Serial Data Transfer – UART / I2C

Computer Engineering 2

Motivation

School of Engineering
InES Institute of Embedded Systems

UART	SPI	I2C
serial ports (RS-232)	4-wire bus	2-wire bus
TX, RX opt. control signals	*MOSI, MISO, SCLK, SS*	*SCL, SDA*
point-to-point	point-to-multipoint	(multi-) point-to-multi-point
full-duplex	full-duplex	half-duplex
asynchronous	synchronous	synchronous
only higher layer addressing	slave selection through $\overline{SS}$ signal	7/10-bit slave address
parity bit possible	no error detection	no error detection
chip-to-chip, PC terminal program	chip-to-chip, on-board connections	chip-to-chip, board-to-board connections
The three interfaces provide the lowest layer of communication and require higher level protocols to provide and interpret the transferred data.		

Agenda

School of Engineering
InES Institute of Embedded Systems

- **Asynchronous Serial Interface**
  - Universal Asynchronous Receiver Transmitter – UART
  - Longer Distances: RS-232 / RS-485 – Electrical Characteristics
  - U(S)ART – STM32F4xxx
- **I2C Bus**
  - Protocol / Operation
  - I2C – STM32F4xxx

Learning Objectives

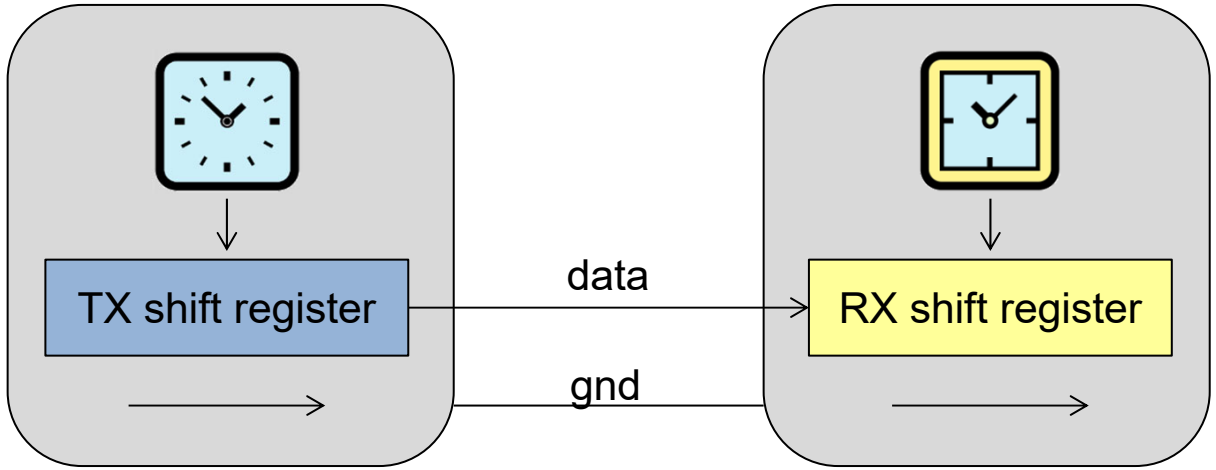
School of Engineering
InES Institute of Embedded Systems

At the end of this lesson, you will be able

- to explain how a UART works, and which synchronization mechanism is used between transmitter and receiver
- to draw and interpret UART timing diagrams including data and overhead bits
- to program the UART interfaces on the STM32F4
- to explain what I2C is and how it works
- to interpret an I2C timing diagram
- to program the I2C interfaces on the STM32F4

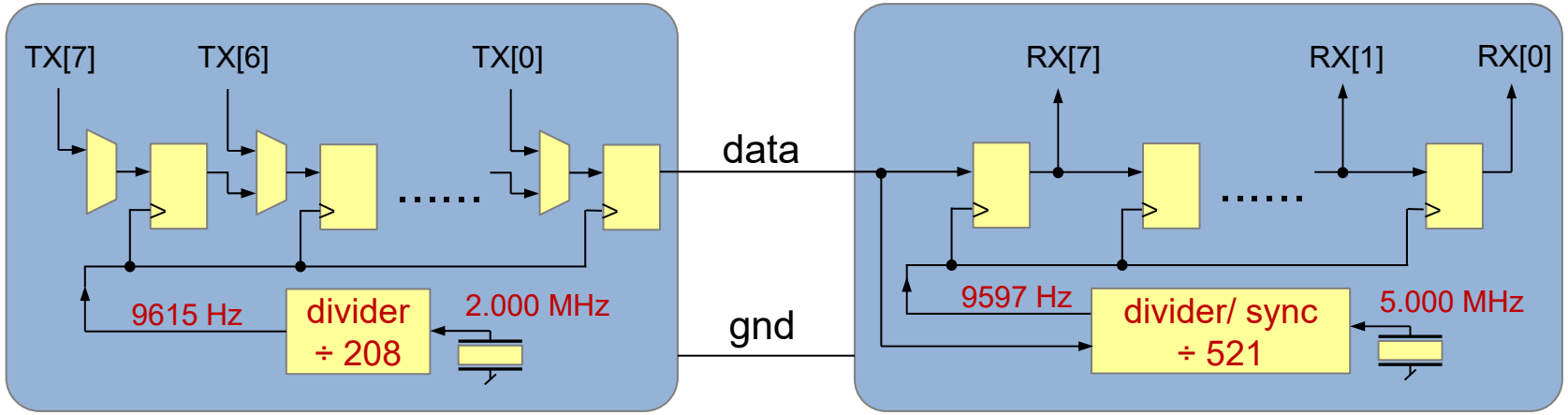
Asynchronous Serial Interface

- **Universal Asynchronous Receiver Transmitter – UART**
  - Connecting shift registers with diverging clock sources
    - Same target frequency
    - Different tolerances and divider ratios
    - Requires synchronization at start of each data item in receiver



Asynchronous Serial Interface

- **Implementation of UART → Shift register**
  - Example: 9'600 Baud with selected quartz frequencies



  - Transmitter and receiver use closest integer to divide clock
    - 2.000 ÷ 208 = 9615 Hz → slightly too fast
    - 5.000 ÷ 521 = 9597 Hz → slightly too slow

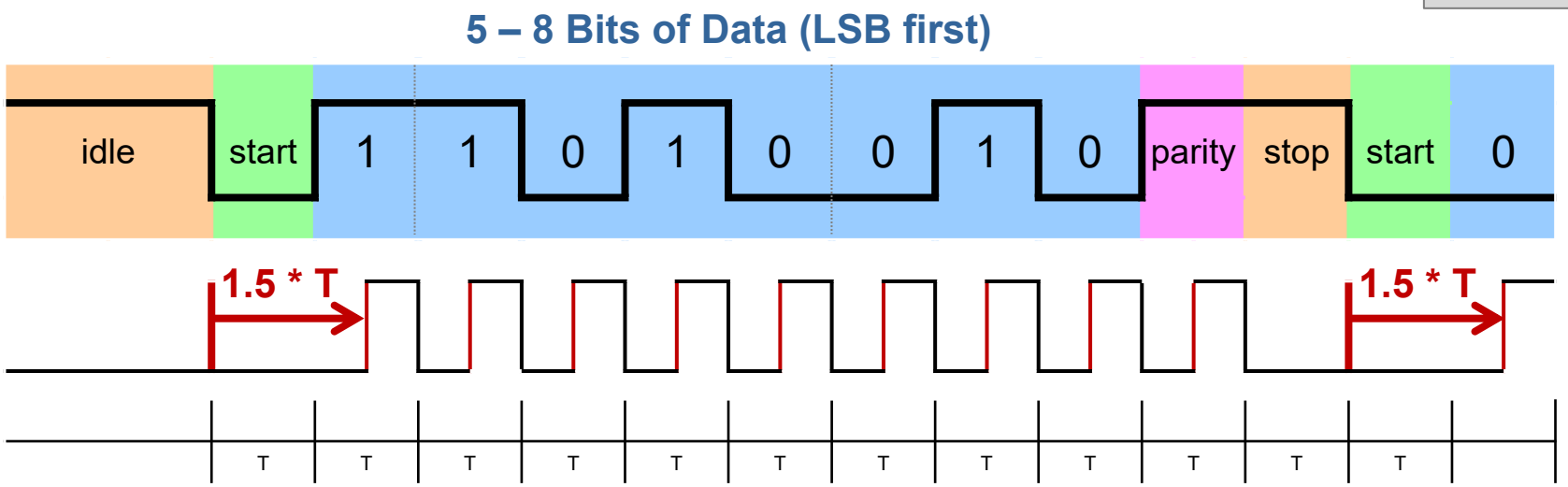
} settings allow successful transmission

Asynchronous Serial Interface

School of Engineering
InES Institute of Embedded Systems

- **UART Timing**

e.g. 9'600 Baud = symbols / s
➔ T = (1 / 9600) s = 0.104 ms



Idle → logic '1'
stop → logic '1'
start → logic '0'

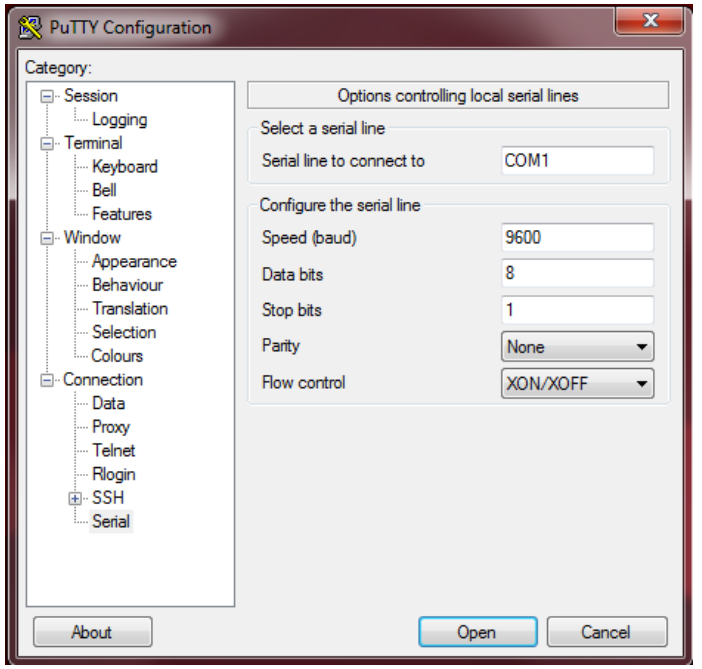
- Transition stop ('1') → start ('0')
  - Receiver detects edge at the start of each data block (5 to 8 bits)
  - Allows receiver to sample data "in middle of bits" → red edges
  - Clocks have to be accurate enough to allow sampling up to parity bit, i.e. max. +/- 0.5 bit times aggregated deviation until last bit

Asynchronous Serial Interface

School of Engineering
InES Institute of Embedded Systems

UART – TX and RX Work with Same Settings

- Transmission rate
  - 2400, 4800, 9600, 19200, 38'400, 57'600, 115'200 … bit/s
  - In case of UART: symbol rate = bit rate i.e. baud corresponds to bit/s
- Number of data bits
  - Between 5 and 8 bit
- Number of stop bits
  - 1, 1.5 or 2 bit
- Parity
  - none
  - mark (logic '1')
  - space (logic '0')
  - even
  - odd



Asynchronous Serial Interface

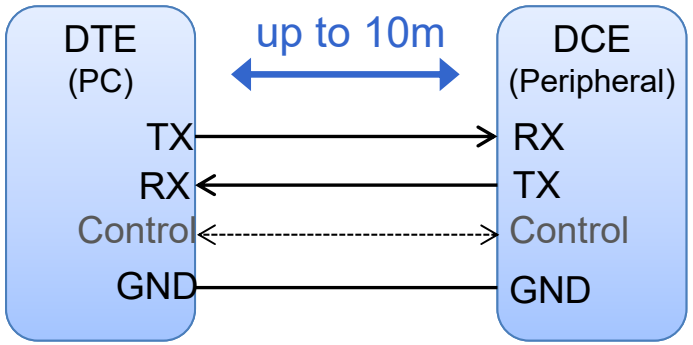
School of Engineering
InES Institute of Embedded Systems

UART Characteristics

- Asynchronous data transfer
  - Mismatch of clock frequencies in TX and RX
  - Each data item (5-8 bits) requires synchronization
  - Requires overhead for synchronization → additional bits (start/stop)
  - Requires effort for synchronization → additional hardware (searching for falling edge of start bit)
- Advantage
  - Clock does not have to be transmitted
  - Transmission delays are automatically compensated
- On-board connections
  - Signal levels are 3V or 5V with reference to ground
  - Off-board connections require stronger output drivers (circuits)

Asynchronous Serial Interface – Electrical Characteristics

RS–232[1)] – Interconnecting Equipment by Cable



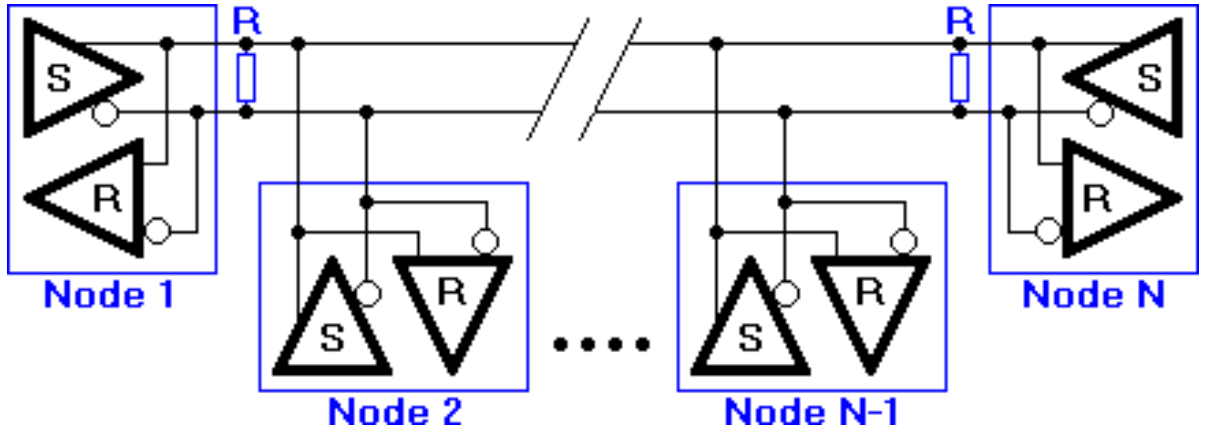
source: Olimex

- Simple, bidirectional point-to-point interface based on UART
- Optional control signals, e.g. CTS – Clear To Send
- Ground → common reference level for all signals (single ended)
- Driver circuit allows transmissions up to ~ 10 m
  - Logic '1' -3V to -15V
  - Logic '0' 3V to 15V

1) standardized by the Electronic Industries Alliance (EIA)

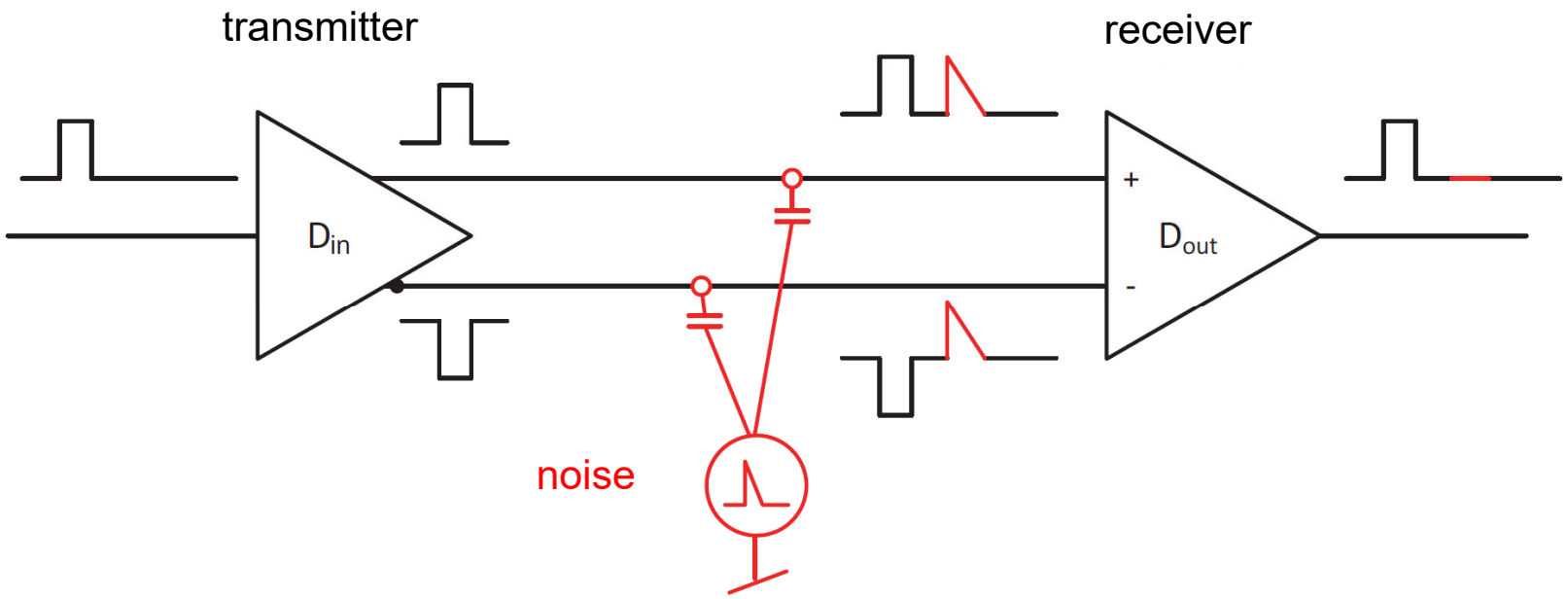
Asynchronous Serial Interface – Electrical Characteristics

- **RS–485 – Differential Transmission Based on UART**
  - Differential signals
    - Less susceptible to disturbances → longer distances, 100+ Meters
  - Transmit and receive share the same lines
    - Multi-point communication
    - Half-duplex
  - Industrial automation → E.g. lowest layer of Profibus



source: http://www.lammertbies.nl/comm/info/RS-485.html

- **RS–485 – Differential Transmission**
  - Transmitter transforms single ended signal into differential signal
  - Capacitive coupled noise affects both lines
  - Receiver forms the difference of the two signals
    - Noise on the two lines cancels itself to a large extent



U(S)ART – STM32F4xxx

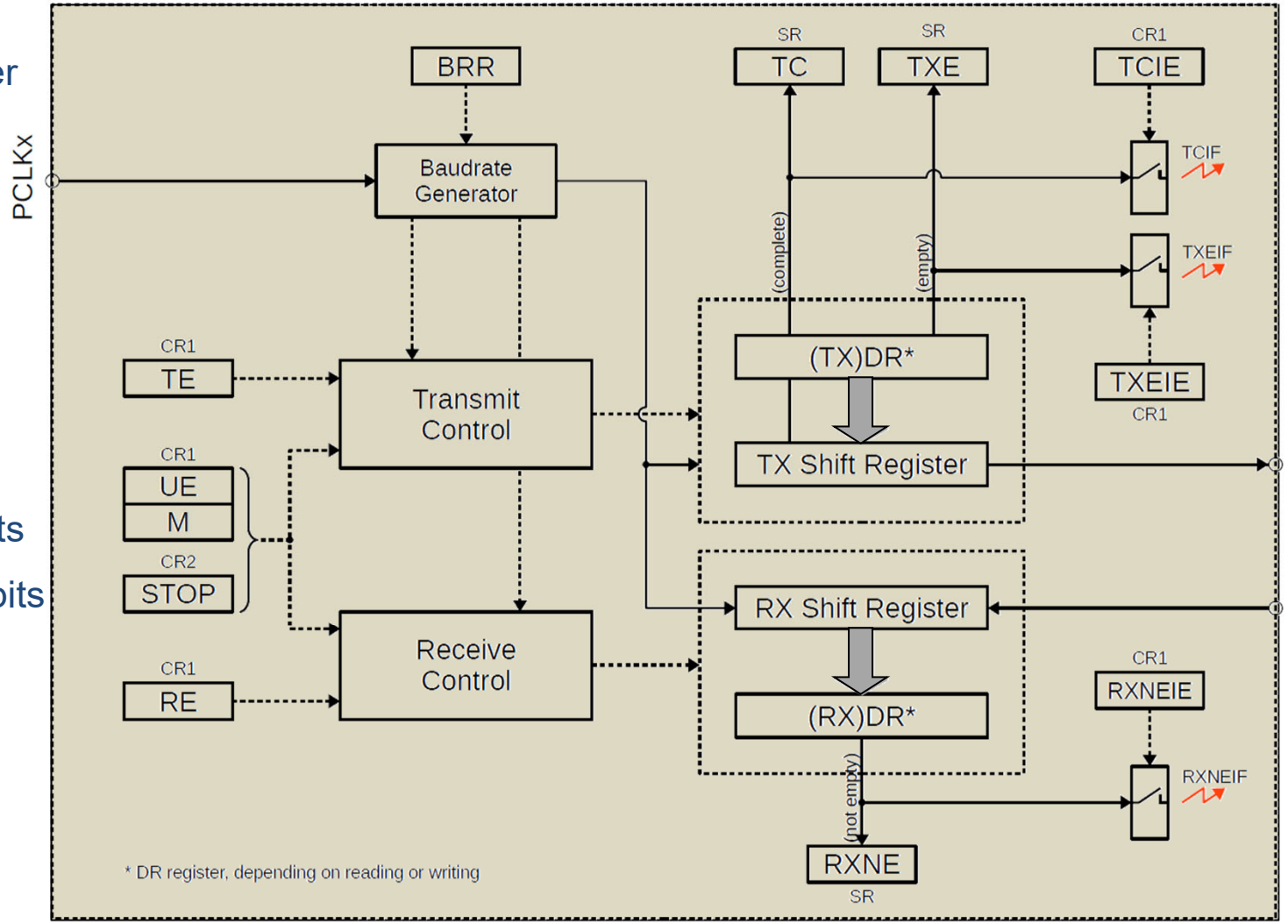
Baud rate register

TX enable

USART enable
M: 8 vs 9 data bits

Number of stop bits

RX enable



To/from GPIO pins

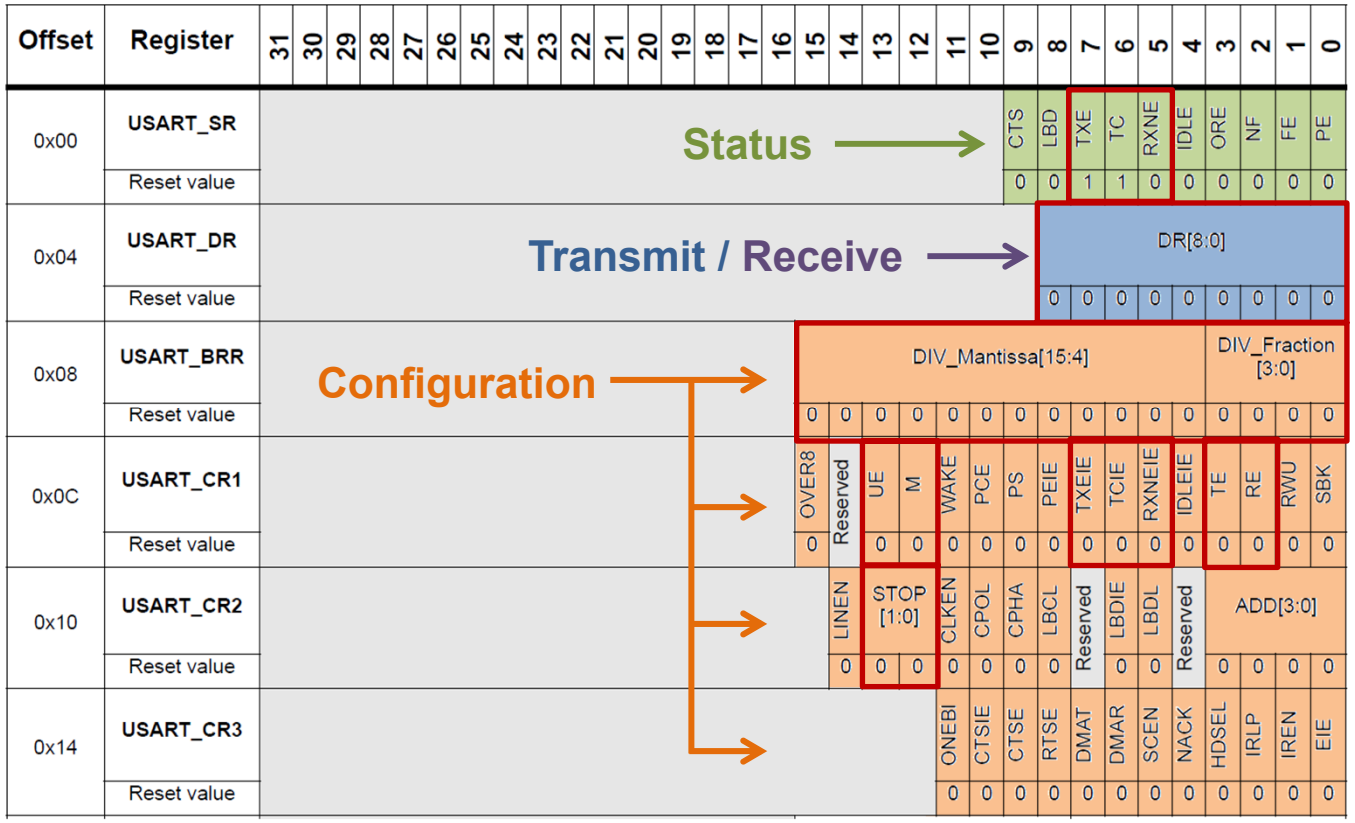
TCIF – Transmission complete interrupt (flag)

TXEIF – Transmit data register empty interrupt (flag)

RXNEIF – Received data register not empty interrupt (flag)

School of Engineering
InES Institute of Embedded Systems

USART Registers



Base addresses
of 8 U(S)ART blocks

```
#define USART1 ( (reg_usart_t *) 0x40011000 )
#define USART2 ( (reg_usart_t *) 0x40004400 )
#define USART3 ( (reg_usart_t *) 0x40004800 )
#define UART4  ( (reg_usart_t *) 0x40004c00 )
#define UART5  ( (reg_usart_t *) 0x40005000 )
#define USART6 ( (reg_usart_t *) 0x40011400 )
#define USART7 ( (reg_usart_t *) 0x40007800 )
#define USART8 ( (reg_usart_t *) 0x40007c00 )
```

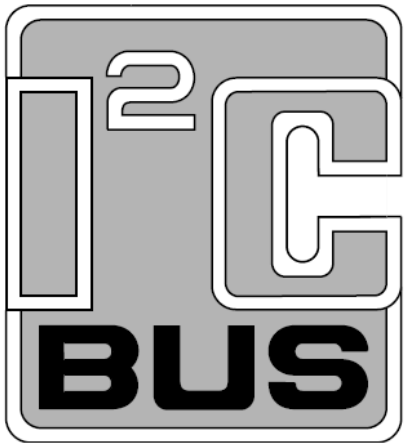
Use of DR and TXE / RXNE is analogous to SPI

I2C Bus

- **I²C – Inter-Integrated Circuit, pronounced I-squared-C**
  - Bidirectional 2-wire
  - Defined by Philips Semiconductors, now NXP
  - First release in 1982

Reference Documentation

UM10204 – I2C-bus specification and user manual
Rev. 6 — 4 April 2014

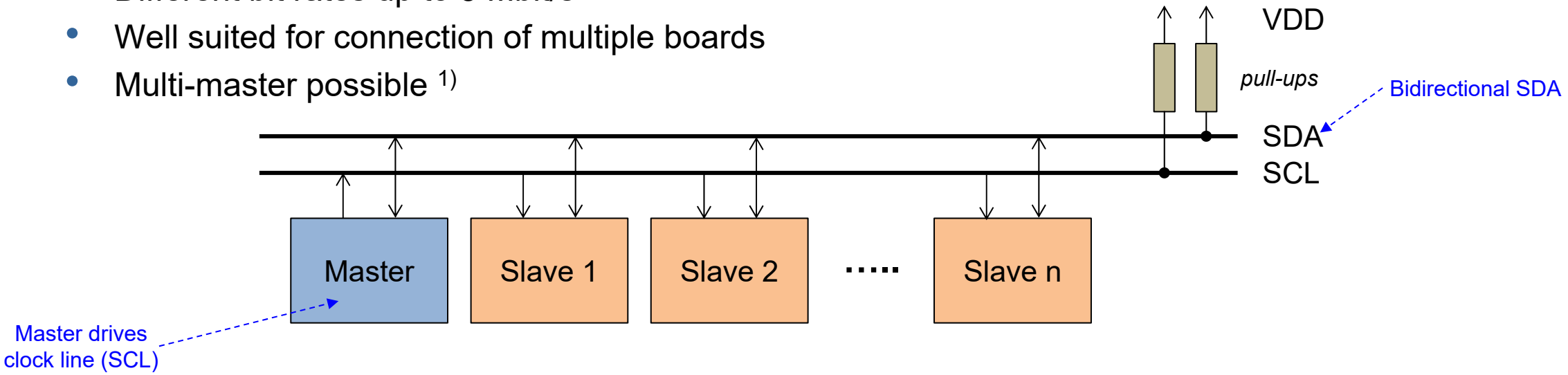


source: NXP

I2C Bus

Overview

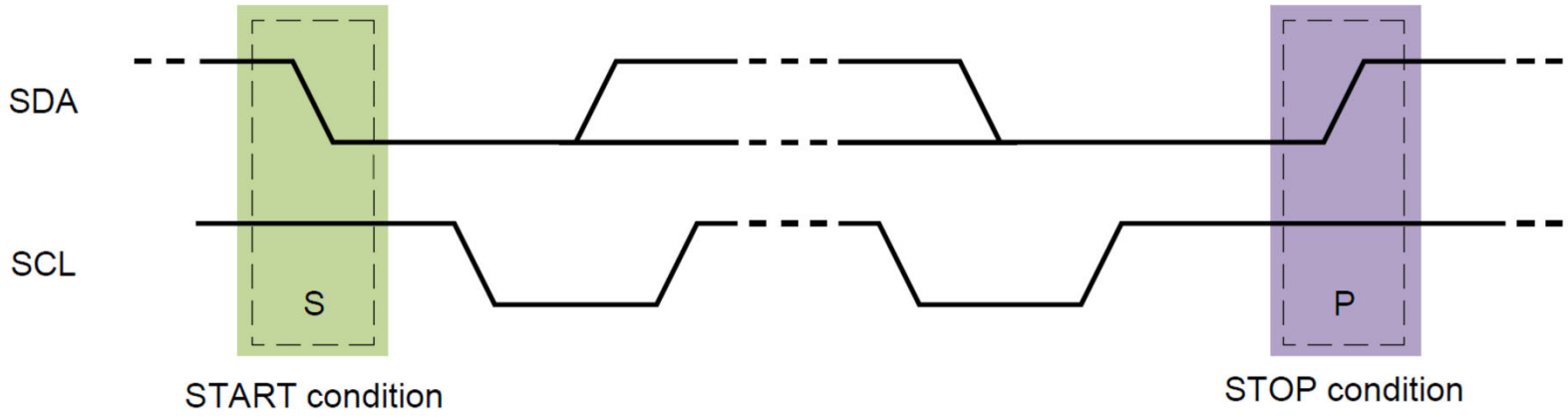
- 2-wire bus Clock → SCL Data → SDA
- Synchronous, half-duplex
- Each device on bus addressable through unique address
  - NXP assigns manufacturer IDs
- 8-bit oriented data transfers
- Different bit rates up to 5 Mbit/s
- Well suited for connection of multiple boards
- Multi-master possible [1)]



1) multi-master not covered in this course

I2C Bus

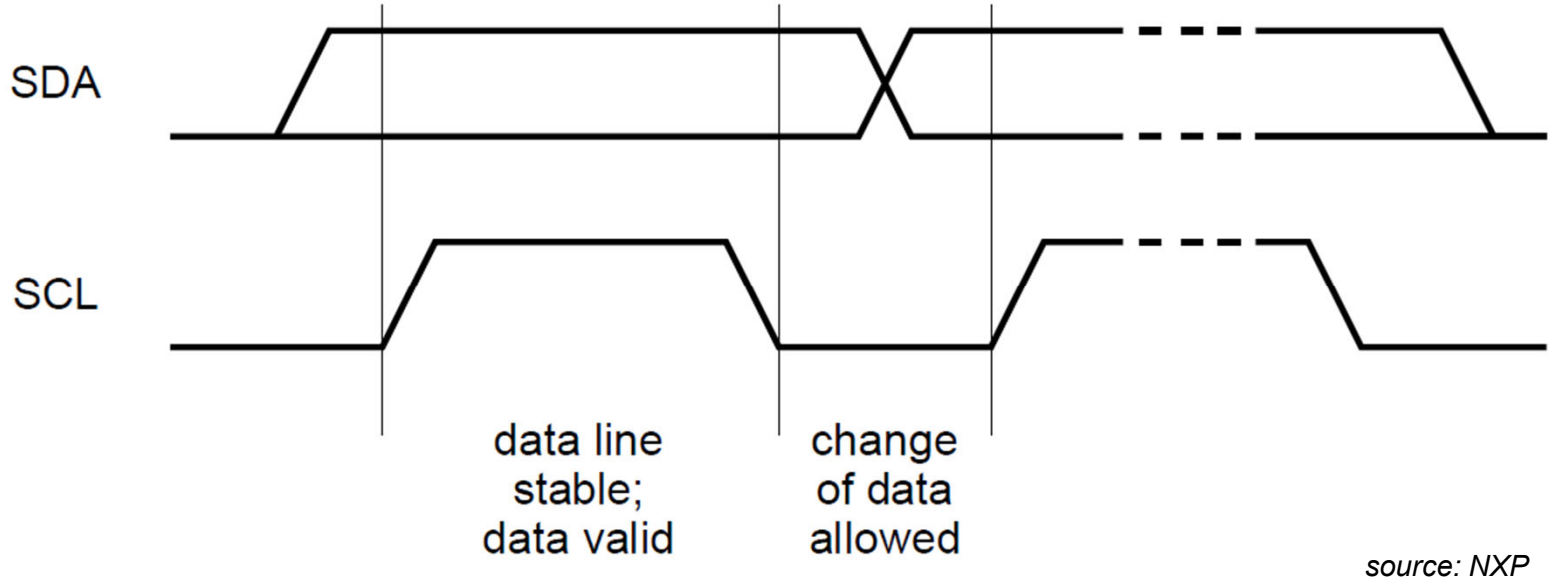
- **Operation**
  - Master drives clock line (SCL)
  - Master initiates transaction through START condition
    - Falling edge on SDA when SCL high
  - Master terminates transaction through STOP condition
    - Rising edge on SDA when SCL high



source: NXP

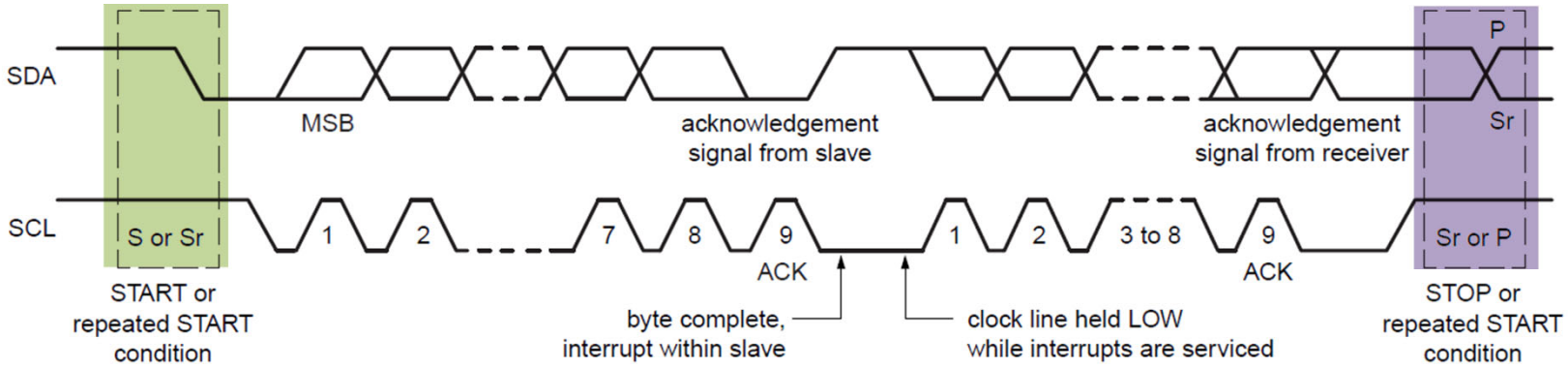
Driving Data on SDA

- Data driven onto SDA by master or by addressed slave
  - Depending on transaction (read/write) and point in time
  - Change of data only allowed when SCL is low
  - Allows detection of START and STOP condition



source: NXP

- **Data Transfer on I2C**
  - 8-bit oriented transfers
  - Bit 9: Receiver acknowledges by driving SDA low
  - Master defines number of 8-bit transfers (STOP)



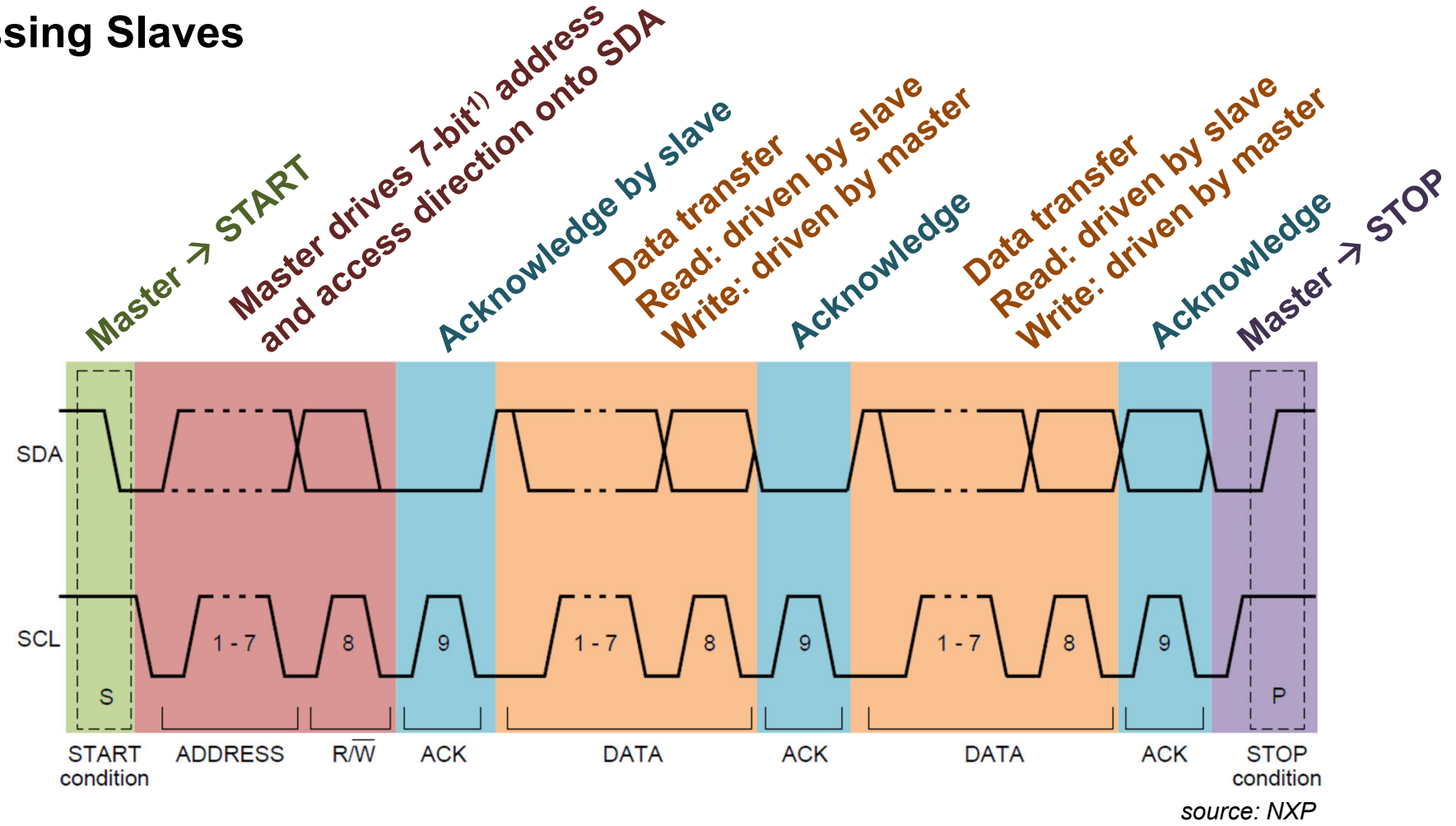
source: NXP

Bit numbering starts with 1; not with 0

'ACK' is active low

School of Engineering
InES Institute of Embedded Systems

- **Addressing Slaves**

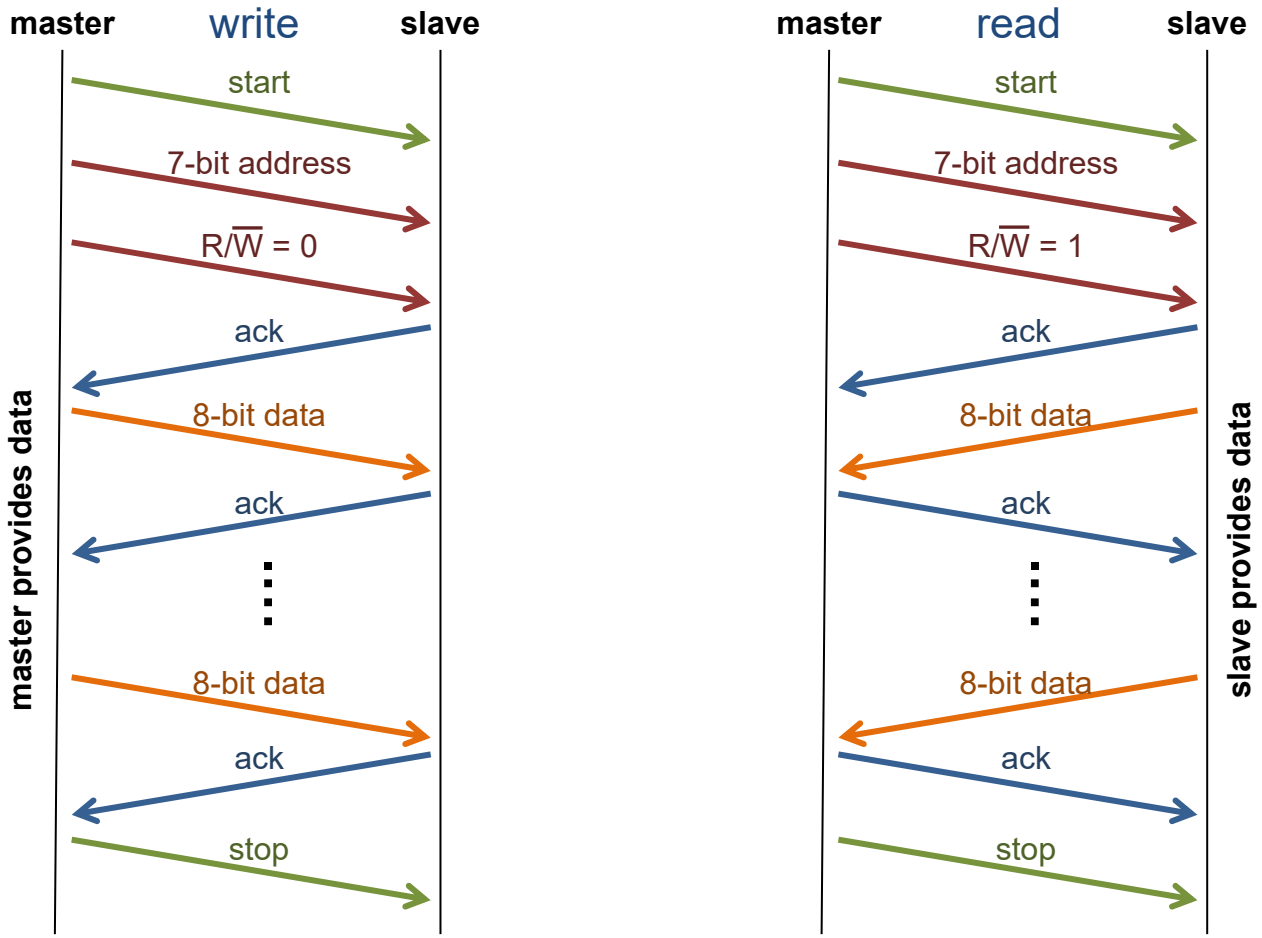


source: NXP

1) 10-bit address scheme available as option

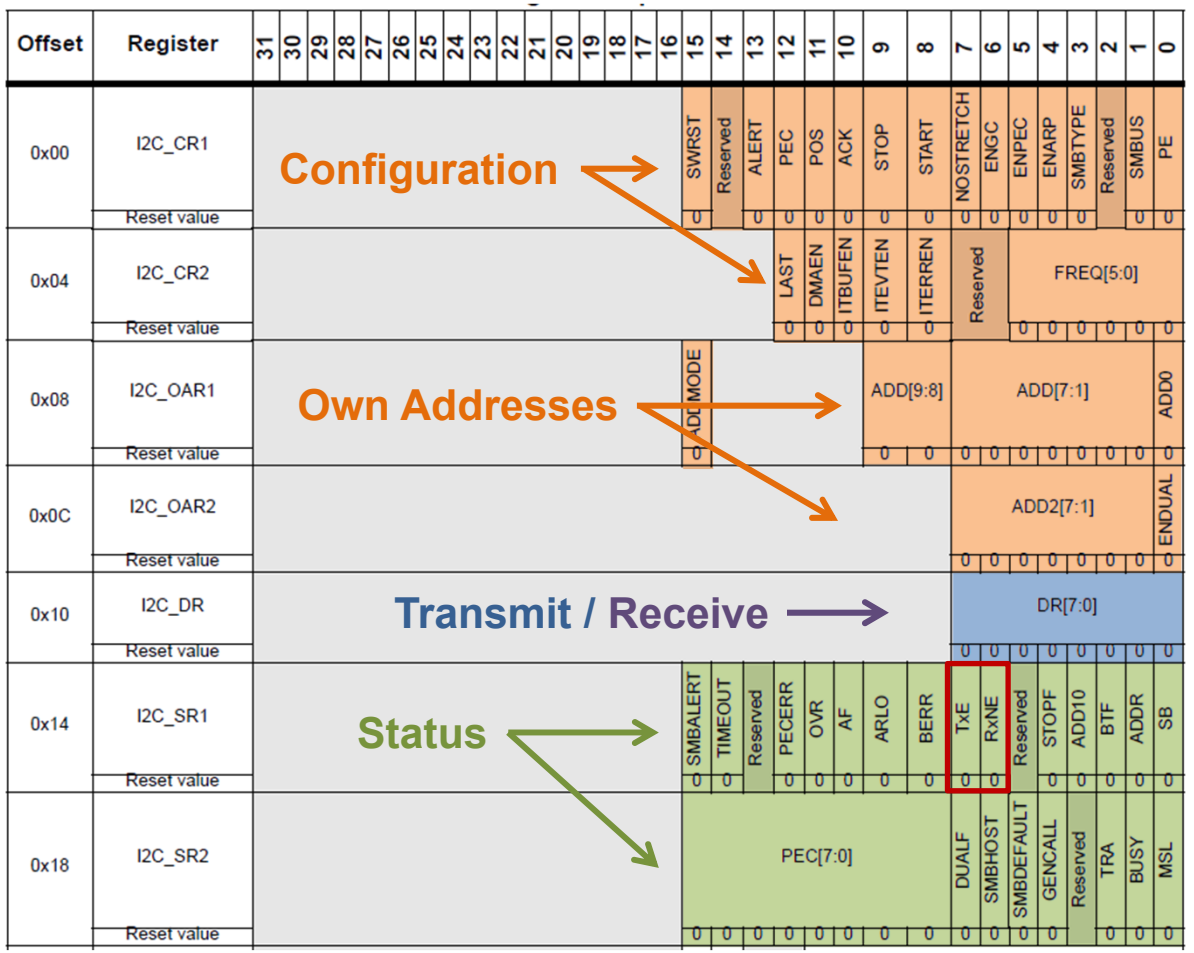
School of Engineering
InES Institute of Embedded Systems

- **Accesses**



School of Engineering
InES Institute of Embedded Systems

I²C Registers



Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	I2C_CR1																	SWRST	Reserved	ALERT	PEC	POS	ACK	STOP	START	NOSTRETCH	ENGC	ENPEC	ENARP	SMBTYPE	Reserved	SMBUS	PE
	Reset value																	0		0	0	0	0	0	0	0	0	0	0	0		0	0
0x04	I2C_CR2																				LAST	DMAEN	ITBUFEN	ITEVTEN	ITERREN	Reserved	FREQ[5:0]						
	Reset value																				0	0	0	0	0		0	0	0	0	0	0	0
0x08	I2C_OAR1																	ADDMODE						ADD[9:8]		ADD[7:1]							ADD0
	Reset value																	0						0	0	0	0	0	0	0	0	0	0
0x0C	I2C_OAR2																									ADD2[7:1]							ENDUAL
	Reset value																									0	0	0	0	0	0	0	0
0x10	I2C_DR																									DR[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x14	I2C_SR1																	SMBALERT	TIMEOUT	Reserved	PECERR	OVR	AF	ARLO	BERR	TxE	RxNE	Reserved	STOPF	ADD10	BTF	ADDR	SB
	Reset value																	0	0		0	0	0	0	0	0	0		0	0	0	0	0
0x18	I2C_SR2																	PEC[7:0]								DUALF	SMBHOST	SMBDEFAULT	GENCALL	Reserved	TRA	BUSY	MSL
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0

Base addresses of 3 I2C blocks

```
#define I2C1 ((reg_i2c_t *) 0x40005400)
#define I2C2 ((reg_i2c_t *) 0x40005800)
#define I2C3 ((reg_i2c_t *) 0x40005c00)
```

School of Engineering
InES Institute of Embedded Systems

Register Bits

I2C_CR1 I²C control register 1

SWRST	Software reset
ALERT	SMBus alert
PEC	Packet error checking
POS	Acknowledge / PEC Position
ACK	Acknowledge enable
STOP	Stop generation
START	Start generation
NOSTRETCH	Clock stretching disable (slave)
ENGC	General call enable
ENPEC	PEC enable
ENARP	ARP enable
SMBTYPE	SMBus type (device vs. host)
SMBUS	SMBus mode (I2C vs. SMBus)
PE	Peripheral enable

I2C_CR2 I²C control register 2

LAST	DMA last transfer
DMAEN	DMA requests enable
ITBUFEN	Buffer interrupt enable
ITEVEN	Event interrupt enable
ITERREN	Error interrupt enable
FREQ[5:0]	Peripheral clock frequency

I2C_DR I²C data register 2

DR[7:0]	8-bit data register

I2C_OAR1 I²C own address register 1

ADDMODE	Addressing mode (7-bit vs. 10-bit)
ADD[9:8]	Interface address
ADD[7:1]	Interface address
ADD0	Interface address

I2C_OAR2 I²C own address register 2

ADD2[7:1]	Interface address in dual adr. mode
ENDUAL	Dual addressing mode enable

I2C_SR1 I²C status register 1

SMBALERT	SMBus alert
TIMEOUT	Timout or Tlow error
PECERR	PEC error in reception
OVR	Overrun/Underrun
AF	Acknowledge failure
ARL0	Arbitration lost (master)
BERR	Bus error
TxE	Data register empty
RxNE	Data register not empty
STOPF	Stop detected (slave)
ADD10	10-bit header sent (master)
BTF	Byte transfer finished
ADDR	ADDR sent (master) / matched (slave)
SB	Start bit (master)

I2C_SR2 I²C status register 2

PEC[7:0]	Packet error checking register
DUALF	Dual flag (slave)
SMBHOST	SMBus host header (slave)
SMBDEFAULT	SMBus device default address (slave)
GENCALL	General call address (slave)
TRA	Transmitter/receiver (R/W bit)
BUSY	Bus busy (communication ongoing on bus)
MSL	Master/slave

Comparison

School of Engineering
InES Institute of Embedded Systems

UART	SPI	I2C
serial ports (RS-232)	4-wire bus	2-wire bus
TX, RX opt. control signals	*MOSI, MISO, SCLK, SS*	*SCL, SDA*
point-to-point	point-to-multipoint	(multi-) point-to-multi-point
full-duplex	full-duplex	half-duplex
asynchronous	synchronous	synchronous
only higher layer addressing	slave selection through $\overline{SS}$ signal	7/10-bit slave address
parity bit possible	no error detection	no error detection
chip-to-chip, PC terminal program	chip-to-chip, on-board connections	chip-to-chip, board-to-board connections
The three interfaces provide the lowest layer of communication and require higher level protocols to provide and interpret the transferred data.		

Conclusions

- **Asynchronous Serial Interface**
  - Transmitter and receiver use diverging clocks
  - Synchronization using start/stop bits → overhead
  - Longer connections require line drivers → RS-232/RS-485
- **SPI**
  - Master/Slave
  - Synchronous full-duplex transmission (MOSI, MISO)
  - Selection of device through Slave Select ($\overline{SS}$)
  - No acknowledge, no error detection
  - Four modes → clock polarity and clock phase
- **I2C**
  - Synchronous half-duplex transmission (SCL, SDA)
  - 7-bit slave addresses