6.1 DESIGN CONSIDERATIONS FOR DATA SAMPLING RATE

In order to cover the wide frequency spectrum of local microseisms, it is essential to generate the digital data pertaining to these tremors in good quantity. For teleseisms, the frequency is highly limited at its upper spectrum but for microseisms any seismic signal upto 100 Hz and if possible, even more than this, may be considered. In view of this requirement, optimum sampling rate has been worked out taking into consideration the data handling capabilities in real time mode of digital data processing system.

The seismic signal generated by three seismometers (in case of three channels station) at each remote station is converted into digital form at a rate of 100 samples per channel per second. (taking care of seismic signal upto 50 Hz.). If it is operated as a single channel station (with only one seismometer) then the signal is converted at a rate of 300 samples per second.(taking care of seismic signal upto 150 Hz.). Thus each outfield station has been designed to generate data at a rate of 4800 bits per second. Thus the seismic data arriving per second at the base station of an intelligent local seismic network (ILSN) from its all the eight remote stations is 40 k bits or 5.0 k bytes.

Each data sample has been kept 16 bit long. Out of
these, 12 bits are of A/D converter output which corresponds to input seismic signal and the remaining 4 bits are for channel gain indication and other allied information. In order to achieve maximum dynamic range the automatic gain ranging has been incorporated.

6.2 DATA FORMATTING AND DATA BLOCK IDENTIFICATION

Out of the 300 samples received in one second, the last one is rejected and in place of it, an other unique word designated as SYNCH-WORD of 16 bits is incorporated. It is embeded in the data stream to identify the start of every data block of one second duration as shown in the fig. (6.1). Each processor card of IFEH module does the similar processing tasks independently on the seismic data received by it from its remote station. Every processor card keeps on checking independently if the fresh data sample received by it is a "Synch word" or not. If the data sample is not the synch word, it is rejected and the same process is continued till the synch word is received by the processor card.

When the processor card detects for the first time the
arrival of synch word, it is retained and is stored in its circular memory which is reserved for the storage of pre event data. The next four data samples which arrive from remote station just after the detection of synch word, are also rejected. In place of these, another four samples containing "Header-information" are stored just next to synch word in the circular memory.

The header information includes- a) time information with one second resolution, b) event number, c) channel number, and d) the number of data samples accumulated during the period which starts from the occurrence of second pulse of the internal clock and ends at the arrival of next synch word transmitted by the remote station. For clarity it is shown in the fig.(6.2).

<table>
<thead>
<tr>
<th>EVENT NO.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sync. word</td>
</tr>
</tbody>
</table>

FIG.6.2 HEADER INFORMATION

The header information before its shifting into circular memory, is kept in some pre determined locations where it is upgraded as and when relevant information is received. Timing information is upgraded after every second whereas the counts of data samples are upgraded with the arrival of synch pulse. Similarly the other desired information is upgraded. Each synch word occurring once in
every second is always followed by the 'Header Information'.

6.3 TECHNIQUE- EVOLVED FOR TIME COMPUTATION

The synch word is unique. Therefore it is easy to detect it from the continuous data stream or bank of digital data. For any kind of counting, interpretation and analysis of data, the "Synch-word" has been taken as reference point. Therefore, it is very essential to determine its exact position in time domain. Once its position is ascertained is known, then the time of arrival of any wave front falling between this synch word and the succeeding one, can be found out with remarkable accuracy. The number of data samples accumulated during the period starting from the occurrence of second pulse and ending at the following synch word, are used to compute the exact time of arrival of the following "synch word".

For example, at an instant, when second pulse occurs, the time is say- 10(H): 20(M): 15(S) and the number of data samples accumulated during the time starting from the occurrence of this second pulse and ending with the next synch word are say- 100. It means that the actual time when synch word has occurred is- 10H:10M:15.330S i.e. it occurred 330 mili seconds later because fresh data sample arrives after every 3.3 mili-seconds.

Once the temporal position of synch word which is unique and distinctly identifiable, is determined accurately, the actual time at any point on seismogram may be computed directly with lot of precision. It can be done
simply by counting the number of data samples falling between synch word and that very point on waveform (seismogram) at which the time is required. The information about exact time is very important and with this 'Simple design technique', it is computed easily and straightforward:

6.4 COMPUTATION OF ARRIVAL- TIME OF WAVE FRONT

First of all the actual time is computed for that synch word which is just before the point at which the exact time information is needed. The time for this synch word is computed as follows:

Actual time "T" contained in header information + (no of data samples "N" contained in header information x 3.3 ) mili second, say it is equal to "X" seconds.

It is the time of the synch word from where one has to count the number of samples falling between this synch word and the point at which the time is to be computed.

Now count the number of samples between the synch word and the point at which the time is to be computed. Say these samples are- "Y".

Hence actual time at this point is equal to "X" + (3.3x Y) mili seconds

6.5 DESIGN- TECHNIQUE FOR- MIXING SEISMIC CLOCK DATA WITH DIGITAL FIELD DATA TELEMETERED DOWN TO BASE STATION

After every second, the clock module which is integral part of the Base-station, sends out real time information in
byte form. Unit and ten of time forms one byte as shown in the fig. (6.3). At the start of every second, the clock module brings at its output port the first byte (seconds) of the current time. After a very short delay (few microseconds only), it generates a pulse which interrupts the micro-processor of every processor card at RST-6.5 for taking this byte in and storing it at predetermined location. The main program is suspended for this period.

![Diagram showing the timing pattern](http://example.com/diagram6.3.png)

**FIG. 6.3 SEISMIC-TIMING PATTERN**

![Diagram showing the interrupt pulse pattern](http://example.com/diagram6.4.png)

**FIG. 6.4 CLOCK DATA AND INTERRUPT PULSE PATTERN**
This total process takes approximately 150 micro-seconds.

After an additional delay of approximately 50 micro seconds, same process is repeated for taking in the next timing byte (minutes) as before. Thus the complete process for taking in the current timing data immediately after every second, is executed with a burst of five pulses which are generated in one milisecond as shown in the fig. (6.4). This sequence of five pulses though generated after every second after the power has been switched on but these would act as interrupts to microprocessor only when the system programming is over and the task of "Execution" has been initiated.

The time is upgraded in this manner simultaneously in all the processor cards after every second. Each pulse of the burst acts as interrupt pulse to command the microprocessor to take in one byte of timing data. Thus after every second, the time is upgraded within a milisecond.

6.6 DESIGN STRATEGY FOR- TRUE EVENT DETECTION- AT THE LEVEL OF INDIVIDUAL REMOTE STATION

The fresh data sample is taken in after every 3.3 m seconds in response to an interrupt at RST- 7.5 of processor card. This interrupt pulse is generated by data recovery logic after every 3.3 mili seconds and is effective only when the system programming is over and execution key has been pressed. The main program for computation of algorithm for true event detection is suspended temporarily to execute
this interrupt service routine.

The processor card under main program is engaged in computing on the incoming data the values of "Short term average" (STA) and "Long term average" (LTA). Every time when new value of STA is available, the ratio of new STA and the previous LTA is computed to detect the occurrence of a true event at the site of remote station whose field data is being processed by its processor card. When the occurrence of true event is detected (i.e., when STA > K LTA where K is an integer and greater than '1') by a processor card (channel) at its remote station, a flag is set in some predetermined locations of its memory to register it.

Every time in the end, as a necessary part of interrupt service routine (ISR) meant for clock time upgradation (ISR for RST 6.5), this very location is checked to find out if event occurrence status flag is set or reset. If it is set (true event has occurred at this remote station), the code (0001) is sent to interface controller. If it is reset (true event has not occurred), then the code (0000) is sent to the Interface controller.

Every processor card of IFEH module continues to send one of these two unique codes as per its processing result to interface controller until it does not finally declare the occurrence of a true event at network level after doing processing on these codes. This processing is discussed at later stage. Interface controller declares the occurrence of true event by issuing a signal designated as event true (ET) signal at one of its output port line. Once the occurrence
of true event is declared finally at network level, then that part of the RST-6.5 interrupt subroutine (of every processor card) which sends either of the two codes for event status flag to interface controller, is not taken up any further.

6.7 DESIGN- PHILOSOPHY FOR- TRUE EVENT DETECTION AT THE LEVEL OF ONE COMPLETE LOCAL SEISMIC NETWORK (ILSN)

The occurrence and detection of true event at any individual remote station of a local network can not be taken up for sure occurrence of true event at network level. For the declaration of occurrence of true event at one complete local network (ILSN), the following design criteria has been evolved out.

Every second, just after the upgradation of clock, one of the selected processor card interrupts the microprocessor of Interface controller at its RST-7.5. As a result of this, the RST-7.5 interrupt subroutine reads one after the other the codes on event occurrence status from the output ports PB of 8255-II and 8255-III of each processor card of IFEH module. The Interface controller starts keeping the record of time from the instant it detects for the first time-the true event occurrence status code sent by any one of the processor cards. The moment it detects again the arrival of code for true event sent by some other processor card, it computes the time lapsed in between the arrival of these two codes meant for true events which have occurred at two different remote stations at different times.
If the time lapsed is less than the time preset for 'Time window' (TW), then the Interface controller checks if true event has occurred at all the required remote stations or not. If all the required processor cards (remote stations) programmed for this purpose, have not sent the true event occurrence code as yet, the true event occurrence is not declared finally at network level and the process continues.

If all the required processor cards (remote stations) have sent the true event occurrence code, then the total time lapsed between the code which was received first of all and the one which was received in the end is computed. If this time is within the preset Time window period, only then the interface controller declares the occurrence of true event at network level otherwise not.

If the total time so computed exceeds the period set for time window (TW), the event is not declared true and the whole process is started fresh again. If finally it is

![FIG. 6-5 TRIGGER WAVEFORMS](image-url)
detected by the interface controller that the true events have occurred at the desired remote stations within the prescribed time window (TW), then it declares the occurrence of true event at network level [65]. The total design concept has been clarified in the fig.(6.5).

The interface controller now issues a signal designated as "ET" to command each processor card to do some additional jobs and to stop certain tasks which were being done by each processor card before the issuance of this signal.

6.8 DESIGN METHODOLOGY FOR SEISMIC EVENT DATA ACQUISITION, PROCESSING AND RECORDING IN REAL TIME MODE

The toughest design challenge in seismology is that the data acquisition, processing and its final recoding into huge secondary storage must be equal to or faster than the combined data generation rate. If design philosophy does not fulfil this requirement, there would be a cumulative data overlapping problem in the intermediary storage device. As discussed earlier, the data generation rate is very high in case of telemetered seismic network array (eight times to the data generated by a single remote station). Sometime the event may continue for a very very long time particularly when large post-event period is desirable. Under such circumstances, the handling of seismic data is very difficult and needs evolution of newer ways and methods.

Handling of seismic data in real time mode at the base station of the ILSN is required in two cases-

(a) when
FIG. 6.6  SYSTEM CONFIGURATION OF BASE STATION OF AN ILSN
occurrence of true event has been detected finally at the network level by the Interface controller, and (b) when the realtime monitoring of all the remote stations of the network is required to check its proper functioning. The total design philosophy worked out under this thesis, has been illustrated in fig. (6.6)

The interface controller identifies if it has to send data from its memory over to PC/XT- RAM for "Real Time Monitoring" (RTM) or for "True event occurrence". The Interface controller does it by checking logic level at line PC6 of 8255-III (IC) as shown in fig.(6.6). When this level is at logical "1", the transfer of data is meant for Real Time Monitoring (RTM) and when this level is at logical "0", the transfer is meant for True Event Data (TED).

Let us first consider the case when data transfer is meant for 'True event data' i.e. the logical level at this line PC6 of 8255-III (IC) is at logical "0".

Case-I

When the true event has been detected finally at network level, then an equal number of data bytes (1200 bytes) from the circular memory of each processor card are transferred into the pre determined locations of PC/XT-RAM. When the storage defined for this purpose in the PC/XT-RAM has been filled to the defined capacity, this data block from PC/XT-RAM as a whole (it is eight time larger than the data bytes sent by each processor card during this period i.e. 9.6 K bytes) is transferred into hard disc of PC/XT.
The process of transferring of data block from PC/XT-RAM to its hard disc takes approximately one second period. The design evolved out is such that when the data block from the PC/XT-RAM is being transferred into hard disc, the freshly incoming data during this period is not ignored but it is attended exactly in the same way as before.

The design approach to meet this requirement is as follows:

After the true event has been declared, the oldest data sample (2 bytes) from the circular memory of each processor card is read out and is sent to the input of its buffer set, as shown in fig.(6.6) and in fig.(6.7). At these very same locations, the fresh incoming data sample telemetered from the remote stations is stored in their respective processor cards (one data sample in each card).

The address counter of the circular memory is then incremented for the address of next sample which has now become oldest. Each processor card resumes its main programme after doing this additional task as a part of interrupt sub-routine at RST- 7.5 which is meant for taking in the fresh data after every 3.3 ms. But this additional task described above, is done as a part of RST-7.5 interrupt sub-routine only when event true signal "ET" has been issued by Interface controller otherwise not.

When the true event has been declared finally, then only, a logic circuit "X" shown in the fig.(6.7), enables a pulse occurring independently after every 3.3 mili seconds to act as interrupt signal at RST-6.5 of interface controller.
As a result of this interrupt at RST-6.5, the data sample made available earlier at buffer set of each processor card is read, into the memory of interface controller one after the other. The buffer set of each processor card is enabled for sufficient period by one of the pulses issued in sequence one after the other by the interface controller at its output port PA of 8255-III.(IC) as shown in fig.(6.8). PA0 line enables the buffer set of first processor card and PA1 that of second card and so on. In response to ISR-
RST-6.5(IC), every time two bytes from each processor card or 16 bytes in total are taken into the memory of interface controller. This process takes approximately 1.3 milliseconds. The same process is repeated again and again after every 3.3 milliseconds in response to the interrupt pulse at RST- 6.5 (IC). and the process is continued till the data upto post-event period is taken.
6.8.1. Design for restructuring of data for its storage in the memory of interface controller

Eight memory blocks (segments) of 2400 locations each have been marked in the memory of interface controller as shown in fig. (6.9), to store the data of eight remote stations (channels). One segment is exclusively reserved for storing the data pertaining to one channel (remote station) only. The 2400 locations in each segment store the data which is received in four seconds duration from one channel (remote station).

Each segment is further sub-divided virtually into two data sub segments designated as data sub segment "A" and data sub-segment"B". Data writing rate into "Circular memory" of each processor card is maintained exactly same as the rate of data retrieval from it. When 'ET' signal is
true, then after every 3.3 milliseconds one data sample is written into it and one data sample is retrieved from it. Therefore at no time, there is any kind of overlapping of data. After every 3.3 ms, the data available at the input of buffer set of each processor card is stored one after the other into its corresponding memory segment.

The transfer of data from circular memory of any processor card into its corresponding data segment in Interface controller memory starts only when Interface controller has detected the arrival of synch word from that very circular memory as shown in fig.(6.10).

In general, these synch words from the circular memory of different processor cards, are sent to their corresponding segments at different times and not simultaneously. It is because—firstly all the remote stations do not start transmission at the same time and
secondly the synch word transmitted by the remote station is not detected by their corresponding processor card at the same time.

The maximum difference in time delay from the start of data storage in any segment which has picked up synch word first of all and the segment which has picked up its synch word in the last, is always less than one second. It means that at any instant of time, the segment which happens to receive its synch word first of all, may have one second duration data (600 bytes) more as compared to one which picks up its synch word in the last.

| CHANNEL-1 (SEG-1) | SUBSEGMENT— A | 0000 TO 1199 |
| SUBSEGMENT— B | 1200 TO 2399 |
| CHANNEL-2 (SEG-2) | SUBSEGMENT— A | 2400 TO 3599 |
| SUBSEGMENT— B | 3600 TO 4799 |
| CHANNEL-3 (SEG-3) | SUBSEGMENT— A | 4800 TO 5999 |
| SUBSEGMENT— B | 6000 TO 7199 |
| CHANNEL-4 (SEG-4) | SUBSEGMENT— A | 7200 TO 8399 |
| SUBSEGMENT— B | 8400 TO 9599 |
| CHANNEL-5 (SEG-5) | SUBSEGMENT— A | 9600 TO 10799 |
| SUBSEGMENT— B | 10800 TO 11999 |
| CHANNEL-6 (SEG-6) | SUBSEGMENT— A | 12000 TO 13199 |
| SUBSEGMENT— B | 13200 TO 14399 |
| CHANNEL-7 (SEG-7) | SUBSEGMENT— A | 14400 TO 15599 |
| SUBSEGMENT— B | 15600 TO 16799 |
| CHANNEL-8 (SEG-8) | SUBSEGMENT— A | 16800 TO 17999 |
| SUBSEGMENT— B | 18000 TO 19199 |

FIG. 6.11 PARTITIONING AND ADDRESSING OF BLOCK 'X' AND BLOCK 'Y'

102
In other word, when the segment which is the last to receive its synch word, has accumulated two second data i.e. 1200 bytes in it, the segment which is the first to receive the synch word, would have 1800 bytes accumulated in its segment by this time.

Since there are eight segments hence there would be eight sub segment "A" and eight sub segment "B" as shown in the fig.(6.11). For the further convenience and future reference all the sub-segments identified as "A"s have been clubbed together into a single data block of 9.6 K bytes capacity and have been designated as "X". Similarly all the eight sub-segments designated as "B"s have also been clubbed together into a single data block of 9.6 K byte capacity and has been designated as "Y". Symbolically both "X" and "Y" are monolithic block (continuous) but in reality both the blocks are non continuous and discrete.

Now when data is being stored into a segment, it is checked if it has accumulated 1200 bytes of data in its sub-segment or not. This data would go in either sub-segment "A" or sub-segment "B". When sub-segment of a segment is filled to its capacity (1200), a counter initially kept reset is incremented by one. Similarly when a sub-segment of any other segment just stores 1200 bytes, the counter is again incremented. When the sub-segment of the segment which was last to receive the synch word, has just received 1200 bytes, the counter will be incremented again. At this instant of time, it would have count of 8 stored in it. The moment count of 8 is detected in this location, a
flag is set into some appropriate location to indicate that block "X" is filled to capacity.

Similarly if the block "Y" is filled, then the flag is set in some other identified location of the memory. The main program keeps on checking if either of these two flags is set. If a flag is found set, then it is checked which one is set. Then the transfer of data over to PC/XT-RAM is started from the block which may be either "X" or "Y".

6.8.2. Design for true event data transfer

When true event signal designated "ET" is issued by
interface controller as shown in the fig. (6.12), the interrupt pulse at RST 6.5 of Interface controller (IC) is enabled after every 3.3 ms. The interrupt pulse at RST- 7.5 (IC) is disabled. The moment, as discussed above, it is detected that all the sub-segments of block "X" have been filled completely i.e. block "X" has by now stored just 9.6 K byte of data (1200 x 8 = 9.6 K byte), the transfer of data from block "X" over to PC/XT-RAM is initiated by main program of Interface controller.

Before an interrupt pulse is issued to PC/XT for data block transfer, first of all, it is checked if PC/XT is ready to accept the data or not. This checking is done by examining the logic level at line PC7 of 8255-III (IC) of Interface controller as shown in fig. (6.12). If this level is at logical "1" (PC/XT is ready), a pulse is generated at line PC0 of 8255-III of IC module as shown in fig. (6.12). This pulse interrupts the PC/XT at its 'INT' for lifting the data block. And now, the data is transferred from the block under "Transfer mode" (The block which has been just filled to its capacity, comes under Transfer mode) over to PC/XT-RAM byte after byte (byte serial form).

6.8.3. Design scheme for transferring the data block in byte serial form from interface controller to PC/XT-RAM

a) The complete data belonging to first sub-segment (i.e. 1200 bytes) of the block under "Transfer mode" is transferred first of all in byte after byte form. After
that, in a similar manner, the complete data of second
sub segment is transferred after reallocating the
starting and end addresses of the second sub-segment. The
process of data transfer from these sub-segments in the
manner as indicated above is continued till the data of
all the sub-segments of one block under transfer mode is
transferred to PC/XT-RAM.

b) So long as, there is no task other than this for
interface controller, the data transfer from one block
over to PC/XT-RAM continues in byte serial form (i.e.
byte after byte). But besides this, there is, in fact,
another important task to be done by the Interface
controller on priority basis. This task is to lift the
then oldest eight data sample from the eight processor
cards (one data sample from each card) of IFEH module
after every 3.3 mili-seconds. It is done after every 3.3
mili seconds when Interface controller is interrupted at
its RST-6.5 as shown in the fig.(6.12). In response to
this interrupt, eight data samples (16 bytes) from eight
processor cards are to be taken in and stored into their
corresponding subsegment (segment) in the interface
controller. This task is done by suspending temporarily
the process of data transfer from the block under
transfer mode.

c) The data sample present at the input of buffer set of
each processor card is lifted one after the other as
explained earlier in fig.(6.8 & 6.12) and accordingly
eight samples are stored into their corresponding sub
segments ie one sample in one sub-segment by reallocating the relevant address of the segment.

The worked out design is such that if data block 'X' is being transferred, the data present at the input of buffer sets are stored automatically in data block 'Y' after every 3.3 milli seconds and vice-versa. When a block is filled to the defined capacity with the data samples received from the circular memories of the processor cards, then only the data of this block is transferred over to PC/XT-RAM and not before that.

The process of data transfer and that of data storage alternates itself between these two blocks - 'X' and 'Y' when data for true event is sent to PC/XT. In nutshell, while the block under transfer mode is engaged in transferring its data over to PC/XT-RAM with intermittent interruption, the other block under storage mode is gathering the data samples from IFEH module after every 3.3 milli seconds in response to interrupt signals at RST-6.5 of Interface controller.

When the block which is gathering data from processor cards gets completely filled (examined by checking a flag), then it goes into transfer mode and starts transferring its data over to PC/XT-RAM. While the other block which has just transferred its data, goes into storage mode and start storing data in it. This process, in alternate, would continue till the data for event and post event periods is stored finally into hard-disk of PC/XT.
6.8.4. Technique evolved to meet the critical design requirement for uninterrupted acquisition of data pertaining to a very long seismic event

The most important 'design criteria' is that the data transfer rate over to PC/XT-RAM from the block under transfer mode, must be much faster than the rate at which the data is being gathered into the other block under storage mode.

As per worked out design scheme, the time required to fill a block to its full capacity i.e. 9.6 K bytes (4800 data samples from eight channels or 600 sample from each processor card) is exactly 2 seconds. Therefore, the block which has just been filled to its capacity, must be vacated totally by transferring its data over to PC/XT-RAM in much less than 2 seconds time. Immediately after this, the data which has been just transferred to PC/XT-RAM must be further transferred to PC/XT hard-disk.

The design has been done in such a way that the total time together in these two tasks (task-1; transfer of data from the block under transfer mode over to PC/XT-RAM and task-2; the further transfer of the data block stored into PC/XT-RAM as a result of task-1, over to PC/XT hard-disk) is always less than 2 seconds. Lesser is this time, better is the design.

The system has been designed to execute the task in 3.3 mili seconds period as follows-

The execution of interrupt service sub-routine (ISR) at
RST 6.5 of Interface Controller takes time of the order of 1.3 mili seconds to transfer the data samples present at the input of buffer sets of IFEH module into the desired data block in interface controller. Thus out of 3.3 ms, every time 1.3 ms is consumed for this purpose. Therefore, only 2 mili seconds are left out for sending data from the data block under transfer mode over to PC/XT- RAM before the arrival of next interrupt. As per design strategy, in these 2 mili seconds the maximum number of bytes must be transferred over to PC/XT- RAM from the data block under transfer mode. The design of hardware and software for this purpose must be most efficient.

It has been done as follows-

After the true event has occurred, the block 'X' is first of all filled to its full capacity with the data samples which are made available at the buffer set after every 3.3 mili seconds (ms). This process of filling a block takes two seconds. In every 3.3 ms (out of 3.3 ms, only 1.3 mili seconds is actually used for lifting the data), 16 byte of data (8 data samples) are received in block 'X'.

The moment block gets filled, Interface controller checks if PC/XT is ready to lift and store this data into its RAM. It is done by checking the logical level at one output line of PC/XT output port as discussed earlier in fig.(6.12). The block which is just filled, now goes to Transfer mode. The other block which was vacant till now, goes to storage mode and starts getting data from the circular memorries of processor cards of IFEH module. First
byte from the block which is under transfer mode, is brought at output port PC of 8255-II (IC) as shown in the fig.(6.12). After this, a pulse is generated at line PC0 of output port PC of 8255-III of IC. This pulse interrupts the PC/XT at its INT. As a result of this, the interrupt subroutine of PC/XT, first of all, changes the logical level "1" to "0" at its line PC7 of 8255-I (PC/XT). Then the data byte available at port PB of 8255-I (PC/XT) is read into and is stored in data block of PC/XT-RAM. After this, it is checked if the data block in PC/XT has acquired 9.6 K byte of data or not. If not, then it goes to scan the arrival of pulse at its line PC1 of 8255-I (PC/XT).

For this task, the interrupt sub-routine of PC/XT takes approximately 35 micro seconds. When PC/XT is doing the above indicated task, the Interface controller is doing concurrently its task of bringing the next byte at the identified output port as before. After issuing the first interrupt pulse, the Interface controller checks if it has sent out all the 9.6 K byte of data from the block under transfer mode. If not, it prepares the address of the next byte to be transferred and brings out the byte on the same output port PC of 8255-II(IC) as before. Instead of an interrupt, it sends out a pulse at its line PB0 of 8255-III (IC) which is being scanned by PC/XT as shown in fig. (6.12) as a part of its interrupt service routine (ISR).

In order to execute the total task involved in bringing out the new data byte at the identified output port and then
for the issuance of the signal to communicate to PC/XT about the availability of new data byte for its immediate lifting, the Interface controller takes approximately 50 micro-seconds every time. The period of 35 micro-seconds which is taken by PC/XT for lifting the byte made available at the output port by the Interface controller and storing it into its RAM and then going over to next job of- scanning the arrival of the pulse which is issued by the Interface controller for indicating the availability of the next byte, run concurrently.

This period is within the period of 50 micro seconds. Hence, the period of 35 micro seconds consumed by PC/XT has no relevance or bearing on timing constraints because it is less than 50 micro seconds and is accounted fully within the period of 50 micro seconds consumed by interface controller for the purpose explained above. Therefore, every byte in actual is transferred in 50 micro seconds over to PC/XT-RAM from the data block under transfer mode.

As explained earlier, for 1.3 mili seconds out of every 3.3 mili seconds, the process of transfer of bytes over to PC/XT- RAM is suspended and during this period of 1.3 mili seconds, the 16 bytes of data from processor card of IFEH module are taken into other block which is now under storage mode. Therefore in every 3.3 mili seconds period, the time actually left out for the transfer of these bytes is (3.3 - 1.3) = 2 mili seconds (ms) only.

The number of bytes which are transferred from the block under transfer mode over to PC/XT-RAM in 2 mili
seconds is $2000/50 = 40$. It means that in the first i.3 ms of every 3.3 ms sample period, the 16 bytes in total are taken from all the eight cards of IFEH module and stored segmentwise (two bytes in each segment) into the data block under storage mode. In the remaining 2 ms, 40 bytes are transferred over to PC/XT-RAM from one segment of the data block under Transfer mode. With this data transfer rate, after the period equal to 240 samples (each sample occurs after every 3.3 ms), the complete data of 9.6 k bytes is transferred from data block under transfer mode to PC/XT-RAM. Therefore, the total time consumed for transfer purpose is $240 \times 3.3 \text{ ms} = 792 \text{ ms} \text{ or } 0.8 \text{ second approximately.}$

Once the complete data from the block under transfer mode has been transferred over to PC/XT-RAM, the Interface controller has no other task but to receive 8 samples of data from IFEH module after every 3.3 ms in response to interrupt at its RST-6.5. It also checks as a part of this interrupt RST-6.5 if 9.6 K byte data has been accumulated into the block under storage mode.

For 9.6 k bytes of storage, at the above indicated data storage rate, it would take exactly 2 seconds. It means, there is still a period of $(2 - 0.8) = 1.2$ seconds left out before the time when data block now under storage mode, would be completely filled up and would become ready as data block for Transfer. In this 1.2 seconds left out period, the data block which has been just transferred into the PC/XT-
RAM is then further transferred into the Hard-disc of PC/XT. The transfer of data block of 9.6 K byte capacity from PC/XT- RAM over to hard disc takes of the order of 0.9 second. Hence total time for the data of a block which is under transfer mode, to get finally transferred from interface controller memory over to PC/XT hard disc via PC/XT- RAM is \( (.8 + .9 ) \) seconds i.e. 1.70 seconds only.

The other data block which is now under storage mode and has to accumulate 9.6 K byte data before becoming qualified for its data transfer, is still in the process of storage. This process will thus continue as long as the event is true irrespective of its length. The event data would be taken finally into Hard-disc along with pre-event data and post event data without loss of any data.

6.8.5. Design methodology for safeguard against overlapping of data into circular memory during event data transfer

The fresh data samples ( coming after every 3.3 ms ) coming from all the remote stations, are stored into circular memory of their respective processor card as per its channel number. From the circular memory which always keeps the incoming data equal to pre event duration, the earliest stored data sample is read out for its transfer over to data block under storage mode in the Interface controller. At this very location, the freshly incoming sample data is written immediately after this. The writing into and retrieval out rate of data in circular memory is
maintained equal to each other so that there is no overlapping of data.

After every 3.3 milliseconds, 16 bytes of fresh data have been acquired from the eight remote stations by the eight processor cards (two in each card) and the same number of bytes have been retrieved out of the eight processor cards to send them over to data block under storage mode as explained above. If the event is not true, no data would be lifted from the circular memory and the fresh incoming data from the remote stations would be written over the earlier stored data in the circular memory. Once the circular memory is filled to its capacity, the it would always contain the data equal to pre event length.

Case-II

When the logic level on line PC6 of 8255 (IC) in fig. (6.12) is detected by the interface controller as '0', then it becomes the case for 'Real time monitoring'.

6.9. "REAL TIME MONITORING" (RTM)

When real time monitoring is desired, a signal on line PC6 of 8255-III (IC) of Interface controller as shown in fig.(6.12), is sent by PC/XT for this purpose. Interface controller after every second checks it to find out if this line is at logical "1" or at logical "0". If it is at logical "1" then it is a command for carrying out the RTM otherwise not. When Interface controller detects the arrival of RTM signal, IC reads one byte of data from a predetermined portion of IC memory kept reserved for storing
realtime monitoring data. This data is received from all the remote stations of the network via processor cards of IFEH module after every second. A pulse is generated in a similar manner as is done in case of true event data transfer to PC/XT-RAM. This pulse interrupts the 8088 microprocessor of PC/XT. As a result of this interrupt, the data byte is lifted by the PC/XT and is stored at predetermined locations into its RAM. The process of lifting data from the identified port is exactly the same as it is in case of event data transfer process explained earlier.
After the transfer of byte into PC/XT-RAM, the Interface controller checks as a part of its interrupt subroutine at RST-6.5 if 16 bytes of data have been sent to PC/XT-RAM or not. Similarly the PC/XT interrupt sub routine checks if it has also received the same number of bytes or not. If not, the process is continued. When 16 bytes of data have been taken into PC/XT-RAM, then PC/XT executes a software written for displaying the "Histogram" on video display unit. The new value of each channel data received from remote stations after every one second are displayed as shown in fig. (6.13). The operator would see perceptible variation in the amplitude of each channel data on the video unit as a confirmation of proper functioning of the remote stations.

The variation in bar height displayed on the screen will confirm the proper functioning of the system. However, there would be some variation in bar height (very very small) due to cultural noise whether there is an event or no event.

6.9.1. Design for transfer of RTM data from remote station to interface controller memory

After every second, one of the two unique codes is sent by each processor card to Interface controller to indicate either the" Occurrence" or "No occurrence" of true event at its remote station. The process of sending out the code to Interface controller is executed by sending an interrupt signal at RST-7.5 of interface controller. The moment
this interrupt is generated, the very first task undertaken by the ISR is to generate a low level signal on the identified line as shown in the fig.(6.12). This line is sensed by every processor card. After sending out the code, none of the processor card does any other job but keep on waiting for it to go high.

After this, Interface controller sends a level "1" on this identified line which is being scanned by all the processor cards. Having done this, the Interface controller goes into a small delay loop. The moment logical level at this very line is sensed "1", all the processor cards immediately send out the latest sample at its respective buffer sets. After completing the delay loop, Interface controller lifts these samples one after the other in the same way as it does in the case of code lifting. And finally these eight samples are stored into the memory of the Interface controller at some predetermined portion.

After this, the Interface controller checks the logical level of an other identified line as shown in the fig.(6.12) if RTM request has been made or not. If there is no request for RTM, the Interface controller processes the codes information to find out if the true event has really occurred or not. If RTM request has been made , then the computation part for the detection of final event at network level is not taken up.Instead, these eight samples or 16 bytes of data which have been taken after codes are first of all transferred over to PC/XT-RAM as explained earlier for displaying them into "Histogram" form.
6.10. DESIGN TECHNIQUE FOR DATA SYNCHRONIZATION AT NETWORK (ILSN) LEVEL

The system comprises of eight channels. One channel is defined to comprise of - one remote station and one transmitter and receiver link along with antennas. It is not essential that all the eight channels are working properly. Secondly, sometime it may be needed to configure the system with less number of channels (accordingly less number of processor cards) due to some technical reasons or some processor cards may go bad.

It is not essential that the "Synch word" transmitted by the eight remote stations would be detected by their respective processor cards simultaneously at the same time. Sometime the synch word on a particular channel (processor card) may not be available even for a very long time due to the failure of transmission link. The interrupt pulse meant to take the data sample in is generated by each processor card separately and independently after every 3.3 milli seconds after the time since when the synch word has been detected by it. Since the synch word by each processor card has been detected at different times and not at one time, therefore the interrupt pulse at each processor card would be generated at different time after every 3.3 milli seconds at its RST-7.5. But all these interrupt pulses at their respective processor card will always appear within 3.3 milli seconds time span and the pattern of appearance of
these pulses will be repeated again and again.

In response to these interrupt pulses at RST-7.5 of each processor card after every 3.3 milli seconds, the earliest stored data sample is retrieved from the respective circular memory and it is brought to its buffer set. In this very location, the newly acquired data sample from remote station is written. The data samples from their respective remote stations are appearing within 3.3 milli seconds and not simultaneously at one time.

Now there is a question how and when these samples made available by each channel at its buffer must be taken into interface controller memory.

The sample data of each processor card (when taken out from its circular memory to the input of its buffer set) remains unchanged for 3.3 milli seconds and is replaced by the new sample data only when next interrupt pulse arrives after every 3.3 milli sec. at its RST-7.5. As explained earlier, these data samples do not appear at the input of their respective buffer simultaneously at one time. It is not possible to take these samples into Interface controller memory as and when these appear at their respective buffer set. If one takes the decision to send these data samples as and when these appear, then no other task can be done at all by the microprocessor of Interface controller. Also interrupt procedure may become unmanageable.

In order to solve this problem, the design scheme has been worked out as follows—
The data available at the input of all buffer sets of eight processor cards has been taken into the memory of interface controller by executing only a single interrupt sub-routine. The interrupt sub-routine is initiated by interrupting the 8085 microprocessor of interface controller at its RST-6.5 after every 3.3 mili-seconds. This interrupt pulse has been created separately and independent of all the other pulses of 3.3 mili-seconds interval which are generated by the eight processor cards of IFEH module. These pulses created independently at an interval of 3.3 mili seconds act as interrupts to interface controller only when the event at network level has been detected otherwise not.

All the eight processor cards receive their respective interrupt pulse at their RST-7.5 at different time with respect to each other but each of them continues to receive it with a periodicity of 3.3 miliseconds. The interrupt pulse at RST-6.5 of microprocessor of Interface controller also occurs with a periodicity of 3.3 miliseconds but it comes at different time with respect to the interrupt pulses appearing at RST-7.5 of processor cards of IFEH module. In case of every interrupt pulse discussed above (eight interrupt pulses - one each at RST 7.5 of eight processor cards of IFEH module and one pulse for RST 6.5 of Interface controller card) the periodicity of its occurrence is though 3.3 miliseconds but each interrupt pulse occurs at different time with respect to others, since the source of generation of these are different.
When a pulse interrupts the 8085 microprocessor of interface controller at its RST-6.5, the data samples available at that instant at the input of buffer sets of eight processor card are lifted in a few hundred microseconds one after the other. In between, again new data samples are brought from circular memories of all the processor cards to the input of their respective buffer sets. The new data samples at the input of buffer sets are made available as and when the interrupt pulse meant for that processor card interrupts it at its RST-7.5. The interrupt sub-routine meant for RST-7.5, if the event is true at network level, provides in the end to the input of its buffer set that data sample which is the oldest at this time in its circular memory. When next interrupt arrives at RST-6.5 of interface controller after 3.3 milliseconds, fresh data sample is present on each buffer set. In the similar manner as discussed above, these data samples are lifted again and again.

6.11. DESIGN CONCEPT OF STA/LTA TRIGGER AT NETWORK LEVEL

It is essential to set the recording to include data prior to the trigger point and after the event has subsided. An algorithm is designed to discriminate between the true events of varying amplitudes and the background noise. The problem is very severe when dealing with very weak events in the background of high level of noise. The worst case is when the area is seismically dormant and background is full of high level of cultural noise (spurious signal). Under
these circumstances the setting up of the trigger parameters for remote stations and then overall trigger conditions are very important design criteria.

While working out the design for Trigger algorithm based on STA/LTA ratio, the following points have been taken into consideration-

a). The number of outstations to be part of the trigger decision.

b). Over what time should these stations be registered.

c). How long should the LTA be taken over ?

d). How long should the STA be taken over ?

e). What is the STA/LTA trigger ratio ?

'STA' and 'LTA'

These are the long term average (LTA) and the short term average (STA) values. For each channel the average signal level is computed for periods set by the seismologist. The system further calculates the ratio of these values to determine if the seismic activity has increased or not. In general prior to an event, the long term value will be low as only local ground noise would be registered. When an event occurs, the STA value will increase all of sudden. Therefore, the LTA value is the measure of local noise and STA is a measure of a seismic event. For large events the ratio between STA and LTA will be high and the event is recognised. When the event is small and distant, the increase in event level over the noise may be small. If the large local events are only of interest
then the seismologist will set the ratio high. When weak micro seismic events must be recorded there is a danger that spurious local noise can create false triggers.

To guard against these, additional checks have been included. These two options are measures against this problem. This relies on the principle that local noise is local and will not be monitored on many stations. Therefore, if an animal disturbed the seismometer at an outstation, it is unlikely that other animal will be disturbing other stations also. However, a seismic event will be registered on all or many stations. The trigger processing system and intelligent front and hardware take this into account and depending on the setting made by the seismologist more than one station must exceed there STA/LTA ratio for recording to occur.

The setting for STA and LTA time period will depend upon experience but when long period seismic events are to be recorded both the STA and LTA time should be long. With higher frequency microseismic events, the parameters over which the averages are calculated should be reduced.

Under further off line processing, the system prints out a log in of the events listing the times when trigger occurred, the channel which triggered and the number of times the channels dropped below trigger level. In addition to this operation, the video can be set to monitor the individual outstation signal levels or their STA/LTA ratios. This is particularly useful when checking the performance of
outstations as it provides a visual monitor of their noise levels. If a station suddenly becomes noisy or totally quite, this could be indicative of a problem.

The algorithm has been designed to include

(i) The number of outstations to be part of the trigger decision (number of channels).
(ii) Over what time should these stations be registered (time window).
(iii) How long should the LTA be taken over for different stations?
(iv) How long should the STA be taken over for different stations?
(v) What is the STA/LTA ratio trigger?
(vi) What should be the post event time?
(vii) What should be the pre event time?

The LTA parameter is set between 1.0 second to 99.0 seconds at a step of one second. The STA parameter is set between 0.1 second to 9.9 seconds with a resolution of .1 second. The pre event period is recommended to be chosen between 5.0 seconds to 15.0 seconds. The post event period has been designed variable as two minutes, 5 minutes, 10 minutes, 15 minutes and 30.0 minutes. The trigger ratio can be set any value between 5 to 10.