TCG TPM
I2C Interface Specification

Family "2.0"
Level 00 Revision 1.00
October 7, 2016

Contact: admin@trustedcomputinggroup.org

TCG

TCG Published

Disclaimers, Notices, and License Terms

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

Without limitation, TCG disclaims all liability, including liability for infringement of any proprietary rights, relating to use of information in this specification and to the implementation of this specification, and TCG disclaims all liability for cost of procurement of substitute goods or services, lost profits, loss of use, loss of data or any incidental, consequential, direct, indirect, or special damages, whether under contract, tort, warranty or otherwise, arising in any way out of use or reliance upon this specification or any information herein.

Contact the Trusted Computing Group at www.trustedcomputinggroup.org for information on specification licensing through membership agreements.

Any marks and brands contained herein are the property of their respective owners.

Table of Contents

List of Figures

List of Tables

1. Scope

The Trusted Computing Group TPM I2C Interface Specification is an industry specification that defines an I2C Interface for TPM 2.0.

As this specification defines only the interface for the I2C-TPM a suitable platform specification must be considered additionally to allow the design of a platform specific I2C-TPM. It is expected that the reader of this specification is familiar with the PTP [2].

This specification is intended for a single TPM on an I2C bus. It does not address multiple TPMs on the same I2C bus.

1.1 Key words

The key words "MUST," "MUST NOT," "REQUIRED," "SHALL," "SHALL NOT," "SHOULD," "SHOULD NOT," "RECOMMENDED," "MAY," and "OPTIONAL" in this document normative statements are to be interpreted as described in RFC-2119, *Key words for use in RFCs to Indicate Requirement Levels.*

1.2 Statement Type

Please note a very important distinction between different sections of text throughout this document. You will encounter two distinctive kinds of text: informative comment and normative statements. Because most of the text in this specification will be of the kind normative statements, the authors have informally defined it as the default and, as such, have specifically called out text of the kind informative comment. They have done this by flagging the beginning and end of each informative comment and highlighting its text in gray. This means that unless text is specifically marked as of the kind informative comment, you can consider it of the kind normative statements.

For example:

Start of informative comment

This is the first paragraph of 1–n paragraphs containing text of the kind *informative comment* ...

This is the second paragraph of text of the kind *informative comment* ...

This is the nth paragraph of text of the kind *informative comment* ...

To understand the TCG specification the user must read the specification. (This use of MUST does not require any action).

End of informative comment

This is the first paragraph of one or more paragraphs (and/or sections) containing the text of the kind normative statements ...

To understand the TCG specification the user MUST read the specification. (This use of MUST indicates a keyword usage and requires an action).

2. Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

[1] I2C-Bus specification and user manual, Rev. 6, 2014-04-04, NXP

[2] TCG PC Client Platform TPM Profile (PTP) Specification, Family 2.0, Level 00, Rev. 43, January 26, 2015, TCG

[3] Trusted Platform Module Library (Part 1 – 4), Family 2.0, Level 00, Rev. 01.16 or later

[4] http://reveng.sourceforge.net/

3. Acronyms and Abbreviations

Acronym / Abbreviation	**Description**
ACK	Acknowledge
CCITT	International Telegraph and Telephone Consultative Committee
CRC	Cyclic Redundancy Check
I2C	Inter-Integrated Circuit
LSB	Least Significant Byte
MSB	Most Significant Byte
NACK	Not Acknowledge
PTP	Platform TPM Profile
SCL	Serial Clock Line
SDA	Serial Data Line
TCG	Trusted Computing Group
TPM	Trusted Platform Module

4. TPM I2C Interface Requirements

4.1 Introduction

Start of informative comment

To allow usage of a discrete TPM in platforms which have only an I2C interface available it has been seen necessary to specify such an interface including a corresponding communication protocol as an extension to the existing PC Client Platform TPM Profile Specification.

End of informative comment

4.2 Requirements

4.2.1 Bus speed

Start of informative comment

The speed of the data transfers is influenced by the participating components as well as by the clock frequency used. I2C provides options for several bus speeds, standard mode (Sm, up to 100 kbit/s), fast mode (Fm, up to 400 kbit/s), fast mode plus (Fm+, up to 1 Mbit/s), high speed mode (Hs-mode, up to 3.4 Mbit/s) and ultra-fast mode (UFm, up to 5 Mbit/s). To allow a wide usage without enhanced HW features, standard mode and fast mode are a good choice with suitable data transfer rates. And as fast mode devices can communicate with standard mode devices fast mode has been chosen.

End of informative comment

1. An I2C-TPM compliant to this specification SHALL be able to operate at Fast mode (Fm).
2. Higher speeds are allowed and SHALL be indicated via the Interface Capability register (see section 6.5.10 of this document).

4.2.2 I2C Device address

Start of informative comment

Each device on the I2C bus needs a unique slave address. I2C offers 2 options, 7-bit slave address is mandatory and 10-bit slave address is optional. Because 10-bit addressing requires 2 bytes overhead 7-bit addressing (1 byte overhead) has been chosen to keep the overhead small.

The default 7-bit I2C device address is 0x2E, the 8th bit indicates the data direction. Therefore the first byte after the START condition will be 0x5D for an I2C read request and 0x5C for an I2C write transmission.

End of informative comment

1. An I2C-TPM compliant to this specification SHALL support one 7-bit I2C device address.
2. Default address is 0x2E.
3. An I2C-TPM compliant to this specification MAY support reconfiguration of the I2C device address.
   a. If supported the reconfiguration SHOULD follow the mechanism defined in section 6.5.15 of this document.
   b. An I2C-TPM MAY implement a vendor defined mechanism for the reconfiguration of the I2C device address.

4.2.3 Fast turnaround

Start of informative comment

I2C offers 2 options for write / read cycles, one is to have 2 separate frames for write and read, and the other is to combine 2 frames using the repeated start condition. The 2nd option allows a slightly higher throughput. Additionally, the repeated start mechanism avoids an allocation of the bus by any other device.

End of informative comment

1. An I2C-TPM compliant to this specification SHOULD support the repeated start condition (Sr) for I2C read after I2C write.

4.2.4 Data rate synchronization

Start of informative comment

I2C is a synchronous bus and the operating speed of the I2C-TPM may sometimes require that the communication speed of the master is throttled down so that the I2C-TPM has sufficient time to store data or to provide correct response data. To allow the synchronization between the I2C-TPM and the bus master it is required that the bus master supports the clock stretching mechanism.

End of informative comment

1. If an I2C-TPM needs to synchronize the data rate on the bus it SHALL use clock stretching.

4.2.5 Supply voltage

Start of informative comment

Besides the supply voltages required by this specification the I2C-TPM may also support other supply voltages.

End of informative comment

1. The I2C-TPM SHALL support a supply and I/O voltage of 1.8V or 3.3V.
2. The I2C-TPM MAY support supply and I/O voltages of both 1.8 and 3.3V.
3. The I2C-TPM MAY support other supply and I/O voltages.

4.2.6 Pull-up resistors

Start of informative comment

I2C needs pull-up resistors to allow the implementation of the wired-AND function. As I2C is critical with respect to the rise time of the bus signals the configuration of the pull-up resistors depends mainly on the bus length and on the number of devices connected to the bus. To allow sufficient flexibility regarding the configuration of the pull-up resistors external resistors will be implemented on the platform.

End of informative comment

1. An I2C-TPM compliant to this specification SHALL NOT have internal pull-up resistors on the SDA and SCL pins.

4.2.7 Host interrupt

Start of informative comment

An I2C-TPM may need a considerable time to process certain requests especially for commands with cryptographic operations. Polling the I2C-TPM during command processing for response availability would create a certain bus load on the one hand and would also create a high system load. Therefore it would be the better choice that the I2C-TPM informs the host once the command processing has been finished and the response is available. Furthermore, for some other events the host may desire to get notified about such events via an interrupt mechanism rather than polling for such an event.

End of informative comment

1. An I2C-TPM compliant to this specification SHALL support a host interrupt (PIRQ#) for signaling of certain events (e.g. response availability).

4.2.8 Availability after reset

Start of informative comment

At power-on after reset the I2C-TPM needs some time to perform the initialization of the device and the initial self-test. Therefore it is necessary to allow the I2C-TPM a certain amount of time to perform such operations before it is able to handle communication requests.

End of informative comment

1. An I2C-TPM compliant to this specification SHALL be available for communication within 30ms after de-assertion of reset (aka TPM_Init).

4.2.9 Locality support

Start of informative comment

With TPM 1.2 the so-called Locality concept has been introduced as hardware based authorization. As TPM 2.0 continues to use this concept the support for Locality is also included in the TPM I2C Interface. To allow simplified implementations it is possible to implement a I2C-TPM with only one locality.

The indication of which localities are supported is done via the Interface Capability register (see section 6.2.10 of this document).

End of informative comment

1. An I2C-TPM compliant to this specification SHALL support one of the following options
   - one locality (locality 0) or
   - 5 localities (locality 0 – locality 4) or
   - All localities supported by this specification (locality 0 – locality 255).

4.2.10 GUARD_TIME

Start of informative comment

GUARD_TIME is the minimum lapse of time at the I2C master measured from the I2C STOP condition until the next I2C START condition. GUARD_TIME might be required by some I2C TPM implementations to recover between 2 separate access cycles.

The Interface Capability register (see section 6.5.10 of this document) indicates whether an I2C-TPM needs the GUARD_TIME and for which condition (write after read, write after write, read after write or read after read).

The default value for GUARD_TIME for all 4 conditions is 250 µs. This value has to be considered until the actual value indicated by the I2C-TPM has been read from the corresponding fields in the Interface Capability register. Additional, before the actual value for GUARD_TIME_Sr has been read from the Interface Capability register the bus master has to use the default value of 250 µs for GUARD_TIME_Sr.

End of informative comment

5. Communication Protocol Fundamentals

The data transmission between the I2C-TPM and the HOST is done through the I2C interface of the I2C-TPM. Additionally, the I2C-TPM offers an indication that response data is available through a dedicated pin (PIRQ#). This low-active signal can also be used as an interrupt signal for other events as defined in section 6.5.5 of this document.

The communication flow between the HOST and the I2C-TPM is a strong dialog. That means the HOST has to wait after a request for the corresponding response from the I2C-TPM before sending a new request.

5.1 Layer Model

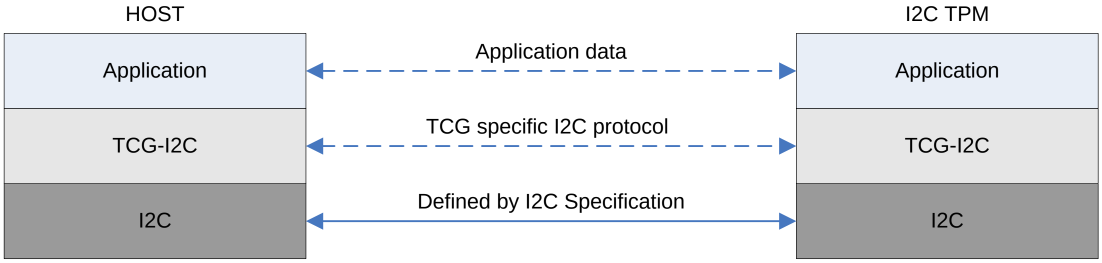


Figure 1 — Layer Model

5.2 Physical Layer I2C

The standardized physical layer is entirely defined in the I2C specification ([1]). Only a subset of those definitions is used for this protocol. See section 4.2 of this document for details.

5.2.1 I2C Protocol Usage Scenarios

5.2.1.1 Regular Register write

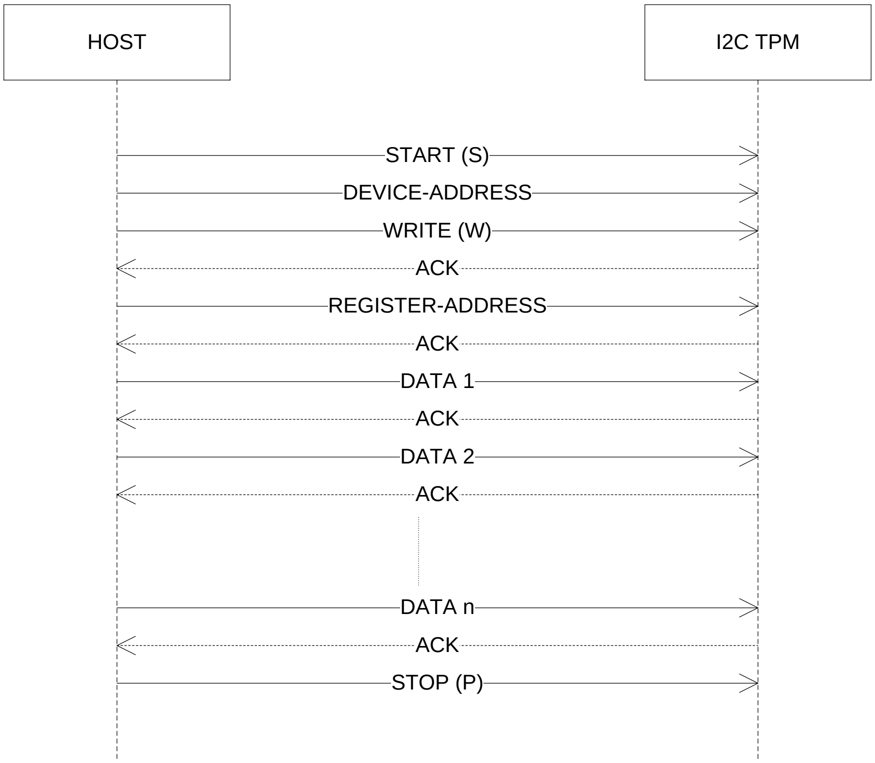


Figure 2 — Register write sequence on the I2C layer

5.2.1.2 Register write with address NACK

Start of informative comment

An address NACK is returned by the I2C-TPM when either an invalid I2C device address is used by the bus master or the I2C-TPM is currently not able to respond to the current bus cycle because of internal reasons. It is a good practice to repeat the current cycle using the correct I2C device address.

End of informative comment

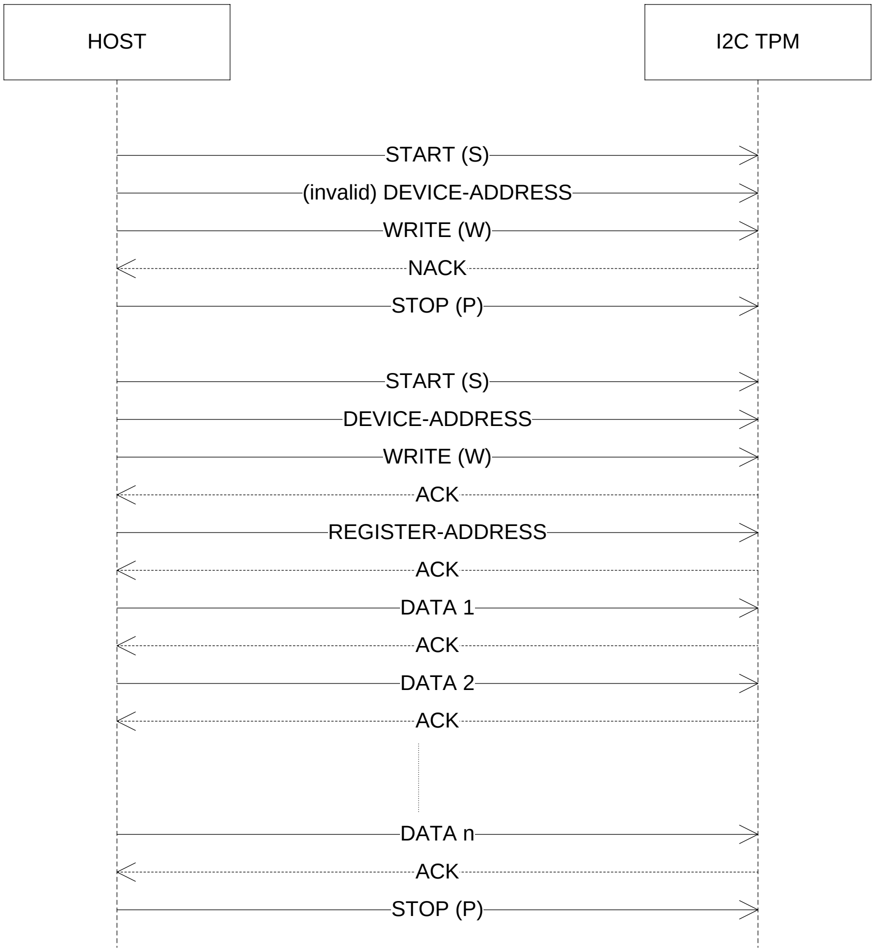


Figure 3 — Register write sequence with address NACK on the I2C layer

5.2.1.3 Register write with data NACK

Start of informative comment

A data NACK is returned by the I2C-TPM when it is not able to receive any more data bytes. It is a good practice to repeat the current cycle entirely if this happens.

End of informative comment

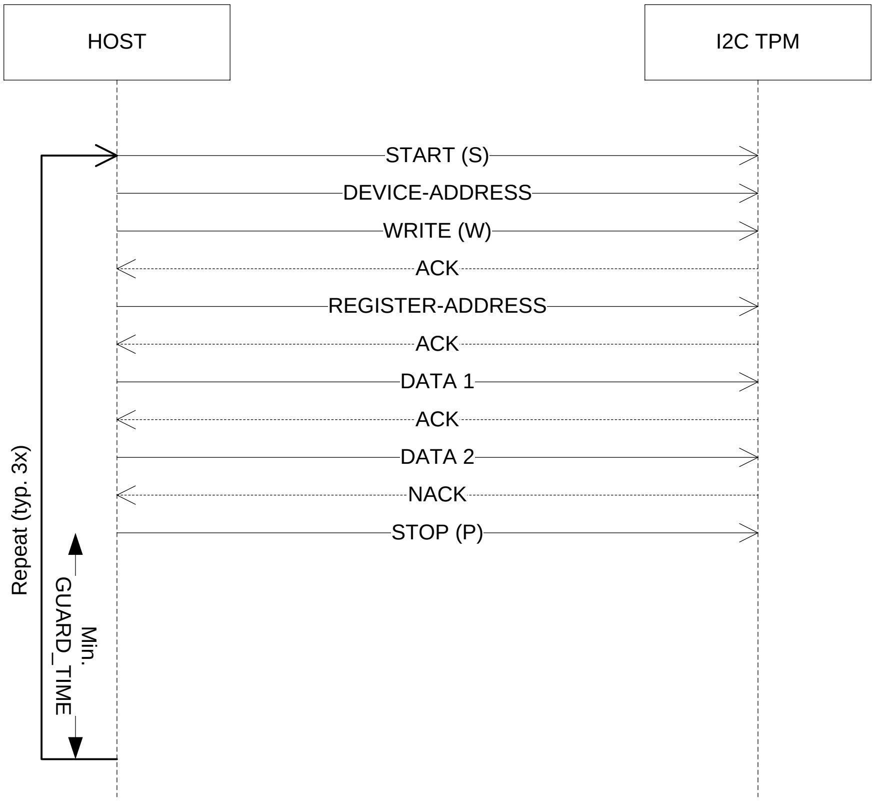


Figure 4 — Register write sequence with data NACK on the I2C layer

1. If the I2C-TPM must return a NACK during the reception of data it SHALL discard the data already written during this cycle.

5.2.1.4 Regular Register read

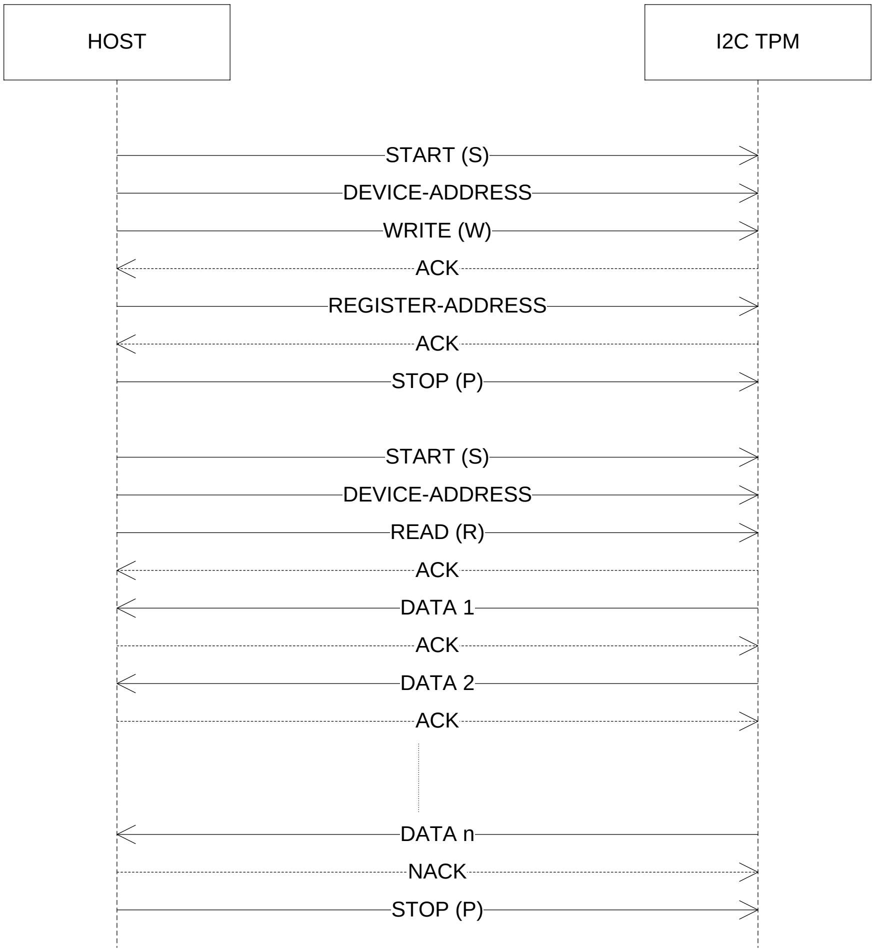


Figure 5 — Register read sequence on the I2C layer

5.2.1.5 Register read with repeated START

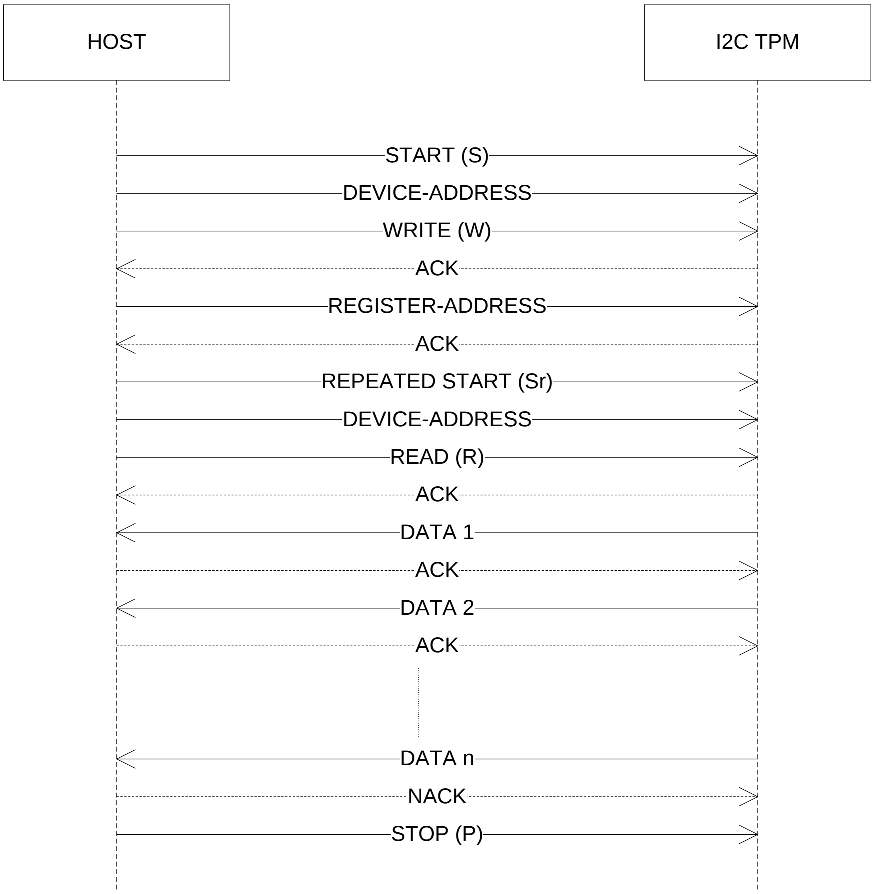


Figure 6 — Register read sequence on the I2C layer using repeated START (Sr)

5.2.1.6 Register read with GUARD_TIME

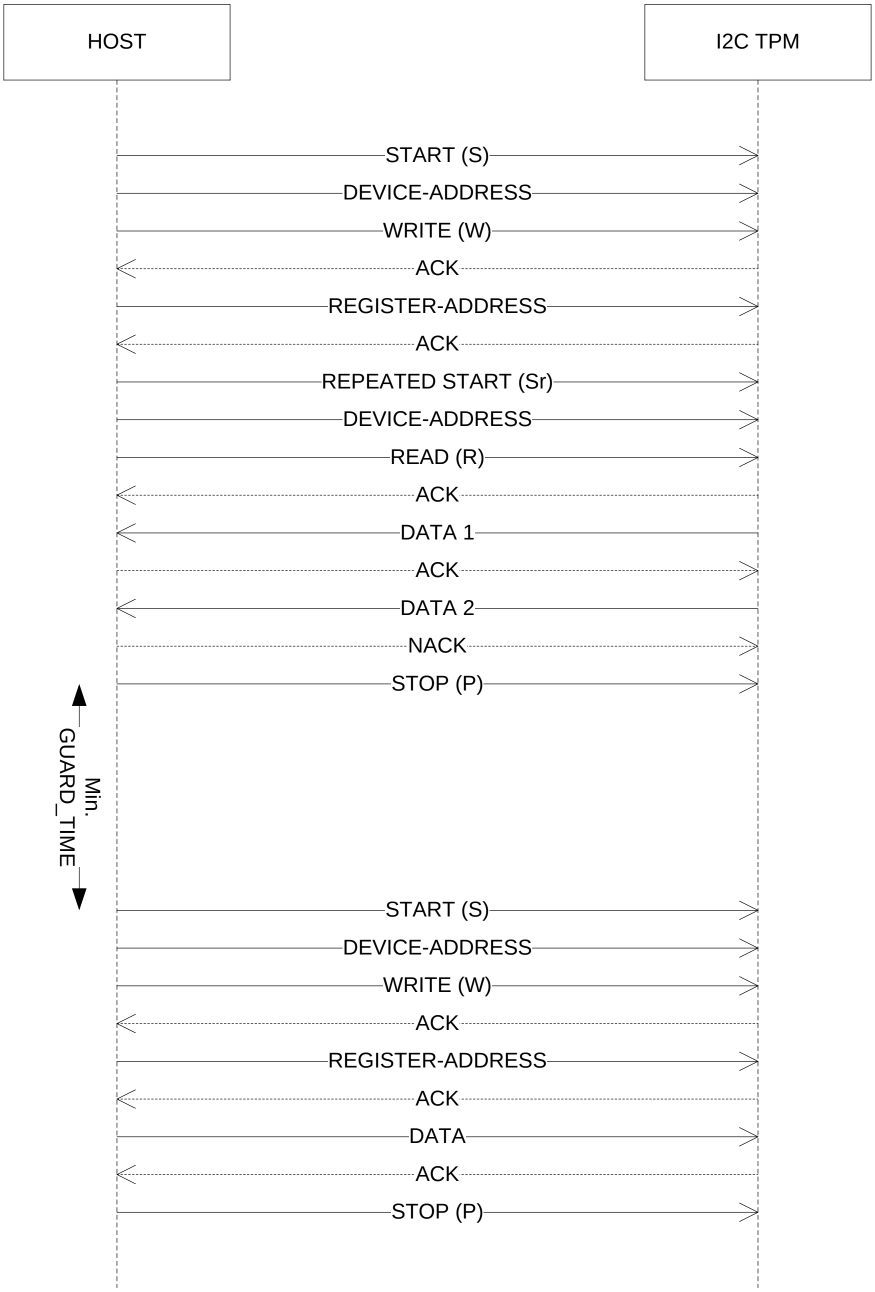


Figure 7 — Register read sequence with GUARD_TIME write after read on the I2C layer

6. Physical Layer TCG-I2C

The physical layer TCG-I2C is used to establish several sub-addresses below a single I2C device address as defined by the I2C specification [1]. These sub-addresses are defined as shown in the following sections of this document. The I2C slave uses different address locations for status, control and data communication registers.

6.1 Byte Ordering

Start of informative comment

Multi-byte numeric values are stored and sent via the bus in little-endian order; for example, the first byte of a two-byte register is the LSB of the stored value while the second byte is its MSB. For example, the master wants to read from a 4-byte register at 0x08 containing data0 at 0x08, data1 at 0x09, data2 at 0x0A and data3 at 0x0B the byte order on the bus looks as following:

Register-Address:								0x08	0x09	0x0A	0x0B	
I2C:	S	DEVICE_ ADDRESS	W	REGISTER_ ADDRESS	Sr	DEVICE_ ADDRESS	R	Data0	Data1	Data2	Data3	P

End of informative comment

6.2 Overruns

1. The I2C-TPM SHALL return a value of 0xFF in the following cases
   - On a read from an invalid register
   - On a read beyond the end of a register
2. If the master writes beyond the end of a register:
   a. The I2C-TPM SHALL update the register designated by the start address.
   b. The I2C-TPM MAY update additional, adjacent registers.

6.3 Handling of Multi-Byte Registers

Start of informative comment

The I2C-TPM has various multi-byte registers (e.g. TPM_INT_ENABLE). Such registers are defined by a base address and a length. In all cases, except the TPM_STS register, the access to such multi-byte registers is only possible from the register base address. E.g. a write to the TPM_INT_ENABLE starts from base address 0x08 and consists of 4 bytes. A read from the same register also starts at the base address but may consist of 1 to 4 bytes (e.g. a read from 0x08 with only 1 byte would return bits 0 to 7 while a read from 0x09 may return 0xFF).

There is one exception concerning TPM_STS which is divided into 3 parts. The 1st part with one byte at 0x18 (bits 0 to 7) may be written as a single byte. The 2nd part with two bytes at 0x19 is read only but may be written without effect (the write is ignored by the I2C-TPM). The 3rd part with one byte at 0x1B (bits 24 to 31) may also be written as a single byte. Consequently, a write to 0x18 may consist of 1 to 4 bytes. The same applies for a read, e.g. a read from 0x18 with 1 byte returns bits 0 to 7, a read from 0x18 with 4 bytes returns the entire TPM_STS register and a read from 0x1B with one byte returns bits 24 to 31.

End of informative comment

1. The I2C-TPM SHALL accept a write to a register base address with a data length = length of the addressed register.
2. The I2C-TPM SHALL accept a read from a register base address and return the corresponding values under consideration of the rules in Table 11.
3. The I2C-TPM SHALL accept a single byte write to the addresses 0x18 or 0x1B (TPM_STS) and change the corresponding value.
4. The I2C-TPM SHALL accept a single byte read from the address 0x1B (TPM_STS) and return the corresponding value.
5. The I2C-TPM SHALL accept a 2-byte read from address 0x19 (TPM_STS) and return the corresponding values.
6. The behavior of the I2C-TPM for any other access is vendor specific.

6.4 I2C-TPM Localities

Start of informative comment

The I2C-TPM supports all localities as defined in the PTP [2] including the extended localities as defined in the TPM specification [3]. Which locality currently accesses the I2C-TPM can be determined from the value in the locality selection register. Locality priority determines which locality is preferred in cases where two or more localities request the I2C-TPM simultaneously. Table 1 shows the locality priorities with 1 as the highest priority and 6 as the lowest.

End of informative comment

Table 1 — TPM Locality Selection Register

Locality Priority	Locality Selection Register value	Locality	Value of locality modifier (see the PTP [2])	Mandatory (M) Optional (O)
5	0x00	0	0000 0001b	M
4	0x01	1	0000 0010b	O
3	0x02	2	0000 0100b	O
2	0x03	3	0000 1000b	O
1	0x04	4	0001 0000b	O
6	0x05 – 0x31	Reserved for vendor use	n.a.	n.a.
	0x32 – 0xFF	32 - 255	0010 0000b – 1111 1111b	O

6.5 I2C-TPM Registers

Start of informative comment

The I2C-TPM registers are used to map the TCG defined TPM Interface for TPM 2.0 (see the PTP [2]) to TPM 2.0 implementations using I2C as the Host interface. Those registers are established as sub-addresses below the single I2C device address as defined by the I2C specification [1].

The following registers are needed for the operation of a TPM at I2C with locality support. For a detailed description of registers (contents, and endianness of multi-byte registers) see the PTP [2].

Table 2 lists all registers of the I2C-TPM. The TPM_ACCESS register has multiple, separate and unique instances, one per locality priority level (see Table 1). All other registers alias to a single register with the locality used to determine whether accesses are permitted

End of informative comment

Table 2 — I2C-TPM Register overview

Address	Name	Length	Description	Master Access
TPM specific registers				
0x00	TPM_LOC_SEL	1	Selection of the locality of the current access	Read / Write
0x01 – 0x03	Reserved	n.a.	Reads return 0xFF	n.a.
0x04	TPM_ACCESS	1	Used to gain ownership of the TPM for this particular locality	Read / Write
0x05 – 0x07	Reserved	n.a.	Reads return 0xFF	n.a.
0x08	TPM_INT_ ENABLE	4	Enables specific interrupts and has the global enable	Read / Write
0x0C – 0x0F	Reserved	n.a.	Reads return 0xFF	n.a.
0x10	TPM_INT_ STATUS	4	Shows which interrupt has occurred	Read / Write
0x14	TPM_INT_ CAPABILITY	4	Provides information about which interrupts this particular TPM supports	Read only
0x18 – 0x1B	TPM_STS	4	Contains general status details	Read / Write
0x1C – 0x1F	Reserved	n.a.	Reads return 0xFF	n.a.
0x20	TPM_HASH_END	1	This signals the end of the hash operation. Only available when locality 4 is selected	Write only
0x21 – 0x23	Reserved	n.a.	Reads return 0xFF	n.a.
0x24	TPM_DATA_FIFO	TPM_STS (burst-count)	Buffer to exchange the data for commands and responses with the HOST. For locality4 this is also aliased to TPM_HASH_DATA	Read / Write
0x25 – 0x27	Reserved	n.a.	Reads return 0xFF	n.a.
0x28	TPM_HASH_START	1	This signals the start of the hash operation. Only available when locality 4 is selected	Write only
0x29 – 0x2F	Reserved	n.a.	Reads return 0xFF	n.a.
0x30	TPM_I2C_ INTERFACE_ CAPABILITY	4	I2C Interface Capability Register	Read only
0x34 – 0x37	Reserved	n.a.	Reads return 0xFF	n.a.
0x38	TPM_I2C_DEVICE_ ADDRESS	2	This register allows changing the I2C device address	Write only
0x3A – 0x3F	Reserved	n.a.	Reads return 0xFF	n.a.

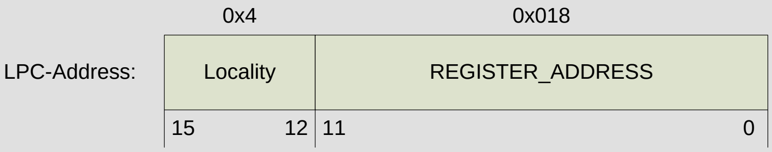
Address	**Name**	**Length**	**Description**	**Master Access**
0x40	TPM_DATA_ CSUM_ENABLE	1	Enables the data checksum calculation and indication via the TPM_DATA_CSUM register	Read / Write
0x41 – 0x43	Reserved	n.a.	Reads return 0xFF	n.a.
0x44	TPM_DATA_CSUM	2	Contains the data checksum when enabled via DATA_CSUM_ENABLE	Read only
0x46 – 0x47	Reserved	n.a.	Reads return 0xFF	n.a.
0x48	TPM_DID_VID	4	Vendor and device ID VID (bits 15:0) DID (bits 31:16)	Read only
0x4C	TPM_RID	1	Revision ID	Read only
0x4D – 0xFF	Reserved	n.a.	Reads return 0xFF	n.a.

6.5.1 TPM_LOC_SEL

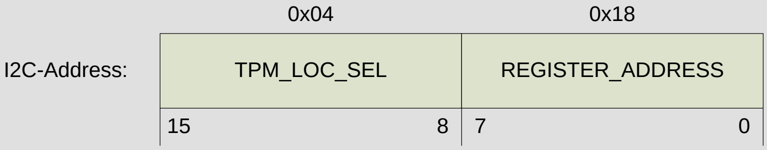
Start of informative comment

This register is used to select the locality which accesses the I2C-TPM. While this register is just the indication of the locality which performs the current communication with the I2C TPM interface it is still necessary to execute the process of getting access to the TPM via the TPM_ACCESS register to become the active locality. Or, in other words, the value in TPM_LOC_SEL may be considered to be the upper nibble of the LPC address.

For example, if locality4 wants to access the register address 0x18 the LPC address would have the value 0x4018:



The same access on the I2C TPM effectively looks as follows:



End of informative comment

Table 3 — TPM Locality Selection Register

Abbreviation:			TPM_LOC_SEL
General Description:			Indication of the accessing Locality
Bit Descriptions:			
7:0	Read/ Write	LocalitySelection	This register is sticky. Read returns the currently accessing locality Write sets the new accessing locality See Table 1 for allowed values, default 0x00

1. The I2C TPM SHALL maintain the value in this register until it gets written with a new value.

6.5.2 TPM_ACCESS

See the PTP [2] section 5.5.2.4 for a detailed description.

6.5.3 TPM_INT_ENABLE

Table 4 — Interrupt Enable

Abbreviation:			TPM_INT_ENABLE
General Description:			Enables specific interrupts and has the global enable. The TPM SHALL implement this register.
Bit Descriptions:			
31	Read/ Write	globalIntEnable	1 = Interrupts controlled by individual bits 0= All interrupts disabled (default) cleared to 0 on reset.
30:8		Reserved	Reads always return 0
7	Read/ Write	commandReadyEnable	1 = Enabled 0 = Disabled (default)
6:3		Reserved	Reads always return 0
2	Read/ Write	localityChangeIntEnable	1 = Enabled 0 = Disabled (default)
1	Read/ Write	stsValidIntEnable	1 = Enabled 0 = Disabled (default)
0	Read/ Write	dataAvailIntEnable	1 = Enabled 0 = Disabled (default)

6.5.4 TPM_INT_STATUS

See the PTP [2] section 5.6.1.2 for a detailed description.

6.5.5 TPM_INT_CAPABILITY

Table 5 — Interrupt Capability

Abbreviation:			TPM_INT_CAPABILITY
General Description:			Provides information about which interrupts this particular TPM supports. The TPM SHALL implement this register.
Bit Descriptions:			
31:8	Read Only	Reserved	Reads always return 0
7	Read Only	commandReadyInt Support	Corresponds to TPM_INT_ENABLE.commandReadyEnable 1 = supported 0 = not supported
6:3	Read Only	Reserved	Reads always return 0
2	Read Only	LocalityChangeIntSupport	Corresponds to TPM_INT_ENABLE.localityChangeIntEnable. 1 = supported 0 = not supported
1	Read Only	stsValidIntSupport	Corresponds to TPM_INT_ENABLE.stsValidIntEnable 1 = supported 0 = not supported
0	Read Only	dataAvailIntSupport	Corresponds to TPM_INT_ENABLE.dataAvailIntEnable. This is a mandatory interrupt. 1 = supported 0 = not allowed

6.5.6 TPM_STS

Start of informative comment

See the PTP [2] section 5.5.2.8 for a description of the state transition behavior.

Reading of the burstCount may be critical when read in single bytes because the low part might change when the high part is read (and vice versa). Therefore it is strongly recommended to read the whole burstCount in one cycle.

End of informative comment

Table 6 — Status Register

Abbreviation:			TPM_STS
General Description:			Contains general status details
Bit Descriptions:			
31:26	Read Only	Reserved	Reads always return 0.
25	Write Only	resetEstablishmentBit	Reads always return 0. Writes (0): Ignored. Writes (1): Reset TPM_ACCESS.tpmEstablished bit if the write occurs from Locality 3 or 4.
24	Write Only	commandCancel	Reads always return 0. A write of a 1 to this field after tpmGo and before dataAvail aborts the currently executing command, resulting in a response of TPM_RC_CANCELLED. A write of 1 to this field after dataAvail and before tpmGo is ignored by the TPM. Writes of 0 are ignored.
23:8	Read Only	burstCount	Indicates the number of bytes that the TPM can return on reads or accept on writes.
7	Read Only	stsValid	This field indicates that TPM_STS.dataAvail and TPM_STS.Expect contain a valid value.
6	Read/ Write	commandReady	Read of 1 indicates TPM is ready, Write of 1 causes TPM to transition its state.
5	Write Only	tpmGo	After software has written a command to the TPM and sees that it was received correctly, software SHALL write a 1 to this field to cause the TPM to execute that command.
4	Read Only	dataAvail	This field indicates that the TPM has data available as a response. When set to 1, software MAY read the ReadFIFO. The TPM SHALL clear the field to 0 when it has returned all the data for the response. Valid indicator: TPM_STS.stsValid = 1
3	Read Only	Expect	The TPM sets this field to a value of 1 when it expects another byte of data for a command. It clears this field to a value of 0 when it has received all the data it expects for that command, based on the TPM size field within the packet. Valid indicator: TPM_STS.stsValid = 1

Abbreviation:			TPM_STS
General Description:			Contains general status details
Bit Descriptions:			
2	Read Only	selfTestDone	This field indicates that the TPM has completed all self-test actions following a TPM2_SelfTest command. Read of 0 indicates self-test is not complete. Read of 1 indicates self-test is complete
1	Write Only	responseRetry	Software writes a 1 to this field to force the TPM to re-send the response. Reads always return 0.
0	Read Only	Reserved	Reads always return 0.

6.5.7 TPM_HASH_END

See the PTP [2] section 4.2 for a detailed description.

6.5.8 TPM_DATA_FIFO

See the PTP [2] section 5.5.2.6 for a detailed description.

6.5.9 TPM_HASH_START

See the PTP [2] section 4.2 for a detailed description.

6.5.10 TPM_I2C_INTERFACE_CAPABILITY

Table 7 — I2C Interface Capability Register

Abbreviation:			TPM_I2C_INTERFACE_CAPABILITY
General Description:			This register provides miscellaneous information about the interface capabilities of the I2C-TPM.
Bit Descriptions:			
31	Read Only	Reserved	Reads always return 0
30	Read Only	GUARD_TIME_Sr	Indicates whether the I2C-TPM needs a GUARD_TIME for repeated START conditions in addition to the conditions defined in Bits 20:17 of this register. 1 – GUARD_TIME needed between Last ACK/NACK to I2C repeated START 0 – No GUARD_TIME needed
29	Read Only	BurstCountStatic	Indicates whether the TPM_STS.burstCount field is dynamic or static 1 = TPM_STS.burstCount is static 0 = TPM_STS.burstCount is dynamic

Abbreviation:			TPM_I2C_INTERFACE_CAPABILITY
General Description:			This register provides miscellaneous information about the interface capabilities of the I2C-TPM.
Bit Descriptions:			
I2C Device Address Change Capabilities			
28:27	Read Only	DevAdrChange	00 – Changing the I2C Device Address is not supported 01 – Changing the I2C Device Address is supported using a vendor defined mechanism 10 – Reserved (not allowed) 11 – Changing the I2C Device Address is supported using the TCG defined mechanism (see 6.5.15)
Locality support Capabilities			
26:25	Read Only	CapLocality	00 – This I2C TPM supports Locality 0 only. 01 – This I2C TPM supports 5 localities (0 – 4). 10 – This I2C TPM supports all localities (0 – 255). 11 – Reserved (not allowed)
I2C Bus Speed Capabilities			
24	Read Only	HsModeSupport	1 - Support for I2C High-Speed Mode (Hs-mode) 0 – I2C High-Speed Mode not supported
23	Read Only	FmPlusSupport	1 - Support for I2C Fast Mode Plus (Fm+) 0 – I2C Fast Mode Plus not supported
22	Read Only	FmSupport	1 - Support for I2C Fast Mode (Fm, mandatory) 0 – Not allowed
21	Read Only	SmSupport	1 - Support for I2C Standard Mode (Sm, mandatory) 0 – Not allowed
GUARD_TIME Capabilities, if indicated the I2C-TPM requires a GUARD_TIME for the given condition Note: Please refer also to bit 30 in this register			
20	Read Only	Read_Read	1 – GUARD_TIME needed between 2 subsequent I2C read operations (I2C STOP to I2C START) 0 – No GUARD_TIME needed
19	Read Only	Read_Write	1 - GUARD_TIME needed between a I2C read operation and the following I2C write operation (I2C STOP to I2C START) 0 – No GUARD_TIME needed
18	Read Only	Write_Read	1 - GUARD_TIME needed between a I2C write operation and the following I2C read operation (I2C STOP to I2C START) 0 – No GUARD_TIME needed
17	Read Only	Write_Write	1 – GUARD_TIME needed between 2 subsequent I2C write operations (I2C STOP to I2C START) 0 – No GUARD_TIME needed
16:9	Read Only	GUARD_TIME	The value in this register defines the GUARD_TIME needed by the I2C TPM if indicated in the bits 20:17 of this register. A value of 0 is only allowed if all bits 20:17 are set to 0. All other values represent the GUARD_TIME in µs (e.g. 0x01 means 1 µs and 0xFA means 250 µs).

Abbreviation:			TPM_I2C_INTERFACE_CAPABILITY
General Description:			This register provides miscellaneous information about the interface capabilities of the I2C-TPM.
Bit Descriptions:			
Interface version detection			
8:7	Read Only	tpmFamily	TPM Family Identifier 00: TPM 1.2 Family 01: TPM 2.0 Family 10 – 11: Reserved
6:4	Read Only	InterfaceVersion	000: TCG I2C interface 1.0 as defined in this specification 001 – 111: Reserved
3:0	Read Only	InterfaceType	0010 – FIFO interface on I2C 0000 – Reserved for PTP use, see the PTP [2] for details 0001 – Reserved for PTP use, see the PTP [2] for details 1111 - Reserved for PTP use, see the PTP [2] for details

6.5.11 TPM_DATA_CSUM_ENABLE

Table 8 — Data Checksum Enable Register

Abbreviation:			TPM_DATA_CSUM_ENABLE
General Description:			Enables the data checksum calculation and indication via the TPM_DATA_CSUM register.
Bit Descriptions:			
7:1		Reserved	Reads always return 0
0	Read/ Write	dataCSumEnable	1 = Data Checksum enabled 0 = Data Checksum disabled (default)

1. The TPM SHALL accept a write to this register after it has performed all power-on initialization (see 4.2.8 for details).
2. The TPM SHALL accept a write to this register when it is in Idle or Ready state (see the PTP [2] section 5.5.2.8 for details).
3. The TPM MAY NOT accept a write to this register when it is in Reception, Execution or Completion state.

6.5.12 TPM_DATA_CSUM

Table 9 — Data Checksum Register

Abbreviation:			TPM_DATA_CSUM
General Description:			Contains the data checksum when enabled
Bit Descriptions:			
15:0	Read only	DataChecksum	Read returns the Checksum of the entire command data at the end of the command transmission or the Checksum of the entire response data at the end of the response transmission Default: 0x00

1. The TPM SHALL use CRC-CCITT (KERMIT) for the calculation of the data checksum (see [4] for further details). The parameters are as follows:
   a. Generator polynomial is 0x1021 ($x^{16} + x^{12} + x^5 + 1$)
   b. The initialization value is 0x0000
   c. Reflection of input data: TRUE
   d. Reflection of output data: TRUE
   e. Final XOR: 0x0000
   f. Test vectors:
      i. The CRC value for the ASCII string "123456789" is 0x8921.
      A 2-byte read from 0x44 will return
      Data0 (LSB) = 0x21 and Data1 (MSB) = 0x89.

ii. The CRC value for the ASCII string “1122334455” is 0xD367.
A 2-byte read from 0x44 will return
Data0 (LSB) = 0x67 and Data1 (MSB) = 0xD3.

iii. The CRC value for the HEX string 00 C1 00 00 00 0C 00 00 00 99 00 01_{16}
(TPM_StartUp(ST_CLEAR) is 0xFBBF.
A 2-byte read from 0x44 will return
Data0 (LSB) = 0xBF and Data1 (MSB) = 0xFB.

iv. The CRC value for the HEX string 80 01 00 00 00 0C 00 00 01 44 00 00_{16}
(TPM2_StartUp(TPM_SU_CLEAR) is 0x6733.
A 2-byte read from 0x44 will return
Data0 (LSB) = 0x33 and Data1 (MSB) = 0x67.

2. If enabled via the TPM_DATA_CSUM_ENABLE register:

a. The TPM SHALL calculate the checksum over the entire command.

b. The TPM SHALL calculate the checksum over the entire response.

c. The TPM SHALL update the command checksum after reception of the last command byte and before the transition of TPM_STS.Expect from 1 to 0. The I2C-TPM SHALL maintain the command checksum from the transition of TPM_STS.Expect from 1 to 0 until TPM_STS.tpmGo is set to 1.

d. The TPM SHALL update the response checksum after the last response byte has been read and before the transition of TPM_STS.dataAvail from 1 to 0. The I2C-TPM SHALL maintain the response checksum from the transition of TPM_STS.dataAvail from 1 to 0 until Host writes a 1 to TPM_STS.commandReady.

6.5.13 TPM_DID_VID

See the PTP [2] section 5.4.1.1 for a detailed description.

6.5.14 TPM_RID

See the PTP [2] section 5.4.1.2 for a detailed description.

6.5.15 TPM_I2C_DEVICE_ADDRESS

Start of informative comment

For I2C devices connected to the same bus each device must have its own unique I2C device address to avoid a bus conflict. There are 3 different possibilities:

- A device has a fixed address which can’t be changed

- A device address may be configured via dedicated pins

- A device address may be changed using a dedicated command or register

The I2C TPM specification defines a register mechanism to change the I2C device address in situations where the bus needs to be shared with other devices which can’t be re-configured. If that is the case a simple write access to the TPM_I2C_DEVICE_ADDRESS register with the new deviceAddress and (this is recommended) the makePersistent bit set will solve such an address conflict.

There may be situations where the new deviceAddress is only needed for one power cycle, in such cases the makePersistent bit should be set to 0.

End of informative comment

Table 10 — I2C Device Address Register

Abbreviation:			TPM_I2C_DEVICE_ADDRESS
General Description:			This register holds the I2C device address.
Bit Descriptions:			
15	Write only	makePersistent	1 = Persistent device address defined by bits 6:0 0= Volatile device address defined by bits 6:0, lost after reset.
14:7		Reserved	Reads always return 0
6:0	Write only	deviceAddress	I2C device address Reads return the current I2C device address Writes set the new I2C device address effective with the next I2C master access Default: 0x2E

6.6 Interface Locality Usage per Register

Start of informative comment

Table 11 shows how the TPM responds to accesses to each of the interface registers based on locality settings for the FIFO interface.

End of informative comment

Table 11 — Register Behavior Based on Locality Setting for I2C

TPM_ACCESS.activeLocality					
Set for This Locality		**Set for other Locality**		**Not Set**	
READ	**WRITE**	**READ**	**WRITE**	**READ**	**WRITE**
TPM_STS Registers					
TPM returns correct value	Fields updated	TPM returns 0xFF	TPM Ignore the write	TPM returns 0xFF	TPM Ignore the write
TPM_ACCESS Registers					
TPM returns correct value	Fields updated	TPM returns correct value	Fields updated	TPM returns correct value	Fields updated
TPM_DATA_FIFO Registers					
TPM returns correct data	TPM accepts data and command	TPM returns 0xFF	TPM Ignore the write	TPM returns 0xFF	TPM Ignore the write
TPM_HASH_START Register					
TPM returns 0xFF	TPM accepts command	TPM returns 0xFF	TPM Ignore the write	TPM returns 0xFF	TPM accepts command and sets TPM_ACCESS. activeLocality for Locality 4
TPM_HASH_DATA Register					
TPM returns 0xFF	TPM accepts data	TPM returns 0xFF	TPM Ignore the write	TPM returns 0xFF	TPM Ignore the write
TPM_HASH_END Register					
TPM returns 0xFF	TPM accepts command and clears TPM_ACCESS. activeLocality for Locality 4	TPM returns 0xFF	TPM Ignore the write	TPM returns 0xFF	TPM Ignore the write

TPM_ACCESS.activeLocality					
Set for This Locality		**Set for other Locality**		**Not Set**	
READ	**WRITE**	**READ**	**WRITE**	**READ**	**WRITE**
TPM_LOC_SEL Register					
TPM returns correct value	Fields updated	TPM returns correct value	Fields updated	TPM returns correct value	Fields updated
TPM_INT_ENABLE Register					
TPM returns correct value	Field updated	TPM returns correct value	Field updated	TPM returns correct value	Field updated
TPM_INT_STATUS Register					
TPM returns correct value	Interrupt cleared	TPM returns correct value	Interrupt cleared	TPM returns correct value	Interrupt cleared
TPM_INT_CAPABILITY Register					
TPM returns correct value	Read-only register	TPM returns correct value	Read-only register	TPM returns correct value	Read-only register
TPM_I2C_INTERFACE_CAPABILITY Register					
TPM returns correct value	Read-only register	TPM returns correct value	Read-only register	TPM returns correct value	Read-only register
TPM_I2C_DEVICE_ADDRESS Register					
TPM returns correct value	Fields updated	TPM returns correct value	Fields updated	TPM returns correct value	Fields updated
TPM_CSUM_ENABLE Register					
TPM returns correct value	Fields updated	TPM returns correct value	Fields updated	TPM returns correct value	Fields updated
TPM_DATA_CSUM Register					
TPM returns correct value	Read-only register	TPM returns correct value	Read-only register	TPM returns correct value	Read-only register
TPM_DID_VID Register					
TPM returns correct value	Read-only register	TPM returns correct value	Read-only register	TPM returns correct value	Read-only register
TPM_RID Register					
TPM returns correct value	Read-only register	TPM returns correct value	Read-only register	TPM returns correct value	Read-only register

6.7 TCG-I2C Protocol Usage Scenarios

6.7.1 Simple access to TPM_ACCESS

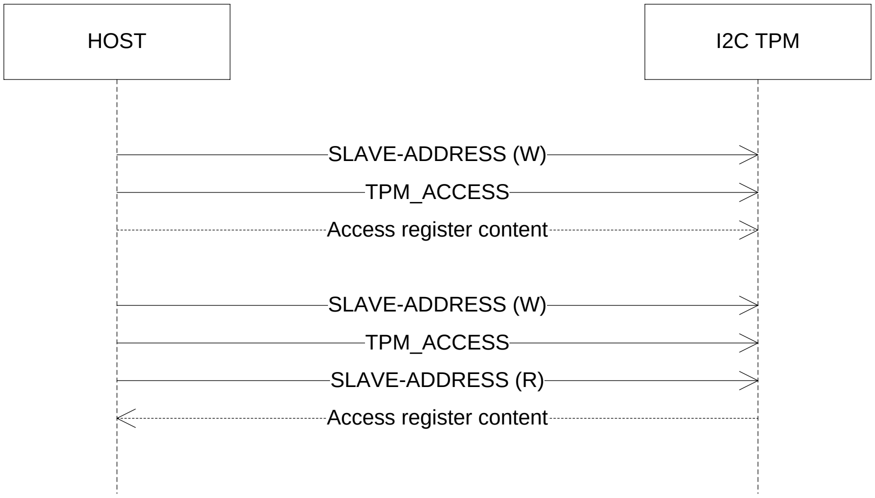


Figure 8 — Write / Read TPM_ACCESS register w/o locality selection

6.7.2 Access to TPM_ACCESS from Locality 0 only

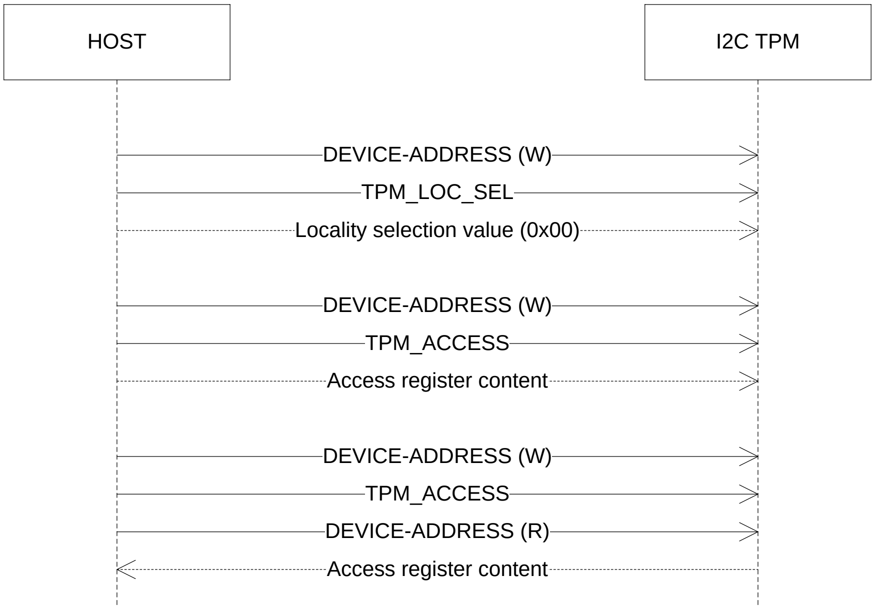


Figure 9 — Write / Read TPM_ACCESS register from Locality 0

6.7.3 Access to TPM_ACCESS from Locality 0 and 2

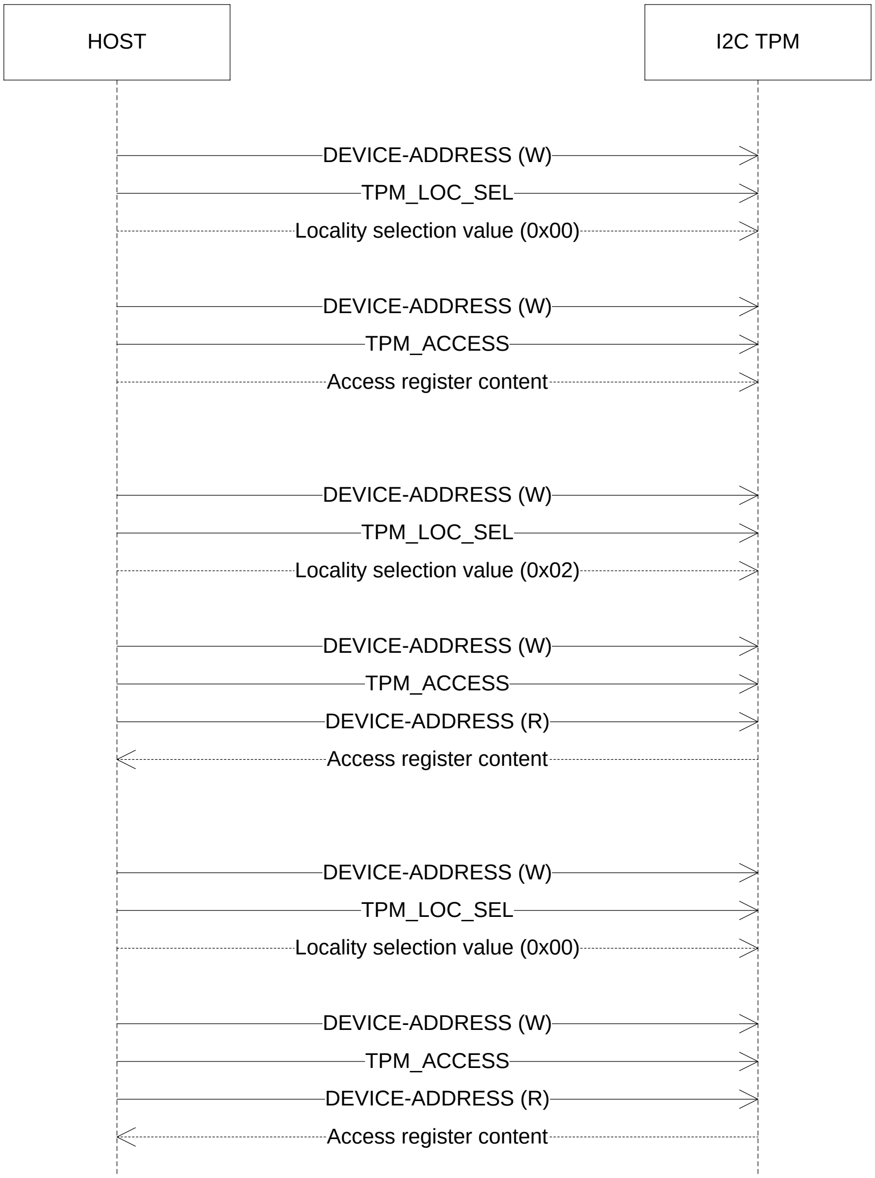


Figure 10 — Write / Read TPM_ACCESS register from Locality 0 and 2

6.7.4 Access to TPM_STS from Locality 0

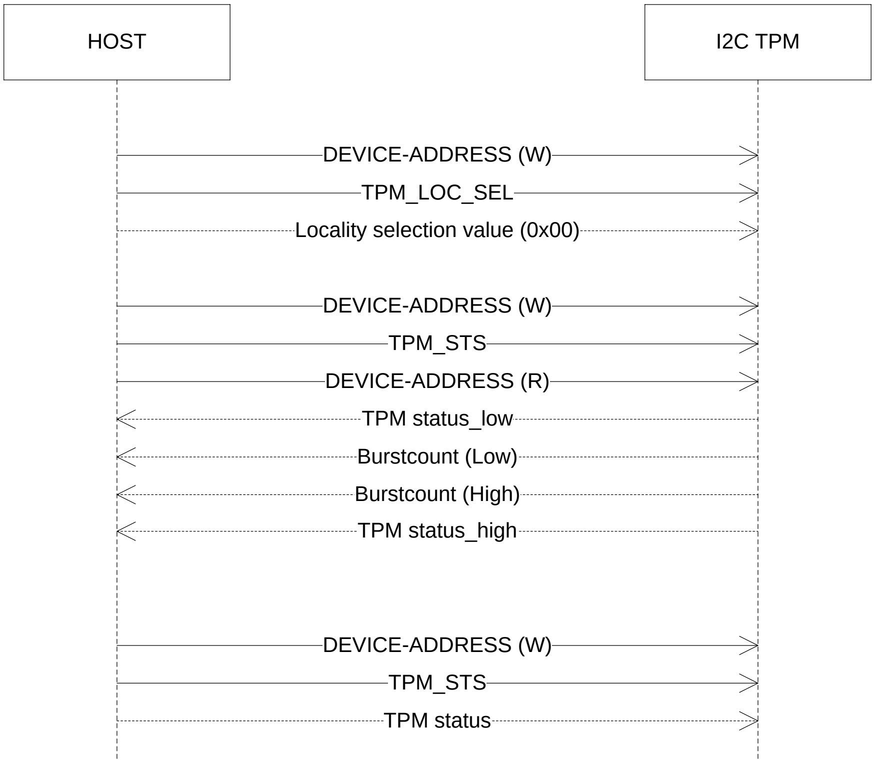


Figure 11 — Read / Write TPM_STS register(s) from Locality 0

6.7.5 Read from TPM_DATA_FIFO from Locality 0

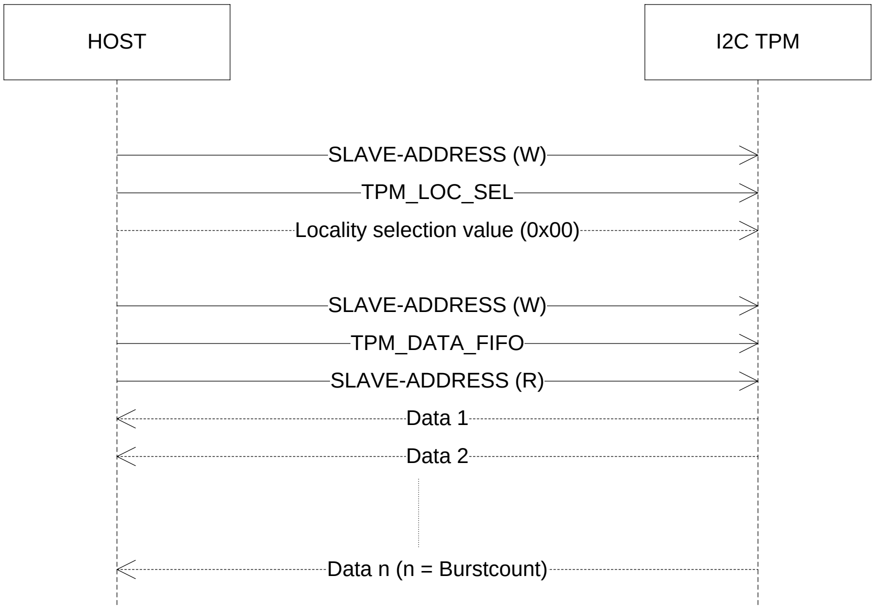


Figure 12 — Read TPM_DATA_FIFO

6.7.6 Write to TPM_DATA_FIFO from Locality 0

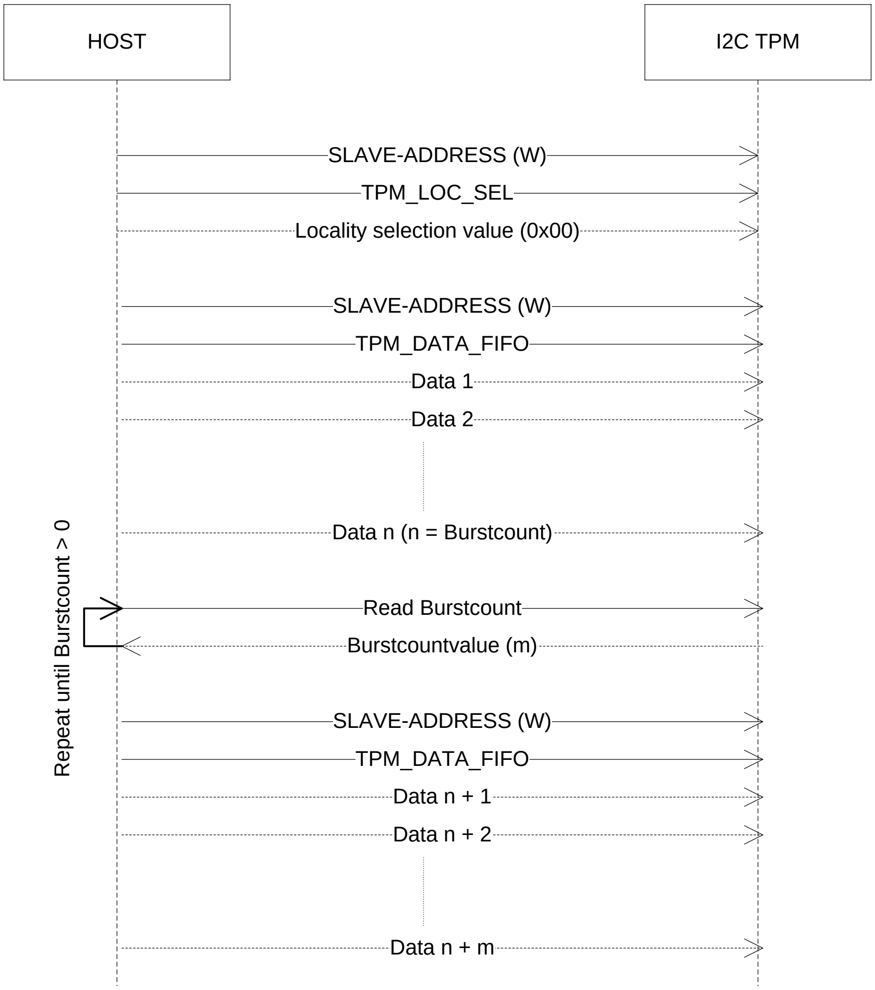


Figure 13 — Write TPM_DATA_FIFO