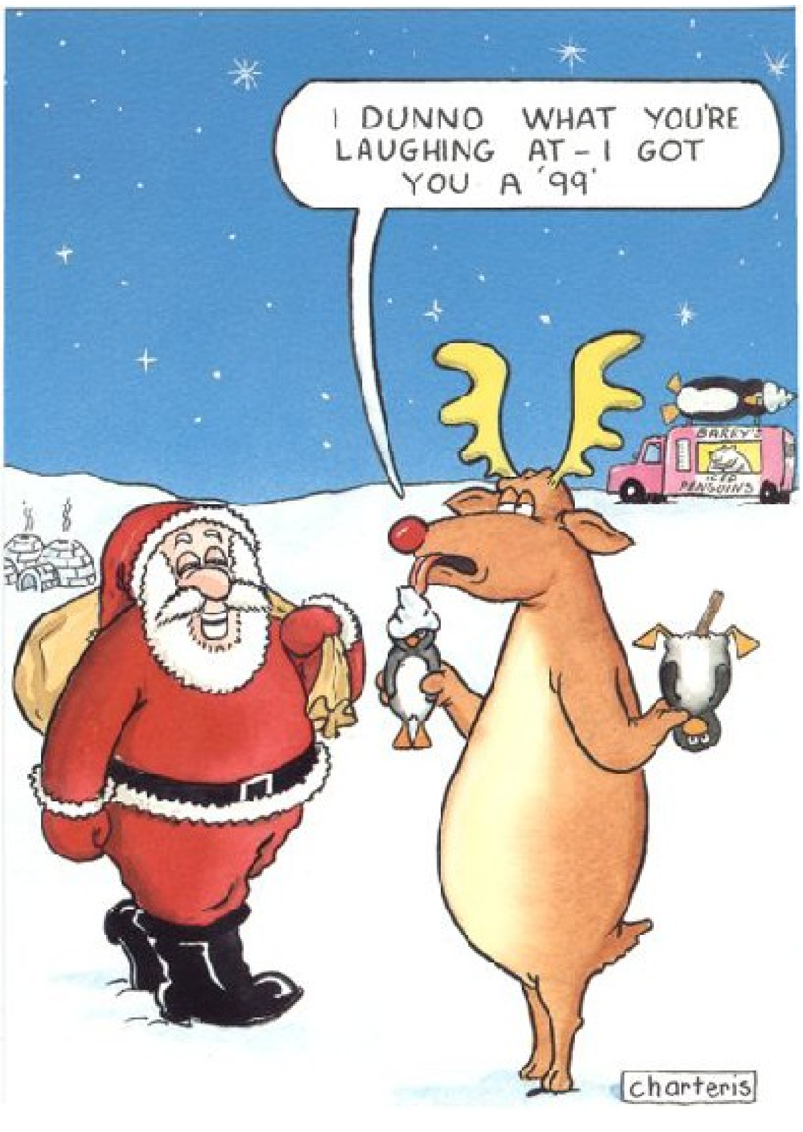
TI*MES

Issue 91

Winter 2005-2006

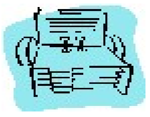
Stanton St. John 2 day workshop 4th and 5th March 2006 at the Stanton St. John Village Hall, Oxfordshire.

YOU CAN NOW PAY FOR YOUR GROUP MEMBERSHIP ON LINE WITH PAYPAL!!!



Supporting the TI-99/4 and /4A, the MYARC GENEVE 9640, Michael Becker SGCPU card,
And any other compatible machine.

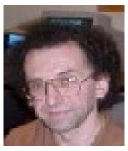
TI*MES index…



Committee Members

Chairman
Trevor Stevens.
249 Southwell Road East, Rainworth, Notts. NG21 0BN
Phone: 01623 406133
chairman@ti99ug.co.uk
SKYPE: trevorstevensmegatech

General Secretary - Richard Twyning
41 Vera Crescent, Rainworth, Notts. NG21 0EU
Phone: 07767 44 56 58
FAX: 07767 449 009
treasurer@ti99ug.co.uk
SKYPE: richardtwyning



Media Librarian (Disk / Cassette / Cartridge)
Francesco Lama
48 Mayfair Road
Cowley, Oxford. OX4 3SR
disklibrary@ti99ug.co.uk

Disclaimer

The views expressed in the articles in this magazine are those of the individual author, and not necessarily the view of the magazine or the group.

Email membership terms and conditions!

At this point we would like to give a warning to those who have subscribed with email membership. You have expressed this preference because you want your TI*MES magazine to be delivered by a more reliable medium than the Royal Mail! However, as you must realise, PC file formats these days are bloated beyond belief and the Microsoft Word file of a newsletter could be as big as 18 Megabytes!!!
PDF files do compress the file size down, but they may still be a considerable size!

Receiving large attachments these days is no big deal, as Yahoo.co.uk offer a free email service which gives you a maximum in-box of 100Megabytes!

If you specify an email address for your email membership then you ***MUST MAKE SURE*** that the email account has ***sufficient capacity*** to be able to receive these attachments!!!

It's not all doom and gloom though! Trevor and I have been pondering converting the magazine to HTML format and putting it on the web site so that people could read it online. Trevor already has the web code to allow us to make it password protected from non-signed-up visitors! We also need to make sure that it only uses bog-standard HTML and GIF or JPEG images so that it is available to a wider audience. Trevor has the habit of automatically using loads of flash and applets on his websites, but I think we need to make things a bit simpler so it can be accessed by a wider range of web browsers.

MAKE A SPACE IN YOUR DIARIES FOR 4th and 5th MARCH 2006 for the TWO DAY WORKSHOP AT STANTON ST. JOHN VILLAGE HALL, NEAR OXFORD. THERE WILL BE MORE ABOUT THIS AGAIN IN RICHARD TWYNING'S ARTICLE.

IMPORTANT INFORMATION

Membership Renewals

For those members who have not yet renewed their membership, membership renewals SHOULD ALL take place at the SAME TIME OF YEAR, in or around JUNE when we send the SUMMER ISSUES out. Some of you may have recently received a renewal notification because we've come up with an automated way of producing reminders, especially now that I work in the direct mail industry! ;-)

If you have access to the interweb, you are now able to pay your group membership directly from the user group website using your PayPal account.
For those who aren't in the know, PayPal is a web site that allows you to send and receive money internationally. It's excellent, and has recently be purchased by a little company called **ebay!**

The group's web address is **www.ti99ug.co.uk** then just click the "**join TI user group**" button!
While still on the subject of the website, the TI picture book has been updated. Now you can see pictures from the old NEC retro computer exhibition, and the recent TI-Treff in Venlo, Netherlands.
Also, soon to come, the TI*MES area will have a passworded zone for the download of some previous issues of TI*MES magazine.

Stanton St. John Workshop extra info

The date of the workshop is advertised as the 4th and 5th of March, but Trevor and I will be travelling down on the Friday night (the 3rd), as I'm sure I've mentioned we are camping in the village hall car park!
I don't think we'll actually be able to enter the room until the Saturday, but if you want to find a B&B in the area on the Friday night, you're welcome to come and join us for a meal in the local pub ;-)

AGM 2006

I've organised the 2006 AGM for the **Saturday 3rd of June**. It will take place again at the Station Hotel, Newstead Village, Nottinghamshire. This is the same place we organized it last year, and although it's called a hotel, they unfortunately don't do any accommodation, but it is a good pub and they do food. The reason I chose it is because it used to be the actual train station and Newstead train platform is right on it's doorstep! You can therefore get a train to the event directly from Nottingham via the Robin Hood Line. Directions to the Station Hotel will appear in TI*MES from the next quarter onwards.

FROM THE CHAIRMAN'S CHAIR BY TREVOR STEVENS

Well the first thing I must say is Merry Christmas and a happy new year to all our readers. Since I last wrote, things have been really silly here for me. I now have a new business up and running which is on the internet fixing computers on line. Things really have been taking off nicely. If you are interested the site is www.doctoremote.co.uk.
Well had no replies to my request for the win994a emulator Ram disks. However I have sorted this out myself. The disks are called rdsk1 rdsk2 and rdsk3. All became clear when I went to the 99er forums about Ram disk. Here are the extracts I have done…

To Cory Burr …
I like the fact that the simulator adds RDSK1,2,3 as these are much faster for file operations then the virtual floppies but they are for the most part pretty useless as they loose their contents when the simulator exits. One of the best peripherals ever created for the TI was the RAMDISK. What made it so powerful was the fact that it was so fast and it was battery backed up. In fact on starting on my current adventure with the simulator, my TI system had not been switched on for nearly 3 years and yet - the batteries kept the contents of bot my Horizon's alive and it just came back up as if it were only yesterday. What I'm getting at here is that it would be great if the ramdisks could be assigned to other logical names and especially great if the contents could be dumped to a file so they could be restored again the next time the simulator is started.

What do you think?
Michael

From Cory Burr…
I thought about this idea of saving the RAMDisks between Win994a runs back when I added them to the application. But then, I thought about what you would end up with, if you saved the RDSK...and that's simply another TI disk file. Of course, that's what DSK1,2,3 already does. So the benefit to make Win994a do that didn't really seem to substantiate the pain to make Win994a do it. I think this feature makes sense on the real TI because floppies are so slow but not on an emulator, where most of the time disk data is held in memory anyway. You'll find that DSK1,2,3 is just as fast as RDSK1,2,3. In fact, they both use the same code with a very slight mod...I think it was just two additional lines of code for RAM instead of hard drive.

I use RDSK1,2,3 to hold what I think the TI sadly lacks, and that's the concept of true TEMPORARY verses PERMANENT files. I know TI must have been thinking along these lines because they included these token words in their BASIC language, but they were never implemented. For me, it's great to have the E/A place my assembly listing files on one of the RAMDisks. I usually don't want it after I've looked at it, and I don't have to remember to do house cleaning. When Win994a terminates, all those files are gone along with everything else I placed there because I didn't want to keep them. If I need the listing again, I can always create it again, or print it out.

Leightym, after reading all your entries, I'm getting the impression that you're very interested in a lot of the non-standard, non-TI things that were/are available for the TI-99/4a. I guess you and I are a little at odds here because that wasn't the "spirit" I'm after with Win994a. I was mainly attempting to build a standard TI emulator, not a collection of third party addons...and I really have no interest there. I know this might cramp the style of some folks, and leave out some nice functionality that some people aren't willing to live without. I know it kind'a leaves a lot of what you're trying to do unsupported. But I hope you can understand and tolerate my position. Win994a is just a personal hobby to me, something I did to have fun. If others find it fun to use also, then great. But I have no plans to give Win994a every bell and whistle ever invented for it.

What I do have plans for is a real, true ANSI standard C compiler that has all the things C99 didn't include. This would be a PC-based compiler. If my plans come to fruition, it will have a "small-model" target that will produce code for both the real TI and Win994a. It will also have a "large-model" target that will produce code only Win994a can execute. The large-model will produce used TMS9940 op-codes that will implement a real on CPU stack mechanism, with op-codes to implement 16meg of virtual data memory pages, and 32 pages of 24k code segments. All this should allow for some very large TMS9940 programs...something TI should have implemented way back when. For me the past is the past. Win994a does well in reproducing that past. What I'd like to do, is to see what the TI would have done had it's family continually evolved like Intel's and Motorola's chips did.

Cory

Leightym
Thanks. I understand and will see about adapting to them as you said as a temp storage area. I like the idea of dropping assembly listings there as well. No need to keep them unless there is an error that needs to be looked into.

A real ANSI c99? That would be cool. I love assembly but c is faster for getting things to work. Frankly this has all been really weird for me as I haven't done ether in my real life for over a decade. After looking at assembly listings for the last couple of days it's starting to all come back and I'm amazed I actually remember what all the addresses are for.

About my interest in the advanced stuff, no worries. I'm not trying to twist your arm on any of them. You can take or leave any of my suggestions and or comments as just enthusiasm for your fine work It's just that when I used the TI as a real computer these things - especially 80 column display - were vital to the ongoing use of the TI as a viable workstation.

Speaking of cool advanced stuff, I still have a couple of 99110 chips that I swear I'm going to make use of some day. Your ANSI c compiler would make that so much easier to develop an updated boot kernel with. If your interested I could send you the data sheets with all the extra op codes listed hint hint - sorry, couldn't resist.

Thanks again for the help and the ear.
Michael

You will note that Cory Burr is the win994a programmer. When I read this, I saw as you have just read that he intends to write a C compiler!

This is the reply to one of the TI fans…

*You do this, and I will kiss your "a*se".*

I have been calling for this for a long time. Every time someone says "why can't we have <insert application here> on the TI", I always say "because we don't have a real C compiler". And instead of generating any impetus to create one, several say "well it wouldn't be that hard in assembly" and usually they are the clueless ones who don't know a lick, and of course the discussion goes no further.

Why should it be so hard to create the tables necessary to output 99xx code from an open source compiler? Maybe not such a behemoth as gcc has become, but what about lcc?

The last serious application I wrote in c99 was 15 years ago. And by the time it was finished (before I did optimization in assembly) at least 1/3 was assembly just because c99 didn't support long ints, unsigned ints, structures etc.

Not to mention that I had to "patch" the file I/O library "in-line" to support binary I/O and INTERNAL *format (not the BASIC INTERNAL, just setting the proper flag to allow open of a non-BASIC INTERNAL file). Along with a real fseek (to a byte, not a record) and an ungetc function.*

It would really be nice to have access to the thousands of ANSI C programs and compile them to 9900 object code or even assembly (although those 6 character labels are rather annoying as well, but at least there are 9900 cross-assemblers for the PC).

Cory Replied
Icc is exactly the open source compiler I'm looking at. It's front end seems to provide all the items and functionality I'm looking for. However, rewriting its backend to output 9940 code is not a trivial matter. I've got several books on how to do this. In a way, it's like teaching Icc to talk 9940. One of the real problems here is a serious, SERIOUS lack of available free memory space in the TI. It doesn't take much of a C program to fill it. In addition, all higher level languages need a fast (supported by CPU op-codes) stack to be efficient. Also, remember that a lot of what we've all come to know and love about C ISN'T in the ANSI C standard...things like fseek and all the f-functions for streaming I/O are add-ons and will have to be added as libraries. And all this extra code will continue to fill TI's meagre memory space, just to get to the point where the first programs can be compiled and tested. That's why I also envision a large-model, where Win994a is given extra machine op-codes to handle memory segmentation at the CPU level. What I'm thinking about is using seven unused bits (given only two bits are used for the interrupt number) in 9940's status register to indicate 128 code segment pages of 24k. In other words, the upper 24k of TI's memory would have virtual memory 128 times. What's the nice thing about using the existing status register to do this is how an RTWP instruction will not only return to the calling routine, but will switch back to the caller's code segment as well. Think about it.

So as you see if Cory does this then this will open up a real flood of new stuff for the TI!

Strange really, My TI is now 23years old and is still going strong. Though I only bring the unit out for high days and holidays I really do enjoy using it. I like to use the emulator however it is not quite the same.

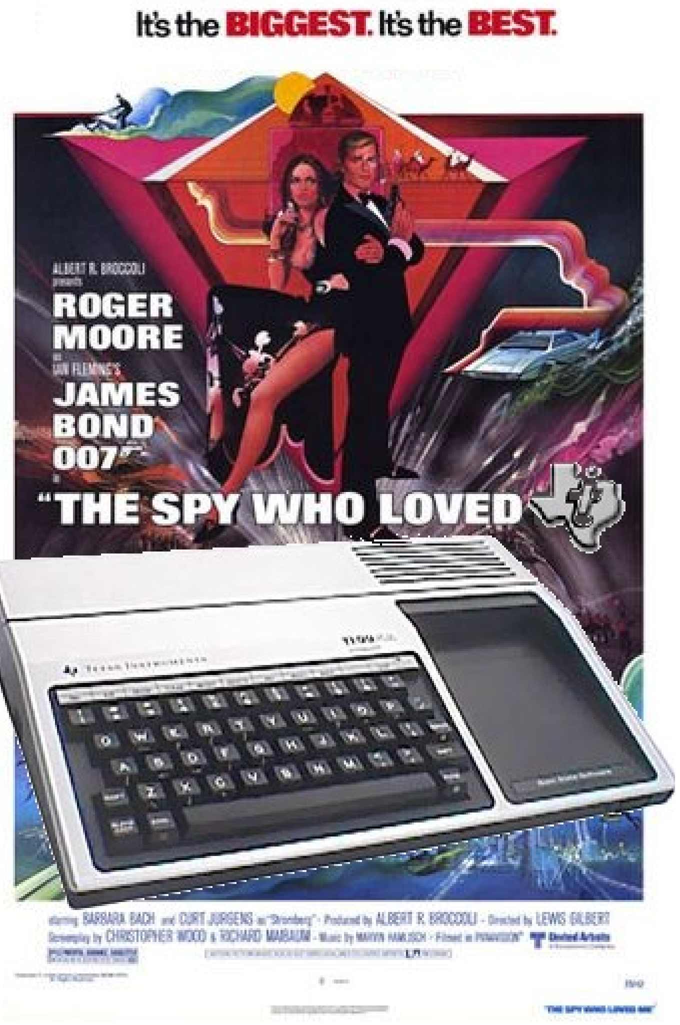
Last week Richard and I joined up with the long lost member Mark Wills. He has written some routines which I believe are in an article in this magazine. So I wait with baited breath to take a peek. Mark has sent me some code that runs as an option5. However it runs sweet as a nut in the emulator.

Well that is about me for this time. Short and sweet I know.
However I will try and do better next time.

Again a Merry Christmas and happy new year.
Trevor Stevens (Chairman)

Fctn / Quit

The Spy Who Loved TI – *By Richard Twyning*



Greetings to our loyal reader!
This article will be shorter than most I think. Depends on how much time I can devote to it this week, because I'm setting myself a deadline, to not only write this article, but to finish the entire magazine and get it printed and posted!
I am looking forward to Christmas, but not as much as things that will be happening after Christmas. I'm really looking forward to the two day workshop and I want it to be as well supported as possible, so I won't feel ashamed, and I won't apologize for the amount that I've published it, or the number of times I've included the directions in the magazine!
I hope lots of you are going to support us in Stanton St. John. I'll print the directions again in this magazine just to make sure you've got them, because you never know if any magazines have gone missing! That leads me to say that if you are a fully paid up member or even an honorary member, please let us know if you think you should have received a newsletter and we'll produce an extra copy and get it sent or emailed out to you.
Ideally we would like all our members to take the option of email membership as it would save all the hassles of postage and the problems of magazines going missing!

I'm not sure what we have planned yet for Stanton St. John, apart from a nice pub meal on the Friday night. I don't know what hardware, if any, I'll be taking with me. I might take my laptop so we can demonstrate Win994A. The trouble with TI hardware now is that it's so heavy! It's much more convenient to take a nice compact machine. I remember a comical moment after the last Sandbach workshop when I bought a PEB from Ross Bennett. I was struggling with it on the way out and the fire hose connector had dropped off and was dragging along the cobbled driveway. "You vandal" Ross shouted in the distance behind me! That's the last workshop we'll ever be having at Sandbach. The pub had gone all bizarre and yuppified! They'd stopped doing food and the atmosphere was just wrong!

I've already planned next year's AGM which you might have seen mentioned briefly at the start of last issue, and at the start of this issue. It's taking place at the same place as last year, the Station Hotel, Newstead Station, Newstead Village, Nottinghamshire. It's got VERY EASY access from the motorway. Access via train is also VERY EASY. It's the old Newstead Station and it's on the Robin Hood line operated by Central Trains between Mansfield and Nottingham. If you can reach Nottingham by train, then Newstead is only 15 minutes by train from Nottingham.

I'll include directions to both Stanton St. John and Newstead at the end of the magazine somewhere ;-)

I'm not intending to write much more as I'll be dangerously close to my usual quota of 24 pages! We usually struggle to fill up the space, but this time, we're struggling to fit everything in!

As I said earlier, I'm looking forward to Christmas, but I'm also looking forward to things beyond Christmas. It was really good recently to get back in contact with Mark Wills and you will notice that immediately following my article, he's managed to write me an article from a bunker on an oil pipeline! It sounds like something out of the James Bond film, The World Is Not Enough which included a pipeline in the plot!

Mark has included some assembly language in his article and he's intending to write a machine code game. Hopefully he will be as excited to see the stuff about the new compiler as everyone else. I've got a very good idea for a game for him, hopefully he'll have something which we can demonstrate at the Stanton St. John workshop, although I don't think Mark will be able to attend as he'll still be in Kazakhstan or Turkey! However, it's excellent to get back in touch with him as he's been a loyal supporter of the group for years and used to be the Vice Chairman and we have had some fun times in the years I've known him.

That's enough from me this time. Have a very good Christmas, and a happy new year.

THE END

BUT...

RICHARD TWYNING WILL RETURN
IN
MOONMINERAKER.

MACHINE CODE MUSINGS AN ARTICLE BY MARK WILLS

Firstly, I'd like to start by apologising to our esteemed editor for the late delivery of this article. You see, I always start out with the best of intentions, and then seem to get distracted by other things, all the time thinking "I must get that article finished", or, "I must get the oil changed in the car", or, "I must get that vasectomy done", and somehow, I never get around to it…

Anyway, this article is approximately ten years late. So I guess I should really, REALLY apologise to our editor! When I promised to submit an article, I was the father of a baby boy. I am now the father of a large boy in secondary school (12), a daughter (11), another son (7), and another son (3). I really MUST get that vasectomy done

For those of you who are sitting there thinking "Who the hell is this guy?", I guess a few words of (re)introduction would not go amiss… So, my name is Mark Wills, (and no, I am no relation to the TI guru Tom Wills, unless there's something my mother hasn't told me). I am English, living in Shropshire, in the UK. I spend very little time in the UK, as I run a business that builds and commissions control systems for the oil & gas industry, which takes me all over the world (except, it would seem, the UK!).

My parents bought me a TI in nineteen eighty something-or-other… Probably 82, and I immediately learned BASIC and many hours were spent in front of the TV with blood shot eyes, typing in programs from magazines and learning how they work. Now I have my own company, writing sophisticated control software, and I owe it all to the TI.

A few weeks ago, I downloaded Win994a out of interest. I was amazed. That is one serious emulator. A real masterpiece of programming. I was also impressed with the fact that it came with Editor Assembler. Just for fun, I decided I would try writing a little machine code program, like this:

```
      DEF START
      REF VSBW


      CLR R0
      LI R1,>2A00
LOOP  BLWP @VSBW
      INC R0
      CI R0,768
      JNE LOOP
STOP  B @STOP
```

Not very impressive, it just fills the screen with asterisks. However, this was the first bit of machine code I had done on a 'TI' for some ten years, and to say I was impressed by own memory would be putting it rather mildly. In fact, when I assembled the program and ran it, my eyebrows temporarily became unencumbered by gravity, and my jaw decided to hang around with my socks for a bit.

Five minutes later, and I've re-joined the TI UK user group (online, you can pay online!) who were so good to me years back, and I'm 'educating' the kids as to what a REAL computer game is…

"Hey kids, come and look at this!"
"What's THAT?"
"THAT… is PARSEC"
"It looks crap, Dad"
"CRAP??!!!! CRAP???!!!! What blasphemy! GO TO YOUR ROOM!"
"Can I have a go?"
"Oh go on then…"

(one hour later…)

"Er, kids, any chance I could have my laptop back, I want to play with the TI emulator"
"Yeah, just let me get past this re-fuelling tunnel and get through the asteroid belt… Do they have this for the X-BOX?"
"Er, no… I'll make a cup of tea then shall I?"

I quickly resolved to write something in machine code, but more impressive than a screen full of asterisk's, really to test myself and see if I could do it… However, this meant I had to find the New Testament (Editor Assembler Manual) the holy grail of TI Computer books… After 24 hours of searching under beds, backs of cars, parents house, attics, airing cupboards, wardrobes, and under the stairs, I found said holy grail, at the bottom of a pile of old books, in the garden shed. Damp, cold, and looking decidedly sorry for itself.

An evening on the radiator had it (almost) restored to it's former glory… Still looking quite good for a 20+ year old book! Oh how I rejoiced when I consulted the section on console ROM/GROM routines…

> "And lo, on the third day, the Lord did speake unto the lowly TI-er and instruct him thus:
>
> Thou shalt NOT access thy VDP memory directly, for it is a sin to covet thy VDP's memory. Though shalt use thy holy BLWP instruction and verily shalt thou choose thy utility as appropriate.
>
> And the utility that thy shall use shall be VSBW,VMBW,VSBR and VMBR.

If thou shalt access thy sacred VDP memory without thy holy utility routines, thou shalt suffer the death of a thousand deaths, and verily shall thy TI crash. Amen."

Well, that's how I remembered it anyway!

So, I thought I would write some machine code to do something "clever". Thus I resolved I would write a "starfield" routine a routine that smoothly scrolls stars across the screen, like you might see in Parsec or similar.

I could use sprites to do it, but that would be too easy… And a waste of sprite resources… No, I'm a man. A REAL MAN. I will use… CHARACTERS… and I will make them move smoothly like sprites… That is the task that I have set myself, and I shan't play another game of Alpiner until it's done (

Thus I ended up with the routine you see below. I shall comment on each section as we come to it, for the benefit of machine code newbies or dabblers. Machine code experts go to the back of the room please, and write a routine to calculate PI to 2 thousand digits. And no talking.

Of course, if you want to type this routine in, simply type it in to editor assembler, leaving out my comments and assemble it. I'd love to hear from anyone that has run this on a 'real' TI (I feel an Ebay session coming on…)

```
* VDP REGISTER DEFINITIONS
VBLNK  EQU >83D7  * VERTICAL BLANK COUNTER
VDPR   EQU >8800  * VDP READ REGISTER
VDPW   EQU >8C00  * VDP WRITE REGISTER
VDPA   EQU >8C02  * VDP ADDRESS REGISTER

       DEF START
       AORG >A000
```

(Ok, in this section, we are simply creating some symbols, which have values. For example, VDPR is a symbol, with the value of 8800 hex (the > sign mean HEX (base 16)). This will make the code somewhat easier to read later on, as we can use a symbol, which has more meaning, in our code, rather than the address. VDPR is short for VDP Read. It is the address of the VDP read register. Contrary to the third commandment earlier in this article, we will be doing VDP access 'in the raw', without the help of console GROM routines, which are very slow, because they require what is known as a context switch. We also set the 'origin' of the code where it will reside in memory, using the AORG directive. In this case, our code will sit in the 24k memory expansion. Also, we also DEFine a start address, which will allow us to run the program later from Editor Assembler option 3.)

```
*---------------------------------------------------
* SECTION 1 - SET UP VDP REGISTERS
* SET THEM UP WITH SAME VALUES AS ED/AS

START  LI R0,>0000
       BL @VDPWR

       LI R0,>01E0
       BL @VDPWR
       SWPB R0
       MOVB R0,@>83D4

       LI R0,>0200
       BL @VDPWR

       LI R0,>030E
       BL @VDPWR

       LI R0,>0401
       BL @VDPWR

       LI R0,>0506
       BL @VDPWR

       LI R0,>0600
       BL @VDPWR

       LI R0,>07F5
       BL @VDPWR
```

Section 1 Description:
We simply set up the various tables of the VDP processor in memory. I would write more on it here, but I am sat in a hotel room in Turkey and don't have access to the Holy Grail (Editor Assembler)

```
*---------------------------------------
* SECTION 2 - DEFINE @ CHARACTER AS BLANK
```

Ok, quite a lot in section 2 we define the @ character as a blank character (the TI Basic equivalent would be CALL CHAR(65,"00")) We do this by writing a block of eight bytes, whose start address is represented by the symbol CHRBUF, and we write these eight bytes to the VDP memory, starting at address >A00, which is the address of the first horizontal row for the @ character:

The subroutine VMBW is used which means VDP Multiple Byte Write. Those of you who are familiar with TI assembly will note we are not using the standard TI VMBW. This is because I chose to replace them with my own, which are faster, as no BLWP (context switch is required). The VMBW routine exactly the same as the TI one however: R0 (register 0) contains the destination VDP address, R1 contains the SOURCE (CPU RAM) address, and R2 contains the number of bytes to write.

```
       LI R0,>A00
       LI R1,CHRBUF
       LI R2,8
       BL @VMBW
```

Next, we place two asterisk symbols on the top line of the screen, at the left hand side, and the right hand side. This is the scrolly border… A scrolly is a message that 'scrolls' across the screen from right to left as the program runs. Most 'demos' contain some sort of scrolly, so does this one!

First, we clear R0, so it points to the start of the screen. Then we use the VSBW (VDP Single Byte Write) subroutine to write an asterisk to the screen R1 contains the ASCII code of the asterisk in its most significant byte.

We then increment the screen by one (INC R0) and again write an asterisk. Then we set the screen address to 30 (decimal), the last but one character on the top line, and write another asterisk. Then, again increment the screen address, and again, write another asterisk. Done. We also load R9 with the start address of our scrolly message. This will be used later.

```
* DISPLAY SCROLLY BORDER
       CLR R0
       LI R1,>2A00
       BL @VSBW
       INC R0
       BL @VSBW
       LI R0,30
       BL @VSBW
       INC R0
       BL @VSBW

       LI R9,TXTMSG *SCROLLY BUF ADDR
```

Next, we have the start of the main loop. R15 is used to place a single pixel into the top row of the @ character. The most significant byte (MSB) is loaded with 1. This will later be shifted through all eight bit positions, giving the illusion that the stars are moving from right to left.

R15 is copied to R1 (because the VSBW routine expects the value to write in the MSB of R1) and we call the VSBW routine. The @ character is now a single pixel.

```
LOOP1  LI R15,>0100
       MOV R15,R1
       LI R0,>A00
       BL @VSBW


*-------------------------------------
* SECTION 3 - DISPLAY THE STARS
```

Having defined the @ character as a single pixel, we write a series of 'stars' (@ characters) to the screen one on each row of the screen. To do this, I defined a buffer in CPU ram, whose start address is represented by the symbol STRARY (star array), and loaded into R3. R1 is loaded with >4000, the MSB of which represent the ASCII code for the @ sign (64 decimal).

The array contains 'offsets' from the left hand column of the screen.
R5 indicates how many stars to write to the screen 23.

```
       LI R1,>4000
       LI R2,32
       LI R3,STRARY
       CLR R4
       LI R5,23
```

The next merry little ditty, below, is a little loop which reads each offset from the array, and adds it to screen address contained in R2. (R2 is initially set to 32, the left most column of screen line 1).

It does this by reading a byte from the array, and also incrementing the array pointer address (MOVB *R3+,R4) so R4 now contains the offset, as read from the array. This is then swapped over to the least significant byte:

```
NXTSTM MOVB *R3+,R4
       SWPB R4
```

Next, the screen offset, in R2, is moved into R0, as the VSBW routine will need it there later:

```
       MOV R2,R0
```

The offset, as read from the array, is now added to the screen offset. The result is in R0, we then write it to the screen:

```
       A R4,R0
       BL @VSBW
```

Next, we increment the screen offset by 32, so that it points to the left most column of the next line down. We also swap R4 back over, so that we don't corrupt the LSB. Then, we decrement the screen row counter (R5) and if it is not 0, we loop back and do the next row of the screen:

```
       AI R2,32
       SWPB R4
       DEC R5
       JNE NXTSTM

*--------------------------------------
* SECTION 4 - SPIN THE STARS
* move 1 bit pos on each vblank
```

Here, we rotate the pixel in the @ character, one bit position to the left. This of course gives the illusion that the stars are moving. Note that there is no Shift Left Circular instruction on the TMS9900 because you don't need one. There is only a shift right circular (SRC) instruction, which is fine… If you do the math, its pretty easy to figure out that by rotating the value 15 bits, the bit we are interested in ends up one position to the left. (It took me an embarrassingly long time to realise that, however.)

We rotate our single pixel, held in R1, and write it to >A00 in VDP (start address of the @ character). HOWEVER, we ONLY do it when a vertical blank is in progress… I.e, when the raster scan is switched off, and the TV/Monitor is moving the guns back up to the top left of the picture tube. The subroutine WVBLNK means 'Wait for Vertical Blank', and does exactly that.

```
       LI R1,>0100
       LI R0,>A00
       LI R2,8
ROTSTR BL @WVBLNK
       BL @VSBW
       SRC R1,15
       DEC R2
       JNE ROTSTR
*--------------------------------------
* SECTION 5 - ERASE STARS
```

When we get to here, we have moved the star through all eight pixel positions… Now we have to physically move them to a different location on the screen (each star one character to the left)… However, before we do that, we have to erase the stars already on the screen. This routine is the same as the 'display stars' routine, above, except it rights spaces to the screen positions, instead of ' characters.

```
       LI R1,>2000
       LI R2,32
       LI R3,STRARY
       CLR R4
       LI R5,23

NXTSTE MOVB *R3+,R4
       SWPB R4
       MOV R2,R0
       A R4,R0
       BL @VSBW
       AI R2,32
       SWPB R4
       DEC R5
       JNE NXTSTE

*---------------------------------
* SECTION 6 - MOVE STARS
```

Now we have erased the stars from their original postions… We now turn to the star array again, read each offset from the array, add 1 to it, and write it back again.

R0 contains the address of the star array, and R3 the number of values in the array (23).

```
       LI R1,>1F00
       LI R0,STRARY
       CLR R2
       LI R3,23
```

We move the byte addressed by R0 into the MSB of R2. This causes a compare to zero (most instructions do) and if it is NOT zero, we jump to the decrement routine. However, if it IS 31, the code falls through, and moves R1 (>1F, 31 decimal) into the byte array position. Thus, the screen offsets 'wrap around' from 0 to 31.

```
MOVSTR MOVB *R0,R2
       JNE DECR
       MOVB R1,*R0+
       DEC R3
       JNE MOVSTR
       JMP LOOP1
```

Here we decrement the value in the array by one, by adding 256 to it. We then write it back to the array. R3 is decremented, and if not zero, we go around and do it all again. Remember, this is all happening in the vertical blank period! We don't have long to do it! I'm not sure if it fits entirely within the VBLANK, its impossible to time it, but it does look very smooth (on my emulator (), with no flickering of the stars, which suggests it does…

```
DECR   AI R2,-256 *DEC LSB
       MOVB R2,*R0+
       DEC R3
       JNE MOVSTR

* UPDATE SCROLLY
```

Now we update the scrolling message on the top row of the screen. Remember at the start of the program we loaded R9 with the address of the scrolly buffer? Well, we will use it now. We load R1 with the current offset into the scrolly buffer (in R9), because the VMBW (VDP Multiple Byte Write) subroutine needs it in R1. R2 holds the number of bytes to write (28). Then we call the VDP Multiple Byte Write routine and write the text to the screen.

```
       LI R0,2
       MOV R9,R1
       LI R2,28
       BL @VMBW
```

Next we increment R9, so it moves one position to the right in the scroll buffer. We also look for a # symbol in the text, which is used as an 'end of buffer' marker (so I could easily change the contents of the buffer without counting the number of characters). Then we jump back up to the top of the program to do it all over again!

```
       INC R9
       C *R9,@HASH
       JNE LOOP1
       LI R9,TXTMSG
       JMP LOOP1
```

Here are the utility subroutines. I must give credit at this point to Karsten, (http://sometimes.99er.net). On his site are some really nice demos you can download and run on MESS or Win994a, and some of them have the source code. I looked at his source code to work out how to do direct VDP access (especially writing to the VDP registers) because I couldn't remember how to do it! Thank you Karsten.

```
*--------------------------------
* SECTION 7 - UTILITY ROUTINES

* VDP SINGLE BYTE WRTE
* R0=ADDRESS IN VDP TO WRITE TO, R1(MSB)=THE BYTE TO WRITE
* NOTE: R8 IS DESTROYED UPON RETURN

VSBW   MOV R0,R8
       ORI R8,>4000
       SWPB R8
       MOVB R8,@VDPA
       SWPB R8
       MOVB R8,@VDPA
       NOP
       MOVB R1,@VDPW
       RT
*--------------------------------------------------
* VDP MULTIPLE BYTE WRITE
* R0=DESTINATION IN VDP, R1=SOURCE ADDRESS IN CPU RAM, R2=NUMBER OF
BYTES TO WRI
* NOTE: R8 IS DESTROYED UPON RETURN

VMBW   MOV R0,R8
       ORI R8,>4000
       SWPB R8
       MOVB R8,@VDPA
       SWPB R8
       MOVB R8,@VDPA
       NOP
VMBW1  MOVB *R1,@VDPW
       INC R1
       DEC R2
       JNE VMBW1
       RT
*---------------------------------
* VDP WRITE TO VDP REGISTER
* R0(MSB)=THE REGISTER TO WRITE TO, R0(LSB)=THE VALUE TO WRITE

VDPWR  ORI R0,>8000
       SWPB R0
       MOVB R0,@VDPA
       SWPB R0
       MOVB R0,@VDPA
       RT
*---------------------------------
* WAIT FOR VERTICAL BLANK
* NOTE: DESTROYS R8

WVBLNK CLR R8
       MOVB R8,@VBLNK
       LIMI 2
BLNKWT CB @VBLNK,R8
       JEQ BLNKWT
       LIMI 0
       RT
*---------------------------------
* SECTION 8 - PROGRAM DATA

* CHARACTER DATA FOR @ SIGN - EMPTY
CHRBUF DATA 0,0,0,0,0,0,0,0
       EVEN

*X OFFSETS FOR STARS


STRARY BYTE 13,26,15,7,23,2,9,2
       BYTE 27,4,24,31,29,4,10
       BYTE 20,13,16,3,8,22,1,28
       EVEN
```

```
HASH   TEXT '#'
       EVEN

* TEXT FOR SCROLLY
TXTMSG TEXT '                               '
       TEXT 'HELLO, THIS IS MARK WILLS '
       TEXT 'HERE. THIS IS THE FIRST '
       TEXT 'BIT OF PROGRAMMING I HAVE '
       TEXT 'DONE ON THE TI FOR 10 '
       TEXT 'YEARS! IT DOESNT DO MUCH, '
       TEXT 'IT JUST SCROLLS SOME STARS '
       TEXT 'ACROSS THE SCREEN... '
       TEXT 'HOWEVER, THE MACHINE CODE '
       TEXT 'FANS AMONG YOU MAY BE '
       TEXT 'INTERESTED TO KNOW THAT '
       TEXT 'ALTHOUGH THE STARS MOVE '
       TEXT 'VERY SMOOTHLY, THEY ARE NOT '
       TEXT 'SPRITES. THEY ARE CHARACTERS! '
       TEXT 'THIS IS HOW I DID IT... '
       TEXT 'A CHARACTER IS LOADED WITH A '
       TEXT 'SINGLE PIXEL, AND THIS IS '
       TEXT 'LEFT SHIFTED THROUGH ALL 8 '
       TEXT 'HORIZONTAL POSITIONS, DURING '
       TEXT 'A VERTICAL BLANK PERIOD... '
       TEXT 'AFTER ALL 8 POSITIONS HAVE '
       TEXT 'BEEN MOVED THROUGH, I ERASE '
       TEXT 'THE CHARACTERS FROM THE SCREEN '
       TEXT 'AND REDRAW THEM ONE CHARACTER '
       TEXT 'POSITION TO THE LEFT. AGAIN, '
       TEXT 'DURING A VERTICAL BLANK. FOR '
       TEXT 'MORE SPEED, SCREEN SCROLLING '
       TEXT 'IS NOT USED - THERE WOULD BE '
       TEXT 'TOO MUCH TIME SPENT ON VDP '
       TEXT 'ACCESS. INSTEAD I USE A BUFFER '
       TEXT 'IN CPU RAM TO KEEP TRACK OF THE '
       TEXT 'X AXIS OF THE STARS... WATCH '
       TEXT 'OUT FOR MORE MACHINE CODE STUFF '
       TEXT 'SOON... ALSO, TRY HTTP://'
       TEXT 'SOMETIMES.99ER.NET FOR SOME '
       TEXT 'NICE MACHINE CODE TI99 DEMOS BY '
       TEXT 'KARSTEN... REALLY NEAT STUFF... '
       TEXT 'GREETS TO TREVOR STEVENS AND '
       TEXT 'RICHARD TWYNING OF TIUG(UK) '
       TEXT '  :-)'
       TEXT '                            #'
       END
```

Type all this source code into Editor Assembler, (without my comments of course), and save it as STARS;S

Next, load the assembler, and give an object name of STARS;O
In options, use the R option (because I am a bit of a girl, and use the 'R' prefix in front of register names, it just reads clearer to me!)

Then, use option 3 of the editor assembler loader, specify STARS;O as the file name, and then press enter again after it loads. Next it will ask you for a program name. Enter START. Sit back and enjoy

Development Environment:

A word about how I wrote this code… I did it the 'old way', using TI's trusted Editor Assembler took me while to remember the function keys though! To debug it, I used Millers Graphics explorer, which is a TI program that EMULATES the 9900, and allows you to single step through the code, and see the code and registers, line by line. This is how I used to do it in the old days. Win994a comes with a windows based TMS9900 assembler, but it just crashes on my machine can't get it to do anything. If anyone knows what I'm doing wrong, drop me a line.

Thanks to Berry Harmsen, who found Explorer for me

I can be contacted by email: mark@markwills.co.uk

Until next time… Keep hacking… I will (By the way, this article was written partly in a hotel room in Keyseri in Turkey, and partly (mostly) 6 feet underground, in a small bunker, housing our computer equipment, on the BTC (Baku, Tbilisi Cheyhan) pipeline. It is 8 degrees in here and my fingers are somewhat numb! Still, it's surprising the work you can get done when you are waiting for Siemens to sort their fibre optics out (

Mark Wills

Francesco Lama – Media Librarian

Many thanks to Francesco for supplying us with a new updated module library listing.

MODULE LIBRARY

TITLE	QTY IN STOCK	PRICE (POUNDS)
32k SUPERSPACE (MODIFIED ROMOX)	1	25.00
ADDITION & SUBTRACTION 1	1	3.00
ADVENTURE + PIRATE TAPE (OTHERS TOO)	5	5.00
ADVENTURE MODULE ONLY	4	3.50
ALPINER	3	8.00
A-MAZING	11	3.00
BEGINNING GRAMMAR	4	3.00
BIG FOOT	1	3.50
BLASTO	2	5.00
CAR WARS	2	4.00
CHISHOLM TRAIL	2	3.50
DISK MANAGER	4	2.00
DISK MANAGER 2	1	4.50
DIVISION 1	1	3.00
EDITOR ASSEMBLER + MANUALS & DISKS	4	25.00
EXTENDED BASIC + MANUAL	4	22.50
EXTENDED BASIC MODULE	5	15.00
HOUSEHOLD BUDGET MANAGEMENT	4	3.50
HUNT THE WUMPUS	2	4.00
HUSTLE (EA VERSION ONLY)		2.00
INDOOR SOCCER	2	4.00
JAWBREAKER 2	2	4.00
MINI MEMORY + LINE BY LINE ASS.	4	15.00
MINI MEMORY AS ABOVE + MINI WRITER	2	18.00
MOON MINE	1	8.00
MULTIPLAN + SOFTWARE + MANUAL	2	30.00
MULTIPLICATION 1	1	3.00
MUNCHMAN	2	3.50
PARSEC	1	4.00
PERSONAL RECORD KEEPING	5	3.50
PERSONAL REPORT GENERATOR	2	5.50
PHYSICAL FITNESS	1	4.00
PICNIC PARANOIA (NOT MK 2 CONSOLES)	1	3.50
PROTECTOR (NOT MK 2 CONSOLES)	4	5.00
PROTYPER	1	20.00
SHAMUS (NOT MK 2 CONSOLES)	3	3.50
SPEECH EDITOR	1	3.50

MODULE LIBRARY continued…

SUPER DEMON ATTACK	1	4.50
SUPER EXTENDED BASIC	1	30.00
TAX/INVESTMENT RECORD KEEPING	1	5.00
TERMINAL EMULATOR 2	5	5.00
THE ATTACK	2	4.00
TI INVADERS	1	4.00
TI LOGO + FOLDER & MANUAL	1	15.00
TI WRITER MODULE	1	8.00
TOMBSTONE CITY	4	4.00
VIDEO CHESS	1	5.00
VIDEO GAMES 1	2	3.50
YAHTZEE	3	3.00
ZERO ZAP	1	3.50

Directions to forthcoming events…

2006 Stanton St. John, 2-Day Workshop

It will take place on the **Saturday 4th and Sunday 5th of March 2006** at the Stanton St. John village hall in Stanton St. John just north east of Oxford.

DIRECTIONS TO STANTON ST. JOHN VILLAGE HALL

FROM THE NORTH OR EAST: approach Oxford along the M40 (whether coming from the Birmingham or the London direction), and leave the motorway at junction 8, heading for Oxford along the A40.
The dual carriage way will continue until you reach the Green Road Roundabout, where you will need to take the fourth exit (signposted Barton, Horton-Cum-Studley etc..). Follow this road down into the dip and up the hill on the other side. Continue along it until you reach the staggered junction at the end. Turn right at this junction and drive on until you reach a pub on your left-hand-side, called "The Talk House". Turn left into the village of Stanton St. John immediately after the pub. You will soon pass a second (and much better) pub, called "The Star Inn". Continue past the pub, and you should see signs for the TI99 Workshop, pointing towards the Village Hall car park, on the right hand side of the road (if you reach the church, on the left hand side, you'll have gone too far).

FROM THE WEST: approach Oxford along the A40 (from the Gloucester direction). You will first come to the Wolvercote roundabout. Take the third exit off it, and continue along the Oxford ring-road. You will come to a second roundabout (Banbury Road), at which you should take the second exit, in order to again continue along the Oxford ring-road. Drive on for several miles, until you reach the Green Road Roundabout. Take the first exit off this (signposted Barton, Horton-Cum-Studley etc..). Follow this road down into the dip and up the hill on the other side. Continue along it until you reach the staggered junction at the end. Turn right at this junction and drive on until you reach a pub on your left-hand-side, called "The Talk House". Turn left into the village of Stanton St. John immediately after the pub. You will soon pass a second (and much better) pub, called "The Star Inn".

Continue past the pub, and you should see signs for the TI99 Workshop, pointing towards the Village Hall car park, on the right hand side of the road (if you reach the church, on the left hand side, you'll have gone too far).

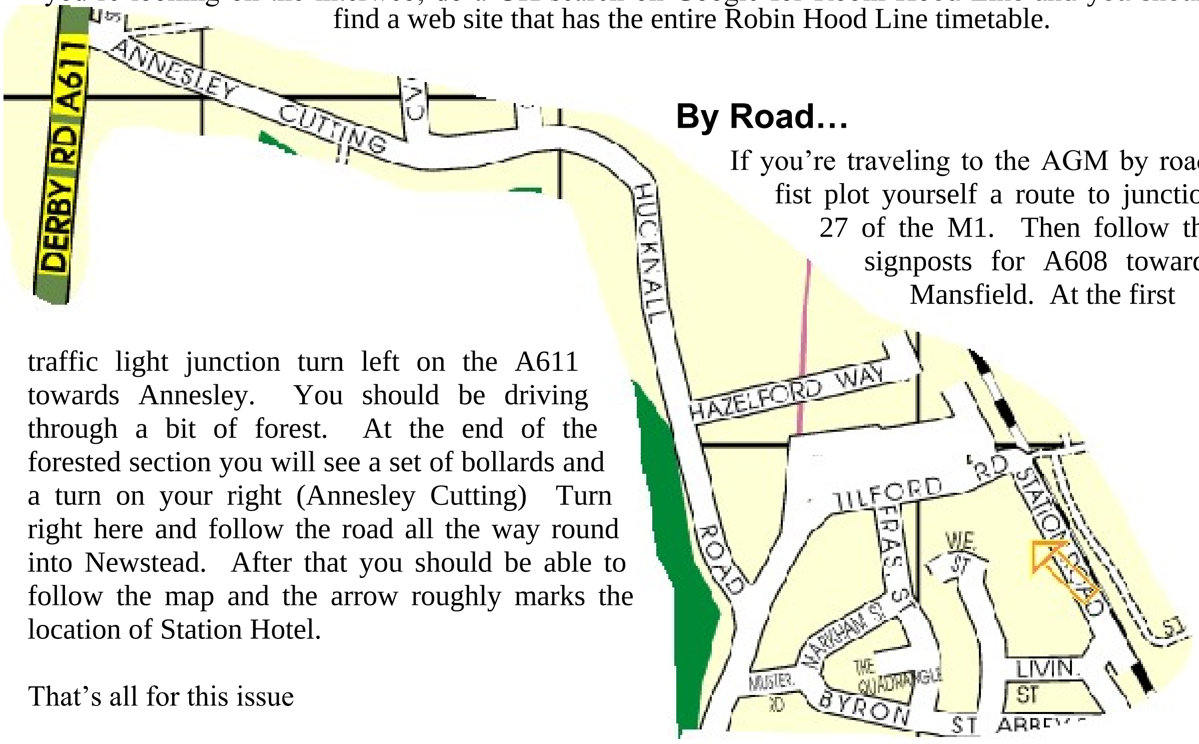
FROM THE SOUTH: approach Oxford along the A34 (from the Newbury direction). Proceed past two main exits for Oxford, signposted Cowley and Bottley (North Hinksey) respectively, and turn off at the one signposted Woodstock and Blenheim Palace. At the subsequent roundabout take the fourth exit (dual carriage way towards Oxford, make sure you do not go up the bank again to the A34!). Continue along this road until you reach the Wolvercote roundabout. Take the second exit off this roundabout , and continue along the Oxford ring-road. You will come to a second roundabout (Banbury Road), at which you should take the second exit, in order to again continue along the Oxford ring-road. Drive on for several miles, until you reach the Green Road Roundabout. Take the first exit off this (signposted Barton, Horton-Cum-Studley etc..). Follow this road down into the dip and up the hill on the other side. Continue along it until you reach the staggered junction at the end. Turn right at this junction and drive on until you reach a pub on your left-hand-side, called "The Talk House". Turn left into the village of Stanton St. John immediately after the pub. You will soon pass a second (and much better) pub, called "The Star Inn". Continue past the pub, and you should see signs for the TI99 Workshop, pointing towards the Village Hall car park, on the right hand side of the road (if you reach the church, on the left hand side, you'll have gone too far).

2006 AGM

It will take place on Saturday 3rd June 2006 at the Station Hotel, Newstead Station, Newstead Village, Nottinghamshire.
As I've already said in the magazine, it's really easy to find.

By Train…

If you can find your way to Nottingham by train, then Newstead is just 15 minutes from Nottingham on the Robin Hood Line which is operated by Central Trains.
Trains run every 30 minutes, but not all of them call at Newstead. If you're looking for a train at Nottingham it will be on the monitor as Worksop because the line terminates at Worksop. Obtain a timetable from Nottingham station, or via the interweb to check which ones stop at Newstead. If you're looking on the interweb, do a UK search on Google for Robin Hood Line and you should find a web site that has the entire Robin Hood Line timetable.



By Road…

If you're traveling to the AGM by road, fist plot yourself a route to junction 27 of the M1. Then follow the signposts for A608 towards Mansfield. At the first traffic light junction turn left on the A611 towards Annesley. You should be driving through a bit of forest. At the end of the forested section you will see a set of bollards and a turn on your right (Annesley Cutting) Turn right here and follow the road all the way round into Newstead. After that you should be able to follow the map and the arrow roughly marks the location of Station Hotel.

That's all for this issue