Serial Interfacing

→ An introduction to serial interface
→ send one bit at time
→ Many serial protocols - CAN ✓, UART
ETHERNET ✓
I2C ✓
SPI ✓
RS232 ✓

→ UART - universal asynchronous receiver/transmitter
→ busy-wait to synchronize with hardware

⊙ I/O synchronization:

→ Performance measures: (goodness of your system)

(i) Latency: ✓

→ time required when I/O device indicated a service and time when service is initiated.

a) Hardware delay ✓
b) software delay ✓

→ for an input device, software Latency is time between new input data ready and software reading data

→ for output device, delay from output device being idle and software giving the device new data to output.

→ for example: ADC conversion (DAC)
software latency ∃ conversion is supposed to started & conversion actually get started.

⊙ In control system time betn when controller is supposed to get started & when it actually start.

ii) Real time -
A system that can guarantee worst case latency
→ soft real time & Hard real time. Δt = time bound

iii) Throughput or bandwidth -
Maximum data flow in bytes/second that can be processed by system.
limited by I/O ✓
limited by software ✓
Δt ⇒ cons. effect
Δt ⇐ bound. error fatal

iv) priority - ✓
→ order of service when two or more requests are made simultaneously.

→ So, the purpose is allow micro-controller to interact with it's I/O devices.

→ So, there are five way's to synchronize software with Hardware (Micro-controller with I/O)

i) Blind cycle :

computer, devices (LCD).

→ Bridge the speed mismatch between two devices.

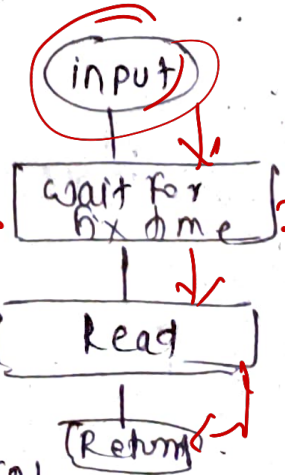
if A speed > B
—→ so miscommunication
so there is need to synchronization

* Blind cycle: - ✓

→ A has to be aware of B speed

→ problem = B speed may vary

Hence slow communication protocol.

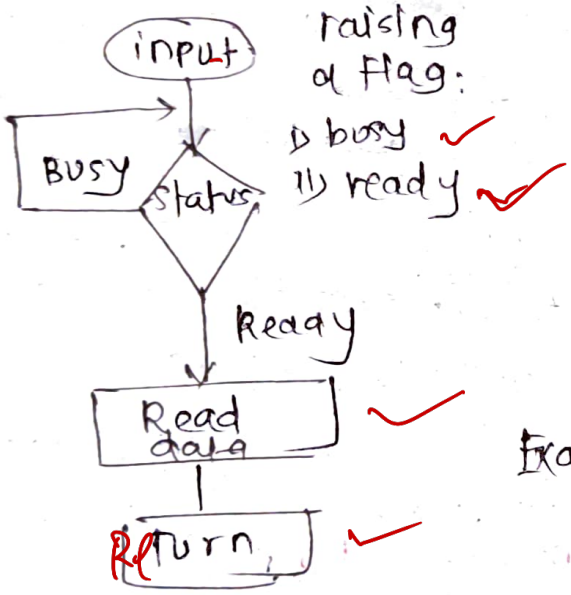


for blind time Δt it waits

→ but for the devices which are not sophisticated we can implement by looking at datasheet of devices ⇒ Δt =

→ For Example :-
LCD display character ; which is ASCII
LCD send.
Wait for 37 µs ... (From datasheet)
& we sure of transmission is complete :

(ii) Busy wait :

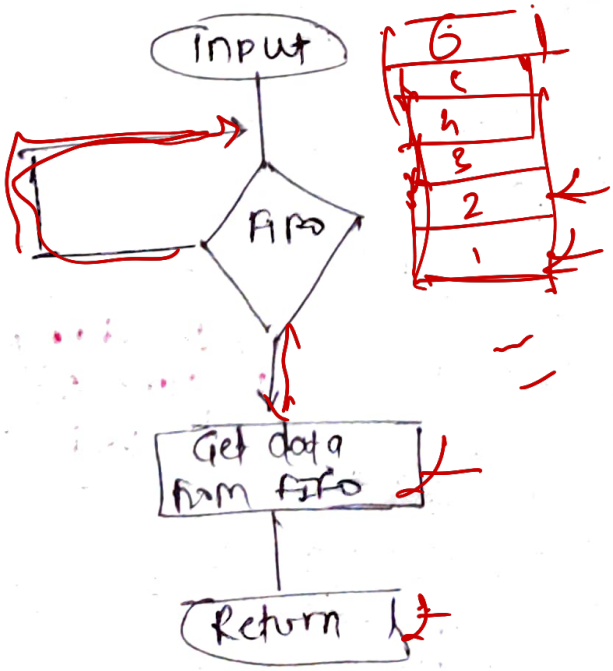


* repetadely checking Flag.

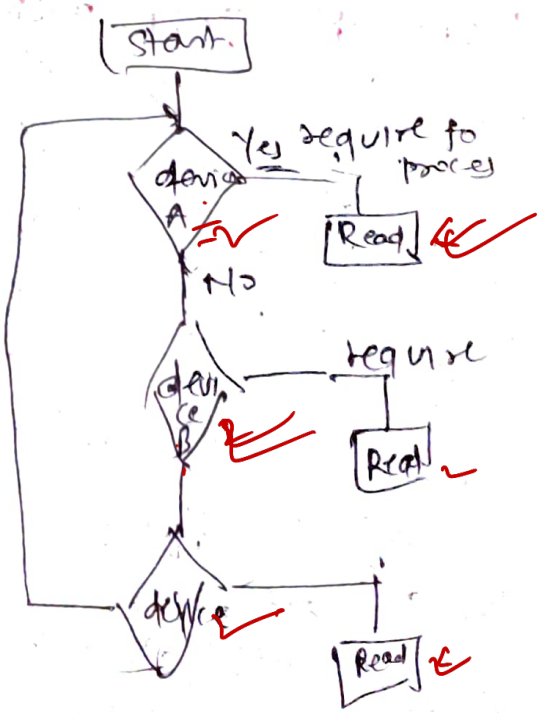
Send data
Check Flag = repetadely
↓
IF Flag set
↓
send another data.

Example :- UART

(iii) FIFO.

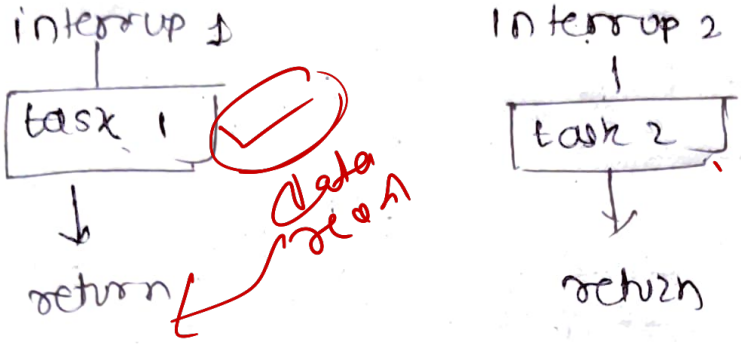


(iv) Polling



v) interrupt :

→ uses hardware to cause special software execution



vi) DMA :

→ Direct memory acess, transfer data directly from memory.

→ With an input device, Hardware will request a DMA transfer when input device has new data

→ DMA reads data & save it in memory without software knowledge

→ With an output device, hardware will request a DMA transfer when output device is idle.

→ DMA gets data from memory & then write to device.

→ When to use :-

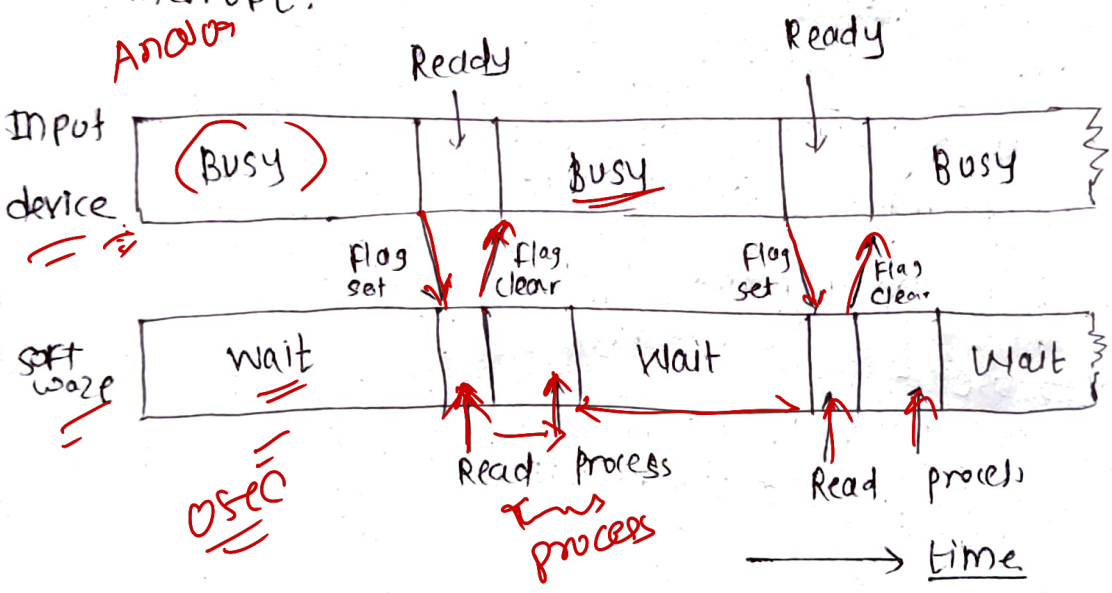
→ high speed data - application system

→ situation where high bandwidth & Low latency required.

- Hardware being in three states
  - idle - disabled or inactive
  - Flag (ii) busy - doing or getting some process (0)
  - Flag (iii) ready - ready to receive (1)

→ Flag setting:-

- Flag = 1 ... when hardware component is complete = read
- read this Flag by software to determine if device is busy or ready
- software can clear Flag, when software component is complete
- Flag can be used for hardware triggering event for interrupt.



I/o bound: →

→ Time for the software to read the data and process the data is less than the time for input device to create new data. This situation is I/o bound

→ Means, the bandwidth is limited by I/o hardware

CPU bound :-

→ If the input device were faster than the software then software waiting would be zero. This situation is CPU bound

→ bandwidth is limited by speed of execution software

Real time system :-

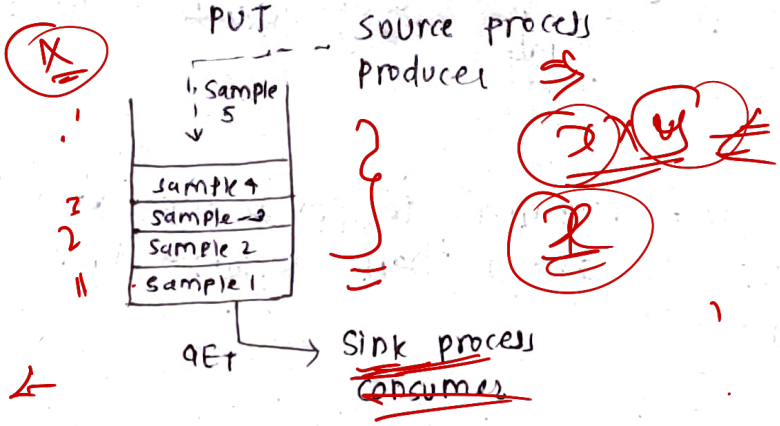
→ bandwidth depends on both input and output

→ data vary over time

→ I/O channels can sometimes be I/O bound but other times could be CPU bound.

① FIFO :-

→ Store & buffer data in FIFO.



→ separates the generation of data from consumption of data, Hence very efficient & hence prevalent in I/O communication.

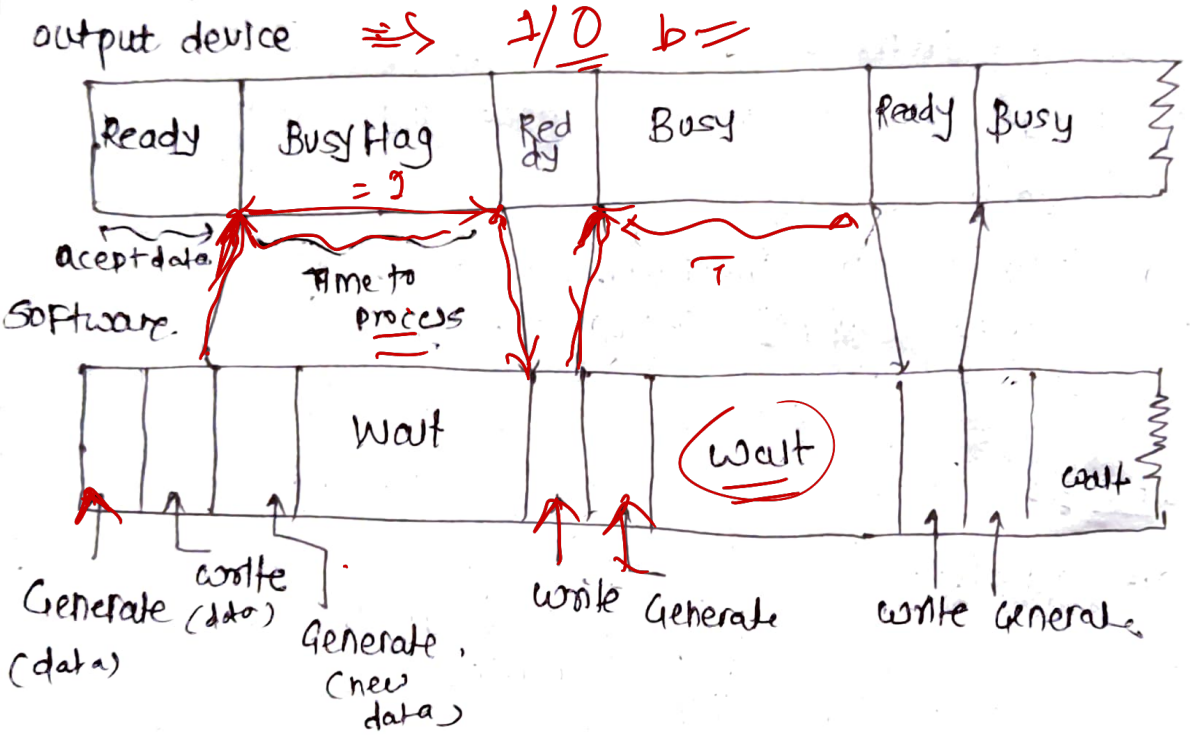
→ used when there is increase or decrease in rates at which data is produced or consumed

→ maintains the order of data

Ex:- busy-wait is unbuffered because hardware & software must wait for transmission of each piece of data.

Ex: interrupts are buffered, because the system allows the input device to run continuously, filling a FIFO a FIFO with data as fast as it can.

* for an output device:



→ As time required to process the data for output device is greater than software time to generate data

Universal Asynchronous Receiver Transmitter (UART)

one of the most coldely used protocol for device to device communication.

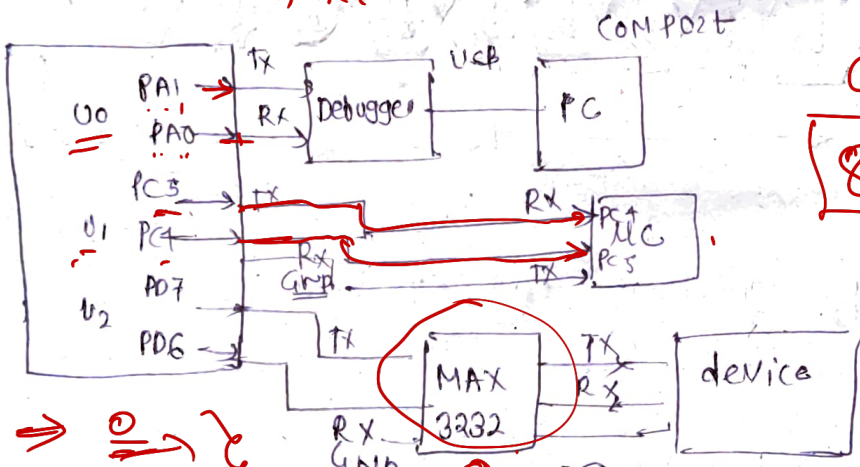
transmitt & Receive serial data - bit by bit

* communication protocol's plays important role in organizing communication between devices

Asynchronous - no clock signal to synchronize the output bits from transmitting device going to receiving end.

8 → Tx
→ Rx
U0 →

TM4C1234GHPM
LP
Cortex M4
8 UART →
PC
MC
Sensor
Bluetooth
Gen



0 ⇒ 0
1 ⇒ 3.3V

RS232

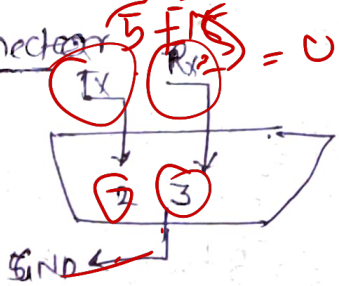
-5to-15 = 1

50 feet

Bluetooth

Vcc
GND
Rx
Tx

⊙ Dog connector



- Baud rate :-

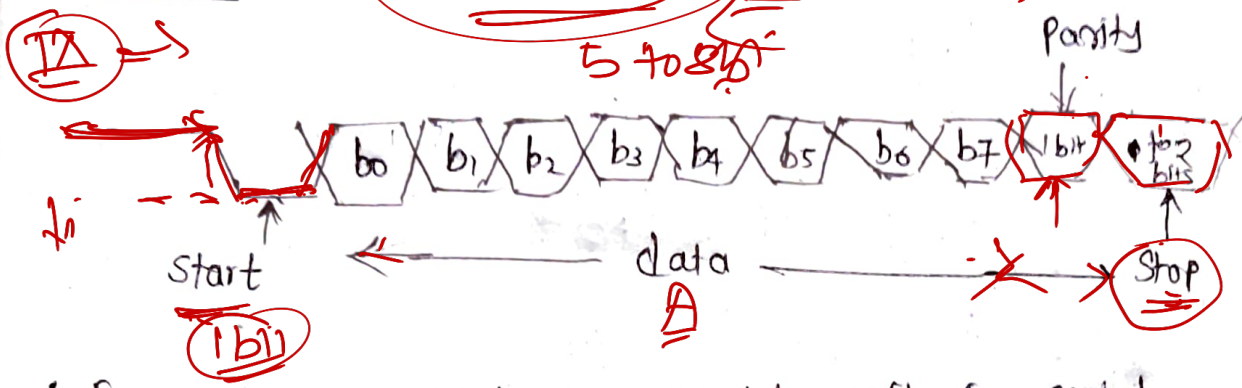
→ The total no. of bits transmitted per second is called baud rate.

→ ~~Baud rate~~ = bit time = $\frac{1}{\text{baud rate}}$

→ Each UART has baud rate control register

9800, 115200 — Register = 5 to 8 bit

- FRAME :-



- A frame is the smallest complete unit of serial transmission.

- START bit :- (1 bit)

→ Initially transmission line held high = when no transmission

→ To start tranfer transmission line is held low for 1 clock cycle.

- Data frame :- (5 bits to 9 bits) 5 to 8 data bits

→ actual data

→ if no parity bit used range = 9 bits.

- Parity bit :-

→ way for receiving UART to tell if any data has changed during transmission.

→ changing by : Electromagnetic radiation, mismatch baud rates, long-distance data transfer.

. stop bit -

→ to singnal END of Data packet.

→ drives the data from Low Voltage to high.

① Asynchronous communication :-

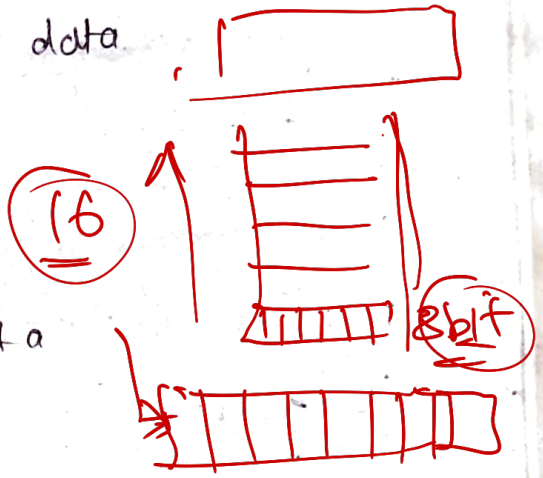
Transmitter side :- →

i) Data register to write data

ii) 16x8 depth FIFO to store data

iii) shift register 10 bit.

UART

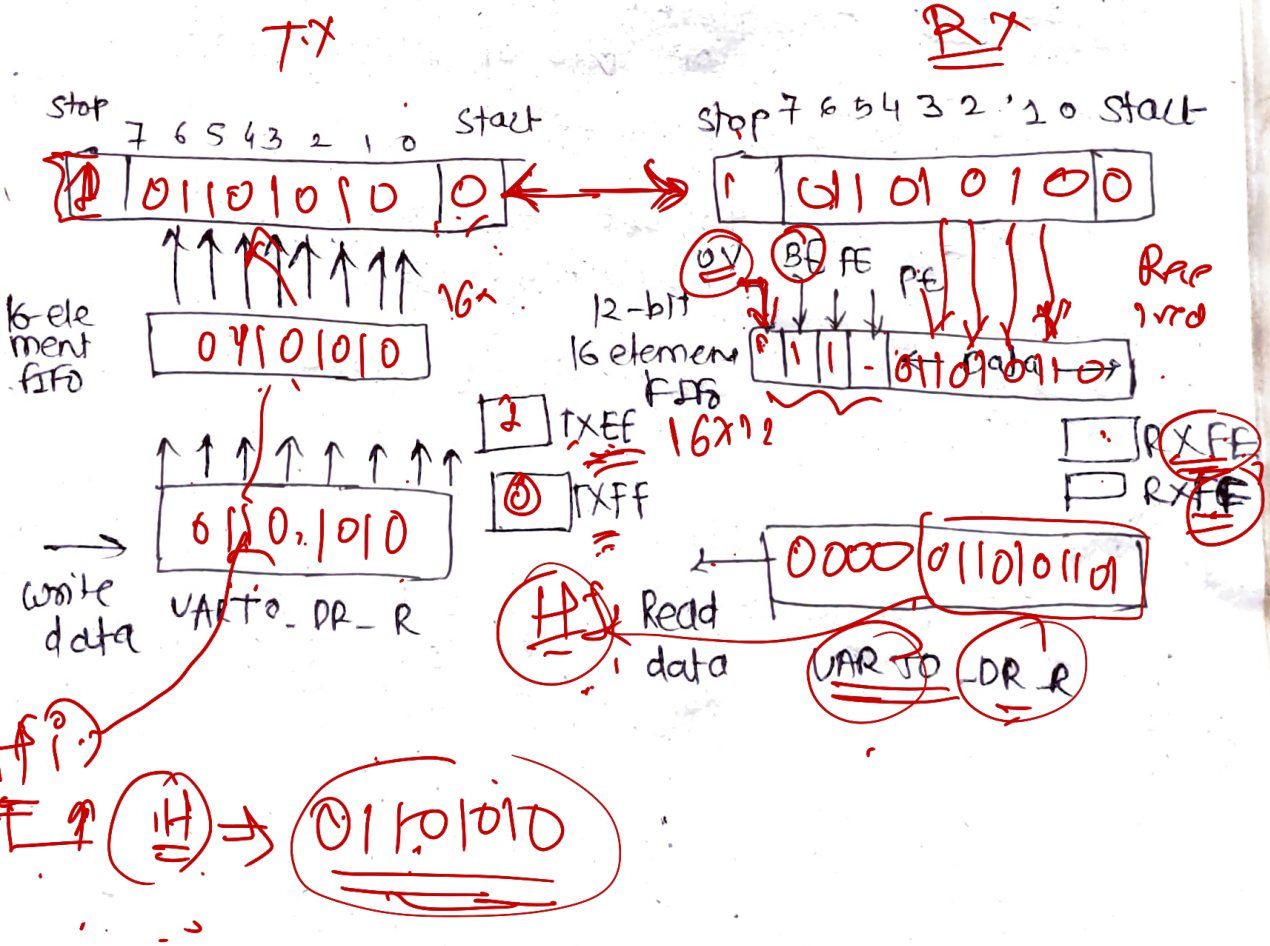


Receiver side :- →

i) Data register to read data

ii) 12 bit FIFO →

iii) 10 bit shift register



TM4C UART :- (API Registers (low level prog)

-> 8 UARTs

-> for Pins refer datasheet ✓

-> for exact register address refer header file

* Initialize the UART :

a) Turn on UART clock

RCGC UART => refer datasheet

=> 0X01

b) Turn on PORT C

RCGCGPIO => refer datasheet

=> to give clock to port C

=> 0X01

c) Tx pin & Rx pin As digital pin,

DEN_R

=> 0X03

d) Alternate function on Pins

AFSEL_R ... enable Alternate function on Pins of selected port

=> 0X03

SYSCTL_PORTX_AFSEL_R

PCTL_R ... to set Alternate function 4 bits Allocated for each port

=>

AMSEL ... disable analog mode

e) if you are using FIFO

UART0_FR_R = 1 ... busy flag will set while transmitter still has onsent bit even if transmitter is disabled

3 2 1 0

TXFF RXFE BL

0 ... when transmit Fifo is empty & last stop bit is sent

9 8

8) UART0_CTL_R — RXE | TXE | UARTEN 0

- control register contains bits to turn on UART

TxE = transmitter enable
RxE = Receiver enable
UARTEN = 1

} 1 to turn on UART

However UARTEN = 0 while initializing ~~You~~ UART.
1

9) Baud rate: UART0_IBRD_R & FBRD_R

UART0_IBRD_R = Value — integer part
UART0_FBRD_R = Value — fractional part

10) UART_LCRH_R = value

- will define frame structure

Unless you dont put value in these register the baud rate registers will not take any effect.

10) Transmit (char data)

while (UART0_FR_R & 0X0020 != 0);

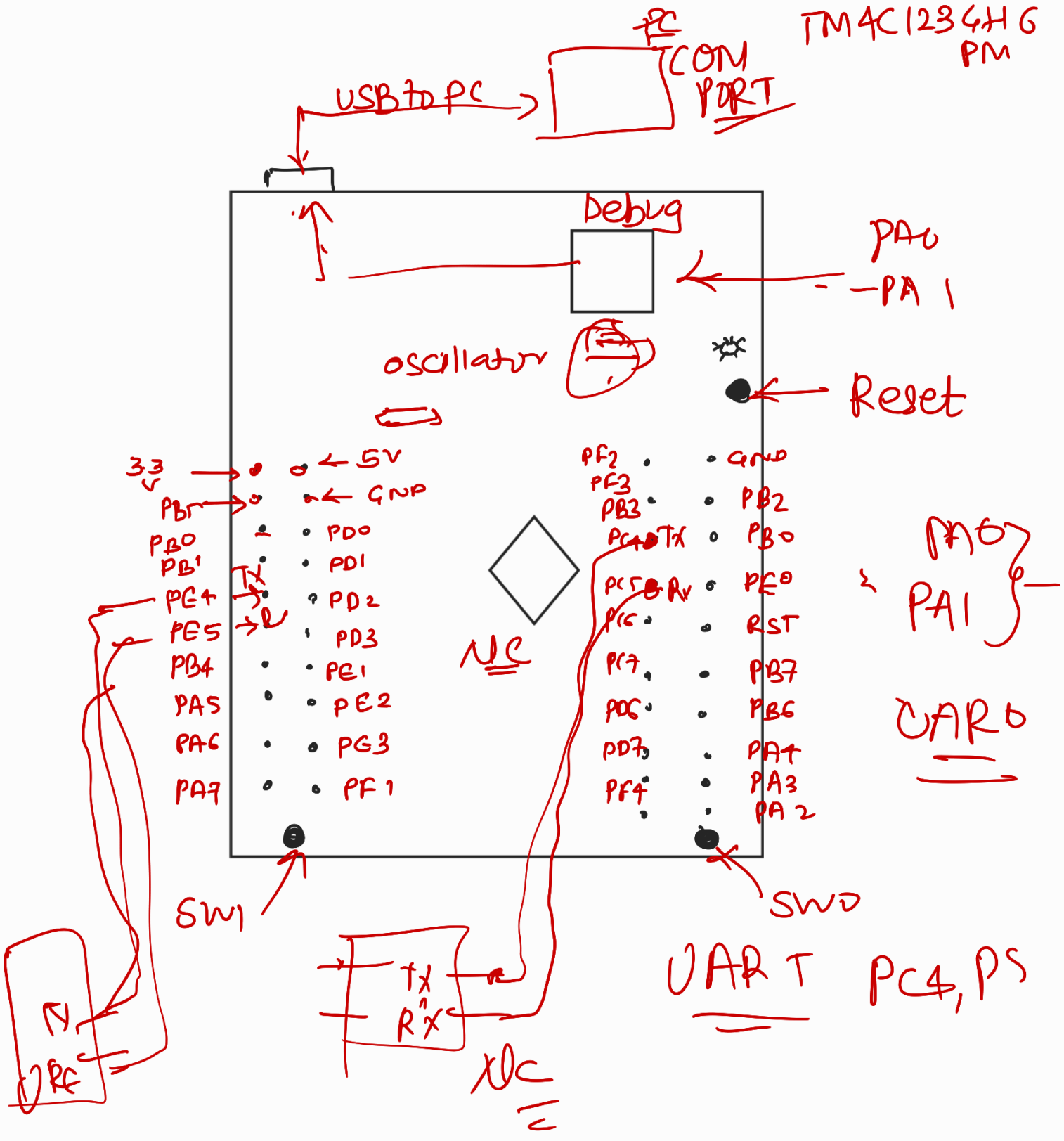
TXFF is 0

UART0_DR_R = data

11) Receive

while (RXFE is 0)

return (characters in UART0_DR_R);


PC
COM PORT
USB to PC
TM4C123GH6PM
Debug
PA0
PA1
oscillator
Reset
3.3V
5V
GND
PB5
PB0
PB1
PE4
PE5
PB4
PA5
PA6
PA7
PD0
PD1
PD2
PD3
PE1
PE2
PE3
PF1
TX
RX
NC
PF2
PF3
PB3
PC4 TX
PC5 RX
PC6
PC7
PD6
PD7
PF4
GND
PB2
PB0
PE0
RST
PB7
PB6
PA4
PA3
PA2
PA0
PA1
UART0
SW1
SWD
TX
RX
GND
VCC
UART PC4, PC5

80MHz ⇒ 115200

$$BRD = \frac{\text{System clock}}{16 \times \text{Baudrate}}$$

$$BRD \Rightarrow \frac{80 \times 10^6}{16 \times 115200}$$

$$\Rightarrow 43.402$$

IBRD = 43

FBRD = (0.402 × 64 + 0.5)

⇒ 26