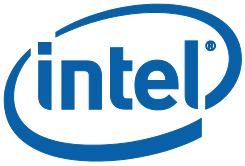


Enhanced Serial Peripheral Interface (eSPI) ECN

Engineering Change Notice

TITLE	Change Alert# pin behavior
DATE	10 January 2014
AFFECTED DOCUMENT	eSPI Base Specification Rev 0.75
DISCLOSURE RESTRICTIONS	CNDA or eSPI Click to Accept License Agreement

Intel hereby grants you a fully-paid, non-exclusive, non-transferable, worldwide, limited license (without the right to sublicense), under its copyrights to view, download, and reproduce the Enhanced Serial Peripheral Interface (eSPI) Specification ("Specification"). You are not granted any other rights or licenses, by implication, estoppel, or otherwise, and you may not create any derivative works of the Specification.

The Specification is provided "as is," and Intel makes no representations or warranties, express or implied, including warranties of merchantability, fitness for a particular purpose, non-infringement, or title. Intel is not liable for any direct, indirect, special, incidental, or consequential damages arising out of any use of the Specification, or its performance or implementation.

Intel retains ownership of all of its intellectual property rights in the Specification and retains the right to make changes to the Specification at any time. No license is granted to use Intel's name, trademarks, or patents.

If you provide feedback or suggestions on the Specification, you grant Intel a perpetual, non-terminable, fully-paid, nonexclusive, worldwide license, with the right to sublicense, under all applicable intellectual property rights to use the feedback and suggestions, without any notice, consent, or accounting. You represent and warrant that you own, or have sufficient rights from the owner of, the feedback and suggestions, and the intellectual property rights in them, to grant the above license.

This agreement is governed by Delaware law, without reference to choice of law principles. Any disputes relating to this agreement must be resolved in the federal or state courts in Delaware and you consent, and will not object, to the exclusive personal jurisdiction of the courts in Delaware.

This agreement is the entire agreement of the parties regarding the Specification and supersedes all prior agreements or representations.

This agreement is hosted at the following location: http://downloadcenter.intel.com/Detail_Desc.aspx?agr=Y&DwnldID=21353

THIS SPECIFICATION IS PROVIDED "AS IS" WITH NO WARRANTIES WHATSOEVER INCLUDING ANY WARRANTY OF MERCHANTABILITY, FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATION, OR SAMPLE.

Except for a limited copyright license to copy this specification for internal use only, no license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted herein.

Intel Corporation and the authors of this specification disclaim all liability, including liability for infringements of proprietary rights, relating to implementation of information in this document and the specification. Intel Corporation and the authors of this specification also do not warrant or represent that such implementation(s) will not infringe such rights.

Implementations developed using the information provided in this specification may infringe the patent rights of various parties including the parties involved in the development of this specification. Except as expressly granted hereunder, no license, express or implied, by estoppel or otherwise, to any intellectual property rights (including without limitation rights under any party's patents) is granted.

All suggestions or feedback related to this specification become the property of Intel Corporation upon submission.

Intel may make changes to the specifications, product descriptions, and plans at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them.

This document is an intermediate draft for comment only and is subject to change without notice. Do not finalize a design based on this document.

Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

* Other names and brands may be claimed as the property of others.

ECN Motivation

eSPI Base Specification 0.75 defines 2 ways of signaling Alert# i.e. through I/O[1] pin or Alert# pin. In the Single Master-Multiple Slaves configuration, Alert# pin must be used. eSPI slave must support both types of Alert mechanism. The Alert mechanism in use is platform specific.

The spec defines consistent Alert# behavior whether the event is signaled through I/O[1] or Alert# pin. By having Alert# pin as an open-drain pin, the spec intends to allow a wired-OR implementation where Alert# pin could be shared by multiple slaves.

Among the Alert# pin behaviors are as follow (same as when Alert# event is signaled through I/O[1]):

- In Section 3.3 – Pin Description
  - Alert# pin is an open-drain output from the slave
  - Alert# pin requires a weak pull-up to be enabled on the pin
- In Section 4.1 – Basic Protocol
  - The Alert event can only be signaled by the slave when CS# is high.
  - The Alert# pin is toggled from tri-state to pulled low by the slave when it decides to request for service.
  - The slave holds the state of the pin until CS# is asserted by the master.
  - Once the CS# is asserted, the slave must release the ownership of the pin by tri-stating the pin.
  - The pin will be pulled high by the weak pull-up.
  - At the last falling edge of the serial clock after CRC is sent, the slave must drive Alert# pin to high until CS# is deasserted to prevent false Alert# event generated after CS# deassertion.
  - After CS# deassertion, the pin is tri-stated by the slave after meeting the t_{SHQZ} Output Disable timing
  - To signal an alert event after CS# deassertion, the slave is only allowed to re-assert the Alert# pin after the t_{SHAA} timing.
  - The Alert event signaled on the pin is asynchronous to the Serial Clock.

The Alert# pin behavior above has several shortcomings to support a wired-OR implementation where Alert# pin could be shared by multiple slaves:

- The need for the slave to drive Alert# pin high at the last failing edge of the serial clock after CRC is sent until CS# is deasserted, will result in contention on the Alert# pin if other slaves are driving the pin low for signaling Alert# event.
- Alert# pin is an input to the master. Without having master to drive the Alert# pin high for the first clock during turn-around will potentially result in false Alert# event where the pin may not able to be pulled high fast enough before CS# deassertion such as in the worst case when transaction is terminated with NO_RESPONSE. On the other hand, the master must not be allowed to drive the Alert# pin high to avoid contention as the pin may be driven low by other slaves for signaling Alert# event.

This ECN addresses the above shortcomings by re-defining the Alert# pin behavior.

ECN Description

Alert# pin is re-defined to have the following behavior:

- As a driven output from slave to master. In a Single Master-Multiple Slaves configuration, a dedicated Alert# pin is required for each of the slaves. Slave must support Alert# pin as a driven output by default. A weak pull-up is not required on Alert# pin in this mode.
- Optionally, the slave may support Alert# pin as an open-drain output. In a Single Master-Multiple Slaves configuration, this allows a wired-OR implementation where multiple slaves may share a single Alert# pin. The slave is only allowed to drive the pin low or tri-state the pin otherwise. A weak pull-up is required on Alert# pin in this mode. The weak pull-up impedance must be sized accordingly to avoid triggering false Alert# event. Slave advertises if Alert# pin as an open-drain output is supported and master configures the slave to operate in this mode.

In summary, the eSPI spec description is updated for the following:

- In Section 3.3 – Pin Description
  - Alert# pin is either a driven, or an open-drain output from the slave
  - When functions as a driven output, Alert# pin does not require a weak pull-up to be enabled on the pin unless for the purpose of terminating the pin to inactive when it is not used. When functions as an open-drain output, Alert# pin requires a weak pull-up to be enabled on the pin

- In Section 4.1 – Basic Protocol
  - The Alert event can only be signaled by the slave when CS# is high.
  - The Alert# pin is toggled by the slave from high to low (when the pin is a driven output), or tri-state to pulled low (when the pin is an open-drain output) when the slave decides to request for service.
  - The slave holds the state of the pin until CS# is asserted by the master.
  - Once the CS# is asserted, the slave must drive the Alert# pin high (when the pin is a driven output), or release the ownership of the pin by tri-stating the pin (when the pin is an open-drain output).
  - The pin will not require a weak pull-up when it is connected to the slave (when the pin is a driven output), or it will be pulled high by the weak pull-up (when the pin is an open-drain output).
  - At the last falling edge of the serial clock after CRC is sent, the slave must drive I/O[n:0] pins to high until CS# is deasserted ~~to prevent false Alert# event generated after CS# deassertion~~.
  - After CS# deassertion, I/O[n:0] pins are tri-stated by the slave after meeting the t_{SHQZ} Output Disable timing
  - ~~To signal an alert event after CS# deassertion, the slave is only allowed to re-assert the Alert# pin after the t_{SHAA} timing.~~
  - The t_{SLAZ} and t_{SHAA} timings are not applicable to Alert# pin.
  - The Alert event signaled on the pin is asynchronous to the Serial Clock.

Changes to the eSPI spec description are as follow:

3.3 Pin Descriptions

eSPI uses the existing SPI I/O buffer. The electrical specification for this new interface is the same as SPI.

eSPI Reset# is typically driven from eSPI master to eSPI slaves. The exception is when eSPI Reset# is generated by eSPI slave, which drives the eSPI Reset# to the eSPI master. eSPI Reset# is the reset to the eSPI interface on both sides.

eSPI master and eSPI slaves must tri-state the interface pins when their respective eSPI Reset# is asserted. The Chip Select#~~,~~ and I/O[n:0] ~~and Alert#~~ pins require weak pull-up to be enabled on these pins whereas the Serial Clock requires a weak pull-down. When functions as a driven output, Alert# pin does not require a weak pull-up to be enabled on the pin unless for the purpose of terminating the pin to inactive when it is not used. When functions as an open-drain output, Alert# pin requires a weak pull-up to be enabled on the pin

The weak pull-up/pull-down should be implemented either as an integral part of the eSPI master buffer or on the board. eSPI slaves must not implement the weak pull-up/pull-down. For Alert# pin configured as open-drain, it is recommended that the weak pull-up be implemented on the board such that its impedance value could be adjusted accordingly when needed.

Refer to Section 8.1 - Electrical Specification for the value of the weak pull-up/pull-down resistor.

After eSPI Reset# is deasserted on the eSPI master, the eSPI master begins driving Chip Select# and Serial Clock pins to their idle state appropriately. The weak pull-up on the Chip Select# and the weak pull-down on the Serial Clock are allowed to be disabled after the eSPI Reset# deassertion. However, I/O[n:0] and Alert# pin (open-drain) continue to have the weak pull-up enabled for the proper operation of the eSPI bus.

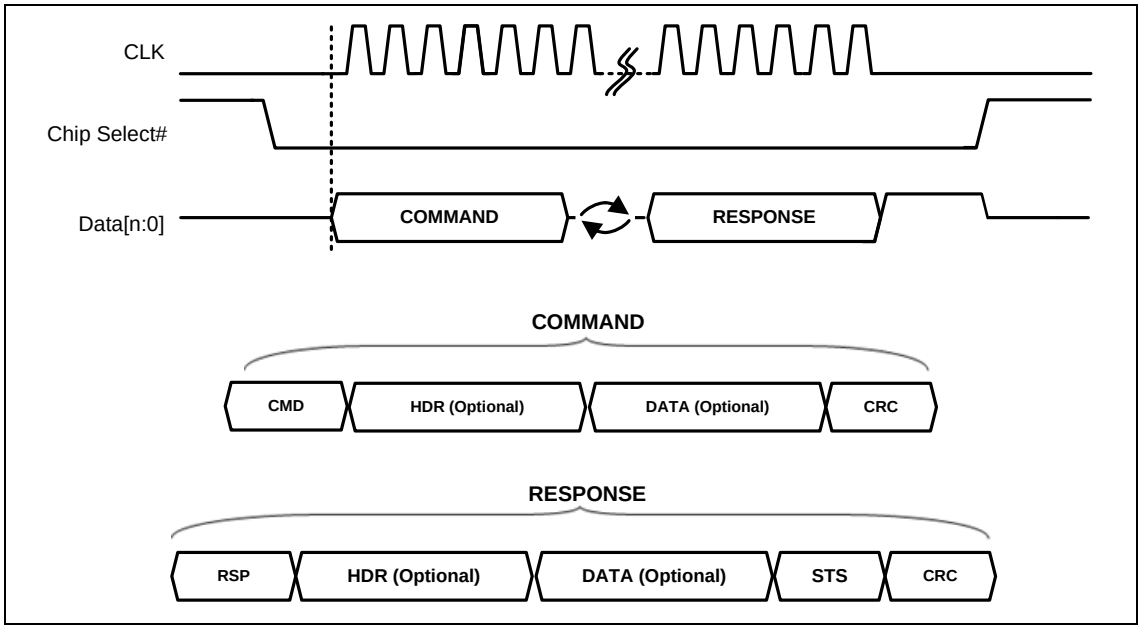
Table 1: eSPI Pin List

Pin Name	Direction	Clock	Description
eSPI Reset#	Master to Slave[1] or Slave to Master[2]	Asynchronous	**Reset#:** Reset the eSPI interface for both master and slaves. **Note:** 1. eSPI Reset# is typically driven from eSPI master to eSPI slaves. 2. eSPI Reset# is generated by eSPI slave, driven from eSPI slave to eSPI master.
Chip Select#	Master to Slave	Asynchronous	**Chip Select#:** Driving Chip Select# low selects a particular eSPI slave for the transaction. Each of the eSPI slaves is connected to a

Pin Name	Direction	Clock	Description
			dedicated Chip Select# pin.
Serial Clock	Master to Slave	-	**Clock:** This pin provides the reference timing for all the serial input and output operations.
I/O [n:0]	Bi-directional	Serial Clock	**I/O:** These are bi-directional input/output pins used to transfer data between master and slaves. The value of 'n' may be 1 or 3 depending on the I/O mode. In Single I/O mode (n=1), I/O[0] is the eSPI master output/eSPI slave input (MOSI) whereas I/O[1] is the eSPI master input/eSPI slave output (MISO).
Alert#	Slave to Master	Asynchronous	**Alert#:** This pin is used by eSPI slave to request service from eSPI master. Alert# is either a driven, or an open-drain output from the slave with default as a driven output. This pin is optional for Single Master-Single Slave configuration where I/O[1] can be used to signal the Alert event.

4.1 Basic Protocol

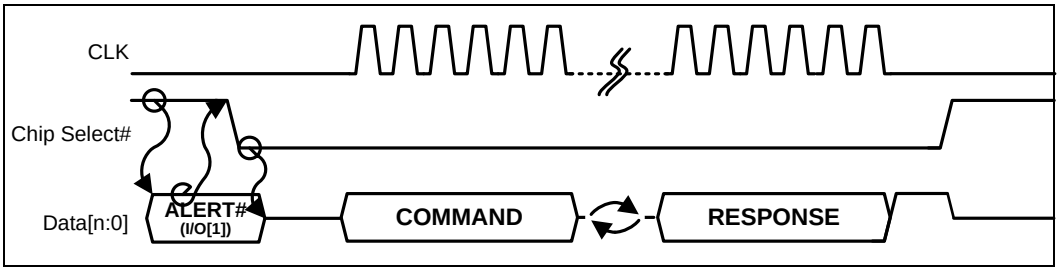
Figure 1: Basic eSPI Protocol



eSPI transaction consists of a Command phase driven by master, a Turn-Around (TAR) phase, and a Response phase driven by the slave. The Command phase consists of a CMD, an optional header (HDR), optional DATA and a CRC. The Response phase consists of a RSP, an optional header (HDR), optional data, a Status and a CRC. CRC generation is mandatory for all eSPI transactions where CRC byte is always transmitted on the bus. However, CRC checking is default disabled after reset and it is enabled by SET CONFIGURATION. When CRC checking is disabled, CRC byte is ignored by the receiver.

A transaction could be initiated by the master through the assertion of Chip Select#, start the clock and drive the command onto the data bus. The clock remains toggling until the complete response phase has been received from the slaves.

Figure 2: Slave Triggered Transaction (Single Master-Slave)



A transaction could be initiated by the slave by first signaling an Alert event to the master. The Alert event could be signaled through two ways. In the Single Master-

Single Slave configuration, the I/O[1] pin could be used by the slave to indicate an Alert event. In the Single Master-Multiple Slaves configuration, a dedicated Alert# pin is required.

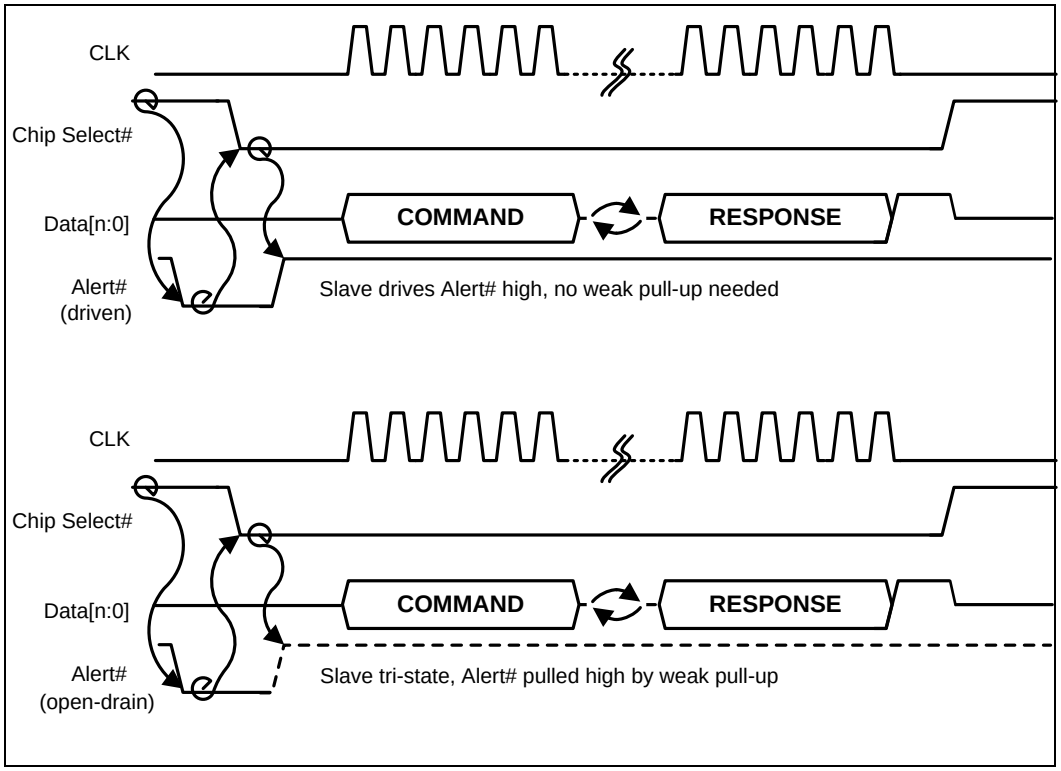
The Alert event can only be signaled by the slave when the slave's Chip Select# is high.

When ~~The pin, either~~ I/O[1] ~~or Alert#~~ is used to signal the Alert# event, it is toggled from tri-state to pulled low by the slave when ~~it~~ the slave decides to request for service. The slave then holds the state of the I/O[1] pin until the Chip Select# is asserted by the master. Once the Chip Select# is asserted, the eSPI slave must release the ownership of the I/O[1] pin by tri-stating the pin within the t_{SLAZ} timing and the pin will be pulled high by the weak pull-up. The master then continues to issue command to figure out the cause of the Alert event from the device and then service the request. At the last falling edge of the serial clock after CRC is sent, the eSPI slave must drive I/O[n:0] ~~and Alert#~~ pins to high until Chip Select# is deasserted. Besides power friendly due to weak pull-up on these pins, the driving to high ensures no false Alert# event is generated by I/O[1] when Chip Select# is deasserted. After Chip Select# deassertion, these I/O[n:0] pins are tri-stated by the slave after meeting the t_{SHQZ} Output Disable timing where the weak pull-ups maintain these pins at high, with the master continues to tri-state the I/O[n:0]. To signal an alert event after Chip Select# deassertion, the slave is only allowed to re-assert the I/O[1] ~~or Alert#~~ pin after the t_{SHAA} timing.

When Alert# pin is used to signal the Alert# event, it is toggled by the slave from high to low (when the pin is a driven output) or tri-state to pulled low (when the pin is an open-drain output) when the slave decides to request for service. The I/O[n:0] pins remain tri-stated by the slave. The slave then holds the state of the Alert# pin until the Chip Select# is asserted by the master. Once the Chip Select# is asserted, the slave must drive the Alert# pin high (when the pin is a driven output), or release the ownership of the pin by tri-stating the pin (when the pin is an open-drain output). The t_{SLAZ} timing is not applicable to the Alert# pin. The master then continues to issue command to figure out the cause of the Alert event from the device and then service the request. At the last falling edge of the serial clock after CRC is sent, the eSPI slave must drive I/O[n:0] pins to high until Chip Select# is deasserted for power friendly reason due to weak pull-up on these pins. After Chip Select# deassertion, these I/O[n:0] pins are tri-stated by the slave after meeting the t_{SHQZ} Output Disable timing where the weak pull-ups maintain these pins at high, with the master continues to tri-state the I/O[n:0]. The t_{SHAA} timing is not applicable to Alert# pin. However, when Alert# pin is configured as open-drain and asserted, the weak pull-up on the pin must be such that the assertion of the CS# for the shortest possible transaction (which causes the slave to tri-state the Alert# pin), is able to pull the Alert# pin high fast enough to the deasserted value before or by the last failing edge of the serial clock at the end of the transaction.

In the case of error condition where Chip Select# is deasserted abruptly by the master, refer to Section 9.2.1.5 for the detail.

Figure 3: Slave Triggered Transaction (Multiple Slave)



The specification does not prevent the use of a dedicated Alert# pin for the Single Master-Single Slave configuration.

In the boundary case where the Alert event assertion aligns with Chip Select# assertion, the slave still tri-state the I/O[1] pin or drive the Alert# pin high (when the pin is a driven output) or tri-state the Alert# pin (when the pin is an open-drain output) after sampling the corresponding Chip Select# assertion. The status is returned during the response phase and the master is then aware of the need to service the slave's outstanding requests.

The Alert event signaled on the pin is asynchronous to the Serial Clock.

eSPI slaves must support both types of Alert mechanism. The method to determine which Alert mechanism to use for each of the eSPI slaves is implementation specific.

eSPI is defined to use packet-based split transaction protocol. On the transmit side, the packets are formed in the Transaction Layer based on the transaction to be sent. The Link Layer extends the packet with a CRC byte.

Similarly on the receive side, the CRC is checked at the receiving Link Layer when CRC checking is enabled. Once the packet passes the CRC check, the packet is sent to Transaction Layer where it is decoded and acted upon.

7.2.1.3 Offset 08h: General Capabilities and Configurations

This register is also reset by the In-band RESET command.

Bit	Type	Default	Description
31	RW	0b	**CRC Checking Enable:** This bit is set to '1' by eSPI master to enable the CRC checking on the eSPI bus. By default, CRC checking is disabled. 0b: CRC checking is disabled. 1b: CRC checking is enabled.
30	RW	0b	**Response Modifier Enable:** This bit is set to '1' to enable the use of Response Modifier by eSPI slave to append either a peripheral (channel 0) completion, a virtual wire (channel 1) packet or a flash access (channel 3) completion to the GET_STATUS response phase. When this bit is a '0', eSPI slave must only use the Response Modifier of "00", i.e. no append. By default, the Response Modifier is disabled.
29	RO	0	**Reserved.**
28	RW	0b	**Alert Mode:** This bit serves to configure the Alert mechanism used by the slave to initiate a transaction on the eSPI interface. 0b: I/O[1] pin is used to signal the Alert event. 1b: ~~A dedicated~~ Alert# pin is used to signal the Alert event. Note: This bit can ~~only~~ be '0' or '1' in a single master-single slave topology. For single master-multiple slave topology, this bit must be programmed to '1'. Alert Mode is allowed to change from default '0' to '1' during runtime in both single or multiple slaves topologies provided when this happens, only a single slave is enabled for generating Alert# event.

<table>
<tr><th>Bit</th><th>Type</th><th>Default</th><th>Description</th></tr>
<tr><td>27:26</td><td>RW</td><td>00b</td><td>I/O Mode Select: eSPI master programs this field to enable the appropriate mode of operation, which will take effect at the deassertion edge of the Chip Select#.

The I/O Mode configured in this field must be supported by both the master and the slave. Single I/O mode is supported by default.

<table>
<tr><th>Encoding</th><th>Operating Mode</th></tr>
<tr><td>00</td><td>Single I/O</td></tr>
<tr><td>01</td><td>Dual I/O</td></tr>
<tr><td>10</td><td>Quad I/O</td></tr>
<tr><td>11</td><td>Reserved.</td></tr>
</table>
</td></tr>
<tr><td>25:24</td><td>RO</td><td>HwInit</td><td>I/O Mode Support: This field indicates the I/O modes supported by the slave.

<table>
<tr><th>Encoding</th><th>Supported I/O Mode</th></tr>
<tr><td>00</td><td>Single I/O</td></tr>
<tr><td>01</td><td>Single and Dual I/O</td></tr>
<tr><td>10</td><td>Single and Quad I/O</td></tr>
<tr><td>11</td><td>Single, Dual and Quad I/O</td></tr>
</table>
</td></tr>
<tr><td>23</td><td>RW</td><td>0</td><td>Open Drain Alert# Select: This bit is set to ‘1’ by eSPI master to configure the Alert# pin as an open-drain output.

By default, Alert# pin operates as a driven output. This bit must only be programmed to ‘1’ if open-drain Alert# pin is supported by the slave.

The bit must be valid when Alert Mode bit is a ‘1’ indicating Alert# pin is used for signaling the Alert event.

0b: Alert# pin is a driven output.

1b: Alert# pin is an open-drain output.</td></tr>
</table>

Bit	Type	Default	Description
22:20	RW	000b	**Operating Frequency:** This field identifies the frequency of operation. **Bits** **Frequency** 000 20 MHz 001 25 MHz 010 33 MHz 011 50 MHz 100 66 MHz Others Reserved. Note: This field has a default value of "000" to reflect $t_{INIT-FREQ}$ (Table 21) of 20MHz max.
19	RO	HwInit	**Open Drain Alert# Supported:** This bit indicates the support of the Alert# pin as an open-drain output by the slave. 0b: Open-drain Alert# pin is not supported. 1b: Open-drain Alert# pin is supported.
18:16	RO	HwInit	**Maximum Frequency Supported:** This field identifies the maximum frequency of operation supported by the slave. **Bits** **Frequency** 000b 20 MHz 001b 25 MHz 010b 33 MHz 011b 50 MHz 100b 66 MHz Others Reserved. The slave that indicates support for the maximum frequency of operation through this field will also support all the lower frequencies on the list. Note: Support for $t_{INIT-FREQ}$ (Table 21) is mandatory.
15:12	RW	0	**Maximum WAIT STATE Allowed:** eSPI master sets the maximum WAIT STATE allowed to be responded by slave before the slave must respond with an ACCEPT, DEFER, NON-FATAL ERROR or FATAL ERROR response code. This is a 1-based field in the granularity of byte time. When "0", it indicates a value of 16 byte time. A byte time corresponds to 8 serial clocks in the Single I/O mode, 4 serial clocks in the Dual I/O mode or 2 serial clocks in the Quad I/O mode.
11:8	RO	0	**Reserved.**

<table>
<tr><th>Bit</th><th>Type</th><th>Default</th><th>Description</th></tr>
<tr><td>7:0</td><td>RO</td><td>HwInit</td><td>Channel Supported: Each of the bits when set indicates that the corresponding channel is supported by the slave.
<table>
<tr><th>Bits</th><th>Channel</th></tr>
<tr><td>0</td><td>Peripheral Channel</td></tr>
<tr><td>1</td><td>Virtual Wire Channel</td></tr>
<tr><td>2</td><td>OOB Message Channel</td></tr>
<tr><td>3</td><td>Flash Access Channel</td></tr>
<tr><td>4:7</td><td>Reserved for platform specific channels</td></tr>
</table></td></tr>
</table>

8.1 Electrical Specification

NOTE: The electrical specification defined in this section is preliminary and it is subjected to change.

Table 2: Electrical Specification

Symbol	Parameter	Condition	Min	Typ	Max	Unit
Vcc	eSPI I/O voltage		1.71	1.8	1.89	V
R_{ON}	Output driver impedance	V_{out} = Vcc/2	15	25	35	Ohm
V_{IL}	Input low voltage				0.3*Vcc	V
V_{IH}	Input high voltage		0.7*Vcc			V
V_{HYS}	Input hysteresis voltage		0.1*Vcc			V
$R^{1}_{reset\text{-}PU}$	Weak pull-up impedance	V_{out} = 0.7*Vcc	10k		30k	Ohm
$R^{1}_{reset\text{-}PD}$	Weak pull-down impedance	V_{out} = 0.3*Vcc	10k		30k	Ohm
$R_{alert\text{-}PU}$	Weak pull-up impedance for Alert# pin	V_{out} = 0.7*Vcc		4.7k[3]		Ohm
C_{in}	Input capacitance				5	pF
C_L[2]	Load capacitance			10		pF
I_{IL}	Input leakage current	0 < V_{in} < Vcc			±10	uA

NOTES:

1. Weak pull-up on eSPI data~~,~~ and Chip Select# ~~and Alert#~~ pins (except Alert# pin) and weak pull-down on eSPI clock must be implemented as an integral part of the eSPI master buffer or on the board.

2. C_L is the test load defined for AC timing measurement.

3. The weak pull-up impedance value is defined for a typical eSPI bus loading when Alert# pin is configured as open-drain. Platform is required to adjust this value accordingly such that when Alert# pin is asserted, the assertion of the CS# for the shortest possible transaction (which causes the slave to tri-state the Alert# pin), is able to pull the Alert# pin high fast enough to the deasserted value before or by the last failing edge of the serial clock at the end of the transaction.

8.2 Timing Parameters

All timing parameters for the Enhanced Serial Peripheral Interface (eSPI) are specified from a device (slave) perspective. The host is required to account for channel effects in meeting the specified timings with the device.

NOTE: The timing parameters defined in this section are preliminary and they are subjected to change.

Table 3: AC Timing Specification

Symbol	Parameter Description
t_{CKH}	Clock High Time
t_{CKL}	Clock Low Time
t_{SLCH}	Chip Select# Setup Time
t_{CLSH}	Chip Select# Hold Time
t_{SHSL}	Chip Select# Deassertion Time
t_{DVCH}	Data In Setup Time
t_{CHDX}	Data In Hold Time
t_{CLQZ}	Output Disable Time during Turn-Around
t_{CLQV}	Output Data Valid Time
t_{CLQX}	Output Data Hold Time
t_{SHQZ}	Output Disable Time after Chip Select# Deassertion
t_{SLAZ}	Chip Select# Assertion to I/O[1] ~~or ALERT#~~ Tri-stated
t_{SHAA}	Chip Select# Deassertion to I/O[1] ~~or ALERT#~~ Assertion
t_{INIT}	eSPI Reset# Deassertion to First Transaction (GET_CONFIGURATION)
$t_{INIT-FREQ}$	Initial Bus Frequency upon eSPI Reset# Deassertion

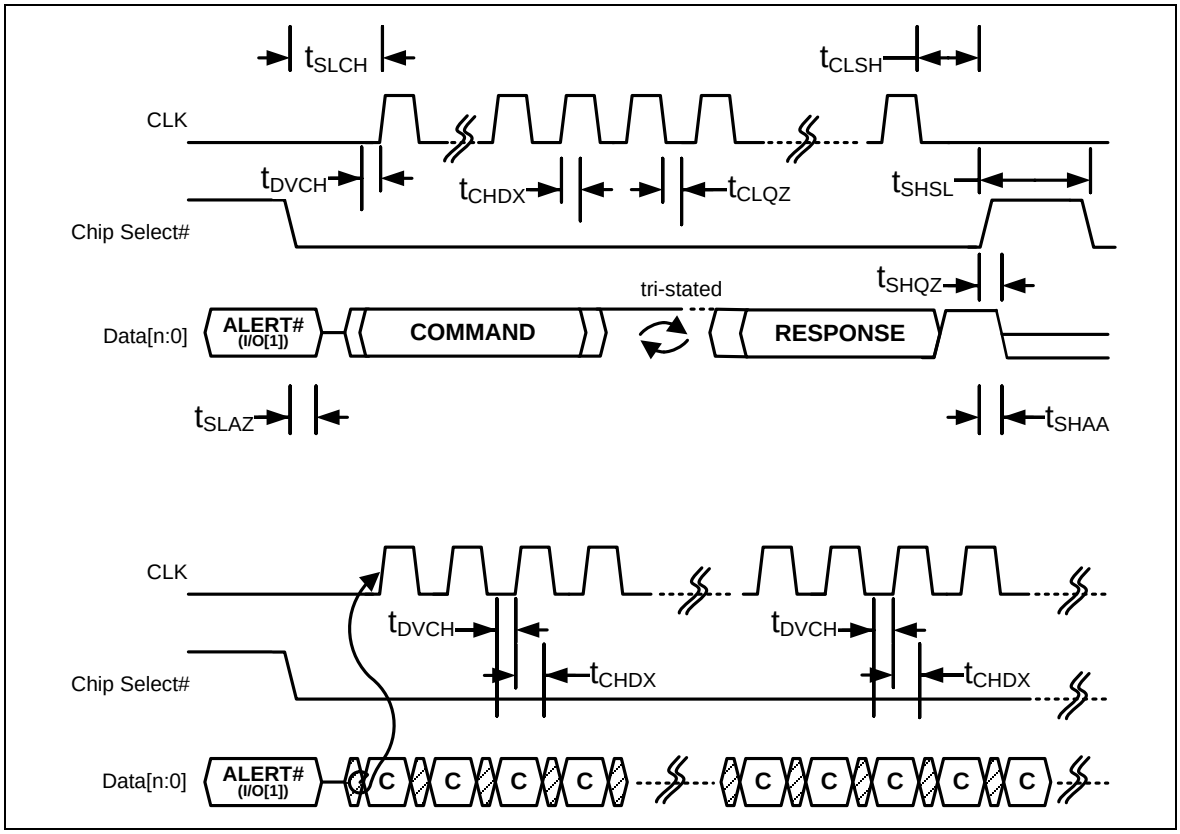
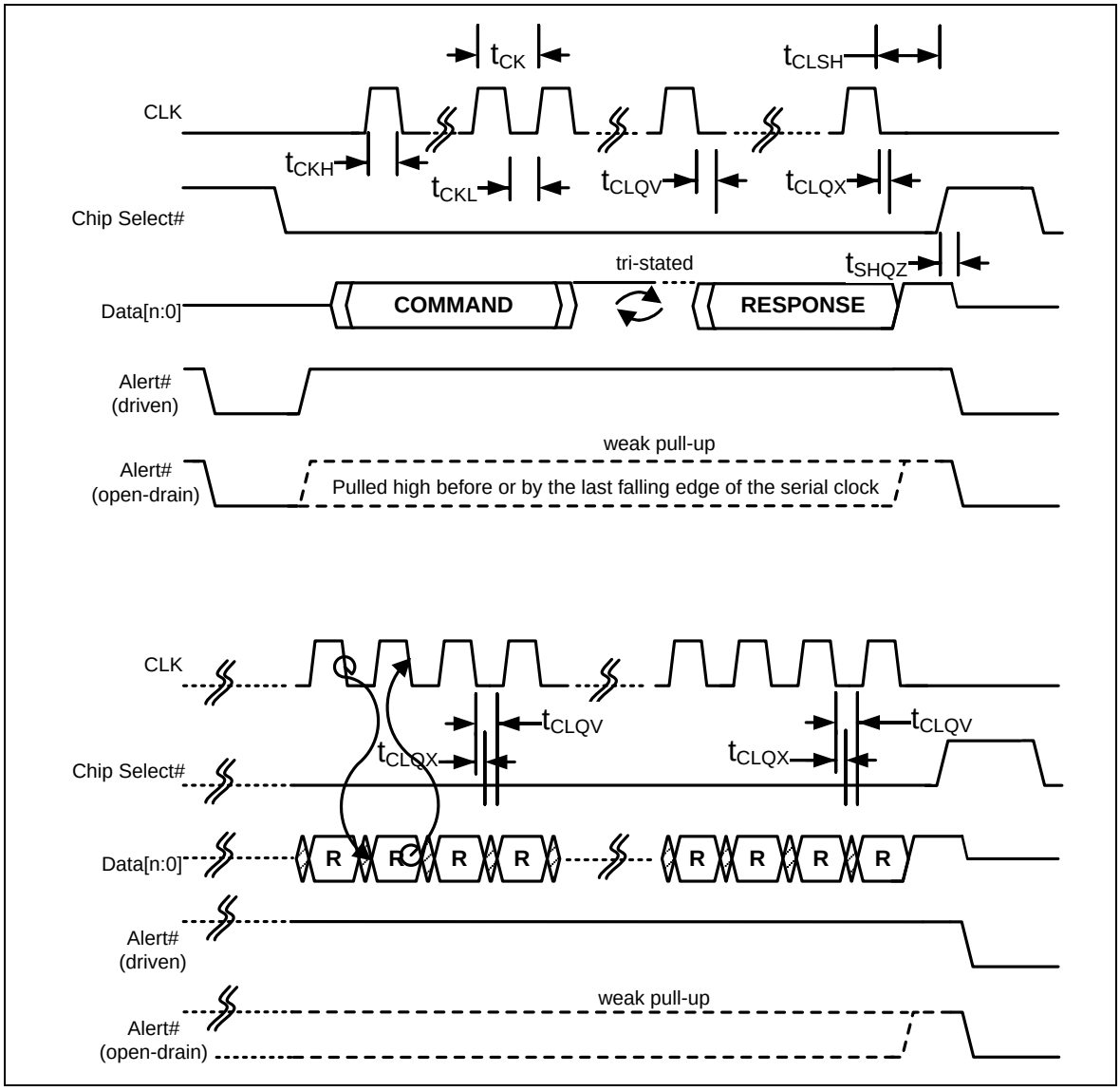


Figure 4: Input Timing Diagram

Figure 5: Output Timing Diagram



§ §