as follows.

1. Host reads 0xAD to determine both the number of frames in the FIFO and the number of operands in the current frame.
2. Host burst reads 0xAE up to the number of operands that are expected to be present.
3. Once the read is complete and the host has no further need for the current data, it writes a value of 01 to offset 0xAC, which clears that entry from the FIFO.

Clearing the FIFO. For core revisions 1.3 and earlier, if the Rx FIFO switches one level back (e.g. level 2 → level 1) by a CLR_RX_FIFO_CUR command at the same time a message is being received, the message may not be stored completely depending on the timing.

Auto Clear Message Available Status. The AUTO_CLR_RX_MSG_AVAIL bit is available only in core revisions 1.4 and later. When activated, it allows the Message Available status bit (offset 0xA6[1]) to be automatically cleared once the Rx FIFO is empty. AUTO_CLR_RX_MSG_AVAIL is a toggle bit; it is set or cleared by writing 1 to toggle it from its current state, and its setting should be verified by reading back the current value.

Receipt of Own Messages. For core revisions 1.3 and earlier, the CEC core will receive its own message if the destination ID matches an ID configured in the CEC_CAPTURE_ID register. Software must discard these messages. Core revisions 1.4 and later do not receive their own transmissions except in Snoop mode.

Access. Any set of registers within the group can be accessed in a burst.

Table 2.7. CEC Receive Data Block (RO)

Offset	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0xAC	CEC_RX_CONTROL							
	RSVD	AUTO_CLR_RX_MSG_AVAIL[1] Self-clear Message Available interrupt if Rx FIFO empty Read: 0 = Manual clr 1 = Auto clear Write: 0 = No effect 1 = Toggle prev. setting	RSVD	RSVD	RSVD	RSVD	CLR_RX_FIFO_ALL Clear All Frames from Rx FIFO 0 = No 1 = Yes (self-resetting bit)	CLR_RX_FIFO_CUR Clear Current Frame from Rx FIFO 0 = No 1 = Yes (self-resetting bit)
0xAD	CEC_RX_COUNT							
	RSVD	CEC_RX_CMD_CNT (RO) CEC Receive FIFO Frame Count – returns the number of frames awaiting reading in the FIFO, 0-3.			CEC_RX_BYTE_CNT (RO) CEC Receive Byte Count – returns the number of operands in the current frame.			
0xAE	CEC_RX_CMD_HEADER							
	CEC_RX_INIT [3:0] CEC Initiator ID – identifies the initiator of the current frame.				CEC_RX_DEST [3:0] CEC Destination ID – identifies the intended target of the current frame.			

Note: This bit is available only in core rev. 1.4 and later.

Offset	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0xAF	CEC_RX_OPCODE							
0xB0	CEC_RX_OPERAND[0]							
0xB1	CEC_RX_OPERAND[1]							
0xB2	CEC_RX_OPERAND[2]							
0xB3	CEC_RX_OPERAND[3]							
0xB4	CEC_RX_OPERAND[4]							
0xB5	CEC_RX_OPERAND[5]							
0xB6	CEC_RX_OPERAND[6]							
0xB7	CEC_RX_OPERAND[7]							
0xB8	CEC_RX_OPERAND[8]							
0xB9	CEC_RX_OPERAND[9]							
0xBA	CEC_RX_OPERAND[10]							
0xBB	CEC_RX_OPERAND[11]							
0xBC	CEC_RX_OPERAND[12]							
0xBD	CEC_RX_OPERAND[13]							
0xBE	CEC_RX_OPERAND[14]							

2.4.3. Feature Abort Registers

The CEC specification allows for the Feature Abort Message as a response to unsupported command requests. The Feature Abort Message consists of 0x00, followed by the opcode of the aborted message, followed by 0x00.

For core revisions 1.1 (CPI 0x83=0x11) and later, a set of registers provides a bit-mapped field corresponding to the 255 possible opcodes that can be supported. To remain firmware-compatible with previous revisions, the bits default to "supported" for all 255 opcodes except for opcode 0xFF, whose default setting is "not supported".

Note: Firmware must set to "supported" any opcode that is sent in a Broadcast message, given that it is not appropriate to abort Broadcast messages.

Table 2.8. CEC Feature Abort Registers

Offset	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0xC0	CEC_ABORT_OPCODES [255:0][1]							
	Opcode 0000 0111	Opcode 0000 0110	Opcode 0000 0101	Opcode 0000 0100	Opcode 0000 0011	Opcode 0000 0010	Opcode 0000 0001	RSVD
0xC1 . . 0xDE	All bits: 0 = Opcode is supported; accept message in receive buffer 1 = Opcode is not supported; send "Feature Abort" when opcode is received							
0xDF	Opcode 1111 1111	Opcode 1111 1110	Opcode 1111 1101	Opcode 1111 1100	Opcode 1111 1011	Opcode 1111 1010	Opcode 1111 1001	Opcode 1111 1000

Note: These registers are available only in core revisions 1.1 (CPI 0x83=0x11) and above.

2.5. Interrupt Service Registers

CPI can be configured to generate an interrupt to the host to notify it of various events. The host driver can either poll for activity or use an interrupt handler routine. Hardware interrupts are signaled to the host on the chip interrupt pin, typically called CINT# but in some cases called RINT#, INT#, etc. In the following discussion, CINT# refers to whatever the interrupt pin is called for the chip in use.

2.5.1. Normal Events

Message Incoming. The CEC subsystem has sensed activity on the CEC line destined for this target device. Host software can poll to await completion or can wait for the Message Available interrupt.

Message Available. The Receive Message FIFO `CEC_RX_CMD_CNT` has gone from 0 to 1. Note that if this bit is cleared while the command count is 1 or greater, receipt of additional frames will **not** cause this bit to be set again; therefore, software should check the command count to ensure all frames have been read. For core revisions 1.4 and higher, the `AUTO_CLR_RX_MSG_AVAIL` bit (CPI 0xAC[6]) can be used to auto-clear the Message Available interrupt once the FIFO has been emptied.

Transmitter FIFO Empty. The CEC subsystem has completed sending the message that was previously loaded. Host software can load an additional message.

Power Status Change. A Standby message has been received. This interrupt is not currently implemented.

Message Sent Event. A message has just been loaded for transmission into the internal Transmit Buffer, or a message in the Transmit Buffer has just been sent out and ACKed. The buffer full/empty status can be checked by looking at the Transmit Buffer Full bit.

2.5.2. Irregularity Events

The following interrupts are for handling CEC line or device irregularity or error conditions. Additional relevant information is available in the Debug registers.

Receiver FIFO Overrun Error. The FIFO is full, and a new message destined for this device has just been detected but was lost.

Start Bit Irregularity. The Start Bit Low or Duration count is outside the expected range. The garbled message will be kept in the FIFO. The message will be NACKed.

Frame Retransmit Count Exceeded. The number of transmit attempts has reached the maximum allowed. This number is either fixed at 5, or can be set at 0xA0[2:0] in core revision 1.1 (CPI 0x83=0x11) and later.

Note: Clearing the interrupt allows the transmission attempts to start again. To avoid the re-send attempt, use the Flush Tx FIFO bit in register 0x87 before clearing the interrupt.

Short Pulse Detected. A LOW period on CEC was detected that was less than the minimum allowable duration. The garbled message will be kept in the FIFO. The message will be NACKed.

2.5.3. Status Indicators

The following instantaneous status bits are provided so that the host can poll to monitor the situation real-time. They can be read at any time, and do not require the host to write back to clear them.

Irregularity Status. This bit was identified as a potential Hot Plug interrupt in previous revisions of this document, but no feature was implemented. Starting from core revision 1.1 (CPI 0x83=0x11), the 0xA6[4] bit is used to indicate that at least one of the bits is active in 0xA7. This timesaving feature allows the interrupt handler to read a single byte register at 0xA6 to determine interrupt status, reading the second byte register at 0xA7 only if 0xA6[4] indicates it is needed.

Transmitter Buffer Full. This bit indicates whether the CEC subsystem has a message in its internal transmit buffer that it is attempting to send out. The Message Sent Event is only an indication that the internal transmit buffer status changed, not whether it became full or empty; the host uses Transmitter Buffer Full to make that determination.

CEC Line Current State. This bit returns the current state of the CEC pin.

2.5.4. Interrupt Enable Register

The Interrupt Enable register selects system events that should generate interrupts to the host processor. Writing any bit to 1 enables the interrupt source, the 'source' being the corresponding bit in the Interrupt Status Register; any enabled source bit of 1 will be passed through as a LOW on the CINT# line. Writing an enable bit to 1 also clears any pending interrupt. Writing 0 disables the Interrupt Status Register bits from being passed out the CINT# line. Note that writing 0 to disable the interrupt does not clear any previously pending interrupt status.

Access: This register is accessed as a single byte.

Table 2.9. CPI Interrupt Enable (R/W)

Offset	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	CEC_INT_ENABLE[15:0]							
0xA4	RSVD	RSVD	**Tx:** Message Sent event 0 = Disable 1 = Enable	RSVD	RSVD	**Tx:** Transmitter FIFO Empty event 0 = Disable 1 = Enable	**Rx:** Message Available event 0 = Disable 1 = Enable	**Rx:** Message Incoming event 0 = Disable 1 = Enable
0xA5	—	—	—	—	**Rx:** Rx FIFO Overrun Error event 0 = Disable 1 = Enable	**Rx:** Short Pulse Detected event 0= Disable 1 = Enable	**Tx:** Frame Retransmit Count Exceeded event 0 = Disable 1 = Enable	**Rx:** Start Bit Irregularity event 0 = Disable 1 = Enable

Note: Bit available only for core revisions 1.1 (CPI 0x83=0x11) and above.

2.5.5. Interrupt Status Register

The Interrupt Status Register shows the current status of interrupt events, even if the event has been disabled (through the Interrupt Enable Register) from generating a physical interrupt on CINT#. Write 1 to interrupt bits to clear the 'pending' status. Writing 0 has no effect. Bits 4, 6, and 7 serve only to show the current state and cannot be cleared directly.

Access: This register is accessed as a single byte.

Table 2.10. CPI Interrupt Status (Read status / Write '1' to clear)

Offset	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	CEC_INT_STATUS[15:0]							
0xA6	CEC line current state (RO) 0 = Low 1 = High	**Tx:** Transmit Buffer Full (RO) 0 = No 1 = Yes	**Tx:** Message Sent event pending 0 = No 1 = Yes	Irregularity status (any bit =1 in 0xA7)[1](RO) 0= No 1= Yes	RSVD	**Tx:** Transmitter FIFO Empty event pending 0 = No 1 = Yes	**Rx:** Message Available event pending 0 = No 1 = Yes	**Rx:** Message Incoming event pending 0 = No 1 = Yes
0xA7					**Rx:** Rx FIFO Overrun Error event pending 0 = No 1 = Yes	**Rx:** Short Pulse Detected event pending 0 = No 1 = Yes	**Tx:** Frame Retransmit Count Exceeded event pending 0 = No 1 = Yes	**Rx:** Start Bit Irregularity event pending 0 = No 1 = Yes

Note: Bit available only for core revisions 1.1 (CPI 0x83=0x11) and later. This bit is a status indicator, reflecting a logical OR of all bits in 0xA7. It goes to 0 only when all bits in 0xA7 have been cleared.

2.6. Interrupt Operation

The Interrupt function can be optionally added to the CINT# signal after startup. If the host enables interrupt sources by setting bits in registers 0xA4-A5, one or more events from an enabled source causes a LOW pulse on the CINT# line. The host will use the low signal to recognize and service the interrupt. This sequence is illustrated in Figure 2.2.

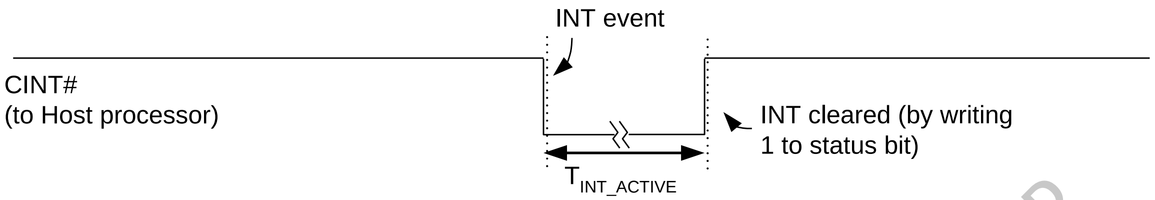

Figure 2.2. CINT# Pulse on Event

When the host clears the interrupt event (by writing 1 to the triggering Interrupt Status Register bit), the CINT# line will go high, as long as there are no more active events.

2.6.1. Tx Interrupt Event Service Procedures

The CPI host typically implements an interrupt-driven handler to manage transmission.

Transmitter FIFO Empty. This event indicates that the transmit buffer is available for writing.

Actions:

1. Clear the interrupt.
2. If a message is available for transmission, write it to the CEC_TX_OPCODE and CEC_TX_OPERAND registers.
3. Activate the message transmission by writing TX_CNT or by using the automatic count feature.

The host can then exit the interrupt handler and await the next event to send subsequent messages.

Frame Retransmit Count Exceeded. Once a transmission has been activated, if after five attempts no ACK is received, the Frame Retransmit Count Exceeded event pending bit will be set, and an interrupt generated if enabled. Further transmissions will be blocked until the status bit is cleared.

Actions:

1. Identify the message that is was not ACKed by using the TX_BFR_ACCESS bit to read back the internal transmit buffer. (Be sure to restore TX_BFR_ACCESS before proceeding.)
2. Clear the interrupt to restart transmission, possibly using the “Flush Tx FIFO” function in the Debug Register group first to remove the message before restarting.
3. Take appropriate action to handle the condition (possibly signaling an error to the user).

2.6.2. Rx Interrupt Event Service Procedures

The CPI host typically implements an interrupt-driven handler to manage reception of messages.

Message Incoming. This event indicates that there is frame activity on the CEC bus that matches the programmed destination device, and that the host can poll to determine whether a complete message is available for parsing.

Utilization of this interrupt is optional and depends on the host capabilities. CEC specifies a desired maximum response time of 200 ms, with an absolute requirement for response within 1 s, so only a host with large interrupt response latency would need to use this interrupt.

Actions:

1. Clear the interrupt.
2. Set a regular call-back procedure to periodically check CEC_RX_CMD_CNT after some period of time has elapsed.
3. On call-back, read CEC_RX_CMD_CNT. Once it becomes non zero, a completely received frame is ready to be read from the message buffer.
4. Follow the Message Available routine below starting from step 2.

The call-back procedure can be disabled once the Message Incoming interrupt has stayed inactive for a certain period of time.

Message Available. This event indicates that at least one complete message is available for processing. If the host can service interrupts with low latency, it is possible to operate in an interrupt-driven mode using this interrupt to determine when to read the buffer.

Actions:

1. Clear the interrupt.
   Note: Core revisions 1.4 and later introduce a new feature bit that allows this interrupt to self-clear once the last message has been read. Refer to the description for CEC_RX_CONTROL (register offset 0xAC).
2. Read the relevant message information.
3. Write CLR_RX_FIFO_CUR to remove the current entry from the FIFO.
4. Check CEC_RX_CMD_CNT for possible additional messages before returning.

Rx FIFO Overrun Error. Receipt of this event means that the FIFO is already full with 3 messages, and that a new message to that destination was detected but could not be stored and was not ACKed.

This error indicates that the host has not been successful in keeping up with bus traffic. CEC allows messages to be NACKed, in which case the initiator will retry later. Therefore, aside from clearing this interrupt and possibly increasing the priority with which the host handles future Receiver FIFO interrupts, no service is required.

2.7. Debug Registers

The Debug registers provide additional information to help troubleshoot bus-related problems. High-level code should not need to depend on these registers for normal operation.

Start Bit Low, Duration Periods. These periods are measured in units that differ depending on the core and product, due to the different clocking source oscillators used.

CEC Snoop Initiator, Current CEC Bus Owner. These values allow the activity on the bus to be monitored. If the CEC Snoop bit is set, CPI will capture CEC activity on the bus from specified device(s). However, the device will **not** ACK these frames – it will simply put the captured data into the receive FIFO for retrieval and analysis by the host. Two variants of this feature are supported.

Single Initiator Snooping. The variant prior to core rev. 1.4 operated as follows:

- Allows Single Initiator Snooping only. Can only snoop on messages sent from the one Initiator Address that is specified via CEC Snoop Initiator Register 0x86[7:4].
- Un-ACKed messages from that initiator, and messages from all other initiators, are not captured.
- While the snooping is enabled (CEC Snoop Bit = 1), the device cannot ACK or receive messages targeted for devices selected in the CEC Capture ID Registers 0xA2 and 0xA3. Therefore Snooping and normal CEC operation cannot co-exist.

Multiple Initiator Snooping. The above method is not as convenient as the variant introduced in rev. 1.4 and later:

- Allows Multiple Initiator Snooping. In this new usage model, when the Snoop bit is set, CEC Capture ID Registers 0xA2 and 0xA3 are redefined to select *initiators* rather than targets. Now, rather than being limited to the one initiator specified in 0x86[7:4], the logic can capture messages from multiple initiators.
- Messages from specified initiators are always captured, regardless of whether some other device has ACKed them or not, and regardless of their intended target.

- Setting Snoop blocks the logic from generating ACK for these selected initiators. However, CEC Snoop Initiator Register 0x86[7:4] is re-defined to pick one initiator whose messages actually do get ACKed. This approach allows the device to continue acting like a normal CEC target, but only as a single device.
- Note that the initiator specified in 0x86[7:4] must also be specified in 0xA3:A2.
- In order to disable all ACK during Snoop mode operation, 0x86[7:4] can be set to 0x0F (broadcast).

Note that software using the Snoop feature must be aware of these changes. The functionality of core rev. 1.4 and later is **not** software backwards-compatible with the earlier method.

ACK/NACK Header Block. During normal operation, CPI automatically ACKs accesses directed to CEC_INIT_ID (offset 0x88). The ACK/NACK bit can be set to not acknowledge these accesses, even while capturing them in the receive FIFO.

Invert ACK to Broadcast Messages. During normal operation, the correct response to Broadcast messages is to leave the CEC line HIGH. Setting this bit forces CPI to NACK these messages by driving the CEC line LOW.

Frame Retransmit Count. Returns the number of tries attempted on the last transmission.

Flush Tx FIFO. If there is a Frame Retransmit Count Exceeded event, the interrupt bit will be set and the transmit attempts will be stopped until the interrupt is cleared. While the interrupt bit is still set, the Flush Tx FIFO bit can be written to clear the FIFO, so that no attempt will be made to retransmit on clearing the interrupt bit.

Access. These registers are accessed as single bytes.

Table 2.11. Debug Registers (R/W)

<table>
<tr><th>Offset</th><th>Bit 7</th><th>Bit 6</th><th>Bit 5</th><th>Bit 4</th><th>Bit 3</th><th>Bit 2</th><th>Bit 1</th><th>Bit 0</th></tr>
<tr><td><b>0x84</b></td><td colspan="8">Start Bit Low Period. Expected range is 3.5 ms to 3.9 ms.
Core rev. before 1.0: Measured in units of 250us±15us. Expected range 0x0E-0F.
Core rev. 1.0 and later: Measured in units of 19us±1us. Expected range 0xB8-CD.</td></tr>
<tr><td><b>0x85</b></td><td colspan="8">Start Bit Duration Period. Expected range is 4.3 ms to 4.7 ms.
Core rev. before 1.0: Measured in units of 250us±15us. Expected range 0x11-12.
Core rev. 1.0 and later: Measured in units of 19us±1us. Expected range 0xE2-F7.</td></tr>
<tr><td><b>0x86</b></td><td colspan="4">CEC Snoop Initiator (RW).
When CEC Snoop is enabled:
– Core rev. before 1.4: values 0000 to 1111 indicate the device that will be snooped. 0xA2-A3 are ignored.
– Core rev. 1.4 and later: values 0000 to 1111 indicate the device that will be ACKed. Devices indicated in 0xA2-A3 are captured but not ACKed.</td><td colspan="4">Current CEC bus owner (RO). Values 0000 to 1111 indicate the identity of the device that most recently controlled the CEC bus.</td></tr>
<tr><td><b>0x87</b></td><td>Flush Tx FIFO
0 = No
1 = Yes
Self-resetting bit</td><td colspan="3">Frame Retransmit Count (RO). Values 0 to 5 indicate the number of retries attempted on the most recent transmission.</td><td>RSVD</td><td>Invert ACK to Broadcast messages
0 = No
1 = Yes</td><td>ACK/NACK Header Block
0 = ACK
1 = NACK</td><td>CEC Snoop
0 = Disable
1 = Enable</td></tr>
</table>

Appendix A – CEC Calibration

Timing requirements for CEC must be strictly controlled and monitored. The CPI hardware manages most of this task. For core revisions 1.0 and later used in receiver products, the calibration is **automatic** at reset, and use of the registers and sequences listed below is **not required**.

However, initial calibration and periodic re-calibration are required for core revisions 0.x (rev. value read from 0x83 has its upper nibble = 0); the calibration must be performed under host control.

A.1. Setup and Calibration Registers

The CEC subsystem implementing CPI provides a means for initial calibration and periodic re-calibration of the timing.

Note: Due to host chip architectural differences, the CEC_SETUP bits reside in **different locations** and have **different polarities** on different chips. Software must be manually configured for these differences. Bit locations for some products are listed below; for those not listed, refer to the *Programmer Reference* for that device.

Table A.1. CEC Setup for SiI9022/9024 and 9220/9222 Tx (R/W)

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Device 0xC0 — **Offset 0x8E** [10]	CEC_SETUP							
	—	—	—	CEC_EN CPI Operation Control 0 = CPI Enabled 1 = Do not use this setting (default)	—	—	CALIB Calibration Enable Control 0 = Normal operation 1 = Enable calibration (write-only; this bit always reads back 0)	SW_CALIB Start/Stop Calibration by Software 0 = Normal operation 1 = Calibrate

Table A.2. CEC Setup for SiI9022/9024A (R/W)

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Device 0xC0 — **Offset 0x8E** [10]	CEC_SETUP							
	—	—	—	CEC_EN CPI Operation Control 0 = Do not use this setting 1 = CPI Enabled (default)	—	—	CALIB Calibration Enable Control 0 = Normal operation 1 = Enable calibration (write-only; this bit always reads back 0)	SW_CALIB Start/Stop Calibration by Software 0 = Normal operation 1 = Calibrate

Table A.3. CEC Setup for SiI9181/9185 Switch (R/W)

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Device **0xD0** — **Offset** **0x08** [00]	CEC_SETUP							
	—	CEC_EN CPI Operation Control 0 = Pass-through mode 1 = CPI Enabled	—	—	—	—	—	—
Device **0xD0** — **Offset** **0x09** [00]	—	—	—	—	—	—	—	CALIB Calibration Enable Control 0 = Normal operation 1 = Enable calibration

Table A.4. CEC Setup for SiI9187/9285/9287/9385/9387/9389 Switch (R/W)

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Device **0xC0** — **Offset** **0x8E** [14]	CEC_SETUP							
	—	—	—	CEC_EN CPI Operation Control 0 = CPI Enabled (default) 1 = Pass-through mode	—	—	CALIB Calibration Enable Control 0 = Normal operation 1 = Enable calibration (write-only; this bit always reads back 0)	SW_CALIB Start/Stop Calibration by Software 0 = Normal operation 1 = Calibrate

Table A.5. CEC Setup for SiI9155/9127/9223/9233 Rx (R/W)

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Device **0xC0** — **Offset** **0x8E** [14]	CEC_SETUP							
	—	—	—	CEC_EN CPI Operation Control 0 = CPI Enabled 1 = Pass-through mode (default)	—	NO_CALIB Prevents automatic calibration from taking place 0 = Normal operation 1 = Block calibration	CALIB Calibration Enable Control 0=Normal operation 1 = Enable calibration (write-only; this bit always reads back 0)	SW_CALIB Start/Stop Calibration by Software 0 = Normal operation 1 = Calibrate

Table A.6. CEC Setup for SiI9251/9261 Rx (R/W)

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Device 0xC0 —— **Offset 0x8E** [04]	CEC_SETUP							
	—	—	—	CEC_EN CPI Operation Control 0 = CPI Enabled (default) 1 = Pass-through mode	—	NO_CALIB Prevents automatic calibration from taking place 0 = Normal operation 1 = Block calibration	CALIB Calibration Enable Control 0 = Normal operation 1 = Enable calibration (write-only; this bit always reads back 0)	SW_CALIB Start/Stop Calibration by Software 0 = Normal operation 1 = Calibrate

A.2. Initial CEC Calibration (core revisions prior to 1.2)

The CPI solution provides two methods to perform calibration: hardware and software.

Note: The SiI9181/9185 Switch offers only the Hardware Calibration method.

Hardware Calibration. The hardware calibration method requires that the design have hardware control of the CEC_D pin, which is used to define the start and end of the calibration cycle.

1. Make sure the SW_CALIB bit is 0 and the CEC_D input is at a high state.
2. Write the CALIB bit to 1.
3. Drive the CEC_D input to a low state.
4. After 10ms +/-1%, return the CEC_D input to a high state.

The CALIB bit clears itself automatically after CEC_D goes high.

Software Calibration. The software calibration method uses two I^{2}C transactions to define the start and end of the calibration cycle.

1. Make sure the SW_CALIB bit starts at 0 and the CEC_D input remains at a high state.
2. Write the CALIB bit to 1.
3. Write the SW_CALIB bit to 1 (alternatively, both SW_CALIB and CALIB can be set to 1 in the same cycle).
4. After 10ms +/-1%, write the SW_CALIB bit to 0.

The CALIB bit clears itself automatically after the SW_CALIB bit is written to 0.

The host must have precise control over the timing of the I^{2}C operations, since the calibration pulse width is measured from the rising edge of the ACK of the first cycle to the rising edge of the ACK of the second cycle.

A.3. Periodic CEC Re-Calibration (core revisions prior to 1.0)

Calibration establishes the correct timing for CEC protocols under the current operating conditions. These timings are derived from an internal ring oscillator whose frequency varies with:

- Process. This parameter is decided by the production lot from which that particular sample of silicon was obtained.
- Temperature. The chip operating temperature will increase significantly after power-up. Ring oscillator frequency decreases with increasing temperature.
- Voltage. The chip operating voltage may be regulated above or below the “typical” value, even while staying within specifications. Ring oscillator frequency decreases with increasing Vcc.

Initial calibration establishes the correct timing according to the production lot, a process characteristic that does not vary over time for a particular sample. It also optimizes timing for the typical operating voltage of that design within the allowed tolerance range.

Re-calibration, however, is required for temperature compensation. It should be applied at an interval that corresponds to expected significant changes in the chip operating temperature.

Note: Core revisions 1.0 (CPI 0x83=0x10) and higher do not require any re-calibration, only a single calibration after hardware reset.

A.3.1. Effect of Temperature

When the silicon is calibrated at a nominal temperature of 25°C, raising or lowering the temperature will have an effect as follows.

- When the temperature is raised from 25°C to 85°C, the Start Bit and Start Bit Low durations fall out of specification (they increase to roughly 4.9 ms and 4.0 ms, respectively). The duration of data bits also increases but stays within specification.
- When the temperature is lowered from 25°C to −10°C, the Start Bit and Start Bit Low durations remain within specification.

As noted above, the ring oscillator circuit of the device decreases in frequency as temperature increases.

A.3.2. Deciding on Re-calibration Interval

The calibration process must be repeated periodically to ensure that the Start Bit and Start Bit Low remain within specification. For every 15°C change in case temperature, the ring oscillator changes frequency and a recalibration of the CEC timing is recommended. The system designer should determine the rate at which the overall design changes 15°C, and set the frequency of calling the HDMI CEC calibration routine as needed.

Calibration can begin any time that the Transmit FIFO is empty. To re-calibrate the timing:

1. Verify that the Transmit FIFO is empty (CPI 0xA6[2]).
2. Disable the Transmit FIFO Is Empty interrupt (CPI 0xA4[2]), since there is no need to trigger an interrupt once the Transmit FIFO is empty.
3. Perform CEC Calibration as described in the “Initial CEC Calibration” section.
4. Re-enable the Transmit FIFO Empty interrupt if desired.

The receive portion of the CEC logic continues to receive data as usual during calibration. Therefore, the ACK and no-ACK situations will be handled by the chip appropriately during calibration.

A.3.3. Effects of Re-calibration

An error can occur if a calibration cycle completes in the middle of message reception, but only if the newly calculated timing value is very different from the previous value. This can generally occur only if the elapsed time between calibration cycles has been too long and a large temperature change has caused a significant change to the oscillator frequency.

Targeted Message Reception. For a message directed to a specified target device, there is risk of “temporary” CEC data loss during a calibration cycle. It is considered temporary because if a targeted frame comes in during calibration and an error occurs, the receiving device will not ACK the cycle and the sending device will retry the transmission. By the time the retry attempt is made, calibration will always be finished.

- The minimum frame is 8 data bits long and it takes 29.5 ms to send (minimum frame time = start bit + 8 data bit + EOM bit + ACK bit = 4.5 ms + 8 * 2.4 ms + 2.4 ms + 2.4 ms = 29.5 ms).
- HDMI CEC Standard version 1.3a requires 1 retry minimum, ensuring that a retry will be attempted.
- By the time the retry attempt is made, 36.7 ms has passed and the calibration cycle will have been completed (minimum frame time + retry free time = 29.5 ms + 3 * 2.4 ms = 36.7 ms).

This worst-case scenario will not occur as long as regular re-calibration is applied.

Broadcast Message Reception. For a broadcast message, there is no requirement for an ACK from a specific receiving device. Therefore, if calibration causes a significant change in the timing values used such that the receiver cannot recover the broadcast cycle, there will be no way to force a retry after the fact.

Regular re-calibration is highly recommended to prevent this situation.

Appendix B – Capability Discovery and Control (CDC)

HDMI 1.4 defines new requirements for CEC operation, in regards to the discovery and use of new Home Ethernet Channel (HEC) and Audio Return Channel (ARC) features. The messaging used over CEC for this operation is known as Capability Discovery and Control (CDC).

Starting from revision 0x1A (and also on revision 0x16), there is hardware support provided to automatically recognize CDC messages. To support CDC messages, set CPI 0xE3=0x83 and leave CPI 0x84 set to its default value of 0xF8.

Table B.1. CDC Arbitration Count (R/W)

Offset	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0xE3 [03]	CDC_ARB							
	CDC Arbitration 0 – Disable 1 – Enable	—	—	CDC Arbitration Count – number of arbitration blocks after the CEC Header				
0xE4 [F8]	CDC_OPCODE							
	CDC Opcode to indicate CDC message Default 0xF8							

Revision History

Revision B, June 30, 2017

Updated to the latest Lattice Semiconductor template.

Revision A05, September 23, 2009

Added revision cross-references for new products.

Revision A04, January 9, 2009

- Corrected Enable register location for Switch devices.
- Added revision cross-references for new products.
- Added note about using Feature Abort with Broadcast messages.

Revision A03, September 8, 2008

- Added descriptions for core revision 1.4 features.
- Added new Snoop functionality description.

Revision A02, August 15, 2007

Added various features for new CPI core revisions.

Revision A01, June 12, 2007

Corrected description of calibration function

Revision A, June 5, 2007

First release of this Programmer Reference.

7th Floor, 111 SW 5th Avenue
Portland, OR 97204, USA
T 503.268.8000
www.latticesemi.com