Lecture 26
EE 421 / CS 425
Digital System Design

Fall 2023

Shahid Masud

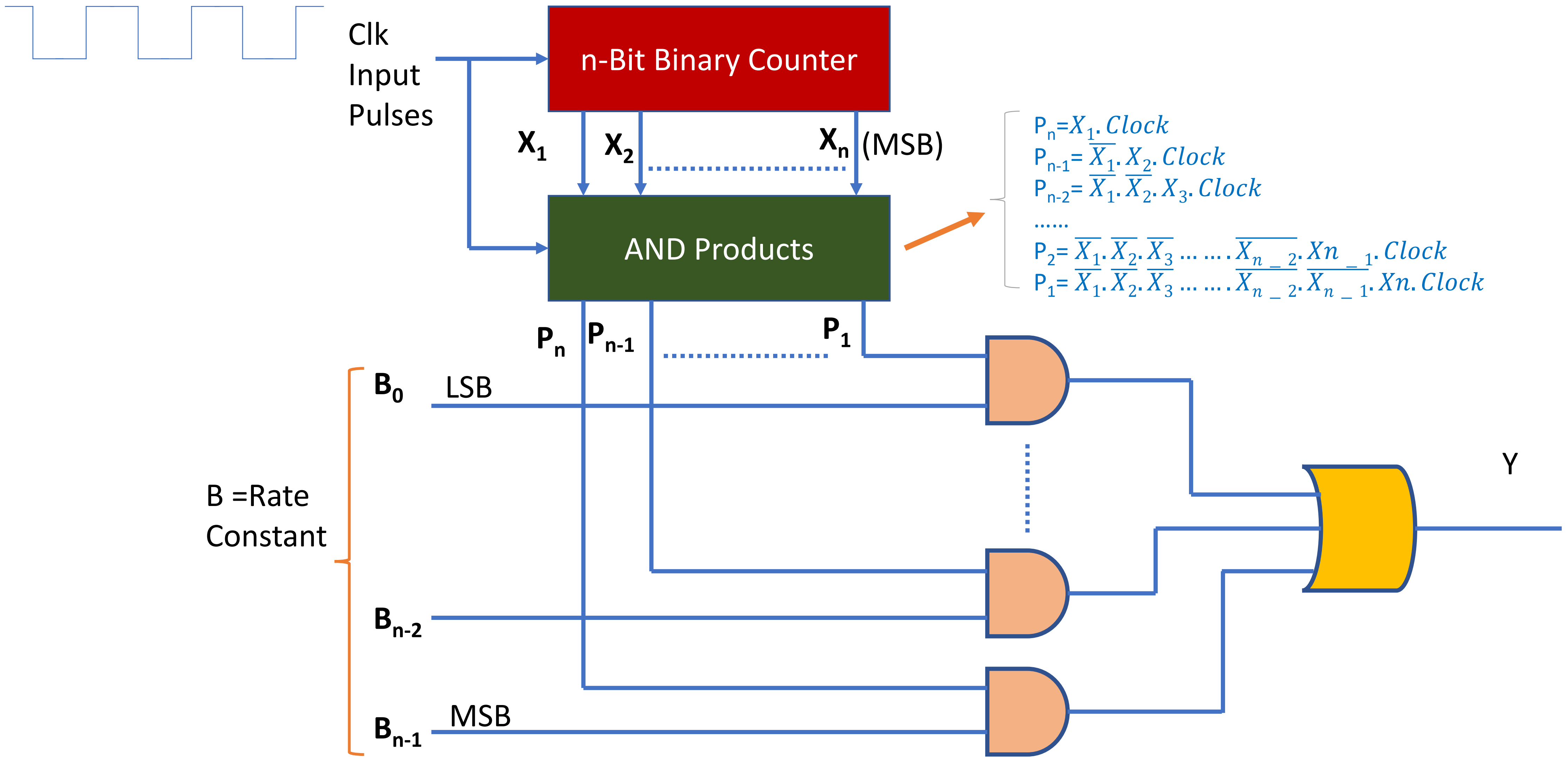
Topics

Quiz 6 Today

Quiz 7 Last Lecture

- BRM Applications
- BRM Chips
- UART Introduction
- UART Configuration
- UART Transmitter Operation
- UART Receiver Operation
- UART Architecture
- UART Registers

Basic Circuit Diagram – DFRM



P_n=$X_1.Clock$
P_{n-1}= $\overline{X_1}.\overline{X_2}.Clock$
P_{n-2}= $\overline{X_1}.\overline{X_2}.X_3.Clock$
......
P_2= $\overline{X_1}.\overline{X_2}.\overline{X_3}\ldots\ldots.\overline{X_{n\_2}}.Xn_{\_1}.Clock$
P_1= $\overline{X_1}.\overline{X_2}.\overline{X_3}\ldots\ldots.\overline{X_{n\_2}}.\overline{X_{n\_1}}.Xn.Clock$

Serially Cascaded Binary Rate Multiplier BRM

Design a BRM to implement Nout = $\frac{63}{320}$. Nin

Break into binary powers (63 x 2 = 126; and 320 x 2 = 640) :

Nout = $\frac{(7 \times 18)}{(10 \times 64)}$. Nin

Nout = $(\frac{7}{10}).(\frac{18}{64})$. Nin

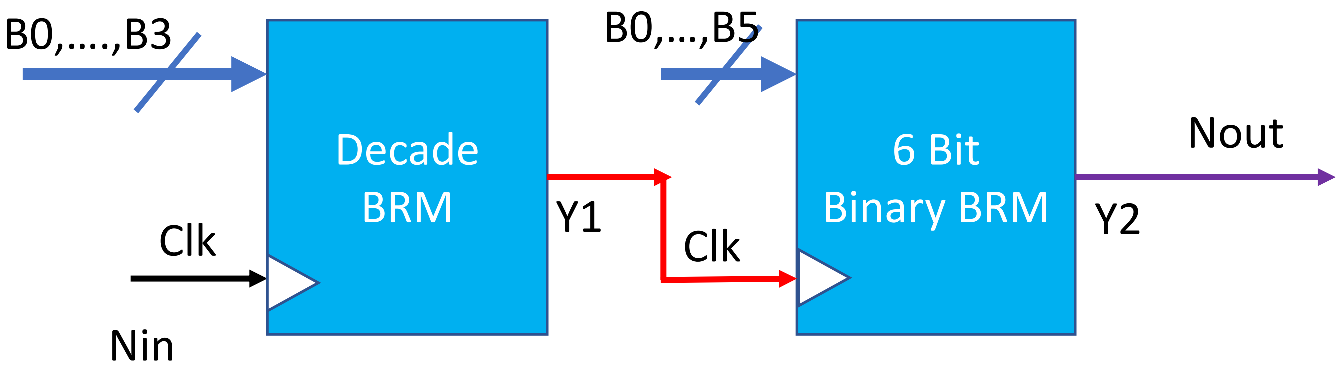
Use:

1 x Decade Counter BRM

1 x 6-Bit Binary Counter BRM

Set B for Decade Counter = 0111

Set B for 6-Bit BRM = 10010

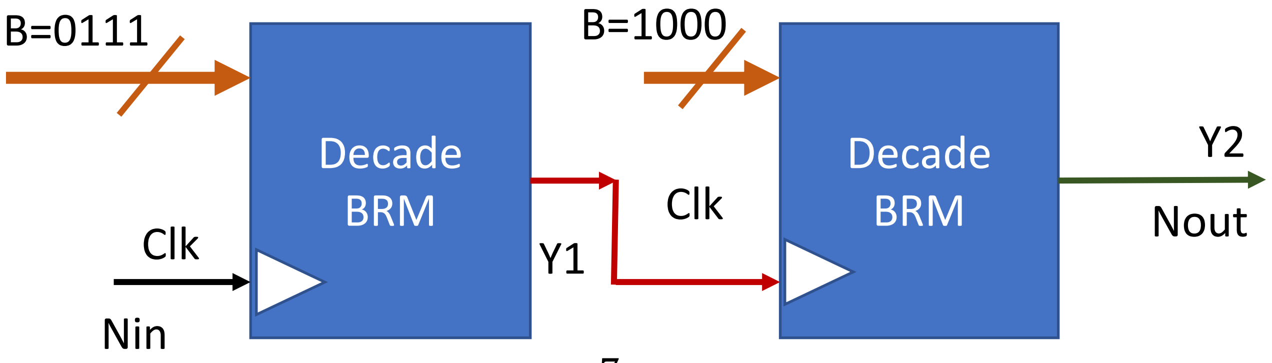


Example 2: Serially Cascaded BRM

Design a BRM circuit that produces: 0.56 of input clock rate
We use two Decade BRMs with
Value of B is set as 7 and 8 respectively

$$Nout = \frac{56}{100}.\,\text{Nin}$$

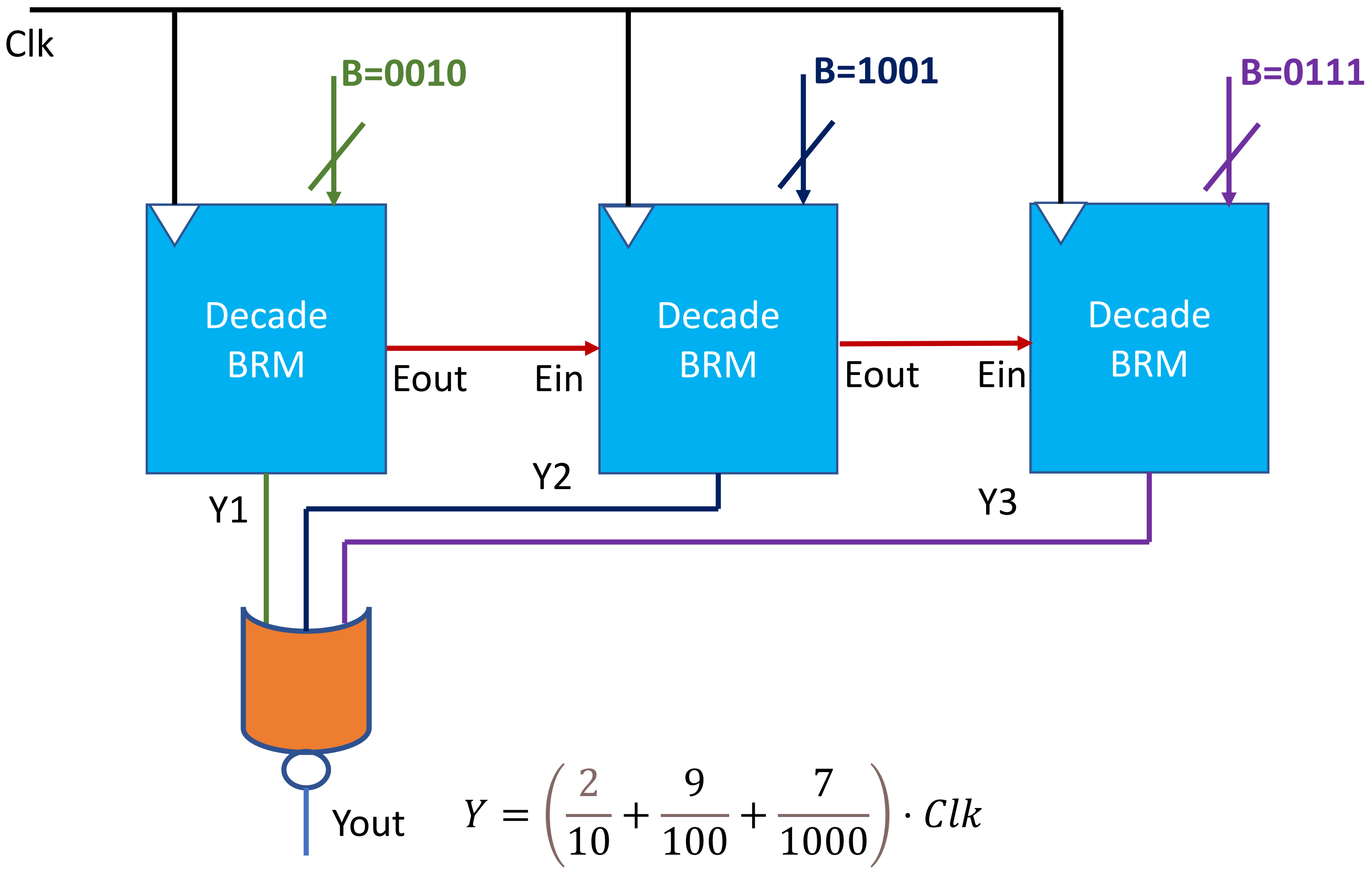
$$N = \frac{7}{10}.\frac{8}{10}.\,\text{Nin}$$



$$Y1 = \left(\frac{7}{10}\right).\,\text{Clk} \qquad Y2 = \left(\frac{7}{10}\right).\left(\frac{8}{10}\right).\text{Clk}$$

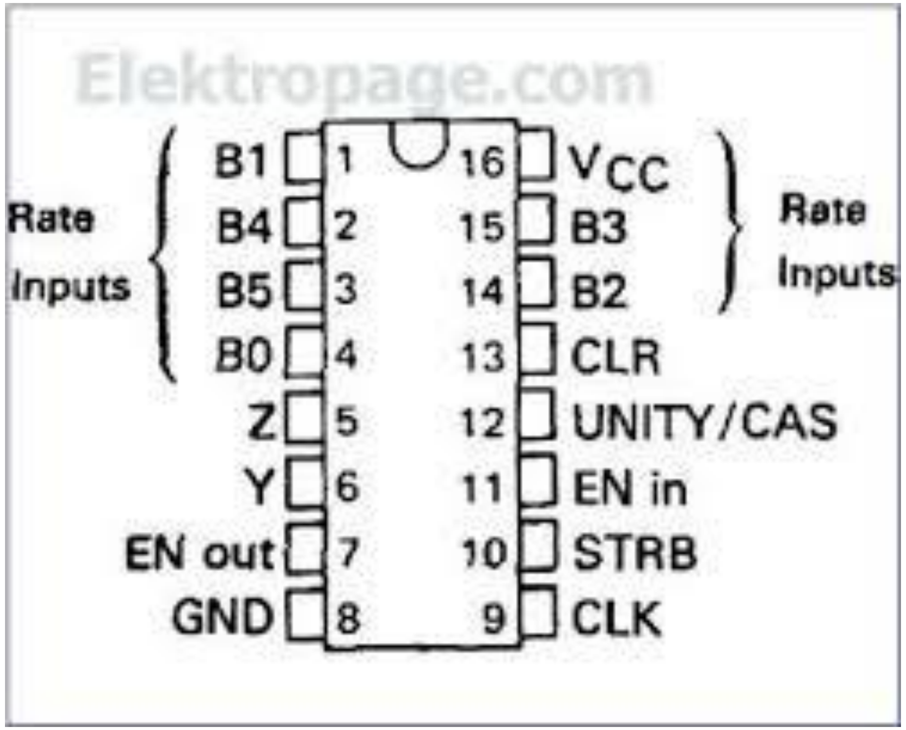
Eg 3: Parallel Cascaded Binary Rate Multiplier

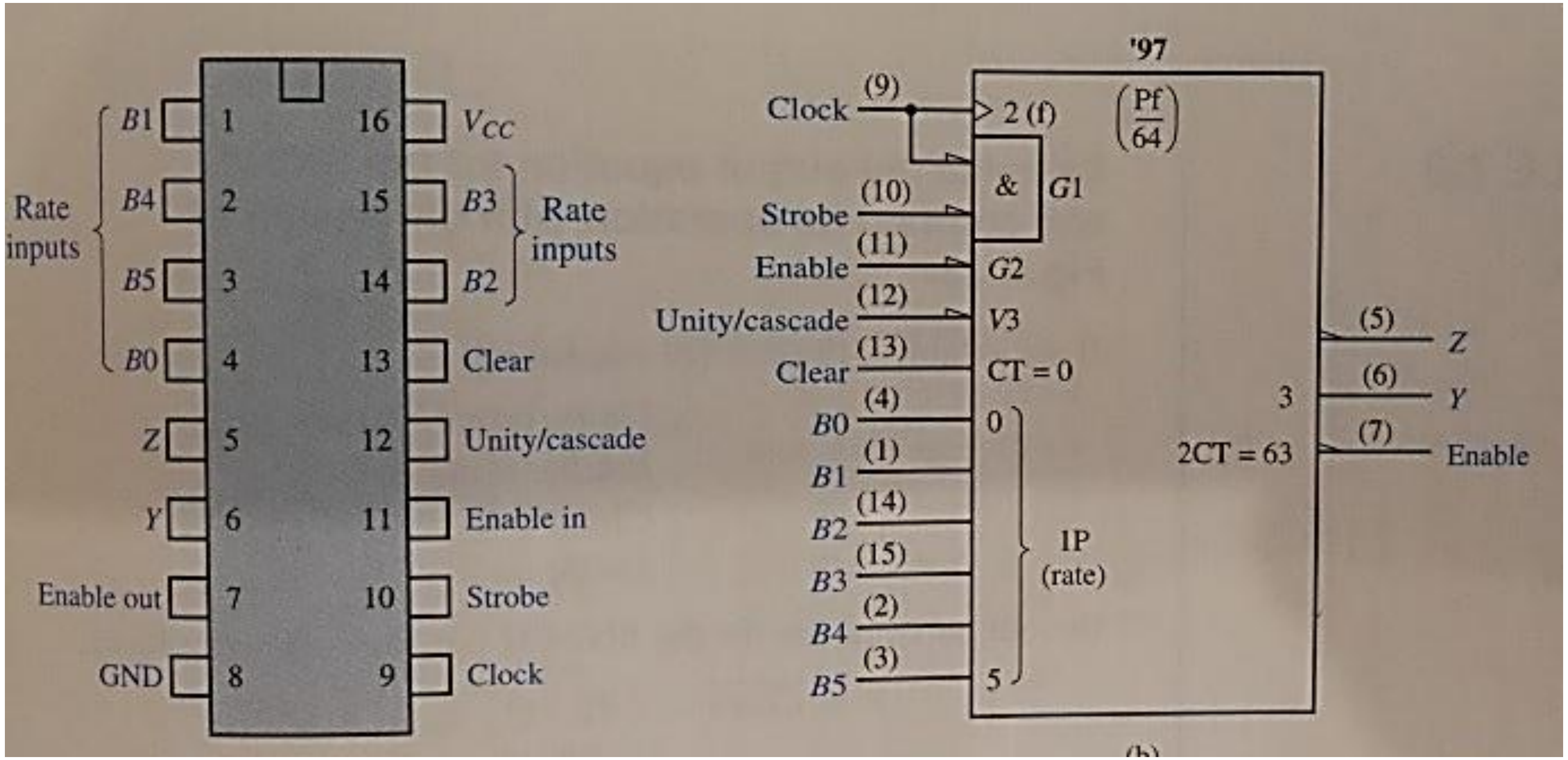
Design a BRM circuit to produce N_{out} = (0.297) × N_{in}



$$Y = \left(\frac{2}{10} + \frac{9}{100} + \frac{7}{1000}\right) \cdot Clk$$

7497 TTL chip 4-Bit and SN7497E chip 6-Bit BRM



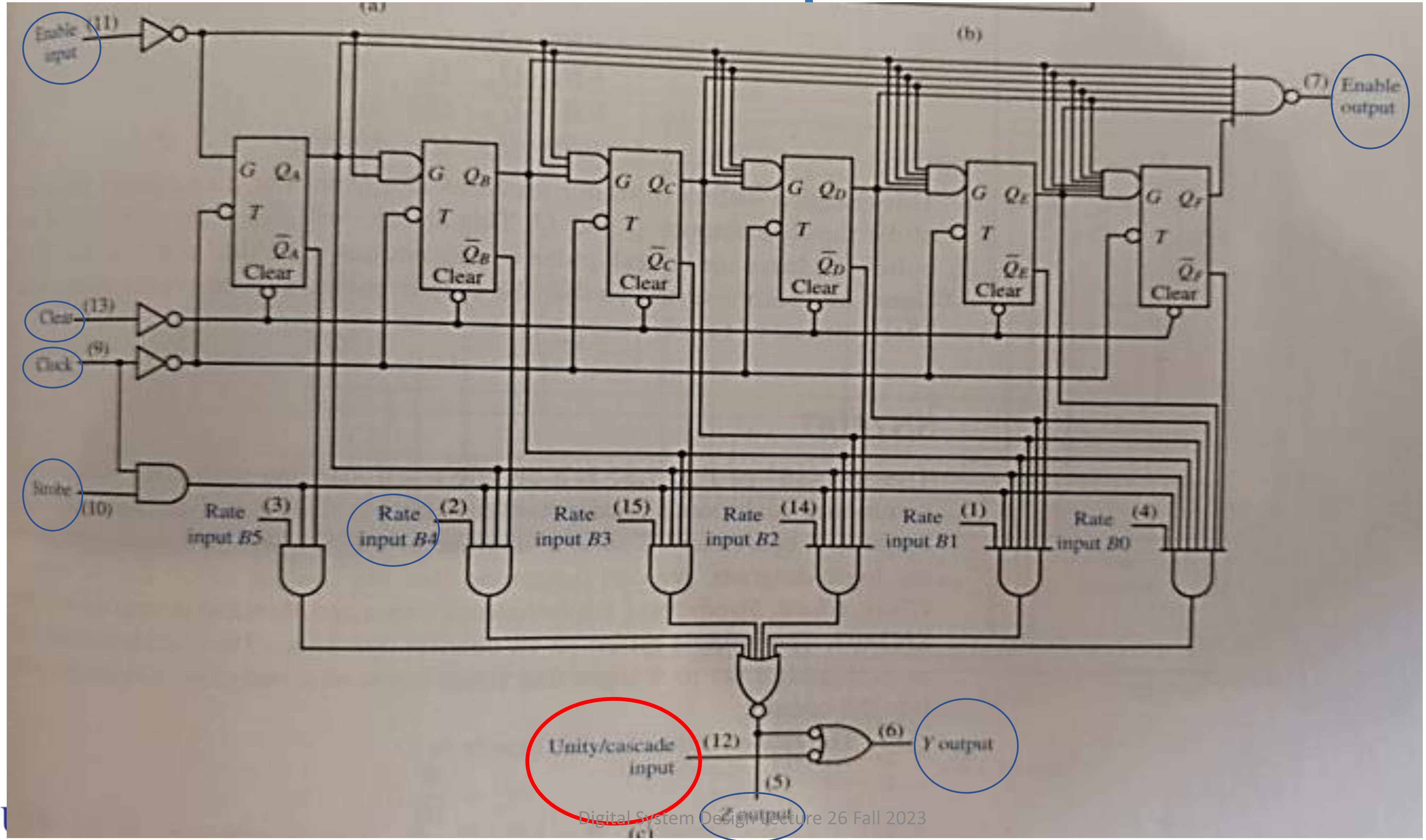


Ref: Images from internet

Pins on 7497 and SN7497E

- Clear signal is active high and asynchronous to drive all flip-flops to all-zero state
- Rising edge on Clock pulses causes counter to advance
- Strobe is an active-low Enable signal for the pulse rate output Y and Z
- Enable input and Enable output allows the modules to be cascaded
- The Unity / Cascade input may be used to combine the outputs of cascaded units, effectively ORing the pulses trains from two modules

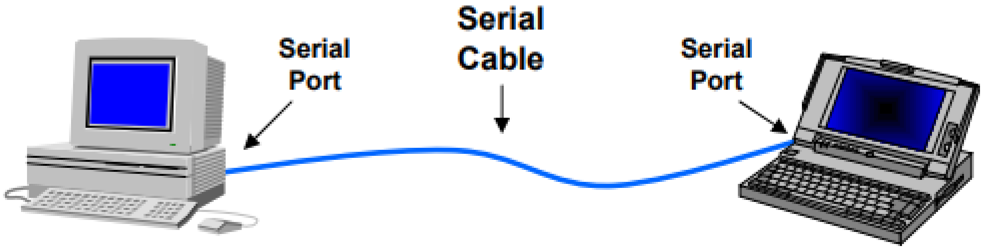
Internal view of BRM chip

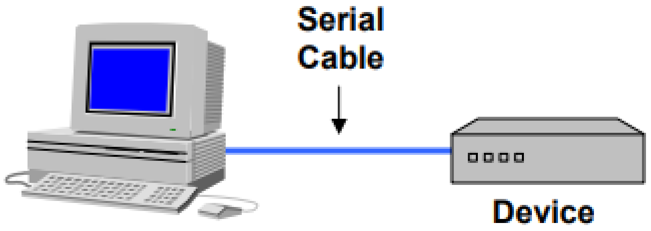


UART RS232 Protocol and Hardware Implementation

UART for Serial Communication

- PC serial port is a UART!
- Serializes data to be sent over serial cable
  - De-serializes received data





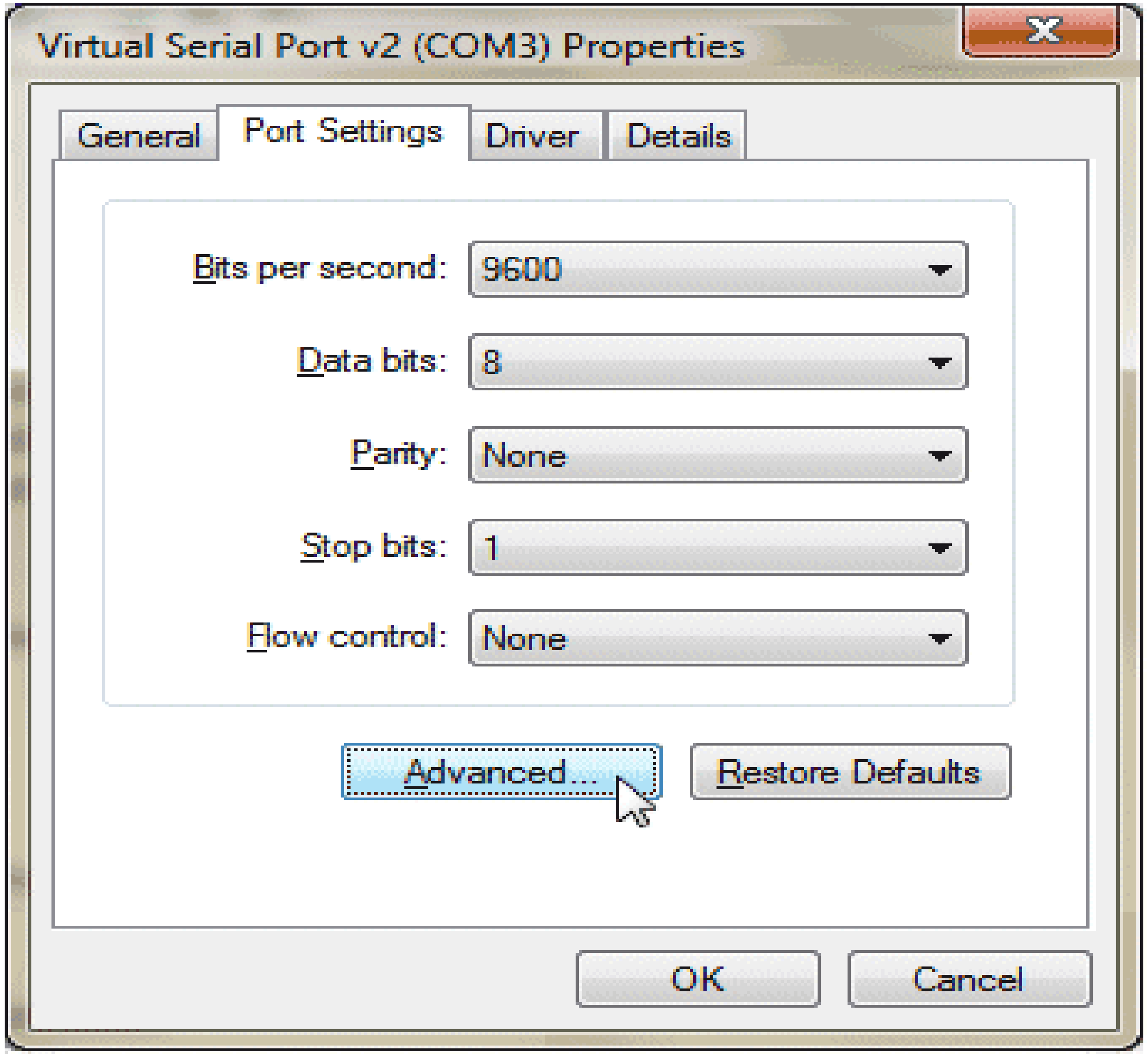
UART Functions

- Outbound data
  - Convert from parallel to serial
  - Add start and stop delineators (bits)
  - Add parity bit
- Inbound data
  - Convert from serial to parallel
  - Remove start and stop delineators (bits)
  - Check and remove parity bit

UART Principle

- UART translates Parallel Data (Bytes or ASCII) to Serial Data for sending and vice versa for receiving.
- Inside UART there are several registers to store status and data for sending and receiving.
- The CPU can read and write these registers through I/O ports.
- For correct communication, both UARTs must make agreement on data frame format and transmission speed (baud rate).

Serial Port Setting on PC using HyperTerminal



Virtual Serial Port v2 (COM3) Properties

Bits per second: 9600

Data bits: 8

Parity: None

Stop bits: 1

Flow control: None

Configurable Baud Rate

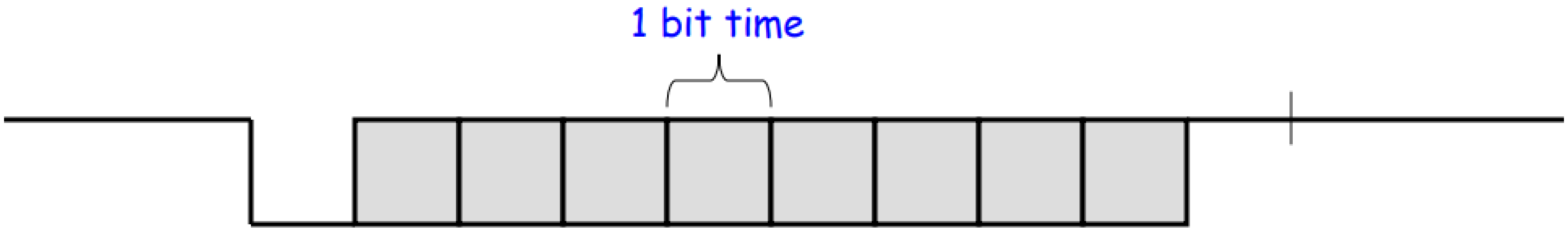
- Definition: “How many times a signal can change per second” Includes data bits plus control bits.
- Possible Baud Rates:
  - 110
  - 150
  - 300
  - 600
  - 1200
  - 2400
  - 4800
  - 9600
  - 19200
  - 38400
  - 57600, 115200, 230400, etc.

UART Throughput

- 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**
- How many KB (kilobytes) per second is this?
  - 1 byte = 8 bits
  - 1 KB = 1,024 bytes
  - So, 1 KB = 1,024 bytes • 8 bits/byte = 8,192 bits
  - Finally, 15,360 bps • 1 KB / 8,192 bits = **1.875 KB/s**

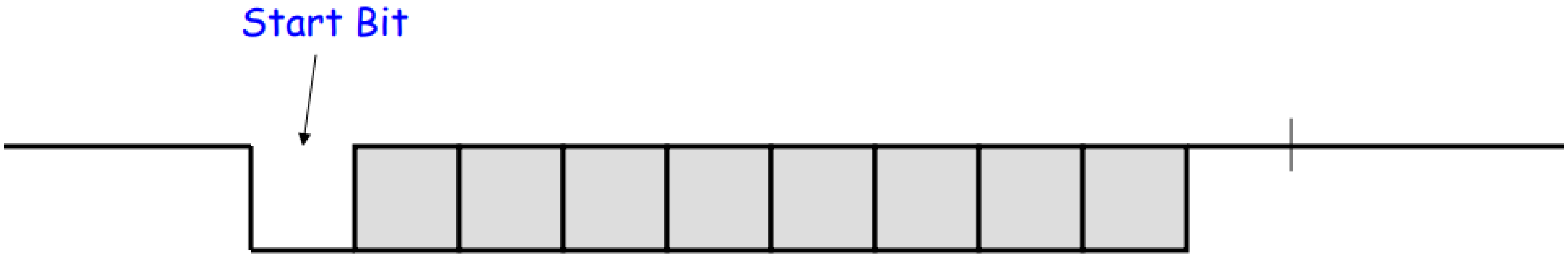
UART Character Transmission

- Each bit has a fixed time duration determined by the transmission rate
- Example: a 1200 bps (bits per second) UART will have a 1/1200 s or about 833.3 μs bit width



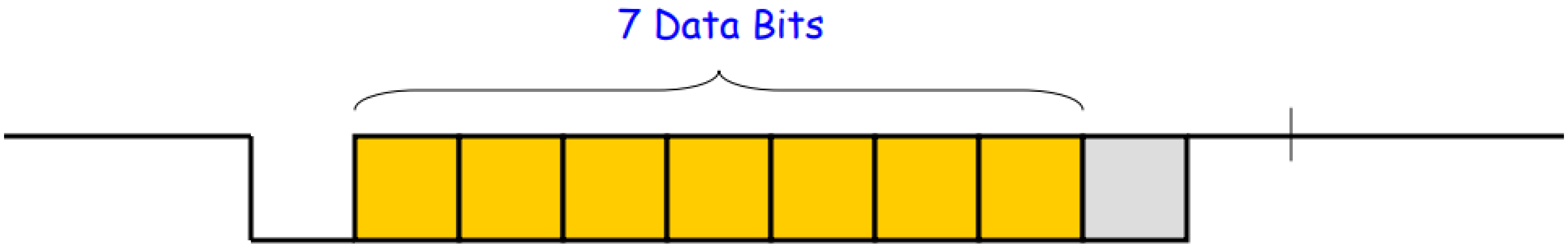
UART Start Bit in Transmission

- The **start bit** marks the beginning of a new word
- When detected, the receiver synchronizes with the new data stream



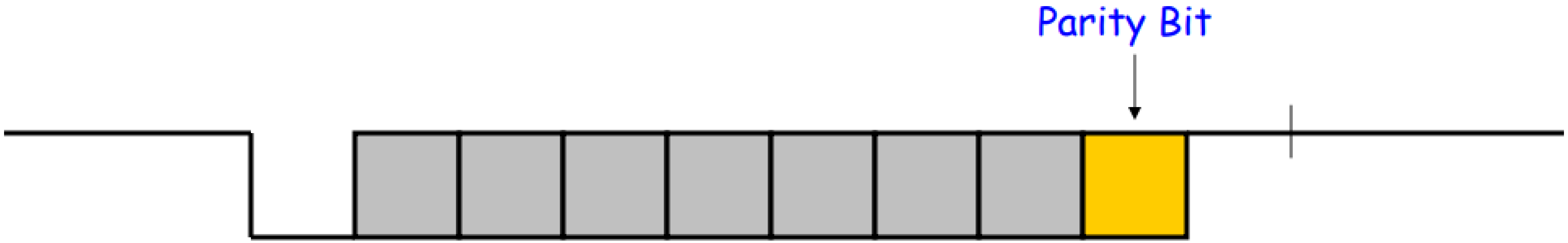
UART Data Bits in Transmission

- Next follows the **data bits** (7 or 8)
- The least significant bit is sent first



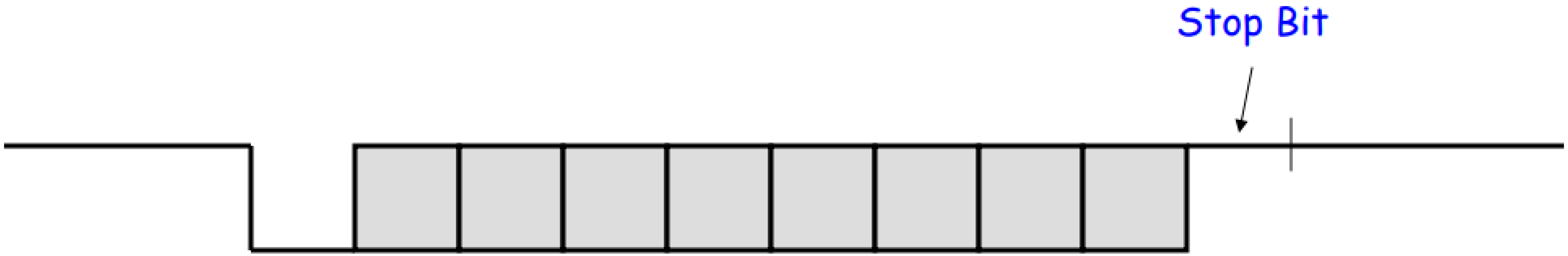
UART Parity Bit in Transmission

- 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



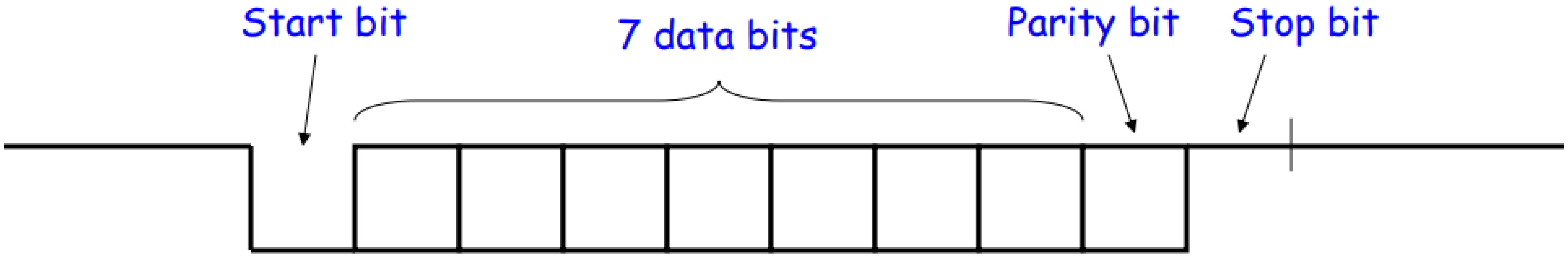
UART Stop Bit in Transmission

- The **stop bit** marks the end of transmission
- Receiver checks to make sure it is '1'
- Separates one word from the start bit of the next word



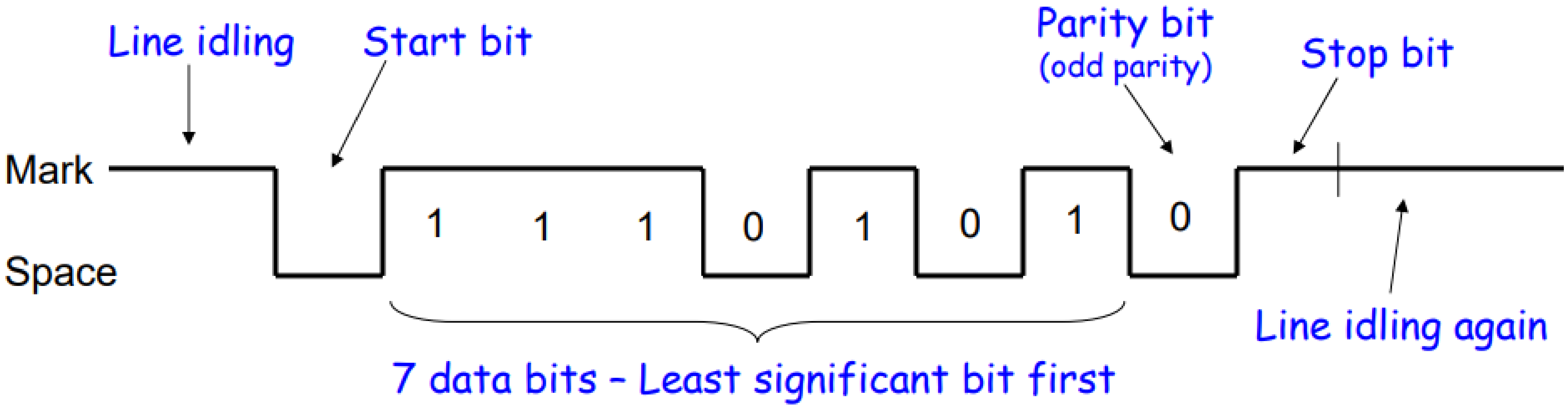
UART Transmission of 7 Bits

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

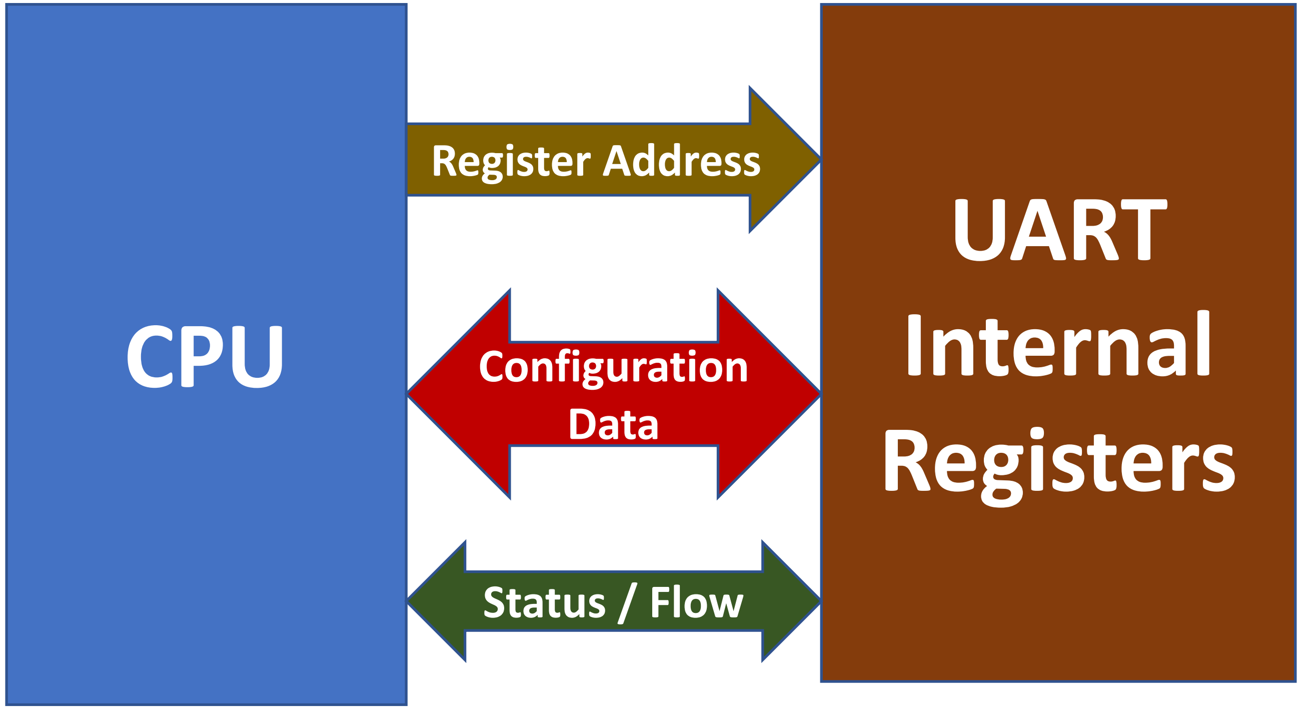


UART Transmit Ascii ‘W’

- Send the ASCII letter 'W' (1010111)



CPU and UART Configuration



Configuration Data

Baud Rate

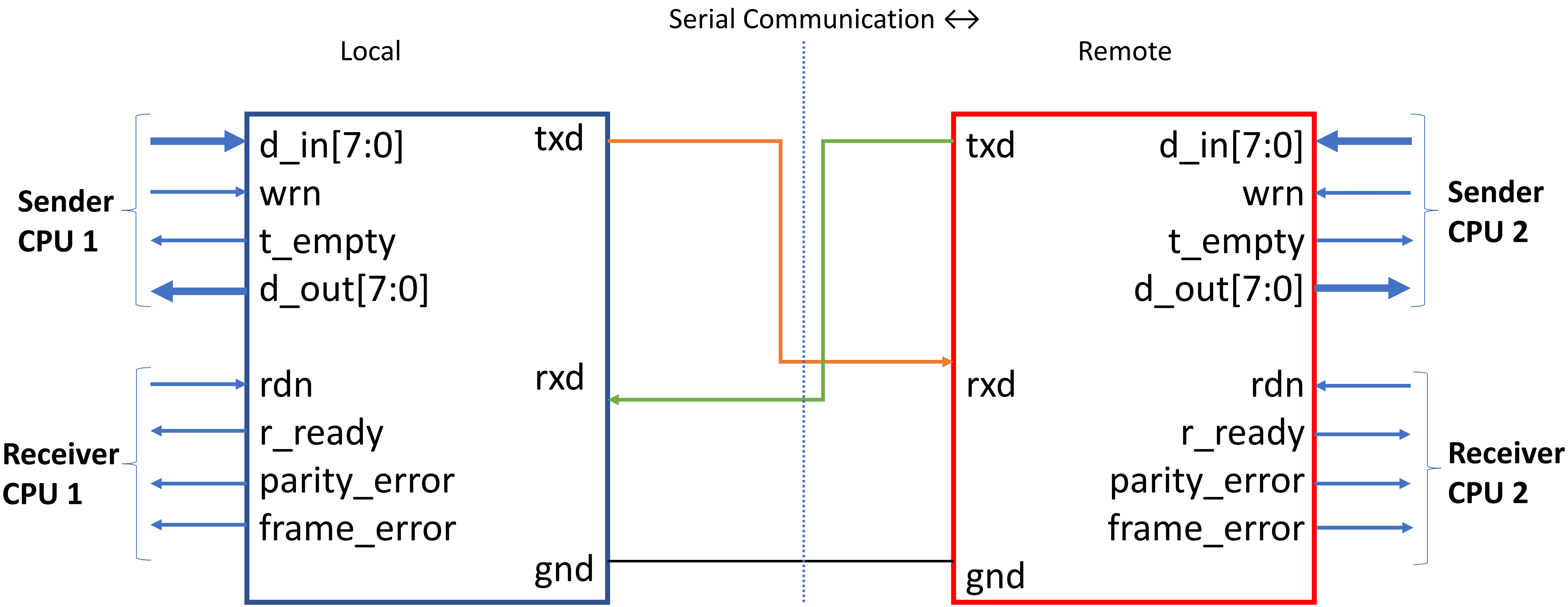
No. of Stop Bits (1, 2)

Parity (Yes, No, Even, Odd)

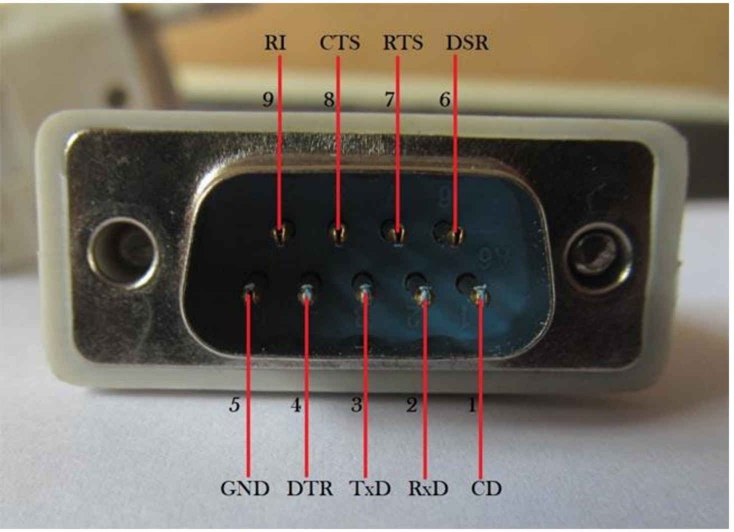
Flow Control (RTS, CTS)

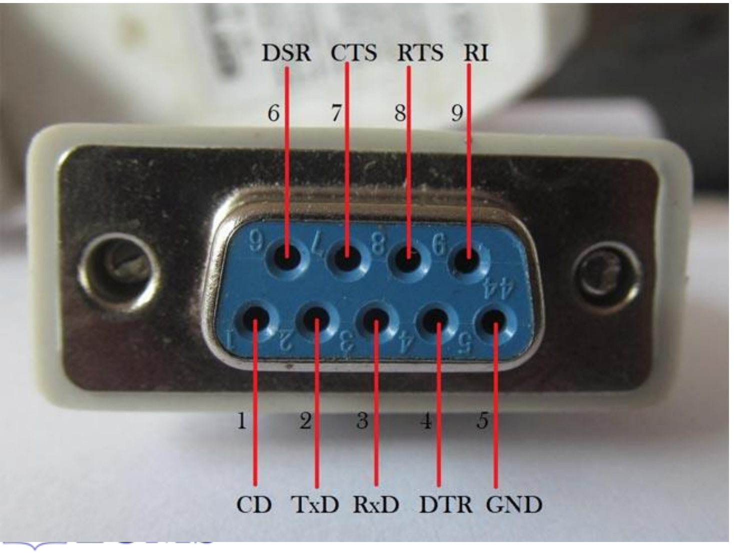
Data Bits (5 to 8)

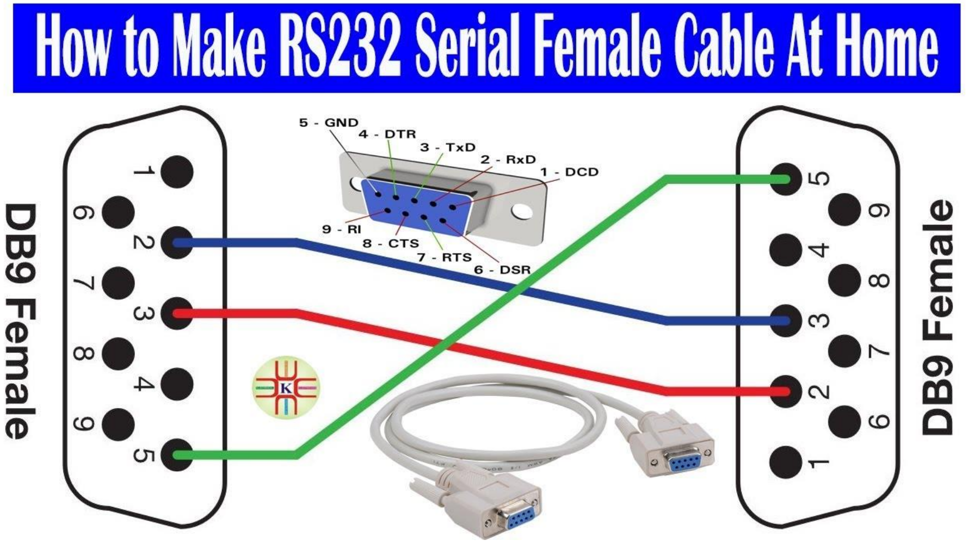
Connecting two UARTs

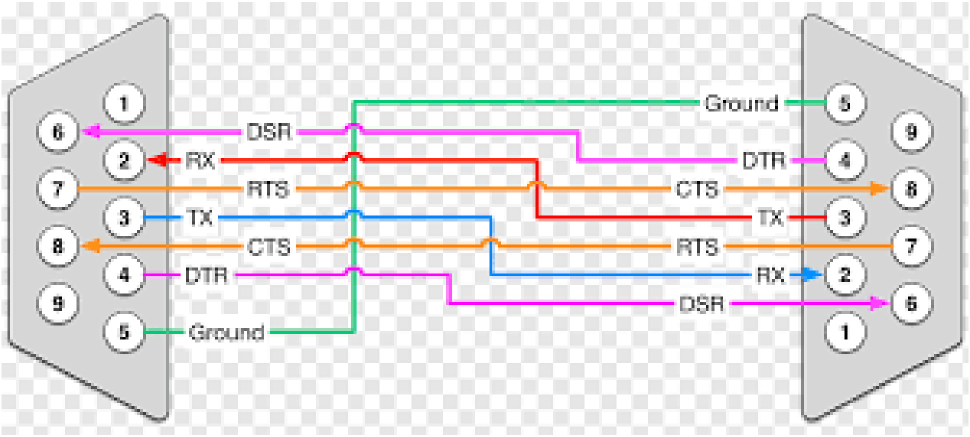


Physical Connections through RS 232





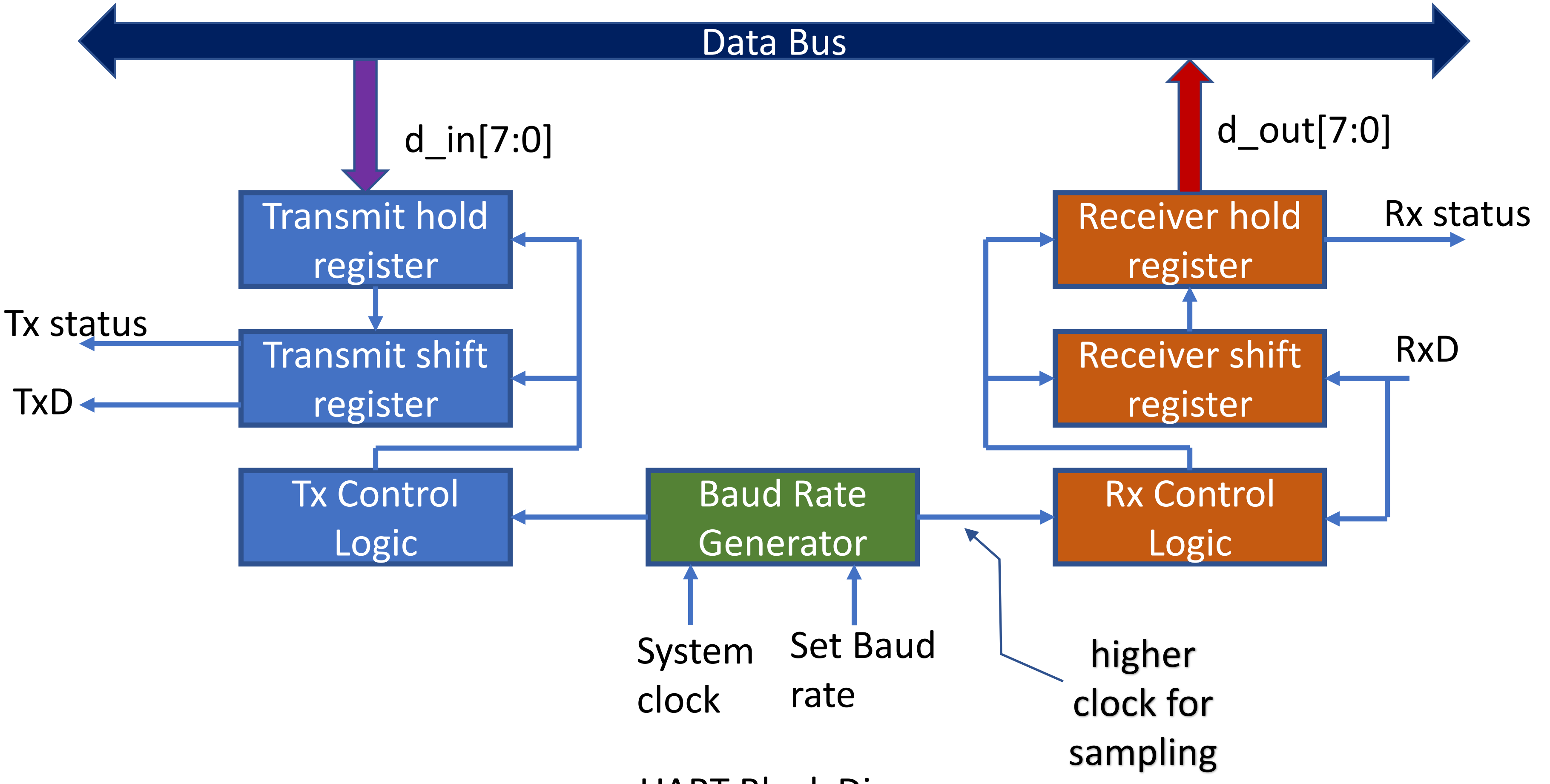




Typical UART Components

- **Baudrate clock generator**: Multiple of the bit rate to improve sampling in the middle of a bit period.
    - For generating this timing information, each UART uses an oscillator generating a frequency of about 1.8432 MHz (as an example).
    - This frequency is divided by 16 to generate the time base for communication.
    - Hence the maximum allowed communication speed is 115200 bps.
- **Input and output shift registers**: Each UART contains a shift register which is the fundamental method of conversion between serial and parallel forms. These registers shifts the data that has to be serially transmitted or serially received.
- **Transmit and receive control**: This control logic checks for the control signals from host processor to start or stop the transmission and reception of the data bits. In case of any error it also generates error signals.
- **Optional transmit and receive buffers:** Buffers can be used to hold the data temporarily.
- **Optional parallel data bus buffer:** This buffer improves the speed.
- **Optional FIFO:** The UART works by writing data from the host processor to its FIFO buffers and feeding the data from the buffer to the serial device in the format dictated by the user.

UART Architecture Block Diagram



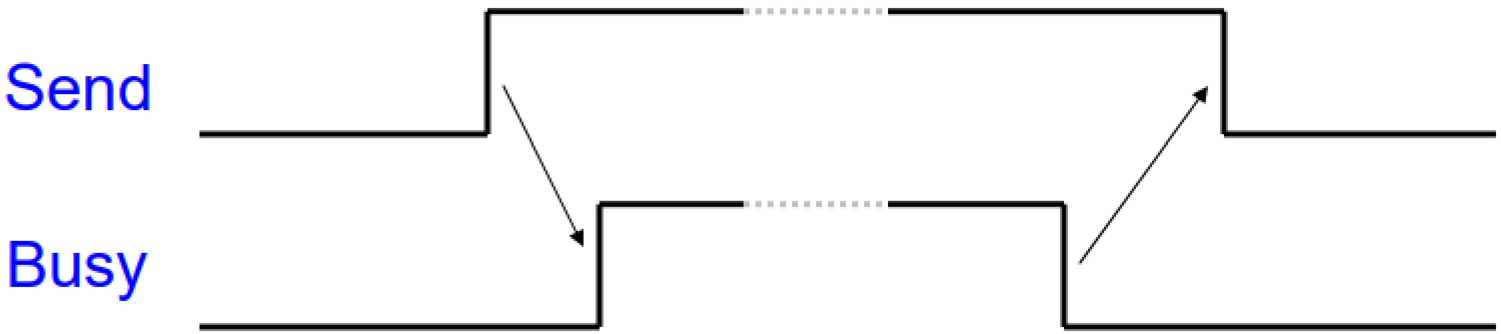
UART Block Diagram

Process of Sending

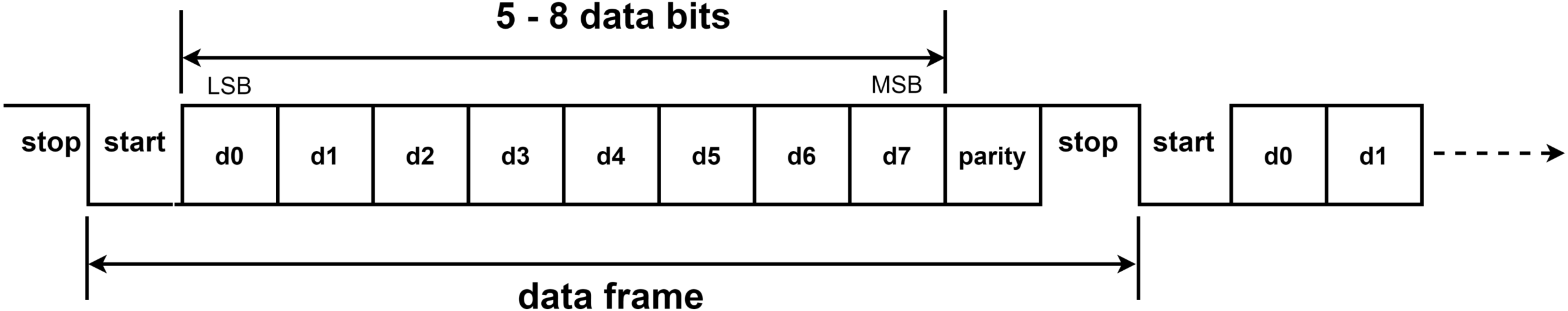
- Asynchronous transmission does not send clock signal to transmit data to the receiver. The sender and receiver must agree on timing parameters in advance and special bits such as start and stop bits are added to each word which is used to synchronize the sending and receiving units.
- A bit called the "**Start Bit**" is added to the beginning of each word that is to be transmitted. The **Start Bit** indicates the start of the data transmission, and it alerts the receiver that a word of data is about to be sent.
- Upon reception of start bit the clock in the receiver goes into synchronization with the clock in the transmitter. The accuracy of these two clocks should not deviate more than 10% during the transmission of the remaining bits in the word.
- The individual bits of the word of data are sent after the start bit. Least Significant Bit (LSB) is sent first.
- The transmitter does not know when the receiver has read at the value of the bit. The transmitter begins transmitting the next bit of the word on next clock edge.
- Parity bit is be added when the entire data word has been sent. This bit can be used to detect errors at the receiver side. Then one Stop Bit is sent by the transmitter to indicate the end of the valid data bits.

UART Transmitter / System Handshaking

- System asserts Send and holds it high when it wants to send a byte
- UART asserts Busy signal in response
- When UART has finished transfer, UART de-asserts Busy signal
- System de-asserts Send signal



Data frame format of UART



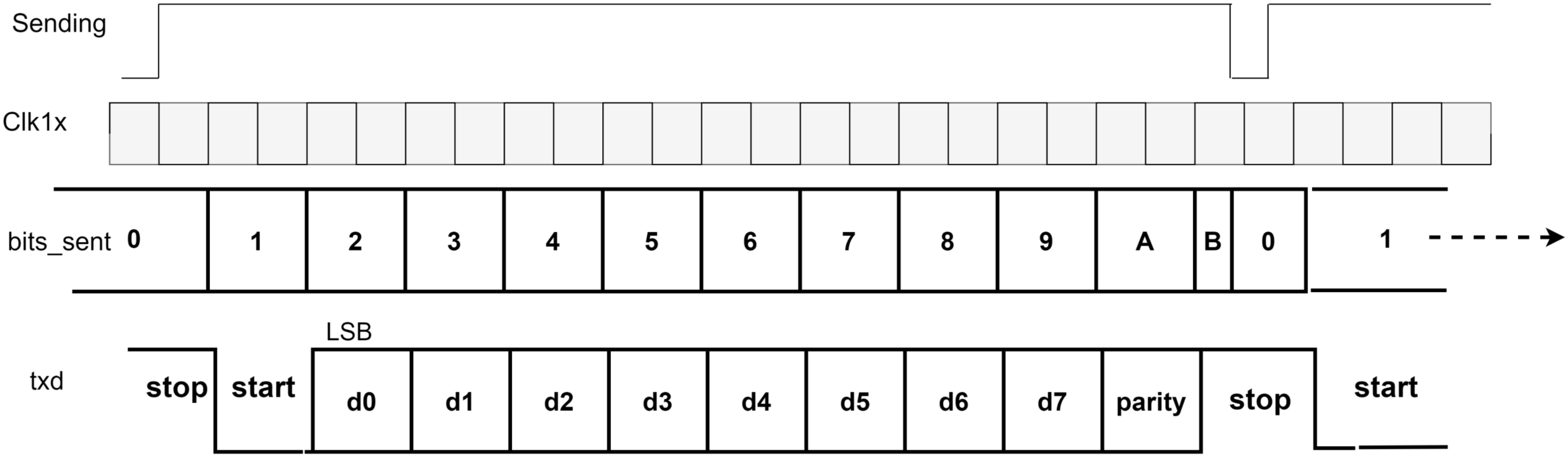
LSB of data bits is transmitted first

Start bit is logic low

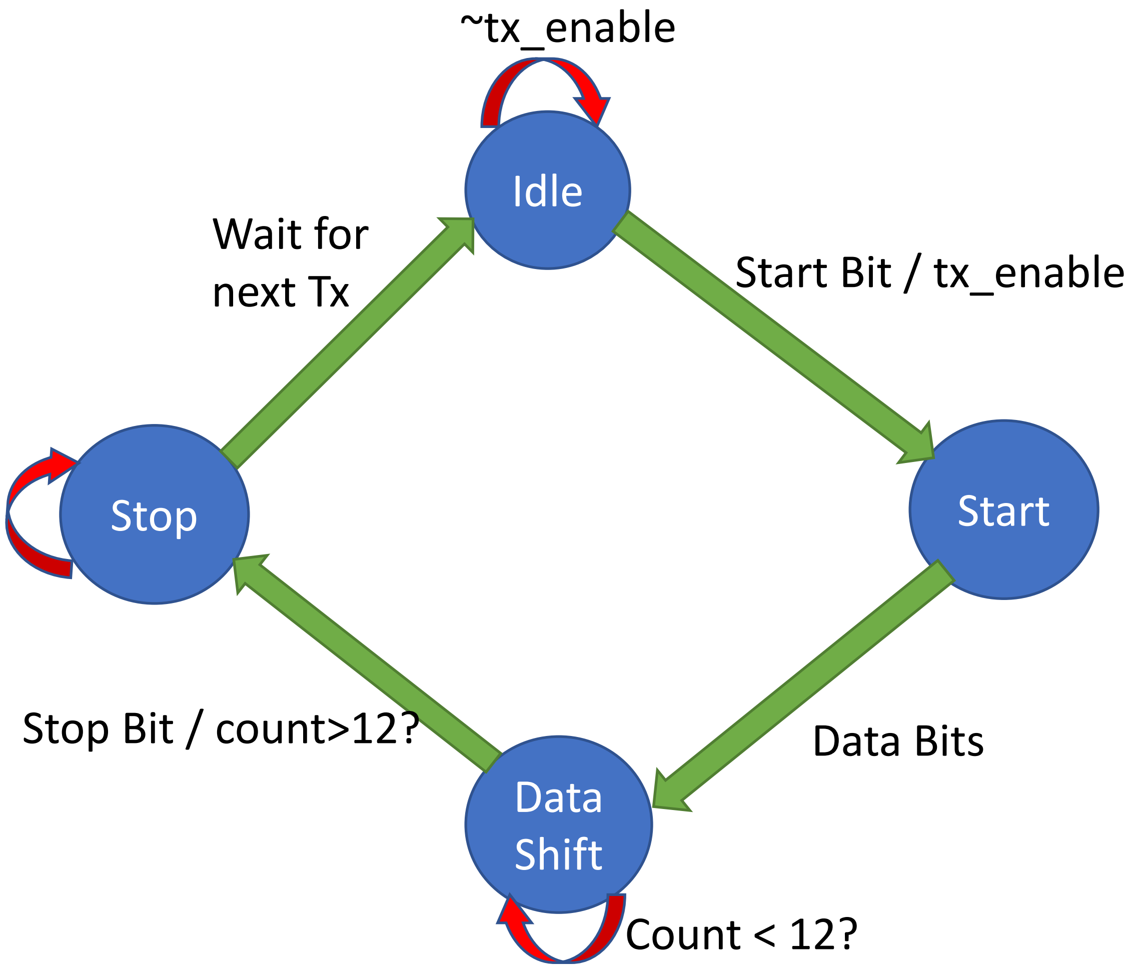
Stop bit is logic high

In RESET Condition, Serial Data Transmission is ‘High’

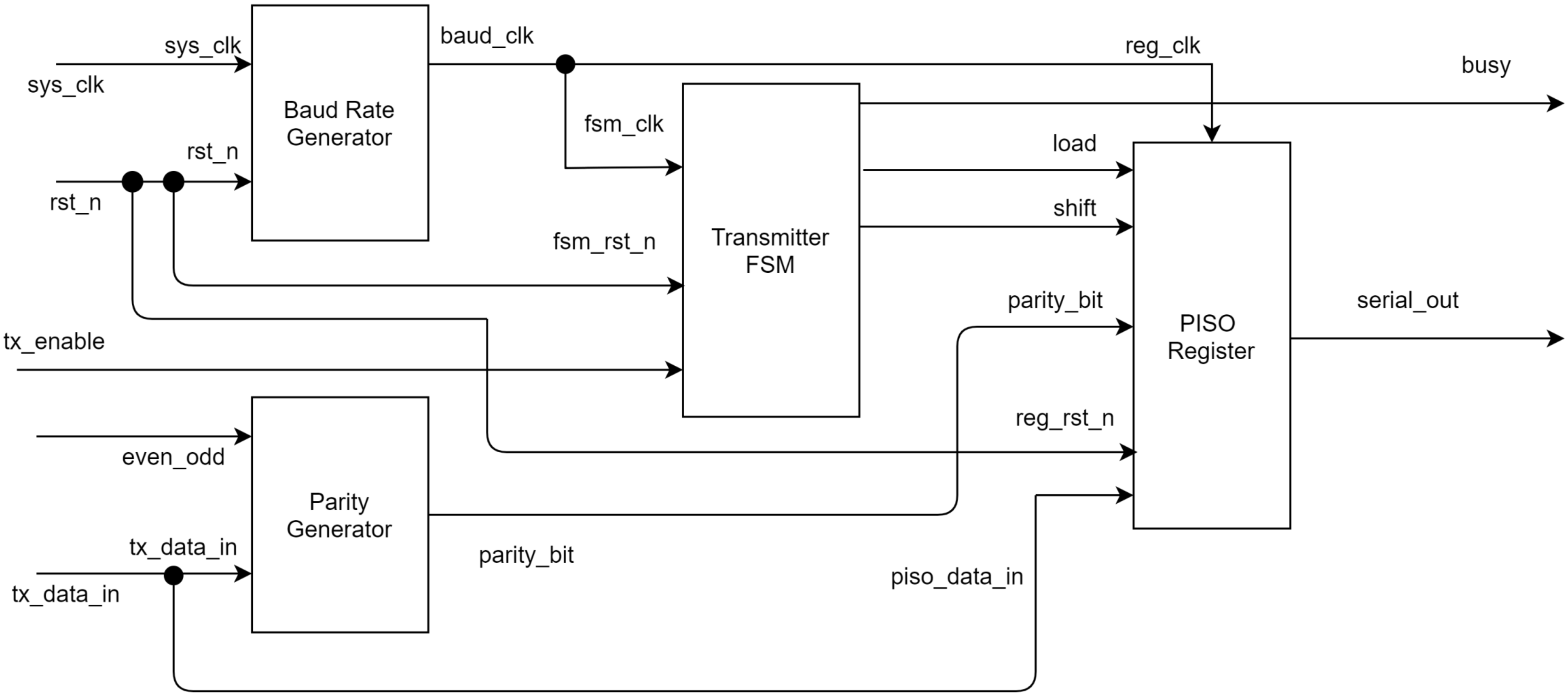
Sending clock of UART Transmitter



FSM for UART Transmitter



UART Transmitter Architecture



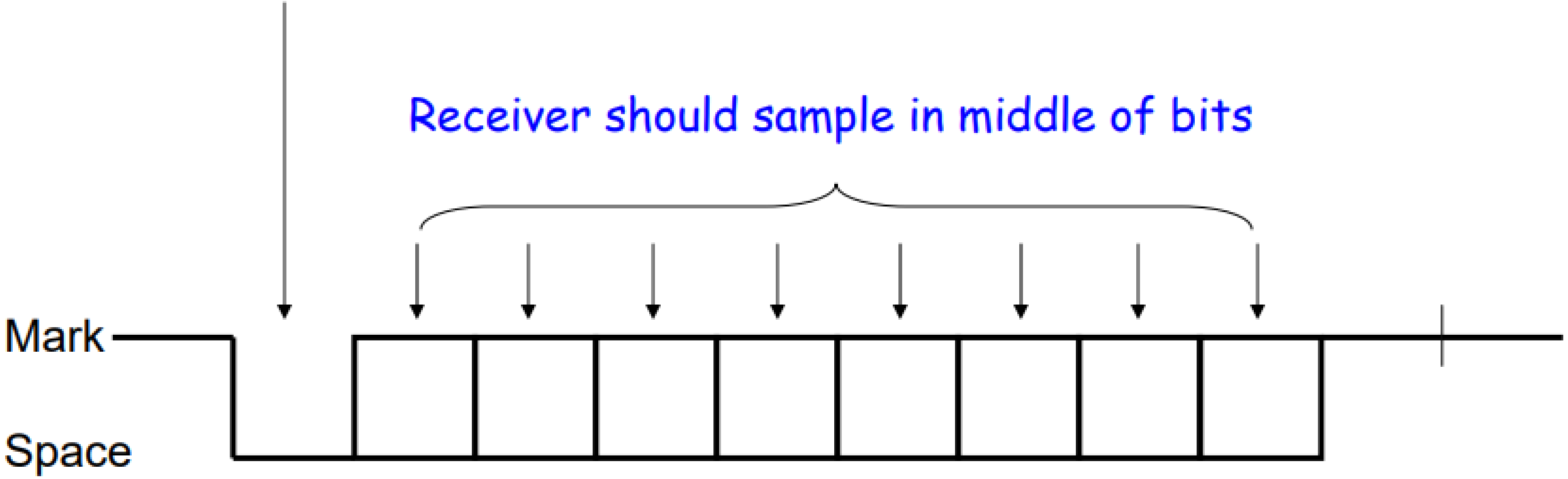
Process of Receiving

- On the receiver side once it receives all of the bits in the data word, it can check for the Parity Bits. To accomplish this task both transmitter and receiver must agree on whether a Parity Bit is to be used.
- Then Stop Bit is encountered by receiver. A missing stop bit may result entire data to be garbage. This will cause a Framing Error and will be reported to the host processor when the data word is read. Framing Error can be caused due to mismatch of transmitter and receiver clocks.
- The UART automatically discards the Start, Parity and Stop bits irrespective of whether data is received correctly or not. If the sender and receiver are configured identically, these bits are not passed to the host. To transmit new word, the Start Bit for the new word is sent as soon as the Stop Bit for the previous word has been sent.
- The transmission speed in asynchronous communication is measured by Baud Rate. A Baud Rate represents the number of bits that are actually being sent over the media. The Baud rate includes the Start, Stop and Parity bits. The Bit rate (Bits per Second-bps) represents the amount of data that is actually sent from the transmitting device to the other device. Speeds for UARTs are in bits per second (bit/s or bps), although often incorrectly called the baud rate. Standard baud rates are: 110, 300, 1200, …..,115200,…..,1843200, and 2764800 bit/s.

UART Character Reception

Start bit says a character is coming, receiver resets its timers

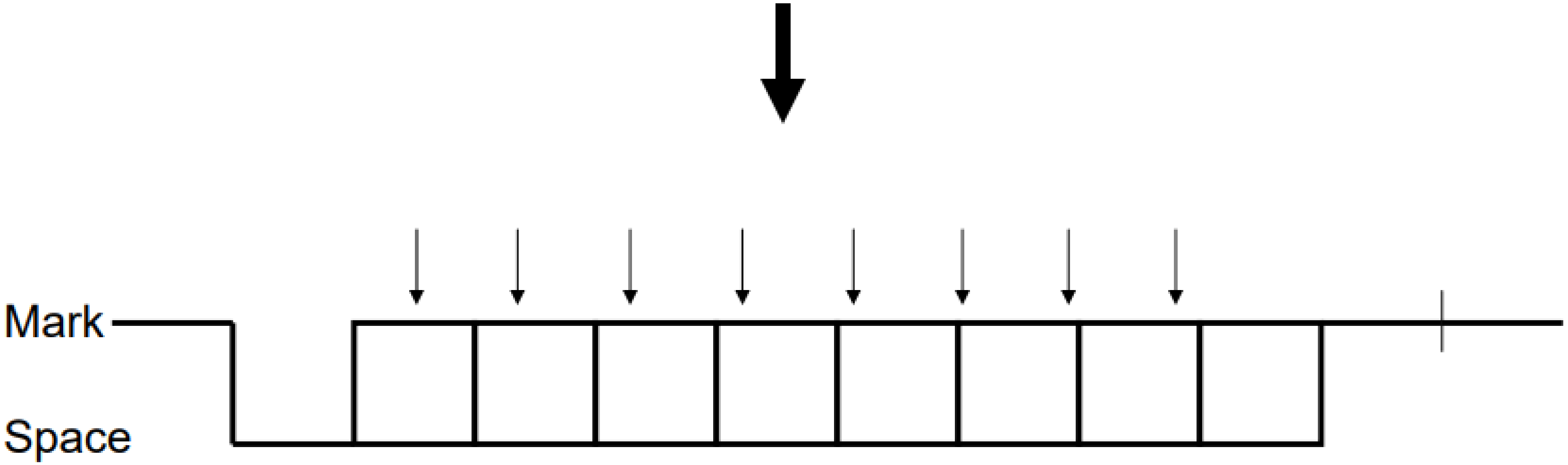
Receiver should sample in middle of bits



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

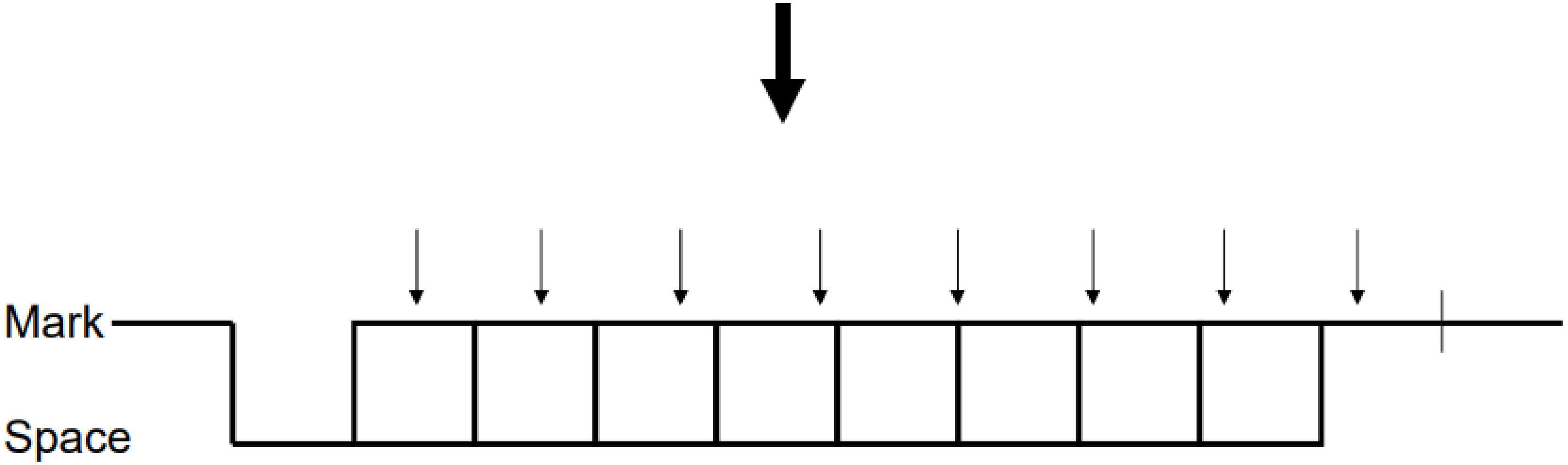
UART Receiver 'Quick' Sampling

If receiver samples too quickly, see what happens…



UART Receiver ‘Slow’ Sampling

If receiver samples too slowly, see what happens…

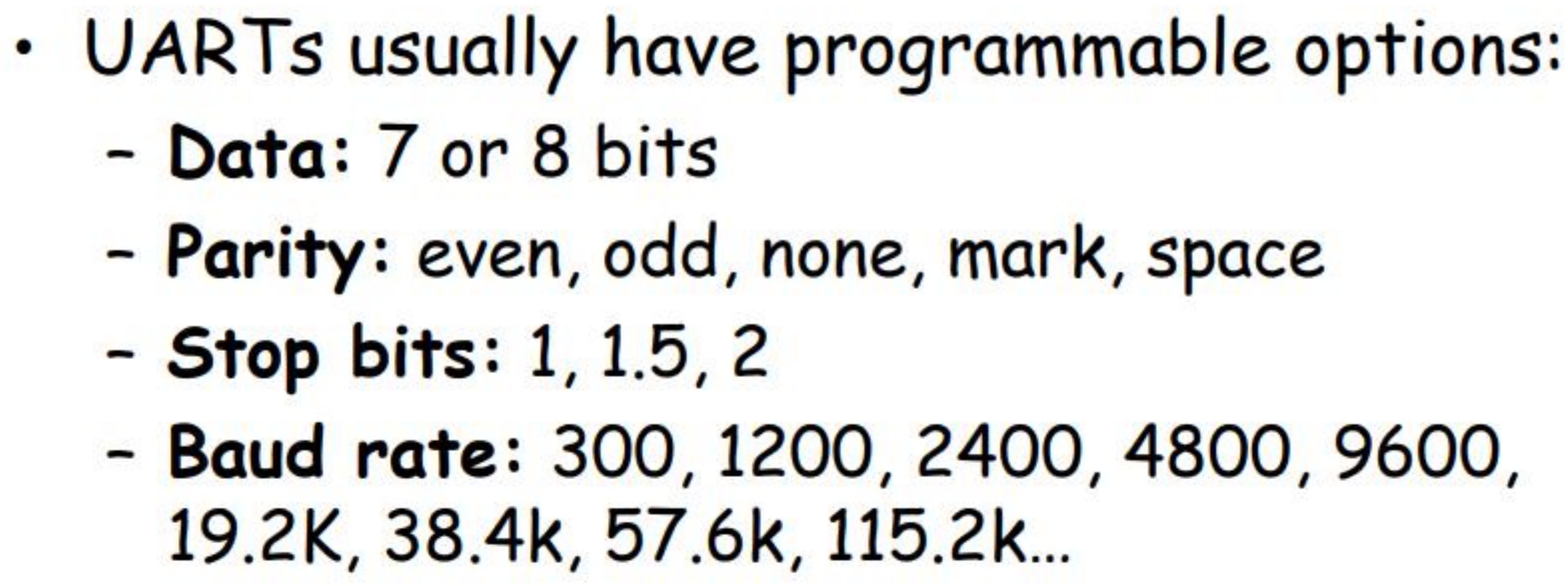


Receiver resynchronizes on every start bit.
Only has to be accurate enough to read 9 bits.

UART Receiving Stop Bit

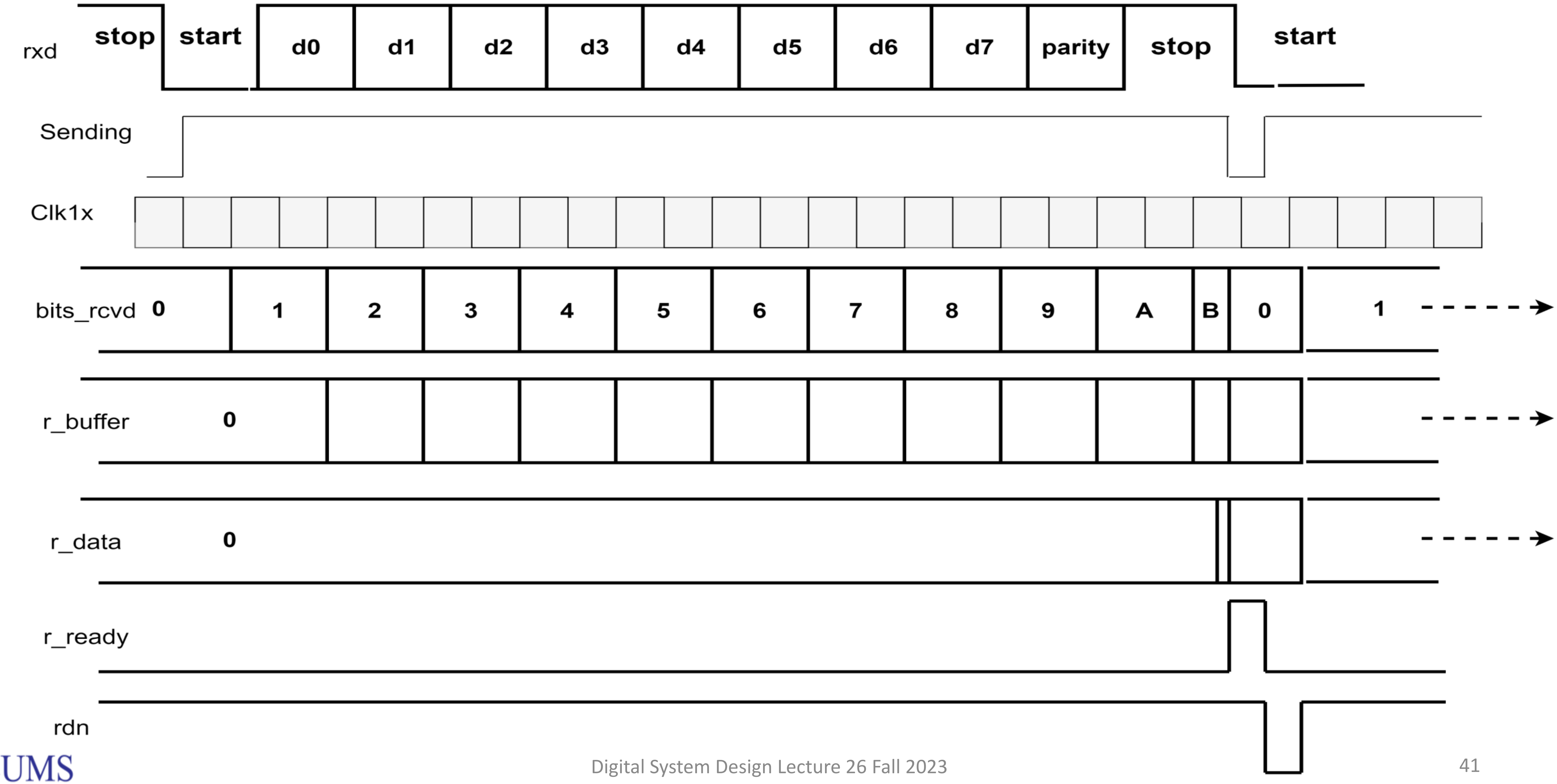
- Receiver also verifies that stop bit is '1'
  - If not, reports "framing error" to host system
- New start bit can appear immediately after stop bit
  - Receiver will resynchronize on each start bit

UART Receiver Configuration

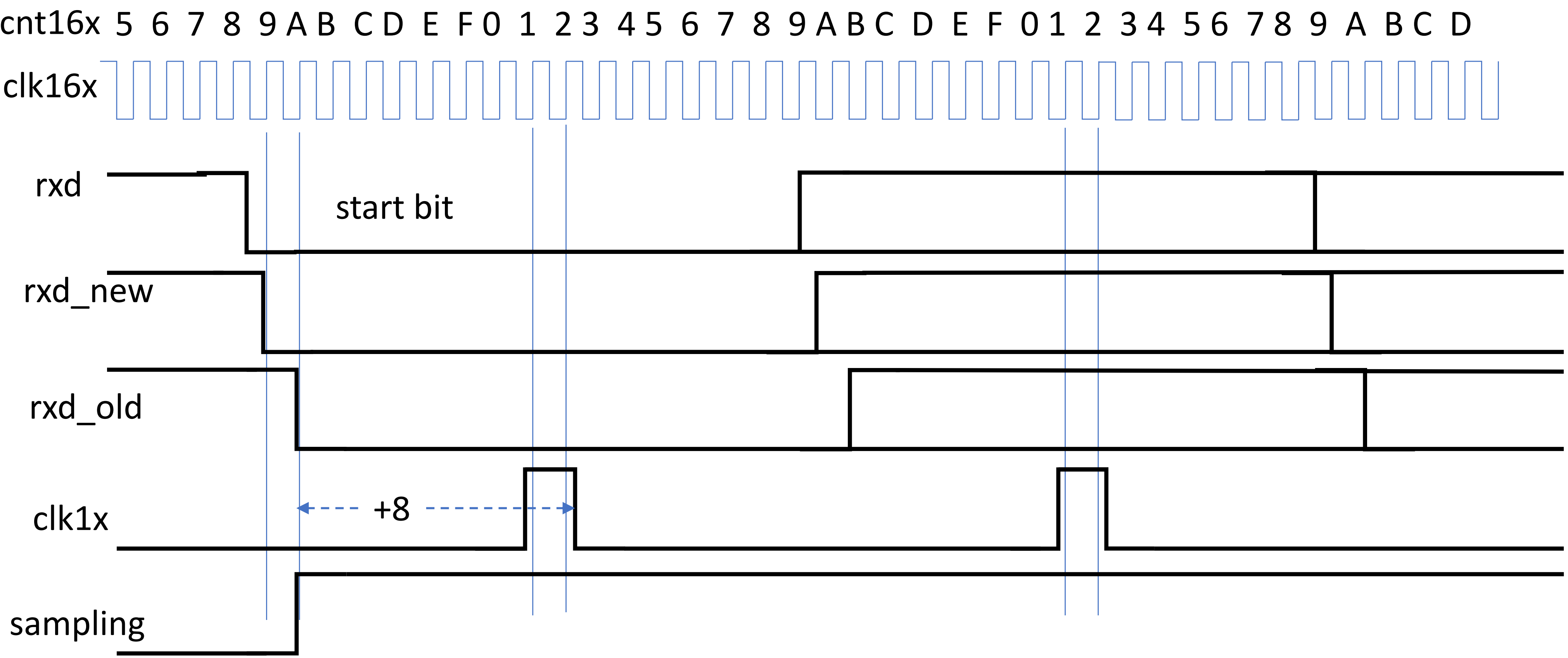


- UARTs usually have programmable options:
  - **Data:** 7 or 8 bits
  - **Parity:** even, odd, none, mark, space
  - **Stop bits:** 1, 1.5, 2
  - **Baud rate:** 300, 1200, 2400, 4800, 9600, 19.2K, 38.4k, 57.6k, 115.2k…

Sampling Clock of UART Receiver



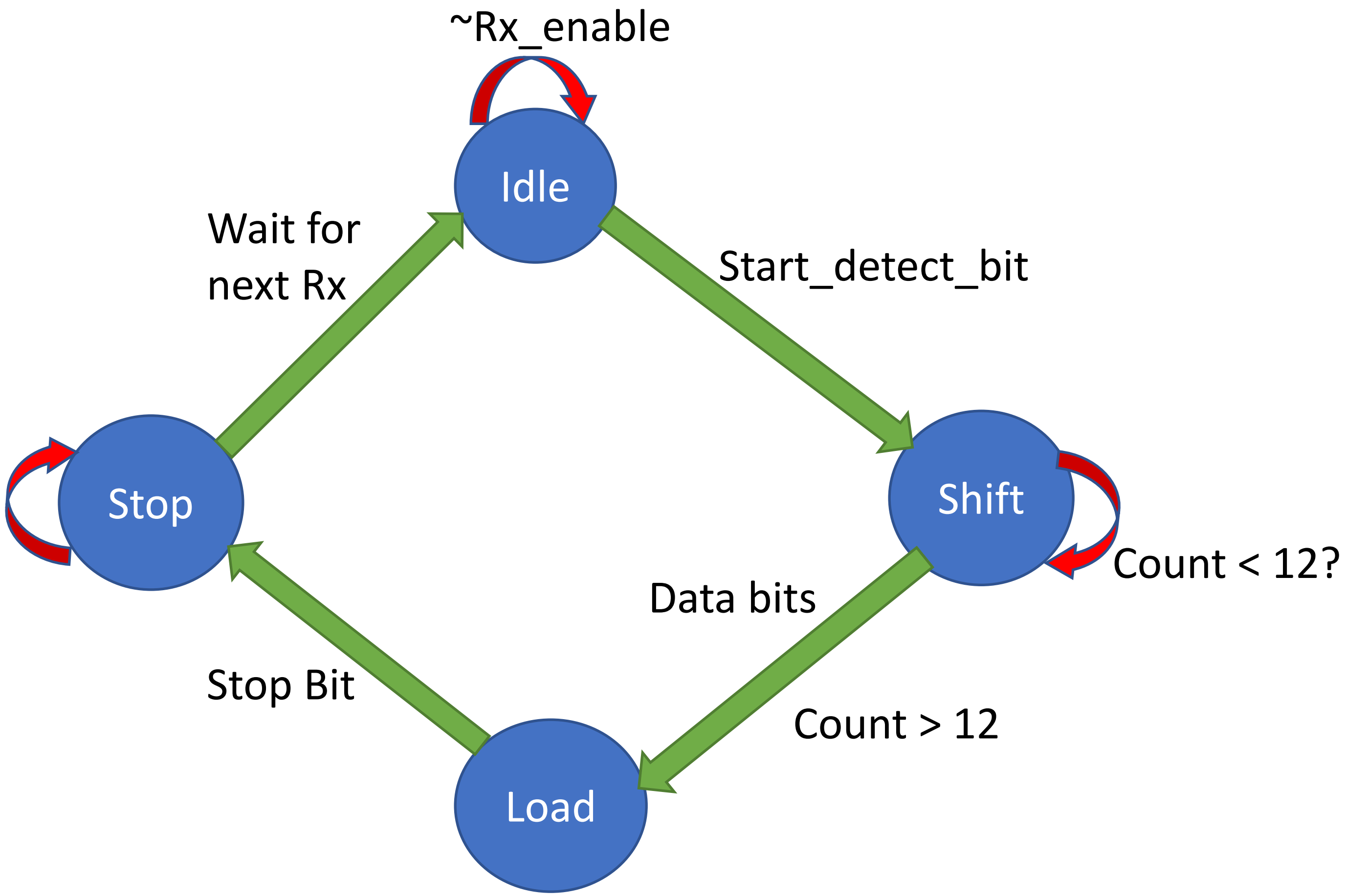
UART Receiver Timing Diagram


cnt16x 5 6 7 8 9 A B C D E F 0 1 2 3 4 5 6 7 8 9 A B C D E F 0 1 2 3 4 5 6 7 8 9 A B C D
clk16x
rxd
start bit
rxd_new
rxd_old
clk1x
+8
sampling

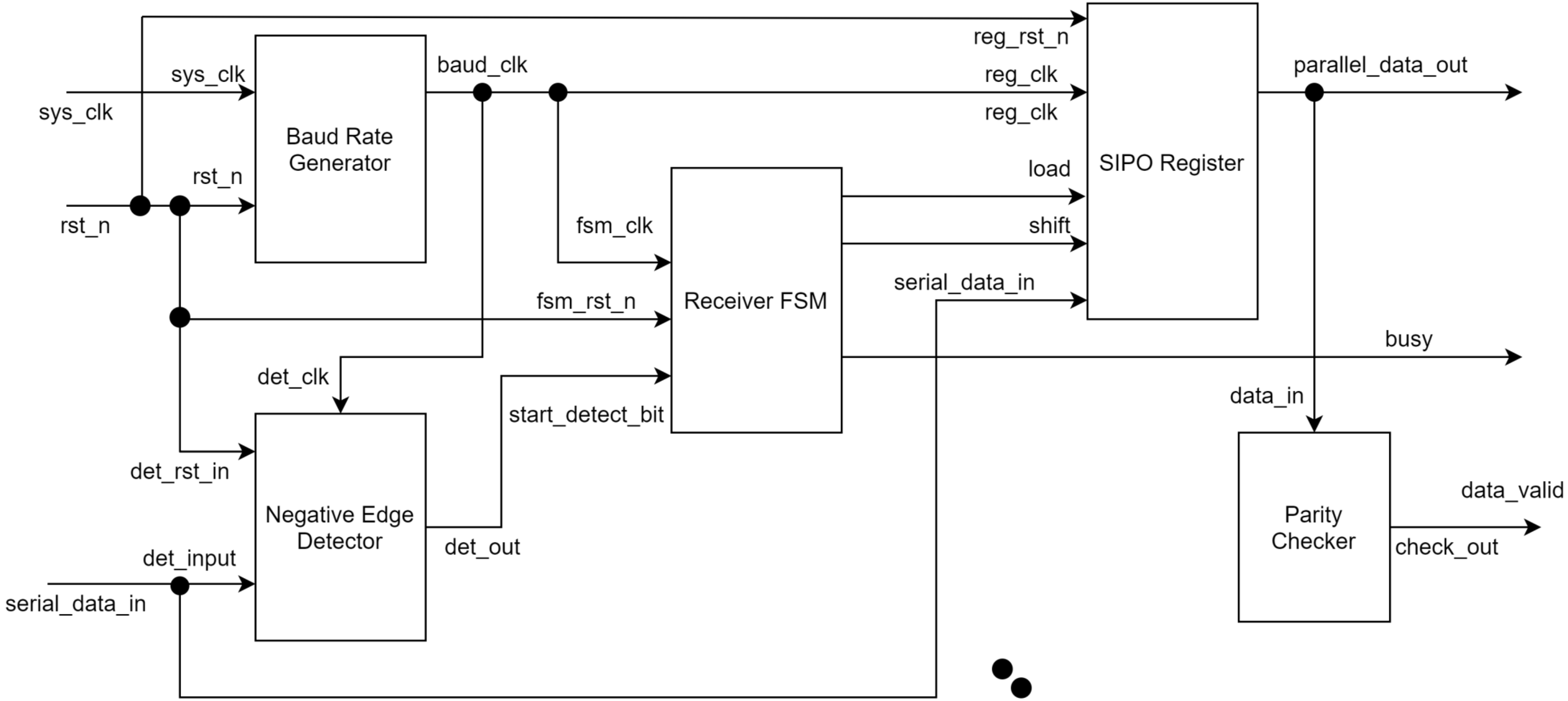
UART Receiver Starting details

- Clk16x is 16 times baud rate
- A 2-bit shift register saves input signal rxd
- When contents of this reg are 10 (rxd_old=1, rxd_new=0) then a falling edge is detected. We let sampling = 1 to sample rxd.
- A 4-bit counter cnt16x is used to generate a sampling pulse clk1x.
- During sampling=1, we let the rising edge of clk1x to appear at about centre place of one rxd bit; and the width of the clk1x pulse to be one clk16x clock cycle.

FSM for UART Receiver



UART Receiver Architecture



UART Errors

Overrun Error

An "**overrun error**" occurs when the UART cannot process the byte that just came in before the next one arrives. The host processor must service the UART in order to remove characters from the buffer. If the host processor does not service the UART and the buffer becomes full, then ***Overrun Error*** will occur.

Framing Error

A "**Framing Error**" occurs when the designated "start" and "stop" bits are not valid. Start bit acts as a reference for the remaining bits. When the "stop" bit is expected if the data line is not in the expected idle state a ***Framing Error*** will occur.

Parity Error

A "**Parity Error**" occurs when the number of "active" bits does not agree with the specified parity configuration of the UART.