TEXAS INSTRUMENTS USER GROUP U.K.

The tenth anniversary of TI pulling out of the home computing market!

Autumn 1993
ISSUE 42

TI99/4A USER'S GROUP (U.K.) CONTACTS.

CHAIRMAN:
Trevor Stevens. Tel 0623 793077
249 Southwell Road East, Rainworth, Nott's. NG21 0BN
VICE CHAIRMAN, CASSETTE LIBRARIAN, PROGRAMING:
Mark Wills. Tel 081 8660677 after 5.30pm
207a Field End Road, Eastcote, Middx. HA5 1QZ
GENERAL SECRETARY:
Richard Twyning. Tel 0623 27670
24 Peel Road, Mansfield, Nott's. NG19 6HB
MEMBERSHIP SECRETARY BACK ISSUES:
Alasdair Bryce. Tel 0389 65903
51 Dumbuie Ave, Silverton, Dumbarton, Scotland. G82 2JH
TREASURER:
Alan Rutherford. Tel 0625 524642
13 The Circuit, Wilmslow, Cheshire. SK9 6DA
TI*MES EDITOR:
Gary Smith. Tel 0636 706767
55 Boundary Road, Newark, Nott's. NG24 4AJ
HARDWARE:
Mike Goddard. Tel 0978 843547
"Sarnia", Cemetary Road, Rhos, Wrexham, Cwld. LL14 2BY
DISK LIBRARIAN JOURNAL EXCHANGE:
Stephen Shaw.
10 Alstone Road, Stockport, Cheshire. SK4 5AH
PUBLICATIONS:
Mike Curtis. Tel 0209 219051
21 Treliske Road, Roseland Gdns, Redruth, Cornwall. TR15 1QE
MODULES:
Francesco Lama. Tel 0865 721582
14 Granville Court, Cheney Lane, Oxford. OX3 0HJ

EDITOR COMMENTS

One thing I have noticed while editing is that most of the articles seem to be written by the same group of people each issue. It would be nice to here from the silent masses. Maybe a Problem page or Facts page may invite a few articles. Let me know what you think!!

DISCLAIMER

All views by contributors to this magazine are strictly their own and do not represent those of the commitee. Contrary opinions are very welcome. Errors will be corrected upon request.

*NEXT COPY DATE.

All articles should be in by the 15th December in order to distribute in Christmas week. Please ensure all copies are as dark as possible and that they fit onto A4 paper with 15mm left and right margins and 15mm top and bottom.
Thankyou for your help.

CONTENTS

Lack of Drive!.

Mark Wills sends his appologies for not having an article in this issue of TI*MES, but his console has been playing up and is out of action at the moment.
Someone is sending him a console in return for a few games cartridges so he will be back on line soon.

There was talk that he would write his article and modem it over so it could be converted to TI-Writer and used but he is very short of time.

He also tells me that the cassette library will be fully operational in a couple of weeks.
He said that he'll be back in the winter issue with more C99 to worry about.

PEB FOR SALE.!

George Michel of 34 Pippin Hill
Denby Village
Nr. Ripley
Derbyshire

Tel : 0332 881067

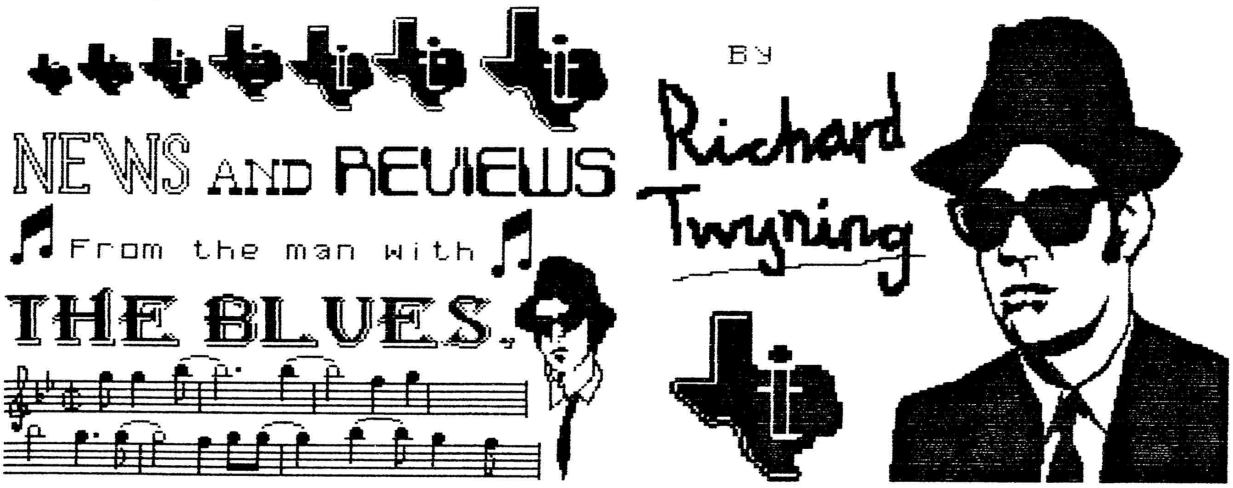
has decided to sell the following:

PEB with
RS232 card
TI Floppy card
One internal + one external SSSD drive
Hard disk controller with Segate ST105 and cables
Cassette player + leads
Navrone Widget with lead
Disk Manager
Minimemory
TI-Writer
EA with manuals
etc etc etc

On Disk: TI-Base
Missing Link
Fast Term
AC circuit analysis
etc etc.

This is all in good condition and will be a bargin for anyone who wants to expand.

If you are interested in this offer then please contact him at reasonable times on the above number.



Dear TI'ers,

I Hope you enjoyed my last article. I did report on many things, and only hinted at other items.

The last item that was scribbled in at the last minute about breaking the final frontier was in reference to a project that we have had in mind for ages.

This might seem far fetched at first, but this project is to build a new computer!

If you remember Myarc's advertising:

"The long winter ended and the ugly duckling is now a GENEVE 9640."

The second winter ended and the GENEVE was incinerated in the long hot summer! But the PHEONIX will arise from the flames! Yes! That's what it's going to be called.

As I am writing this article todays date is the 28th of June 1993 and it's 2:31pm. I'm living a bit of the executive lifestyle! I'm writing this section of the article on paper to type up later! I've got the bed out on the patio, and I'm catching some radiation from our nearest star!

I've got the phone, and I'm waiting for a call from Mark Wills to finalize the details of a meeting which, by the time you are reading this, will have hopefully taken place on the 10th of July.

Trevor has come up with the idea of taking his new caravan for the weekend, which is quite a home from home and sleeps five people and includes a shower, electric flush toilet, full gas cooker with main oven, grill, and four ring hob. Not forgetting the fridge of course!

The meeting is on a Saturday, and we are travelling down on the Friday night. Gary is travelling across from work

placement in Swindon and is also spending the weekend in the caravan.

The meeting is at Mark Wills' place in Eastcote. We are going to discuss the necessary features of the machine, and possibly do some designing of certain areas. The machine will probably wait until we've got our 9958 project released.

The 80 column card can then help to fund the PHEONIX. Gary and I visited Mark and his friend Dave on the 29th of May. I drove down to Gary at Swindon and we got the train to London for £29 Return!!!

Mark met us in Paddington station, and we travelled to Eastcote on the tube, via Baker Street station. We took the circuit diagram of the V9990 board and Dave had a look over it and was pleased to see that we hadn't made the same mistakes as the Dutch on the output of the video signal. The Dutch board sends 5 Volts directly to the RGB lines and could seriously blow your monitor!

As you will have seen in last quarter's issue, the circuit that handles CRU addressing is complete, and Dave and Gary used it as a start for the V9958 design. After doing that, and sticking the 4464 RAM's on the diagram, everyone realized that the 4464 RAM's were obsolete, and supplies are beginning to dry up.

That's when ideas about how to solve the problem came into existence. In the TI world, and in the TI Expansion Box, there are some classic pieces of engineering. The best of all is the method of mapping the cards in and out, (CRU addressing). There is also the MYARC 512K card with the re-partitionable RAM Disk and Print Spooler, and for the GENEVE, there's the MEMEX card that detects when other cards are making an access to the same area of RAM, and it disables part of its memory to avoid damaging hardware.

The new piece of hardware that Gary and Dave designed can be safely added to the list of classics. It's operation is top secret, but it allows the 9958, which is designed for 192K of Dynamic RAM, to use a single 512K static chip!

That's all I'm going to say about the design. There'll be more on it in a little hardware review later in the article.

We cut things a bit close when it was time to leave Mark's place. We left at 8:10pm, and the last Return train from Paddington to Swindon was at 9pm. This doesn't sound too bad, but the journey from Eastcote into the centre of London takes 45 minutes, and then we had to get back to Paddington via Baker Street. Typically for British Rail, the 9pm train left at 9:15pm, so things worked out O.K.

Mark gave Gary the data manual for the Hitachi GDP 6440 Video Chip, and we spent the journey back working out new hardware possibilities, such as a laser printer port actually driven by the chip, because it supports its own laser printer!

Now, things are up to date!

I'm the same as Mark when it comes to writing articles. I tend to write them a bit at a time when I get the inspiration.

I also tend to make efficient use of any available hardware now that I've got the Amiga. I can write articles almost anywhere and then format them properly later with TI-Writer.

All of the stuff you have just read is being typed using PROTEXT Word Processor which runs on a PC. This was from a free disk on Personal Computer World, and it runs with no trouble on the KCS board on the Amiga.

The trouble with most word processors on the PC is that they try and be too clever. Standard text should be 80 columns wide, but things like Word Perfect totally screw things up when they try and be clever with font sizes etc.

They treat everything as a bit map, and it has to be converted to post script, and if you want to print straight text files to the laser, you have to convert it to Laserjet mode, rather than True Type, and you've got dozens of directories full of True Type fonts and it's all just a big waste of space.

The Funnelweb editor takes up just a couple of PROGRAM files, and it loads in less than two seconds from hard disk.

PROTEXT however is one of the rare PC word processors that treat text as text, and saves documents as a true text file. Admitedly, it does try and be clever in its own little way, and sticks crappy characters in front of every word for some sad IBM reason if you do a straight Save command, but if you select Save ASCII, it outputs a proper file. And it works on columns. Another sad thing about IBM's is that they take the operation of the TAB character too literally, and if you press the TAB key, thinking you're simply moving your cursor along faster, then you'd be wrong.

It puts tab characters into the text which appear on screen as very wide space characters!

I think the price of Word Perfect is somewhere in the sad range of £300, or about £1200 if you want the UNIX version!. Every one will keep mindlessly buying it though, and a few more will be conned into upgrading to Word Perfect 6!

And god help the people who spent their own money on Word Perfect for Windows! It's not too bad if a company buys it, because they don't know any better.

The company I've just got a placement with spend £17.50 on a pack of ten high density disks. That works out at 14.4Megs of storage. Trevor and I can get 100 low density disks for £26. That works out at 72Megs if their used as PC disks at 720K each, or if their formatted as actual Amiga disks at 880K each, then it works out at 88Megs! For £13 we can get fifty disks and store 44Megs! It's Beyond Me!

That's enough of the moaning! (for now!!)

Back to the subject of real computing.

Our meeting did take place on the 10th of July at Mark's place. Trevor and I left his place with the caravan at about 4pm on the Friday, and began driving down. After stopping for 15 minutes for some nutrition and beverage, we hit the M25 at about 6:20pm.

After parking on the M25 and turning the engine off at one point, we finally arrived at the camp site in Iver at about 7:50pm and set the caravan up. Gary travelled to Slough (Pronounced Sluff!) by train, and we met him in Slough station at about 8:50pm. Then we spent about half an hour driving around Slough, trying to find a Fish and Chip shop!

Apparently, there's only one, and for anyone unfortunate enough to get stranded in Sluff, it's opposite the NISSAN dealer's showroom!

They did very nice jumbo sausages, and Gary and Trevor had Kebabs. We took the food back to the caravan to eat. The camp site at Iver is only a few minutes away from Sluff by car.
It's a very good site, and they've got quite a bit of land. It must be worth loads for land like that so close to London.

The site is called Home Farm Cottage, and driving to the actual caravan field is a bit of an experience. The actual cottage is about forty yards up the drive from the main road. You have to drive past for another 50 yards, and then do a sharp left and manoeuvre the caravan between a tree on your left, and a shed on your right. Then you've got another 100 yard drive which is fairly straight which takes you past a mini orchard and hen houses etc. Then the road bends right, and you've got to get through a narrow gate which takes you into an orchard. After another 20 or 30 yard drive at the side of the orchard, you finally open up into the field. When we arrived there where about four vans there across the other sides of the field. We set Trevors van up right at the side of the orchard. The orchard wasn't fenced off, and we could literally reach out of the window and grab an apple!
We had driven so far from the road to get to the field that you could hardly hear the traffic. The only noises were from a field of sheep which was adjacent to the orchard, and from 747's a few miles away in and out of Heathrow Airport.

We took plenty of First Aid with us which was kept in the medicine cupboard. Gary expressed a dislike for the local brew, and was pleased I'd taken a can of Mansfield!
Whilst eating our chips, we read some of the stuff that Gary had very kindly brought for us, such as two issues of Micropendium, which contained quite a few interesting things, such as completely new releases of MDOS, GPL, MYARC Disk Manager V. All of these are now up to 1.15H, and now include, hopefully, full support for floppies on a hard disk controller.

In Micropendium was an article from Stephen Shaw, and it was nice to see our group getting its name onto the world map. A thing which I'm sure will happen more often, once we get the new 80 column card project in full gear.

Also in Micropendium was a new Extended BASIC cartridge, Extended BASIC 3, that contains just about every important piece of software, such as Editor/Assembler, V.5 TI-Writer, Terminal Emulator II, DM1000, Archiver 3.03 etc. Everything fits into a 256K cartridge!

We also had the first sight of Gary's amazing cassette port parallel printer interface which is currently based on only two 74LS174 flip-flops and two transistors. He says he can change it to work with three chips, and do away with the transistors altogether. The version he brought with him to the meeting however is the one that was described in last quarters TI*MES.

(Remember what I said about taking every opportunity to type my article. At the moment, I've got the company laptop PC at home on the coffee table. Our hardware manager has gone on holiday and left me in charge of his laptop!) Back to the subject!

Gary has had the cassette interface semi-working, and has just tested it by wires onto the Clock and Data inputs. As he touched the wires onto the two lines, the printer started doing things, but there was too much noise for him to control it properly.
It does prove that data is getting through to the printer though, even if it's random control characters.

He's got some little routines in an old issue of TI*MES which flip the motor control pins on the 4A cassette port. Using these, I'll be able to write a small TMS9900 routine for Extended BASIC which will take entire strings and pass them to the interface via the cassette port.

The routine will take data in any of the following ways:

```
CALL LINK("CASPRN",A$) ! Prints the contents of the string A$.
CALL LINK("CASPRN","Hello!") ! Prints `Hello!' on the printer.
CALL LINK("CASPRN",CHR$(12)) ! Prints ASCII 12 to the printer,
                               which is the Form Feed character.
```

The problem with this however, as I mentioned in last quarters' article, is that to do this you will need at least a 32K Memory Expansion, or a Mini Memory. Anyone interested in a standalone, or PE Box 32K expansion should contact either myself, Gary Smith, Trevor Stevens, or Mark Wills.
Depending how many people express an interest in the £12 memory expansion for the side of the console we can estimate the number of boards that we would initally need to make, and work out the cost of having that number produced.

Back to the subject again!

We finally got to bed on the Friday night at about

midnight, after polishing off half of the medical supplies, and talking about the new computer, and the 80 column card project.

We were due at Mark's the following morning at 9:00am, but we didn't get up until about 8:00AM. And we had to have a good breakfast!

Trevor supplied the sausages, eggs, and bacon, and I supplied the grilled waffles! I took the video camera, and filmed the proceedings, before we demolished the food. We sat down to eat at exactly 9:00AM. After washing the pots, and sorting everything out, we finally left for Mark's at around 10:00AM and arrived at 10:25AM!

Colin Hinson had already beaten us to it, and was unloading things from his car.

It's amazing what you think you know about TI, and then realize you didn't!

QUESTION: Why does every other micro computer manufacturer number the bit positions the opposite way to TI?

ANSWER: When TI first produced their first microprocessor, there were no other microprocessors on the market. The only thing they had got to refer to where mainframes, so they numbered it the same way that mainframes are numbered!!!!

Colin told us some very interesting stories about TI. He said,

Censored!

Sorry folks! Sensitive information!

Colin brought a modified 99/4A with him which contained a TMS9995 and some clever circuitry to get round the multiplexing problem. The problem with the TMS9995 is that it's only got an 8-bit data bus. The TMS9900 has got a 16-bit data bus. The GENEVE doesn't have any 16-bit wide RAM at all, whereas, the 4A's scratchpad (PAD) RAM is full 16-bit memory, and is connected to the full 9900 data bus. It is after this that the multiplexing is done and the rest of the system is only 8 bits wide. The data bus into the expansion box is 8 bits wide. That's the reason the GENEVE is hardware compatible with the 4A. Only an 8-bit data bus is required in the expansion box, so the 9995 can support this with no trouble.

If you want to put a 9995 into the console, however, problems with this arise because you have to take into account the 16-bit wide scratchpad.

You have to multiplex the 9995 up to 16 bits for the scratchpad etc., but leave it at 8 bits for the rest of the

system. You also have problem of providing a suitable clock for the 9995. The 4A has a 48MHz crystal which is sent into the TMS9904 which divides it down for the 9900. This has to be modified, and a seperate 12MHz crystal added to the 9995 board.

Colin also brought some TMS9995's with him. About £1000 worth of chips! He's also got a TMS9996! Yes! A TMS9996! Imagine a chip as advanced as a TMS9995, but with a full 16-bit data bus! It's got the same instruction set as the TMS9995, with Load Workspace Pointer, Load Status Register, and Signed Multiply and Divide instructions being added on top of the TMS9900 instructions.

The reason Colin built the 9995 modification was so that he had a usable 9995 machine. Channel 4 Television wanted some 9995 based machines to control Video Tape Recorders for horse racing programs!

TI have a development system which plugs directly into the CPU socket of a 9995 based machine. It makes the machine think that it is actually being run by a 9995, but in fact, it is being run by a large box costing about £5000 which allows you to do many more things that you can with an actual TMS9995. These include single stepping the machine and tracing the flow of interrupts.

Unfortunately, the console had been left collecting dust for so long that some of the chips had internally disintegrated. When we switched it on we got a constant black screen, which is very interesting, as you will read in a moment.

I like what Mark told me about Colin a couple of days before the meeting. He said to Colin that he was thinking of building a new computer, and Colin said that he had designed the modification to allow the 4A console to use the TMS9995.

Mark said, "have you got a copy of the circuit diagram?"

To which Colin replied,

> "Yes, I have, and I think I might even have a completed console still lying around somewhere!"

That's the ultimate definition of being really cool. You just happened to have designed a modification for the console that lets you use a TMS9995 (like you do), and you then build one and mislay it around the house somewhere (like you do!)!!!!!

We did discuss the PHEONIX project, and the V9958 project. Mark was very keen to use the V9990, but as you will read later, this would cause some problems with memory clashes when both chips tried to talk to the CPU at the same time, because the 9990 doesn't have the lower modes of the TMS9918, V9938, or V9958. So somehow, both chips would have to be used at once.

In the end, we didn't seem to make much progress on the PHEONIX project. The project started when we first discussed the

2022 note: The Phoenix project was still in the planning stage.... in 2022

80 column card project. Mark's friend Dave knows a good thing when he sees it. Mark told him how good the 4A architecture was and explained it to him, and he liked it immediately. So, when Mark got involved with the 80 column card project, Dave said that he could even re-design the 4A from scratch and incorporate the 9938 or 9958 straight into the design. That's where the first idea of designing the new computer came from.

The initial idea was to use the TMS9995, because the TMS99000 is no longer available and the TMS9900 is in short supply, and is expensive.

Then we thought, that if we were designing a new computer, we might as well use a newer CPU which didn't necessarily have the same instruction set, but had the power to emulate our instruction set with no great difficulty. You have probably realized by now that I'm not giving any secrets away of the design. It's too top secret!!!

After having a bit of a discussion about designing the new machine, we went into the computer room to have a look at the Texas Instruments TMS34020 board in action. The resolutions and colours are amazing, and it would be nice to have a 24-bit colour card on the 4A, but the average 4A owner hasn't got the disk capacity to store the 24-bit images. This is apparent when you compare GIF images to TI-Artist images. It's not unusual to have a 200 sector GIF file. I've even got one or two images for G6 and G7 mode which are 500 or 600 sectors long.

When you start talking about 24-bit colour, the difference can be as great again. The only answer to this would probably be to buy a Ron Walters SCSI card, which I think I will possibly do this year, so I'll be running my existing MYARC HFDCC ST506/412 interface, and the SCSI card in the same box.

Mark's got loads of amazing pictures just lying about on the Amiga hard disk, including part of a map of China, and he loaded it and created a new town and a road, and the resolution is so good that you couldn't tell the difference!

The first time Gary and I went to Mark's place in May, Mark had a problem with his console. This is apparently the same problem that Trevor had with his console, and the same thing appeared to be wrong with Colin's 9995 console. Mark's console went down again on the day of the meeting, and we checked everything. Trevor found a break in the wire of Mark's console AC adapter, but after fixing it, the console still wouldn't work.

The strange thing is that all three consoles, Trevor's, Mark's, and Colin's 9995 console, all powered up with black screens. It must be a common problem.

The most likely cause that Trevor worked out was that over the years, the TMS9918 or TMS9929 VDP (depending on NTSC or PAL) got so hot that the white heatsink compound, that is supposed to seal it to the metal block on the screening plate, became runny and ran down into the socket and caused a bad connection between the VDP chip pins and the socket. Trevor managed to get his sorted out by taking the chip out and spraying it with contact cleaner. Mark's problem however, seemed a little bit more serious, and he still hasn't managed to fix it.

Colin had to leave early, so we carried on trying to sort out the console without him. When we finally realized what time it was it was around 5 or 6 o'clock.

We hadn't had any dinner, and despite planning our evening meal carefully in advance, we decided on chips again, and invited Mark back to the caravan. We got some chips in Eastcote in an amazing shop that had some amazing women in it!

When it was my turn and the goddess behind the counter said "can I help you?", I thought to myself, yes, but I'd better have some chips first!!!!!

Well, what a night it was when we got back to the caravan! There was enough medicine in the First Aid cupboard, and I shared out the medical supplies. I think we had enough for two cans each.

I am sorry to say that I had my usual chip order, which is two jumbo's and chips, which is exceedingly bad for diet. That's why I've taken up swimming! It's doing my upper body good, but my few extra bits of flab aren't moving very fast!!! It must be the sausage rolls that I'm tempted to buy at work from time to time! I do like to keep on the move though whilst typing my article. At the moment I've got the laptop in a car park!!!

I think this could be my first article that's been typed entirely on something other that the 4A or GENEVE! It will be arranged properly, and printed with Funnelweb though!!!!

I can't trust any other machine for producing the output exactly as I want it other than the GENEVE.

PC and Amiga software seems to be too **Hit and Miss** when you're doing anything serious.

I've got quite a professional looking CAD program on the Amiga, but I can't trust the output for anything serious.
I've also got an Amiga CAD program that is supposedly worth £700, but it's totally unusable. You have to click several different things before you can even draw a line!

(I'm possibly making TI history now! I'm typing this bit in the car travelling at around 85MPH between junction 27 and 26 of the M1! Is this the fastest article in TI history?
Don't worry, my dad's driving, and I'm safely in the passenger seat!

We're checking out directions to a company which I've got to visit tomorrow to do an installation. I thought it would be a good opportunity to do a bit of my article, and also start typing up my work placement diary. I've got both of them in memory, and can flip between them with Control Y.

The PC world has finally caught up with Programmers Editor!)

Back to the subject!!!

When we finished our supper, and had finished discussing things in the TI world we decided to visit the local Public Beverage Establishment for further medication! And further discussion!

Then we decided to go back to Mark's again, and spent until 2am watching Mark's interesting selection of videos! Very enlightening!!!

The following morning we didn't get up until about 10am, and then had a full cooked dinner of potatoes and meat pie etc.!!!!

This was followed by apple pies in rice pudding. That was my recipe!

(I'm now typing in Loughborough! Sat outside my friend Nick's house on Storer Road. He officially finished his placement at SYNEL U.K., but as the hardware manager is away, and the other new employee is away too, it's left me and another bloke to cope with support.

Since our software is crap and uses too many files, the boss has called Nick back in for the day to give me some help.

The only trouble is that I've got to give him a lift to and from Loughborough! This means I've got to work on a plan if we finish work at 5:30, and I've got to bring Nick back home to Loughborough, because I'm due at Trevor's tonight for our first computing evening since his return from holiday.)

Back to the subject!!! AGAIN!!!!!

After dinner we relaxed over the latest edition of TI*MES and a cup of tea. The discussion turned to the 80 column card. Then we took Gary back to Sluff to catch his return train, and then we went back to pack up and hitch up the caravan and set off back to Mansfield.

It wasn't a complete success as far as the new computer was concerned, but we did return with a copy of one of Mark's video's!!!!!!

I have been asked by Alasdair Bryce to mention something in this article about new hardware products.

In my last article I stated that the V9990 project may have been revived, but this is currently not on the cards.

This is primarily because of the expense of mounting the actual V9990 chip. It's a surface mount job, and 128 pins, and the response that Gary and I received means that even if we got more people involved in the production, the amount that we would eventually sell would not make it worth our while to produce.

Another problem with the card would be that it would work as a totally separate graphics system. The V9990 does not possess the lower 9918/9929 modes that are still present on the 9938 and 9958. Therefore, the 9918 or 9929 couldn't be totally replaced by the V9990. It would be possible to run 80 column software such as Funnelweb through it with minor video address changes (because both VDP's could not be easily made to reside at the same address), but the average user would not want the hassle of having both a T.V. and a monitor running at the same time, which the V9990 would require. The T.V. signal would come from the 9918/9929 which still displayed standard 4A graphics, and the high-res V9990 modes would have to come through the monitor.

The trouble with this though would be that while the 4A modes were active, the V9990 would display complete crap, and while the V9990 modes are active, the 4A would display complete crap.

The board would only work in theory however, because we would have to rely on the fact that while we were talking to the 9918 or 29, the V9990 would ignore the commands.

And the 4A would ignore the commands for the V9990 while we were talking to that. The only way it would work would be if the V9990 was given its own CRU address.

The big advantage of the V9990 however, is that it's the same chip that's used in the Super Nintendo, and gives 32768 colours, and a top resolution of 1024 by 768. Oh yes, and it's got a few more sprites! 125 to be exact!

Have no fear, the 9958 is here! Derek Hayward, Gary Smith, Trevor Stevens, Mark Wills, and myself are developing a new 80 column card.

WHY? Well, the Dutch made a bit of a bad job of it. If you read the Dutch notes that Derek published in last quarters issue, then you will notice that they say that there's an error in the 99/4A GROM!!

That's like saying there's an error in the 4A's cassette port because it won't run SCSI hard disks!!!!

The 9918 and 9929 power up in pattern mode (TI-BASIC and Extended BASIC mode). The machine goes straight into the TI Title screen.

The 9938 and 9958 power up in higher resolution, so when the system powers up and the 4A sends the chip the data to produce the TI-Title screen, of course it won't display properly, because the new VDP chip is expecting high-res commands. That's the problem. It's not a bug in the GROM at all!!!!

So, if we don't consider the Dutch card, how can our card hope to compete with the other 80 column cards that are available in the U.S.A.????

For a start, the 9958 will only use 64K * 4 bit Dynamic

RAM chips, which are obsolete! As mentioned earlier, by using a revolutionary piece of hardware designed by Gary Smith, and Mark's friend Dave, the chip will use a single Static RAM!

The 9938 and 9958 have a maximum address range of 192K RAM, but our piece of hardware also allows the chip to access, theoretically, any amount of Static VDP RAM. Just imagine the possibilities for word processing packages. Half a Meg of memory usable as a text buffer without taking any CPU RAM!
That's better than a PC can manage! And if we use more CRU bits as extra address bits, then we can double the memory for every bit we use.

If we used just five more CRU bits as additional memory mapping bits, then that would give us 16 Megabytes of VDP RAM!

At each CRU address, Gary says we've got 128 free CRU bits!

If I reach for my pocket computer and do 2 to the power 128, I get $3.402823668 * 38^{10}$. Without using any CRU bits, the chip itself can be made to address 384K, so if you multiply that value by 384, then it gives you an addressing range of $1.306684289 * 10^{41}$.

In Megabytes, this would be:
130,668,428,900,000,000,000,000,000,000,000,000,000 Megs.

Which is (if I've calculated this correctly), ONE HUNDRED AND THIRTY MILLION MILLION MILLION MILLION MILLION MILLION MILLION BYTES!!!

Or ONE HUNDRED AND THIRTY MILLION, MILLION, MILLION TERRABYTES!!!!

If you could buy RAM for £1 per Megabyte, then it would still take several times the assets of the world wide IBM corporation to populate the board. And it would take rather a big board!

Think of the possibilities for animation programs! Several Megabytes of frames that can be mapped into the currently viewed screen by just altering the states of CRU bits!

Does that sound better than currently available 80 column cards then?!?!?

The 9958 also allows 19268 colours! None of the Amiga's 4096 colour crap.
The 4A will leave the Amiga standing for colour animation! It will be quite impressive with half a Meg of memory, but with the extra CRU bits used it would be totally mind blowing!

At the time of writing this, Gary is unfortunately down with a very serious ear infection. I had one myself a few years ago, and that was a bit worrying, but Gary's is ALOT worse, and it's affecting his balance, so the prototyping of the test board

is delayed. The revolutionary piece of hardware that controls the additional memory is already tested however, so there's no worry over the whole thing collapsing at the last minute.

What I've described above is physically possible, but maybe not financially possible, unless we can get hold of eight 16Mbit SIMMS for a reasonable price!

Asgard software have agreed to sell the new boards, and apparently they are going mad over it, and have contacted Mark several times, asking when they can get hold of an early release board to test.

Another bit of late news just in! Todays date is the 5th of September. Last night I phoned Gary to see how he was, and he has informed me that, while he has had some time being ill, he's gone right through the circuit design of the 9958 card, and he's totally redesigned the revolutionary piece of hardware and corrected a small part of the address decoding.

I said, "Have we still got half a Meg of VRAM?" To which he replied, "No!" He said, he thought he would include an extra CRU bit while he was at it, and he's designed it with a FULL Meg of VRAM as standard.

Trevor has returned from his holiday, and I called round to see him this morning after I'd been swimming. I did do him a note, as I thought he'd probably not returned, but he'd already returned, so me and my friend Paul from college went in for half an hour. Trevor said he put the 4A on as soon as he arrived, and he left it for a bit to warm the chips up!

Late news just in!

Article update!

Todays date is the 20th of September, as you will see if I do a get date and time function in Protext word processor!>>>>20 September 1993 21:32:59. <<<The time says 21:32:59, but it is in fact 22:36:24. The KCS, PC emulator on the Amiga won't let me alter the real time clock on the PC board! So it's stuck at Greenwich Mean Time, rather than British Summer Time!

I've just returned from swimming with Trevor!

Yep! I roped him into going! I'm going again tomorrow! It's the lifeguards!!!!! Eat your heart out Erika Eleniak!!

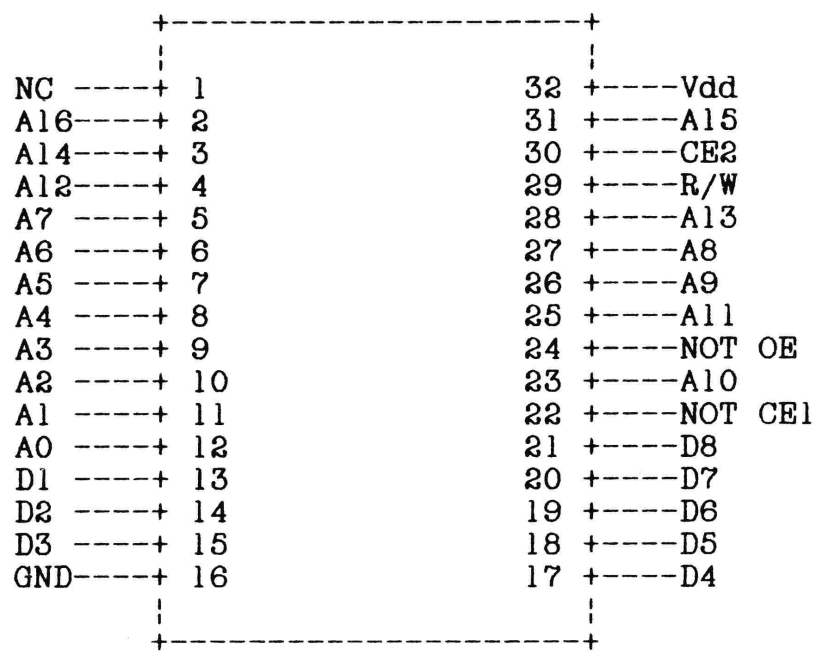
Where was I before my hormones overtook me?!?!?!?!?!?

Oh, yes, Gary phoned before I went out saying that there were a few memory problems on the new 80-column card. Financial errors, rather than technical. He's been pricing up some chips, and found that Static RAMs are becomming a bit expensive. £21.30 for a 128K * 8 bit (full 128K RAM chip).

However, I think I've found the solution whilst look through my RS book. RS are doing a Toshiba chip which is the TC551001APL-70L which has an access time of 70 nanoseconds. This is somewhat surprising, since when buying 25-99 they work out at £15.75. This is £1.55 cheaper than the NEC version which only has an access time of 85 nanoseconds!!!

I can't see how they're working out the pricing for that. It does have an asterisk by the side of the Toshiba chip which suggests that it's recently introduced stock, so maybe Toshiba are deliberately underpricing themselves initially, in an attempt to flood the market.

For the technically minded (Gary) here's the pin out.



SIZE: 1048576 bits
ORGANIZATION: 131072 words * 8 bits
ACCESS TIME: 70ns max.
SUPPLY: 5V d.c. +/- 10%
SUPPLY CURRENT: 70mA max.
STANDBY SUPPLY CURRENT: 3mA max.

PRICES: 1-24 = £21.00
25-99 = £15.75
100+ = £12.60

That's enough for now about the 80-column card. There'll be a progress report in my next article.

The name we are currently thinking of for the board is 80TIMES80. The board will also possibly be a half height board, and I might try and convince Gary to design a doodar that'll let you interface it straight to the side of the console!!!

Well, what's next in the line of hardware?

I won't do anything on RAM Disks, because Trevor said he was going to do that. I will say though, that Bud Mills now has an 8Meg, and possibly even a 16Meg HORIZON RAM Disk. I know that the 8Meg card will definitely work on both the 99/4A and the GENEVE. On the 4A, you can access all 8Megs as memory by bank switching around in 8K blocks. The GENEVE will do the same thing, but you need to take wires directly from the RAM Disk to the Gate Array on the GENEVE.

The other BIG piece of hardware is Bud Mills SCSI card.

Well, what's a SCSI card? Well, it stands for Small Computer Systems Interface, and it's not a hard disk interface.

It's a standardized general purpose interface that can be used to connect alot of devices such as plotters, printers, etc.

It's also Multi-drop, which means you can get about 8 devices on a single cable by setting jumpers which correspond to ID wires on the cable. The most common use for SCSI is, however, to connect hard disks, so now it's mainly advertised as a hard disk interface. Today there are two main hard disk interfaces. IDE and SCSI. Most PC owners seem to use IDE, but SCSI seems more powerful, because somehow it seems to let you have larger drives. I don't really know alot about SCSI, and even less about IDE, so I think Gary would be the man to answer any questions.

The advantage with finally having a SCSI card available is that we're back in the mainstream again, after Lewis Phillips hit the iceberg and then ran off with he lifeboats!

The DSR for the SCSI card is currenty being re-written so that we can actually access the larger drives. The problem with the MYARC HFDCC is that it still uses 256 bytes per sector as TI floppies use. The control chip on the FDCC will allow up to 64K per sector though!

If Lou programmed his DSR properly, then we could have had larger drives from the HFDCC. With the new DSR being re-written for the SCSI card, we'll be able to hopefully run a few ONE GIG drives!

Imagine running the disk library from that!
I could probably manage that if I get my full 200 Megs of ST506/412 drives on-line, but I think I'll invest this year in a SCSI card and either a half or a full GIG drive! I might even investigate re-writeable optical drives.

I just hope that the SCSI card has got a selectable CRU, so that I can run it along side the MYARC HFDCC. I could leave the odd 80000000 bytes here and there just for GIF's or

TI-Artist, or GRAPHX. Well, I must try and keep this article short. The time now is 13:58 and the date is 22-09-93.

I've got the rest of today and tonight to finish the article, and then I've got to transfer it to TI-Writer so that it can be properly, and professionally formatted.

Tomorrow night, we are editing the newsletter at Trevors house, so it must be finished in time. We are also hoping to have a game of Chess between Battle Chess on the Amiga, and Video Chess on the Cray 1!!!! There'll be a report on the result of this in the next issue.

Our 32K card for the side of the console is not off of the ground officially yet. The 80-column card project is too important yet. I've also been doing some calculations for the cartridge port. By using just two bytes as a memory map (which gives us a 16-bit address) we can access 64K * 8K , which gives us 536870912 bytes. Half a GIG! Therefore we've got an 8K supercart, 32K superspace, and when TI release their first Quantum circuit, we'll have the Half Gig Nob Off cart!!! Therefore, it's quite possible that we could design a 256K E/A cartridge that uses the chips that we're using for the 80-column card that I've described above. I am trying to keep this article as short as possible for this issue, so that's it for now as far as new hardware, and hardware projects are concerned.

I will close by mentioning a bit about my CAD program. Unfortunately, things ha ve been hectic for the last couple of months, and I've not had time to do much with the program. I have got some interesting routines off of my friend Nick, however, which include, hidden line removal, 3D calculations, and ray tracing etc. If possible, and if memory allows me, then as much of this as is possible will be incorporated into the program. What I have done so far is test out various routines with C99. The floating point handling is fully tested, and I'm completely clear on how it works. The trouble with C99 is that it doesn't have a standard built-in type for floating point numbers, so it has to be included as a seperate include file which handles the definition and storage of the numbers.

When trying them out however, they are so easy to use that you would think they were actually included as a physical part of the language. I've fully tested he passing of floating point numbers into functions, and it works 110%!!! I think I've chosen an official name for the CAD program though, "TI- Graphic Designer". Hopefully, I will have found the time to do some more work on the program by the time the next issue is out.

Well, that's about it for now!

Things to look forward to for the next issue:

Trevor and I are writing an Extended BASIC game which will hopefully be finished for the Winter edition, and will be printed in TI*MES so you can type it in and play over Christmas.

And there'll be the report on how Video Chess manages against Battle Chess.

And there'll be an update on hardware projects which will mainly be the 80-column card for the moment.

ONE LINER:

```
1 IF X=7 THEN PRINT SEG$(A$,
N+1,28) :: N=(N+ABS(N<23))*A
BS(N<23) :: GOTO 1 ELSE CALL
 CHAR(X+96,RPT$("0",14-(X)*2
)&"FFFF") :: A$=RPT$("`abcde
fedcba",5) :: X=X+1 :: GOTO
1! BY JOHN MARTIN
```

DISK LIBRARY REPORT....

Library fees are one pound per side copied, plus one pound per order for post and packing. You send the disks!
OR if you prefer the library to supply disks, add on 50p per disk required.

>UTIL32 is now full following the addition of Jim Peterson's PRINTALL Version 1.8, a multi-column printing utility.

>UTIL26 has added FANCYLIST which reads a DV163 merge file and prints various types of token in differing fonts.

>FUNLWEB version 5.0 80 column only. EDITOR ONLY- add on files to main disks. ON TWO DISKS. Includes lots of extras- UPDATED TO JULY 1993.

>EURO-FORMATTER is for an option of the above, and includes text files on the new Editor. TWO DISKS.

>FUNLWEB VN 5.0 40 COLUMN EDITOR- updated to July 1993- includes multi language and 8-bit modes and lots of extras! Bare bones version at present does not like to exit. TWO DISKS.

new series of util disks-

UTIL A:
DEFRAGMENTER uses just ONE disk drive, and puts fragmented files back into one piece quite quickly. Up to 32 files maximum. plus DIGITISER which reads sound into the cassette port and "digitises" it- note that this is a "one byte sampler" using square waves, so the quality can be difficult!

UTIL B:
KWIKDUMP- Harrison- Fctn 7 dumps screen TEXT-no graphics- fast. ; REMINDER- scans a DV80 address list and prints labels date flagged; ; GROM Master-experimental program to drive a self build gram type device; ; SAY- barry boone- experimental speech program ; SEARCH- adds to XB: CALL LINK("SEARCH",A$(),"LOOK",FOUND) to quickly locate a particular string in an array ; READER.

>MICROPENDIUM- programs from the magazine- CHEMICAL Elements-Regena-10/92; ; SKI/UTAH -Regena-11/92-Tourist guide! ; SPRITES -W Shepard, 10/92- demo ; XBCOMPARE- Barry Traver-11/92- compare XB progs for modificatios ; plus game FORE/AFT and utility HELPWRITER- Romstadt and Harrison- insert HELP screens into XB programs to call up at the touch of a button, and return to calling screen.

>TI GAMES VIDEO. About 5 hours of play showing various modules in use, including some "unreleased" modules. No commentary. Two pounds per week rental plus fifteen pounds returnable deposit.

>PHILLIPS 1. This series of disks is based upon the many programs of John Phillips. This first disk contains STARGAZER 1, 2, and 3, runnable from XB. These are educational programs based around the patterns of the Constellations. Also on the disk is to be found: MUNCHMOBILE, and a version of SCRABBLE, which with the loader on this disk has the LARGE character set.

>PHILLIPS 2. MUNCHMAN II-the two screen version with transporter pads. REQUIRES A LOADER eg FUNLWEB. D STATION- loader included is inadequate, use funlweb. 4A FLYER with manual on disk, MR PACMAN (by W Becherer), and STRIKE 3, a baseball simulation, which like all baseball simulations I have seen I personally find quite impossible.

>PHILLIPS 3. TWO DISKS. THE COMMENTED SOURCE CODE for 4A FLYER. Includes assembled versions to run from EA3 and EA5 and the manual, on disk.

>PHILLIPS 4. TWO DISKS. The commented source code for MUNCHMAN II. Includes versions to run from EA3 and EA5.

>PHILLIPS 5. Commented source code for STARGAZER 1.

>PHILLIPS 6. Commented source code for STARGAZER 2.

>SOKOBAN. A new machine code game from our group Treasurer, a block moving puzzle said to have 50 solvable screens. I can't get past level 5 however! Versions for Ed As Opt 2 and 3, the latter of which should load with most XB loaders such as that to be found with Funlweb.

>THE ULTIMATE ACCEPT AT by Bruce Harrison, a machine code CALL LINK for Extended Basic programmers. The title says it all!!!

>TIME CALCULATOR by Brice Harrison, another XB utility with machine code assistance to handle tricky calculations of time.

The complete library list is available on FOUR SSSD disks, just send along some blank disks and return post and packing!

Stephen Shaw

~~~~~~

2/03/92 18:41 ARCHIVE MATERIAL

HEX BUS for CC40 and 99/2 ON TI74 AND TI95:

HEX BUS

THE HEX-BUS CONNECTION:
By Dan H. Eicher via Lima Group March 1992

One facet of the whole TI saga that has been largely overlooked is TI's introduction of a new bus standard in the last days of their involvement within the mass consumer market. That new bus was called the Hex-Bus.

The Hex-Bus was planned as a standard bus throughout the TI family of low end computers. The computers that had built in Hex-Bus interfaces were: 99/8, 99/2, and the CC-40 (CC stands for Compact Computer), and the 99/4(a) was to have an interface cable. For one hundred dollars you received a box the plugged into the i/o port of your comsole and at the other end had a Hex-Bus plug.

What was so grand about this new scheme was the peripherals themselves, some of the peripherals that where actually produced where: disk drives, wafer tape drives, rs-232 interfaces, printer plotters, printers, and video interfaces.

These peripherals were VERY compact, all of them fit on a 4X4" printed circuit board. All Hex-Bus units could be used on ANY computer that had a hex bus port! That means that you could use your rs-232 on your 99/8 and when you went on a camping trip you just unpluged it and took it along with your CC-40.
~~~~~~

Several years after TI went out of the home computer market, they produced the TI-74 Basic Calc (which was much like a trimmed down CC-40) & the TI-95 Procalc (this is a much improved model over TI's original 95 programmable line). Each of these new computers had a new I/O port on the back that looked much different then the old Hex-Bus port. TI slipped and they put out a technical reference manual for the TI-74 (I say slipped because we all know how TI likes to give out technical information....not at all). I believe you can still order one of these TI-74 technical reference manuals from TI Cares.

When people got these technical manuals and started to look them over they where surprised to find out the Hex-Bus was back, only in a different package. In fact all that needs to be done to hook up a Hex-bus peripheral to one of these new computers is change the pin configuration (nothing has to be added, just the wires need to be moved around and put in a new package). The only thing new to this new bus (TI calls it, a Dock-bus port), is a power line has been added so a peripheral unit can provide power to either the 74 or 95. This was done because TI did NOT set up a power adapter jack in either of these two computers, so it was battery powered only. If you bought their printer and pluged it into a power supply, the power supply would provide power to both the printer and the host computer. This will not work with the older Hex-bus peripherals, nor is the CC-40 designed to accept power from one of the new Dock-bus peripherals.

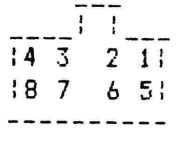
One of the major draw backs to the CC-40 was that there was not a built in cassette tape interface, and Hex-Bus disk drives and Wafer tape units are all but non-existant, this leaves the user without any means to save programs on a permenant media. The 74 & 95 also did not have cassetter interfaces built in so TI sold a unit that plugged into the Dock-Bus that allowed you to save programs to cassette.

At first many CC-40 owners where overjoyed, thinking they at last had found a solution to their mass storage delima, but it was not to be. TI did not build the code into either the CC-40 or the cassette interface that would allow the CC-40 to save to tape using this device. Although TI did build a version of the CC-40 called the CC-40+ that had a built in cassette interface, these unfortuneatly were only built for use "in-house".

I would like to note that even today, many CC-40 and TI-74 see service daily. TI wrote several custom programming packages for these two machines. Different government and private agencys send their employees into the field armed with a CC-40 or TI-74 and custom software, tailored to the type of calculations they do. I have even heard of a automated car wash in Germany that runs off of a CC-40.

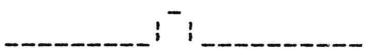
I have included a copy of the shape and pin ins/outs of the Hex-Bus and Dock-bus for any who may be interested.

Hex-Bus:



Pin	Purpose
1	D0 Data-LSB
2	D1 Data
7	D2 Data
8	D3 Data-MSB
5	HSK - Hand Shake
3	BAV - Bus available
4	GND
6	Protective GND

Dock-Bus



Pin	Purpose
1	System Power distribution out
2	System Power distribution in
3	D0 Data-LSB
4	D1 Data
5	D2 Data
6	D3 Data-MSB
7	HSK - Hand Shake
8	BAV - Bus available
9	System Reset
10	GND

For you experimenters here is the color codeing
of the HexBus Port:

Pin	Color	Pin	Color
1	Grey	5	Brown
2	Yellow	6	Green
3	Red	7	Black
4	Orange	8	Blue

++

NO MYSTERY ABOUT 32767

by Alexander Hulpke

If you had to remember a phone number, which would be easier: 123-456-7890 or 314-159-2653 ? Surely the first one ! If you had to describe a two dimensional object, which would be easier: A circle or a oddly shaped polygon? Most likely the first.

What is the reason for these examples: They show, that "simplicity" is very often a question of your point of view: If you are regarding numbers, as a ten-fingered person using the decimal system, the different symbols in their natural order are quite simple. If you regard objects, you look at their geometrical properties. If you were to regard them as numbers, for example calculating the area, the "simplicity" would be quite different: A polygon will most likely have an rational area, for example 523/45. The area of the unit circle cannot be described rational, it is an irrational number, its decimal evaluation starting with 3.141592653 .

You have seen, that "simplicity" is not always the same, it depends on what you think to be the important part of am object.

Let's think, you were a computer. As we all know, computers can only count 1 and 0. Which numbers would be more simple: 1000 0000 0000 0000 or 1100 0011 0101 0000? Most likely the first one. Since we are decimal creatures, we should want to regard these numbers as decimals: The first one is 32768, the second one is 50000, much more "simple" than 32768. You see, that the powers of two: 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192, 16384, 32768 and 65536 are "simple" numbers for dual creatures like computers, which also means that building something using these numbers is easy. For example, it means: Line 0 is Hi Level, the other lines are ALL Lo Level.

This is the reason for the proliferation of these powers in computers: We have a 16 Bit processor, having 64k=65536 Bytes Address Space (1k is 1024 Byte), the screen is 32 characters wide, each an 8*8 grid, thus having 256 horizontal resolution (the lower vertical resolution is due to the compatibility with the TV system, which is not a digital invention). We have 256 characters (though Basic allows only 128, thus saving some memory), that may fill a string up to size 255 (+1 size byte is 256). etc. etc.

But why 32767 which is clearly 2 to the power of 15 -1 ! Why the -1 ???

We all know, that there are two kinds of numbers: positive and negative. If we have to count apples, surely no negative counts can be possible. Thus, if we have numbers 1 to 65536, we could use them.

Now instead of counting apples, lets look at the thermometer (having only an integer scale). It might be necessary to use also negative numbers. If you have only positive numbers, what do you do? You would surely take (perhaps half of them) and CALL them negative numbers. So you would get 32768 positive and 32768 negative numbers. But wait: There also is a a number, neither positive, nor negative; the 0. We have to spare one number to be able to display a 0. Shell we take a positive or a negative one? To understand what is done, lets look closely at calculating with a finite set of numbers:

If its 22 o'clock and you "add" 4 hours, its 2. We were using a system of 0 to 23 and were "throwing the 24 away", regarding it again as zero. Thus 26=24+2 becomes 0+2=2.

What we have done is usually called "modulo arithmetic". A funny result is, that also negative numbers can be positive. Modulo 24 you can say either 22 or -2 since 2+22=0.

This arithmetic has very interesting properties. For example, you can show that a full calculations; which means, that you may add, subtract, multiply AND divide each two numbers and get a result again IN THE SAME SET, you started in (The mathematicians call a set, that has these properties "field") if and only if the number, by which you were doing the modulo arithmetic is a prime number. If you only add, subtract and multiply, it can be any number (this is called "ring"). Further examinations of these numbers lead far into the subject of abstract algebra, and we shall not cover this subject in this article much further.

Coming to the computer again, we already know that the powers of two are very important. Especially, since we have a 16 Bit processor, we have 16 Bit registers, which means, we have 65536 possible numbers for each register. So we will do again modulo arithmetic (note that 65536 is NOT prime, thus we cannot do fully division, but most times get a remainder, so the DIV command in machine language will always give quotient and remainder): As we have seen, -10 will be the "same" as 65526. There comes a very neat idea: if you add 65536 (which is in fact the 0 in our modulo arithmetic) to a number, you will get a positive number, that is the "same". We will cover this later.

This fact leads us to the system of number representations, used in the 9900. It is called "two's complement" and works like the following: To create a negative number, just invert all bits, and add 1 to the result. Thus -32767 is: 32767 equals 0111 1111 1111 1111 inverted: 1000 0000 0000 0000, add 1: 1000 0000 0000 0001 equals 32769, 32767+32769 = 65536 equals 0. Voila!

The interesting property is: Since its modulo arithmetic, its fully compatible with larger numbers. You can use either positive numbers 0 to 65535 or numbers -32768 to 32767 with the SAME routines for calculation. Its only dependent on your point of view!

If you regard numbers as negative, you can easily say if a number is negative, just by looking at it's first bit, that contains the sign. Of course, its not as simple as using Bits 1 to 15 for the number and Bit 0 for the sign, but then you would need different calculating procedures. Now we can understand the multiple occurences of 32767: Whenever Basic passes a number to Assembler, the number is regardes as to be signed. When we address memory, as for LOAD and PEEK, we calculate only with positive numbers. So we must subtract the number from 65536 to get the correct negative number if the address is too big to be a "register positive".

Other routines, that do not need to get numbers that big, would be very confusing: What is a negative position in a string? Thus SEG$ and POS are limited to 32767. Negative line numbers are not allowed, thus the largest line number is 32767 By the way: Line number 0 may not be entered, but you can create it by some manipulations. A program with line 0 will run correctly.

The fact that RES will place 32767 for each line number that could not be found is quite arbitrary. If you look at address >2314 in the Basic Interpreter (see TI Intern, pg.137), you will detect, that >7FFF equals 32767 is taken as an error value. I guess this was done, since most time this line will not exist and the program will break with an error instead of running into other routines that were prohibited. In fact only the line number 32767 generated by RES is arbitrary, the other occurences of 32767 in the TI are due to the internal number format.

NEVER RELEASED OFFICIAL T.I. PERIPHERALS: THE MILTON BRADLEY MBX EXPANSION SYSTEM

described by Charles Good
Lima Ohio User Group

This device was (and maybe still is) literally years ahead of the competition when first introduced to the public at the January 1983 Consumer Electronics Show. When attached to the TI99/4A it allows speech recognition with specific Milton Bradley game and educational modules. The user speaks instructions into a microphone, and the 99/4A understands the spoken words and responds accordingly. With the MBX system, our old fashioned 99/4A's can do tricks that even the most sophisticated modern home game machines can't do. Voice recognition is NOT available even today for Nintendo and Sega game systems. These days you can find voice recognition hardware costing hundreds of dollars advertised in Computer Shopper for use with MSDOS and MAC computers. In T.I.'s last complete price list of 99/4A products published in June 1983, the MBX system lists for $129.95. It's too bad only only about 300 (? I think a few more-sjs) were ever made!

Although the title of this article might suggest that the MBX system was made by T.I., this is not so. The MBX was manufactured and sold by the Milton Bradley Company. T.I., under license from Milton Bradley, manufactured and sold the specific software modules designed for use with the MBX. The MBX system comes packaged in a box with the the same kind of "photograph of the product on a black background with white letters" style found on 99/4A console boxes.

The actual MBX hardware is in the same gray plastic used for the most recent 99/4A consoles and official T.I. cassette program recorders.

As a registered 99/A owner, I received by mail in early November 1983 from T.I. (not from Milton Bradley) an advertisement describing the MBX and MBX specific software. Apparently Milton Bradley intended to have the MBX on store shelves for the 1983 Christmas season, but canceled all further production after BLACK FRIDAY. There is no serial number on my recently purchased used MBX, but it bears a sticker that says "MBX Control number 8310". This may mean that my unit was manufactured in the 10th week of 1983. My guess of 300 MBX units actually produced is based on the very limited availability of this product for sale at TI shows I have attended and in the possession of T.I. owners known to me, as well as the fact that UNISOURCE once advertised that they had 200 MBX's for sale.

There are three parts to the MBX "system", the control box, the joystick, and the headset/microphone. The heart of the system is of course the control box. It measures 10 x 7.5 x 2.5 inches and includes its own built in speech synthesizer. This box plugs into the joystick AND the cassette recorder ports of the 99/4A console. The MBX system is designed to be used with just the console and specific software cartridges. There is no provision in any of the MBX software modules for disk usage. Since one of the MBX connections occupies the cassette jack, you can't use a cassette recorder either. You must disconnect the regular speech synthesizer to use MBX. To hear speech, the two speech synthesizers cannot coexist.

The control box has a side port for the required AC power source. On the front of the control box are two 9 pin male D ports for joysticks, a jack for the headset/microphone, and an on/off switch. When you slide this switch to the ON position the MBX control box responds by saying "ready" in a well modulated female voice. This voice, and all speech generated by the MBX system, comes from a speaker at the top back of the control box, not from the monitor speaker. Music and other non-speech sounds continue to be heard from the monitor's speaker.

Only spoken words (synthesized speech) are heard from the MBX system's speaker. You have to turn on the MBX before you turn on the console in order for the 99/4A to recognize the presence of the MBX. When activated, the MBX system disables the FCTN/0 QUIT console keypress. On the top of the control box is a 64 position membrane keypad.

The top row of 8 keys on this keypad functions in the same way with all the MBX software modules that utilize this keypad. These top row keys include RESET, VOLUME UP (the volume of the speech coming from the MBX's built in speaker, not the music and sounds coming from the monitor speaker), VOLUME DOWN, MIC (toggles on and off the ability of the microphone headset to "hear" spoken words), YES, NO, PAUSE (stops game action), and GO. The action of other 56 positions on the control box keypad is specific to the particular software module in use. A very decorative keypad overlay comes with those software modules designed to utilize the rest of the MBX control box keypad. These overlays slip easily and snugly over the top of the keypad.

The headset superficially resembles a set of "walkman" earphones, but in fact contains no earphone speakers. The things that cover your ears are just pads. The microphone is positioned in front of your mouth and its position is adjustable. Physically the headset unit is flimsy. The wire leading to the microphone is thin and subject to stretching and damage at the point where it enters the adjustable microphone arm of the headset as the microphone arm is adjusted back and forth. Fortunately a handheld microphone designed to plug into a cassette recorder will also work with the MBX if the headset microphone breaks. The advantage of the headset over a handheld microphone is that the headset allows easy two handed manipulation of the special MBX joystick.

One joystick comes as standard equipment with each MBX system. A second joystick is listed in T.I.'s last 99/4A price list for $29.95 and can be plugged into the second joystick port on the control box. This would give each of two players their own separate joystick. In actual use of the MBX software modules a second joystick isn't really needed.

Only one player at a time uses the joystick. The two joystick ports on the control box respond the same. There is no "joystick #1" and "joystick #2" as there is with the 99/4A console. The MBX joysticks are very fancy and cannot be used by themselves directly from the 99/4A's joystick port. Likewise, you can't use regular joysticks from the MBX console. Movement of the MBX joystick handle is very smooth.

The device is described in promotional literature as a "triple-axis analog control that allows 360 degree object rotation and left to right and front to back proportional control of all movements." The word ANALOG suggests infinitely variable control. The MBX joystick's arm appears to produce the same kind of 8 direction movement typical of joysticks.

The "analog" infinitely variable control is the rotating knob on the end of this joystick arm. With some MBX games this knob will rotate the object under control to face any direction, for example to orient a gun prior to shooting. In the MBX baseball game this knob controls the force of a batter's swing. Minimum swing power results in a bunt. A trigger style fire button is included with the MBX joystick, as well as three other buttons. These three buttons resemble mouse buttons and have specific purposes when using specific MBX software modules.

How does MBX allow the 99/4A to respond to voice commands? At the beginning of each session with an MBX software cartridge that allows voice recognition as an option, the user is asked if he wants to use voice recognition. This is always optional. All the MBX cartridges can be used WITHOUT voice recognition by using the keyboard and/or the MBX keypad for input instead. If voice recognition is chosen, the user is asked which commands are to be given in voice.

It is possible to use voice for some commands and the console keyboard or MBX keypad for other commands, or to have all non joystick input by voice. The computer then directs the user to speak the possible commands (big, small, left, right, pencil, pen, centerfield, shortstop, etc) into the MBX microphone. This "voice training" of the MBX to recognize the user's voice patterns is repeated twice. Voice patterns are stored digitally on chips inside the MBX for the duration of the session, until the MBX is reset or shut off. This voice pattern storage is probably similar to that of some modern telephone answering machines.

My home answering machine does not store the greeting message on cassette tape. Instead, my "This is the Good household answering machine..." message that greets incoming calls is stored on a chip and played to callers every time I don't answer the phone quickly enough. As with the MBX, I can quickly erase my "stored on a chip greeting" and replace it with another on my answering machine. An MBX user can use any word desired for a particular command, as long as the user is consistent in using this word. For example, in CHAMPIONSHIP BASEBALL a user can speak the imaginary name of a fielder when asking for a particular fielding position. During voice training the computer can ask the user to speak the word "shortstop" and the user can reply "Tony". As long as the user remembers that Tony is playing shortstop, the game will work OK.

After voice training the game begins and the computer responds to sounds it hears via the MBX microphone. Users have to be careful to ONLY speak when they want the computer to perform some action. Casual conversation by the user can result in unexpected things happening as the computer interprets some of this conversation as specific spoken commands. The solution to this problem is to turn off the MIC using the MBX keypad when response to voice commands is temporarily not desired. A small symbol continuously on screen indicates the ON/OFF status of the microphone.

How well does it work? How reliable is MBX's voice recognition? It is about 80-90% reliable. Sometimes the MBX either totally ignores a verbal command, or the command is incorrectly interpreted as a different verbal command. Part of the problem is that during the excitement of game play, a player's voice may sound different than it did during voice training. In CHAMPIONSHIP BASEBALL it can be very annoying to command a throw to "second" and instead see the ball thrown to "centerfield". All voice commands can instead be activated from the 99/4A keyboard of the MBX's keypad with almost 100% reliability.

All the modules designed for use with the MBX, even those that absolutely REQUIRE the MBX, can be used totally without voice recognition. For really serious accurate game play, one should bypass the MBX's voice recognition feature. My testing panel is divided in their preference for voice recognition. Meaghan, my 5 year old daughter, likes to use voice recognition. I think she finds voice easier then reading the MBX overlay or memorizing complex 99/4A keypress sequences. Ian and Colin, ages 12 and 9, both prefer not to use voice recognition. High scores are important to these two serious game players, and such scores are easier to obtain with accurate game control.

What software is available? The following cartridges by Milton Bradley were specifically designed for use with the MBX expansion system. All these include speech synthesis and many also allow voice recognition. The speech synthesis of these software modules (but not speech recognition) can be accessed using the regular TI speech synthesizer without using the MBX system. They were officially released by T.I. in 1983 and 1984. The last (June 1983) 99/4A catalog published by TI lists these modules for $50 and $60.

Quoting from the booklet TEXAS INSTRUMENTS HOME COMPUTER PROGRAM LIBRARy that came packaged with many TI modules sold in late 1983:

"The BRIGHT BEGINNING SERIES includes four games which teach elementary programming, music, and other learning concepts. Ages 4-8."

HONEY HUNT
SOUNDTRACK TROLLEY
TERRY TURTLE'S ADVENTURE (MBX system required)
I'M HIDING (MBX system is required).

"The ARCADE PLUS SERIES has six arcade style games that take you from home town ball parks to meteor belts far, far away."

CHAMPIONSHIP BASEBALL (MBX system required)
SEWERMANIA
SUPER FLY
SPACE BANDITS
BIGFOOT
METEOR BELT

NAME	MUST have MBX system attached to console to use this software	Uses MBX keypad overlay	Uses MBX joystick's analog rotation knob	Uses MBX joystick's buttons 1,2 and 3	Uses SPEECH RECOGNITION as an option
TERRY TURTLE'S ADVENTURE	Yes	Yes	No	No	Yes
I'M HIDING	Yes	Yes	No	No	Yes
CHAMPIONSHIP BASEBALL	Yes	Yes	Yes	Yes	Yes
BIGFOOT	No	No	No	Limited Use	No
SEWERMANIA	No	No	No	Yes	Yes
SUPERFLY	No	No	Yes	Yes	Yes
METEOR BELT	No	No	Yes	Yes	Yes
SPACE BANDITS	No	No	No	Yes	Yes
HONEY HUNT	No	Yes	No	No	No
SOUNDTRACK TROLLEY	No	Yes	No	No	No

NOTES: -Most software cartridges allow use of the MBX joystick as an option.
-METEOR BELT requires two MBX joysticks if the MBX system is used in a two player game.
-All software cartridges allow limited use of the MBX keypad for RESET, PAUSE, and GO.

Comparison of Extended Basic, c99, Assembly, USCD Pascal - listings

SPIRAL

Wesley Richardson - Bluegrass 99 Computer Society BLUEGRASS 99 COMPUTER SOCIETY, INC.

A comparison of different programming languages doing the same thing.

Jon Keller wrote his SPIRAL program in c99 and in Extended BASIC. With the help of several people, his routine has been translated to four other languages. The TI-99/4A has the capabillty of running programs in c99, BASIC, Extended BASIC, UCSD Pascal, FORTH, LOGO, TMS9900 Assembly Language, PILOT 99, and possibly some others. The ease of writing the program, the ease of use, and the speed of execution varies widely between these languages. The run times for SPIRAL are given below:

EXTENDED BASIC	29m 11s	- listing below
LOGO	21m 13s	- no listing
USCD PASCAL	17m 20s	- listing below
FORTH	1m 20s	- no listing
c99	0m 58s	- listing below
9900 ASSEMBLY	0m 16s	- listing below

One of the criteria used in translating the program to the other languages, was to use the same algorithm in each program. Where possible, the names of the variables were also kept the same, to assist in correlation of the program listings.

The Extended BASIC version will probably be understood by the largest number of TI-99/4A owners. While having the greatest ease of generating the program, it also has the slowest run time. This program could be used in standard BASIC by changing the multiple statement lines so that only one statement was on each program line.

```
100 REM SPIRAL EXTENDED BASIC
110 REM JON KELLER
120 REM BLUEGRASS 99 COMPUTER SOCIETY
130 REM APRIL 1986
140 CALL CLEAR :: CN=33
150 ROW=13 :: COL=17
160 CALL HCHAR(12,16,CN)
170 N=2
180 FOR Z=1 TO 11
190 RD=-1 :: CD=-1
200 FOR X=1 TO 2
210 FOR Y=1 TO N
220 ROW=ROW+RD
230 CALL HCHAR(ROW,COL,CN)
240 NEXT Y
250 FOR Y=1 TO N
260 COL=COL+CD
270 CALL HCHAR(ROW,COL,CN)
280 NEXT Y
290 RD=1 :: CD=1
300 NEXT X
310 N=N+2 :: COL=COL+1 :: ROW=ROW+1
320 NEXT Z
330 CN=CN+1 :: IF CN<128 THEN 150
340 END
```

The LOGO version may surprise people by running faster than the Extended BASIC version. One limitation in LOGO was the characters or tiles above 95 are not pre-defined, so the routine has been recycled

to give the same number of tiles to be printed to the screen. After editing the LOGO procedure definitions, you only need to save the PROCEDURES to disk or cassette. Type SPIRAL as the starting procedure. In design, LOGO and FORTH are very similar. Both languages allow the definition of procedures or words based on previously defined words. A structure is built with increasing levels until the desired function becomes the calling word.

The UCSD Pascal version of SPIRAL will probably have the least number of people try it. Pascal is very similar in structure to c99. The program is written as a TEXT file, and then is compiled to a CODE file to run or execute. It was anticipated that the Pascal version would be much faster, but apparently on the TI, Pascal does not run as fast as on other machines. Since SPIRAL has the "USES SUPPORT;" it is necessary to have the file SYSTEM.LIBRARY on the ED-FILR: disk during both compiling and when execution is called. The SYSTEM.LIBRARY file can be T(ransfered) from the COMPILR: disk. After the program has run about 10 minutes, the screen will blank because no keys have been pressed. The program is still running. Press any key to return the screen display.

```
(* PASCAL SPIRAL PROGRAM *)
(* BY WALTER LIGHTNER *)
(* BLUEGRASS 99 COMPUTER SOCIETY *)
(* APRIL 1986 *)
PROGRAM SPIRAL;
USES SUPPORT;
VAR ROW, COL, N, Z : INTEGER;
    RD, CD, X, Y : INTEGER;
    CN : INTEGER;
BEGIN
  PAGE(OUTPUT);
  CN:= 33;
    REPEAT
    ROW:= 13; COL:= 20;
    GOTOXY(19,12);
    WRITE(CHR(CN));
    N:= 2;
    FOR Z:= 1 TO 11 DO
      BEGIN
      RD:= -1;
      CD:= -1;
      FOR X:= 1 TO 2 DO
        BEGIN
        FOR Y:= 1 TO N DO
          BEGIN
          ROW:= ROW + RD;
          GOTOXY(COL,ROW);
          WRITE(CHR(CN))
          END;

        FOR Y:= 1 TO N DO
          BEGIN
          COL:= COL + CD;
          GOTOXY(COL,ROW);
          WRITE(CHR(CN))
          END;
        RD:= 1; CD:= 1
        END;
      N:= N +2;
      COL:= COL + 1;
      ROW:= ROW + 1
      END;
```

```
    CN:= CN + 1;
    UNTIL CN > 127;
END.
================================
```

One advantage of FORTH is the ability to test 'words' from the console. Although FORTH does not have quite the speed of c99 or Assembly, during the development stage of the program, FORTH would be easier to debug.

```
================================
```

To enter the c99 SPIRAL program, use the Editor/Assembler Editor, and save the program on disk as SPIRAL;C. Use option 5 of E/A to load C99C and answer 'Y' to the first two questions. Use SPIRAL;C as the input name and SPIRAL;S as the output.
After compiling with no errors, load the E/A Assembler to assemble SPIRAL;S with no options and the output file SPIRAL;O. To run the program, load SPIRAL;O and load CSUP. The program calls dsk2.conio so this c99 file must be in disk drive 2. Give START as the program name to start.

```
/* c99 SPIRAL PROGRAM SOURCE CODE */
/* BY JON KELLER */
/* BLUEGRASS 99 COMPUTER SOCIETY */
/* APRIL 1986 */
/* ### is a shifted 3 */

#define one 1
#define none -1
#include dsk2.conio

int row,col,cn;

main(){ /* MAIN PROGRAM */
  int ff.bp,cd,rd,x,y,n,c;
  cn=33;
  putchar(FF);
  while(cn!=126){
    n=2; row=14; col=18; x=1; bp=0;
    while(bp!=1){
      x=1;
      cd=none;
      rd=none;
      while(x<3){
        y=1;
        while(y<n){
          row=row+rd;
          if(row>24){row=24;bp=1;}
          hchar();
          y++;}
        y=1;
        while(y<n){
          col=col+cd;
          hchar();
          y++;}
        rd=one;
        cd=one;
        x++;}
      n=n+2;
      col++;
      row++;}
```

```
      cn++;}
    putchar(FF);
    puts("again ? Y or N \n");
    c=getchar();
    if ((c=='y')|(c=='Y'))main();

    } /* END OF MAIN PROGRAM */

 hchar(){ /* HCHAR ROUTINE */
    locate(row,col);
    putchar(cn);
    }
 return();
```

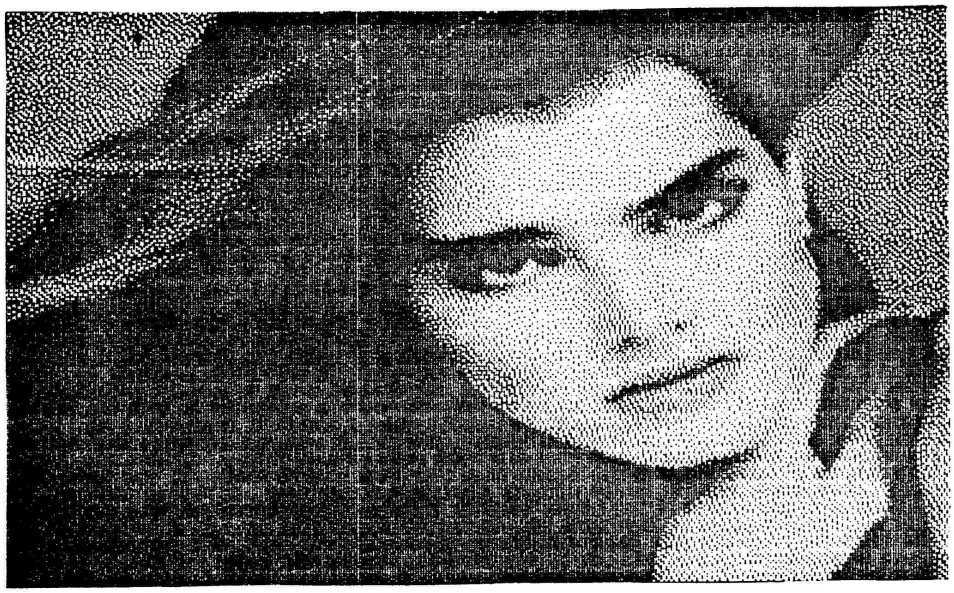
==========

TMS9900 Assembly language is the winner in terms of speed, using only 15.7 seconds to run. To assemble, be sure to include 'R' as an option. The object code will load from E/A #3, and give 'GO' as the program name to start. You had better watch closely and be good with your stopwatch if you are going to time this one.

```
        DEF  GO
        REF  VSBW
*************************************
*          SPIRAL PROGRAM           *
*   TI-99/4A ASSEMBLY LANGUAGE      *
* BLUEGRASS 99 COMPUTER SOCIETY     *
*            APRIL 1986             *
*************************************
REGS    BSS  32
CN      EQU  9
ROW     EQU  7
COL     EQU  6
N       EQU  5
Z       EQU  8
D       EQU  10
X       EQU  4
Y       EQU  3
*********************************
GO      LWPI REGS
        LIMI >0000
        LI   CN,33
LIN160  LI   ROW,12
        LI   COL,16
        BL   @WRITEC
        LI   ROW,13
        LI   COL,17
        LI   N,2
        LI   Z,1
LOOPZ   LI   D,-1
        LI   X,1
LOOPX   LI   Y,1
LOOPY1  A    D,ROW
        BL   @WRITEC
        INC  Y
        C    Y,N
        JLE  LOOPY1
        LI   Y,1
LOOPY2  A    D,COL
        BL   @WRITEC
```

```
        INC   Y
        C     Y,N
        JLE   LOOPY2
        LI    D,1
        INC   X
        CI    X,2
        JLE   LOOPX
        INCT  N
        INC   COL
        INC   ROW
        INC   Z
        LI    R2,12
        C     Z,R2
        JLT   LOOPZ
        INC   CN
        CI    CN,128
        JLT   LIN160
        LIMI  2
        BLWP  @0
** WRITE CHARACTER ROUTINE **
WRITEC LI     R0,32
        MPY   ROW,R0
        AI    R1,-32
        A     COL,R1
        AI    R1,-1
        MOV   R1,R0
        MOV   CN,R1
        SWPB  R1
        BLWP  @VSBW
        RT
        END
```

I would like to thank all of the people who helped in the preparation of the programs for this article. I know I have learned several things, and I hope this will give you some exposure to the variety of programming languages available for your TI-99/4A.



TI Artist image printed on Epson FX-80 printer

The Art of Assembly - Part 4

Memory Saving Tips

By Bruce Harrison

Back in the first installment of this series, we made the bold assertion that Memory is your Master. On the TI, that becomes apparent whenever one tries to do a really big job on this computer. Our Word Processor, which we use to prepare these articles, fills nearly all of the TI's memory capacity. On many occasions in writing and refining that program, we did "scrubdowns" on the source code, trying to find places where we could accomplish the same function with fewer bytes. That was necessary to add new features to the program. It's not unusual in a program that size (about 150 pages of source code) that one can find places to save several hundred bytes. After a couple of scrubdowns, this gets tougher.

In this article we'll pass along some of the lessons learned in that experience, and hope your Assembly programs will benefit. We'll start with one small concrete example.

Let's assume you have a variable called CURSCR, which is going to keep track of what screen in VDP RAM you're currently looking at. Since there are less than ten screens possible, you decide to make that variable a single byte:

```
CURSCR BYTE 0
```

That's fine until you discover that for many of its uses, you need to transfer that variable into a register to perform some action, and then need to transfer it back to the variable location. Look what that requires when we want the variable value in R4:

```
CLR  R4         Clear the register
MOVB CURSCR,R4  Move the byte in
SWPB R4         Right justify the byte
(do some operation)
SWPB R4         Move value to left byte R4
MOVB R4,CURSCR  Put byte back at CURSCR
```

That's okay if you only do it at one place in the program, but if it's required at many places, the one byte you saved by making the variable a byte will cost you dearly. If it were a word in memory as CURSCR DATA 0, then the above code would read:

```
MOV  CURSCR,R4
(do some operation)
MOV  R4,CURSCR
```

This takes six fewer bytes to perform than the previously shown code, because you skip the clear operation and you also skip the two SWPB operations. Yes, you could do the same thing as the first operation by:

```
MOVB CURSCR,R4
SRL  R4,8
```

```
        (do some operation)
        SWPB R4
        MOVB R4,CURSCR
```

But that still takes four more bytes than the operation would take by the second example above.

Let's look at another small example, from the Menu Driver we showed in Part 3 of this series. After the keystroke has been accepted, we did the following:

```
ACC1    MOV  R8,R5
        S    NUMASK,R5
```

And so on until we branch to the address contained in R5. Actually, we needn't have moved the keystroke from R8 to R5, since we really made no other use of R8 in that section of code. Therefore we could eliminate the instruction MOV R8,R5 entirely, and just substitute R8 for R5 in the rest of that section of the source code.

That particular change will only save us two bytes of memory, but it would be part of a wider "scrubdown" effort, in which many bytes might be saved over the whole program.

Just for a moment, we'll digress into the subject of Macros. The TI Assembler doesn't make any provision for them, but we don't use that Assembler. We prefer using Art Green's RAG Assembler, which does provide a capabilty to use Macros. A Macro is a sort of second cousin to a subroutine, but instead of being located at one place in memory and called from many other places, a Macro simply replicates a section of code wherever it's invoked. Our small subroutine MOVBTS, for example, could be implemented as a Macro. We would save some overhead that way, since the main program wouldn't need the BL **MOVBTS** instruction, which in itself uses four bytes.
Nevertheless, we don't recommend using Macros on the TI, because that will become a bad habit, and larger sections of code will be replicated over and over again in your programs, eating up valuable memory space. On the PC computer, we have resorted to using some very small Macros, to perform such functions as setting segment registers. Excessive use of Macros instead of subroutines is another reason that PC programs become overly large.

You may well ask why, then, do we use (and recommend) Art Green's RAG Assembler. That's simple. The RAG Assembler provides the best error reporting scheme of any Assembler we've seen. If, for example, you have an undefined symbol in your code, it tells you on-screen at which line of which file the erroneous label occurs, and shows you that line of source code. This makes it much easier to track down and correct source code errors.
[The rag assembler is available from your user group disk library]

Let's get off our soapbox now and get back to some serious business. There are many ways to save bytes in programs. We often find that savings can be made simply by changing the way we perform an operation. Here's an example from one of the subroutines in Part 2 of this series. Let's look at our screen clearing subroutine:

```
CLS   LI   R2,SCRWID  Sets R2 to characters in screen line
      LI   R5,>2000   Sets left byte R5 to space
```

```
        LI    R3,SCRLI    Point R3 at SCRLI
        MOV   R3,R1       Point R1 at SCRLI also
LOOP1   MOVB  R5,*R3+     Move one byte and increment R3
        DEC   R2          Decrement R2
        JNE   LOOP1       If not zero, repeat
        CLR   R0          Point R0 to screen origin
        LI    R2,SCRWID   Set R2 again
        LI    R4,24       24 rows to clear
LOOP2   BLWP  VMBW        Write SCRWID bytes to screen
        A     R2,R0       add that many bytes to R0
        DEC   R4          Decrement R4
        JNE   LOOP2       If not zero, repeat
        RT                Return to calling program
```

2022 NOTE: This looks like a TI Writer formatter error- I would expect to see BLWP @VMBW The @ has turned on the emphasis (bold)

The part at LOOP2 will serve as an example. We could have written that as:

```
LOOP2 BLWP VMBW       Write SCRWID bytes to screen
      AI   R0,SCRWID  add SCRWID bytes to R0
      DEC  R4         Decrement R4
      JNE  LOOP2      If not zero, repeat
      RT              Return to calling program
```

That would work every bit as well, but since R2 already contains SCRWID while we're executing this loop, using the instruction A R2,R0 saves us two bytes. Similarly, in the section before LOOP1, we anticipated needing R1 pointed to SCRLI, so we moved R3 to R1, rather than having to LI R1,SCRLI. That also saves two bytes.

Moving or adding values from register to register rather than from immediate values should be the practice whenever possible. Such moves not only save memory, but also execute faster.

Another practice we encourage is maximizing use of the integer math operations. In our Menu Driver, for example, we wanted to double a number in the range of 0 through 7. We could have accomplished that this way:

```
LI   R3,2     Place 2 in R3
MPY  R3,R5    Multiply by the value in R3
MOV  R6,R5    Put result back into R5
```

What we actually did was simply to SLA R5,1. This saves bytes and execution time. Whenever the range of possible outcomes is 0 through 32767 or less, doubling can be done in this manner. (Negative numbers can also be doubled this way.) There will of course be instances when the MPY instruction must be used, because the result will be too large to fit in one register, but every time one can use the shortcut SLA instruction, memory and time will both be saved.

Similarly, one can divide by two with a simple SRL or SRA instruction. In general, any time one needs to multiply or divide by an integral power of two (2, 4, 8, etc.), one should look and see whether the expected range of the outcome will permit shifting the number rather than using MPY or DIV to perform the operation.

There are of course exceptions to any rule. In our music programs, we perform timing of note durations using a loop. One of our customers disassembled our code, and told us that one operation in that loop could

have been performed by a simple compare operation instead of the DIV that we used. He was correct, except that we used DIV on purpose to kill time in that loop. A compare operation would take far less time to execute, but then we'd have had to find some other way to waste time in the loop, otherwise our whole scheme for timing durations would need revision.

We said in our last installment that this one would include some right and wrong examples. Here's one. Let's suppose that you have a menu on the screen, and you wish to branch out to one of six labels (FUNCT1 through FUNCT6) from that menu. Assume for the moment that the key value in question is in R8. Here's the wrong way to do that branching:

```
            AI    R8,->31   Remove number mask plus 1
            JLT   OUTRNG    If lower than 0, key is out of range
            JGT   CMP1      If greater than 0, jump ahead
            B     FUNCT1    Else function 1 chosen
CMP1        CI    R8,1      Has #2 been chosen?
            JGT   CMP2      If greater, jump ahead
            B     FUNCT2    Else GOTO function 2
CMP2        CI    R8,2      Has #3 been chosen?
            JGT   CMP3      If greater, jump ahead
            B     FUNCT3    Else GOTO function 3
CMP3        CI    R8,3      Has #4 been chosen?
            JGT   CMP4      If greater, jump ahead
            B     FUNCT4    Else GOTO function 4
CMP4        CI    R8,4      Has #5 been chosen?
            JGT   CMP5      If greater, jump ahead
            B     FUNCT5    Else GOTO function 5
CMP5        CI    R8,5      Function 6?
            JGT   OUTRNG    If greater, key is out of range
            B     FUNCT6    Else perform function 6
OUTRNG      (IGNORE THE KEYSTROKE)
```

Now here's the right way to do it. Start by putting a lookup table in the data section like this:

```
LUT         DATA FUNCT1,FUNCT2,FUNCT3 ... ,FUNCT6
```

Now the branching can be done like this:

```
            AI    R8,->31   Remove number mask plus 1
            JLT   OUTRNG    If result less than zero, jump
            CI    R8,5      Is number greater than 5?
            JGT   OUTRNG    If so, jump
            SLA   R8,1      Else double the number
            MOV   LUT(R8),R5 Put selected address in R5
            B     *R5       Branch to the address in R5
OUTRNG      (IGNORE THE KEYSTROKE)
```

This takes many fewer bytes than the code shown above as the wrong way. We'll leave calculation of how many bytes fewer as an exercise for the student. On a casual first look, the twelve bytes used by the lookup table might seem wasteful, but overall we have a significant saving by "spending" those twelve bytes. This is similar to an Extended Basic situation in which a chain of IF THEN statements is replaced by an ON GOTO. That saves both bytes and execution time in XB, just as this "right" way does in Assembly.

This method will work very nicely when there's only one menu in your program. See the source code given in Part 3 for an efficient way to

handle more than one menu.

One more method of memory saving we should mention, and that's what we'll call recycling. (Recycling is a fashionable term nowadays.) A small example or two should give you the idea. In our Golf Score Analyzer, we have a part of the data section of our code devoted to the copyright notice. It looks like this:

```
CPYRT   BYTE 14       length of first line
        TEXT 'Copyright 1991'
        BYTE 17       length of second line
        TEXT 'Harrison Software'
```

Altogether that takes up 33 bytes. It's used only once, at the very beginning of the program, then becomes wasted memory space. At later stages of the program, we needed an area of 56 bytes length to store a temporary record of a round of golf. Ordinarily one would give that a label and a BSS like:

```
TEMREC BSS 56
```

But by placing that label just before the copyright notice, we could make it:

```
TEMREC BSS 23
```

Thus the rest of the 56 bytes in TEMREC overwrites the copyright notice, which we don't need anymore.

In a pinch, we could also use some of the area filled by the code that places that copyright notice on the screen as data storage. We didn't do that in this instance, but we did recycle the area in memory where the Extended Basic LOAD program is loaded to store user data about courses played. Thus when our program ends, the original Extended Basic program which got our Assembly program going has been destroyed. (When exiting our program, we take steps to insure that Extended Basic will "know" that it has no program in memory.)

One final option for dealing with the shortage of memory is the use of program overlays, where a new section of program is brought in from disk and written over an existing section of code or data. We resorted to that method for some utility features in our Word Processor, but we'll save that rather complex topic for a later article in this series.

We hope these few examples will serve to inspire you in finding ways to save memory in your own programs. In our next article, we plan to include more subroutines that will be directly usable in your programs.

~~~~~~
~~~~~~

PROGRAMMING MUSIC THE EASY WAY

MUSIC

PART 5 AND FINAL

by Jim Peterson

In previous installments I have shown you how to program music by an easy method which requires you to specify a duration or a frequency only when it changes from one note to the next. Now, here is an even easier method - auto-chording.

With this method, you do not have to key in the accompaniment - you just specify the chord and GOSUB to the proper line to play the type of chord.

Almost all sheet music has guitar chords printed above the upper staff - those little 6x4 grids with black dots on them. And those guitar chords are al- ways labeled with the name of the chord they represent.

The most common chord is a major chord, represented by a letter - A, C or whatever, or a letter followed by a flat or sharp sign. For those, use GOSUB 1000. The second most common chord is the 7th chord, which has the letter followed by a 7, such as C7. For those, GOSUB 1100.

You might come across a minor chord, denoted by a small m after the letter, such as Cm. In that case, GOSUB 1200.
And for a minor 7th, such as Cm7, GOSUB
1300.

There are many more complex chords, but I have not tried to allow for them all in this easy method. If you come to one of them, just try playing on through with the previous chord - it will usually sound alright.

To program music in this way, use the scale that I showed you in Part 1, but you will probably have to set the star- ting frequency considerably higher than 110. Merge in one or the other of the following routines, then program the music just as I showed you before, but only A and B. Give A the number for the melody and B the number for the chord, then GOSUB to the proper line number for that type of chord. If the next note does not have a guitar chord above it, it is the same chord so you do not have to give B a value again, just GOSUB to the same line number.

Now, here is the first routine, to play simple harmony. Let me give you a tip to save you some time. When you are keying in a series of program lines which are all nearly the same, key in the first one, Enter it, then use FCTN 8 to bring it back to the screen. Use the editing keys to change the line num- ber and make other necessary changes, Enter it, use FCTN 8 to bring it back, etc.

```
110 D=3 :: V1=1 :: V2=9 :: V
3=9
1000 X=X+1+(X=4)*4 :: ON X G
OSUB 1010,1020,1030,1040 ::
RETURN
1010 FOR J=1 TO T*D :: CALL
SOUND(-999,N(A),V1,N(B),V2,N
(B)/1.585,V3):: NEXT J :: RE
TURN
1020 FOR J=1 TO T*D :: CALL
SOUND(-999,N(A),V1,N(B),V2,N
(B)/1.334,V3):: NEXT J :: RE
```

```
1030 FOR J=1 TO T*D :: CALL
SOUND(-999,N(A),V1,N(B),V2,N
(B)/2,V3):: NEXT J :: RETURN
1040 FOR J=1 TO T*D :: CALL
SOUND(-999,N(A),V1,N(B)/1.58
5,V2,N(B)/1.334,V3):: NEXT J
 :: RETURN
1100 X=X+1+(X=9)*4 :: ON X G
OSUB 1110,1120,1130,1140 ::
RETURN
1110 FOR J=1 TO T*D :: CALL
SOUND(-999,N(A),V1,N(B)/1.49
7,V2,N(B)/1.585,V3):: NEXT J
 :: RETURN
```

```
1120 FOR J=1 TO T*D :: CALL
SOUND(-999,N(A),V1,N(B)/1.49
7,V2,N(B)/1.334,V3):: NEXT J
 :: RETURN
1130 FOR J=1 TO T*D :: CALL
SOUND(-999,N(A),V1,N(B)/1.49
7,V2,N(B)/2,V3):: NEXT J ::
RETURN
1140 FOR J=1 TO T*D :: CALL
SOUND(-999,N(A),V1,N(B)/1.58
5,V2,N(B)/1.334,V3):: NEXT J
 :: RETURN
1200 X=X+1+(X=4)*4 :: ON X G
OSUB 1210,1220,1230,1240 ::
RETURN
1210 FOR J=1 TO T*D :: CALL
SOUND(-999,N(A),V1,N(B),V2,N
(B)/1.679,V3):: NEXT J :: RE
TURN
1220 FOR J=1 TO T*D :: CALL
SOUND(-999,N(A),V1,N(B),V2,N
(B)/1.334,V3):: NEXT J :: RE
TURN
1230 FOR J=1 TO T*D :: CALL
SOUND(-999,N(A),V1,N(B),V2,N
(B)/2,V3):: NEXT J :: RETURN
1240 FOR J=1 TO T*D :: CALL
SOUND(-999,N(A),V1,N(B)/1.67
9,V2,N(B)/1.334,V3):: NEXT J
 :: RETURN
1300 X=X+1+(X=4)*4 :: ON X G
OSUB 1310,1320,1330,1340 ::
RETURN
1310 FOR J=1 TO T*D :: CALL
SOUND(-999,N(A),V1,N(B)/1.49
7,V2,N(B)/1.679,V3):: NEXT J
 :: RETURN
1320 FOR J=1 TO T*D :: CALL
SOUND(-999,N(A),V1,N(B)/1.49
7,V2,N(B)/1.334,V3):: NEXT J
 :: RETURN
1330 FOR J=1 TO T*D :: CALL
SOUND(-999,N(A),V1,N(B)/1.49
7,V2,N(B)/2,V3):: NEXT J ::
RETURN
1340 FOR J=1 TO T*D :: CALL
SOUND(-999,N(A),V1,N(B)/1.67
9,V2,N(B)/1.334,V3):: NEXT J
 :: RETURN
```

That routine will play straight 3-part harmony, but I like this one better, although it does not work well with some pieces.

```
110 D=30 :: S=1 :: V1=1 :: V
2=5 :: V3=7
1000 FOR J=1 TO T :: X=X+1+(
X=4)*4 :: ON X GOSUB 1010,10
20,1030,1040 :: GOSUB 2000 :
: NEXT J :: RETURN
1010 CALL SOUND(-999,N(A),V1
,N(A)*1.01,V1,N(B),V3):: RET
URN
1020 CALL SOUND(-999,N(A),V1
,N(A)*1.01,V1,N(B)/1.585,V3)
:: RETURN
1030 CALL SOUND(-999,N(A),V1
,N(A)*1.01,V1,N(B)/1.334,V3)
:: RETURN
1040 CALL SOUND(-999,N(A),V1
,N(A)*1.01,V1,N(B)/2,V3):: R
ETURN
1100 FOR J=1 TO T :: X=X+1+(
X=4)*4 :: ON X GOSUB 1110,11
20,1130,1140 :: GOSUB 2000 :
: NEXT J :: RETURN
1110 CALL SOUND(-999,N(A),V1
,N(A)*1.01,V1,N(B),V3):: RET
URN
1120 CALL SOUND(-999,N(A),V1
,N(A)*1.01,V1,N(B)/1.585,V3)
:: RETURN
1130 CALL SOUND(-999,N(A),V1
,N(A)*1.01,V1,N(B)/1.334,V3)
:: RETURN
1140 CALL SOUND(-999,N(A),V1
,N(A)*1.01,V1,N(B)/1.497,V3)
:: RETURN
1200 FOR J=1 TO T :: X=X+1+(
X=4)*4 :: ON X GOSUB 1110,11
20,1130,1140 :: GOSUB 2000 :
: NEXT J :: RETURN
1210 CALL SOUND(-999,N(A),V1
,N(A)*1.01,V1,N(B),V3):: RET
URN
1220 CALL SOUND(-999,N(A),V1
,N(A)*1.01,V1,N(B)/1.679,V3)
:: RETURN
1230 CALL SOUND(-999,N(A),V1
,N(A)*1.01,V1,N(B)/1.334,V3)
:: RETURN
1240 CALL SOUND(-999,N(A),V1
,N(A)*1.01,V1,N(B)/2,V3):: R
ETURN
1300 FOR J=1 TO T :: X=X+1+(
X=4)*4 :: ON X GOSUB 1110,11
20,1130,1140 :: GOSUB 2000 :
: NEXT J :: RETURN
1310 CALL SOUND(-999,N(A),V1
,N(A)*1.01,V1,N(B),V3):: RET
URN ! continued->
```

```
1320 CALL SOUND(-999,N(A),V1
,N(A)*1.01,V1,N(B)/1.679,V3)
:: RETURN
1330 CALL SOUND(-999,N(A),V1
,N(A)*1.01,V1,N(B)/1.334,V3)
:: RETURN
1340 CALL SOUND(-999,N(A),V1
,N(A)*1.01,V1,N(B)/1.497,V3)
:: RETURN
2000 FOR Y=1 TO D :: NEXT Y
:: RETURN
```

Both of those routines cycle through four inversions of the chord, to avoid a monotonous drone.

There are many ways to vary those rou- tines. Just for instance, right after each N(B) put *2 to raise the harmony above the melody. Also try *4. Or alternate *2 and *4. Experiment! Have fun!

==========

INSTRUCTIONS AND HINTS
FOR TI-WRITER WORD PROCESSOR
by Dick Altman Part Two

Dick Altman: TI Writer Part Two

Reminders:
1. Dick uses the Formatter option to print out!
2. TI Writer clones operate just about the same eg Funlweb etc.

When you wish to reload a file from a disk back into the word processor, it's EASY! When you first bring up the word processor in the Editor mode, you are automatically in the command line. Just type LF (for Load File) and hit ENTER, then type in DSK1.(and the name you gave it) then hit ENTER again and wait a few seconds for the work to be loaded into your computer from the disk.

In the book you will find two methods of printing your work- as is direct from the Editor, or saving to disk and then printing using the Formatter.

I prefer printing from the disk, not from the computer. You will find a command of 'Print File'. That's not the one I use! The one I have become accustomed to using may take a few seconds longer, but it is the one I learned first, and I have just stuck with it. It is as follows.

After I have finished typing my letter or whatever, return to the command line with FCTN 9, there type a Q (for Quit) hit ENTER, then S (for Save) and ENTER, then DSK1.TERRY or whatever name I want to give the file instead of TERRY, then ENTER. I usually use a short two or three character name. I have even been known to use #1, or #2, or something like that (the file name cannot be more than 10 characters long, and you can't have any spaces in a file name). Then, after the work goes from the computer to the disk, you can either print it now or sometime next week.

The command to go to the printer at this point is like this: Q (for Quit) ENTER, then E (for Exit) and ENTER again. This takes you back to the master menu. This time, you select #2, or THE FORMATTER. After it comes up, you have to type in DSK1.(filename) and hit ENTER [the clone programs vary somewhat here!]

Then you have to type in the command telling it to go from the disk to the printer, instead of to the screen. Without knowing what kind of printer you have, I can't give exactly the correct command here, but it will be something like this: PIO.LF or RS232.BA=4800.LF.

Then you will have five more choices, mostly for which you will just press ENTER for each of them. Perhaps you might wish more than one copy, so on the correct one you would punch in that number. Be sure your printer is turned 'on' before hitting the last ENTER,(the one that says "PAUSE AT END OF THE PAGE?) because you will be printing immediately.

For your purposes (manuscript writing) you may want it double spaced. That is simply a dot command of '.LS 2' (LS for Line Spacing of course!) and if you want it triple spaced, just change the 2 to a 3.

There are many, many more commands available, such as merging either parts of two different files, or merging a whole file into the middle of another, or putting in headers at the tops of every page, and footers at the bottom, all automatically. Such things as page numbers, or requirements for manuscripts, etc., but those can be found as you need em.

[more articles on TI Writer in following issues].

The word processor does have a capacity beyond which you have to save your work to disk, and start with a clean slate. It is approximately 20,000 characters including blanks. I have only run into it when transferring a long story to disk. I was entering a 10,000 word story, and I got 'MEMORY FULL-SAVE OR PURGE' flashing at me at the top or command line after about 4,000 words (I wish it would ring a bell or something). At that point 'save' your work and retire that file name.

You will just have to trial and error it for your job! Of course, the length canNOT be judged just by the line numbers on the left side of your screen. Think about whether you are using only one window, or two, or the maximum of three. I am using just one window while I do this work, as I explained earlier, so that will make my capacity come much farther down the line numbers than if I were using all three windows! 80 characters (or columns) wide, instead of the 37 I am using. If and when the MEMORY FULL bit happens to you, remember that when you save it this time to a disk, then for pete's sake don't save the next time to the same file name!

There is also available in 'FREEWARE'
circles an excellent disk set called "FUNLWEB" which was done by Tony and Will McGovern in Australia. It replaces the need for a cartridge to have TI-WRITER word processing capabilities. As far as I can tell, it does exactly the same things the cartridge does and much, much faster! It runs in Extended Basic and contains excellent utility programs.

The FORMATTER command for the underscore is merely the ampersand (Shift 7) and it can be used anywhere. If you want to underline more than one word you have to connect them with what is called a caret. It is above the 6, or Shift 6. If you wish, the AMPERSAND can be printed in your work, but not the caret. Merely type in two ampersands and only one of them will be printed!

CONTROL COMMANDS

ASCII CODES	FUNCTION	FORMAT
0	Terminate Tabulation	CTRL U, SHIFT 2, CTRL U
7	Sound the buzzer	CTRL U, SHIFT G, CTRL U
8	Backspace	CTRL U, SHIFT H, CTRL U
9	Horizontal tabulation	CTRL U, SHIFT I, CTRL U
10	Line feed	CTRL U, SHIFT J, CTRL U
11	Vertical tabulation	CTRL U, SHIFT K, CTRL U
12	Form feed	CTRL U, SHIFT L, CTRL U
13	Carriage return	CTRL U, SHIFT M, CTRL U
14	Print enlarged characters	CTRL U, SHIFT N, CTRL U
15	Print condensed characters	CTRL U, SHIFT O, CTRL U
17	Select printer	CTRL U, SHIFT Q, CTRL U
18	Turn off condensed printing	CTRL U, SHIFT R, CTRL U
19	Disable printer	CTRL U, SHIFT S, CTRL U
20	Turn off enlarged printing	CTRL U, SHIFT T, CTRL U
27	Escape	CTRL U, FCTN R, CTRL U
27;48	Set line spacing 8 per inch	CTRL U, FCTN R, CTRL U, 0
27;50	Set line spacing 6 per inch	CTRL U, FCTN R, CTRL U, 2
27;51	Set line spacing n/216 per inch	CTRL U, FCTN R, CTRL U, 3,n
27;52	Turn Italic Character set on	CTRL U, FCTN R, CTRL U, 4
27;53	Turn Italic Character set off	CTRL U, FCTN R, CTRL U, 5
27;56	Disable paper-end detector	CTRL U, FCTN R, CTRL U, 8
27;57	Select paper-end detector	CTRL U, FCTN R, CTRL U, 9
27;65	Set line spacing(1/72 to 85/72 inch)	CTRL U, FCTN R, CTRL U, A,n
27;66..	Set up 8 vertical tab pos.	CTRL U, FCTN R, CTRL U, B ++
27;67	Set form length up to 127 lines	CTRL U, FCTN R, CTRL U, C,n
27;68..	Set up to 12 horizontal tab positions	CTRL U, FCTN R, CTRL U, D ++
27;69	Turn on emphasized printing	CTRL U, FCTN R, CTRL U, E
27;70	Turn off emphasized printing	CTRL U, FCTN R, CTRL U, F
27;71	Turn on double printing	CTRL U, FCTN R, CTRL U, G
27;72	Turn off double printing	CTRL U, FCTN R, CTRL U, H
27;75	Turn on normal density graphic printing	CTRL U, FCTN R, CTRL U, K
27;76	Turn on dual density graphic printing	CTRL U, FCTN R, CTRL U, L
27;77	Turn Elite mode ON	CTRL U, FCTN R, CTRL U, M
27;78	Set skip-over perforation	CTRL U, FCTN R, CTRL U, N
27;79	Release skip-over perforations	CTRL U, FCTN R, CTRL U, O
27;80	Turn Elite mode OFF	CTRL U, FCTN R, CTRL U, P
27;81	Set a column width	CTRL U, FCTN R, CTRL U, Q
27;82;n	Select 1 of 8 int'l char.sets	CTRL U, FCTN R, CTRL U, Rn

[ESC]R1 will set CHR$(6) to a pound sign on an Epson FX80.

SEE YOUR PRINTER MANUAL FOR MORE!!!!!!!!

FROM THE CHAIRMANS CHAIR

By T. STEVENS (c)1993

Well here we are again with another addition of TI*MES. As you all know I took over the Editors job from Alan Bailey some three issues ago. Alan I am sad to report, has since died of his illness. I have had many long talks with his wife and it would appear that he died as result of stomach cancer. I hereby on behalf of all members of the group send the family our condolences. I also again say that all his hard work towards the group will be greatly missed. His wife also expresses that if you wish to send something could you please make a donation to Cancer Research, as this is what Alan would have wanted.

Having started on a low note I now move onto something a little more light hearted. I am sure Alan would have agreed that life has to go on. So I start with a bit of sillyness really. I am no chess player. I know how the game works and have played (badly I might add) several times. If you have the VIDEO CHESS game you will know that you can play a game whereby you can play both sides of the board and also set up a board. Well I was watching Channel 4 and the coverage of the Masters game, when it struck me that I could set up the game that was in progress and let the computer do some work. I set up the game board from the TV. caught up with the moves and Short was to play. Our TI was working at the high level selection and it started to calculate. Would believe our computer worked out a far better move and would have allowed him a better offensive and defensive situation than the one he played. On the discussion the so called on line super computer did'nt even come close to the move the Texas worked out. It only became apparent when the experts discussed it and one of them came out with the same move. However before this move was fully discussed Short made his move and the game was only eight moves from defeat. So your little TI really does know how to crunch the old numbers and come out with some really good results. The next test is to run some other programs against ours. I will let you know what happens.

This quarter has been really busy for me so I have not really got my act together with regard to my article. So I am not this quarter, going to do the Sprite tutorial. This will I promise return next quarter. However I have been asked several times about RAMDISKS and what are they. Several of our members seem keen on the idea but wonder what it is all about. I will be discussing my Ram Disk, a MYARC, as that is the one I own.

What is a RAM DISK. Well put in a simple term, it is a electronic disk drive, like your five and a quarter disk drive. However being constructed as a electronic card it has no moving parts, unlike your mechanical drive. The RAM DISK has banks of memory inside which depending on the setup or memory can be formatted as conventional drive. This means that you can use it with all the file formats that are available to the disk user.

What advantages does it have you may ask?
Well it has quite a few really. In the first instance as discussed above, it gives you an extra drive. However as you may or may not know the TI Disk Controller can only handle 3 disk drives. With a ram disk it does not require a disk controller to use it. The on board eprom which is in the card contains software to handle RAM DISK access. This, in effect gives your computer 4 drives. The Ram Disk also gives you MEGA SPEED!!! Due to the non moving parts its access time is faster than a Hard Drive. To give you an idea of the speed here are the load times for TI BASE. This loads numerous files (340 Sectors worth). From hitting the Extended Basic selection key to date prompt it takes the following times. Standard Disk (1 Min 4.9 Secs) RamDisk (14 seconds). If you load a standard program it just blinks and its in. Also with the MYARC card you also have on board, as standard a print spooler. This allows you to pass text data to the Ram Disk and it will store it in the memory to feed to your printer. So if you have a long document, that will not fit into your printers memory, then it will hold onto it and feed your printer when it requires it. Doing this releases your computer from PIO tasks and allows you to do something else. That's real multitasking.

What disadvantages does it have?
Well not that many really. You do have to load your ram disk with the files that you want to use. However you can battery back the card as I do and this will keep you information as long as there is power. On my system I have a small home made transformer and 9 volt NI-CAD which is left on. This plugs into the rear of the card with a jack plug. RAM DISKS however are not for long term storage of data and have to be backed up by floppy disks or hard drives. They are sometimes prone to data corruption but this is very rare. You also sometimes come across a program that clashes with the RAM disks and when accessing the RAM drive the program looks at the wrong DSR location for the now disabled drive and throws a benny. This is not the fault of the RAM DISK but the programmer. This I might add is fortunately few and far between.

The MYARC RAM DISK plugs into the expansion box. When fitted this in effect creats as a three fold function card. First it is a full 32K card. So when you get it you take out your TI 32k. Next is the Ram Disk and third as discussed the Print Spooler.

The card when bought could be unpopulated and just 32k. The next was a populated card with 128k of ram. The highest card is 512K or half a meg. I have the 512 card. When you get the card with 128 or 512 of Ram you can use the Ram Disk. You can decide on start up how you want the Ram Disk configured. This is done from Basic or Extended Basic with the CALL PART(x,y) command. What this does is to tell the Ram Card to Format your disk into Ram Disk and Print Spooler. For example with a 512 card you have available 480k of space. (32k used for the 32k replacement) With the 128 96k. With the 512 card the maximum partion available is 400k (x) for Ram Disk and 80k (y) for Print Spooler. This equates to 1600 sectors of Disk space. As usual 2 Sectors are used for shop keeping as with your floppy. This now leaves you 1598 sectors. Thats more than two 720 sector disks or nearly 5 single sided 360 sector floppies. I think that is enough for me. The next thing you can do is decide what

drive number you want your Ram Disk to run as. This uses the command CALL EMDK(Drive No). ie CALL EMDK(1) will make your Ram Disk emulate drive 1. This will not clash with your drive 1 as it over rides your drive 1 and makes it unusable. So if you wish you could say load an Extended game in the ram disk, and have another Extended game in Floppy drive one. If both had the LOAD file on them then you can boot up from Load on either by switching the EMDK number. The print spooler can also be moved about. You can if you wish access the following:- RS 232/1, RS 232/2 or PIO
These are the commands :- SP,SP/1,SP/2 or SPPIO. If you want to abort from this in basic you just enter Call ABPS.

The Ram Disk will also except a disk name and use it by using the CALL VOL("name"). This ables you to use programs such as TI ARTIST that looks for disk names instead of drive numbers. I use this feature a lot as TI ARTIST is always loading modules. So with ram disk it only takes a second to load and saves time.

Also you can look at your Ram Disk to see what is on it without having to go through the task of loading another program such as DM1000 or similar. All you do is issue the command CALL RDDIR. This does a complete DIRectory of you Ram Disk.

All the usual Floppy disk commands are available such as Delete, Restore, Input etc.

There is also another plus which can be very useful if you are an Assembly Language programmer. Instead of using the Ram Disk feature you can use the memory available to you to bank switch. This means that you can load up four banks of memory with program information. By running your main program and using CRU bits >1002 and >1004 you can bring on line the relavent bank for use when required. This means you can have very large programs.

As you can see there are lots of things that you can do with a Ram Disk. I find mine very useful and to be honest I would not be without it. The MYARC can still be obtained second hand but are few and far between. However there are still other makes which are available. The best known is Horizon. This is still being sold by Bud Mills Services 166 Dartmouth Drive, Toledo OH 43614-2911
The basic model which is unpopulated is $125. With memory available up to 8 Megs. However this is very expensive. To give you a price for my card (512) it would cost 325 dollars. However there are firms about still selling second hand stuff. I have seen 192k Horizons for sale for 192 dollars. If you are interested then drop me a line and I can provide information on this type of source

Well that's it folks 'til next time. If any of you want a special article on anything you are not sure about then drop me a line. I will then try to get some information collated and brought to everyone in the magazine.

Harrison Time Calculator

NEW TO THE DISK LIBRARY - TIME CALCULATOR from Bruce Harrison.

Time Calculator Instructions

The Harrison Time Calculator is an Extended Basic program with built-in Assembly enhancement. Its purpose is to handle calculating numbers in Hours, Minutes, and Seconds. The time inputs may be made in either the "normal" 12 hour clock format or in the "military" 24 hour format.

The program is called TIMECAL. At startup, the program simply puts a menu on the screen. Six items are on the menu, so selection requires only one keypress on the keys 1 through 6. The six selections are:

1. ELAPSED TIME
2. CUMULATIVE SUM
3. TIME MULTIPLY
4. TIME DIVIDE
5. SET 12 OR 24
6. EXIT PROGRAM

The first selection, Elapsed Time, is for cases like "how much time is it from 10:22:35 to 3:30:46?" In elapsed time mode, you simply answer the prompts for Start time and Stop time, and the program reports the correct time interval as "Elapsed Time". This will correctly "roll over" either the 12 or 24 hour limits, provided you've set the time standard correctly. The hours limit defaults to 12, so that "Civilian" time inputs will be accepted and handled correctly. If you want to set up for a 24 hour clock, instead of 12, use selection 5 from the menu.

Item 2 on the menu is for those cases such as trying to copy records or CDs onto cassette tapes. Often, the timings of the selections are shown on the label or jacket, in minutes and seconds. This function will allow you to keep a running total as you enter times for the various selections, so you'll know when you've reached some limit, like the 30 or 45 minutes on one side of a cassette.

In this mode, you'll get a prompt that says "INPUT A TIME". You would put in the minutes and seconds for the first number to be taped. When you press "ENTER" with a time in place, another prompt will appear that says "PLUS OR MINUS (P/M or +/-)". This prompt requires a single keypress of the "P" key or the "M" key or the "+" key or "-" key. In most cases, you'll be adding, so P will be the most often used. (In fact, the program at this point defaults to "add" for any keypress other than M or m or -, so that you can just press the space bar at this prompt to add the new value to the current total.) The minus situation is provided for the case where your running total has gone over the allowed limit, and you want to "take out" one or more numbers to get within the allowed time.

This mode will not allow the cumulative time to go below zero. If you attempt to subtract more than the current running total, you'll get an error message. (Star Trek may be able to handle negative time, but our program can't.)

After you've pressed P or M or + or -, the program will calculate and display the cumulative sum, and prompt "KEEP GOING? (Y/N)". Pressing Y or y will allow you to continue this cumulative sum, while N or n will take you back to the first menu.

Item 3 on the menu is designed for the old "cook-book" problem, where you find that the recipe says to roast the lamb at 350 degrees for 25 minutes per pound. The leg you're roasting weighs 3.78 pounds, according to the label, so how much time is 25 minutes times 3.78?

When you select item three, you'll be able to answer that question exactly. At the first prompt, you'd enter 00 hours, 25 minutes, 00 seconds. At the multiplier prompt you'd answer 3.78. The program will then quickly tell you to the second how long to roast that leg. (1 Hour, 34 Minutes, 30 seconds) Of course you'll want to "manually" round off the seconds, but having the hours and minutes correct will be close enough even for Julia Child.

Item 4 on the menu is there mainly just to complete the functions. We really don't know what you'll do with this function, except perhaps if you wanted to correctly divide an eight-hour workday into 12 equal time periods, (40 minutes each) or some other insane thing. We'll leave to your own imagination how to make use of this function, but it's there for you as part of the package.

Item 5 is there to change the hours limit. A very short menu will be presented. Press 2 to set a 24 hour limit, or 1 to select the "default" 12 hour limit. This can be changed back and forth whenever you like. To exit back to the main menu without changing the time standard, press Function-9.

The TIME Input Field

One of the special features of this program is the built-in Assembly routine GETTM, which provides the special input field for the Hours, Minutes, and Seconds. The field is divided by colons between the two places for each sub-field. The cursor will skip over these colons, so you don't have to type in anything but numbers or spaces. Let's walk through an example to show how this special input field works. Suppose you wanted to enter 1 hour, 45 minutes, no seconds. The cursor starts out in the HOURS sub-field, so you could type either 1 and a space or a space and 1 in that sub-field, or you could type 1 and Function-D, or Function-D and 1. When you've done any of those four things, the cursor will be in the MINUTES sub-field, so you'd type 45, and the cursor will skip to the SECONDS sub-field. Since we said seconds was zero, you'd just leave that sub-field blank and press ENTER. The program will now write over the field with "01:45:00".

A single-digit entry for any of the sub-fields can be placed in either the left or right position in that field, and will be correctly interpreted if the other position is blank. Before pressing ENTER, you can move the cursor left and right with the Function-S and Function-D keys. The cursor will skip over the colons, and will not go beyond the limits of the field.

Illegal entries will be rejected after ENTER has been pressed, and the field will be cleared. The MINUTES or SECONDS sub-fields will not accept numbers higher than 59. The HOURS sub-field defaults to 12 as its maximum allowed, but this can be changed to a 24 hour limit through menu selection.

For those adventurous souls who want to use the Time Input field in their own XB programs, we have provided the object file, with ALSAVE and ALLOADM as well, so you can "submerge" the routine with your own program. For those even more adventurous, we've included the source file, complete with annotation, so you can delve into how this little gem does its job, or you could even modify it!

The menu driver that's used with DATA statements in the XB program to provide "instant" menus on screen is also provided on the disk as the object file MENDRV, along with its source file MENDRVS. This can be used in your own programs as well.

That's it! The program disk has been released to Public Domain, and so may be freely copied, shared, and so on. We ask only that all copies include the complete contents of the disk, which includes these instructions, source code, object file, etc. Enjoy!

-THE PROGRAM IS AVAILABLE FROM THE DISK LIBRARY ON ONE SSSD DISK- ask for TIME CALCULATOR!

==========

Review: Ti-Pei

REVIEW by Stephen Shaw

TI-PEI a game for one by William Reiss.

It is nice to find a well written and enjoyable game program which is not so widely known.

TI-PEI loads from disk with Extended Basic. It is an interesting type of solitaire program very loosely based on Mah-Jongg tiles.

Given an oddly shaped pile of tiles you have to match pairs of tiles, which are removed, and end up with no tiles left.

You usually have a number of choices of which tiles to match, and if you remove tiles in the wrong sequence it can make further play impossible.

The documentation suggests the game may be five times harder than card-type solitaire (=patience) but I think I would dispute that- from my playing of the program it is either equal to or slightly easier than the usual 7-pile patience we usually play.

Documentation is perhaps the weak point, but the game can easily be picked up. Perhaps the hardest thing is to consider the very flat display is really a pile of tiles, as there is no attempt at a 3-D look. Each layer of tiles is colour coded, but I have been playing the game on a mono monitor with no great trouble.

The program is stated to run on a Geneve (in GPL mode), and with either a Myarc Mouse or an Asgard Mouse (NOT a Mechatronic mouse). It does not unfortunately recognise joystick input, which is a pity!

The tiles are presented in what looks like a random order, but if you get stuck you have the option of playing the last hand again! There are 144 tiles, comprised of 4 suites valued 1-9, so each tile could initially match with 3 others (unlikely though as the others are probably at the bottom of the pile).

If you like playing patience, this is certainly a program you will enjoy - all you need to do is locate someone reliable to sell it to you!

Stephen Shaw

==========

ARTICLE BY STEPHEN SHAW

I have previously quoted the puzzles from New Scientist magazine, many of which are very suitable for computer solution. This one is Enigma 725, set by Robert High in New Scientist of 3rd July 1993.

In the following diagram each letter stands for a digit. Each digit is represented only by one letter.

Example first with sample values:

```
       T=1   E=3   A=2
          A=2   T=1
```

In the second row, the values are derived by taking the differences between the two letters above, ignoring any minus sign, so that taking the first two letters T and E, the difference is 2 which is A. These values apply only to this example!!!

Now solve this:

```
   P   E   R   F   E   C   T
     P   U   Z   Z   L   E
```

I can tell you that T does not equal 1, and that there is only ONE solution. A quick Extended Basic program I threw togther here found the answer in 55 seconds and proved there was no other solution in 75 seconds. Can you beat that?

==========

```
1 REM  WITHOUT RUNNING THIS
2 REM   PROGRAM- WHAT DOES
3 REM   IT DO?
4 REM  CAN YOU DO IT MORE
5 REM  EFFICIENTLY?
6 REM
7 REM
8 REM  -A DEMONSTRATION TO
9 REM  SHOW THAT YOU CAN DO
10 REM  THINGS THE HARD WAY
11 REM        OR
12 REM  THE EASY WAY.
13 REM
14 REM  X AND Y TO BE >0
15 REM
100 INPUT "FIRST NUMBER:":X
101 X=ABS(X)
110 INPUT "SECOND NUMBER:":Y
111 Y=ABS(Y)
120 Z=0
130 IF X/2=INT(X/2)THEN 150
140 Z=Z+Y
150 X=INT(X/2)
160 Y=2*Y
161 PRINT X;
170 IF X<>0 THEN 130
180 PRINT Z
190 END
```

==========

HARRISON BASIC COMPILER

COMPILER... a report by Stephen Shaw

I have right now a true compiler which is capable of producing code for the following extended basic program:

```
100 B=48.3
110 A$="THIS WAS A VERY LONG TEST"
120 S=1 :: R=20 :: T=4
130 FOR I=-1 TO -R STEP -S :: FOR J=2 TO 30 STEP T
140 GOSUB 190
150 NEXT J :: NEXT I
160 FOR K=10 TO 1000 STEP 20 :: PRINT K :: NEXT K
170 PRINT B;A$
180 DISPLAY AT(12,7)BEEP:"TEST COMPLETE" :: STOP
190 PRINT I;J :: RETURN
```

This program demonstrates the ONLY parts of the new compiler which at this moment works, further extensions are to come.

The only extended basic function which is ruled out at present is the use of user sub programs (eg CALL MYPROG).

PRINT uses a lot of processor time and the compiled program runs about twice as fast - if PRINT is removed, the compiled program runs about six times faster.

When I say "true compiler" I mean a program which takes an ordinary extended basic program as input (in merge format) and outputs assembly source code for an assembler to work upon. By additional processes, the output is made into a "hidden code" extended basic program- which could load from cassette if the 32k ram was present.

Here are the steps:
1. Write your XB program and SAVE it to DISK in MERGE (DV163) format.
2. Load and run the compiler, which will ask you for the name of your saved XB program and ask for an output file name.
3. The compiler will create disk files: a. Assembly source code for assembling; b. Assembly data file which will be accessed while assembling. c. A merge format extended basic program to have the object code inserted into it.
4. Assemble the source code using Art Greens assembler. This takes as input the source file, accesses the data file, and various supplied "library" files, and outputs machine code object code.
5. Load Harry Wilhelms extended basic program HML (supplied) which takes as input the object file. Merge in the shell file and you are left with what looks like an odd extended basic program, which runs quite quickly.

At the time of writing supported features are: Variable equates (S=1); FOR-NEXT-STEP (slightly limited at present); PRINT (numeric variables only at present); GOSUB; GOTO; END; STOP. A limited form of IF...THEN, which like basic can at present only jump to a line number, and no ELSE right now.

At present there is no limit to the number of FOR-NEXT loops you can stack.

Perhaps a little limited right now but a good start.

The compiler is being put together by Bruce Harrison. Watch for more news.

TIPPSY

The tips for the quarter are below. However no one has sent in any tips for publication. Come on out there in TI Land, speak up and lets us hear your tips.

1. This tip in fact takes advantage of a very rare bug in the TI Extended Rom chip. How would you like to use your TI with the screen and text colours of your choice, which survives a run break.
As you may or may not know, when you break a running program in Extended Basic everthing is reset including the colours. However if you break a program when it is in the middle of an ACCEPT AT statement then it will not reset everything. Try this as an experiment:

```
FOR I=0 TO 14 :: CALL COLOR(I,13,1) :: NEXT I :: CALL
SCREEN(4):: ACCEPT AT(1,1):A$
```

Do not use a line number just enter this lot in command mode. You will notice that everything stays the same. It will however bomb if the TI comes up with an error warning. All you do is retype the line.

2. This one is for Editor Assembler owners with Disks. Did you know that if you have a option 5 Loading program with the name UTIL1 then you do not have type in the name. All you do is treat it as you would a TI Extended Load file. That means turn on your machine, select Editor assembler. Go through to the Load section. Select OPTION 5 and then press enter twice. Bingo your program loads.

Item 3 was incorrect and could cause hardware damage.

4. Disk drive resistor packs when fitting two half heights and one external full height. You will require a pack on drive 2 and also on drive 3 as the two TI card sockets input data through two seperate path connectors.

5. TI Writer tip. Did you know that if you wish to force a page break in a document you can use a CTRL P or CTRL 9 to in bed your command this then can be used by printing via the PF (Print File) command or through the formatter.

Well that's it for this time, *** MEMORY FULL ***!!!!!