Subscription Fees

- ❑ 6 issues USA $35
- ❑ 6 issues Canada/Mexico $42.50
- ❑ 6 issues other countries surface mail
  - ___ Surface mail $40
  - ___ Air mail $52

Outside U.S., pay via postal or international money order or credit card; personal checks from non-U.S. banks will be returned.

ADDRESS CHANGES: Subscribers who move may have the delivery of their most recent issue(s) delayed unless MICROpendium is notified six weeks in advance of address changes. Please your old address as it appears on your mailing label when making an address change.

Check each item ordered (or list on separate page and enter total amount here

_____

Check/MO ❑ ❑ ❑

Card No._____

Expiration Date_____

(Minimum credit card order is $9)

Signature_____

(Required on credit card orders.)

No sales tax on magazine subscriptions. Texas residents add 8.25% sales tax on other items, including back issues and disk subscriptions. Credit card orders add 5%.

Name_____

Address_____

City_____ State____ ZIP_____

The set of numbers at the top of your mailing label indicates the cover date of the last issue of your subscription.

PERIODICALS

Micropendium
P.O. Box 1343
Round Rock, TX 78680

T

45894

OH

A 1/99
CHARLES GOOD
P.O. BOX 647
VENEDOCIA

Covering the TI99/4A and Geneve home computers

MICROpendium

Volume 15 Number 3 May/June 1998 $6

TI and the Internet?

See Page 6

XBASIC Programs

3TO5COLCAT
ARRANGER
OVERLAYER

The Art of Assembly

It's all downhill from here, Bruce Harrison says

TI Marketing: Still cracking the whip?

REVIEWS:

TI SCSI Cataloger
Atari Super Storm

CONTENTS

MICROpendium

MICROpendium (ISSN 10432299) is published bimonthly for $35 per year by Burns-Koloen Communications Inc., 502 Windsor Rd., Round Rock, TX 78664-7639. Periodical postage paid at Round Rock, Texas. POSTMASTER: Send address changes to MICROpendium, P.O. Box 1343, Round Rock, TX 78680-1343.

All correspondence should be mailed to MICROpendium at P.O. Box 1343, Round Rock, TX 78680.

Foreign subscriptions are $42.50 (Canada and Mexico); $40 surface mail to other countries; $52 airmail to other countries.

All editions of MICROpendium are mailed from the Round Rock (Texas) Post Office.

Mailing address: P.O. Box 1343, Round Rock, TX 78680.

Telephone & FAX: (512) 255-1512

Internet E-mail: jkoloen@earthlink.net

Home page: http://www.earthlink.net/~jkoloen/

John Koloen Publisher

Laura Burns Editor

COMMENTS

Fest West '98 — the aftermath

Tom Wills put in an enormous amont of time and effort into getting Fest West '98 — Lubbock off the ground. After the event he revealed some of the behind-the-scene problems that occurred, most of which were created by TI's mystifying marketing department. We're reprinting most of the article in this issue. I was amazed at how petty a large corporation can be. Perhaps it will surprise you as well.

Lima coverage next month

Next issue we hope to give a report on the 1998 Lima Multi Users Group meeting May 15-16. The MUG, free to participants and vendors, has traditionally been one of the highlights of the year for TI99/4A users. However, the 1998 MUG will be the last, according to Charles Good, its chief organizer. Many thanks to Charlie for his efforts over the years.

Scheduled speakers for this year include Dan Eicher, Lew King, Ted Zychowicz, Bob Carmany, Bruce Harrison, Delores P. Werth and Jim Krych.

If it's not one thing, it's another

Some subscribers to the MICROpendium disk series reported problems with a number of the files and programs included on the disk. Indeed, there were some problems, which I've corrected. The problems come are the result of transferring files from our Mac to the TI. I'll be more vigilant in the future. In any case, I'm including the problematic files from March/April as an archived file on the May/June disk.

BUGS AND BYTES

MATIUG considers BBS

The Milwaukee Area TI User Groupmay restart the MATIUG BBS, according to information posted on the TI list server by Ted Zychowicz. He asked list server subscribers whether they would call the BBS. He said the BBS could be online as early as June.

Membership inf MATIUG is $15 per year when started in January. Memberships started at a later date are $10. Members receive a newsletter and access to the group's software library.

For more information, contact Zychowicz by e-mail at tedzychowicz@juno.com, or MATIUG, c/o Gene Hitz, 4122 N. Glenway, Milwaukee, WI 53222.

Continued on page 4

BUGS AND BYTES

Continued from page 3

Sister Pat has web page

Sister Pat Taylor, known to many TI users for her computer ministry at Marian Hall, the nursing home where she is a resident, now has a Web site. Users can find it at: http://users.mwci.net/~sisterpat/.

OPINION

TI marketing department cracked whip on FW98

BY TOM WILLS

Tom Wills was the driving force behind Fest West '98—Lubbock and published the following in the newsletter of the SouthWest Ninety-Niners, the sponsoring user group.—Ed.

Fest West '98—Lubbock is history, and it all went well, right? Well, maybe not so right.

Most people think the planning all went well. All I've publicly stated until now were the positive aspects of setting up FW98. However, there were many negatives that occurred during the process which I want to pass along.

One of the first things I was able to secure was the use of the TI logo as used on the TI web site. Plus, I was able to get our Fest West '98 web page included as a link from the TI99/4A home page on the TI web site.

For those who don't know, a web page is a "page" you go to on the Internet. A web site and home page are the main page of the Internet on a particular server. This is a bit of gray area, as the home page/web site for the SouthWest Ninety Niners User Group is just one of many on the server run by The River Communications.

In our case, the home page (or web site) is the user group page. The FW98 web page is a branch of that web page. A "link" is a selection on one web page that when it is double-clicked will take you to another web page.

In the example above, the Texas Instruments web site had a web page devoted to the TI99/4A. The webmaster (the person whose job it is to keep the Internet server up and running) had secured permission for me to use the TI logo.

He also agreed to include the FW98 web page link on the TI99/4A web page, and we would, in return, include the TI99/4A web page link on our web page. This was a good agreement for all. As part of the agreement to use the logo, I had to agree that it would be used only in conjunction with Fest West related activities, and nothing else.

OPINION

After several months of using the logo, and having the link between the TI web page and ours, TI's marketing department decided it was inappropriate for us to use their logo and I was to discontinue using it immediately! And I was to remove all references to the TI web site from our server and they had the reference to our web site removed from the TI web site.

Then, to top it off, I was told that I could not refer to the users of the TI99/4A computers as "Tiers." The reason? That's what they call their employees. Call TI99/4A users TI99ers met with approval, but not by much.

Which just goes to show the TI marketing department is still one of TI's worst enemies. Heaven forbid that we actually look like we were having anything to do with Texas Instruments. And any appearance that TI was in any way helping sponsor FW98 was strictly to be discounted. They were just cooperating with us. And that is an optimistic way of stating it.

Anyone who was accessing the FW98 web page probably remembers seeing the changes around the end of September. That was why.

But that wasn't the end of it. I was repeatedly called by TI and told to change this, that, and the other thing on our web page to meet their liking. For a short while, I thought FW98 was going to be called off by TI.

The last thing that TI interfered with was the declaring of FW98 Saturday as a special day. They objected to it on the grounds that they were going to be celebrating their 25th year in Lubbock and want that date declared a special day.

So, even though it was requested by a Lubbock resident, a concerned citizen, that Feb. 14 be declared a special day, and per the request of the mayor's office that we set aside about 10 minutes for her to make her presentation, Texas Instruments was apparently able to shut down this idea.

Even the Monday before FW98, everything was "go" on the presentation. On Saturday, Feb. 14, the mayor was a no-show. After asking what had happened, the mayor's office said the mayor had been called out of town at the last minute. They said they notified TI. Strangely, TI didn't relay the information to us.

Well, due to the nitpicking of the TI marketing department and the way they kept us in line, so to speak, it has cost them some business. I have decided that the laptop computers we are buying at work will not be TI laptops. Their marketing department left me with a bad taste in my mouth. And, as they are the same department that basically ruined it for the TI99/4A to begin with, I want no more to do with them than necessary. Who knows what they'd do to us next?

I am not trying to influence anyone else with my thinking. Nor is this to be construed as an official stand of the SouthWest Ninety Niners User Group. These paragraphs are mine alone.

I just wanted everyone to know that it wasn't all easy going. I do not blame the Lubbock TI officials for any of the above problems. I feel it all came out of the Dallas home office.

Does the TI have a place on the Net?

Coming closer to the Internet

BY MICHAEL ZAPF

This article is intended to inform you about some basics of the Internet and to show that we can connect a TI to the Internet.

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 weren't many services. You could at most send some messages, but gradually networks became very sophisticated. Today networks often consist of a server that is equipped with many resources, such as large hard drives, 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 Defense instructed scientists to develop an American internet that is fail-save in case of the 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 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 electronic 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 can find many thousands of them. Messages sent to a newsgroup are routed to a special computer (also called a "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 a newsgroup called "comp.sys.ti", for instance.

In order to enhance file transfer on the Internet, the File Transfer Protocol (FTP) was specified. FTP made it possible to store files at well-known locations on the net and retrieve them.

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), often just called "the web." 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 start of web addresses). This allows you 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, multimedia elements can be integrated so that images, sounds, video clips, and so on can be sent along with the text. The latest trend in the web is to transport functionality to the client using a script language called "JavaScript" or the platform-independent programming language "Java." Now web pages not only present one static layout to the client but provides him with small programs called "applets" that are downloaded and executed automatically at the client's computer.

While the development of distributed file access and remote execution was more interesting for subsections of the Internet, the global data exchange started to grow exponentially with the invention of the World Wide Web

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, information, connectivity, home banking, electronic commerce, and much more.

It is clear by now that this enhancement becomes a true threat to the effec-

Continued on page 8

Continued from page 7

tiveness 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, replacing the current one.

The Infrastructure

If someone destroys 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 text. 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 development.

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 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 does just a small job and is soon ready to continue with any other work. If somebody seems to work unsatisfactorily, he can be easily replaced.

A Matter of Layers

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 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.

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 recipient. 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 information can again be found in

Continued on page 10

Continued from page 9

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.

Addresses in IP are represented by four integer numbers between zero and 255, separated by a dot, such as "141.2.150.16". A special service called "Domain Name Service" converts these numbers to a better readable format, such as "www.vsb.cs.uni-frankfurt.de" (note that the parts of both addresses do not directly correspond to each other). The latter format is normally used when typing in an address, but it is always converted to the numeric format. While the first one to three numbers refer to a network, the remaining numbers identify a host in this network.

The IP layer maintains a structure called a "routing table." It contains IP adresses of hosts and networks (host part set to zero) and the corresponding output device. If a host has two interfaces it uses this table to decide which way to forward the data. Data are grouped in packets with an IP header that contains the source and destination IP address.

Transport Layer — The network layer can work only with data strings of limited length, also called packets. This means that longer data strings are broken into suitable lengths 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 correct order. 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 1,500 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.

Hosts may even provide both interfaces but programs running on it are completely unaware of this fact — all they need to know are the correct destination addresses.

2. The higher layers have no idea about 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.

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.

LANs (Local Area Networks) are often composed in a simpler way, so you might ask why such complex processing must be done on 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. At the time when the concepts were gathered, the word "computer" was not directly associated with a "Wintel" PC as it is today. Since a worldwide 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.

Connecting the TI

There are already questions like: "Can we write a web browser that runs on the TI?" My answer is "No." It's at least unlikely. The problem is that the requirements of the WWW are very strong; many web pages already have Java applets, requiring a Java interpreter. And even if there's no Java, there are a lot of graphics, interactive fields, and so on.

However, the Net is not the Web. That means that the Internet is much more than just the WWW; it offers newsgroups, e-mail, or file transfer. These three applications should be focused on by us. They don't need graphics or the like, they belong to the basic functionality of the Internet, and they are by far more

Continued on page 12

Continued from page 11

important than the graphics-oriented WWW. So, if someone says, the Internet consists of colorful pages with sound and more, tell him he's wrong.

The first step is to build a simple TCP/IP stack. All that functionality that I described above is only necessary if we want a complete TCP/IP stack that also offers server and router functionality. However, we need only a small subset of these features. Our TI will not forward any data, and it will only connect by a point-to-point link via a serial interface (modem). Taking this into account, we can shrink the huge implementation to a handy size that will fit into the TI's limited memory.

Next we must write client applications. The first one will be a File Transfer Protocol client. This allows us to connect to an FTP server which offers files for download (just like a BBS) and which can also store uploaded files. There are already numerous servers that offer TI files. The problem that everyone faces when downloading files from such a server is how to transfer the files to the TI — since we cannot connect the TI directly to the server at this time. If we had such an FTP client, this would be possible.

After that, we can write a POP mail client. "POP" stands for Post Office Protocol and defines how to send and retrieve e-mails from a mail server. This should be even less complicated than the FTP client.

The same holds for the news client that eventually allows all of us to use the newsgroups, e.g., comp.sys.ti. The advantage of the newsgroups compared to e-mail distributors (like "majordomo") is that you can decide whether you want to download some news article or not; e-mails are transferred completely; and very often, you are absolutely not interested in the message but you had to wait for it to be transferred. Unlike mailing lists, the newsgroups do not maintain a list of subscribers. Being subscribed just means to download the messages of this group from the news server.

Requirements

In contrast to what many people suggested, I think it is wiser to rewrite the TCP/IP stack completely in assembler. It won't be useful to try to port existing TCP/IP implementations because we still have to truncate it wherever possible. On the other hand, all those implementations are written in C, and even if we had a standard ANSI C compiler, it would result in too much code which will execute too slowly.

It could be possible to fit the whole TCP/IP stack into the 32K of the TI99/4A. However, we must allocate some memory for buffers that are necessary when passing data between the different layers, let alone the application that is to use the stack, e.g., the FTP client. All this makes it seem unlikely that a "normal" TI could suffice. As I use a Geneve, the first implementation will be on the Geneve which offers a very good base in my opinion. It has a lot of memory

and a (relatively) fast CPU. I think that memory is the bigger problem — it could be possible that an expanded TI will also do.

Another problem is the ISP (Internet Service Provider). There is virtually no use of asking anyone of them if we can connect our TI to their machine. If they do remember it, they'll laugh and try to convince us to buy a cheap PC. (Some local ISPs in the U.S. provide dial-in service using a command line *interface. These systems typically provide support for e-mail systems such as ELM or PINE as well as FTP and text-based access to the Web. They can be accessed by virtually any computer capable of running a terminal emulator program. I have used them at speeds as low as 2400 baud with good results.—Ed.)*

Speed is normally no issue in the Internet; there is no definition of a minimum speed, and you can experience very long waiting times even if you have a fast computer. The modem connection speed could become a problem, however: We should expect a minimum rate of 14,400 bits per second which is faster than most older RS232 cards can deliver. This is not compliant to the TCP/IP specifications, but who really knows them. Every PC can connect with 14,400 bps, so what?

As I use a Geneve, the first implementation will be on the Geneve which offers a very good base in my opinion. It has a lot of memory and a (relatively) fast CPU.

I was glad to hear that many of you actually managed to get a higher (yet reliable) connection rate than 9600 bps. This means that we won't get any problems with the connection.

Activities

During recent months I gathered some interested people around the world who want to assist me in writing the TCP/IP stack. We are using a mail server which is called:

tcpip99@vsb.informatik.uni-frankfurt.de

People who want to join simply need to send the message "subscribe tcpip99" to majordomo@vsb.informatik.uni-frankfurt.de. However, don't be surprised if you do not find any traffic on it currently. The work had not enough progress up to now so that it could be divided among the people. This does not mean that there are severe, unsolvable problems. As the membership in this list is free, I would ask everyone to stay patient. As of March 1998, the current state is as

Continued on page 14

Continued from page 13

follows:

Working SLIP driver — It is possible to connect the Geneve to a SLIP driver on a PC. The driver uses input and output queues and processes escape sequences correctly.

IP driver — The driver is not yet complete. One of its most important components was completed some months ago — the reassembly algorithm. It was successfully tested as a stand-alone program using a file of mixed fragments of text strings (together with offsets and identificators) as input.

ICMP — Messages in the Internet Control Message Protocol are used to convey important information to hosts concerning the connection. One often used message is the "Echo request" that should be answered by an "Echo reply." The "ping" program in common TCP/IP implementations uses these messages to check if a host is present and how long it takes for messages to travel between the hosts. Using the (incomplete) ICMP implementation, the Geneve proved to be capable of answering pings from a standard PC — that means the PC accepted the replies as if they came from any other TCP/IP-capable host!

The Future

Now for the future plans. The next step is to eventually integrate the reassembly algorithm in the IP driver. After that, we enter layer 4. TCP is a little hard for the beginning, so I suggest to take UDP as a start. This means we have to design what is called "sockets" in common TCP/IP implementations. UDP is rather easy, so it should not take too long before we can indeed transfer application data between a PC and the Geneve. Finally, TCP must be implemented. If this is done, we can start writing applications like file transfer programs, e-mail readers and so on.

Addendum: How to Receive Bytes Asynchronously

One of the first interesting things I found out is how to open the serial interface for asynchronous reception of bytes. The RS232 port triggers an interrupt each time a byte is successfully received so that it can be read by the DSR (Device Service Routine). Alternatively, the port can be polled for newly received bytes, but this does not seem suitable to me for an interface driver in the TCP/IP stack. The program is written in assembler and runs on the TI or the Geneve with an RS232 card.

The program uses the so-called Circular Input Buffer (CIB). It is located in the VDP RAM and uses the following pointers in the PAD:

>8300 Start address of CIB (word)
>8302 Buffer length (byte)
>8303 Buffer bottom, offset to start (byte)
>8304 Buffer top, offset to start (byte)

The device must be opened by a special OPEN call with the leftmost bit set to one. This turns on the external interrupt so that the computer can be notified if a new byte must be read from the data register of the RS232 port. When the buffer is full, the program exits.

Every ring buffer uses two pointers — bottom and top. If both are equal, the buffer is empty. If top is just below bottom, the buffer is full. In this case, the character >FE is stored in the buffer, indicating an overflow. The character >FF indicates bad data (e.g. parity error). Beware: Both could also be data that has been received normally.

```
*******   Simple CIB receiver  *******

        DEF  MAIN
        REF  DSRLNK,VMBW

* PAB definition
* Device is opened with the >80 code, enabling
* the interrupt trigger
* The buffer address is of no use here.

PABDAT  DATA >8000,>1800,>FF00,>0000,>0017
        TEXT 'RS232.BA=9600.DA=8.PA=N'
        EVEN

* Special characters

FEFF    DATA >0000,>0000,>183C,>7EFF
        DATA >FFFF,>FFFF,>FFFF,>FFFF

MAIN    LI   R0,>0F80
        LI   R1,PABDAT
        LI   R2,33
        BLWP @VMBW          write PAB
        LI   R0,>0FF0
        LI   R1,FEFF
        LI   R2,>0010
        BLWP @VMBW          define >FE,>FF

        LI   R0,>0F89
        MOV  R0,@>8356
        CLR  @>837C
        LIMI 0              no interrupts now
        BLWP @DSRLNK        OPEN the RS232 port
        DATA 8
        JEQ  OUT

        CLR  R0
```

Continued on page 16

Continued from page 15

```
        MOV  R0,@>8300    CIB to be seen on screen
         LI   R1,>FF00     255 characters long (maximum)
        MOV  R1,@>8302    write that and clear BUFBOT
        SWPB R1
        MOVB R1,@>8304    and clear BUFTOP, too.

        LIMI 2            Off we go...
LOOP    MOVB @>8303,R1
        AI   R1,>FF00
         CB   R1,@>8304
         JNE  LOOP         continue until buffer full
        JMP  QUIT         then just exit

OUT     MOVB R0,@>D000    can be PEEKed later (-12288)
QUIT    BLWP @>0000

        END
```

Please send your remarks and questions to the following e-mail address: zapf@vsb.informatik.uni-frankfurt.de

If you have a web browser on your computer, you are welcome to visit my home page at: http://www.uni-frankfurt.de/~zapf/

References

[1] W. Richard Stevens: *TCP/IP Illustrated, Volume 1, The Protocols.* Addison-Wesley

[2] W. Richard Stevens: *TCP/IP Illustrated, Volume 2, The Implementation.* Addison-Wesley

MDOS 6.00 Y2K compliant

The following was posted by Tim Tesch on the TI list server.—Ed.

MDOS version 6.00 and later are year 2000 compliant.

"While I could only test MDOS using the date routine, I am confident that the Ge is good for the next 60 years," he said. The two routines which calculate the date now into consideration the fact that the year 2000 is a leap year. This means that dates in and beyond will be properly recognized and displayed.

In addition to changing the date code, MDOS 6.00 displays the full year when a DIRectory is requested. Programs not set up to use the full year will still display only last two digits, but it is a step forward.

"My only big concern right now is for anyone using databases coded for the two-c system," Tesch said. "Perhaps someone more familiar with Multiplan and other DBs c enlighten the TI/Geneve community on how this could impact their calculations. I s this not to alarm everyone, only to caution you."

Remainder of Extended BASIC program listed here

Those who tried to input the 3TO5COLCAT by Leonard W. Taffs, which appeared in the January/ February 1998 edition got about half way through the listing where it abruptly ended. We don't have any idea why this happened. However, we are chagrined. The listing stopped at line 440. Following is the remainder of the program. The program in its entirety will be included on the May/ June MICROpendium disk.

```
440 DISPLAY AT(A,1):AR$(I)::
 DISPLAY AT(A+2,1):RPT$(" ",
28):: A=A+1 :: IF A=23 THEN
A=1
450 CALL KEY(0,K,S):: IF S<>
1 THEN 470
460 CALL KEY(0,K,S):: IF S<>
1 THEN 460
470 NEXT I
480 DISPLAY AT(24,1):"READ A
GAIN OR PRINT (A/P)"
490 CALL KEY(0,K,S):: IF S<1
 THEN 490
500 IF (K=65)+(K=97)THEN CAL
L CLEAR :: GOTO 390 ELSE IF
(K=80)+(K=112)THEN 510 ELSE
IF (K=81)+(K=113)THEN 850 EL
SE IF K=13 THEN 490
510 PRINT :: INPUT "3, 4, or
 5 COLUMNS? ":C :: DISPLAY A
T(22,1):RPT$(" ",56):: IF C=
3 THEN C3=1 ELSE IF C=4 THEN
 C4=1 ELSE IF C=5 THEN C5=1
515 IF C=0 THEN 390 ELSE IF
C=9 THEN 860
520 PRINT :"NOW INITIALIZING
 ARRAYS FOR PRINTING.......
." :: PR=1 :: OPEN #3:"PIO",
VARIABLE 132
530 INC2=INC-1 :: IF C3 THEN
 X=INC2/3 ELSE IF C4 THEN X=
INC2/4 ELSE IF C5 THEN X=INC
2/5 ! X established
540 X$=STR$(X):: P=POS(X$,".
",1):: IF P THEN ADJ=1
550 X=INT(X)+ADJ :: X2=X
560 FOR I=1 TO INC :: DISPLA
Y AT(23,18):I
570 IF AR$(I)="" THEN 680
580 IF J1 THEN 590 ELSE Z=Z+
1 :: IF Z<=X THEN RS1$(Z)=AR
$(I):: IF Z=X THEN J1=J1+1 :
: Z=0 :: GOTO 630 ELSE 630
590 IF J2 THEN 600 ELSE Z=Z+
1 :: IF Z<=X THEN RS2$(Z)=AR
$(I):: IF Z=X THEN J2=J2+1 :
: Z=0 :: GOTO 630 ELSE 630
600 IF J3 THEN 610 ELSE Z=Z+
1 :: IF Z<=X THEN RS3$(Z)=AR
$(I):: IF Z=X THEN J3=J3+1 :
: Z=0 :: GOTO 630 ELSE 630
610 IF J4 THEN 620 ELSE Z=Z+
1 :: IF Z<=X THEN RS4$(Z)=AR
$(I):: IF Z=X THEN J4=J4+1 :
: Z=0 :: GOTO 630 ELSE 630
620 IF J4 THEN Z=Z+1 :: IF Z
```

Continued on page 18

3T05COLCAT

Continued from page 17

```
<=X THEN RS5$(Z)=AR$(I):: IF
Z=X THEN J5=1 :: Z=0
630 !
640 !
650 CALL KEY(0,F,G):: IF G<>
1 THEN 670
660 CALL KEY(0,F,G):: IF G<>
1 THEN 660
670 STP=0 :: NEXT I
680 REM ** PRINT HARDCOPY **
690 PR=1
700 IF C3 THEN IF PR THEN PR
INT #3:TAB(10);AR$(0);"   ";
INC-1;"Files   ";DATE$:TAB(4
);R3$ :: GOTO 750
710 IF C5 THEN OPEN #5:"PIO"
 :: PRINT #5:CHR$(15);CHR$(2
0);CHR$(27);CHR$(77);:: CLOS
E #5 ! print Condensed 5 col
720 IF C4 THEN OPEN #5:"PIO"
 :: PRINT #5:CHR$(15);CHR$(2
0);CHR$(27);CHR$(77);:: CLOS
E #5 ! print Condensed 4 col
730 IF C4 THEN IF PR THEN PR
INT #3:TAB(10);AR$(0);"   ";
INC-1;"Files   ";DATE$:TAB(5
);R4$
740 IF C5 THEN IF PR THEN PR
INT #3:TAB(20);AR$(0);"   ";
INC-1;"Files   ";DATE$:TAB(3
);R5$
750 REM ** PRINT HARDCOPY
760 FOR I=1 TO X
770 IF C3 THEN IF PR THEN PR
INT #3:TAB(4);RS1$(I)&"|| ";
TAB(32);RS2$(I)&"|| ";TAB(58
);RS3$(I)
780 IF C4 THEN IF PR THEN PR
INT #3:TAB(5);RS1$(I)&"|| ";
TAB(33);RS2$(I)&"|| ";TAB(61
);RS3$(I)&"|| ";TAB(89);RS4$
(I)
790 IF C5 THEN IF PR THEN PR
INT #3:TAB(3);RS1$(I);"|";TA
B(31);RS2$(I);"|";TAB(57);RS
3$(I);"|";TAB(83);RS4$(I);"|
";TAB(110);RS5$(I)
800 NEXT I
830 ! PRINT #3:"1234567890";
:: NEXT I :: PRINT #3:RS1$(1
);"|";RS2$(1);"|";RS3$(1);"|
";RS4$(1);"|";RS5$(1) ! test
 line
840 PRINT #3: : : : : : : ::
PRINT #3:CHR$(27);"@@" :: CL
OSE #3
850 PRINT :: INPUT "ANOTHER
DISK? (Y/N) ":AG$ :: IF (AG$
="Y")+(AG$="y")THEN CALL CLE
AR :: GOTO 870 ELSE PRINT :A
R$(0):INC-1;"Files":
860 CALL SCREEN(11):: GOTO 8
80
870 DISPLAY AT(13,6):"Re-Run
ning Program" :: RUN
880 CALL CLEAR :: PRINT "THA
NK YOU IF YOU USED THIS": :"
PROGRAM.^ Please Send Any":
:"^ comments to author at":
:
890 PRINT "^ 4124 E. FIRST S
TREET": :"^ TUCSON, AZ. 8571
1-1006": :"E-MAIL wltakts@az
starnet.com": :"^^^^ THANKS!
Leonard": : :
```


3T05COLCAT

```
900 DISPLAY AT(24,1):"Press
<ANY KEY> to Terminate" :: C
ALL KEY(0,K,S):: IF S<1 THEN
900
910 STOP
```

EXTENDED BASIC

Arranger program manipulates columns in TI-Writer files

BY JACQUES GROSLOUIS

Many articles have been written describing how to use TI-Writer files to create small databases. Programs have been written to sort these databases using any selected column. One of these programs is CHARTBASE by Jerry Stern, which appeared in the May 1989 issue of MICROpendium. This program used the tab settings in a TI-Writer file to identify the various columns.

The program listed below, ARRANGER, uses many routines from CHARTBASE and allows you to rearrange the order of columns. In addition, you may insert blank columns with widths of 1 to 9 characters between existing columns. This can be useful where an existing file is missing a column of information. TI-Writer or Funnelweb text editor is used to add the required data.

The program first asks you to name your input and output files. The input file must contain tabs, otherwise the program will crash.

After naming the output file, you can decide to add tabs to the output file. The first record of the input file is then displayed with each column identified by a letter of the alphabet, such as "ABCD." The required rearrangement is entered by using the identified columns, such as "DCBA."

Additional columns are created by inserting numerals — "D4C4BA" would add two extra columns of four characters each. A display of the old and new record is then presented and you are allowed to change the arrangement by pressing "Y." An option to rerun the program is presented before exiting.

For Funnelweb users and owners of Horizon RAMdisks, the name of the output file is placed in the FWB mailbox ready to be called up if the file requires editing.

ARRANGER

```
100 ! SAVE DSK1.ARRANGER !13
3
110 CALL CHAR(128,"0000FF"):
: CALL CLEAR :: CALL BLUE !1
40
120 DISPLAY AT(1,7):"Column
Arranger":"    By Jacques Gr
oslouis" :: CALL HCHAR(3,1,1
```

Continued on page 20

EXTENDED BASIC

Continued from page 19

```
28,32)!252
130 DIM P(16),R$(300):: V$="
ABCDEFGHIJKLMNOP" !133
140 DISPLAY AT(8,1):"Name of
 file to load?":"DSK1.INPUT"
 !205
150 DISPLAY AT(11,1):"Name o
f output file ?":"DSK1.OUTPU
T" !136
160 ACCEPT AT(9,4)VALIDATE(U
ALPHA,DIGIT,"._")SIZE(-12):S
$ :: IF S$=" " THEN 160 ELSE
 S$="DSK"&S$ !237
170 ACCEPT AT(12,4)VALIDATE(
UALPHA,DIGIT,"._")SIZE(-12):
SS$ :: IF SS$=" " THEN 170 E
LSE SS$="DSK"&SS$ !110
180 ON ERROR 140 :: DISPLAY
AT(14,1):"Include tab line?:
N" :: ACCEPT AT(14,19)SIZE(-
1)VALIDATE("ynYN")BEEP:Q$ !1
80
190 OPEN #1:S$,DISPLAY ,VARI
ABLE 80,INPUT :: L=0 !185
200 IF EOF(1)THEN 220 !010
210 L=L+1 :: LINPUT #1:R$(L)
:: IF LEN(R$(L))<3 THEN L=L-
1 :: GOTO 200 ELSE 200 !183
220 CLOSE #1 :: ON ERROR STO
P :: L=L-1 :: R$(0)=R$(L+1)!
139
230 N=(POS(R$(0),CHR$(213),5
)-5):: IF N=0 THEN N=16 !091
240 FOR T=1 TO N :: P(T)=ASC
(SEG$(R$(0),T+4,1))-133 :: N
EXT T :: P(T)=80 !215
250 CALL HCHAR(1,1,32,544)::
 DISPLAY AT(1,1):L;" Record
fields start at:" !048
260 FOR T=1 TO N :: DISPLAY
AT(T+1,1):P(T);TAB(5);SEG$(R
$(1),P(T),MIN(20,P(T+1)-P(T)
));TAB(27);CHR$(T+64):: NEXT
 T !201
270 DISPLAY AT(22,1):"Enter
rearrangement required":"Dig
its to insert # of spaces":A
$ !033
280 C$=SEG$(V$,1,N):: ACCEPT
 AT(24,1)SIZE(-16)VALIDATE(C
$,DIGIT):A$ :: IF A$="" THEN
 270 !061
290 CALL HCHAR(21,1,32,128):
: FOR T=1 TO 1 :: GOSUB 370
:: NEXT T :: B$="" :: DISPLA
Y AT(24,8):"OK to continue N
" !118
300 CALL KEY(0,K,S):: IF S<1
 THEN 300 ELSE IF K=78 OR K=
110 THEN 250 !077
310 OPEN #2:SS$,DISPLAY ,VAR
IABLE 80,UPDATE :: CALL MAIL
BX(SS$)!207
320 CALL HCHAR(23,1,32,64)::
 FOR T=1 TO L :: GOSUB 370 :
: PRINT #2:B$ :: B$="" !220
330 NEXT T :: IF Q$="Y" OR Q
$="y" THEN GOSUB 420 !023
340 CLOSE #2 :: DISPLAY AT(2
4,4):"Rearrange Another?:Y"
!227
350 CALL KEY(0,K,S):: IF S<1
 THEN 350 ELSE IF K=89 OR K=
121 THEN 110 !247
360 END !139
```


EXTENDED BASIC

```
370 DISPLAY AT(18,1):"Old:";
R$(T):: FOR W=1 TO LEN(A$)::
 A=ASC(SEG$(A$,W,1))-64 !057
380 IF A<0 THEN A=A+16 :: B$
=B$&RPT$(" ",A):: GOTO 400 !
175
390 B$=B$&SEG$(R$(T),P(A),MI
N(LEN(R$(T))+1,P(A+1))-P(A))
!207
400 NEXT W :: DISPLAY AT(20,
1):"New:";B$ !069
410 RETURN !136
420 PRINT #2:CHR$(128);CHR$(
134);CHR$(128);CHR$(213);!18
1
430 FOR T=1 TO LEN(A$):: PRI
NT #2:CHR$(P(T)+133);:: NEXT
 T !082
440 FOR X=LEN(A$)+1 TO 16 ::
 PRINT #2:CHR$(213);:: NEXT
X !121
450 PRINT #2:CHR$(128);CHR$(
134):: RETURN !086
26400 SUB MAILBX(A$)!254
26405 CALL PEEK(8198,A,B)::
IF A=170 AND B=85 THEN 26410
 ELSE CALL INIT !012
26410 FOR Z=1 TO LEN(A$)!246
26415 Y=ASC(SEG$(A$,Z,1)):
CALL LOAD(-24577+Z,Y)!251
26420 NEXT Z :: Y=32 !184
26425 FOR Z=LEN(A$)+1 TO 80
:: CALL LOAD(-24577+Z,Y):: N
EXT Z !146
26430 SUBEND !168
29505 SUB BLUE !149
29510 ! SWITCHES DISPLAY TO
WHITE ON BLUE; JLS 7/88 !230
29515 CALL SCREEN(5):: FOR L
=0 TO 14 :: CALL COLOR(L,16,
1):: NEXT L :: SUBEND !202
```

THE ART OF ASSEMBLY — PART 70

The bathtub curve, or it's all downhill from here

BY BRUCE HARRISON

Today we're starting with a subject that's involved more with probability and statistics than with programming. This should be of interest to the more general segment of the readers of MICROpendium than to the Assembly buffs. To make it even more eye-catching, we've included a picture, shown as Fig. 1.

The Picture Explained

No, we didn't make the picture in Fig. 1 using some exotic equation with our plotter program. There probably is an equation for this curve, but we simply fudged the curve using our drawing program.

This curve is normally used in connection with the life cycle of some product or other, and depicts the frequency of failures for the product over time. Notice

Continued on page 22

THE ART OF ASSEMBLY PART 70

Continued from page 21

that neither time nor failure rate is quantified on the graph. The time span may be just a few years or a whole lifetime, but the general shape of the curve is what's important.

The first part of a life cycle shows that failure rates are initially very high while the product is in development. Failures come down during this early period, as the product gets improved upon. Thus the early part of the curve is referred to as a "learning curve." After this learning part, the failure rate levels off into what's called the "random failures" portion, with a relatively constant low failure rate. During this flat part of the curve, things fail at random intervals, but not too many failures occur.

As you can see, the biggest part of the curve in terms of time is this random failure part. Finally, though, time catches up with our product, and failure rate increases rapidly in what's called the "wearout" phase of the product's life cycle.

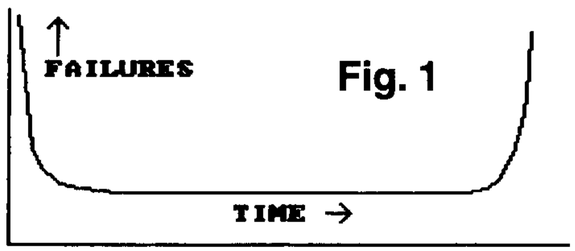

Fig. 1

This may well be what awaits us in the near future with our beloved TI-99/4A. We've been enjoying that flat part of the curve for more than ten years now, but we are about to be slapped in the face by the other end of the bathtub. That's the thinking which has inspired our friend Mike Wright and others to create a full-blown emulation of the TI-99/4A on modern PC computers. In theory, at least, we can transfer our beloved computer into a product that's in the early part of the random failure phase, and thus can continue to enjoy the TI's unique capabilities while the machines themselves are dying out very rapidly.

Other Applications

The Bathtub Curve can also apply to other things, such as the life cycle of a programmer. When first we start programming a computer, we suffer frequent failures of one kind or another during the learning period. As we learn, our failures level off, so there are little mistakes or bugs here and there, but these become infrequent and easily corrected. Sooner or later, however, we reach the wearout phase.

In humans, this may affect either the mind or the body. For most of us, including your author, it seems the body starts to fall apart first. Thus the fingers don't always hit the right keys, the eyes get more myopic, and the energy required even to type simple programs gets harder to muster. Thus while the mind is still productive of sound ideas, the body won't allow them to be realized as programs. In other words, that steep back end of the bathtub is catching up with us.

THE ART OF ASSEMBLY PART 70

Colorful Filling

Today's sidebar has nothing to do with bathtubs, but deals with a question posed many moons ago by an anonymous reader. We answered a letter from that reader a long time ago, but the fact that he was concerned enough to write us triggered our putting some source code in today's column.

The question was first put to us through Laura Burns as "how do you fill an area with color?" That was later expanded in a letter from the reader to your author as "how do we fill a bounded area in a bit-map screen with some selected color?" Those who've used our drawing program know that this capability is included in that program, so the source code for today's sidebar was readily available from that disk. The code in today's sidebar is just a fragment, so it can't be assembled or run as is.

We sent along a copy of the drawing program, complete with all its source code, to our reader. Here, though, we're going to lead you through the filling process in some detail.

For openers, the computer must be operating in its bit-map mode, and there must be an enclosed area drawn in pixels on the screen. The area can be of nearly any shape, but for purposes of discussion let's assume it's a square.

The first thing we have to do in this fragment is to determine the position of our drawing cursor. The cursor in this case is a sprite in the shape of a + sign. The sprite attribute table has been set to >3800, so as not to interfere with the bit-map screen.

Thus, to get the position coordinates for sprite #0 (our cursor), we set R0 to >3800. Reading the byte at >3800 gives us the y position of the sprite's upper left corner in dot-rows. We offset this number by 5 so that we've got the position of the center of the +. We read the next byte from VDP RAM to get the x position, then offset that by 3 to get to the center of the +. Thus we have in R8 the dot-row position of the center of our cursor and in R7 the dot-column position of the center of our cursor.

The first thing that the routine does after getting its position information is to start looking on the current row for a "filled" pixel. That is, it's examining the pattern table in VDP RAM, moving left until it finds a pixel that's "on." It does that by using the subroutine PLOTCK. That subroutine looks at the pixel pointed to by R8 and R7, and returns with status EQ if the pixel is on. If the pixel at R8, R7 coordinates is off, status will return with an NE indication.

Thus the instruction just following SKLFT will cause the routine to stop moving left as soon as it finds a turned on pixel. At that point (label NMSL) we save the dot-column position from R7 into R13, then move the original starting dot-column position back into R7 and start moving to the right. The loop starting at SKRT moves R7 to the right until a filled pixel is found, then exits its

Continued on page 24

Continued from page 23

loop to label NMSR. At that point, we save the right side position into R14.

The section of code just after NMSR sets R7 to the center of the current row of the enclosed area, then moves up one row. If that takes it beyond our picture area, we have reached the top of our enclosed figure. If the pixel at the center of the next upward row is on, that also indicates that we've reached the top of the enclosed area. In our drawing program, dot-rows from 0 through 7 are used to display information, and are not part of the picture area, hence the checking of R8 against 8, and the JLT GOLFT0. If the pixel we checked in the new row is not turned on, then we start over scanning this new row at label STLFT.

At label GOLFT0, we start scanning downward, moving left and right, and filling the pixels with whatever color is currently set for drawing. This section of code first finds the left boundary on each row, then turns on and colors the pixels moving right until it reaches the right boundary of that row of the figure. It also center-seeks on each row, thus insuring that when we move downward, we're in the center of the row just above.

This process continues until we've reached either the bottom of the enclosed area or the bottom of the picture area at row 176. When the whole area is filled, we jump to the code at FILLX, re-set conditions for the cursor, and then go back to scanning the keyboard and joystick for user input.

Along the way during our left-right scanning, we also check to see when we've reached the left and right boundaries of the picture area. Thus a "closed" area may actually be open on one side, bounded by the edge of the screen, and the fill process will still work correctly. Similarly, the closed area can be bounded by the top or bottom of the picture area.

In your own programming, you might want to use the whole screen for your picture area, instead of having our top and bottom limits. In such a case in the section following NMSR, after DEC R8, you'd simply omit the CI R8,8 line, so your "seek top" would continue through dot-row 0. In the section of code after NMRT, following the INC R8, you'd want to CI R8,191 instead of the CI R8,175 that's there. This would allow your filling to go all the way to the bottom row of the screen.

Nobody's Perfect

This method works very nicely for most shapes, but on occasion gets fooled. Circles, rectangles, and squares get filled perfectly. Triangles can be tricky, and sometimes they will wind up with a narrow corner unfilled. In the drawing program, we've provided a way to fill those tiny gaps by just putting the program into "pen down" condition and moving the cursor through the unfilled area.

Some Cautions

This method won't work if you try filling horizontally adjacent areas with

different colors. That's not because of the program per se, but because of the way the color bytes themselves are organized in VDP memory. One byte of color affects an area of eight pixels in the horizontal direction. Thus, when you fill another area that's within the eight pixel zone of an already filled area, the new color will bleed into the already filled area. Presumably, you could correct for this by using a background coloring in the overlapped area, but that's beyond the scope of our current discussion.

The Subroutines

You'll notice that we've shown the entire PLOT subroutine, but that the main routine doesn't use all of that subroutine. Instead, it uses only the part starting at label PLOTF. That's done because the necessary work of that first part of PLOT has already been performed by the PLOTCK routine, so we save some time by jumping in at PLOTF to actually put a pixel on the screen and color it.

These subroutines are not entirely of our own design, but were derived from material given us by John C. Johnson of Cedar Rapids, Iowa. We're not completely sure that the annotations in these are correct, but we know that the subroutines work as intended, and that's all that matters. We hope this stuff will be helpful to any of our readers who are struggling with bit-map mode.

Next issue, we're taking the plunge into the deep dark water of floating point math. We hope you'll be there to float along with us.

SIDEBAR 70

```
* SIDEBAR 70
* A FRAGMENT FOR FILLING A CLOSED AREA
* WITH COLOR IN BIT-MAP MODE
* Code by Bruce Harrison
* PUBLIC DOMAIN
* 9 MAR 1995
*
FILL    LI    R0,>3800      POINT AT SPRITE 0
        BLWP  @VSBR         READ Y-POSITION
        MOVB  R1,R8         PLACE IN R8
        MOV   R8,@OLDROW    STASH IN MEMORY
        SRL   R8,8          RIGHT JUSTIFY
        AI    R8,5          ADD 5 DOT ROWS
        INC   R0            NEXT VDP ADDRESS
        BLWP  @VSBR         READ X-POSITION
```

Continued on page 26

Continued from page 25

```
        MOVB R1,R7          MOVE TO R7
        MOV  R7,@OLDCOL     STASH IN MEMORY
        SRL  R7,8           RIGHT JUSTIFY
        AI   R7,3           ADD 3 DOT COLUMNS
STLFT   MOV  R7,@ENDOC      STASH R7
SKLFT   BL   @PLOTCK        CHECK PIXEL AT R8,R7 POSITION
        JEQ  NMSL           IF SET, JUMP
        DEC  R7             ELSE MOVE LEFT ONE DOT COLUMN
        JLT  NMSL           IF PAST LEFT EDGE, JUMP
        JMP  SKLFT          ELSE REPEAT MOVING LEFT
NMSL    MOV  R7,R13         STASH AWAY COL
        MOV  @ENDOC,R7      PUT ORIGINAL COLUMN BACK
SKRT    BL   @PLOTCK        CHECK PIXEL
        JEQ  NMSR           IF SET, JUMP
        INC  R7             MOVE RIGHT ONE COLUMN
        CI   R7,255         CHECK FOR RIGHT EDGE
        JGT  NMSR           IF GREATER, JUMP
        JMP  SKRT           ELSE CONTINUE MOVING RIGHT
NMSR    MOV  R7,R14         STASH AWAY R7
        S    R13,R14        GET DIFFERENCE IN COLUMNS
        CI   R14,2          CHECK FOR 2
        JLT  GOLFT0         IF LESS, JUMP
        SRL  R14,1          TAKE HALF OF RIGHT COL
        A    R14,R13        ADD TO LEFT COL
        MOV  R13,R7         PUT CENTER POSITION IN R7
        DEC  R8             MOVE UP ONE ROW
        CI   R8,8           COMPARE TO TOP OF PICTURE AREA
        JLT  GOLFT0         IF LESS, JUMP
        BL   @PLOTCK        CHECK PIXEL
        JNE  STLFT          IF NOT SET, JUMP
*
* WHEN WE REACH HERE, WE'VE FOUND THE TOP CENTER OF THE AREA
*
GOLFT0  INC  R8             DOWN ONE ROW
        MOV  @ENDOC,R7      GET OLD COLUMN
GOLFT   BL   @PLOTCK        CHECK PIXEL
        JEQ  GORT0          IF SET, JUMP
```



```
        DEC  R7             ELSE LEFT ONE COLUMN
        JLT  GORT0          IF <0, JUMP
        JMP  GOLFT          ELSE KEEP MOVING LEFT
GORT0   MOV  R7,R13         SAVE R7
GORT    INC  R7             RIGHT ONE COLUMN
        CI   R7,255         CHECK FOR EDGE
        JGT  NMRT           IF GREATER, JUMP
        BL   @PLOTCK        CHECK PIXEL
        JEQ  NMRT           IF SET, JUMP
        MOVB @LINCLR,R9     PUT LINE COLOR BYTE IN R9
        BL   @PLOTF         PLOT AND COLOR ONE PIXEL
        JMP  GORT           THEN REPEAT
NMRT    MOV  R7,R14         STASH R7
        S    R13,R14        TAKE DIFFERENCE
        CI   R14,2          CHECK FOR 2
        JLT  FILLX          IF LESS, EXIT
        SRL  R14,1          HALVE R14
        A    R14,R13        ADD TO R13
        MOV  R13,R7         CENTER COLUMN
        INC  R8             NEXT ROW
        CI   R8,175         CHECK BOTTOM OF PICTURE AREA
        JGT  FILLX          IF GREATER, JUMP
        BL   @PLOTCK        ELSE CHECK PIXEL
        JNE  GOLFT          IF NOT SET, REPEAT FOR THIS ROW
FILLX   CLR  R4             CLEAR REG 4
        CLR  R10            CLEAR REG 10
        MOV  @OLDCOL,R7     OLD COLUMN BACK
        MOV  @OLDROW,R8     OLD ROW BACK
        BL   @NODRW         RESET THE SPRITE
        B    @KJSCAN        BACK TO SCANNING MODE
*
* SUBROUTINES
*
*
* FOLLOWING WRITES ONE PIXEL TO SCREEN AT LOCATION POINTED
BY
* R8 (DOT ROW) AND R7 (DOT COLUMN)
```

Continued on page 28

Continued from page 27

```
*
PLOT    MOV  R7,R3          MOVE DOT COLUMN TO R3
        MOV  R8,R4          AND DOT ROW TO R4
        MOV  R4,R5          DOT ROW ALSO IN R5
        ANDI R5,7           R5 HAS DOT ROW MODULO 8
        SZC  R5,R4          SO DOES R4
        SLA  R4,5           MULTIPLY R4 BY 32
        A    R5,R4          ADD R5, SO R4 HAS DR MOD. 8 * 32 +
DR MOD 8
        MOV  R3,R0          MOVE DOT COL TO R0
        ANDI R0,>FFF8       R0 HAS DC - DC MOD 8
        S    R0,R3          R3 HAS DC MOD 8
        A    R4,R0          ADD R4
        SWPB R0             SWAP BYTES
        MOVB R0,@>8C02      WRITE LOW ADDRESS BYTE
        SWPB R0             SWAP
        MOVB R0,@>8C02      WRITE HIGH ADDRESS BYTE
        NOP                 WASTE TIME
        MOVB @>8800,R1      READ THE BYTE
PLOTF   SOCB @M(R3),R1      OVERLAY MASK FROM TABLE M
PLOTF0  ORI  R0,>4000       SET THE 4000 BIT IN R0
        SWPB R0             SWAP
        MOVB R0,@>8C02      WRITE LOW BYTE OF ADDRESS
        SWPB R0             SWAP
        MOVB R0,@>8C02      WRITE HIGH BYTE OF ADDRESS
        NOP                 WASTE TIME
        MOVB R1,@>8C00      WRITE MODIFIED BYTE BACK TO VDP
        MOV  R9,R9          IS COLOR TO BE SET?
        JEQ  PLOTX          IF NOT, JUMP AHEAD
        ANDI R0,>3FFF       STRIP OFF "4" FROM R0
        AI   R0,>2000       ADD >2000 TO POINT AT COLOR TABLE
ENTRY
        BLWP @VSBR          READ THAT BYTE INTO R1
        MOVB R1,R2          MOVE THE BYTE TO R2
        ANDI R2,>F000       STRIP ALL BUT LEFT NYBBLE
        CB   R2,R9          COMPARE TO LEFT BYTE R9
        JEQ  PLOTX          IF EQUAL, COLOR ALREADY SET
```



```
        ANDI R1,>0F00       ELSE STRIP OFF LEFT NYBBLE R1
        AB   R9,R1          REPLACE WITH LEFT NYBBLE R9
        BLWP @VSBW          THEN WRITE COLOR BYTE BACK
PLOTX   RT                  RETURN
PLOTCK  MOV  R7,R3          MOVE DOT COLUMN TO R3
        MOV  R8,R4          AND DOT ROW TO R4
        MOV  R4,R5          DOT ROW ALSO IN R5
        ANDI R5,7           R5 HAS DOT ROW MODULO 8
        SZC  R5,R4          SO DOES R4
        SLA  R4,5           MULTIPLY R4 BY 32
        A    R5,R4          ADD R5, SO R4 HAS DR MOD. 8 * 32 +
DR MOD 8
        MOV  R3,R0          MOVE DOT COL TO R0
        ANDI R0,>FFF8       R0 HAS DC - DC MOD 8
        S    R0,R3          R3 HAS DC MOD 8
        A    R4,R0          ADD R4
        SWPB R0             SWAP BYTES
        MOVB R0,@>8C02      WRITE LOW ADDRESS BYTE
        SWPB R0             SWAP
        MOVB R0,@>8C02      WRITE HIGH ADDRESS BYTE
        CLR  R1             CLEAR REGISTER 1
        CLR  R2             CLEAR REGISTER 2
        MOVB @>8800,R1      READ THE BYTE
        MOVB @M(R3),R2      GET MASK BIT INTO R2
        COC  R2,R1          SEE IF R1 HAS A ONE AT MASK BIT
        RT
*
* DATA SECTION
*
M       DATA >8040,>2010,>0804,>0201  MASK DATA
LINCLR  BYTE >10            LINE DRAWING COLOR
OLDROW  DATA 0              OLD DOT-ROW
OLDCOL  DATA 0              OLD DOT-COLUMN
ENDOC   DATA 0              STORAGE WORD
```

GENEVE

Program displays ASCII values onscreen

BY MARTIN ZEDDIES

With ASC_DISPL you can display all characters of the actual loaded character set under MDOS. (It has been tested under MDOS 5.0.) With this table it is easy to calculate the ASCII values of the characters you like to see on the monitor. That make it much easier to display frames or easy graphics on the text-monitor screen, such as a batch-start menu for your system.

The program displays a table of ASCII characters based on keystrokes you input. If you have a complete ASCII key table in your hand, it is easy to get the correct keypress-combination to display the character you want to get.

There are a lot of these key tables in the the TI and Myarc computer world but no one will give you an information about the display which appears when you press the different key-combinations. My program gives you the 'char-looking-like' information you need to create the graphic you want.

This software is written on the Geneve. The code may run on a TI 99/4A with 80-column card if you compile it with the correct header-files for your card. I think it is worth trying to run it on your TI system.

ASC_ENG_C

```
/* Sorry folks !                                                        */
/* Althrough this is the international version of my MDOS ASC-display pro- */
/* gram the following comments are all in German because that is my native */
/* speech. But if you have ideas or question feel free to contact me in    */
/* german or englich language. You will find my adress 8 lines down in     */
/* this C-Source-Code !                                                 */
/* ---------------------------------------*/
/* Dieses Programm erstellt eine ASCII-Zeichensatztabelle ueber den gerade */
/* benutzen ASC-Zeichensatz im MDOS 5.00                                */
/* Teststatus: MDOS 5.00 / C99MDOS-Compiler CP                          */
/* Version #: 19970915 - ZE                                             */
/* Bei Fragen oder Anregungen :                                        */
/* Martin Zeddies, Hauptstrasse 26,D-38446 Wolfsburg-Reislingen, Germany  */
/* Phone&Fax: +49-5363-71125                                            */
#include "DSK5.STDIO_H";              /* wird von LOCATE benoetigt */
#include "DSK5.VIDEO_H";              /* weil HCHAR benutzt wird */
#define soff 20                       /* Zeichenoffset zum Zeilenbegin */
#define dver "V#: 19970906-ZE"        /* Achtung diese define-Zuweisung arbeitet
                                      es wird weder ein '=' noch das ';' am
```


GENEVE

```
                                      Zeilenende benutzt */
int z=3,s=5,stab=3,i,j,a,c,zeichen;
main()
{
 c=vidmode(1);                        /*TEXT90 26Zeilen * 80 Zeichen */
 locate(1,4);
 puts("\fZE-software still produce software for the MYARC GENEVE 9640");
 locate(24,1);
 puts("ASCII-Chart of the active under MDOS loaded characterset      ");
 puts(dver);
 locate(z,soff-10+s);
 for (i=0; i<16; i++)
 {
  locate(z,i*stab+soff);
  putchar(hexdigit(i));
 }
 zeichen=0;                    /* mit diesem Zeichen beginnen */
 for (i=0; i<16; i++)
 {                                    /* Zeilenschleife */
  locate(i+5,7);
  putchar(hexdigit(i));
  for(j=zeichen; j<(zeichen+16); j++)
  {                                   /* Spaltenschleife */
   locate(i+5,(j-zeichen)*stab+soff);
   if ((j<32) || ((j>128) && (j<160)))
    a=64;
   else
    a=j;
   vchar(i+5,(j-zeichen)*stab+soff,j,1);
  }                                   /* Spaltenschleife */
  zeichen=j;
 }                                    /* Zeilenschleife */
locate(26,1);                         /* Am Programmende Cursor unten links */
}
/* Hier beginnen die lokalen Unterfunktionen */
hexdigit(dezi)
int dezi;
{
 if( dezi<10)
  dezi=dezi+48;
 else
  dezi=dezi+55;
return dezi;
}
```

EXTENDED BASIC

Program creates keyboard overlays for any program

Overlayer, by Mike DeFrank, is an Extended BASIC program that prints keyboard overlays for a number of popular programs, including TI-Writer, Multiplan, and Terminal Emulator II. Users may also create overlays for other programs. The program is designed to output the overlays to a Gemini 10-X printer, so you may have to modify the printer commands for your printer.

The program is menu-driven and easy to use.

OVERLAYER

```
10 ! OVERLAYER (8507.20) !04
9
20 ! BY MIKE DEFRANK !159
30 ! 4374 NW 9TH AVE
     POMPANO BEACH, FL 33064
 !086
40 ! (305)-946-2724 !088
50 ! OVERLAY STRIP GENERATOR
 FOR THE GEMINI 10-X PRINTER
 !105
60 DEVICE$="PIO" :: DIM CTRL
$(12),FCTN$(12)!223
70 GOSUB 200 ! SETUP SCREEN
!044
80 !MAIN CONTROL LOOP !067
85 DISPLAY AT(10,4):"
                  ":"     1) D
ESIGN OVERLAY    ":"
                  ":"
                   " !045
90 DISPLAY AT(14,4):"  2) SE
LECT FROM LIST   ":"
                  ":"
                   " !216
95 DISPLAY AT(17,4):"  3) EX
IT PROGRAM       ":"
                  ":"
                   " !100
100 CALL SOUND(-5,1800,1)::
DISPLAY AT(22,2):"CHOICE > "
 !032
102 CALL KEY(0,K,S):: IF S=0
 THEN 102 :: IF 51<K>49 THEN
 102 !192
104 S=K-48 :: DISPLAY AT(22,
10):CHR$(K)!036
110 IF S=3 THEN CALL CLEAR :
: END ELSE CALL HCHAR(10,1,3
2,480)!177
120 IF S=2 THEN 140 !149
130 GOSUB 300 :: GOTO 150 !
DESIGN !065
140 GOSUB 4000 ! SELECT !098
150 GOSUB 600 ! SET PRINTER
!125
160 POINTER=1 :: GOSUB 700 !
PRINT 1ST HALF OF OVERLAY !1
63
170 POINTER=7 :: GOSUB 700 !
PRINT 2ND HALF OF OVERLAY !1
49
180 GOSUB 1000 ! RESET PRINT
ER !165
190 GOTO 80 !159
```


EXTENDED BASIC

OOPS!	REFORMAT	SCREEN COLOR	NEXT PARAGRH	DUPE LINE	LAST PARAGRH
DELETE CHAR	INSERT CHAR	DELETE LINE	ROLL DOWN Ñ	NEXT WINDOWº	ROLL UP ñ

WORD TAB	NEW PARAGRH	NEW PAGE	WORD WRAP		TI
TAB	INSERT LINE	COMMAND/ESC	LINE NUMBERS	QUIT	WRITER

Sample printout from Overlayer.

```
200 !SECTION A: SET SCREEN !
031
210 CALL DELSPRITE(ALL):: CA
LL CLEAR :: CALL SCREEN(15):
: CALL COLOR(13,5,1):: CALL
MAGNIFY(3)!214
220 CALL CHAR(91,"00102844EE
282838",93,"00382828EE442810
",123,"0010305E825E3010",125
,"001018F482F41810")! SPECIA
L GRAPHICS !200
230 A$=RPT$("08",7):: CALL C
HAR(128,"0F"&A$,129,"FF",130
,"FF"&A$,131,"F8080808080808
08",132,"0F",133,"F8",134,"0
8"&A$)! OVERLAY GRAPHICS !23
8
240 CALL CHAR(136,"00006E848
4640000"&"0000E688C8860000"&
"0000E8A8C8AE0000"&"0000EA4E
4E4A0000")! SPRITE WORD GRAP
HICS !076
250 CALL CHAR(140,"00FFFFFFF
FFFFFFF"&RPT$("00",8)&"00FEF
EFEFEFEFEFE"&RPT$("00",8))!
SPRITE BOX GRAPHICS !136
260 DISPLAY AT(2,4):"OVERLAY
 STRIP DESIGNER" !129
270 DISPLAY AT(4,2):"
                     ":"
                      ":"
                       ":"
1 2 3 4 5 6 7 8 9 0 =" !186
280 CALL SPRITE(#1,136,5,25,
211)! SPRITE WORDS !142
290 RETURN !136
300 !SECTION B: DESIGN !014
310 CALL COLOR(0,8,5):: R,R1
=25 :: R2=33 :: A,CNT=0 :: R
ESTORE 450 !085
320 DISPLAY AT(10,2):"
                    ":"   SPE
CIAL GRAPHICS KEYS  ":"
                     ":"
                      "
!202
324 DISPLAY AT(14,2):"  FCTN
 R: [   FCTN T: ]   ":"
                     ":"
```

Continued on page 34

EXTENDED BASIC

Continued from page 33

```
                        " !132
326 DISPLAY AT(17,2):"  FCTN
 F: {    FCTN G: }  ":"
                       ":"
                        " !179
330 DISPLAY AT(23,12):"
       ":" TEXT WILL BE AUTO
-CENTERED" !109
340 CNT=CNT+1 :: IF CNT=12 T
HEN R=R2 :: RESTORE 450 :: C
NT=1 !030
350 CALL SPRITE(#2,140,CNT+2
,R,14+(CNT*16)):: READ B$ ::
 IF R=R1 THEN A$="(CTRL" ELS
E A$="(FCTN" !211
360 CALL SOUND(-5,1800,1)::
DISPLAY AT(22,2):A$&" "&B$&"
):" :: ACCEPT AT(22,12)SIZE(
12):S$ !112
370 BOXSIZE=12 :: GOSUB 500
:: IF R=R1 THEN CTRL$(CNT)=S
$ ELSE FCTN$(CNT)=S$ !086
380 IF R=R2 AND CNT=11 THEN
DISPLAY AT(23,8):"          "
 :: GOTO 390 ELSE 340 !057
390 A=A+1 :: IF A=1 THEN A$=
"CTRL" ELSE A$="FCTN" !103
400 CALL SPRITE(#2,140,16,17
+(A*8),211)!200
410 CALL SOUND(-5,1800,1)::
DISPLAY AT(22,2):"TEXT: "&A$
 :: ACCEPT AT(22,08)SIZE(-09
):S$ !191
420 BOXSIZE=09 :: GOSUB 500
!220
430 IF A=1 THEN CTRL$(12)=S$
 :: GOTO 390 !173
440 FCTN$(12)=S$ :: CALL HCH
AR(10,1,32,480):: CALL DELSP
RITE(#2):: RETURN !188
450 DATA 1,2,3,4,5,6,7,8,9,0
,=,XXX !004
500 !SECTION C: FORMAT TEXT
STRINGS !205
510 N=POS(S$,"[",1):: IF N T
HEN S$=SEG$(S$,1,N-1)&CHR$(1
64)&SEG$(S$,N+1,LEN(S$)-N)::
 GOTO 500 !126
520 N=POS(S$,"]",1):: IF N T
HEN S$=SEG$(S$,1,N-1)&CHR$(1
65)&SEG$(S$,N+1,LEN(S$)-N)::
 GOTO 500 !129
530 N=POS(S$,"{",1):: IF N T
HEN S$=SEG$(S$,1,N-1)&CHR$(1
66)&SEG$(S$,N+1,LEN(S$)-N)::
 GOTO 500 !160
540 N=POS(S$,"}",1):: IF N T
HEN S$=SEG$(S$,1,N-1)&CHR$(1
67)&SEG$(S$,N+1,LEN(S$)-N)::
 GOTO 500 !163
580 N=INT((BOXSIZE-LEN(S$))/
2):: S$=RPT$(" ",N)&S$&RPT$(
" ",BOXSIZE-(LEN(S$)+N))&CHR
$(245)!077
590 RETURN !136
600 !SECTION D: SET PRINTER
!134
610 CALL COLOR(0,5,1)!173
620 CALL SOUND(-5,1800,1)::
DISPLAY AT(22,2):"PRINTER DE
VICE NAME:" :" >"&DEVICE$ !21
7
630 ON ERROR 900 :: OPEN #1:
DEVICE$,VARIABLE 132 !056
640 PRINT #1:CHR$(27)&CHR$(6
```


EXTENDED BASIC

```
4):: ON ERROR STOP !071
660 PRINT #1:CHR$(27)&CHR$(6
5)&CHR$(6)! SET LINEFEED SPA
CING TO 6/72 !169
670 PRINT #1:CHR$(15):: ! PR
INT #1:CHR$(27);"4" !137
680 PRINT #1:CHR$(27)&CHR$(7
1)!235
690 RETURN !136
700 !SECTION E: PRINT HALF O
F OVERLAY !022
710 DISPLAY AT(19,2):" STATU
S: PRINTING OVERLAY" !129
720 PRINT #1:RPT$(CHR$(10),3
)! ADVANCE PAPER 3 LINES !20
6
730 ! PRINT TOP ROW !091
740 IF POINTER=7 THEN A=9 EL
SE A=12 !193
745 PRINT #1:RPT$(CHR$(95),8
0)!246
750 PRINT #1:CHR$(43)&RPT$(R
PT$(CHR$(45),12)&CHR$(43),5)
&RPT$(CHR$(45),A)&CHR$(43)!2
23
760 ! PRINT CTRL ROW !157
770 PRINT #1:CHR$(124);:: FO
R B=POINTER TO POINTER+5 ::
PRINT #1:CHR$(27);"4";CTRL$(
B);" ";CHR$(27);"5";CHR$(124
);:: NEXT B
780 ! PRINT DIVIDER LINE !15
9
790 IF POINTER=7 THEN 810 !0
09
800 PRINT #1:CHR$(43)&RPT$(R
PT$(CHR$(45),12)&CHR$(43),5)
&RPT$(CHR$(45),12)&CHR$(43):
 GOTO 820 !211
810 PRINT #1:CHR$(43)&RPT$(R
PT$(CHR$(45),12)&CHR$(43),4)
&RPT$(CHR$(45),12)&CHR$(43)&
RPT$(CHR$(32),09)&CHR$(43)!0
84
820 ! PRINT FCTN ROW !147
830 PRINT #1:CHR$(124);:: FO
R B=POINTER TO POINTER+5 ::
PRINT #1:CHR$(27);"4";FCTN$(
B);" ";CHR$(27);"5";CHR$(124
);:: NEXT B !064
835 PRINT #1:CHR$(124)&RPT$(
RPT$(CHR$(95),12)&CHR$(124),
5)&RPT$(CHR$(95),A)&CHR$(124
)!124
840 !PRINT BOTTOM OF OVERLAY
 !252
855 PRINT #1:CHR$(43)&RPT$(R
PT$(CHR$(45),12)&CHR$(43),5)
&RPT$(CHR$(45),A)&CHR$(43)!2
23
860 RETURN !136
900 !SECTION F: ERROR MSG !2
33
910 DISPLAY AT(19,2)BEEP:"DE
VICE ERROR - PRESS ENTER" !0
87
920 CALL KEY(0,K,S):: IF S=0
 THEN 920 !223
930 IF K<>13 THEN 920 !144
940 DISPLAY AT(19,1):"" :: R
ETURN 620 !157
1000 !SECTION G: RESET PRINT
ER !032
1010 PRINT #1:CHR$(27)&CHR$(
64)!237
```

Continued on page 36

EXTENDED BASIC

Continued from page 35

```
1020 CALL HCHAR(19,1,32,192)
!026
1030 CLOSE #1 :: RETURN !161
4000 !SECTION H: SELECT !026
4005 ON WARNING NEXT !215
4010 DISPLAY AT(10,1):"
                          01 T
I WRITER  06 TE 1200      02 M
UTIPLAN   07 TE-II        03 T
I FORTH   08 FASTTERM     " !17
6
4020 DISPLAY AT(14,1):" 04 T
I ARTIST  09 BASIC        05 G
RAPHX     10 ED/AS
                          " !06
4
4030 DISPLAY AT(17,1):"
                          " !15
3
4040 CALL SOUND(-5,1800,1)::
 DISPLAY AT(22,2):"CHOICE:"
:: ACCEPT AT(22,10)VALIDATE(
DIGIT)SIZE(2):S :: IF S<1 OR
 S>10 THEN 4040 !114
4050 ON S GOTO 4060,4070,408
0,4090,4100,4110,4120,4130,4
140,4150 !143
4060 RESTORE 5000 :: GOTO 42
00 !065
4070 RESTORE 5010 :: GOTO 42
00 !075
4080 RESTORE 5020 :: GOTO 42
00 !085
4090 RESTORE 5030 :: GOTO 42
00 !095
4100 RESTORE 5040 :: GOTO 42
00 !105
4110 RESTORE 5050 :: GOTO 42
00 !115
4120 RESTORE 5060 :: GOTO 42
00 !125
4130 RESTORE 5070 :: GOTO 42
00 !135
4140 RESTORE 5080 :: GOTO 42
00 !145
4150 RESTORE 5090 :: GOTO 42
00 !155
4200 DISPLAY AT(19,2):"STATU
S: GENERATING OVERLAY" !219
4210 FOR A=1 TO 2 :: FOR B=1
 TO 12 !023
4220 CALL SOUND(-5,1800,1)::
 CALL SPRITE(#2,140,B+2,(A*8
)+17,14+(B*16))!096
4230 READ S$ :: IF B=12 THEN
 BOXSIZE=09 ELSE BOXSIZE=12
!150
4240 GOSUB 500 :: IF A=1 THE
N CTRL$(B)=S$ ELSE FCTN$(B)=
S$ !229
4250 NEXT B :: NEXT A !049
4260 CALL DELSPRITE(#2):: CA
LL HCHAR(10,1,32,480):: RETU
RN !028
5000 DATA OOPS!,REFORMAT,SCR
EEN COLOR,NEXT PARAGRH,DUPE
LINE,LAST PARAGRH,WORD TAB,N
EW PARAGRH,NEW PAGE,WORD WRA
P,,TI !044
5005 DATA DELETE CHAR,INSERT
 CHAR,DELETE LINE,ROLL DOWN
],NEXT WINDOW},ROLL UP [,TAB
,INSERT LINE,COMMAND/ESC,LIN
E NUMBERS,QUIT,WRITER !207
5010 DATA HOME,TAB,NXT UNL C
```


EXTENDED BASIC

```
ELL,FORWARD CHAR,FORWARD WOR
D,CHNGE WINDOW,REL/ABS REF,,
,,CANCEL,MULTI !025
5015 DATA LOWER RIGHT,,,BACK
 CHAR,BACK WORD,,HELP,RECALL
,BACKSPACE,DEL FORWARD,,PLAN
 !052
5020 DATA ,,,,,,,INSERT LINE
,,,,TI !155
5025 DATA DELETE CHAR,INSERT
 CHAR,ERASE LINE,FORWARD SCR
,TAB },BACK SCR,BUFFER IN,BU
FFER OUT,ESCAPE,,QUIT,FORTH
!086
5030 DATA A}CLEAR I.,B}CLEAR
 C.,C}INPUT D.,D}DRAW,E}ALPH
A N.,F}FILL,H}H OR V,I}INVER
T,K}K-LINE,L}LINE,M}MIRROR,T
I !111
5035 DATA N}SWAP,O}CIRCLE,P}
POINT,Q}DISC,R}RAYS,S}STORE,
V}FRAME,X}BOX,Z}ZOOM,FCTN .}
PEN,FCTN ;}SPEED,ARTIST !109
5040 DATA ,,,,,,,,,,,GRAPHX
!220
5045 DATA SLOWER,FASTER,DRAW
 TOGGLE,ERASE TOGGLE,NO HELP
,ZOOM TOGGLE,COLOR MENU,LINE
 MODE,CIRCLE MODE,COPY MENU,
MAIN MENU,!147
5050 DATA RESTART,,CANCEL,LO
G ON,LOG OFF,TRANSFER,,,,,T
E 1200 !069
5055 DATA ,,,,,,,,,QUIT,!08
3
5060 DATA SPEAK,OUTPUT,CANCE
L,TRANSFER,WRAP TOGGLE,CASE
TOGGLE,PAGE,,,EXIT,,TE-II !1
93
5065 DATA ^G}BELL,^H}BACKSPA
CE,^J}LINEFEED,^L}FORM FEED,
^.}ESCAPE,^/}BREAK,^Q}X-ON,^
S}X-OFF,,FCN V}DELETE,QUIT,!
004
5070 DATA MODEM BAUD,SPOOL T
OGGLE,MODEM PARITY,MODEM POR
T,PRNTR PARITY,PRNTR PORT,PR
NTR BAUD,,,40/80 TOGGLE,QUIT
,FAST !125
5075 DATA DELETE,^T,^N,BREAK
,WINDOW },^E,FORE COLOR,BACK
 COLOR,,PAGE TOGGLE,QUIT,TER
M !159
5080 DATA ,,,,,,,,,,BASIC !
115
5085 DATA DELETE,INSERT,ERAS
E,BREAK,BEGIN,PROCEED,AID,RE
DO,BACK,,QUIT,!085
5090 DATA DELETE CHAR,INSERT
 CHAR,ERASE LINE,ROLL UP[,NE
XT WINDOW},ROLL DOWN],TAB,IN
SERT LINE,ESCAPE,,QUIT,EDITO
R !158
5095 DATA DELETE,INSERT,ERAS
E,BREAK,BEGIN,PROCEED,AID,RE
DO,BACK,,QUIT,ASSEMBLER !196
```

PUNN sponsors Web site

The Portland Users of Ninety-Nines have a web site. Address is www.rdrop.com/users/tedpet/.

c99 code listed

BY VERN JENSEN

Below are the files snake;l and snake;c. These are companion files to last issue's Beginning c99 column. Refer to that article for more information about these programs.—Ed.

Another typo of mine in the Jan/Feb edition was just pointed out to me. In the BITTEST/C source code listing, the for loop of the PutBinary() function should read:

```
for (n=0; n <= 15; n++)
```

Originally, it appeared as below:

```
for (n=0; n < 15; n++)
```

snake;l

```
DSK2.SNAKE;O
DSK2.CSUP
DSK2.GRF1
DSK2.SOUND
```

snake;c

```
#include "DSK2.GRF1;H"
#include "DSK2.RANDOM;C"
#include "DSK2.SOUND;H"

#define kLives  3   /* Starting number of lives   */
#define kSpeed  12  /* Starting snake speed       */
#define kApples 10  /* Number of apples per level */
#define kDelay  8   /* Delay before tail moves after eating an apple */

#define cBody   128 /* Char for the snake's body  */
#define cHead   129 /* Char for the snake's head  */
#define cWall   136 /* Char for the wall          */
#define cApple  144 /* Char for the apple         */
#define cLeaf   145 /* Char for the apple leaf    */

#define kMaxLen 100 /* Maximum length of the snake */
#define FALSE   0
#define TRUE    1


  /* Variables to keep track of the snake's position */
int   hDelta, vDelta;
char  headRow, headCol;
char  array[kMaxLen][2];
char  tailDelay;
```



```
char  length;

char  done;  /* TRUE when stats need to be displayed. */
char  dead;  /* TRUE if snake died */

char  speed;
char  level;
char  lives;
char  apples;

char  g,row,col,ticks;
int   n,x,y,keyCode,status;


main()
{
  Grf1();
  Randomize();
  Screen(2);
  InitChar();

  while (1)
  {
    level = 1;
    lives = 3;
    dead  = FALSE;

    TitleScreen();

    while (!dead)
    {
      DoStats();
      Play();
    }

    Clear();
    SpDel(0);
    Display(12,12,"Game Over");
    Wait(120);
  }
}


InitChar()
{
    /* Color text to white */
  for (n=0; n < 32; n++)
```

Continued on page 40

BEGINNING C99 PART 8

Continued from page 39

```
    Color(n,16,1);

  Color(cBody/8,3,1);
  ChrDef(cBody,"FFFFFFFFFFFFFFFF");

  Color(cWall/8,12,1);
  ChrDef(cWall,"FFFFFFFFFFFFFFFF");

  Color(cApple/8,9,1);
  ChrDef(cApple,"66FFFFFFFFF7E24");
  ChrDef(cLeaf,"3060C0");

  ChrDef(cHead,
    "183C5A7E3C3C7EFF80CCFAFFFFFACC80FF7E3C3C7E5A3C1801335FFFFF5F3301");
}


TitleScreen()
{
  Clear();
  SpDel(0);

  Display(3,5,"S N E A K Y   S N A K E");
  Display(8,3,"Use the arrow keys to move");
  Display(9,3,"the snake into the apples.");
  Display(10,3,"Collect ten apples to");
  Display(11,3,"advance to the next level.");
  Display(12,3,"Each time you eat an apple,");
  Display(13,3,"the snake grows longer and");
  Display(14,3,"moves faster.");
  Display(16,3,"Good luck!");
  Display(24,9,"-PRESS ANY KEY-");

  do
  {
    keyCode = Key(0,&status);
  } while (status != 1);
}


DoStats()
{
  Clear();
  SpDel(0);

  Display(8,6,"Level");
  HChar(8,12,'0' + level,1);
```


BEGINNING C99 PART 8

```
  Display(8,19,"Lives");
  HChar(8,25,'0' + lives,1);

  Wait(60);
  Display(14,11,"Get Ready!");
  Wait(120);

  SetUp();
}


SetUp()
{
  Clear();

  done = FALSE;
  apples  = kApples;
  headRow = 23;
  headCol = 16;
  array[0][0] = headRow;
  array[0][1] = headCol;

  hDelta  = 0;
  vDelta  = -1;
  speed   = kSpeed;
  length  = 0;
  tailDelay = 4;


    /* Draw Wall */
  HChar(1,1,cWall,32);
  HChar(24,1,cWall,32);
  VChar(1,1,cWall,24);
  VChar(1,32,cWall,24);

  DrawLevel();
  PutApple();
}


DrawLevel()
{
  char curLevel;

  curLevel = level;
  while (curLevel > 7)
    curLevel = curLevel - 7;
```

Continued on page 42

Continued from page 41

```
switch (curLevel)
{
  case 2:
    HChar(6,9,cWall,15);
    VChar(6,9,cWall,12);
    VChar(6,23,cWall,12);
    break;
  case 3:
    HChar(11,1,cWall,32);
    HChar(11,15,32,3);
    break;
  case 4:
    HChar(6,7,cWall,18);
    HChar(14,7,cWall,18);
    break;
  case 5:
    HChar(11,3,cWall,27);
    break;
  case 6:
    VChar(6,7,cWall,10);
    VChar(6,15,cWall,10);
    VChar(6,23,cWall,10);
    break;
  case 7:
    HChar(6,7,cWall,18);
    HChar(15,7,cWall,18);
    VChar(6,16,cWall,9);
    break;
  default:
    break;
  }
}


Play()
{
  while (!done)
  {
    GetTime(); /* Clear timer */
    ticks = 0;

      /* Move all values in array down one element (slow, but easy) */
    for (n=length; n > 0; n-)
    {
      array[n][0] = array[n-1][0];
      array[n][1] = array[n-1][1];
    }
```



```
    /* Put snake's new position at top of array */
  array[0][0] = headRow;
  array[0][1] = headCol;

    /* Erase tail if it is moving */
  if (tailDelay > 0)
  {
    tailDelay-;
    length++;
  }
  else
  {
    row = array[length][0];
    col = array[length][1];
    HChar(row,col,32,1);
  }


  MoveSnake();

    /* Wait until it is time to move the snake again */
  if (!done)
  {
    do
    {
      ticks = ticks + GetTime();
    } while (ticks < speed);
  }
 }
}


MoveSnake()
{
  status = Joyst(1,&x,&y);
  if (!status)
    status = Joyst(2,&x,&y);

  if (status)
  {
      /* Don't do anything if the joystick position is diagonal. */
    if (x == 0 || y == 0)
    {
      if (x != 0)
      {
        hDelta = x/4;
        vDelta = 0;
```

Continued on page 44

BEGINNING C99 PART 8

Continued from page 43

```
    }
    else
    {
      hDelta = 0;
      vDelta = -y/4;
    }
  }
}
else  /* Read keys if no joystick value is read. */
{
  keyCode = Key(0,&status);
  if (keyCode >= 'a')
    keyCode = keyCode - 32;

  if (keyCode == '9')
  {
    done = TRUE;
    dead = TRUE;
  }
  if (keyCode == 'E')
  {
    hDelta = 0;
    vDelta = -1;
  }
  else if (keyCode == 'X')
  {
    hDelta = 0;
    vDelta = 1;
  }
  else if (keyCode == 'S')
  {
    hDelta = -1;
    vDelta = 0;
  }
  else if (keyCode == 'D')
  {
    hDelta = 1;
    vDelta = 0;
  }
}


headRow = headRow + vDelta;
headCol = headCol + hDelta;

g = GChar(headRow, headCol);
```


BEGINNING C99 PART 8

```
  /* Erase old head with body */
HChar(array[0][0],array[0][1],cBody,1);

  /* Draw the snake's head */
if (vDelta == -1)
  HChar(headRow, headCol, cHead, 1);
else if (hDelta == 1)
  HChar(headRow, headCol, cHead+1, 1);
else if (vDelta == 1)
  HChar(headRow, headCol, cHead+2, 1);
else
  HChar(headRow, headCol, cHead+3, 1);


if (g == cApple)      /* Snake hit apple */
{
  apples-;
  speed-;
  tailDelay = kDelay;
  SpDel(0); /* Delete leaf sprite */

  if (apples == 0)
  {
    Sound1(2,262,2);
    Sound1(2,330,2);
    Sound1(2,392,2);
    Sound1(2,523,2);
    Sound1(2,392,2);
    Sound1(2,330,2);
    Sound1(2,262,2);

      /* Create exit holes */
    HChar(1,16,32,1);
    HChar(24,16,32,1);
    VChar(12,1,32,1);
    VChar(12,32,32,1);
  }
  else
  {
    Sound1(5,587,2);
    Sound1(5,684,2);

/*  Sound2(5,392,2,5,587,2);
    Sound2(5,494,2,5,784,2);
*/
    Wait(30);
    PutApple();
  }
```

Continued on page 46

Continued from page 45

```
  }
  else if (g != 32)      /* Snake hit itself or the wall */
  {
    for (n=0; n <= 15; n++)
      SoundN(5,6,n);

    lives-;
    done = TRUE;

    if (lives == 0)      /* Game Over */
      dead = TRUE;
  }
  else if (headRow == 1 || headRow == 24 ||   /* Snake went out exit hole.
*/
           headCol == 1 || headCol == 32)
  {
    level++;
    lives++;
    done = TRUE;
    ExitLevel();
  }
  else  /* Snake hit nothing */
  {
      /* Play the "move" sound */
    Sound1(2,392,2);
  }
}


ExitLevel()  /* Move snake out the exit */
{
    /* Move all values in array down one element */
  for (n=length; n > 0; n-)
  {
    array[n][0] = array[n-1][0];
    array[n][1] = array[n-1][1];
  }

    /* Put snake's new position at top of array */
  array[0][0] = headRow;
  array[0][1] = headCol;

    /* Draw the snake's head */
  HChar(headRow, headCol, cBody, 1);


    /* Erase the tail */
  for (n=length; n >= 0; n-)
```



```
  {
    GetTime(); /* Clear timer */
    ticks = 0;

      /* Erase the tail */
    row = array[n][0];
    col = array[n][1];
    HChar(row,col,32,1);

    do  /* Wait until it is time to move again */
    {
      ticks = ticks + GetTime();
    } while (ticks < 2);
  }
}


PutApple()  /* Place an apple randomly on the board */
{
  do  /* Find a random spot that is empty */
  {
    row = GetRnd(2,23);
    col = GetRnd(2,31);
    g = GChar(row,col);
  } while (g != 32);

  HChar(row,col,cApple,1);

    /* Add green leaf sprite */
  Sprite(0,cLeaf,13,row*8-9,col*8-5);
}


GetRnd(lowNum, highNum)  /* Returns a number between lowNum and highNum */
char lowNum, highNum;
{
  return Rnd(highNum-lowNum)+lowNum;
}


Display(myRow, myCol, myString)
char myRow, myCol, *myString;
{
  Locate(myRow, myCol);
  PutS(myString);
}
```

Continued on page 48

BEGINNING C99 PART 8

Continued from page 47

```
Wait(theDelay)
int theDelay;
{
  GetTime();    /* Clear timer */

  while (theDelay > 0)
  {
    theDelay = theDelay - GetTime();
  }
}


GetTime()
{
#asm
        LIMI 2                  ALLOW INTERRUPTS
        LIMI 0                  STOP THEM
        MOV  @>8378,8           GET VDP COUNT
        CLR  @>8378             CLEAR COUNTER
#endasm
}
```

MICROREVIEWS

TI-SCSI Cataloger and Atari Super Storm

By CHARLES GOOD

TI-SCSI CATALOGER by Dave Connery

Several programs will catalog SCSI drives. For 99/4A users this is the most useful SCSI cataloger I have seen to date. It appears to be written entirely in Extended BASIC, which is surprising when you consider that SCSI users have until recently complained about the lack of a good 99/4A cataloging program. This software is not limited to cataloging SCSI drives. It apparently can catalog any device including floppy drives, RAMdisks and maybe hard and floppy disk controller (HFDC) drives.

When you load TI-SCSI Cataloger you are asked for a device name. You can use DSD1 or SCSI or WDS1 or a longer path. If you don't put a period after the device name the software will do it for you. Press <enter> and your device is cataloged on screen and optionally to a printer. At the top of the screen you see the kind of device you are accessing. When I catalog a disk the display says "90k disk" for SSSD, "180k disk" for DSSD, and "360k disk" for DSDD. My SCSI hard drive displays as "40.2 meg H.D." Also displayed are the volume name of the device and the number of sectors used and free. Below that are the file names in groups of 12. This is a very comprehensive, informative display.

If there are more than 12 files to be cataloged the software displays the first 12, waits several seconds, displays the next 12, waits several seconds, and displays the next 12, etc., until all files have been displayed. The deliberate delay between pages of file names gives you plenty of time to take some action if you want. Possible actions are (N)ext page (immediately, rather than waiting for the delay to end), (P)revious page, move the cursor (U)p and (D)own the current group of 12 files, (R)emove a file, (C)hange drive to be cataloged, e(X)it to XBs and (V)iew a directory.

The (V)iew feature is lacking in the somewhat faster Harrison SCSI cataloger. You move the cursor to a subdirectory name. Such names are identified as subdirectories in the screen display. When the cursor is next to a subdirectory you press V and you can then catalog that directory. You cannot easily move up a directory tree structure toward the root directory, but you can move down into the tree structure cataloging subdirectories within subdirectories.

TI-SCSI Cataloger is not a complete disk manager. The software does mot make or delete directories, format disks or hard drives, execute software or view files. It is most useful to 99/4A owners who want an easy way to catalog a hard drive. It works fine on a Geneve in TI mode, but Geneve users already have Clint Pulley's Directory Manager which is far more complete than TI-SCSI Cataloger.

This is fairware and comes on a SSSD disk. The author requests a donation of a few bucks and/or a phone call expressing appreciation for his efforts. Send me $1 and I will mail it to you on a TI disk, or send me an e-mail and I will e-mail it to you for free in either PC99 or TIFILES format.

SUPER STORM by Atari

This is a 16K Atari game cartridge for the 99/4A. You need nothing except the console, a monitor, and joysticks to play it. I'll bet you never heard of this title! It is, in fact, a never-released game specifically written for the 99/4A by Atari back in 1983. The title does not show up in any list of the Atari games lists I have checked. This means that at least the name is different from the name of any Atari 2600 or other game system cartridge.

The story of how this "new" game cartridge is only now available to TIers is interesting. Kyle Crichton of Competition Computer almost literally rescued cartridges from an Atari dumpster after the company was taken over by its present owner. Kyle purchased, with the rights to resell, a whole bunch of dumpstered cartridges, including many cartridges for familiar 99/4A Atari games. A few samples of Super Storm and another never-released

Continued on page 50

MICROREVIEWS

MICROREVIEWS

Continued from page 49

game called Robotron were in this batch of cartridges. Kyle negotiated a deal with Atari's owners that allows him to burn eproms for these two games and sell them as long as one of the cartridges he purchased is consumed in the process. Super Storm is available for sale now and Robotron soon will be.

Super Storm comes with no documentation. The folks at Atari couldn't find any. Despite the lack of documentation my 17-year-old son Colin and I have more or less figured out how the action goes. You have a ship that moves around in a green sea at the bottom of the screen. Giant drops of slime rain down on the ship and if one hits the ship you are sunk. The joystick moves a green wedge-shaped shield which can be permanently positioned with the fire button. You get up to 20 of these wedges to position. You are supposed to position multiple wedges so that the slime drips are deflected to the left or right so that they do not fall into the sea. As drops fall into the sea the sea level rises and when it gets too high everything is flooded and the game ends.

Colin, who has played a lot of computer games including most of the TI games, says, "Super Storm is an average TI/Atari game compared to others like TI Invaders and Centipede. The game gets harder and harder as you go on. The water rises putting you higher up on the screen and giving you less time to aim your cannon and the bombs fall more rapidly."

Although I have never heard of Super Storm it turns out that Bill Gaskill has. Bill probably knows more about TI cartridges than anybody else. He has researched all the old computer magazines. Although Bill has never seen the game he was able to send me the following very interesting information:

"Below you will see an entry from the cartridge NOTES page on my Web-Site, No. 29, which talks about the SLIME cartridge. I believe SLIME and SUPER STORM to be one and the same.

"29. The only references that I have been able to locate for the Slime cartridge appear in the June 1983 issue of the President's Letter from the International 99/4 Users-Group on page 2 and in the August 1983 issue of *Compute!* magazine on page 36. The IUG article focuses on the Summer Consumer Electronics Show and in one paragraph states: At the Atari booth, demos were being run on soon-to-be-released titles such as Pac-Man, Defender, Donkey Kong, Centipede and Dig Dug. Atari Publishing also announced that it will be bringing to market four additional titles under a licensing agreement with Synapse. These will include Shamus, Protector, Picnic Paranoia and Slime. The *Compute!* article also talks about the Summer CES, but focuses heavily on the reorganization at Atari, including the forming of Atari Publishing (Atarisoft) with the announcement of that division's intention to make software titles for competing computers, including the TI99/4A.

"The original Slime program was written by Mike Hales for the Atari 800 in 1982 (source: The Giant List of Classic Game Programmers by James Hague) and was manufactured by Synapse Software, 820 Coventry Rd., Kensington, CA 94707 (415) 527-7751 (source: *Antic* magazine June 1982, p.38 under 'New Products'). The game's theme tells you that Plexarian Invincibles threaten all life on Earth. These invaders hover in the sky and drop layer after layer of SLIME into the Sargasso Sea. Their intention is to raise the level of the oceans until all human life is drowned. If that happens, the SLIME-breathing Invincibles will colonize the Earth. You must stop them with meager defenses, or mankind perishes.

"In the Slime game large drops of slime fall from the sky onto your ships. If one drop hits your ship, it will sink. Use the triangular diverters to aim the slime into buckets on the sides of the screen. If slime falls into the ocean the level of the ocean rises. When the ocean level reaches the top of the screen the game is over.

"If one reads the excellent article 'Pole Position, Jungle Hunt coming,' authored by Laura Burns in the March 1984 issue of MICROpendium, on page 12 she describes the theme of a game cartridge named Super Storm that is identical to that of Slime. This leads one to the logical assumption that Slime was to be released as Super Storm in its TI99/4A version. An Atarisoft ad in the January 1984 issue of *BYTE* magazine on page 409 shows the Super Storm cartridge as being available for the TI99/4A. It does not list Slime as being a TI99/4A product.

"Maybe you can use some of this worthless trivia in your review? If so, please feel free to do so."

Bill also sent me the text of a review of Slime published in the April 1983 issue of *ANTIC Magazine.*

Thanks Bill! Bill's Internet Web site is a treasure trove of interesting information about the 99/4A, one of the two best 99/4A sites. If you have Internet access you should check it out at http://www.gj.net/~lucky7/.

After seeing and playing with Super Storm there is no doubt in my mind that Super Storm and Slime are the same game. You can view a screen shot of the Atari Slime at http://www.tiac.net/users/jgoodman/atari/s_slime.html and see that it looks very similar to Super Storm. Super Storm is available for $29.95 as a cartridge from Competition Computer.

ACCESS

Charles Good (source for TI-SCSI Cataloger), P.O. Box 647, Venedocia OH 45894; phone (419) 667-3131; e-mail good.6@osu.edu

Competition Computer (source for Super Storm), 350 Marcella Way, Millbrae CA 94030; phone (800) 471-1600 6 a.m.-3 p.m. local time M-F

MICROREVIEWS

NEWSBYTES

Becker, Wright to speak at TI Tref

According to the TImes newsletter from the United Kingdom, Michael Becker of Germany will demonstrate hardware produced by his user group and Mike Wright of CaDD Electronics in the U.S. will demonstrate the latest version of PC99 at the 13th TI Tref at the Beeches Hotel in Nottingham, England, Oct. 9-11.

For further information, contact Richard Speed, 8 Corfe Close, Southwater, Horsham, West Sussex RH13 7XL, UK, or RichardSpeed@ResolutionGroup.Com.

German group testing loader

Michael Becker says his SNUG user group is working on a multi-loader for DSR's (ASCII and HSGPL).

"We are in beta-test with a new DSR for the EVPC (our 80-column-card), with the feature of in-system-modify of the DIP-switch, loadable palette-set, loadable configuration and so on. This DSR make use of the internal NOVRAM of the card."

HSGPL-program V1.14a, a multilingual version, also is in beta-test.

"We can make a second release of our old BwG disk controller, if necessary. Up to 30 cards are possible. The card supports up to four floppy-drives with maximum DS/DD (360K)." The card has a MM58274 clock-chip as ("CLOCK") and supports a second DSR with the old IBM 320K format, like the rare double-density TI disk controller. It supports the obsolete WD1773 and works with PC-Transfer in Corcomp mode.

"The EVPC2-card is possible," Becker said on the TI list server. "We are counting the pre-orders, than we will decide if we can produce the card for an aceptable price."

The SGCPU is nearly sold-out. A second-release is possible.

For more information, send e-mail to Becker at michael.becker@man.adtranz.de.

Leader dies, CONNI disbands

Richard (Dick) Beery, serving his second term as president of the Central Ohio Ninety Niners (CONNI) died suddenly on March 19.

Beery, a member of the group since its founding in 1983, taught classes in TI BASIC to club members.

John Parkins, past president of CONNI, received a notice that the club's incorporation charter would end in 1998. The club voted April 15 not to renew it, refunding dues to members who had already paid and donating other funds to Neighborhood Services Inc. in memory of Beery.

Play-In released for Midi-Master

Bruce Harrison is creating a new program for MIDI-Master owners, called Play-In. This program, according to Harrison, allows the user to play music on the MIDI instrument and have that music recorded in the TI's memory. From there, it can be saved to disk, recovered from disk and played back through the instrument. The program also includes a "de-compile" feature, so music contained in memory can be written out to a D/V 80 file in SNF format. That file can be edited with Editor/Assembler or Funnelweb,

Continued on page 54

NEWSBYTES

NEWSBYTES

Continued from page 53

then recompiled using MIDI-Master V2.5A or V2.5Z.

The program is scheduled for release at the May 1998 Lima Multi User Group fair. For further information, contact Harrison at 5705 40th Place, Hyattsville, MD 20781.

USER NOTES

Turbo modification for Myarc controller

This item was written by Jerry Coffey and has appeared in several user group newsletters. It's a hardware project, so any reader who attempts it is entirely responsible for the outcome.

This "turbo" modification locks out the "read after write" (write verify) routine usually performed by the Myarc disk controller. Here are the details:

Find the 74LS251 chip at the top center of the controller board, above the DIP switches and beside the large FDC chip (marked WD1770).

Solder a wire from the No. 2 pin of the 74LS251 through a switch to ground (e.g. the wide trace of the DIP switches or any trace connected to that wide trace). Fig. 1 shows the location of pin No. 2 when viewed from the bottom (non-component side) of the board.

You can tell it is working if your controller writes as fast as it reads. Normally, the write takes twice as long.

Fix for TPA and V5.0 of MDOS

Users of MDOS V5.0 may find

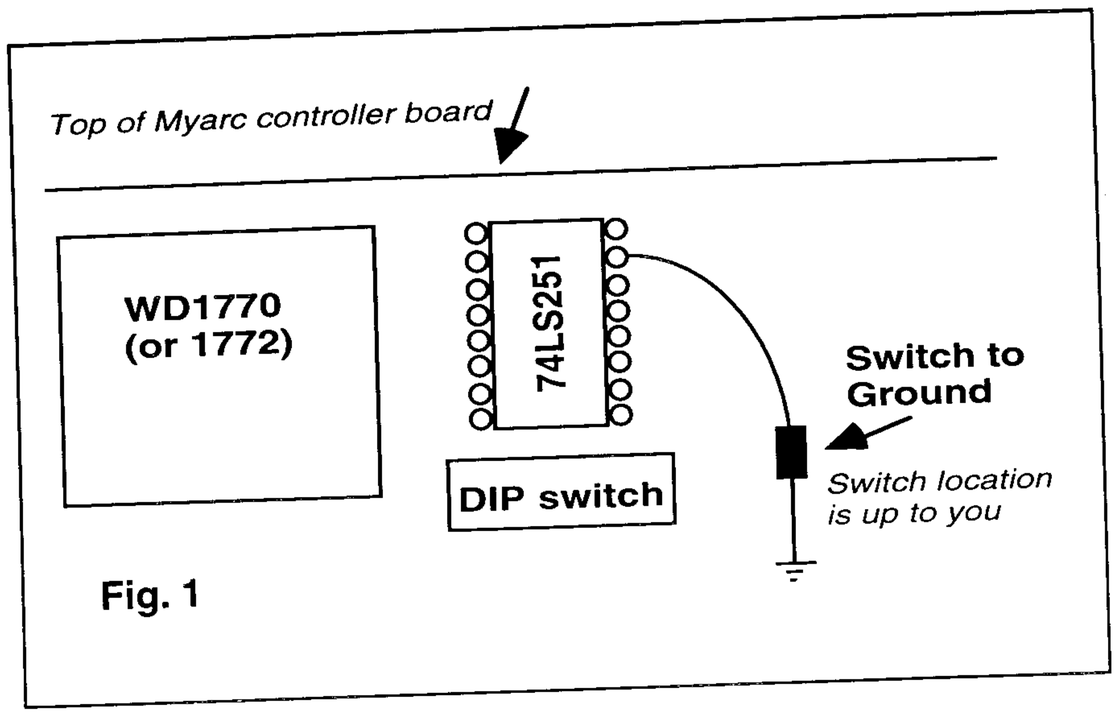

Fig. 1

USER NOTES

that TPA (The Printer's Apprentice) may not work in TIMODE, though it works in MDOS mode. Here is a fix for the problem, posted on the TI list server by Tim Tesch.

After disassembly, here's the fix.. Load John Birdwell's DSKU. You want to sector-edit the TPA file. In the very first sector you will see the following hexadecimal string:

028100241306

Assuming you have version 1.3 of TPA, change the "13" to "1A" so the string reads:

028100241A06

This should correct TPA.

What you will find is that TPA does not have sufficient memory. When you run TPA, it requests 36*8K pages, or 288K of memory. The error trapping, which the above modification will fix, allowed TPA to try loading even when there was insufficient memory to operate!

The only other problem I see is that TPA does not check properly for missing files. If TPB/TPC are not present, an error is not generated, and that means a lockup or strange effects. I suggest backing up TPA if you have not done so already, just in case.

CLASSIFIEDS

Download files TI to V9T9. .25 per file. Maximum of 25, TI Assembly Language Manual $5, Power Supply $5, 9 Games for Geneve, $9, add $1 S&H 765-664-6001.

FOR SALE
TI-RS232 card, $45 (obo); Fast Mac V34 modem (28.8K), works with any computer, $35 (obo). Call 512-255-1512 or email koloen@earthlink.net.

Classified ads are 10 cents per word. Payment (checks, money order, or credit card) must accompany ad. Mail to: MICROpendium Classifieds, P.O. Box 1343, Round Rock, TX 78680.

DISKS/BACK ISSUES

❑ Back Issues,$3.50 each to March 1996, later $6 each. List issues on separate sheet.
No price breaks on sets of back issues. Free shipping USA. Add $1, single issues to Canada/ Mexico. Other foreign shipping 75 cents single issue surface, $2.80 airmail. Write for foreign shipping on multiple copies.
OUT OF STOCK: V1#1-2; V2#1

GENEVE PUBLIC DOMAIN DISKS (SSSD unless specified)

These disks consists of public domain programs available from bulletin boards. If ordering DSDD, specify whether Myarc or CorComp.

	SSSD	DSSD	DSDD
❑ Series 1	$9	$7	$5
❑ Series 2	$9	$7	$5
❑ Series 3	$9	$7	$5
❑ Series 4	$9	$7	$5
❑ Series 5	$9	$7	$5
❑ Series 6	$9	$7	$5