Embedded RTOS System
Inter-Integrated Circuit
(I2C, I^{2}C, IIC)

Hung-Yi Lin, Ph.D.
Dept. of Electrical Engineering
National University of Kaohsiung

2025/04/10

I2C Bus

- Designed for low-cost, medium data rate applications.

  (Phillips Semiconductor, 1980s)

Characteristics:

- Serial, byte-oriented;
- Multiple-master;
- Moderate speeds:
  - Standard mode: 100Kbits/s
  - Fast mode: 400Kbits/s
  - High speed mode: 3.4 Mbits/s

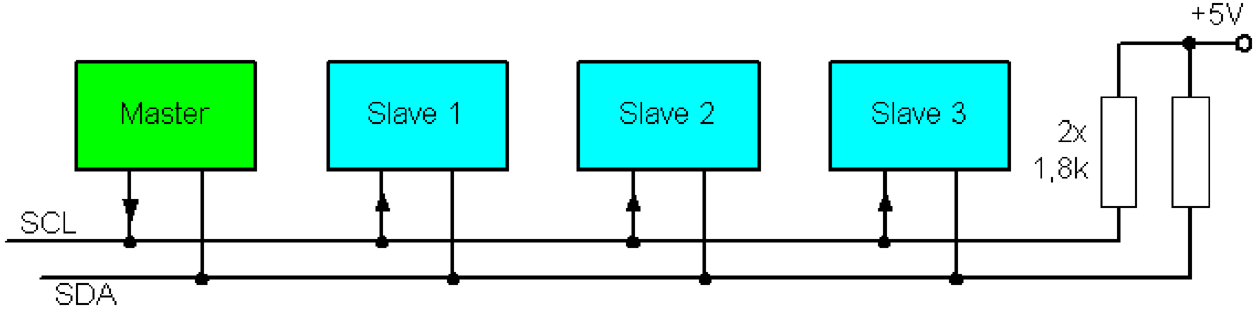
Many microcontrollers come with built-in I2C controllers.

Ref: 成大資工 I2C wiki

I2C Compatibility

- System Management Bus (SMBus)
- Power Management Bus (PMBus)
- Intelligent Platform Management Interface (IPMI)
- Display Data Channel (DDC)
- Advanced Telecom Computing Architecture, (ATCA)

- I²C uses a 7-bit address space but reserves 16 addresses,
  - It can communicate with up to **112 nodes** on a single bus.
  - New generation of I²C bus can communicate with more nodes (support 10-bit address space)



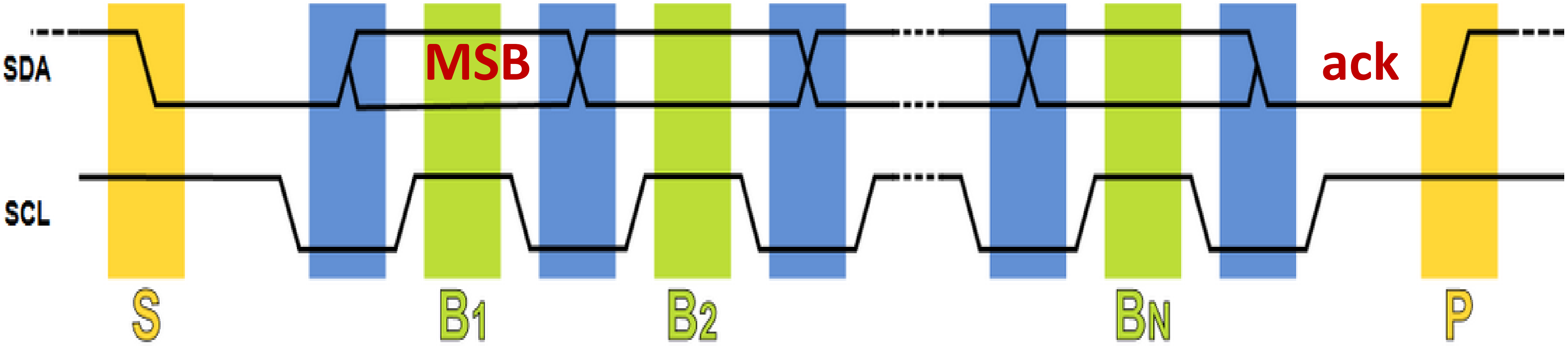
I2C Signaling

- Bus = “wired-AND” configuration
  - Open collector/drain drivers on SDA & SCL
  - Resistor pulls bus up to logic 1. (R: 1K~4.7K)
  - Any sender can pull the bus down to 0, even if other senders are trying to drive the bus to 1.
  - Sender “releases” SDA by disabling its driver, allowing SDA to be pulled up to logic 1
- Data on SDA must be stable while SCL high
  - Data on SDA is sampled while SCL is high
  - SDA may change only while SCL low
- Exceptions:
  - SDA 1->0 while SCL=1 signals START condition
  - SDA 0->1 while SCL=1 signals STOP condition

I2C Data Format

Start:
SDA 1->0 while SCL=1

Stop:
SDA 0->1 while SCL=1



SDA stable
while SCL=1

Transmit 8-bit byte (MSB first)

Four I2C operating Modes

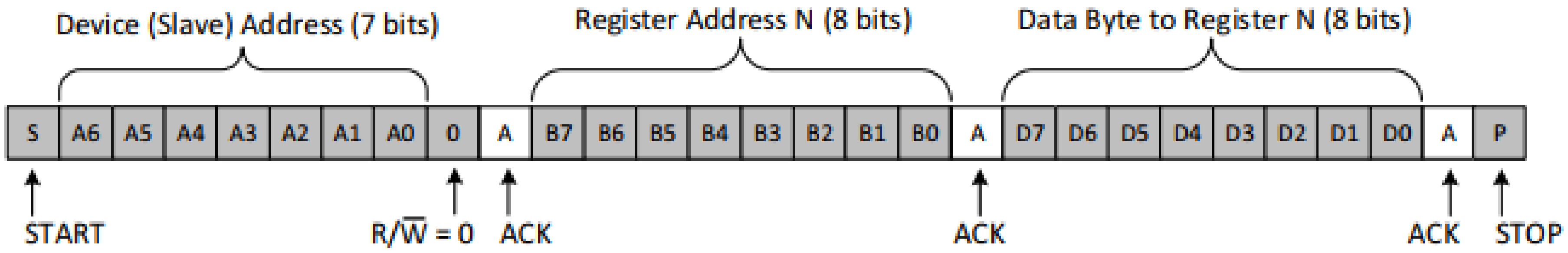
- master transmit
  - master node is sending data to a slave
  - Module issues START and ADDRESS, and then transmits data to the addressed slave device
- master receive
  - master node is receiving data from a slave
  - Module issues START and ADDRESS, and receives data from the addressed slave device
- slave transmit
  - slave node is sending data to the master
  - Another master issues START and the ADDRESS of this module, which then sends data to the master

Four I2C operating Modes

- **slave receive**
  - **slave node is receiving data from the master**
  - **Another master issues START and the ADDRESS of this module, which then receives data from the master.**
- ***Some devices only support slave modes – sensors, memories, etc***

Basic Data Formats

Write to slave device



Device (Slave) Address (7 bits)

Register Address N (8 bits)

Data Byte to Register N (8 bits)

S	A6	A5	A4	A3	A2	A1	A0	0	A	B7	B6	B5	B4	B3	B2	B1	B0	A	D7	D6	D5	D4	D3	D2	D1	D0	A	P

START

R/$\overline{W}$ = 0

ACK

ACK

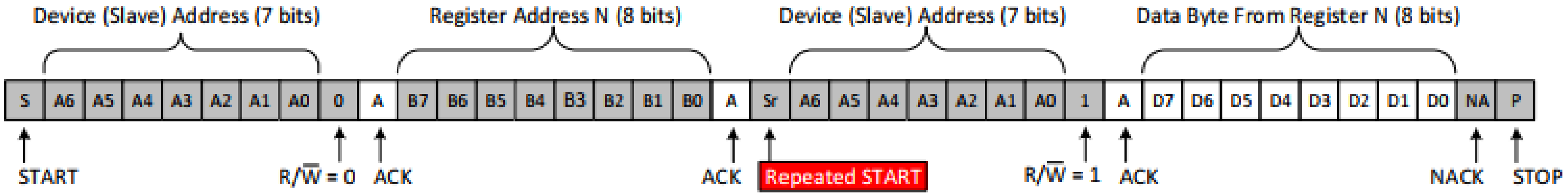
ACK

STOP

▨ Master Controls SDA Line

□ Slave Controls SDA Line

Read from slave device



Device (Slave) Address (7 bits)

Register Address N (8 bits)

Device (Slave) Address (7 bits)

Data Byte From Register N (8 bits)

S	A6	A5	A4	A3	A2	A1	A0	0	A	B7	B6	B5	B4	B3	B2	B1	B0	A	Sr	A6	A5	A4	A3	A2	A1	A0	1	A	D7	D6	D5	D4	D3	D2	D1	D0	NA	P

START

R/$\overline{W}$ = 0

ACK

ACK

Repeated START

R/$\overline{W}$ = 1

ACK

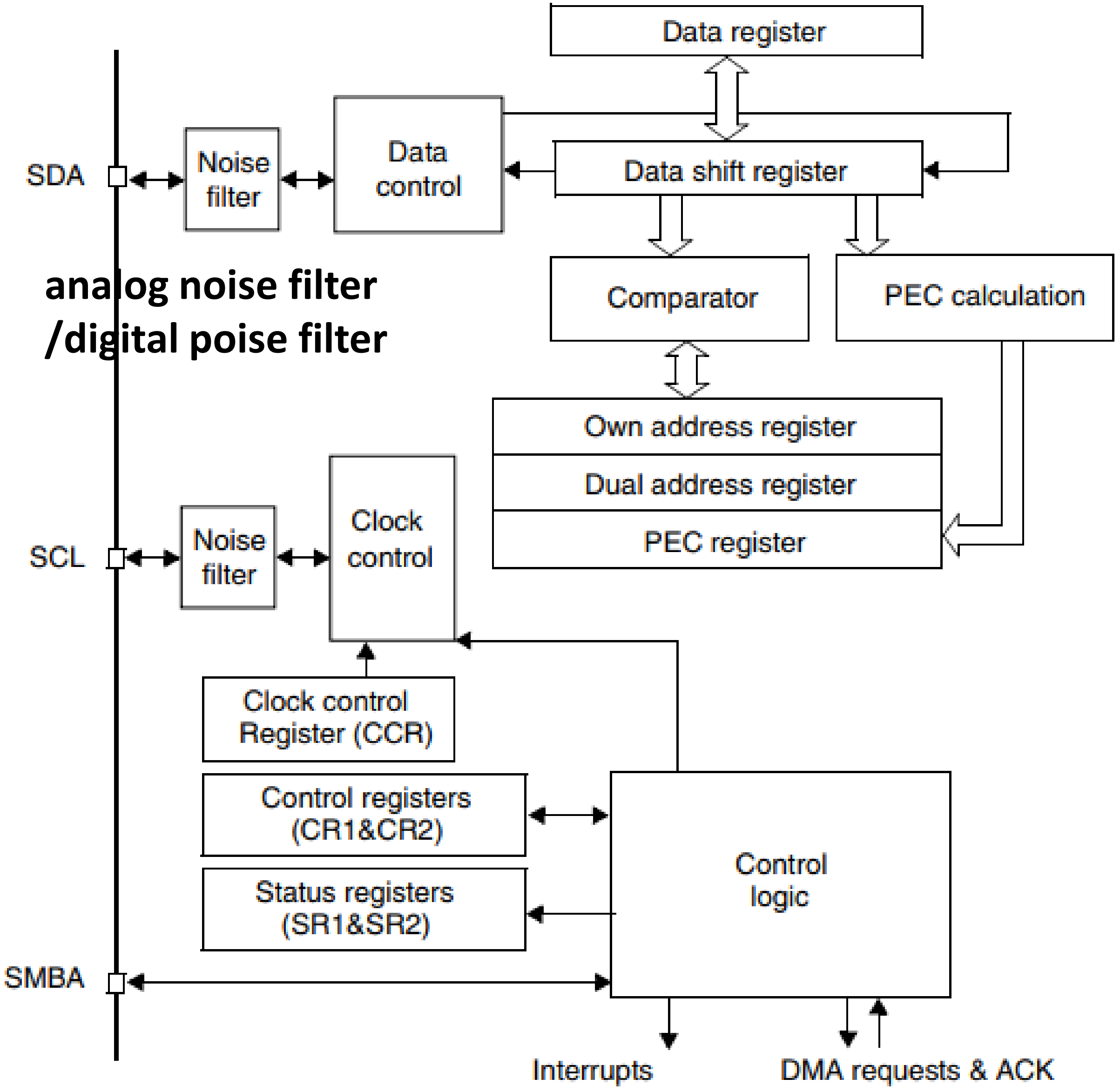
NACK

STOP

STM32 I2C Module

- Standard I2C compliant bus interface.
  - All I2C bus-specific sequencing, protocol, arbitration, timing
  - 7-bit and 10-bit addressing
  - Standard (≤ 100KHz) or Fast (≤ 400KHz) speed modes
  - Multi-master capability – use as master or slave
- Also supports standards:
  - SMBus (System Management Bus)
  - PMBus (Power Management Bus)
- DMA support – between memory and data register
- 2 interrupt vectors – data transfer complete and errors

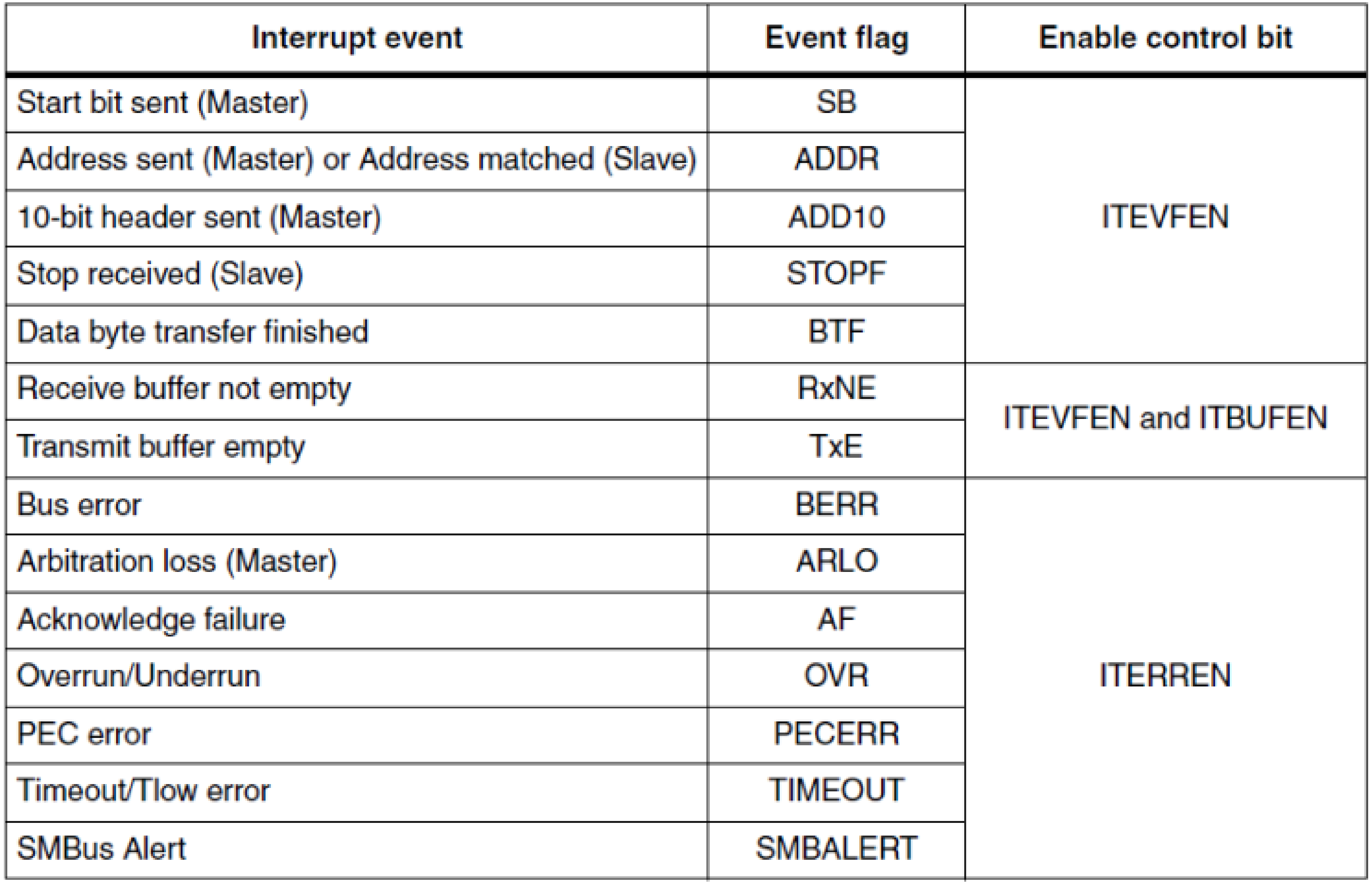
STM32 I2C Module



analog noise filter
/digital poise filter

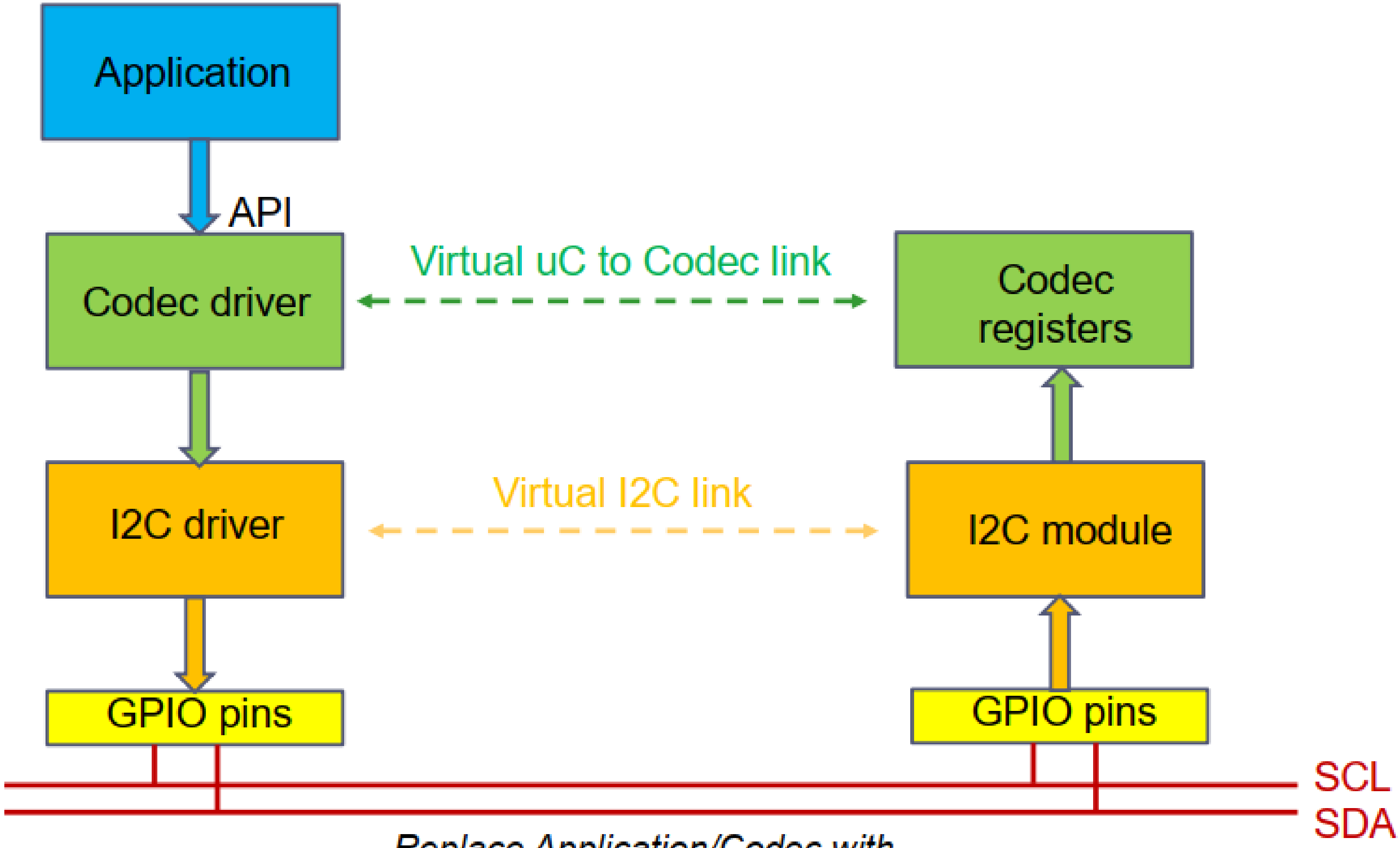
Packet Error Code

STM32 I2C Interrupts



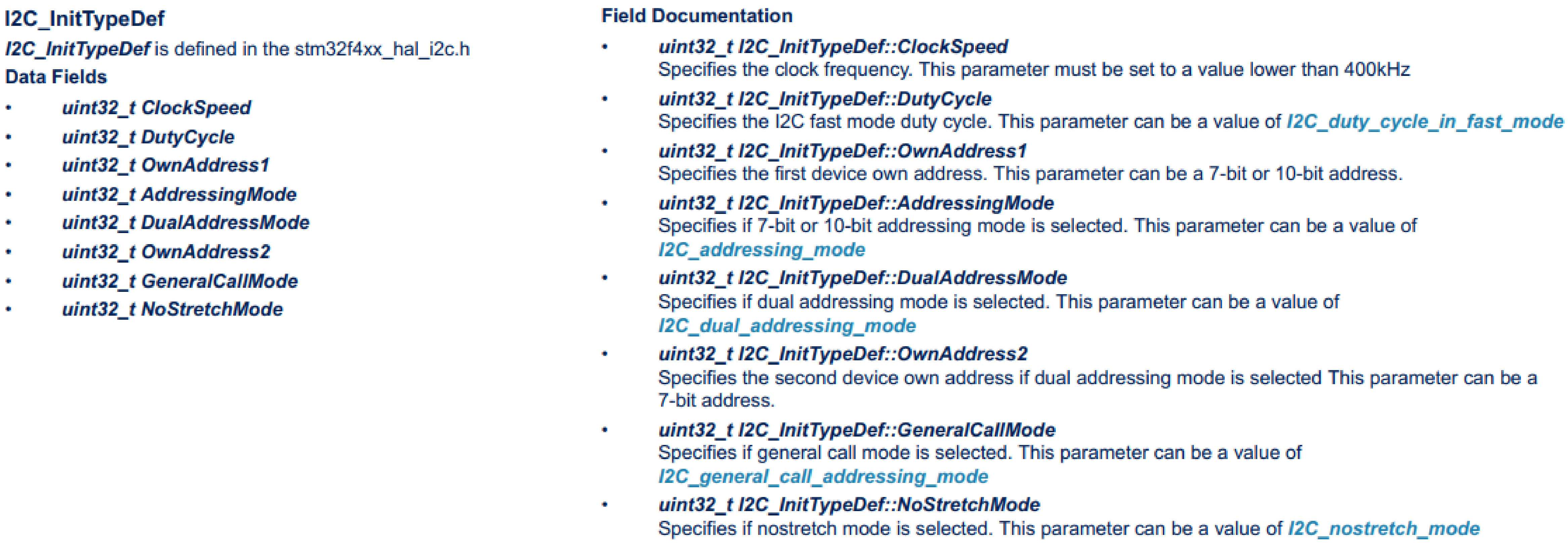
Interrupt event	Event flag	Enable control bit
Start bit sent (Master)	SB	ITEVFEN
Address sent (Master) or Address matched (Slave)	ADDR	
10-bit header sent (Master)	ADD10	
Stop received (Slave)	STOPF	
Data byte transfer finished	BTF	
Receive buffer not empty	RxNE	ITEVFEN and ITBUFEN
Transmit buffer empty	TxE	
Bus error	BERR	ITERREN
Arbitration loss (Master)	ARLO	
Acknowledge failure	AF	
Overrun/Underrun	OVR	
PEC error	PECERR	
Timeout/Tlow error	TIMEOUT	
SMBus Alert	SMBALERT	

Hierarchical Module Design



Replace Application/Codec with other functions that use I2C driver

I2C initType Structure



I2C_InitTypeDef

I2C_InitTypeDef is defined in the stm32f4xx_hal_i2c.h

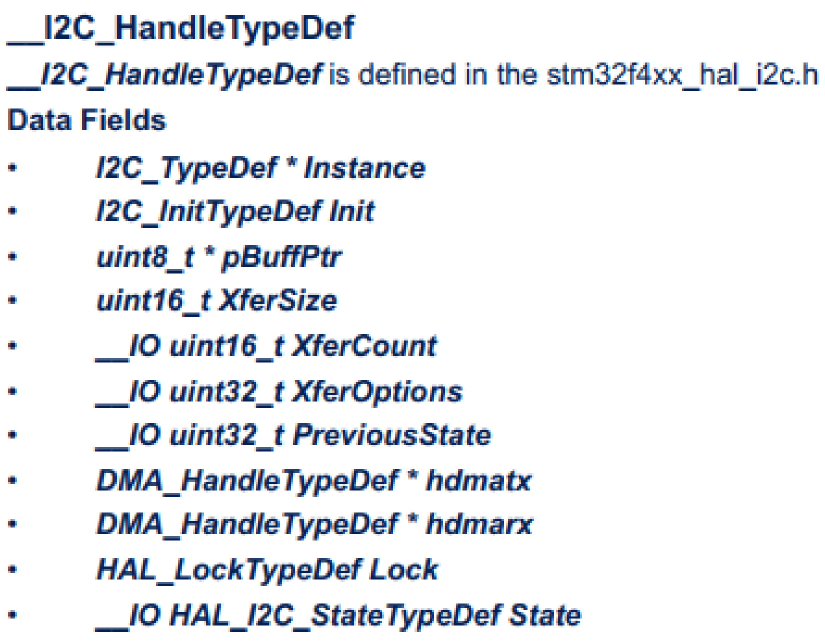
Data Fields

- ***uint32_t ClockSpeed***
- ***uint32_t DutyCycle***
- ***uint32_t OwnAddress1***
- ***uint32_t AddressingMode***
- ***uint32_t DualAddressMode***
- ***uint32_t OwnAddress2***
- ***uint32_t GeneralCallMode***
- ***uint32_t NoStretchMode***

Field Documentation

- ***uint32_t I2C_InitTypeDef::ClockSpeed***
  Specifies the clock frequency. This parameter must be set to a value lower than 400kHz
- ***uint32_t I2C_InitTypeDef::DutyCycle***
  Specifies the I2C fast mode duty cycle. This parameter can be a value of ***I2C_duty_cycle_in_fast_mode***
- ***uint32_t I2C_InitTypeDef::OwnAddress1***
  Specifies the first device own address. This parameter can be a 7-bit or 10-bit address.
- ***uint32_t I2C_InitTypeDef::AddressingMode***
  Specifies if 7-bit or 10-bit addressing mode is selected. This parameter can be a value of ***I2C_addressing_mode***
- ***uint32_t I2C_InitTypeDef::DualAddressMode***
  Specifies if dual addressing mode is selected. This parameter can be a value of ***I2C_dual_addressing_mode***
- ***uint32_t I2C_InitTypeDef::OwnAddress2***
  Specifies the second device own address if dual addressing mode is selected This parameter can be a 7-bit address.
- ***uint32_t I2C_InitTypeDef::GeneralCallMode***
  Specifies if general call mode is selected. This parameter can be a value of ***I2C_general_call_addressing_mode***
- ***uint32_t I2C_InitTypeDef::NoStretchMode***
  Specifies if nostretch mode is selected. This parameter can be a value of ***I2C_nostretch_mode***

I2C HandleType Structure

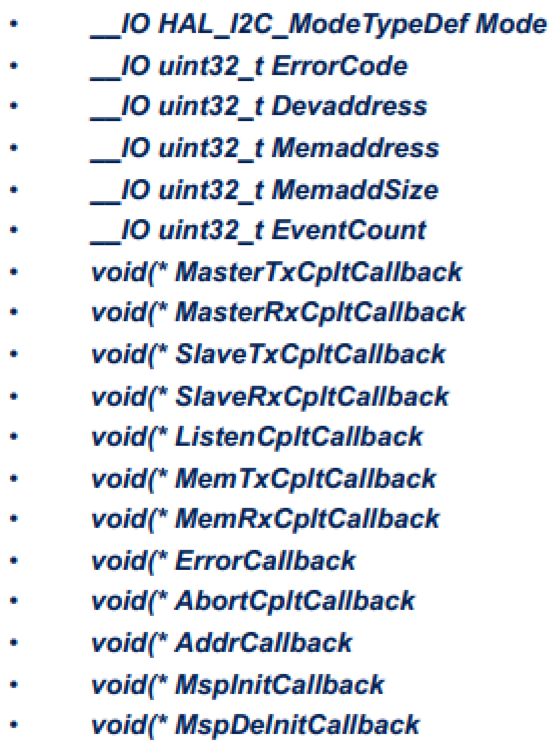


__I2C_HandleTypeDef

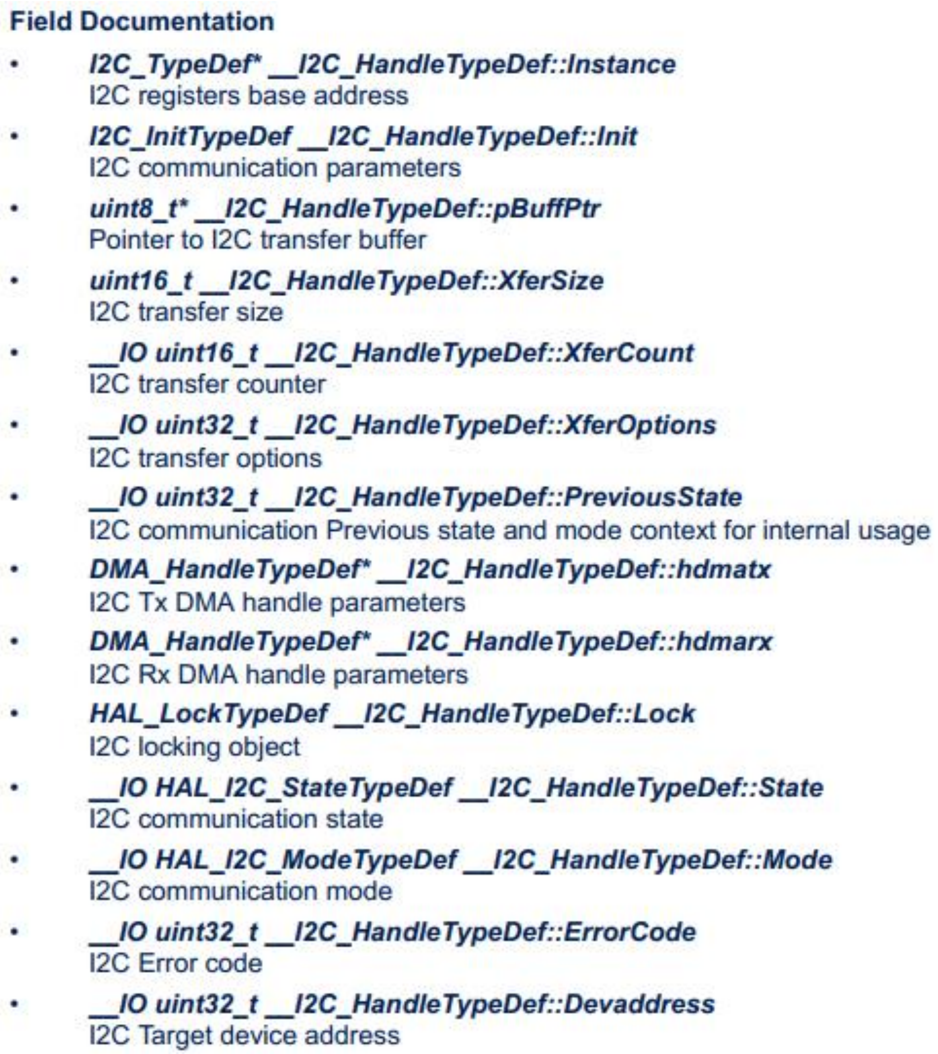
__I2C_HandleTypeDef is defined in the stm32f4xx_hal_i2c.h

Data Fields

- ***I2C_TypeDef * Instance***
- ***I2C_InitTypeDef Init***
- ***uint8_t * pBuffPtr***
- ***uint16_t XferSize***
- ***__IO uint16_t XferCount***
- ***__IO uint32_t XferOptions***
- ***__IO uint32_t PreviousState***
- ***DMA_HandleTypeDef * hdmatx***
- ***DMA_HandleTypeDef * hdmarx***
- ***HAL_LockTypeDef Lock***
- ***__IO HAL_I2C_StateTypeDef State***



- ***__IO HAL_I2C_ModeTypeDef Mode***
- ***__IO uint32_t ErrorCode***
- ***__IO uint32_t Devaddress***
- ***__IO uint32_t Memaddress***
- ***__IO uint32_t MemaddSize***
- ***__IO uint32_t EventCount***
- ***void(* MasterTxCpltCallback***
- ***void(* MasterRxCpltCallback***
- ***void(* SlaveTxCpltCallback***
- ***void(* SlaveRxCpltCallback***
- ***void(* ListenCpltCallback***
- ***void(* MemTxCpltCallback***
- ***void(* MemRxCpltCallback***
- ***void(* ErrorCallback***
- ***void(* AbortCpltCallback***
- ***void(* AddrCallback***
- ***void(* MspInitCallback***
- ***void(* MspDeInitCallback***

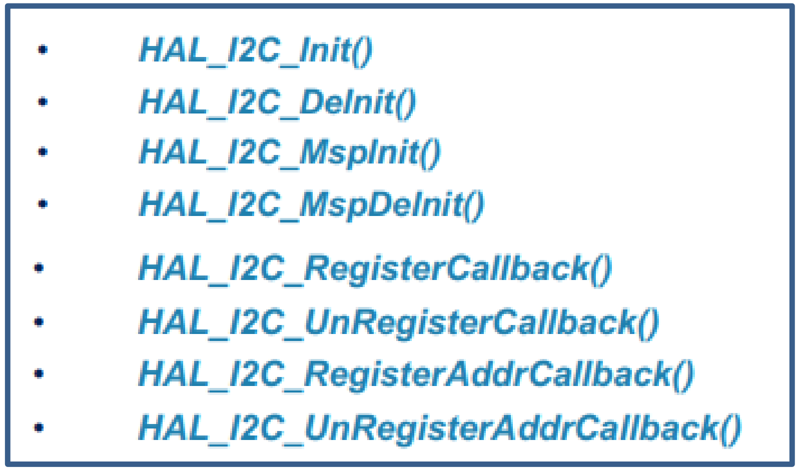


Field Documentation

- ***I2C_TypeDef* __I2C_HandleTypeDef::Instance***
  I2C registers base address
- ***I2C_InitTypeDef __I2C_HandleTypeDef::Init***
  I2C communication parameters
- ***uint8_t* __I2C_HandleTypeDef::pBuffPtr***
  Pointer to I2C transfer buffer
- ***uint16_t __I2C_HandleTypeDef::XferSize***
  I2C transfer size
- ***__IO uint16_t __I2C_HandleTypeDef::XferCount***
  I2C transfer counter
- ***__IO uint32_t __I2C_HandleTypeDef::XferOptions***
  I2C transfer options
- ***__IO uint32_t __I2C_HandleTypeDef::PreviousState***
  I2C communication Previous state and mode context for internal usage
- ***DMA_HandleTypeDef* __I2C_HandleTypeDef::hdmatx***
  I2C Tx DMA handle parameters
- ***DMA_HandleTypeDef* __I2C_HandleTypeDef::hdmarx***
  I2C Rx DMA handle parameters
- ***HAL_LockTypeDef __I2C_HandleTypeDef::Lock***
  I2C locking object
- ***__IO HAL_I2C_StateTypeDef __I2C_HandleTypeDef::State***
  I2C communication state
- ***__IO HAL_I2C_ModeTypeDef __I2C_HandleTypeDef::Mode***
  I2C communication mode
- ***__IO uint32_t __I2C_HandleTypeDef::ErrorCode***
  I2C Error code
- ***__IO uint32_t __I2C_HandleTypeDef::Devaddress***
  I2C Target device address

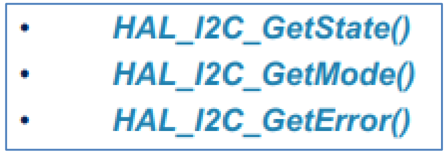
I2C Firmware API

I2C initialization/de-initialization



- *HAL_I2C_Init()*
- *HAL_I2C_DeInit()*
- *HAL_I2C_MspInit()*
- *HAL_I2C_MspDeInit()*
- *HAL_I2C_RegisterCallback()*
- *HAL_I2C_UnRegisterCallback()*
- *HAL_I2C_RegisterAddrCallback()*
- *HAL_I2C_UnRegisterAddrCallback()*

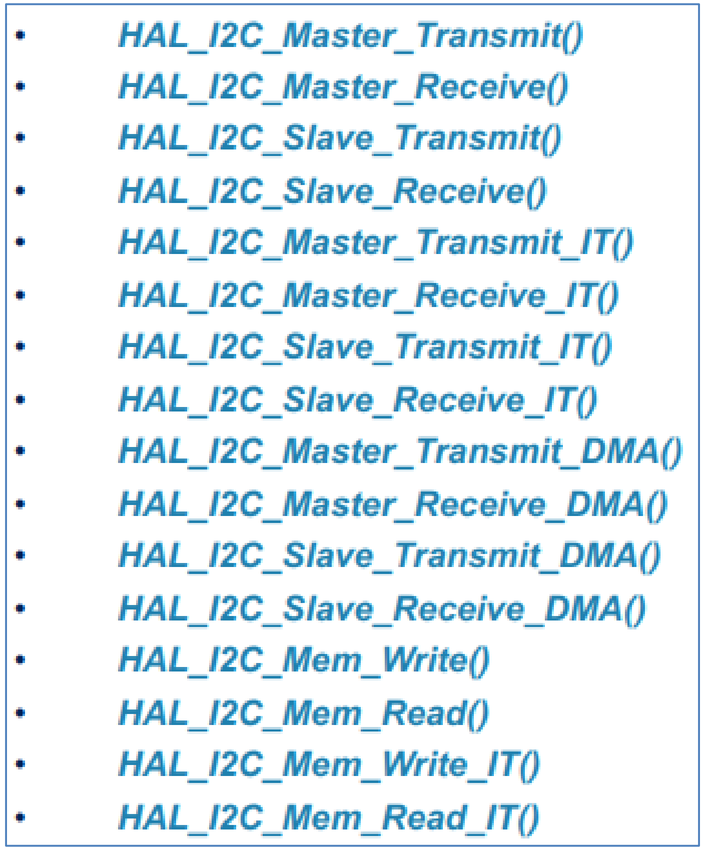
Peripheral state, mode and error functions



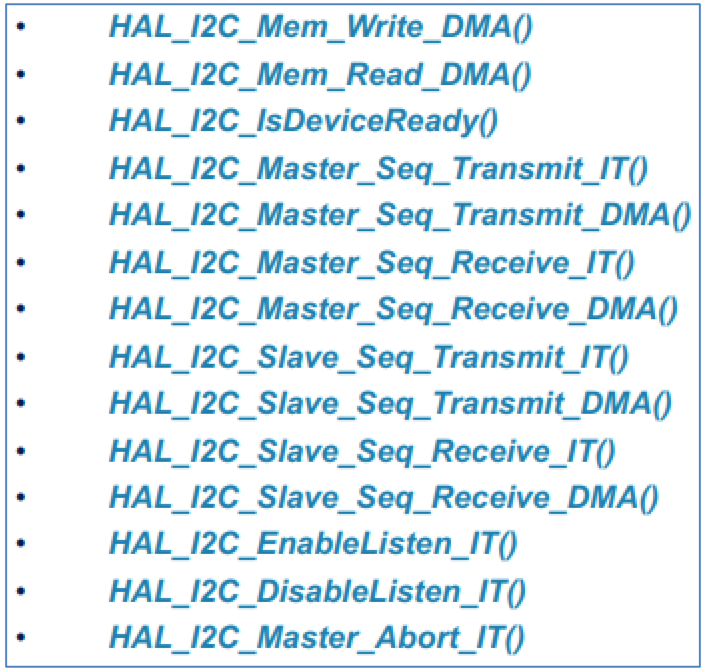
- *HAL_I2C_GetState()*
- *HAL_I2C_GetMode()*
- *HAL_I2C_GetError()*

I2C Firmware API

I2C IO operation



- *HAL_I2C_Master_Transmit()*
- *HAL_I2C_Master_Receive()*
- *HAL_I2C_Slave_Transmit()*
- *HAL_I2C_Slave_Receive()*
- *HAL_I2C_Master_Transmit_IT()*
- *HAL_I2C_Master_Receive_IT()*
- *HAL_I2C_Slave_Transmit_IT()*
- *HAL_I2C_Slave_Receive_IT()*
- *HAL_I2C_Master_Transmit_DMA()*
- *HAL_I2C_Master_Receive_DMA()*
- *HAL_I2C_Slave_Transmit_DMA()*
- *HAL_I2C_Slave_Receive_DMA()*
- *HAL_I2C_Mem_Write()*
- *HAL_I2C_Mem_Read()*
- *HAL_I2C_Mem_Write_IT()*
- *HAL_I2C_Mem_Read_IT()*



- *HAL_I2C_Mem_Write_DMA()*
- *HAL_I2C_Mem_Read_DMA()*
- *HAL_I2C_IsDeviceReady()*
- *HAL_I2C_Master_Seq_Transmit_IT()*
- *HAL_I2C_Master_Seq_Transmit_DMA()*
- *HAL_I2C_Master_Seq_Receive_IT()*
- *HAL_I2C_Master_Seq_Receive_DMA()*
- *HAL_I2C_Slave_Seq_Transmit_IT()*
- *HAL_I2C_Slave_Seq_Transmit_DMA()*
- *HAL_I2C_Slave_Seq_Receive_IT()*
- *HAL_I2C_Slave_Seq_Receive_DMA()*
- *HAL_I2C_EnableListen_IT()*
- *HAL_I2C_DisableListen_IT()*
- *HAL_I2C_Master_Abort_IT()*

Labs

- Practice 1: EEPROM 24LC02:
- Practice 2: OLED display:
  - https://hackmd.io/@hylin/Syj8CTUskg