Data Transfer Technique

polling:

Although the meaning of the data transmitted by the various process, operator and computer peripherals differs, there are many common features which relate to transfer of data from the interface to the computer.

A characteristic of most interface devices is that they operate synchronously with respect to the computer and that they operate at much lower speeds. This difference in speed would severely limit the speed of operation of the computer if it directly controlled the device; however, for maximum flexibility of operation program control is desirable. Operation in this way is known as ' **programmed transfer** ' and involves the use of the CPU. The alternative is direct memory access "DMA".

A major problem in data transfer is timing. It may be thought that under programmed transfer, the computer can read or write at any time to a device, i.e. can make an **unconditional transfer**. For some process output devices, (e.g. switches & indication lights) these would be connected to a digital output interface or for D/A convector, unconditional transfer is possible; they are always ready to receive data.

For other output devices; (e.g. printer & communications channels), which are not fast enough to keep up with are computer but must accept a sequence of data items without missing any item, unconditional transfer cannot be used.

The computer must always be sure that the device is ready to accept the next item of data, hence either a timing loop to synchronize the computer to the external device or **'conditional transfer'** has to be used.

1. **conditional wait**

A simple example of conditional transfer is show bellow. Assuming that the data is being transferred to printer which operates at 40 characters/second, the computer will find that the device is ready ones every 25 ms. The three instruction involved in performing the test will take approximate 5 μ sec (the actual time will depend on the speed of the processor); thus the condition test will be carried out about 5000 times for each character transmitted. The computer will spend 99.98% of its time in checking to see if the device is ready and only 0.02% of that time doing useful work.

```
TEST : IN A , (STATUS)
       BIT 0 , A
       JR  Z , TEST      repeat until A is zero
       OUT  DATA , A
```

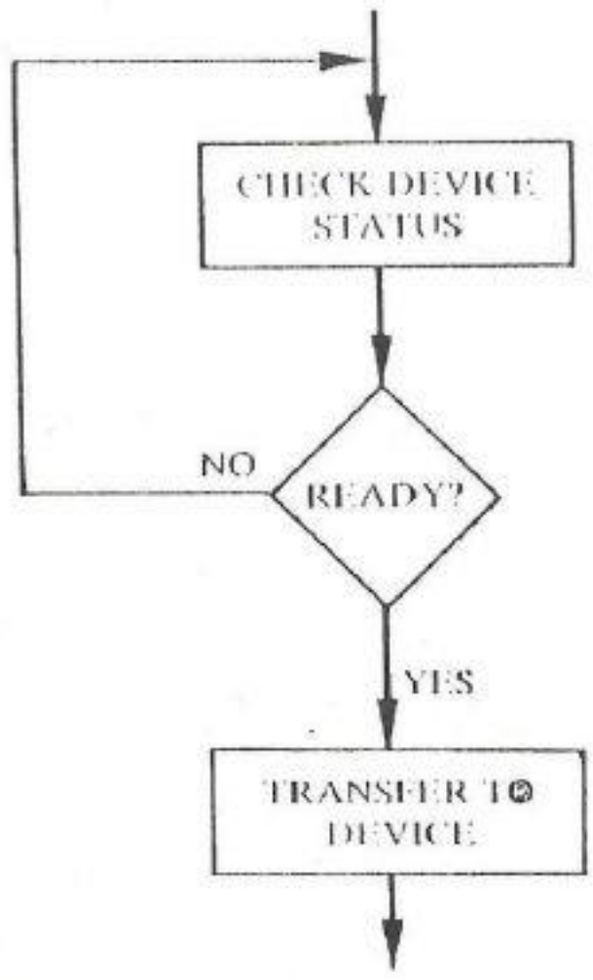

Fig. Conditional transfer (busy wait).

2. Conditional delay loop:

As an alternative to providing, on the interface, a status line which can be tested by the computer, a timing loop generated in the computer by loading a register and then decrementing it a specified number of times can be used, e.g. :

```
        LD    B,25        ; load register B with time delay
Loop:   DEC   B           ; decrement B
        JR    NZ, Loop    ; repeat until B is zero
```

To ensure that no transfer is made before the peripheral is ready, the time delay must be slightly greater than maximum delay expected in the peripheral; thus in term of use the CPU this method is even more inefficient than the use of conditional wait. It does slightly simplify and reduce the cost of the interface.

3. Continues method (periodic checks):

An alternative arrangement for conditional transfer, which allows the computer to continue doing useful work if the device busy.

In this method a check is made to see if the device is ready: if it is ready then the transfer is made; otherwise the computer continues with other work and returns at some later time to check if the device is ready.

The technique avoids the inefficiency of waiting in a loop for a device to become ready, but presents the programmer with the difficult task of arranging the software such that all devices are checked at frequent intervals.

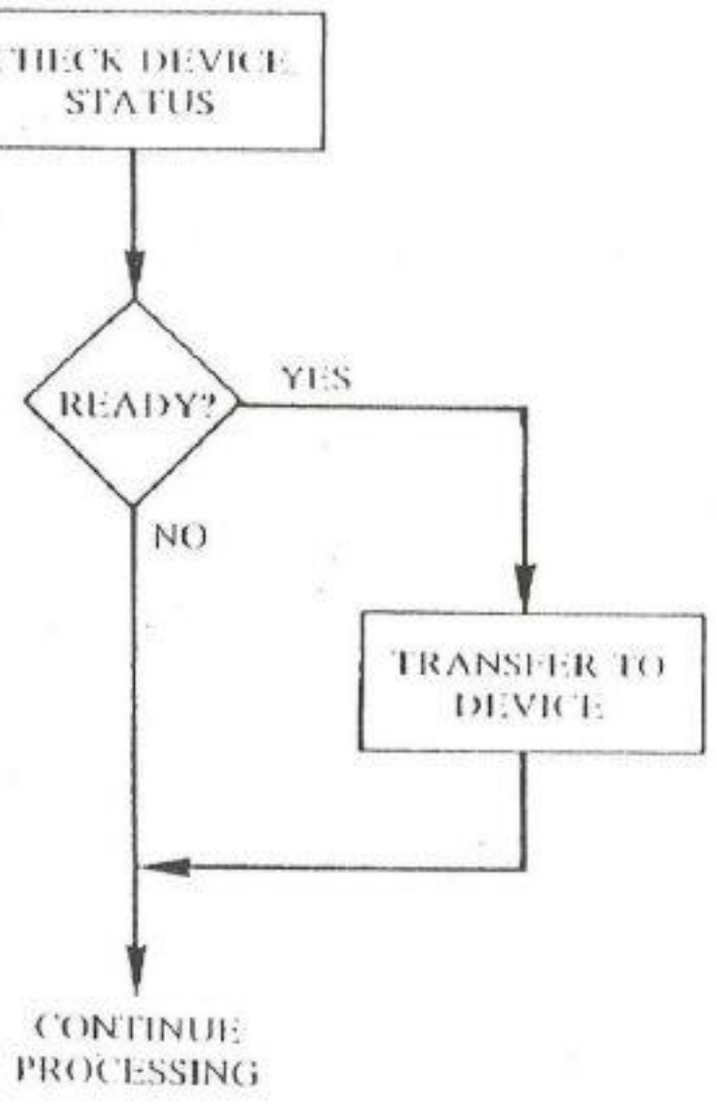

Fig. Conditional transfer.

Interrupts:

An interrupts is a mechanism by which the flow of the program can be temporarily stopped to allow a special piece of software – an Interrupt Service Routine, or Interrupt Handler – to run. When this routines has finished, the program was temporarily suspended is resumed. The process is illustrated bellow:

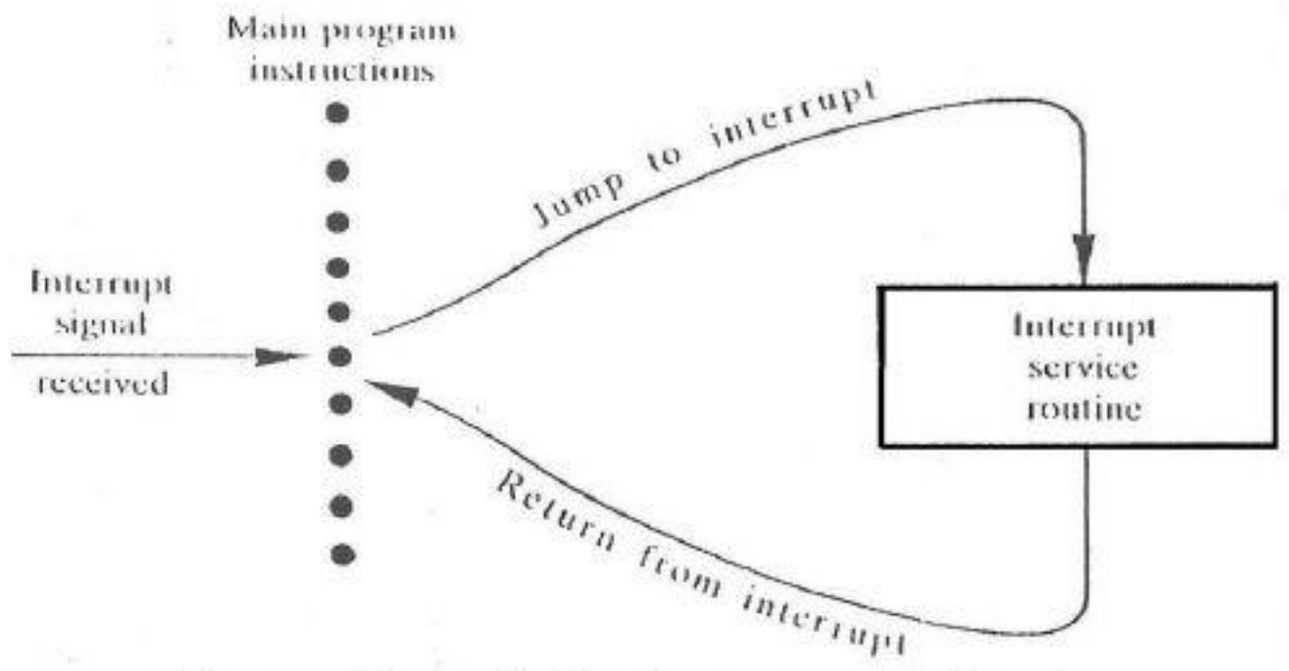

Fig. Interrupt-initiated program control transfer.

Interrupts are essential for the correct operation of most real-time computer systems; in addition to providing a solution to the conditional wait problem they are used for:

- **Real-time clock:** The external hardware provides a signal at regulated spaced intervals of time; the interrupt service routine (ISR) counts the signal and keeps a clock.
- **Alarm input :** Various sensors can be used to provide a change in a logic level in the event of an alarm.
- **Manual override:** Use of an interrupt can allow external control of system to allow for maintenance and repair.
- **Hardware failure indication:** Failure of external hardware or of interface units can be signalled to the processor through the use of an interrupt.
- **Debugging aids:** Interrupt are frequently used to insert breakpoints or traces in the program during program testing.
- **Operating system:** interrupts are used to force entry to the operating system before the end of a time slice.
- **Power failure warning:** It is simple to include in the computer system a circuit that detects very quickly the loss of power in the system and provides a few milliseconds warning before the loss is such that the system stops working.

1. Saving and restoring registers

Since an interrupt can occur at any point in a program precautions have to be taken to prevent information which is being held temporally in the CPU registers from the being overwritten.

All CPUs automatically save the contents of the program counter. This is vital: if the contents are not saved then a return to the point in the program at which the interrupt occurred could not be made.

2. Interrupt input mechanisms

A simple form of interrupt input is shown below. In between each instruction the CPU checks the IRQ line. If it is active, an interrupt is present and ISR is entered: if it is not active the next instruction is fetched and cycle repeats.

Note that an instruction involves more than one CPU clock cycle and that the interrupt line is checked only between instructions. Because several clock cycles may elapse between successive checks of the interrupt line, the interrupt signal must be latched and only cleared when the interrupt is acknowledged.

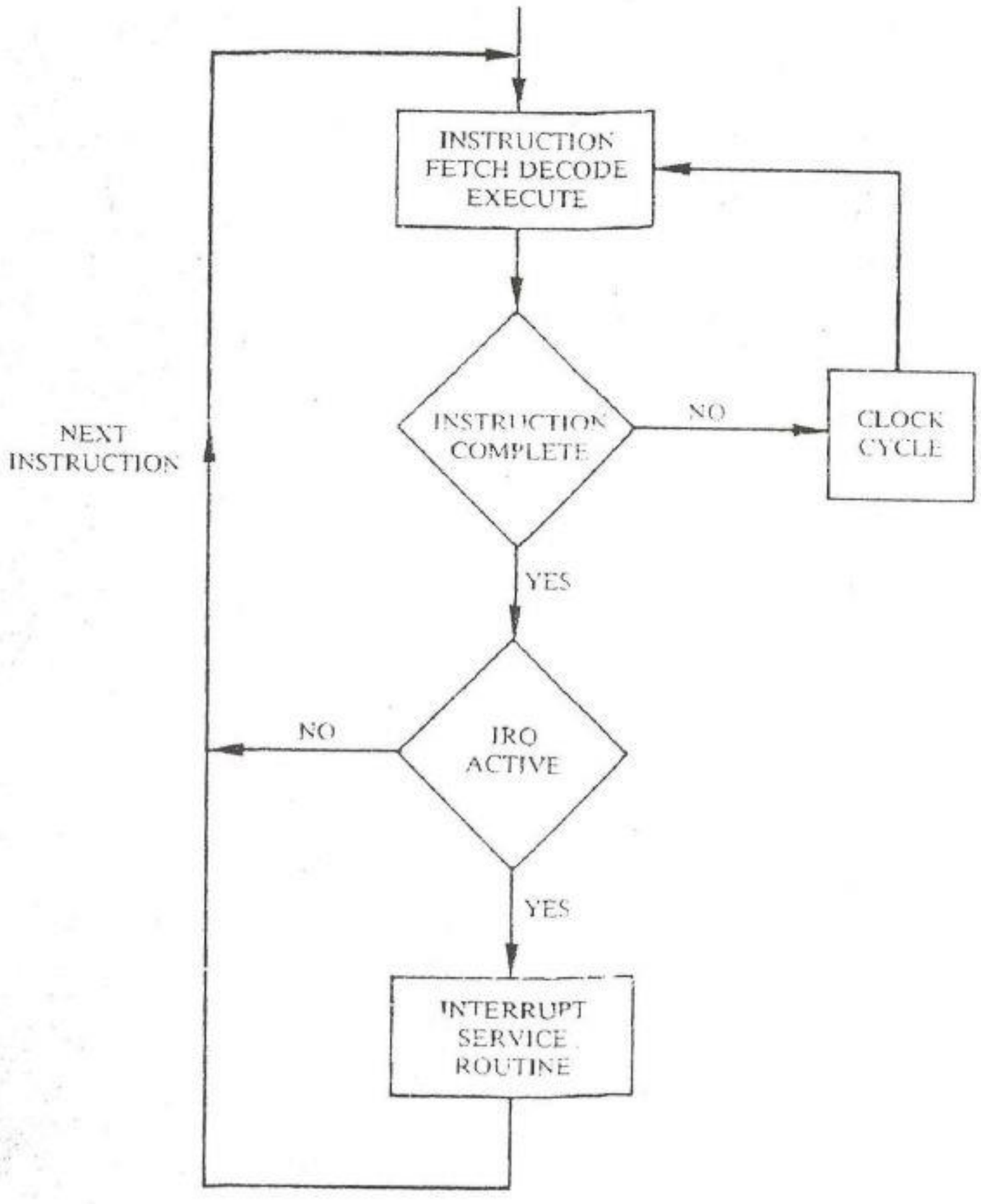

Fig. Flowchart of basic interrupt mechanism.

The use of polling with either busy wait or periodic checks on device status provides the simplest method in term of the programming requirements and in the testing of programs. Interrupt driven systems are much more difficult to debug since many of errors may be time dependent. At high data transfer rates the use of interrupts is inefficient because of the overheads involved in the interrupt service routine (saving and restoring the environment), hence polling is often used .

Direct memory access (DMA):

There are three modes for used DMA

1- Burst mode

In burst mode the DMA controller takes over the data highways of the computer and locks out the CPU for the period of time necessary to transfer. The use of burst mode can seriously affect the response time of a real time system to an external event and because of their mode not be acceptable.

2- Distributed mode

In this mode the DMA controller takes occasional machine cycle from the CPU's control and uses each cycle to transfer a byte of information to or from fast memory to the backing memory. In real-time system, if software-timing loops are used then the loss of machine cycle will affect the time taken to complete the loop.

3- Cycle-stealing method

In this mode transfer data only during cycles when the CPU is not using the data bus. Therefore the program proceeds at the normal rate completely unaffected by DMA data transfers. This is, however, the slowest method of transfer to backing store.

Communications:

The use of distributed computer systems implies the need for communication. As the distance between the source and receiver increases it becomes more difficult, when using analog techniques, to obtain a high signal to noise ratio; this is particularly so in an industrial environment where there may be numerous sources of interference. Analog systems are therefore generally limited to short distance.

The use of parallel digital transmission provides high data transfer rates but is expensive in term of cabling and interface circuitry and again is normally only used over short distances (or when very high rates of transfer are required).

Serial communication techniques can be characterized in several way

1- Mode

a) Asynchronous

b) Synchronous

2- Quantity

a) Character-by-character

b) Block

3- Distance

a) Local

b) Remote i.e. wide area

4- Code

a) ASCII

b) Other

Asynchronous and synchronous transmission techniques:

Asynchronous transmission implies then both the transmitter and receiver circuits use their own local clock signals to gate data on and off the data transmission line.

In order that the data can be interpreted unambiguously there must be some agreement between the transmitter and receiver clock signals. This agreement is forced by the transmitter periodically sending synchronization information down the transmission line.

The most common form of asynchronous transmission is the character-by-character system which is frequently used for connecting terminals to computer equipment and was introduced for transmission of information over telegraph lines.

It is sometimes called the stop-start system. In this each character which is transmitted is preceded by 'start' bit and followed by one or by two 'stop' bits (see the figure below).

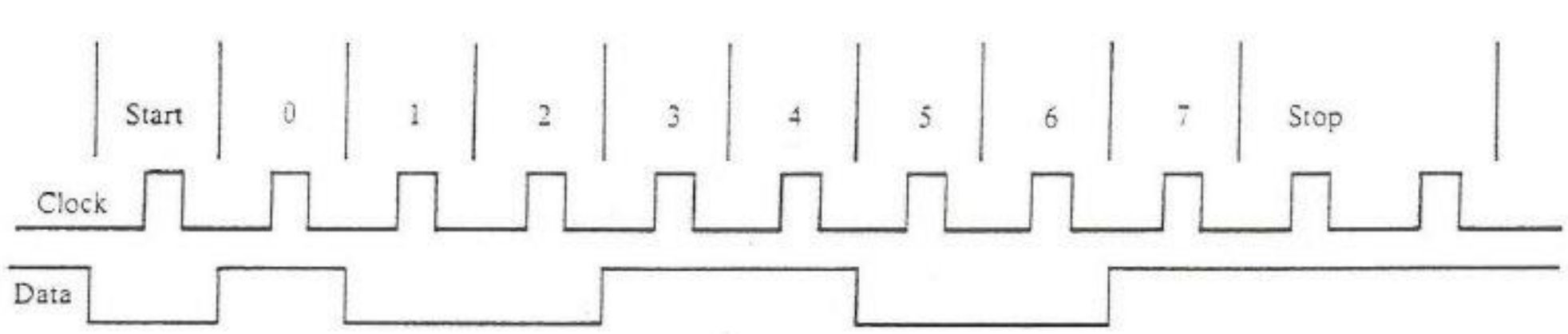

Fig. Asynchronous transmission.

The start bit is used by receiver to synchronize its clock with the incoming data; the signal must remain synchronized for the time taken to receive the following eight data bits and two stop bits.

The advantage of the stop-start system is that, particularly at the lower transmission rates, the frequencies of the clock signal generators do not have to be closely matched.

The disadvantage of the system is that for each character transmitted (8bits) three or four extra bits information have also to be transmitted, i.e. the overall information ratio is not very high.

The range of transmission speeds used for this system is from (75) bits/sec to (9600) bit/sec. The standard speed are (75 ,110 ,300 ,600 ,1200, 1800, 2400, 4800, 9600)b/s.

To overcome the problem of transmitting redundant bits, synchronous systems designed to transmit large volumes of data over short period of time, such as computer-to-computer systems, use block synchronous transmission techniques. Here, the characters are grouped into records, e.g., blocks of 80 characters, and each record is preceded by a synchronization signal and terminated with a stop sequence .

The synchronization sequence is used to enable the receiver to synchronize with the transmit clock.

In order to establish effective communication it is necessary to transmit more than just a synchronization signal – the additional information is called the protocol.

A simple protocol is:-

1. At the start of a transmission, bit synchronization is achieved by the transmitter sending out a sequence of 0s and 1s.
2. Followed by the ASCII code "SYN". The transmitter will continue to send the "SYN" code until receiver responds by sending back the code "ACK" or a preset time elapses (device time out).
3. If time out occurs, the transmitter sends the bit pattern of 0s and 1s again.
4. Once contact has been established the transmitter will send out "SYN" characters during any idle period and the receiver will respond by sending back "ACK".
5. The line will only be completely idle when the transmitter has send "EOT" (end of transmission) character.

The text is broken up into blocks and each block is preceded by an "EXT" (start of text) character and ended by an "ETX" (end of text).

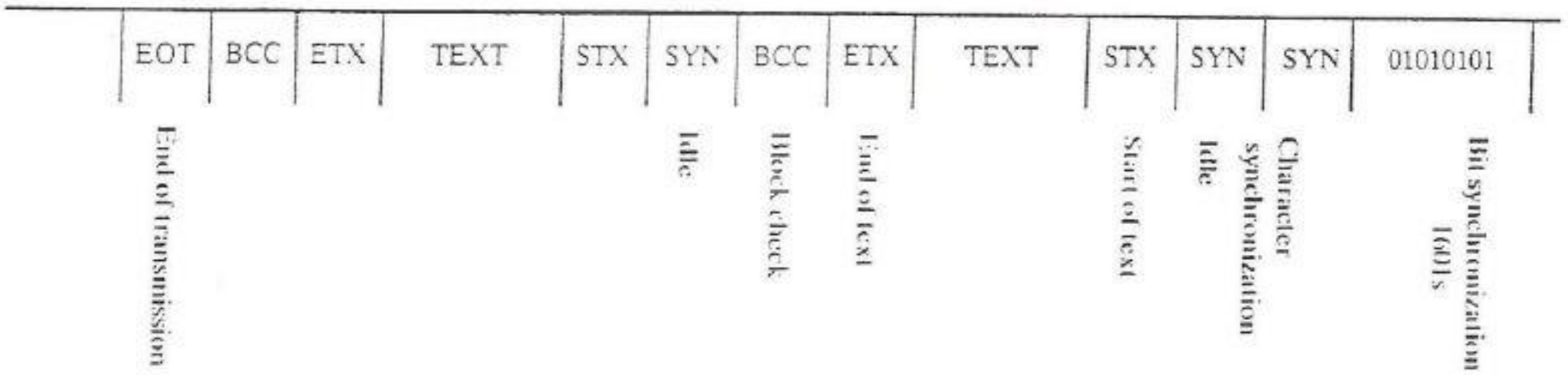

Fig. Synchronous transmissions

Following the ETX will be an integrity check on the data, typically this will take the form of a parity check.