PRL TN-80-2002

PRL TECHNICAL NOTE

Automating a networked telescope on an e-mail trigger for rapid observations of Gamma Ray Bursts

D. V. Subhedar, T. Chandrasekhar & N. M. Ashok

Physical Research Laboratory
Navarangpura, Ahmedabad 380 009

PRL TN-80-2002

PRL TECHNICAL NOTE

Automating a networked telescope on an e-mail trigger for rapid observations of Gamma Ray Bursts

D. V. Subhedar, T. Chandrasekhar & N. M. Ashok

Physical Research Laboratory
Navarangpura, Ahmedabad 380 009

Automating a networked telescope on an e-mail trigger for rapid observations of Gamma Ray Bursts

D. V. Subhedar, T. Chandrasekhar & N. M. Ashok
Astronomy & Astrophysics Division
Physical Research Laboratory
Ahmedabad 380 009

Abstract: Gamma Ray Bursts are transient in nature and their timing and position in the sky is unpredictable. Many spacecrafts continuously monitor the sky for their occurrence. When a burst is detected, the information on its time, location and strength is immediately passed on to the ground based central agency, GCN. The GCN promptly sends e-mail messages to the registered observatory sites all over the world. Since the entire phenomenon is random and short-lived, only an automated telescope triggered by an incoming e-mail message can record such an event. One such automated recording system has been successfully developed and will be described in this technical note. The software developed for this can be used for controlling not only a telescope but also any networked system remotely by just sending an e-mail message from anywhere in the world.

1. Introduction

Gamma Ray Bursts (GRB) are intense flashes of high-energy photons that occur randomly and are uniformly distributed on the sky. They were discovered in 1967 but were not detected at other wavelengths (X-ray, UV, optical, IR, and radio) for another 30 years due to lack of rapid and precise localizations. The breakthrough came about in 1997 with the advent of the Beppo-SAX X-ray satellite and the first detection of the optical counterpart of a GRB named later as GRB970508. The occurrence rate of GRBs is approximately one per day at random points in the sky.

It is now widely accepted that GRBs originate at cosmological distances with isotropic energy releases ranging from 10^{51} to 10^{54} ergs. Current theoretical models invoke collapse of a massive star in a blackhole or the merging of two compact objects to explain the immense energy released. GRBs are expected to emit most of their visible light when the burst's narrow jet or turbulent expanding fireball interacts with the surrounding medium. The bulk of the burst's energy is between 100KeV to 1 MeV. The GRBs exhibit a great variety in their temporal behavior. The light curves of the observed GRBs show that they may last from about 30 milliseconds to almost 1000 seconds. A few bursts evolve smoothly to lower frequencies (such as X-ray, optical, IR, and radio) as the time passes. This afterglow mechanism is not well understood. Most of the resulting afterglows are faint ($m_v \sim 16$ to 21) and last only for 12 to 24 hours. A very bright GRB $m_v \sim 13$ (visual) can fade to $m_v \sim 24$ (visual) – a decrease in flux by a factor of 25000 – in less than a day. This fast fading implies that in the study of GRB afterglows, speed is most essential.

GRBs occur at random times from unpredictable directions. Also, even after detecting a GRB, it is difficult to determine the direction of arrival of the gamma ray photons due to the poor angular resolution of the detectors- they can be detected only within ~10 sq. arcmin with the present day detectors. Pinpointing the celestial coordinates of the bursts would benefit greatly, the study of the GRBs. The recent observations of these bursts from many satellite based platforms carrying Gamma Ray and X-ray detectors and ground based and satellite based optical telescopes (including Hubble Space Telescope) have given boost to their studies and many theories have been suggested about the mechanism of their generation.

A number of spacecrafts in orbit like High Energy Transient Experiment (HETE2), Rossi X-ray Timing Explorer (RXTE), Interplanetary Gamma Ray Burst Timing Network (IPN) consisting of Ulysses/NEAR/GGS-Wind missions, and BeppoSax (SAX) detect GRBs and provide directional information with some error-box. The onboard process for responding to a burst, however, takes some time that varies with spacecraft. The IPN takes a few hours to pinpoint a GRB but the HETE2 spacecraft does it much faster (within a few minutes). One of the methods of GRB position determination on the ground is to follow it up at other wavelengths (for example in IR and optical) with an instrument of appropriate field of view and angular resolution (~seeing limited) once its coarse direction is known.

The advent of the High Energy Transient Explorer Satellite (HETE2) has speeded up the burst response times significantly. The expectation is that observatories will obtain precise (arc minute) location of GRBs (through an e-mail message) within a few minutes of an occurrence of a burst. This will allow astronomers to see a burst as it occurs, at least for a burst of a longer duration even after allowing moderate slew time for a telescope to point to a GRB in the sky. In case of a shorter duration burst, the afterglow observations can be immediately started. It will be desirable that the telescope starts tracking the GRB within a minute or so after receiving the e-mail message notifying the GRB event. Considering that the message itself is sent after a few minutes of detection of the GRB by the HETE2 spacecraft and the intercontinental propagation delay of a minute or so is required for the e-mail to arrive, it is more than adequate to check every new e-mail message at a one minute time interval. A typical 8 inch amateur telescope equipped with a good quality CCD camera should reach, in a dark site, $m_V \sim 18$ (visual) with an exposure of a few minutes. Such a set up might catch the brightest afterglow even 24 hours after the burst. In the CCD image containing many objects, the GRB source can be identified by its diminishing intensity in its successive images [1,2].

Multi wavelength information of GRBs and their afterglow is of crucial importance for understanding and constraining the emission mechanisms of the GRBs. A red shift in the optical spectrum of the GRBs will provide the evidence of their extra-galactic origin.

The GRB Coordinates Network (GCN), at GSFC, NASA, USA, collects the Right Ascension (RA) and Declination (Dec) coordinates of the GRBs detected by the spacecrafts mentioned above and distributes that information in near-real time to the registered sites (numbering about 250 till this date) all over the globe. These sites initiate rapid optical follow up observations with their instruments. The GCN uses a number of methods for distributing the notices containing GRB coordinates and are listed below.

- **Dedicated phone:** Continuous phone modem connection resulting in high line charges
- **Internet Socket:** Fast and suitable for automated instruments
- **E-mail:** To any network address
- **Dialed phone:** Slower but cheaper than dedicated phone
- **Pager:** RA, Dec, UT and intensity displayed on a pager.

For the sites (receiving GCN notices) outside USA, only the Internet Socket and E-mail distribution methods are of any use, the former being faster but requires a separate IP address. We at Astronomy & Astrophysics Division of PRL, have presently joined the GCN for receiving e-mail notices and receive following types of **GCN E-mail Notices** on our e-mail address taken specifically for this project:

- RXTE_PCA
- RXTE_ASM
- IPN_POS
- SAX_WFC
- HETE and
- ALEXIS

All these have a similar format. The minor variations in their format have been taken care of in processing the text of their message while extracting the RA and Dec values of the GRB position notified therein. These E-mail Notices are sent from the GCN in a "TOKEN: Value" format, chosen so, because this format is the best compromise between a human readable and a machine-readable. A human readable format is useful for sites having manual control of the telescope. A machine-readable format, on the other hand, is suitable for the sites having daemons waiting for the incoming notices for automating their telescope instrument. **One such automatic system for a robotic telescope control has been developed for our site and is the subject matter of this note.**

In addition to notifying the real GRB events, the GCN, everyday, sends four to five Test Notices for dummy events at random, to enable the sites to test the readiness of their instrument set up for recording of the GRBs either manually or automatically. A printout of one such Test E-mail message is attached as **Appendix A1** of this write-up. The authors found these test notices very useful for extensively checking their automated recording set up during the development of the project. The **Appendix A2** is a printout of an e-mail notice for a real Gamma Ray burst detected by the HETE2 spacecraft.

2. The Instrument Set up

The instrument set up for the optical follow up observations of a GRB event consists of a **robotic telescope (8" Meade LX200** Schmidt-Cassegrain), a **CCD camera (Celestron PixCel 255**) and a **Pentium PC**. (See **Fig 1**) and the set up is installed in a small observatory building near the main telescope building at IR Observatory Campus, Gurushikhar, Mt. Abu, Rajasthan. The main telescope building houses the 1.2-m IR telescope. The **robotic telescope** communicates with the **PC** on a serial port **COM1** and the **CCD camera** is interfaced on a parallel port **LPT1**. The **Pentium PC** is networked with the PRL LAN on a 64KBPS link between Gurushikhar and PRL Main Campus at Ahmedabad. This **Pentium PC** works as a client for the **IMAP4** mail server in the PRL Computer Center for accessing GCN e-mail notices. When initiated at the start of the observations for the night, the software program connects to the **IMAP4** server on a communication socket every minute (the reasons for choosing this period have been already explained above) and checks for a new e-mail notice for a Gamma Ray Burst. Upon receipt of such a message, the text of the e-mail is parsed for extracting the RA and Dec coordinates of the GRB position and a command is given to the telescope to slew it to that location and start tracking it. The **Meade LX200** telescope goes in a **Track Mode** by default, once the slew motion is completed, without requiring a command to do so. With the help of **Windows Multimedia** support available on the **Pentium PC**, a pre-recorded audio announcement for a fresh GRB event is played on the speakers connected to the **PC** via a standard sound card. This announcement informs the observatory assistant present on the telescope floor of a fresh GRB event to enable him to start the CCD camera for the GRB follow up observations. (Alternately, the CCD camera could be commanded in advance in the evening to acquire and record images of the field of view of the sky seen by the telescope at a regular interval throughout the night. In that case, no action from the observatory assistant will be required when the GRB message arrives and the recording will be fully automatic). As an additional precaution, simultaneously, a telephone modem connected to the second serial port **COM2** of the **Pentium PC** rings a selected telephone extension in the office/residence of the Observer. When he lifts the telephone handset, he hears through the telephone receiver the same audio announcement of a GRB event. The software does it automatically by using the speakerphone commands of the modem. This alerts the observing scientist who may initiate any further action (like tracking the GRB event with the main 1.2-m IR telescope suspending temporarily his ongoing science programme etc). The entire operation is completed within a maximum of 80 seconds (explained later) from the receipt of a GCN e-mail notice. This is a completely automated operation and runs unattended throughout the night waiting for a GRB event to take place.

3. The Challenge

A completely unattended operation to automatically follow up a GRB in the sky within a minute from receiving the e-mail notice from the GCN was a challenge in itself as no readymade application software was available for the same.

A lot of literature survey was done in the journals on Astronomy and Computer Software magazines as well as on the Internet without success. It became obvious that we had to develop our own program package suitable to our needs. Many commercial mail utilities like Pine, Outlook Express, Yahoo Mail etc were also studied. These utilities can be instructed to automatically route the incoming e-mail message to various folders by predetermined rules on the basis of its source address, the subject line and the presence of a search string pattern etc. These utilities, however, cannot process the text of the e-mail message and hence cannot be tailored to control a robotic telescope without a manual intervention.

A Windows based application was our first choice for achieving a better control of the computer hardware. A search was started for identifying the Windows version of the C++ language that supports network programming for e-mail access from a remote server and telescope control from a serial port. After a lot of thinking it was decided that two packages (one in Visual C++ and one in Visual Basic) would be required to be developed in a multitasking mode – one for communicating with the e-mail server and the other for the telescope control. The Windows 98 was the best operating system for this as the CCD camera that is used as a backend instrument for this telescope also has control software that is Windows based. A Windows operating system environment is device independent and is best suited for multitasking, without any clash amongst the tasks particularly when they are controlling peripheral devices on the computer ports [8].

The Microsoft Visual Studio 6.0 supplies both of these required components (- the **Microsoft Visual C++ 6.0** and **Microsoft Visual Basic 6.0**) on the same CD and was found to be ideal for our application. We decided that both the application programs, one for the network access and other for telescope control, running simultaneously in a multitasking mode, could communicate with each other through a stored hard disk file through a common path for achieving a completely unattended operation. **Appendix B** gives the flow chart of the complete software developed for the purpose of automating the robotic telescope.

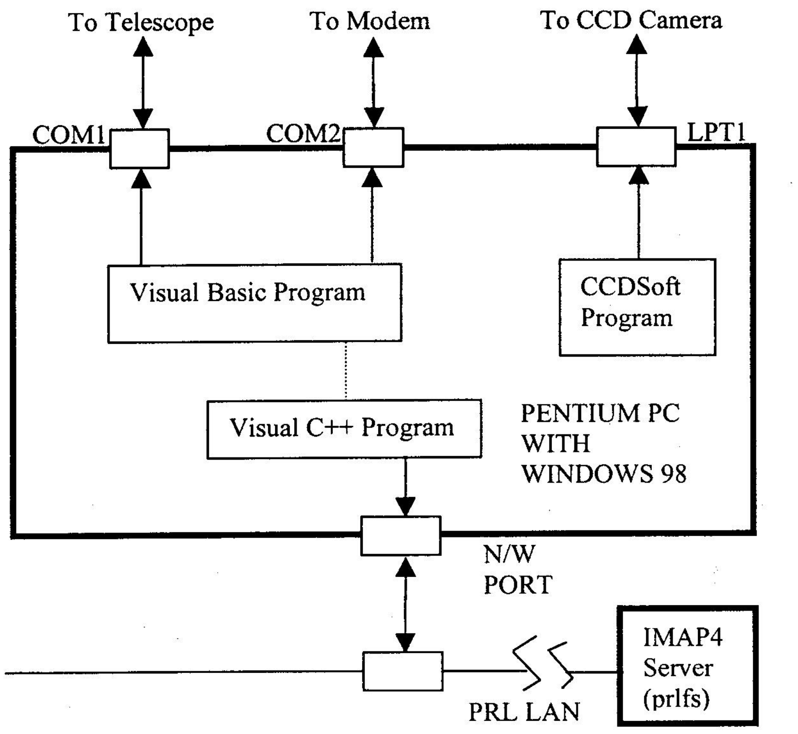


Fig 1:THE INSTRUMENT SET UP FOR AUTOMATIC TELESCOPE MOTION FOR TRACKING A GAMMA RAY BURST

4. The Hardware and Software Components of the Project

4.1 The Pentium PC

Any Pentium PC with Windows 98 Operating System installed on it is suitable for this program. The PC should have 2 serial ports, 1 parallel port, and an Ethernet connection for the LAN and Multimedia support. A 20 GB hard disk is good enough for a dedicated system.

4.2 IMAP4 Server

IMAP4 is the fourth revision of the **I**nternet **M**essage **A**ccess **P**rotocol and deals exclusively with the handling of e-mails on a client side. An **IMAP** client can manipulate a remote mailbox in many ways. It can fetch the body and/or the header of the message into the client machine leaving the original still in the server mailbox till it is specifically commanded for its deletion. The **IMAP4** protocol assumes a reliable data stream such as provided by the Transmit Control Protocol (TCP).

An **IMAP4** connection consists of the establishment of a client/server network connection, an initial greeting from the server, and subsequent client/server interactions. All interactions transmitted by a client and the server are in the form of lines; that is, character strings that end with a Carriage Return and a Line Feed. A client command begins an operation. All commands are numbered sequentially [3].

We have developed a client program for fetching a GCN e-mail notice from the ***prlfs*** server in the PRL Computer Center under the **IMAP4** protocol on port no. 143. The program software was developed in **Microsoft Visual C++ 6.0** using its Integrated Development Environment (IDE) as a Win32 Console Operation employing Windows socket functions. The program is described below.

4.3 Microsoft Visual C++ 6.0 program

This program in Visual C++ (VC++), is initiated everyday in the evening at the start of the observation session. Prior to this, the observer manually logs on to the *prlfs* server and empties the inbox of any previous e-mail messages. The observer also deletes all the text from a particular file on the hard disk (providing the link between the VC++ and VB program as explained later) storing a previous GCN E-mail Notice. This process ensures that only a new GRB event will trigger a telescope slew movement.

Algorithm for the Visual C++ program

For using Windows socket functions, a *winsock.h* header file is required. *Windows.h* header file at the beginning of the program calls this file. The program starts with declaring a SOCKET as an unsigned integer. The **socket()** function that follows creates a socket of the type specified (PF_INET, SOCK_STREAM, IPPROTO_TCP) and returns a handle to that socket. PF_INET decides the Internet protocol family, SOCK_STREAM decides a stream socket and the last parameter informs the socket function that the TCP/IP protocol is to be used for communication.

After calling the **socket()** function, the members of the structure **sockaddr_in** are initialized by AF_INET family specifying that the socket will be used over the Internet, the address of the IMAP4 server will be 202.41.133.33 and the port for communicating with it will be 143.

This declaration is followed by a **connect()** function which connects the client machine to the server at the other end of the network. The server sends a message of greeting which the client needs to **receive()** in its buffer [4,5,7]. At this stage, the client issues a fixed sequence of commands for logging in, selecting inbox (optional), fetching e-mail text and finally logging out from the server. For any command from the client, if the **IMAP4** server does not respond within 6 seconds (decided by **timeout.tv_sec** component in the timeout timeval structure of the Visual C++ program), the program is terminated after displaying an error message to that effect. This may only happen when the network is handling a heavy traffic causing a delay in transmission and reception of data packets, which is normally not expected for the GRB observations that are carried out during night. The e-mail text (or the response from the server to a fetch command) is stored in a text file on the specified folder on the hard disk of the client machine. This cycle repeats at one-minute interval till the program is manually terminated after the observing session is over. Thus, an E-mail Notice of a fresh GRB event occurring anytime during the night will be stored on the hard disk within a maximum of 60 seconds. The client can choose to send this sequence of commands at a shorter interval than 60 seconds if he decides to do so. This, however, will increase the traffic on the network on one hand and may make the response from the IMAP4 server slower, which may eventually result in termination of the program due to a timeout error.

The **Appendix C** of this write-up shows the complete Visual C++ source listing of the program for accessing the E-mail message every minute and storing it on the hard disk in a specific folder.

4.4 Visual Basic Program

Using the Microsoft Visual Basic 6.0 and its Integrated Development Environment (IDE), we have developed a Visual Basic (VB) project for controlling the telescope movement automatically and making an audio announcement of a GRB event for the observer. To start with, to the main **Form** of the VB project during its development, a few controls were added from the IDE ToolBox and are listed below:

- **Label controls**: Label1 and Label2
- **TextBox controls**: Text1 and Text2
- **Timer control**: Timer1 and Timer2
- **Microsoft Communication controls**: MSComm1 and MSComm2.
- **Microsoft Multimedia Control**: MMControl1

From the Properties window of the **Timer** control, the Interval property was set at 20 seconds for **Timer1** and around 10 seconds for **Timer2**. Both the communication controls (**MSComm1** and **MSComm2**) were set at 9600-baud rate, no parity bit, 8 data bits and 1 stop bit. The **TextBox1** was set for Multiline Property and Vertical and Horizontal Scroll Bars [8]. **MMControl1** device type was selected as Wave Audio and wave file containing an audio announcement for a GRB event was attached to it. Code lines were added to the procedures for the various components selected to achieve the required operation explained in the algorithm below.

The Algorithm for the Visual Basic 6.0 program

The VB program essentially controls the telescope that slews to the GRB position in the sky as per the RA and Dec coordinates notified in the GCN e-mail notice from a stored file. As explained before, this file is updated in the background at every one-minute interval by the VC++ program running concurrently on the same Pentium PC machine.

At the interval of 20 seconds (decided by **Timer1**), the VB program accesses a disk file containing the fresh e-mail message as stated above. If an event has occurred, the program then checks for the required string patterns in the text message and extracts the RA and Dec coordinates of the GRB. The two labels, **Label1** and **Label2** respectively display these in the main **Form**, and the e-mail text is displayed in the **TextBox1** for convenience of an operator who may want to monitor it occasionally. (See p.15). At this point, the **Timer1** is disabled and the **Timer2** is enabled.

The RA and Dec coordinates of the GRB are converted into the command strings in the necessary format in ASCII characters to be sent on the PC serial port **COM1** through **MSComm1** control for moving the telescope. The **LX200** telescope checks the validity of these coordinates and if the GRB is found to be located above the local horizon at the time, the telescope slews to the GRB location and starts tracking it automatically till the operation is manually terminated. On the other hand, if the GRB is found to be below the local horizon at the time, the telescope responds with a line of message to that effect without moving the telescope. This message is received by the **MSComm1** control on the serial input and is displayed in the **TextBox2**. If no such message is received, it implies that the GRB was above the local horizon and the telescope had started tracking it. Thus, the robotic telescope starts tracking the GRB within a maximum of 80 seconds from the receipt of a GRB E-mail Notice. This figure is estimated as follows. In the worst case, the fresh e-mail might have arrived just after the VC++ program finished its command sequence with a logout command, requiring another 60 seconds to go before starting the next sequence. Subsequently, by the time the e-mail message was stored, the 20 second interval in **Timer1** in the VB program had just finished requiring another 20 seconds to go before it could read the e-mail thus causing a total delay of 80 seconds to command the telescope. In addition, the operator may start the VC++ and VB programs manually and randomly at the start of the observation session and hence both of them may not be running in exact synchronization. This also will have some effect on the delay in moving the telescope after receiving the GCN e-mail.

The **Timer1** can be set for any interval that is lesser than 20 sec to improve the response time of the telescope to an e-mail. We chose this interval to avoid any clash between the GRB observation programme and a routine astronomy program running in the background. With this setting, the telescope is not commanded to move by the e-mail before it completes the motion previously commanded by the routine astronomy programme.

In addition to giving a command to the telescope, simultaneously, with the help of **Windows Multimedia** support available on the **Pentium PC**, a pre-recorded audio announcement for a fresh GRB event is played repeatedly on the speakers connected to the **PC** via a standard sound card. The second timer, **Timer2,** decides the period of this repetition. This informs the observatory assistant present on the telescope floor of a fresh GRB event to enable him to start the **CCD** camera attached to the **LX200**. (As mentioned earlier, the **CCD** camera can be commanded in the evening, at the start of the observation session, to take images of the field of view of the sky seen by the robotic telescope at a regular interval, irrespective of the direction in which it is pointing. In that case, the observatory assistant need not take any specific action after hearing the audio announcement and the system will be operated in a fully automatic mode). As an additional precaution, simultaneously, an external telephone data/voice modem (**D-Link DFM-560ES**) connected to the second serial port **COM2** of the **Pentium PC** rings a selected telephone extension in the office/residence of the Observer. When the observer lifts the telephone handset, he hears through his telephone receiver the same audio announcement of a GRB event repeatedly till he disconnects the call. The software does it automatically by using the speakerphone commands of the modem and by connecting the sound card output to the microphone jack provided on the modem. This alerts the observing scientist who may initiate any further action (like tracking the GRB event with the main 1.2-m IR telescope, suspending temporarily his ongoing science programme etc).

After reading the GCN e-mail, there will not be any read operation on the disk file due to the termination of the 20 second timer (**Timer1**) and the telescope will not be commanded again. This is perfectly in order as the probability of another GRB detection in the same night from a different source is statistically very small.

The **Appendix D** of this note shows the source listing of the Visual Basic program for controlling the robotic telescope after extracting the RA and Dec coordinates of a GRB from the GCN E-mail message freshly recorded on the hard disk file.

4.5 A Robotic Telescope

A new breed of Robotic Telescopes has entered the Amateur Astronomy market. These telescopes can be remotely controlled by a Personal Computer. This long-standing demand from the community of astronomers forced the manufacturers to bring many affordable models in the market to suit to the individual users. Meade **LX200** from Meade Instruments Corporation, USA, is one such telescope and has been chosen for this project.

The **8" Model of Meade LX200** includes:

- 8" Schmidt-Cassegrain optical tube assembly (D = 203mm, F = 1280mm-f/6.3);
- heavy duty fork mount with 4"- dia. Sealed polar ball bearing,
- quartz-microprocessor-controlled 5.75" worm gears on both axes,
- multi-function power panel display on the drive base,
- manual and electrical slow motion controls on both axes,
- setting circles on RA and Dec,
- handheld keypad Electronic Command Center with digital readout display ,
- PPEC Smart Drive,
- 9-speed drive control on both axes,
- GO TO controller,
- High-Precision pointing,
- 64,340-object onboard celestial software library,
- 25 ft power cord for 230V 50 Hz operation and
- a variable height field tripod.

The **LX200**'s internal software supports the RS-232 serial interface requiring a serial communication program. Such programs are readily available in the market. (As described above, we however, developed our own program in Microsoft Visual Basic 6.0 due to our very specific requirement of a totally unattended operation).

The **LX200** Command Set consists of 17 commands for general telescope information, 7 commands for telescope motion, 4 commands for home position, 29 commands for library objects and 11 miscellaneous commands for focus and reticle control etc. Each of these commands has a specific format of ASCII characters and returns a response in ASCII characters also in a specific format of its own informing about the success or failure of a command given to the telescope.

The RS232 interface communication with the telescope in either direction is set at 9600-Baud Rate, No Parity, 8 date bits and 1 stop bit.

The **LX200** telescope can be used in various modes. For visual observing and planetary photography operations, it can be set up in the altazimuth mode. We have, however, set up the telescope in the equatorial mode by fixing an equatorial wedge onto the field tripod. The wedge was adjusted for the observatory's latitude after the field tripod was properly leveled. The telescope has built in software that helps the observer in making a precise polar alignment, in a few trials, to achieve a good pointing accuracy. These settings remain in order unless the telescope is installed at a new location. The information on the longitude and latitude of the observatory site as well as that of the time and date once entered in the telescope is stored in it and is remembered and updated even when the electrical power to the telescope is shut off.

Once slewed to an astronomical object, the **telescope starts tracking it by a default action** (to keep the object precisely at the secondary focus where a CCD or any other detector is mounted). The Smart Drive software in the telescope can be trained to eliminate all the periodic errors from the RA worm gear to maintain accurate tracking of the selected object for hours together. A CCD camera or a photometer can be attached to the telescope as a back-end instrument for any science requirement. We have attached a **Celestron PixCel 255** CCD camera to the telescope for this program for taking images of the GRB and the field around it at a regular interval.

4.6 The CCD Camera

A **Celestron PixCel 255** camera is attached to the **Meade LX200** robotic telescope as a backend instrument. Some of the features of the camera include:

- It is based on the Texas Instrument TC255 CCD with 320 X 240 pixels that are 10 micron square (giving 17'.2 X 12'.8' FOV for f/3.15 system using a telecompressor optics with the camera that gives ~3".2/pixel resolution in the image).
- Double Correlated Sampling Readout with 16 bit A/D for the lowest possible noise
- Convenient and fast parallel interface offers full frame download times under 4 seconds
- Thermoelectric cooling and vibrationless fan
- Telescope port for use as an autoguider
- Advanced CCDSoft software for data acquisition, display and processing
- Track and Accumulate for long duration exposures

The Camera is connected to the Pentium PC through the parallel port and works under **CCDSoft** software from **Software Bisque Inc** under Windows 98 Operating System.

It is planned that the camera will be kept ready throughout the night and will be taking images of the field of view of the sky seen by the robotic telescope at a regular interval. The camera will start taking the pictures of the field containing the freshly detected GRB, once the telescope is commanded to track it automatically by the VB software discussed above. The CCD images will be stored on the hard disk for off-line processing.

4.7 Data/Voice Modem

The **D-Link DFM-560ES Data/Voice/Fax modem** of external type with a speakerphone attachment is connected to the Pentium PC on **COM2** port and has added a tremendous facility to the set up of the GRB observations. As explained in the introduction, the GRB event is a random phenomenon and it is humanly impossible to wait for it to occur for the entire night. With the help of the modem and the speakerphone attachment, it is now possible to give an audio alert to the scientist immediately after receiving the GCN e-mail notice to enable him to take appropriate actions at a very short notice.

The modem accepts Hayes compatible AT commands that can be sent on a serial port under Visual Basic Program.

5. Applications of the software package

The in-house software developed for our GRB programme, thus controls a robotic telescope automatically when triggered by an incoming e-mail message. The utility of this software, however, is not restricted to the GRB programme alone. In fact, by using this software, any telescope can be remotely operated by just sending an e-mail message in a proper format. The need of controlling a telescope located at a remote site from the place of research is felt by many observing scientists. These days, the observing scientists cannot make frequent field trips due to a variety of reasons, the most important one being the loss of valuable research time that one needs to spend in travelling to and from an observatory usually located in high mountains.

In a way, the software package described in this Technical Note, in some respects, is even better than a commercially available **Internet Astronomy Server/Client Suite** sold by **Software Bisque Inc; USA.** In the **Internet Astronomy** software, a remote observer sitting on a client machine can operate the observatory telescope connected to the server. The Server and the Client machines need to remain connected over the Internet (or a LAN or WAN etc). In our software, however, the observer can just send an e-mail in a prescribed format from any PC, even from his home computer. The home computer does not need to have any client software running on it. Any standard Mail Utility Program will do the job.

When the observer wants to move to another stellar source, he just sends another e-mail containing the RA and Dec coordinates of the new source to be observed. Our software even offers password security. The telescope can be operated remotely by an e-mail message only by that observer for whom the telescope time has been allotted on that particular night.

Such a facility, with a few modificatios, can be installed in all major Observatories in the country. The IIA telescope at Hanle, the IUCAA telescope near Pune, The NCRA's GMRT near Pune, the UPSO at Naini Tal are a few example sites that may get benefited by this software development for GRB and other astronomy observations.

Our software package can be used even for non-astronomy applications wherein any networked instrument in the Observatory or a Laboratory can be operated from a remote location by just sending an e-mail message at an appropriate time. The possibilities are really endless!

6. Conclusions

A completely automatic system for a robotic telescope has been developed and tested successfully for tracking Gamma Ray Bursts within about a minute of receiving a GCN e-mail notice about its occurrence.

The playing of the audio announcement in the observatory and sending it over a telephone to an observing scientist automatically are the added utilities for a project of this type wherein a real time alert message is very useful for optimising the observations.

All the hardware and software was tested in the Thaltej Campus of the Laboratory. The set up has been moved to the IR Observatory site at Gurushikhar, Mt. Abu. And is ready for carrying out the actual GRB follow-up observations in the near future.

Acknowledgements

The authors are greatly indebted to Prof. B. G. Anandarao, Chairman, Astronomy & Astrophysics Division, PRL, Ahmedabad, for his continuous encouragement, wholehearted support and valuable suggestions during the testing of the instrument set up.

The authors are thankful to Dr. Kailash C. Sahu of Space Telescope Science Institute (STSC), Baltimore, USA, for suggesting us to undertake this work during his visit to PRL and giving us the necessary information on the agencies involved in the programme. Authors appreciate the initial consultations and motivation provided by Dr. S. N. Pradhan in developing the network program for communicating with the IMAP4 server. Mr. A. D. Bobra, along with M/s G. G. Dholakia, R. Shukla and Kadamb Thakkar of PRL Computer Center helped in debugging the program by downloading various versions of IMAP servers to make the program work successfully.

Mr. D. B. Pancholi, Technician, helped in installing and making the polar alignment of the robotic telescope a number of times whenever the telescope was taken to the terrace of the building for night observations.

We also appreciate the motivation given to us by Prof. G. S. Agarwal, Director, PRL, from time to time, during the execution of the project. Financial support for the project came from Dept. of Space, Govt. of India.

References

1. Scott Barthlemy, http://gcn.gsfc.nasa.gov/motivation.html,

2. Gerald J Fishman & Dieter H Hartmann, Gamma Ray Bursts,
Feature Article :Scientific American, March 1998

3. Vijay Mukhi, IMAP4, Charting the future of E-mail,
A tutorial on http://www.vijaymukhi.com

4. W. Richard Stevens, Unix Network Programming,
Prentice Hall of India Pvt. Ltd, New Delhi, 1994

5. Martin Hall, Windows Sockets, An open Interface for Network Programming under Microsoft Windows, Version 1.1, 20 January, 1993

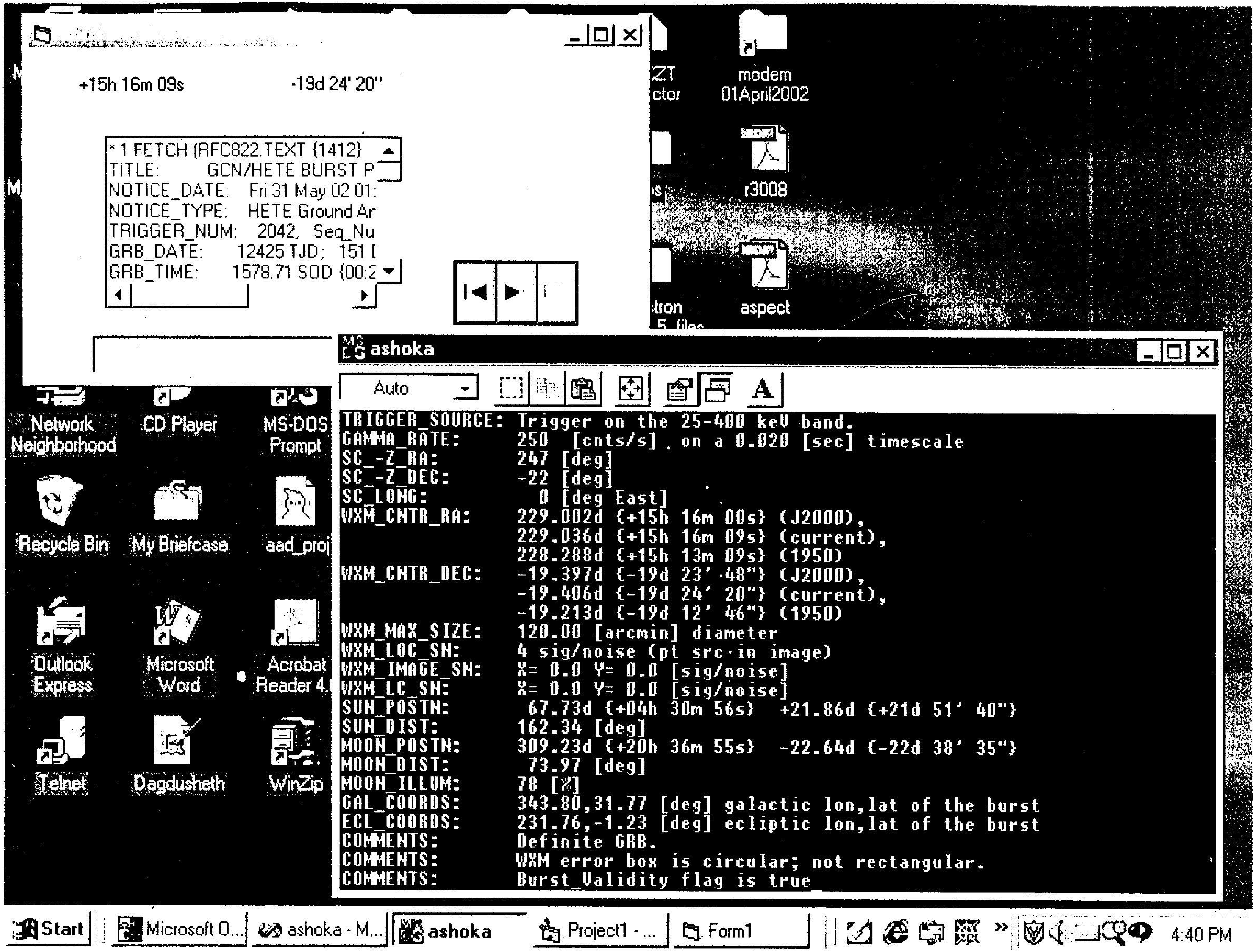
6. CD ROM: Microsoft MSDN Library, Visual Studio 6.0,
Microsoft Corporation, USA.

7. Michael J. Young, Mastering Visual C++ 6,
BPB Publications, New Delhi, 1998

8. Bob Reselman, Richard Peasly and Wayne Pruchniak, Using Visual Basic 6, Prentice-Hall of India Private Limited, New Delhi 110020, December 1999

9. Instruction Manual,
8" LX200 Schmidt-Cassegrain Telescope,
Meade Instruments Corporation, California, USA.

10. User's Manual,
Celestron PixCel 255 Advanced CCD Camera,
Celestron International, California, USA.



```
+15h 16m 09s          -19d 24' 20"

* 1 FETCH (RFC822.TEXT {1412}
TITLE:          GCN/HETE BURST P
NOTICE_DATE:    Fri 31 May 02 01:
NOTICE_TYPE:    HETE Ground Ar
TRIGGER_NUM:    2042,  Seq_Nu
GRB_DATE:       12425 TJD;  151 (
GRB_TIME:       1578.71 SOD (00:2
```

```
TRIGGER_SOURCE: Trigger on the 25-400 keV band.
GAMMA_RATE:     250  [cnts/s] , on a 0.020 [sec] timescale
SC_-Z_RA:       247 [deg]
SC_-Z_DEC:      -22 [deg]
SC_LONG:          0 [deg East]
WXM_CNTR_RA:    229.002d {+15h 16m 00s} (J2000),
                229.036d {+15h 16m 09s} (current),
                228.288d {+15h 13m 09s} (1950)
WXM_CNTR_DEC:   -19.397d {-19d 23' 48"} (J2000),
                -19.406d {-19d 24' 20"} (current),
                -19.213d {-19d 12' 46"} (1950)
WXM_MAX_SIZE:   120.00 [arcmin] diameter
WXM_LOC_SN:     4 sig/noise (pt src in image)
WXM_IMAGE_SN:   X= 0.0 Y= 0.0 [sig/noise]
WXM_LC_SN:      X= 0.0 Y= 0.0 [sig/noise]
SUN_POSTN:       67.73d {+04h 30m 56s}  +21.86d {+21d 51' 40"}
SUN_DIST:       162.34 [deg]
MOON_POSTN:     309.23d {+20h 36m 55s}  -22.64d {-22d 38' 35"}
MOON_DIST:       73.97 [deg]
MOON_ILLUM:     78 [%]
GAL_COORDS:     343.80,31.77 [deg] galactic lon,lat of the burst
ECL_COORDS:     231.76,-1.23 [deg] ecliptic lon,lat of the burst
COMMENTS:       Definite GRB.
COMMENTS:       WXM error box is circular; not rectangular.
COMMENTS:       Burst_Validity flag is true
```

A typical screen display on the Pentium PC CRT monitor after receiving a GCN e-mail notice

APPENDIX A1

A typical GCN E-mail Notice for a dummy event sent for testing the GRB recording equipment

```
TITLE:            BACODINE BURST POSITION NOTICE
NOTICE_DATE:      Thu 24 Jan 02 22:23:25 UT
NOTICE_TYPE:      Original
TRIGGER_NUM:      -1
GRB_RA:           251.98d {+16h 47m 55s} (J2000),
                  252.00d {+16h 48m 00s} (current),
                  251.53d {+16h 46m 06s} (1950)
GRB_DEC:          +35.00d {+35d 00' 13"} (J2000),
                  +35.00d {+35d 00' 00"} (current),
                  +35.09d {+35d 05' 27"} (1950)
GRB_ERROR:        5.0 [deg radius, statistical only]
GRB_INTEN:        1000 [cnts]    Peak=1000 [cnts/sec]
GRB_TIME:         80604.00 SOD {22:23:24.00} UT
GRB_DATE:         12298 TJD;    24 DOY;    02/01/24
GRB_SC_AZ:          0.00 [deg]                          {XScan=0.00}
GRB_SC_EL:          0.00 [deg]  {Zenith_angle=90.00}   {Scan=90.00}
SC_X_RA:            0.00 [deg] (J2000)
SC_X_DEC:           0.00 [deg]
SC_Z_RA:            0.00 [deg]
SC_Z_DEC:           0.00 [deg]
SUN_POSTN:        307.11d {+20h 28m 27s}  -19.07d {-19d 04' 18"}
SUN_DIST:          75.20 [deg]
MOON_POSTN:        70.66d {+04h 42m 39s}  +20.97d {+20d 58' 26"}
MOON_DIST:        124.01 [deg]
PROG_VERSION:     7.39
PROG_LEVEL:       3
COMMENTS:         Test Coordinates.
```

APPENDIX A2

A Typical GCN E-mail Notice for a Gamma Ray Burst detected by HETE2 Spacecraft

```
TITLE:           GCN/HETE BURST POSITION NOTICE
NOTICE_DATE:     Fri 31 May 02 01:54:22 UT
NOTICE_TYPE:     HETE Ground Analysis
TRIGGER_NUM:     2042,   Seq_Num: 3
GRB_DATE:        12425 TJD;   151 DOY;   02/05/31
GRB_TIME:        1578.71 SOD {00:26:18.71} UT
TRIGGER_SOURCE: Trigger on the 25-400 keV band.
GAMMA_RATE:      250  [cnts/s]  on a 0.020 [sec] timescale
SC_-Z_RA:        247 [deg]
SC_-Z_DEC:       -22 [deg]
SC_LONG:           0 [deg East]
WXM_CNTR_RA:     229.002d {+15h 16m 00s} (J2000),
                 229.036d {+15h 16m 09s} (current),
                 228.288d {+15h 13m 09s} (1950)
WXM_CNTR_DEC:    -19.397d {-19d 23' 48"} (J2000),
                 -19.406d {-19d 24' 20"} (current),
                 -19.213d {-19d 12' 46"} (1950)
WXM_MAX_SIZE:    120.00 [arcmin] diameter
WXM_LOC_SN:      4 sig/noise (pt src in image)
WXM_IMAGE_SN:    X= 0.0 Y= 0.0 [sig/noise]
WXM_LC_SN:       X= 0.0 Y= 0.0 [sig/noise]
SUN_POSTN:        67.73d {+04h 30m 56s}  +21.86d {+21d 51' 40"}
SUN_DIST:        162.34 [deg]
MOON_POSTN:      309.23d {+20h 36m 55s}  -22.64d {-22d 38' 35"}
MOON_DIST:        73.97 [deg]
MOON_ILLUM:      78 [%]
GAL_COORDS:      343.80,31.77 [deg] galactic lon,lat of the burst
ECL_COORDS:      231.76,-1.23 [deg] ecliptic lon,lat of the burst
COMMENTS:        Definite GRB.
COMMENTS:        WXM error box is circular; not rectangular.
COMMENTS:        Burst_Validity flag is true.
```

APPENDIX B

VISUAL C++ PROGRAM
START
CREATE SOCKET
CONNECT TO IMAP4 SERVER
START 60 SEC TIMER
TIMER OUT?
NO
YES
SEND LOGIN COMMAND
SEND SELECT INBOX COMMAND (OPTIONAL)
FETCH EMAIL OR IMAP4 RESPONSE SAYING NO EMAIL
SEND LOGOUT COMMAND
STORE FILE ON HARD DISK
VISUAL BASIC PROGRAM
START
START 20 SEC TIMER
TIMER OUT?
NO
YES
READ FILE
GRB EMAIL?
NO
YES
STOP 20 SEC TIMER
MOVE ROBOTIC TELESCOPE TO TRACK GRB
MAKE AUDIO ANNOUNCEMENT
DIAL OBSERVER'S TELEPHONE AND PLAY AUDIO ANNOUNCEMENT IN IT

FLOW DIAGRAM
OF THE SOFTWARE
FOR RAPID FOLLOW UP
OBSERVATIONS OF
GAMMA RAY BURSTS

APPENDIX C

```
//The source listing for the Visual C++ PROGRAM for automatically checking and
//recording every minute a GCN E-mail Notice, if any, from the IMAP4 server at PRL
//Computer Center

#include <windows.h>
#include <iostream.h>
#include <time.h>
#include <stdio.h>
#define NO_FLAGS_SET     0
#define MSG_LENGTH       5000

void main()
{
time_t  ltime;
long    mtime;
char    b[20];
char    RecMsg[MSG_LENGTH];
char    StoreMsg[MSG_LENGTH];
FILE    *fp;
char    filename[40];
strcpy(filename,"C:\My Documents\visc++\ashoka\nmatc.txt");

//Run every minute for 12 hours (720 minutes)
for (int i=0;i<720;i++)
{
time(&ltime);
mtime=ltime+60;
        while(mtime!=ltime)
        {
                time(&ltime);
        }
        itoa(mtime,b,10);
        cout<<b<<"\n";

char    SendMsg_1[28];
char    SendMsg_2[19];
char    SendMsg_3[26];
char    SendMsg_4[13];

//Socket startup sequence
WSADATA    WSAData;

int      iRetVal;
```

```
iRetVal=WSAStartup(MAKEWORD(1,1),&WSAData);
if (iRetVal=0)
{
        cout<<"Startup Not Successful",iRetVal;
}

//Open a TCP socket
SOCKET      sockfd;
sockfd=socket(PF_INET,SOCK_STREAM,IPPROTO_TCP);
if (sockfd==INVALID_SOCKET)
{
        cout<<"\ncan't open stream socket";
}

unsigned      long    temp;
temp=inet_addr("202.41.133.33");

unsigned      short   tport;
tport=htons(143);

//sockaddr_in structure

SOCKADDR_IN      grb;
memset((char *) &grb,0,sizeof(grb));

grb.sin_family=AF_INET;
grb.sin_addr.S_un.S_addr=temp;
grb.sin_port=tport;

//Preparing a set of sockets

FD_SET      readfds;
FD_SET      writefds;
FD_ZERO(&readfds);
FD_ZERO(&writefds);
FD_SET(sockfd,&readfds);
FD_SET(sockfd,&writefds);

TIMEVAL     timeout;
timeout.tv_sec=6;
timeout.tv_usec=0;

//connect socket

if (0!=(connect(sockfd, (SOCKADDR *) &grb, sizeof(grb))))
{
        closesocket(sockfd);
        cout<<"Connect not successful";
}

//Read Socket

int     status_1;
memset((char *) &RecMsg,0,sizeof(RecMsg));

status_1=recv(sockfd,RecMsg,MSG_LENGTH,NO_FLAGS_SET);

if (status_1==SOCKET_ERROR)
        {
                cout<<"Socket Error %d",WSAGetLastError();
        }

cout<<RecMsg;

//First command to PRL server

memset((char *) &RecMsg,0,sizeof(RecMsg));

status_1=2;
status_1=select(0,0,&writefds,0,&timeout):
if(status_1==1)
{
        strcpy(SendMsg_1,"a001 login thaltej grburst\r\n");
        cout<<"sending "<<SendMsg_1;
        send(sockfd,SendMsg_1,sizeof(SendMsg_1),NO_FLAGS_SET);
}
if(status_1==0)
{
        cout<<"timeout first send\n";
}

status_1=2;
status_1=select(0,&readfds,0,0,&timeout);
if(status_1==1)
{
        recv(sockfd,RecMsg,sizeof(RecMsg),NO_FLAGS_SET);
        cout<<RecMsg;
}
if(status_1==0)
{
        cout<<"timeout first receive\n";
}
```

```
//Second command to PRL server

memset((char *) &RecMsg,0,sizeof(RecMsg));

status_1=2;
status_1=select(0,0,&writefds,0,&timeout);
if(status_1==1)
{
        strcpy(SendMsg_2,"a002 select inbox\r\n");
        cout<<"sending "<<SendMsg_2;
        send(sockfd,SendMsg_2,sizeof(SendMsg_2),NO_FLAGS_SET);
}
if(status_1==0)
{
        cout<<"timeout second send\n";
}

status_1=2;
status_1=select(0,&readfds,0,0,&timeout);
if(status_1==1)
{
        recv(sockfd,RecMsg,sizeof(RecMsg),NO_FLAGS_SET);
        cout<<RecMsg;
}
if(status_1==0)
{
        cout<<"timeout second receive\n";
}

//Third command to PRL server

memset((char *) &RecMsg,0,sizeof(RecMsg));

status_1=2;
status_1=select(0,0,&writefds,0,&timeout);
if(status_1==1)
{
        strcpy(SendMsg_3,"a003 fetch 1 rfc822.text\r\n");
        cout<<"sending "<<SendMsg_3;
        send(sockfd,SendMsg_3,sizeof(SendMsg_3),NO_FLAGS_SET);
}
if(status_1==0)
{
        cout<<"timeout third send\n";
}
```

```
status_1=2;
status_1=select(0,&readfds,0,0,&timeout);
if(status_1==1)
{
        recv(sockfd,RecMsg,sizeof(RecMsg),NO_FLAGS_SET);
        cout<<RecMsg;
        strcpy(StoreMsg,RecMsg);
}
if(status_1==0)
{
        cout<<"timeout third receive\n";
}

//Fourth command to PRL server

memset((char *) &RecMsg,0,sizeof(RecMsg));

status_1=2;
status_1=select(0,0,&writefds,0,&timeout);
if(status_1==1)
{
        strcpy(SendMsg_4,"a004 logout\r\n");
        cout<<"sending "<<SendMsg_4;
        send(sockfd,SendMsg_4,sizeof(SendMsg_4),NO_FLAGS_SET);
}
if(status_1==0)
{
        cout<<"timeout fourth send\n";
}

status_1=2;
status_1=select(0,&readfds,0,0,&timeout);
if(status_1==1)
{
        recv(sockfd,RecMsg,sizeof(RecMsg),NO_FLAGS_SET);
        cout<<RecMsg;
}
if(status_1==0)
{
        cout<<"timeout fourth receive\n";
}


//Closing the socket
closesocket(sockfd);
```

```
//Cleanup of the socket application called
WSACleanup();
fp=fopen(filename,"wb");
fprintf(fp,"%s",StoreMsg);
fclose(fp);
}
        return;
}
```

APPENDIX D

Visual Basic Program for GRB Observations.
Project4 in VB

```
'The source listing of a Visual Basic Program for periodically reading the GCN E-mail
'Notice from hard disk file and controlling the robotic telescope and modem

Private Sub Form_Load()
Dim intLen As Integer
Dim intRAloc As Integer
Dim intDECloc As Integer
Dim strFileName As String
Dim strText As String
Dim strBuffer As String
Dim strFind As String
Dim strRA As String
Dim strRAprv As String
Dim strDEC As String
Dim strDECprv As String
Dim strDECcomm As String
Dim lngResult As Long
Dim FileHandle%
Dim strTel As String

'Use COM1 Port for Telescope Control
MSComm1.CommPort = 1
MSComm1.Settings = "9600,N,8,1"
MSComm1.PortOpen = True

'Use COM2 Port for Modem
MSComm2.CommPort = 2
MSComm2.Settings = "9600,N,8,1"
MSComm2.PortOpen = True

'Disable Timer2
Timer2.Enabled = False

'Set Multimedia Parameters
MMControl1.Notify = False
MMControl1.Wait = True
MMControl1.Shareable = False
MMControl1.DeviceType = "WaveAudio"
MMControl1.Command = "Open"
MMControl1.FileName = "C:\WINDOWS\Desktop\grb.wav"
End Sub
```

```
Private Sub MSComm1_OnComm()
Text2.Text = Text2.Text & MSComm1.Input
End Sub

Private Sub Timer1_Timer()
MSComm1.InBufferCount = 0

strTel = Label1.Caption & Label2.Caption
FileHandle% = FreeFile
strFileName = "c:\My Documents\visc++\ashoka\nmatc.txt"
Open strFileName For Input As #FileHandle%

Do While Not EOF(FileHandle%)
Line Input #FileHandle%, strBuffer
strText = strText & strBuffer & vbCrLf
Loop
Close #FileHandle%

'TEST Co-ordinates
strFind = "GRB_RA:"
lngResult = InStr(strText, strFind)

If lngResult <> 0 Then
   intRAloc = lngResult + 72
   strRA = Mid(strText, intRAloc, 12)
   strRAcomm = "#:Sr " & Mid(strRA, 2, 2) & ":" & Mid(strRA, 6, 2) & "." &
Mid(strRA, 10, 1) & "#"
   Label1.Caption = strRA

   strFind = "GRB_DEC:"
   lngResult = InStr(strText, strFind)
   intDECloc = lngResult + 72
   strDEC = Mid(strText, intDECloc, 12)
   strDECcomm = "#:Sd " & Mid(strDEC, 1, 3) & Chr$(223) & Mid(strDEC, 6, 2) & "#"
   Label2.Caption = strDEC

   If strTel <> strRA & strDEC Then
   grbaction
   End If

End If

'HETE BURST
strFind = "WXM_CNTR_RA:"
lngResult = InStr(strText, strFind)
If lngResult <> 0 Then
   intRAloc = lngResult + 76
   strRA = Mid(strText, intRAloc, 12)
   strRAcomm = "#:Sr " & Mid(strRA, 2, 2) & ":" & Mid(strRA, 6, 2) & "." &
Mid(strRA, 10, 1) & "#"
   Label1.Caption = strRA

   strFind = "WXM_CNTR_DEC:"
   lngResult = InStr(strText, strFind)
   intDECloc = lngResult + 76
   strDEC = Mid(strText, intDECloc, 12)
   strDECcomm = "#:Sd " & Mid(strDEC, 1, 3) & Chr$(223) & Mid(strDEC, 6, 2) & "#"
   Label2.Caption = strDEC

   If strTel <> strRA & strDEC Then
   grbaction
   End If

End If
```

'SAX_WFC Burst

```
strFind = "GRB_SAX_RA:"
lngResult = InStr(strText, strFind)

If lngResult <> 0 Then
   intRAloc = lngResult + 78
   strRA = Mid(strText, intRAloc, 12)
   strRAcomm = "#:Sr " & Mid(strRA, 2, 2) & ":" & Mid(strRA, 6, 2) & "." &
Mid(strRA, 10, 1) & "#"
   Label1.Caption = strRA

   strFind = "GRB_SAX_DEC:"
   lngResult = InStr(strText, strFind)
   intDECloc = lngResult + 78
   strDEC = Mid(strText, intDECloc, 12)
   strDECcomm = "#:Sd " & Mid(strDEC, 1, 3) & Chr$(223) & Mid(strDEC, 6, 2) & "#"
   Label2.Caption = strDEC

   If strTel <> strRA & strDEC Then
   grbaction
   End If

End If
```

'IPN Burst

```
strFind = "GRB_IPN_RA:"
lngResult = InStr(strText, strFind)
If lngResult <> 0 Then
```

```
   intRAloc = lngResult + 84
   strRA = Mid(strText, intRAloc, 12)
   strRAcomm = "#:Sr " & Mid(strRA, 2, 2) & ":" & Mid(strRA, 6, 2) & "." &
Mid(strRA, 10, 1) & "#"
   Label1.Caption = strRA

   strFind = "GRB_IPN_DEC:"
   lngResult = InStr(strText, strFind)
   intDECloc = lngResult + 84
   strDEC = Mid(strText, intDECloc, 12)
   strDECcomm = "#:Sd " & Mid(strDEC, 1, 3) & Chr$(223) & Mid(strDEC, 6, 2) & "#"
   Label2.Caption = strDEC

   If strTel <> strRA & strDEC Then
   grbaction
   End If

End If

'RXTE_ASM Burst
strFind = "GRB_RXTE_RA:"
lngResult = InStr(strText, strFind)

If lngResult <> 0 Then
   intRAloc = lngResult + 96
   strRA = Mid(strText, intRAloc, 12)
   strRAcomm = "#:Sr " & Mid(strRA, 2, 2) & ":" & Mid(strRA, 6, 2) & "." &
Mid(strRA, 10, 1) & "#"
   Label1.Caption = strRA

   strFind = "GRB_RXTE_DEC:"
   lngResult = InStr(strText, strFind)
   intDECloc = lngResult + 96
   strDEC = Mid(strText, intDECloc, 12)
   strDECcomm = "#:Sd " & Mid(strDEC, 1, 3) & Chr$(223) & Mid(strDEC, 6, 2) & "#"
   Label2.Caption = strDEC

   If strTel <> strRA & strDEC Then
   grbaction
   End If

End If
Text1.Text = strText
End Sub
```

```
Private Sub Timer2_Timer()
'Play Audio File Repeatedly
MMControl1.Command = "play"
MMControl1.Command = "prev"
End Sub

Public Sub grbaction()
Text2.Text = ""

'Send Commands to Robotic Telescope
MSComm1.Output = strRAcomm
MSComm1.Output = strDECcomm
MSComm1.Output = "#:MS#"

'Disable Timer1 and Enable Timer2
Timer1.Enabled = False
Timer2.Enabled = True

'Send Commands to Modem

'Reset Modem
lngResult = 0
strMyString = ""
MSComm2.Output = "ATZ" & vbCrLf

Do Until lngResult <> 0
strMyString = strMyString & MSComm2.Input
lngResult = InStr(strMyString, "OK")
Loop

'Dial Extension
lngResult = 0
strMyString = ""
MSComm2.Output = "ATDT 2119;" & vbCrLf

Do Until lngResult <> 0
strMyString = strMyString & MSComm2.Input
lngResult = InStr(strMyString, "OK")
Loop

'Set Audio Mode
lngResult = 0
strMyString = ""
MSComm2.Output = "AT + FCLASS = 8" & vbCrLf
Do Until lngResult <> 0
strMyString = strMyString & MSComm2.Input
```

```
lngResult = InStr(strMyString, "OK")
Loop

'Speakerphone on
lngResult = 0
strMyString = ""
MSComm2.Output = "AT + VSP = 1" & vbCrLf

Do Until lngResult <> 0
strMyString = strMyString & MSComm2.Input
lngResult = InStr(strMyString, "OK")
Loop

'Analog source destination selection
lngResult = 0
strMyString = ""
MSComm2.Output = "AT + VLS = 9" & vbCrLf

Do Until lngResult <> 0
strMyString = strMyString & MSComm2.Input
lngResult = InStr(strMyString, "OK")
Loop

'Set Microphone Gain
lngResult = 0
strMyString = ""
MSComm2.Output = "AT + VGM = 220" & vbCrLf

Do Until lngResult <> 0
strMyString = strMyString & MSComm2.Input
lngResult = InStr(strMyString, "OK")
Loop

'Set Speaker Gain
lngResult = 0
strMyString = ""
MSComm2.Output = "AT + VGS = 200" & vbCrLf

Do Until lngResult <> 0
strMyString = strMyString & MSComm2.Input
lngResult = InStr(strMyString, "OK")
Loop

End Sub
```

```
Private Sub Form_Unload(Cancel As Integer)
MSComm1.PortOpen = False
MSComm2.PortOpen = False
MMControl1.Command = "close"
Timer2.Enabled = False
End Sub
```