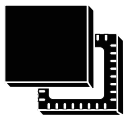
ST33KTPM-IDevID

Datasheet

ST33KTPM provisioned for device identity and attestation



UFQFPN32 (5 × 5 × 0.55 mm)

Product status link
ST33KTPM-IDevID

Features

TPM features

- Flash memory-based trusted platform module (*TPM*)
- Compliant with Trusted Computing Group (*TCG*) trusted platform module (*TPM*) Library specifications 2.0, revision 1.59 errata version 1.5 and *TCG* PC Client Platform *TPM* Profile (*PTP*) for *TPM* 2.0 version 1.05
- Fault-tolerant firmware loader that keeps the *TPM* fully functional when the loading process is interrupted
- SP800-193 compliant for protection, detection and recovery requirements
- Targeted certifications:
  - Common Criteria EAL4+ in compliance with the *TPM* 2.0 protection profile (augmented with AVA_VAN.5, resistant to high-potential attacks)
  - *FIPS* 140-3 with physical security level 3
  - *TCG* certification
- 192 KB *NV* memory available to store keys and data
- *SPI* communication bus running at up to 66 MHz
- *I²C* communication bus running at up to 1 Mb/s

IDevID and IAK provisioning

- ST33KTPM-IDevID devices are provisioned with 2 signing keys ECC NIST P-384 and 2 certificates to support device identity and attestation use cases, compliant with TCG TPM 2.0 Keys for Device identity and Attestation v1.1

Hardware features

- Highly reliable flash memory with error correction code
- Extended temperature range: −40°C to 105°C
- Electrostatic discharge (ESD) protection up to 4 kV (HBM)
- 1.8 V or 3.3 V supply voltage range

Security features

- Active shield
- Monitoring of environmental parameters
- Hardware and software protection against fault injection and side channel attacks
- *NIST* SP800-90A and AIS20-compliant deterministic random-bit generator (DRBG)
- *NIST* SP800-90B and AIS31-compliant true random-number generator (*TRNG*)

- Cryptographic algorithms:
  - *RSA* key generation (1024, 2048, 3072 and 4096 bits)
  - *RSA* signature (*RSASSA-PSS*, *RSASSA-PKCS*1v1_5)
  - *RSA* encryption (*RSAES*-OAEP, RSAES-*PKCS*1-v1_5)
  - SHA-1, SHA-2 (256 and 384 bits), SHA-3 (256 and 384 bits)
  - *HMAC* SHA-1, SHA-2, and SHA-3
  - AES-128, 192, and 256 bits
  - *ECC* key generation (*NIST* P-256/384)
  - *ECC* secret sharing (*ECDH*)
  - *ECC* signature (*ECDSA*, ECSchnorr, *ECDAA*)
- Device provided with three endorsement keys (*EK*) and *EK* certificates (RSA2048, *ECC NIST* P-256 and *ECC NIST* P-384)
- Device provisioned with three 2048-bit *RSA* key pairs to reduce the *TPM* provisioning time

Product compliance

- Compliant with Microsoft® Windows® 10 and 11
- Compliant with Linux® drivers
- Compliant with Intel® vPro® technology
- Compliant with *TCG* test suite for *TPM* 2.0
- Compliant with the open-source *TCG TPM* 2.0 *TSS* implementation

1 Description

The ST33KTPM-IDevID is a trusted platform module provisioned with keys and certificates to support use cases like device identity and attestation. The ST33KTPM-IDevID is compliant with TCG TPM 2.0 keys for device identity and attestation 1.1.

The IDevID key and X509 certificate provide the device identity defined by IEEE 802.1AR and are compatible with cloud system authentication such as AWS IoT, and Microsoft Azure, and can also be used in TLS 1.3 session to authenticate client.

The ST33KTPM-IDevID comes also with bundle files including device leaf certificates for a specific reel that are made available through authenticated download. This improves the supply chain security as the ST33KTPM-IDevID devices identified can be activated thanks to the bundle files after reception of reels, or can be revoked in case of lost shipment. This also facilitates the loading of leaf certificates in the cloud system before getting access to the physical platforms embedding ST33KTPM-IDevID devices.

The ST33KTPM-IDevID is based on product ST33KTPM2X compliant with the *TCG TPM* library specifications 1.59. It offers a target serial peripheral interface (*SPI*) or a target *I²C* interface, both compliant with the TCG PC client TPM profile specifications and PC client platform TPM profile (*PTP*) version 1.05.

ST33KTPM-IDevID devices are certified Common Criteria, TCG, and *FIPS*140-3.

It offers resilience services during the TPM firmware upgrade process, and self-recovery of TPM firmware and critical data upon failure detection.

The ST33KTPM-IDevID operates in the –40°C to 105°C extended temperature range.

The ST33KTPM-IDevID generic parts are loaded with X509 certificates signed by an STSAFE-TPM certification authority described in the provisioning profile. Upon customer request, STMicroelectronics can support a customer-specific provisioning profile and assign a dedicated certification authority linked to a specific ordering code.

ST33KTPM-IDevID devices with generic profile are based on ST33KTPM2X product offered in UFQFPN32 Ecopack2 package.

ST33KTPM-IDevID devices with customer profile are based either on ST33KTPM2X product offered in UFQFPN32 Ecopack2 package or on ST33KTPM2I product offered in WLCSP Ecopack2 package.

2 Datasheet scope

2.1 ST33KTPM-IDevID product

This datasheet describes the ST33KTPM-IDevID product based on the product ST33KTPM2X provisioned with a generic provisioning profile.

A unique 3-letter ID printed on the package (Area L) defines each provisioning profile. Provisioning profiles can be generic or customer specific. The description of the provisioning is defined in a separate application note including the reference to the provisioning profile ID (refer to [AN6324]) .

This datasheet includes the ordering code of ST33KTPM-IDevID devices with generic provisioning profiles.

The ordering codes of ST33KTPM-IDevID devices with customer-specific profiles are defined in documents with restricted access to those customers. These documents are provided by your sales contact.

The procedure to download bundles files is described in the application note [AN6350], "Online certificate distribution for STSAFE-TPM products".

3 UFQFPN32 pin and signal description

The figure below gives the pinout of the UFQFPN32 package in which the devices are delivered. Table 1 describes the associated signals.

Figure 1. UFQFPN32 pinout

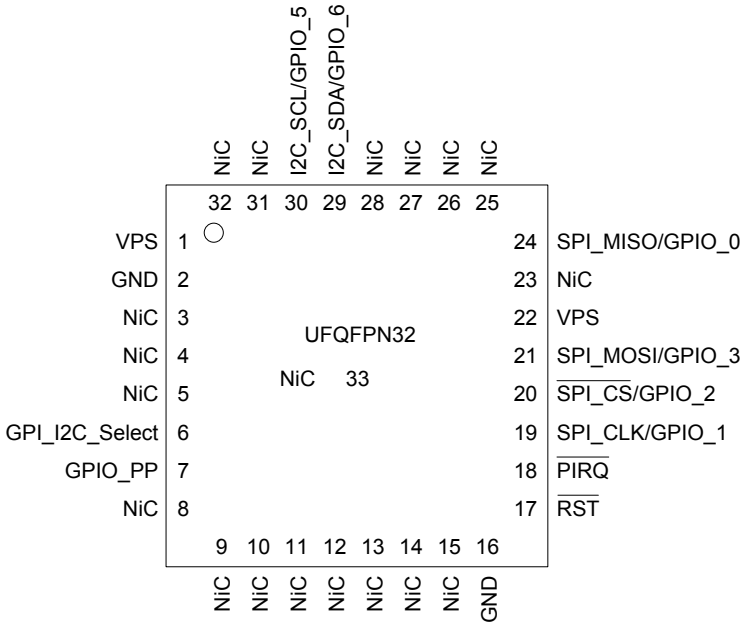


Table 1. UFQFPN32 pin descriptions

Signal	Type	GPIO configuration during reset	Description
VPS	Input	—	**Power supply**. This pin must be connected to 1.8 V or 3.3 V DC power rail supplied by the motherboard.
GND	Input	—	**Ground**, has to be connected to the main motherboard ground.
$\overline{\text{RST}}$	Input	VWPD	**Reset**, active low, used to re-initialize the device. Must not be unconnected. External pull-up resistor is required if it cannot be driven.
SPI_MISO	Output	VWPD	***SPI* master input, slave output** (output from slave)
SPI_MOSI	Input	VWPD	***SPI* master output, slave input** (output from master)
SPI_CLK	Input	VWPD	***SPI* serial clock** (output from master)
$\overline{\text{SPI\_CS}}$	Input	VWPD	***SPI* chip (or slave) select**, internal pull-up (active low; output from master)
$\overline{\text{PIRQ}}$	Output	VWPD	**IRQ**, active low, open drain, used by the *TPM* to generate an interrupt
GPIO_PP	Input	VWPD	**Physical presence** (*PP*), active high, internal very weak pull down. Used to indicate physical presence to the *TPM*. The *GPIO* function could be modified by activating the *GPIO*s mapped with the *NV* storage index feature.
GPI_I2C_Select	Input	WPU	This pin must be connected to an external pull-down resistor to activate the *I²C* protocol during product boot time. It can remain unconnected for the *SPI* protocol. This pin is internal weak pull-up by default and becomes internal floating after *I²C* activation.
NiC	—	—	**Not internally connected**: not connected to the die. May be left unconnected but no impact on *TPM* if connected.
GPIO_X	Input/ output	VWPD	The *GPIO* function could be modified by activating the *GPIO*s mapped with the *NV* storage index feature. *GPIO* availability is dependent of bus interface (for example, GPIO_5 and GPIO_6 are available with the *SPI* interface activated).
I2C_SDA/GPIO_6	Input/ output	VWPD	**Bidirectional *I²C* serial data** (open drain without a weak pull-up resistor) / General-purpose input/output if *SPI* is activated[1]
I2C_SCL/GPIO_5	Input	VWPD	**Input *I²C* serial clock** (open drain without a weak pull-up resistor) / General-purpose input/output if *SPI* is activatedGeneral-purpose input/ output[1]

1. *The GPIO function could be modified by activating the GPIOs mapped with the NV storage index feature.*

Note: The UFQFPN32 package has a central pad (PIN33) on the bottom, which is not connected to the die. This pin does not impact the TPM, be it connected or not.

4 Serial peripheral interface (*SPI*) and *TPM* registers

The *SPI* interface implemented in this device complies with the *TCG* PC Client-specific *TPM* Platform specifications [PTP 2.0 r1.05] for the *TPM* 2.0 library.

The *SPI* supports only one clock mode (CPOL=0, CPHA=0).

Data is transferred serially between master and slave; the most significant bit (*MSB*) first, least significant bit (*LSB*) last.

Addresses and commands are transferred msb first for the entire field, e.g. the 24-bit address is transferred by sending b23 first, then b22 all the way up to b0.

The *TPM* drives data on the falling edge of the *SPI* clock.

The *TPM* samples data on the rising edge of the *SPI* clock.

The *TPM* always decodes a 24-bit address in the 0xD4_xxxx range when the *TPM* $\overline{\text{SPI\_CS}}$ pin is asserted.

The ST33KTPM-IDevID supports an interrupt interface line directly from the *TPM* to the main platform *MCU*. This interrupt is supported using a dedicated *GPIO* ($\overline{\text{SPI\_PIRQ}}$, active low). The device detects a change in the *TPM* status when the *TPM* triggers a falling edge on pin $\overline{\text{SPI\_PIRQ}}$.

4.1 SPI communication protocol flow control

4.1.1 SPI communication

The *TPM* flow control mechanism operates on a transaction basis and can transfer data of various sizes. The *TPM* can transfer or receive 1 up to 32 bytes of data per transaction.

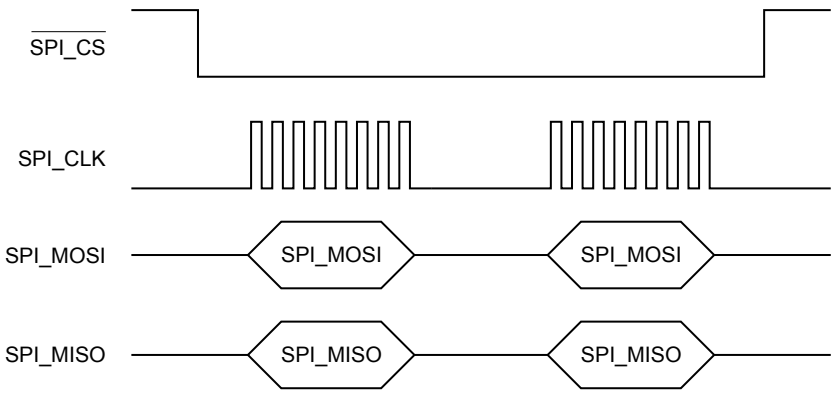
Because the *SPI* does not support any acknowledge signal, a specific way to synchronize between the host and the *TPM* has been defined.

In order to synchronize the start of a command, the *SPI* master must set low the $\overline{\text{SPI\_CS}}$ pin when sending a command and set the $\overline{\text{SPI\_CS}}$ pin back to high at the end of the transaction.

An *SPI* start of frame is detected when the $\overline{\text{SPI\_CS}}$ signal goes low. This causes the *TPM* to drive the SPI_MISO signal low. An end of frame is detected when the $\overline{\text{SPI\_CS}}$ line goes to the high voltage level.

The transmit sequence begins when the *TPM* receives the SPI_CLK signal, the $\overline{\text{SPI\_CS}}$ signal goes low and the most significant bit of the data is present on the SPI_MOSI pin.

Figure 2. SPI transmit sequence



The *TPM* host initiates every command by transmitting a single byte on the SPI_MOSI pin that tells the *TPM* if it is a read or write operation in a register. This byte also contains the size of the transfer including the 1st byte. The command processing continues with the transfer of three more bytes containing the address of the targeted register.

At this time, depending on the targeted register, the *TPM* is able to insert 1 or more Wait states before acknowledging the transfer. A Wait state is a Low state (0) sent on the SPI_MISO pin by the *TPM*. An Acknowledgment is a byte, whose last bit is ‘1’, sent on the SPI_MISO pin by the *TPM*.

In the case of a register read command, the *TPM* sends on the SPI_MISO pin the number of requested bytes from the requested register.

In the case of a register write command, the *TPM* reads on the SPI_MOSI pin the number of bytes previously declared and modifies accordingly the *TPM* state machine.

Table 2. SPI bit protocol

Bit transfer order on SPI_MISO/SPI_MOSI pins	Byte on SPI_MISO/SPI_MOSI pins	Usage	Notes
RFU	67 for 64B transactions 11 for 8B transactions	Reserved for future use for larger register sizes	-
57-63 – last bits on wire	7	Data[30:24]	-
56		Data[31]	msb of 4th *LSB*
49-55	6	Data[22:16]	-
48		Data[23]	msb of 3rd *LSB*
41-47	5	Data[14:8]	-
40		Data[15]	msb of 2nd *LSB*
33-39	4	Data[6:0]	-
32		Data[7]	msb of 1st *LSB*
Optional flow control can be done in this window. See Protocol flow control for SPI read access and Protocol flow control for SPI write access for details. This is the only place in the bit transfers where flow control can be performed.			
31	3	Addr[0]	lsb of address
9-30	1-3	Addr[22] down to Addr[1]	-
8	1	Addr[23]	msb of address
2-7	0	bits[5:0] Size of transfer where bit[5] of this field is the 3rd bit transferred on the wire, and bit [0] is the 8th bit on the wire. This field is 0's based count of the bytes. Any byte count from 1 to 64 is legal.	Bit [5:0] decode '11_1111' = 64 bytes ' etc. for 63 down to 6 bytes '00_0100' = 5 bytes '00_0011' = 4 bytes '00_0010' = 3 bytes '00_0001' = 2 bytes '00_0000' = 1 byte
1		Reserved; bit[6]	-
0 – first bit on wire		Byte0, bit[7] Read/Write	1=read, 0 = write

4.1.2 Protocol flow control for SPI read access

Following the standard specification, depending on the targeted register, the *TPM* is authorized to insert 1 Wait state before acknowledging the read access. A Wait state is a Low state (0) sent on the SPI_MISO pin by the *TPM*. An Acknowledgment is a byte, whose last bit is '1', sent on the SPI_MISO pin by the *TPM*.

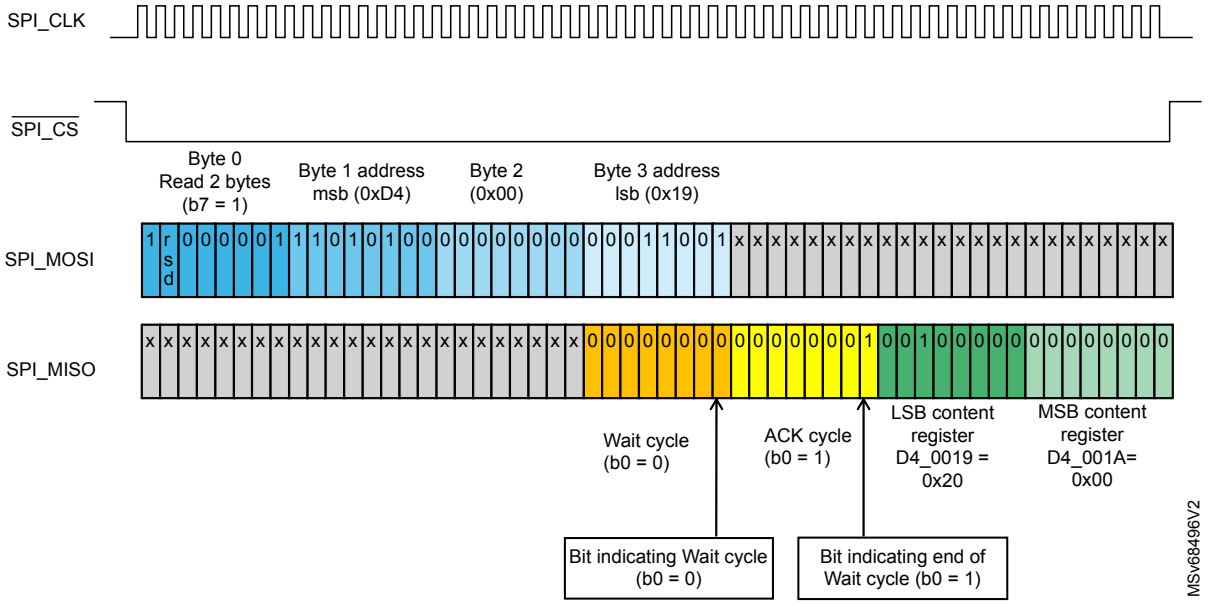
The ST *TPM* uses only one wait state for *SPI* read access.

On read access, if the data are not available, the ST *TPM* uses the dedicated bits defined by the standard specification:

- stsValid (bit 7 in the Status register): This bit indicates whether TPM_STS_x.dataAvail and TPM_STS_x.Expect are valid. They are valid when stsValid = 1.
- tpmRegValidSts (bit 7 in the Access register): This bit indicates whether all other bits in this register contain valid values. All values are valid when tpmRegValidSts = 1.

The host should repeat read access and consider that the *TPM* is out of order only after TIMEOUT_B (2 seconds) of no data availability.

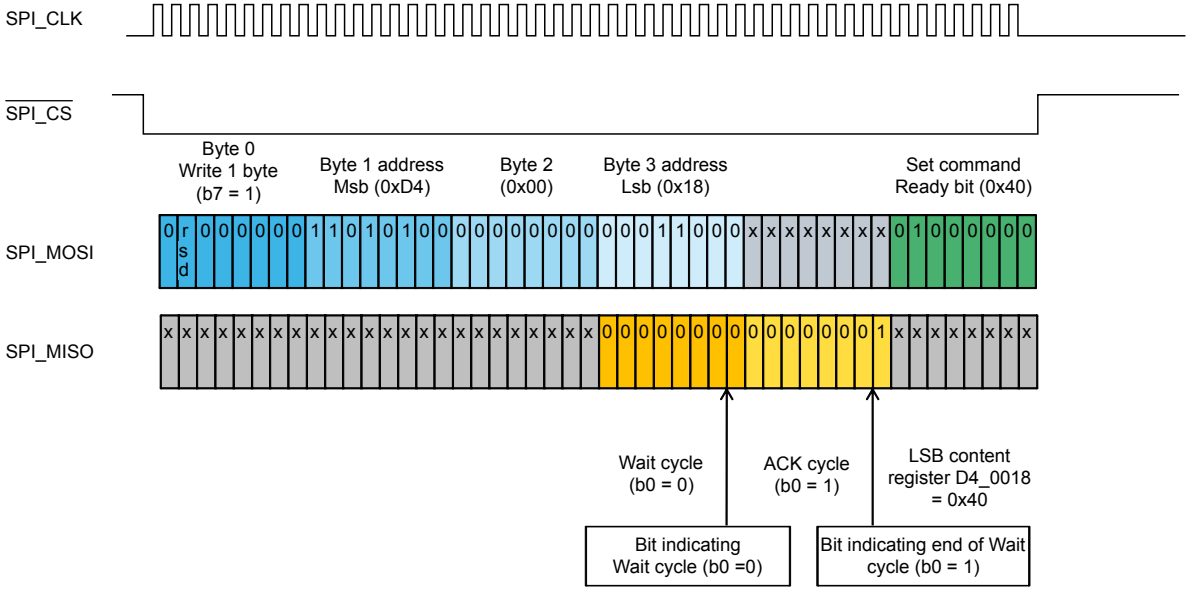
Figure 3. SPI Read register example



4.1.3 Protocol flow control for SPI write access

Following the standard specification, the *TPM* is authorized to insert 1 or more Wait states before acknowledging the write access. A Wait state is low state (0) sent on the SPI_MISO pin by the *TPM*. An Acknowledgment is a byte, whose last bit is '1', sent on the SPI_MISO pin by the *TPM*.

Figure 4. SPI write register example



4.2 Register space addresses

The following table shows *TPM* register space starting from the base address 0xD4_0000.

4.2.1 FIFO interface

Table 3. List of FIFO register space addresses

Locality 0	Register name	Locality 1	Register name	Locality 2	Register name	Locality 3	Register name	Locality 4	Register name	Register description
0x0000	TPM_ACCESS_0	0x1000	TPM_ACCESS_1	0x2000	TPM_ACCESS_2	0x3000	TPM_ACCESS_3	0x4000	TPM_ACCESS_4	Used to gain ownership of the TPM for this particular Locality
0x000B-0x0008	TPM_INT_ ENABLE_0	0x100B-0x1008	TPM_INT_ ENABLE_1	0x200B-0x2008	TPM_INT_ ENABLE_2	0x300B-0x30008	TPM_INT_ ENABLE_3	0x400B-0x4008	TPM_INT_ ENABLE_4	Interrupt Configuration Register
0x000C	TPM_INT_ VECTOR_0	0x100C	TPM_INT_ VECTOR_1	0x200C	TPM_INT_ VECTOR_2	0x300C	TPM_INT_ VECTOR_3	0x400C	TPM_INT_ VECTOR_4	SIRQ vector to be used by the TPM (SIRQ pin not available)
0x0013-0x0010	TPM_INT_ STATUS_0	0x1013-0x1010	TPM_INT_ STATUS_1	0x2013-0x2010	TPM_INT_ STATUS_2	0x3013-0x3010	TPM_INT_ STATUS_3	0x4013-0x4010	TPM_INT_ STATUS_4	Shows which interrupt has occurred
0x0017-0x0014	TPM_INTF_ CAPABILITY_0	0x1017-0x1014	TPM_INTF_ CAPABILITY_1	0x2017-0x2014	TPM_INTF_ CAPABILITY_2	0x3017-0x3014	TPM_INTF_ CAPABILITY_3	0x4017-0x4014	TPM_INTF_ CAPABILITY_4	Provides the information about supported interrupts and the characteristic of the burstCount register of the particular TPM.
0x001A-0x0018	TPM_STS_0	0x101A-0x1018	TPM_STS_1	0x201A-0x2018	TPM_STS_2	0x301A-0x3018	TPM_STS_3	0x401A-0x4018	TPM_STS_4	Status Register. Provides status of the TPM
-	-	-	-	-	-	-	-	0x4020	TPM_HASH_END	This signals the end of the hash operation in Locality 4.
0x00027-0x0024	TPM_DATA_ FIFO_0	0x1027-0x1024	TPM_DATA_ FIFO_1	0x2027-0x2024	TPM_DATA_ FIFO_2	0x3027-0x3024	TPM_DATA_ FIFO_3	0x4027-0x4024	TPM_HASH_DATA / TPM_DATA_FIFO_4	ReadFIFO or WriteFIFO, depending on the current bus cycle (read or write). These four addresses are aliased to one inside the TPM. In Locality 4, it is the HASH_DATA FIFO.
-	-	-	-	-	-	-	-	0x4028	TPM_HASH_START	This signals the start of the hash operation in Locality 4.
0x0037-0x0034	TPM_DATA_CSUM_ENABLE_0	0x1037-0x1034	TPM_DATA_CSUM_ENABLE_1	0x2037-0x2034	TPM_DATA_CSUM_ENABLE_2	0x3037-0x3034	TPM_DATA_CSUM_ENABLE_3	0x4037-0x4034	TPM_DATA_CSUM_ENABLE_4	Activation of the data checksum
0x003F-0x0038	TPM_DATA_CSUM_0	0x103F-0x1038	TPM_DATA_CSUM_1	0x203F-0x2038	TPM_DATA_CSUM_2	0x303F-0x3038	TPM_DATA_CSUM_3	0x403F-0x4038	TPM_DATA_CSUM_4	Checksum data locations

Locality 0	Register name	Locality 1	Register name	Locality 2	Register name	Locality 3	Register name	Locality 4	Register name	Register description
0x0080-0x0083	TPM_XDATA_FIFO_0	0x1080-0x1083	TPM_XDATA_FIFO_1	0x2080-0x2083	TPM_XDATA_FIFO_2	0x3080-0x3083	TPM_XDATA_FIFO_3	0x4080-0x4083	TPM_XDATA_FIFO_4	Extended ReadFIFO or WriteFIFO, depending on the current bus cycle (read or write). Transactions to this address may be any size from 1B to maxTransferCapability identified in the capability register.
0x00BF-0x0084	Reserved	0x10BF-0x1084	Reserved	0x20BF-0x2084	Reserved	0x30BF-0x3084	Reserved	0x40BF-0x4084	Reserved	-
0x0F03-0x0F00	TPM_DID_VID_0	0x1F03-0x1F00	TPM_DID_VID_1	0x2F03-0x2F00	TPM_DID_VID_2	0x3F03-0x3F00	TPM_DID_VID_3	0x4F03-0x4F00	TPM_DID_VID_4	Vendor and device ID
0x0F04	TPM_RID_0	0x1F04	TPM_RID_1	0x2F04	TPM_RID_2	0x3F04	TPM_RID_3	0x4F04	TPM_RID_4	Revision ID
0x0F7F-0x0F05	Reserved	0x1F7F-0x1F05	Reserved	0x2F7F-0x2F05	Reserved	0x3F7F-0x3F05	Reserved	0x4F7F-0x4F05	Reserved	-

4.2.2 Command response buffer interface

Table 4. List of CRB register space addresses

Locality 0	Register name	Locality 1	Register name	Locality 2	Register name	Locality 3	Register name	Locality 4	Register name
0x0003-0x0000	TPM_LOC_STATE_0	0x1003-1000	TPM_LOC_STATE_1	0x2000-0x2003	TPM_LOC_STATE_2	0x3000-0x3003	TPM_LOC_STATE_3	0x4000-0x4003	TPM_LOC_STATE_4
0x000B-0x0008	TPM_LOC_CTRL_0	0x100B-1008	TPM_LOC_CTRL_1	0x200B-0x2008	TPM_LOC_CTRL_2	0x300B-0x30008	TPM_LOC_CTRL_3	0x400B-0x4008	TPM_LOC_CTRL_4
0x000F-0x000C	TPM_LOC_STS_0	0x100F-100C	TPM_LOC_STS_1	0x200F-0x200C	TPM_LOC_STS_2	0x300F-0x300C	TPM_LOC_STS_3	0x400F-0x400C	TPM_LOC_STS_4
0x002F-0x0010	Reserved	0x102F-1010	Reserved	0x202F-0x2010	Reserved	0x302F-0x3010	Reserved	0x402F-0x4010	Reserved
0x0037-0x0030	TPM_INTERFACE_ IDENTIFIER_0	0x1037-1030	TPM_INTERFACE_ IDENTIFIER_1	0x2037-0x2030	TPM_INTERFACE_ IDENTIFIER_2	0x3037-0x3030	TPM_INTERFACE_ IDENTIFIER_3	0x4037-0x4030	TPM_INTERFACE_ IDENTIFIER_4
0x003F-0x0038	TPM_CRB_CTRL_EXT_0	0x103F-0x1038	Reserved	0x203F-0x2038	Reserved	0x303F-0x3038	Reserved	0x403F-0x4038	Reserved
0x0043-0x0040	TPM_CRB_CTRL_REQ_0	0x1043-0x1040	TPM_CRB_CTRL_REQ_1	0x2043-0x2040	TPM_CRB_CTRL_REQ_2	0x3043-0x3040	TPM_CRB_CTRL_REQ_3	0x4043-0x4040	TPM_CRB_CTRL_REQ_4
0x0047-0x0044	TPM_CRB_CTRL_STS_0	0x1047-0x1044	TPM_CRB_CTRL_STS_1	0x2047-0x2044	TPM_CRB_CTRL_STS_2	0x3047-0x3044	TPM_CRB_CTRL_STS_3	0x4047-0x4044	TPM_CRB_CTRL_STS_4
0x004B-0x0048	TPM_CRB_CTRL_ CANCEL_0	0x104B-0x1048	TPM_CRB_CTRL_ CANCEL_1	0x204B-0x2048	TPM_CRB_CTRL_ CANCEL_2	0x304B-0x3048	TPM_CRB_CTRL_ CANCEL_3	0x404B-0x14048	TPM_CRB_CTRL_ CANCEL_4
0x004F-0x004C	TPM_CRB_CTRL_ START_0	0x104F-0x104C	TPM_CRB_CTRL_ START_1	0x204F-0x204C	TPM_CRB_CTRL_ START_2	0x304F-0x304C	TPM_CRB_CTRL_ START_3	0x404F-0x404C	TPM_CRB_CTRL_ START_4
0x0057-0x0050	TPM_CRB_CTRL_INT_0	0x1057-0x1050	TPM_CRB_CTRL_INT_1	0x2057-0x2050	TPM_CRB_CTRL_INT_2	0x3057-0x3050	TPM_CRB_CTRL_INT_3	0x4057-0x4050	TPM_CRB_CTRL_INT_4
0x005B-0x0058	TPM_CRB_CTRL_CMD_ SIZE_0	0x105B-0x1058	TPM_CRB_CTRL_CMD_ SIZE_1	0x205B-0x2058	TPM_CRB_CTRL_CMD_ SIZE_2	0x305B-0x3058	TPM_CRB_CTRL_CMD_ SIZE_3	0x405B-0x4058	TPM_CRB_CTRL_CMD_ SIZE_4
0x005F-0x005C	TPM_CRB_CTRL_CMD_ LADDR_0	0x105F-0x105C	TPM_CRB_CTRL_CMD_ LADDR_1	0x205F-0x205C	TPM_CRB_CTRL_CMD_ LADDR_2	0x305F-0x305C	TPM_CRB_CTRL_CMD_ LADDR_3	0x405F-0x405C	TPM_CRB_CTRL_CMD_ LADDR_4
0x0063-0x0060	TPM_CRB_CTRL_CMD_ HADDR_0	0x1063-0x1060	TPM_CRB_CTRL_CMD_ HADDR_1	0x2063-0x2060	TPM_CRB_CTRL_CMD_ HADDR_2	0x3063-0x3060	TPM_CRB_CTRL_CMD_ HADDR_3	0x4063-0x4060	TPM_CRB_CTRL_CMD_ HADDR_4
0x0067-0x0064	TPM_CRB_CTRL_RSP_ SIZE_0	0x1067-0x1064	TPM_CRB_CTRL_RSP_ SIZE_1	0x2067-0x2064	TPM_CRB_CTRL_RSP_ SIZE_2	0x3067-0x3064	TPM_CRB_CTRL_RSP_ SIZE_3	0x4067-0x4064	TPM_CRB_CTRL_RSP_ SIZE_4
0x006F-0x0068	TPM_CRB_CTRL_RSP_ ADDR_0	0x106F-0x1068	TPM_CRB_CTRL_RSP_ ADDR_1	0x206F-0x2068	TPM_CRB_CTRL_RSP_ ADDR_2	0x306F-0x3068	TPM_CRB_CTRL_RSP_ ADDR_3	0x406F-0x4068	TPM_CRB_CTRL_RSP_ ADDR_4
0x007F-0x0070	Reserved	0x107F-0x1070	Reserved	0x207F-0x2070	Reserved	0x307F-0x3070	Reserved	0x407F-0x4070	Reserved
0x0880-0x0080	TPM_CRB_DATA_ BUFFER_0	0x1880-0x1080	TPM_CRB_DATA_ BUFFER_1	0x2880-0x2080	TPM_CRB_DATA_ BUFFER_2	0x3880-0x3080	TPM_CRB_DATA_ BUFFER_3	0x4880-0x4080	TPM_CRB_DATA_ BUFFER_4
0x0FFF-0x0881	Reserved	0x1FFF-0x1881	Reserved	0x2FFF-0x2881	Reserved	0x3FFF-0x3881	Reserved	0x4FFF-0x4881	Reserved

4.3 Register descriptions

Detailed descriptions of each register can be found in *TCG* PC client specific platform *TPM* profile for [PTP 2.0 r1.05].

4.4 Additional information

- The *TPM* implements a $\overline{\text{PIRQ}}$ but does not implement an SIRQ pin.
- Any write operation to the TPM_ACCESS_x register with more than one field set to a '1' is not accepted.
- Interface capability:
  - TPM_STS_x.burstCount is dynamic.
  - Interrupt detection mode "Level low" is supported (other modes "Level high" and "Edge" are not supported).
  - Interrupts localityChange and dataAvailable are supported (Interrupt stsValid and commandReady are not supported).
- Transaction size:
  - The *TPM* accepts transactions to offset 0x0024-0x0027 which are of lengths from 1 to 32 bytes (legacy FIFO).
  - The *TPM* accepts transactions to offset 0x0081-0x0083 which are of lengths from 1 to 32 bytes (extended FIFO).
- Vendor and device ID for the *TPM*
  - TPM_DID_VID_X = 0x0003104A; // DID = 0x0003 and VID = 0x104A (ID of STMicroelectronics defined by [Vendor Registry])
- Revision ID for the *TPM*
  - TPM_RID_X = 0x03; // 0xFFFFFF = reserved
  - TPM_RID_X is a counter incremented by 1 when the HWINTF library version is updated in future products. TPM_RID_X identifies the HWINTF library (*TPM* bus interface driver) version in the register level.
- TPM_DATA_CSUM_ENABLE and TPM_DATA_CSUM are available in the *SPI* interface as already defined in [PTP 2.0 r1.05] for *I²C* interface. They are managed only on the current active locality.

4.5 Recommendations

For optimized performance results, the use of an extended data FIFO with 32-byte burst is mandatory.

When the falling edge occurs on the $\overline{\text{SPI\_CS}}$ signal, SPI_CLK must be at the low logic level as defined in Figure 12.

If the device does not show this behavior, add a 56 pF capacitor (example value) on the $\overline{\text{SPI\_CS}}$ line to slow the falling edge on $\overline{\text{SPI\_NSS}}$ or $\overline{\text{SPI\_CS}}$.

5 I²C interface protocol

5.1 TCG PC client options

The implemented protocol is compliant with the *PTP* specification (see [PTP 2.0 r1.05] for details) with the following options:

- Bus speed is limited to 1 Mbits/s.
- 10-bit target address is NOT supported.
- Target address reconfiguration is supported in the range of 0x08 to 0x77.
- Clock stretching is used.
- 1.8 V and 3.3 V are supported.
- Five localities are supported.
- Guard time is lower than 20 µs.
- The supported interrupts are “locality change” and “data available”.
- Burst count is dynamic. The maximum burst is 256 bytes.
- Data checksum is supported.
- Default *I²C* address used: 0x2E.

5.2 Platform constraints

This section explains the constraints on the ST33KTPM-IDevID *TPM* device.

5.2.1 I²C bus activities

As there is no line to select an *I²C* device, all the bus activities wake up the ST33KTPM-IDevID *TPM* device when it is in idle mode, and delay its response when it is in execution mode only if this is the TPM address.

To avoid loading the *I²C* bus and the ST33KTPM-IDevID, it is necessary to adapt the *TPM* response availability check to leave enough time for the device to process the commands. To do so, prefer interrupts on “data available” events when possible. If not possible, a polling mechanism with an incremental delay can also be used.

5.2.2 TPM acknowledgement

Sometimes, the ST33KTPM-IDevID *TPM* device is in low-power mode and cannot process an *I²C* transaction immediately. In this case, the transaction is not acknowledged. The host has to repeat it until the *TPM* device acknowledges the transaction.

When the ST33KTPM-IDevID *TPM* device is in low power mode, it requires about 80µs to wake up. For a clock at 1 MHz, the three first *TPM* accesses are *NACK* (with a guard time equal to 20 µs).

5.2.3 Clock stretching

In the case of a DRTM or H-CRTM sequence, the ST33KTPM-IDevID *TPM* uses clock stretching to control the hash data flow. Clock stretching is activated on the second frame following the hash data or the end of the hash. The duration can be up to 2 ms for the end of the hash.

6 Integration requirements

6.1 Integration information

- The duration of TPM_HASH_DATA exceeds 250 µs if SHA-384 and dual-bank management are implemented.
- In Windows® 10 or 11, for using a 3072-bit or 4096-bit *RSA* key, update the following, if it exists in the registry editor:
  For a 3072-bit *RSA* key:
  `[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\TPM]`
  `"TimeoutCommandCreate"=dword:0057E40`
  For a 4096-bit *RSA* key:
  `[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\TPM]`
  `"TimeoutCommandCreate"=dword:00AFC80 .`
- In linux, refer to tpm.h, under directory drivers/char/tpm, TPM2_DURATION_LONG_LONG (300000 ms by default), must be increased when using a 3072-bit or 4096-bit *RSA* key. The value is defined in Table 2.

Interrupt mode with Windows® could be activated under three conditions:

- PIRQ signal connected and driven by Host controller
- UEFI activate TPM Interrupt mode
- Add in Windows® registry editor (create item if does not exist):
  [HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\TPM]
  "EnableInterruptSupport"=dword:00000001

6.2 *TPM* Interrupt signal usage (PIRQ) driver implementation

PIRQ is the *TPM* interrupt signal.

The *TPM* host drivers should enable the PIRQ before using it with the TPM_INT_ENABLE_x or TPM_CRB_INT_ENABLE_x registers defined in [PTP 2.0 r1.05].

Once an interrupt is asserted, the *TPM* do not assert another interrupt until it receives a TPM_EOI (End_Of_Interrupt), even if new events that could cause an interrupt occur.

When an event occurs that causes the *TPM* to signal an interrupt (PIRQ signal falling edge), the *TPM* host must read the appropriate fields in the TPM_INT_STATUS_x or TPM_CRB_INT_STS_x register to determine the cause (for instance localityChangeIntOccured or dataAvailIntOccured), and take appropriate action.

After that event, the *TPM* host runs TPM_EOI through the TPM_INT_STATUS_x or the TPM_CRB_INT_STS_x register.

The *TPM* interrupt enablement sequence is:

Table 5. *TPM* interrupt enable sequence

Step number	Description
Step 1	The host driver reads the TPM_INT_ENABLE register to get the interrupt polarity. This *SPI* frame gives: MOSI: 0x83 0xD4 0x00 0x08 0x00 0x00 0x00 0x00 0x00 MISO: 0x00 0x00 0x00 0x00 0x01 0x08 0x00 0x00 0x00 TISREAD: locality 0: register 008 (TPM_INT_ENABLE_0): bytes number: 4: Data 08:00:00:00: Bit 3 set : typePolarity : Low level Default value for this register
Step 2	The host driver writes the TPM_INT_ENABLE register to enable the globalIntEnable and dataAvailIntEnable bits, and the interrupt polarity. This *SPI* frame gives: MOSI: 0x03 0xD4 0x00 0x08 [Flowcontrol] 0x09 0x00 0x00 0x80 MISO: 0x00 0x00 0x00 0x00 [Flowcontrol] 0xFF 0xFF 0xFF 0xFF

Step number	Description
	TISWRITE: locality 0: register 008 (TPM_INT_ENABLE_0): bytes number: 4: Data 09:00:00:80: Bit 3 set : typePolarity : Low level Bit 0 set : dataAvailIntEnable : Enabled Bit 31 set : globalIntEnable : Interrupts controlled by individual bits
Step 3	The host driver reads the TPM_INT_ENABLE register to check whether the write operation has been executed. This *SPI* frame gives: MOSI : 0x83 0xD4 0x00 0x08 0x00 0x00 0x00 0x00 0x00 MISO : 0x00 0x00 0x00 0x00 0x01 0x09 0x00 0x00 0x80 TISREAD: locality 0: register 008 (TPM_INT_ENABLE_0): bytes number: 4: Data 09:00:00:80: Bit 3 set : typePolarity : Low level Bit 0 set : dataAvailIntEnable : Enabled Bit 31 set : globalIntEnable : Interrupts controlled by individual bits [*TPM* state is in Command execution see figure 3 [PTP 2.0 r1.05]] Interrupt PIRQ is in Low level
Step 4	Check interrupt cause. MOSI: 0x80 0xD4 0x00 0x10 0x00 0x00 MISO: 0x00 0x00 0x00 0x00 0x01 0x01 TISREAD: locality 0: register 010 (TPM_INT_STATUS_0): bytes number: 1: Data 01: Bit 0 : dataAvailIntOccured : interrupt corresponds to “Data available”: *TPM* answer is ready to be read. [Get *TPM* answer]
Step 5	Send TPM_EOI This *SPI* frame gives: MOSI : 0x00 0xD4 0x00 0x10 [Flowcontrol] 0x01 0x01 MISO : 0x00 0x00 0x00 0x00 [Flowcontrol] 0x01 0xFF TISWRITE : locality 0: register 010 (TPM_INT_ENABLE_0): bytes number: 1: Data 01: Bit 0 set : dataAvailIntOccured : Writing a 1 to this field clears the dataAvailIntOccured interrupt.

7 TPM 2.0 library

7.1 Supported commands

The commands supported in the library mode *TPM* 2.0 are compliant with [TPM 2.0 P1 r159], [TPM 2.0 P2 r159], [TPM 2.0 P3 r159], [TPM 2.0 P4 r159], and [PTP 2.0 r1.05].

Table 6. Summary of supported commands

Commands		Support
Signals	_TPM_INIT	X
	_TPM_HashStart	X
	_TPM_HashData	X
	_TPM_HashEnd	X
Startup	TPM2_Startup	X
	TPM2_Shutdown	X
Testing	TPM2_IncrementalSelfTest	X
	TPM2_SelfTest	X
	TPM2_GetTestResult	X
Session commands	TPM2_StartAuthSession	X
	TPM2_PolicyRestart	X
Object commands	TPM2_Create	X
	TPM2_Load	X
	TPM2_LoadExternal	X
	TPM2_ReadPublic	X
	TPM2_ActivateCredential	X
	TPM2_MakeCredential	X
	TPM2_Unseal	X
	TPM2_CreateLoaded	X
	TPM2_ObjectChangeAuth	X
Duplicate commands	TPM2_Duplicate	X
	TPM2_Rewrap	X
	TPM2_Import	X
Asymmetric primitives	TPM2_RSA_Encrypt	X
	TPM2_RSA_Decrypt	X
	TPM2_ECDH_KeyGen	X
	TPM2_ECDH_ZGen	X
	TPM2_ECC_Parameters	X
	TPM2_ZGen_2Phase	X
Symmetric primitives	TPM2_EncryptDecrypt	X(1)
	TPM2_EncryptDecrypt2	X(1)
	TPM2_Hash	X
	TPM2_HMAC	X
Random number generator	TPM2_GetRandom	X

Commands		Support
Random number generator	TPM2_StirRandom	X
Hash/HMAC/Event sequences	TPM2_HMAC_Start	X
	TPM2_MAC_Start	-
	TPM2_HashSequenceStart	X
	TPM2_SequenceUpdate	X
	TPM2_SequenceComplete	X
	TPM2_EventSequenceComplete	X
Attestation commands	TPM2_Certify	X
	TPM2_CertifyCreation	X
	TPM2_Quote	X
	TPM2_GetSessionAuditDigest	X
	TPM2_GetCommandAuditDigest	X
	TPM2_GetTime	X
	TPM2_CertifyX509	X
Anonymous attestation	TPM2_Commit	X
	TPM2_ECC_Ephemeral	X
Signature verification	TPM2_VerifySignature	X
	TPM2_Sign	X
Command audit	TPM2_SetCommandCodeAuditStatus	X
Integrity collection (PCR)	TPM2_PCR_Extend	X
	TPM2_PCR_Event	X
	TPM2_PCR_Read	X
	TPM2_PCR_Allocate	X
	TPM2_PCR_SetAuthPolicy	-
	TPM2_PCR_SetAuthValue	-
	TPM2_PCR_Reset	X
Enhanced authorization (EA) commands	TPM2_PolicySigned	X
	TPM2_PolicySecret	X
	TPM2_PolicyTicket	X
	TPM2_PolicyOR	X
	TPM2_PolicyPCR	X
	TPM2_PolicyLocality	X
	TPM2_PolicyNV	X
	TPM2_PolicyCounterTimer	X
	TPM2_PolicyCommandCode	X
	TPM2_PolicyPhysicalPresence	X
	TPM2_PolicyCpHash	X
	TPM2_PolicyNameHash	X
	TPM2_PolicyDuplicationSelect	X
	TPM2_PolicyAuthorize	X
	TPM2_PolicyAuthValue	X

Commands		Support
Enhanced authorization (EA) commands	TPM2_PolicyPassword	X
	TPM2_PolicyGetDigest	X
	TPM2_PolicyTemplate	X
	TPM2_PolicyAuthorizeNV	X
	TPM2_PolicyNvWritten	X
Hierarchy commands	TPM2_CreatePrimary	X
	TPM2_HierarchyControl	X
	TPM2_SetPrimaryPolicy	X
	TPM2_Change_PPS	X
	TPM2_Change_EPS	X
	TPM2_Clear	X
	TPM2_ClearControl	X
	TPM2_HierarchyChangeAuth	X
Dictionary attack functions	TPM2_DictionaryAttackLockReset	X
	TPM2_DictionaryAttackParameters	X
Miscellaneous management functions	TPM2_PP_Commands	X
	TPM2_SetAlgorithmSet	-
Field upgrade	TPM2_FieldUpgradeStart	-(2)
	TPM2_FieldUpgradeData	-(3)
	TPM2_FirmwareRead	-
Context management	TPM2_ContextSave	X
	TPM2_ContextLoad	X
	TPM2_FlushContext	X
	TPM2_EvictControl	X
Clocks and timers	TPM2_ReadClock	X
	TPM2_ClockSet	X
	TPM2_ClockRateAdjust	X
Capability commands	TPM2_GetCapability	X
	TPM2_SetCapability	-(4)
	TPM2_TestParms	X
Nonvolatile storage	TPM2_NV_DefineSpace	X
	TPM2_NV_UndefineSpace	X
	TPM2_NV_UndefineSpaceSpecial	X
	TPM2_NV_ReadPublic	X
	TPM2_NV_Write	X
	TPM2_NV_Increment	X
	TPM2_NV_Extend	X
	TPM2_NV_SetBits	X
	TPM2_NV_WriteLock	X
	TPM2_NV_GlobalWriteLock	X
	TPM2_NV_Read	X

Commands		Support
Nonvolatile storage	TPM2_NV_ReadLock	X
	TPM2_NV_ChangeAuth	X
	TPM2_NV_Certify	X
Proprietary commands	TPM2_VendorCmdSetMode	X
	TPM2_VendorCmdSetCommandSet	X
	TPM2_VendorCmdSetCommandSetLock	X
	TPM2_VendorCmdRestoreEK	X
	TPM2_VendorCmdFieldUpgradeStart	X
	TPM2_VendorCmdFieldUpgradeData	X
	TPM2_VendorCmdGPIOConfig	X(1)
	TPM2_VendorCmdGetRandom2	X
	TPM2_VendorCmdGetRandom800_90B	X
	TPM2_VendorCmdChangeObjectDeletionAuth	X(1)

1. *Commands deactivated by default. See TPM command support configuration.*
2. *Handled by TPM2_VendorCmdFieldUpgradeStart*
3. *Handled by TPM2_VendorCmdFieldUpgradeData.*
4. *TPM command defined in Appendix B: Referenced documents*

7.2 Cryptographic support

Table 7. Cryptographic algorithm support table

Algorithm Name	Value	Support	Comment
TPM_ALG_ERROR	0x0000	-	-
TPM_ALG_RSA	0x0001	X	1024, 2048, 3072 and 4096 bits
TPM_ALG_SHA	0x0004	X	-
TPM_ALG_SHA1	0x0004	X	-
TPM_ALG_HMAC	0x0005	X	SHA-1, SHA-256, SHA-384, SHA-512 SHA3-256 and SHA3-384
TPM_ALG_AES	0x0006	X	128, 192, 256 bits
TPM_ALG_MGF1	0x0007	X	-
TPM_ALG_KEYEDHASH	0x0008	X	-
TPM_ALG_XOR	0x000A	X	-
TPM_ALG_SHA256	0x000B	X	-
TPM_ALG_SHA384	0x000C	X	-
TPM_ALG_SHA512	0x000D	X	-
TPM_ALG_NULL	0x0010	X	-
TPM_ALG_SM3_256	0x0012	-	-
TPM_ALG_SM4	0x0013	-	-
TPM_ALG_RSASSA	0x0014	X	-
TPM_ALG_RSAES	0x0015	X	-
TPM_ALG_RSAPSS	0x0016	X	-
TPM_ALG_OAEP	0x0017	X	-

Algorithm Name	Value	Support	Comment
TPM_ALG_ECDSA	0x0018	X	-
TPM_ALG_ECDH	0x0019	X	-
TPM_ALG_ECDAA	0x001A	X	-
TPM_ALG_SM2	0x001B	-	-
TPM_ALG_ECSCHNORR	0x001C	X	-
TPM_ALG_ECMQV	0x001D	-	-
TPM_ALG_KDF1_SP800_56A	0x0020	X	-
TPM_ALG_KDF2	0x0021	-	-
TPM_ALG_KDF1_SP800_108	0x0022	X	-
TPM_ALG_ECC	0x0023	X	ECC crypto support summary
TPM_ALG_SYMCIPHER	0x0025	X	-
TPM_ALG_CAMELLIA	0x0026	-	-
TPM_ALG_SHA3_256	0x0027	X	-
TPM_ALG_SHA3_384	0x0028	X	-
TPM_ALG_SHA3_512	0x0029	-	-
TPM_ALG_SHAKE128	0x002A	-	-
TPM_ALG_SHAKE256	0x002B	-	-
TPM_ALG_SHAKE256_192	0x002C	-	-
TPM_ALG_SHAKE256_256	0x002D	-	-
TPM_ALG_SHAKE256_512	0x002E	-	-
TPM_ALG_CMAC	0x003F	-	-
TPM_ALG_CTR	0x0040	X	-
TPM_ALG_OFB	0x0041	X	-
TPM_ALG_CBC	0x0042	X	-
TPM_ALG_CFB	0x0043	X	-
TPM_ALG_ECB	0x0044	X	-
TPM_ALG_CCM	0x0050	-	-
TPM_ALG_GCM	0x0051	-	-
TPM_ALG_KW	0x0052	-	-
TPM_ALG_KWP	0x0053	-	-
TPM_ALG_EAX	0x0054	-	-
TPM_ALG_EDDSA	0x0060	-	-
TPM_ALG_EDDSA_PH	0x0061	-	-
TPM_ALG_LMS	0x0070	-	For hybrid signature in field upgrade
TPM_ALG_XMSS	0x0071	-	-
TPM_ALG_KEYEDXOF	0x0080	-	-
TPM_ALG_KMACXOF128	0x0081	-	-
TPM_ALG_KMACXOF256	0x0082	-	-
TPM_ALG_KMAC128	0x0090	-	-
TPM_ALG_KMAC256	0x0091	-	-

Table 8. **ECC crypto support summary**

Name	Value	Support	Comments
TPM_ECC_NONE	0x0000	-	-
TPM_ECC_NIST_P192	0x0001	-	-
TPM_ECC_NIST_P224	0x0002	-	-
TPM_ECC_NIST_P256	0x0003	X	-
TPM_ECC_NIST_P384	0x0004	X	-
TPM_ECC_NIST_P521	0x0005	-	-
TPM_ECC_BN_P256	0x0010	X	Curve to support *ECDAA*
TPM_ECC_BN_P638	0x0011	-	Curve to support *ECDAA*
TPM_ECC_SM2_P256	0x0020	-	-
TPM_ECC_BP_P256_R1	0x0030	-	-
TPM_ECC_BP_P384_R1	0x0031	-	-
TPM_ECC_BP_P512_R1	0x0032	-	-
TPM_ECC_CURVE_25519	0x0040	-	Curve to support *EdDSA*
TPM_ECC_CURVE_448	0x0041	-	Curve to support *EdDSA*

7.3 PCR banks

The default configuration supports two banks of 24 *PCR*s:

- One for SHA256 *PCR*s
- One for SHA384 *PCR*s

TPM2_PCR_Allocate can be used to change the default bank configuration. It is possible to allocate two banks of 24 PCRs for each maximum hash size (one bank for SHA-384 and one bank for SHA-512).

If TPM2_PCR_Allocate attempts to allocate more than two banks, the *TPM* device returns the TPM_RC_NO_RESULT error.

7.4 Dictionary attack mitigation

All information about DAM is available in [TPM 2.0 P1 r159], section 19.8.2 Lockout Mode Configuration Parameters.

maxTries (*NV*) – The *TPM* is in Lockout mode as long as failedTries equals this value. When a new owner is installed, maxTries is set to its default value as specified in the relevant platform-specific specification.

recoveryTime (*NV*) – This value indicates, in seconds, the rate at which failedTries is decremented. This can be set to a large value (2^{32} - 1) which essentially inhibits automatic exit from Lockout mode. When a new owner is installed, this value is set to its default value as specified in the relevant platform-specific specification.

lockoutRecovery (*NV*) – This value indicates the delay in seconds between attempts to use lockoutAuth. The time delay only applies after an authorization failure using lockoutAuth. A value of zero indicates that a system reboot (TPM2_Startup(TPM_SU_CLEAR)) is required between lockout attempts.

The parameters maxTries, recoveryTime, and lockoutRecovery **are set with TPM2_DictionaryAttackParameters()**. This command requires authorization with lockoutAuth.

The DAM parameters have the following values:

- maxTries = 32 (dec)
- recoveryTime = 7200 (dec) seconds (2 hours)
- lockoutRecovery = 86400 (dec) seconds (24 hours)

7.5 Resource availability and capability

Table 9. Resource availability and capability

Capability name	Support	Description
TPM_PT_HR_TRANSIENT_MIN	5	The maximum number of transient objects that can be held in the *TPM* RAM.
TPM_PT_HR_PERSISTENT_MIN	7	The minimum number of persistent objects that can be held in the *TPM NVM*. This number was calculated using the following example allocations: • 3 slots intended for root keys (PPK, SRK, 1 *EK*) • 3 slots intended for OS/application usage • 1 slot intended for the platform hierarchy The *TPM* supports at least two persistent 3072-bit *RSA* keys.
TPM_PT_HR_LOADED_MIN	4	The maximum number of authorization sessions that can be held in the *TPM* RAM.
TPM_PT_ACTIVE_SESSIONS_MAX	64	The maximum number of authorization sessions that can be active concurrently.
TPM_PT_PCR_COUNT	24	The number of *PCR*s implemented in a bank.
TPM_PT_PCR_SELECT_MIN	3	The maximum number of bytes in a TPMS_PCR_SELECT.sizeOfSelect.
TPM_PT_NV_COUNTERS_MAX	0	The maximum number of *NV* indexes that are allowed to have the TPMA_NV_COUNTER attribute SET. The value zero indicates that there is no fixed maximum.
TPM_PT_NV_INDEX_MAX	2048	The maximum size of an *NV* index data area.
NV_MEMORY_SIZE	192 Kbytes	Maximum available size of *NVM* and objects in bytes. It can be split into: • 13 Kbytes allocated exclusively to objects to allow support for the minimum PTP requirements and *EK* certificates (7 objects, 6 counters and 2 *PIN*s). • 179 Kbytes: maximum size of *NVM* for user data (objects and *NV* indexes) (for instance, it can be any of the following with policy digest: 115 RSA4096 keys (SHA512), 151 RSA3072 keys (SHA384), 220 RSA2048 keys (SHA256), 568 *ECC NIST* P-256 keys (SHA256), 414 *ECC NIST* P-384 keys (SHA384) or 730 NV indexes of 138 bytes data size (SHA256).
TPM_PT_MEMORY	2	Indicates that the *NV* memory used for persistent objects is shared with the *NV* memory used for *NV* index values.
Maximum Hybrid NVM size	512 bytes	Maximum size in RAM for data of *NV* index with NV_Orderly attribute including metadata (16 bytes per NV Index).

Other *TPM* capabilities are available in Appendix A.3: TPM_CAP_TPM_PROPERTIES: TPM_PT (PT_FIXED).

7.6 TPM vendor capability

Table 10. cTPM_CAP_TPM_PROPERTIES values

Value	Property name	Comments(1)
0x105	cTPM_PT_MANUFACTURER	0x53544D20 // “STM”
0x106	cTPM_PT_VENDOR_STRING_1	The first 4 characters of the main part number in ASCII.(1)
0x107	cTPM_PT_VENDOR_STRING_2	The second next 4 characters of the main part number in ASCII.(1)
0x108	cTPM_PT_VENDOR_STRING_3	The third next 4 characters of main part number in ASCII.(1)
0x109	cTPM_PT_VENDOR_STRING_4	The fourth next 4 characters of main part number in ASCII.(1)
0x10A	cTPM_PT_VENDOR_TPM_TYPE	0x00000000
0x10B	cTPM_PT_FIRMWARE_VERSION_1	TPM FW version(1)
0x10C	cTPM_PT_FIRMWARE_VERSION_2	0x00000000
0x12D	cTPM_PT_MODES	Flags that indicate if FIPS mode is activated (bit 0).

1. *Refer to Table 84. TPM_CAP_TPM_PROPERTIES: TPM_PT (PT_FIXED) for the values.*

Table 11. cTPM_CAP_COMMANDS values

Vendor commands	Command index	NV	Extensive	Flushed	cHandles	rHandles	V
TPM2_VendorCmdSetMode	0x307	1	0	0	1	0	1
TPM2_VendorCmdSetCommandSet	0x309	1	0	0	1	0	1
TPM2_VendorCmdRestoreEK	0x30A	1	1	0	1	0	1
TPM2_VendorCmdSetCommandSetLock	0x30B	1	0	0	1	0	1
TPM2_VendorCmdFieldUpgradeStart	0x30C	1	0	0	1	0	1
TPM2_VendorCmdFieldUpgradeData	0x30D	1	0	0	0	0	1
TPM2_VendorCmdGetRandom2	0x30E	0	0	0	0	0	1
TPM2_VendorCmdGPIOConfig	0x30F	1	0	0	1	0	1
TPM2_VendorCmdChangeObjectDeletionAuth	0x310	1	0	0	1	0	1
TPM2_VendorCmdGetRandom800_90B	0x311	0	0	0	0	0	1

Table 12. cTPM_CAP_VENDOR_PROPERTY values

Value	Capability name	Description
0x100	cTPM_CAP_VENDOR_PROPERTY	Output information listed below (count/start/vendorSpecificSize/vendorSpecific[])

cTPM_CAP_VENDOR_PROPERTY

```
tU32      count;
tU32      start;
tU16      vendorSpecificSize;
tU8       vendorSpecific[vendorSpecificSize];
```

Table 13. cTPM_SUBCAP_VENDOR_INTERNAL_DATA

Type	Name	Description
UINT32	count	`0x00000001` (only 1 property is provided)
UINT32	start	`cTPM_SUBCAP_VENDOR_INTERNAL_DATA (0x00000000)` (nor property starting point, useless)
UINT16	vendorSpecificSize	The dynamic size of the vendor specific area
BYTE[vendorSpecificSize]	vendorSpecific	Vendor-specific information: • DigestFactoryLength (1 byte) • DigestFactory (48 bytes) • DigestCurrent *TPM* instance 1 length (1 byte) • DigestCurrent *TPM* instance 1 (48 bytes) • DigestCurrent *TPM* instance 2 length (1 byte) • DigestCurrent *TPM* instance 2 (48 bytes) • DigestKeyLength (1 byte) • DigestKey (48 bytes) (digest (SHA384) of the public key (*ECC* or *LMS*) used to authenticate flash images) • *TPM* instance 1 checksum (4 bytes) • *TPM* instance 2 checksum (4 bytes) • HW code (4 bytes) • Configuration management node (4 bytes) • *TPM* upgrade counter (4 bytes) • *TPM* upgrade counter limit (4 bytes) • *TPM* failure counter (4 bytes) • *TPM* failure counter limit (4 bytes) • Inactive *TPM* instance version (4 bytes) • Inactive *TPM* instance configuration management node (4 bytes) • *TPM* library version (4 bytes) • HWINTF library version (4 bytes) • FactoryFWDowngradeAbility (1 byte): (0x0D: locked for product configuration) • Active *TPM* instance (1 byte): (Indicates the active *TPM* instance: 0 (instance 1), 1 (instance 2)). Instance 1 is active with the default factory configuration when both instances are loaded with the same version firmware. *MSB* set if *TPM* firmware autorecovery has been executed.

Table 14. cTPM_SUBCAP_VENDOR_LOW_POWER_MODE

Type	Name	Description
UINT32	count	`0x00000001` (only 1 property is provided)
UINT32	start	`cTPM_SUBCAP_VENDOR_LOW_POWER_MODE (0x00000001)` (nor property starting point, useless)
UINT16	vendorSpecificSize	Dynamic size of the vendor-specific area.
BYTE[vendorSpecificSize]	vendorSpecific	Data related to the current power-mode state: RFU (1 byte) RFU (1 byte) LowPowerAuto(1 byte): • 1: Active • 0: Not active BootToLowPowerTimeout (1 byte): 0..255 seconds CmdToLowPowerTimeout (1 byte): 0..255 seconds modeLock(1 byte): Mask indicating which mode is locked.

Table 15. cTPM_SUBCAP_VENDOR_GET_PRODUCT_INFO

Type	Name	Description
UINT32	count	0x00000001 (only 1 property is provided)
UINT32	start	cTPM_SUBCAP_VENDOR_GET_PRODUCT_INFO (0x00000003)
UINT16	vendorSpecificSize	104
BYTE[vendorSpecificSize]	vendorSpecific	• Unique serial number (7 bytes) • Pad 0 (1 byte) • Product identification number (PIN) (2 bytes) • Master product identification number (MPIN) (2 bytes) • Product internal revision (1 byte) • Pad 0 (3 bytes) • Firmware kernel version (4 bytes) • Pad 0 (3 bytes) • External Chameleon (3 bytes) • Pad 0 (74 bytes) • Cryptographic library version (4 bytes)

Table 16. cTPM_SUBCAP_VENDOR_GET_GPIO_CONFIG_INFO

Type	Name	Description
UINT32	count	0x00000001 (only 1 property is provided)
UINT32	start	cTPM_SUBCAP_VENDOR_GET_GPIO_CONFIG_INFO (0x00000004)
TPML_GPIO_CONFIG	vendorSpecific	Count (4 bytes) Followed by count TPMS_GPIO_CONFIG structures: GPIO name (4 bytes) GPIO NV index (4 bytes) GPIO mode (4 bytes)

Table 17. cTPM_SUBCAP_VENDOR_GET_FW_HASH

Type	Name	Description
UINT32	count	0x00000001 (only 1 property is provided)
UINT32	start	cTPM_SUBCAP_VENDOR_GET_FW_HASH (0x00000005)
UINT16	vendorSpecificSize	48
BYTE[vendorSpecificSize]	vendorSpecific	SHA384 of a set of CRC16 computed over sets of 256 bytes of the active FW instance

Table 18. cTPM_SUBCAP_VENDOR_GET_SETCMDLIST

Type	Name	Description
UINT32	count	0x00000001 (only 1 property is provided)
UINT32	start	cTPM_SUBCAP_VENDOR_GET_SETCMDLIST (0x00000006)
UINT16	vendorSpecificSize	The dynamic size of the vendor-specific area
BYTE[vendorSpecificSize]	vendorSpecific	List of commands (represented by their command codes) managed by TPM2_SetCommandSet with the states of the three internal flags for each of them: For each command managed by TPM2_SetCommandSet: Command code of the command managed by TPM2_SetCommandSet (4 bytes) Activated flag state (1 byte): Activated (1), deactivated (0) DisableLock flag state (1 byte): Locked by TPM2_SetCommandSet (1), unlocked by TPM2_SetCommandSet (0) DisableLockClear flag state (1 byte): DisableLock flag enabled (0), DisableLock flag unlocked irreversibly by TPM2_VendorCmdSetCommandSetLock (1)

Table 19. **cTPM_SUBCAP_VENDOR_TPMA_MODES**

Type	Name	Description
UINT32	count	0x00000001 (only 1 property is provided)
UINT32	start	cTPM_SUBCAP_VENDOR_TPMA_MODES (0x00000007)
UINT16	vendorSpecificSize	The dynamic size of the vendor specific area
BYTE[vendorSpecificSize]	vendorSpecific	UINT32 used to add data to TPMA_MODES structure reported by TPM using TPM2_GetCapability(capability=TPM_CAP_TPM_PROPERTIES, property = TPM_PT_MODES) as defined in the [TCG FIPS L1]: • Bit 0 : FIPS_140_2 bit support • Bit 1 : FIPS_140_3 bit support • Bits 3:2 : FIPS_140_3_INDICATOR • Bits 5:4 : FIPS_140_3_CUMULATIVE_INDICATOR • Bit 7:6 : FIPS_140_3 level mode – Level 1 (0x00) – Level 2 (0x01) – Level 3 (0x02) – Level 4 (0x03) • Bit 8 : FIPS_140_3 level lock information: – Unlocked (0x00) – Locked (0x01) • Bits 31:9 : RFU

7.7 TPM2_CreatePrimary

According to the TPM2.0 specification [TPM 2.0 P1 r159], it is critical that the *TPM* manufacturer makes sure that the injected key pairs are only associated with a single template. The *TPM* device meets this requirement thanks to an association between the injected *RSA EK* key pairs during manufacturing, and the default template defined in [TCG EK Cre Profile TPM 2.3].

When the public area is different from the default template, or after execution of the `TPM2_ChangeEPS` command, the process to create a primary key uses the standard derivation from the endorsement primary seed (eps) based on a *DRBG* function.

Note that the protection profile [TPM 2.0 PP] restricts primary key objects to *RSA* 2048-bit key, *ECC* 256, and *AES* 128 as a minimum.

7.8 Deterministic random number generation (DRNG) implementation options

The *TPM* device generates random numbers by processing the TPM2_GetRandom command defined in the TPM2.0 library. Random numbers are generated from a SHA256 deterministic random bit generator (*DRBG*) (SP800-90A compliant) automatically seeded (Instantiate SP800-90A function) with a true random number generator (*TRNG*) (SP800-90B and AIS-31 PTG.2 class compliant). The Hash-*DRBG* (*HDRBG*) seeding operation is carried out afterwards:

- Reset of each *TPM* device
- A reseed period equal to 2^{20} generations

The SHA256 *DRBG* could also be reseeded on demand by processing the TPM2_StirRandom command defined in the TPM2.0 library that implements the reseed operation as defined in SP800-90A. Reseed is performed by combining a fresh *TRNG* output, the chip unique identifier and some user-defined data (limited to 128 bytes) passed as input to TPM2_StirRandom.

Standard command TPM2_GetRandom provides random bytes limited to a fixed length (32 bytes) whereas proprietary command Section 9.4.2: GetRandom2 provides a variable length of random data.

The *DRBG* provides a security strength of 256 bits (according to SP800-57) suitable for generating AES-256 keys and ECC P-521 or higher keys.

8 TPM 2.0 endorsement key certificate support

STMicroelectronics manages its own certificate authority (*CA*) infrastructure to issue *EK* certificates.

There are two chains of certificates from the *EK* certificate up to the STSAFE root certificate:

- One based on *RSA* cryptography.
- The other based on *ECC* cryptography.

This product includes four *EK*s and four *EK* certificates that do not expire:

- A 2048-bit *RSA* key and a certificate signed with an ST intermediate CA using a 4096-bit *RSA* key.
- An *ECC NIST* P-256 key and a certificate signed with an ST intermediate *CA* using an *ECC* NIST P-384 bit key.
- An *ECC NIST* P-384 key and a certificate signed with an ST intermediate *CA* using an *ECC NIST* P-384 bit key.

Figure 5. RSA certificate authority structure

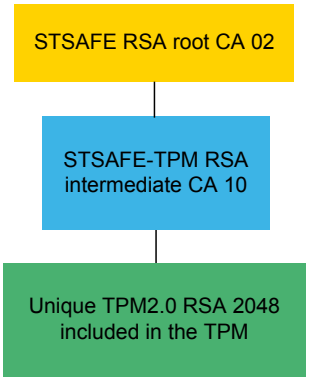
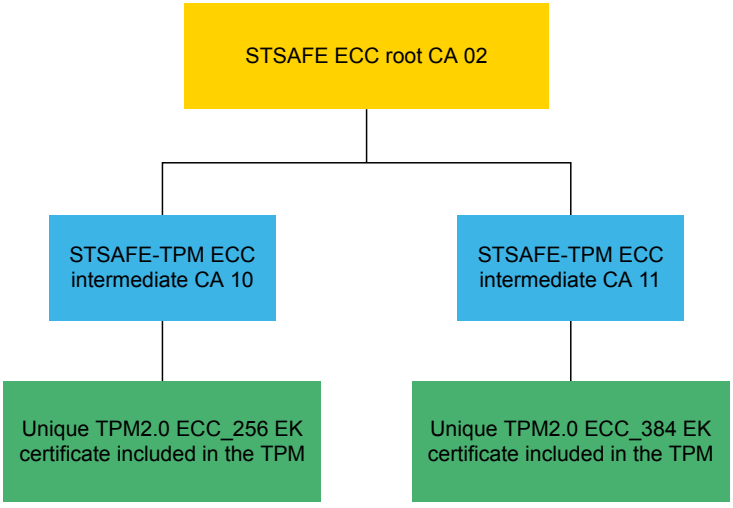


Figure 6. ECC certificate authority structure



8.1 *TPM* 2.0 *RSA* 2048 *EK*

This section and subsections provide information on the *TPM* 2.0 *RSA* 2048 *EK* template and certificate.

8.1.1 *TPM* 2.0 *RSA* 2048 *EK* template

The default template for the *TPM* 2.0 *RSA* 2048 *EK* is defined in [TCG EK Cre Profile TPM 2.3], Table 2: Default EK Template (TPMT_PUBLIC) L-1: RSA 2048 (Storage).

8.1.2 *TPM* 2.0 *RSA* 2048 *EK* certificate

nvIndex

For a certificate computed for the 2048-bit RSA key the index value is **0x01C00002**.

Index payload

Table 20. Certificate NV index payload

Name	Type	L	Value
cert	Byte[]	X	The certificate as defined in [TCG EK Cre Profile TPM 2.3]. The certificate size X is 1299 bytes.

Intermediate *CA*

The intermediate *CA* required to validate the *RSA EK* certificate can be retrieved from this URL: http://sw-center.st.com/STSAFE/stsafetpmrsaint10.crt.

8.2 *TPM* 2.0 *ECC NIST* P-256 *EK*

This section and subsections provide information on the *TPM* 2.0 *ECC NIST* P-256 *EK* template and certificate.

8.2.1 *TPM* 2.0 *ECC NIST* P-256 *EK* template

The default template for the *ECC NIST* P-256-bit key is defined in [TCG EK Cre Profile TPM 2.3], Table 3: Default EK Template (TPMT_PUBLIC) L-2: ECC NIST P256.

8.2.2 *TPM* 2.0 *ECC NIST* P-256 *EK* certificate

nvIndex

For a certificate computed for an *ECC NIST* P-256-bit key the index value is **0x01C0000A**.

Index payload

Table 21. Certificate NV index payload

Name	Type	L	Value
cert	Byte[]	X	The certificate as defined in [TCG EK Cre Profile TPM 2.3]. The certificate size X is maximum 680 bytes long which is the most frequent size. Certificates with lower sizes, such as 679, 678, and lower, may be observed with lower occurrences. This certificate length variability is due to the randomness of the *ECC* signature values and the ASN.1 formatting rules leading to signature fields with variable lengths.

Intermediate *CA*

The intermediate *CA* required to validate the *ECC EK* certificate can be retrieved from this URL: http://sw-center.st.com/STSAFE/stsafetpmeccint10.crt.

8.3 *TPM* 2.0 *ECC NIST* P-384 *EK*

This section and subsections provide information on the *TPM* 2.0 *ECC NIST* P-384 *EK* template and certificate.

8.3.1 TPM 2.0 *ECC NIST* P-384 *EK* template

The default template for the *ECC NIST* P-384-bit key is defined in [TCG EK Cre Profile TPM 2.3], Table 6: Default EK Template (TPMT_PUBLIC) H-3: ECC NIST P384.

8.3.2 TPM 2.0 *ECC NIST* P-384 *EK* certificate

nvIndex

For a certificate computed for an *ECC NIST* P-384-bit key the index value is **0x01C00016**.

Index payload

Table 22. Certificate NV index payload

Name	Type	L	Value
cert	Byte[]	X	The certificate as defined in [TCG EK Cre Profile TPM 2.3]. The certificate size X is maximum 709 bytes which is the most frequent size. Certificates with lower sizes, such as 708, 707, and lower, may be observed with lower occurrences. This certificate length variability is due to the randomness of the *ECC* signature values and the ASN.1 formatting rules leading to signature fields with variable lengths.

Intermediate *CA*

The intermediate *CA* required to validate the *ECC EK* certificate can be retrieved from this URL: http://sw-center.st.com/STSAFE/stsafetpmeccint11.crt.

9 Proprietary commands

9.1 Mode selection

9.1.1 Purpose

This section describes the commands used by the platform firmware to configure the expected device behavior.

The following requirements are covered by this specification:

- Ability to configure the Low-power mode

9.1.2 TPM2_VendorCmdSetMode command description

TPM2_VendorCmdSetMode command authorization

The TPM2_VendorCmdSetMode command is authorized by platform authorization or the owner authorization.

Caution: *This command must be processed during platform provisioning in the pre-OS environment (before loading the OS) by the OEM platform firmware. Indeed, platforms for which a password was defined for AuthValue (as specified in [TPM 2.0 P2 r159]) during boot time cannot use the TPM2_VendorCmdSetMode command at OS level. STMicroelectronics provides UEFI and BIOS vendors with a dedicated support for efficient implementation.*

TPM2_VendorCmdSetMode ordinal definition

The table below shows the ordinal definition of the TPM2_VendorCmdSetMode command.

Table 23. Definition of (UINT32) TPM_CC constants (numeric order) <IN/OUT, S>

Name	Command Code	Nonvolatile write	Decrypt	Encrypt	Comment
TPM_CC_VendorCmdSetMode	0x20000307	Y	-	-	-

TPM2_VendorCmdSetMode command parameter definition

Table 24. TPM2_VendorCmdSetMode command

Type	Name	Description
TPMI_ST_COMMAND_TAG	tag	TPM_ST_SESSIONS
UINT32	commandSize	-
TPM_CC	commandCode	TPM_CC_VendorCmdSetMode
TPMI_RH_PROVISION	@authorization	TPMI_RH_PROVISION (OWNER + PLATFORM). Auth Index: 1 Auth Role: USER
TPMS_MODE_SET	modeSet	Bitmap defining the target mode the ST33KTPM-IDevID must switch to.

Table 25. TPM2_VendorCmdSetMode response

Type	Name	Description
TPM_ST	tag	-
UINT32	responseSize	-
TPM_RC	responseCode	-

9.1.3 TPM2_VendorCmdSetMode structure definition

Table 26. TPMS_mode_set

Type	Name	Description
BYTE	CmdToLowPower	Timer (s) between SPI activity and low power (relevant if TPMLowPowerAuto is enabled).
BYTE	BootToLowPower	Timer (s) between boot and first low power (relevant if TPMLowPowerAuto is enabled).
BYTE	modeLock	See Table 28.
BYTE	mode	See Table 27.

Table 27. Mode parameter

b7	b6	b5	b4	b3/b2	b1	b0	Low-power configuration
-	-	-	-	-	-	x	RFU
-	-	-	-	-	-	x	RFU
-	-	-	-	-	0	-	Standby activated during reset(1)
-	-	-	-	-	1	-	RFU
-	-	-	-	00	-	-	TPMLowPower (disabled)
-	-	-	-	01	-	-	Reserved
-	-	-	-	10	-	-	Reserved
-	-	-	-	11	-	-	TPMLowPowerAuto (enabled) (1)

1. *Default value.*

Table 28. modeLock parameter

b7	b6	b5	b4	b3/b2	b1	b0	Mode configuration
-	-	-	-	00	-	-	TPMLowPowerLock CLEAR (1)
-	-	-	-	x1	-	-	TPMLowPowerLock SET
-	-	-	-	1x	-	-	TPMLowPowerLock SET
-	-	-	-	-	1	-	RFU

1. *Default value.*

The modeLock parameter of the command can permanently lock the low power mode configuration. Any further modification of the configuration is ignored.

9.2 *TPM* field upgrade commands

This section details the formats of the commands used for updating the *TPM* firmware (explained in Field upgrade).

9.2.1 Command format

TPM2_VendorCmdFieldUpgradeStart

```
#define TPM_CC_VendorCmdFieldUpgradeStart                0x2000030C
```

Table 29. TPM2_VendorCmdFieldUpgradeStart command

Type	Name	Description
TPMI_ST_COMMAND_TAG	tag	TPM_ST_SESSIONS
UINT32	commandSize	Size of the command
TPM_CC	commandCode	TPM_CC_VendorCmdFieldUpgradeStart
TPMI_RH_PLATFORM	@authorization	TPM_RH_PLATFORM Auth Index:1 Auth Role: USER
TPM_ST_FU_BLOBZERO	blob	Blob of type 0x0 (see Type 0x00 blob)

Table 30. TPM2_VendorCmdFieldUpgradeStart response

Type	Name	Description
TPM_ST	tag	TPM_ST_SESSIONS
UINT32	responseSize	Size of the response
TPM_RC	responseCode	TPM_RC_SUCCESS if command has been successfully executed

TPM2_VendorCmdFieldUpgradeData

```
#define TPM_CC_VendorCmdFieldUpgradeData                 0x2000030D
```

Table 31. TPM2_VendorCmdFieldUpgradeData command

Type	Name	Description
TPMI_ST_COMMAND_TAG	tag	TPM_ST_NO_SESSIONS
UINT32	commandSize	-
TPM_CC	commandCode	TPM_CC_VendorCmdFieldUpgradeData
TPM_ST_FU_BLOBDATA	blob	Blob of type 0x01 (see Figure 9) / 0xFF (see Figure 10)

Table 32. TPM2_VendorCmdFieldUpgradeData response

Type	Name	Description
TPM_ST	tag	TPM_ST_NO_SESSIONS
UINT32	responseSize	Size of the response
TPM_RC	responseCode	TPM_RC_SUCCESS if command has been successfully executed
UINT32	fuStatus	Indicates the status of the field upgrade session: 0x00000000 (on-going), 0x000000FF (finished)

Command authorization

The Auth Role in the TPM2_VendorCmdFieldUpgradeStart command means that the user allows the command to be used in a password, *HMAC* or policy session.

The TPM2_VendorCmdFieldUpgradeData command has no authorization.

9.2.2 Command sequences

In order to be able to complete a field upgrade successfully, the following sequence of commands has to be sent to the *TPM* device:

1. A session must be opened:
   A password session or TPM2_StartAuthSession with a *HMAC* or policy session
2. In the case of a policy session, execute the expected sequence of policy commands.
3. Send TPM2_VendorCmdFieldUpgradeStart using data from the type 0x0 blob of the flashable image and with the proper authorization.
4. Send *n* TPM2_VendorCmdFieldUpgradeData commands using the different blobs (type 0x01 / 0xFF) from the flashable image sequentially. When output indication `fuStatus` is equal to 0xFF, the field upgrade process is finished.
5. Restart the *TPM* device to complete the field upgrade.

9.3 TPM command support configuration

9.3.1 Purpose

TPM command support configuration is used to activate or deactivate a set of *TPM* commands.

TPM command support configuration is based on the processing of two commands:
TPM2_VendorCmdSetCommandSet and TPM2_VendorCmdSetCommandSetLock.

1. TPM2_VendorCmdSetCommandSet offers the ability to:
   - Activate or deactivate at least one command affected by TPM2_VendorCmdSetCommandSet
   - Make the change persistent after reset
   - Lock the state (activation/deactivation) of the command irreversibly.

   Commands affected by TPM2_VendorCmdSetCommandSet:
   - TPM2_ChangeEPS
     By default, this command is activated.
   - TPM2_VendorCmdChangeObjectDeletionAuth
     By default, this command is deactivated.
   - TPM2_EncryptDecrypt
     By default, this command is deactivated.
   - TPM2_EncryptDecrypt2
     By default, this command is deactivated.
   - TPM2_VendorCmdRestoreEK
     By default, this command is activated.
   - TPM2_VendorCmdGPIOConfig
     By default, this command is deactivated.
   - By default, this command is deactivated.
2. TPM2_VendorCmdSetCommandSetLock: gives the ability to:
   Irreversibly lock the ability to lock a command activation/deactivation.
   Command affected by TPM2_VendorCmdSetCommandSetLock:
   - TPM2_VendorCmdSetCommandSet

TPM2_EncryptDecrypt and TPM2_ChangeEPS could be irreversibly deactivated by using the TPM2_VendorCmdSetCommandSet command during platform provisioning for example.

If the TPM2_EncryptDecrypt and TPM2_ChangeEPS commands must remain reversible, TPM2_VendorCmdSetCommandSetLock must be used during platform provisioning to remove the irreversible lock functionality from TPM2_VendorCmdSetCommandSet.

9.3.2 TPM2_VendorCmdSetCommandSet command description

Command authorization

TPM2_VendorCmdSetCommandSet reuses the authorization defined in the library specifications.
The TPM2_VendorCmdSetCommandSet command is authorized:

- By platform authorization

Impacted commands

Table 33. List of de/activable command codes

Logical command	Command Code	Default state
TPM2_ChangeEPS	0x00000124	Activated
TPM2_EncryptDecrypt	0x00000164	Deactivated
TPM2_EncryptDecrypt2	0x00000193	Deactivated
TPM2_VendorCmdRestoreEK	0x2000030A	Activated
TPM2_VendorCmdGPIOConfig	0x2000030F	Deactivated
TPM2_VendorCmdChangeObjectDeletionAuth	0x20000310	Deactivated

Ordinal definition

Table 34. Definition of (UINT32) TPM_CC constants (numeric order) <IN/OUT, S>

Name	Command Code	Nonvolatile write	Decrypt	Encrypt	Comments
TPM_CC_VendorCmdSetCommandSet	0x20000309	Y	-	-	-

Command parameter definition

Table 35. TPM2_VendorCmdSetCommandSet command

Type	Name	Description
TPMI_ST_COMMAND_TAG	`tag`	TPM_ST_SESSIONS
UINT32	`commandSize`	-
TPM_CC	`commandCode`	TPM_CC_VendorCmdSetCommandSet
TPMI_RH_PLATFORM	`@authorization`	TPM_RH_PLATFORM Auth Index: 1 Auth Role: USER
UINT32	`CommandCode`	Command to be activated or deactivated depending on SetClearFlag.
UINT32	`SetClearFlag`	Clear (0): command must be deactivated. Set (1): command must be activated.
UINT32	`Lock`	Clear (0): command state (activated or deactivated) is not locked. Set (1): command state (activated or deactivated) must be irreversibly locked.

Table 36. **TPM2_VendorCmdSetCommandSet response**

Type	Name	Description
TPM_ST	`tag`	-
UINT32	`responseSize`	-
TPM_RC	`responseCode`	-

Command information

If the SetClearFlag value differs from 0 and 1, *TPM* returns error code RC_VALUE.

If the command code value is not in the list of the allowed values listed in Table 33. List of de/activable command codes, the product returns RC_VALUE.

If `commandCode` corresponds to TPM2_EncryptDecrypt, the same action (activation/deactivation) is performed on TPM2_EncryptDecrypt2. Likewise, if `commandCode` corresponds to TPM2_EncryptDecrypt2, the same action (activation/deactivation) is performed on TPM2_EncryptDecrypt.

The new configuration is saved to the nonvolatile area and so is persistent after reset.

If the *Lock* value is SET (1), then according to the SetClearFlag value, the concerned command is irreversibly locked (the command may be definitively activated or deactivated).

If TPM2_VendorCmdSetCommandSet is executed for an already locked command (activated or deactivated command), the device returns RC_FAILURE.

If TPM2_VendorCmdSetCommandSetLock is applied, and then TPM2_VendorCmdSetCommandSet is applied with input parameter Lock set to SET (1), the *TPM* device returns TPM_RC_DISABLED.

If TPM2_VendorCmdSetCommandSetLock is applied, and then TPM2_VendorCmdSetCommandSet is applied with input parameter Lock set to CLEAR (0), the *TPM* devices processes SetClearFlag and CommandCode, and then activates or deactivates command support in persistent memory.

9.3.3 TPM2_VendorCmdSetCommandSetLock command description

Command authorization

TPM2_VendorCmdSetCommandSetLock reuses the authorization defined in the library specifications.

The TPM2_VendorCmdSetCommandSetLock command is authorized:

- By platform authorization

Impacted commands

Table 37. **List of impacted command codes**

Logical command	Command Code	Default state
TPM2_VendorCmdSetCommandSetLock	0x20000309	Activated

Command information

The device processes this command and irreversibly deactivates the effect of the Lock parameter in the TPM2_VendorCmdSetCommandSet command. In this case, it is no more possible to irreversibly activate/deactivate a command.

This command should not be used during integration tests, but only in the production environment. The firmware upgrade does not modify the irreversible lock. If the TPM2_VendorCmdSetCommandSetLock command has been executed, the additional commands added by firmware upgrade cannot be activated or deactivated irreversibly anymore.

Ordinal definition

Table 38. Definition of (UINT32) TPM_CC constants (numeric order) <IN/OUT, S>

Name	Command Code	Nonvolatile write	Decrypt	Encrypt	Comments
TPM_CC_VendorCmdSetCommandSetLock	0x2000030B	Y	-	-	-

Command parameter definition

Table 39. TPM2_VendorCmdSetCommandSetLock command

Type	Name	Description
TPMI_ST_COMMAND_TAG	tag	TPM_ST_SESSIONS
UINT32	commandSize	-
TPM_CC	commandCode	TPM_CC_VendorCmdSetCommandSetLock
TPMI_RH_PLATFORM	@authorization	TPM_RH_PLATFORM • Auth Index: 1 • Auth Role: USER

Table 40. TPM2_VendorCmdSetCommandSetLock response

Type	Name	Description
TPM_ST	tag	-
UINT32	responseSize	-
TPM_RC	responseCode	-

9.3.4 Command state transitions for TPM2.0

The following table shows the state transitions for `TPM2_SetCommandSet`.

Table 41. Summary of TPM2.0 command state transitions

Previous command executed	No `TPM2_SetCommandSetLock` executed				`TPM2_SetCommandSetLock` applied first	
Current action: `TPM2_SetCommandSet`	`Lock` value is CLEAR		`Lock` value is SET		`Lock` value is CLEAR	`Lock` value is SET
	`SetClearFlag` is SET	`SetClearFlag` is CLEAR	`SetClearFlag` is SET	`SetClearFlag` is CLEAR	`SetClearFlag` is SET or CLEAR	
Next state of command support in persistent memory	Activates command support	Deactivates command support	Activates command support	Deactivates command support	Activates or deactivates command support.	No change
Next state for `Lock` value of command support	No change Command support could still be activated or deactivated.		Command support is irreversibly activated.	Command support is irreversibly deactivated.	-	No change.
-	Default state This state should not be kept in the field to avoid denial of service.		State for commands that must be irreversibly activated or deactivated.		State of commands for which the activation must stay configurable after platform provisioning, but must not be vulnerable to "Denial of Service" due to the disabling of "irreversible locking".	

9.4 Deterministic random number generator

9.4.1 Purpose

The TPM2_GetRandom2 command returns the next requested bytes from the deterministic random number generator (DRBG). The output is limited to the maximum size defined for the buffer.

All these requirements are supported by one proprietary command: TPM2_GetRandom2.

9.4.2 TPM2_VendorCmdGetRandom2 command description

Command authorization

TPM2_VendorCmdGetRandom2 reuses the authorization definition of TPM2_GetRandom defined in the library specifications.

Thus, the TPM2_VendorCmdGetRandom2 command does not require any authorization.

Command processing

If bytesRequested is longer than TPM2B_MAX_BUFFER (see Appendix A.3: TPM_CAP_TPM_PROPERTIES: TPM_PT (PT_FIXED) for TPM2B_MAX_BUFFER.size) can accommodate, no error is returned but the TPM returns as much data as a TPM2B_DATA buffer can contain.

Ordinal definition

Table 42. Definition of (UINT32) TPM_CC constants (numeric order) <IN/OUT, S>

Name	Command Code	Nonvolatile write	Decrypt	Encrypt	Comments
TPM_CC_VendorCmdGetRandom2	0x2000030E	N	-	Y	-

Command parameter definition

Table 43. TPM2_VendorCmdGetRandom2 command

Type	Name	Description
TPMI_ST_COMMAND_TAG	tag	TPM_ST_SESSIONS if an audit or encrypt session is present; otherwise, TPM_ST_NO_SESSIONS
UINT32	commandSize	-
TPM_CC	commandCode	TPM_CC_VendorCmdGetRandom2
UINT16	bytesRequested	Number of bytes to return

Table 44. TPM2_VendorCmdGetRandom2 response

Type	Name	Description
TPM_ST	tag	-
UINT32	responseSize	-
TPM_RC	responseCode	-
TPM2B_MAX_BUFFER	randomBytes	The random bytes

9.5 GPIO support

9.5.1 Purpose

The proprietary command called TPM2_VendorCmdGPIOConfig is used to configure the general-purpose inputs/outputs (*GPIO*s).

9.5.2 TPM GPIO support

The *TPM GPIO* feature can be used to configure the *GPIO*s of a device by using an *NV* index.

The value of the *NV* index and the level of a *GPIO* defined as output are linked together by the *TPM* once the *GPIO* configuration and the *NV* index have been defined. It is then possible to control the *GPIO* level of the output pin according to the access rights to the *NV* index. This is a way of indicating a specific *TPM* state to an external device.

The *TPM GPIO*s are by default configured to support the *GPIO*s required by the *PTP* and the optional physical presence (*PP*) pin.

It is possible to keep a *TCG*-compliant *TPM* by reconfiguring the *PP* function pin. This pin can be used as *GPIO*.

When this preconfigured I/O is reconfigured as a *GPIO*, the related function is no more available.

9.5.3 TPM2_VendorCmdGPIOConfig command description

General description

TPM2_VendorCmdGPIOConfig is a proprietary command used to configure a *GPIO*. This command can be sent several times. The *GPIO* configuration is modified after the successful execution of the command.

Platform authorization is required for this command.

[PTP 2.0 r1.05] reserves a specific range of *NV* index that can be used to map a *GPIO* to an *NV* index.

These *GPIO NV* indexes can be created, deleted, read or written thanks to standard TPM2.0 commands such as TPM2_NV_DefineSpace, TPM2_NV_UndefineSpace, TPM2_NV_Read or TPM2_NV_Write (see Using the standard *TPM* commands to control the GPIOs for more details).

The proprietary TPM2_VendorCmdGPIOConfig command allows the *NV* index handle and *GPIO* mode linked to a specific *GPIO* to be configured.

If the given *NV* index handle is not in the defined range, that is between 0x01C4 0000 and 0x01C4 000F, the *TPM* device returns TPM_RC_VALUE. If the given *NV* index is already defined in *NV* memory, the *TPM* returns TPM_RC_NV_DEFINED. If the given *NV* index is already linked to another *GPIO*, the *TPM* device returns TPM_RC_VALUE.

The IO can be reconfigured as input, output or standard. The last configuration is used to restore the default function of the pin. The *TPM* device can return TPM_RC_VALUE if the *GPIO* name or *GPIO* function is not relevant. In standard mode, if the given *NV* index handle is not set to 0x0000 0000, the *TPM* device returns TPM_RC_VALUE.

After successful completion of this command, the configuration is saved to *NVM*.

For a new *GPIO* configuration linked to a previously created *NV* index, it is necessary to delete this *NV* index before setting the new configuration. If the *NV* index is not deleted, the TPM2_VendorCmdGPIOConfig command returns TPM_RC_NV_DEFINED.

It is possible to lock the *GPIO* configuration definitively thanks to the proprietary command called TPM2_VendorCmdSetCommandSet.

Definition of the command ordinal

Table 45. Definition of (UINT32) TPM_CC constants (numeric order) <IN/OUT, S>

Name	Command Code
TPM_CC_VendorCmdGPIOConfig	0x2000030F

TPM2_VendorCmdGPIOConfig command and response

Table 46. TPM2_VendorCmdGPIOConfig command

Type	Name	Description
TPMI_ST_COMMAND_TAG	tag	TPM_ST_SESSIONS
UINT32	commandSize	-
TPM_CC	commandCode	TPM_CC_VendorCmdGPIOConfig {NV}
TPMI_RH_PLATFORM	@authHandle	TPM_RH_PLATFORM Auth Index: 1 Auth Role: USER
TPML_GPIO_CONFIG	GPIO_Config	*GPIO* configuration list (see TPML_GPIO_CONFIG).

Table 47. TPM2_VendorCmdGPIOConfig response

Type	Name	Description
TPM_ST	tag	See clause 6 in [TPM 2.0 P3 r159].
UINT32	responseSize	-
TPM_RC	responseCode	-

TPML_GPIO_CONFIG

Table 48. Definition of TPML_GPIO_CONFIG structure

Parameter	Type	Description
count	UINT32	Maximum number of *GPIO* configurations
gpioConfig[count] {:GPIO_COUNT}	TPMS_GPIO_CONFIG	List of configuration (see Table 49. Definition of TPMS_GPIO_CONFIG structure)
#TPM_RC_SIZE	-	Response code when count is greater than the possible number of *GPIO*

TPMS_GPIO_CONFIG

Table 49. Definition of TPMS_GPIO_CONFIG structure

Parameter	Type	Description
GPIO name	TPMI_GPIO_NAME	The *I/O* name
NV index	TPMI_RH_NV_INDEX	The handle of the *NV* index
GPIO mode	TPMI_GPIO_MODE	Input, output, or standard mode
#TPM_RC_VALUE	-	Response code when an incorrect value is used

TPMI_GPIO_NAME

The following table gives the list of reconfigurable IO pins.

Table 50. Definition of TPMI_GPIO_NAME (UINT32) type

Name	Value	Description		Supported modes
		SPI	*I²C*	
GPIO_0	0x0000 0000	NA	GPIO_0	GPIO_INPUT_XXX or GPIO_OUTPUT_XXX
GPIO_1	0x0000 0001	NA	GPIO_1	GPIO_INPUT_XXX or GPIO_OUTPUT_XXX
GPIO_2	0x0000 0002	NA	GPIO_2	GPIO_INPUT_XXX or GPIO_OUTPUT_XXX
GPIO_3	0x0000 0003	NA	GPIO_3	GPIO_INPUT_XXX or GPIO_OUTPUT_XXX
GPIO_5	0x0000 0005	GPIO_5	NA	GPIO_INPUT_XXX or GPIO_OUTPUT_XXX
GPIO_6	0x0000 0006	GPIO_6	NA	GPIO_INPUT_XXX or GPIO_OUTPUT_XXX
GPIO_PP	0x0000 0009	GPIO_PP	GPIO_PP	GPIO_INPUT_XXX or GPIO_OUTPUT_XXX

TPMI_GPIO_MODE

The following table shows *GPIO* mode. Note that the *GPIO* interface cannot be configured to wake up the device.

Table 51. Definition of TPMI_GPIO_MODE (UINT32) type

Name	Value
GPIO_STANDARD	0x00000000
GPIO_INPUT_FLOATING	0x00000001
GPIO_INPUT_PULLUP	0x00000002
GPIO_INPUT_PULLDOWN	0x00000003
GPIO_OUTPUT_OPENDRAIN	0x00000004
GPIO_OUTPUT_PUSHPULL	0x00000005

9.5.4 Using the standard *TPM* commands to control the GPIOs

This section describes the standard *TPM* commands that have been modified in order to control the *GPIO*s.

TPM2_NV_DefineSpace command

Once a pin has been reconfigured as a general-purpose IO by the TPM2_VendorCmdGPIOConfig command, the related *NV* index referenced by GPIO_NAME must be created in non-volatile memory thanks to the TPM2_NV_DefineSpace command.

Additional conditions are checked when the *NV* index value is linked to a *GPIO*:

- If the *NV* index is not defined as an ordinary *NV* index, then the TPM_RC_ATTRIBUTES error is returned.
- If a data size is not equal to 1 then the TPM_RC_SIZE error is returned.
- If the TPMA_NV_CLEAR_STCLEAR, TPM_NV_WRITEDEFINE, TPMA_NV_WRITE_STCLEAR, TPMA_NV_READ_STCLEAR and TPMA_NV_ORDERLY attributes are not CLEAR then TPM_RC_ATTRIBUTES is returned.

TPM2_NV_Write command

The *GPIO* output level can be changed thanks to the TPM2_NV_Write command. Data set to 0x01 puts the *GPIO* to the high level and data set to 0 puts the *GPIO* to the low level. For any other data, the *TPM* returns TPM_RC_NV_RANGE.

TPM2_NV_Write has no effect on the level of an input *GPIO*.

TPM2_NV_Read command

TPM2_NV_Read can be used to read the *NV* index value of an input *GPIO*.

TPM2_NV_Read can also be used to read the current level of an output *GPIO*. For a *GPIO* set to high level, the data read are 0x01, and for a GPIO set to low level, the data read are 0x00.

A standard *NV* index must be written before using this command. This requirement is also applicable to a *GPIO* index (input or output).

TPM2_NV_Certify command

It is also possible to certify the state of a *GPIO* with the TPM2_NV_Certify command.

9.5.5 GPIO state after reset

The new *GPIO* configuration is applied just after the execution of the TPM2_VendorCmdGPIOConfig command and after each reset of the *TPM* during the boot phase.

There is a transient state between reset and the initialization of the *GPIO* during the *TPM* boot phase, where the *GPIO* configuration reverts to a default state.

The following table gives the default state of each *GPIO* just after reset.

Table 52. Default states of the GPIOs

Name	Mode
GPIO_0	GPIO_INPUT_PULLDOWN
GPIO_1	GPIO_INPUT_PULLDOWN
GPIO_2	GPIO_INPUT_PULLDOWN
GPIO_3	GPIO_INPUT_PULLDOWN
GPIO_5	GPIO_INPUT_VERY_WEAK_PULLDOWN
GPIO_6	GPIO_INPUT_VERY_WEAK_PULLDOWN
GPIO_PP	GPIO_INPUT_VERY_WEAK_PULLDOWN

9.5.6 Reading the current GPIO configuration

It is possible to read the current *GPIO* configuration thanks to the TPM2_GetCapability command.

The following table gives the parameters to be set in the TPM2_GetCapability command (see [TPM 2.0 P3 r159] for details) in order to read the current configuration. The TPM device returns a list with the TPML_GPIO_CONFIG type.

Table 53. Parameter values of the TPM2_GetCapability command

Parameter name	value
capability	TPM_CAP_VENDOR_PROPERTY (0x100)
property	0x00000004
propertyCount	0x00000001

9.6 EK reinstatement

9.6.1 Purpose

The *EK* restoration command is similar to TPM2_ChangeEPS [TPM 2.0 P3 r159]. The difference is that this command restores the STMicroelectronics factory *EK* primary keys. The command execution randomizes `ehProof` similarly to TPM2_ChangeEPS.

9.6.2 TPM2_VendorCmdRestoreEK command description

Command authorization

TPM2_VendorCmdRestoreEK reuses the authorization defined in the library specifications.
The TPM2_VendorCmdRestoreEK command is authorized:

- By platform authorization

Ordinal definition

Table 54. Definition of (UINT32) TPM_CC constants (numeric order) <IN/OUT, S>

Name	Command Code	Nonvolatile write	Decrypt	Encrypt	Comments
TPM_CC_VendorCmdRestoreEK	0x2000030A	Y	-	-	-

Command parameter definition

Table 55. TPM2_VendorCmdRestoreEK command

Type	Name	Description
TPMI_ST_COMMAND_TAG	tag	TPM_ST_SESSIONS
UINT32	commandSize	-
TPM_CC	commandCode	TPM_CC_VendorCmdRestoreEK
TPMI_RH_PLATFORM	@authorization	TPM_RH_PLATFORM Auth Index:1 Auth Role: USER

Table 56. TPM2_VendorCmdRestoreEK response

Type	Name	Description
TPM_ST	tag	-
UINT32	responseSize	-
TPM_RC	responseCode	-

9.7 TPM2_VendorCmdGetRandom800_90B command

TPM2_VendorCmdGetRandom800_90B is a vendor command. It is used to get random numbers from the *TRNG*, health-tested in accordance with NIST SP800-90B and ready to seed an external *DRBG* according to [SP800-90Ar1].

9.7.1 Command authorization

This command does not use any authorization.

9.7.2 Command processing

If bytesRequested is longer than TPM_PT_MAX_CAP_BUFFER (see Appendix A.3: TPM_CAP_TPM_PROPERTIES: TPM_PT (PT_FIXED) for TPM2B_MAX_BUFFER.size) can accommodate, no error is returned but the *TPM* returns as much data as a TPM2B_DATA buffer can contain.

9.7.3 Description

The command generates blocks of a maximum size of 1024 bytes and complies with [SP800-90B].

9.7.4 Ordinal definition

Table 57. Definition of (UINT32) TPM_CC constants (numeric order) <IN/OUT, S>

Name	Command Code	Nonvolatile write	Decrypt	Encrypt	Comments
TPM_CC_VendorCmdGetRandom800_90B	0x20000311	-	-	Y	-

9.7.5 Command parameter definition

Table 58. TPM2_VendorCmdGetRandom800_90B command

Type	Name	Description
TPMI_ST_COMMAND_TAG	tag	TPM_ST_SESSIONS if an audit or encrypt session is present; otherwise, TPM_ST_NO_SESSIONS.
UINT32	commandSize	-
TPM_CC	commandCode	TPM_CC_VendorCmdGetRandom800_90B
UINT16	bytesRequested	Number of bytes to generate.

Table 59. TPM2_VendorCmdGetRandom800_90B response

Type	Name	Description
TPM_ST	tag	-
UINT32	responseSize	-
TPM_RC	responseCode	-
TPM2B_MAX_BUFFER	randomBytes	The random bytes.

9.8 Storage hierarchy object deletion restriction to TPM owner

The *TPM* specifications support the use case where all objects linked to Storage or Endorsement hierarchies can be cleared by the platform firmware role linked to the Platform hierarchy. This use case is required for platform refurbishment, or to support “back to factory” services.

Some systems do not require such services and must on the other hand prevent the risk of denial of service linked to the deletion of objects provisioned in the *TPM* memory by the Platform hierarchy.

This proprietary feature allows the storage of objects to the Storage hierarchy that could be exclusively deleted by the *TPM* owner authorization.

In order to prevent objects from being deleted with the TPM_RH_PLATFORM authorization, a new vendor command, named `TPM2_VendorCmdChangeObjectDeletionAuth`, has been introduced. To execute this command, use the following sequence:

- Execute the TPM2_ClearControl command to disable the usage of TPM2_Clear. This step can also be done after the TPM2_VendorCmdChangeObjectDeletionAuth command execution by using the TPM_RH_LOCKOUT authorization.

Note: If the flag ObjectDeletionWithPlatformAuth is permanently cleared when performing this action, the TPM2_Clear command is permanently disabled.

- Execute the TPM2_VendorCmdChangeObjectDeletionAuth command to clear a new internal global permanent flag named `ObjectDeletionWithPlatformAuth` inside the *TPM*. The `PlatformAuth` parameter of the command must be set to `NO`. Refer to `TPM2_VendorCmdChangeObjectDeletionAuth command` for more information.

Clearing `ObjectDeletionWithPlatformAuth` has the following impacts:

- The TPM2_ClearControl command checks this flag to disable the usage of the command if the used authorization is `TPM_RH_PLATFORM`. The `TPM_RH_LOCKOUT` authorization remains effective.

- The TPM2_EvictControl command checks this flag to disable the eviction of objects that do not belong to the platform hierarchy if the TPM_RH_PLATFORM authorization is used. The TPM_RH_OWNER authorization remains effective for objects that do not belong to the platform hierarchy.

The `ObjectDeletionWithPlatformAuth` flag is set by default.

9.8.1 TPM2_VendorCmdChangeObjectDeletionAuth command

TPM2_VendorCmdChangeObjectDeletionAuth is a vendor command. It is used to prevent objects located in the storage hierarchy from being flushed with the TPM_RH_PLATFORM authorization.

Command authorization

TPM2_VendorCmdChangeObjectDeletionAuth uses:

- the owner authorization

Description

This command performs the following steps:

- If parameter `platformAuth` = `NO`:
  It clears the `ObjectDeletionWithPlatformAuth` global flag, which modifies the standard TPM2.0 specification behavior of the TPM2_ClearControl and TPM2_EvictControl commands.
- If parameter `platformAuth` = `YES`:
  It sets the `ObjectDeletionWithPlatformAuth` global flag, which restores the standard TPM2.0 specification behavior of the TPM2_ClearControl and TPM2_EvictControl commands.

This command is disabled by default. It can be enabled by using the TPM2_VendorCmdSetCommandSet vendor command.

Error

This command does not output errors if:

- `platformAuth` = `NO` and `ObjectDeletionWithPlatformAuth` is cleared to 0.
- `platformAuth` = `YES` and `ObjectDeletionWithPlatformAuth` is set to 1.

Ordinal definition

Table 60. Definition of (UINT32) TPM_CC constants (Numeric order) <IN/OUT, S>

Name	Command Code	Nonvolatile write	Decrypt	Encrypt	Comments
TPM_CC_VendorCmdChangeObjectDeletionAuth	0x20000310	Y	-	-	-

Command parameter definition

Table 61. TPM2_VendorCmdChangeObjectDeletionAuth command

Type	Name	Description
TPMI_ST_COMMAND_TAG	tag	TPM_ST_SESSIONS
UINT32	commandSize	-
TPM_CC	commandCode	TPM_CC_VendorCmdChangeObjectDeletionAuth
TPMI_RH_OWNER	@authorization	TPM_RH_OWNER
Auth Index:1	-	-
Auth Role: USER	-	-
TPMI_YES_NO	platformAuth	`YES` if platform authorization can be used to execute TPM2_ClearControl and TPM2_EvictControl
		`NO` if platform authorization cannot be used to execute TPM2_ClearControl and TPM2_EvictControl.

Table 62. **TPM2_VendorCmdChangeObjectDeletionAuth response**

Type	Name	Description
TPM_ST	tag	-
UINT32	responseSize	-
TPM_RC	responseCode	-

9.8.2 Modification of existing commands

TPM2_ClearControl

Error TPM_RC_AUTH_FAIL is output by the TPM2_ClearControl command if:

- The command execution authorization is TPM_RH_PLATFORM.
  AND
- `ObjectDeletionWithPlatformAuth` is cleared to 0.

TPM2_EvictControl

Error TPM_RC_AUTH_FAIL is output by TPM2_EvictControl if:

- The command execution authorization is TPM_RH_PLATFORM.
  AND
- `ObjectDeletionWithPlatformAuth` is cleared to 0.
  AND
- The object is stored in the OWNER hierarchy

10 Technical features

10.1 RSA key pairs background generation

This product supports the generation of prime numbers as a background task, when there is no *TPM* command to process. Primes are stored in dedicated slots and are used to speed up the generation of *RSA* key pairs when required by an input *TPM* command.

The product supports eight slots to store seven 2048-bit and one 3072-bit RSA key pairs.

For ordinary keys creation, command execution times are short if key pairs are available in the dedicated slots. Command execution times might reach several seconds if no key pair is available.

10.2 RSA key pairs provisioned at the ST factory

Three 2048-bit RSA keys are provisioned in *RSA* key slots during the ST manufacturing process. These keys are generated by the same public-key infrastructure (*PKI*) as the one used for *EK* provisioning. These keys are available for *RSA* key creation command for ordinary objects from the first power-up of the device. The response time to the first three commands is deterministic and faster. These keys can only be used once and are erased once they are linked to a *TPM* object.

10.3 Low-power modes

10.3.1 Definitions

Sleep event: an event that causes the product to enter a low-power mode.

Wakeup event: an event that causes the product to exit the low-power mode it is in.

Low-power modes: there are the three low-power configurations available on the product and activated by the TPM2_VendorCmdSetMode command (refer to Proprietary commands). Namely these modes are:

- TPMLowPower (disabled)
- TPMLowPowerAuto (Enabled)

They are described in the following sections.

10.3.2 TPMLowPower (disabled)

In this mode, the ST33KTPM-IDevID is able to reach the I_{CC} idle power level (see Table 67) but only when the ST33KTPM-IDevID is inactive. No sleep or wake-up event occurs.

10.3.3 TPMLowPowerAuto (Enabled)

In this mode, the ST33KTPM-IDevID is able to reach the I_{CC} standby power level (see Table 67) according to the ST33KTPM-IDevID activity.

10.3.3.1 TPMLowPowerAuto sleep condition

When TPMLowPowerAuto is enabled, the ST33KTPM-IDevID device automatically switches from the I_{CC} idle to the I_{CC} standby power level (see Power consumption characteristics) according to the ST33KTPM-IDevID activity.

To avoid the low-power impact on the *TPM* activity during the boot time, the BootToLowPower timer is defined as the time between the *TPM* power-on and the first standby state request (default value 150 seconds). The BootToLowPower timer is configurable by the TPM2_SetMode command (see Section 9: Proprietary commands).

To avoid the low-power impact on successive ST33KTPM-IDevID command sequences, the CmdToLowPower timer is defined as the time between the moment the ST33KTPM-IDevID enters the idle state and the moment when it could enter the standby state (default value 2 seconds).

10.3.3.2 TPMLowPowerAuto wake-up condition

The first falling edge on Chip Select causes the ST33KTPM-IDevID to wake up. The wake-up time is 80 µs.

10.4 Double firmware instances

This product is provided with two slots reserved for TPM applicative code that are populated with two identical code versions after manufacturing. This code duplication improves resilience.

10.5 Field upgrade

This chapter concern *TPM* firmware update from current firmware version to the same or a later *TPM* version.

10.5.1 Field upgrade overview

10.5.1.1 Firmware to upgrade

The *TPM* application code in the ST33KTPM-IDevID device is divided into several parts named code blocks:

- Code memory loader (*CML*)
- TPM2.0 firmware instance 1
- TPM2.0 firmware instance 2

The *CML* is the first “user” code executed after device reboot. It is a small code block that is not upgradable. Its role consists in verifying the integrity of a *TPM* instance, and starting its execution.

Only one of the two *TPM* instances is active at a given time. The field upgrade process consists in updating the TPM2.0 instance that is not active with a flashable image. On the next reboot after a successful field upgrade, the *CML* switches automatically to the upgraded instance, which becomes the active one. No TPM2_Clear command (to erase internal data) is required before or after a *TPM* firmware update. Persistent data are not affected by firmware updates.

10.5.1.2 Field upgrade sequence

Field upgrade is based on the processing of two commands (refer to *TPM* field upgrade commands for more details):

- TPM_VendorCmdFieldUpgradeStart
- TPM_VendorCmdFieldUpgradeData

The first one initiates the field upgrade process if several verifications are successful whereas the second one transfers the different firmware parts with flashable image. The TPM_VendorCmdFieldUpgradeData command is received multiple times until all the flashable image has been loaded into the *TPM* device (Flashable image indicates how the firmware is mapped onto the two commands).

10.5.1.3 Cryptographic protection

The field upgrade process is secured by the use of cryptography.

- Integrity:
  The *TPM* firmware integrity is protected by chained digests computed with SHA-384.
- Confidentiality:
  The *TPM* firmware confidentiality is protected with *AES* encryption with a 256-bit session key.
- Authenticity:
  Firmware authenticity is guaranteed by the use of a 384-bit *ECC* and *LMS* signature computed over the metadata (see Section 10.5.2.1.1 described in Type 0x00 blob) of the uploaded firmware. The private part of this key pair is kept in an HSM used for firmware image signing.

10.5.1.4 Loading protection

The *TPM* device authorizes the field upgrade to be processed in the following use cases without any restriction:

- If the version of the flashable image is strictly greater than the version of the active firmware.
- If the version of the flashable image is equal to the version of the active firmware and strictly greater than the version of the inactive firmware.

These rules permit the user to select one of two strategies:

- Performing a single update for a given flashable image. The two *TPM* instances then have different versions and only the most recent one can be executed.
- Performing two updates with the same flashable image. The two *TPM* instances are upgraded to the same version, thus allowing self-recovery.

The *TPM* device authorizes the field upgrade to be processed, with some restrictions, if the version of the flashable image is the same as the version of the two *TPM* instances.

The goal of this rule is to allow several loadings of a firmware for test purposes. In order not to hijack this rule to perform replay attacks, the *TPM* device increments an internal counter (called the upgrade counter) that is checked against a limit at the beginning of each field upgrade process. The internal counter value and the limit can be retrieved with a proprietary capability (see Table 81). When the limit (256) is reached, the field upgrade feature is deactivated until a new flashable image, with a version strictly greater than the installed one, is loaded.

A counter, known as the failure counter, is provided to avoid voluntary interruptions of the field upgrade process, which could be used by replay attacks. It is incremented if an error is detected during the field upgrade process or if the process is stopped before completion. This counter is reset only if a *TPM* firmware update has been done successfully. If the limit value of this counter is reached, the field upgrade feature is permanently deactivated. This is only the case if 256 consecutive loading trials are not completed successfully.

10.5.1.5 *Resilience*

Thanks to the presence of two instances, resilience is guaranteed in the following use cases:

- If a field upgrade process fails, the active *TPM* instance remains fully functional.
- If the *CML* cannot start the active *TPM* instance due to a firmware integrity error detection, *CML* tries to execute the second instance if its version is the same as that of the active one.

As the *TPM* device is delivered with the two instances set by default to the same version, this second case of resilience is guaranteed by default from factory.

10.5.2 Flashable image

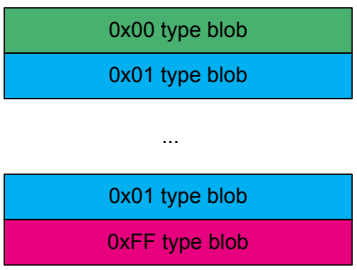
The new TPM2.0 firmware to upload to the ST33KTPM-IDevID device is distributed in the form of a file called flashable image and identifiable by its extension *.fi*.

10.5.2.1 *.fi file format*

The *.fi* file is encoded in classical ANSI format. It results in the concatenation of several parts called blobs. There are three different types of blobs:

- Type 0x00 blob
- Type 0x01 blob
- Type 0xFF blob

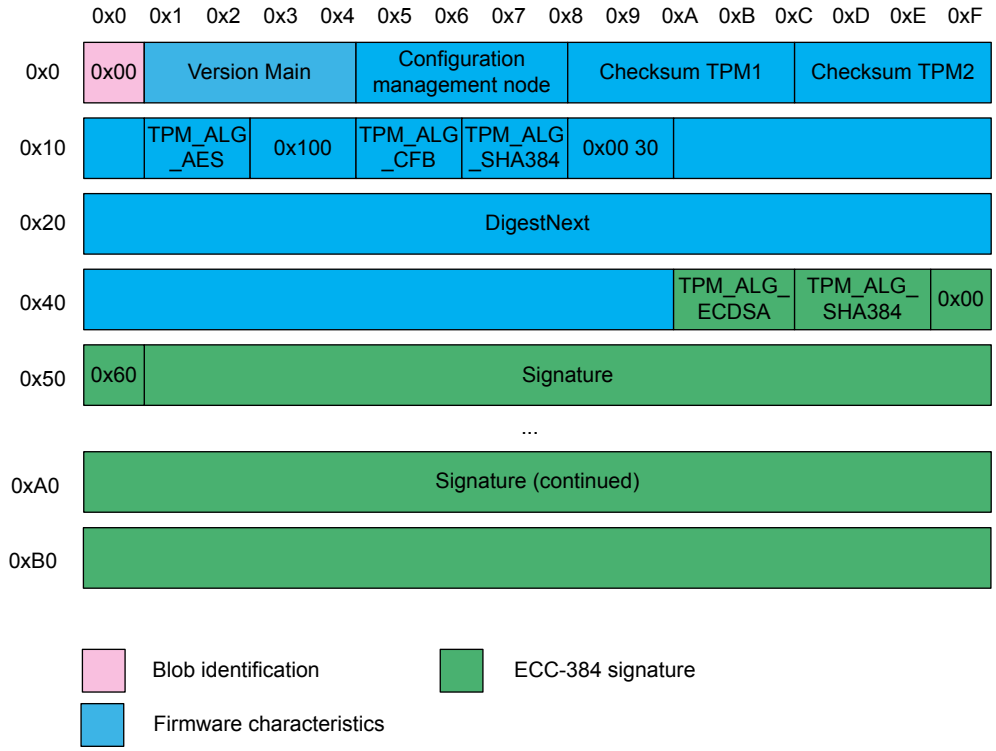
Figure 7. Format of a *.fi* file



10.5.2.1.1 Type 0x00 blob

The figure below gives the format of the type 0x00 blob.

Figure 8. Type 0x00 blob format



The TPM_ALG_AES, TPM_ALG_SHA384, and TPM_ALG_ECDSA constants mentioned in the figure are all defined in [TPM 2.0 P4 r159], table 9.

The eight fixed values in the blob correspond to:

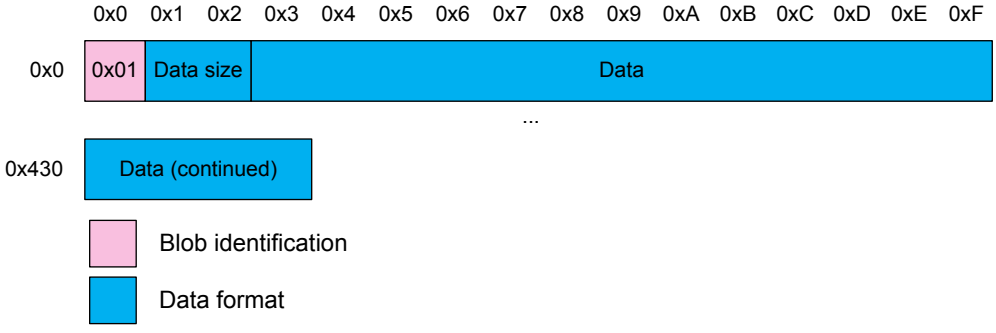
- Value “0x00” at @0x0 of the blob is the blob type
- Value “0x01 0x00” at @0x13 of the blob is the *AES* bit length used to encrypt type 0x01 blobs
- Value “0x00 0x30” at @0x19 of the blob is the length of the DigestNext field (in bytes)
- Value “0x00 0x60” at @0x4F of the blob is the length of the *ECC* signature (in bytes)

The signature is computed over the blue part of the blob with the private part of the firmware upgrade (*FU*) signature key. The type 0x00 blob is not encrypted.

10.5.2.1.2 Type 0x01 blob

The figure below gives the format of type 0x01 blobs.

Figure 9. Type 0x01 blob format



The first byte of the block indicates the blob type. *Data size* indicates the number of bytes of the data part that contains part of the code block plus the size of the digest (48 bytes).

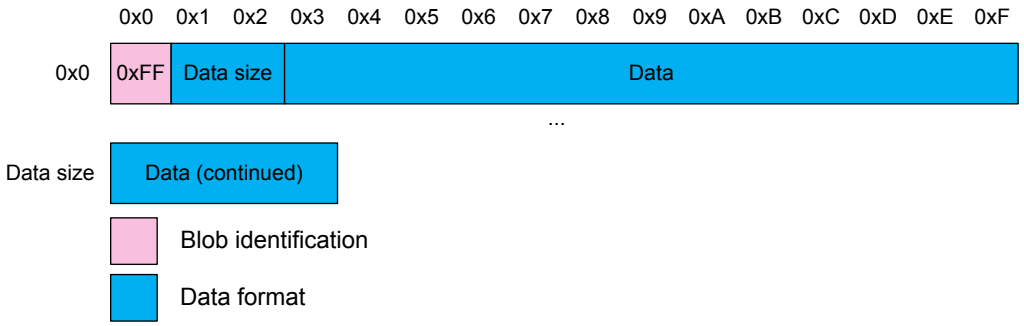
Data size of a type 0x01 blob is fixed at 0x430 (that is 1072 bytes).

The blue part of the blob is encrypted in AES 256 with the session key.

10.5.2.1.3 Type 0xFF blob

The figure below gives the format of type 0xFF blobs.

Figure 10. Type 0xFF blob format



The blue part of the blob is encrypted with the session key.

Data have the same format as for blob 0x01.

The value of the data size is comprised between 0x31 (49) and 0x430 (1072) bytes.

10.5.2.2 *Image identification*

Several data in the type 0x00 blob (see Type 0x00 blob) part of the flashable image can be used to identify the firmware to upload:

- **VersionMain** (4 bytes) that identifies the version of the firmware being uploaded.
  - The first two bytes define the major version
  - The second two bytes define the minor version.
- **VersionAdditional** (4 bytes) that contains:
  - The configuration management node used to generate the firmware.
- **Checksum TPM1** (4 bytes) whose:
  - first two bytes are equal to 0
  - last two bytes are equal to the CRC-16 computed over the TPM2.0 instance 1.
- **Checksum TPM2** (4 bytes) whose:
  - first two bytes are equal to 0
  - last two bytes are equal to the CRC-16 computed over the TPM2.0 instance 2.
- **DigestNext** (48 bytes) that identifies the firmware uniquely.

10.5.3 Field upgrade process, normal sequence

Field upgrade sequence

The field upgrade process is split over the *CML* and the *TPM* device.

Table 63. Field upgrade steps

Step	Command	Processing
0	TPM_VendorCmdFieldUpgradeStart	The input of this command is the type-0x00 blob.
1 to N	TPM_VendorCmdFieldUpgradeData	The inputs of these commands are: • first the type-0x01 blobs

Step	Command	Processing
		• then the type-0xFF blobs in the order provided in the flashable image.
N+1	TPM_Init	Starts the upgraded code instance.

TPM state during the sequence

During the field upgrade sequence, the *TPM* device is in *FU* mode.

FU mode is the default mode for field upgrade. This mode is entered after field upgrade initiation. Only one command is accepted in this mode:

- TPM2_VendorCmdFieldUpgradeData

When an attempt is made to execute another command, the *TPM* device returns TPM_RC_UPGRADE, and exits *FU* mode.

Restrictions

A flashable image can be uploaded to the *TPM* device if it fulfills specific conditions:

- The version to be uploaded must be greater than or equal to the installed version (refer to Loading protection for a detailed description).
- The major version of the flashable image must be equal to the major version of the *TPM* product firmware. If not, the loading process does not start.

Post upgrade verification

The field upgrade completion can be verified by using TPM2_GetCapability with properties `cTPM_PT_FIRMWARE_VERSION_1` after the next TPM reboot.

The capability must be retrieved around 2 seconds after TPM_Init.

10.5.4 Field upgrade process, abnormal sequence

TPM error states

Depending on the error encountered during the field upgrade, the *TPM* device could stop the field upgrade process and continue executing the code of the active instance. There is no reduced mode. The active instance remains fully functional until next reboot.

If an error occurs during field upgrade, or if the process is interrupted, the failure counter is incremented. The failure counter is compared to its maximum allowed value at the beginning of the field upgrade process to determine if the field upgrade can be started or not. The counter is reset on a successful upgrade.

In the case of multiple consecutive uploads of the same firmware version, the upgrade counter is incremented. If this counter reaches its maximum authorized value, field upgrade is temporally deactivated until an image that fulfills the incremental version rules is loaded.

Error detection

In order to determine whether an error occurred during field upgrade, several checks can be made:

- Determine the installed FW version by using TPM2_GetCapability (TPM2.0) commands. If the firmware has not been modified, the upgrade was not completed successfully.
- If an error is sent in the response to a TPM2_VendorCmdFieldUpgradeData command, the *TPM* device stops the field upgrade process. The current instance remains functional, but the uploaded instance is not valid and can be modified thanks to a new field upgrade.
- The *TPM* upgrade counter detailed in Loading protection is indicated in the response to the TPM2_GetCapability (TPM2.0) command (TPM_CAP_VENDOR_PROPERTY, see Table 81).

Field upgrade restart

A field upgrade that did not complete successfully (error detected) cannot be resumed from its previous state. A new complete field upgrade must be carried out.

Field upgrade abort

A field upgrade can be aborted by sending a `DigestNext` value that does not correspond to the correct one (a null digest for example). Field upgrade stops and the *TPM* device exits *FU* mode and returns to the normal operating state.

Command execution

During a field upgrade session, the only authorized command is TPM2_VendorCmdFieldUpgradeData. If another command is received, the *TPM* device answers by sending the TPM_RC_UPGRADE error. If the offending command is TPM2_VendorCmdFieldUpgradeStart, in addition, the *TPM* device aborts *FU* mode.

11 Electrical characteristics

This section summarizes the operating and measurement conditions, and the DC and AC characteristics of the device. The parameters in the DC and AC characteristics tables that follow are derived from tests performed under the measurement conditions summarized in the relevant tables.

The users must check that the operating conditions in their circuit match the measurement conditions when relying on the quoted parameters.

11.1 Absolute maximum ratings

Table 64. Absolute maximum ratings

Symbol	Parameter	Value	Unit
V_{PS}	Supply voltage	–0.3 to 3.63	V
V_{IO}	Input or output voltage relative to ground	–0.3 to V_{PS} +0.3	V
T_A	Ambient operating temperature	–40 to +105	°C
T_{STG}	Storage temperature (Please refer to [AN2639])	–65 to +150	°C
V_{ESD}	Electrostatic discharge voltage according to JESD22-A114, human body model	4000	V
	Electrostatic discharge voltage, charged device model according to JEDEC standard JS-002	up to 500	V

Note: Stresses listed above may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of the specification is not implied.

Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

11.2 DC and AC characteristics

T_A = −40 to 105 °C.

The voltage (V_{PS}) must be in one of the two authorized ranges: 1.8 V ±10% or 3.3 V ±10%.

The voltage on all inputs or outputs must not exceed V_{PS} + 0.3 V or be lower than V_{PS} - 0.3 V.

Table 65. DC characteristics (V_{PS} = 1.8 V or 3.3 V ± 10%)

Symbol	Parameter	Condition	Min.	Typ.	Max.	Unit
V_{IL}	Input low voltage	-	–0.3	-	0.3 × V_{PS}	V
V_{IH}	Input high voltage	-	0.7 × V_{PS}	-	V_{PS} + 0.3	V
V_{OH}	Output high voltage	I_{OH} = −2.5 mA	V_{PS} − 0.4	-	-	V
V_{OL}	Output low voltage	I_{OL} = 3 mA	-	-	0.4	V
POR	Power on reset voltage(1)	-	-	1.45	1.62	V
WPD_R	Weak pull-down resistor	-	-	40	-	kΩ
$VWPD_R$	Very weak pull-down resistor	-	-	400	-	kΩ
PU_R	Pull-up resistor	-	-	40	-	kΩ
C_{IN}	Input capacitance	-	-	-	10	pF

1. *V_{PS} voltage from which the device starts to run or V_{PS} voltage below which the device starts to switch off during a shutdown.*

11.3 Overshoot and undershoot

The *TPM* has been tested in accordance with JEDEC standard JESD78D.

- Overshoot
  - Tolerated overshoot : up to 5.4V
  - Duration(1): 10ms
- Undershoot
  - Tolerated undershoot : down to -1V
  - Duration(1): 10ms

1. *Cumulative sum during product life*

11.4 Performance and power consumption characteristics

The values provided in the table below were measured at T_A = −40 to 105 °C.

Table 66. Power-on and warm reset timing characteristics

Symbol	Parameter	Condition	Min.	Typ.	Max.	Unit
t_{WL}	$\overline{RST}$ pin low state pulse width for reset	-	1	-	-	ms
t_{INIT}	Minimum time for TPM_ACCESS_x registers to contain valid data from *TPM* reset	-	-	-	500	µs
t_{CMD}	Time required before sending first *TPM* command from *TPM* reset rising edge	-	-	-	50	ms
t_{POR_RST}	POR to $\overline{RST}$ rising edge time	C_{LOAD} = 30 pF	0	30	-	ms

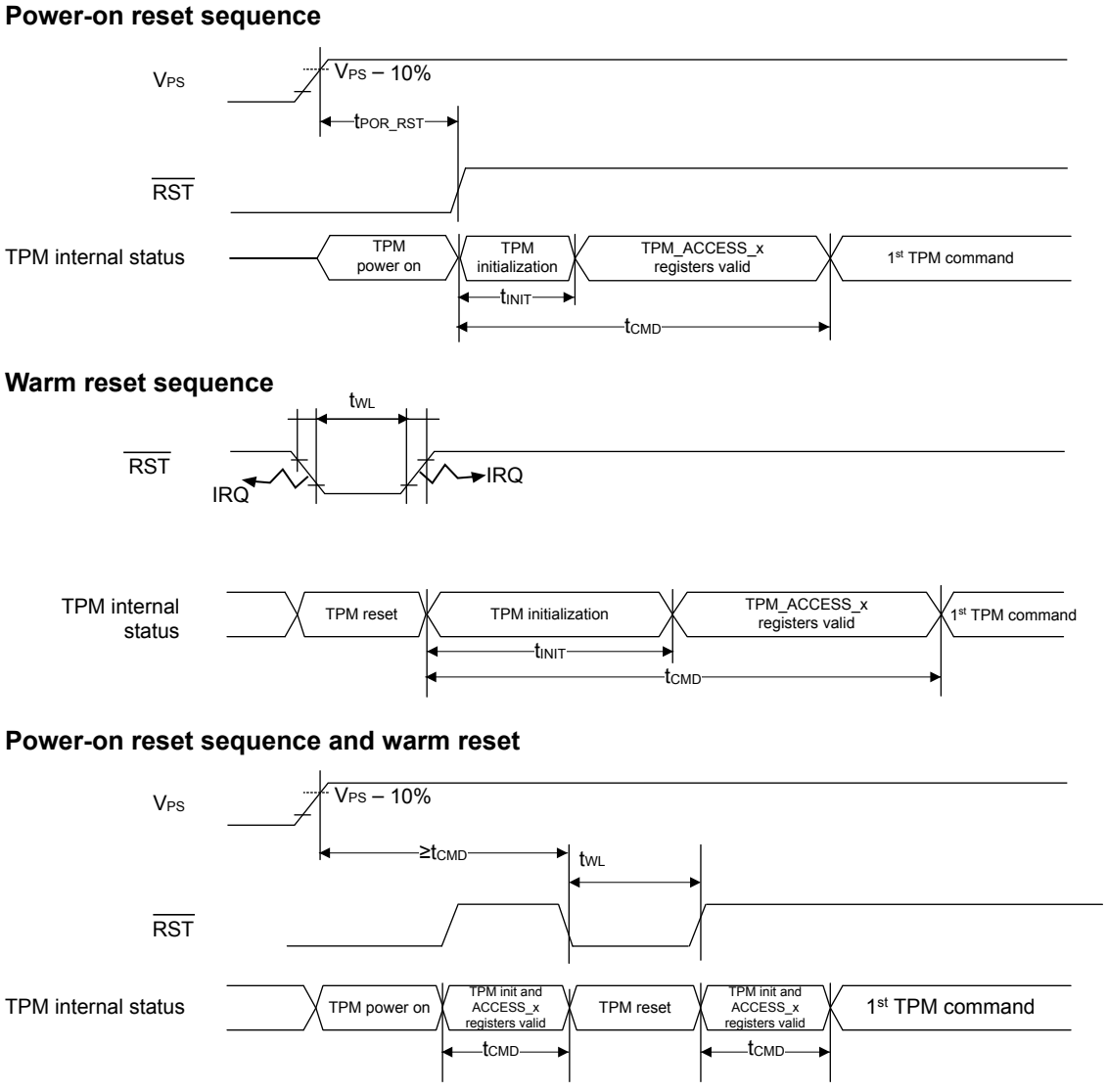
Table 67. Power consumption characteristics

The values provided in the table below were measured at T_A = 25 °C.

Symbol	Parameter	Typ.	Max.	Unit
I_{CC} run	Normal *TPM* operation	-	18	mA
I_{CC} idle	Supply current when not processing any commands.*	4.8(1)	-	mA
I_{CC} standby	Supply current when the device is in deep sleep mode.(2)	30(1)	-	µA
I_{CC} reset	Supply current when the device is in reset ($\overline{RST}$ at low level) and when the background activities are finished .	30	-	µA

1. *This value could be modified following specific GPIO configuration (value given in optimal condition).*
2. *Activated by default. See Section 9: Proprietary commands and Section 10: Technical features.*

Figure 11. Power on and warm reset sequence



Note: The power-on sequence to be followed on ST33KTPM-IDevID is:

1. *The RESET pin must not be tight to high prior to the V_{CC} power pin.*
2. *The RESET pin must be tied low prior or simultaneously with the VCC pin.*
3. *The voltage applied to the V_{CC} pin must be less than 0.3 V before starting a new power-on sequence.*

11.5 SPI characteristics

Table 68. SPI electrical characteristics

Data based on design simulation and/or characterization results, not tested in production.

Symbol	Parameter		Min.	Max.	Unit
f_{SCK}	*SPI* clock frequency with max baud rate (f/2)		-	66	MHz
$t_{c(SCK)}$	$1/f_{SCK}$		15.15	-	ns
$t_{w(sckh)}$	SPI_CLK high time		48	52	%
$t_{r(SCK)}$	SPI_CLK rising edge time		-	2(1)	ns
$t_{f\ (SCK)}$	SPI_CLK falling edge time		-	2(1)	
$t_{su(SI)}$	Data input setup time		5	-	ns
$t_{h(SI)}$	Data input hold time		5	-	
$t_{v(SO)}$	Data output valid time		-	13	
$t_{h(SO)}$	Data output hold time		0	-	
$t_{SU(NSS)}$	Setup time		6	-	
$t_{h(NSS)}$	Hold time		6	-	
$t_{dis(SO)}$(2)	Data output disable time	V_{PS} = 1.8 V	0	21	
		V_{PS} = 3.3 V	0	12.5	
$t_{a(SO)}$(3)	Data output access time	V_{PS} = 1.8 V	0	21	
		V_{PS} = 3.3 V	0	12.5	
t_{delay}	Minimum delay time between SPI_CLK going low and $\overline{SPI\_NSS}$ going low, or between SPI_NSS goes high, and SPI_CLK goes high		7	-	

1. *Figure given for maximum SPI frequency*
2. *The Min. time is the minimum time required to invalidate the data output; the Max. time is the maximum time required for the data output to go to high impedance.*
3. *The Min. time is the minimum time required to drive the data output; the Max. time is the maximum time required to validate the data output.*

Figure 12. SPI slave timing diagram - SPI_CLK low (by default)

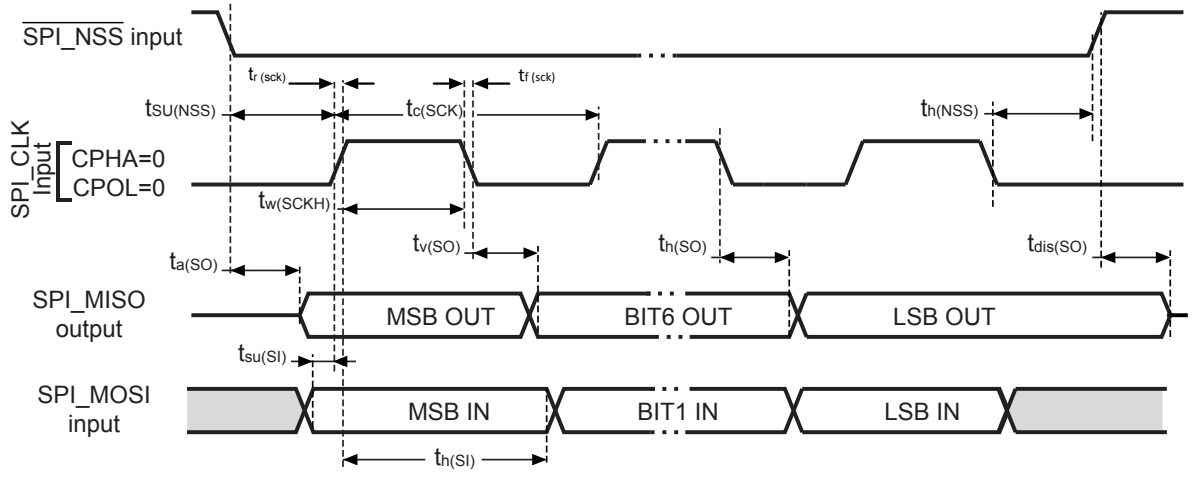
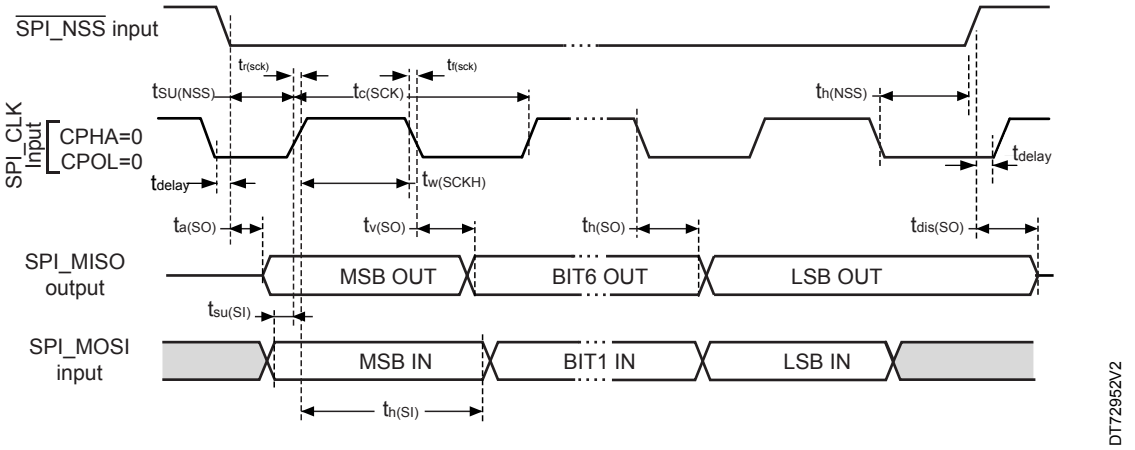


Figure 13. SPI slave timing diagram - SPI_CLK high (by default)



Note:

1. *For both drawings, measurements are made at CMOS levels: 0.3 × V_{PS} and 0.7 × V_{PS}.*
2. *When no communication is ongoing the data output line of the SPI (SPI_MOSI in master mode, SPI_MISO in slave mode) has its alternate function capability released. In this case, the pin status depends on the I/O port configuration.*

11.6 I²C characteristics

Table 69. I²C timing characteristics

Symbol	Parameter [1]		Standard-mode		Fast-mode		Fast-mode Plus		Unit
			Min	Max	Min	Max	Min	Max	
I2C_SCL[2]	I²C communication clock frequency (typical)	3.3 V	0	100	0	400	0	1000	kHz
		1.8 V							
t_{LOW}	I2C_SCL clock low time	3.3 V	4.7	-	1.3	-	0.5	-	µs
		1.8 V							
t_{HIGH}	I2C_SCL clock high time		4.0	-	0.6	-	0.26	-	
$t_{SU;\ DAT}$	I2C_SDA setup time		250	-	100	-	50	-	ns
$t_{HD;\ DAT}$	I2C_SDA data hold time		0	- [3]	0	-[3]	0	-	
$t_{VD;\ DAT}$	Data valid time		-	3.45	-	0.9	-	0.45	µs
$t_{VD;\ ACK}$	Data valid acknowledge time		-	3.45	-	0.9	-	0.45	
t_{rCL}	I2C_SCL signal rise time		-	1000	20	300	-	120	ns
t_{rCL1}	I2C_SCL signal rise time after a repeated Start condition and after an acknowledge bit		-	1000	20	300	-	120	ns
t_{fCL}	I2C_SCL signal fall time		-	300	20 × V_{CC}/5.5	300	20 × V_{CC}/5.5	120	ns
t_{rDA}	I2C_SDA signal rise time		-	1000	20	300	-	120	ns
t_{fDA}	I2C_SDA signal fall time		-	300	20 × V_{CC}/5.5	300	20 × V_{CC}/5.5	120	ns
$t_{HD;\ STA}$	Start condition hold time		4.0	-	0.6	-	0.26	-	µs
$t_{SU;\ STA}$	Repeated Start condition setup time		4.7	-	0.6	-	0.26	-	
$t_{SU;\ STO}$	Stop condition setup time		4.0	-	0.6	-	0.26	-	
t_{of}	Output fall time from 0.7 × V_{CC} to 0.3 × V_{CC}		5	120	5	120	5	120	ns
t_{BUF}	Stop to Start condition time (bus free)		4.7	-	1.3	-	0.5	-	µs
C_b	Capacitive load for each bus line		-	400	-	400	-	100	pF
R_{PU}	Pullup resistance for each bus line		1	-	1	-	1	-	kΩ

1. *The I²C characteristics are the requirements from I²C bus specification rev. 05. They are specified by design when the I2C timing registers are correctly programmed. These characteristics are not tested in production.*
2. *In controller mode, the I2C_SCL frequency decreases with the load capacitance on the I2C_SCL line.*
3. *The maximum $t_{HD;DAT}$ could be 3.45 µs and 0.9 µs for Standard mode and Fast mode, but must be less than the maximum of $t_{VD;DAT}$ or $t_{VD;ACK}$ by a transition time. This maximum must only be met if the device does not stretch the LOW period (t_{LOW}) of the I2C_SCL signal. If the clock stretches the I2C_SCL signal, the data must be valid by the set-up time before it releases the clock.*

Figure 14. **I²C bus protocol**

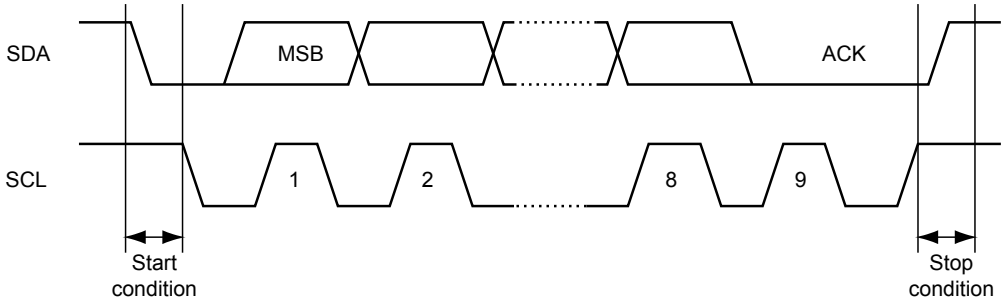
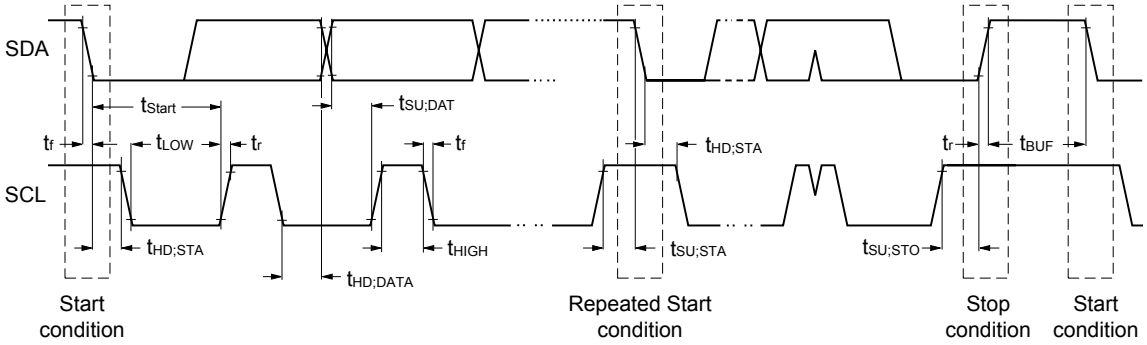


Figure 15. **Typical application with I²C bus and timing diagram**



11.7 AC measurement conditions

Table 70. AC measurement conditions

T_A = −40 to 105 °C.
f = 1 MHz, unless otherwise specified.

Parameter	Value (1)
Input rise and fall times	10 ns max
Input pulse voltage	V_{IL} to V_{IH}
Input timing reference voltage	0.5 × V_{PS}
Output timing reference voltage	V_{OL} to V_{OH}

1. *Measurement points are at CMOS levels: 0.3 × V_{PS} and 0.7 × V_{PS}.*

Figure 16. AC testing input/output waveforms

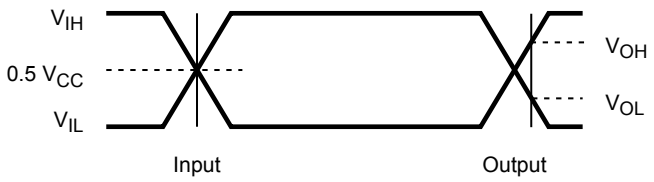
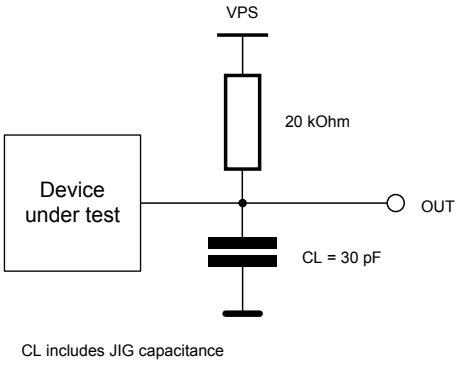


Figure 17. AC testing load circuit



11.8 Environment sensor characteristics

Table 71. Environment sensor characteristics

Symbol	Parameter	Min.	Typ.	Max.	Unit
T_{LOW}	Temperature sensor low threshold	-	-	–40	°C
T_{HIGH}	Temperature sensor high threshold	125	-	-	°C

Note: Sensors could have spurious detection. The device controller must take this fact into account including in the implementation.

12 Electrical integration guidance

This section gives some guidance on how to integrate the ST33KTPM-IDevID device in an application.

12.1 Recommended power supply filtering

The power supply of the device should be filtered using the circuit shown in the figure below.

Figure 18. Recommended filtering capacitors on V_{CC}

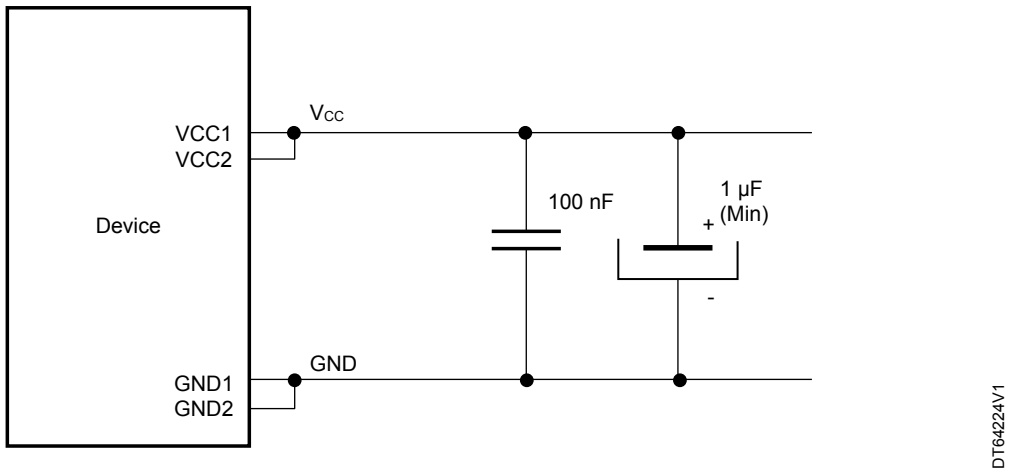


Table 72. V_{CC} rising slope

Data based on design simulation and/or characterization results, not tested in production.

Symbol	Parameter	Min.	Typ.	Max.	Unit
S_{VCC}	V_{CC} rising slope	2	-	$2 \cdot 10^3$	V/ms

Note: Measurement must be done between 1.36 V and 1.62 V. If V_{CC} rising slope requirement is unreachable for the concerned platform or if there is any other noisy environment at boot, a "power-on reset and warm reset sequence" must be run, as defined in Figure 11.

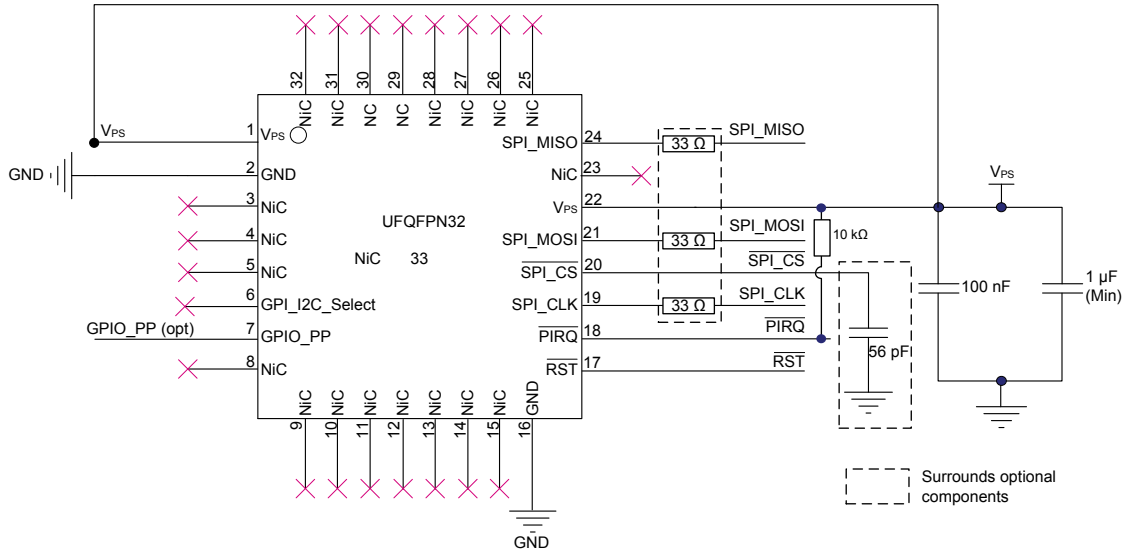
12.2 $\overline{\text{SPI\_CS}}$ optional filtering

Recommendation for SPI_CS integration: It is mandatory that SPI_CLK is at the low logic level when the falling edge occurs on the SPI_CS signal. An external capacitance of 56 pF is recommended on SPI_CS for that purpose. This capacitor might not be required depending on the intrinsic line capacitance, the SPI bus frequency, or both.

12.3 Device integration for SPI communication

The figure below shows the typical hardware implementation of the ST33KTPM-IDevID device for *SPI* communication.

Figure 19. Typical hardware implementation for SPI communication (UFQFPN32 package)



Note: The use of a low-value resistor (typically 33 Ω) on SPI_MISO, SPI_MOSI and SPI_CLK can be recommended for line adaptation when the signals are affected by parasite spikes. Its use is mandatory to avoid disturbance of the ramp-up and ramp-down signals.

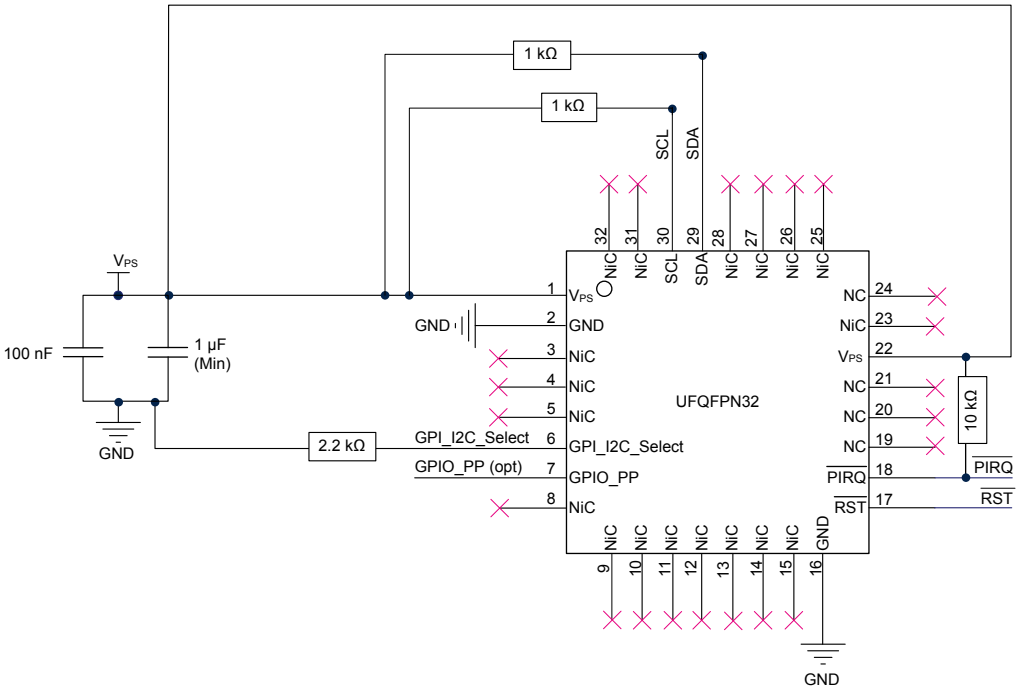
Note: The capacitor on $\overline{\text{SPI\_CS}}$ is optional (see $\overline{\text{SPI\_CS}}$ optional filtering or Recommendations).

Note: The pull-up resistor on the PIRQ line is mandatory to optimize the power consumption in standby mode.

12.4 Device integration for I²C communication

The figure below shows the typical hardware implementation of the ST33KTPM-IDevID device for *I²C* communication.

Figure 20. Typical hardware implementation for *I²C* communication (UFQFPN32 package)



Note: The pull-up resistor on the PIRQ line is mandatory to optimize the power consumption in standby mode.

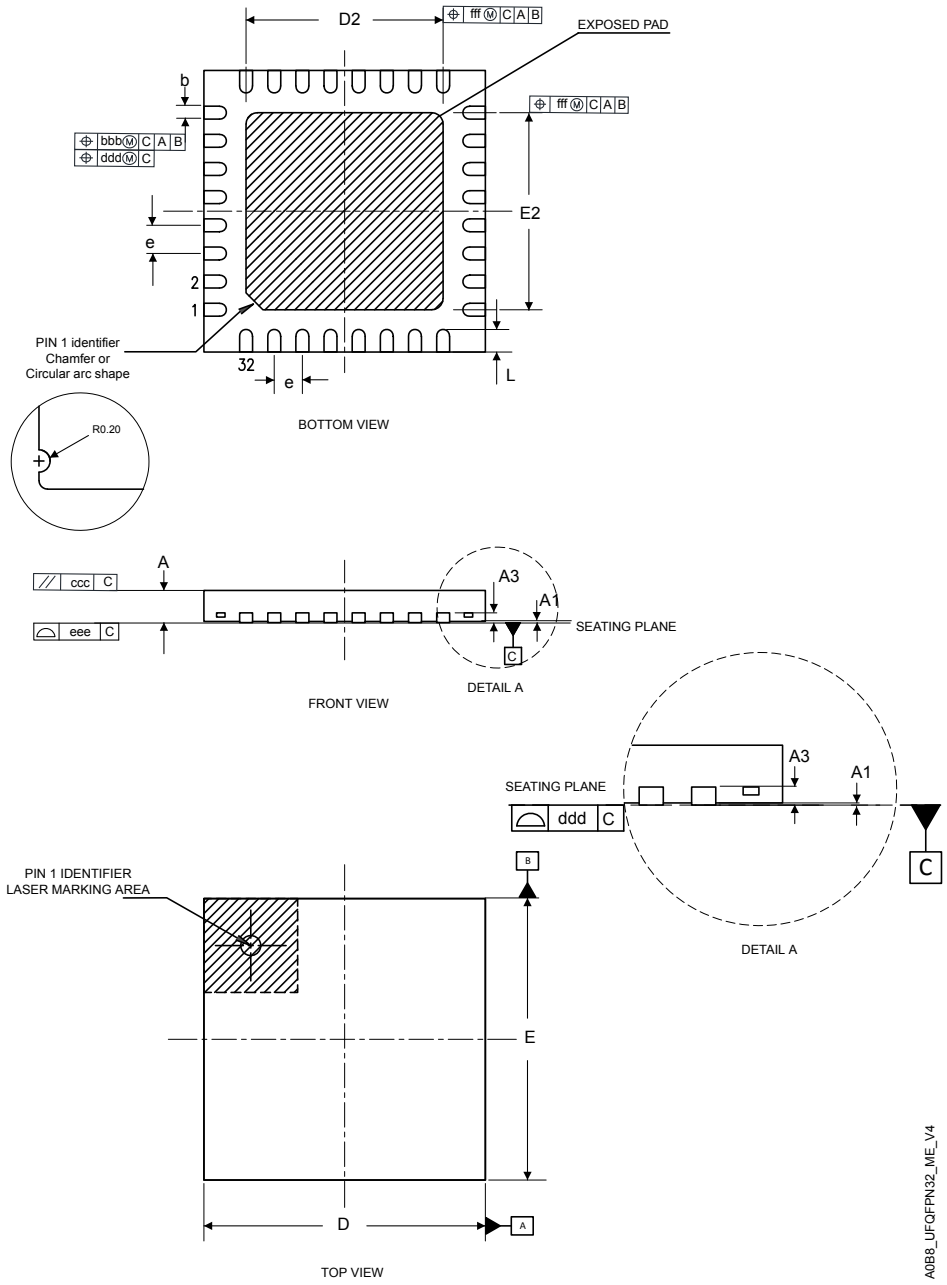
13 Package information

In order to meet environmental requirements, ST offers these devices in different grades of ECOPACK packages, depending on their level of environmental compliance. ECOPACK specifications, grade definitions and product status are available at: www.st.com. ECOPACK is an ST trademark.

13.1 UFQFPN32 package information

This UFQFPN is a 32 pins, 5x5 mm, 0.5 mm pitch ultra thin fine pitch quad flat package.

Figure 21. UFQFPN32 - Outline



1. *Drawing is not to scale.*
2. *All leads/pads should also be soldered to the PCB to improve the lead/pad solder joint life.*

3. *There is an exposed die pad on the underside of the UFQFPN package. It is recommended to connect and solder this backside pad to PCB ground.*

Table 73. UFQFPN32 - Mechanical data

Symbol	millimeters(1)			inches(2)		
	Min	Typ	Max	Min	Typ	Max
A(3)(4)	0.50	0.55	0.60	0.0197	0.0217	0.0236
A1(5)	0.00	-	0.05	0.000	-	0.0020
A3(6)	-	0.15	-	-	0.0060	-
b(7)	0.18	0.25	0.30	0.0071	0.010	0.0118
D(8)(9)		5.00 BSC			0.1969 BSC	
D2	3.50	3.60	3.70	0.139	0.143	0.147
E(8)(9)		5.00 BSC			0.1969 BSC	
E2	3.50	3.60	3.70	0.139	0.143	0.147
e(9)	-	0.50	-	-	0.02	-
N(10)			32			
K	0.15	-	-	0.006	-	-
L	0.30	-	0.50	0.0119	-	0.0199
R	0.09	-	-	0.004	-	-

1. *All dimensions are in millimetres. Dimensioning and tolerancing schemes are conform to ASME Y14.5M-2018 except European .*
2. *Values in inches are converted from mm and rounded to 4 decimal digits.*
3. *UFQFPN stands for Ultra thin Fine pitch Quad Flat Package No lead: A ≤ 0.60mm / Fine pitch e ≤ 1.00mm.*
4. *The profile height, A, is the distance from the seating plane to the highest point on the package. It is measured perpendicular to the seating plane.*
5. *A1 is the vertical distance from the bottom surface of the plastic body to the nearest metallized package feature.*
6. *A3 is the distance from the seating plane to the upper surface of the terminals.*
7. *Dimension b applies to metallized terminal. If the terminal has the optional radius on the other end of the terminal, the dimension b must not be measured in that radius area.*
8. *Dimensions D and E do not include mold protrusion, not to exceed 0,15mm.*
9. *BSC stands for BASIC dimensions. It corresponds to the nominal value and has no tolerance. For tolerances refer to Table 74*
10. *N represents the total number of terminals.*

Table 74. Tolerance of form and position

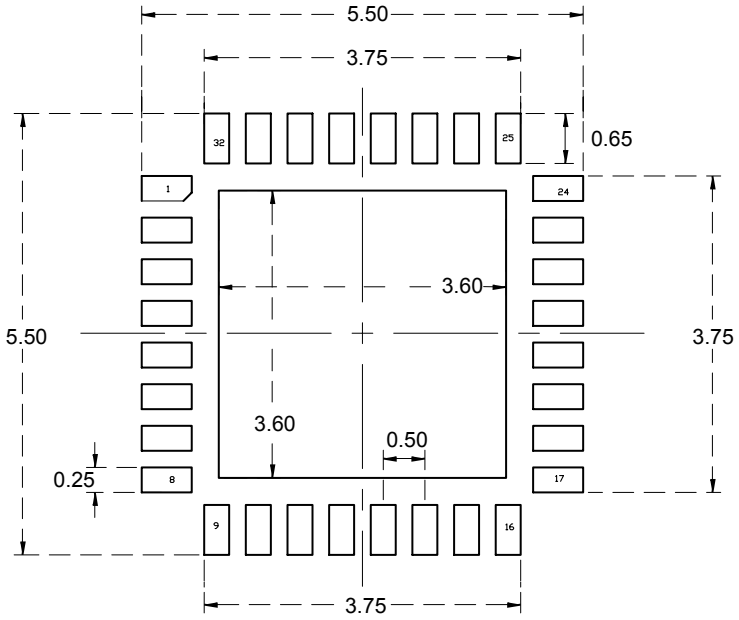
Symbol(1)	Tolerance of form and position(2) In millimeters	Tolerance of form and position(3) In inches
aaa	0.15	0.006
bbb	0.10	0.004
ccc	0.10	0.004
ddd	0.05	0.002
eee	0.10	0.004
fff	0.10	0.004

1. *For the tolerance of form and position definitions see Table 75.*
2. *All dimensions are in millimetres. Dimensioning and tolerancing schemes are conform to ASME Y14.5M-2018 except European .*
3. *Values in inches are converted from mm and rounded to 4 decimal digits.*

Table 75. Tolerance of form and position symbol definition

Symbol	Definition
aaa	The bilateral profile tolerance that controls the position of the plastic body sides. The centres of the profile zones are defined by the basic dimensions D and E.
bbb	The tolerance that controls the position of the terminals with respect to Datums A and B. The centre of the tolerance zone for each terminal is defined by basic dimension e as related to datums A and B.
ccc	The tolerance located parallel to the seating plane in which the top surface of the package must be located.
ddd	The tolerance that controls the position of the terminals to each other. The centres of the profile zones are defined by basic dimension e.
eee	The unilateral tolerance located above the seating plane wherein the bottom surface of all terminals must be located = coplanarity
fff	The tolerance that controls the position of the exposed metal heat feature. The centre of the tolerance zone is the data defined by the centrelines of the package body

Figure 22. UFQFPN32 - Footprint example



1. *Dimensions are expressed in millimeters.*

13.1.1 UFQFPN32 thermal characteristics of packages

The table below provides the thermal characteristics of the UFQFPN32 package.

Table 76. Thermal characteristics

Parameter		Symbol	Value
Recommended operating temperature range	Ambient temperature	T_A	−40 to 105 °C
	Case temperature	T_C	-
	Junction temperature	T_J	−37 to 108 °C
Absolute maximum junction temperature		-	125 °C
Maximum power dissipation		-	66 mW
Theta-JA, -JB and -JC	Junction to ambient thermal resistance	θ_{JA}(1)	35 °C/W
	Junction to case thermal resistance	θ_{JC}	5 °C/W
	Junction to board thermal resistance	θ_{JB}	20 °C/W
Psi-JT	Junction-to-top of the package thermal characterization parameter	Ψ_{JT}	0.2 °C/W

1. *According to JESD51-2 (still air condition).*

14 UFQFPN32 - tape and reel delivery packing

Surface-mount packages can be supplied with tape and reel packing. The reels have a 13" typical diameter.

Reels are in plastic, either anti-static or conductive, with a black conductive cavity tape. The cover tape is transparent anti-static or conductive.

The devices are positioned in the cavities with the identifying pin (normally pin “1”) on the same side as the sprocket holes in the tape.

The STMicroelectronics tape and reel specifications are compliant with the EIA 481-A standard specification.

Table 77. UFQFPN32 - Packages on tape and reel

Package	Description	Tape width	Tape pitch	Reel diameter	Quantity per reel
UFQFPN32	Ultrathin fine pitch quads flat pack no-lead package	12 mm	8 mm	13 in.	3000

Figure 23. UFQFPN32 - Reel diagram

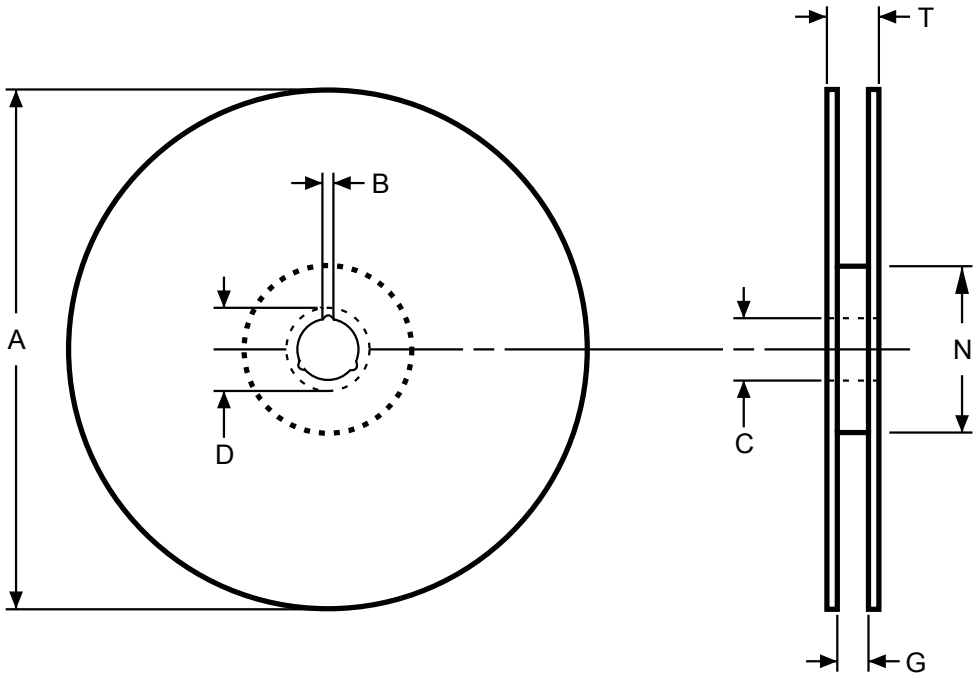
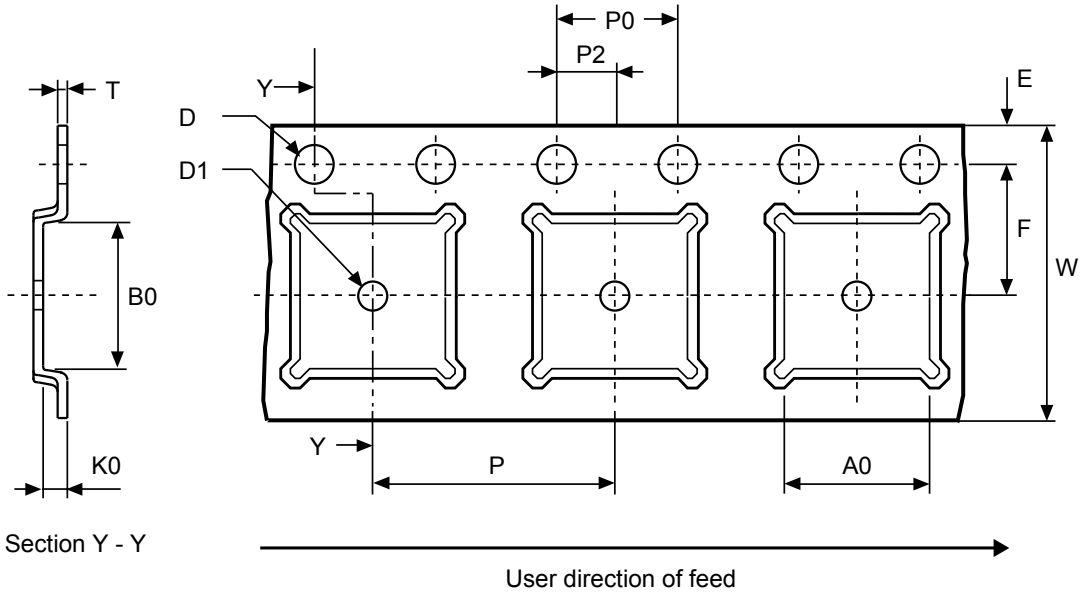


Table 78. UFQFPN32 - Reel dimensions

Reel size	Tape width	A Max.	B Min.	C	D Min.	G Max.	N Min.	T Max.	Unit
13”	12	330	1.5	13 ±0.2	20.2	12.6	100	18.4	mm

Figure 24. **UFQFPN32 - Embossed carrier tape**



1. *Drawing is not to scale.*

Figure 25. **UFQFPN32 - Chip orientation in the embossed carrier tape**

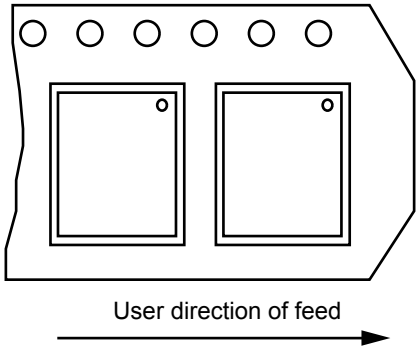


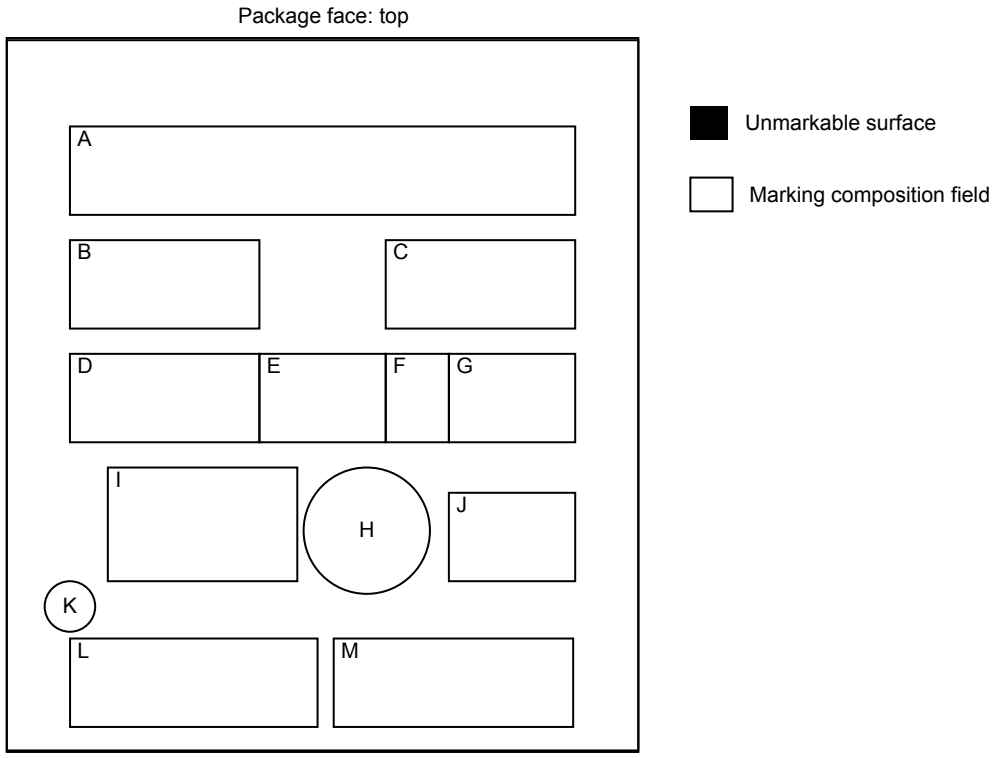
Table 79. **UFQFPN32 - Carrier tape dimensions**

Package	A0	B0	K0	D1 Min.	P	P2	D	P0	E	F	W	T Max.	Unit
UFQFPN 5×5	5.3 ±0.1	5.3 ±0.1	0.75 ±0.1	1.5	8 ±0.1	2 ±0.05	1.55 ±0.05	4 ±0.1	1.75 ±0.1	5.5 ±0.1	12 ±0.3	0.3 ±0.05	mm

15 UFQFPN32 package marking information

Parts marked as E or ES (for engineering sample) are not yet qualified and therefore not approved for use in production. ST is not responsible for any consequences resulting from such use. In no event will ST be liable for the customer using any of these engineering samples in production. ST's Quality department must be contacted prior to any decision to use these engineering samples to run a qualification activity.

Figure 26. UFQFPN32 - Standard marking example



Legend:

A: Marking area – Up to 8 digits
B: Marking area – 3 digits
C: BE sequence (LLL)
D: Country of origin (3 characters allowed (max.))
E: Assembly plant (PP)
F: Assembly year (Y)
G: Assembly week (WW)
H: Second level interconnect
I: Standard STMicroelectronics logo
J: Diffusion traceability plant (WX)
K: Dot(1)
L: Provisioning area
M: Chip position

1. *The dot on the back side indicates the pin 1 location.*

16 Ordering information

Table 80. Ordering information

Ordering code	Product line	Initial part number ordering code	Provisioning profile reference(1)	Package	Minimum ordering quantity	Marking area A	Marking area B	Marking area L
ST33K2X32DKG9ST1	ST33KTPM-IDevID	ST33KTPM2X32DKG9	ST1	UFQFPN32	3000	KTPM	KG9	ST1

1. *Provisioning profiles are described in specific application notes (see [AN6324] for ST1 provisioning profile).*

17 Support and information

Additional information regarding ST TPM devices can be obtained from the www.st.com website.

For any specific support information you can contact STMicroelectronics through the following e-mail: *tpmsupport@list.st.com*.

STMicroelectronics has put in place a Product Security Incident Response Team (ST PSIRT). We encourage you to report any potential security vulnerability that you might suspect in our products through the ST PSIRT web page: https://www.st.com/psirt.

Appendix A TPM 2.0 capability

A.1 TPM_CAP_VENDOR_PROPERTY: SUBCAP_VENDOR_INTERNAL_DATA

Value: 0x0000 0100

1. TPM command to get capability.
   [TPM request]

```
80 01 (tag: TPM_ST_NO_SESSIONS)
00 00 00 16 (commandSize: 22)
00 00 01 7A (commandCode: TPM_CC_GetCapability)
00 00 01 00 (capability: TPM_CAP_VENDOR_PROPERTY)
00 00 00 00 (property: 0)
00 00 00 01 (propertyCount: 1)
```

   [TPM response]

```
80 01 (tag: TPM_ST_NO_SESSIONS)
00 00 01 0F (responseSize: 287)
00 00 00 00 (responseCode: TPM_RC_SUCCESS)
00 (moreData: NO)
(capabilityData @ TPMS_CAPABILITY_DATA)
00 00 01 00 (capability: TPM_CAP_VENDOR_PROPERTY)
00 00 00 01 (count: 1)
(vendorProperty[0])
00 00 00 00 (property: TPM_SUBCAP_VENDOR_INTERNAL_DATA)
(data @ TPM2B_MAX_BUFFER)
01 06 (size: 262)
(buffer)
30
C5 62 A9 92 52 1C 91 52 55 05 68 F9 BB 09 7E 5C
13 5B 56 0B 9C DB B5 67 31 08 08 23 1E 51 1C E6
5B 4A D1 30 F5 A6 5F F8 71 AA 7D 2D 83 38 CB E0
30 C5 62 A9 92 52 1C 91 52 55 05 68 F9 BB 09 7E
5C 13 5B 56 0B 9C DB B5 67 31 08 08 23 1E 51 1C
E6 5B 4A D1 30 F5 A6 5F F8 71 AA 7D 2D 83 38 CB
E0 30 C5 62 A9 92 52 1C 91 52 55 05 68 F9 BB 09
7E 5C 13 5B 56 0B 9C DB B5 67 31 08 08 23 1E 51
1C E6 5B 4A D1 30 F5 A6 5F F8 71 AA 7D 2D 83 38
CB E0 30 76 51 FF D6 82 BE FD B3 DF 0B AA F5 E7
A0 F5 81 76 DD 9D 92 2B B0 C1 56 C7 FF 4B 9C C0
67 A2 3E EF 13 83 63 60 01 4A 43 21 5B FA ED 71
D2 B6 5D 00 00 D1 30 00 00 E1 19 54 50 30 00 00
00 22 59 00 00 00 00 00 01 00 00 00 00 00 00 00
00 01 00 00 09 01 01 00 00 22 59 05 02 02 00 07
01 03 00 0D 00
```

2. Decoding capability

Table 81. TPM_CAP_VENDOR_PROPERTY decoding values (SUBCAP_VENDOR_INTERNAL_DATA)

Offset in the structure	Value	Byte (in hexadecimal)
0	DigestFactoryLength (1 byte)	30
1	DigestFactory (48 bytes)	C5 62 A9 92 52 1C 91 52 55 05 68 F9 BB 09 7E 5C 13 5B 56 0B 9C DB B5 67 31 08 08 23 1E 51 1C E6 5B 4A D1 30 F5 A6 5F F8 71 AA 7D 2D 83 38 CB E0
49	DigestCurrent TPM instance 1 length (1 byte)	30
50	DigestCurrent TPM instance 1 (48 bytes)	C5 62 A9 92 52 1C 91 52 55 05 68 F9 BB 09 7E 5C 13 5B 56 0B 9C DB B5 67 31 08 08 23 1E 51 1C E6 5B 4A D1 30 F5 A6 5F F8 71 AA 7D 2D 83 38 CB E0
98	DigestCurrent TPM instance 2 length (1 byte)	30
99	DigestCurrent TPM instance 2 (48 bytes)	C5 62 A9 92 52 1C 91 52 55 05 68 F9 BB 09 7E 5C 13 5B 56 0B 9C DB B5 67 31 08 08 23 1E 51 1C E6 5B 4A D1 30 F5 A6 5F F8 71 AA 7D 2D 83 38 CB E0
147	DigestKeyLength (1 byte)	30
148	DigestKey (48 bytes) (Digest (SHA384) of the public key used to authenticate flash memory images.)	76 51 FF D6 82 BE FD B3 DF 0B AA F5 E7 A0 F5 81 76 DD 9D 92 2B B0 C1 56 C7 FF 4B 9C C0 67 A2 3E EF 13 83 63 60 01 4A 43 21 5B FA ED 71 D2 B6 5D
196	TPM instance 1 checksum (4 bytes) - CRC16	00 00 D1 30
200	TPM instance 2 checksum (4 bytes) - CRC16	00 00 E1 19
204	Hardware code (4 bytes) (generic hardware chameleon code)	54 50 30 00 (TP0)
208	Configuration management node (4 bytes)	00 00 22 59
212	TPM upgrade counter (4 bytes)	00 00 00 00 (0)
216	TPM upgrade counter limit (4 bytes)	00 01 00 00 (256)
220	TPM failure counter (4 bytes):	00 00 00 00 (0)
224	TPM failure counter limit (4 bytes):	00 00 01 00 (256)
228	Inactive TPM instance version (4 bytes)	00 09 01 01 (9.257)
232	Inactive TPM instance Configuration management node (4 bytes)	00 00 22 59
236	TPM library version (4 bytes)	05 02 02 00
240	HWINTF library version (4 bytes)	07 01 03 00
244	FactoryFWDowngradeAbility (1 byte) 0x0D for lock mode, which corresponds to the product configuration)	0D
245	Active TPM instance (1 byte): (Indicates the active TPM instance: 0 for instance 1, 1 for instance 2)	00

A.2 TPM_CAP_VENDOR_PROPERTY: SUBCAP_VENDOR_LOW_POWER_MODE

Value: 0x00000100

1. TPM command to get capability.
   [TPM request]

```
80 01 (tag: TPM_ST_NO_SESSIONS)
00 00 00 16 (commandSize: 22)
00 00 01 7a (commandCode: TPM_CC_GetCapability)
00 00 01 00 (capability: TPM_CAP_VENDOR_PROPERTY)
00 00 00 01 (property: 1)
00 00 00 01 (propertyCount: 1)
```

   [TPM response]

```
80 01 (tag: TPM_ST_NO_SESSIONS)
00 00 00 20 (responseSize: 32)
00 00 00 00 (responseCode: TPM_RC_SUCCESS)
00 (moreData: NO)
(capabilityData @ TPMS_CAPABILITY_DATA)
00 00 01 00 (capability: TPM_CAP_VENDOR_PROPERTY)
(data @ TPML_VENDOR_PROPERTY)
00 00 00 01 (count: 1)
 (vendorProperty[0])
00 00 00 01 (property: TPM_SUBCAP_VENDOR_SETMODE_INFO)
(data @ TPM2B_MAX_BUFFER)
00 06 (size: 7)
(buffer)
00 00 01 96 02 00
```

2. Capability decoding

Table 82. TPM_CAP_VENDOR_PROPERTY decoding properties (SUBCAP_VENDOR_LOW_POWER_MODE)

Value	Byte
Reserved	00
Reserved	00
LowPowerAuto (1 byte): 1 Active, 0 not active	01
BootToLowPowerTimeout (1 byte): 0..255 seconds	96
CmdToLowPowerTimeout (1 byte): 0..255 seconds	02
modeLock (1 byte): mask indicating which mode is locked	00

Table 83. modeLock(1 byte) decoding

Bit	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Definition	Res	Res	Res	Res	Low-power mode: – 1: Locked – 0: Not locked	Low-power mode: – 1: Locked – 0: Not locked	Res	Res
Value	0	0	0	0	0	0	0	0

A.3 TPM_CAP_TPM_PROPERTIES: TPM_PT (PT_FIXED)

Table 84. **TPM_CAP_TPM_PROPERTIES: TPM_PT (PT_FIXED)**

Definition	Raw data	Value
TPM_PT_FAMILY_INDICATOR	0x322E3000	2.0
TPM_PT_LEVEL	0	0
TPM_PT_REVISION	0x9F	1.59
TPM_PT_DAY_OF_YEAR	0x9	9
TPM_PT_YEAR	0x7E7	2023
TPM_PT_MANUFACTURER	0x53544D20	“STM”
TPM_PT_VENDOR_STRING_1	0x53543333	"ST33"
TPM_PT_VENDOR_STRING_2	0x4B54504D	"KTPM"
TPM_PT_VENDOR_STRING_3	0x32580000	"2X"
TPM_PT_VENDOR_STRING_4	0x00000000	""
TPM_PT_VENDOR_TPM_TYPE	0x0	0
TPM_PT_FIRMWARE_VERSION_1	0x00.09.01.01	9.257
TPM_PT_FIRMWARE_VERSION_2	0x0	0
TPM_PT_INPUT_BUFFER	0x400	1024
TPM_PT_TPM2_HR_TRANSIENT_MIN	0x5	5
TPM_PT_TPM2_HR_PERSISTENT_MIN	0x7	7
TPM_PT_HR_LOADED_MIN	0x4	4
TPM_PT_ACTIVE_SESSIONS_MAX	0x40	64
TPM_PT_PCR_COUNT	0x18	24
TPM_PT_PCR_SELECT_MIN	0x3	3
TPM_PT_CONTEXT_GAP_MAX	0xFFFF	65 535
TPM_PT_NV_COUNTERS_MAX	0x0	0
TPM_PT_NV_INDEX_MAX	0x800	2048
TPM_PT_MEMORY	0x2	2
TPM_PT_CLOCK_UPDATE	0x2710	10000
TPM_PT_CONTEXT_HASH	0xC	12
TPM_PT_CONTEXT_SYM	0x6	6
TPM_PT_CONTEXT_SYM_SIZE	0x100	256
TPM_PT_ORDERLY_COUNT	0xFF	255
TPM_PT_MAX_COMMAND_SIZE	0xBA0	2976
TPM_PT_MAX_RESPONSE_SIZE	0xBA0	2976
TPM_PT_MAX_DIGEST	0x30	48
TPM_PT_MAX_OBJECT_CONTEXT	0x672	1650
TPM_PT_MAX_SESSION_CONTEXT	0x146	326
TPM_PT_PS_FAMILY_INDICATOR	0x1	1
TPM_PT_PS_LEVEL	0x0	0
TPM_PT_PS_REVISION	0x105	1.05
TPM_PT_PS_DAY_OF_YEAR	0x0	0

Definition	Raw data	Value
TPM_PT_PS_YEAR	0x0	0
TPM_PT_SPLIT_MAX	0x80	128
TPM_PT_TOTAL_COMMANDS	0x71	113
TPM_PT_LIBRARY_COMMANDS	0x69	105
TPM_PT_VENDOR_COMMANDS	0x8	8
TPM_PT_NV_BUFFER_MAX	0x400	1024
TPM_PT_MODES	0x1	TPMA_MODES_FIPS_140_2
TPM_PT_MAX_CAP_BUFFER	0x400	1024

A.4 TPM_SUBCAP_VENDOR_GET_FW_HASH

Table 85. TPM_SUBCAP_VENDOR_GET_FW_HASH decoding values

Definition	Value
Instance 1 firmware hash	82 E8 16 7E FC 74 6B 07 E0 15 5D B4 E9 E6 9D 51 66 90 59 F4 19 F7 2B BE F8 82 F8 D4 5D D3 D0 CE 00 F3 9A 59 19 7B E9 6E 58 E2 9D E3 62 4E 3B 67
Instance 2 firmware hash	CC 0F 13 FF 52 32 3B F9 6E 60 B4 C6 A2 AA B2 D0 99 6E FD 81 F0 50 B6 73 02 B6 54 63 14 51 7F 7B 27 25 0E 19 82 80 0E A5 24 93 BC C3 13 2A 5C BA

A.4.1 Example for instance 1

[TPM request]

```
80 01 (tag: TPM_ST_NO_SESSIONS)
00 00 00 16 (commandSize: 22)
00 00 01 7A (commandCode: TPM_CC_GetCapability)
00 00 01 00 (capability: TPM_CAP_VENDOR_PROPERTY)
00 00 00 05 (property: 5)
00 00 00 01 (propertyCount: 1)
```

[TPM response]

```
80 01 (tag: TPM_ST_NO_SESSIONS)
00 00 00 49 (responseSize: 73)
00 00 00 00 (responseCode: TPM_RC_SUCCESS)
00 (moreData: NO)
(capabilityData @ TPMS_CAPABILITY_DATA)
00 00 01 00 (capability: TPM_CAP_VENDOR_PROPERTY)
(data @ TPML_VENDOR_PROPERTY)
    00 00 00 01 (count: 1)
    (resp: Active FW Hash)
    (vendorProperty[0])
        00 00 00 05 (property: TPM_SUBCAP_VENDOR_FW_HASH)
        (data @ TPM2B_MAX_BUFFER)
            00 30 (size: 48)
            (buffer)
            82 E8 16 7E FC 74 6B 07 E0 15 5D B4 E9 E6 9D 51
            66 90 59 F4 19 F7 2B BE F8 82 F8 D4 5D D3 D0 CE
            00 F3 9A 59 19 7B E9 6E 58 E2 9D E3 62 4E 3B 67
```

A.5 TPM_SUBCAP_VENDOR_GET_GPIO_CONFIG_INFO

[TPM request]

```
80 01 (tag: TPM_ST_NO_SESSIONS)
00 00 00 16 (commandSize: 22)
00 00 01 7a (commandCode: TPM_CC_GetCapability)
00 00 01 00 (capability: TPM_CAP_VENDOR_PROPERTY)
00 00 00 04 (property: 4)
00 00 00 01 (propertyCount: 1)
```

[TPM response]

```
80 01 (tag: TPM_ST_NO_SESSIONS)
00 00 00 71 (responseSize: 113)
00 00 00 00 (responseCode: TPM_RC_SUCCESS)
00 (moreData: NO)
(capabilityData @ TPMS_CAPABILITY_DATA)
  00 00 01 00 (capability: TPM_CAP_VENDOR_PROPERTY)
  (data @ TPML_VENDOR_PROPERTY)
    00 00 00 01 (count: 1)
    (resp: GPIO Config)
    (vendorProperty[0])
      00 00 00 04 (property: TPM_SUBCAP_VENDOR_GPIO_CONFIG)
      (data @ TPM2B_MAX_BUFFER)
        00 58 (size: 88)
        (buffer)
        00 00 00 07 00 00 00 00 00 00 00 00 00 00 00 00
        00 00 00 01 00 00 00 00 00 00 00 00 00 00 00 02
        00 00 00 00 00 00 00 00 00 00 00 03 00 00 00 00
        00 00 00 00 00 00 00 05 00 00 00 00 00 00 00 00
        00 00 00 06 00 00 00 00 00 00 00 00 00 00 00 09
        00 00 00 00 00 00 00 00
```

Table 86. **TPM_SUBCAP_VENDOR_GET_GPIO_CONFIG_INFO decoding values**

Value	Byte (in hexadecimal)
Count (4 bytes)	0x00000007
GPIO name (4 bytes)	0x00000000 (GPIO_0)
GPIO NV index	0x00000000 (default value)
GPIO mode	0x00000000 (default value)
GPIO name (4 bytes)	0x00000001 (GPIO_1)
GPIO NV index	0x00000000 (default value)
GPIO mode	0x00000000 (default value)
GPIO name (4 bytes)	0x00000002 (GPIO_2)
GPIO NV index	0x00000000 (default value)
GPIO mode	0x00000000 (default value)
GPIO name (4 bytes)	0x00000003 (GPIO_3)
GPIO NV index	0x00000000 (default value)
GPIO mode	0x00000000 (default value)
GPIO name (4 bytes)	0x00000005 (GPIO_5)
GPIO NV index (4 bytes)	0x00000000 (default value)
GPIO mode (4 bytes)	0x00000000 (default value)
GPIO name (4 bytes)	0x00000006 (GPIO_6)
GPIO NV index (4 bytes)	0x00000000 (default value)

Value	Byte (in hexadecimal)
GPIO mode (4 bytes)	0x00000000 (default value)
GPIO name (4 bytes)	0x00000009 (GPIO_PP)
GPIO NV index (4 bytes)	0x00000000 (default value)
GPIO mode (4 bytes)	0x00000000 (default value)

A.6 TPM_SUBCAP_VENDOR_GET_SETCMDLIST

[TPM request]

```
80 01 (tag: TPM_ST_NO_SESSIONS)
00 00 00 16 (commandSize: 22)
00 00 01 7A (commandCode: TPM_CC_GetCapability)
00 00 01 00 (capability: TPM_CAP_VENDOR_PROPERTY)
00 00 00 06 (property: 6)
00 00 00 01 (propertyCount: 1)
```

[TPM response]

```
80 01 (tag: TPM_ST_NO_SESSIONS)
00 00 00 43 (responseSize: 67)
00 00 00 00 (responseCode: TPM_RC_SUCCESS)
00 (moreData: NO)
(capabilityData @ TPMS_CAPABILITY_DATA)
 00 00 01 00 (capability: TPM_CAP_VENDOR_PROPERTY)
 (data @ TPML_VENDOR_PROPERTY)
     00 00 00 01 (count: 1)
     (resp: SETCMDLIST
     (vendorProperty[0])
         00 00 00 06 (property: TPM_SUBCAP_VENDOR_GET_SETCMDLIST)
         (data @ TPM2B_MAX_BUFFER)
             00 2a (size: 42)
             (buffer)
             00 00 01 24 00 00 00 00 00 01 64 00 00 00 00 00
             01 93 00 00 00 00 00 03 0A 01 00 00 00 00 03 0F
             00 00 00 00 00 03 10 00 00 00
```

Table 87. TPM_SUBCAP_VENDOR_GET_SETCMDLIST decoding values

Command Code	Activated	Locked	Lock blocked	Command name
00 00 01 24	00	00	00	TPM_CC_ChangeEPS
00 00 01 64	00	00	00	TPM_CC_EncryptDecrypt
00 00 01 93	00	00	00	TPM_CC_EncryptDecrypt2
00 00 03 0A	01	00	00	TPM2_VendorCmdRestoreEK
00 00 03 0F	00	00	00	TPM2_VendorCmdGPIOConfig
00 00 03 10	00	00	00	TPM2_VendorCmdChangeObjectDeletionAuth

A.7 TPM_SUBCAP_VENDOR_TPMA_MODES

[TPM request]

```
80 01 (tag: TPM_ST_NO_SESSIONS)
00 00 00 16 (commandSize: 22)
00 00 01 7A (commandCode: TPM_CC_GetCapability)
00 00 01 00 (capability: TPM_CAP_VENDOR_PROPERTY)
00 00 00 07 (property: 7)
00 00 00 01 (propertyCount: 1)
```

[TPM response]

```
80 01 (tag: TPM_ST_NO_SESSIONS)
00 00 00 1D (responseSize: 29)
00 00 00 00 (responseCode: TPM_RC_SUCCESS)
00 (moreData: NO)
(capabilityData @ TPMS_CAPABILITY_DATA)
00 00 01 00 (capability: TPM_CAP_VENDOR_PROPERTY)
(data @ TPML_VENDOR_PROPERTY)
    00 00 00 01 (count: 1)
    (resp: TPMA_MODES)
    (vendorProperty[0])
        00 00 00 07 (property: TPM_SUBCAP_VENDOR_TPMA_MODES)
        (data @ TPM2B_MAX_BUFFER)
            00 04 (size: 4)
            (buffer)
            00 00 00 03
```

Table 88. Decoding values for 00 00 00 03

Bits 31:5	Bit 4	Bits 3:2	Bit 1	Bit 0
RFU	FIPS_140_3_CUMULATIVE_INDICATOR	FIPS_140_3_INDICATOR	FIPS_140_3 bit support	FIPS_140_2 bit support
0b00..00	0b0	0b00	0b1	0b1
RFU	Disabled	Disabled	Enabled	Enabled

Note: FIPS_140_3_CUMULATIVE_INDICATOR and FIPS_140_3_INDICATOR are dynamic bits useable only for certification lab.

A.8 TPM_SUBCAP_VENDOR_GET_PRODUCT_INFO

[TPM request]

```
80 01 (tag: TPM_ST_NO_SESSIONS)
00 00 00 16 (commandSize: 22)
00 00 01 7A (commandCode: TPM_CC_GetCapability)
00 00 01 00 (capability: TPM_CAP_VENDOR_PROPERTY)
00 00 00 03 (property: 3)
00 00 00 01 (propertyCount: 1)
```

[TPM response]

```
80 01 (tag: TPM_ST_NO_SESSIONS)
00 00 00 81 (responseSize: 129)
00 00 00 00 (responseCode: TPM_RC_SUCCESS)
00 (moreData: NO)
(capabilityData @ TPMS_CAPABILITY_DATA)
  00 00 01 00 (capability: TPM_CAP_VENDOR_PROPERTY)
  (data @ TPML_VENDOR_PROPERTY)
    00 00 00 01 (count: 1)
       (vendorProperty[0])
      00 00 00 03 (property: TPM_SUBCAP_VENDOR_GET_PRODUCT_INFO)
      (data @ TPM2B_MAX_BUFFER)
        00 68 (size: 104)
        (buffer)
        4A E0 A0 59 DC 07 17 00 48 02 47 02 44 00 00 00
        02 04 01 03 00 00 00 54 50 30 00 00 00 00 00 00
        00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
        00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
        00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
        00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
        00 00 00 00 01 06 07 04
```

Table 89. TPM_SUBCAP_VENDOR_GET_PRODUCT_INFO decoding values

Definition	Value
Unique serial number (7 bytes)	4A E0 A0 59 DC 07 17
Pad 0 (1 byte)	00
Product identification number (PIN) (2 Bytes)	48 02 (0x248)
Master product identification number (MPIN) (2 bytes)	47 02 (0x247)
Product internal revision (1 Byte)	44 (D)
Pad 0 (3 bytes)	00 00 00
Firmware kernel version (4 bytes)	02 04 01 03 (3.1.4.2)
Pad 0 (3 bytes)	00 00 00
External chameleon (3 bytes)	54 50 30 (TP0)
Pad 0 (74 bytes)	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Cryptographic library version (4 bytes)	01 06 07 04

A.9 TPM2_GetTestResult

Below an example of TPM2_GetTestResult without any issue reported :

[TPM Request]

```
80 01 (tag: TPM_ST_NO_SESSIONS)
00 00 00 0A (commandSize: 10)
00 00 01 7C (commandCode: TPM_CC_GetTestResult)
```

[TPM Response]

```
80 01 (tag: TPM_ST_NO_SESSIONS)
00 00 00 63 (responseSize: 99)
00 00 00 00 (responseCode: TPM_RC_SUCCESS)
(outData @ GETTESTRESULT_DATA)
 00 53 (size: 83)
 00 00 00 00 (selftest_failure_status: FATAL_ERROR_NONE)
 00 00 00 00 (algos_status: 0)
 (context_CPSN)
 00 00 00 00 00 00 00
 (context_nonce)
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 (diagnostic_information)
 00 00 00 00
 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00
 FF FF FF FF (die_integrity: ACTIVE_SHIELD_SAFE)
00 00 00 00 (testResult: TPM_RC_SUCCESS)
```

Appendix B Referenced documents

The following materials are to be used in conjunction with or are referenced by this document.

[TPM 2.0 P1 r159]	TPM Library, Part 1, Architecture, Family 2.0, rev 1.59, TCG
[TPM 2.0 P2 r159]	TPM Library, Part 2, Structures, Family 2.0, rev 1.59, TCG
[TPM 2.0 P3 r159]	TPM Library, Part 3, Commands, Family 2.0, rev 1.59, TCG
[TPM 2.0 P4 r159]	TPM Library, Part 4, Supporting routines, Family 2.0, rev 1.59, TCG
[TPM 2.0 rev159 Err 1.5]	Errata Version 1.5 for Trusted Platform Module Library Family 2.0 Revision 1.59, TCG
[PTP 2.0 r1.05]	TCG PC Client Platform TPM Profile (PTP) for TPM 2.0 Version 1.05 Revision 14, TCG
[PTP 2.0 errata]	Errata version 1.0 for PC Client Platform TPM Profile (PTP) for TPM 2.0 Version 1.05 Revision 0.14, TCG
[PKCS#1]	PKCS#1: v2.1 RSA Cryptography Standard, RSA Laboratories
[AN2639]	Application note, Soldering recommendations and package information for Lead-free ECOPACK microcontrollers, STMicroelectronics
[AN6324]	ST33KTPM-IDevID - Provisioning profile ST1 description
[AN6350]	Online certificate distribution for STSAFE-TPM products
[TCG EK Cre Profile TPM 2.3]	TCG EK credential profile for TPM Family 2.0 Level 0. Specification Version 2.3 Revision 2, 23 July 2020, TCG.
[TPM 2.0 PP]	TCG Protection Profile for PC Client Specific TPM 2.0 Library Revision 1.59; Version 1.3
[SP800-90B]	Recommendation for the entropy sources used for random bit generation, January 2018, NIST
[SP800-90Ar1]	Recommendation for random number generation using deterministic random bit generators, June 2015, NIST
[Algorithm registry]	TCG Algorithm Registry Family "2.0", Revision 1.32
[Vendor Registry]	TCG TPM Vendor ID Registry Version 1.02 Revision 1.00
[TPM 2.0 IDevID &IAK]	TPM 2.0 Keys for Device Identity and Attestation, version 1.1, TCG

Revision history

Table 90. **Document revision history**

Date	Revision	Changes
21-Oct-2025	1	Initial release.

Glossary

3D Three-dimensional

AES Advanced encryption standard

BIOS Basic input/output system

CA Certification Authority

CC Common Criteria

CML Code memory loader

CRC Cyclic redundancy check

CRT Chinese remainder theorem

CSP Critical security parameter

DES Data encryption standard

DRBG Deterministic random bit generator

DXE Driver execution environment

EC Elliptic curve

ECC Elliptic curve cryptography

ECDA Elliptic curve direct anonymous attestation

ECDAA Elliptic curve direct anonymous attestation (algorithm)

ECDH Elliptic curve Diffie–Hellman

ECDSA Elliptic curve digital signature algorithm

EdDSA Edwards-curve digital signature algorithm

EK Endorsement key

ESD Electrostatic discharge

FIPS Federal Information Processing Standards

FU Field upgrade

GPIO General purpose input/output

HBM Human body model

HDRBG Hash-*DRBG*

HMAC Hash-based message authentication code or keyed-hash message authentication code

I/O Input/output

IAK Initial attestation key and certificate

IDevID Initial (factory installed) device identity certificate

I²C Inter-integrated circuit

LMS
Leighton–Micali signatures

LSB Least significant byte

MCSP Manufacturer certificate authority service provider

MCU Microcontroller unit

MSB Most significant byte

NACK Not acknowledge

NIST National Institute of Standards and Technology

NV Nonvolatile

NVM Nonvolatile memory

OEM Original equipment manufacturer

PCR Platform configuration register

PIN Personal identification number

PKCS Public key cryptographic standards

PKI Public-key infrastructure

PP Physical presence

PQC Post quantum cryptography

PSS Probabilistic signature scheme

PTP Platform *TPM* Profile

RNG Random number generator

RSA Public-key cryptosystem (created by Ron Rivest, Adi Shamir and Leonard Adleman)

RSAES Rivest Shamir Adelman encryption/decryption scheme

RSASSA Rivest Shamir Adelman signature scheme with appendix

SHA Secure Hash algorithm

SPI Serial peripheral interface

TCG Trusted Computing Group®

TDES Triple DES cryptographic algorithm

TPM Trusted platform module

TRNG True random number generator

TSS TPM software stack

UEFI Unified extensible firmware interface

WPC Wireless power consortium

X.509 X.509 is a standard format for public key certificates.

Contents

List of tables

List of figures

IMPORTANT NOTICE – READ CAREFULLY

STMicroelectronics NV and its subsidiaries (“ST”) reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice.

In the event of any conflict between the provisions of this document and the provisions of any contractual arrangement in force between the purchasers and ST, the provisions of such contractual arrangement shall prevail.

The purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST’s terms and conditions of sale in place at the time of order acknowledgment.

The purchasers are solely responsible for the choice, selection, and use of ST products and ST assumes no liability for application assistance or the design of the purchasers’ products.

No license, express or implied, to any intellectual property right is granted by ST herein.

Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.

If the purchasers identify an ST product that meets their functional and performance requirements but that is not designated for the purchasers' market segment, the purchasers shall contact ST for more information.

ST and the ST logo are trademarks of ST. For additional information about ST trademarks, refer to www.st.com/trademarks. All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.