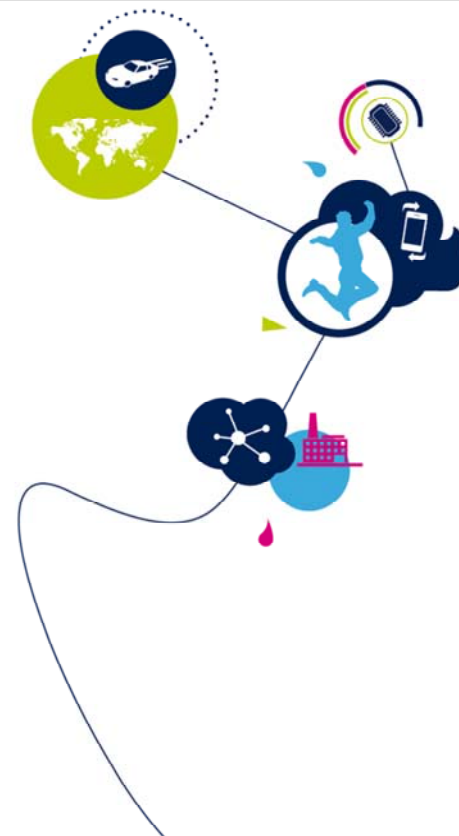
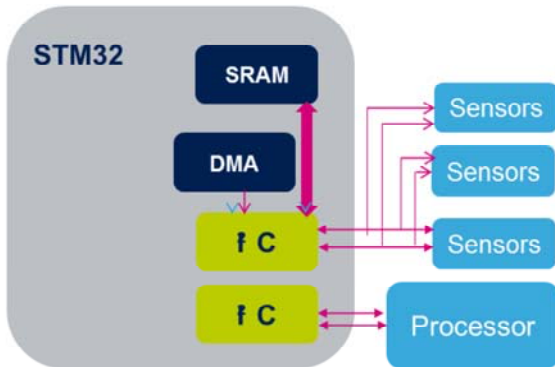


# STM32MP1 – I<sup>2</sup>C

Inter-Integrated Circuit  
Revision 1.0



Hello, and welcome to this presentation of the STM32 I<sup>2</sup>C interface. It covers the main features of this communication interface, which is widely used to connect devices such as microcontrollers, sensors, and serial interface memories.



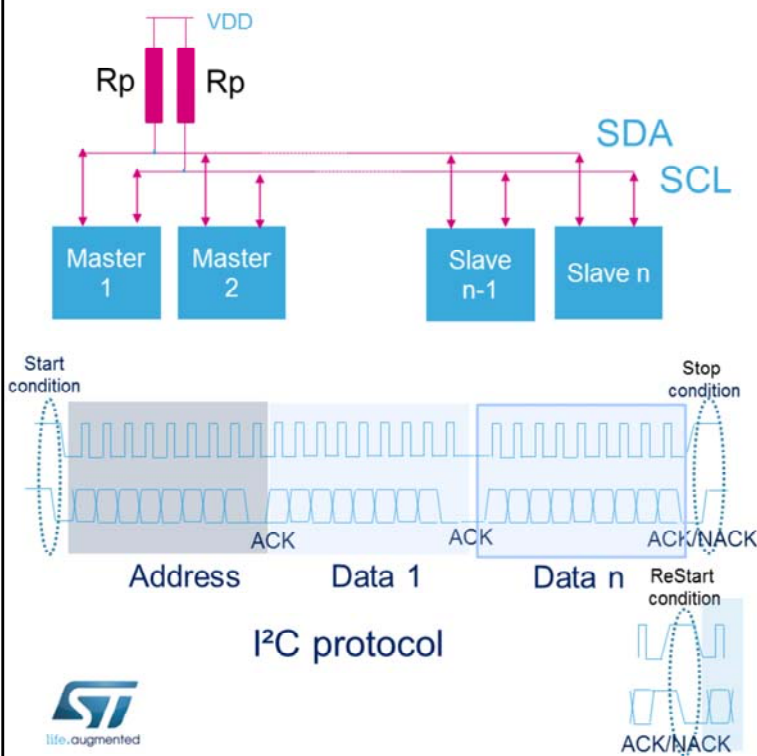
- Provides an inter-integrated circuit communication interface
  - I<sup>2</sup>C-bus specification and user manual, rev 3
    - Standard, Fast and Fast-mode Plus (1 MHz)
  - SMBus 3.0 hardware support
  - PMBus 1.3 Compatibility

### Application benefits

- Easy-to-use event management
- Fully programmable timing values
- Functional in low-power Stop modes

The I<sup>2</sup>C interface is compliant with the NXP I<sup>2</sup>C-bus specification and user manual, Revision 3; the SMBus System Management Bus Specification, Revision 3; and the PMBus Power System Management Protocol Specification, Revision 1.3.

This peripheral provides an easy-to-use interface, with very simple software programming, and full timing flexibility. Additionally, the I<sup>2</sup>C peripheral is functional in some low-power stop modes.



- Multi-master and slave capability
- 20 mA output drive capability for Fast mode Plus
- Controls all I<sup>2</sup>C bus-specific sequencing, protocol, arbitration and timing values
- 7- and 10-bit addressing modes
- Multiple 7-bit address support
- Optional clock stretching

The I<sup>2</sup>C peripheral supports multi-master and slave modes. The I<sup>2</sup>C IO pins must be configured in open-drain mode. The logic high level is driven by an external pull-up. The IO pins support the 20 mA output drive required for Fast mode Plus. The peripheral controls all I<sup>2</sup>C bus-specific sequencing, protocol, arbitration and timing values. 7- and 10-bit addressing modes are supported, and multiple 7-bit addresses can be supported in the same application. The peripheral in master mode supports slave clock stretching and clock stretching from slave side. In slave mode configuration of the peripheral, the clock stretching can be disabled by software.

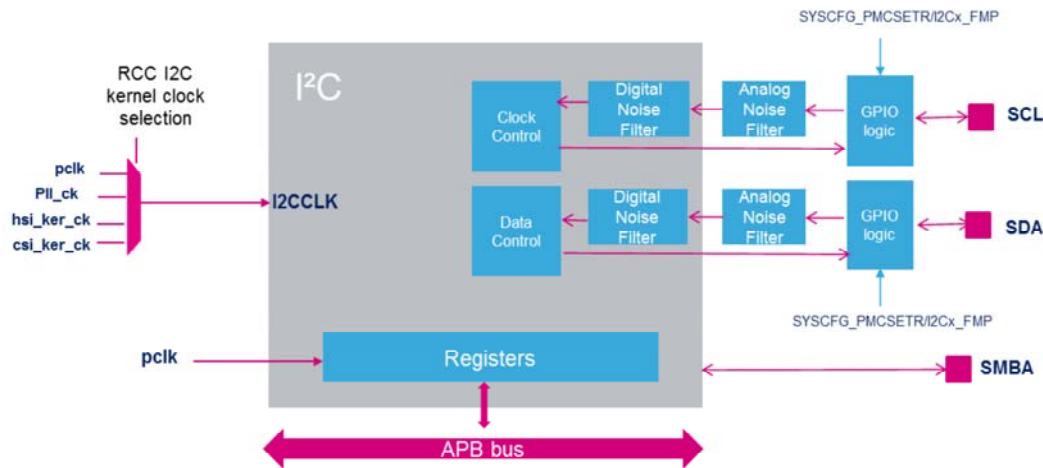
- Programmable setup and hold times
- Programmable analog and digital noise filters on SCL and SDA lines
- Wakeup from Stop mode on address match
- Independent clock allowing communication baud rate independent from system clock



The Setup and Hold times are programmable by software. Analog and digital glitch filters on the data and clock lines can be configured by software.

The peripheral can wake up the MCU from Stop mode when an address match is detected.

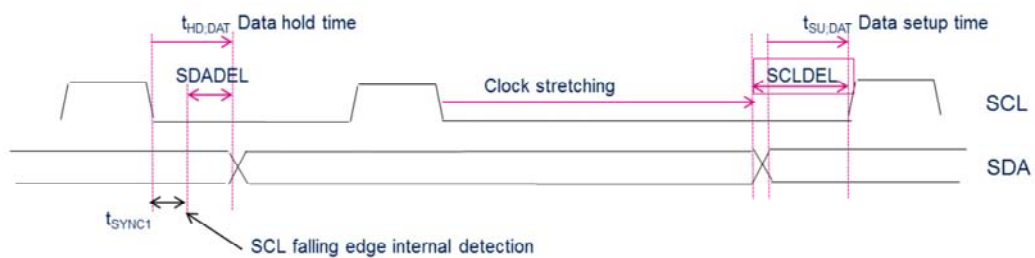
The peripheral has an independent clock domain, which allows a communication baud rate independent from the system clock.



Here is the I<sup>2</sup>C block diagram. The registers are accessed through the APB bus, and the peripheral is clocked with the I<sup>2</sup>C clock, which is independent from the APB clock. The I<sup>2</sup>C clock can be selected from among the PLL, APB clock, the high-speed internal RC oscillator, with a frequency from 8 to 64 MHz, and the low-power internal 4 MHz RC oscillator. Analog and digital noise filters are present on the SCL and SDA lines. A 20 mA driving capability is enabled using the control bits in the System configuration registers. In addition, an SMBus Alert pin is available in SMBus mode.

## Full flexibility in timing generation

- Setup and hold times between SDA and SCL lines during transmission are programmable through the PRESC, SDADEL and SCLDEL fields in the I<sup>2</sup>C Timing Register (I2C\_TIMINGR).
  - SDADEL is used to generate Data Hold time.  $t_{SDA\Delta EL} = [SDA\Delta EL * (PRESC+1) + 1] * t_{I2CCLK}$
  - SCLDEL is used to generate Data Setup time.  $t_{SCL\Delta EL} = (SCL\Delta EL+1) * (PRESC+1) * t_{I2CCLK}$



The I<sup>2</sup>C setup and hold times can be configured by software through the I<sup>2</sup>C Timing register.

The SDADEL and SCLDEL counters are used during transmission, in order to guarantee the minimum Data Hold and Data Setup times.

The I<sup>2</sup>C peripheral waits for the programmed Data Hold time after detecting a falling edge on the clock line before sending the data. After the data is sent, the clock line is stretched low during the programmed Data Setup time.

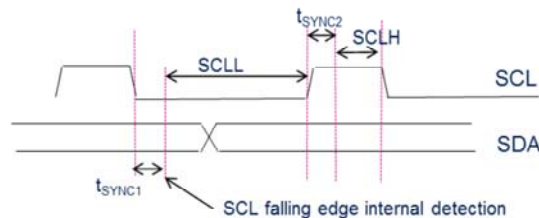
The total Data Hold time is greater than the programmed SDADEL counter. This is due to the fact that SDADEL delay is only added once the SCL falling edge is internally detected. The time ( $t_{SYNC1}$ ) needed for this internal detection depends on the SCL falling edge, the input delay due to the filters, and the delay due to the internal SCL synchronization with the I<sup>2</sup>C clock. However, the setup time is not impacted by these internal delays.

# I<sup>2</sup>C master clock generation

7

## Full flexibility in master clock generation

- SCL Low and High duration programmable through I2C\_TIMINGR
  - SCL Low counter:  $(SCLL+1) * (PRESC+1) * t_{I2CCLK}$ 
    - Starts counting after internal detection of the falling edge of SCL .
    - After the count, SCL is released.
  - SCL High counter:  $(SCLH+1) * (PRESC+1) * t_{I2CCLK}$ 
    - Starts counting after internal detection of the rising edge of SCL .
    - After the count, SCL is driven low.
- SCL period =  $t_{SYNC1} + t_{SYNC2} + [(SCLL+1) + (SCLH+1)] * (PRESC+1) * t_{I2CCLK}$



The I<sup>2</sup>C master clock's low- and high-level durations are configured by software in the I<sup>2</sup>C Timings register.

The SCL low- and high-level counters start after the detection of the edge of the SCL line. This implementation allows the peripheral to support the master clock synchronization mechanism in a multi-master environment as well as the slave clock stretching feature.

Therefore, the total SCL period is greater than the sum of the counters. This is linked to the added delays due to the internal detection of the SCL line edge. These delays,  $t_{SYNC1}$  and  $t_{SYNC2}$ , depend on the SCL falling or rising edge, the input delay due to the filters, and the delay due to the internal SCL synchronization with the I<sup>2</sup>C clock.

The rising edge depends on pull-up resistor and SCL line capacitance. The falling edge depends on the I/O port parameters defined in the datasheet. In order to properly configure clock speed, these edges can be either measured or calculated. They are needed in order to properly configure I<sup>2</sup>C peripheral in the STM32CubeMX tool, then the settings of the Timing Register can be automatically calculated by this tool.

# Slave addressing mode

8

## Large number of slave addresses

- I<sup>2</sup>C can acknowledge several slave addresses. 2 address registers:
  - I2C\_OAR1: 7- or 10-bit mode.
  - I2C\_OAR2: 7-bit mode only. OA2MSK[2:0] allow masking of 0 to 7 LSBs of OAR2

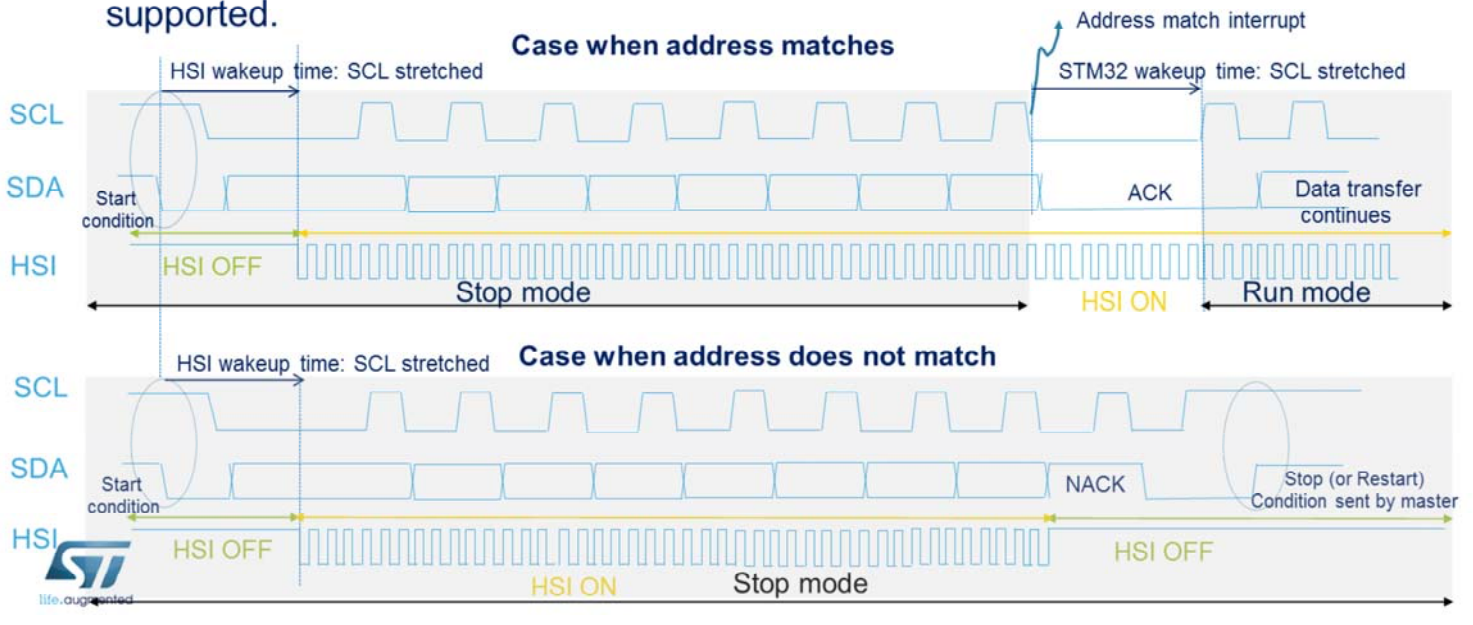
OA2MSK[2:0]	Address match condition
000	Address[7:1] = OA[7:1]
001	Address[7:2] = OA[7:2] (bit 1 is don't care)
010	Address[7:3] = OA[7:3] (bits 2:1 are don't care)
...	
111	All addresses are acknowledged except I <sup>2</sup> C reserved addresses



The I<sup>2</sup>C slave can acknowledge several slave addresses. The slave addresses are programmed into two registers. Own Address Register 1 can be programmed with a 7- or a 10-bit address. Own Address Register 2 can be programmed with a 7-bit address, but the Least Significant Bits of this address can be masked through the OA2MSK register, in order to acknowledge multiple slave addresses. The two Own Address Registers can be enabled simultaneously.

# Wakeup from Stop mode on address match

- When I2CCLK clock is set to HSI or CSI, the I<sup>2</sup>C is able to wake up the MCU from Stop mode when it receives its slave address. All addressing modes are supported.



The I<sup>2</sup>C peripheral supports Wakeup from Stop mode on address matches. To do this, the I<sup>2</sup>C peripheral clock must be set to the HSI or the CSI oscillator. Only the analog noise filter is supported when the Wakeup from Stop feature is enabled. All addressing modes are supported.

When the device is in Stop mode, the high-speed internal oscillator is switched off. When a Start condition is detected, the I<sup>2</sup>C peripheral enables the high-speed internal oscillator, which is used to receive the address on the bus.

After an address is received in Stop mode, a wakeup interrupt is generated if the address matches the programmed slave address.

If the address does not match, the high-speed internal oscillator is switched off, no interrupt is generated, and the device remains in Stop mode.

Clock stretching must be enabled because the I<sup>2</sup>C peripheral stretches the clock line low after the Start condition, until the high-speed internal oscillator is started. After having received

an address that matches the programmed slave address, the I<sup>2</sup>C peripheral also stretches the clock line low until the STM32MP1 device is awoken.

# Simple master mode management

For payloads  $\leq 255$  bytes: only 1 write action needed!

- START = 1
- SADD: slave address
- RD\_WRN: transfer direction
- NBYTES = N: number of bytes to be transferred
- AUTOEND = 1: STOP automatically sent after N data

AUTOEND	Description
0: Software end mode	End of transfer software control after NBYTES bytes of data are transferred: Transfer Complete (TC) flag is set and an interrupt generated if enabled. A Restart or Stop condition can be requested by software.
1: Automatic end mode	Stop condition is automatically sent after NBYTES bytes of data are transferred.



Master mode software management is very simple. Only one write action is needed to handle a master transfer with a payload smaller than 255 bytes. The full protocol is managed by the hardware.

In order to start a transfer in Master mode, I<sup>2</sup>C Control Register 2 must be written with the Start condition request, the slave address, the transfer direction, the number of bytes to be transferred, and the End of Transfer mode. End of Transfer mode is configured by the AUTOEND bit. If it is set, the Stop condition is automatically sent after the programmed number of bytes is transferred.

If the AUTOEND bit is not set, the end of transfer is managed by software. After the programmed number of bytes is transferred, the Transfer Complete (TC) flag is set and an interrupt is generated, if enabled. Then a Repeated Start or a Stop condition can be requested by software.

The data transfer can be managed by interrupts or by the DMA.

# Easy to use event management

- For payloads > 255: The RELOAD bit must be set in I2C\_CR2.
- AUTOEND = 0 has no effect when the RELOAD bit is set

RELOAD	Description
0: No reload	NBYTES bytes of data are transferred, followed by STOP or RESTART.
1: Reload mode	NBYTES is reloaded after NBYTES bytes of data are transferred: data transfer will resume. Transfer Complete Reload (TCR) flag is set and an interrupt generated if enabled.



When the payload is greater than 255 bytes, the RELOAD bit must be set in I<sup>2</sup>C Control Register 2. In this case, the Transfer Complete Reload (TCR) flag is set after the programmed number of bytes has been transferred. The additional number of bytes to be transferred is programmed when the TCR bit is set, and then, the data transfer will resume. The I<sup>2</sup>C clock is stretched low as long as the TCR bit is set. The RELOAD bit is used in Master mode when the payload is greater than 255 bytes, and in Slave mode when Slave Byte Control is enabled.

When the RELOAD bit is set, the AUTOEND bit has no effect.

- By default : I<sup>2</sup>C slaves use clock stretching. Clock stretching can be disabled by software.
- Reception : Acknowledge control can be done on selected bytes in Slave Byte Control (SBC) mode with RELOAD = 1
  - SBC = 1 enables the NBYTES counter in Slave mode (Tx and Rx modes).
  - SBC = 1 is allowed only when NOSTRETCH = 0.

SBC	Description
0: No reload	NBYTES bytes of data are transferred, followed by a STOP or RESTART.
1: Reload mode	NBYTES is reloaded after NBYTES bytes of data are transferred: data transfer will resume. Transfer Complete Reload (TCR) flag is set and an interrupt generated, if enabled.



By default, the I<sup>2</sup>C slave uses clock stretching. The clock stretching feature can be disabled by software.

In Receive mode, the slave acknowledge on received byte behavior can be configured when Slave Byte Control mode is selected, together with the RELOAD bit being set. When the SBC bit is set, the number of bytes counter is enabled in Slave mode. Clock stretching must be enabled when Slave Byte Control is enabled.

In Receive mode, when Slave Byte Control is enabled with the RELOAD bit set and the number of bytes to be transferred is 1, the Transfer Complete Reload flag is set after each received byte and the SCL line is stretched. This is done after data reception and before the acknowledge pulse. The Receive Buffer Not Empty flag is also set, so the data can be read. In the TCR subroutine, an Acknowledge or NOT Acknowledge can be programmed to be sent after the byte is received.

It is recommended to clear the SBC bit in transmission, as

there is no use for the byte counter in I<sup>2</sup>C Slave Transmitter mode.

In SMBus mode, Slave Byte Control mode is used in transmission for sending the PEC (packet error code) byte.

## Seamless SMBus 2.0 support

- ARP (Address Resolution Protocol): Device default address, Arbitration in slave mode
- Host Notify protocol support: host address
- Alert support: Alert pin and Alert Response support
  - Timeout and bus idle detection
- Command and data acknowledge control in SBC mode
- Packet Error Checking (PEC) hardware calculation



The I<sup>2</sup>C peripheral provides hardware support for the SMBus. The SMBus Address resolution protocol is supported through the device default address and arbitration in Slave mode.

The Host Notify protocol is supported with host address support.

The Alert protocol is supported through the SMBus Alert pin and Alert Response address.

The SMBus clock low timeout and Cumulative clock low extend times can be detected, with a programmable duration. The Bus Idle condition can be detected with a programmable duration.

Command and data acknowledge control is supported through Slave Byte Control mode.

The Packet Error Code (PEC) byte is calculated by hardware.

# SMBus: PEC (Packet Error Checking)

- Automatic PEC sending/checking
- NBYTES (data transfer counter) is used to:
  - Automatically checks the Packet Error Code (PEC) byte in reception, after NBYTES-1 bytes are received
    - Automatic NACK sending in case of failure
  - Automatically sends the Packet Error Code (PEC) byte in transmission, after NBYTES-1 bytes are sent
  - Slave Byte Control mode (SBC bit) must be set to Slave mode to enable the NBYTES counter



The Packet Error Code (PEC) byte is automatically sent in transmission, and checked in reception.

The data transfer counter, initialized with the NBYTES value, is used to automatically check the PEC byte in reception, after NBYTES minus one byte are received. If the received PEC byte does not match the calculation, a Not Acknowledge is automatically sent after the PEC byte. In transmission, the internally calculated PEC byte is automatically sent after NBYTES minus one byte. Slave Byte Control mode must be enabled in Slave mode in order to enable the NBYTES counter and allow automatic PEC reception or transmission.

Interrupt event	Description
Receive buffer Not Empty	Set when the Receive buffer contains received data and is ready to be read.
Transmit buffer interrupt Status	Set when the Transmit buffer is empty and is ready to be written.
Stop detection	Set when a Stop condition is detected on the bus.
Transfer Complete Reload	Set when RELOAD = 1 and NBYTES bytes of data have been transferred.
Transfer Complete	Set when RELOAD = 0 AUTOEND = 0 and NBYTES bytes of data have been Transferred.
Address matched	Set when the received slave address matches one of the enabled slave addresses.
NACK reception	Set by hardware when a NACK is received after a byte transmission.

- DMA requests can be generated when the Receive Buffer is *Not Empty* or when the Transmit buffer is *Empty*



Several events can trigger an interrupt:

The Receive Buffer Not Empty flag is set when the receive buffer contains received data and is ready to be read. The Transmit Buffer interrupt status is set when the transmit buffer is empty and is ready to be written. The Stop Detection flag is set when a Stop condition is detected on the bus.

The Transfer Complete Reload flag is set when the RELOAD bit is set and NBYTES bytes of data have been transferred.

The Transfer Complete flag is set when the RELOAD and AUTOEND bits are cleared and NBYTES bytes of data have been transferred.

The Address Match flag is set when the received slave address matches one of the enabled slave addresses.

The NACK reception flag is set when a Not Acknowledge is received after a byte transmission.

DMA requests can be generated when the Receive Buffer Not Empty or Transmit Buffer Empty flag is set.

Interrupt event	Description
<b>Bus Error Detection</b>	Set when a misplaced Start or Stop condition is detected.
<b>Arbitration Loss</b>	Set in the event of an arbitration loss.
<b>Overrun/Underrun error</b>	Set in Slave mode with NOSTRETCH = 1, when new data is received while the previous byte has not yet been read, or when new data must be transmitted while it is not written yet.
<b>SMBus: PEC error</b>	Set when the received PEC does not match the PEC register content.
<b>SMBus: Timeout error</b>	Set when a timeout or extended clock timeout occurred.
<b>SMBus: Alert pin detection</b>	Set when SMBHEN = 1 (SMBus host configuration), ALERTEN = 1 and a SMBALERT event (falling edge) is detected on SMBA pin.

Several errors flags can be generated.

A Bus Error Detection flag is set when a misplaced Start or Stop condition is detected. The Arbitration Loss flag is set in the event of an arbitration loss. An Overrun or Underrun Error flag is set in Slave mode with clock stretching disabled, when an overrun or an underrun error is detected.

In SMBus mode, a PEC Error flag is set when the received PEC does not match the calculated PEC register content. A Timeout Error flag is set when a timeout or extended clock timeout is detected. An Alert pin detection flag is set in the SMBus Host configuration, when Alert is enabled and a falling edge is detected on the SMBA pin.

Mode	Description
<b>CRun</b>	Active.
<b>CSleep (MPU or MCU sub-system state)</b>	Active. Peripheral interrupts cause the device to exit CSleep mode.
<b>Stop + LPStop</b>	Registers content is kept. If the I2C is clocked by HSI or CSI clock, the address recognition is functional. The I2C address match condition causes the device to exit the Stop mode.
<b>LPLV-Stop</b>	Registers content is kept but the I2C is not functional.
<b>Standby</b>	Powered-down. The peripheral must be reinitialized after exiting Standby mode.



Here is an overview of the I2C instance status in specific low-power modes. This status depends on state of MPU or MCU sub-system domains and to which of them the peripheral instance is allocated.

The I<sup>2</sup>C peripheral is active in CRun and CSleep modes. In Stop and LP-Stop modes, the peripheral registers content is kept and if the I<sup>2</sup>C peripheral is clocked by HSI or CSI clock, the address recognition is functional. The I2C address match condition causes the device to exit Stop mode. In LPLV-Stop mode, the peripheral is no more functional but the registers content is kept. In Standby mode, the peripheral is powered down and must be reinitialized after exiting Standby mode.

- I2Cx SMBUS timeout counter can be stopped when core is halted



For each I<sup>2</sup>C peripheral, a bit is available for debugging purposes in the Debug Component that can be used to stop the SMBUS timeout counter when the core is halted.

# STM32MP1 instances features

I <sup>2</sup> C features	I2C1	I2C2	I2C3	I2C4	I2C5	I2C6
7-bit addressing mode	X	X	X	X	X	X
10-bit addressing mode	X	X	X	X	X	X
Standard-mode (up to 100 Kbit/s)	X	X	X	X	X	X
Fast-mode (up to 400 Kbit/s)	X	X	X	X	X	X
Fast-mode Plus with 20 mA output drive IOs (up to 1 Mbit/s)	X	X	X	X	X	X
Independent clock	X	X	X	X	X	X
Wakeup from Stop on address match	X	X	X	X	X	X
SMBus	X	X	X	X	X	X
Optional runtime allocation	MPU MCU	MPU MCU	MPU MCU	MPU only	MPU MCU	MPU only



x: supported

STM32MP1 microcontrollers embed six I<sup>2</sup>C peripherals, all with the same set of features. I2C1, I2C2, I2C3 and I2C5 can be controlled by the Cortex-M4 core, either directly or through DMA1/2, but it can also be managed by the Cortex-A7 CPU subsystem as well. I2C4 and I2C6 are secure communication blocks dedicated to the Cortex-A7 core.

- For more information, refer to these trainings linked to this peripheral:
  - System Configuration Controller (SYSCFG)
  - Reset and Clock Controller (RCC)
  - Power controller (PWR)
  - Interrupts (NVIC and GIC)
  - Direct memory access controller (DMA)



For more information related to this peripheral, you can also refer to these trainings:

- System configuration controller
- Reset and Clock controller
- Power controller
- Interrupts controller
- Direct memory access controller

- For more details, please refer to following webpages:
  - [www.nxp.com](http://www.nxp.com) :
    - UM10204 I2C-bus specification and user manual
  - [www.sbs-forum.org](http://www.sbs-forum.org)
    - System Management Bus (SMBus) Specification
  - <http://www.powersig.org/>
    - PMBus™ Power System Management Protocol Specification



For more details, please refer to the I<sup>2</sup>C-bus specification and user manual from the NXP web site.  
The SMBus specification can be found in the Smart Battery System implementers forum.  
The PMBus Power System Management Protocol specification can be found in the Power Management Bus implementers forum.