Chapter 26

The Data Acquisition System for the GMRT

R. K. Singh

The Giant Meterwave Radio Telescope (GMRT), an array of thirty antennas, situated at Khodad, Narayangaon, is by and large completely controlled by observers, sitting at their desks, with the help of a centralized networked system of computers at the Central Electronics Building (CEB).

The software system that achieves this can be broadly divided into two parts, (i) one that deals with the control and monitoring of the antennas (ONLINE) and (ii) one that deals with the control, monitoring as well as data acquisition (and processing) of the digital backends (DAS). Both of these systems have a large number of small embedded subsystems that can be controlled across the network. In discussion in this chapter is limited to the DAS.

Any reasonable Data Acquisition System (DAS) for the GMRT must fulfill the following major requirements.

1. Interfacing with the hardware for acquisition of data.
2. Control of the correlator both for configuration as well as required periodic updates of parameters.
3. Monitoring of the status of the correlator and flagging the data appropriately.
4. Online processing of the data for reduction and archiving.

A functional description of the GMRT correlator can be found in chapter 9, and it is assumed that the reader is familiar with it.

26.1 Data Acquisition

Let us first estimate the maximum data rate produced by the GMRT correlator. For each antenna there are four associated data streams (two sidebands USB and LSB, in each of two polarizations RR and LL) each of 16 MHz bandwidth. Therefore, one requires four sampling points per antenna with sampling rate of 32 MHz. The present correlator has only 60 sampling points, and handles only one side band. For each stream there is a corresponding FFT unit in the correlator. This FFT unit carries out a 512 data point transform in real time, i.e. one 512 point transform every 16μsec. The 512 point transform corresponds to 256 complex numbers, i.e. 256 frequency channels. The time to acquire 512 data samples, i.e. 16μsec is called a FFT cycle. The number of distinct pairs of antennas (including an antenna with itself, i.e. self correlations) that can be made from 30 antennas is $30 \times (30+1)/2 = 465$. Therefore, 465 Multiplier and Accumulator (MAC) units are required to correlate all data from 30 antennas. Each MAC unit accepts 4 data streams i.e. two polarizations from two antennas (it makes no sense to correlate USB with LSB) and multiplies them to produce 128 complex numbers each for two polarizations, or 256 values for one polarization. In either case, it is 256 complex numbers, which the MAC units sum for a duration of a STA (Short Term Accumulation) cycle, which is 4096 FFT cycles. One STA cycle is equivalent to $4096 \times 16\mu\text{sec} = 66\text{ms}$. The MAC data format is such that 4 bytes encode one complex number. The actual number of MACs in the correlator is 176×3 (there are 3 Racks with 176 MAC units each) or 528, i.e. there are $528 - 465$ redundant MACs. Therefore the total amount of data produced per second per side band is $528 \times (1\text{sec}/66\text{ms}) \times 256 \times 4\ \text{ bytes} = 8$MB. The total data including both side bands would be $= 16$MB/sec.

16MB/s is a huge data rate to be sustained on any general purpose machine, or to be stored on any media. This means that would would need another piece of hardware in the correlator to carry out Long Term Accumulation (LTA). Such a hardware element was planned, but has not been implemented so far. Instead, in the present correlator system, the STA cycle has been configured for 8192 FFT cycles, i.e. the STA cycle duration is 132 ms. This brings down the sustained data rate per side band to 4 MB per second. Even this requires a special interface cards to input the data into the general purpose machine for processing. The host computer currently used for the DAS is a pentium based machine running the Linux operating system.

26.2 Correlator Control

The correlator system for the GMRT serves the following four major functions depicted in the schematic shown in the Figure 26.1.

1. Analog to Digital Conversion.
   As we mentioned earlier that the full GMRT would require $30 \times 4 = 120$ sampling points, for thirty antennas with two side bands in each of two polarizations. The current correlator system suffices half of this requirement. The sampling takes place at 32 MHz in order to have data for a maximum of 16 MHz bandwidth. There is no control required for this unit of the correlator.

2. Delay DPC unit.
   Signals originating from a given point in the source, and received via two different antennas, traverse different path lengths before they arrive at the samplers. This different path lengths arise because of the different locations of the two antennas as well as the different cable lengths between the two antennas and the correlator (and are called the geometric delay and the fixed delay respectively). As discussed in chapter 4 the geometric delay changes with the Hour Angle of the source. In order to compensate for the differential delays that the signals from different streams have suffered, the Delay unit has to be periodically updated with the current values of time delays that have to be applied. Therefore, delay values are required to be transmitted from the host computer down to this unit of the correlator periodically. The Delay unit of the correlator can delay only for the integral number of sampling time intervals, which is 1/32MHz. Finer delay corrections are made in the FFT unit, where delay values are converted into a phase gradient across the band.

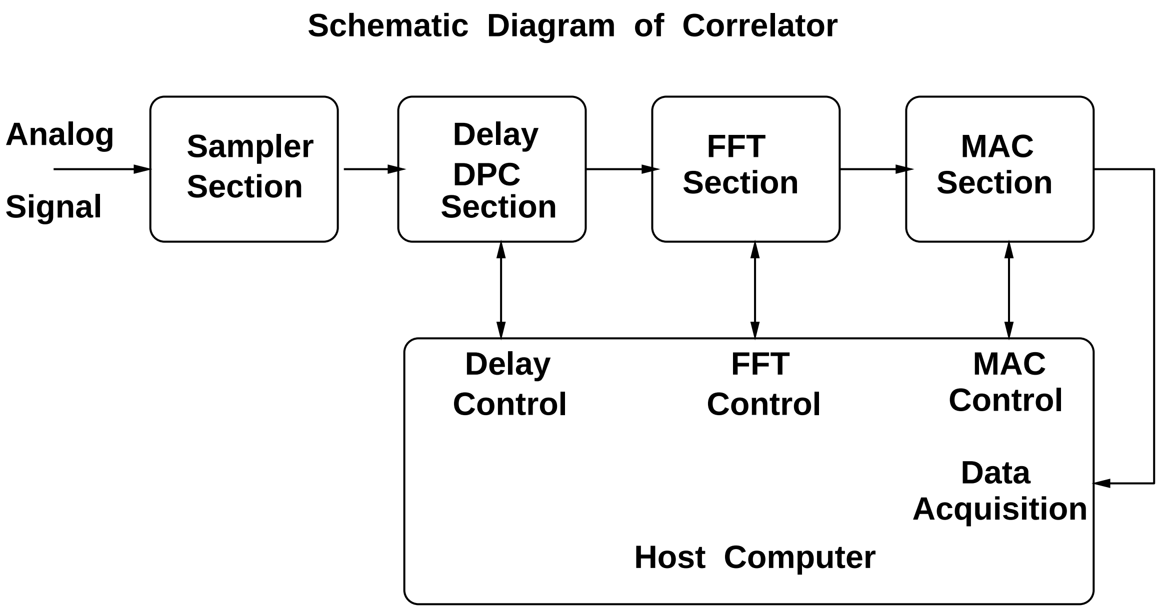

Figure 26.1: The schematic block diagram of the correlator. The four major blocks of the hardware as well as the host computer are shown.

3. FFT unit.

   This section of the correlator carries out 512 points FFT every $1/32\mathrm{MHz} \times 512 = 16\mu$sec. The two other major tasks that this unit performs are, 1) fringe phase subtraction, and 2) fractional sampling time delay correction (FSTC, see chapter 4). This requires periodic updates of the phase and FSTC values that are to be applied, which has also to be supplied by the host computer.

4. MAC unit.
   The MAC unit can be configured in a variety of modes. This configuration is usually done by the host computer during the initialization sequence.

All these units (except the samplers) need to be initialized. The exact initialization required depends on the observing mode. To achieve this, at the start of the observation appropriate pieces of programs are loaded into the controllers, which then behave like embedded systems.

26.3 Monitoring the health of the correlator

The quality of the data acquired, depends on the health of the correlator for the duration of the observation. It is desirable to have flag bits in the recorded data indicating the state of the correlator at the time of the observation. There exists a planned set of parameters that are to be monitored and a method of transmitting such information to the host computer exists, but the actual monitoring has not been implemented yet. Hopefully, this will be done sometime in the future.

26.4 Online processing of the data

26.4.1 The network of acquisition and processing

The actual data acquisition and online processing of the data are carried out over a network of computers. The connectivity of the network is shown in Figure 26.2.

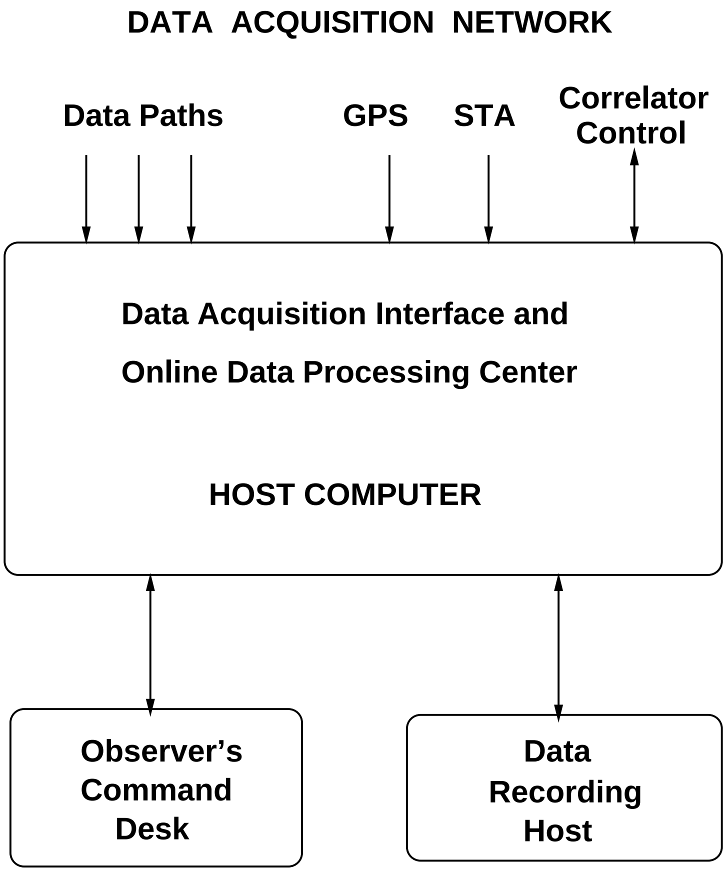

Figure 26.2: The various connections that exist to the correlator host computer. The connections on the upper side are to the correlator hardware, while on the lower side are to other computers on the observatory network.

The present correlator handles one side band of 16 MHz bandwidth. The MAC unit consists of three racks, and the data is hence received over three channels. The host computer uses a custom made card to accept data at three separate ports. In order to maintain standard time for reference for the data blocks, the host computer is also provided with minute pulses from the Global Positioning System (GPS). One of the fundamental cycles of the correlator operation is the Short Term Accumulation (STA) cycle, over which the MAC unit accumulates the data; it is equivalent to 8192 cycles of FFT. The clock edges that signify the beginning of STA cycles are also provided to the computer. The STA cycle timings are used to maintain the synchronization in the communication between the host computer and the correlator components. This STA clock edge also signifies the beginning of the STA data cycle. The actual communication for the control of the correlator components takes place via another custom made, in-house developed card which can communicate to sixteen different components sequentially.

So far, the connections were described on the correlator side of the interface (host) computer. The major data processing and correlator control is carried out by the same host computer. On the other side of the interface computer is the 'network' i.e. a connection via the standard Internet Protocol over Ethernet. One of the computers on this network side is used by the observers to control and monitor the entire GMRT system. Commands are issued by this computer ('Observers Command Desk' in Figure 26.2) to the host computer which in turn manages the correlator. Another network component, the 'Data Recording Host' (figure 26.2) is used for data storage and off-line processing. This computer is the main compute server and has a huge data storage capability.

26.4.2 The software layout

A large number of separate programs work together in order to achieve the parallelism that is a natural requirement for a real time acquisition, processing, and control. These units will now be described one by one. The Figure 26.3 shows the important components of the whole package.

The low level programs

There are three low level components of the package that establish communications with correlator.

1. The data collector.
   As stated above each MAC racks passes data over a separate channel, therefore, this low level software, the data collector, accepts the data from three channels. Each MAC rack is composed of 11×16 MAC chips each of which gives out 256 complex numbers per STA cycle. Each complex number is coded in 32 bits, i.e. two 16 bit real numbers. Each MAC chip has two buffers; one is used for accumulating the incoming data over a STA cycle, and another is used for transmitting data accumulated during the previous STA cycle. Therefore, one MAC rack outputs $11 \times 16 \times 256 \times 4 \ \text{bytes} = 176$ KB of data per STA cycle. One of the bits of the 16 bit word is reserved to indicate the beginning of a fresh STA cycle.

   We use a custom made PCI bus based interface card to input this data. The interface card has four ports, one of which is unused. These ports are 16 bits wide data ports with separate control lines. Three of the ports are configured to accept the data on an independent external clock. The interface card has total 64/128 KB total memory, which is equally distributed among the 4 ports. Three ports are configured for streaming mode operation, where the 16/32 KB memory for a port is divided into two halves, such that when one half is full, an interrupt is generated for the host computer's CPU for transferring this data into the memory of the computer. While this transfer continues from one half of the data, the external data continues to fill the other half of the memory for each port. This process continues endlessly.

   The user level software programs are expected to process the data at this rate on the average. This low level program receives the data from all the three channels and stores them in separate local buffers. For every block of data corresponding to the half of port memory (8 KB/16 KB, depending on the port memory size), the PC time is noted. This time corresponds to the end of the block of data just arrived, and can also be thought of as the time corresponding to the beginning of the next data block. This time will be used for time stamping of data by the next level software referred to as 'acq' in Figure 26.3.

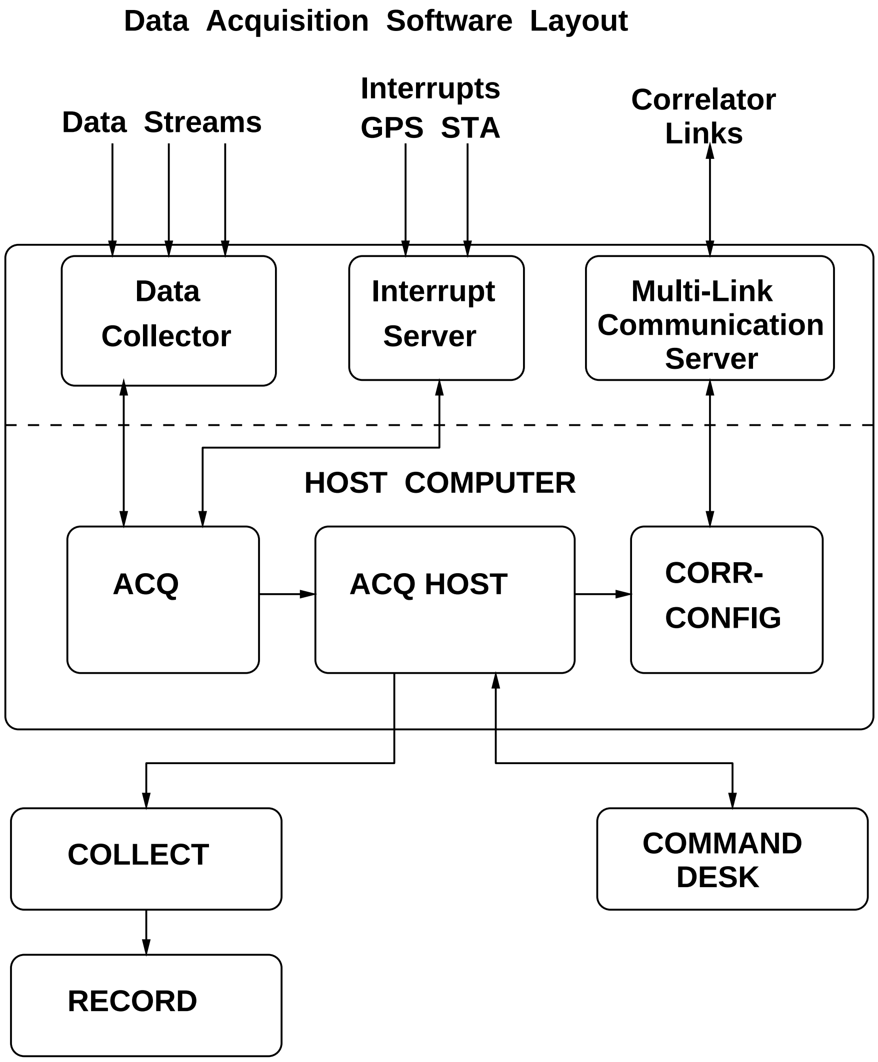

Figure 26.3: All the major programs used in DAS are shown here. The components in the solid box and above the dashed line are 'low level programs' that deal with the hardware interface. The ones inside the solid box but below the dashed line are 'high level programs' which deal with controlling the hardware as well as collecting and processing the data. The programs below the solid box are 'peripherial' or 'network' programs.

The low level program makes no effort to look inside the data buffers and synchronize three streams into one logical stream.

The data collector stores a good fraction of a second's worth of data in three separate circular buffers. Hence it is highly unlikely that the higher level program 'acq' (even if it is temporarily busy elsewhere) will be unable to drain out this data before the buffers overflow.

2. GPS and STA interrupt server.
The GPS minute pulse, and the STA clock edge are used to generate interrupts to the host processor. When an interrupt occurs, the interrupt server program notes

down the PC time and maintains a list of last few such times. On request from the higher level programs, the interrupt server simply supplies this list (for the GPS or STA interrupts, as requested). For both STA and GPS the time interval between two successive interrupts should in normal circumstances be a fixed constant, and therefore, the interrupt server can optionally be requested to build statistics on time intervals at which the interrupted occurred and attempt to interpolate over temporary glitches. Note that as discussed further below the PC time noted by the interrupt server could in principle be wrong for a variety of reasons. These errors need to be recognized and corrected for by the higher level programs.

3. Multi link communication server.
   This low level program communicates with the correlator through a home made custom communication protocol. It is primarily used for configuring the correlator. A higher level correlator configurator program sends and receives data through this low level program.

The high level programs

There are three high level processes that run on the correlator host computer which are responsible for the real time processing of the data as well as correlator control.

1. The acquisition center, the 'acq' program.
   This program has three major responsibilities.
   (a) It acquires the data from the data collector, and looks for the STA cycle markers described above.
   (b) It still maintains the data in three separate streams, but marks each data block in each stream with a sequence number. The sequence number is derived from the time noted as the beginning of STA cycle for that block. This basically synchronizes the three streams.
   (c) The program places the data (in three separate circular buffers), and also considerable book keeping information in a common memory resource (shared memory). This shared memory can be accessed by other programs for further real time processing. Note that more than one program can simultaneously read the shared memory, but only one program, viz. 'acq' can write to the shared memory.
2. The correlator configurator, the 'corr_config/fstop' program.
   This program is has two main functions
   (a) To initialize the embedded correlator controller cards and prepares them for the specified observation mode.
   (b) To periodically set the model values (delay, fringe stop, FSTC). The delay value is transmitted to the Delay section of the correlator and the fringe stop and FSTC to the FFT section.

   The current model values have to be down loaded to the embedded systems at a time and in a manner such that the normal workings of those systems are not affected. Further the protocol should be such that higher level program must know in advance at what instant of time these new values will be made effective by the low level software. For this purpose, two basic norms are followed.

(a) The model values are down loaded at times which are not close to the beginning of a new STA cycle. At the start of an STA cycle the embedded systems have several time critical tasks to perform.

(b) The embedded systems are initialized with a STA cycle sequence number. This sequence number is incremented by all the embedded systems at the start of new STA cycle, so that all the embedded systems are properly synchronized. When model values are down loaded, the configurator also passes the STA sequence number at which these values are to be effected. The activity across all the components of correlator are hence synchronized. The protocol takes into account the fact that erroneous conditions which require corrective action could arise.

3. The acquisition manager, the ‘acqhost / acq30’ program.
This program (which is the the central program) receives interactive commands on one side (i.e. over the network) and on the other side (i.e. the correlator host computer and the correlator’s embedded systems) controls all activities much like a band master. It is the responsibility of this program to continuously ensure the logical coherence of the observation.

When ‘acq30’ is started, it scans a set of files (that could also be specified via user commands) that describe the correlator hardware configuration. These files contain information such as what are the coordinates of the antennas, which antenna is connected to which samplers, which samplers is connected to which FFT pipeline, and which FFT pipelines are connected to which MAC chips. From this information ‘acq30’ builds a complete mapping of all the connectivities in the correlator. ‘acq30’ reads the data from the shared memory created by ‘acq’. Recall that in this shared memory one has ’raw’ data, i.e. the data is organized into three streams (one per rack) and in each stream the ordering depends on the ordering of the MAC chips in that rack. ‘acq30’ merges these three streams into one stream using the sequence number information planted in each data block by ‘acq’ (see above). Further it also notes the common IST arrival time of the data corresponding to a given STA sequence. It was originally intended that the merging of the three streams of data into one stream would be effected by the embedded software of the correlator, but this was not realized. Once the data has been merged into one stream, ‘acq30’ uses its knowledge of the correlator connectivity, to determine which piece of the data corresponds to which antennas, and which samplers, and which FFT pipelines. The ‘acq30’ program however has no way of determining whether the specific antennas whose data is being read are all pointing to the specified source or not. ‘acq30’ simply trusts that when a scan is started (see below) the antennas are pointing at the specified source. Given the source position and the antenna co-ordinates, ‘acq30’ computes the appropriate delay, fringe stop and FSTC values. The calculation of such quantities for a given point of time is referred to as ‘model calculation’, and the values as ‘model parameters’ or ‘model values’. For as long as a given scan continues, ‘acq30’ computes the model parameters periodically and sends them to the appropriate program (corr_config/fstop) to be set. From its mapping of antennas to samplers to FFT pipelines, ‘acq30’ can determining which delay value to send to which delay control card etc.

‘acq30’ also converts the data format from the ‘MAC format’ described above (i.e. 4 bytes for each complex word) to the IEEE standard floating point format. The Long Term Accumulation (LTA) is also performed by this program. The observer specifies the number of STA cycles for which the data should be further summed before recording. This summation is carried out at the same point as the conversion to IEEE floating point format. After performing LTA 'acq30' sends the resulting data on the network to a separate program 'collect' which runs on another computer (the 'Data Recording Host' in Figure 26.2). 'collect' makes this data available to other monitoring and recording programs.

'acq30' accepts (and passes on to down stream programs) several user commands, of which the four major ones are:

(a) Init:Initiate an observation session. During initialization time parameters which will remain constant during the observation session (eg. the bandwidth, the polarization mode, the number of channels to record, which antennas to record etc.) have to be specified.

(b) StartScan: Start a scan on a specified source. Required parameters include source position, observing frequency, observer's name etc.

(c) StopScan: This indicates the end of the present scan. At this stage one can start a new scan on the same or different source using the 'StartScan' command.

(d) End Close the observation session. On receiving this command 'acq30' informs all other dependent programs to shut down and then shuts itself down.

The peripheral programs on the network

There are three other peripheral programs on the network that complete the entire data acquisition chain. They are:

1. 'dassrv'.
   A program 'dassrv' running on the 'Observer's Command desk' (see Figure 26.2) interfaces between the main GMRT control program 'ONLINE' and 'acq30' running on the 'Host Computer'. An observer sitting at this desk hence has therefore control over both the antenna system and the data acquisition system. Commands can be entered from this desk interactively or via an observation file.

2. 'collect'
   As discussed above once the LTA of the data has been done by 'acq30', it is sent (along with the associated model values, times, and flags) to 'collect' (which runs on the 'Data Recording Host' in Figure 26.2). 'collect' performs the simple job of creating a shared memory resource and placing the data it receives into it in a well specified format. Any number of programs can then access this data either for recording, archiving, or online monitoring.

3. 'record'
   This is the last in the chain of Data Acquisition System (DAS). As stated earlier, this program takes the data from the sharable resource created by 'collect' and writes the data in a specific format (locally called the 'lta format') onto the hard disk. From here it can be transferred to tape for long term storage.

Sequence of program execution

When the correlator host computer boots, the three low level programs (in Figure 26.3 are started. The correlator itself is usually started before this computer is booted, but in principle the correlator can be brought up at any time before higher level programs are executed. Next, 'acq' is started, and it connects to two low level programs, the data collector, and the interrupt server. As long as 'acq' is alive it takes data from the data collector, time stamps it using the information from the interrupt server (for more details on time-stamping see below) and then places the time-stamped data in a shared memory. 'acq' does not care whether any such programs exist to use this shared memory.

At this point, the correlator configurator program (corr-config), is run to setup the correlator in the required mode. Ideally, this should be done when 'acq30' is instantiated, but as of now this is not recommended. 'acq30' and the peripheral programs ('collect', 'dassrv', 'record') can then be started. At this point the entire data acquisition chain is up. The 'record' program can be started anytime after 'collect' has been started. Any number of 'record' programs could run at a time. Similarly, any number of monitoring and online display programs can run simultaneously to follow the progress of the observation.

A large number of small tools are developed to look at the raw data as placed in the shared memory of 'acq' on the host computer. These programs provide invaluable tools to debug the correlator problems, and also to make consistency checks at run time. These programs do not interfere with the normal observations, therefore, any number of them could be run at a time.

26.4.3 Time stamping of data and it's accuracy

The time at which a given astronomical signal was received by the GMRT is a fundamental parameter. At the GMRT we attempt to stamp the data with an accuracy of better than 100μsec. This time resolution is very good given our baseline lengths and operating frequencies.

A complex algorithm is followed to keep the time information to within 100μsec accuracy. As stated earlier time information is collected from three sources, (i) the Host Computer Clock (usually called the 'PC Clock'), (ii) the GPS minute pulse and (iii) the STA pulse. All these sources could possibly be in error. Below we describe in more detail the time information available.

1. The most basic source of time is the data itself. Recall that the low level program, 'data collector', reads the PC time at the end of every 16 KB of data that is received from the MAC. This time information is available to 'acq' along with every block of data. The 'data collector' has no knowledge about where in the data a new STA cycle begins because it simply collects the data coming at an uniform rate as a continuous stream. 'acq' figures out the beginning of the STA cycle, by looking for the synchronizing bit in the data (see above). Once the synchronizing bit is found 'acq' can use the time-stamping of each data block to associate a PC time to the beginning of the data block of a given STA cycle.

2. The GPS minute pulse is provided to the Host Computer as an interrupting pulse. The 'interrupt server' reads the PC time at this event and maintains a list of the last several such times. 'acq' goes through this list, looking for a set of contiguous events for which the interval between successive events is the same to within 100μsec. Once it finds such a set, it uses their time of arrivals to establishes a linear equation between IST and the PC time. This equation (which is communicated to all programs which require to know the exact time) is used to go back and forth between IST (which is essentially what is required for astronomical purposes) and the PC clock, (which is what is readily available to all programs). The linear equation accounts for an offset as well as a constant drift between the PC time and IST. The equation is continuously updated using the latest "good" GPS pulses, so higher order drifts are also taken care off.

3. At the start of a new STA cycle a synchronizing pulse is sent to all the components of the correlator as well as the host computer. This pulse is picked up by the 'interrupt server' which maintains a list of the time of arrivals of the last several such pulses and provides it to 'acq' on request. As discussed above, this same information is acquired also from the synchronization bit in the data. This redundancy allows for a consistency check to be made. The STA interrupts are used for two purposes (i) to make this redundancy check, and (ii) to set up a linear equation between the STA pulse time of arrivals and the PC time. Once this equation has been set up, a sequence number can be assigned to any given STA pulse based on its time of arrival.

There are several possible sources of error in the time keeping, viz.

1. It is possible that the system (OS) may be busy when a given event (say GPS or STA pulse arrival) took place, and by the time it registers this event and notes down the PC time an unknown delay has occurred.
2. The basic unit of scheduling for the OS is a unit called one CLOCK TICK (10ms), and occasionally the OS makes an error of exactly one CLOCK TICK. This error is the simplest to detect and correct for.
3. Sometimes, (mainly because of hardware glitches), an expected event does not take place, or there a number of spurious events.

The fundamental assumption that is used in correcting for these errors is that the GPS cycles, STA cycles, and the STA coded in data are all driven by external agents, therefore, even if there is a momentary glitch, sanity will prevail again after a while. Similarly drifts in the time of arrivals are expected to be slow and not discontinuous. The only discontinuous event that could occur is the missing of one CLOCK TICK, but since this leads to an error with a well determined signature (a jump in time by exactly one CLOCK TICK), it is very easily recognized. In practice this sort of error rarely occurs. 'acq' uses a number of complex heuristics as well as continuous redundancy checks to interpolate short term hardware glitches and experience has shown that it is quite robust to short term glitches.

While ignoring short term glitches, 'acq' should nonetheless track slow drifts of the PC time. To do this, after every 16 STA cycles it makes an attempt to arrive at a new equation between the STA sequence number and PC time. This equation is not accepted unless it is found to be matching within 100 microseconds of the previous equation. Similarly, every minute when a new GPS pulse is received, an attempt is made to arrive at a new equation.

Sometimes 'acq' cannot update its equations for more than a threshold time interval (because the updated equations are very different from the existing equations). This indicates that a much more serious problem has occurred. Generally what has happened at this point is that the PC time has jumped. 'acq' then attempts to reestablish equations afresh from a reasonably long sequence of good time intervals for STA and GPS. For example, it is demanded that a contiguous set of 4 GPS pulses must be within 100 microseconds of error, before the IST equation is accepted. Since 'acq' is starting afresh, the deviation from the previous equation is not checked as it would normally be. Similarly for the STA equation it requires that least half of 64 intervals in a stretch of 64 STA cycles must be good before deriving a fresh STA equation. When the PC time jumps a time accuracy of less than 100μsec is not guaranteed for the time interval between the jump and the time when a new set of equations are established. However, there are a limited number of environmental factors which cause the PC time to jump. These are known and are avoided during observations.

26.5 Further desirable features

A few of the features that would be desirable are:

- Monitoring the health of the digital system (correlator) is possibly the most important in this list. One should identify the parameters that are to be monitored and define actions to be taken on specific conditions. The conditions should also get flagged in the data.
- The communication between the 'acq30' and the 'corr-config' is not good yet. It causes variety of problems. For example, the 'acq30' has no way of figuring whether the control values that it wanted set at a given time have been set, and if so, whether at the exact time specified.

  Therefore, a better connection with a well defined protocol, must be created between 'acq30' and correlator configurator in order to achieve the following goals.

  - To reduce the time delays between the time when the request is made and the time when the control values are set in the correlator.
  - To have better timing control in setting the values.
  - To be able to report back to 'acq30' the errors encountered by the configurator, so that the corrective measures are initiated to counter the error, and also to allow 'acq30' to flag the data appropriately.
- Communication between the program for 'online control of antenna' and 'data acquisition system' is truly minimal. A better ties with the 'online system' is required, so that the problems in antenna pointing, or with the analog systems can be reported to the 'acq30' for appropriate flagging of the data. At the moment an user has to take the log file generated by 'online system' and associate such conditions to the data manually.

The critical low level tools and routines already exist for the fulfillment of the items in the list.