بسم الله الرحمن الرحيم

In this section we will develop a simple device driver using the Universal Asynchronous Receiver/Transmitter (UART) --> communication protocol .

This serial port allows the microcontroller to communicate with devices such as other computers, printers, input sensors, and LCDs. Serial transmission involves sending one bit at a time, such that the data is spread out over time.

The total number of bits transmitted per second is called the baud rate.
The reciprocal of the baud rate is the bit time, which is the time to send one bit.

1. USB --> use to send big size files , USB3's speed = 5G

2. UART --> used to send characters to be printed through printers and as we know each char. represents by ASCI code

Each UART will have a baud rate control register, which we use to select the transmission rate.

A frame is the smallest complete unit of serial transmission. Figure 8.3 plots the signal versus time on a serial port, showing a single frame, which includes a start bit (which is 0), 8 bits of data (least significant bit first), and a stop bit (which is 1). There is always only one start bit, but the Stellaris ® and Tiva ® UARTs allow us to select the 5 to 8 data bits and 1 or 2 stop bits.

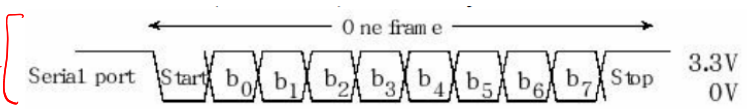


Figure 8.3. A serial data frame with 8-bit data, 1 start bit, 1 stop bit, and no parity bit.

The UART can add even , odd, or no parity bit.

we will know later to avoid noise can happen between 0,3.3 we will use cicuit interface including amplifier to make 0 --> 5v , 1 --> -5 ,hence noise can't be 11.

we will employ the typical protocol of 1 start bit, 8 data bits, no parity, and 1 stop bit. This protocol is used for both transmitting and receiving. for this lect.

The information rate, or bandwidth, is defined as the amount of data or useful information transmitted per second. From Figure 8.3, we see that 10 bits are sent for every byte of usual data. Therefore, the bandwidth of the serial channel (in bytes/second) is the baud rate (in bits/sec) divided by 10.

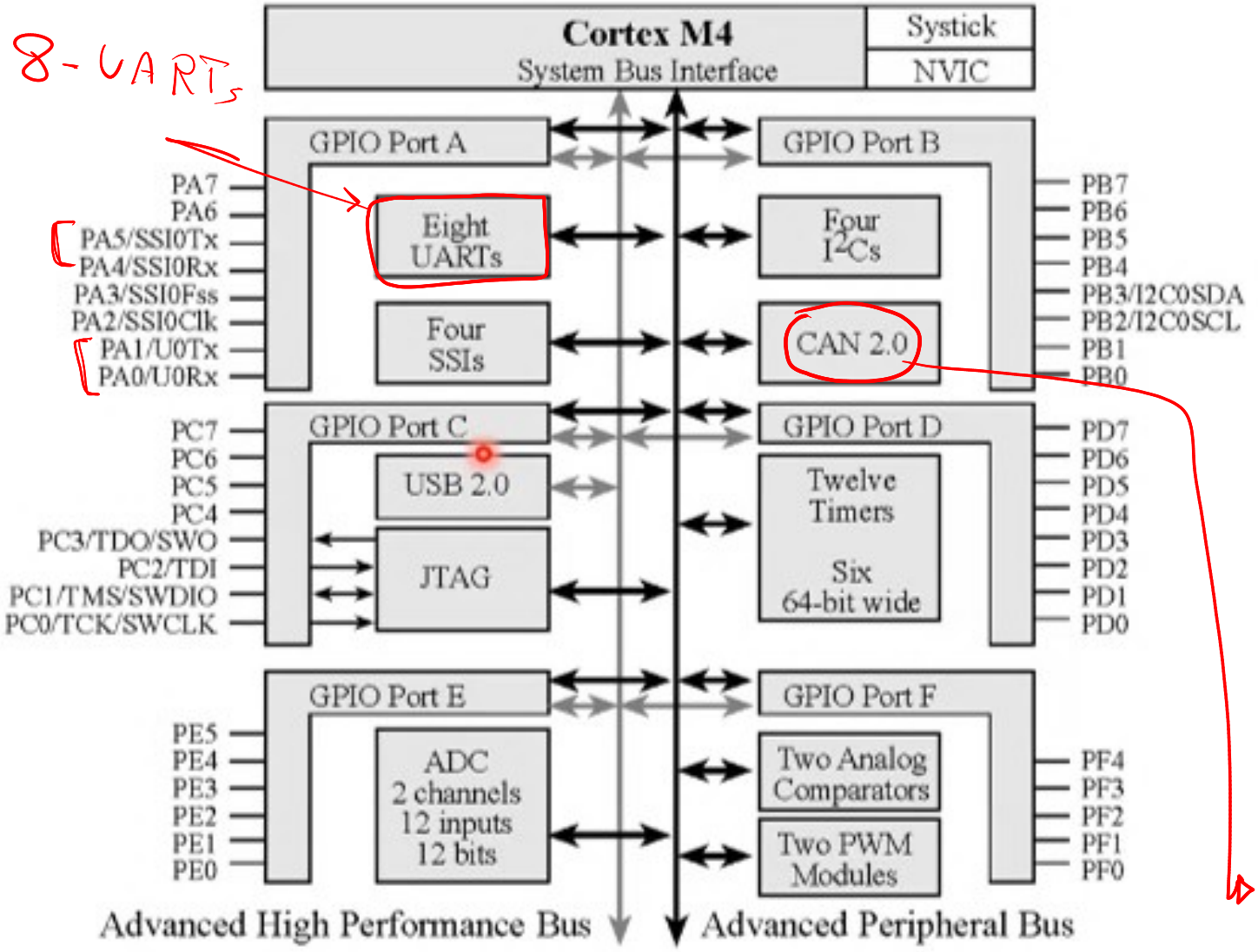
Common Error: If you change the bus clock frequency without changing the baud rate register, the UART will operate at an incorrect baud rate.

Table 8.2 shows the three most commonly used RS232 signals. The RS232 standard uses a DB25 connector that has 25 pins. The EIA-574 standard uses RS232 voltage levels and a DB9 connector that has only 9 pins. The most commonly used signals of the full RS232 standard are available with the EIA-574 protocols. Only **TxD**, **RxD**, and **SG** are required to implement a simple bidirectional serial channel, thus the other signals are not shown (Figure 8.4). We define the **data terminal equipment** (DTE) as the computer or a terminal and the **data communication equipment** (DCE) as the modem or printer.

DB25 Pin	RS232 Name	DB9 Pin	EIA-574 Name	Signal	Description	True	DTE	DCE
2	BA	3	103	TxD	Transmit Data	-5.5V	out	in
3	BB	2	104	RxD	Receive Data	-5.5V	in	out
7	AB	5	102	SG	Signal Ground			

Texas Instruments TM4C123

8-UARTs



ARM Cortex-M4
+ 256K Flash
+ 32K RAM
+ JTAG
+ 43 Ports
+ SysTick
+ ADC
+ UART

→ used in Cars

PA0 [1-bit] → used for RX

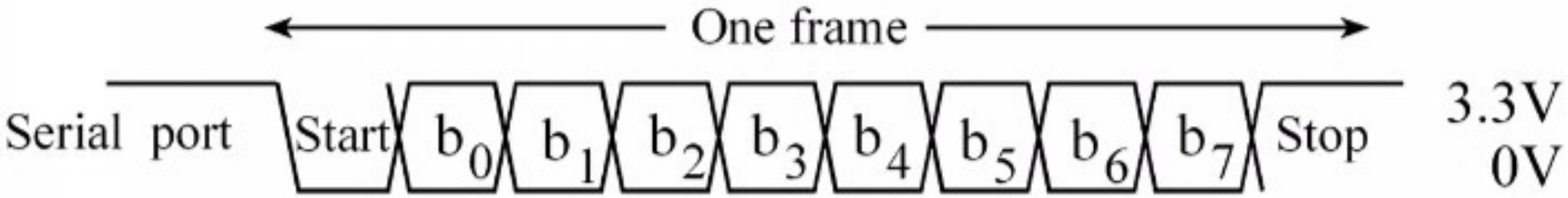
PA1 " → " " TX

SysTick Timer

```
;------------SysTick_Wait------------
; Time delay using busy wait.
; Input: R0  delay parameter in units of the core clock
;        80 MHz(12.5 nsec each tick)
; Output: none
; Modifies: R1
SysTick_Wait
    SUB  R0, R0, #1    ; delay-1
    LDR  R1, =NVIC_ST_RELOAD_R
    STR  R0, [R1]      ; time to wait
    LDR  R1, =NVIC_ST_CURRENT_R
    STR  R0, [R1]      ; any value written to CURRENT clears
    LDR  R1, =NVIC_ST_CTRL_R
SysTick_Wait_loop
    LDR  R0, [R1]      ; read status
    ANDS R0, R0, #0x00010000 ; bit 16 is COUNT flag
    BEQ  SysTick_Wait_loop   ; repeat until flag set
    BX   LR
```

Universal Asynchronous Receiver/Transmitter (UART)

❑ UART (Serial Port) Interface



- ❖ Send/receive a *frame* of (5-8) data bits with a single (start) bit prefix and a 1 or 2 (stop) bit suffix
- ❖ Baud rate is total number of bits per unit time
  - o Baudrate = 1 / bit-time
- ❖ Bandwidth is data per unit time
  - o Bandwidth = (data-bits / frame-bits) * baudrate

time one bit takes to transmi. or recieved

baudrate → bit/time

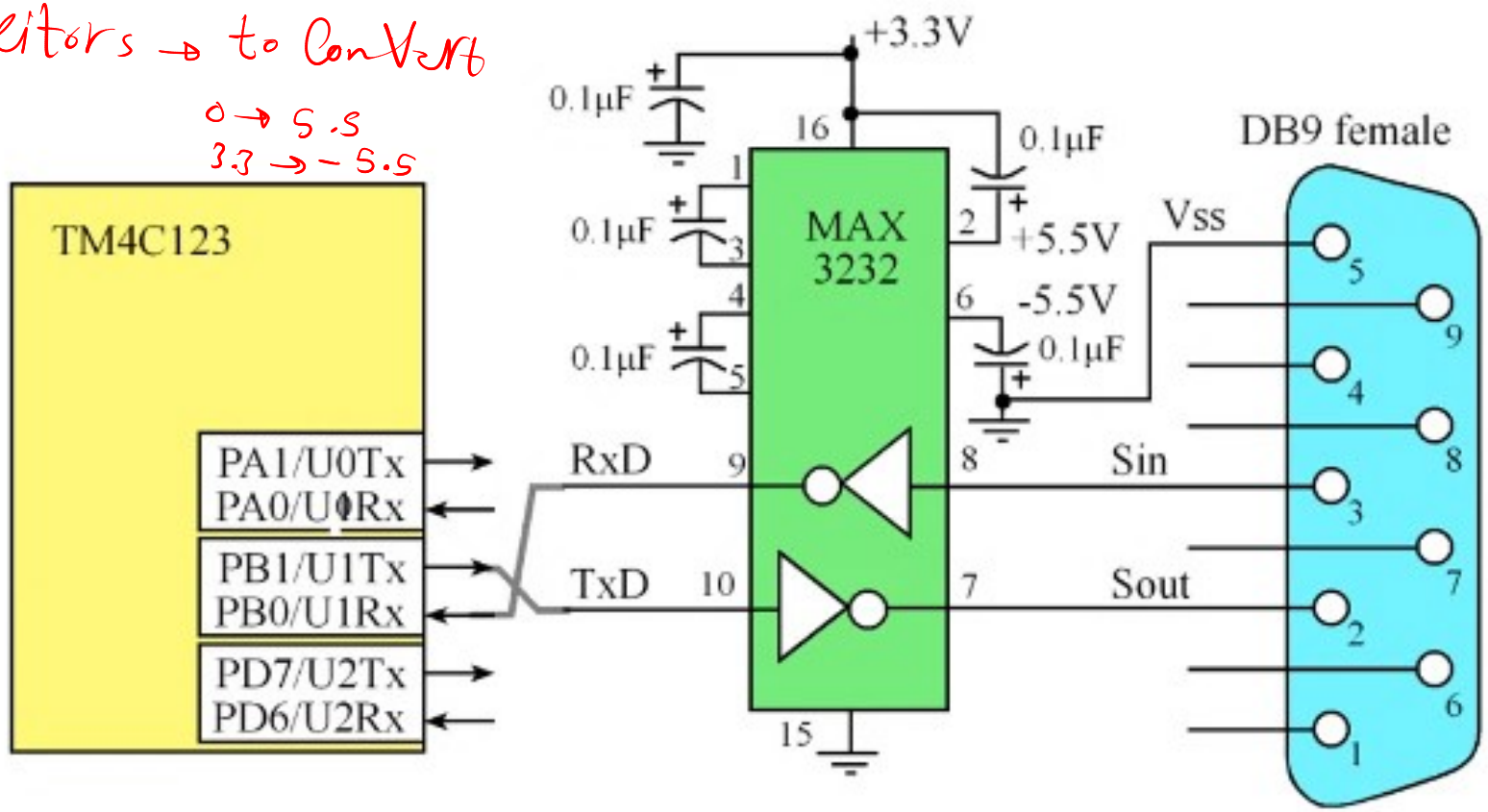
frame frame frame

10-bit 10-bit 10-bit

data-bits

RS-232 Serial Port

Capacitors → to convert
0 → 5.5
3.3 → −5.5



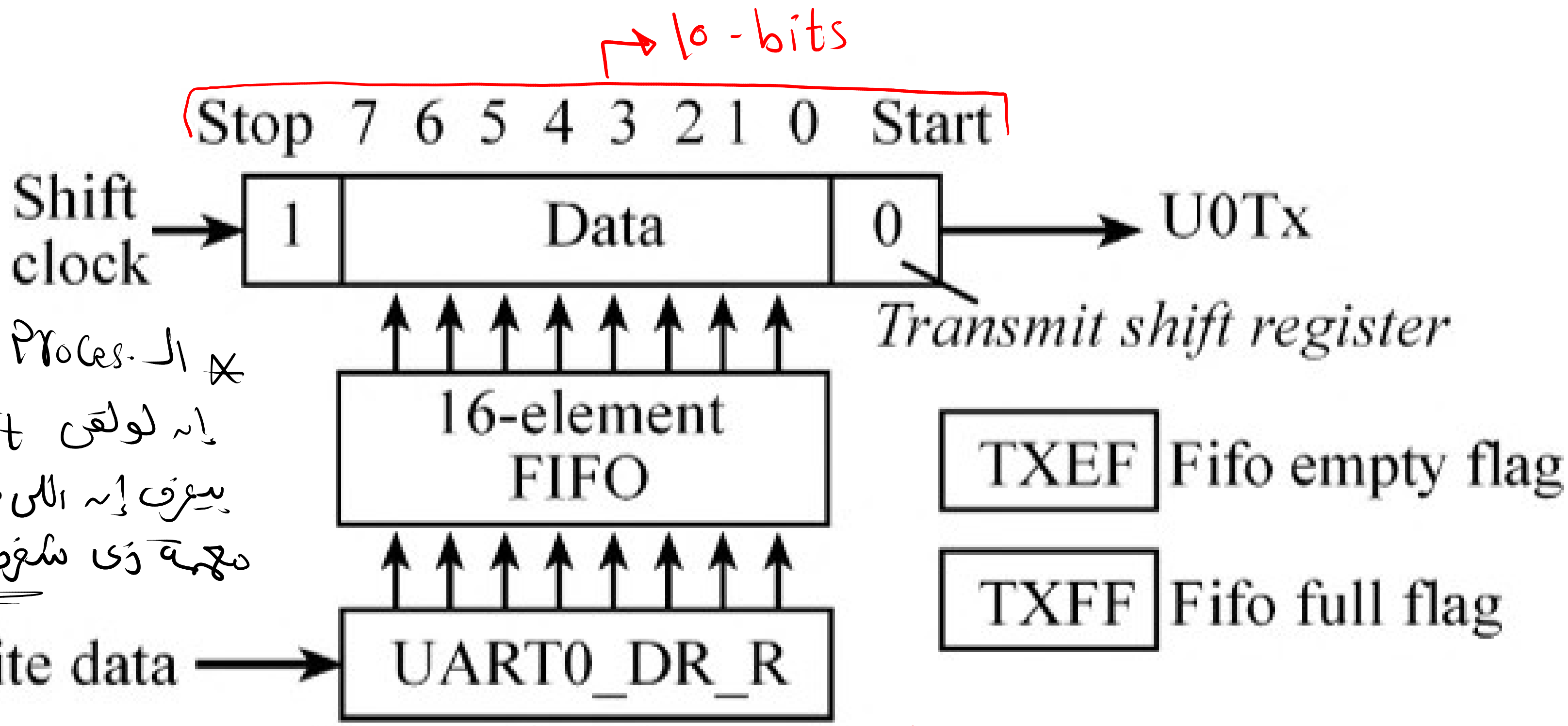
DB25 Pin	RS232 Name	DB9 Pin	EIA-574 Name	Signal	Description	True	DTE	DCE
2	BA	3	103	TxD	Transmit Data	-5.5V	out	in
3	BB	2	104	RxD	Receive Data	-5.5V	in	out
7	AB	5	102	SG	Signal Ground			

ده شرح السلايدز اللى فوق

RS232 is a non-return-to-zero (NRZ) protocol with true signified as a voltage between -5 and -15 V. False is signified by a voltage between +5 and +15 V. A MAX3232 converter chip is used to translate between the +5.5/-5.5 V RS232 levels and the 0/+3.3 V digital levels. The capacitors in this circuit are important, because they form a charge pump used to create the ±5.5 voltages from the +3.3 V supply. The RS232 timing is generated automatically by the UART. During transmission, the Maxim chip translates a digital high on microcontroller side to -5.5V on the RS232/EIA-574 cable, and a digital low is translated to +5.5V. During receiving, the Maxim chip translates negative voltages on RS232/EIA-574 cable to a digital high on the microcontroller side, and a positive voltage is translated to a digital low. The computer is classified as DTE, so its serial output is pin 3 in the EIA-574 cable, and its serial input is pin 2 in the EIA-574 cable. When connecting a DTE to another DTE, we use a cable with pins 2 and 3 crossed. I.e., pin 2 on one DTE is connected to pin 3 on the other DTE and pin 3 on one DTE is connected to pin 2 on the other DTE. When connecting a DTE to a DCE, then the cable passes the signals straight across. In all situations, the grounds are connected together using the SG wire in the cable. This channel is classified as **full-duplex**, because transmission can occur in both directions simultaneously.

We will begin with transmission, because it is simple. The transmitter portion of the UART includes a data output pin, with digital logic levels as drawn in Figure 8.5. The transmitter has a 16-element FIFO and a 10-bit shift register, which cannot be directly accessed by the programmer (Figure 8.5). The FIFO and shift register in the transmitter are separate from the FIFO and shift register associated with the receiver. To output data using the UART, the software will first check to make sure the transmit FIFO is not full (it will wait if **TXFF**is 1) and then write to the transmit data register (e.g., **UART0_DR_R**). The bits are shifted out in this order: start, $\mathbf{b_0}$, $\mathbf{b_1}$, $\mathbf{b_2}$, $\mathbf{b_3}$, $\mathbf{b_4}$, $\mathbf{b_5}$, $\mathbf{b_6}$, $\mathbf{b_7}$, and then stop, where $\mathbf{b_0}$ is the LSB and $\mathbf{b_7}$ is the MSB. The transmit data register is write only, which means the software can write to it (to start a new transmission) but cannot read from it. Even though the transmit data register is at the same address as the receive data register, the transmit and receive data registers are two separate registers.

UART0_DR_R will not send to the FIFO if the TXFF flag = 1 → * بيقولك إنه أنا دلوقتى عندى الـFIFO ده ممكن يخزن 16-bit اللى هما يعنى حرفين فا أنا لو دخلت بالـsequence اللى هو T ← e ← أنا كده مليت الـFIFO والـTXFF هيبقى بـ "1" ومش هيدخل الحرف الـ "s" اللى هو بعد "e" هيستنى لحد ما الـFIFO تفضى وساعتها TXFF فلاج؟ = 0



* الـ Proces. بيترجم إنه لو لقى Start بـ "0" يعرف إنه اللى بعده معلومات مهمة زى شوية رسايل كده

Transmit data register → Can hold only 8-bit and only write by user

32-bit-R

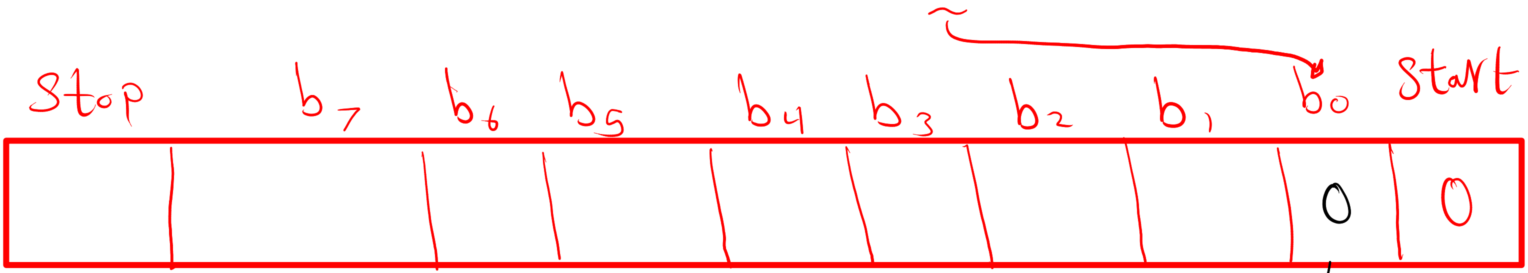
(8 * 4) bits

R0 = "Test"

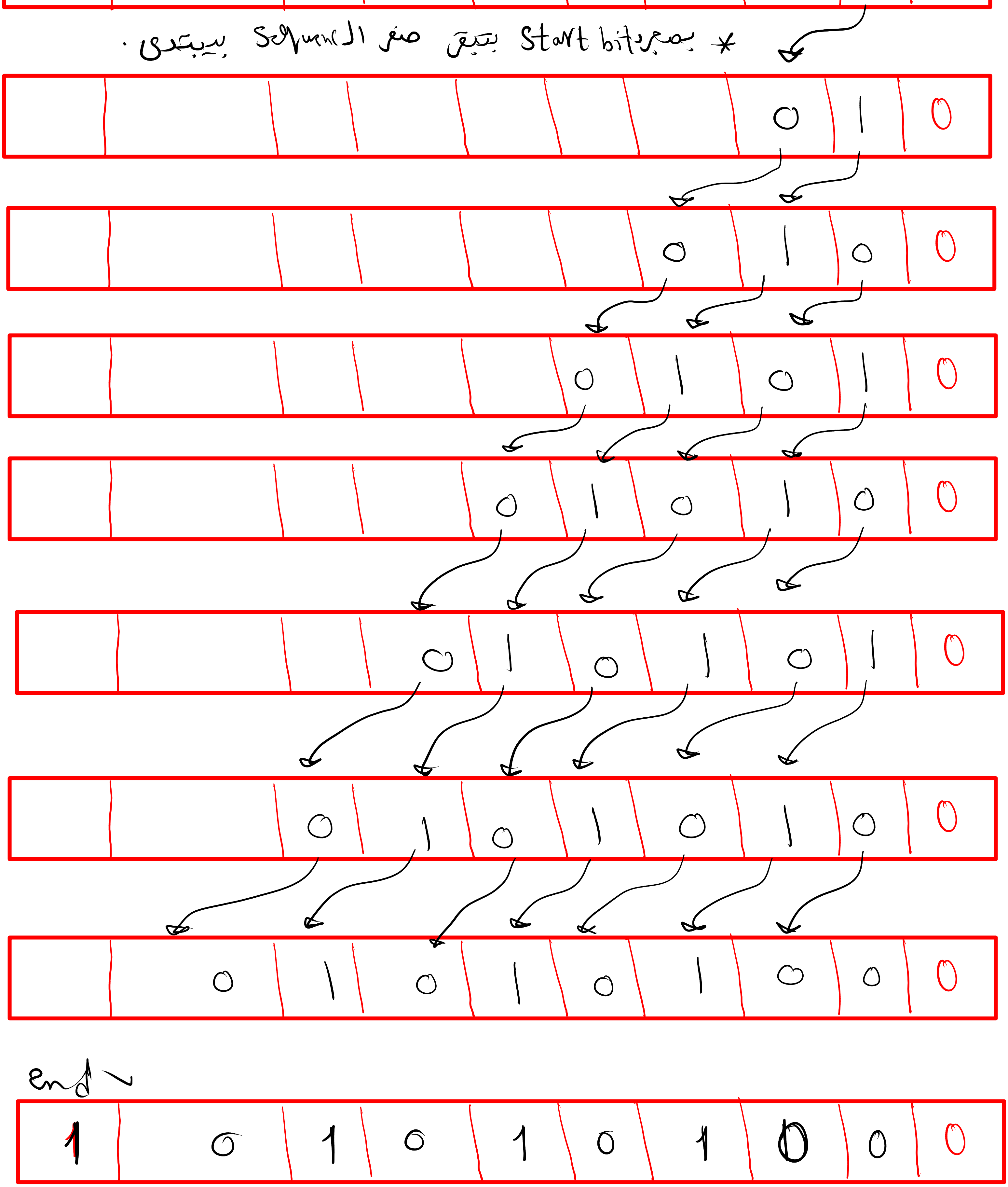
Char [as we know] = 1-byte = 8-bits

بالمختصر كده لو أنت عاوز تطبع كلمة إنت كتبتها وليكن مثلا "Test" فا انت بتدخل كل حرف بنظام sequence كده يعنى (T → e → s → t)

if we transmiter "T" = 0X54 = 01010100



* بمجرد start bit بتبقى صفر الـ sequence بيبتدي .



* بمجرد ما stop bit تبقى بواحد الـ sequence بيخلص