Hello, and welcome to this presentation of the STM32 I²C 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.
The I²C interface is compliant with the NXP I²C-bus specification and user manual, Revision 3; the SMBus System Management Bus Specification, Revision 2; and the PMBus Power System Management Protocol Specification, Revision 1.1.

This peripheral provides an easy-to-use interface, with very simple software programming, and full timing flexibility. Additionally, the I²C peripheral is functional in low-power stop modes.
The I²C peripheral supports multi-master and slave modes. The I²C IO pins must be configured in open-drain mode. The logic high level is driven by an external pull-up. The I²C alternate functions are available on IO pins supplied by VDD, which can be from 1.71 to 3.6 volts, and on IO pins supplied by VDDIO2, which can be from 1.08 to 3.6 volts. This allows communication with external chips at voltages different from the STM32L4 main power supply. A typical use case is communication with an application processor in sensor hub applications. The IO pins support the 20 mA output drive required for Fast mode Plus. The peripheral controls all I²C 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 supports slave clock stretching and clock stretching from slave can be disabled by software.
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 I²C block diagram. The registers are
accessed through the APB bus, and the peripheral is
clocked with the I²C clock, which is independent from the
APB clock. The I²C clock can be selected between the
system clock, APB clock and the high-speed internal 16
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.
The STM32L4 embeds noise filters on I²C data and clock lines. The analog noise filters can filter spikes up to 50 ns and can be enabled or disabled by software. By default, analog noise filters are enabled. The digital noise filters can be enabled on the SDA and SCL lines instead of the analog noise filters. These filters suppress spikes with a programmable length from 1 to 15 I²C clock periods. The digital filters offer an extra filtering capability compared to the 50 ns required by the I²C standard. The digital filter value is fixed by software, while the analog filter value may vary with process, temperature and voltage. Take care that the digital filter is disabled by hardware when the Wakeup from Stop feature is enabled. In this case, only the analog filter can be enabled.
The I²C setup and hold times can be configured by software through the I²C Timing register. The SDADEL and SCLDEL counters are used during transmission, in order to guarantee the minimum Data Hold and Data Setup times.

The I²C 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 I²C clock. However, the setup time is not impacted by these internal delays.
The I²C master clock’s low- and high-level durations are configured by software in the I²C 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 I²C clock.
The I²C 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.
The I²C peripheral supports Wakeup from Stop mode on address matches. To do this, the I²C peripheral clock must be set to the high-speed internal 16 MHz RC 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 I²C 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 I²C 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 I²C peripheral also stretches the clock line low until the STM32L4 device is woken up.
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, I²C 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

---

### AUTOEND Table

<table>
<thead>
<tr>
<th>AUTOEND</th>
<th>Description</th>
</tr>
</thead>
</table>
| 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 |
software.
The data transfer can be managed by interrupts or by the DMA.
When the payload is greater than 255 bytes, the RELOAD bit must be set in I²C 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 I²C clock is stretched low as long as TCR 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, the $I^2C$ slave uses clock stretching. The clock stretching feature can be disabled by software.

In reception, 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 reception, 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 SCL 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 I²C Slave Transmitter mode. In SMBus mode, Slave Byte Control mode is used in transmission for sending the PEC (packet error code) byte.
The I²C peripheral provides hardware support for 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.

### 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 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.
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.

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

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

<table>
<thead>
<tr>
<th>Interrupt event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bus Error Detection</td>
<td>Set when a misplaced Start or Stop condition is detected</td>
</tr>
<tr>
<td>Arbitration Loss</td>
<td>Set in the event of an arbitration loss</td>
</tr>
<tr>
<td>Overrun/Underrun error</td>
<td>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.</td>
</tr>
<tr>
<td>SMBus: PEC error</td>
<td>Set when the received PEC does not match the PEC register content</td>
</tr>
<tr>
<td>SMBus: Timeout error</td>
<td>Set when a timeout or extended clock timeout occurred</td>
</tr>
<tr>
<td>SMBus: Alert pin detection</td>
<td>Set when SMBHEN=1 (SMBus host configuration), ALERTEN=1 and a SMB ALERT event (falling edge) is detected on SMBA pin</td>
</tr>
</tbody>
</table>
The I²C peripheral is active in Run, Low-power run, Sleep, and Low-power sleep modes. I²C interrupts cause the device to exit Sleep or Low-power sleep modes. Address detection is active is Stop modes. The I²C peripheral generates a wakeup interrupt in the event of an address match. In Standby and Shutdown modes, the peripheral is powered down and must be reinitialized after exiting Standby or Shutdown mode.
For each I²C peripheral, a bit is available for debugging purposes in the MCU Debug Component that can be used to stop the SMBUS timeout counter when the core is halted.
Here is an example of a sensor hub application that requires several I²C peripherals. One or several I²C masters are used to interface with external sensors. It is possible to configure the STM32L4 device in Low-power sleep mode for this purpose, in order to reduce power consumption during data transfers. This is called Batch Acquisition Mode.

Only the required communication peripheral + 1 DMA + 1 SRAM are configured with the clock enabled in Low-power sleep mode.

The Flash memory is put in power-down mode and the Flash clock is gated off during Low-power sleep mode.

Note that the I²C clock can be at 16 MHz even in Low-power sleep mode, allowing support for 1 MHz Fast-mode Plus.

An I²C slave is used to communicate with the application processor. The STM32L4 device can be put in Stop mode and wakes up only when addressed by the...
application processor.
The STM32L4 devices embed up to four I²C peripherals, with this full set of features. Wakeup from Stop 1 mode is supported by all I²C instances. Wakeup from Stop 2 mode is supported only by I2C3.

![STM32L4 instances features table]

<table>
<thead>
<tr>
<th>Feature</th>
<th>PC1</th>
<th>PC2</th>
<th>PC3</th>
<th>PC4 (*)</th>
</tr>
</thead>
<tbody>
<tr>
<td>7-bit addressing mode</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>10-bit addressing mode</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Standard-mode (up to 100 kbit/s)</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Fast-mode (up to 400 kbit/s)</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Fast-mode Plus with 20mA output drive I/Os</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>(up to 1 Mbit/s)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Independent clock</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>SMBus</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wakeup from Stop 1 mode</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wakeup from Stop 2 mode</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

*: supported

(*) only on STM32L49x/4Ax and STM32L45x/46x devices
Related peripherals

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

For more information related to this peripheral, you can also refer to these peripheral trainings:
- System configuration controller
- Reset and Clock controller
- Power controller
- Interrupts controller
- Direct memory access controller
For more details please refer to the I²C-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.