Databook, V 1.0, Nov 2005

TPM

SLB 9635 TT 1.2

Trusted Platform Module

CONFIDENTIAL

Automotive, Industrial & Multimarket

Chipcard and Security ICs

Never stop thinking

SLB 9635 TT 1.2
CONFIDENTIAL
All characteristics described in this document might change without further notice.

Revision History: **2005-11-15** V 1.0

Previous Version: none

Page	Subjects (major changes since previous version)

Edition 2005-11-15
Published by Infineon Technologies AG,
St.-Martin-Strasse 53,
D-81669 München, Germany

Attention please!

The information herein is given to describe certain components and shall not be considered as a guarantee of characteristics.

Terms of delivery and rights to technical change reserved.

We hereby disclaim any and all warranties, including but not limited to warranties of non-infringement, regarding circuits, descriptions and charts stated herein.

Infineon Technologies is an approved CECC manufacturer.

Information

For further information on technology, delivery terms and conditions and prices please contact your nearest Infineon Technologies Office in Germany or our Infineon Technologies Representatives worldwide (see address list).

Warnings

Due to technical requirements components may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies Office.

Infineon Technologies Components may only be used in life-support devices or systems with the express written approval of Infineon Technologies, if a failure of such components can reasonably be expected to cause the failure of that life-support device or system, or to affect the safety or effectiveness of that device or system. Life support devices or systems are intended to be implanted in the human body, or to support and/or maintain and sustain and/or protect human life. If they fail, it is reasonable to assume that the health of the user or other persons may be endangered.

CONFIDENTIAL Distribution under NDA only

1 Overview

The Infineon SLB 9635 TT 1.2 Trusted Platform Module (TPM) is an integrated circuit and software platform that provides computer manufacturers with the core components of a subsystem used to assure authenticity, integrity and confidentiality in e-commerce transactions and Internet communications. The SLB 9635 TT 1.2 is a complete solution implementing version 1.2 of the Trusted Computing Group[1] specifications (TCG), which is an industry group founded in 2003 by AMD, HP, IBM, Intel, Microsoft and now including more than 50 companies. For details about the TCG specification please refer to www.trustedcomputinggroup.org.

The basic principle to realize these issues is to insert a trusted subsystem - called the "root of trust" - into the PC platform, which is able to extend its trust to other parts of the whole platform by building a "chain of trust", where each link extends its trust to the next one. As a result, the TPM extends its trustworthiness, providing a Trusted PC for secure transactions.

To simplify system integration into existing PC mainboards, the SLB 9635 TT 1.2 uses the LPC interface (Low Pin Count) as defined by Intel (see [1]). This standardized interface is available on nearly every board and provides sufficient bandwidth. The TPM is basically a secure controller with added cryptographic functionality:

- Hash algorithm (SHA-1)
- Asymmetric key procedures (RSA, key lengths of up to 2048 bit)
- Secure key storage
- True random number generator
- Unique key that identifies each TPM

With these capabilities, the TPM is able to calculate hash-values of the BIOS at boot time as an integrity metric. Once this metric is available, it is saved in a secure memory location. Optionally, it could be compared to some predefined values and the boot process could be aborted on mismatch.

During the boot process, other integrity metrics are collected from the platform, e.g. of the boot loader and the operating system itself. Device drivers may be hashed, even hardware like PCI cards can be detected and identified. Every metric obtained is concatenated to the already available metrics. This gives a final metric, which describes the operational state of the whole platform and the state of its system integrity.

A challenger may now ask the platform for these metrics and make informed decisions on whether to trust it based on the metric values obtained. To support the privacy issue, the user of the platform may restrict the TPM in answering to any challenge, but the user is never able to make the TPM report false metrics. Moreover, the user is able to create several identities for his interactions.

Offering these features to a system, the TPM can be used in a wide field of applications, e.g. in a remote access network to authenticate platforms to a server and vice versa. Concerning e-commerce transactions, contracts can be signed with digital signatures using the TPM´s asymmetric encryption functionality. Regarding a network scenario, the client PCs equipped with a TPM are able to report their platform status to the server so that the network

[1] The TCG ist the successor organization of the Trusted Computing Platform Alliance (TCPA) which was founded in 1999 by COMPAQ, HP, IBM, Intel, and Microsoft. The TCPA stopped its operations and the TCG has adopted all specifications of the TCPA.

administration is aware of their trustworthiness. In conclusion, the TPM acting as a service provider to a system helps to make transactions more secure and trustworthy.

1.1 System Integration

Integration of the TPM into the system is simplified through the use of the LPC interface. See below for a general block diagram of a PC with a TPM. For a more detailed description, refer to **chapter 1.4**.

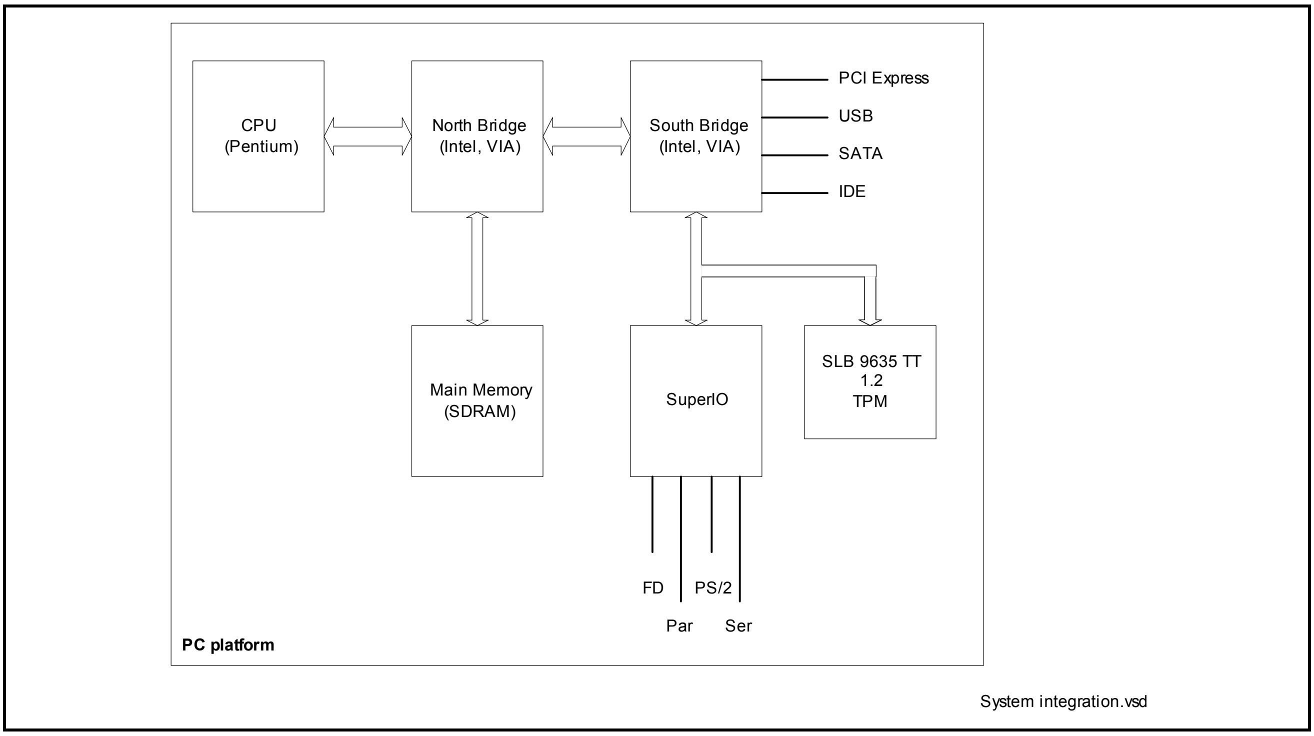


Figure 1-1 Integration of Infineon TPM into Standard PC Mainboard

CONFIDENTIAL Distribution under NDA only

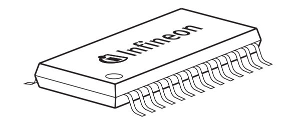
Trusted Platform Module
TPM

SLB 9635 TT 1.2

CMOS

1.2 Features

- TCG-compliant Trusted Platform Module
- Security architecture based on Infineon security controller family
- ROM for TCG firmware
- EEPROM for TCG firmware and data
- Hardware hash accelerator for SHA-1 algorithm
- Advanced Crypto Engine (ACE) for asymmetric key operations (up to 2048-bit key length)
- Power saving sleep mode
- 3.3V power supply
- LPC interface
  - Different addresses possible by pin-strapping
  - Support of Intel LT architecture LPC extensions
- Tick counter
- Security features
  - Over-/Undervoltage detection
  - Low/High frequency sensor
  - Reset filter
  - Memory encryption
  - Shield

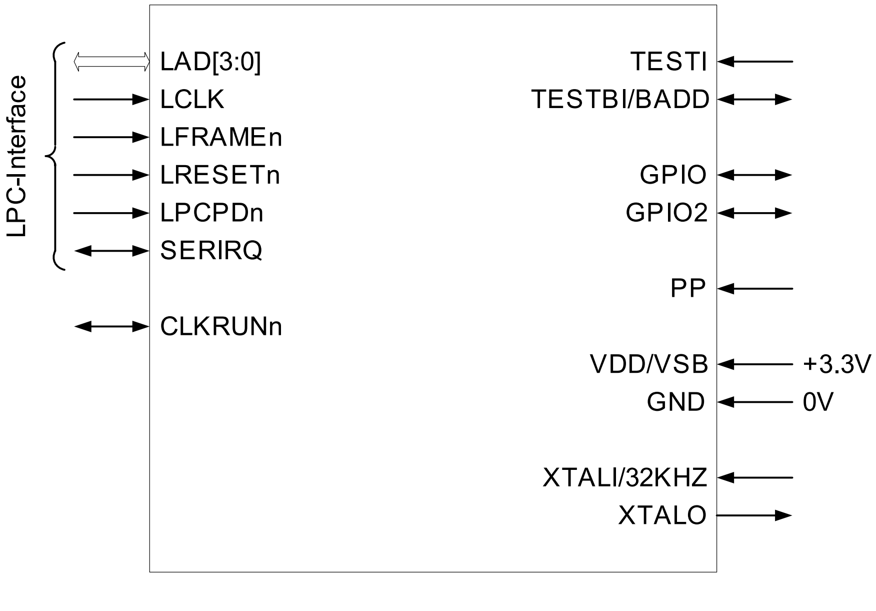


P-TSSOP-28-1,-2

Type	Package
SLB 9635 TT 1.2	P-TSSOP-28-1/2

1.3 Logic Symbol

The logic symbol gives an overview of the TPM interfaces.



TPM Logic Symbol.vsd

Figure 1-2 Logic Symbol of the TPM

1.4 Typical Schematic

The following figure shows the typical schematic for the TPM. The LPC interface pins connect directly to the LPC bus signals on the mainboard (or daughter-card). **LCLK** and **LRESETn** are usually the **PCICLK** and **PCIRST#** signals of the PCI bus (refer to the mainboard chipset description). The **TESTBI/BADD** pin is strapped to logical '1' giving a base address for the TPM of $4E_H/4F_H$ since addresses $2E_H/2F_H$ are usually used by the SuperIO chip.

The **PP** pin is connected to a jumper which allows the pin to be at either low or high level. This is needed since there has to be a way to signal physical access to the TPM. The default position for this jumpers is a connection to GND. If the user needs to show that he can physically access the platform, the jumper is changed to connect the pin to 3.3V. The TPM firmware reads this pin and reacts accordingly. If this feature is not used, the pin can be left open (it has an internal pull-down).

If the **CLKRUN#** signal is not available on the platform, the **CLKRUNn** pin must be connected to GND.

The **XTALI** pin is driven by a single-ended 32.768 kHz clock which always must be running while **VSB** is available (otherwise, this would be seen as an attack). Pin **XTALO** must be left unconnected in this case. Alternatively, a crystal can be connected between **XTALI** and **XTALO**.

The **TESTI** pin must be connected to GND.

The four decoupling capacitors should be placed as short as possible to the respective **VDD** and **VSB** pins of the chip. If necessary, a block decoupling capacitor of approx. 1µF should be added.

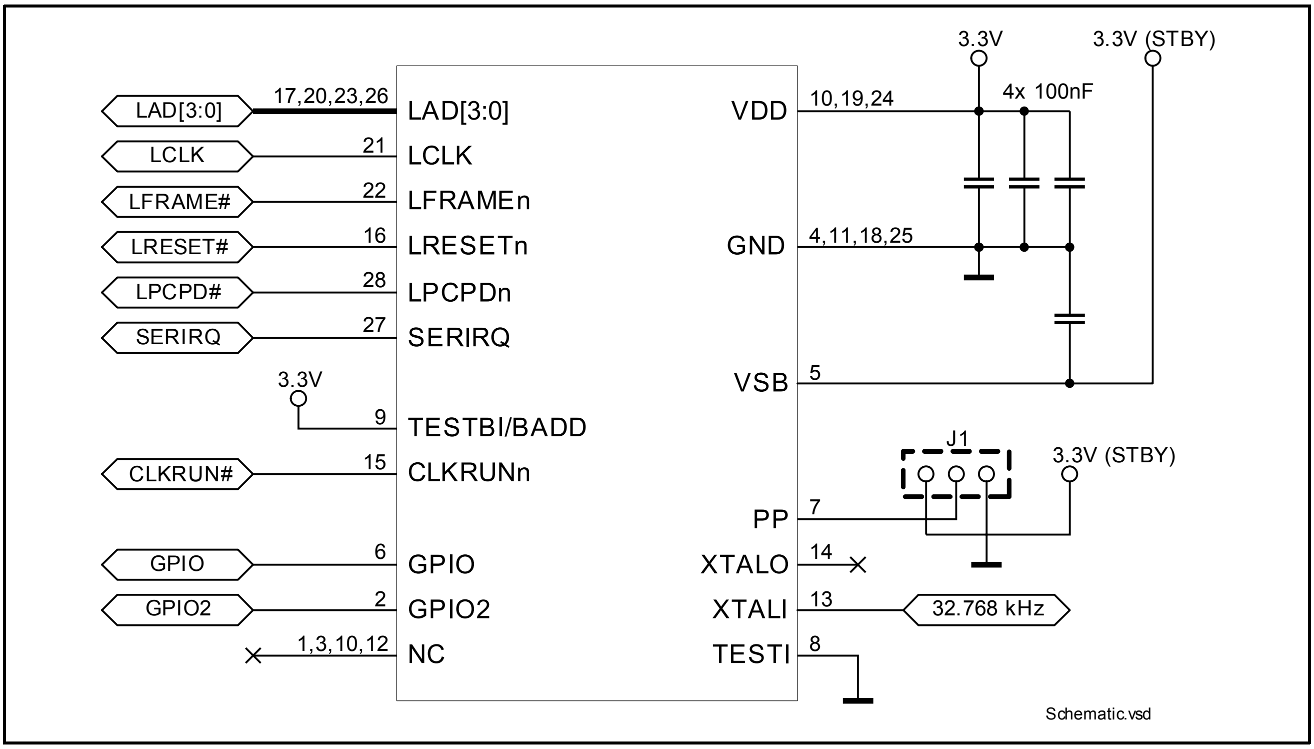


Figure 1-3 Typical Schematic

Note: Pin 10 is not connected for current device, but it reserved for VDD connection.

1.4.1 Crystal Connection

If a crystal is used for the tick counter frequency supply, the schematic shown in **figure 1-4** should be used. The crystal is a fundamental wave 32.768 kHz clock crystal. The parameters and the values of the capacitors are given in **table 1-1**.

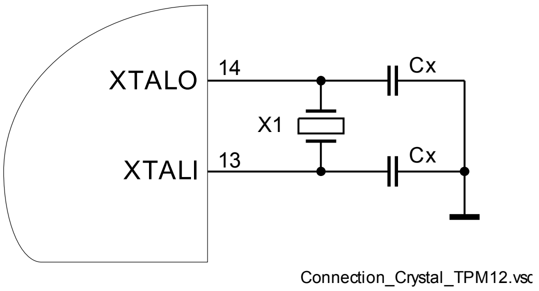
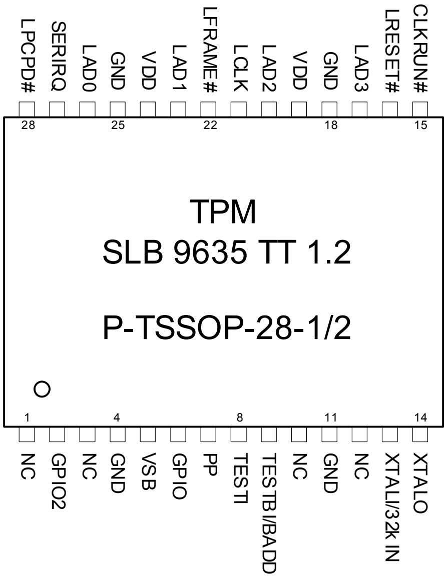


Figure 1-4 Connection of Crystal

Table 1-1 Crystal Ocillator Parameters

Parameter	Min	Nom	Max	Comment
f_{X1}		32.768 kHz		Crystal frequency
C_{shunt}		1.5 pF		crystal shunt capacitance
$C_{motional}$		2 fF		crystal motional capacitance
Cx		12.5 pF		external load capacitance
R_{X1}			50 kΩ	crystal series resistance
P_{X1}			1 µW	power dissipation in crystal

1.5 Pin Configuration



Pinning_TSSOP-28_TPM12.vsd

Figure 1-5 Pin Configuration of the TPM in P-TSSOP-28-1/2 Package

1.6 Pin Description

Table 1-2 Pin Description

Pin number	Pin Name	Description	Type[1]	Electrical Char.	Reset State
21	LCLK	LPC/PCI clock, nominal 33 MHz	I	PCI 3.3 V	-
22	LFRAMEn	LPC framing signal	I	PCI 3.3 V	-
17	LAD3	LPC command/address/data bus	Bi	PCI 3.3 V	Ts
20	LAD2				
23	LAD1				
26	LAD0				
16	LRESETn	LPC/PCI reset	I	PCI 3.3 V	-
28	LPCPDn	LPC power down	I	PCI 3.3 V	-
27	SERIRQ	Serial interrupt request	Bi	PCI 3.3 V	Ts
15	CLKRUNn	CLKRUN# signal	Bi	PCI 3.3 V	Ts
9	TESTBI/ BADD	Test port/Base address select	Bi	ISO	
8	TESTI	Test port	I	ISO	-
7	PP	Physical presence	I	ISO	-
6	GPIO	General Purpose I/O	Bi	ISO	
2	GPIO2	General Purpose I/O	Bi	ISO	
13	XTALI/ 32kIN	Input of internal crystal oscillator or input for single-ended external clock (32.768 kHz)	I	-	-
14	XTALO	Output of internal crystal oscillator.	O	-	-
19,24	VDD	3.3 V power supply	(supply)	-	-
5	VSB	3.3 V power supply (standby)	(supply)	-	-
4,11,18, 25	GND	Ground	(supply)	-	-
1,3,10,12	NC	Not connected internally	-	-	-

[1] I - input only, O - output only, Bi - bidirectional, Ts - tristate

1.7 LCLK

This pin provides the external clock for the chip. The clock frequency is 33 MHz nominal. This pin is typically connected to the PCI clock of the host.

Note: The PCI specification allows a clock range from DC to 33 MHz. However, the minimum frequency for the chip is 8 MHz since some low-frequency security sensors will trigger if the clock falls below this value. However, when the chip is in sleep mode, the clock may stop.

1.8 LFRAMEn

LPC framing signal. This pin is connected to the LPC LFRAME# signal and indicates the start of a new cycle on the LPC bus or the termination of a broken cycle. The signal is active low.

1.9 LRESETn

External reset signal. Asserting this pin unconditionally resets the LPC interface and - based on the state of the TPM - the controller core. The signal is active low and is typically connected to the PCIRST# signal of the host.

1.10 LAD[3:0]

Multiplexed LPC command, address and data bus. Connect these pins to the LAD[3:0] pins of the LPC host.

1.11 LPCPDn

If this pin is asserted, the chip will go into sleep mode. This signal is active low. Connect this pin to the LPCPD# signal of the LPC bus. It is mandatory to assert this signal before entering power state S3 of the platform (or stopping the LCLK for other reasons). If this is not done, the TPM will not enter sleep mode with low power consumption but will initiate an internal security reset.

The signals LAD[3:0], SERIRQ and CLKRUNn will be tristated after this pin is asserted.

1.12 CLKRUNn

This pin adds support for the CLKRUN# protocol used in mobile devices. Connect to the CLKRUN# signal of the mobile chipset. If this feature is not used, the pin must be connected to GND.

1.13 SERIRQ

Interrupt pin. Connect this pin to the SERIRQ signal of the LPC bus. The interrupt number used is programmed by software.

1.14 TESTBI/BADD

Base address select pin. The chip uses two addresses in the I/O space (similar to a SuperIO). If this pin is connected to GND, addresses $2E_H/2F_H$ are used. If it is strapped to VSB, addresses $4E_H/4F_H$ are used. Since normally $2E_H/2F_H$ are used by the SuperIO, it is recommended to connect this pin to VSB.

1.15 GPIO, GPIO2

These pins are general purpose I/O pins. Pin GPIO is defined as GPIO-Express-00, please refer to [5] and the PCI-SIG ECN “Trusted Configuration Space for PCI Express”.

The behaviour of GPIO2 is like GPIO, please refer to section 10.4.2.

1.16 Test Port (TESTI, TESTBI/BADD)

These pins are used for IFX test purposes. For normal operation, connect TESTI to GND.

1.17 PP Pin

This pin should be connected to a jumper. The standard position of the jumper should connect the pin to GND. If the pin is connected to VSB, some special commands are enabled (for instance, the command TPM_ForceClear, also refer to [6]).

If the hardware physical presence feature is not used on the platform, the pin can be left unconnected (it has an internal pull-down).

[6] includes commands which require physical presence at the platform. Physical presence can be proven either by the use of a software command during the POST or by the use of a "Physical Presence Switch" connected to PP. Before any use of hardware physical presence, usage of this pin must be explicitely enabled with the command TSC_PhysicalPresence (TPM_PHYSICAL_PRESENCE_HW_ENABLE). This should be done by the platform manufacturer before shipment of the system. Please see [6] for further details.

1.18 Power Supply Pins

All VDD pins should be connected together and routed to a 3.3 V power supply. The VSB pin should be connected to a 3.3V power supply which is active in standby mode. Likewise, all GND pins should be connected and routed to GND. Bypass capacitors should be placed as close as possible to the chip (100 nF). If necessary, a block decoupling capacitor of 1 µF or higher should be placed near the chip.

Table 1-3 TPM Power Supply

Platform power state	TPM VDD	TPM VSB	TPM LCLK	TPM LPCPDn
S0, S1, S2	active	active	active	deasserted
S3	inactive	active	inactive	asserted before entering S3
S4, S5	inactive	active	inactive	asserted

In table 1-3, an overview is given on which supply rails should be active when the platform is in different power states.

2 LPC Interface

The LPC interface consists of signals which are electrically compliant to the 3.3 V PCI bus. The TPM module supports only a subset of the cycles defined in the LPC specification. Only simple IO cycles are implemented.

2.1 LPC Signals

The TPM device uses the following LPC signals: LCLK, LFRAME#, LRESET#, LAD[3:0], LPCPD#, and SERIRQ. Additionally, CLKRUN# is supported. The LPC clock is assumed to be 33 MHz.

2.2 Standard IO cycles

The standard IO cycles use 16-bit addressing. For a detailed description, refer to the LPC specification [1].

2.3 Byte Ordering

The TCG specification [5] defines all multi-byte fields as big endian. To meet the big-endian requirement for the these fields, the TPM receives the MSB (most significant byte) first on the LPC bus.

2.4 Special TPM LT IO cycles

The new TPM-Read and TPM-Write special cycle are similar to the existing LPC IO-Read and IO-Write cycle formats. The only difference is a changed START field. Otherwise, they are equivalent to standard cycles. For a detailed description, refer to the LPC specification [1] and the TPM Interface Specification [5].

As for standard cycles, word or double word accesses are broken into single byte accesses to consecutive addresses by the chipset. For example, a 32-bit read from address 1000_H will be translated into four consecutive 8-bit reads from addresses 1000_H, 1001_H, 1002_H and 1003_H.

2.5 SYNC Field Usage

The SYNC field can be used to indicate access errors to the chipset (for instance, reading data from an empty FIFO). For normal accesses without errors, the SYNC field is driven to SYNC OK by the state machine of the TPM module.

A SYNC ERROR is indicated if a read to an empty FIFO or a write to a full FIFO of the LPC module occurs. This applies only to the legacy port of the LPC interface, all LT ports do not produce SYNC ERRORs at all. Instead, cycles are either accepted, wait stated or master aborted. This feature must be explicitely enabled by setting the EREN bit (see chapter 3.1.5). This bit is reset after a module reset; that means that an access to an empty FIFO will be acknowledged with a SYNC OK by default.

This behaviour has been chosen since on some mainboards, the generation of an NMI is enabled when a SYNC ERROR field occurs. Since an NMI is normally handled by the panic function of the OS, this leads to a blue screen (for Windows 2000, XP) if a SYNC ERROR occurs. However, depending on the driver implementation, the occurrence of a SYNC ERROR due to a read from an empty FIFO does not always have to be critical.

The SYNC field can also be used to enlarge cycles (i.e. to insert wait states on the bus). This feature is used to prevent data loss on the LT interface from and to the FIFO. The implemented wait mechanism always signals "Long Wait" values because the time needed to free or fill the FIFO is not predictable.

*Note: This is described in **chapter 5.5**, Legacy IO Cycles will stay unchanged, that means no wait generation on the legacy ports.*

2.6 LFRAMEn Usage

The signal LFRAMEn is used by the host to indicate the start of cycles and the termination of cycles due to an abort or time-out condition. This signal is to be used by the device to know when to monitor the bus for a cycle.

This signal is used as a general notification that the LAD[3:0] lines contain information relative to the start or stop of a cycle, and that the device monitors the bus to determine whether the cycle is intended for it. The use of LFRAMEn allows the device to enter a low-power state internally. When no activity is present on the bus, the device can decouple its statemachine from the bus and internally gate its clock.

When the device samples LFRAMEn active, it must immediately stop driving the LAD[3:0] lines on the next clock and monitor the bus for new cycle information.

The LFRAMEn signal functions as described in **[1]**.

2.7 Usage of LPCPDn Signal

The LPCPDn signal is asserted before power on the LPC bus is turned off. Turning off power also means that the clock disappears. To enable the devices to cope with this, the LPCPDn signal is asserted at least 30 µs before the power down.

The LCLK signal always stops in a low state. Upon receiving the LPCPDn signal, the device drives the LAD[3:0] lines to tristate (to prevent driving high into a powered-down host).

After LPCPDn goes back inactive, the LPC interface must always be reset using the LRESETn signal. The LPCPDn signal may occur asynchronously to LCLK. LCLK must be running at least 30 µs after LPCPDn is asserted and at least 30 µs again before LPCPDn is deasserted.

Refer also to **[1]**, section 8.2 for further information about the LPCPDn protocol.

Note: The mentioned minimum timing for LPCPD# to clock off of 30 µs might not be sufficient for the TPM. If the device is busy doing internal calculations and the clock is stopped before these calculations can be aborted, a security reset might occur inside the TPM. This does not really impair functionality since the device is reset anyway; however, the current drawn in that state might be up to 5 mA instead of the expected quiescent current of 500 µA. It is recommended that the time between the assertion of LPDPC# and the stopping of the clock should be at least 200 µs. If it can be assumed that the system is powered down while key generations are active, this time should be at least 3.5 ms; this allows programming of newly generated keys into the NVM of the device before power is lost. However, this behaviour is application-profile *dependent and is expected to happen with a very low probability only. For a timing diagram, please refer to* ***section 13.2.1*** *as well.*

2.8 SERIRQ Protocol

The LPC interface module supports the SERIRQ protocol as defined in **[3]**. Please refer to **chapter 7** for a detailed description of the use of that signal.

This page has been left blank intentionally.

3 Legacy Port

This chapter describes the registers used for the legacy port. For a description of the LT port, please refer to the TCG specification.

3.1 LPC Configuration Registers

The central configuration register set of the LPC interface supports ACPI compliant PnP configuration. The configuration registers are structured as a subset of the Plug and Play Standard registers, defined in Appendix A of [2]. All system resources (i.e. I/O address space and IRQ line) assigned to the device are configured in, and managed by, the central configuration register set.

3.1.1 Index-Data Register Pair

Access to the device configuration registers is via an Index-Data register pair, using only two system I/O byte locations. The base address of this register pair is determined during reset, according to the state of the hardware strapping option on the TESTBI/BADD pin.

Table 3-1 Base Address Options

TESTBI/BADD	I/O Address		
	Config Register[1]	Index Register	Data Register
0	$2E_H$	Config Register	Index Register + 1
1	$4E_H$	Config Register	Index Register + 1

[1] Note that the location of the Config register can be changed using the registers located at offsets 26h and 27h.

The Index register is an 8-bit R/W register located at the selected base address (Base+0). It is used as a pointer to the configuration register file, and holds the index of the configuration register that is currently accessible via the Data register.

Reading the Index register returns the last value written to it (or the default of 00h after reset). The Data register is an 8-bit virtual register, used as a data path to any configuration register. Accessing the Data register actually accesses the configuration register that is currently pointed to by the Index register.

The location of the Index register is also used to enable or disable the configuration of the device. Initially (after reset), the Index/Data pair is inactive. A write access to the Config register with the value of 55h enables the Index/Data register pair, a subsequent write access with the data value of AAh disables the register pair.

The location of the Config register can be changed by enabling the Index/Data pair and then writing the new address into the configuration registers at locations 26h and 27h. After writing these registers, the chip should be brought out of the configuration mode by writing AAh to the (old) Config register. All subsequent accesses have to be done using the new Config (and thus also the new Index/Data) addresses. All semantics of the Config register are identical at the new location. Please refer to figure 3-1 for a logical view of this config mechanism.

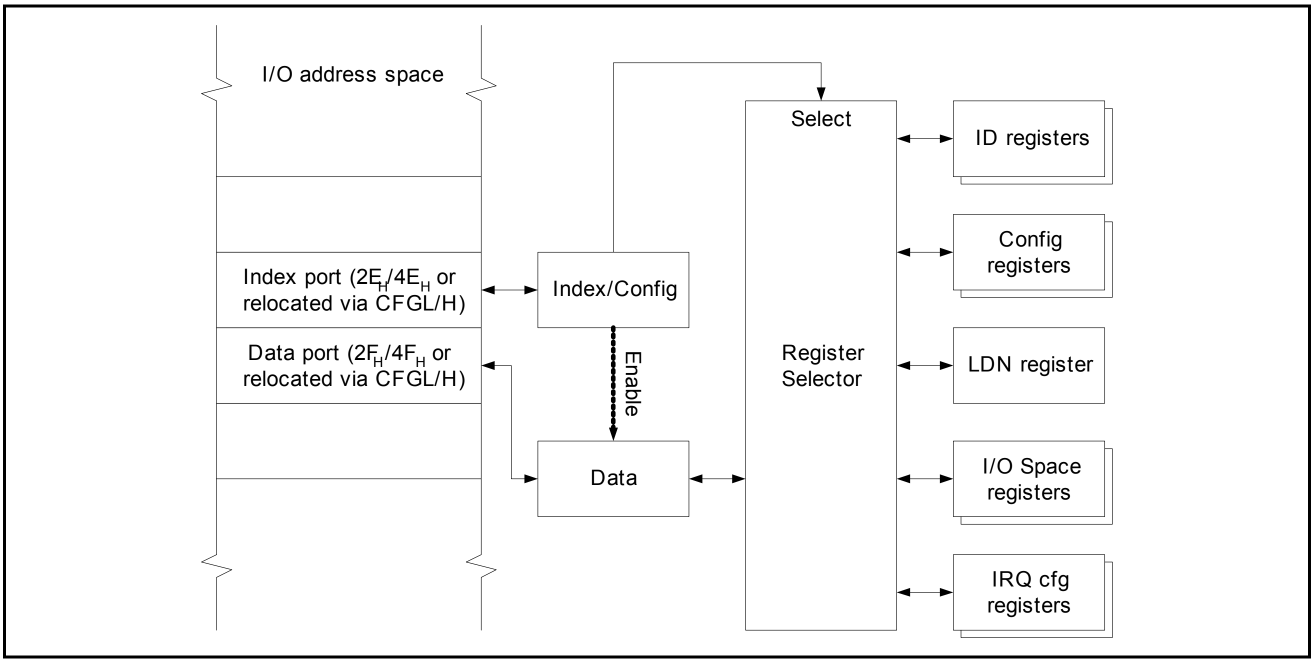


Figure 3-1 Logical View of Configuration Registers

3.1.2 Standard Configuration Register Definitions

Unless otherwise noted, all registers described here are read/write. All reserved bits return 0 on read and must not be modified unless noted otherwise. Read-modify-write accesses shall be used to prevent values of reserved bits being changed during write. Write-only registers shall not use read-modify-write during updates.

All bits marked as ’0’ are reserved as well, that means they return ’0’ on reads while write accesses to these bits are discarded.

3.1.2.1 Register Address Overview

Table 3-2 Register Address Overview

Register	Address	Reset	Function
RES	00_H-06_H	00_H	reserved
	08_H-$1F_H$		
	22_H-25_H		
	28_H-$2F_H$		
LDN	07_H	00_H	Logical Device Select
ID1	20_H	$0B_H$	Chip ID
ID2	21_H	00_H	
CFGL	26_H	xE_H[1]	Config Address
CFGH	27_H	00_H	
DAR	30_H	00_H	Device Activate
SEN	38_H	00_H	Sync Error Enable
IOLIMH	60_H	00_H	IO Base Address
IOLIML	61_H	00_H	
IRQSEL	70_H	00_H	IRQ Channel Select
IRQTYPE	71_H	02_H	IRQ Type Select
IDVENL	$F1_H$	$D1_H$	PCI Vendor ID
IDVENH	$F2_H$	15_H	
IDPDL	$F3_H$	$0B_H$	PCI Device ID
IDPDH	$F4_H$	00_H	
RID	$F5_H$	xx_H	Revision ID
WRFIFO	base + 0[2]	00_H	Write FIFO
RDFIFO	base + 1[2]	00_H	Read FIFO
STAT	base + 2[2]	05_H	Status Register
CMD	base + 3[2]	xx_H	Command Register

[1] upper nibble of reset value depends on pin TESTBI/BADD when LRESETN is deasserted.

[2] the base address depends on the setting of the IO base address.

3.1.2.2 Reserved Registers

RES
Reserved registers **(00_H-06_H)** **Reset Value: 00_H**

7	6	5	4	3	2	1	0
RES							

RES
Reserved registers **(08_H-$1F_H$)** **Reset Value: 00_H**

7	6	5	4	3	2	1	0
RES							

Field	Bits	Type	Description
RES	7:0	r	**Reserved** Return 0 on reads. Writes are discarded.

3.1.2.3 LDN Register

LDN
Logical Device Select **(07_H)** **Reset Value: 00_H**

7	6	5	4	3	2	1	0
LDN							

Field	Bits	Type	Description
LDN	7:0	r	**Logical Device Select** This register selects the current logical device. Since only one device is supported, the register is read-only and always returns 00_H.

3.1.2.4 Device Configuration Registers

ID1
Chip ID Register 1 **(20_H)** **Reset Value: $0B_H$**

7	6	5	4	3	2	1	0
ID1							

ID2
Chip ID Register 2 **(21_H)** **Reset Value: 00_H**

7	6	5	4	3	2	1	0
ID2							

Field	Bits	Type	Description
ID1	7:0	r	**Device identification, low byte** Same as PCI device ID
ID2	7:0	r	**Device identification, high byte** Same as PCI device ID

CFGL
Config Address, Low Byte **(26_H)** **Reset Value: xE_H**

7	6	5	4	3	2	1	0
CFGL							

CFGH
Config Address, High Byte **(27_H)** **Reset Value: 00_H**

7	6	5	4	3	2	1	0
CFGH							

Field	Bits	Type	Description
CFGL	7:0	rw	**Low byte of 16-bit configuration address** Reset value depending on TESTBI/BADD pin when reset is deasserted: 0 If TESTBI/BADD = 0, reset value is $2E_H$ 1 If TESTBI/BADD = 1, reset value is $4E_H$ Also refer to **section 1.14**.
CFGH	7:0	rw	**High byte of 16-bit configuration address**

RES
Reserved registers **(22_H-25_H)** **Reset Value: 00_H**

7	6	5	4	3	2	1	0
RES							

RES
Reserved registers **(28_H-$2F_H$)** **Reset Value: 00_H**

7	6	5	4	3	2	1	0
RES							

Field	Bits	Type	Description
RES	7:0	r	**Reserved** Return 0 on reads. Writes are discarded.

3.1.2.5 Device Activate Register

DAR
Device Activate Register **(30_H)** **Reset Value: 00_H**

7	6	5	4	3	2	1	0
0	**0**	**0**	**0**	**0**	**0**	**0**	**ACT**

Field	Bits	Type	Description
ACT	0	rw	**Activate Device** Setting this bit activates the (logical) device. This is normally used by the LPC host to deactivate non-functional or not properly configured devices.

3.1.2.6 SYNC Error Enable Register

SEN
Sync Error Enable **(38_H)** **Reset Value: 00_H**

7	6	5	4	3	2	1	0
0	**0**	**0**	**0**	**0**	**0**	**0**	**EREN**

Field	Bits	Type	Description
EREN	0	rw	**Error Generation Enable** Setting this bit enables SYNC Error generation on the LPC bus. The conditions for throwing an LPC error is either reading from an empty FIFO or writing to a full FIFO. Note that this is only valid for cycles on the legacy port.

3.1.2.7 IO Space Config Registers

IOLIMH
IO Limit High Byte **(60_H)** **Reset Value: 00_H**

7	6	5	4	3	2	1	0
IOLIMH							

IOLIML
IO Limit Low Byte **(61_H)** **Reset Value: 00_H**

7	6	5	4	3	2	1	0
IOLIML							

Field	Bits	Type	Description
IOLIMH	7:0	rw	**IO Limit High Byte**
IOLIML	7:0	rw	**IO Limit Low Byte** These two registers define the (16-bit) IO lower limit address for the IO descriptor 0. *Note: Only the upper 12 bits are decoded. That means the IO base address must always programmed to lie on a 16-byte boundary.*

3.1.2.8 Interrupt Configuration Registers

IRQSEL
IRQ Select Register **(70_H)** **Reset Value: 00_H**

7	6	5	4	3	2	1	0
0	0	0	0	IRQ			

IRQTYPE
IRQ Type Register **(71_H)** **Reset Value: 02_H**

7	6	5	4	3	2	1	0
ACK	0	0	0	0	0	LEV	TYPE

Field	Bits	Type	Description
IRQ	3:0	rw	**Interrupt Request** Selects the IRQ. A value of 1 selects IRQ1, a value of 2 select IRQ2 and so on. A value of 0 is not a valid interrupt selection. This selection is only valid for the legacy port. If the interrupt request number set here is either 0 or the same than set for the non-legacy port (in register INT.VECTOR in the LT space), no interrupts will be generated for the legacy port.
ACK	7	rw	**Interrupt Acknowledge** If zero (default), interrupts are acknowledged explicitely by setting the IRQC bit in the CMD register. If set to one, interrupts are acknowledged implicitly by reading the STAT register.
LEV	1	rw	**Interrupt Active Level** Selects the active level of the interrupt request. 0 means active low, 1 means active high.
TYPE	0	rw	**Interrupt Type** Selects the type of the interrupt. 0 means edge triggered, 1 means level triggered interrupts.

For level-triggered interrupts, an acknowledge during the ISR (Interrupt Service Routine) is mandatory. If this is not done, a new interrupt will be generated immediately after exiting the interrupt service routine.

The acknowledge can be signaled in two different ways, either explicitely by setting the IRQC bit in the CMD register or implicitly by reading the STAT register. For the non-legacy ports, the acknowledge is done by writing a ’1’ to one or more bits (corresponding to the type of interrupt just handled) of the TPM.INT.STATUS register.

For edge-triggered interrupts, an acknowledge is not mandatory. Though the interrupt signal stays active, a new interrupt is only generated when a new edge occurs.

3.1.2.9 Identification Registers

IDVENL
PCI Vendor ID, Low Byte **($F1_H$)** **Reset Value: $D1_H$**

7	6	5	4	3	2	1	0
1	**1**	**0**	**1**	**0**	**0**	**0**	**1**

IDVENH
PCI Vendor ID, High Byte **($F2_H$)** **Reset Value: 15_H**

7	6	5	4	3	2	1	0
0	**0**	**0**	**1**	**0**	**1**	**0**	**1**

IDPDL
PCI Device ID, Low Byte **($F3_H$)** **Reset Value: $0B_H$**

7	6	5	4	3	2	1	0
0	**0**	**0**	**0**	**1**	**0**	**1**	**1**

IDPDH
PCI Device ID, High Byte **($F4_H$)** **Reset Value: 00_H**

7	6	5	4	3	2	1	0
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**

Field	Bits	Type	Description
IDVENL	7:0	r	**PCI Vendor ID** IDVENL and IDVENH indicate the value of $15D1_H$ which is the PCI vendor ID of Infineon AG.
IDVENH	7:0	r	
IDPDL	7:0	r	**PCI Device ID** IDPDL and IDPDH indicate a product ID value of $000B_H$ which identifies the LT TPM (SLB 9635 TT 1.2).
IDPDH	7:0	r	

3.1.2.10 Revision ID Register

RID
Revision ID **($F5_H$)** **Reset Value: xx_H**

7	6	5	4	3	2	1	0
RID							

Field	Bits	Type	Description
RID	7:0	r	**Revision ID** Indicates the revision ID of the firmware to the host.

3.1.3 Data Registers

WRFIFO
Write FIFO **(Base Address + 0_H)** **Reset Value: 00_H**

7	6	5	4	3	2	1	0
WFIFO							

RDFIFO
Read FIFO **(Base Address + 1_H)** **Reset Value: 00_H**

7	6	5	4	3	2	1	0
RFIFO							

Field	Bits	Type	Description
WFIFO	7:0	w	**Write FIFO**
RFIFO	7:0	r	**Read FIFO**

Both FIFOs have a depth of 8 bytes each. The addresses of these registers are relative to the IO base address set in the IOLIMH and IOLIML registers. Each byte written to the WRFIFO register is put into the FIFO of the device. A read from address RDFIFO returns one byte from the device FIFO. If the EREN bit is set and no byte is available for reading or no space is available for writing, the transfer is aborted with a SYNC error by the LPC statemachine (using the LPC SYNC mechanism, see [1]). If the EREN bit is not set, read and write accesses are aborted without an error indication.

In both cases, a read access returns the last byte again while a write access is simply discarded.

3.1.4 Status Registers

STAT
Status Register **(Base Address + 2_H)** **Reset Value: 05_H**

7	6	5	4	3	2	1	0
RDA	IRQA	RES	RFUI		FOK	LPA	XFE

Field	Bits	Type	Description
RDA	7	r	**Receive Data Available** This bit is set when data is available in the FIFO and may be read via the RDFIFO register.
IRQA	6	r	**Interrupt Request Active** This bit is set if an interrupt request is active.
RES	5	r	**Reserved** Reads as 0, writes have no effect.

Field	Bits	Type	Description
RFUI	4:3		**Reserved for future use, implemented** Bits 3 has RW permissions from the host and is RO from the ECO core. Bit 4 is vice versa.
FOK	2	r	**Firmware OK** This bit is set to '1' after reset and is '0' after the firmware of the TPM is ok.
LPA	1	r	**Loop Active** Indicates an active loop on the host side of the LPC interface. All bytes written to the interface are mirrored back to the host immediately.
XFE	0	r	**Transmit FIFO Not Full** This bit is set when data may be written into the FIFO using the WRFIFO register.

3.1.5 Command Registers

CMD
Command Register **(Base Address + 3_H)** **Reset Value: XX_H**

7	6	5	4	3	2	1	0
0	IRQC	0	0	RFUI	RFUI	LP	DIS

Field	Bits	Type	Description
IRQC	6	w	**Interrupt Request Clear** Writing a 1 to this bit clears the IRQA bit in the STAT register (see **chapter 3.1.4**). This bit is not sticky.
RFUI	3	rw	**Reserved for future use, implemented** Bit 3 can be written and read by the host and is read only from the ECO2 side.
RFUI	2	-	**Reserved for future use, implemented**
LP	1	rw	**Loop** Closes a loop on the LPC interface (host side). All bytes written from the host are mirrored back immediately. This bit can only be used in factory test environments to prevent Denial-Of-Service attacks.
DIS	0	rw	**Disable Interrupt** When this bit is set to 1, no interrupts are generated on the SERIRQ line. Setting this bit to 0 again will generate an interrupt if a request is pending.

The address of this register is relative to the IO base address set in the IOLIMH and IOLIML registers. After reset, the loop and interrupts are disabled. This is equivalent to writing a value of 01_H into the register.

The write-only bits return 0 on reads.

4 Locality and Access Functionality

The TPM interface features accesses through different logical ports, all of which are using the same physical interface signals. The distinction between these ports is done by the START field of the LPC bus cycle on the one hand and the address range on the other hand.

Six logical ports are defined, one legacy port and 5 so-called LT ports. These ports are called localities. The locality legacy (or locality -1) is accessed using the standard LPC IO cycles. All other localities must be accessed using the new LPC LT cycles (also refer to **chapter 2.2** and **chapter 2.4**).

Only one port is allowed to issue commands to the TPM through the LPC interface at a given time. There is only one FIFO structure (one transmit FIFO and one receive FIFO) for all logical ports. This implies that all internal FIFO status bits are the same for all communication ports. The scheduling of these ports is described in the following text.

4.1 LPC Access Rights

The TPM features 6 levels of locality, locality legacy and locality 0 to 4, where locality legacy has the lowest and locality 4 the highest priority. Only one locality is allowed to access the TPM at a time. An access is defined as the sequence of sending a request to the TPM and reading back the response after the operation has finished.

The different localities have different meanings, some TCG commands are only accepted by the TPM if they are generated using locality 4, other TCG commands may require a locality 1 to 3 and many operations do not require any locality, these would be accepted on the legacy port and on the port describing locality 0.

The locality legacy accesses the TPM through standard LPC I/O cycles, where localities 0..4 use the new LPC special cycles to identify the LT address space. The access to these localities is done through memory cycles which are mapped to the LT address space (by the platform chipset). The mapping is shown in **table 4-1**.

Table 4-1 Address Mapping for LT Space

System Address	LPC Address (using new LPC START cycle)	Locality
FED4_0xxx$_H$	0xxx$_H$	0
FED4_1xxx$_H$	1xxx$_H$	1
FED4_2xxx$_H$	2xxx$_H$	2
FED4_3xxx$_H$	3xxx$_H$	3
FED4_4xxx$_H$	4xxx$_H$	4

The access permission is done with a semaphoring mechanism using the command bits described in **section 5.2**. Accesses from the locality legacy do not use the semaphoring mechanism. They will only be granted if no ACTIVE.LOCALITY bit is set, otherwise they will be master-aborted. A master-abort means that the TPM does neither generate any SYNC signal on the LPC bus nor reacts internally on any legacy LPC activity.

For localities 0..4, the protocol is as follows: each software agent, when it wishes to use the TPM, will write a '1' to REQUEST.USE. If the TPM is idle, the first agent that writes its bit will become user. The TPM will set ACTIVE.LOCALITY for the locality that gains access to the TPM. All other localities that have written the REQUEST.USE bit will poll on the ACCESS register and read a '0' for ACTIVE.LOCALITY. The winning locality will read a '1' on this bit, and may start issuing commands to the TPM. The state diagram is shown in **figure 4-1**.

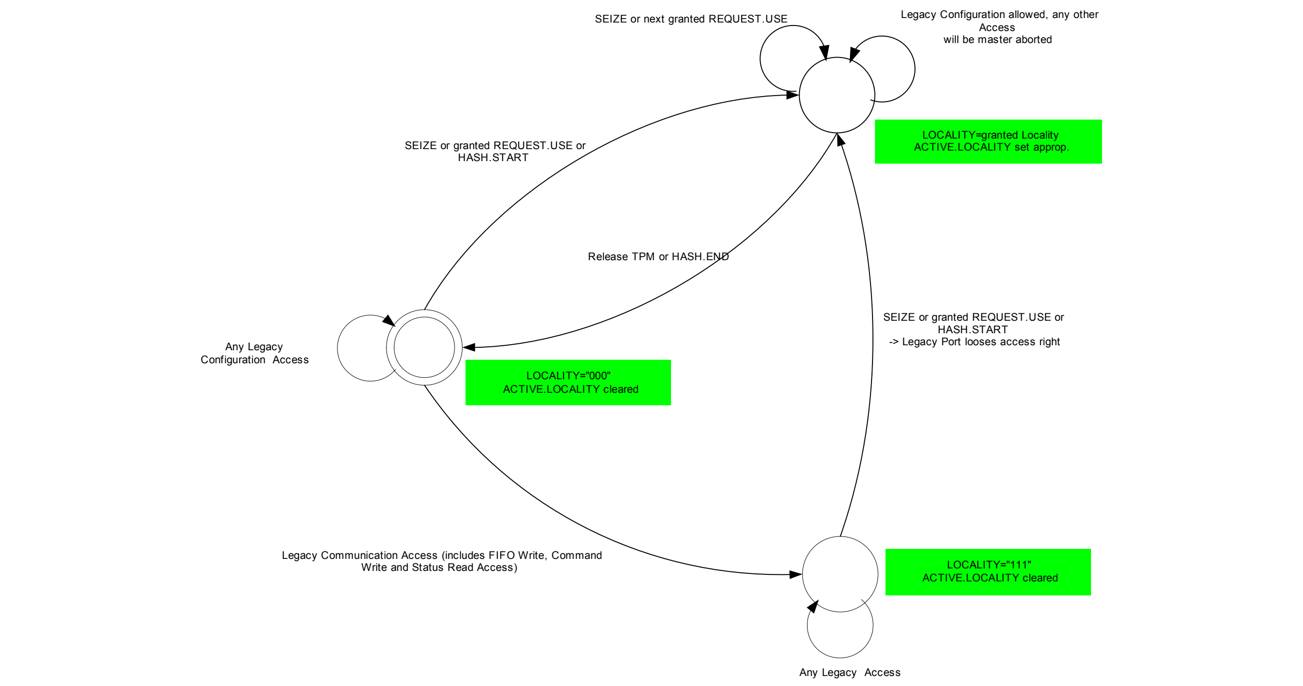

SEIZE or next granted REQUEST.USE
Legacy Configuration allowed, any other Access will be master aborted
LOCALITY=granted Locality
ACTIVE.LOCALITY set approp.
SEIZE or granted REQUEST.USE or HASH.START
Release TPM or HASH.END
SEIZE or granted REQUEST.USE or HASH.START
-> Legacy Port looses access right
Any Legacy Configuration Access
LOCALITY="000"
ACTIVE.LOCALITY cleared
Legacy Communication Access (includes FIFO Write, Command Write and Status Read Access)
LOCALITY="111"
ACTIVE.LOCALITY cleared
Any Legacy Access

Figure 4-1 Locality Access State Diagram

When the winning locality is finished with the operation, it will write a '1' to its ACTIVE.LOCALITY bit. This indicates that it has finished its series of commands. The TPM will look at all pending REQUEST.USE bits and grant the access to the highest locality with its bit set by setting ACTIVE.LOCALITY for that locality. The others continue waiting.

Software can read the ACCESS register and determine that the TPM is in use by some other agent, so it knows that the TPM is functioning, but busy.

If software, for some reason, decides that another locality's software is not playing fair or is hung, then it can seize the TPM from the present user as long as that user is at a lower locality. Software writes a '1' to SEIZE. This forces the TPM to stop honoring cycles from the other locality, and only honor the new locality's requests.

Each register has its own access rights which describe if the register is updated on a write or can be read if the associated ACTIVE.LOCALITY is set respectively not set. If the access cycle is not accepted by the TPM, it will be master aborted (no LPC SYNC cycle will be generated and no action is done on the internal registers). **Table 4-2** shows which operation is done by the TPM on each register depending on the ACTIVE.LOCALITY bit.

*Note: The locality legacy is divided into **configuration** and **communication** accesses. Legacy **configuration** is always allowed, even if an LT locality holds the TPM. Only legacy **communication** is suppressed (master aborted) if any LT locality has its ACTIVE.LOCALITY bit set.*

*Note: In **table 4-2**, “abort” means that no valid SYNC is generated when a cycle is seen by the interface which shall be aborted. The data present in an aborted write access cycle must not change the addressed register.*

Table 4-2 LT register - Access matrix

	ACTIVE.LOCALITY set for this locality		ACTIVE.LOCALITY set for other locality		ACTIVE.LOCALITY not set at all	
	READ	WRITE	READ	WRITE	READ	WRITE
STS	read	write	abort	abort	abort	abort
INT.ENABLE	read	write	read	abort	read	abort
INT.VECTOR	read	write	read	abort	read	abort
INT.STATUS	read	reset Intr.	read	abort	read	abort
INT.CAPABILITY	read	- (abort)	read	- (abort)	read	- (abort)
ACCESS	read	write	read	write	read	write
READFIFO	read[1]	abort	abort	abort	abort	abort
WRITEFIFO	abort	write	abort	abort	abort	abort
Configuration Registers	read	write	read	abort	read	abort
HASH.START	abort	write	abort	abort	abort	write[2]
HASH.DATA	abort	write	abort	abort	abort	abort
HASH.END	abort	write[3]	abort	abort	abort	abort

[1] If **STS.DATA.AVAIL** is not set, this access is ’abort’

[2] The write to HASH.START sets **ACCESS.ACTIVE.LOCALITY** of locality 4.

[3] The write to HASH.END is an implicit release of the TPM (like a ’1’ - write to the **ACCESS.ACTIVE.LOCALITY** bit of locality 4)

This page has been left blank intentionally.

5 LT Register Description

The new LPC cycles with changed START field open a new address space for the LT functionality of the TPM. This chapter describes the registers which are included in this space and implemented inside the TPM.

5.1 LT Register Space

Table 5-1 lists the addresses decoded by the TPM. Note that only the ACCESS register has multiple copies, one per locality. The other addresses alias to a single register, with the locality used to determine if accesses are permitted or aborted. The latter is done with a LPC master abort cycle, where the TPM does not react on an access at all.

The register descriptions are shown for one locality only. The addresses for the registers have their locality identifier at the highest significant nibble (e.g. TPM.INT.STATUS register for locality 3 has address 3010h). The differences for Locality 4 are shown in table 5-2.

Table 5-1 LT Address Space

Offset	Register Name	Description
x000h	TPM.ACCESS.x	Used to coordinate the ownership for a particular locality
x00Bh-0008h	TPM.INT.ENABLE.x	Global and specific interrupt enable bits and interrupt type definition
x00Ch	TPM.INT.VECTOR.x	SERIRQ vector (corresponds to interrupt number) to be used by the TPM
x013h-x010h	TPM.INT.STATUS.x	Indicates the cause(s) of an interrupt
x017h-x014h	TPM.INT.CAPABILITY.x	Shows which interrupts and what interrupt types are supported by the TPM
x01Ah-x018h	TPM.STATUS.x	LT Status Register. Provides status of the TPM LT interface.
x027h-x024h	DATAFIFO.x	Read and write FIFO address, depending on transaction. These 4 addresses are aliased to one inside the TPM. The TPM is not required to check that the addresses on LPC are incrementing modulo 4, even though platform hardware would most likely send it that way. The read or write data could be performed by accessing x024h over and over without using the other addresses.
xF03h-xF00h	DID/VID.x	Vendor and device ID
xF04h	RID.x	Revision ID
xF7Fh-xF05h		TCG defined configuration registers
xF80h[1)]	LEGACY1	Alias registers to legacy IO space
xF84h	LEGACY1B	Alias registers to legacy IO space (unused)

Table 5-1 LT Address Space (cont'd)

Offset	Register Name	Description
xF88h	LEGACY2	Alias registers to legacy IO space
xF8Ch	LEGACY2B	Alias registers to legacy IO space (unused)
xF90h	TO	8-bit timeout value for the currently executing command (in seconds). This value is set by FW to indicate a measure how long the current command might take. The register is cleared on read (by hardware). If the command still takes longer than the indicated time, the register is re-set by firmware.
xFFFh-xF91h		Vendor defined configuration registers
All addresses not defined in the table above		Reserved, reads return FFh, writes are dropped.

[1] Note that these addresses must only be used for locality 0, that means addresses 0F80h, 0F84h, 0F88h and 0F8Ch. Addresses xF80h, xF84h, xF88h and xF8Ch for x=1, 2, 3, 4 must not be used. For the interface implementation, these addresses are reserved.

Table 5-2 Locality 4 Special Registers

Offset	Register Name	Description
4020h	TPM.HASH.END	Signals the end of the hash operation. This command is carried out as a single write to 4020h. Writes to 4021h to 4023h are not decoded by the TPM. The actual value of the write operation is not relevant. HASH.END is an implicit release of the TPM. This is equivalent to writing a '1' to **ACCESS.ACTIVE.LOCALITY**. The HASH.END operation is only accepted if **ACCESS.ACTIVE.LOCALITY** is set for locality 4.

Table 5-2 Locality 4 Special Registers (cont'd)

4027h-4024h	TPM.HASH.DATA/ DATAFIFO.4	This port is used to send data that the TPM is to hash. Between HASH.START and HASH.END, writes to this register are treated as data and part of the HASHed value. Outside that window, writes to this register are treated as part of a TPM command. These four addresses are aliased to one inside the TPM. The TPM is not required to check that the addresses in the LPC cycles are incrementing modulo 4. The hash could be done by writing to 4024h repeatedly without using the other addresses.
4028h	TPM.HASH.START	Signals the start of the hash operation. This command must be done by a single write to address 4028h. Accesses to addresses 4029h to 402Bh are not decoded. The actual value of the write operation is not relevant. The HASH.START operation is only accepted either if **ACCESS.ACTIVE.LOCALITY** is set for locality 4 or no ACCESS.ACTIVE.LOCALITY bit is set. If the HASH.START is accepted, all write accesses to the data FIFO are delayed on the LPC bus (using wait states) until the firmware has reacted appropriately (to prevent inadvertently hashing wrong data). The ACCESS.ACTIVE.LOCALITY bit for locality 4 is set with this command. After a HASH.START has been issued correctly, the TPM will only accept a HASH.END or locality 4 HASH.DATA. All other cycles will be master-aborted.

5.2 ACCESS Register

ACCESS
Access Register **(x000$_H$)** **Reset Value: 00$_H$**

7	6	5	4	3	2	1	0
RVAL	–	ACTL	BSEI	SEIZ	PREQ	REQU	EST

Field	Bits	Type	Description
RVAL	7	r	**ACCESS.REG.VALID** This bit is a '1' to indicate that the other bits in this register are valid. If this bit is ’0’, then software must ignore the other bits in this register. This bit is '0' at reset and remains a '0' until the TPM has gone through its self-test and initialization and has established correct values in the other bits.
ACTL	5	rw	**ACCESS.ACTIVE.LOCALITY** If this bit is set, it indicates that a locality is active (e.g. may access the TPM). Writing a '1' to this bit clears both, the ACTIVE.LOCALITY bit and the REQUEST.USE bit and thus relinquishes control of this locality or takes back the access request (if only REQUEST.USE was set). Setting this bit from the host side aborts all currently running operations in the TPM if the writing locality is the currently active locality as well. An abort also occurs when a locality is granted access after having set its REQUEST.USE bit.
BSEI	4	rw	**ACCESS.BEEN.SEIZED** If this bit is set, it indicates that a locality had the TPM taken away while this locality had the ACTIVE.LOCALITY bit set. SW can use this bit to determine if it needs to abort an entire task and begin it again when it gets the TPM back again. Writing a '1' to this bit clears it.
SEIZ	3	w	**ACCESS.SEIZE** When SW writes a '1' to this bit, the TPM will reset the **ACCESS.ACTIVE.LOCALITY** bit and remove ownership for localities less than the locality which is writing this bit. Setting this bit does not affect the state of the **ACCESS.REQUEST.USE** bit for any locality except the one issuing the seize. For this locality, the TPM will set the **ACCESS.ACTIVE.LOCALITY** bit and clear the **ACCESS.REQUEST.USE** bit. When a write to ACCESS occurs, and ACCESS.SEIZE is set, then ACCESS.ACTIVE.LOCALITY and ACCESS.REQUEST.USE are ignored. If either or all of those 3 bits are set, the cycle is treated as a write only to ACCESS.SEIZE and the other bits are ignored.
PREQ	2	r	**ACCESS.PENDING.REQUEST** This bit indicates if some other locality is requesting usage of the TPM. This bit can be used by SW to determine if it should relinquish control of the TPM so some other locality can use it.

Field	Bits	Type	Description
REQU	1	rw	**ACCESS.REQUEST.USE** Setting this bit, a locality is requesting the TPM. This bit is cleared by the TPM when the requesting locality is granted access. This bit can only be cleared from the host software by writing a ’1’ to the **ACCESS.ACTIVE.LOCALITY** bit (of this locality). When there are multiple REQUEST.USE bits set, and the TPM is arbitrating for the next user, it chooses the highest locality that has its REQUEST.USE bit set.
EST	0	r	**ACCESS.TPM.ESTABLISHED** This bit will return a value of '1' until the first time that a secure environment has been established. The register will return a value of ’0’ thereafter. Once this bit is cleared, it will remain cleared independent of any reset, power cycling or any other event. This bit is ’0’ after reset and is set to ’1’ after TPM internal initialization is complete if and only if no secure environment has ever been established. **ACCESS.REG.VALID** should not be set until this bit has the correct value. The firmware must not set this bit to ’1’ after reset, even if **ACCESS.REG.VALID** is marked as invalid, unless '1' is the known correct value. This bit is set to ’1’ again when a RESET.TPM.ESTABLISHED command is issued. This command must be done from locality 3 or 4, otherwise it is not accepted. This allows the system to be returned to a clean state after manufacturing tests or when the system is given to a new owner.

5.3 Interrupt Registers

5.3.1 INT.ENABLE Register

INT.ENABLE
Interrupt Enable Register **(x008$_H$)** **Reset Value: 0000 0008$_H$**

31								0
GIE	RFU	CRE	RFU	LEV	LOE	VAE	AVE	

Field	Bits	Type	Description
GIE	31	rw	**INT.ENABLE.GLOBAL.INT.ENABLE** If this bit is cleared, no interrupts will be generated. If set, then interrupt generation is controlled by bits 0:2 of this register
RFU	30:8 6:5	–	**INT.ENABLE.RESERVED** Return all ’0’ on read accesses.

Field	Bits	Type	Description
CRE	7	rw	**INT.COMMAND.READY.ENABLE**
LEV	4:3	rw	**INT.ENABLE.TYPE.LEVEL** These bits set the interrupt type and level. The default is level triggered, active high. 00 level triggered, active high 01 level triggered, active low 10 edge triggered, rising 11 edge triggered, falling
LOE	2	rw	**INT.ENABLE.LOCALITY.CHANGE.INT.ENABLE** If this bit is set, the Locality Change Interrupt is enabled. This interrupt is caused whenever any locality moves from REQUEST.USE to ACTIVE.LOCALITY. *Note: If the TPM has no ACTIVE.LOCALITY set when REQUEST.USE is written, the TPM will move directly from REQUEST.USE to ACTIVE.LOCALITY without causing the interrupt.*
VAE	1	rw	**INT.ENABLE.STS.VALID.INT.ENABLE** If this bit is set, the **STS.VALID** Change Interrupt is enabled. This interrupt indicates a '0' to '1' transition on STS.VALID.
AVE	0	rw	**INT.ENABLE.DATA.AVAIL.INT.ENABLE** If this bit is set, the DATA.AVAIL Interrupt is enabled. This interrupt indicates that a transition from a '0' to a '1' occurred on **STS.DATA.AVAIL**.

If an interrupt is disabled, it will neither cause a SERIRQ# transition nor will the appropriate status bit be set when the interrupt condition occurs.

5.3.2 INT.VECTOR Register

INT.VECTOR
Interrupt Vector Register **(x00C$_H$)** **Reset Value: 00$_H$**

7	6	5	4	3	2	1	0
RFU				SIV			

Field	Bits	Type	Description
RFU	7:4	–	
SIV	3:0	rw	**INT.VECTOR.SERIRQ.VECTOR** Interrupt number used by the TPM. Can be anything from 0 to 15, platform compatibility must be checked before setting the vector.

5.3.3 INT.STATUS Register

INT.STATUS
Interrupt Status Register **(x010$_H$)** **Reset Value: 0000 0000$_H$**

31					0
RFU	CRS	RFU	LOS	VAS	AVS

Field	Bits	Type	Description
RFU	31:8 6:3	–	**INT.STATUS.RESERVED** Return all ’0’ on read accesses.
CRS	7	rw	**INT.STATUS.COMMAND.READY.INT.STAT** Indicates, when ’1’, that a Command Ready Interrupt occurred. Writing a '1' to this bit clears the interrupt.
LOS	2	rw	**INT.STATUS.LOCALITY.CHANGE.INT.STAT** Indicates, when ’1’, that a Locality Change Interrupt occurred. Writing a '1' to this bit clears the interrupt.
VAS	1	rw	**INT.STATUS.STS.VALID.INT.STAT** Indicates, when ’1’, that a STS.VALID Interrupt occurred. Writing a '1' to this bit clears the interrupt.
AVS	0	rw	**INT.STATUS.DATA.AVAIL.INT.STAT** Indicates, when ’1’, that a DATA.AVAIL Interrupt occurred. Writing a '1' to this bit clears the interrupt.

If an event remains set in the INT.STATUS register after the first one has been cleared out, the IRQ slot will be pulled low for one SERIRQ cycle (which triggers a SERIRQ) and then go high again in the next ones. The interrupt routine therefore has to reset all handled events with one write to INT.STATUS.

5.3.4 INT.CAPABILITY Register

INT.CAPABILITY
Interrupt Capability Register **(x014$_H$)** **Reset Value: 8000 00FF$_H$**

31										0
GSP	RFU	BCS	CRF	IEF	IER	ILL	ILH	LOP	VAP	AVP

Field	Bits	Type	Description
GSP	31	r	**INT.CAPABILITY.GLOBAL.INT.SUPPORT** Indicates if interrupts are supported at all by this TPM; ’1’ = supported '0' = not supported
RFU	30:9	–	**INT.CAPABILITY.RESERVED** Return all ’0’ on read accesses.

Field	Bits	Type	Description
BCS	8	r	**BURST.COUNT.STATIC** Indicates whether the **STS.BURST.COUNT** field is statuc or dynamic; ’1’ = static, '0' = dynamic
CRF	7	r	**INT.CAPABILITY.COMMAND.READY** command ready interrupts; ’1’ = supported, '0' = not supported
IEF	6	r	**INT.CAPABILITY.EDGE.FALLING** falling edge interrupts; ’1’ = supported, '0' = not supported
IER	5	r	**INT.CAPABILITY.EDGE.RISING** rising edge interrupts; ’1’ = supported, '0' = not supported
ILL	4	r	**INT.CAPABILITY.LOW.SUPPORT** active low level interrupts; ’1’ = supported, '0' = not supported
ILH	3	r	**INT.CAPABILITY.HIGH.SUPPORT** active high level interrupts; ’1’ = supported, '0' = not supported
LOP	2	r	**INT.CAPABILITY.LOCALITY.CHANGE.INT.SUPPORT** corresponds to interrupt bit no. 2; ’1’ = supported, '0' = not supported
VAP	1	r	**INT.CAPABILITY.STS.VALID.INT.SUPPORT** corresponds to interrupt bit no. 1; ’1’ = supported, '0' = not supported
AVP	0	r	**INT.CAPABILITY.DATA.AVAIL.INT.SUPPORT** corresponds to interrupt bit no. 0; ’1’ = supported, '0' = not supported

5.4 STS Register

STS
Status Register **(x018$_H$)** **Reset Value: XX00 0800$_H$**

31									0
–	BCNT	VAL	CDR	GO	DAV	EXP	RFU	RET	RFU

Field	Bits	Type	Description
BCNT	23:8	r	**STS.BURST.COUNT** If **STS.DATA.AVAIL** is set, then this register provides the number of reads from the FIFO that can be done without inserting wait states. If STS.DATA.AVAIL is not set, then this register provides the number of writes to the FIFO that can be done without wait state insertion. This field is dynamic, it changes with each byte that is read or written. The higher 8 bits are always ’0’, the counter shows the maximum number of available/free bytes in the 8-byte deep FIFO.
VAL	7	r	**STS.VALID** This bit indicates that both **STS.DATA.AVAIL** and **STS.EXPECT** are correct. If STS.VALID is not set, then **STS.DATA.AVAIL** and **STS.EXPECT** are not guaranteed to be correct. SW that is using **STS.DATA.AVAIL** or **STS.EXPECT** must poll on STS until STS.VALID is set. If the receive FIFO (as seen from the TPM) is not empty, this bit is cleared by hardware.
CDR	6	rw	**STS.COMMAND.READY** When '1', indicates that the TPM is ready to receive a new command. Powers up to '0' and is set to '1' when TPM is ready to accept the first command. COMMAND.READY is set only after FIFO is empty, so if SW starts sending a command and then restarts the command, SW needs to poll on STS.COMMAND.READY until it is '1'. This bit is cleared when the first byte of data is written into the receive FIFO (as seen from the TPM). SW writes a '1' to this bit to abort any currently processed command. The TPM will also empty the read and write FIFOs. *Note: If a command has been executed and SW writes a '1' to this bit before reading back the response, the TPM will clear the FIFOs. The pending response will be lost.*
GO	5	w	**STS.TPM.GO** After SW has written a command to the TPM and sees that it was correctly received, SW will write a '1' to this bit to cause the TPM to execute that command. This bit is not sticky.

Field	Bits	Type	Description
DAV	4	r	**STS.DATA.AVAIL** This bit indicates that the TPM has data available as a response. This bit may only be evaluated by the SW if the STS.VALID bit is set. When the host reads from the FIFO without the DATA.AVAIL bit being set **or** the DATA.AVAIL bit set but the STS.VALID bit reset, this will result in a master abort (the host will see FF$_H$ being read from the FIFO). When the response has been correctly received, SW must write a ’1’ to STS.COMMAND.READY to indicate that the response was correctly received. DATA.AVAIL is cleared if SW writes a '1' to COMMAND.READY, even though the response data has not been read.
EXP	3	r	**STS.EXPECT** The TPM sets this bit when it expects another byte of data for a command. It clears this bit when it has received all the data it expects for that command, based on the size field within the packet. This bit is set after COMMAND.READY is written and the FIFO is ready to receive data. This bit must stay set until the TPM has received the number of bytes it expects for that command. This bit may only be evaluated by the SW if the STS.VALID bit is set.
RFU	2	–	**STS.RESERVED** Return ’0’ on read
RET	1	w	**STS.RESPONSE.RETRY** SW writes a '1' to this bit to force the TPM to re-send the last response. Reads of this bit always return '0'.
RFU	0	–	**STS.RESERVED** Return ’0’ on read

5.5 DATAFIFO Register

DATAFIFO
Data Port Register **(x024$_H$)** **Reset Value: XXXX XXXX$_H$**

31			0
DATA	DATA	DATA	DATA

Field	Bits	Type	Description
DATA	31:24 23:16 15:8 7:0	rw	**DATAFIFO.DATAFIFO** Bidirectional communication port. Host ⇨TPM: SW writes packets to this port. A command consists of multiple bytes, but the host writes it 1 byte at a time. SW should read **STS.BURST.COUNT** to determine how many consecutive bytes it should send to the TPM. TPM ⇨Host: SW reads packet return data and return status from this port. The return packet for a command is multiple bytes, but the host reads it 1 byte at a time. Software should read **STS.BURST.COUNT** to determine how many consecutive bytes it may burst. A read to the FIFO when the DATA.AVAIL bit is ’0’ is aborted. The TPM does neither drops writes on LPC when it is not able to accept data nor does it abort reads on LPC if no data is in the FIFO (and DATA.AVAIL = ’1’). Instead it wait-states the LPC bus until the access can be handled. SW is not required to read **STS.BURST.COUNT**, but should do so for better general system performance.

5.6 DID/VID Registers

DID
Device Identification Register **($xF00_H$)** **Reset Value: 000B 15D1$_H$**

31	0
DID	VID

Field	Bits	Type	Description
DID	31:16	r	**DID.DID** Device ID of the TPM SLB 9635 TT V1.2: 000B$_H$
VID	15:0	r	**DID.VID** PCI Vendor ID of Infineon AG: 15D1$_H$

5.7 RID

RID
Revision Identification Register **($xF04_H$)** **Reset Value: 10$_H$**

7	6	5	4	3	2	1	0
RID							

Field	Bits	Type	Description
RID	7:0	r	**RID.REVISION.ID** Revision ID of the TPM in BCD (2 digits). First revision is 1.0, to be changed if chip updates (mask changes) occur.

5.8 Legacy Registers

LEGACY1
Legacy Register 1 **(0F80$_H$)** **Reset Value: xx$_H$**

<table>
<tr><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="8">DATAL1</td></tr>
</table>

LEGACY1B
Legacy Register 1B **(0F84$_H$)** **Reset Value: 00$_H$**

<table>
<tr><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="8">RES</td></tr>
</table>

LEGACY2
Legacy Register 2 **(0F88$_H$)** **Reset Value: xx$_H$**

<table>
<tr><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="8">DATAL2</td></tr>
</table>

LEGACY2B
Legacy Register 2B **(0F8C$_H$)** **Reset Value: 00$_H$**

<table>
<tr><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="8">RES</td></tr>
</table>

Field	Bits	Type	Description
DATAL1	7:0	rw	**LEGACY1.DATA** Legacy **index** configuration register, for a description please refer to **chapter 3.1.1**. Only the uppermost 14 bits are decoded for the register address, that means the LEGACY1 register can be accessed on 0F81$_H$ - 0F83$_H$ as well. This register can be accessed only from locality 0.
DATAL2	7:0	rw	**LEGACY2.DATA** Legacy **data** configuration register, for a description please refer to **chapter 3.1.1**. Only the uppermost 14 bits are decoded for the register address, that means the LEGACY1 register can be accessed on 0F89$_H$ - 0F8B$_H$ as well. This register can be accessed only from locality 0.

Field	Bits	Type	Description
RES	7:0	r	**LEGACY1B.RES, LEGACY2B.RES** These registers are reserved. Write accesses have no effect, read accesses return FF$_H$. Only the uppermost 14 bits are decoded for the register addresses, that means the LEGACY1B register can be accessed on 0F85$_H$ - x087$_H$ as well and the LEGACY2B register can be accessed on 0F8D$_H$ - 0F8F$_H$ as well. These registers can be accessed only from locality 0.

5.9 Timeout Register

TO
Timeout Register **(xF90$_H$)** **Reset Value: 00$_H$**

7	6	5	4	3	2	1	0
TIMEOUT							

Field	Bits	Type	Description
TIMEOUT	7:0	r	**TO.TIMEOUT** Timeout value for the excecuting command (in seconds). This register can be read from all localities. Holds all zeros when no command is executing or a command is completed. When a command is sent to the TPM and the GO bit is written, the TPM will place the timeout value for that command into this register. The register is cleared by hardware on read accesses, but this only occurs if the read access is done from the active locality. If the command is not finished after the indicated time, the TPM firmware will re-set this register to the expected remaining time. The TPM may complete the command before the timeout is reached.

5.10 Other Configuration Registers

The address space from xF05$_H$ - xF7F$_H$ is reserved for future TCG defined configuration registers. All reads to that address space return 00$_H$ (not implemented). Writes will be aborted.

The address space from xF91$_H$ - xFFF$_H$ is reserved for vendor defined configuration registers. All reads to that address space return 00$_H$ (not implemented). Writes will be aborted.

This page has been left blank intentionally.

6 General Overview

6.1 Block Diagram

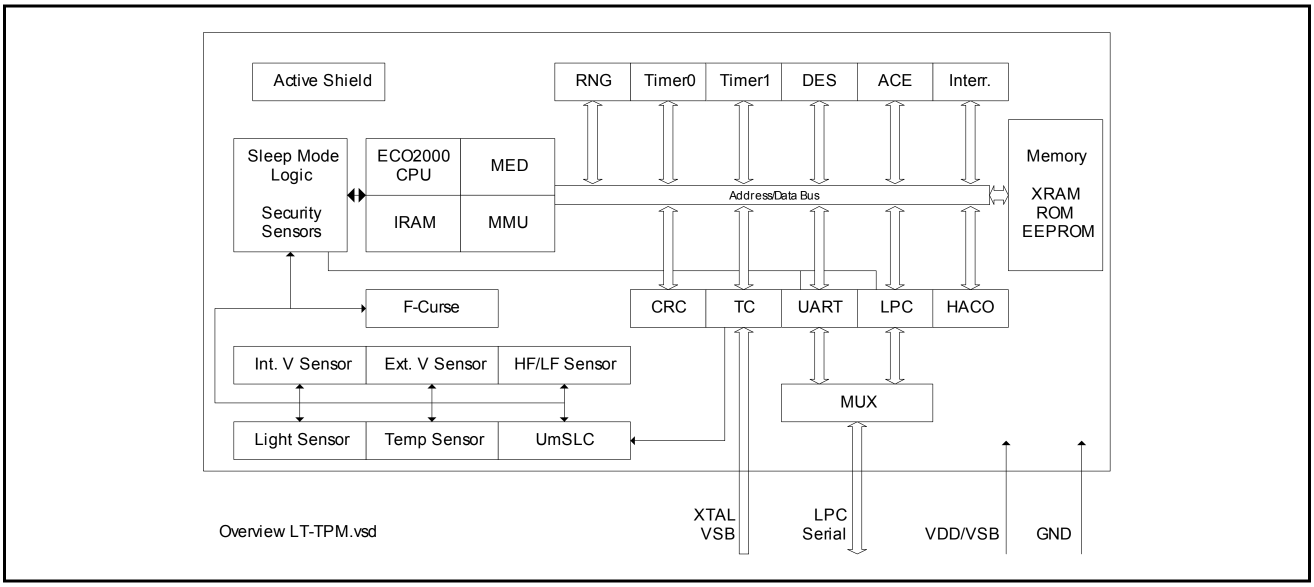


Figure 6-1 Block Diagram of the TPM

The TPM is based on an Infineon 66PE secure controller core. This core contains a variety of modules like CPU core, internal RAM, ROM, EEPROM, cryptographic accelerators etc. Additionally, a hardware hash accelerator and a specialized interface have been added (Low Pin Count interface, LPC, see [1]). This LPC interface is the main interface of the chip.

The clock for the controller is derived from the LPC clock which runs at nominally 33 MHz. This clock is scaled down by a factor of 8 and then fed to the PLL of the secure controller core.

6.2 Hardware Interface

The TPM uses the Low Pin Count (LPC) Interface as defined by Intel to connect to the surrounding system (for details on this bus, please refer to [1]). An 8-byte FIFO decouples the LPC bus from the internal controller bus. The LPC interface of the TPM only supports slave I/O functionality, i.e. only cycle types I/O read and I/O write (and the LT specific versions of these two cycles). All other cycles are ignored by the chip.

The configuration of the (legacy) host side of the LPC is done via the standard plug-and-play mechanism. For this purpose, a standard register set is available (refer to **chapter 3.1**).

The interface supports the LPCPDn line which is used to detect a power-off condition on the LPC bus. For signalling interrupts, the SERIRQ protocol is used.

6.3 Hash Accelerator

The hash accelerator supports the SHA-1 hash algorithm. It is used by the internal firmware to speed up hash calculations which greatly enhances the performance compared to a software solution.

6.4 Firmware

The firmware of the chip includes a basic operating system and provides the TCG functionality. It also provides the secure code download capability[1] for field upgrades. For a description of the command set which is available at the logical interface, please refer to [5] and [6].

For further description of the embedded software (firmware) and the device drivers, please refer to chapter 10.

[1] It should be noted that firmware can only be downloaded to the chip if it has been crypted and signed by Infineon. This prevents unauthorized changes of the firmware.

7 Serial Interrupt Request

The LPC interface module supports the SERIRQ protocol as defined in [3].

7.1 SERIRQ Cycle Control

There are two modes of operation for the SERIRQ Start Frame.

7.1.1 Quiet (Active) Mode

Any device on the bus may initiate a start frame by driving the SERIRQ line low for one clock, while the SERIRQ signal is idle. After driving low for one clock, the SERIRQ line must be immediately tristated without at any time being driven high. A start frame may not be initiated while the SERIRQ line is active. The line is idle between stop and start frames and is active between start and stop frames. This mode of operation allows the SERIRQ line to be idle when there are no pending interrupts which should be most of the time.

Once a start frame has been initiated by the LPC interface module, the host controller will take over driving the SERIRQ line low in the next clock and will continue to drive the line low for a period between three to seven clocks. This makes a total low pulse width of four to eight clocks. Finally, the host controller will drive the SERIRQ back high for one clock and then tristate the signal.

When the LPC interface module needs to deliver an interrupt, it must initiate s start frame in order to update the host controller unless the SERIRQ is already in an SERIRQ cycle and the interrupt can be delivered in that cycle.

Note: A start frame is generated by the LPC interface module if the interrupt state changes. This also occurs if, for instance,only the interrupt level is changed while interrupts are disabled.

7.1.2 Continuous (Idle) Mode

Only the host controller can initiate a start frame to update the IRQ information. All other SERIRQ agents become passive and may not initiate a start frame. SERIRQ will be driven low for four to eight clocks by the host controller. This mode has two functions. It can be used to stop or idle the SERIRQ line or the host controller can operate SERIRQ in continuous mode by initiating a start frame at the end of ever y stop frame.

Upon reset, the SERIRQ bus defaults to continuous mode, therefore only the host controller can initiate the first start frame. The LPC interface module must continuously sample the stop frame pulse width to determine the next SERIRQ cycle mode.

7.2 SERIRQ Data Frame

Once a start frame has been initiated, the LPC interface module will watch for the rising edge of the start pulse and start counting IRQ/Data frames from there. Each IRQ/Data frame is three clocks long: sample phase, recovery phase, and turn-around phase.

During the sample phase, the LPC interrupt module must drive the SERIRQ line low if and only if an interrupt request is available. If no interrupt request is available, the line must be left tristated.

During the recovery phase, the SERIRQ signal must be driven high if and only if it was driven low in the previous sample phase.

During the turn-around phase, the SERIRQ line must be tristated.

The LPC interface module will drive the SERIRQ line low at the appropriate sample point if an interrupt request is available, regardless of which device initiated the start frame.

Note: The SERIRQ line is driven low if an interrupt is active and the interrupt level is set to 'active low' or if no interrupt is pending and the level is set to 'active high'. In all other cases, SERIRQ is left tristated. This also implies that the LPC interface module never drives a '1' onto that line.

The sample phase of each IRQ/Data frame follows the low-to-high transition of the start frame pulse by a number of clocks which is equal to the IRQ/Data frame times three, minus one (e.g. the IRQ5 sample clock is the sixth IRQ Data frame, the sample phase is located at the seventeenth clock after the rising edge of the start pulse since (6 x 3 - 1) = 17).

7.3 Stop Cycle Control

Once all IRQ/Data frames have been completed, the host controller will terminate the SERIRQ activity by initiating a stop frame. A stop frame is indicated when the SERIRQ line is low for two or three clocks. If the low time is two clocks, then the next SERIRQ cycle's sample mode is the quiet mode, and any SERIRQ device may initiate a start frame in the second clock or later after the rising edge of the stop frame pulse.

If the length of the stop frame is three clocks, then the next SERIRQ's sample mode is the continuous mode, and only the host controller may initiate a start frame in the second clock or later after the rising edge of teh stop frame pulse.

7.4 Reset and Initialization

The SERIRQ bus uses the PCI reset signal as its reset signal (i.e. the LRESETn signal). The SERIRQ pin must be tristated while the LRESETn signal is asserted. With the reset signal, the SERIRQ controller of the LPC interface module is put into the (continuous) idle mode. This means that the first start frame will be initiated by the host controller. The SERIRQ controller of the LPC interface module then follows with the continuous/quiet mode protocol (as defined by the stop frame pulse width) for subsequent SERIRQ cycles.

8 TPM Windows Device Driver

For all communications between the TPM Software Stack and the TPM there is a OS level device driver included in the delivery package. In conjunction with the TCG device driver library (TDDL), an easy to use interface is available to transfer the TCG byte streams to/from the TPM. For a detailed description of the TDDL please refer to [10].

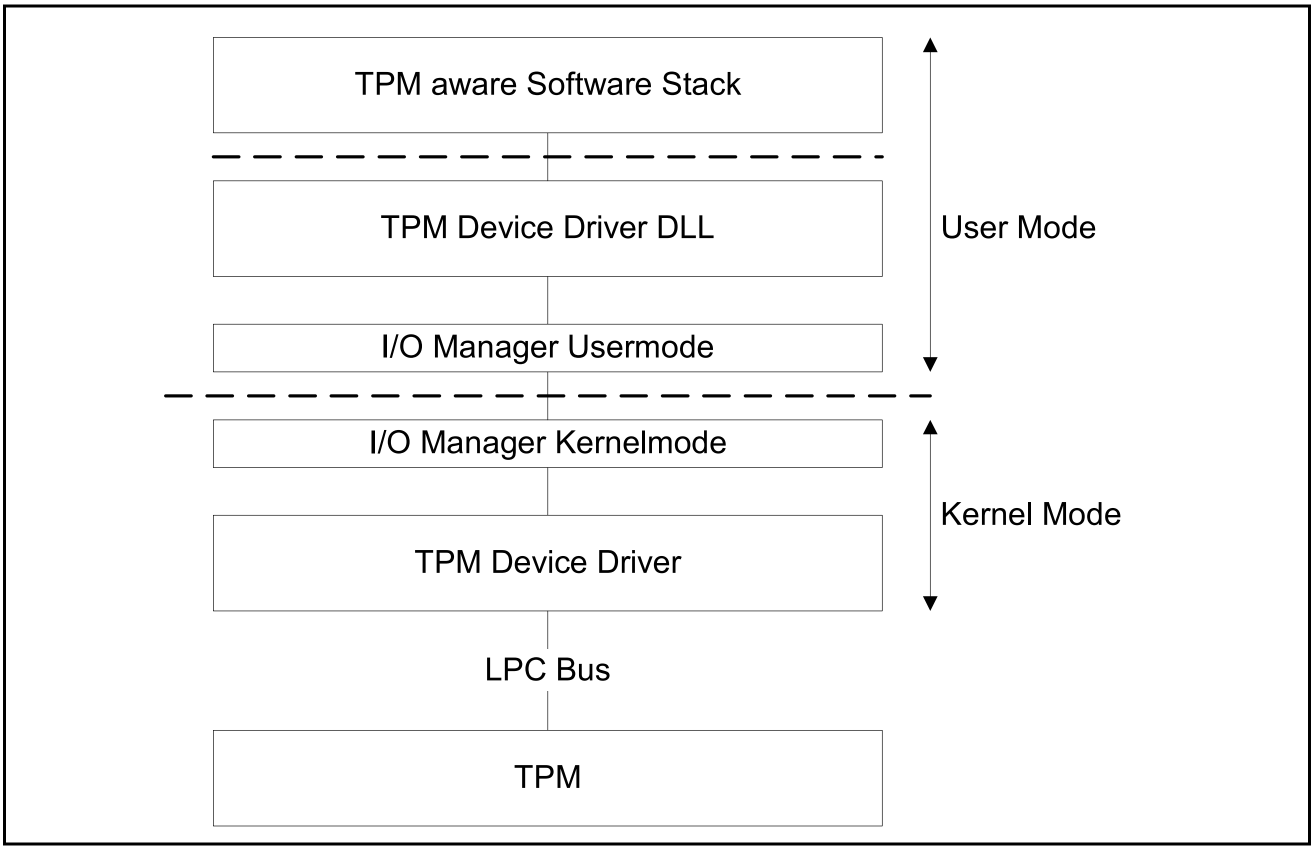


Figure 8-1 TPM Software Stack

The TPM Device Driver is implemented as a Class/Function Driver supporting the Windows Driver Model (WDM). It is WHQL certified for Windows 2000, Windows 2003 Server and Windows XP Operating Systems.

9 TPM BIOS Device Driver

The TPM Bios Device Driver package consists of the Memory Absent (MA) driver and the Memory Present (MP) driver as specified in [7]. For a detailed description please also refer to [9].

10 TPM Embedded Software

The embedded software of the Infineon TPM SLB9635TT1.2 has been developed to be fully compliant to [5] and [6]. Nevertheless, in some cases the implementation deviates from the referenced specifications or vendor specific extensions are necessary.

Furthermore the actual parameters of the implementation need to be documented as the specified values are only minimum requirements.

10.1 Available Resources

Table 10-1 Available Resources

Number of PCRs	24
Amount of free NV Memory	1.5 kBytes
Maximum number of concurrent sessions	up to 20 depending on the number of loaded keys
Maximum number of volatile Keys	Up to 10 2048-bit Keys can be loaded into the volatile storage depending on the number of open sessions
Nonvolatile Keyslots	6 for up to 2048-bit key smaller keys will still use a complete keyslot
Delegation Rows	2
Family Table Rows	8
Maximum number of concurrent saved context other than key	32
Maximum number of monotonic counters	8
I/O-Buffer size[1]	1280 Byte

[1] To calculate the correct amount of data which can be exchanged with the TPM (for instance using TPM_NV_WriteValue), care shall be taken to consider 10 Bytes for communication layer overhead and also to consider the overhead required for the TCG command layout.

10.2 Command Ordinal List

The following ordinals are implemented. All other ordinals will return TPM_BAD_ORDINAL.

Table 10-2 Command Ordinal List

TPM_ORD_ActivateIdentity	TPM_ORD_MakeIdentity
TPM_ORD_AuthorizeMigrationKey	TPM_ORD_MigrateKey
TPM_ORD_CertifyKey	TPM_ORD_NV_DefineSpace
TPM_ORD_CertifyKey2	TPM_ORD_NV_ReadValue
TPM_ORD_ChangeAuth	TPM_ORD_NV_ReadValueAuth
TPM_ORD_ChangeAuthOwner	TPM_ORD_NV_WriteValue
TPM_ORD_CMK_ApproveMA	TPM_ORD_NV_WriteValueAuth
TPM_ORD_CMK_ConvertMigration	TPM_ORD_OIAP
TPM_ORD_CMK_CreateBlob	TPM_ORD_OSAP
TPM_ORD_CMK_CreateKey	TPM_ORD_OwnerClear
TPM_ORD_CMK_CreateTicket	TPM_ORD_OwnerReadInternalPub
TPM_ORD_CMK_SetRestrictions	TPM_ORD_OwnerSetDisable

Table 10-2 Command Ordinal List (cont'd)

TPM_ORD_ContinueSelfTest	TPM_ORD_PCR_Reset
TPM_ORD_ConvertMigrationBlob	TPM_ORD_PcrRead
TPM_ORD_CreateCounter	TPM_ORD_PhysicalDisable
TPM_ORD_CreateEndorsementKeyPair	TPM_ORD_PhysicalEnable
TPM_ORD_CreateMigrationBlob	TPM_ORD_PhysicalSetDeactivated
TPM_ORD_CreateWrapKey	TPM_ORD_Quote
TPM_ORD_DAA_JOIN	TPM_ORD_Quote2
TPM_ORD_DAA_SIGN	TPM_ORD_ReadCounter
TPM_ORD_Delegate_CreateKeyDelegation	TPM_ORD_ReadPubek
TPM_ORD_Delegate_CreateOwnerDelegation	TPM_ORD_ReleaseCounter
TPM_ORD_Delegate_LoadOwnerDelegation	TPM_ORD_ReleaseCounterOwner
TPM_ORD_Delegate_Manage	TPM_ORD_ReleaseTransportSigned
TPM_ORD_Delegate_ReadTable	TPM_ORD_ResetLockValue
TPM_ORD_Delegate_UpdateVerification	TPM_ORD_SaveContext
TPM_ORD_Delegate_VerifyDelegation	TPM_ORD_SaveState
TPM_ORD_DisableForceClear	TPM_ORD_Seal
TPM_ORD_DisableOwnerClear	TPM_ORD_SelfTestFull
TPM_ORD_DSAP	TPM_ORD_SetCapability
TPM_ORD_EstablishTransport	TPM_ORD_SetOperatorAuth
TPM_ORD_ExecuteTransport	TPM_ORD_SetOwnerInstall
TPM_ORD_Extend	TPM_ORD_SetOwnerPointer
TPM_ORD_FieldUpgrade	TPM_ORD_SetTempDeactivated
TPM_ORD_FlushSpecific	TPM_ORD_SHA1Complete
TPM_ORD_ForceClear	TPM_ORD_SHA1CompleteExtend
TPM_ORD_GetCapability	TPM_ORD_SHA1Start
TPM_ORD_GetCapabilityOwner	TPM_ORD_SHA1Update
TPM_ORD_GetPubKey	TPM_ORD_Sign
TPM_ORD_GetRandom	TPM_ORD_Startup
TPM_ORD_GetTestResult	TPM_ORD_StirRandom
TPM_ORD_GetTick	TPM_ORD_TakeOwnership
TPM_ORD_IncrementCounter	TPM_ORD_TickStampBlob
TPM_ORD_Init	TPM_ORD_UnBind
TPM_ORD_KeyControlOwner	TPM_ORD_Unseal
TPM_ORD_LoadContext	TSC_ORD_PhysicalPresence
TPM_ORD_LoadKey2	TSC_ORD_ResetEstablishmentBit

10.3 Dictionary Attack Prevention

To protect the authorization values stored within the TPM against Dictionary Attacks, a suitable prevention mechanism has been implemented. The details to understand and configure this prevention mechanism is available as separate document upon request [8].

10.4 NV Storage

10.4.1 Predefined NV Indices

The following predefined NV Indices, besides those mandatorily defined in [5], [6] and [7], are available:

Table 10-3 Predefined NV Indices

Value	Index Name	Default Size	Attributes
0x1000F000	TPM_NV_INDEX_EKCert	The total size of this index is 1.75 kBytes, actual size can vary depending on the size of the certificate currently loaded. This index holds the pre-loaded Infineon TPM Endorsement Key Certificate.	TPM_NV_PER_OWNERWRITE TPM_NV_PER_OWNERREAD
0x30000001	TPM_NV_INDEX_VRSNCert	The total size of this index is 0.6 kBytes, actual size can vary depending on the size of the certificate currently loaded. This index holds the pre-loaded Verisign Class 3 Primary CA Certificate.	TPM_NV_PER_OWNERWRITE

10.4.2 Reserved NV Indices

The following reserved NV Indices, besides those mandatorily defined in [5], [6] and [7], are available:

Value	Index Name	Default Size	Attributes
0x00011680	TPM_NV_INDEX_GPIO_80	This index is reserved and before use it must be allocated via TPM_NV_DefineSpace. For this index the same rules are applicable as for TPM_NV_INDEX_GPIO_00, see [1] for further details.	—

10.5 Extensions and Deviations from the TPM Main-Specification

10.5.1 TPM_ContinueSelftest

If TPM_ContinueSelftest is called, all outstanding selftests are performed before the TPM returns to the caller with the appropriate error return code either TPM_SUCCESS or TPM_FAILEDSELFTEST.

10.5.2 TPM_GetTestResult

The TPM will return to the caller with outData = UINT16, a bit-field describing the result of the selftest.

10.5.3 TPM_OwnerClear, TPM_ForceClear

The TPM will maintain the TPM_PERMANENT_FLAG allowMaintenance at FALSE as the maintenance functionality is not supported.

10.5.4 TPM_GetCapability

If TPM_CAPABILITY_AREA = TPM_CAP_VERSION_VAL the following data will be returned: TPM_VERSION with revMajor and revMinor the hexadecimal representation of the embedded software version currently installed on the TPM. For example, the embedded software version 00.90 will be returned with revMajor = 0x00 and revMinor = 0x5A.

Further capabilities regarding the Dictionary Attack Prevention can be retrieved via capArea = TPM_CAP_MFR. For a detailed description refer to [8].

10.5.5 TPM_SetCapability

Further capabilities regarding the Dictionary Attack Prevention can be set via capArea = TPM_SET_VENDOR. For a detailed description refer to [8].

10.5.6 TPM_FieldUpgrade

Start of informative comment:

The TPM fieldupgrade process is divided into several parts. The first part should be getting all necessary information to do an appropriate update.

End of informative comment.

Table 10-4 FieldUpgrade-Specific return codes

Name	Value	Description
TPM_BASE	0x00000000	The start of TPM return codes
TPM_FU_BAD_PARAMETER	TPM_BASE + 0x81	One or more parameter is bad
TPM_FU_WRONG_RND_VALUE	TPM_BASE + 0x82	The parameter file and the fieldupgrade data file are inconsistent
TPM_FU_DECRYPTION_ERROR	TPM_BASE + 0x88	The encryption of the fieldupgrade data are not compliant to the TPM to be upgraded
TPM_FU_VERIFY_ERROR	TPM_BASE + 0x89	The signature of the fieldupgrade data does not match the expected value
TPM_FU_UNEXPECTED_ERROR	TPM_BASE + 0x8A	An error occurred with no further explanation
TPM_FU_WRONG_ROM_CRC	TPM_BASE + 0x8B	The fieldupgrade data don't fit to the TPM
TPM_FU_WRONG_VERSION	TPM_BASE + 0x8C	The version of fieldupgrade data can't loaded on to the current version of the TPM
TPM_FU_WRONG_UPDATE_STATUS	TPM_BASE + 0x8D	The sequence of the fieldupgrade is wrong

IDL Definitions of subCommand

```
#define TPM_FieldUpgradeInfoRequest     0x10
#define TPM_FieldUpgradeStart           0x34
#define TPM_FieldUpgradeUpdate          0x31
#define TPM_FieldUpgradeComplete        0x32
```

IFX_FieldUpgradeInfo

Definition:

```
typedef struct tdIFX_FieldUpgradeInfo{
  BYTE infoVersion;
  BYTE[4] chipVersion;
  TCPA_VERSION ver;
  BYTE[3] date;
  UINT16 maxDataSize;
  UINT16 romCRC;
  BYTE keyInfoKTTP;
  BYTE keyInfoSig;
  UINT16 flagsFieldUpgrade;
} IFX_FIELDUPGRADEINFO;
```

Parameters

Table 10-5 Parameters for TPM_FieldUpgrade

Type	Name	Description
BYTE	InfoVersion	This SHALL be the version of IFX_FieldUpgradeInfo.
BYTE[4]	chipVersion	Returns the Identifier of the Chip Version
TCPA_VERSION	ver	Version number defined in TCPA main specification.
BYTE[3]	date	This SHALL be the date info:dd.mm.yy in decimal notation
UINT16	maxDataSize	This SHALL be max. Data Size which can be sent to the TPM by FieldUpgrade subcommands.
UINT16	romCRC	This SHALL be the CRC of the complete ROM.
BYTE	keyInfoKTTP	This SHALL be the version of the named Key.
BYTE	keyInfoSig	This SHALL be the version of the named Key.
UINT16	flagsFieldUpgrade	0x0001: ownerSetFlag indicates the the TPM has currently an Owner 0x0002: noOwnerFUBlockFlag indicates that the FieldUpgrade without OwnerAuth is not possible. This bit can only be set to TRUE by a special FieldUpgrade version; it cannot be set to FALSE except during the TPM manufacturing process. 0x0004: firmwareValidFlag indicates that the Firmware is valid

10.5.6.1 TPM_FieldUpgradeInfoRequest

Start of informative comment:

This capability provides all relevant information about the TPM chip to the caller to be able to start with the Upgrade process. Because not part of the TPM_Fieldupgrade thread this subcommand can always be called under normal operation conditions.

End of informative comment.

Type

TCPA protected capability

Incoming Operands and Sizes

PARAM		HMAC		Type	Name	Description
#	SZ	#	SZ			
1	2			TCPA_TAG	tag	TPM_TAG_RQU_COMMAND
2	4			UINT32	paramSize	Total number of input bytes including paramSize and tag
3	4			TCPA_COMMAND_CODE	ordinal	Command ordinal, fixed value of TPM_FieldUpgrade
4	1			BYTE	subCommand	infoRequest
5	2			UINT16	inInfoRequestSize	0x00

Outgoing Operands and Sizes

PARAM		HMAC		Type	Name	Description
#	SZ	#	SZ			
1	2			TCPA_TAG	tag	TPM_TAG_RSP_COMMAND
2	4			UINT32	paramSize	Total number of output bytes including paramSize and tag
3	4			TCPA_RESULT	returnCode	The return code of the operation. See section 4.3. of TCPA main specification
4	2			UINT16	outInfoRequestSize	Outgoing size of infoRequest
5	<>		<>	BYTE []	outInfoRequestData	Outgoing data (for further information see IFX_FieldUpgradeInfo)

Description

This command SHALL supports the caller with the necessary information about max. size of net data, version and other relevant data.

Specific Error Return Codes

TCPA_BAD_PARAMETER:	invalid subCommand
TCPA_BAD_PARAM_SIZE:	wrong paramSize

10.5.6.2 TPM_FieldUpgradeStart

Start of informative comment:

This capability starts the Upgrade Process with the first block of Data (Parameterblock).

End of informative comment.

Type

TCPA protected capability

Incoming Operands and Sizes

PARAM		HMAC		Type	Name	Description
#	SZ	#	SZ			
1	2			TCPA_TAG	Tag	TPM_TAG_RQU_AUTH1_COMMAND
2	4			UINT32	paramSize	Total number of input bytes including paramSize and tag
3	4	1s	4	TCPA_COMMAND_CODE	ordinal	Command ordinal, fixed value of TPM_FieldUpgrade
4	1	2s	1	BYTE	subCommand	TPM_FieldUpgradeStart
5	2	3s	2	UINT16	sizeOfData	
6	<>	4s	<>	BYTE[]	fieldUpgradeData	
7	4			TCPA_AUTHHANDLE	authHandle	The authorization handle used for Owner authorization
8	20	2_{H1}	20	TCPA_NONCE	nonceEven	Even nonce newly generated by TPM to cover outputs
		3_{H1}	20	TCPA_NONCE	nonceOdd	Nonce generated by system associated with authHandle
9	1	4_{H1}	1	BOOL	continueAuthSession	Ignored
10	20			TCPA_AUTHDATA	resAuth	The authorization digest for the returned parameters. HMAC key: ownerAuth.

Outgoing Operands and Sizes

PARAM		HMAC		Type	Name	Description
#	SZ	#	SZ			
1	2			TCPA_TAG	tag	TPM_TAG_RSP_AUTH1_COMMAND
2	4			UINT32	paramSize	Total number of output bytes including paramSize and tag
3	4	1s	4	TCPA_RESULT	returnCode	The return code of the operation. See section 4.3. of TCPA main specification
		2s	4	TCPA_COMMAND_CODE	ordinal	Command ordinal, fixed value of TPM_FieldUpgrade
4	2	3s	2	UINT16	outStartSize	0x00
5	20	2_{H1}	20	TCPA_NONCE	nonceEven	Even nonce newly generated by TPM to cover outputs
		3_{H1}	20	TCPA_NONCE	nonceOdd	Nonce generated by system associated with authHandle
6	1	4_{H1}	1	BOOL	continueAuthSession	Continue use flag, fixed value of FALSE
7	20			TCPA_AUTHDATA	resAuth	The authorization digest for the returned parameters. HMAC key: ownerAuth.

10.5.6.3 TPM_FieldUpgradeUpdate

Start of informative comment:

This capability shall be called as often as the complete Firmware is upgraded.

End of informative comment.

Type

TCPA protected capability

Incoming Operands and Sizes

PARAM		HMAC		Type	Name	Description
#	SZ	#	SZ			
1	2			TCPA_TAG	tag	TPM_TAG_RQU_COMMAND
2	4			UINT32	paramSize	Total number of input bytes including paramSize and tag
3	4			TCPA_COMMAND_CODE	ordinal	Command ordinal, fixed value of TPM_FieldUpgrade
4	1			BYTE	subCommand	TPM_FieldUpgradeUpdate
5	2			UINT16	inUpdateSize	Size from TPM_FieldUpgrade (infoRequest). Must be mod 8
6	<>			BYTE[]	inUpdateData	TPM_FieldUpgrade Data.

Outgoing Operands and Sizes

PARAM		HMAC		Type	Name	Description
#	SZ	#	SZ			
1	2			TCPA_TAG	tag	TPM_TAG_RSP_COMMAND
2	4			UINT32	paramSize	Total number of output bytes including paramSize and tag
3	4			TCPA_RESULT	returnCode	The return code of the operation. See section 4.3. of TCPA main specification
4	2			UINT16	outUpdateSize	0x00

Description

This command SHALL incorporate blocks of upgrade data.

10.5.6.4 TPM_FieldUpgradeComplete

Start of informative comment:

This capability is the last command of the TPM_FieldUpgrade.

End of informative comment.

Type

TCPA protected capability

Incoming Operands and Sizes

PARAM		HMAC		Type	Name	Description
#	SZ	#	SZ			
1	2			TCPA_TAG	tag	TPM_TAG_RQU_COMMAND
2	4			UINT32	paramSize	Total number of input bytes including paramSize and tag
3	4			TCPA_COMMAND_CODE	ordinal	Command ordinal, fixed value of TPM_FieldUpgrade
4	1			BYTE	subCommand	TPM_FieldUpgradeComplete
5	2			UINT16	inCompletetSize	Max. Size from TPM_FieldUpgrade (infoRequest)
6	<>			BYTE[]	inCompletetData	TPM_FieldUpgrade Data.

Outgoing Operands and Sizes

PARAM		HMAC		Type	Name	Description
#	SZ	#	SZ			
1	2			TCPA_TAG	tag	TPM_TAG_RSP_COMMAND
2	4			UINT32	paramSize	Total number of output bytes including paramSize and tag
3	4			TCPA_RESULT	returnCode	The return code of the operation. See section 4.3. of TCPA main specification
4	2			UINT16	outCompleteSize	0x00

11 Performance Values

11.1 Hashing (SHA1)

TPM_SHA1Start:	≤ 2 ms
TPM_SHA1Update (1024 Bytes):	≤ 5 ms
TPM_SHA1CompleteExtend (0 Bytes):	≤ 2.5 ms

11.2 Symmetric Cryptography

TPM_SaveContext:	≤ 50 ms
TPM_LoadContext:	≤ 40 ms

11.3 Asymmetric Keygeneration

TPM_CreateWrapKey (1024 Bit):	typical 3.5 s
TPM_CreateWrapKey (2048 Bit):	typical 20.2 s

11.4 Asymmetric Cryptography

TPM_Sign (1024 Bit RSA Key):	≤ 325 ms
TPM_Sign (2048 Bit RSA Key):	≤ 600 ms
TPM_Seal (2048 Bit RSA Key):	206 ms
TPM_Unseal (2048 Bit RSA Key):	367 ms

11.5 Transport encryption

The overhead caused by transport encryption is 30 ms – 60 ms depending on the amount of data.

This page has been left blank intentionally.

12 Characteristics

12.1 Electrical Characteristics

Table 12-1 Absolute Maximum Ratings

Parameter	Symbol	Values			Unit	Note/Test Condition
		min.	typ.	max.		
Operating temperature	T_A	0	–	70	°C	–
Supply voltage	VDD, VSB	-0.3	–	3.6	V	–
Voltage on any pin (pins 1..14)	V_{max}	-0.3	–	VSB + 0.3	V	–
Voltage on any pin (pins 15..28)	V_{max}	-0.3	–	VDD + 0.3	V	–
ESD robustness HBM: 1.5 kΩ, 100 pF	$V_{ESD,HBM}$	2000			V	According to EIA/ JESD22-A114-B
ESD robustness	$V_{ESD,SDM}$	500			V	According to ESD Association Standard DS5.3.1 - 1999
Latchup immunity	I_{LATCH}	100			mA	According to EIA/ JESD78

Note: ***Stresses above those listed here may cause permanent damage to the device. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.***
Maximum ratings are absolute ratings; exceeding only one of these values may cause irreversible damage to the integrated circuit.

12.2 Functional Operating Range

Table 12-2 Functional Operating Range

Parameter	Symbol	Limit Values			Unit
		Min	Typ	Max	
Supply Voltage (LPC pads)	*VDD*	3.0	3.3	3.6	V
Supply Voltage (core)	*VSB*	3.0	3.3	3.6	V
Operating Temperature	T_A	0		70	°C
Useful lifetime[1]				5	y
Operating lifetime[1]				5	y
Average T_A over lifetime			55		° C

[1] The useful lifetime of the device is 5 (five) years with a duty cycle (that means, a power-on time) of 100%. An useful lifetime of 7 (seven) years can be guaranteed for a duty cycle of 70%. For both scenarios, it is assumed that the device will be used for calculations for approximately 5% of the maximum useful lifetime.

12.3 DC Characteristics

Table 12-3 Current Characteristics

T_A = 0°C to 70°C, *VDD* = *VSB* = 3.3V +- 0.3V unless otherwise noted

Parameter	Symbol	Limit Values			Unit
		Min	Typ	Max	
Power consumption[1]	I_{VDD}		1.4[2]	5	mA
Power consumption[3]	I_{VDD}			0.5	mA
Power consumption[1]	I_{VSB}		3.4[2]	25	mA
Power consumption[3]	I_{VSB}			0.5	mA

[1] Assuming operating state S0, that means active. Note that since the device is mostly in an internal sleep state in a "typical" application, the typical average current consumption is far less than the maximum value.

[2] It is assumed that in a normal environment, the device is in an internal sleep state for approximately 90% of the operating time of the platform.

[3] Assuming operating state S3, i.e. powerdown and clock stopped. If the clock remains active, the current consumption is 1 mA (maximum value, valid for both I_{VDD} and I_{VSB}). Obviously, this value is zero if the TPM is not powered in S3 state (this is platform dependent).

Note: Current consumption does not include any currents flowing through resistive loads on output pins! For the definition of power/operating states, please refer to the ICH chipset specifications of Intel and to ***section 1.18****.*

Table 12-4 DC Characteristics for non-LPC Pins

Pins GPIO, GPIO2, PP, TESTBI/BADD, TESTI, XTALI, XTALO

T_A = 0°C to 70°C, VDD = VSB = 3.3V +- 0.3V unless otherwise noted

Symbol	Parameter	Condition	Min	Max	Units	Notes
V_{IH}	Input high voltage		0.7VSB	VSB	V	
V_{IL}	Input low voltage		0	0.2VSB	V	
I_{IH}	Input high leakage current	V_{IN} = VSB	4	200	µA	Pins TESTI and PP
I_{IL}	Input low leakage current	V_{IN} = 0V	-10	10	µA	Pins TESTI and PP
I_{IH}	Input high leakage current	V_{IN} = VSB	-10	10	µA	Pin TESTBI/ BADD and all GPIO pins
I_{IL}	Input low leakage current	V_{IN} = 0V	-200	-2	µA	Pin TESTBI/ BADD and all GPIO pins
V_{OH}	Output high voltage	I_{OH} = -2µA[1]	0.7VSB		V	Pin TESTBI/ BADD and all GPIO input pins
V_{OH}	Output high voltage	I_{OH} = 0.5mA	0.7VSB		V	All GPIO output pins
V_{OL}	Output low voltage	I_{OL} = 1mA		0.4	V	Pin TESTBI/ BADD and all GPIO pins
V_{PP}	Peak-to-peak voltage	$V_{IL} \geq$ 0V $V_{IH} \leq VSB$ square wave	1.4		V	Pin XTALI (external clock supply)
I_{LEAK}	Leakage current		-10	10	µA	Pin XTALI, pin XTALO when in test mode
f	Clock frequency[2]		32,441	33095	kHz	Pin XTALI
D	Duty cycle[2]	$C_{XTALO} \leq$ 2pF	45	55	%	Pin XTALI

[1] Note that the device has only weak internal pullups. It is recommended that external pullups are used.

[2] Valid if an external single ended clock is used (also refer to **section 1.4**). Note that it is mandatory that the capacitance on pin XTALO is less than 2pF in that case!

Table 12-5 DC Characteristics for LPC Pins

Pins LCLK, LFRAMEn, LAD[3:0], LRESETn, LPCPDn, SERIRQ, CLKRUN#
T_A = 0°C to 70°C, *VDD* = *VSB* = 3.3V +- 0.3V unless otherwise noted
DC characteristics according to 3.3V PCI signaling environment

Symbol	Parameter	Condition	Min	Max	Units
V_{IH}	Input high voltage[1]		0.5*VDD*	*VDD*+0.5	V
V_{IL}	Input low voltage[1]		-0.5	0.3*VDD*	V
I_{IH}	Input high leakage current[1][2]	V_{IN} = VDD	-10	10	µA
I_{IL}	Input low leakage current[1][2]	V_{IN} = 0V	-10	10	µA
V_{OH}	Output high voltage[3]	I_{OH} = -0.5mA	0.9*VDD*		V
V_{OL}	Output low voltage[3]	I_{OL} = 1.5mA		0.1*VDD*	V
C_{IN}	Input capacitance[4]			10	pF

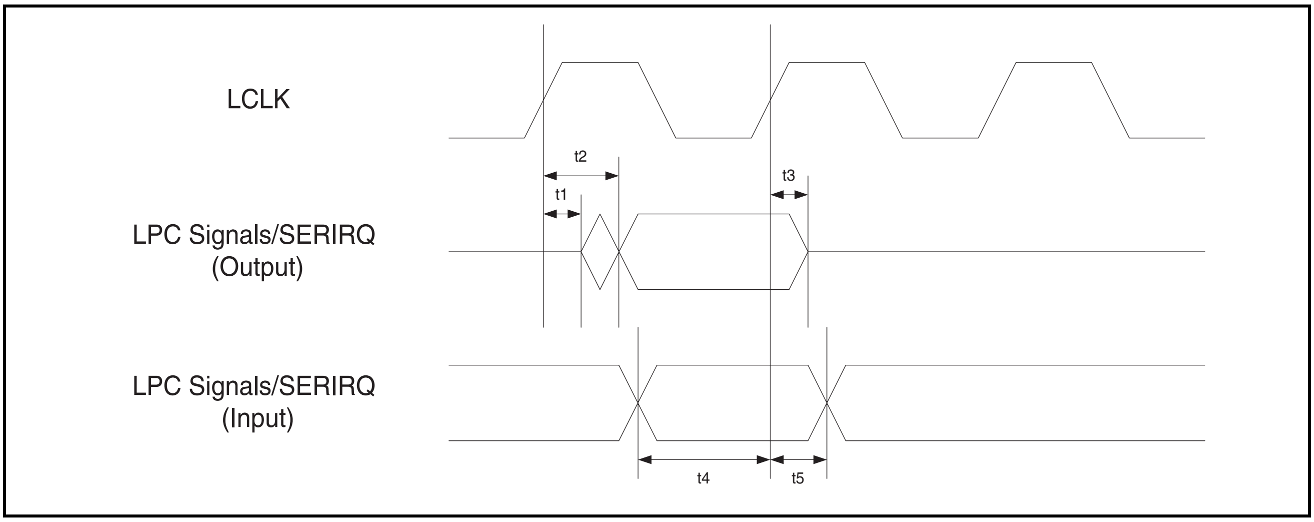
[1] *VDD* = *VSB*
[2] Input leakage currents include Hi-Z output leakage for all bidirectional buffers with tristate outputs
[3] valid for pins LAD[3:0] and SERIRQ
[4] guaranteed by design

13 AC Characteristics

13.1 LPC Signals and SERIRQ Timing

Table 13-1 LPC and SERIRQ Timing

Symbol	Parameter	min	typ	max	unit
	LPC clock frequency (pin LCLK)	32	33	34	MHz
t1	Float to active delay	2		11	ns
t2	Output valid delay	2		11	ns
t3	Active to float delay			28	ns
t4	Input setup time	7			ns
t5	Input hold time	0			ns



13.2 LPC Powerdown and LPC Reset Timing

13.2.1 LPC Powerdown

LCLK must be stopped in low state and LRST# must be asserted (active low) together with LPCPD#. VDD must be switched off. The VDD powered signals LFRAME#, LAD[3:0], SERIRQ and CLKRUN# must be tri-stated (or also driven low) when VDD is off.

The LPC module is reset directly by LRST# and the whole chip except the tick counter is reset at the end of the LPC power down cycle.

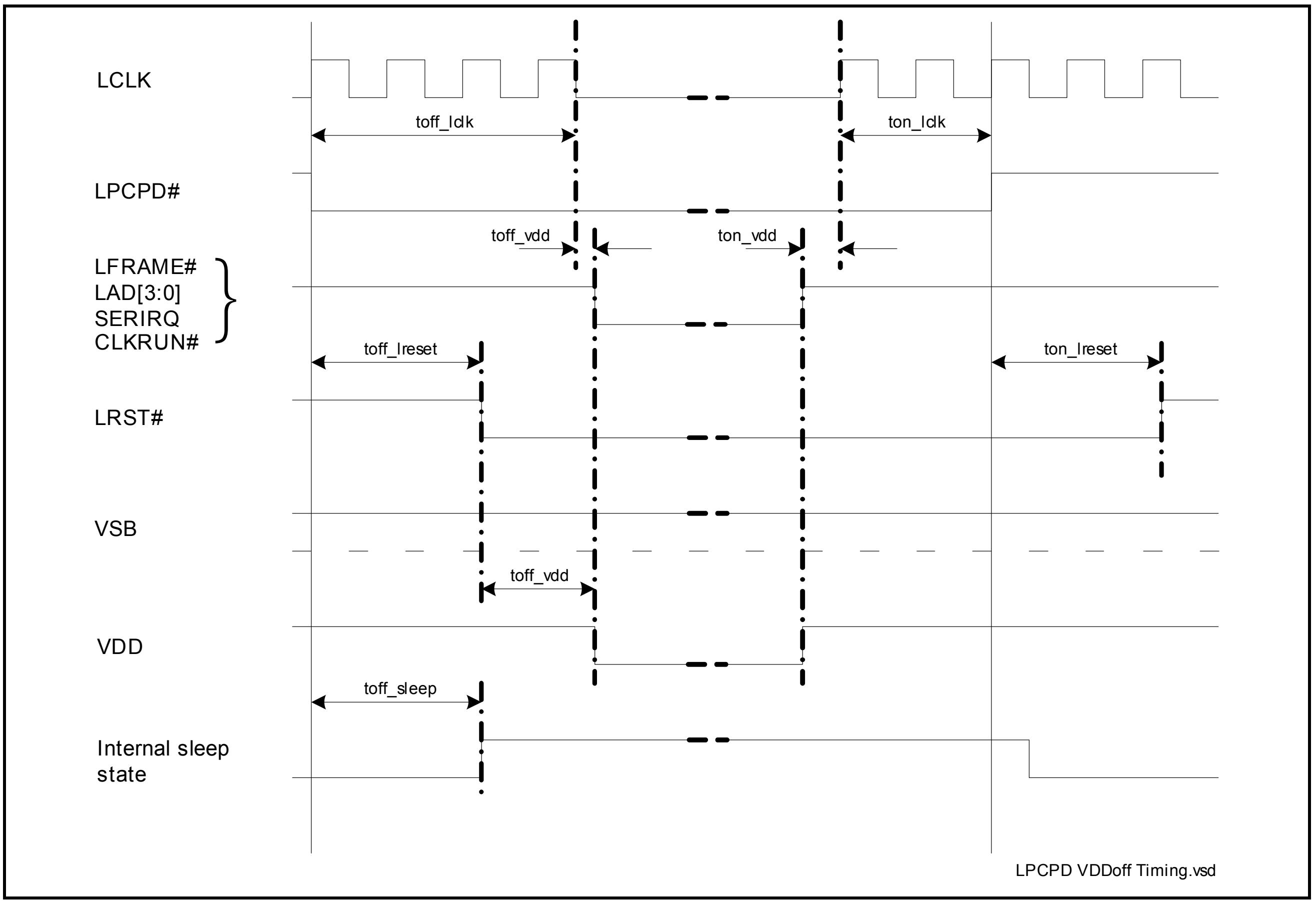


Figure 13-1 LPC Powerdown Timing

Table 13-2 LPCPD Timing Parameters

Parameter	Symbol	min	typ	max	Unit
LPCPD# active to LCLK off (= low)	toff_lclk	30[1]			µs
LCLK on to LPCPD# inactive	ton_lclk	100			µs
LPCPD# active to LRST# active	toff_lreset	30			µs
LPCPD# inactive to LRST# inactive	ton_lreset	32			µs
LRST# active or LCLK off to VDD off	toff_vdd	0			µs
VDD on to LCLK on	ton_vdd	0			µs
LPCPD# active to internal sleep state	toff_sleep	0	40	3500[2]	µs

[1] Must be 3500µs if minimum power consumption on VSB must be reached in any case. See parameter toff_sleep and **section 2.7**!

[2] Maximum value is needed only for certain cases when the TPM firmware currently updates keys in non-volatile memory (this is application-profile dependent and expected to happen with a very low probability). In most cases, the TPM is already in sleep. The TPM stops key generation/update and goes into security reset state if the clock is switched off before sleep state was reached. The current consumption is higher than specified for sleep state then! Normal operation is resumed at LPCPD# going inactive by resetting the whole chip (except tick counter; LRESET# clears LPC interface module).

13.2.2 LPC Reset

A reset request is assumed if LPCPD# and LRST# go active at about the same time. The clock should not be stopped and power should stay on in that case!

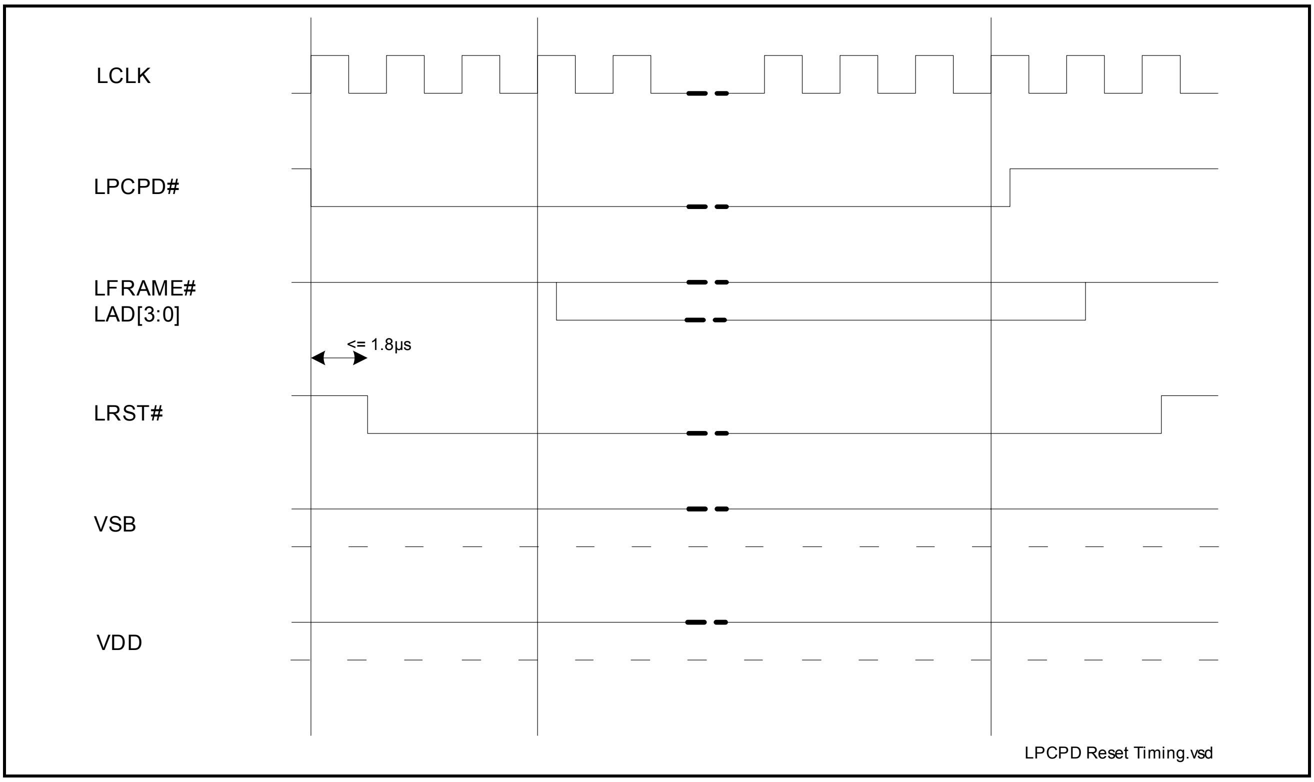


Figure 13-2 LPC Reset Timing

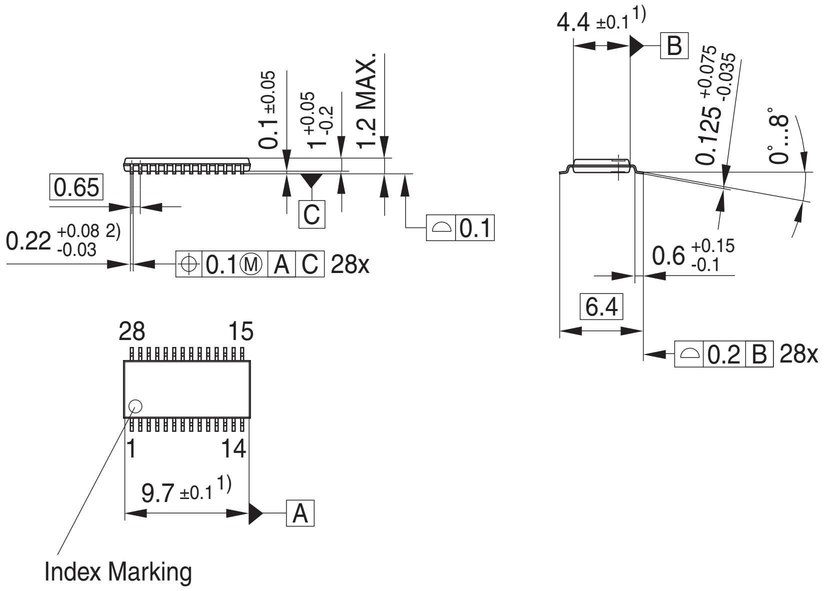
Table 13-3 LPC Reset Timing Parameters

Parameter	Symbol	min	typ	max	Unit
LRST# pulse width	—	1			µs

This page has been left blank intentionally.

14 Package Dimensions

All dimensions are given in millimeters (mm) unless otherwise noted.



1) Does not include plastic or metal protrusion of 0.15 max. per side
2) Does not include dambar protrusion

Figure 14-1 Package Dimensions P-TSSOP-28-1/2

14.1 Packing Type

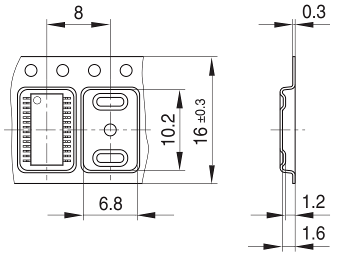


Figure 14-2 Tape & Reel Dimensions P-TSSOP-28-1/2

Tape & Reel (reel diameter 330 mm), 3000 pcs.

14.2 Recommended Footprint

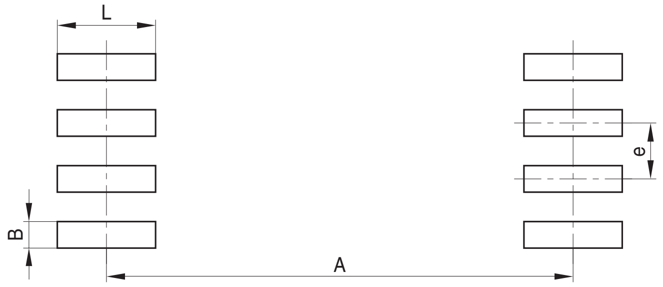


Figure 14-3 Recommended Footprint P-TSSOP-28-1/2

e	0.65 mm	25.6 mil
A	6.10 mm	240 mil
L	1.30 mm	51 mil
B	0.40 mm	16 mil
Controlling dimension is mm		

14.3 Chip Marking

Figure 14-4 Chip Marking

Line 1: SLB9635TT12

Line 2: G <datecode> KMC => K indicates assembly site code, MC indicates mold compound code

Line 3: 00 <Lot number>

The 00 is a internal FW indication (only at the time of manufacturing due to field upgrade option)

15 References

[1] —, “Low Pin Count (LPC) Interface Specification”, Version 1.1, Intel

[2] —, “Plug and Play ISA Specification”, Revision 1.0a, Microsoft

[3] —, “Serial IRQ Specification for PCI Systems”, Version 6.0, Intel

[4] —, “Peripheral Connect Interface (PCI) Specification”, Rev. 2.2

[5] —, “TCG PC Client TPM Interface Specification”, Version 1.2, 2005-05-19, TCG

[6] —, “TPM Main Specification”, Version 1.2, Rev. 87, 2005-05-24, TCG

[7] —, “PC Client Implementation Specification”, Version 1.2, 2005-05-31, TCG

[8] —, “TPM Dictionary Defense Logic”, Version 1.00, 2005-10-20, IFX

[9] —, “TPM BIOS Porting Design Guide”, Version 2.00.0000, 2005-10-12, IFX

[10]—, “TCG Software Stack Specification (TSS)”, Version 1.2, 2005-11-02, TCG

This page has been left blank intentionally.

http://www.infineon.com

Published by Infineon Technologies AG