5.1 Input/Output (I/O) Interface

1) With reference to the computer system diagram (Figure 1) in the case study notes,

(a) What are the type (input, output or bi-directional) of processor pins you would connect each of the module's data signal to?

Table 1a

No.	Module	Type (Input, Output or Bidirectional)
i.	Buttons	Input
ii.	Wifi	Bi-Directional
iii.	Touch Screen Controller	Input/Bi-Directional**
iv.	System Memory	Bi-Directional
v.	Display	Output

** For 'smart' modules that allows configuration, e.g. Processor is able to write to the Touch Screen Controller to configure the screen size and scan rate etc.

(b) Given the following interface requirement, select the appropriate module from the device information list in the case study notes to connect to the processor.

[Suggested Solution]

When choosing which module to connect, select based on the two criteria that allow modules to communicate with each other, i.e. they have to be compatible in Electrical Signal level and Communication Protocol.

i. Wifi.
   a. WIFI0010AC
   b. 3V electrical interface. VOH, VOL, VIH and VIL compatible with Processor SPI interface.

ii. Touch Screen Controller
   a. TS002UART
   b. 3V electrical interface. VOH, VOL, VIH and VIL compatible with Processor UART interface.

iii. System Memory
   a. DRAM0002-16M16, SRAM0002-2M16, NOR0001-1M. Actual choice depends on other requirement such as memory speed (read/write), cost etc.
   b. 3V electrical interface. VOH, VOL, VIH and VIL compatible with Processor Parallel Bus interface.
   c. 16 bit data bus compatible with Processor 16 bit parallel bus interface. We can also use 2X 8-bit memory if required.

Table 1b

No.	Module	Interface Type
i.	Wifi	SPI
ii.	Touch Screen Controller	UART
iii.	System Memory	Parallel Bus

(c) For each of the interface shown in **Table 1b**, is the data transfer serial or parallel?
[Suggested Solution]

Wifi. SPI Bus => 1-bit data in each direction (MOSI, MISO) => Serial Bus. Full duplex.

Touch Screen Controller. UART interface => 1-bit data in each direction (TX, RX) => Serial Bus. Full Duplex.

System Memory. Parallel Bus => multiple data bits transferred simultaneously => Parallel Bus.

(d) Describe the difference between synchronous and asynchronous interface. Are the interfaces shown in **Table 1b** asynchronous or synchronous? Explain.
[Suggested Solution]

Synchronous – has a separate clock signal to synchronize data transfer.

Asynchronous – has no clock signal. Data transfer is based local clocks at transmitter and receiver. Some sort of synchronization bits/pattern is typically used to align the two clocks.

i. Wifi – SPI interface. This interface has a common clock line (SCLK) to synchronise the data transfer. => synchronous
ii. Touch Screen Controller – UART interface. There is no common clock line between transmitter and receiver, all sync information are embedded in the data line signals. => Ansychonous.
iii. System Memory. Parallel Bus Interface. A Strobe signal is used to tell the receiver when to latch in the data on the data bus. => Synchronous.

(e) With reference to the SPI wifi module information in the case study notes, what is the maximum speed that data can be transferred between the wifi module and the processor?

[Suggested Solution]

Note: SPI bus will transfer one bit of data at each clock edge (rising or falling).

Wifi module (WIFI0010AC)

- Max SPI Clock on Wifi Module = 50Mhz
- Max Processor SPI clock = 50Mhz
- => 50Mbps transfer rate (max).

Note: When determining the data transfer rate, always choose the slowest module in the transfer chain. E.g. if the max SPI clock on Wifi Module is 40 Mhz instead, then the max speed that data can be transferred will be 40Mbps.

(f) With reference to the device information of the system memory you have chosen in 1(c) above, what is the maximum data transfer rate achievable between the system memory and the processor? Assuming that data on the parallel bus gets transferred on each rising edge of the data strobe.

[Suggested Solution]

Note: Need not discuss all examples below, just need to touch on one of them, the rest are applicable under the same concept. Difference from serial transfer is the multiple bits transmission per clock edge, hence the multiplication factor applied.

**Slowest device in the chain determines the transfer rate of the system.

DRAM (DRAM0002-16M16)

- Max data strobe rate supported by DRAM = 100Mhz
- Max Processor Parallel bus strobe rate = 200Mhz
  ⇨ Max supported data strobe rate on this link = 100Mhz
- 16-bit interface => 16 bits data transferred with each strobe
  ⇨ 100M * 16 = 1600Mbps.

SRAM (SRAM0002-2M16)

- Max data strobe rate supported by SRAM = 200Mhz
- Max Processor Parallel bus strobe rate = 200Mhz
  ⇨ Max supported data strobe rate on this link = 200Mhz
- 16-bit interface => 16 bits data transferred with each strobe
  ⇨ 200M * 16 = 3200Mbps.

NOR Flash (NOR0001-1M)

- Max data strobe rate supported by DRAM = 10Mhz
- Max Processor Parallel bus strobe rate = 200Mhz

⇨ Max supported data strobe rate on this link = 10Mhz

- 16-bit interface => 16 bits data transferred with each strobe

⇨ 10M * 16 = 160Mbps.

5.2 Data Transfer

2) **Fig. 2** shows the logical waveform of an asynchronous data frame.

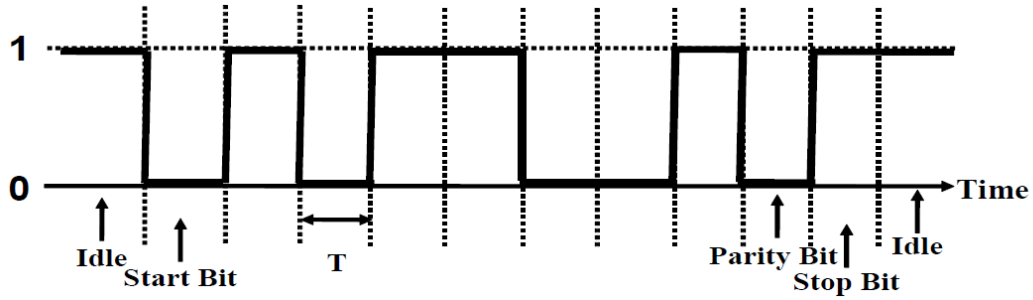

Fig. 2

Answer the following question with reference to **Fig. 2**:

(a) Specify the 7-bit ASCII character that is transmitted (LSB is transmitted first).

[Suggested Solution]

7-bits: 1001101_2 → 0x4D

ASCII character → M

(b) Assume there are no errors in the transmitted waveform, state whether even or odd parity is being used.

[Suggested Solution]

Even parity. Data+Parity bits → even number of '1's.

(c) Assume two of the bits received in the character are erroneous; can the receiver detect the error? Can you extend the example to determine the limitations of parity checking?

[Suggested Solution]

No, the parity of the data sequence remains unchanged if two bits are erroneous. Hence, the parity check will fail. In general, if there is an even number of errors bits, parity checking fails.

(d) If 7-bit ASCII characters are transmitted continuously in the format shown (with no idle periods between the frames) at a baud rate of 9600, calculate the following:

i. The value of **T** in **Fig 2**.
ii. The data transfer rate of this serial interface in characters per second (cps).

[Suggested Solution]

(i) T = 1/9600 = 0.104 ms.

(ii) 1 packet consists of Start, Data, Parity and Stop Bits.
9600 bits per second / 10 bits (1 packet) = 960 characters per second.

(e) Re-compute the data transfer rate (cps) if the transmission does not use any parity bit. Compare the results with those obtained in Q3d(ii), and state your observation(s).

[Suggested Solution]

9600 bits per second/9 bits (1 packet) = 1066 characters per second.

Throughput is higher, but the integrity of the data packet cannot be ascertained.

Which option to choose will depend on the signal line condition (noisy?) between the transmitter and receiver.

(f) Assume the baud rate of the transmitter is 4800, but the baud rate of the receiver is configured as 9600. Based on **Fig. 2**, determine, if any, the ASCII character(s) that will be received.

[Suggested Solution]

Possible multiple-sampling scenario

4800 baud rate: 7-bits: 1001101_2 → 0x4D

Original Frame							
S	D (lsb to msb)	P	E				
0	1011001	0	1				

Oversampled Frames (2x)							
Frame #1				Frame #2			
S	D (lsb to msb)	P	E	S	D (lsb to msb)	P	E
0	0110011	1	1	0	0001100	1	1

Frame #1: 0x66 → lowercase letter ‘f’

Frame #2: 0x 18 → special character ‘CAN’

Parity errors will be flagged for both frames (odd numbers of ‘1’s received).

[OPTIONAL TUTORIAL QUESTIONS]

**Below questions are optional. Not necessary to be covered during tutorial

3) There has been a trend of 'serialization' of interface bus standard. E.g. USB replacing Parallel Port Interface, SATA replacing IDE in HDD.
   (a) What are the advantages that Serial Bus has over Parallel Bus interface that enticed industry player to move in this direction?
   (b) What are the scenarios in which Parallel bus will still be preferred over Serial bus? What are the design considerations that need to be put in place in such cases?

   [Suggested Solution]

   (a) Serial Bus has *less* data lines compared to Parallel Bus.
   -> Less issue with signal skew and less probability of being influenced by crosstalk.
   -> Less bulky so easier to route PCB trace and cabling (SATA vs IDE Cable).
   -> Even though data is transferred one bit at a time, modern serial standards are able to achieve relatively high data transfer speed (USB3.0 raw speed is up to 5Gbps). This is achieved through good design and testing conformance/certifications.

   (b) Parallel bus is still commonly used when sustained high speed is needed. E.g. interfacing to DDR SDRAM (DDR3 SDRAM data rate scales up to beyond 20GByte/s). But designing these high speed parallel buses require deep knowledge of electrical circuit behaviour, so as to reduce the effect of signal skew and crosstalk. And typically, the length of the parallel bus is very short (order of cm rather than metre), with lots of consideration on grounding and shielding. High speed circuit design is almost an art.

4) In Q3(f) above, a wrong sampling result is observed when a wrong baud rate is used. This can actually be used to implement auto baud rate detection. Briefly describe how this can be done.

   [Solution]

   A Carriage Return has the ASCII value of 0x0D and would be sampled as different value with different baud rate assumption. If the receiving device knows that the transmitting device sends a 0x0D, then it'll be able to derive the baud rate based on the value sampled. Below is an example for the case when receiving device is sampling at 115200 Baud. If the transmitting device is sending 0x0D at 38400 baud, the data received will be 0x1C. Note that other Ascii characters can also be used as long as that value yields different result when sampled under different baud rate.

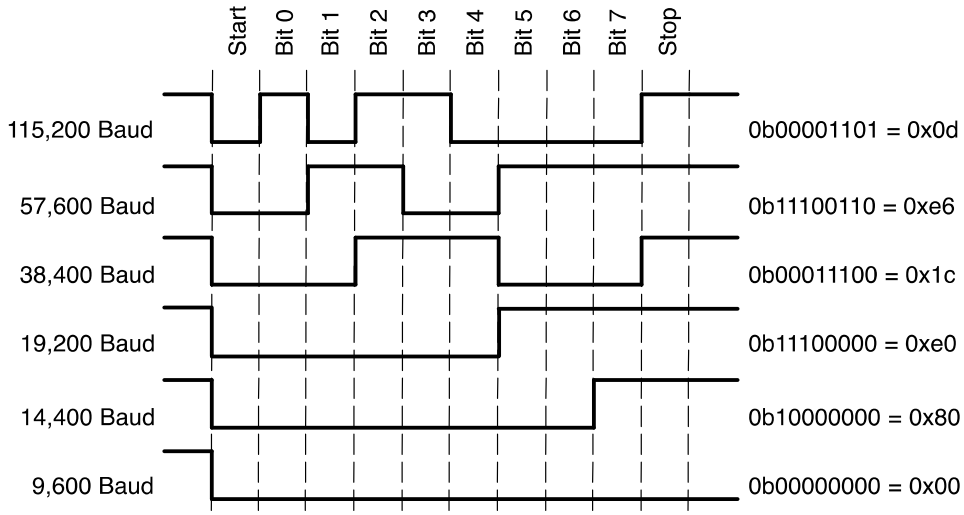

Figure 3. Bit Patterns for Baud Rates Between 115K and 9,600

If data sampled is 0x00. That means baud rate is 9600 or below. In this case, the receiver baud rate has to be adjusted to 9600 and resample the UART data. Below is the chart showing the different values that will be sampled by virtue of the transmitting baud rate.

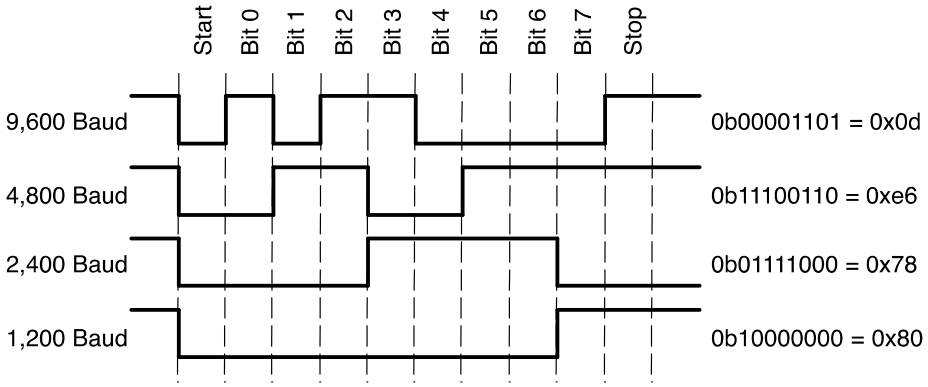

Figure 4. Bit Patterns for Baud Rates Between 9,600 and 1,200