Dr. Erik Baigar
email: *erik@baigar.de*, Homepage *http://www.baigar.de*
Gotzmannstraße 5/I
D81245 München/Aubing
Tel.: +49/(0)89/63496363
SMS: +49/(0)173/3723182
Fax: +49/(0)1803/551863496
AB: +49/(0)721/151479577

Alternativ: Richard-Strauß-Str. 9
D87616 Marktoberdorf
Tel.: +49/(0)8342/95876
email, SMS, Fax, AB wie oben

Aubing , 3. August 2007

-

Dear valued reader,

this report is a slightly modified version of the report issued 3/2007 in the cctech mailing list, dealing with vintage computers, their conservation and operation. For the resulting discussion, please see http://www.classiccmp.org/cctech.html.

This report describes in a chronological order the steps, problems and thoughts which led to make an old computer from one of the first Panavia Tornado aircraft operational again. The computer is one of the typical black boxes (see above) and the type label reads

.

During this project the circuitry was investigated, support and debugging tools where built and programmed, the assembler language was decyphered, a macro-assembler written and last but not least you will find a “Hello World“ example with an externally connected LCD display.

Those who are interested, please enjoy and do not hesitate to contact me with any hints or questions,

best regards,

Erik Baigar

Report on the Resoration of the "Programmer Electronic Control":
(Status of the project from the 3/15/2007 with small changes and notes.)

Dear vintage computing fans,

in this posting I want to report the community on the restoration and reverse-engineering project I did over the last two years as a hobby. Maybe someone is interested in this or even has valuable information.

Beginning of 2005 I got this small airborne computer via eBay and I bought it mainly because of the 96kbit of core memory inside. Those days some of these boxes have been sold by ABEX and it is still on their web page:

http://www.abex.co.uk/sales/electronic/other/programmer/item.htm

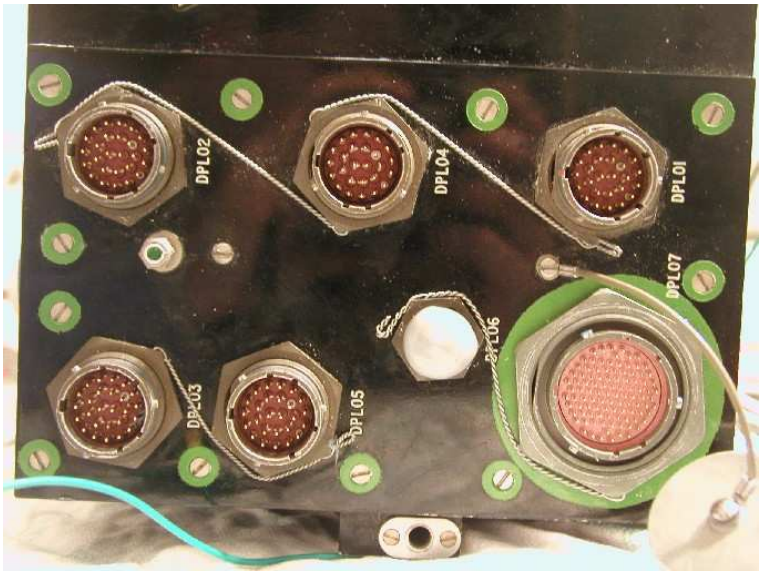
But they do not have them any more :-(:-(:-((see below)

In this forum William Maddox mentioned an eBay auction regarding one of these in his Nov/17/2003s posting "Latest eBay finds, including a real museum piece". After getting the unit I disassembled it and noticed that this must be a complete computer (size a little bigger than a shoe box) and I tried to collect information (e.g. on pprune.org and usenet's rec.aviation.military). But as expected no one was able supply significant information so I started

PHASE0 (January 2005):

Inspecting of all PCBs I drew basic wiring schemes and located the major parts of the system as timing generation, core logic, core drivers, microcode, ALU and so on. It became clear: THIS IS A 12-BIT MACHINE[1].

The number of connectors of the box is limited (one for power, 5 plugs connecting to some kind of serial differential links [2] and only one very big 135-pin beast). Since the big plug was the only one equipted with a protection cover I developed the hope, that everything neccessary to operate the unit is within this one box I have got and not distributed over more devices.



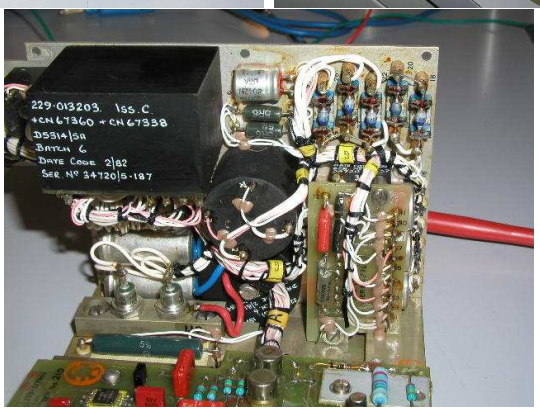
Here I made the decision to continue the reverse-engineering with the goal to make the unit OPERATIONAL.

PHASE1 (March 2005):

I put the core memory aside and investigated the power supply. Since I do not have got the 110V-tree phase AC of some weired frequency, the small included switch mode power supply seems to require, I instead figured out what voltages it generates and supplied them from external supplies (5V, 8A; -5V, +12V and variable voltage for core). I also located the "power good" signal and applied this, too. See pictures:

[1]Note: Here I had some hope, that the command-set will be inspired e.g. by PDP8.

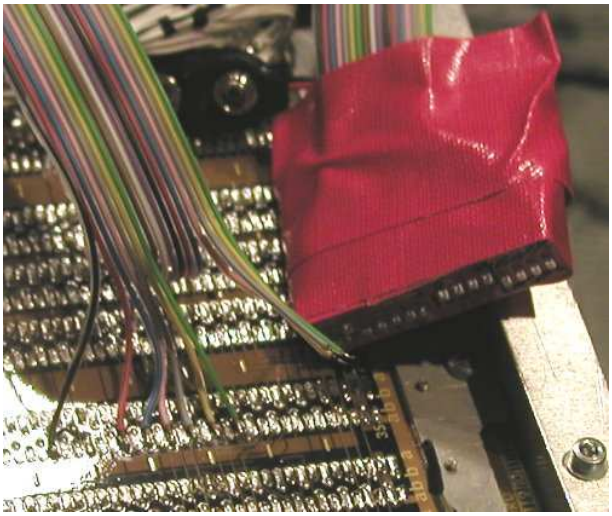
[2]One had a dedicated receiver, the other four links are MUXed to one common receiver/transmitter.



Now I was able to supply power and observed with the oscilloscope that there is activity after power on. Now I thoroughly examined the core section until I had a good idea how it works, leading to

PHASE2 (April 2005):

Supplying data to the data lines by using a 12-bit DIP-switch. This allowed me to make the processor reading the same word of data over and over again and therby stepping through the whole address space.



With this I characterized the core memory timing, I tried to identify important lines, refined my schematics and connected a logic analyzer to all vital signals. In applying different patterns, I already discovered read instructions, 2-cycle instructions, read-modify-write instructions and some instructions freezing the unit and requiring a power cycle (not only a reset pulse[3]):

[3]Those days I could hardly believe, that there is no ROM and that there are commands halting the computer totally. The horror scenario would be, that during flight a bit of the core gets wrongly rewritten to memory leading to a freeze of this box. This would require to shot down all Systems, reprogram the unit and power on again - incredible for a aircraft! Now (August/2006) I know: It is really true!

In parallel I did detective work on the stickers and posted my information to a thread in rec.aviation.military. This lead to the suspicion that the unit is from an early tornado aircraft (probably only from one of the first prototypes) and was responsible for controling some kind of display:

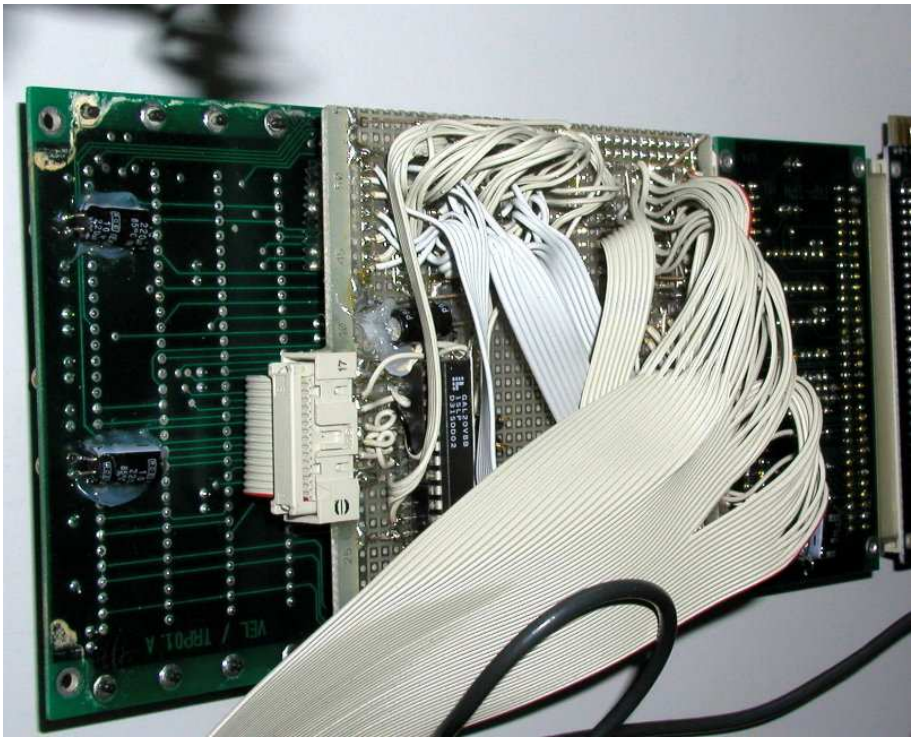
Anyhow these experiments showed me that (a) the machine is not completely dead and that (b) the command set is not compatible (this was my hope first) with the well known PDP8 or any other known 12-bit machine. So what was needed was

PHASE 3 (Sep 2005)

an in-circuit analyzer and the possibility to read and write core memory. With quite a amount of reverse- engineering, thinking and try-and-error I discovered a DMA mechanism which in my opinion was used to load software onto the unit via the big plug - but this mechanism was probably unused during normal operation (big plug sealed off).

Since I hate any PC-stuff and I needed a powerful and real-time capable tool to generate the appropriate timing for writing into the core, I decided to use a vintage transputer[4] from a ParsyTec-Megaframe, running the operating system helios (beautiful unix-type parallel OS) which I connected to the Programmer Electronic Control using a homemade interface PCB and flat ribbon cables:

[4]Transputers are somewhat like modern microcontrollers with four high speed links, integrated memory, FPU and some peripherals but with a different programming philosophy than modern processors. They where used in massively parallel computers with 1024 and more nodes, which where connected via their links. For more details, see Ram's Transputer page: board http://www.classiccmp.org/transputer/



During this process I refined my schematics again, analyzed the ALU and the timing generation and made more professional connections to the logic analyzer:

Here much efford went into establishing all the tools needed - but it was worth the efford, since in this environment the complete cycle of

- finishing a modification (solder iron),
- booting the helios (UNIX!),
- triggering the logic analyzer,
- launching the test software,
- read out the results from memory, oscilloscope and analyzer,

only takes about 20 seconds. To be on-topic here: Vintage techniques involved in the setup:

- Sun Sparc 20 for GPIB-interfacing to logic analzyer, and for booting the helios client via BBK-S4 board.
- One T805 entry-node of a Parsytec PowerXplorer[5] for development and ompiling of the helios software. This is booted from the SunSparc 20 as well.
- SGI Indigo2 as frontend and data processing.

Soon the timing matched what the processor does on its own during memory access.

PHASE4 (Nov 2005):

I inserted the core memory and tested my tools on word 0 only (read, write) and it was perfect. So the hope was to recover information still sitting in the memory. But after doing tests on word 0 and than reading the remaining 4095 words, I only discovered some type of test pattern beeing inside the memory. But now the homemade tools worked very well allowing cycles of

1. write test pattern,
2. let unit execute some cycles,
3. observe dataflow (HP1661A logic analyzer) and read out core memory,
4. put data into files for later analysis (dinotrace).

One of these cycles takes around 15 seconds.

[5]This type of machine is a interesting beast, too. Since somewhen Transputers where outperformed by always faster processors, Parsytec made the clever trick to use the established environment from the Transputer based parallel computers and supplied a PowerPC processor to each Transputer. In this way the transputers and all the readily available software could be used for communication and IO, whereas the faster PowerPCs where used to do the number crunching.

PHASE5 (Jan 2006):

Doing lots of cycles I discovered lots of commands and instructions. So this led to defining an own assembler language. In parallel I implemented an assembler[6] understanding the already discovered commands and generating files suitable for writing into the core memory. So it became more and more comfortable to write test programs to analyze new instructions.

In this manner I analyzed systematically the whole 12 bit space to discover jump instructions, arithmetic instructions and so on. In summary this is a VERY ARCHAIC and unusual machine:

- THE MACHINE HAS 1 ACCUMULATOR
- THERE EXISTS ONE INDEX REGISTER (indirect addressing)
- CALCULATION IS DONE IN 1-complement!!!
- NO CARRY-FLAG (essentially all arithmetic only uses bit 0-11, bit 12 is sign/carry)
- SHIFT-COMMANDS use a hidden 12bit SHIFT REGISTER[7]!
- There is a MULTIPLY-INSTRUCTION (did not expect this!)!!!
- Only one instruction for ABSOLUTE JUMPS using a very delicate table jump algorithm. This can be used to simulate someting like a CALL (no stack on this unit!) - Really ugly!

Doing this it became obvious that the unit has problems in accessing the upper 4k words of memory. In doing so it freezes, requiring a power cycle to recover. So far I was not able to locate the problem and unfortunately all attempts to get hands on a second unit for swapping the micro code and sequence generator failed.

It must be a problem with micro-code since the unit can execute code in the upper half, but data access fails[8].

In writing bigger programs I encountered problems in my transputer setup related to refresh cycles on the transputer PCB:

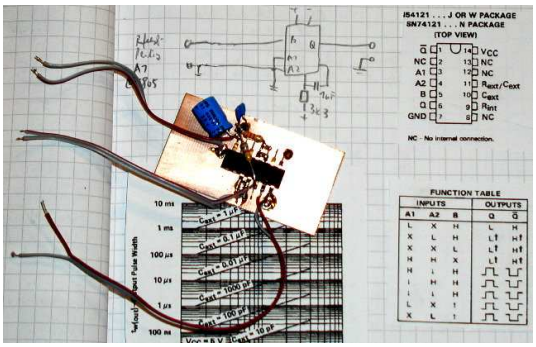
[6] Today this is a two pass macro assembler, written in TCL/Tk. It understands about 25 different commands and the indirect adressing mode, too.

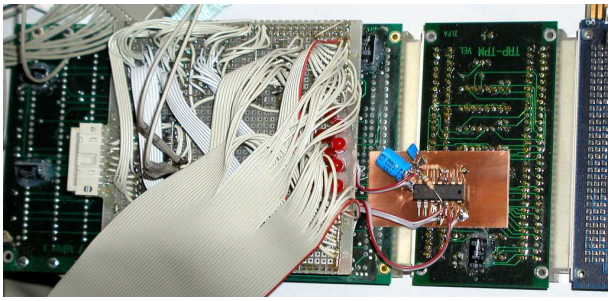
[7] Today I know, that this hidden shift register is identical to the above mentioned INDEX register. I discovered this since some of my programs failed due to a mystic changing INDEX register ;-)

[8] Since the unit can execute code in the upper half, the core loginc must be OK.

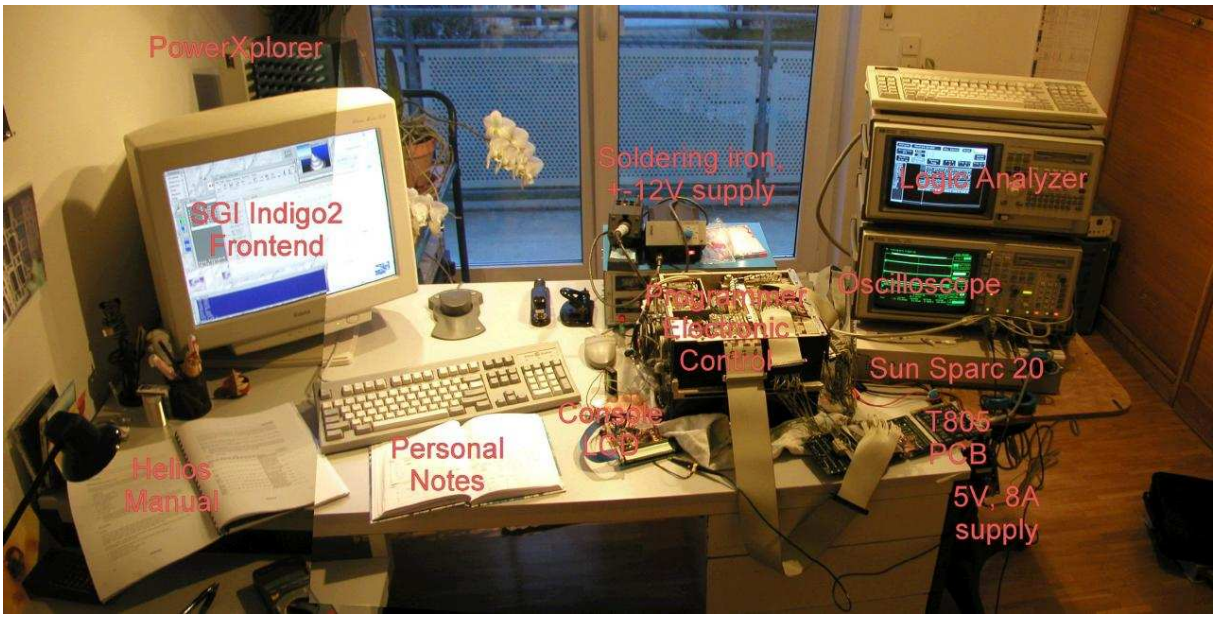
PHASE 6 (Nov 2006):

The box itself runs at 1.2us cycle time for core accesses, but the core timing has to be accurate to <30ns for proper operation. The built-in refresh of the DRAM on the transputer lead to an error-rate of approx 1% in my tools writing and reading core. So a small PCB had to be added to the transputer setup to detect this refresh cycles and to synchronize the helios software to them:





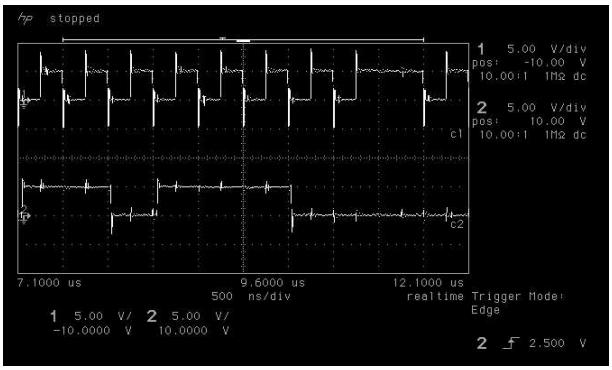
With this the error rate dropped to 0.004% which is more than enough for the application. From there on the complete setup did not change any more:



Today the assembler understands approximately 25 instructions, generates appropriate warnings and errors and warns if an instruction will freeze the unit (TCL together with m4 makes it very easy to write such a macro-assembler and tcl past 8.4.7 is reasonable fast to lead to very short compile times even on my slow hardware)...

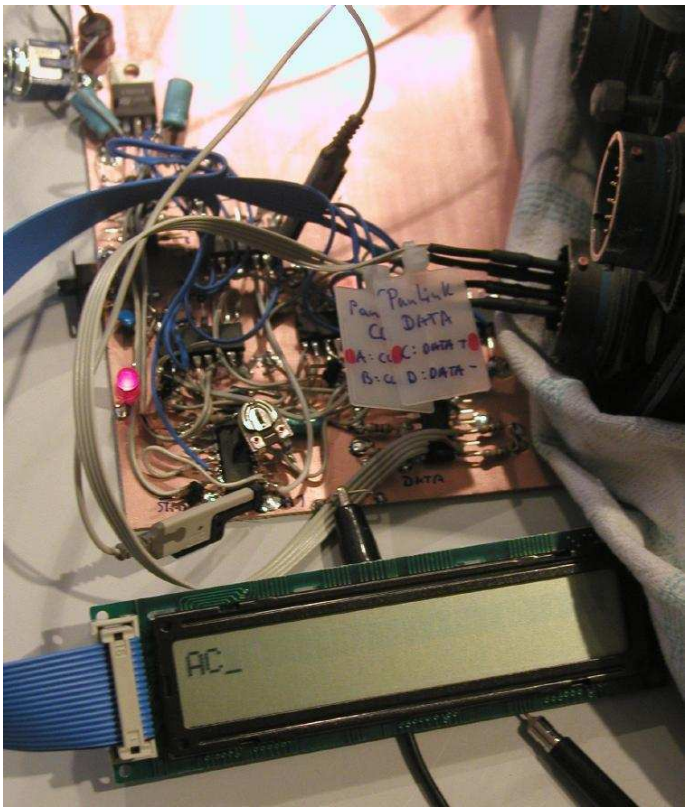
PHASE 7 (Jan 2007):

A closer look at the serial links revealed that they are some type of serial, differential SPI buses, allowing messages of 12 bit + 2bit identifier + some kind of address. Not all identifiers can be sent out on all plugs. These interfaces (as my limited understanding of this is) should be Panavia-link or short panlink interfaces. But sorry - I do not know any details about this:



The latest action was, to build a small interface PCB to decode the messages, i.e. seperate data from address and identifier and generate CS and CE signals. With this it is possible to connect different devices to these serial links in the future.

The first device I connected two weeks ago was a small LCD display and look - it works:



In the assembler, this “Hello World“ looks like this (including initialization of the display):

```
reset: MTA 0b00111000 ; Display Initialisieren (8bit, 5*8 chars)
       WDPL3.ID0  0   ; Befehl ans Display (RS=0)

       DELAY(4200)    ; Warten 4.2ms bis Display initialisiert

       MTA 0b00000001 ; Display Loeschen
       WDPL3.ID0  0   ; Befehl ans Display (RS=0)

       DELAY(1640)    ; 1.64ms Wait bis sich Display geloescht hat

       MTA 0b00001111 ; Cursor an, Display ist jetzt leer mit Cursor
       WDPL3.ID0  0   ; Befehl ans Display (RS=0)

       DELAY(110)     ; 110us Wait, dann hat sich das Display konfiguriert

       MTA 0b00000110 ; Auto-Increment-Mode, d.h.Cursor wandert
       WDPL3.ID0  0   ; Befehl ans Display (RS=0)

       DELAY(50)      ; Nach 50us Wait ist der Cursor an und alles bereit

       ;
       ; Text ans Display ausgeben
       ;
       MTA Text        ; Zeiger auf Text
       STA Index       ; initialisieren
loop:  LDI Index       ; Naechsten Buchstaben holen
       LDA 0           ; und Daten ans Display (RS=1)
       RJAZ Fin        ; falls nicht 0 (das waere Ende)
       WDPL3.ID0  1    ; schicken. Danach
       INC Index       ; Index erhoehen und warten...
       DELAY(50)       ; 50us Wait
       rjmp loop       ; Naechsten Buchstaben

Fin:   rjmp Fin              ; Endlos-Schleife!
```

In this code fragmet, DELAY is a macro defined within the assembler and consisting of a simple loop.

So this is the current state on beginning of March 2007!

F U T U R E :

The following things are still on my to-do-list:

1. The unit has got a 12bit-start/compare timer which I currently do not have access to.
2. Approx 40 of the possible bit-patterns are not valid commands now - either I do not understand their meaning yet or the microcode is somehow defective.
3. I can read data from the plugs, but I did not discover a sync mechanism: When has data completely arrived???
4. There is still the problem of the unit freezing in accessing the upper memory half.
5. On each serial link, there seems to be an interrupt signal. I do not know how to enable the interrupts and what action they cause (I suppose they cause a jump to a certain address in memory).

So thank you for reading all this junk of bad English and maybe something was interesting for you. For me the work on this unit was a hobby and real fun. I spent around one day a month on this unit and my big hope would be to

GET HANDS ON A SECOND

OF THESE "Programmer Electronic Controls"

Best regards,

happy computing,

Erik.