

**Overview**

In this unit your students should:

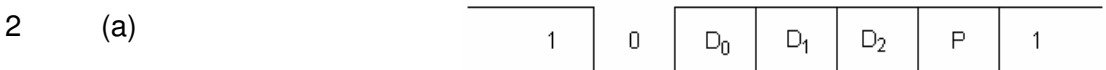
- appreciate the need for start and stop bits in serial transmission of words
- be able to relate the bandwidth of a serial transmission to its bit rate
- understand the operation of serial transmitter and receivers
- learn about the different pieces of information carried by a packet
- understand the protocol for transmitting packets in an ethernet system

This should not require more than 5 hours of class time.

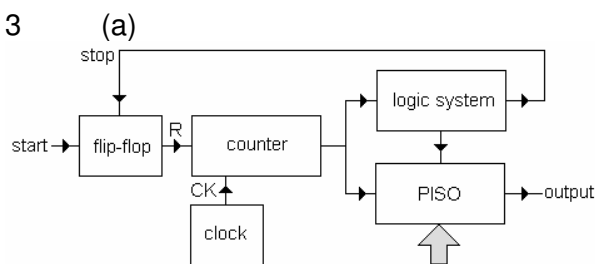
Hour	Suggested Activity
1	<p>Launch your students straight into the <b>Serial Transmitter</b> practical. You might want to save time by programming the PICs in advance. Most should manage to have a working system by the end of the session. They may need help with step 5.</p> <p>Ask them to answer the questions on page 1 of the <b>Asynchronous transmission</b> exercises before the next session.</p>
2	<p>Discuss their answers to page 1 of the <b>Asynchronous transmission</b> exercises.</p> <p>Then let them continue with the <b>Serial Receiver</b> practical. As before, you might want to save time by programming the PICs in advance. They may need some help with step 4. Step 9 is extension work.</p> <p>Ask them to answer the questions on page 2 of the <b>Asynchronous transmission</b> exercises before the next session.</p>
3	<p>Discuss their answers to page 2 of the <b>Asynchronous transmission</b> exercises before getting them to answer the questions on page 3.</p> <p>Ask them to answer questions 1 and 2 on page 159 of the textbook before the next session.</p>
4	<p>Get all students to start the <b>Sending Packets</b> practical. Let them program the PICs themselves, as the programs are quite short.</p> <p>As they reach step 8, divide students into groups of up to eight and give each student in the group a different address CBA. Nominate one member of the group as the transmitter, leaving the other seven as receivers. You will need to ensure that all eight share a common 0 V line as well as a common data line.</p> <p>Students will have an opportunity to do step 8 in the next session.</p> <p>Ask them to answer questions 3 and 4 on page 159 of the textbook before the next session.</p>
5	<p>Students who need more time to finish the questions from the textbook should use this session to do so before completing the <b>Sending Packets</b> practical.</p> <p>In the meantime, students could continue with step 8 of the practical. Ultimately, each student in a group of eight could have a transmitter and a receiver, both connected to the same data line as the other seven students.</p> <p>Ask them to revise <b>Time Division Multiplexing</b> for a formal test next session.</p>

**Model Answers**

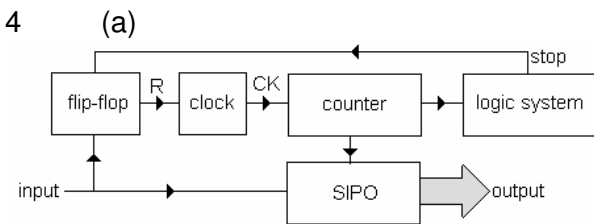
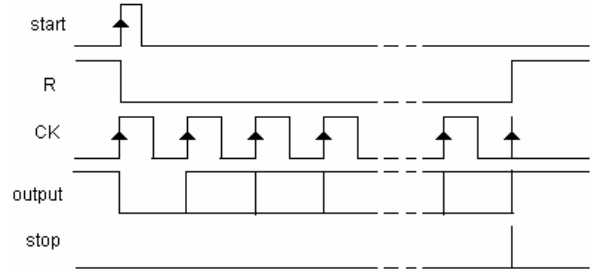
- 1 (a) The **destination address** tells each computer on the link which one is the intended recipient of the data in the packet. The **source address** tells each computer on the link which one of them is placing the packet on the link. The **data payload** is the information being transmitted (e.g. text, video, music ...). The **checksum** can be used to check that the data payload has not been corrupted in transmission.
- (b) A computer that wishes to place a packet on the line waits until the line has been quiet for 10  $\mu$ s. It then transmits the packet, one bit at a time, comparing the actual state of the line with what it should be. If the actual state does not match the required state, another computer is probably trying to transmit at the same time. Both computers stop transmitting immediately, wait a for a short random length of time and start the procedure all over again.



- (b) The stop bit leaves the line high so that when it is pulled low by the start bit the receiver knows that a fresh word is being transmitted.
- (c) The state of the parity bit is determined by the transmitter to ensure that there is an even number of 1s in the word PD<sub>2</sub>D<sub>1</sub>D<sub>0</sub>. This means that it can be used at the receiver to find out if the word has been corrupted in transmission as a change of any one bit results in an odd number of 1s.



- (b) A rising edge at **start** sets the flip-flop. This turns on the clock. The first clock pulse loads the word into the PISO register, and subsequent clock pulses place the bits, one after the other at the **output**. The logic system detects when the stop bit is placed at the output, pulsing **stop** high to reset the flip-flop and turn the clock off again.



- (b) A falling edge at the beginning of the start bit at the input sets the flip-flop. This sets the clock going, with rising edges at the SIPO register to latch each bit as it arrives at the input. When all the bits have been retrieved, the logic system pulses **stop** high to reset the flip-flop, the clock freezes and the received word is presented at **output**.

