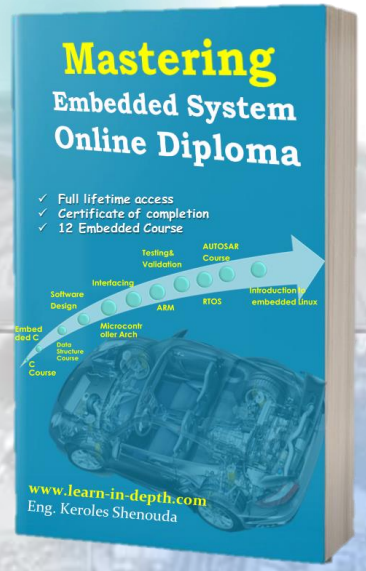
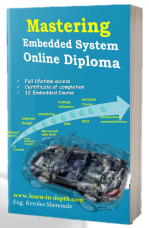
Mastering Embedded System

Online Diploma

- ✓ Full lifetime access
- ✓ Access on Android mobile and PC (Windows)
- ✓ Certificate of completion
- ✓ 12 Embedded Course





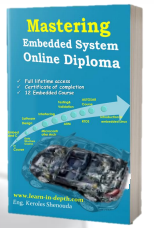
#LEARN_IN DEPTH

#Be_professional_in_embedded_system

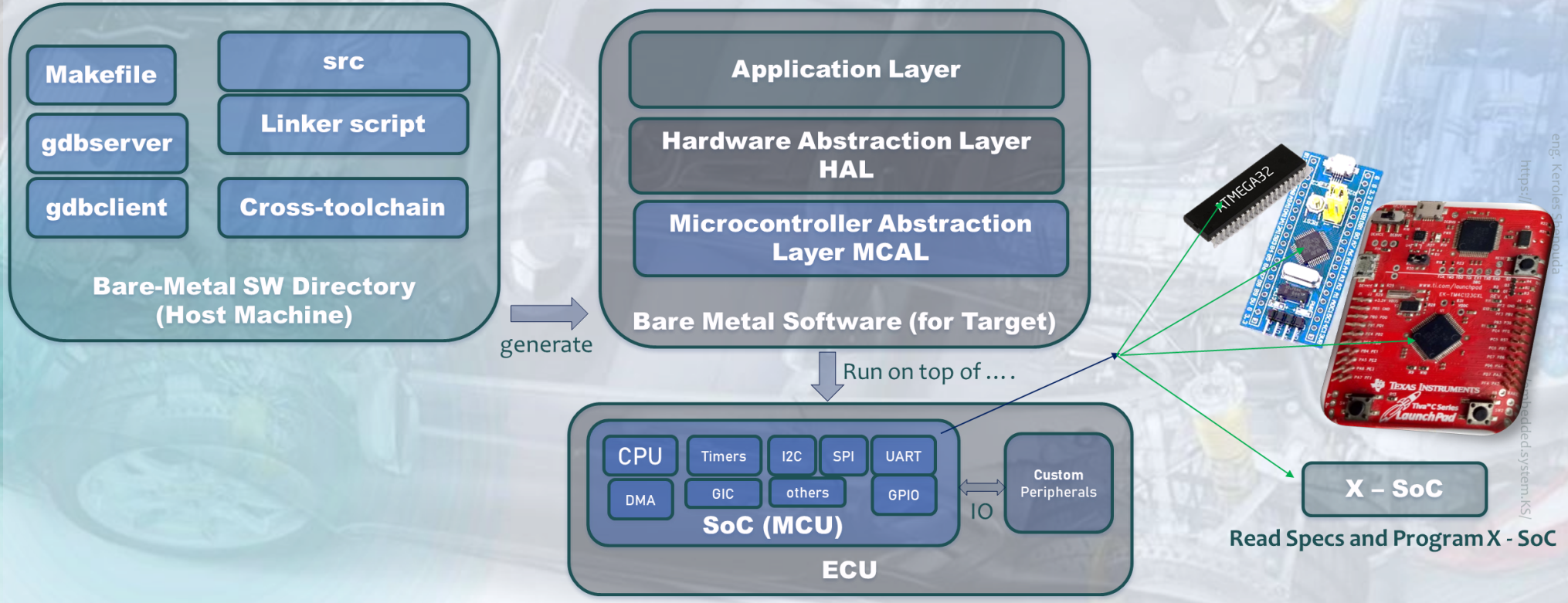
Unit 8 (MCU Interfacing) . lesson 2 UART

- Interfacing Main Concepts
  - Embedded communication
  - Serial Vs Parallel
  - Serial Synchronization
  - TX/RX relations
  - Bit Rate vs. Baud Rate & Data Throughput
  - Single ended vs differential Wire
- Communication Specs Big Picture
- UART Protocol
  "Universal asynchronous receiver-transmitter"
- Wire standard Conjugation with USART protocol
- USART Controller General Circuit
- UART General Configurations
- USART Block Diagram on Atmega32
- USART Block Diagram on STM32F103X
- USART Block Diagram on TM4C123

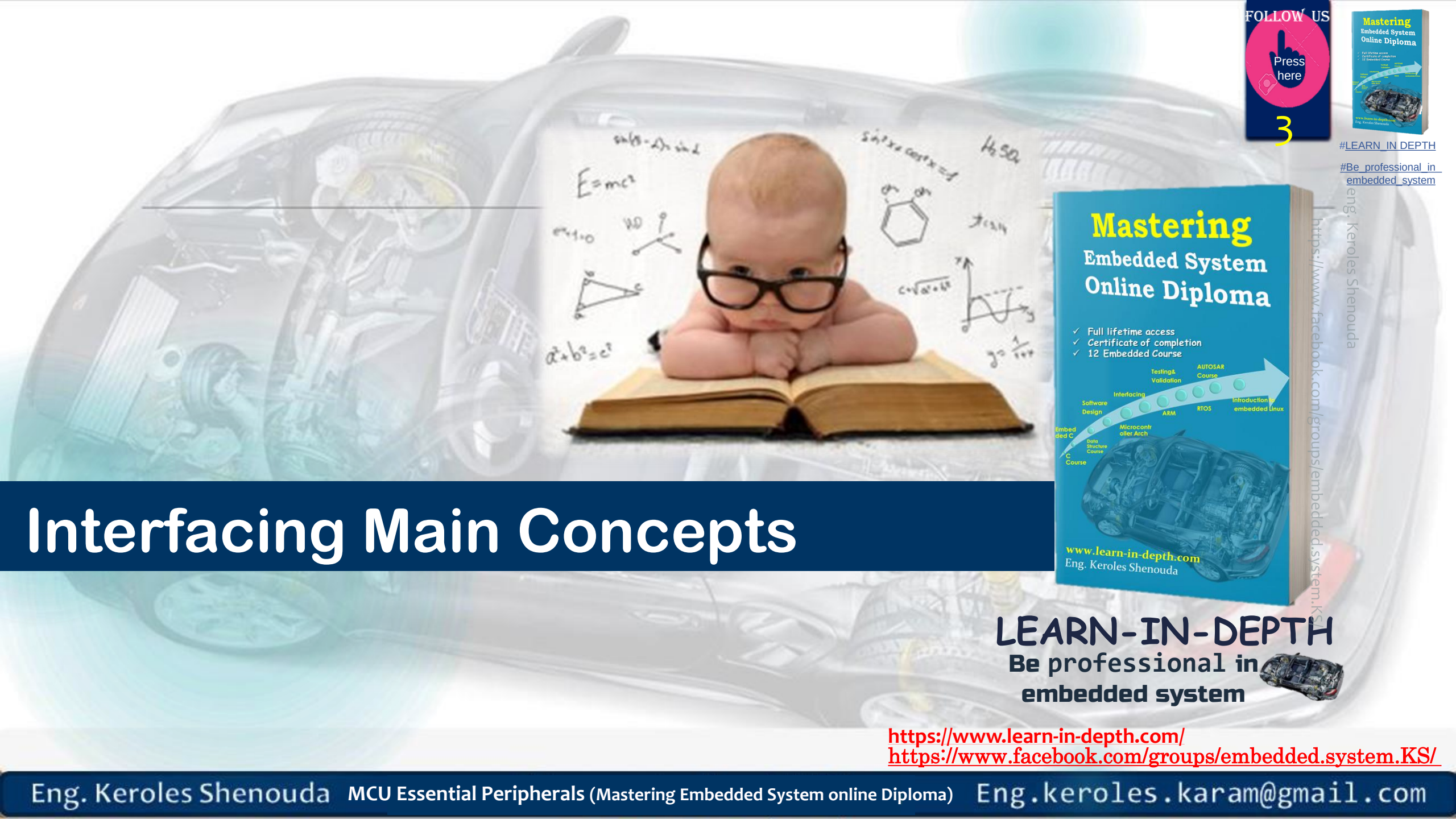
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

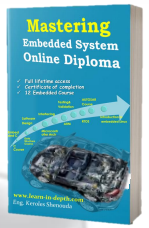


Big Picture



https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

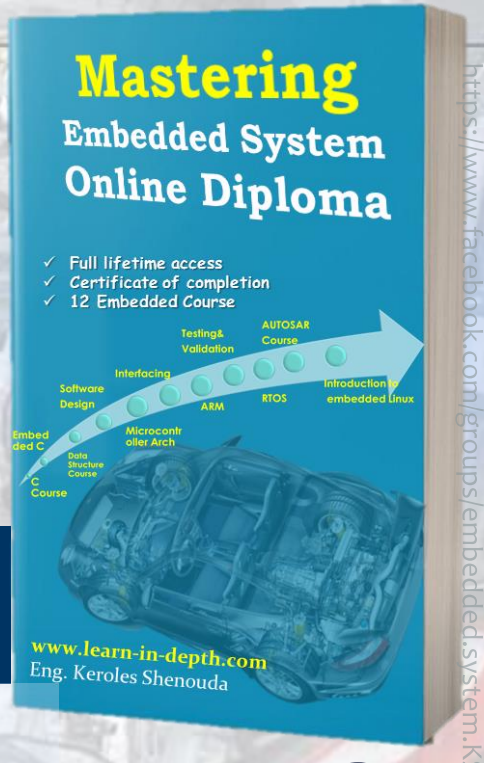

FOLLOW US
Press here
3
Mastering
Embedded System
Online Diploma
#LEARN_IN DEPTH
#Be_professional_in_embedded_system
Mastering
Embedded System
Online Diploma
✓ Full lifetime access
✓ Certificate of completion
✓ 12 Embedded Course
Embed ded C
C Course
Data Structure Course
Software Design
Microcontr oller Arch
Interfacing
ARM
Testing& Validation
RTOS
AUTOSAR Course
Introduction to embedded Linux
www.learn-in-depth.com
Eng. Keroles Shenouda
https://www.facebook.com/groups/embedded.system.KS/
Eng. Keroles Shenouda
Interfacing Main Concepts
LEARN-IN-DEPTH
Be professional in embedded system
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/
Eng. Keroles Shenouda MCU Essential Peripherals (Mastering Embedded System online Diploma) Eng.keroles.karam@gmail.com



#LEARN_IN DEPTH

#Be_professional_in_embedded_system

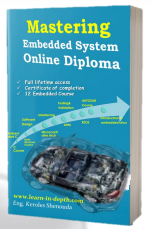
Embedded communication



LEARN-IN-DEPTH
Be professional in embedded system

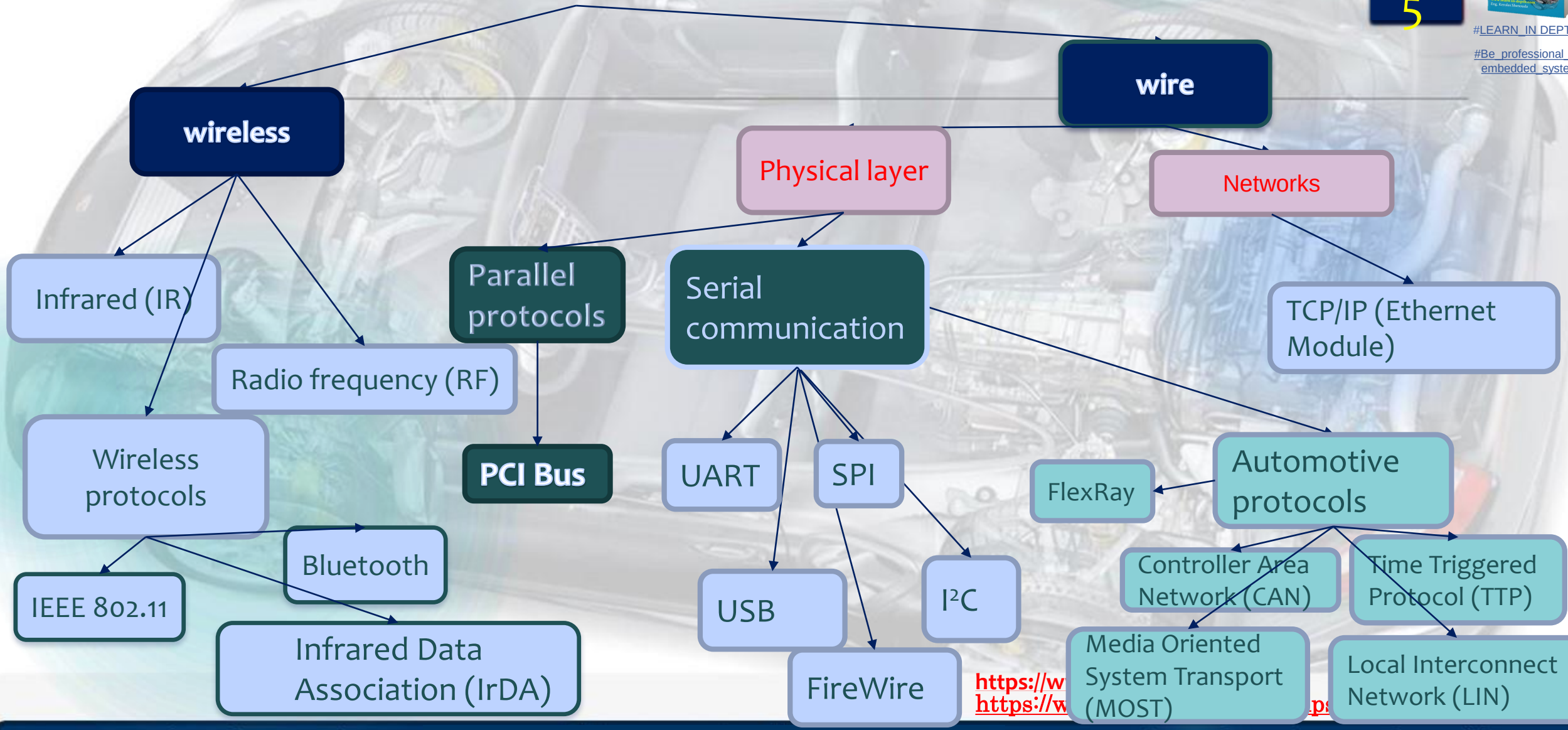
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

Embedded communication

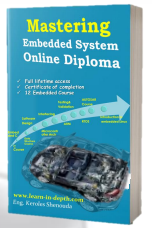


#LEARN_IN DEPTH

#Be_professional_in embedded_system



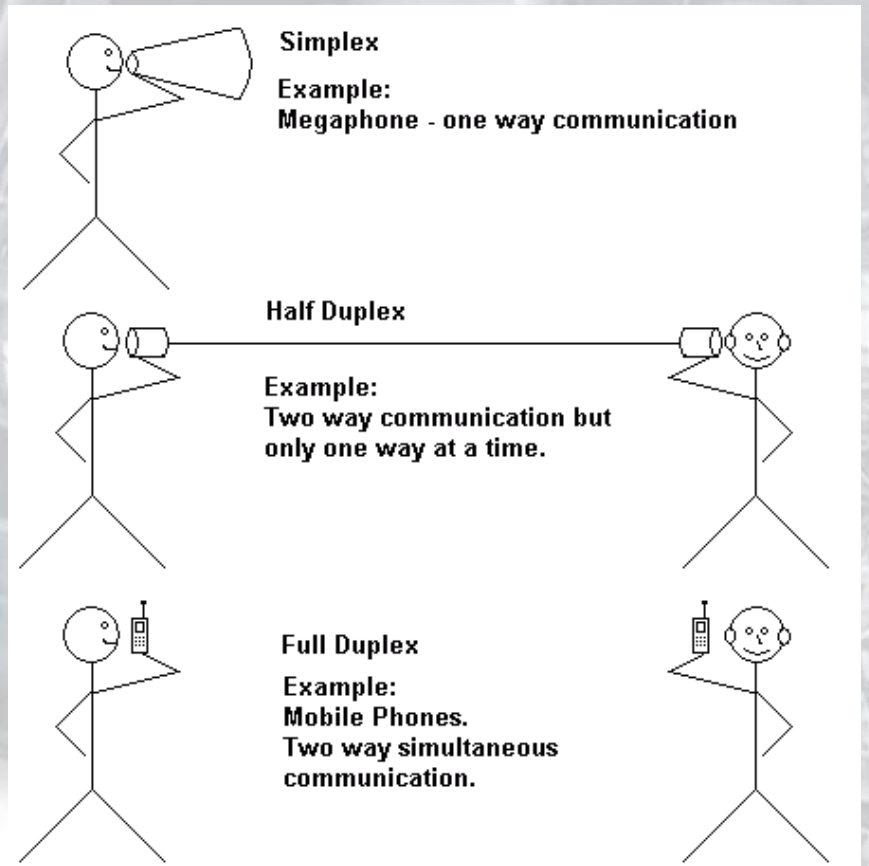
https://w
https://w



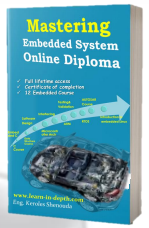
#LEARN_IN DEPTH

#Be_professional_in_embedded_system

Simplex, Half-duplex, Full-duplex



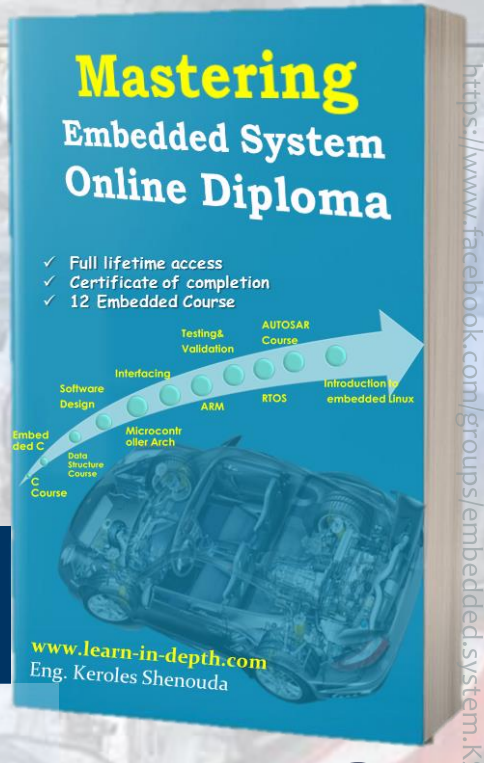
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



#LEARN_IN DEPTH

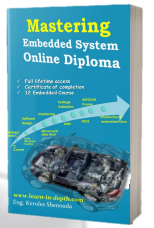
#Be_professional_in_embedded_system

Serial Vs Parallel

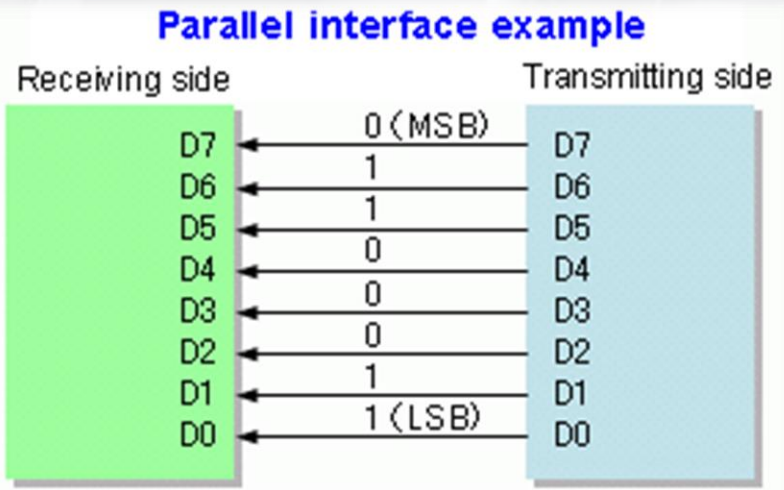


LEARN-IN-DEPTH
Be professional in embedded system

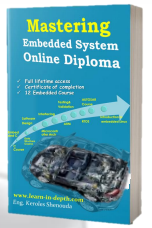
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



Serial Vs Parallel

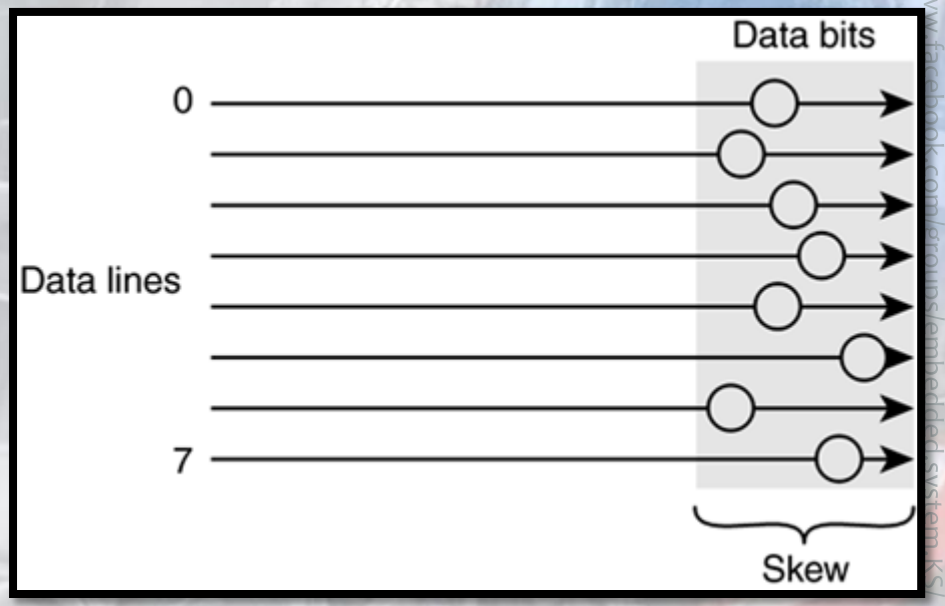


https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

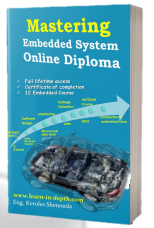


Parallel Interfacing (Data Skew)

- One of the difficulties presented by parallel bus architecture is a phenomenon known as skew.
- If 8 or 16 bits of data are sent simultaneously in parallel, small differences in propagation delay along each data line may occur, and not all bits may arrive at the destination at precisely the same moment.
- Skew refers to the difference in arrival time for each bit comprising a data word. The skew window is the time difference within which all bits must arrive.



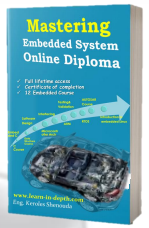
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



Serial Vs Parallel

	Parallel Communication	Serial Communication
Cable	Use large number of wires	Use less number of wires
Cable length	Can't use lengthy cables. EMI limits data rate	Use long shield cables, protected from EMI
Communication modes	Only single shared path is available. Hence, can be only half-duplex	Can have separate paths for transmission and reception. Hence, can be full-duplex
Communication error	Bits get corrupted due to capacitance effects between cable wires	Only one bit is communicated at a time
Data rate	It is faster.	Latest techniques offer faster / comparable rates. E.g.: PCI-Ex, SATA

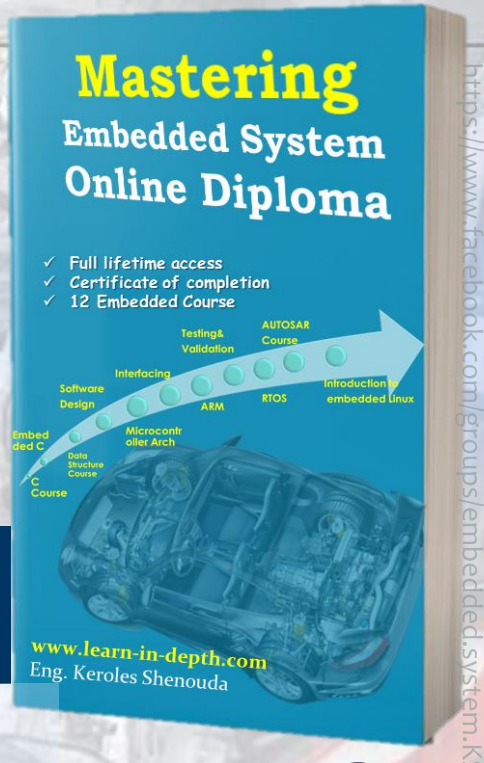
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



#LEARN_IN DEPTH

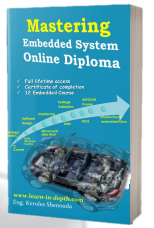
#Be_professional_in_embedded_system

Serial Synchronization



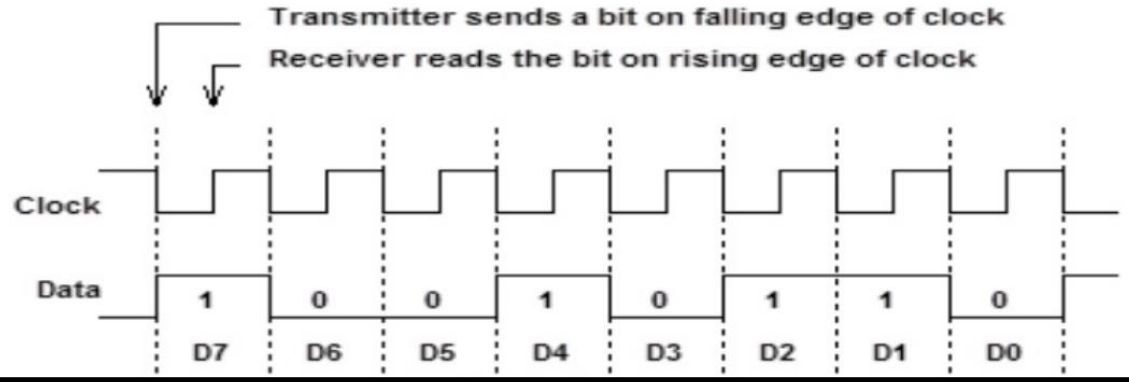
LEARN-IN-DEPTH
Be professional in embedded system

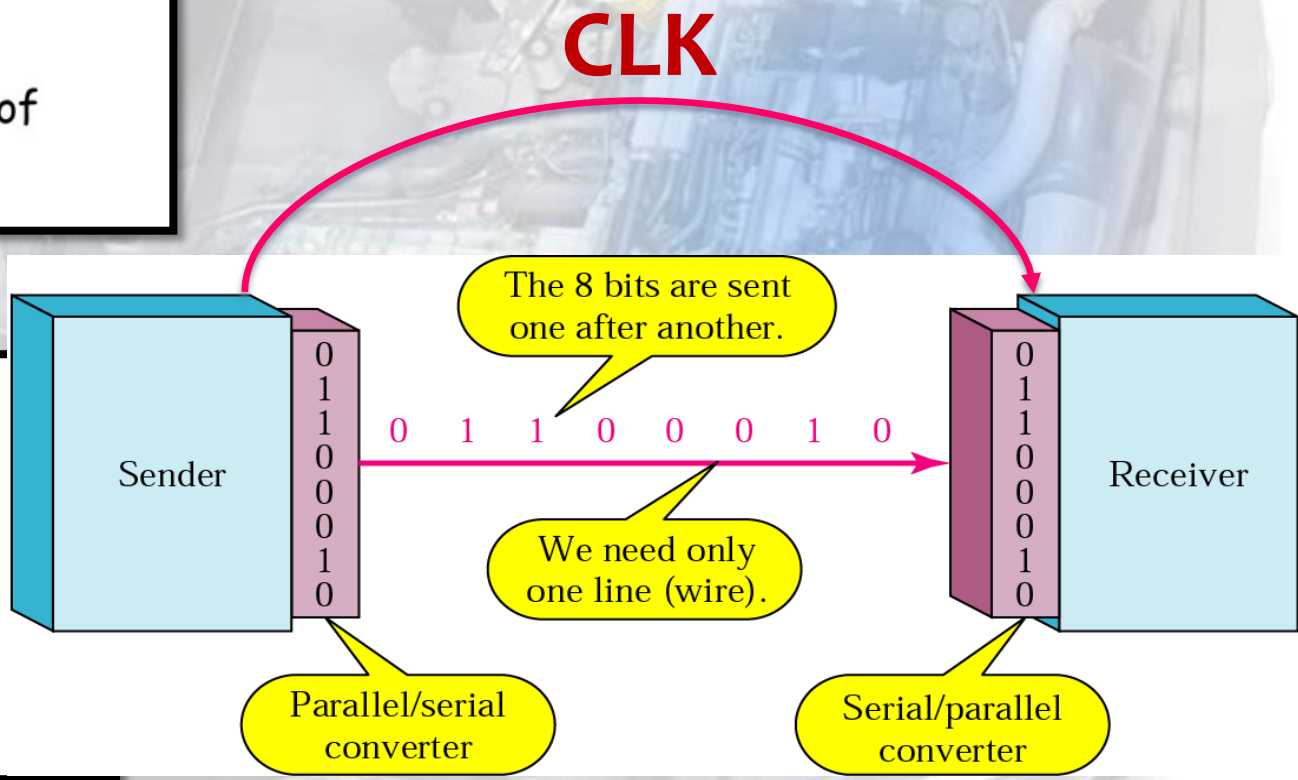
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



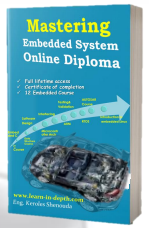
Synchronous Communication

- Transmitter & receiver use common clock
- Used for short distance, high volume, in blocks (instead of individual characters) and fast transfer





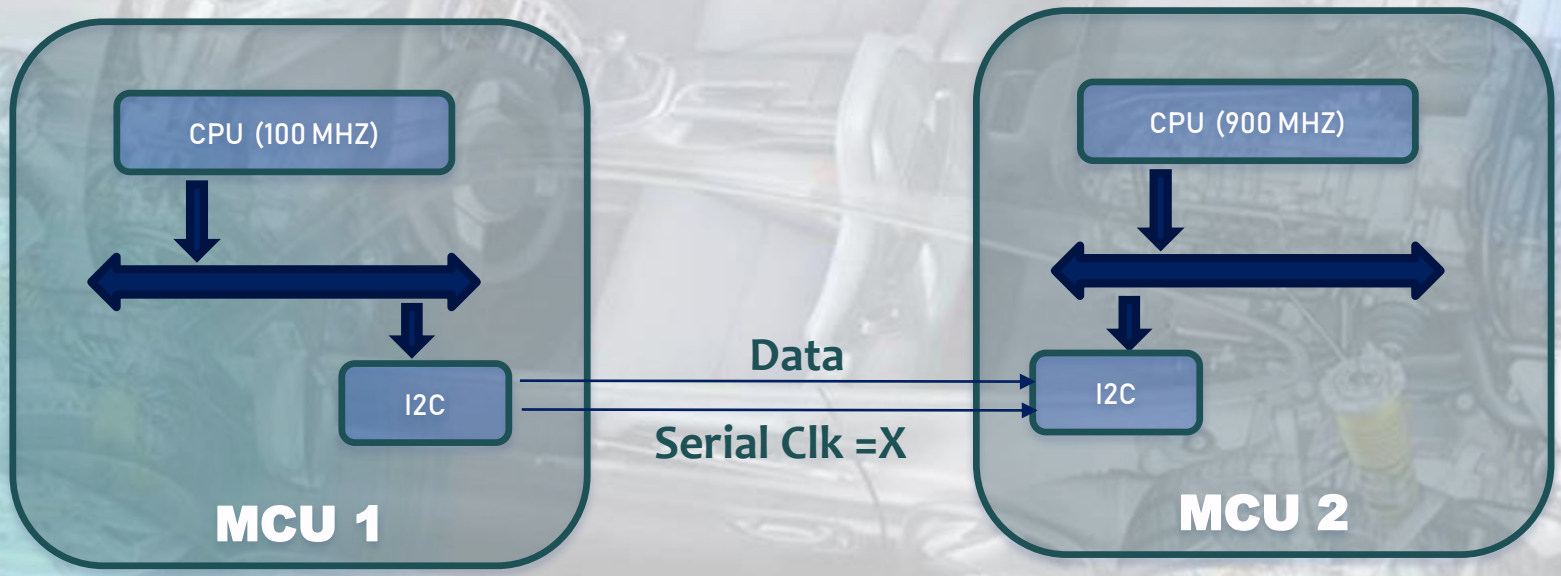
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



#LEARN_IN DEPTH

#Be_professional_in_embedded_system

Take Care, Serial Clock is different than CPU Clock

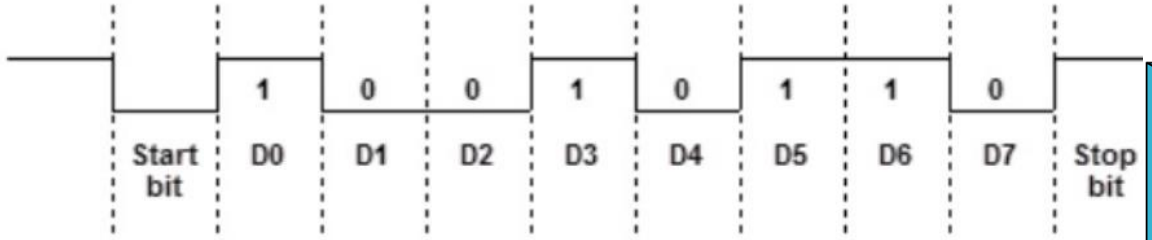


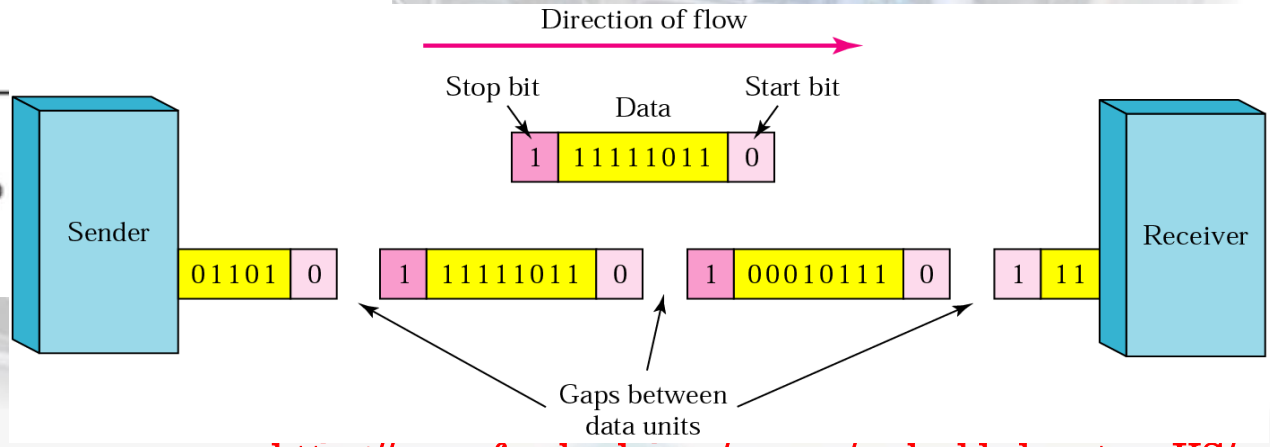
Serial Clock is Shared clock

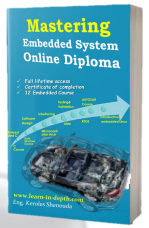
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

Asynchronous Communication

- Framing
  - Start bit, data bits, parity bit, stop bits - Popular format 8N1
- Character oriented
- Standard communication speeds:
  - 110, 150, 300, 600, 1200, 2400, 4800, 9600, ..., 115.2 kbaud

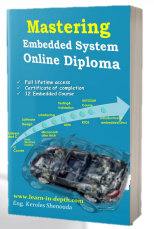




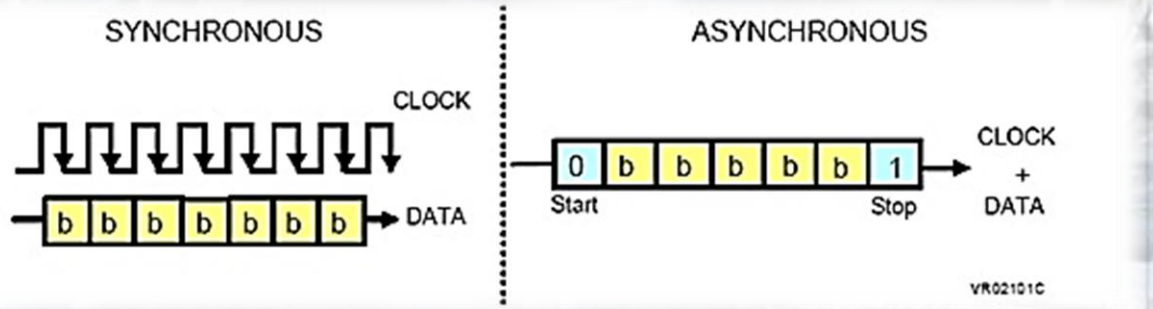


Asynchronous

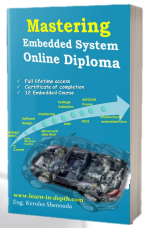
Asynchronous = No Clock



Synchronous VS Asynchronous



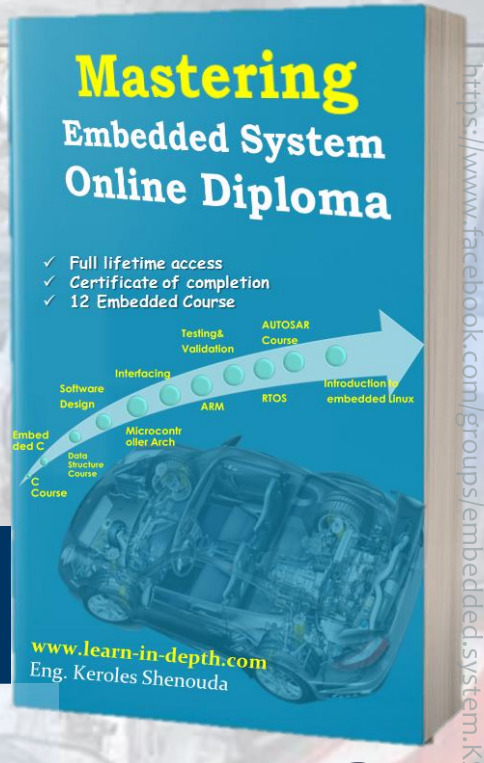
Synchronous	Asynchronous
• Fast transmission • Needs a common clock signal, or some way of sharing it • May have to wait briefly until data can be sent	• Slower transmission, due to the extra bits and the gaps • Cheap and easy to implement = no clock sharing • Can transmit when ready



#LEARN_IN DEPTH

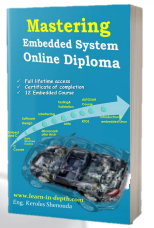
#Be_professional_in_embedded_system

TX/RX relations



LEARN-IN-DEPTH
Be professional in embedded system

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



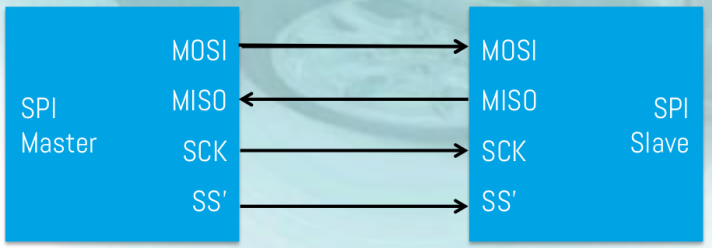
#LEARN_IN DEPTH

#Be_professional_in_embedded_system

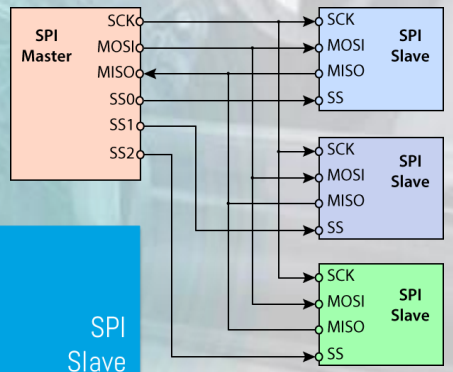
TX/RX relation

Master/Slave

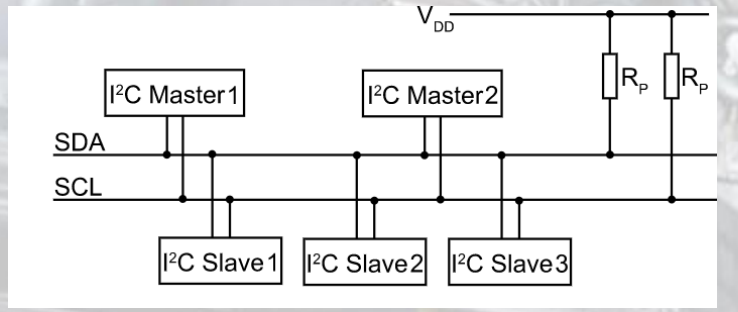
Single Master Single Slave



Single Master Multi Slave

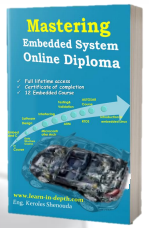


Multi Master Multi Slave



Peer to peer

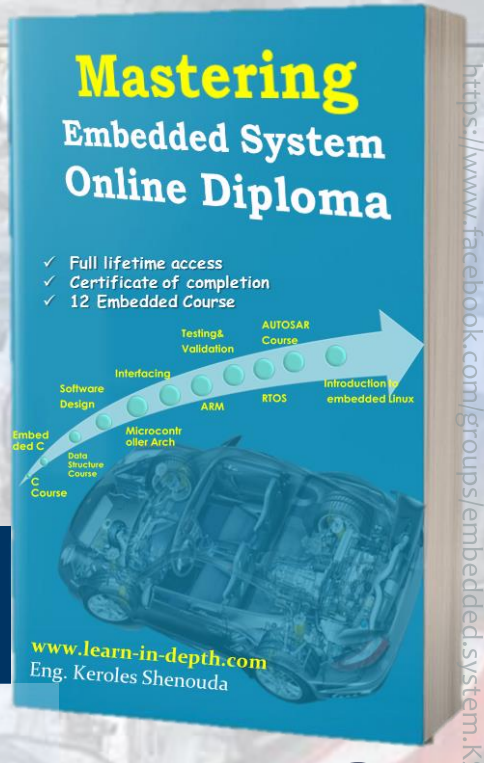
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



#LEARN_IN DEPTH

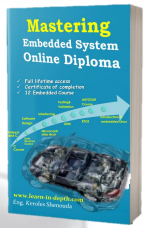
#Be_professional_in_embedded_system

Bit Rate vs. Baud Rate & Data Throughput



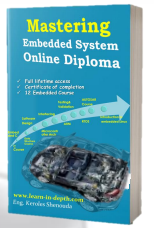
LEARN-IN-DEPTH
Be professional in embedded system

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



Bit Rate vs. Baud Rate

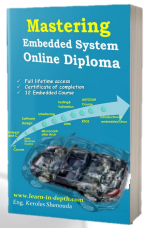
- Bit Rate: how many ***data bits*** are transmitted per second?
- Baud Rate: how many ***symbols*** are transmitted per second?
- These may be different
  - Extra symbols (channel changes) may be inserted for framing, error detection, acknowledgment, etc. These ***reduce*** the bit rate
  - A single symbol might encode more than one bit.
    - This ***increases*** the bit rate.



Throughput

$$\text{Throughput (bps)} = \frac{\text{Amount of Effective Data}}{\text{Data Size transmitted}} * 100\%$$

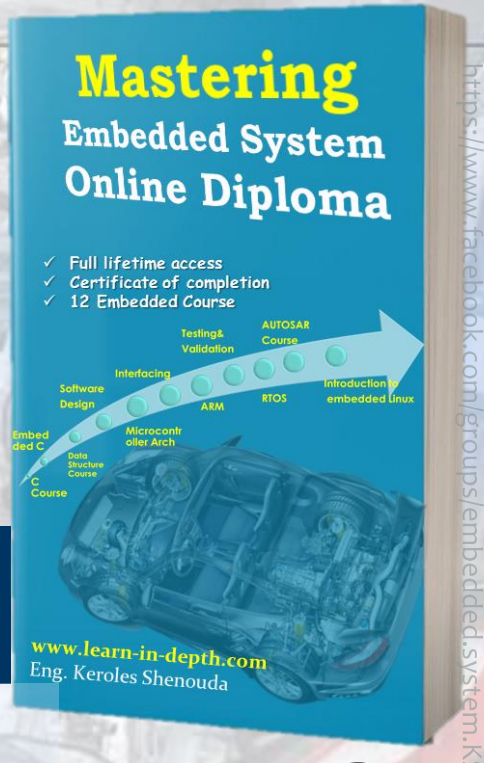
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



#LEARN_IN DEPTH

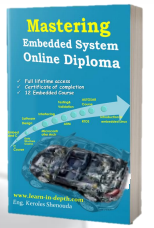
#Be_professional_in_embedded_system

Single ended vs differential Wire



LEARN-IN-DEPTH
Be professional in embedded system

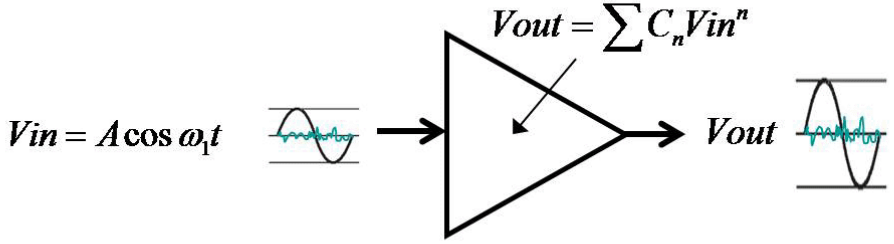
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



Single ended vs differential

SINGLE-ENDED

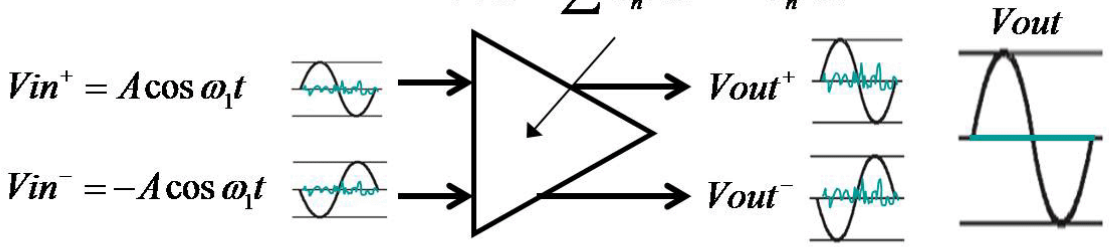
$$Vout = \sum C_n Vin^n$$



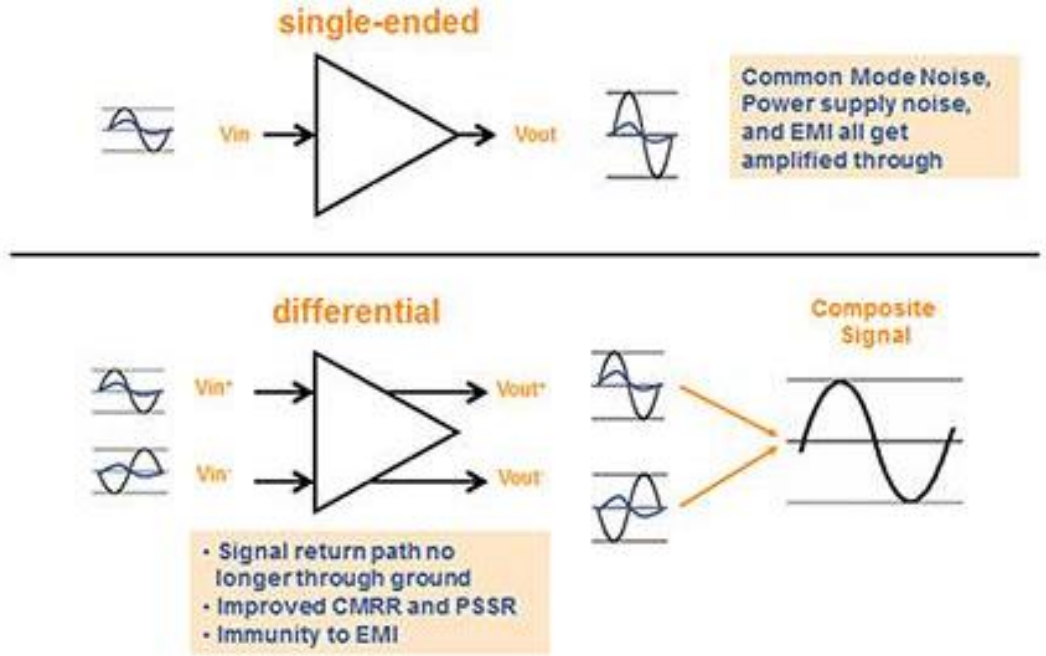
$$Vout = \left(C_0 + A^2C_3/2\right) + \left(C_1 - A^3C_4/4\right)\cos\omega_1 t + \left(A^2C_3/2\right)\cos 2\omega_1 t + \left(A^3C_4/4\right)\cos 3\omega_1 t + \ldots$$

DIFFERENTIAL

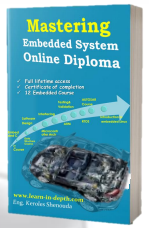
$$Vout = \sum C_n Vin^{+n} - C_n Vin^{-n}$$



$$Vout^+ - Vout^- = (0) + \left(0 - A^3C_4/2\right)\cos\omega_1 t + (0)\cos 2\omega_1 t - \left(A^3C_4/2\right)\cos 3\omega_1 t + \ldots$$



https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

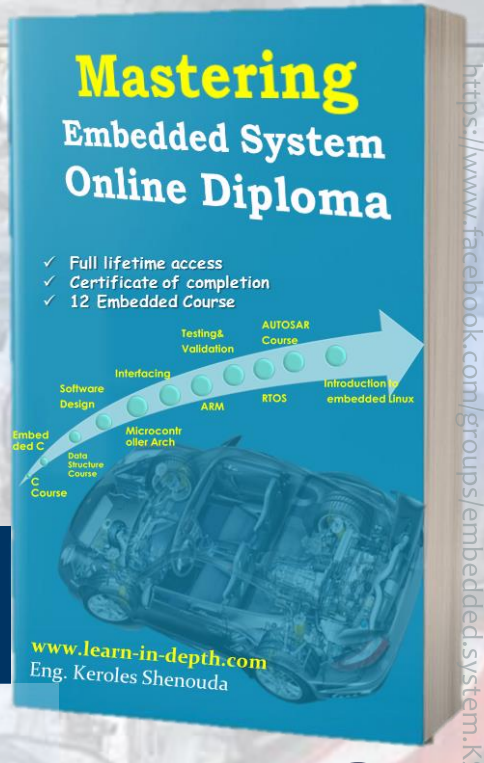


#LEARN_IN DEPTH

#Be_professional_in_embedded_system

Communication Specs Big Picture

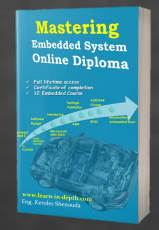
SUMMARIZE EVERYTHING BY YOUR HAND ☺



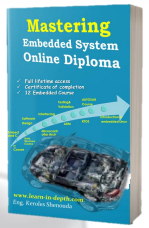
LEARN-IN-DEPTH
Be professional in embedded system

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

Communication Specs

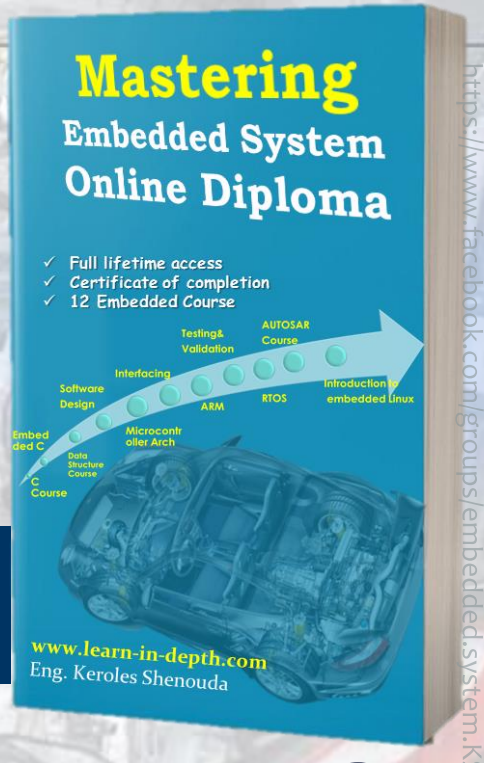


- Medium
  - Wire
  - Wireless
- N2N Relation
  - Peer to peer
  - Master/Slave
- Mode
  - Serial
  - Parallel
- Direction
  - Simplex
  - FD
  - HD
- Single ended / Differential wire
- Synch.
  - Synch
  - ASynch
- Throughput = Data bits / Frame



#LEARN_IN DEPTH

#Be_professional_in_embedded_system



UART

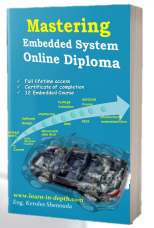
"Universal asynchronous receiver-transmitter"

- **UART** - Universal Asynchronous Receiver/Transmitter
- **USART** - Universal Synchronous/Asynchronous Receiver/Transmitter

LEARN-IN-DEPTH

Be professional in embedded system

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

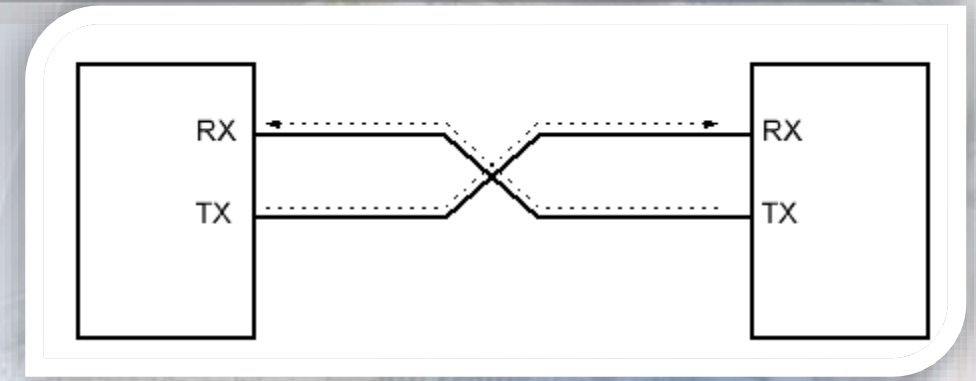


#LEARN_IN DEPTH

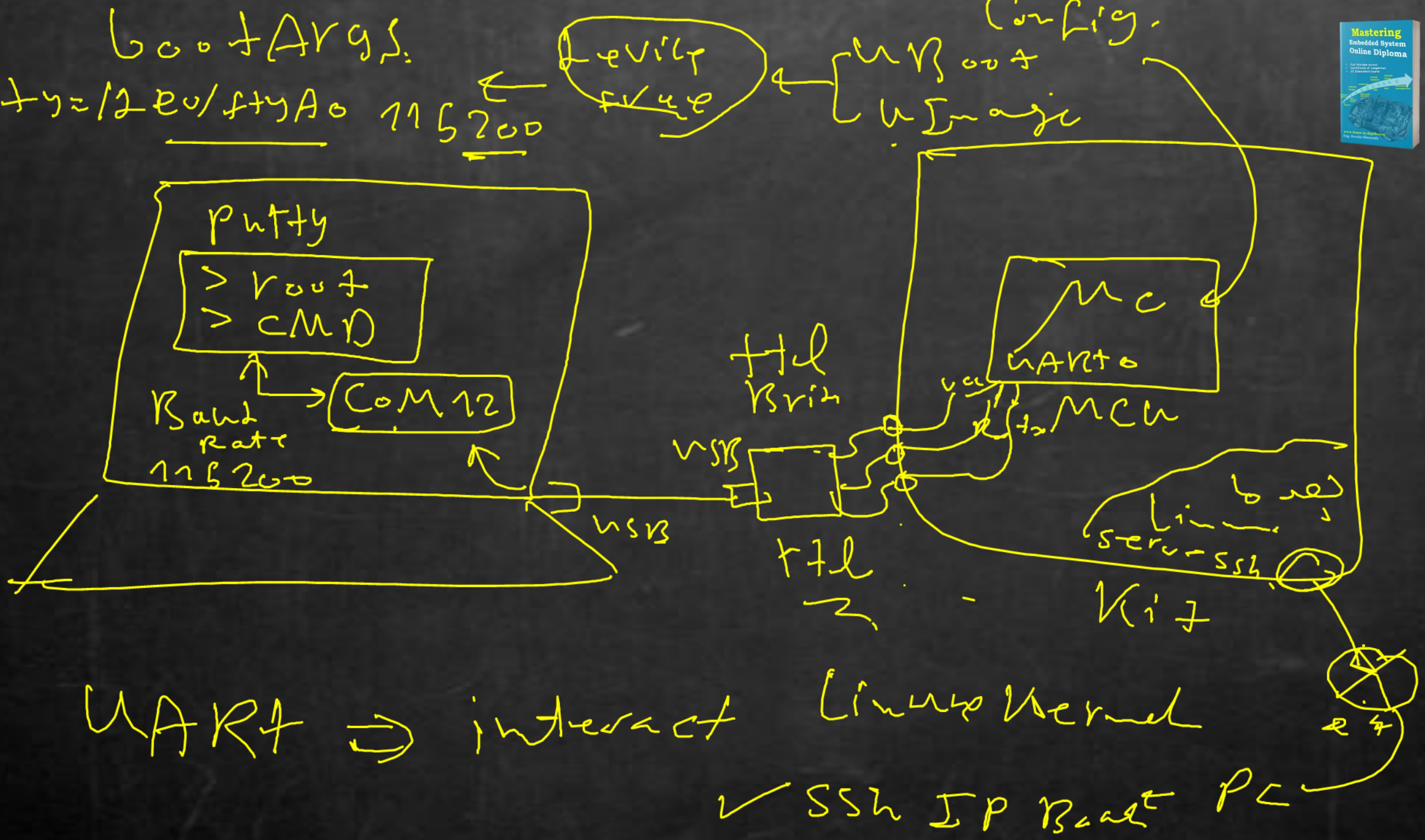
#Be_professional_in_embedded_system

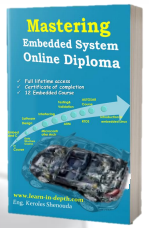
UART - Universal Asynchronous serial Receiver and Transmitter

- Full Duplex mode.(two wires)
- Asynchronous == No Clock
- Peer to peer
- Used to:
  - Make a communication channel between our micro-controller and our computer
  - talk to some sensors/chips, the most used ones are serial backpacks for lcd's and graphical lcd's and GPS modules that use almost always an serial interface.
  - The serial protocol is a fairly old protocol created many years ago it was used by terminals
  - It is Most used with **Embedded Linux** to show the Linux Kernel log until the login by the UART



https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

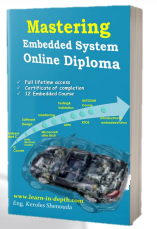

bootArgs:
tty=/dev/ttyA0 115200
devicetree
uBoot
uImage
Config.
Putty
> root
> CMD
Baud rate 115200
COM12
USB
HW Brdg
USB
RTL
MCU
UART0
Vcc
Tx
Linux
server-ssh
Kit
UART ⇒ interact
Linux Kernel
✓ SSH IP Boot PC



Framing the data

- Data chunk
  - The real meat of every serial packet is the data it carries.
  - The amount of data in each packet can be set to anything from 5 to 9 bits. Certainly, the standard data size is your basic 8-bit byte
- Synchronization bits
  - **stop bit(s)**. these bits mark the beginning and end of a packet. There's always only one start bit, but the number of stop bits is configurable to either one or two .
  - The start bit marks the beginning of a new word, When detected, the receiver synchronizes with the new data stream

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

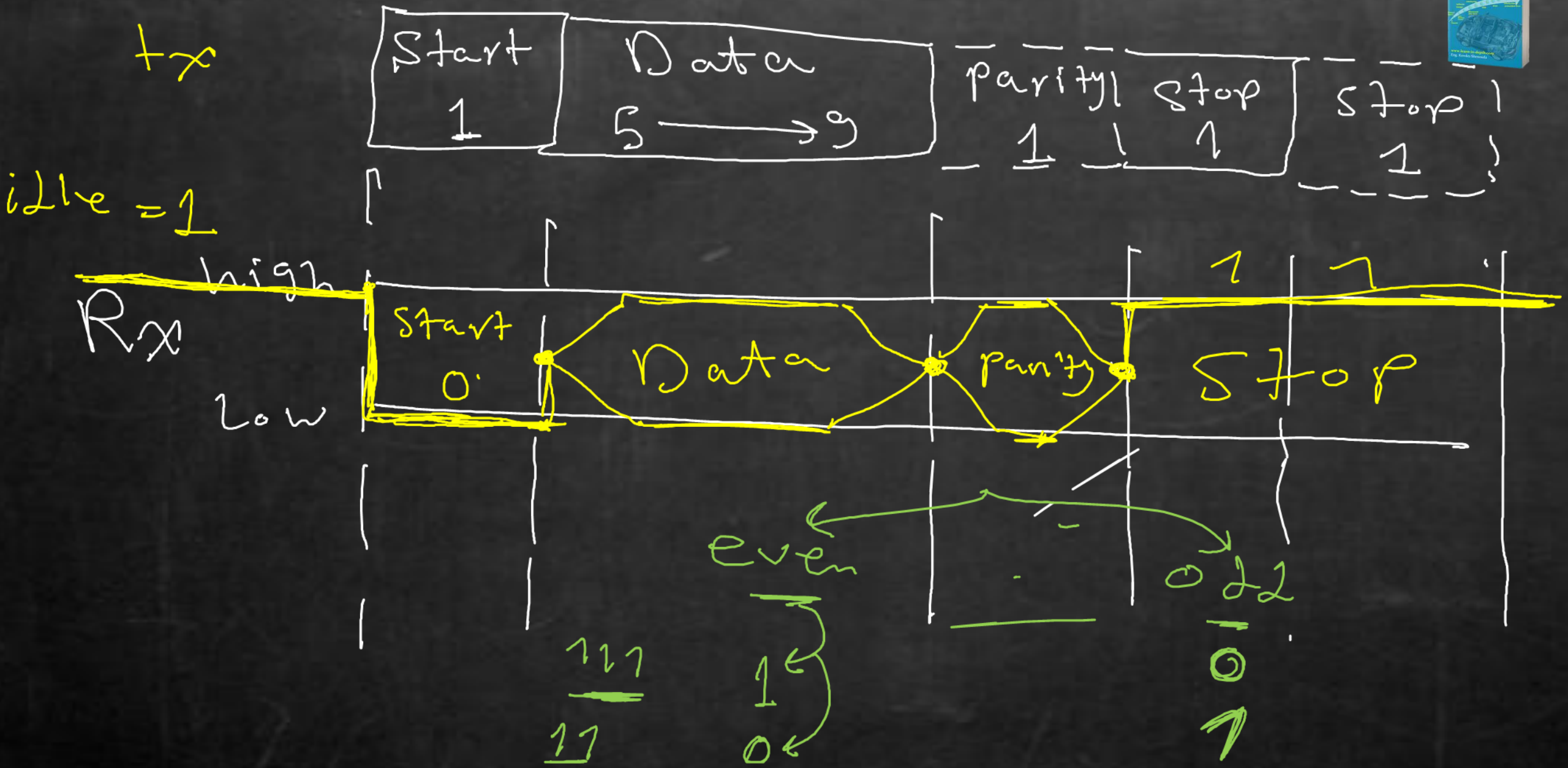


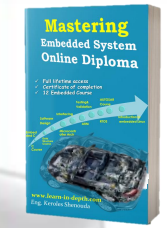
Framing the data

7 bits of data	(count of 1-bits)	8 bits including parity	
		even	odd
0000000	0	00000000	00000001
1010001	3	10100011	10100010
1101001	4	11010010	11010011
1111111	7	11111111	11111110

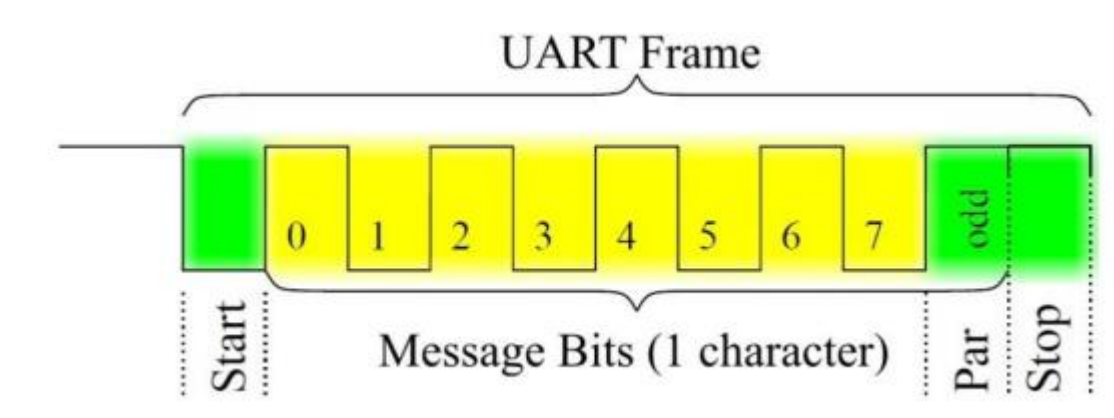
- Parity bits
- The parity bit is added to make the number of 1's even (even parity) or odd (odd parity)
- This bit can be used by the receiver to check for transmission errors

UART Frame.


tx
Start
1
Data
5 → 9
parity
1
stop
1
stop
1
idle = 1
high
Rx
low
Start
0
Data
parity
stop
1
1
1
even
0
1
0
1
1
1
0
1



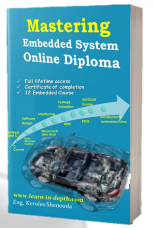
UART "Asynchronous " frame



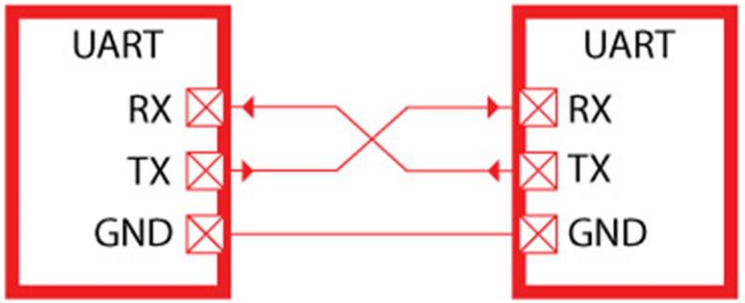
Each data frame consists of a **start bit**, a variable number of **data bits**, an **optional parity bit**, and **1 or 2 stop bits.** The most common configuration is 1 start bit, 8 data bits, no parity bit, and 1 stop bit ("8N1").

In asynchronous mode, there is no clock line: data is transmitted on the transmit line (Tx) and received on the receive line (Rx).
The UART is initialized by configuring control registers that determine the **baud rate, parity, number of stop bits:**

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

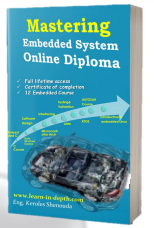


Why use a UART?

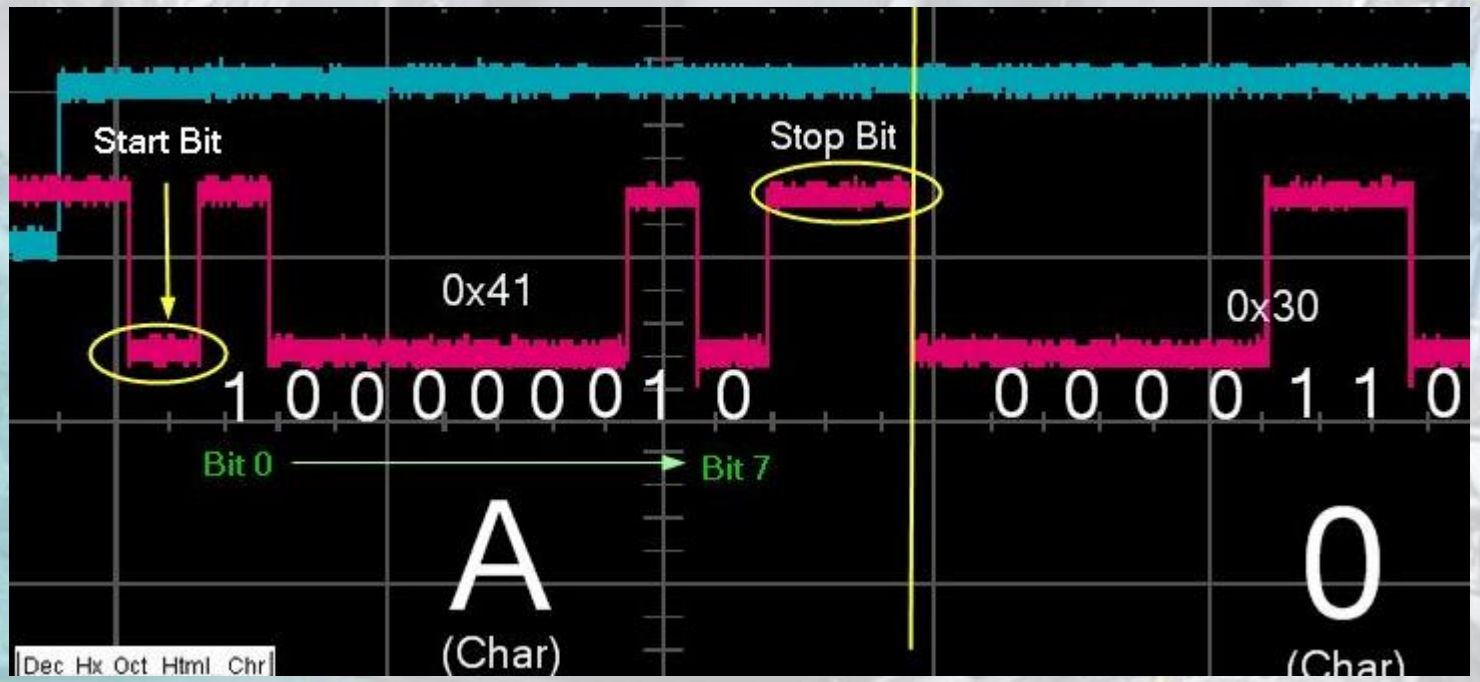


- A UART may be used when:
  - - High speed is not required
  - - A cheap communication line between two devices is required
- Asynchronous serial communication is very cheap
  - - Requires a transmitter and/or receiver
  - - Single wire for each direction (plus ground wire)
  - - Relatively simple hardware

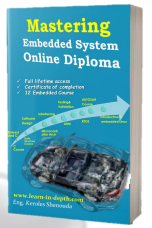
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



UART frame



https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

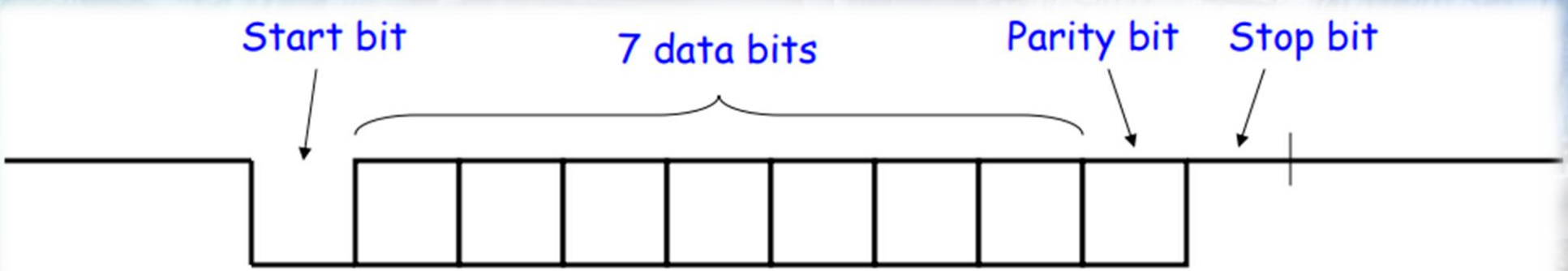


#LEARN_IN DEPTH

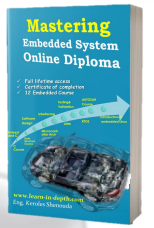
#Be_professional_in_embedded_system

UART Character Transmission

- Assume the data width 7 bit
- In the configuration shown, it takes 10 bits to send 7 bits of data

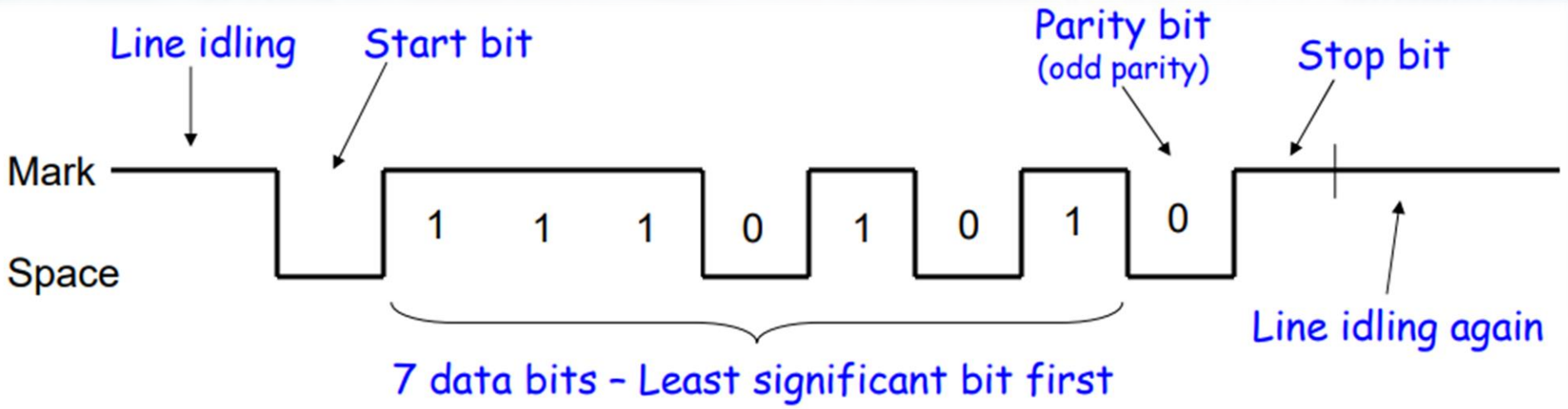


https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

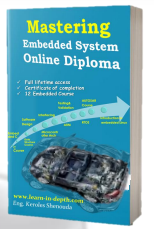


UART Character Transmission

- Assume the data width 7 bit
- In the configuration shown, it takes 10 bits to send 7 bits of data
- • Send the ASCII letter 'W' (1010111)

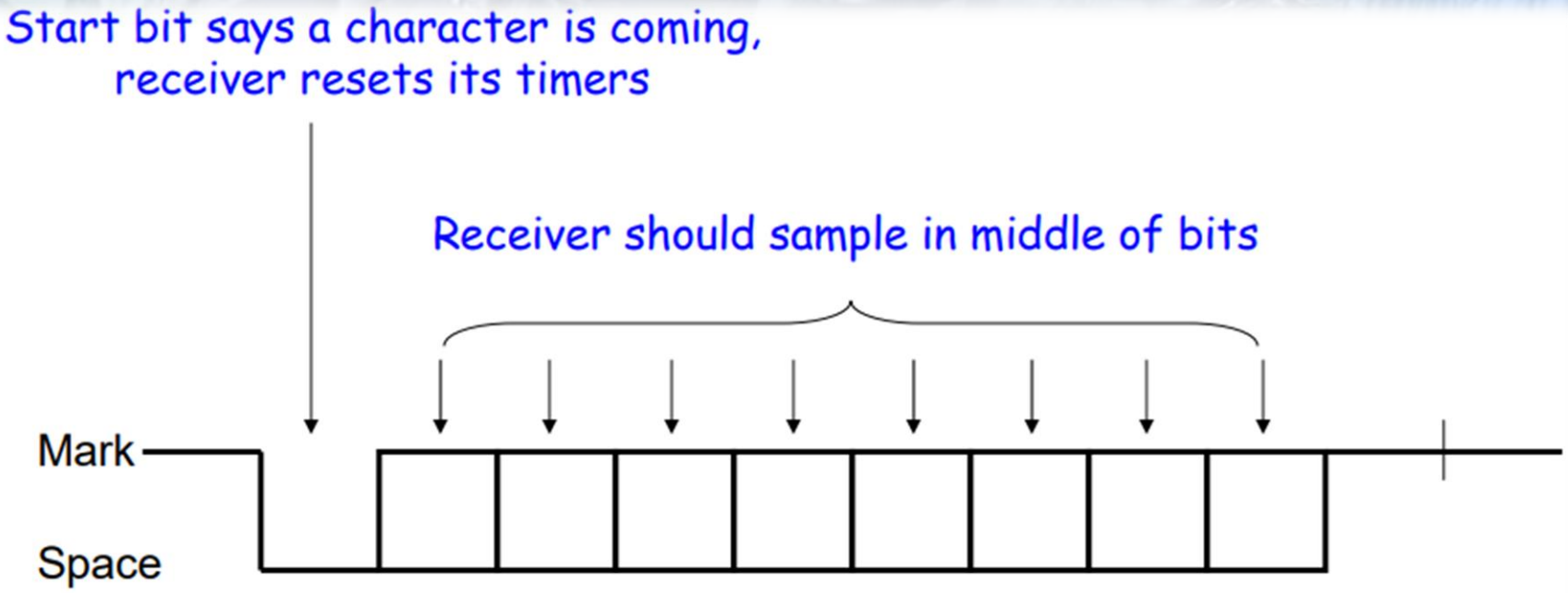


https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

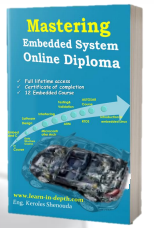


UART Character Reception

- Receiver uses a timer (counter) to time when it samples.
- **Transmission rate (i.e., bit width) must be known!**

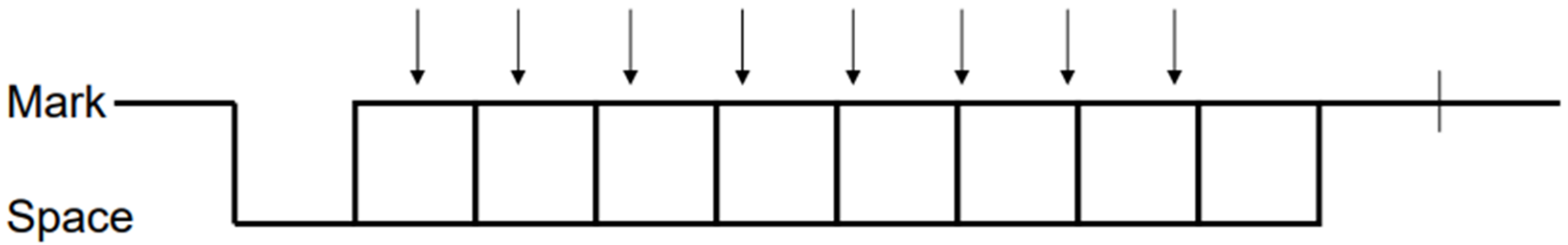


https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

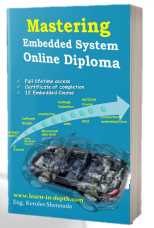


UART Character Reception

If receiver samples too quickly, see what happens…

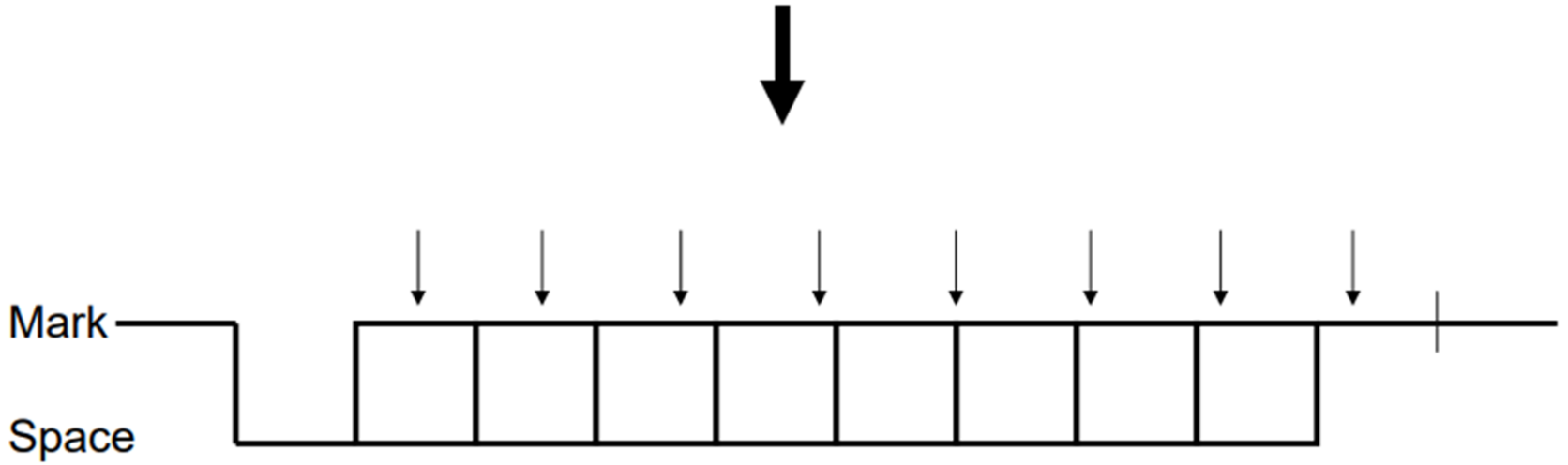


https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



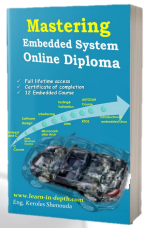
UART Character Reception

If receiver samples too slowly, see what happens…



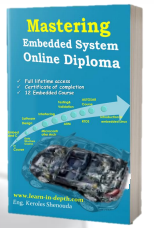
Receiver resynchronizes on every start bit.
Only has to be accurate enough to read 9 bits.

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



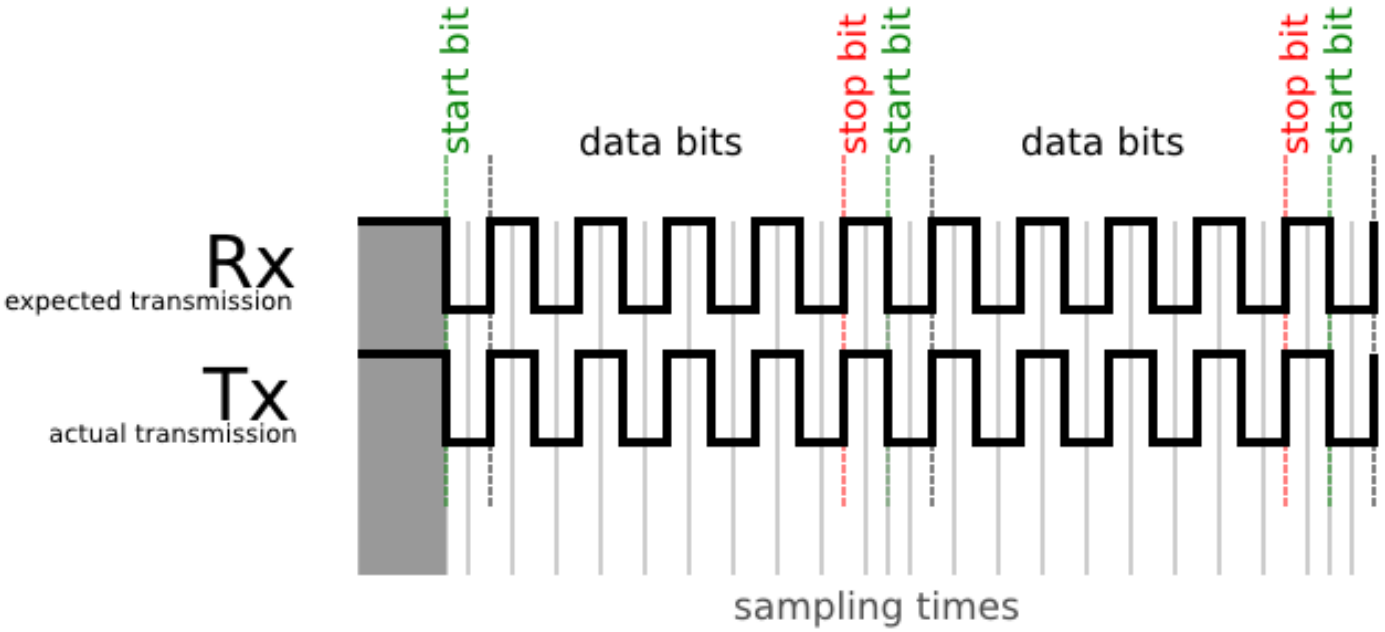
UART Character Reception

- Receiver also verifies that stop bit is '1'
  - If not, reports "**framing error**" to host system
- New start bit can appear immediately after stop bit
  - Receiver will **resynchronize** on **each start bit**



So to keep the receiver Sampling in middle of bits

Receiver and Transmitter Need to be on the Same Baud rate / Bitrate



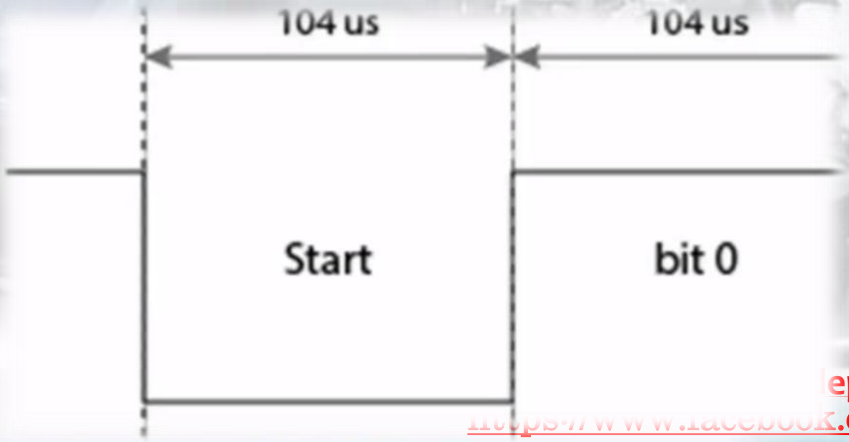
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

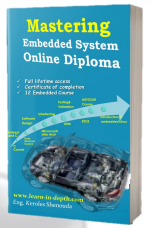
Baud Rate

- Baud Rate is The "symbol rate" of the transmission system
- **For a UART**, baud rate **is same as the number of bits per second (bps)**
- Each bit is 1/(rate) seconds wide
- Example: - 9600 baud 9600 Hz
  - 9600 bits per second (bps)
  - Each bit is 1/(9600 Hz) ≈ 104.17 μs long

Bit Duration

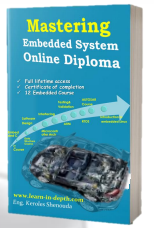
- Each bit is transmitted for a fixed duration
- The duration must be known to Tx and Rx
- Baud rate (f) determines the duration (T)
- Baud rate is the number of transitions per second
  - Typically measured in "bits per second (bps)"
- T = 1/f
  - F = 9600 baud, T = ~104 microsec





Data Throughput Example

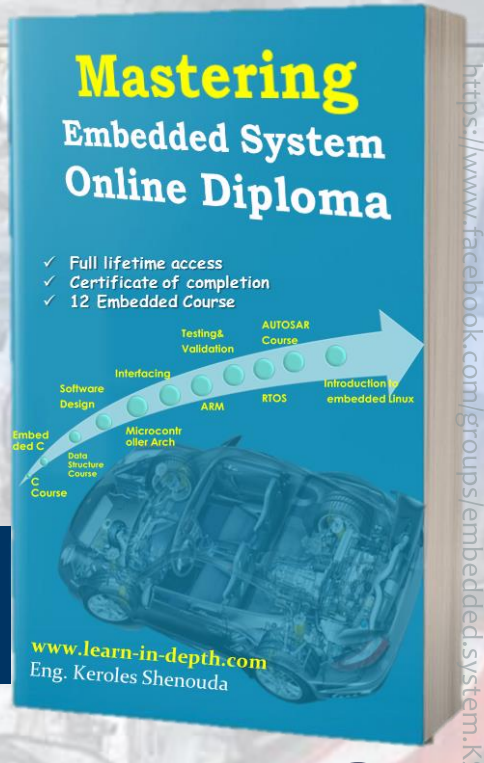
- Data Throughput Example
  - Assume 19200 baud, 8 data bits, no parity, 1 stop bit
    - 19200 baud → 19.2 kbps
    - 1 start bit + 8 data bits + 1 stop bit → 10 bits
  - It takes 10 bits to send 8 bits (1 byte) of data
  - 19.2 kbps • 8/10 = **15.36 kbps**



#LEARN_IN DEPTH

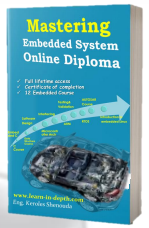
#Be_professional_in_embedded_system

What is the difference between USART vs UART



LEARN-IN-DEPTH
Be professional in embedded system

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



USART

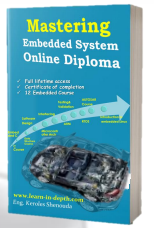
Universal Synchronous/Asynchronous Receiver/Transmitter

- The first difference between a **USART** and a **UART** is the way in which the serial data may be clocked.
- A UART generates its data clock internally to the microcontroller and synchronizes that clock with the data stream by using the **start bit** transition. There is **no incoming clock signal that is associated with the data**,

so in order to properly receive the data stream the receiver needs to know ahead of time what the baud rate should be.

- A **USART**, on the other hand, can be set up to run in synchronous mode. In this mode the sending peripheral will generate a clock that the receiving peripheral can recover from the data stream without knowing the baud rate ahead of time. Alternatively, the link will use a completely **separate line to carry the clock signal**. The use of the external clock allows the data rate of the USART to be much higher than that of a standard UART, **reaching up to rates of 4 Mbps**.

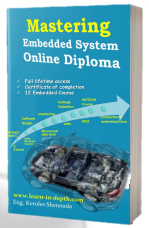
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



Are USARTs and UARTs the same?

- Technically the answer is **no**. A USART generally has more capabilities that a standard UART and the ability to generate clocked data allows the USART to operate at baud rates well beyond a UART's capabilities.

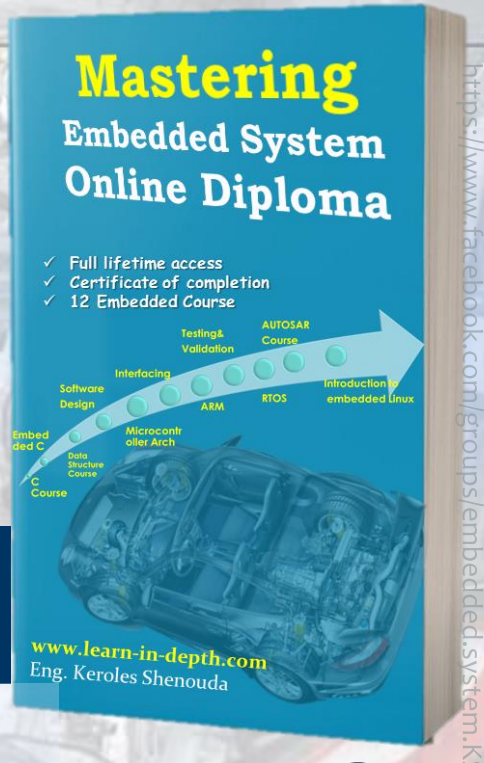
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



#LEARN_IN DEPTH

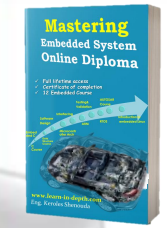
#Be_professional_in_embedded_system

Wire standard Conjugation with USART protocol



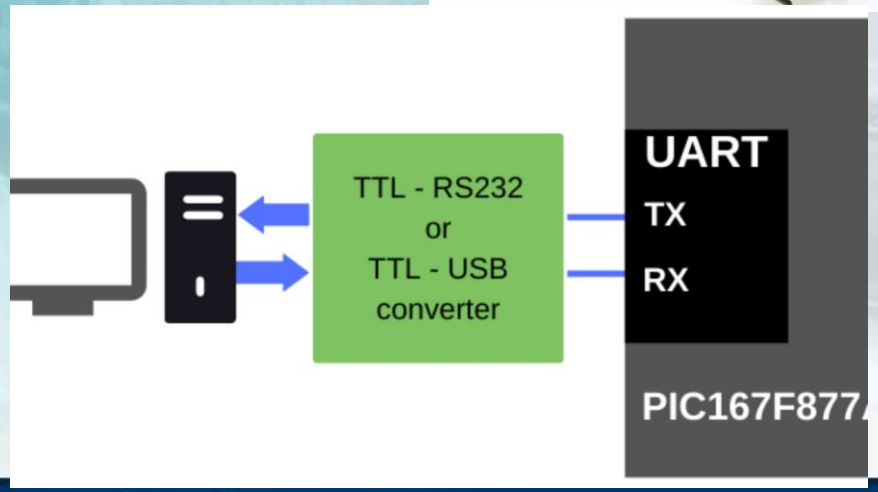
LEARN-IN-DEPTH
Be professional in embedded system

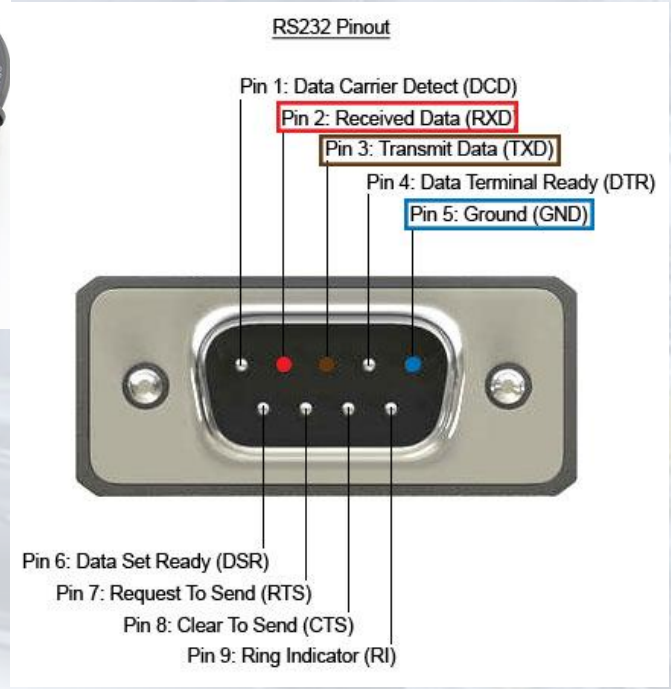
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



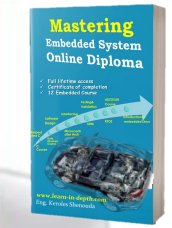
Wire standard Conjugation with USART protocol

- RS-232
- USB
- … etc

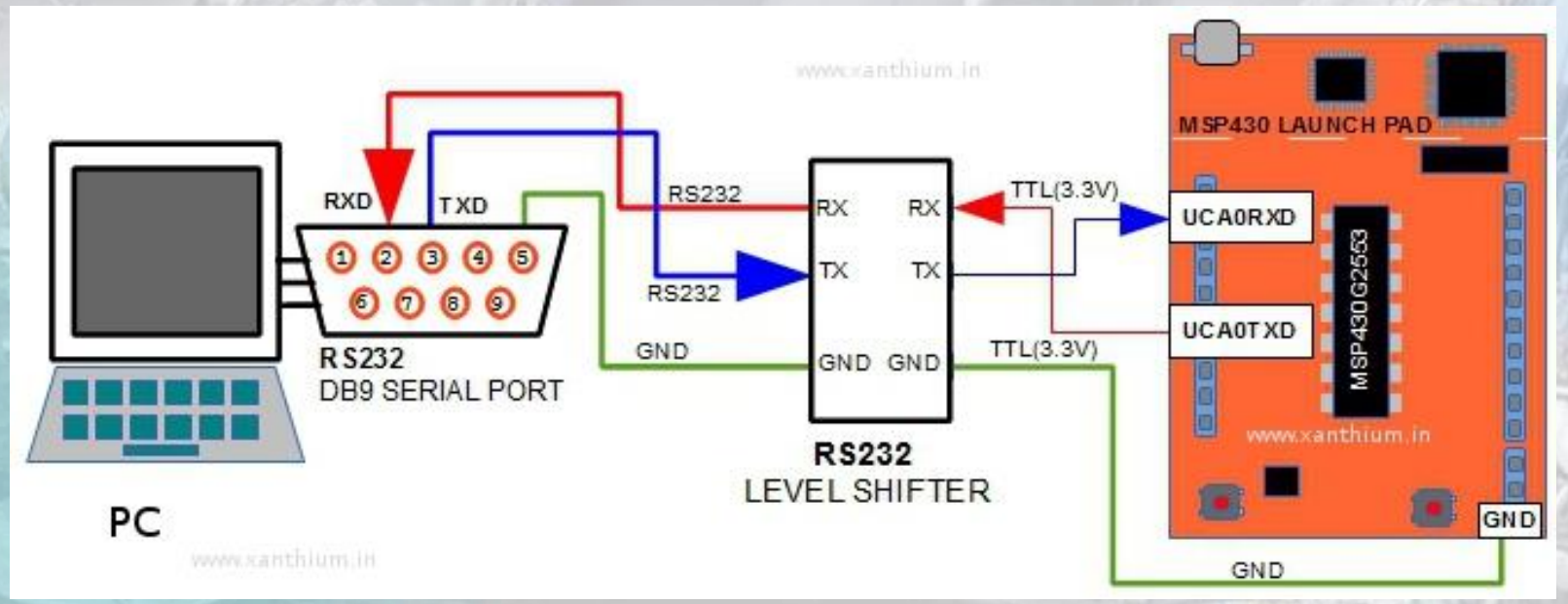




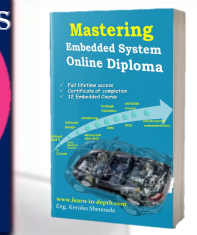
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



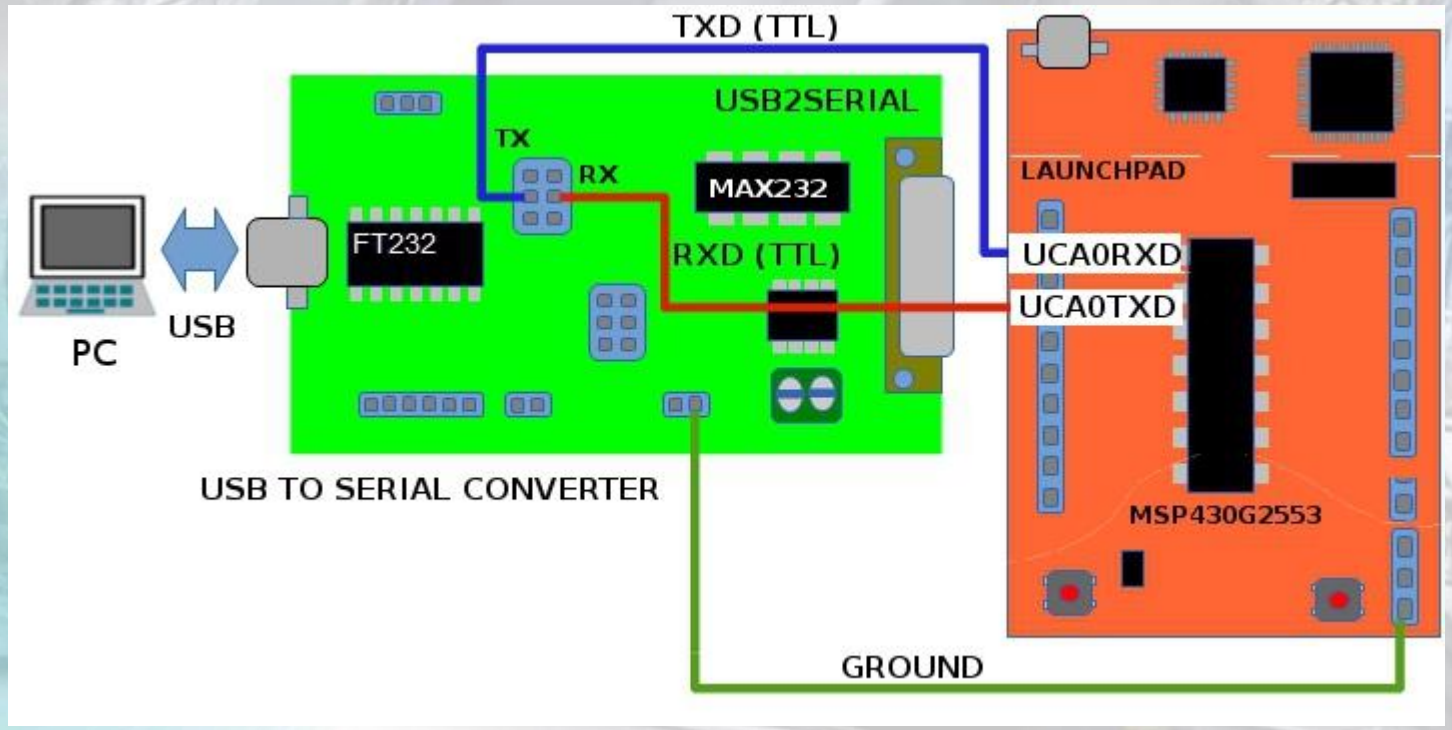
Example 1



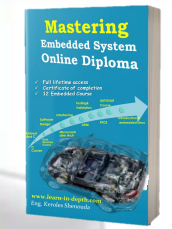
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



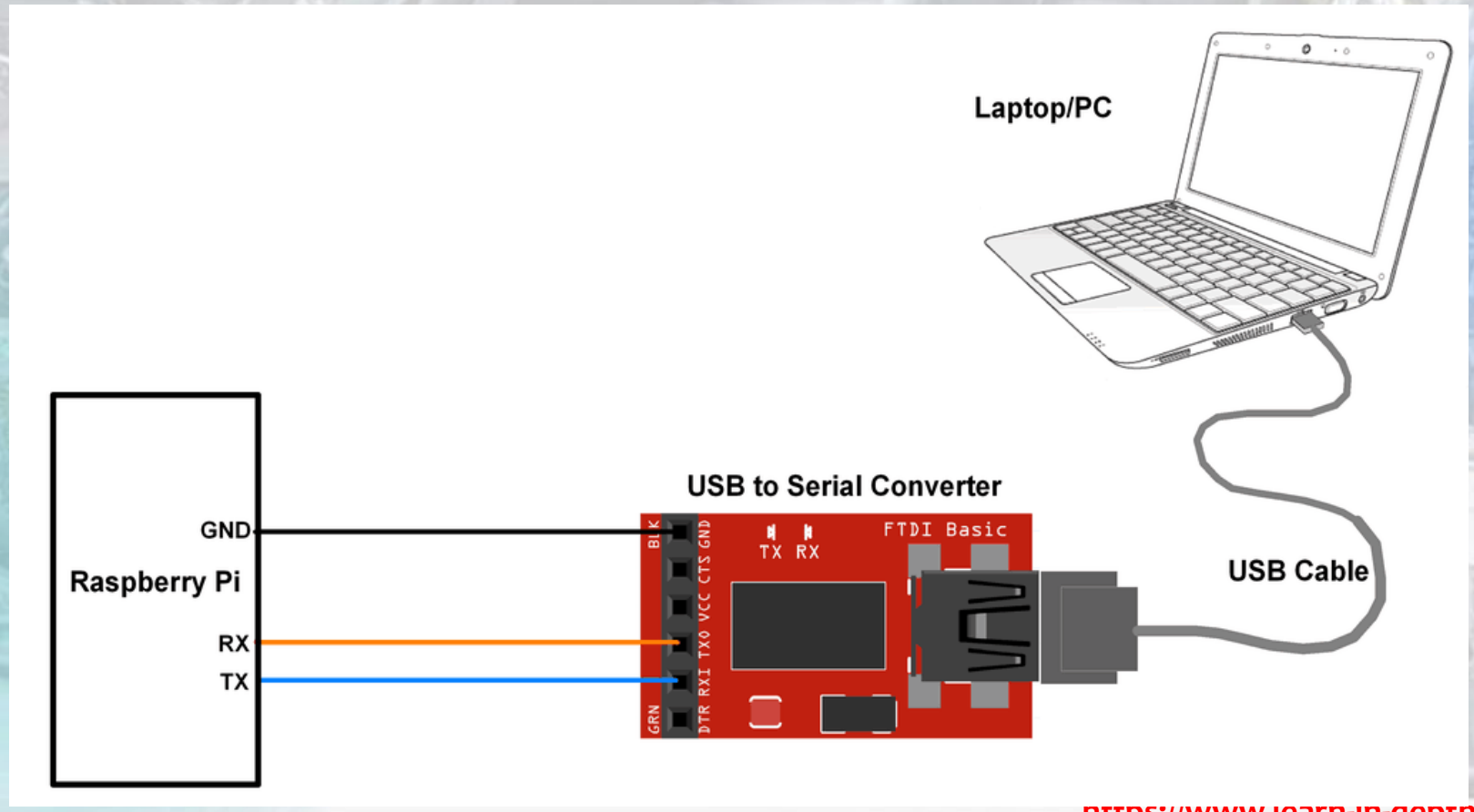
Example 2



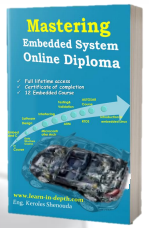
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



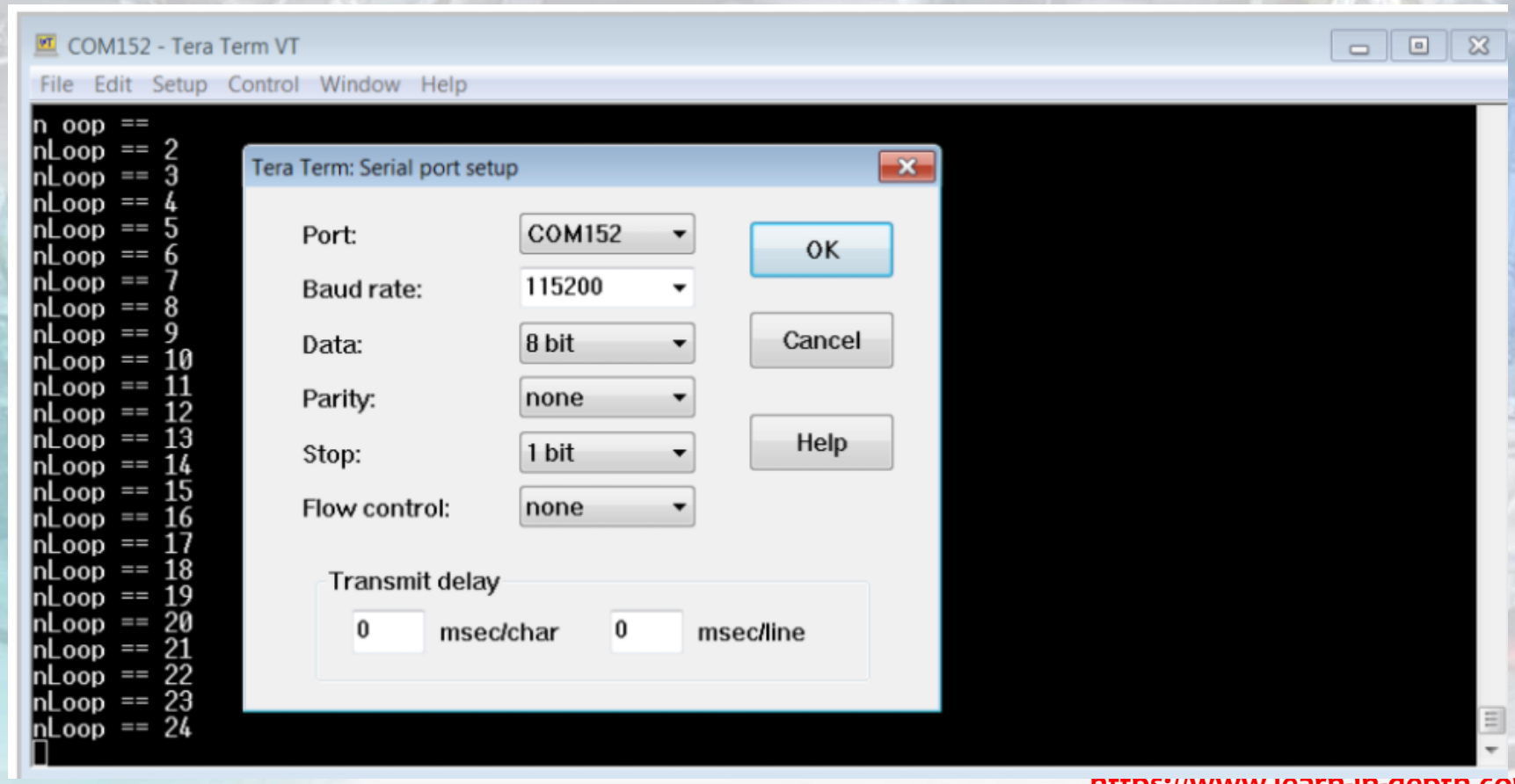
Example



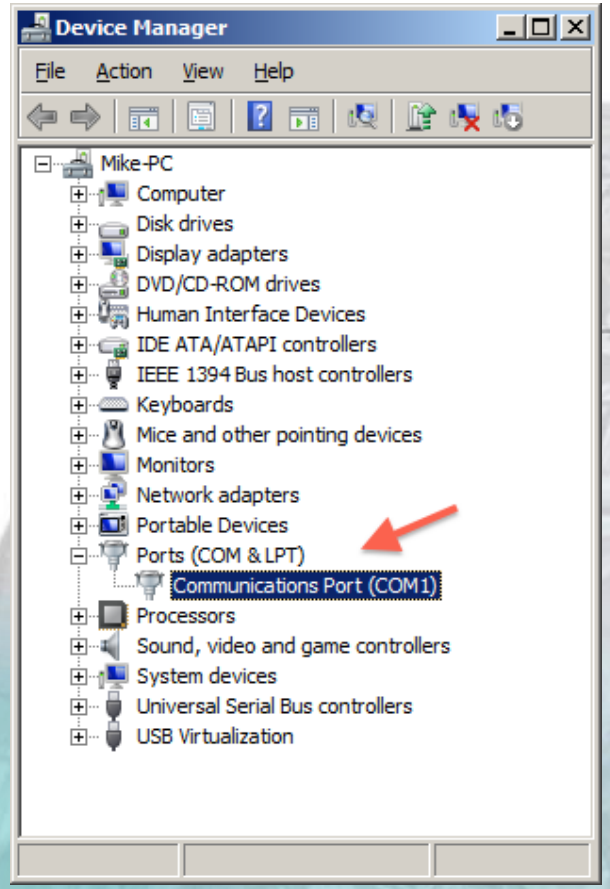
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

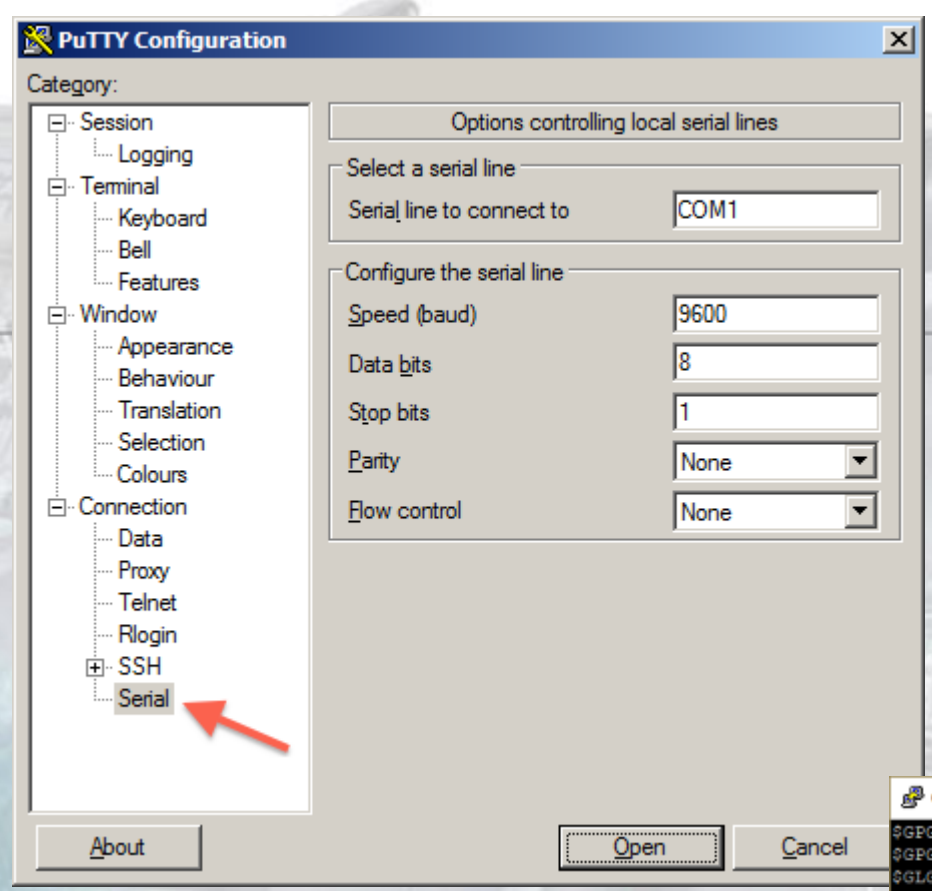


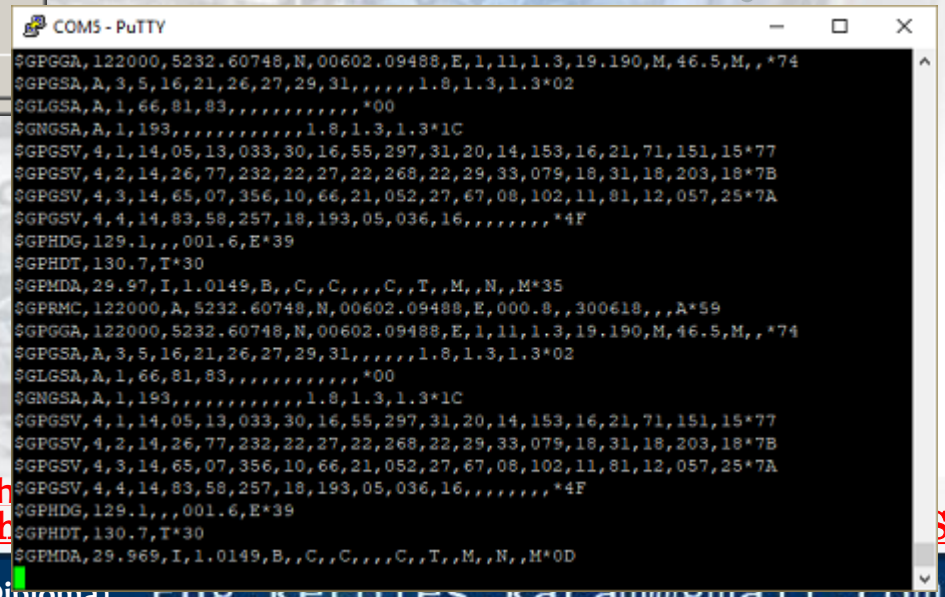
Example

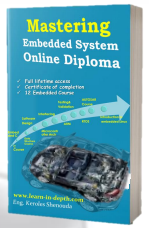


https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



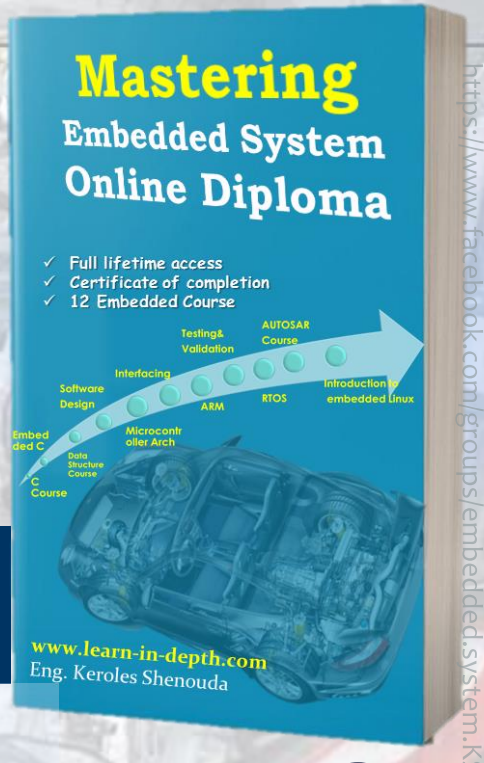






#LEARN_IN DEPTH

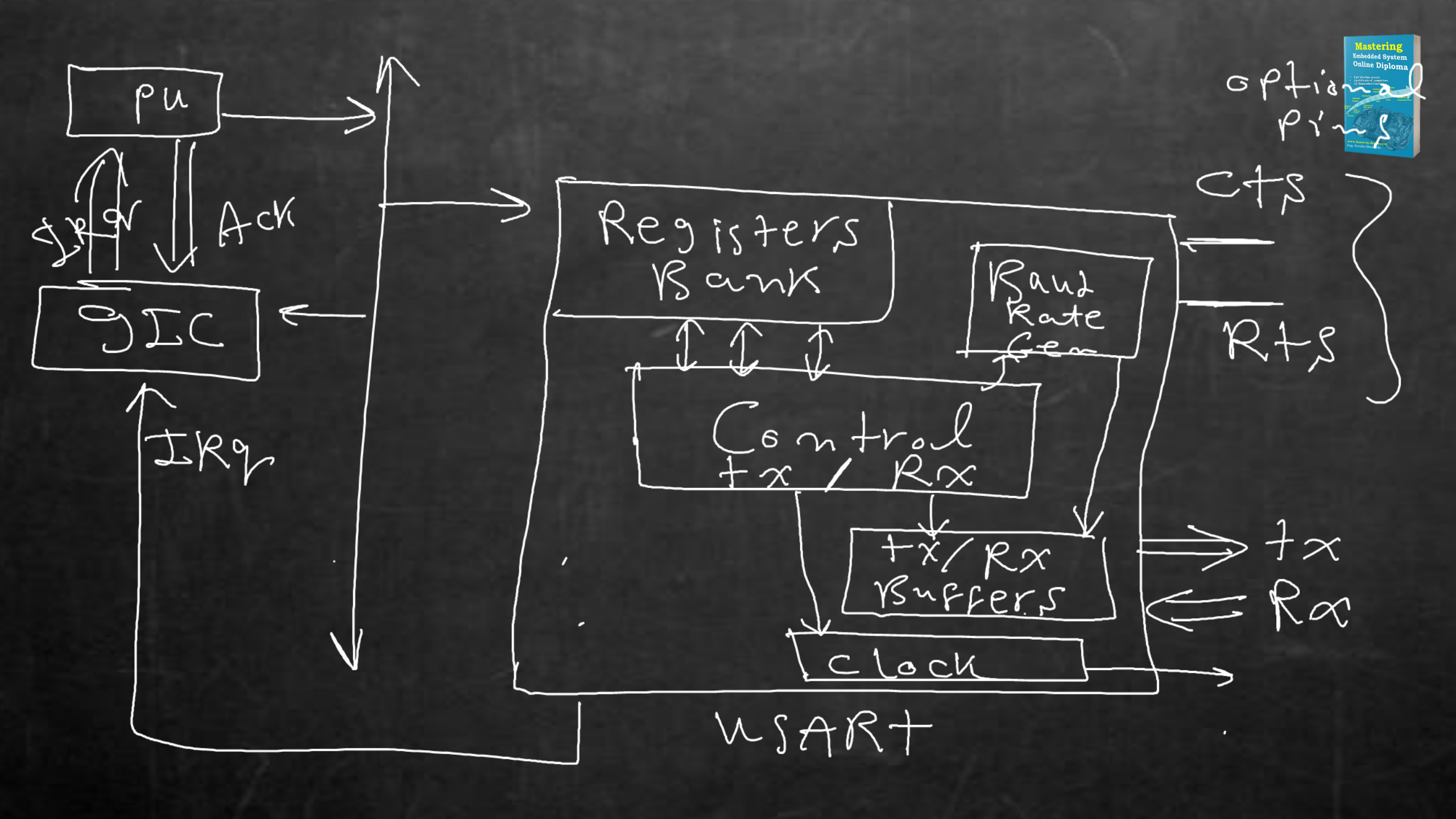
#Be_professional_in_embedded_system

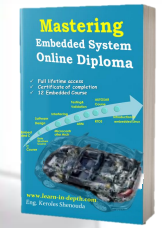


USART Controller General Circuit

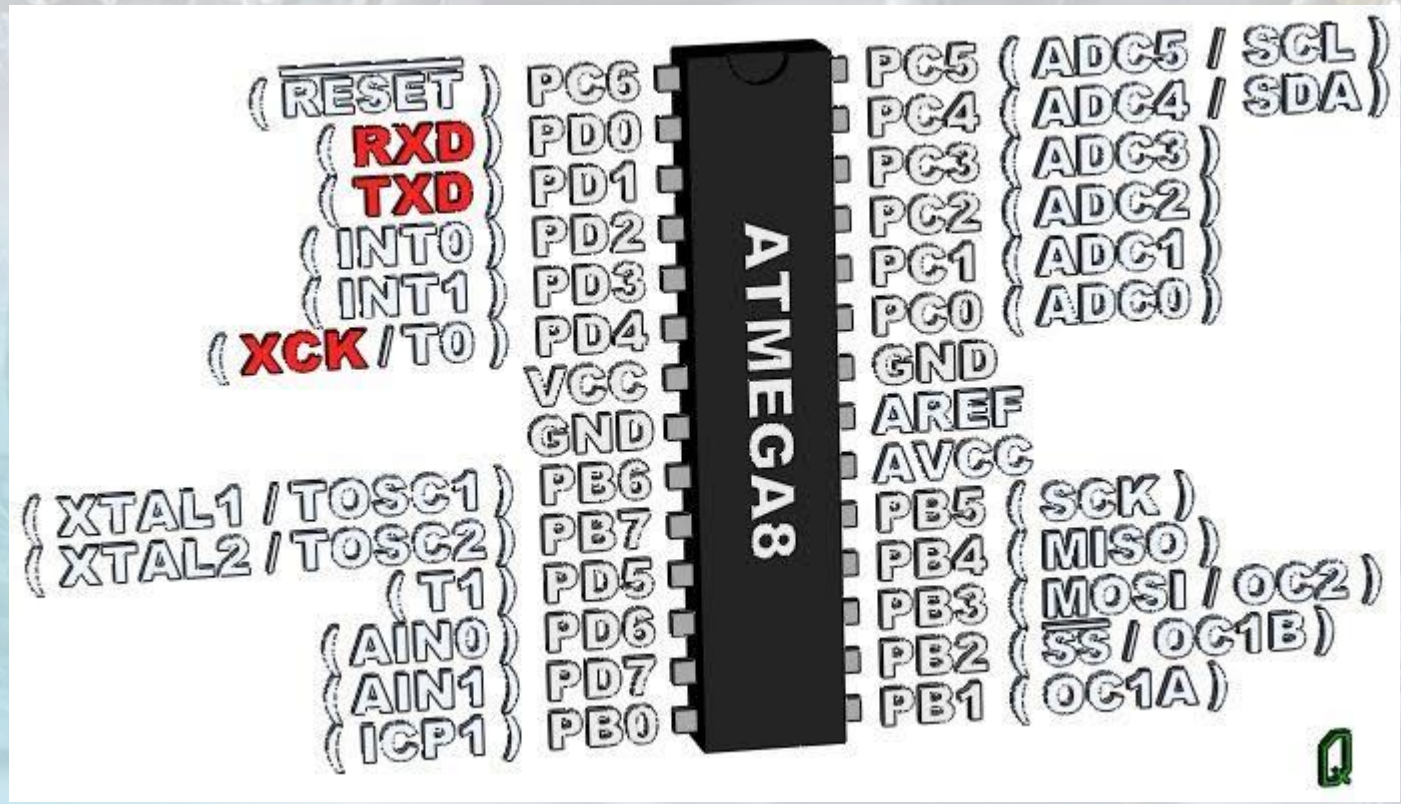
LEARN-IN-DEPTH
Be professional in embedded system

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

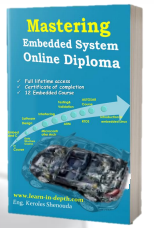

pu
Req
Ack
PIC
IRq
Registers Bank
Baud Rate Gen
Control Tx / Rx
Tx / Rx Buffers
Clock
USART
optional Pins
CTS
RTS
Tx
Rx



Atmega UART Pins

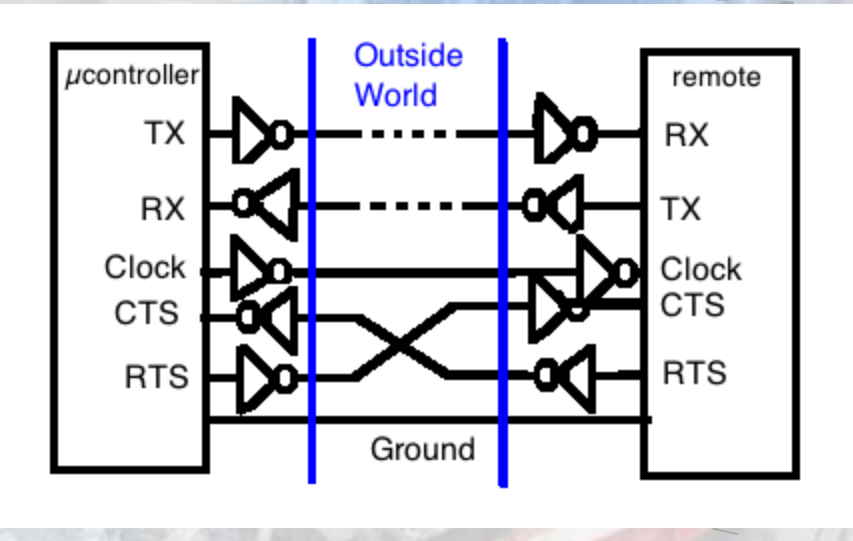


https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

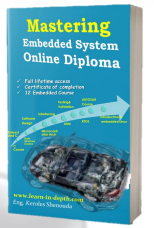


USARTS PINS (FOR FLOW CONTROL)

- A USART has three extra pins, clock, Clear To Send (CTS), and Ready To Send (RTS).
- **Instead of having a prearranged baud rate**, one of the two devices can send out a clock signal, a square wave at the baud frequency. This clock signal is used to synchronize the two devices
- The Clock signal isn't available on 9-pin RS-232 connectors that was introduced on the IBM PC-AT in 1984.
- USARTs support two **flow control** signals, CTS and RTS, that are used to throttle data flow between the two devices. A receiver may run low on buffer space and, instead of throwing away data, can turn RTS on, saying "hey I'm not ready, please stop". RTS is connected to CTS on the transmitter, when the transmitter notices that RTS is on, it stops transmitting. Once the receiver processes the data and frees up some buffer space, it will turn RTS off again. The transmitter detects CTS is off and can continue transmitting. This is known as **hardware handshaking**.

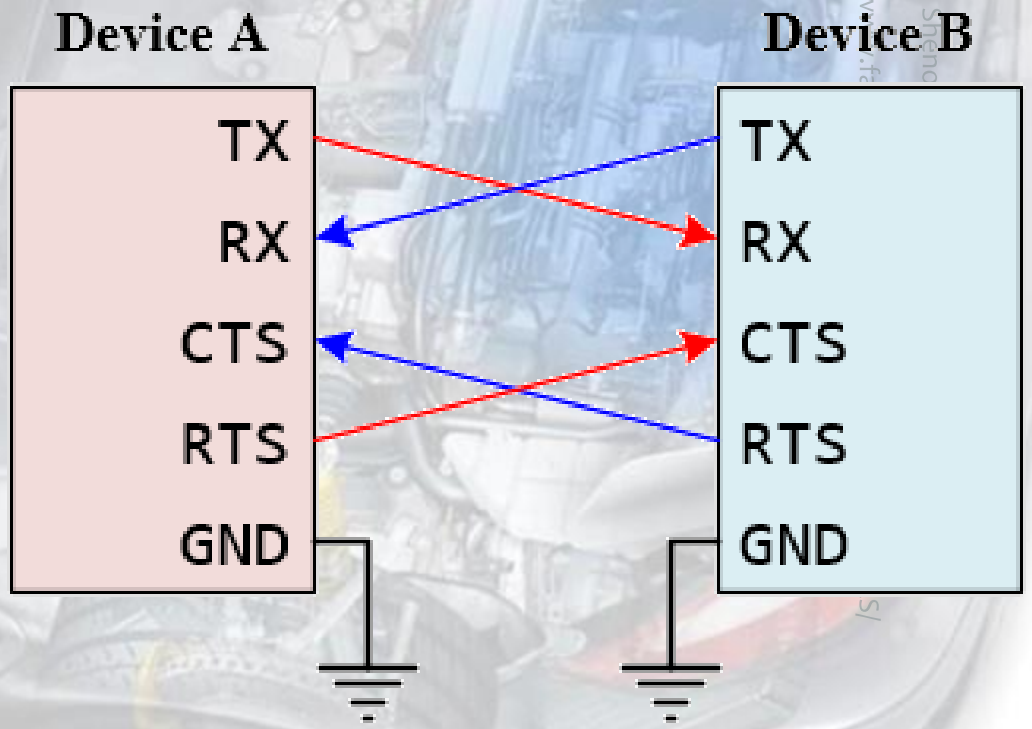


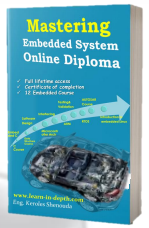
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



RTS/CTS Flow Control

- When hardware flow control is enabled, two extra wires are needed in addition to the data lines. They are called **RTS** (Request to Send) and **CTS** (Clear to Send). The CTS signal is an input to the UART that is set by the other UART in the system when it is OK to send data on the bus. The RTS signal is an output of the UART informing the other UART on the bus that it is ready to receive data. The RTS line of one UART is connected to the CTS line of the other UART and vice versa. These lines are only valid before transmission is started. If the signal is set or cleared after a transfer is started the change will only affect the next transfer.

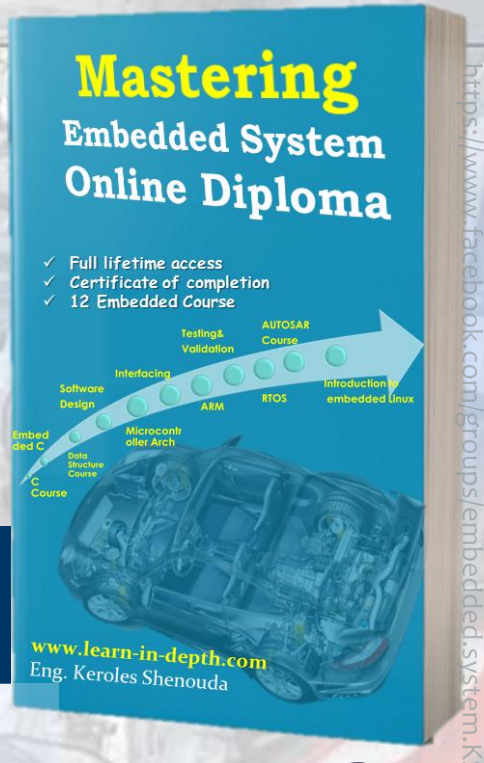




#LEARN_IN DEPTH

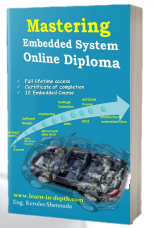
#Be_professional_in_embedded_system

UART General Configurations



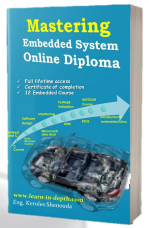
LEARN-IN-DEPTH
Be professional in embedded system

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



UART Generic Configurations

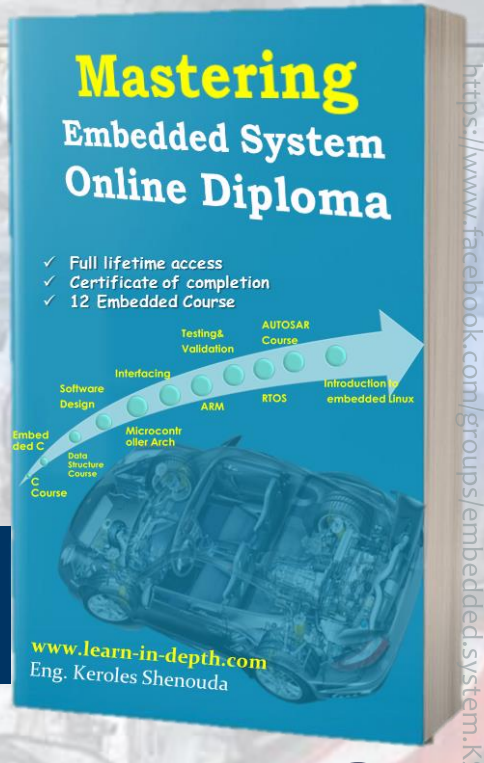
- Both UART communicating devices must operate at the same baud rate and have same configuration for successful communication:
- **Baud Rate**
  - Standard Mode: 110 to 115,200 bps
  - Fast Mode: 230,400 ~ 921,600 bps, up to 4 Mbps
- **Modes**: Full-duplex, half-duplex, TX only, RX only
- **Data Bits**: 5 ~ 9 bits
- **Endianess**: Some UART devices offer the option to send the data in either LSb or MSb. The UART is almost always LSb.
- **Parity Types**: None, Even, Odd, Mark/Space
- **Stop Bits**: 1 or 2
- **Flow Control**: None, Hardware (CTS/RTS)



#LEARN_IN DEPTH

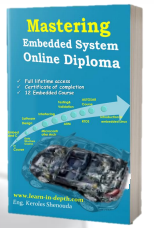
#Be_professional_in_embedded_system

USART Block Diagram on Atmega32



LEARN-IN-DEPTH
Be professional in embedded system

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

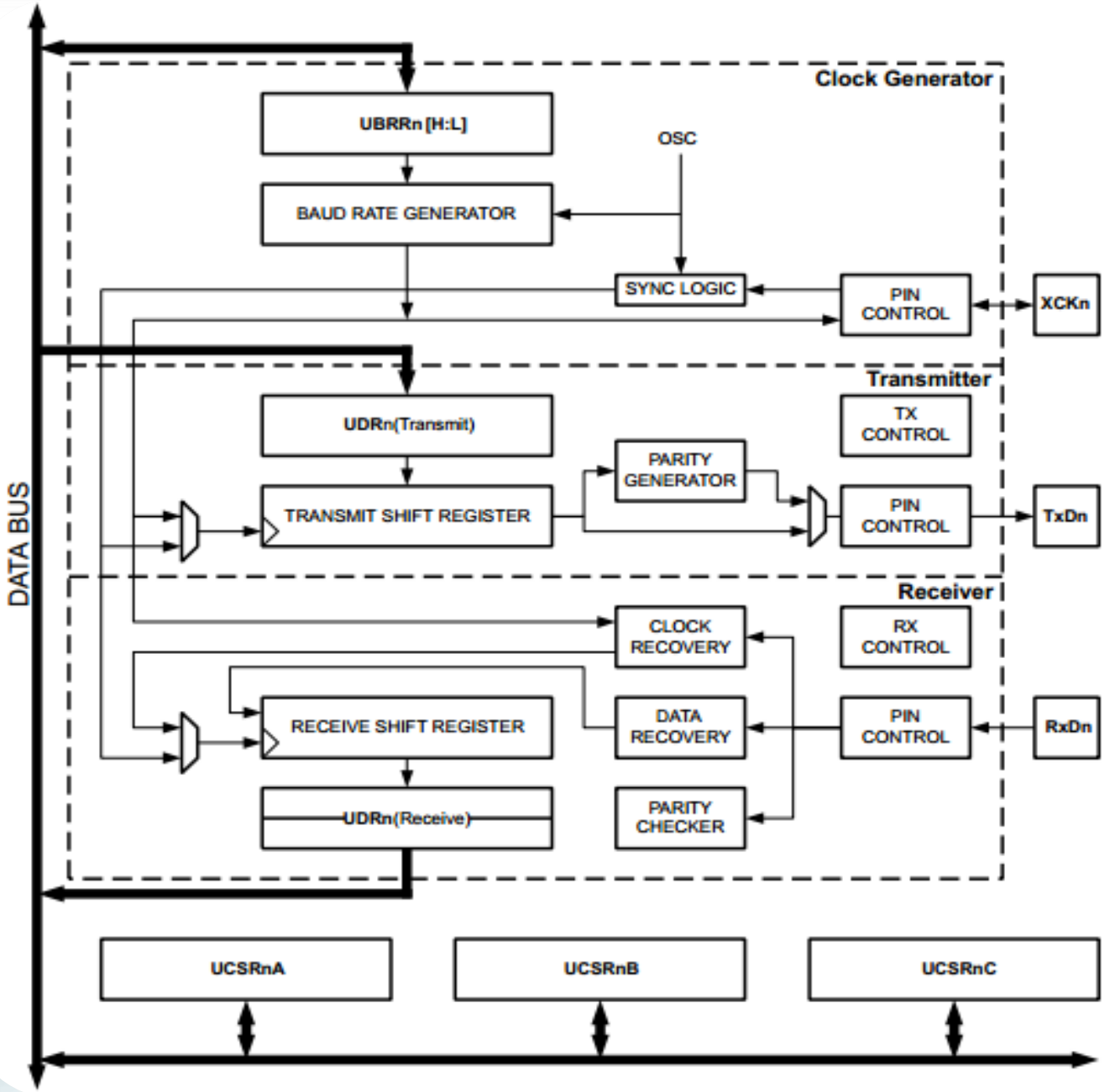


USART - Universal Synchronous and Asynchronous serial Receiver and Transmitter

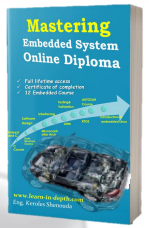
23.1. Features

- Full Duplex Operation (Independent Serial Receive and Transmit Registers)
- Asynchronous or Synchronous Operation
- Master or Slave Clocked Synchronous Operation
- High Resolution Baud Rate Generator
- Supports Serial Frames with 5, 6, 7, 8, or 9 data bits and 1 or 2 stop bits
- Odd or Even Parity Generation and Parity Check Supported by Hardware
- Data OverRun Detection
- Framing Error Detection
- Noise Filtering Includes False Start Bit Detection and Digital Low Pass Filter
- Three Separate Interrupts on TX Complete, TX Data Register Empty and RX Complete
- Multi-processor Communication Mode
- Double Speed Asynchronous Communication Mode

USART Block Diagram



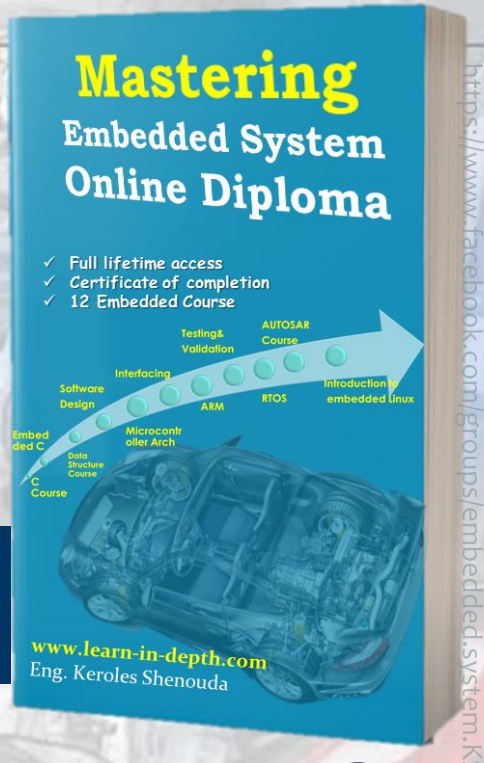
USART Block Diagram



#LEARN_IN DEPTH

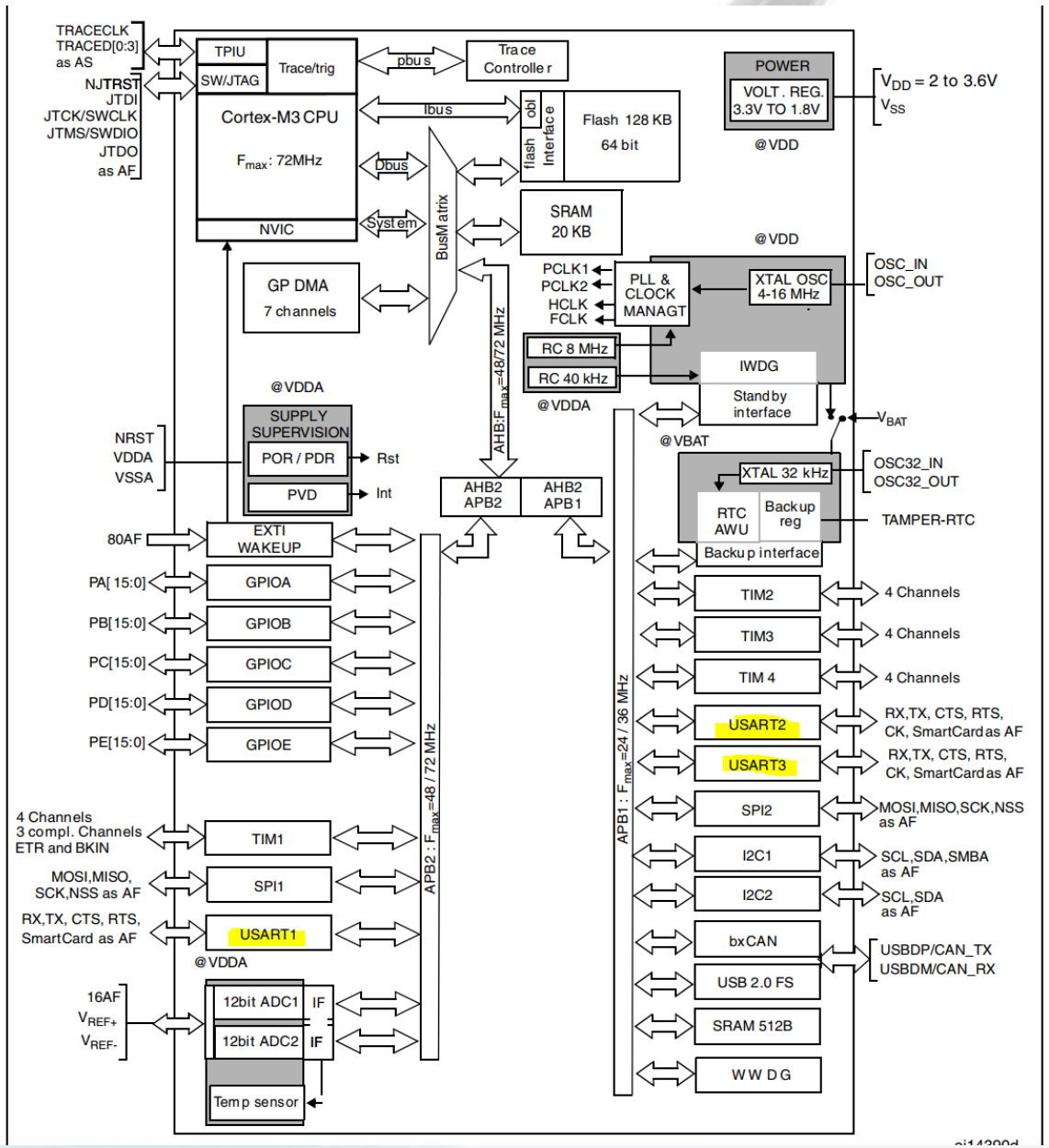
#Be_professional_in_embedded_system

USART Block Diagram on STM32F103X



LEARN-IN-DEPTH
Be professional in embedded system

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



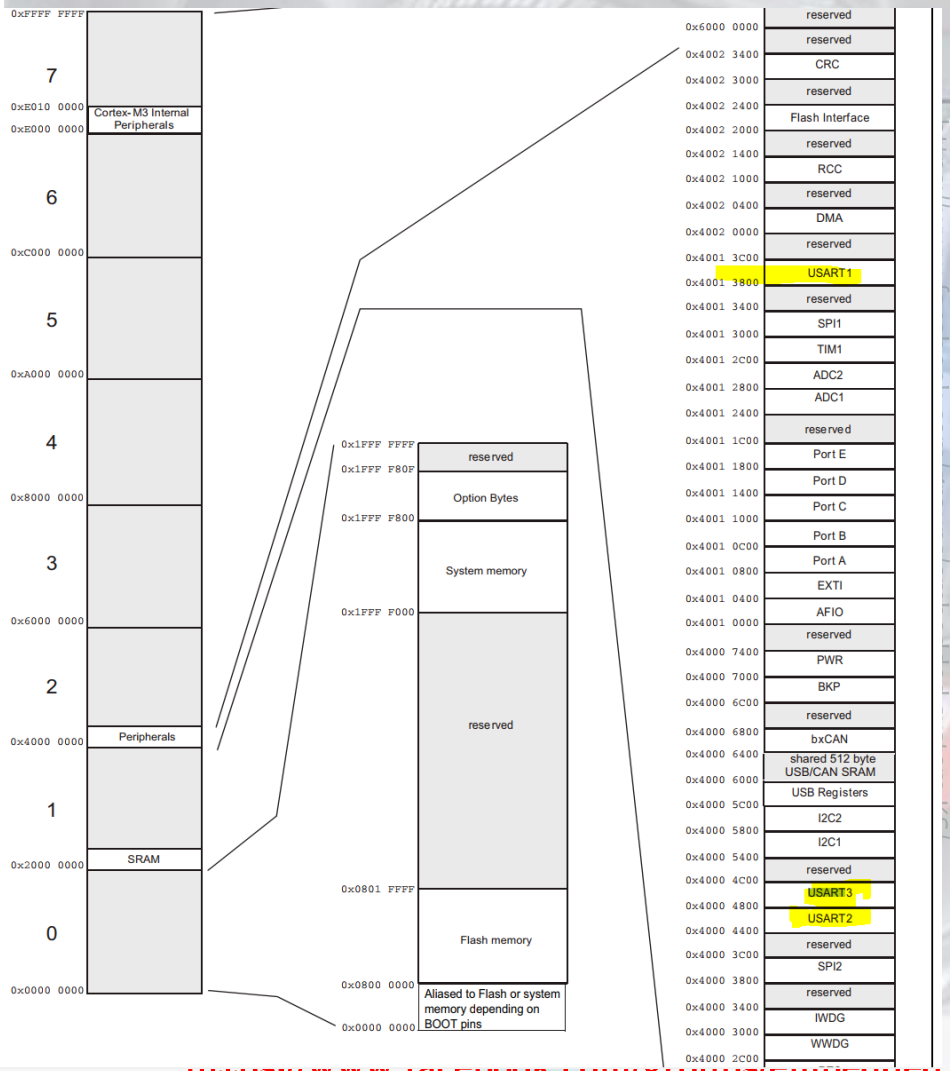
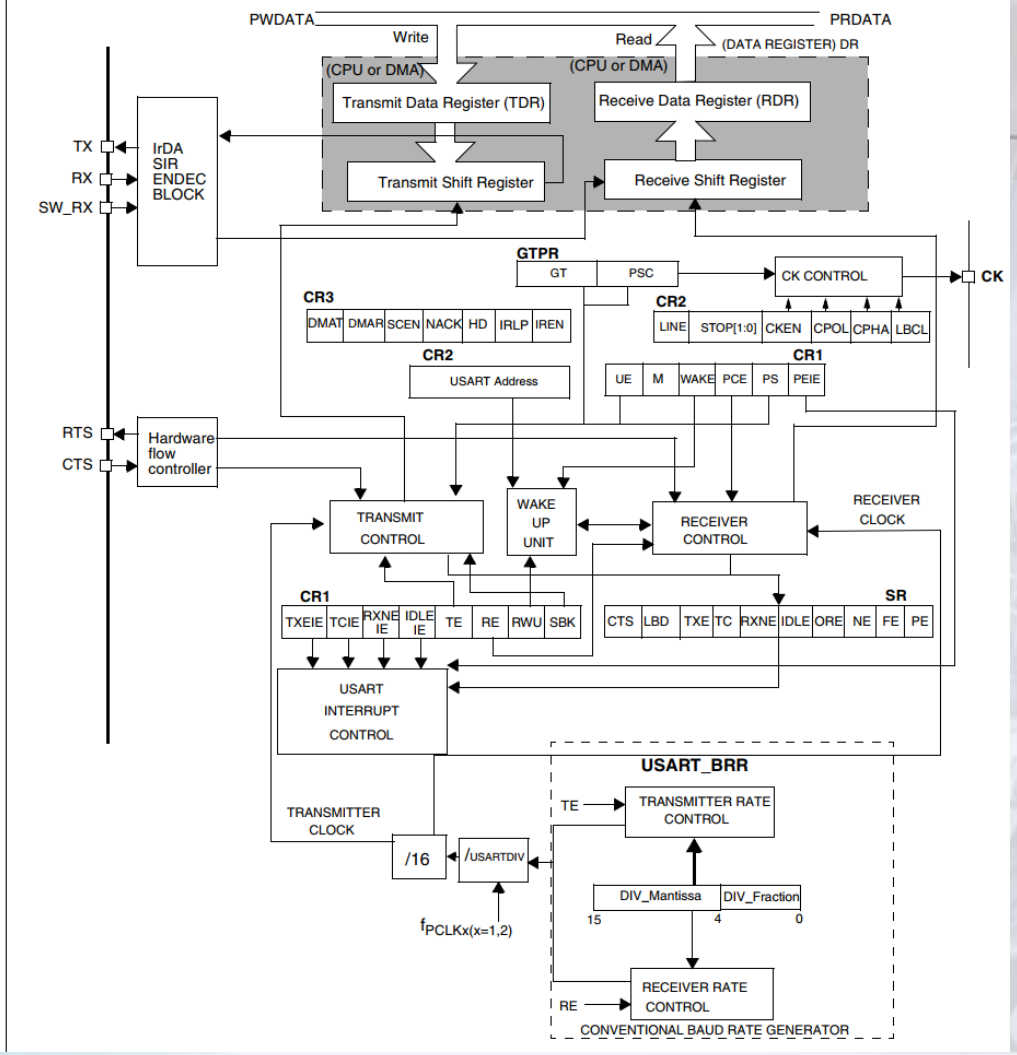
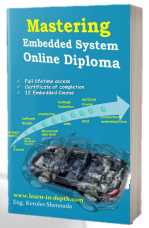


Figure 279. USART block diagram



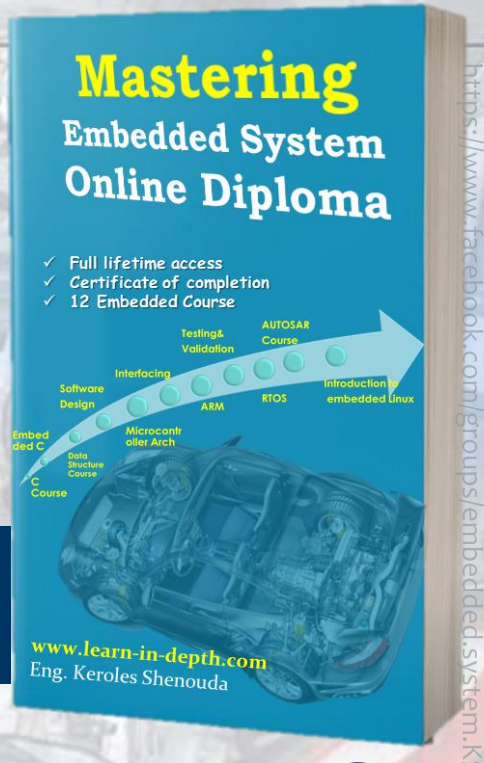
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



#LEARN_IN DEPTH

#Be_professional_in_embedded_system

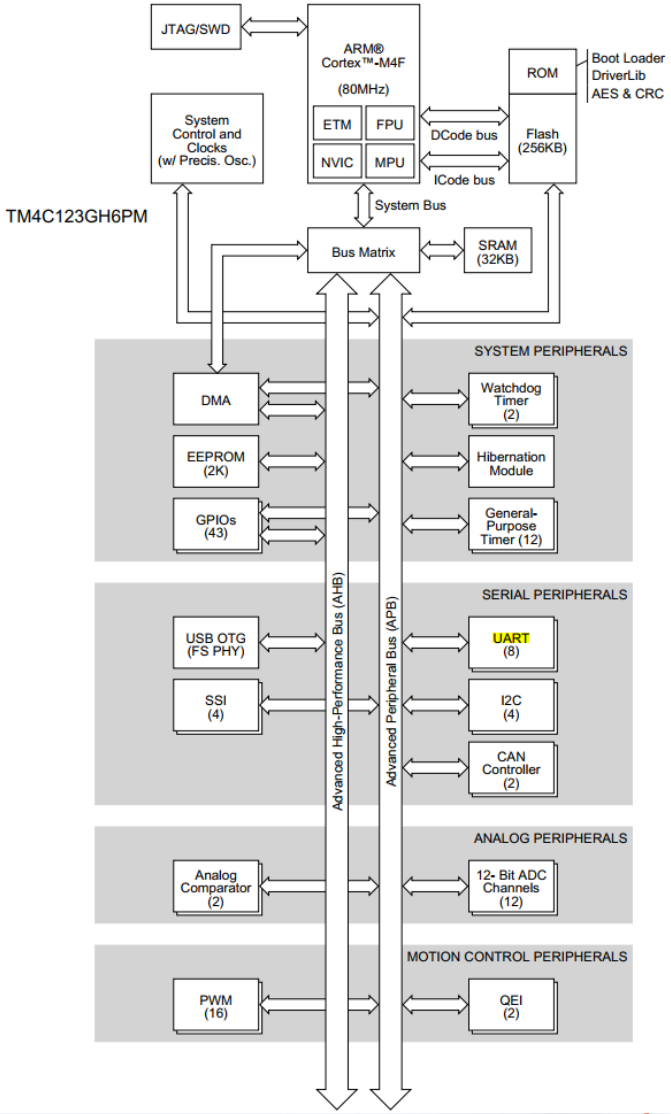
USART Block Diagram on TM4C123



LEARN-IN-DEPTH
Be professional in embedded system

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

Figure 1-1. Tiva™ TM4C123GH6PM Microcontroller High-Level Block Diagram



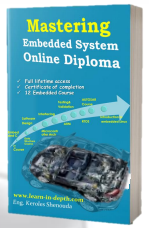
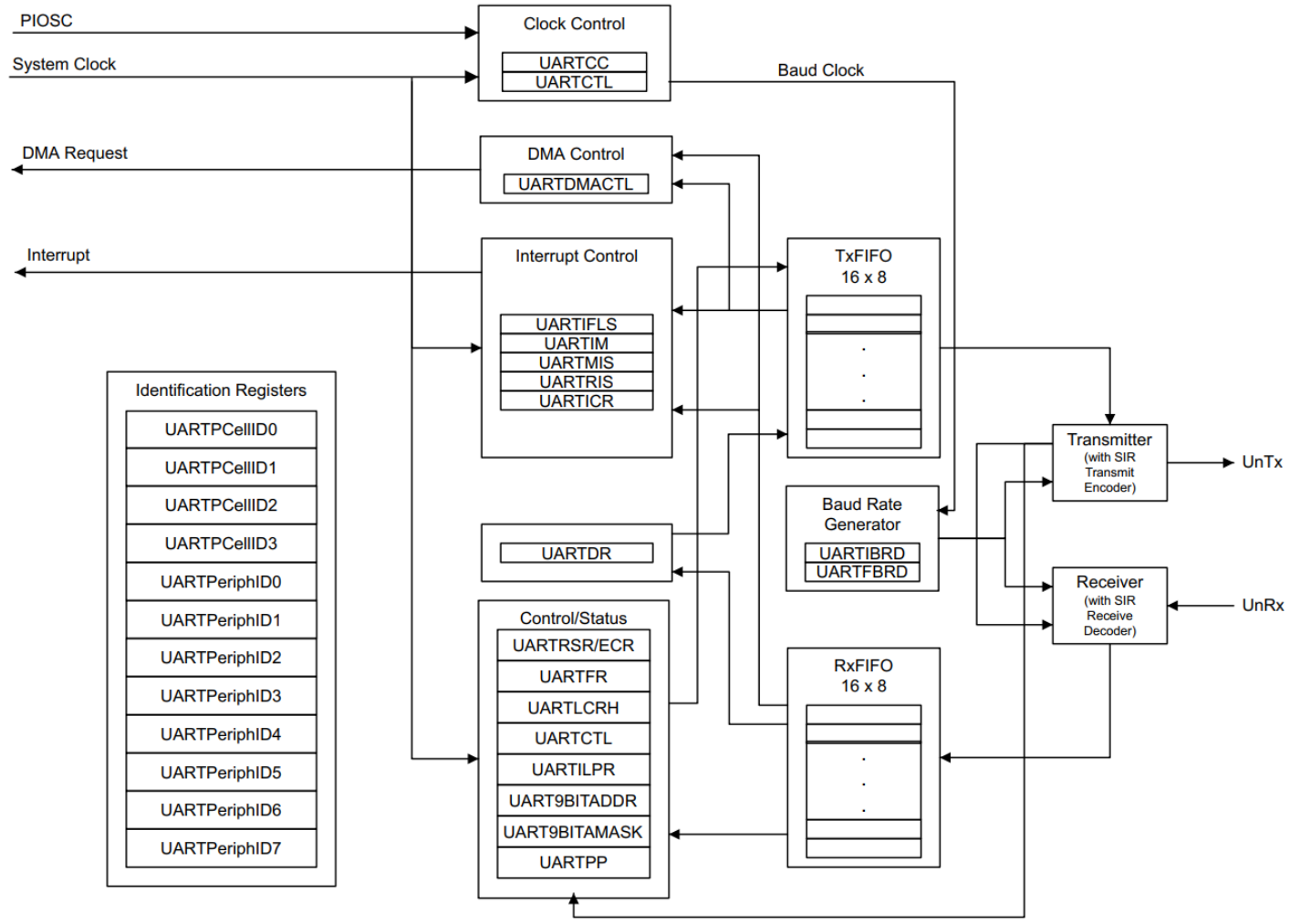


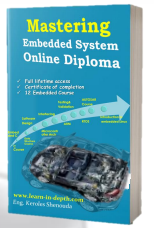
Table 2-4. Memory Map *(continued)*

Start	End	Description	For details, see page ...
0x4000.6000	0x4000.6FFF	GPIO Port C	658
0x4000.7000	0x4000.7FFF	GPIO Port D	658
0x4000.8000	0x4000.8FFF	SSI0	967
0x4000.9000	0x4000.9FFF	SSI1	967
0x4000.A000	0x4000.AFFF	SSI2	967
0x4000.B000	0x4000.BFFF	SSI3	967
0x4000.C000	0x4000.CFFF	UART0	903
0x4000.D000	0x4000.DFFF	UART1	903
0x4000.E000	0x4000.EFFF	UART2	903
0x4000.F000	0x4000.FFFF	UART3	903
0x4001.0000	0x4001.0FFF	UART4	903
0x4001.1000	0x4001.1FFF	UART5	903
0x4001.2000	0x4001.2FFF	UART6	903
0x4001.3000	0x4001.3FFF	UART7	903

14.1 Block Diagram

Figure 14-1. UART Module Block Diagram





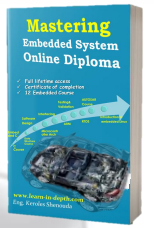
#LEARN_IN DEPTH

#Be_professional_in embedded_system

References

- https://docs.google.com/viewer?a=v&pid=sites&srcid=ZmtlLnV0bS5teXxyaWR6dWFuLXMtd2Vic2l0ZXxneDo2ODU0NzlkM2JkOTg4MjRk
- http://www.avrprojects.net/index.php/avr-projects/sensors/38-humidity-and-temperature-sensor-dht11?showall=&start=1
- http://www.cse.wustl.edu/~lu/cse467s/slides/dsp.pdf
- http://www.avr-tutorials.com/
- Microprocessor: ATmega32 (SEE3223-10) http://ridzuan.fke.utm.my/microprocessor-atmega32-see3223-10
- http://circuitdigest.com/article/what-is-the-difference-between-microprocessor-and-microcontroller
- http://cs4hs.cs.pub.ro/wiki/roboticsisfun/chapter2/ch2_7_programming_a_microcontroller
- Embedded Systems with ARM **Cortex-M Microcontrollers in** Assembly Language and C Dr. Yifeng Zhu Third edition June 2018

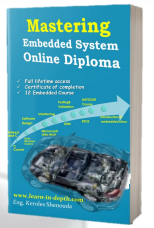
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



References

- http://techdifferences.com/difference-between-interrupt-and-polling-in-os.html
- http://www.bogotobogo.com/Embedded/hardware_interrupt_software_interrupt_latency_irq_vs_fiq.php
- Preventing Interrupt Overload Presented by Jiyong Park Seoul National University, Korea 2005. 2. 22. John Regehr, Usit Duogsaa, School of Computing, University.
- First Steps Embedded Systems Byte Craft Limited reference
- COMPUTER ORGANIZATION AND ARCHITECTURE DESIGNING FOR PERFORMANCE EIGHTH EDITION **William Stallings**
- **Getting Started with the Tiva™ TM4C123G LaunchPad Workshop**

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



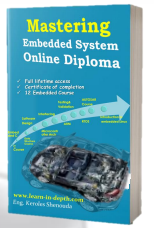
#LEARN_IN DEPTH

#Be_professional_in_embedded_system

References

- Tiva™ TM4C123GH6PM Microcontroller DATA SHEET
- Interrupts and Exceptions COMS W6998 Spring 2010
- THE **AVR MICROCONTROLLER**. AND EMBEDDED SYSTEMS Using Assembly and C. Muhammad Ali **Mazidi**.

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



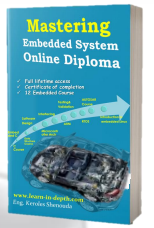
#LEARN_IN DEPTH

#Be_professional_in embedded_system

References

- https://docs.google.com/viewer?a=v&pid=sites&srcid=ZmtlLnV0bS5teXxyaWR6dWFuLXMtd2Vic2l0ZXxneDo2ODU0NzlkM2JkOTg4MjRk
- http://www.avrprojects.net/index.php/avr-projects/sensors/38-humidity-and-temperature-sensor-dht11?showall=&start=1
- http://www.cse.wustl.edu/~lu/cse467s/slides/dsp.pdf
- http://www.avr-tutorials.com/
- Microprocessor: ATmega32 (SEE3223-10) http://ridzuan.fke.utm.my/microprocessor-atmega32-see3223-10
- http://circuitdigest.com/article/what-is-the-difference-between-microprocessor-and-microcontroller
- AVR Microcontroller and Embedded Systems: Using Assembly and C (Pearson Custom Electronics Technology) 1st Edition https://www.amazon.com/AVR-Microcontroller-Embedded-Systems-Electronics/dp/0138003319

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



#LEARN_IN DEPTH

#Be_professional_in_embedded_system

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/