TI*MES

This T**T ruined Treff 2006 for us!

We need to send him a Christmas present on behalf of Texas Instruments…

Texas Instruments Laser Guided Bomb!!!

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! J

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. **J**

Important Information

Membership Renewals

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 been 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 AND movies from our very successful workshop in Stanton St. John near Oxford.
Our website now has a passworded zone where you can read or download previous and current issues of TI*MES magazine.

From the Chairman's Chair

By Trevor Stevens

Unfortunately, this time around, Trevor hasn't had time to produce an article.

He would however like to make an official statement, aimed at the BAA security staff at Stanstead Airport…

The Bastards!!!!!

From Rainworth With Love – ***By Richard Twyning***

Dear Reader!

Well, as you might have guessed, we didn't make it to the Treff this year!
We missed our flight because of the amateurish security measures at Stanstead Airport!
We've put a claim in and they're initially denying responsibility, so we need to keep at it!
You will realise that for this reason my article isn't the one advertised at the end of my last article. This happened in the Bond movies too! At the end of The Spy Who Loved Me, they proudly announced that James Bond would return in For Your Eyes Only, but there was a little space film that almost went unnoticed had it not broken box office records and coined the phrase "Blockbuster!"
For this reason, "Cubby" Broccolli wanted to cash in on the success and produce a space film of his own! Moonraker was the result.

Unfortunately time and "other projects" have prevented me from producing much of an article. I'll spend what time I have on trying to get this issue out of the door and save anything else for the Christmas issue!

Upcoming events for your diary…

Two-day AGM in Poole, Dorset in May 2007

We are still planning to hold a two day workshop in the Poole area in May 2007. We're fairly sure that it will be the last weekend in May.
We're struggling to find a venue at the moment. Most of the hotels down there cater for weddings quite a bit, so they want extortionate amounts of money for room hire!
John Murphy went to check out one hotel that seemed quite promising, but he said the room didn't seem very secure for leaving the stuff in over night.

I'm still on the case and will try my best to have something to publish in the Christmas issue.

Treff 2007

First, an apology, since we printed the details of the 2007 TI-Treff in the last issue, Berry Harmsen has informed us that the hotel has contacted him and told him that they are converting the room he booked into a restaurant, so we can't use it for the Treff anymore! **L**

Watch this space for new details as soon as they become available **J**

That's it from me for another quarter, if you can call it a quarter. Apologies for the delay in receiving issue. We'll be making the Christmas issue extra special.

THE END

BUT…

*RICHARD TWYNING **WILL** RETURN*

IN

GOLDEN TI.

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	10	3.00
BEGINNING GRAMMAR	4	3.00
BIG FOOT	0	3.50
BLASTO	1	5.00
CAR WARS	1	4.00
CHISHOLM TRAIL	1	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	3	3.50
HUNT THE WUMPUS	1	4.00
HUSTLE (EA VERSION ONLY)		2.00
INDOOR SOCCER	1	4.00
JAWBREAKER 2	1	4.00
MINI MEMORY + LINE BY LINE ASS.	3	15.00
MINI MEMORY AS ABOVE + MINI WRITER	2	18.00
MULTIPLAN + SOFTWARE + MANUAL	2	30.00
MULTIPLICATION 1	1	3.00
MUNCHMAN	1	3.50
PARSEC	1	4.00
PERSONAL RECORD KEEPING	4	3.50
PERSONAL REPORT GENERATOR	1	5.50
PHYSICAL FITNESS	1	4.00
PROTECTOR (NOT MK 2 CONSOLES)	3	5.00
PROTYPER	1	20.00
SHAMUS (NOT MK 2 CONSOLES)	2	3.50
SPEECH EDITOR	1	3.50
SUPER EXTENDED BASIC	1	30.00
TERMINAL EMULATOR 2	5	5.00
THE ATTACK	1	4.00
TI INVADERS	1	4.00

MODULE LIBRARY continued…

```
TI LOGO + FOLDER & MANUAL ..... 1 ..... 15.00
TI WRITER MODULE ..... 1 .....  8.00
TOMBSTONE CITY ..... 3 .....  4.00

VIDEO CHESS ..... 1 .....  5.00
VIDEO GAMES 1 ..... 1 .....  3.50

YAHTZEE ..... 2 .....  3.00
```

Mark Wills, back with a vengeance!

Mark is back from the Sudan and now working in the worlds most civilised country… Scotland. He's based in Aberdeen and found the time to start new projects on the TI. This assembly code is a little demo that puts stars on the screen. We have a fully compiled version saved as a disk file for Win994A which we will make available for download on the group's website. **J**

Stars Demo

```
vblnk equ   >83d7
vdpr  equ   >8800
vdpw  equ   >8c00
vdpa  equ   >8c02

;register definitions - because I'm a girl and I like 'R0' instead of '0'!
r0    equ   0
r1    equ   1
r2    equ   2
r3    equ   3
r4    equ   4
r5    equ   5
r6    equ   6
r7    equ   7
r8    equ   8
r9    equ   9
r10   equ   10
r11   equ   11
r12   equ   12
r13   equ   13
r14   equ   14
r15   equ   15

      aorg  >6000       ; console ROM

      data  >aa01       ; standard identifier
      data  >0
      data  >0
      data  menu
      data  >0

menu  data  >0          ; definition of menu item
      data  start
```

```
      byte  10          ; length of text
      text  "MARK WILLS"
      even

start limi  0           ; turn off interrupts
      clr   r0          ; top of screen
      li    r1,txt            ; address of text buffer
      li    r2,17       ; length of buffer
      bl    @vmbw       ; write it to screen
      li    r0,32
      clr   r2
reset li    r1,>2100
next  bl    @vsbw
      inc   r0
      ci    r0,768
      jne   next
      li    r0,32
      ai    r1,>0100
      ci    r1,>6900
      jne   next
      jmp   reset

; VDP SINGLE BYTE WRTE
; replaces the console version, which requires a context switch (BLWP)
; this one is twice as fast
; r0=address in vdp to write to
; r1(msb)=the byte to write
; NOTE: r8 is destroyed upon return
vsbw  mov   r0,r8       ; copy address to r8 (because we will modify it)
      ori   r8,>4000    ; needed when vdp operation is a write
      swpb  r8          ; send low byte first
      movb  r8,@vdpa    ; send it to vdp
      swpb  r8          ; get high byte
      movb  r8,@vdpa    ; send it to vdp
      nop               ; waste some time
      movb  r1,@vdpw    ; write the data to vdp write register
      rt                ; return to caller

; VDP MULTIPLE BYTE WRITE
; writes a buffer of data from cpu ram into vdp ram
; r0=destination in vdp
; r1=source address in cpu ram
; r2=number of bytes to write
; NOTE: r8 is destroyed upon return
vmbw  mov   r0,r8       ; copy address to r8 (because we will modify it)
      ori   r8,>4000    ; needed when the vdp operation is a write
      swpb  r8          ; low byte first
      movb  r8,@vdpa    ; send the low byte
      swpb  r8          ; get the high byte
      movb  r8,@vdpa    ; send it to vdp
      nop               ; give vdp processor some time
vmbw1 movb  *r1,@vdpw   ; move a byte into vdp. vdp address auto increments
      inc   r1          ; move to next address in buffer
      dec r2                ; decrement counter
      jne   vmbw1       ; if not finished write the next byte
      rt                ; return to caller

      even
txt   text "HELLO FROM WILLSY"
      even
```

Mem Check article

I've been playing with Win994a a lot – I just can't stop playing with it – It's quite funny really, I mean I went out and bought just about the most expensive laptop I could find – dual processors, Bluetooth, Infra red, wireless, 2GB ram, DVD Writer, yadda yadda yadda, and what do I do with it? I play Parsec, Alpiner, and write machine code for an orphan computer!

As some of you will know, Win994a has, in addition to the 32K extended memory that an original expanded TI has, an extra 16MB of memory! Now, for our little TI's, that is an AWFUL lot of memory!

Now, obviously 16mb in 32k doesn't quite go, so the author of Win994a, Cory Burr, has devised a neat way of getting that memory accessible. It is 'paged' into a 4K window at >7000 to >7FFF.

But how do you select which bank you want to access I hear you ask? Well, Cory has enhanced TI's original technique of writing to the ROM space at >6000 to >6FFF. Writing any value to >6000 will page in the first 4K of the 16MB expansion memory. Writing any value to >6001 will page in the second 4K bank, and so on. Very neat, and very simple, and retains compatibility with the original TI architecture.

So, I thought I would write something to demonstrate this. As far as I know, I am the first person to do anything with the extra memory – and I am a little embarrassed that my first attempt isn't a bit more flash, but it illustrates the techniques nicely I think.

The program below is a memory checker for the entire 16mb expansion memory. It sweeps all 16MB with >AA and then >55 (inverted bit patterns) and reads the values back, making a note of any errors found. Of course, since this is running in an emulator, there shouldn't be any errors, (or you would be in big trouble) but I have written the code from the "TI's" perspective, not the PC's – therefore, the "TI" doesn't *know* it's running on a PC, and should make no assumptions.

It gives a report at the end of any errors that it made find (should be zero of course!) and you are done.

I wanted to make this code into a cartridge, but I can't find a way of doing it – the Cart Creator in Win994a want's .BIN files, but Asm994a outputs .BIN files. If anyone knows the answer, please let me know.

Ok, regards to all TI'ers everywhere – keep on keepin' on!

Mark Wills

Here is the code, which is written for Asm994a, the assembler that comes with Win994a:

```
;
; Mem Check
;
; Utility program written for Win994a to access and 'test' the 16mb expanded memory
afforded by Win994a
; Note: this program only works with Win994a
;
; Mark Wills - August 2006
;
             def START
             aorg >a000          ; cpu ram, upper 24k

; executable entry point
START  limi 0              ; turn off interrupts

             li r3,>6000         ; memory bank pointer
             li r7,1                   ; memory bank pointer for display purposes
             li r4,>ff           ; dummy value to write to rom area (any value will do)
             clr r5              ; errors counter

nxtpge mov r4,*r3                ; write to ROM area (pages in expanded memory
                                 ; to >7000->7FFF) and move to next page

             bl @wrtAA           ; write >AA's to expanded memory
             bl @rdAA            ; read them back from expanded memory
             bl @wrt55           ; write >55's to expanded memory
             bl @rd55            ; read them back again

             inc r3              ; select next memory page
                   inc r7              ; increment memory bank counter for display
                         ; purposes
             ci r3,>6fff         ; have we done the last memory page?
             jeq   finish        ; if so, hang
             b @nxtpge           ; else do the next memory page

; write >AA's to expanded memory
wrtAA        mov r11,@r11sav     ; save return address
             clr r0              ; screen address
             li r1,wAAmsg        ; address of message
             li r2,24                  ; length of message
             bl @vmbw            ; write message to screen
             li r0,25                  ; screen address
             mov r7,r2           ; bank address
             bl @wi              ; write the address
             li r0,>7000         ; address of expanded memory
             li r1,>aaaa         ; data to write
waalp        mov r1,*r0+         ; write to expanded memory and move to next
                                 ; word
             ci r0,>8000         ; check for end of 4k page
             jne   waalp         ; if not, do next word
             mov @r11sav,r11     ; restore return address
             rt                  ; else return to caller

; read >AA's from expanded memory
rdAA         mov r11,@r11sav     ; save return address
             clr r0              ; screen address
             li r1,rAAmsg        ; address of message
             li r2,24                  ; length of message
             bl @vmbw            ; write message to screen
             li r0,25                  ; screen address
             mov r7,r2           ; bank address
             bl @wi              ; write the address
             li r0,>7000         ; address of expanded memory
raalp        mov *r0+,r1         ; read from expanded memory and move to next
                                 ; word
```

```
            ci r1,>aaaa         ; is the value read equal to >AAAA?
            jeq aanw            ; if so, check next word
            inc r5              ; else increment errors counter
aanw        ci r0,>8000         ; check for end of 4k page
            jne   raalp         ; if not, do next word
            mov @r11sav,r11     ; restore return address
            rt                  ; else return to caller

; write >55's to expanded memory
wrt55       mov r11,@r11sav     ; save return address
            clr r0              ; screen address
            li r1,w55msg        ; address of message
            li r2,24                 ; length of message
            bl @vmbw            ; write message to screen
            li r0,25                 ; screen address
            mov r7,r2           ; bank address
            bl @wi              ; write the address
            li r0,>7000         ; address of expanded memory
            li r1,>5555         ; data to write
w55lp       mov r1,*r0+         ; write to expanded memory and move to next
                                ; word
            ci r0,>8000         ; check for end of 4k page
            jne   w55lp         ; if not, do next word
            mov @r11sav,r11     ; restore return address
            rt                  ; else return to caller

; read >55's from expanded memory
rd55        mov r11,@r11sav     ; save return address
            clr r0              ; screen address
            li r1,r55msg        ; address of message
            li r2,24                 ; length of message
            bl @vmbw            ; write message to screen
            li r0,25                 ; screen address
            mov r7,r2           ; bank address
            bl @wi              ; write the address
            li r0,>7000         ; address of expanded memory
r55lp       mov *r0+,r1         ; read from expanded memory and move to next
                                ; word
            ci r1,>5555         ; is the value read equal to >AAAA?
            jeq nw55            ; if so, check next word
            inc r5              ; else increment errors counter
nw55        ci r0,>8000         ; check for end of 4k page
            jne   r55lp         ; if not, do next word
            mov @r11sav,r11     ; restore return address
            rt                  ; else return to caller

; finished - write number of errors found
finish      li r0,32                 ; screen address
            li r1,finmsg        ; address of message
            li r2,21                 ; length of message
            bl @vmbw            ; write to screen
            mov r5,r2           ; get number of errors in r2
            li r0,54                 ; screen address
            bl @wi              ; write it

die         limi 2              ; enable interrupts for FCTN/QUIT
die1        b @die1             ; loop forever

;
; write an integer to the screen (with leading zero's)
; r0=screen address
; r2=value to write
;
wi          mov r11,@r11sa2
            mov r3,r8           ; save r3
            li r6,buf           ; address of buffer in r6
            clr r1              ; r1 used in DIV instruction
            li r3,10000         ; divisor
```

```
              div r3,r1           ; divide by 10000 - result in r1, remainder in r2
              ai r1,48            ; add ascii offset
              swpb r1             ; get lsb in msb
              movb r1,*r6+        ; move ascii digit to buffer
              clr r1

              li r3,1000          ; divisor
              div r3,r1           ; divide by 10000 - result in r1, remainder in r2
              ai r1,48            ; add ascii offset
              swpb r1             ; get lsb in msb
              movb r1,*r6+        ; move ascii digit to buffer
              clr r1              ; r1 used in DIV instruction

              li r3,100           ; divisor
              div r3,r1           ; divide by 10000 - result in r1, remainder in r2
              ai r1,48            ; add ascii offset
              swpb r1             ; get lsb in msb
              movb r1,*r6+        ; move ascii digit to buffer
              clr r1              ;  r1 used in DIV instruction

              li r3,10                 ; divisor
              div r3,r1           ; divide by 10000 - result in r1, remainder in r2
              ai r1,48            ; add ascii offset
              swpb r1             ; get lsb in msb
              movb r1,*r6+        ; move ascii digit to buffer
              clr r1              ; r1 used in DIV instruction

              ai r2,48            ; final remainder (units column) in r2, add ascii
                                  ; offset
              swpb r2             ; get lsb in msb
              movb r2,*r6         ; move ascii digit to buffer

              li r1,buf           ; address of buffer
              li r2,5                  ; write 5 bytes to vdp
              bl @vmbw            ; do the write
              mov r8,r3           ; restore r3
              mov @r11sa2,r11     ; restore return address
              rt                  ; return to caller

; vdp multiple byte write
; r0=destination in vdp, r1=source address in cpu ram, r2=number of bytes to write
; side effects: r2 zero'd on exit, r0 changed
vdpw   equ >8C00           ; vdp write register
vdpa   equ >8C02           ; vdp address register

vmbw   ori r0,>4000
              swpb r0             ; push address to VDP
              movb r0,@vdpa; push low byte
              swpb r0             ; rotate
              movb r0,@vdpa; push high byte
              nop                 ; spin the wheels a bit
vmbw1  movb *r1+,@vdpw     ; push data
              dec r2              ; decrement counter
              jne vmbw1           ; repeat if not done
              rt                  ; return to caller

; program data
wAAmsg text "Writing >AA's to bank   "
w55msg text "Writing >55's to bank   "
rAAmsg text "Reading >AA's from bank "
r55msg text "Reading >55's from bank "
finmsg text "* Done, Errors Found: "
r11sav data >0000
r11sa2 data >0000
buf           bss 6               ; buffer to hold the ascii representation of a register
(only 5 bytes required)

              end
```

Type the code into Notepad, and save as MemCheck.a99 (put quotes around the file name, and then windows won't add the .TXT file extension)

Then, start Asm994a. Click on Add Assembly Source File. Choose the MemCheck.a99 file.

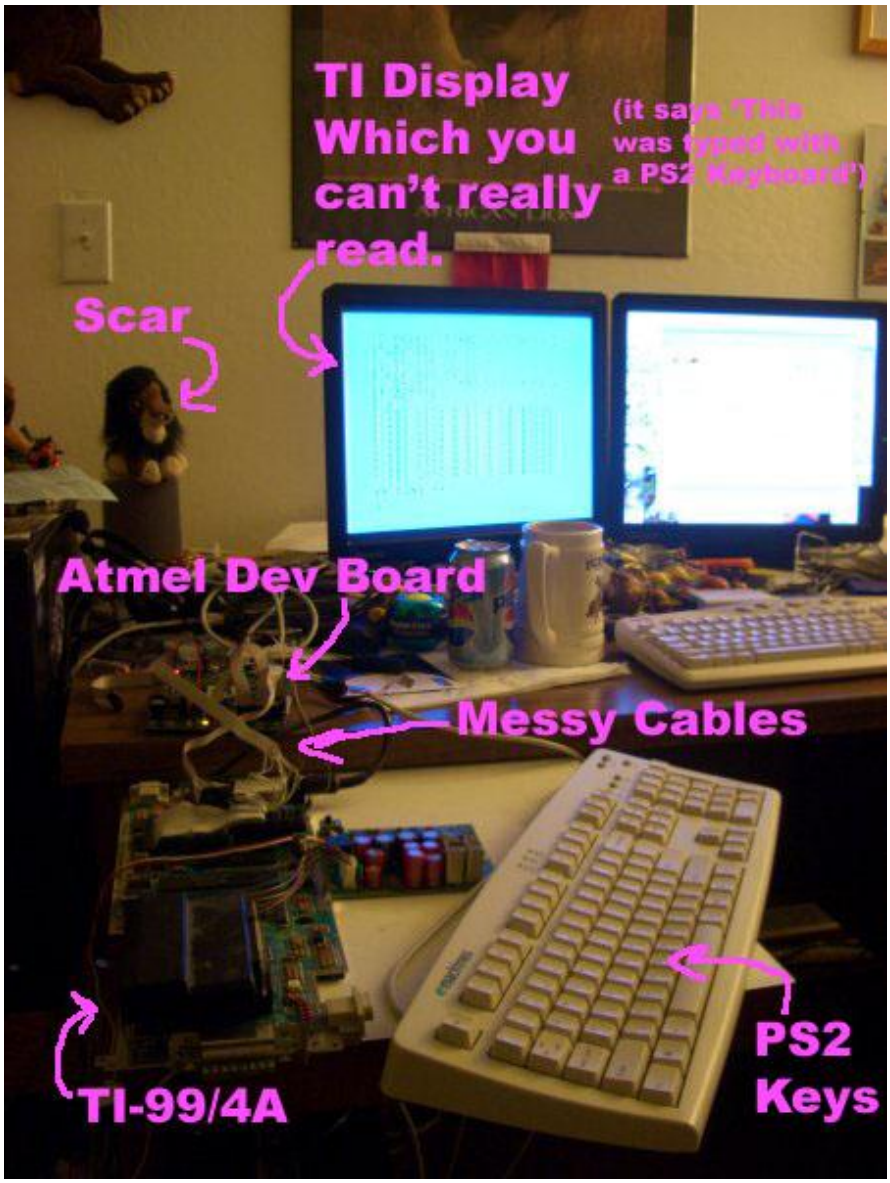
Then, click on Save As from the File menu, and enter the project name MemCheck. Enter a path and filename into the Listing and Hex object file boxes. Make sure you click the tick boxes that say "Define Registers, Produce Hex File, and Produce Listing File". Now click Assemble. The file should assemble without errors – if it doesn't there is a typo somewhere!

The assembler will produce an object file in the path you specified with the name you specified (memcheck.obj probably). You need to import this onto a "TI Disk" with the Win994a Disk Manager (use the import Text file option – it works fine).

Give the imported file the name MEMCHECK;O

Then run the Editor Assembler cartridge. Select Option 3 (Load and Run) – Enter the filename (e.g DSK1.MEMCHECK;O) the object file will load. It will ask you for another file name – just press enter. It will then ask you for a Program Name, Enter START and press enter. The program will now run.

Enjoy.



The Diary of a TI-99/4a Developer - TURSI.

I've *finally* had time to play with my Atmel programmer kit, and my first project was the one I bought it for - the TI-99 to PS/2 keyboard interface.

It took two days to get everything together, but I'm pleased to say that it's working quite well now.

The code's running on an ATMega16 at 16MHz, using two IO ports to talk to the TI and a couple of pins on a

third for the PS/2 interface. (Using the fourth port for debug LEDs). There was no real rhyme or reason behind choosing the Mega16 over other processors, but a minimum of two ports are needed, and it looks like the 16MHz clock speed is about what it takes to stay far enough ahead of the TI, though I will probably see about optimising the code some more. The code and tables are about 3.5k.

You will have already seen a picture of the project is on the previous page.

I'm pleased that all the extra keys worked the way I wanted - arrow keys work as FCTN-E/S/D/X, all the function keys (well, 1-10) map as FCTN 1-0, all the extended characters work (and so does the original key mappings! You can, for example, get a question mark either with Shift-/ or Alt-P ;). Many extended keys on the keyboard (F11, F12, volume, Windows keys) map to Control and Letter combinations so if you wanted, you could actually use them in your TI programs.

I'll release the project source at a little later date, yet - I still need to see whether I can talk back to the keyboard and control the Caps Lock LED (if not I'll mount the debug LED on the case), and I'm considering building a few to sell *if* I can get a solid enough installation.

It's a bit ironic I have a real keyboard working on the real TI before my emulator, but, the tables I built for the emulator way back in the day are what I used to get this going.

I did apparently blow out the serial port in my dev board, which is hindering debugging, but I guess that's par for the course with hardware.

And today I successfully got the backwards communication going, so the interface can reset the keyboard and control the LEDs.

Since the LEDs were working, I even went one further and got Num Lock and Scroll Lock to do something. Num Lock, as expected, unlocks the numeric keypad (not so useful on the TI maybe, but then, who am I to say what's useful?) Scroll Lock has a unique function - when it's active, it locks the arrow keys into letter mode.

Normally, if you press the Up arrow key, the PS2 adapter will tell the TI that FCTN and E are pressed - to give the normal up arrow. For games and menus where you are not expected to press FCTN, you can activate scroll lock, and then the up arrow key will simply return E. (Works the same for all four arrow keys, as well as the arrow keys on the numeric keypad when Num Lock is off).

I've been working on ghost keys appearing when multiple keys are pressed at once (not related to the TI ghost key problem, although that can technically still happen), in this case it's caused by my “metakey” flags having issues, but I think I've got all that cleared up too. If you *try* to cause weird behaviour you can, but I think overall it's in really good shape.

I've got six circuit boards for the PS/2 adapters here now.. one for me, one to give away (potentially), and four to drop on EBay and see if anyone's interested. ;) It's working pretty well now that I've worked the bugs out.

I spent a long time trying to decide how to package it - everything from hiding the entire circuit inside TI keyboards and shipping those to just selling the controller chip and leaving the rest to the user. In the end I had the circuit boards made and will assemble them and ship them with all cables attached.

To install, you'll still need to solder two wires, sadly, because the TI keyboard connector does not provide power! So you need to provide +5v and Ground. You'll also need to route the PS/2 cable outside the case (or mount a PS/2 jack to the case, however, I will be providing a cable with the jack inline). And then just unplug the TI's keyboard and plug this cable in.

What initially seemed very simple became a bit trickier when it came to handling edge cases. Single key presses on the PC keyboard which become multiple key presses on the TI presented some interesting challenges. But it's working pretty reliably now.

I don't actually know how the old XT keyboard adapter worked. But because this one interfaces directly to the CRU chip, the TI can not tell the difference between the PS/2 and the original keyboard. This means that the PS/2 keyboard should be compatible with every TI keyboarding application.

I should note that the two keys I do not support are Break and Print Screen. Print Screen actually reports itself as two key presses (extended Shift and Number Pad *), and Break reports as a series of two make and break commands. I didn't want to special case the state machine for just two keys. Most other keys on most PS/2 keyboards, even some of the special media keys, will return as *something* on the TI (most of the extra keys are mapped to CTRL plus another key).

Unfortunately, my plan to build them this weekend was blown when I finally opened my box from Digikey and found I'd ordered the wrong processors. So I have a new order in and will do that next week.

I sat down and put the first prototype board together tonight, but sadly (and perhaps not surprisingly), it didn't work first try. ;)

I did find a routing error on the boards (my fault), which was powering the crystal oscillator backwards. The error is sort of recoverable, by placing the oscillator on the bottom of the board instead of the top, the power pins are correctly placed. If another board run should ever be made, I've corrected the layout for that.

But that doesn't seem to be the only fault as even with I run on internal oscillator it's not coming up properly (and internal oscillator is too slow for full use anyway.)

A bit disappointing, since I had it working both on the dev board and in a standalone direct-wire configuration, but I'll suss out the issues and get it going. A bit of a slow start since I've had everything taken apart for so long now, but I thought I'd update.

Tonight was a good night, I got the basic problems worked out and the chip is working like it's supposed to now. Unfortunately, my testing found that not all PS2 keyboards are alike ({sarcasm}oh, surprise{/sarcasm}), so the next step is figuring out why some of them crash my chip when I set the LEDs. (I've got some basic ideas about where, looks like I'm missing a bit for some reason, but I need to do some more troubleshooting to figure out why, and why it doesn't happen on the keyboard I built the code code. For what it's worth, the keyboards that fail are older, cheaper models, but that's what most people would want to use, so I'll have to take a stab at it.

hee.. I should note that it was my software that was crashing, not some funky hardware issue. There's a timing error that's causing it to miss or add a bit now and then when sending commands, which is a bit funky since it's a single bi-directional wire used for communication. (If I wasn't setting the LEDs I'd have been done long ago. **J**).

I don't actually have a spec I'm working to, which is probably some of my problems, but the most useful pages have been these:
http://www.beyondlogic.org/keyboard/keybrd.htm
http://heim.ifi.uio.no/~stanisls/helppc/keyboard_commands.html

All AT and PS/2 keyboards support the bi-directional communication, though, that's how you reset it, control the LEDs, set the repeat rate, and so forth.

There's surprisingly more to this than even I expected. At the most basic level, you're right. But the PS/2 keyboard has scan codes for every key - these are converted to the appropriate keys on a TI keyboard. Some keys on a PC keyboard (such as the backtick `) map to *two* keys on a TI (FCTN-C). Then we have keys like the pipe (|) which is a shifted key on the PC, but a FCTN key on the TI. Some PC keys send multiple bytes (extended keys). And then there's dealing with the release codes on the PC to tell you when the key is up again.

Mapping to the switches was easy in theory, but I'm doing the mapping in software, not hardware, meaning the

processor has to read the column select from the TI, and put the correct row data on the output pins before the TI reads them. I'm not 100% convinced this part is fast enough, but it does the right thing 99.9% of the time. When it gets it wrong, you get the incorrect character typed. (Of course, an addressable buffer or maybe dual-port RAM could be used to solve that, but I wanted to see if this could be done on one chip. ;))

All that I knew in advance, expected, and actually was able to reuse some Classic99 code to get it mapping nicely. No worries there. ;)

The next issue was dealing with auto repeat. You can't turn auto repeat off on a standard PS/2 keyboard (or any normal AT keyboard). So I had to be able to detect auto repeat and ignore it, so it didn't mess up the TI's auto repeat.

A baffling issue was dealing with the fact that, on the PC, it's common to type by overlaying your keystrokes - ie: you press the next key before fully releasing the previous key. This was something I hadn't even considered. And it wouldn't be a big problem, except for the single->multiple key mapping I discussed above. A contrived case I used to test was to play with the left and right arrow keys:

PC: LEFT down, RIGHT down, LEFT up, RIGHT up.
TI: Left, Right & Left, D

... D? Why D? Because the 'LEFT up', which mapped to 'FCTN S', released the FCTN and S keys, but RIGHT still needed FCTN depressed. So I had to add ref counting to the shift keys. That took a while to get right (and it's not 100%, but it's self-calibrating and recovers itself nicely).

And currently, I'm dealing with the rogue keyboards and unexpected hangs. I'm getting closer - as of today it works 100% on my newer keyboards and about 80% on the old one, and I've isolated when the old one quits working, I just need to check the sequence that's killing it. (I'm suspecting a parity bit being wrong, but it's late and I'm tired).

Currently the program is running a lot slower than it should, too (though response time for normal typing is quite good). Once it's working 100% I just have to optimise a bit (largely removing debug code), and then we're good to go. The main issue I need to solve is the delay period between receiving data and sending a command - going to take a look at how other people have done that because I need about 2/10s of a second delay before I send anything if I want it to work, and that's noticeable when you tap Caps Lock.

Originally I'd hoped to sell these things, but at this point I'll probably just sell off the prototypes for my cost and post the rest on my web page. I've spent too much on little errors to ever make my money back.

Definitely doing it for love, but the professional circuit boards were in hopes of making it presentable enough for sale, though it doesn't address how to mount the product. ;) Seeing the RAVE showed me the right way to get power into the board, so if I did a rev 2 I think I'd want to do the same thing as they did with the power cable. It's a shame I messed up the clock (and fried one in the process), but at least the workaround was easy and works. I considered kit form, too, but there's so little assembly needed, really.

One feature of my translation system that I was pleased with is that even though all the keyboard re mapping works, the original TI key presses still work, too, so if you want a question mark, you can type Shift-/ *or* Alt-I, and still get it. Helps for those who are still used to the TI layout.

I'm not sure how easy it is the support VGA output from the TIM - my understanding (and I may be wrong) was that the 9958 only supports 15kHz sync, which VGA monitors can't support. You're right, though; the supply of 15kHz RGB monitors is fading rapidly. (You could get an old JAMMA arcade machine and put your TI in that to get a larger monitor? **J**)

Replacing the video chip with an upgraded one that could do extended modes and VGA output was one of my dreams, but the current speed of embedded processors means that I won't be doing it without learning FPGAs. I

was trying to find the 9918A core that's already out there in the MSX world, apparently, but I had no luck, and starting from scratch is probably not a good project for my first learning attempt. ;)

Still, I'm investigating ideas for a next project already, hopefully I'll have time.

I was able to make substantial improvements in my timing. Specifically, I was clocking my control data on the wrong part of the waveform. Also thanks to finally knuckling down and recording the waveforms (by feeding the clock and data lines into my sound card and recording it with Sound Recorder), I was able to find some of the bit counting errors and fix them as well. That recording *also* showed me why my clock inhibit wasn't working (it was going high instead of low), and I was able to get that going too. Finally, I implemented a few of the last controls like retransmit on request (I have one keyboard that gets it's clocking confused while I'm sending control data and requests me to resend, so I needed that), and finally came up with a reliable way to ignore the Break key (which is a bit of a pain in the butt). However, I'll still see if I can map Control-Break to FCTN-4, I think that'll work.

The system is running *very* reliably now, and works on all the keyboards I have (except one which seems to be defective and is now in my trash ;)).

Still a few more tests to do! When running in debug mode, I could trigger a bad scan about one in 100 key presses by rapidly toggling between certain keys (it comes down to timing on the TI side - some keys take less time to detect than others. And you really have to hammer on it, but, if I can easily reproduce it I do want to try to fix it.) Experimentally, I dropped a 20MHz clock on the chip instead of the 16MHz it's rated for, and that does in fact appear to work. On a release build, with quick testing of 200 keystrokes or so, I never saw the problem. I still need to test the release build on the 16MHz clock to decide whether to go with the 20 instead (which is an over clock). (8MHz is definitely too slow).

But it's really promising. It's looking very solid. Once I settle on the clock and test it on the board, I'll have to play a few games and make sure that's good. But I'm pleased with the progress so far! Definitely close!

And we're up! I've completed my testing, and got the first board (besides the prototype) assembled and installed into my own machine tonight. I spent about 30 minutes typing little programs into BASIC and Extended BASIC, and playing Munch Man, and it performed flawlessly the entire time. Even better than I'd hoped for.

I've posted photos in my gallery. After some sleep, I'll work out what I'm doing with the spares.

I've posted photos in my gallery. After some sleep, I'll work out what I'm doing with the spares.
http://www.harmlesslion.com/gallery/ti99_ps2
Just take a look.

It's finally done. A few last minutes tweaks and bug fixes and I'm quite happy with the keyboards now.

I had found a problem a few days ago with one of my design decisions - the shift keys do not map to the real TI shift keys. This is because some PS2 shifted keys are FCTN keys on the TI.

However, when I finally installed it in my real TI, I remembered that the RAM Disk has a feature whereby you hold shift to bypass any boot menu.

Since I didn't want to break the shift handling code, I remapped the left Windows key (which was previously Control-Y) to the TI's shift key, and it works perfectly for skipping the RAM disk. I also made several improvements to the start-up and command handling, and so the devices are shipping with version 1.2 of the firmware.

Spent some time last night pricing out parts... a run of 20 is going to cost me about $650... I certainly hope there's enough interest. ;) I'm planning to price the kits about $45, which is a bit less than the two prototypes on EBay went for. These are for full boards with a solder mask and silkscreen, socketed processors, and all.

The new code is nearly done. I'm working on a mechanism to permit the TI to upload an alternate scan table, which will allow custom remapping of most keys (some are hard coded and I won't override those).

Also, took a suggestion for reset - the Rave XT apparently allowed you to hook up to the console reset line (on the cartridge port), and Alt-F9 would toggle it. I've gone one further, and mapped Alt-F9 through Alt-F12 to generic outputs - you can hook them up as you like on your machine (suggested uses - reset and load lines ;)). These aren't tested yet though.

The new board layout includes pass throughs and jacks for both power and keyboard, removing the need to solder anything for installation. A jack is also intended for connecting the PS/2 jack.

This is where I'm hitting a snag - I don't want to build all the cables. ;) Does anyone know a source for cables with the same pin count as the keyboard cable (single row, 15 pins)? We could use IDE or floppy cables, of course, but the potential for error on installation is higher and they're far larger than we need. I would need 20 of these.

Also, I need a source of cables for the PS/2 cable. I came across a couple for older Intel motherboards:
http://www.worldofcables.com/oscatalog/CablesProduct/FLT-2900.htm
http://www.cables4computer.com/Product....20.html

Of those, the 'mouse' cable would be the most ideal, since it has the smallest pin connector, but I would need a source of 20 of them to do my run.

And, finally, of course, I would need a run of 20 cables with the thick 4-pin connector on each end to handle the power supply.

Any help sorting out the cables would be appreciated. If worst comes to worst, I may leave you all to sort out the cables in your own way, but I'm not de-soldering cables from dead keyboards again, oi.

I've released my PS/2 keyboard project with the fixes I planned for Rev2, except for the programmability feature. I found some old conversations talking about the 9901 being a bit fragile to reverse voltage, which my plan ran the risk of doing, plus the software for reprogramming it was a serious pain.

The following improvements are in the new version over the one I shipped:
-Rev 2 board released (not yet tested)
-Remapped Esc to FCTN-9 (back)
-Remapped both Windows keys to Shift on TI
-Optimised code a little
-Flipped Page up and Page down between Fctn-4 and Fctn-6 to correct E/A behaviour
-Added output pins - active low when Alt F9 through F12 pressed
-Bug fix: when going from keys shifted on the TI to not shifted on the TI, release the old one first (ex: typing '"s")
-Reset keyboard controller with Control-Alt-Delete
-Bug fix setting column table more than needed when reading -1 from scan code table
-No longer set Port D to output at startup (wasn't needed anymore anyway)

The output pins are the big one. There are four output pins, triggered in turn by Alt-F9 through Alt-F12. Normally the output is tri-stated, when you press the key sequence, the output is driven low.

The most useful uses for these I could think of are to drive reset and load, rather than using a button. I've no idea what the other two might be used for; how about driving some of Thierry's hacks, speed control, etc?

I've posted a zip file containing the .hex file as well as the schematic and PCB layout for http://expresspcb.com at my page at http://harmlesslion.com/software/adapter

The new PCB layout includes two ports for power and two ports for keyboard, so you can use cables to pass through power and keyboard. I'm not sure if the original keyboard will work, since the Atmel is continuously driving those lines, though. Also, I had no luck finding cables for the keyboard. You could probably use a floppy cable if you didn't mind it hanging over the side of the connector, though, and were careful about lining it up. (It's what I'm using on my test board.)

Also, the PS/2 connector I laid out does not match any of the connectors I was talking about above, necessarily, as I never got my hands on one. So if you use one, check the pin out is right.

At this time, I don't have any intention of doing a run of boards. I am selling programmed chips to anyone interested (and they seem to work fine wire-wrapped or even point-to-point if you don't mind ugly bits). However, I'd be happy to work with anyone interested in organizing a run.

Found a notable bug in the PS/2 Keyboard adapter I released -- when in place, the joysticks don't work. This is because it does not turn off it's outputs for unrecognized control patterns (ie: scanning the joystick lines), and so the joystick inputs can't be seen by the 9901.

There's no workaround that I can see for this. I've been toying with a new release, though, that piggybacks the 9901 instead of needing a circuit board. *IF* I get time to do that, I will try to ensure joysticks work.

This diary was extracted from the 99er.net forums.

History of project:

v1.2
-Initial release, rev 1 board

v1.3
-Rev 2 board released (not yet tested)
-Remapped Esc to FCTN-9 (back)
-Remapped both Windows keys to Shift on TI
-Optimised code a little
-Flipped Page up and Page down between Fctn-4 and Fctn-6 to correct E/ behaviour
-Added output pins - active low when Alt F9 through F12 pressed
-Bug fix: when going from keys shifted on the TI to not shift ed on the TI, release the old one first (ex: typing '"s")
-Reset keyboard controller with Control-Alt-Delete
-Bug fix setting column table more than needed when reading -1 from scan code table
-No longer set Port D to output at start-up (wasn't needed anymore anyway).

For anymore information regarding this project please contact. **turi@harmlesslion.com**

teaching Biology and Plant Biology at the Lima Campus of The Ohio State University ever since. In 1975 Charles married a Lima woman with four young children, and Charles and his wife had three more children. This made Charles a proud father of seven. In 1982, Charles and his wife were searching for an appropriate home computer for the family. They attended a Commodore Vic 20 demo

at a local store and were very unimpressed when the computer ran out of memory about half way through a Basic program designed to plot a circle on the screen. Therefore they purchased a home computer that could do the job nicely, a TI-99/4A.

Biography prepared by Glenn Bernasek
from personal recollections and from biographical material provided by Charles Good.

Inducted to the TI99'ers Hall of Fame on December 24, 2004