COL215P Digital Logic and System Design

Serial Asynchronous Data Transfer

15 May 2022

Parallel vs Serial Data Transfer

- Parallel
  - multiple wires carry bits in parallel : 8-bit, 16-bit, 32-bit, 64-bit . . .
  - address, read data, write data may use same wires or separate wires
- Serial
  - only one bit at a time
  - read data, write data may use same wires or separate wires, no separate address wires
  - less expensive than parallel transfer

Examples of Series/parallel interface standards

Parallel

- ISA (Industry Standard Architecture)
- EISA (Extended ISA)
- VLB (VESA (Video Electronics Standards Association) Local Bus)
- PCI (Peripheral Component Interconnect)
- PATA (Parallel Advanced Technology Attachment)
- AGP (Accelerated Graphics Port)

Serial

- RS232
- USB (Universal Serial Bus)
- Fire wire
- Fibre channel
- PCIe (Peripheral Component Interconnect express)
- SATA (Serial Advanced Technology Attachment)

Is parallel transfer faster than serial?

- Transferring multiple bits in parallel means higher throughput

BUT

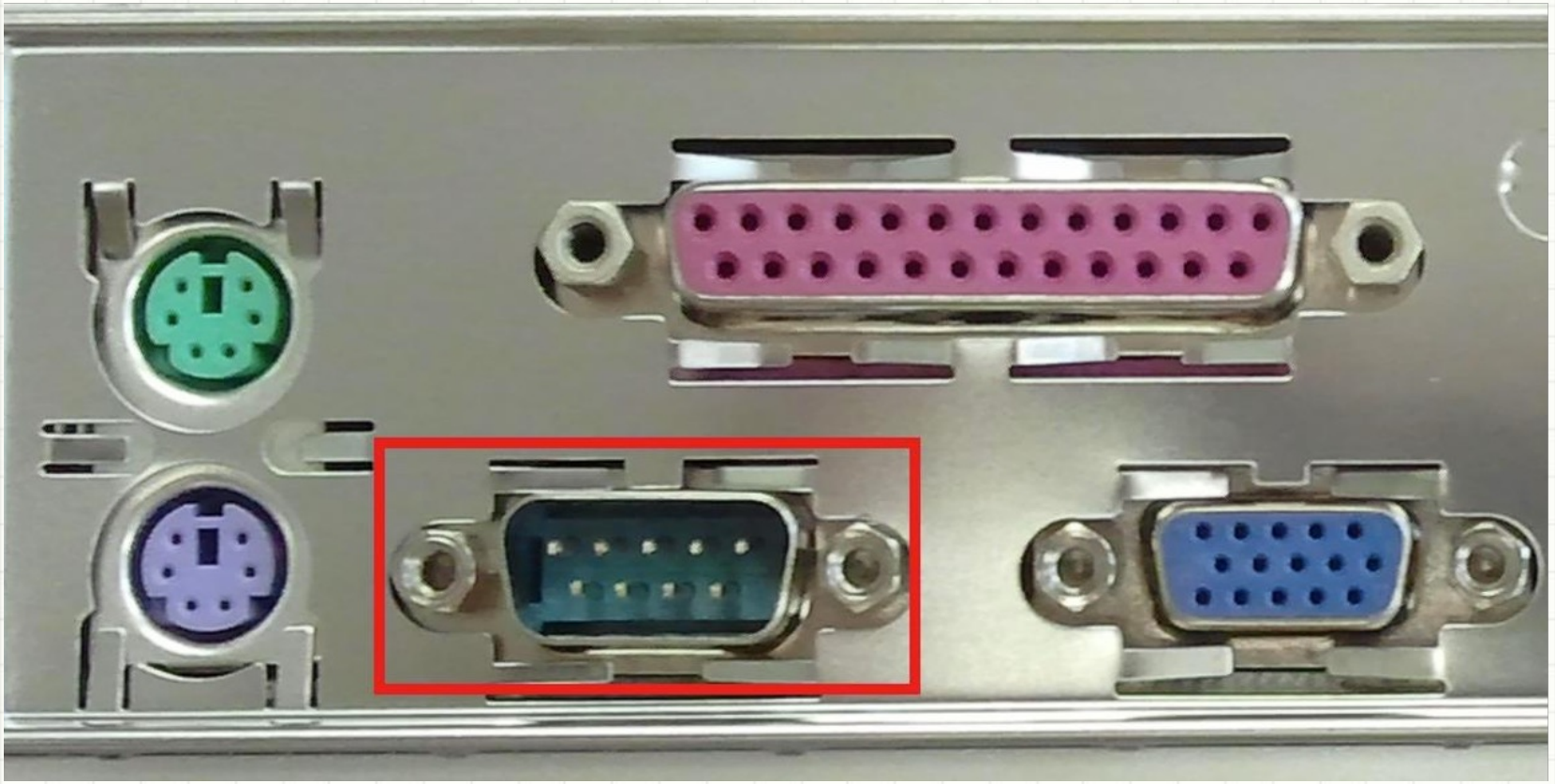
- At high speed, keeping all the bits synchronized in parallel paths is very difficult
- Longer the signals travel, more skew is likely among the bits of parallel paths

=> parallel is faster over smaller distances (mm, cm), serial is faster over longer distances (m)

Timing in serial transfer

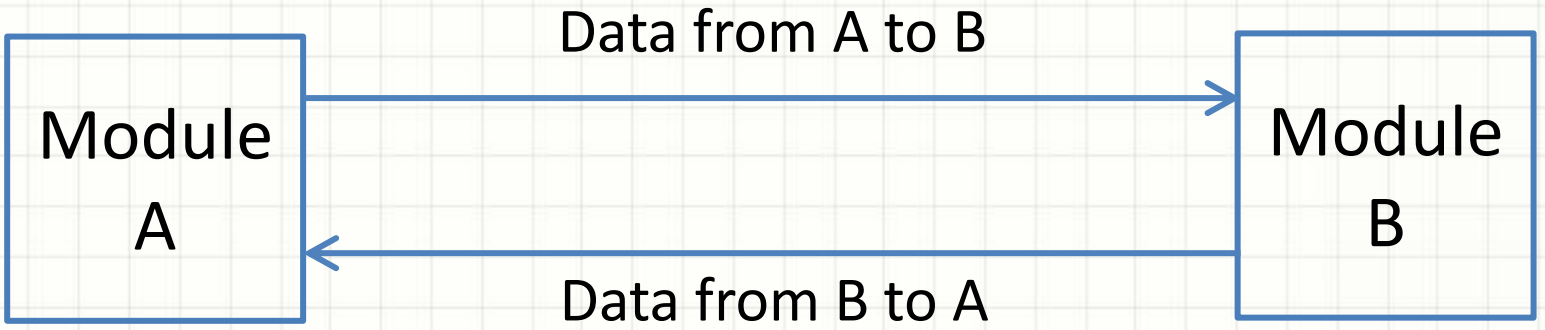
- Synchronous
  - shared clock
  - same frequency and phase
- Mesochronous
  - transfer clock signal, separately or with data
  - same frequency but not phase
- Plesiochronous
  - locally generated clock
  - nearly same frequency, changing phase

Serial port / COM port on PCs



RS232 has been a common standard
75 to 115.2 K bits/s

Serial Asynchronous Interface

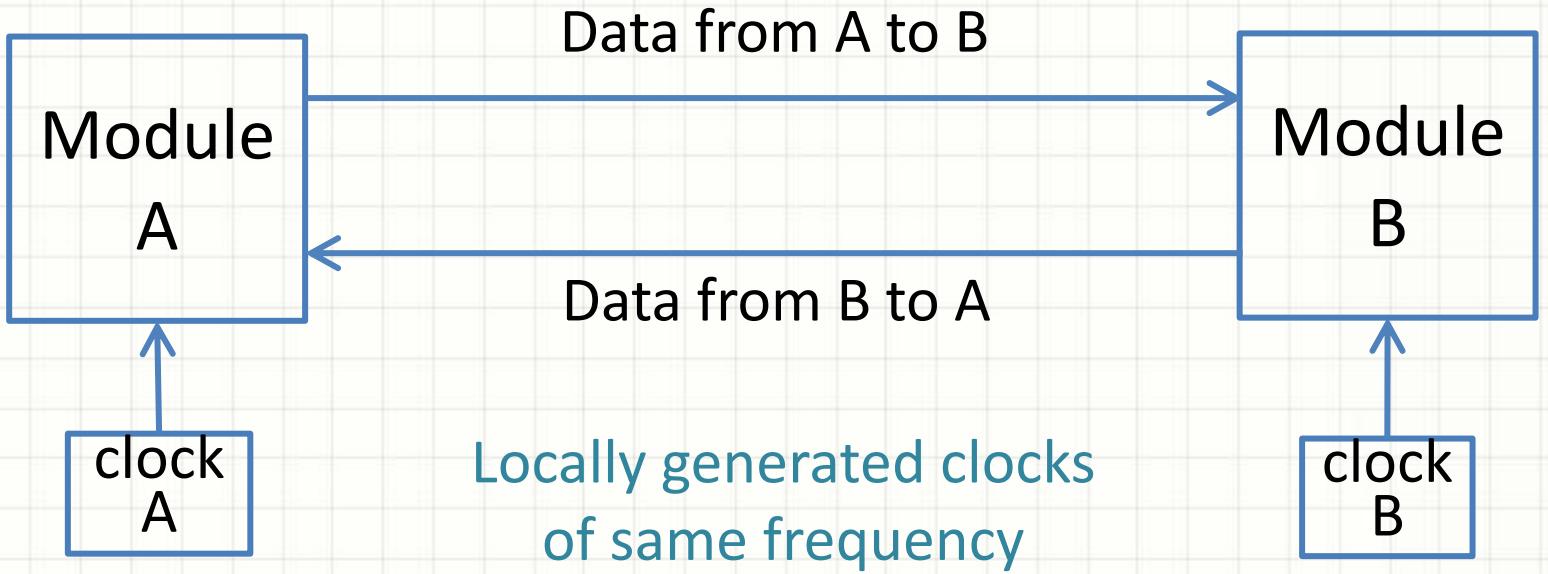


No shared clock

Shared data rate (baud rate)

Clocks for Asynchronous Interface

(Plesiochronous)



Locally generated clocks
of same frequency
(within some tolerance)
but different phases

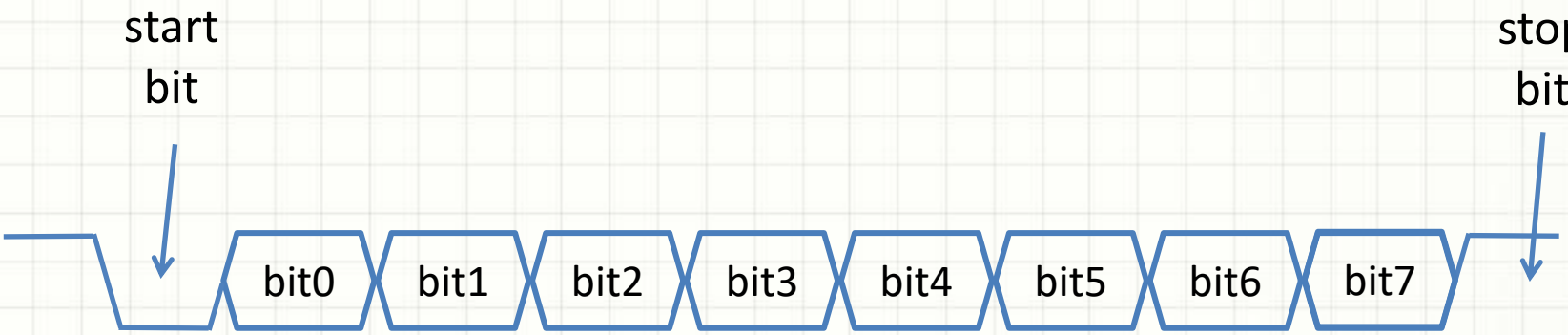
Synchronizing clock phases

- Always start sending data with a transition
- Signal is ‘1’ in idle condition
  (to distinguish it from a broken connection)
- The first bit is ‘0’
- ‘1’ to ‘0’ transition indicates start of Tx clock period to Rx
- This ‘0’ is called start bit, it is not a part of data

How is transition ensured at start?

- Arbitrary gap between two data units
- What if the last data bit is ‘0’ and there is no gap between two data units?
- Stop bit(s) = ‘1’ ensure a ‘1’ to ‘0’ transition for every data unit

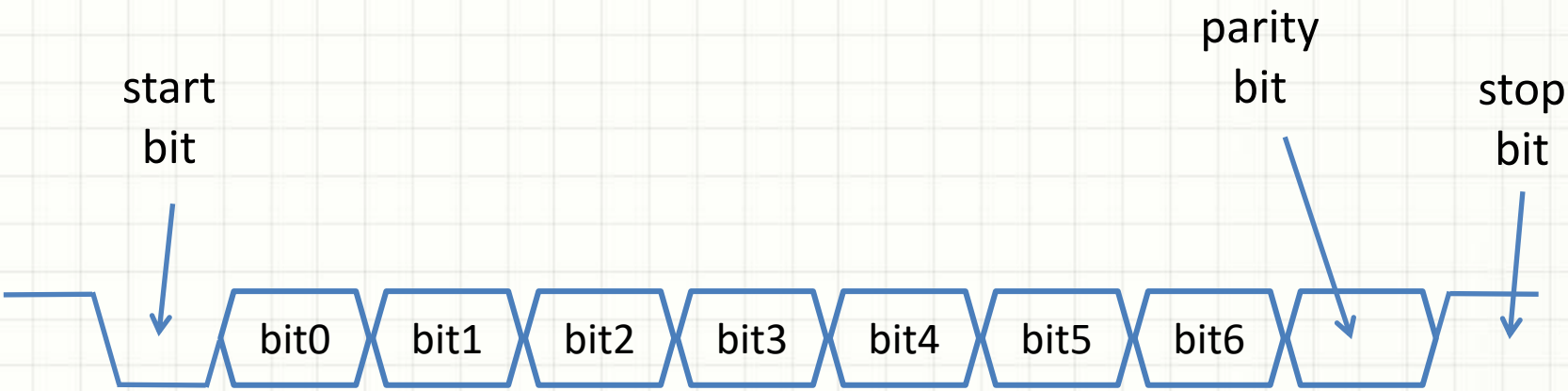
Data Framing Example



Variants

- Data bits : 5 to 8
- Parity : even / odd / none
- Start bits : 1
- Stop bits : 1 or 2
- Data rate : 300 / 600 / 1200 / 2400 / 4800 / 9600 . . . bits/sec (bauds)

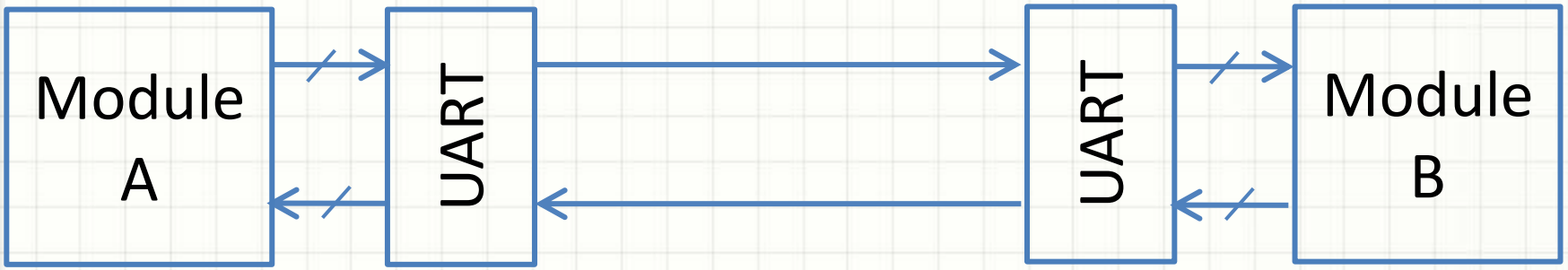
Example with parity bit



Serial receiver and transmitter

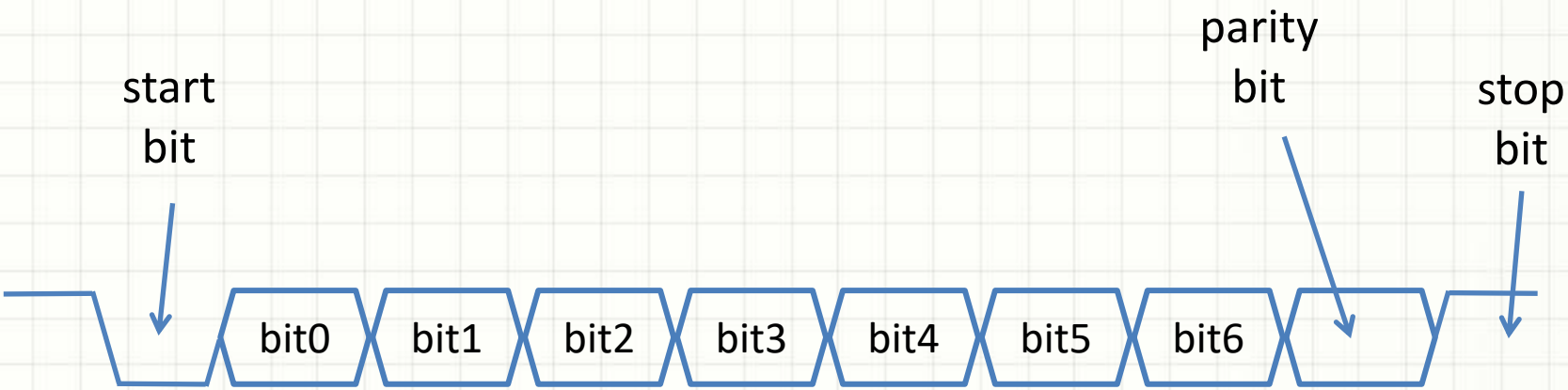
- Transmitter:
  - Parallel-in Serial-out register
  - Load data in parallel in a register
  - Shift out the bits serially
- Receiver:
  - Serial-in Parallel-out register
  - Shift the bits serially into a register
  - Read out data in parallel

UART
(Universal Asynchronous Receiver Transmitter)



"universal" => multiple variants are supported

Receiving Asynchronous Data



Wait for start bit
Collect data bits
Check parity bit
Check stop bit(s)

Problems in real systems

Presence of noise, distortion

- Spurious zeroes should not be interpreted as start bit
- Duration of data bits may appear as shorter or longer than normal

Solution

- Sample the input at a higher frequency – usually 8 or 16 times the baud rate
- Reject short 0's
- Look at the data bits in middle of the intervals.

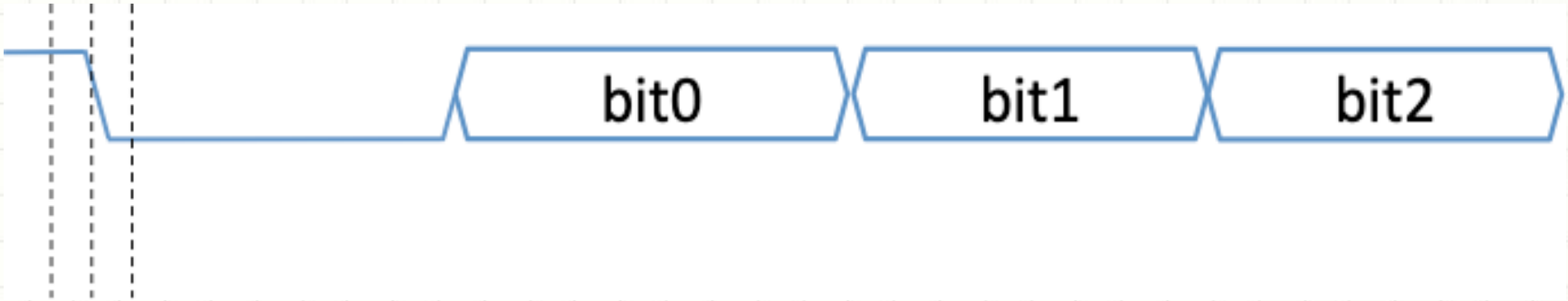
Wait for start bit

start
bit

1’s followed by a 0

Sample at fast rate for accuracy

start
bit

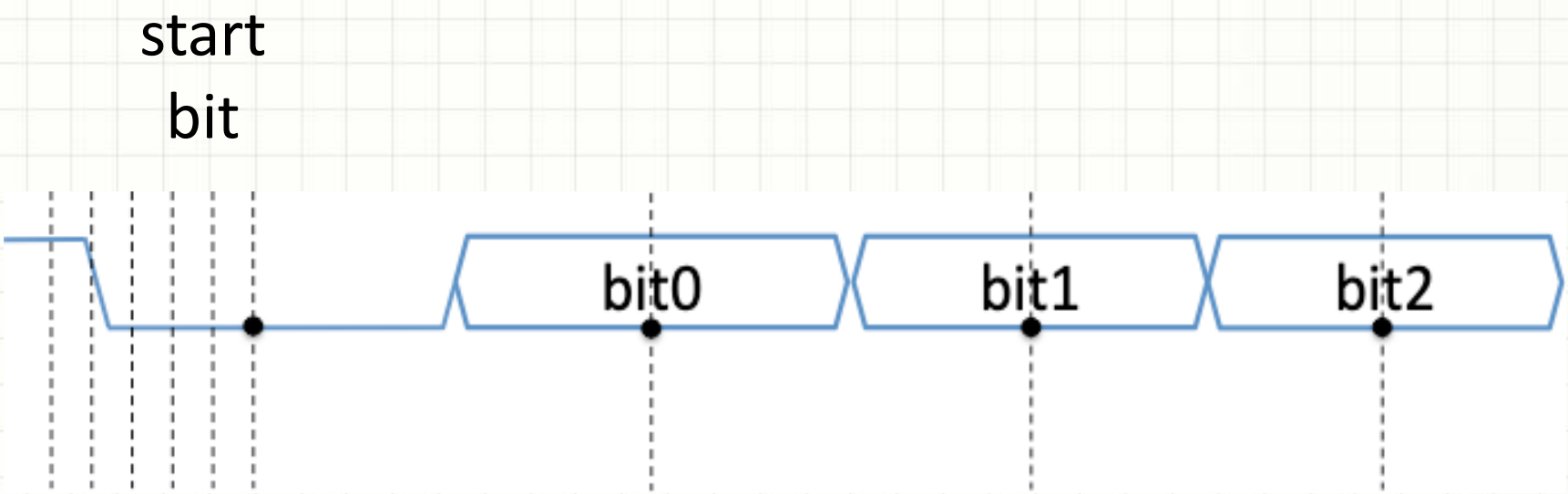


Receiver clock is 8 or 16 times baud rate

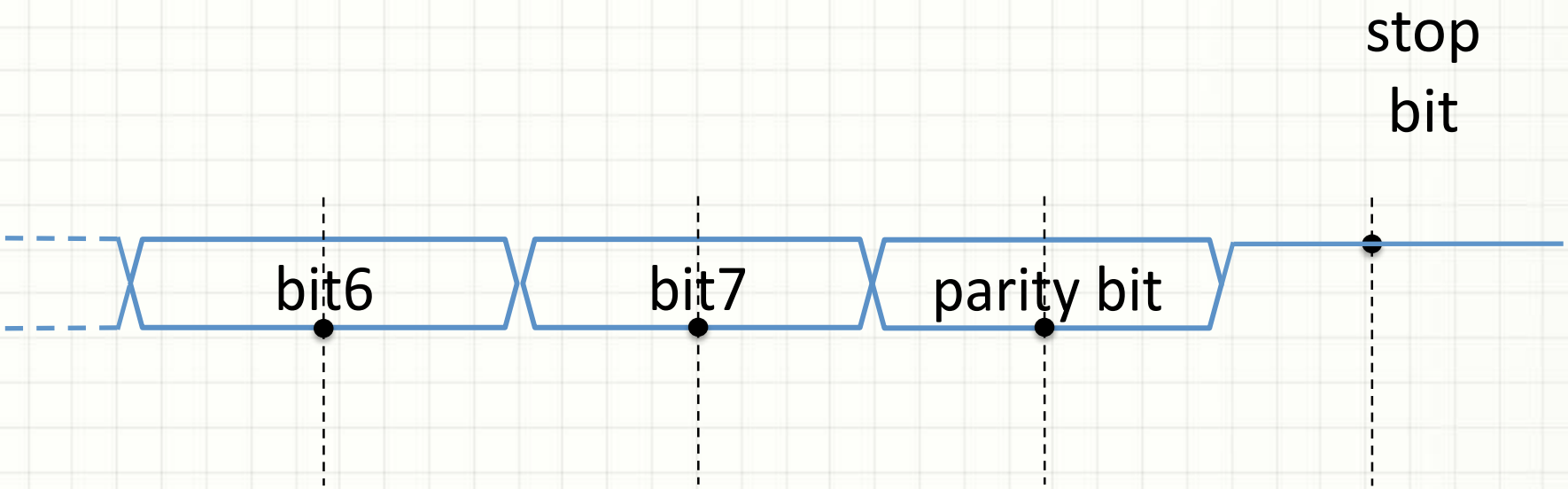
Reach middle of the start bit

With 16x clock, ensure 8 consecutive 0's in order to ignore the spikes

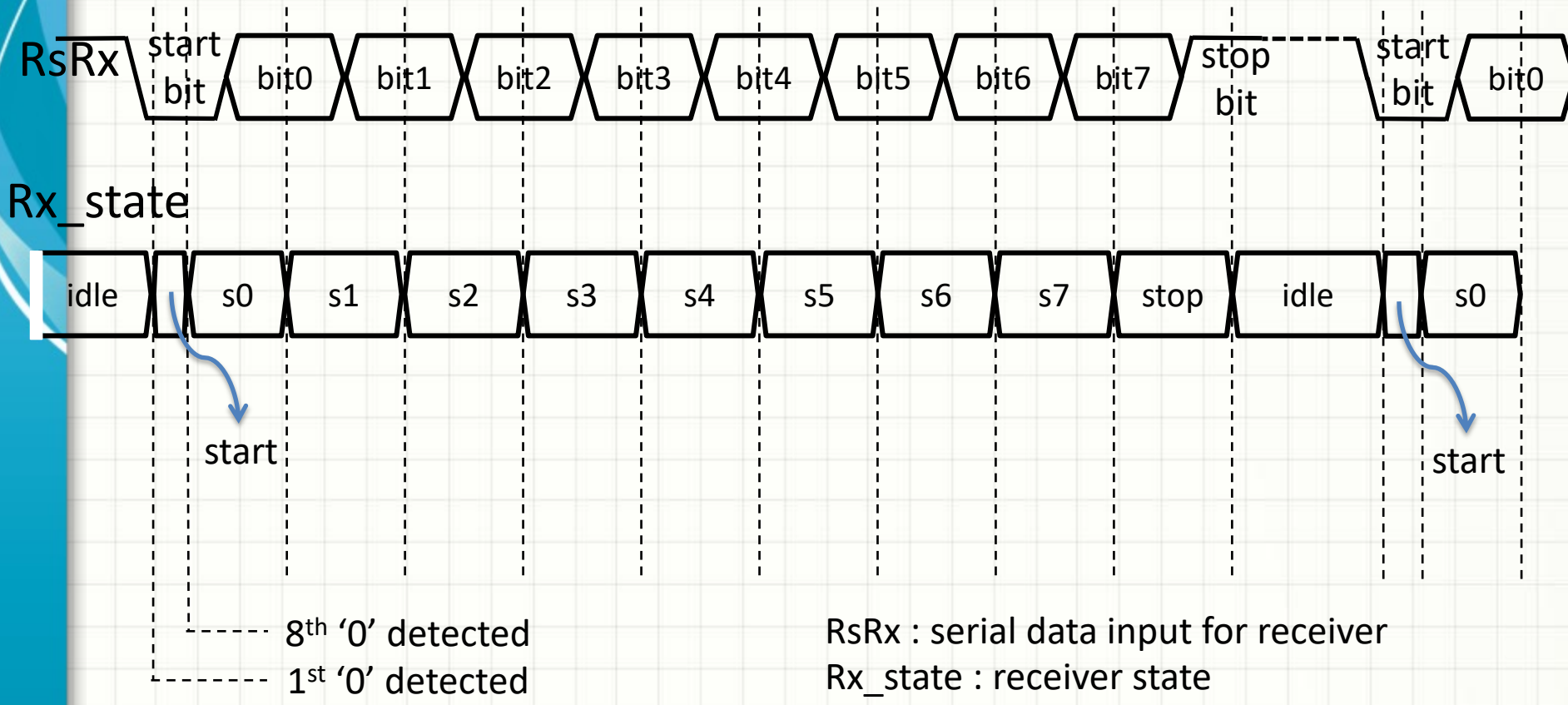
Sample subsequent bits in middle



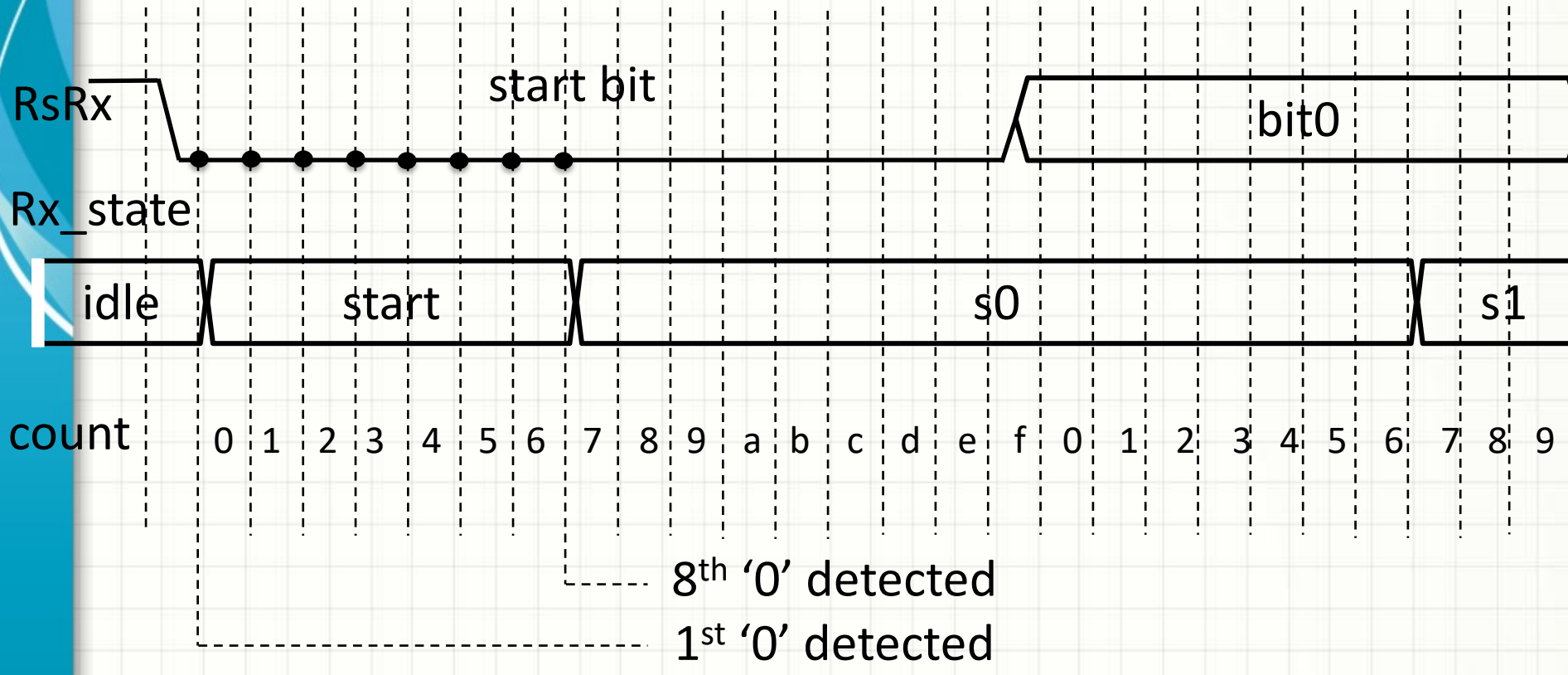
Check parity and stop bit(s)



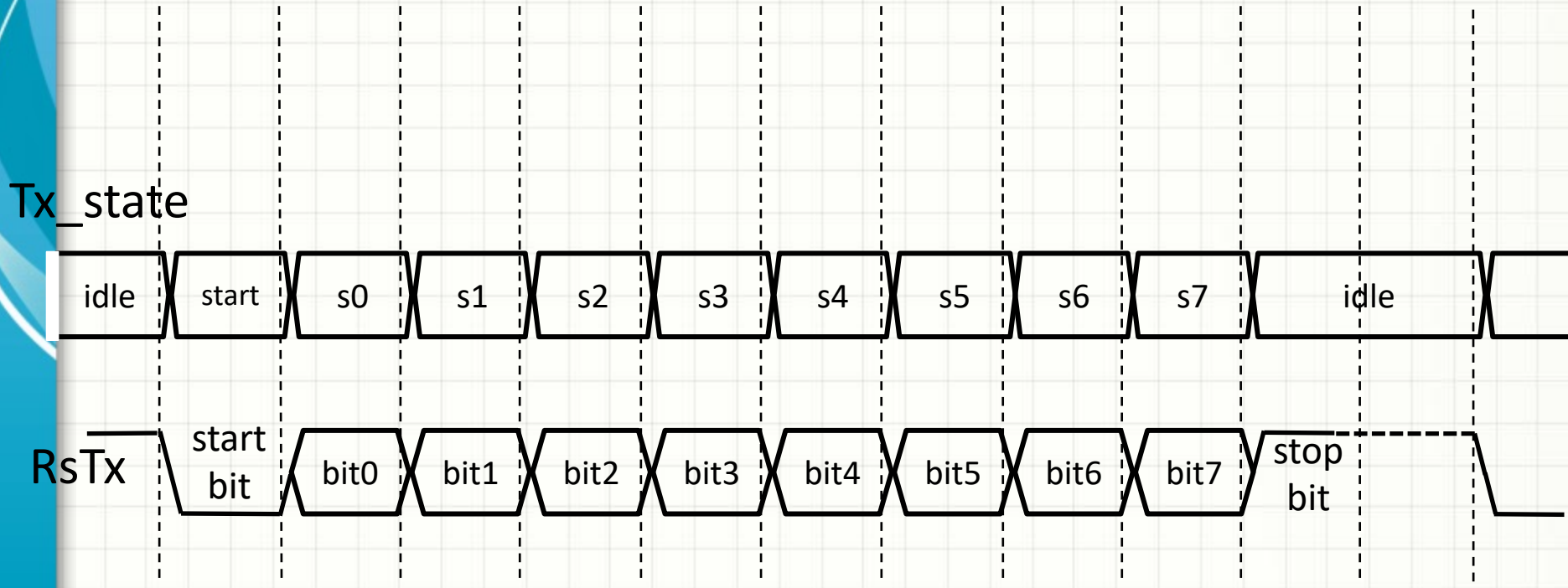
Receiver States



Receiver States - Zoomed

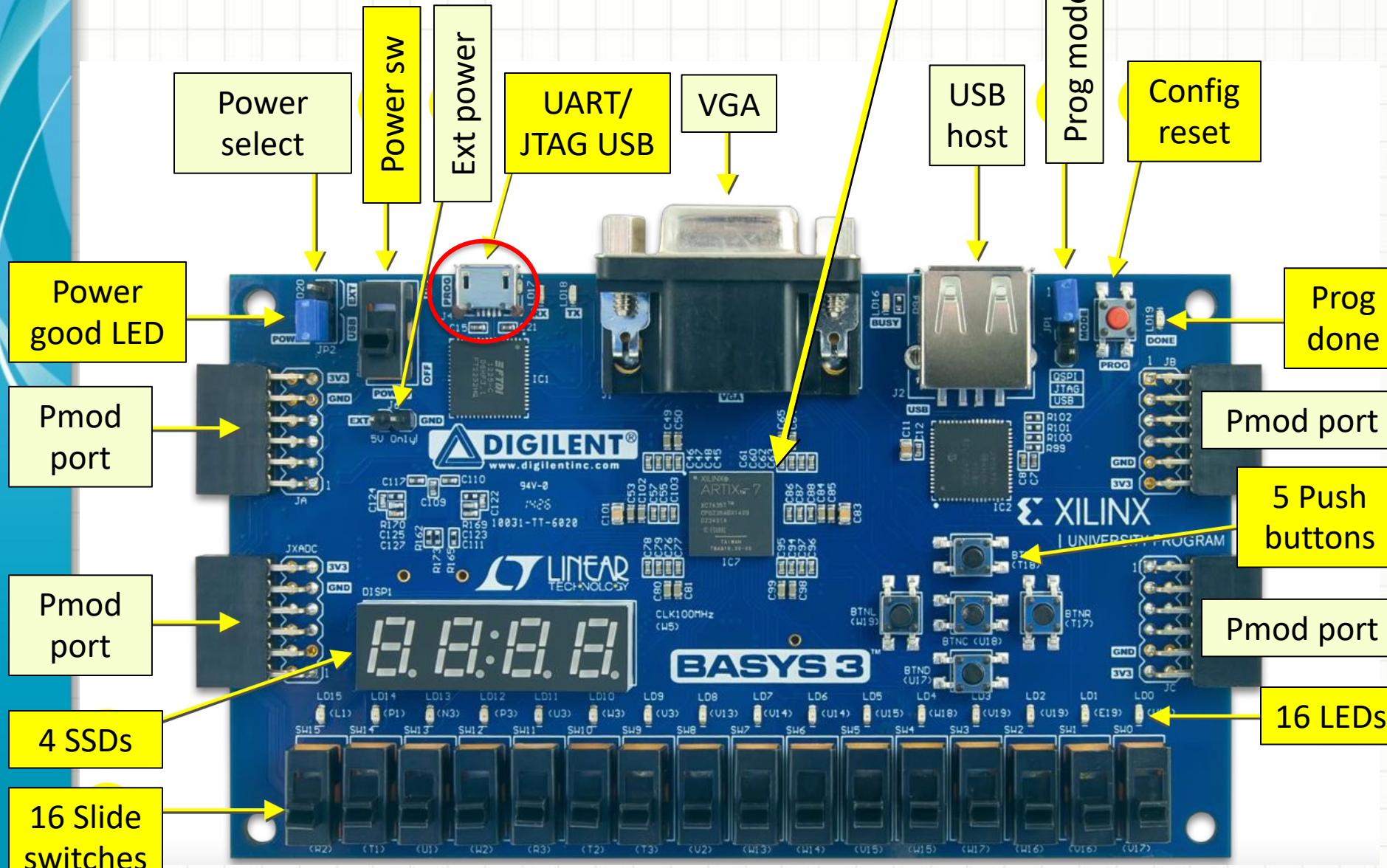


Transmitter States



RsTx : serial data output from transmitter
Tx_state : transmitter state

Implementing UART on BASYS 3 Board



RsRx and RsTx pins in xdc file

##USB-RS232 Interface

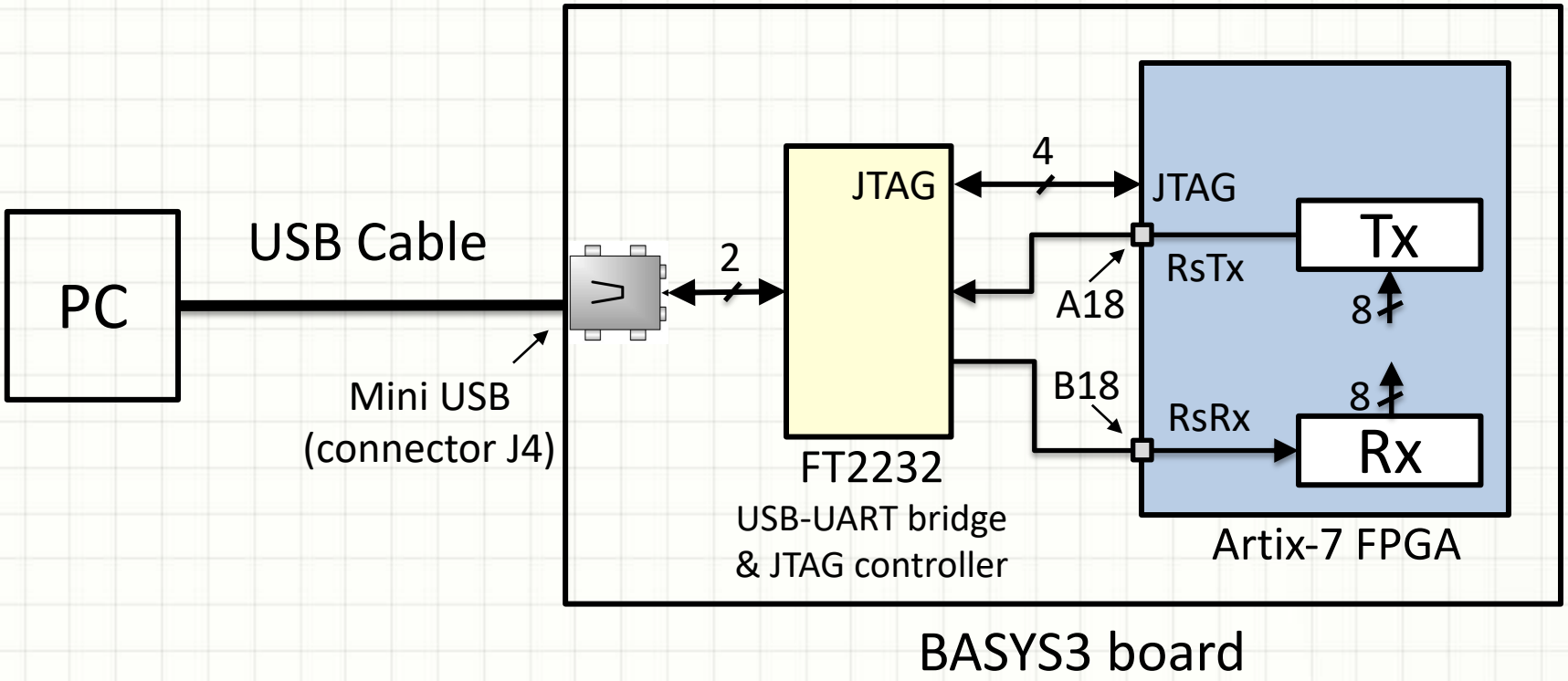
#set_property PACKAGE_PIN B18 [get_ports RsRx]

#set_property IOSTANDARD LVCMOS33 [get_ports RsRx]

#set_property PACKAGE_PIN A18 [get_ports RsTx]

#set_property IOSTANDARD LVCMOS33 [get_ports RsTx]

Tx and Rx on BASYS3 board



THANKS