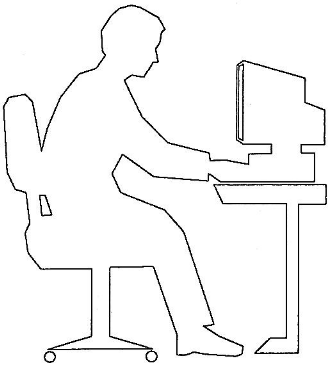
■ Texas Instruments Brisbane User Group (TIBUG) ■



BUG BYTES

April/May 1997

Editor's Note

Dennis Remmer
dennis@dstc.edu.au

Welcome to the second issue for 1997.

This issue I'm going to branch out a little and bring some more non-specific articles which should be of interest, in addition to a few TI-specific pieces.

Use your TI for the Internet? Maybe it isn't such a silly idea - and there's a good deal of discussion going on about how to do it.

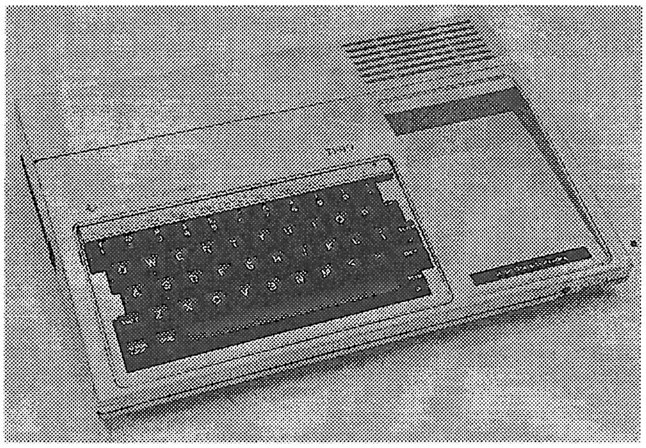
Some more news (?) from the TI company itself to see what they're up to these days.

I've also undertaken to review some of the TI 99 - related Web sites out there. All views expressed are my own.

The next meeting will be held at John Campbell's house, 107 Kylie Ave, Ferny Hills, on Friday night 4th April, starting at 7:30p.m.

A personal note of thanks to Col Christensen for so kindly making available (i.e giving away) his vast collection of TI hardware, software, paperware and memorabilia at the last meeting. A very kind gesture.

Best Regards...



A picture of an all-silver TI-99! Does anybody have any more information on the status of such a machine (i.e was it a prototype casing)?

Source:
http://staff.motiv.co.uk/~kevan/old_collection/manufacturer-ti.html

TI Buys Microsoft :-)

Charles Good
cgood@im3.com

April 1, 1997. In an announcement that stunned most industry observers Texas Instruments announced today that it had purchased Microsoft from Bill Gates. In a news conference today Tex Lubbock, TI President and CEO discussed his plans for the new Microsoft.

"We are developing a new universal computer operating system to replace Microsoft's DOS and Windows 95 products. The new operating system will be called FUNNELWEB 97 and will be so small that it will work on computers with as little as 48K of ram. For years now, software and operating systems for modern PC's have earned the nickname of "fatware" because of ever growing software memory demands. We are reversing this trend by modifying the FUNNELWEB operating system of this company's highly successful 99/4A computer so that the operating system can run on all computers. Now owners of one kind of computer will feel very comfortable operating any other kind of computer because all computers will soon use our single operating system.

"FUNNELWEB 97 will have several technological innovations that will make it far superior to Windows 95. Windows 95 was designed for illiterate users and that is why there were all those little pictures on the Windows 95 screen. FUNNELWEB 97 is designed for a more sophisticated audience, specifically, people who know how to read. FUNNELWEB 97 will introduce to the computing world the fantastic new concept of MENUS CONTAINING WORDS

CONTENTS

instead of screens full of little pictures. These pictures required lots of memory to draw on screen and were one of the reason why old fashioned operating systems such as Windows 95 were such memory hogs. With FUNNELWEB 97 all you have to do is press the number or letter that is displayed in front of a group of words. This either runs the software described by the words or leads you to additional menus with more descriptive words. This revolutionary system based on words instead of pictures will save lots of ram memory which will then be available for the running of software.

"FUNNELWEB 97 will have several other innovative features not found in Windows 95, such as a file viewer and a sector editor. Just move the cursor next to a file name, press "V", and view the file just as it appears encoded on disk. If you don't like what you see you can directly alter the disk with FUNNELWEB 97's sector editor. Neither of these features is found in the now outdated Windows 95 product.

"I want to encourage all members of the public to bring their pentium computers, pentium pro computers, and 99/4A-Geneve computers to the Texas Instruments/ Microsoft table at the May 24, 1997 MUG Conference at Lima Ohio. This is when FUNNELWEB 97 will have its official release and you can try out our new operating system on your computer. Then you can enjoy the thrill of seeing your PC boot up and start running software almost as quickly as the 99/4A can boot up and run software."

On the way to the Internet with the Geneve and TI: Part I. The Internet

Michael Zapf

zapf@diamant.vsb.informatik.uni-frankfurt.de

(Editor: There is an important development project underway by various people on the Internet to investigate implementing certain Internet and network technologies for the TI and/or Geneve. Michael Zapf is one of the main players in this project, and here provides the opening chapter.)

1. Introduction

There are computer networks for quite some decades now. Computer scientists soon realized the advantages of connecting computers to enhance calculations, distribute data, share expensive hardware between different participants and so on. At first there were not many services; you could at most send some messages, but gradually networks became very comfortable. Today networks often consist of a server that is equipped with many resources, e.g. large hard drives and fast processors, and many clients that are computers on their own but use the offered services from the server by loading files from it or dispatching difficult jobs to it. Computers are connected to each other in a network, and networks may also be connected to each other. This forms what is called an internet.

As is often the case, military requirements are the starting point of many inventions. The U.S. Department of Defence instructed scientists to develop an internet on the American territory that is fail-save in case of a destruction of participating nodes. That means that there must not be a center where all the messages have to pass through; instead, the net should be able to re-route the data when connections become inaccessible. This led to the development of the ARPAnet.

Further need of scientific know-how and a increasingly relaxing military situation in the world allowed more and more scientific sites to join the net that was now called the "Internet". The development gained speed at a dramatic rate: Protocols were defined for different services in the Internet; computers that were miles away could be used as if you were working with it directly. And then even the border was crossed, and the Internet started spreading around the world.

For many years the Internet was mainly used for message exchange. Because of its analogy to the real world this exchange was called e-mail (electronic mail). Even today, e-mail is one of the most popular services of the Internet; what could be easier than to type in a few words, execute a send command; and only some minutes later a reply arrives - although your peer sits in an office on the other side of the world. And unlike telephoning, you need not make sure you can actually reach him - your message will be presented to him as soon as he takes a look in his mailbox.

Another interesting institution is the USENET that was soon integrated into the Internet. The USENET consists of a collection of so-called newsgroups; today you will find many thousands of them. Messages sent to a newsgroup are routed to a special computer (also called news server) that stores them and distributes them to other news servers. Users can subscribe to these newsgroups at a news server of their choice and download all messages of the newsgroups they are interested in. This allows a lot of interesting discussions on every possible subject, about the TI-99/4A in comp.sys.ti, for instance.

In order to enhance file transfer on the Internet, the File Transfer Protocol was specified. Now it was possible to store files at well-known locations in the net and retrieve them when necessary. While the development of distributed file access and remote execution was more interesting for subsections of the Internet (subnets), the global data exchange started to grow exponentially with the invention of the World Wide Web (WWW). In fact, it is still based on the Internet and uses a new protocol "on top", the HyperText Transfer Protocol (which you often see as "http" at the left side of WWW addresses). This allows to define files that appear as hypertext (text with links to other files) when they are downloaded to the user's computer. A special language (HyperText Markup Language, HTML) is used to compose these documents. Furthermore, multimedial elements are considered so that images, sounds, video clips and so on can be sent along with the text.

With these features the Internet became interesting for virtually everyone. More and more companies try to advertise their products; newspapers and magazines offer excerpts of their printed products; there is entertainment, knowledge, connectivity, home banking, electronic commerce and much more. It is clear by now that this enhancement becomes a true threat to the effectiveness of the Internet for everyone, including its fathers, the scientists. The more participants start sending around their data, the slower the whole net becomes because the bandwidth (amount of data that can pass a network connection in a specific time) is limited. Even worse, the number of Internet addresses will be exhausted in the next few years so that a new addressing scheme had to be already defined that will replace the current one.

2. The infrastructure

If someone lays a bomb in the building where your favorite BBS is located, what happens? It will take quite some time before you can get online again - in case the BBS is ever restored some day. Not so with the Internet - if a server crashes, it will take only a short time for the adjacent net nodes to realize this and to revise their respective routing decisions. (The pratice shows that this does not always work very reliably - but at least it is possible.)

This implies that participating hosts (network nodes) have many more things to do than simply to receive or send texts. And since there are so many applications that want to utilize the network functionality, the software must be very well designed to be usable in very different situations without constraining future developments.

A good real-world example is the situation where the bosses of two companies want to arrange a meeting. Each one has a secretary who takes the messages from him or passes received messages to him, in this case there is even a clerk that delivers the messages inside the company. The secretary herself is free to choose a transmission medium to her peer at the other company; her boss does not care. She, by herself, does not care about the job the telecommunication service has to perform to transmit the fax that she decided to use. The telecom service, on the other hand, is not interested in the message itself, but only to deliver it as requested. Her peer notices her fax device throwing out a sheet; she takes it, checks it briefly to see if it was correctly transmitted but is not interested in the content. She just looks at the recipient and drops it in the appropriate box. Another clerk comes by, fetches all the papers in this box, and brings them to the boss. The advantage is that everyone just does a small job and is soon ready to continue with any other work. If somebody seems to work unsatisfactorily, he can easily be replaced.

This seemed to be a model for the realization of the global Internet. The most successful strategy proved to be a paradigm that states that the network software should be organized in layers where only layers of the same depth understand each other.

Each layer receives outgoing data from its next upper layer, modifies them and hands them over to the next lower layer. Incoming data is at first processed by a lower layer before it reaches the next one. This restriction that data can only be passed from one layer to the next one generates the impression of a stack that must at first be worked down, then be rebuilt. Therefore, we also use the term "protocol stack". A protocol is a template for the communication between different peers; beside the real data, it includes information for the recipients that have to process the data. In real life, people normally say "hello" to each other before they start a communication the first time; or, if they don't have visual contact, they call each other by name before.

The layers are named by their functionality and do not prescribe a special protocol:

```
Layer 4: Transport Layer
Layer 3: Network Layer
Layer 2: Data link layer
Layer 1: Physical layer
```

The lower the number, the closer to the phone line or network cable the layer can be found. Applications are set on top of this stack and communicate only with layer 4. Each layer adds its own header to the outgoing application data. In detail:

Physical Layer: This layer is concerned with the transmission of the bits, the specification of the electrical values, the hardware (plugs), the transmission rate. It is specific to the kind of connection you are using; for serial transmission, it is the RS-232 specification. Outgoing byte strings from layer 2 are converted into bit strings; incoming data bits are converted to byte strings before they are sent to layer 2.

Data Link Layer: The bytes of layer 1 are grouped in so-called frames of special length; a checksum is calculated that ensures a correct transmission. In case of an Ethernet where several hosts are connected to one wire, the header contains the network card addresses of the sender and receiver. This is of course not necessary with point-to-point protocols such as PPP or SLIP that are used among two hosts that use a serial connection (e.g. a modem). Flag bytes are used to decide whether the incoming data is to be passed up or control data for this layer.

Network Layer: The data we got from layer 2 has been checked, now we need to see if we are the true recipient. This can again be found in the header, and it need not refer to the same host that can be found in the layer 2 header because layer 2 only knows about our local network but nothing about the world outside. If we are the final recipient, the data is passed up. Otherwise it continues its journey, and it is the task of this layer to decide where to forward the data. An example of a protocol is IP, the Internet protocol.

Transport Layer: The network layer can only work with data strings of limited length, also called packets. This means that longer data strings are broken in suitable pieces and then sent to the network layer. In the other direction, the situation is more complicated. Nothing guarantees that the incoming packets are complete and in the right ordering. The transport layer cares for the completeness of the transmission; if we order 10 kilobytes, it will try to deliver them, regardless of the packet size prescribed by the lower levels. An example of a layer 4 protocol is TCP, the Transmission Control Protocol.

You can see that this structure implies a lot of overhead on the transmitted data: Each layer (except layer 1 that will not be of further interest) adds its special header that enables the corresponding layer on the recipient's host to correctly process the data. From the upper layers to the lower layers, the amount of transmitted data increases; in the opposite direction, the amount decreases with every header being stripped away. To give you an example:

In an Ethernet network which uses TCP/IP/IEEE-802.3, we have 18 bytes for data link, 20 bytes for IP, 20 bytes for TCP and then the payload bytes. One frame is normally 1500 bytes long.

The inestimable advantage of this layer strategy is that each layer:
1. can be replaced without influencing the others
2. can rely on a guaranteed service of the adjacent layers

1. means that if we decide to use Ethernet instead of a serial connection, it is only a matter of the data link layer, but the functionality of the network and transport layers still remain the same.

2. The higher layers have no idea what happens to their data in lower layers, nor what the meaning for higher layers might be. Our application (to be found in a layer greater than four) does not have to bother about the fragmentation of the data, checksums, or the correct ordering. It simply expects the transport layer to deliver exactly the amount of data that it wants and to do the right thing to the data it sends to the transport layer. On the other hand, the transport layer does not know what the meaning of the data is.

LANs (Local Area Networks) are often composed in a simpler way, so you might ask why there is such a problem with the Internet. The reason is clear: The inventors of this protocol stack were wise enough not to require a special computer system that can take part. Since a world-wide system is difficult to change, the smaller the components are, the quicker they are replaced. And even if we have completely new transmission media or processor types, the Internet will continue to work.

3. Overview on the TCP/IP Protocol Stack

After the last section you will now be able to figure out what is meant by this term. The TCP/IP stack is the main protocol stack in the Internet; the data link layer is freely selectable, e.g. Ethernet or serial line (PPP, SLIP), the network layer is controlled by the Internet Protocol, the transport layer uses the Transmission Control Protocol. There are of course other important protocols that are, however, only used for special services, but they are nonetheless important and will be described later. I will only describe those protocols that are of major interest for us, and so I will not explain any further Ethernet issues. While I wrote this chapter I noticed that it's becoming longer and longer so that I will at first give an overview how the different protocols work without getting too far into the details.

3.0 Request for Comment: RFC

A strange name for a set of specifications, isn't it? Since the beginning of the Internet (in those days still the ARPAnet), the documentations for the various protocols and utilities, proposals (even jokes on April Fool's Day) were collected under this label at special Internet sites. Everyone who wants to find precise informations about a special subject must take a look in this list that comprises more than 2000 entries at the beginning of 1997. Many of these RFCs are updates to former ones which are obsolete. There are many ways how to get a copy of an RFC:

Using FTP: The RFCs can be found in the "InterNIC Directory and Database Services" server at 'ds.internic.net'. Change to the directory 'rfc' and you will find the RFCs as text or postscript files. (Note to German users: You can use the server at 'nic2.nic.de')

Using e-mail: It is possible to 'order' an RFC by e-mail. Just send the following message (nothing more; no subject) to 'mailserv@ds.internic.net':

document-by-name rfcXXXX

and replace XXXX by the corresponding RFC number. You can request more than one RFC by using 'document-by-name rfcXXXX, rfcYYYY' or separate lines. You should also get the index by typing

document-by-name rfc-index

I will provide the latest RFC number with each following subsection. I strongly encourage you to get the corresponding RFC because the informations that you can read below cannot cover all necessary aspects.

3.1 Point-to-Point Protocol (PPP) --- RFC 1661

I'll start with the Point-to-Point Protocol because it's more widely used for modem connection than SLIP (serial line Internet Protocol, RFC 1055).

Receiving (data flow from lower to higher numbered layers):

Suppose our interface card has received a stream of bytes (in fact, out layer 1 program has received them) and sends it to the data link layer which we want to use PPP. The tasks for PPP are

- group the bytes to maximum length strings (frames, approx. 1500 bytes)
- check the consistency of the transmitted data by checking the CRC value
- demultiplex the data for the different upper-layer services
- negotiate transmission options with the other end

Of every frame, the first five bytes are kept by PPP as well as the last three ones. The remaining bytes are passed on to the service that is determined by the fourth and fifth byte.

Sending (data flow from higher to lower numbered layers):

In the opposite direction, some upper layer passed data to this layer. What must be done is

- calculate the CRC value
- write the frame header before the data and the CRC and end byte after them
- put the whole frame to the interface driver as soon as possible.

As the data link layer driver is fixed to one interface, it is often considered to be the interface driver itself. If there are more interfaces, each one has its separate data link layer driver.

3.2 Internet Protocol (IP) version 4 -- - RFC 791

Receiving:

The data we received comes from layer 2 and is considered to be IP data. What must be done now is at first to check if this host is in fact the recipient. It is possible that our host is to forward the data because it has two connections to different networks (this is called a gateway). If this is the case, the next receiver is determined by using a special directory, called routing table. This table also tells which interface to use in order to reach a destination, so in case of PPP, the data is just put to the appropriate data link layer driver. If our host is indeed the recipient, the IP layer looks more closely at the data. Again there is a header (20 bytes) that describes among other things the

- length of the packet
- packet ID and offset
- type of transmitted data (TCP, ICMP, UDP, ...)
- source and destination IP address

The IP layer strips off the header and sends the remainder to the selected service (according to the type).

Sending:

The data that came from an upper level is cut in suitable pieces (fragments) for the lower layer. As I already said, IP uses a so-called routing table to find out where to send the packets. A typical entry in the table contains the IP address of the destination host A, the IP address of a gateway B, and the interface name I. This tells IP: To send the packets to A, the driver of interface I needs to send it to B which will forward the data.

If our host is a dead end with a PPP connection, there should be only one entry, namely the other side of the PPP connection as the gateway for 'default' delivery, which means any host.

IP addresses are composed of four bytes to define a location in the Internet. The value is written as four decimal numbers, separated by dots: For example, one of the addresses of the InterNIC server that is mentioned above is '198.49.45.10'. The name 'ds.internic.net' is another way of addressing, but it must at first be translated to these numbers by the Domain Name Service (DNS). Only the numbers can be used in IP datagrams. The numbers are specifically structured, but I won't explain that in this overview.

3.3.1 Transmission Control Protocol (TCP) --- RFC 793

The Internet Protocol is said to be connectionless and unreliable. This does not imply that it is badly working; it simply means that IP does neither care of the ordering of the transmitted packets, nor that all packets have arrived. As this is not acceptable for real applications like FTP where we would rather have files without ugly holes and not shuffled up, another protocol is used to guarantee this. TCP adds another 20 bytes as header, and when this one is stripped away, we eventually get our application data. In addition, several programs may require network access simultaneously. Even FTP needs two connections; one for the data, another for control bytes. In order to send the data to the correct application, the concept of 'ports' was introduced. Each application allocates as many ports as it wants, and if they are granted to it, it can start its communication. Since TCP is a bidirectional service, both sender and recipient need to define ports for their communication. The ports are just an operating system construct but no physical devices.

Another important aspect of TCP is flow-control. This is achieved by using the 'sliding window' strategy: The receiver continously informs the sender about the size of its 'window'. If the receiver does not manage to process the data fast enough, the window 'closes', and when it is shut, the sender cannot proceed with the transmission. (Only data classified as 'urgent' can still be sent.) While the receiver processes the data, it lets the window slide open again.

After the connection establishment took place where the participants exchanged their respective sequence numbers, each one is free to send data to the other side. An explicit termination procedure is required to close the connection.

Receiving:

The sequence number of the data from the IP layer tells TCP where to put the segment it has received into the buffer. If no segment is actually missing up to now, TCP sends an acknowledgement to the other side. Unless the application gets the data from the buffer, TCP decreases the window size. If there is a hole in the buffer, TCP continues to send acknowledgements for the end of the contiguous block from the start of the buffer. The sender, not receiving any acks for the latest segments, tries to retransmit the segment that seems to be missing. When the hole is closed, an ack of the whole block can be sent (the last segments of which could have been there quite long until the missing segments were filled in).

This situation is more often encountered than you would possibly imagine. Especially when the connection is poor, the lower layers could have discarded some data so that some segments could not be reconstructed. The effect of IP's silently discarding packets with checksum errors is that the corresponding segment is not acknowledged. So the connectionless and unreliable character of IP is effectively worked around by the TCP protocol.

Sending:

During the connection establishment a maximum segment size is negotiated between the two participants. The data from the application is split into segments of this size, and the TCP header is written before it. If the last window size of the receiver is larger than the segment size, the segment is sent to the IP layer.

3.3.2 User Datagram Protocol (UDP) --- RFC 768

There are situations when the full-blown TCP machinery is not necessary; for example, when small packets shall be sent, when we are not interested that every packet does reach the destination or when the flow control is performed by the application itself. UDP is a very simple protocol (the RFC is only three pages long; TCP's is 85 pages). The IP packets (datagrams) are equipped with the already described source and destination ports; there is no connection establishment and no automated acknowledgement. If this is desired, it is up to the application to implement it.

Although UDP seems to be of rare use, it is needed by the Domain Name Service (DNS) that translates textual Internet references like ‘ds.internic.net’ to four-byte IP addresses.

3.3.3 Internet Control Message Protocol (ICMP) --- RFC 792 (v4)

This is yet another important protocol that must be present in every TCP/IP implementation. By ICMP, hosts are transmitting messages that are of major importance for the current or future connections.

ICMP messages are normally 4 bytes, followed by message-specific content bytes. The types of messages can be

- echo request/reply
- destination unreachable
- time exceeded (measured by hop count)
- redirect
- source quench
- router solicitation/advertisement
- parameter error
- timestamp request/reply
- information request/reply
- address mask request/reply

The first three ones are used very often. The echo request from any host on the net must be answered by an echo reply; some nets consider hosts that do not reply as crashed and cut dial-up connections.

4. The FTP application --- RFC 959

Now I want to show you one very important application that works on the TCP/IP stack: the File Transfer Protocol application. Although the usage of HTTP is growing, FTP is still the major protocol for uploading and downloading files on the Internet.

Beside the file transfer capabilities inside a subnet where users log on an FTP server by a password, there is another possibility for everyone to download files, called ‘anonymous FTP’. If the system maintainer allows this kind of access, any user can log on by identifying himself as user ‘anonymous’ or ‘ftp’ and typing in his e-mail address as password. After that, a special part of the directory tree of the FTP host is available for browsing and downloading. Sometimes there is also a special subdirectory (usually called ‘incoming’) that is writable so that files can be uploaded; the maintainer should sort the files into appropriate directories.

4.1 A sample session with our FTP host

At first we should watch an example of an FTP session by logging on the FTP server in my subnet. Whenever there is a <Return> it means that the user has to hit the Return key to continue after typing the text before.

```
(some_prompt) ftp www.vsb.cs.uni-frankfurt.de
<Return>

Connected to diamant-atm.vsb.cs.uni-
frankfurt.de.

220 www.vsb.cs.uni-frankfurt.de FTP server
(Version 1.2.3 Fri Jan 10 12:02:30 MET 1997)
ready.

Name (www.vsb.cs.uni-frankfurt.de:anyone):
anonymous <Return>

Guest login ok, send your complete e-mail
address as password.

Password: anyone@some.where.out.there <not
shown on your display, Return>

230 Guest login ok, access restrictions apply.

ftp> cd pub/people/mz <Return>

250 CWD command successful.

ftp> binary <Return>

200 Type set to I.

ftp> get fract20.xmo

200 PORT command successful.

150 Opening BINARY mode data connection for
fract20.xmo (61739 bytes).

226 Transfer complete.

local: fract20.xmo remote: fract20.xmo

61739 bytes received in 61.7 seconds (1
Kbytes/s)

ftp> bye <Return>

221 Goodbye.

(some_prompt) _
```

What we did was to download the file fract20.xmo from the FTP server at ‘vsb.cs.uni-frankfurt.de’ (it is the same as the WWW server, hence the name). As you see - no sign of segmentation, fragmentation, dropped frames or the like. It seems as if we were doing just an ordinary file copy or a familiar BBS file download, not involving all that fuss about this TCP/IP stack. But be sure, it was involved ever since the first <Return>.

4.2 A closer look at the FTP session

After entering the ‘cd’ command that changes the directory to the one with the file and using ‘binary’ that tells the computer that the files to be transferred should not undergo any conversions, we are ready to download the file ‘fract20.xmo’ by using ‘get’.

We suppose that everything is set up correctly and that the FTP server has just received the FTP command “RETR fract20.xmo” (this is the actual command that is transmitted as that very string) and take a look at the actions of the different layers.

FTP application (141.2.150.16): We have just received the command to send the contents of the file “fract20.xmo”. The client should have said on which port it expects the data to arrive, so that we try to send the data to the client “socket” (IP address and port). As we do not (have to) care what the transfer details are, e.g. packet size, we just write the file to its ftp port. As this would be much faster than the network can transport the data, the write operation is blocking which serves as a brake. When there are no more data, the connection is closed.

TCP (141.2.150.16): The application (whatever it is) continues to feed data into the layer. If the client told us that it cannot receive more data (closed window), we don’t accept more bytes from the application (and let it block). If the client announced that it is able to receive data (open window), the data are at first partitioned in segments of a maximum length (as was negotiated at the start of the connection). Now that we know the receiver’s IP address (e.g. 141.2.28.160) and port (e.g. 1048) we construct a header with these informations, segment sequence number and checksum for each segment and just drop these segments to the network layer.

IP (141.2.150.16): The segments that arrive from the upper layer get another header before each one that contains our IP address (141.2.150.16) and the one of the recipient (141.2.28.160). But where’s this 141.2.28.160? No idea, but our routing table says: Send it to 141.2.29.2, it probably knows more about it. And this one is reachable via interface en2.

Ethernet (141.2.150.16): The protocol layer above us sent us some data for 141.2.29.2, we'll at first find out the identification of this Ethernet interface in this network (called "address resolution"). Now that we know it, we encapsulate the data once more and send it to this Ethernet address.

Ethernet (141.2.29.2): There are some data for us. We check the frames and pass them up.

IP (141.2.29.2): Ah, there are some packets. But - it's not for us, the recipient is 141.2.28.160. Taking a look in our routing table, we find that this one is connected to the PPP connection ppp160. So down again with the packet.

PPP (141.2.29.2): The packets we receive must be destined for the one on the other side of my connection. So we encapsulate them again and put them on their way.

PPP (141.2.28.160): There are some data for us. Up to the IP layer.

IP (141.2.28.160): Check the recipient; OK, it's us, no more forwarding. What protocol must be used? The field in the header says it's TCP, so we strip off the header and pass it on to the TCP layer.

TCP (141.2.28.160): There are segments coming up from the network layer that are obviously destined for an application above us. We take the segment number and put it in our buffer at the appropriate place. If it was the segment we expected, we'll send an ACK (acknowledge) of the segment to the server. If not, we just don't acknowledge this incoming segment and rely on the server eventually retransmitting missing segments. The application continues to read the data from our buffer (connected to the indicated port) so that our window is opening again. The other side must be informed of this.

FTP application (141.2.28.160): After we sent the command, the data that arrives at the port (which number was transmitted to the FTP server before) is simply stored in a file that normally has the same name as the remote file. The data arrives asynchronously which means that we need to use a blocking read so that no data are lost. When the connection is closed by the server, there are no more data, and the transfer is complete.

As you can see, each layer "speaks the same language" as its peer on the other side. And - as is often the case - there is another computer in between (141.2.29.2) that forwards the IP packets.

This concludes the first part of my Internet tutorial. The second part will examine implementation issues, especially what is needed to implement a minimal FTP client.

MICROREVIEWS

Speed Reading, P-GRAM Reloader, Assembly Poker, AMS Video Titler, Casino (Klondike) Solitaire

By CHARLES GOOD

SPEED READING by Bruce Harrison

This excellent assembly language software is not only good for improving your reading speed, it is also useful for proofreading articles you have written on your TI and it can be used as a text file reader. It will display on-screen text files of unlimited size.

When you boot the program either as XB DSK1.LOAD or from EA5 you see a title screen and then are asked for the path of a DV80 file. This prompt will support long path names needed by hard drive users. Next you are asked for the reading speed, between 115 and 1,028 words per minute. Next you are asked for the window size. The answer, a number between 3 and 20, determines the number of text lines displayed simultaneously on the monitor. You are given the option of a 20-, 30- or 40-column display. The lettering in all these displays is the same size no matter what text width you specify, corresponding to what you normally see in the TI's 40-column text mode. Finally you are asked for a "bookmark." This is the record number of the DV80 file where you wish viewing to begin and is useful if you viewed part of the text file in an earlier Speed Reading session and wish to continue where you left off. You can ignore the bookmark prompt by just pressing <enter>. After answering all these prompts you press <enter> to begin speed reading.

The software then reads in a big chunk of text from the designated text file and begins displaying it on screen. You see the number of screen lines designated in your "window" entry as wide as your "column" size entry. The text has word wrap. After a short interval, giving you time to read the entire initial text window, the text starts scrolling up the screen with new material appearing at the bottom of your defined window and old text disappearing out the top. The speed of scrolling is determined by your "words per minute" setting. You can change this speed on the fly, while viewing scrolling text, by pressing the F(aster) or S(lower) key. It is really fun to watch the effect of such an on-the-fly speed change. Your current words per minute speed is indicated at the bottom of the screen. You can pause the scrolling at any time with the space bar. During a pause is a good time to change text scrolling speed. Press <space> again to resume scrolling.

You can exit in the middle of a file by pressing "Q". The program then displays your "bookmark," a number one greater than the last record of the file that you read. When you start Speed Reader again you can input this bookmark number and start reading your text file exactly where you left off.

If you can't remember the instructions or program features, just load the on disk "TESTFILE" into the program and use it to practice your speed reading. TESTFILE is actually the program doc file. Bruce provides an external program to print these docs if you want a hard copy.

Speed Reading is one of the only 99/4A programs that can display very large DV80 files. It sucks a big chunk of the file into memory, displays the file, sucks more into memory, etc., indefinitely. Speed Reader's text buffer is larger than that of any 40-column version of TI writer. Bruce Harrison says he developed Speed Reader in part so he could display on screen the large (more than 200 sectors) Sherlock Holmes book files I have made available to the TI community.

Bruce has decided to get rich quick with this software. Unlike a lot of his offerings, it is commercial rather than public domain. I can't send it to you. To get your copy on a SSSD disk you need to send $5 directly to Bruce. This includes the cost of shipping and media.

P-GRAM RELOADER v1.0
by Tony Knerr

This will reload the entire contents of your P-GRAM+ card all in one smooth continuous operation. All four GRAM banks and the DSR are loaded in this process. If you normally keep the same software in your P-GRAM then you will like P-GRAM Reloader. You can experiment with your P-GRAM, temporarily loading in some different software. Later you can restore your original P-GRAM software suite easily with P-GRAM Reloader. This is particularly useful if you want to experiment with a module that uses module RAM at >6000. Doing this will erase the version of extended basic that almost everyone keeps in bank one of their P-GRAM+. You can then quickly use P-GRAM Reloader to restore your P-GRAM's normal extended basic. P-GRAM Reloader is also useful if you think your P-GRAM's memory has been corrupted.

P-GRAM Reloader will work from floppy, RAMdisk, or HFDC-controlled hard drive. You have to put all your P-GRAM files on the same floppy, RAMdisk or hard drive directory. Working from floppy probably requires a DSDD-formatted disk to hold all the P-GRAM files. This means you need a double density (not an official TI) disk controller. If you have a TI controller you can keep all your P-GRAM files on a RAMdisk.

Setting up your P-GRAM files for use by Reloader is a bit tricky, but you will only have to do this once. You copy the P-GRAM DSR and all your GRAM files to the working disk or directory. You then rename most of the GRAM files, following the instructions in Reloader's documentation. Finally, you have to use a sector editor to change the header of one of the GRAM files.

P-GRAM Reloader runs from Editor/Assembler Option 3 or TI BASIC. TI BASIC use requires a CorComp or Myarc floppy disk controller. Just run the software and your P-GRAM will be reloaded, overwriting everything already in the P-GRAM's memory.

This is software that every P-GRAM+ owner should have. It is not meant for the regular, non plus, P-GRAMs that only have one bank of GRAM memory and it is not for GRAM Krackers, Gramulators, or other GRAM devices. P-GRAM Reloader is public domain. Send me $1 and I will mail it to you on a SSSD disk.

ASSEMBLY POKER
by Marcel's Software

Marcel Barbeau is Bruce Harrison's son. Sales of this game augment Marcel's allowance. The software was actually written by Bruce Harrison. It boots from Extended BASIC or EA5.

This is one of the best poker games I have seen on the TI. There is some skill and mystery involved in this game for one player against the computer. You don't know what your computer opponent's hand looks like until the hand of poker is over, and if you fold you never find out.

The rules of five-card draw poker game play are literally "according to Hoyle." This is a one-player game. You play against the computer, whose cards you don't see until the end of the hand. Your cards are displayed at the bottom of the screen in sorted order with a printed statement telling you about your hand ("ace high," or "pair of 10s").

You and the computer each start the game with a $200 pool of money. Each hand starts with you and the computer putting a $4 ante into the pot. You each get the opportunity to bet and "see" one another's bets. At this stage, the computer player may bluff on a garbage hand, since this hand can be improved later by drawing replacement cards.

When replacement cards are offered, if the computer player takes fewer than three cards this means that it has a good hand. If it takes no replacements, this means it has at least a straight. The number of replacement cards taken by the computer player is really the only clue the human player has concerning the contents of the computer's hand. After replacement cards an additional round of betting occurs. This time the computer player will not bet highly on a bad hand and may fold.

The human and computer player start a session with $200 each. The game ends when either player has less than the $4 ante for a new hand. However, you can end the game early by having the human player answer "N" to the "Another hand?" prompt. Thus, Assembly Poker makes an excellent short TI game. If you have a few minutes to kill just slip the SSSD disk into DSK1. The software boots quickly and you can then play a few quick hands of poker.

The game is fun with realistic play action, and the graphics are well done. There is no music or speech, but good poker players like it quiet so they can concentrate on their game. Assembly Poker is commercial. To get it send $5 to Marcel Barbeau. This includes the cost of shipping and handling.

AMS VIDEO TITLER
by Bruce Harrison

I have previously reviewed Video Titler, software designed to create neat videotape title sequences by artfully changing between in-memory full screen graphics. The original let you simultaneously load two full screen graphics into memory. This version lets you load in up to 41 screens, storing them in the AMS card's memory.

You can load TI-Artist "_P" pictures with or without their "_C" color file and/or you can load color pictures created with Harrison's Drawing Program. You load these one at a time, specifying the path and file name for each when prompted. Without an AMS card you can fit only two of these graphics into memory, which can make using Titler quite cumbersome. With the common 256K AMS card you can load 20 graphics. The 128K AMS card will let you load 9 graphics and if you have upgraded to a 512K AMS card you can load up a whopping 41 frames. Once this is done you just press a key to zip from one graphic to the next in sequence. At any time you can pause the VCR and load more pictures, replacing your choice of those already in memory or filling as yet unused memory. Such changing pictures in memory in the middle of a recording session should not be necessary with the AMS version of Titler. It is almost always required if you use the non-AMS version limited to only two in memory graphics.

There are a zillion different neat ways of wiping the screen from one graphic to the next; slow or fast, top to bottom or bottom to top, corner to corner, edges to center or center to edges, etc. Each of these methods for paging through the graphics in memory is controlled by a single keypress.

If you don't want to make videotape titles you can just use AMS Titler to page through libraries of TI-Artist pictures. This would result in a really neat business or user group video presentation.

The software has an XB loader but is really an EA5 program that resides in low memory. It uses high memory to store graphics. The AMS card can be set up to bank 4K blocks of high memory. Each graphic uses three of these 4K memory banks. Video Titler displays in bit map mode, dumping the contents of the desired graphic into VDP RAM.

AMS card owners have been asking for more useful AMS-capable software. Here it is! Bruce Harrison tells me that anybody with enough money to buy an AMS card should be willing to pay a little for AMS capable software. AMS Video Titler is not public domain. You can buy your copy from Bruce on a DSSD disk for $10. This includes shipping, handling, seven samples of on disk graphics to load up, and over-the-phone product support.

CASINO (KLONDIKE) SOLITAIRE
by Ken Gilliland and Notung Software

This can be played from extended basic all by itself right off the disk, or it can be added as a module to Notung's TI Casino suite. The game, labeled Klondike Solitaire when played as part of TI Casino, differs little from your basic one-deck one-player generic solitaire.

Klondike Solitaire is almost identical to the solitaire that comes bundled with Windows 3.1 played on an IBM-compatible, except that in Klondike Solitaire there is some wagering. My wife sometimes spends hours playing the Windows version of solitaire on her IBM. It is one of the ways she relaxes after a hard day at work and dealing with the spouse and kids. Solitaire is a very enjoyable, slow, easy way to kill some time. There is some skill involved and a single game doesn't take very long. You can play just one quick game or spend several hours trying to build up your winnings.

You start off by buying a deck of cards. There are cheap, medium and expensive decks. The more you pay for your deck, the greater are your potential winnings. If you are playing the standalone game, without the TI Casino interface, you are given your choice of either $500, $1,000, or $5,000 at the beginning of the game. When the game ends because you run out of money or because you get tired of playing, then you go back to Extended BASIC command mode and all your money disappears. If you are using the TI Casino interface you can save your winnings in an account for use in a later game, or you can literally write yourself a check for any surplus funds you have accumulated.

The computer starts each game by shuffling the deck and dealing some cards out into card stacks. The remaining undealt cards make up the "deck" from which you may draw one card at a time as needed. The top card in each card stack is exposed. You can arrange cards in these stacks in decreasing numerical order using alternating colors. The object of the game is to build four suit stacks in numerical order starting with ace (on the bottom of each suit stack) and ending with king. These suit stacks are in separate piles, different piles than the card stacks. You may move any exposed card from the card stacks or the most recently exposed card from the deck to any of the four suit stacks if that card is the next in the numerical order suit pile. You are allowed to go through the "deck" a maximum of three times per game. The game ends when you have completed all four suit stacks or when you can't add any more cards to the suit stacks. Your winnings are computed based on the number of cards in your suit stacks, $x per card. The money awarded per card depends on how much you paid for your deck at the beginning of the game. You win some money in almost all of the games you play. The trick is to win back more than you initially paid for the deck. When a game ends you can keep your money and exit Solitaire or shuffle the deck again and play another game.

As in all the TI Casino games, a mouse is required. You move the cursor over a card you want to move and press the fire button. Use the joystick handle to drag the card where you want it and press the fire button to drop the card. The computer won't let you cheat. You can't drop a card on a pile where it isn't supposed to go. You also use the joystick to uncover cards in the "deck" by placing the cursor on the deck and pressing the fire button to display the next card. Joystick use in Klondike Solitaire is almost identical to mouse use in Windows 3.1 solitaire.

If Solitaire is your favorite game, then it is best played directly from the disk without incorporating it into TI Casino. The game boots much faster when played outside of the TI Casino environment.

Configuring Klondike Solitaire into TI Casino, a separate product, also by Ken Gilliland, is a bit tricky. You have to follow closely the instructions in the Solitaire documentation. If you already have the original TI Casino and the extra cost extra features TI Casino Supplement, then adding Klondike Solitaire will almost completely fill a DSSD disk. In fact, you have to remove all the joke files except one in order to fit everything onto a DSSD disk. TI Casino plus the Supplement plus Klondike Solitaire is not hard drive friendly. It is difficult to configure all the files for long path names.

When you add Klondike Solitaire to TI Casino you completely reconfigure the upstairs of the casino, adding rooms for additional casino games. These additional, as yet unreleased, casino games include five-card stud, wheel of fortune (different from roulette), horse racing, single-deck blackjack and paigow poker. When you click on any of these game rooms you are told that the game isn't installed and are returned to the upstairs casino lobby. I look forward to these additional games and will probably buy them all, because the TI Casino suite is my personal favorite 99/4A and Geneve entertainment software. I suspect that adding any more games to the existing TI Casino suite will require either a DSDD compatible disk controller card, a large RAMdisk, or a convenient software method of running the entire package from two drives.

Klondike Solitaire is commercial and sells for $7 plus postage. It is available from Notung Software, Ramcharged Computers and other TI dealers. These dealers also sell other parts of the TI Casino series.

ACCESS

Harrison Software, and Marcel's Software (Speed Reading, Assembly Poker, AMS Video Titler): 5705 40th Place, Hyattsville, MD 20781. Phone (301) 277-3467

Notung Software (Klondike Solitaire and other games in the TI Casino suite): 7647 McGroarty St., Tujunga, CA 91042

Charles Good (source for P-GRAM Loader): P.O. Box 647, Venedocia, OH 45894. Phone (419) 667-3131. Preferred e-mail good.6@osu.edu (other previously published e-mail addresses are still valid until the end of 1996).

TI Web Site Reviews

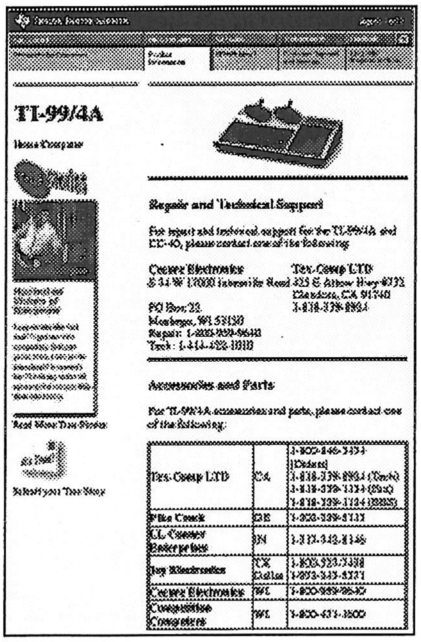
Dennis Remmer

dennis@dstc.edu.au

Sitename: **TI 99/4a Home Computer**

Owner: **Texas Instruments**

Address: `http://www.ti.com/calc/docs/994a.htm`



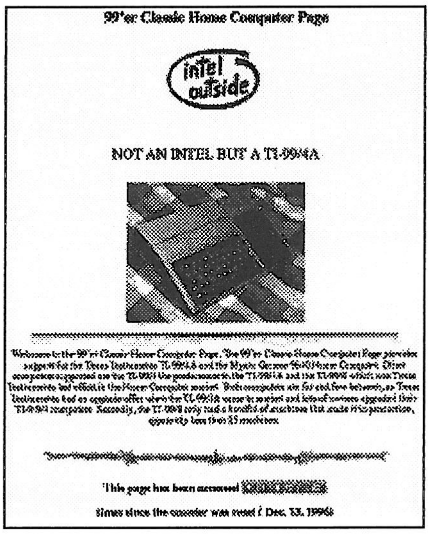
Comments: This site is the full extent of Texas Instruments's official recognition of their Home Computer. I guess we should be thankful at least for something! This page branches off from their Calculators product information page as 'Other Products'. Information provided includes Repair & Technical Support contact (Cecure & Tex Comp), and a brief list of some suppliers of accessories & parts in the US.

Rating: 2/10

Sitename: **99'er Classic Home Computer Page**

Owner: **John Van Weelie**
jvweelie@mgl.ca

Address: `http://www.mgl.ca/~jvweelie/index.htm`



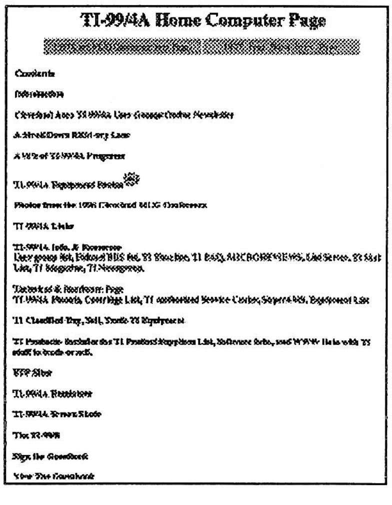
Comments: One of several really good starting points for TI and Geneve information on the 'Net. Site is updated fairly regularly, and includes sections and links on general information, modules, FAQ's, the mail servers, TI-99/4A pinouts, a cartridge list, SuperAMS, a repair information page, product and suppliers list, ftp sites, and emulators. The design is fairly ordinary, but the content is excellent.

Rating: 7/10

Sitename: **TI-99/4A Home Computer Page**

Owner: **Rich Polivka**
polivka@gwis.com

Address: `http://w3.gwis.com/~polivka/994apg.html`



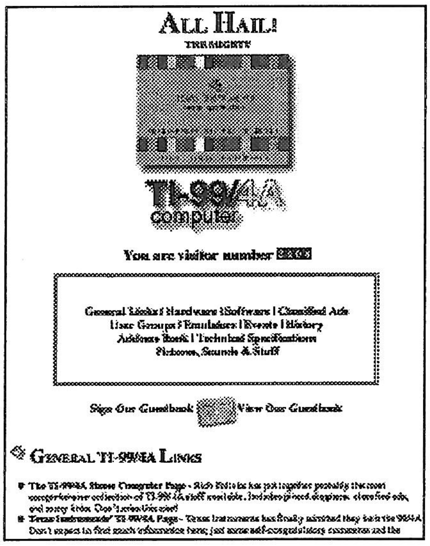
Comments: Another excellent collection of TI references and information, particularly with respect to local user groups and conference activity. Information includes various Cleveland area newsletters, some nice photos, a User group list, Fidonet BBS list, TI Timeline, TI FAQ, MICROREVIEWS, List Server, TI Mail List, TI Magazine, TI Newsgroup, a nice TI Classifieds section, lots of screen shots, emulator information, a terrific TI-99/8 page, and typical vendor/supplier lists and information. There is also a mechanism for you to be kept advised by email of changes to the site. A good site for general and slightly leftfield TI information.

Rating: 8/10

Sitename: TI-99/4a Shrine

Owner: John F. McDonnell
jmcdonnell@worldnet.att.net

Address: `http://www.geocities.com/Athens/7374/ti.html`



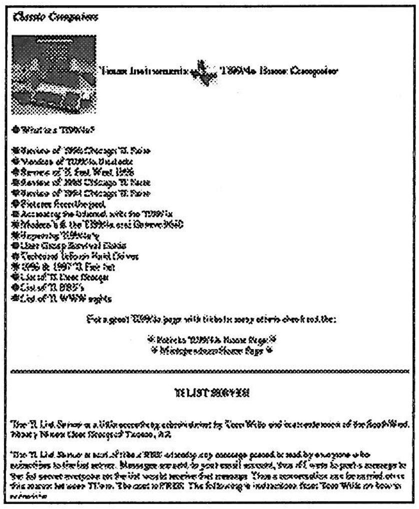
Comments: Wow! The best TI website out there. John has obviously put some thought into the simple design and usability of the aptly named Shrine. He has scoured the net to find the most appropriate information & links, and provide a single point of entry to all these resources. This is a regularly updated website with information grouped into General TI-99/4a Links, Hardware - Original, Hardware - Third Party, Software, Classified Ads & Auctions, Emulators, Events, History, Pictures, Sound, and lots more. I was hard-pressed to imagine anything he's left out. Everything is cross-referenced, and the links are kept up-to-date.

Rating: 10/10

Sitename: TI 99/4a

Owner: Gary W. Cox
garycox@netten.net

Address: `http://www.netten.net/~garycox/ti99idx.htm`



Comments: A farily basic design conceals what is mostly Gary's own work, including reviews of the Chicago TI Faire, Fest West, information on communications with the TI and Geneve, repairing TI 99/4a's, a User Group Survival Guide, technical information on Hard Drives, and lists for User Groups, BBS's and links to other sites. A good collection of one man's efforts, with a focus on events and technical information.

Rating: 7/10

TI Brisbane User Group

Committee

President: Rex Jones
3396 1662

Vice-President: John Campbell
3351 3107

Secretary: Garry/Col Christensen
3888 4857 / 3284 7783

Treasurer: Val Jones
3396 1662

Editor: Dennis Remmer
3870 5710
dennis@dstc.edu.au

Disk Librarian: John Reynolds
3357 9758

Tape Librarian: Bruce Campbell
3351 3107

Book Librarian: Trevor Campbell
3351 3107

Module Librarian: Bruce Campbell
3351 3107

All items within this newsletter may be reprinted providing the source and author are acknowledged. The views expressed in articles published in TIBUG are those of the author and do not necessarily reflect the views of the Editor, Committee Members or Members of this User Group. All items, articles, programs etc. in this Newsletter are believed to be public domain.

Contributions to TIBUG are invited from both members and non-members. Articles for inclusion in the succeeding bi-monthly newsletter are required at least two weeks before the monthly meeting and may be included in that newsletter at the discretion of the Editor.

Most original articles by members of TIBUG in this newsletter are on available on disk and are available to other User Groups on request.

Submissions of articles, reviews, comments and letters from members is encouraged, however the Editor asks that those submitting keep the following in mind:

Submissions should be about the TI Community in particular, computers in general, or of sufficient general interest. The preferred media is computer file, preferably in ASCII (Text) or Microsoft-Word compatible format, submitted on MacIntosh or IBM-compatible floppy disk or via Electronic Mail to the Editor. Handwritten submissions are acceptable but please remember that they have to be retyped. Other submissions, such as typed, printed or photocopied are welcome but must of reproducible quality.

Submissions are best sent directly to the Editor:

Dennis Remmer - TIBUG Editor
PO Box 30, Toowong QLD 4066
(Email: dennis@dstc.edu.au)

This space available (cheap!)

for your articles and notices

TI Brisbane User Group
PO Box 3051
Clontarf MDC, QLD 4019
Australia