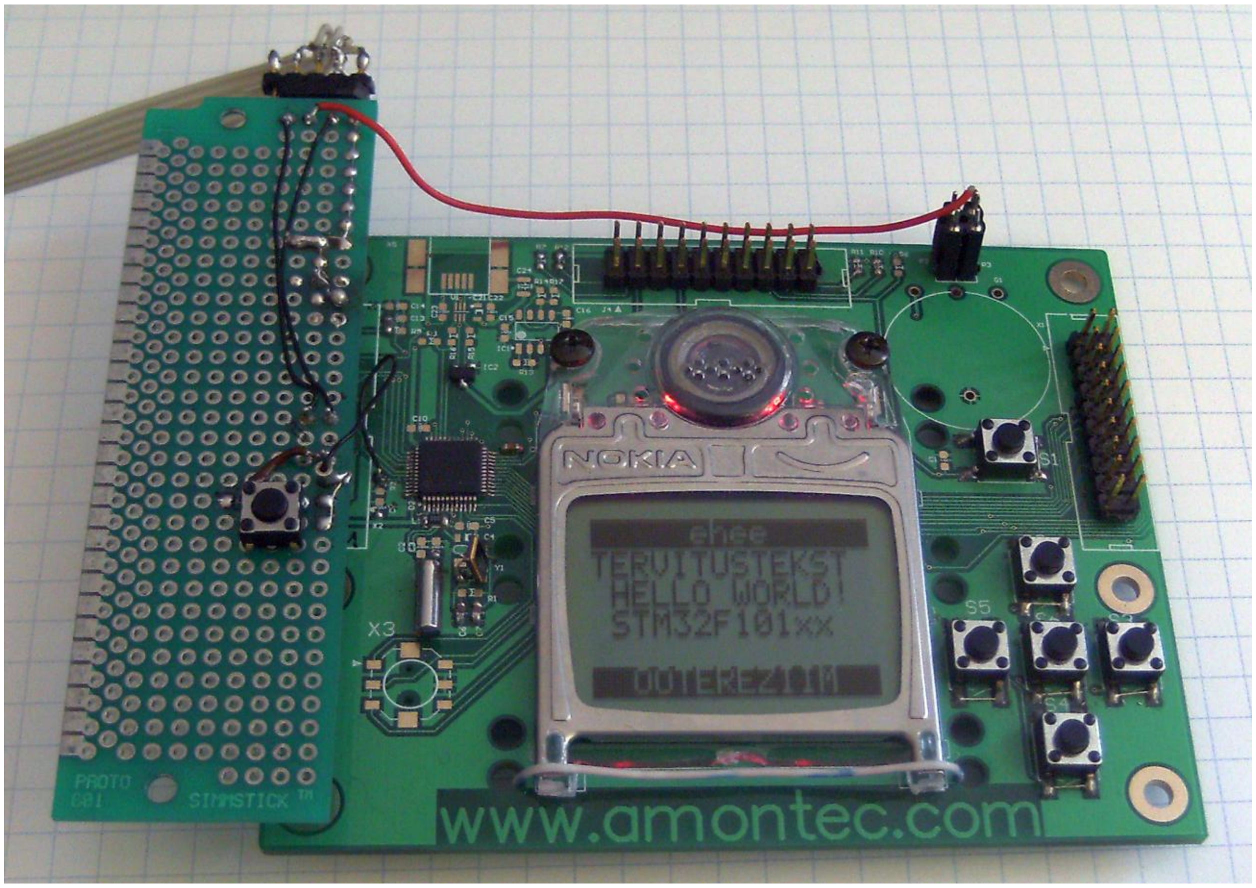
Antti-Brain

Issue 8

March 2009



Hello, from Amontec STM32!

Revised on March 31, 2009

Editorial

This time I had to rip-out some unfinished articles. Too unfinished ones, well they are saved to be included in the next issues.

Antti Lukats
Antti.Lukats@googlemail.com

http://groups.google.com/group/antti-brain

Cover Story

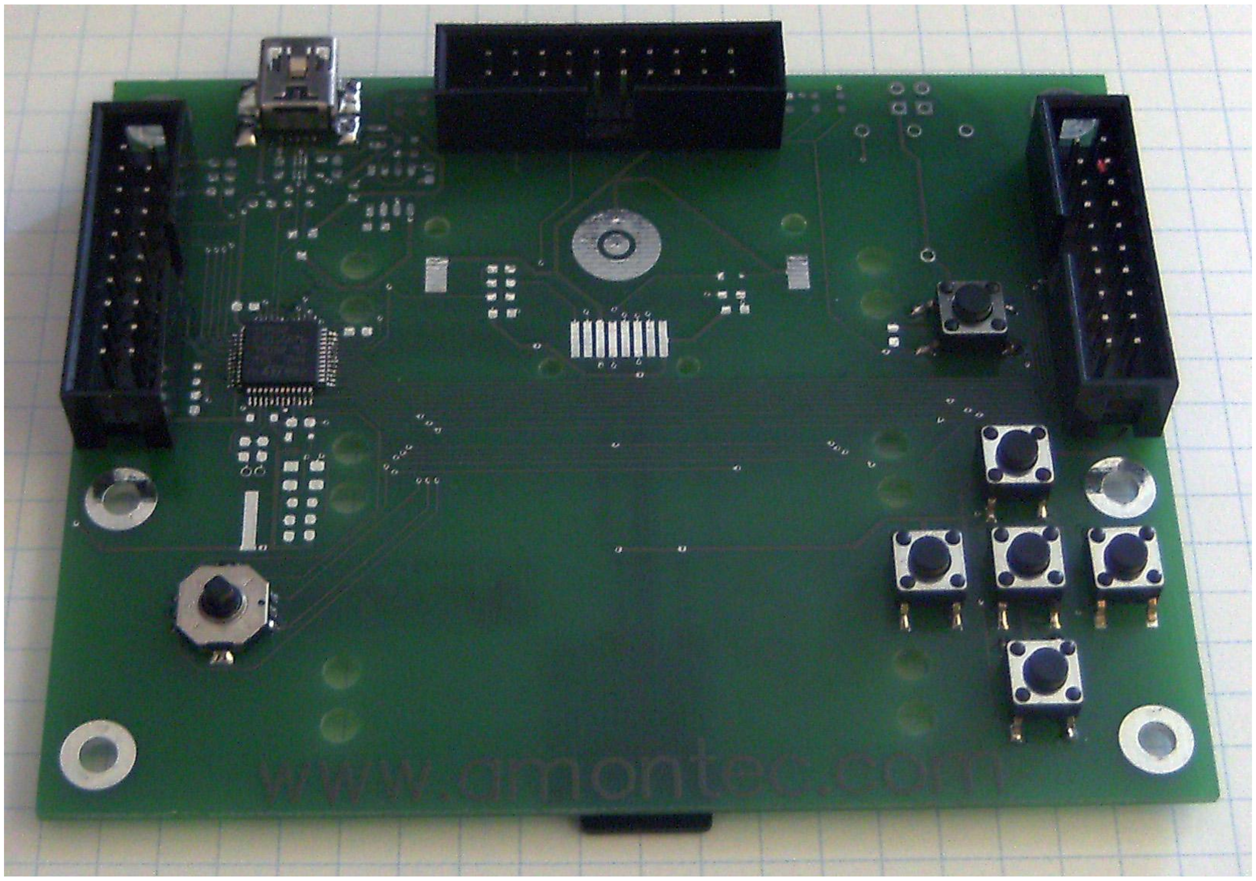
Amontec Cortex-M3 Evaluation Board (with STM32 from ST).

Features

- Arm Cortex-M3 MCU (STM32F)
- Mini USB (for power and charger also)
- 20 Pin ARM JTAG (mates with all Amontec JTAG Cables)
- Two 20 pin user extension headers
- Micro-SD Socket (at the bottom side)
- 12MHz crystal
- 32KHz RTC crystal
- Monochrome Graphical display (Nokia 3310) with Speaker
- LiOn Battery and charger
- Reset Button
- JOY Buttons (on the right)
- Joypad (on the left)

Pictured on the cover page is the very first prototype, "rev A" with too many small but nasty issues. Like the buttons had all short circuit, and the special pad for the speaker (the big round seen on the other photo) had solder mask on it.

The very first board was made with the UART Bootloader, it does work without the crystal being connected what is pretty nice feature. Well the UART Bootloader test was just to get it tested, there should be no reason to use UART Bootloader as JTAG port offers much more features for programming and debugging. So the next test was with JTAG, using Amontec JTAGKey.



PCB Revision B – most of the problems are fixed. Actually only one did remain, the USB connector footprint has no holes for the plastic fixture pegs.

And YES, the small black thing at the lower edge, bottom is micro-SD Card. And YES, the small application that says hello on the cover page was loaded from the micro-SD card using the factory pre-programmed Bootloader.

This may sound like nothing special, but well, I had the rev A board working for a long time, and many times I maybe would have used it for some small test, but then each time I was too lazy to connect the programming cable. And each time I did think, if it only would have a Bootloader from the SD Card!

Now it is has ☺

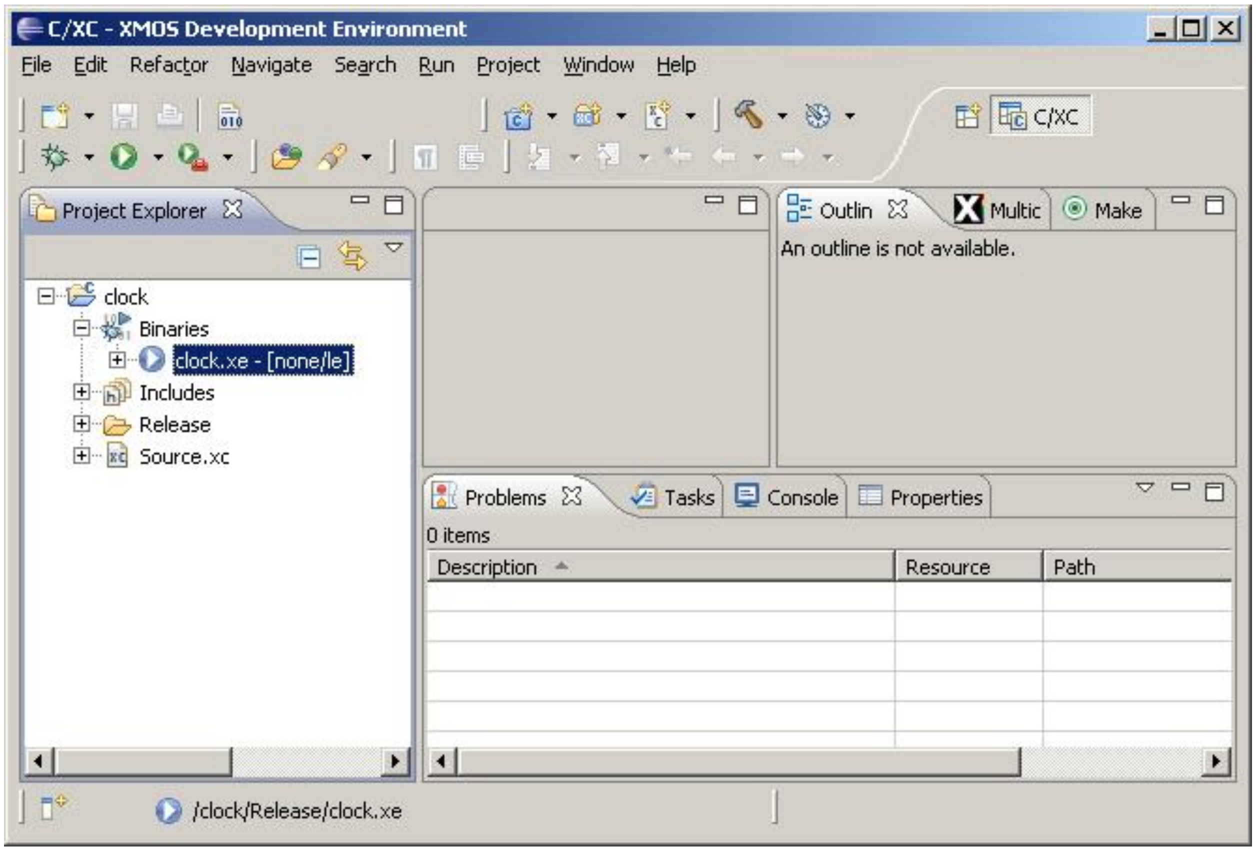
And it's ready for rev C, that should be production ready version.

XMOS

Unfortunately I missed XMOS booth at Embedded World, but when I returned from the show I had the XC-1 board waiting for me. My son said “COOL!” the very instant I plugged the USB cable, and both my kids had some fun with the preloaded music demo.

XC-1 Getting Started

Getting the first application compiled and running on the XMOS XC-1 board was boringly easy.



An outline is not available.

0 items

Here it is step by step:

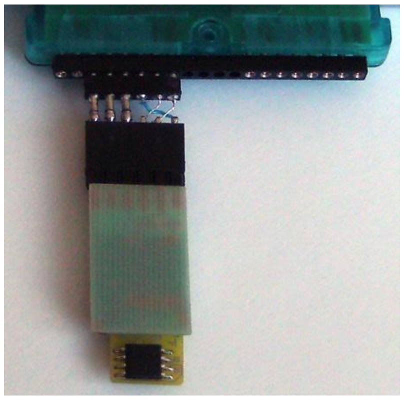
1. Download and install XMOS Desktop Tools (takes about 10 minutes)
2. Download some XC-1 Example (unzip it outside the workspace folder!)
3. Start the IDE
4. Create New Project
5. Import the .XC file from the Example [from step 2]
6. Build..
7. Run..

And the newly compiled application was indeed running on the XC-1 hardware. Please note there was nothing needed for the USB drivers or any hardware setup. Also no need to change any source code or any parameters or configuration options. One note: you have to select (highlight) the XE file in order to be able to run the application on the hardware (the launch menu is not visible otherwise).

The above wasn’t really my own XC design yet. Still to be made ☺

Silicon Blue FPGA Part III

For easy programming of the SPI Flash I made special software for NDS, well while troubleshooting it, first attempt I wrote too few bytes to Flash, the result was that the L04 did turn off the pre-configuration pull-ups but did not start correctly (as bitfile was only partial). After fixing that the new flashed SPI memory did work as well. For this test I just made copy of the 1MByte flash that I had programmed with iceMAN65. Silicon Blue own tools did not seem to work with AT45DB161D, but my own tool programmed it Ok, and it also worked to properly configure the SB FPGA. Hum, I still need conversion from the SB kind of hex to pure binary for my SPI flash tool. Ok, the hex is standard Intel hex, but the bin is some weird ascii binary. My programming software only reads pure binary files.

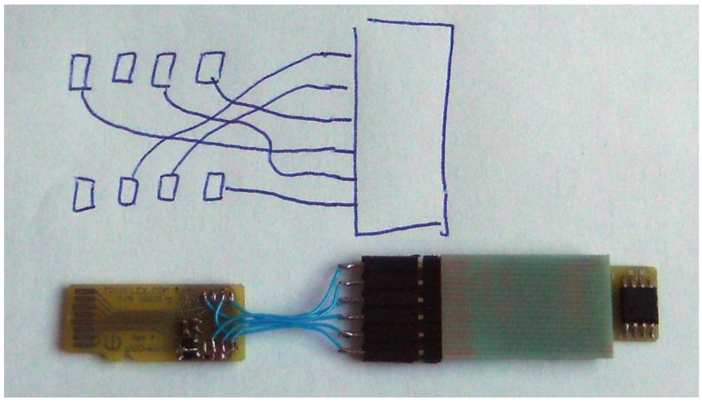


This is the UD Flash card adapter setup. Plugged into GameIO cart is Sipsik™ to Digilent PMOD header adapter, then Pmod Micro-SD adapter, and finally micro-UD SPI Flash card with AT45DB161D mounted.

But no, KEIL's hex2bin cannot convert the Silicon Blue Hex file, says conversion failed. Ha, the hex2bin from KEIL does not understand lower case ASCII ! And Silicon Blue tools use lower case.

Ok, I added support for ASCII binary (ascii file of 0x30 and 0x31) to the programming software, so now the SB tool generated files can be directly used.

Here is a photo how a micro-SD adapter and micro-UD card are used for development, it shows the AT45DB161D pin connection to Pmod SD Card adapter, so that the inserted micro-UD SPI Flash card will have 1:1 connection of the function pins.



And another success.. the new ice65L04 based FPGA Stamp PCB worked also at the production site where the engineers used the just developed SPI Flash programmer software, and UD Card and micro-SD adapter connected in similar fashion for SPI Flash break-out development.

SB Soft-Core Processor

Now it’s time to try out some soft-core processor in the SB FPGA, isn’t it? Well trying out AX8 an AVR compatible core based SoC, it failed at first attempts on the iceMAN65 and it still isn’t working properly. Trying lightweight 8080 core, this doesn’t seem to work at all.

Got some luck with Mproz3 what is a really minimalistic soft-core. Some very simple test program worked when ROM was implemented from logic, and failed completely when used initialized RAM. Maybe the problem was start up behavior after FPGA configuration, Mproz3 namely has no reset input to the core at all. Fixing up some sort of reset on configuration or more accurately just delaying the clock the core some clock cycles. Yes it helped the ROM from logic version is now starting up reliable on each reconfiguration (it was like 50:50 before applying the clock delay circuit).

And I was now fully able to verify the small test program was actually executed correctly, the all program was only setting output port to 0, then 1 and looping forever. I had the clock delay input connected to button on board so pressing that button stopped clock. And when tried out, the LED was either off or more bright, that is either static 0 or static 1. When the test program was running the LED was on with some medium brightness. So the loop was executed.

```
        br      main
        dc.w    iret
        br      int
iret:   dc.w    0

main:
loop:   add     #0000,#0001,io     ; LEDS=1
        add     #0000,#0000,io     ; LEDS=0
        br      loop

int:    br int

#0000:  dc.w    0
#0001:  dc.w    1

io=$7fff
```

Maybe the initialized RAM version would also work now? No. Still no luck, as soon as switch the code memory to block RAM it doesn't execute any code any more. Hum but I don't see the RAM init values in the generated bit file. So maybe the initialized RAM inferring doesn't work properly. That would explain that no instructions are executed. Yes, so it is the RAM init area in the bit file is all 0's!

Skipped here is lots of troubleshooting.. only the result matters, and yes Mproz3 now does successfully execute instructions from initialized block RAM. There is still some issues to be solved, but the possibility to execute code from RAM is proven now.

The remaining problem wasn't so easy to find and fix. Somehow the RAM writes makes the FPGA go nuts. Finally I connected RAM Write Clock enable to external 2 position slider switch. The results:

When write clock is initially disabled, can force FPGA reconfiguration any number of times, the soft core work ok all the times.

When enable the write clock after configuration, the soft-core keeps operating.

When enable the write clock and force reconfiguration, the soft-core stops working (50:50 probability, sometimes failure is after second reconfiguration attempt).

Now comes the real strange thing: if the soft-core once stopped working, then setting the write clock to disabled, and doing new reconfiguration will not bring the FPGA back to live!

And even more funny, if you remove power for less than one minute then the FPGA also will not properly start.

So once the write enable caused the soft-core to stop working, then it needed to let the FPGA to go into full power off to make it working again. Simple reconfiguration is not enough.

The one minute is because the Silicon Blue FPGA takes very little current and the core voltage is hold on the capacitors for quite many good seconds after power is removed from the iceMAN65 board.

Mproz3 is special in that sense that the original code has no reset wire, so the soft-core will run away immediately when the FPGA is configured and clock is present.

It looks that for the SB FPGA some startup module is required to keep the soft-core in reset, and the RAM in write disabled until the clock is stable, and ? Well it is still to be tested what is needed. My very first attempt that delayed the external clock a few cycles, and the RAM write clock a few cycles more did not fix the issue with failure at configuration.

It could be the L8080 also had similar problem, why it did not work at all. Yes, by disabling the writes to the inferred ROM/RAM the L8080 seems to work also.

Making a special ROC (Reset on Configuration) IP Core, connecting to the Mproz3 top level, using the reset signal to disable the write clocks to the RAM blocks. Now the startup is reliable!

But does the Mproz3 now really work properly? I had only tested a very primitive program that did set the output port to 0x0000 then 0x0001 and looping forever. That program did not write to RAM at all. For the full test I do modify the Mproz3 assembler, only 3 lines of code changed and already it does generate from template the initialized memory file for Silicon Blue. I create a very small program that increments a RAM location and does then write it to output port. And yes, now the LED is blinking, so the soft-core is fully working, including read/write to block RAM.

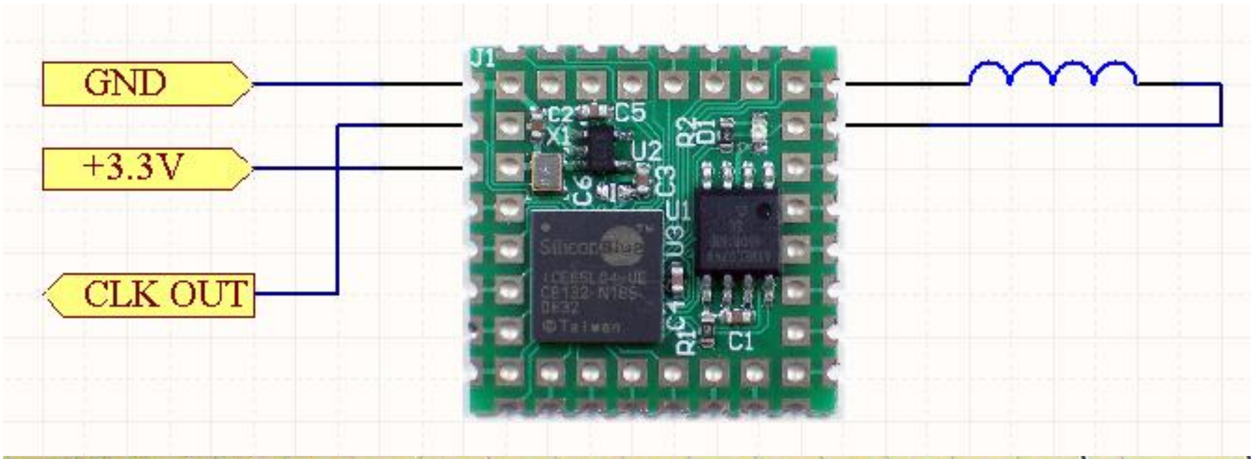
SB-Stamp LED Blinking

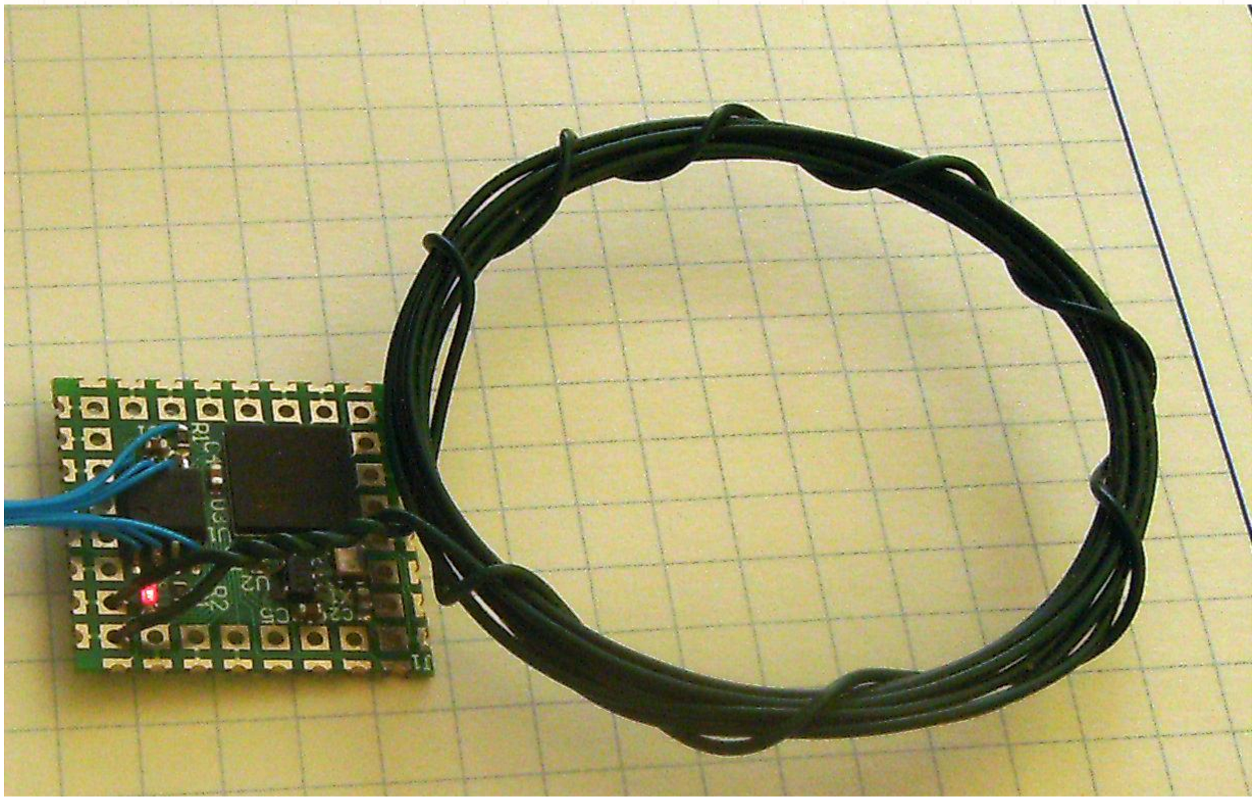
SB-STAMP PCB was tested with my LED ON design at the factory, but it was not actually tested if the on-board oscillator works, and if the 50Mhz clock is useable.

So I take my LED blinker from Stamp-Actel project and try it out with the SB-Stamp. The stamp-PCB is wired to empty micro-UD PCB so I can insert into the same programming adapter I use for programming micro-UD SPI flash cards.

Programming done, power cycle, and yes the LED is blinking! Well SB tools did report max clock to be 168mhz for the design so I assumed it would work with 50mhz. Sure the design was only one 24 bit counter to divide the 50Mhz to human eye blink rate.

Inductive Oscillator





MicroFPGA Part I

Prehistory

Most of the work on the concept was done at the end of 2006, I used maybe 3 weeks of computer time to generate MicroBlaze based MicroFPGA bit-streams for all Xilinx FPGA's supported by their tools at that time. Also made PicoBlaze based versions for the smaller devices where MicroBlaze does not fit at all. The very first versions used an array of GPIO ports, later I changed the I/O access to ASIO simple bit addressable I/O port.

PinAPI™ was developed long before MicroFPGA, at the time when MSDOS 3.22 was just released. While PinAPI was just a software API, then MicroFPGA ASIO peripheral could be described as hardened PinAPI, a special processor peripheral that can do indexed I/O access with single bit granularity.

SD Card Access

This was as simple as expected – taking all known working code, modifying the I/O drive functions, loading to S3A Starterkit, and voila readme.txt from micro-SD Card was read OK with DOSFS as FAT support library.

Spartan-3A DNA Reader

I have tested Ken Chapman's KCPSM based device DNA Reader of course, but the assembly is not that easy to read. Not that it much complicated, but well here is what I did write for MicroFPGA:

```
unsigned int dna_high;
unsigned int dna_low;

void DNA_Read() {
	int i;

	SetPin(DNA_CLK, 0);
	SetPin(DNA_SHIFT, 0);

	// Load DNA Value
	SetPin(DNA_READ, 1);
	PulsePin(DNA_CLK, 1);
	// Go shift mode
	SetPin(DNA_READ, 0);
	SetPin(DNA_SHIFT, 1);
	// 57 bits of DNA shift register to read out
	dna_high=0;
	for (i=0;i<(57-32);i++) {
		dna_high<<=1;
		dna_high |= GetPin(DNA_DOUT);
		PulsePin(DNA_CLK, 1);
	}
	for (i=0;i<(32);i++) {
		dna_low<<=1;
		dna_low |= GetPin(DNA_DOUT);
		PulsePin(DNA_CLK, 1);
	}
}

int main (void) {
	LCD_Init();
```

```
        DNA_Read();
        xil_printf("%08X%08X",dna_high,dna_low);
}
```

Written as it was, it immediately worked and displayed on the LCD same value as Xilinx DNA reader demo. The hardware to connect DNA to MicroFPGA (ASIO) is actually only routing as the DNA primitive pins are connected to virtual pins.

The above C code will work without change on any Spartan-3A based MicroFPGA. Nice? Of course the DNA value could be transmitted over UART or written to serial flash memory or used to detect the device/board and choose action to perform.

<to be continued>

The concept is maybe not so explained in this story, well a short explanation of MicroFPGA is that:

A MicroFPGA allows ANY FPGA to be used as if it would be a Microcontroller (not FPGA). Sure such use would be limited, but for some applications it is a great quick start.

Organize my Life™

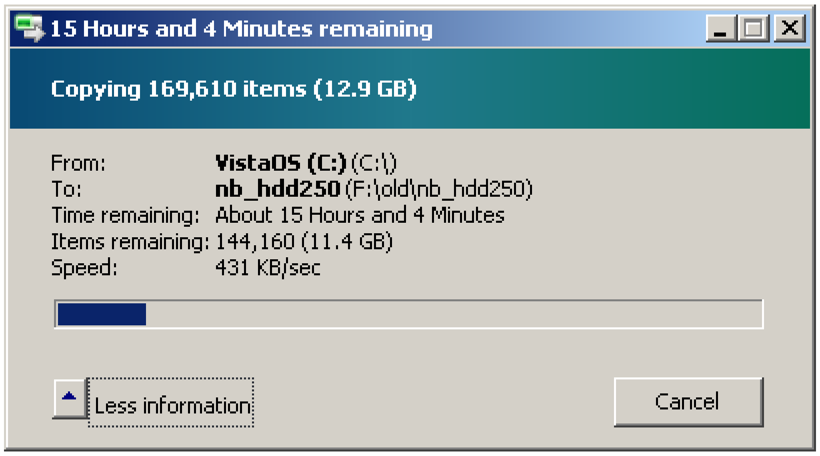
For about 8 months ago I did get a new notebook, as usual the best deal for reasonable money at that time. It was again ASUS despite the fact that my previous ASUS had only fatal errors and was later replaced to the newer model. Well the new ASUS did die too, blue screen of death during power down, later CRCDISK.SYS error during boot, and the auto-power off during power up. I was very frightened as I had no backups at all from the projects on the internal hard disk. The notebook luckily started next morning but I decided that I really must make full backup of everything. As I also have old projects on maybe 6 HDD's taken out from computers when the C drive no longer so I wanted to finally start the project "Organize my Life".

Step 1: Need new hard disk? Well I had some backup HDD's from 120G to 500G size. But the 500G HDD was SATA and I had no SATA enclosure. I could have cleaned one of the existing HDD's to use as the new backup unit, but I decided to keep the existing HDD's as emergency backups of old projects so I did get a brand new 1TB drive.

Step 2: Backup everything! Started file copy to the external 1TB disk from download folder at desk top PC.. 12GB copied, file copy error! I look the HDD root directory and it contains only trash names. No files. So it not that there is one wrong entry, no - the root directory is read from wrong place. Naturally the 12G of files are not accessible. Luckily I did file copy not move. Ok, I decide not to try format the HDD, just back to the store and get new one.

Copying some download files to 1TB, data transfer speed changes from 37KB/s to 11MB/s? How come can it be the data rate is so different? I had no other applications running just the file copy! But the low data rate explains why some tools run so slow on the notebook.

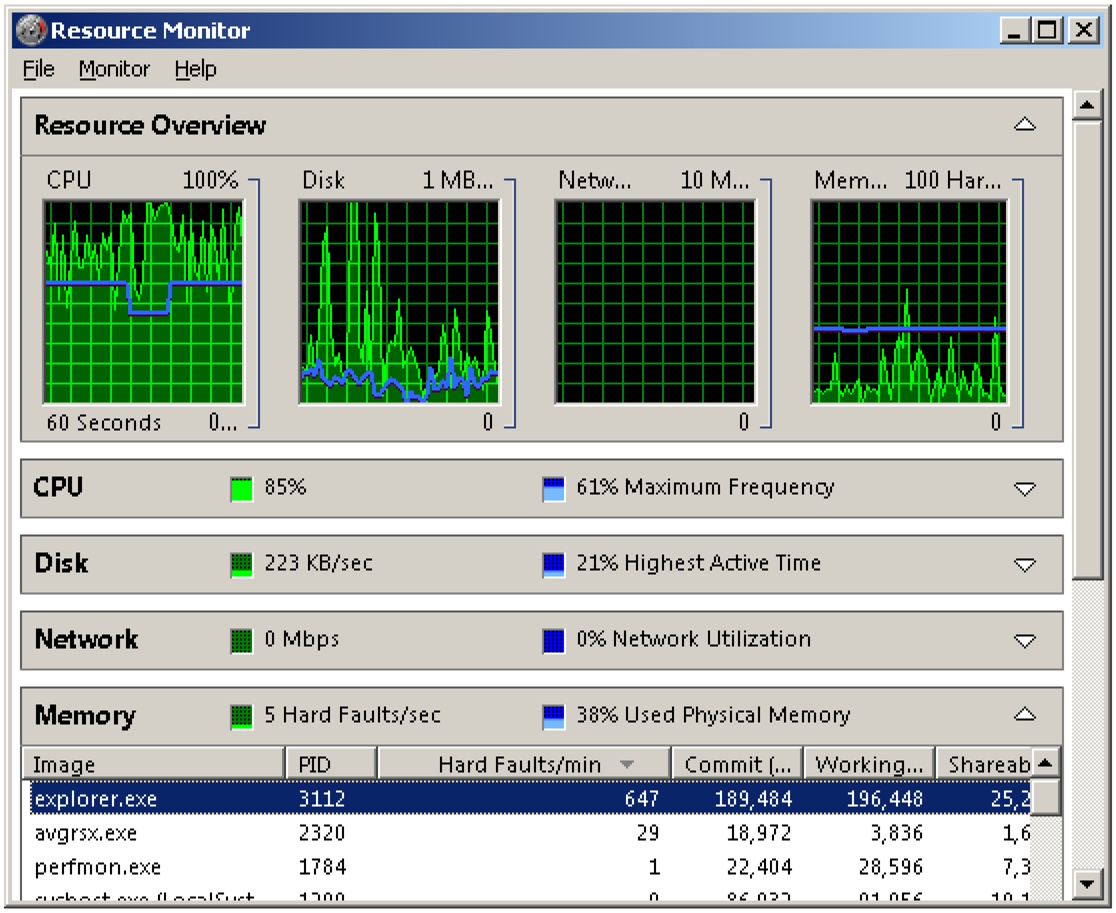
Starting file copy on project files on the notebook:


15 Hours and 4 Minutes remaining

Copying 169,610 items (12.9 GB)

From: VistaOS (C:) (C:\)
To: nb_hdd250 (F:\old\nb_hdd250)
Time remaining: About 15 Hours and 4 Minutes
Items remaining: 144,160 (11.4 GB)
Speed: 431 KB/sec

That is ouch: 169.610 files to copy! The time remaining changed from 3 to 22 hours. Ok I am on the country house and supposed to do some work in the forest, so I don't care about the time.



CPU 🟩 85% 🟦 61% Maximum Frequency

Disk 🟩 223 KB/sec 🟦 21% Highest Active Time

Network 🟩 0 Mbps 🟦 0% Network Utilization

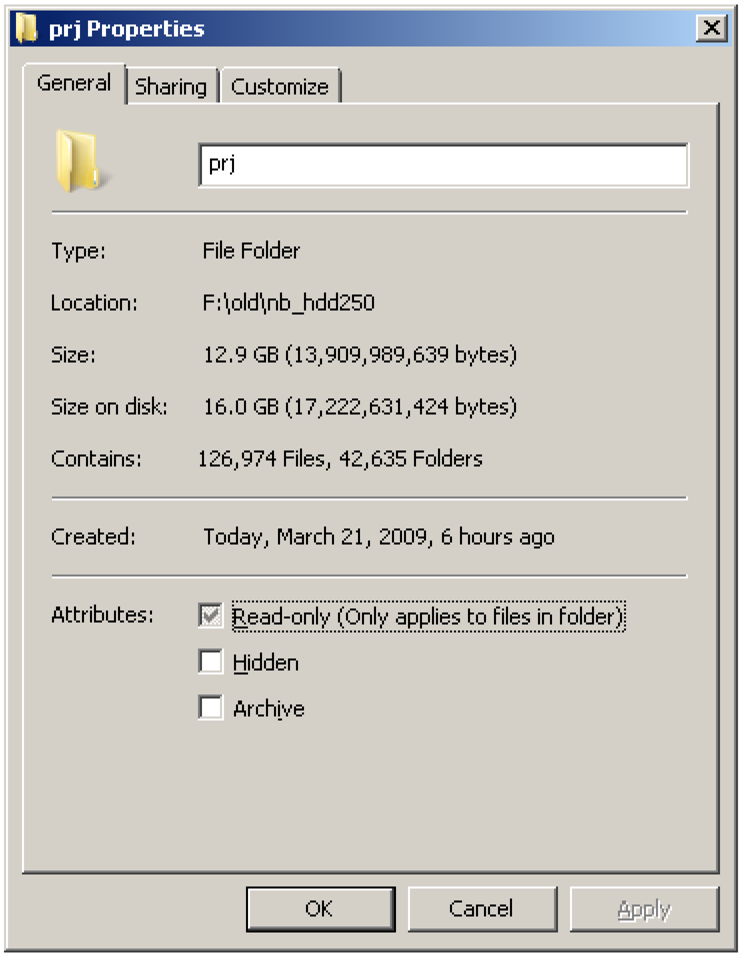
Memory 🟩 5 Hard Faults/sec 🟦 38% Used Physical Memory

Image	PID	Hard Faults/min	Commit (...	Working...	Shareab
explorer.exe	3112	647	189,484	196,448	25,2
avgrsx.exe	2320	29	18,972	3,836	1,6
perfmon.exe	1784	1	22,404	28,596	7,3

Maybe this explains why the notebook is working so slow – it is still something I do not fully understand, in the old times it was all very simple the memory was either OK or BAD. But now, there are "Hard Faults" on memory reported? If there are hard faults, so there are also less serious faults? So is the computer memory bad or good?

The AVG scanner is still running, I failed to disable it ☹ looked all placed even tried the help, but there is no place to disable the AVG protection.

Unfortunately I cannot tell how long the copy actually did take, I started it around 8 and at 14 when I returned from forest it was already completed.



prj

Type: File Folder

Location: F:\old\nb_hdd250

Size: 12.9 GB (13,909,989,639 bytes)

Size on disk: 16.0 GB (17,222,631,424 bytes)

Contains: 126,974 Files, 42,635 Folders

Created: Today, March 21, 2009, 6 hours ago

Attributes: [✓ Read-only (Only applies to files in folder)] Hidden Archive

I wanted to be sure that the copy actually did transfer the files but it really did. Well it only 126900 files not 169000 but still a lot. And it was good thing to do dumb copy as trying to sort out 42000 folders? Well that work still has to be done, but I can make some cleanup scripts that would hopefully deletes most of the files and directories.

<to be continued…>

Shorts

ATtiny13A

Has 3x3 mm QFN package, that nice unfortunately only 6 I/O pins, 2 pins of the 10 are DNC (do not connect. Still it is nice package and it is cheaper than SiLabs in same sized package. Well the C8051F30x have 8 I/O's not 6, but it costs more.

ATtiny24A

Well this is nice, finally! 3x3 mm QFN package with more than 8 useable I/O. Has weird 0.45mm pitch, and the corner pads have different shape. But it is nice and small. Digikey doesn't yet carry this device, so need checkout the availability.

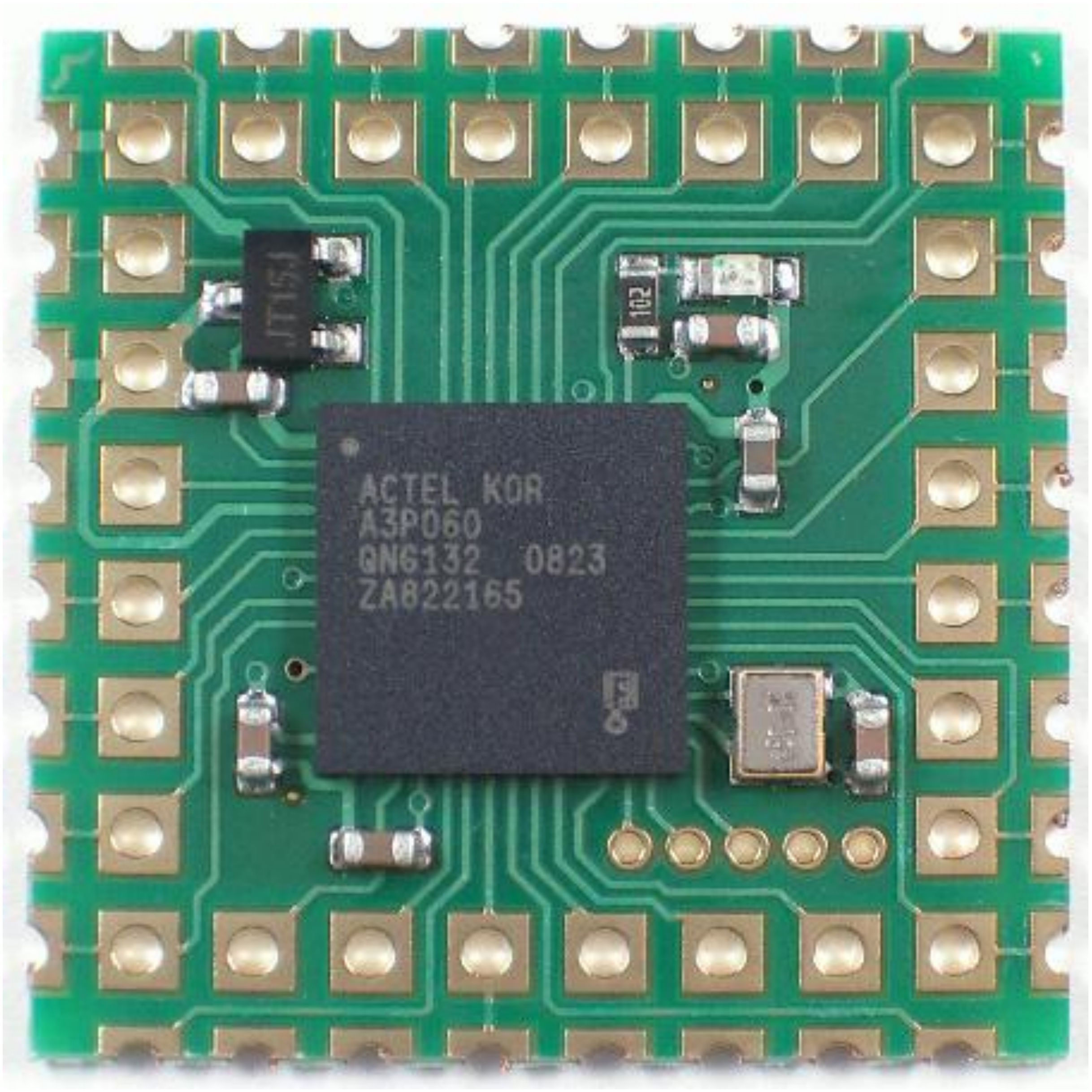
PSoC with power drivers

A MCU + 4 channels 1A/32V FET's. More fun for color led fans.

SiLabs Radio (continued)

And here it is, test setup to verify the Si4721 Transmit function. And yes, the radio is fresh new from local store, just obtained for test purposes. The test was really simple, a small loop that tuned the FM transmitter to 87.5Mhz, and then did reset chip, and loop. The TXO (transmit out) pin of the IC is not yet connected to any antenna or circuit, just a 6 mm long PCB track on the adapter board. That was sufficient to hear the noise (or difference in the noise) at around the transmit frequency during the transmitter tune function.

Actel Stamp PCB Rev B



This is production version, definitely. And yes they are ready at the fab, and tested as well.

References

- http://www.trioflex.com

Instead of adding the URL links at the end of each issue, I will be adding them to the Trioflex online link collection, so they can be updated more frequently.