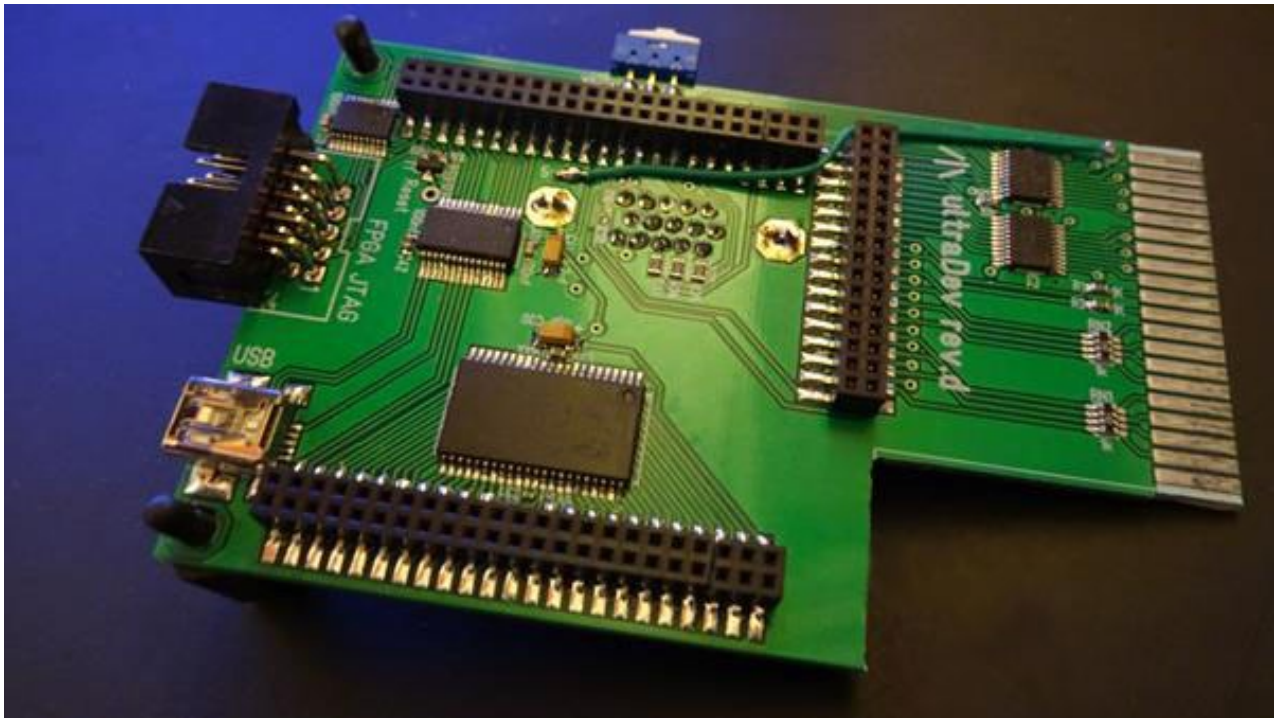
ultraDev

The Atari Development Cartridge



User Manual

Version 0.5

Table of Contents

1. Welcome

ultraDev is a FPGA based development cartridge for Atari computers.

How does it works?
After a reset the Atari stays in an upload screen and waits for an upload from the host computer. You can upload and start a Prg-file from the host computer command line via USB.

With a small cable from inside the Atari the cartridge is able to reset the Atari.

Which means you just run the command line tool again and the cartridge resets the Atari, uploads and starts the prg.

If you do not want to open up your Atari and solder in the cable you can also use a small resetter code which is included in the 68k library.

How about debugging?
If you want to debug you can tell the command line tool to start bugaboo before and directly load the Prg-file.
With the command line tool it is possible execute bugaboo commands after bugaboo loaded the Prg-file. Very useful to set breakpoints or just execute automatically after loading.

Debug screen
You can connect a VGA monitor to the cartridge to output debug text values on a separate screen (40x32). You can use up to 8 colours. It is even possible to upload a font to create custom chars. A bitmap mode with 128x128 resolution is also supported. Have a look to the 68k Library and examples how to use it.

1.1. Electrostatic-sensitive device warning

ultraDev is an electrostatic-sensitive device. What does this mean? (Text from Wikipedia)

"An electrostatic-sensitive device (often abbreviated ESD) is any component (primarily electrical) which can be damaged by common static charges which build up on people, tools, and other non-conductors or semiconductors. ESD commonly also stands for electrostatic discharge.

As electronic parts like computer central processing units (CPUs) become packed more and more densely with transistors the transistors shrink and become more and more vulnerable to ESD."

ultraDev has no case if you touch the PCB you can damage the components. So it's a good idea to touch a grounded device before you touch the PCB.

But it's not that dramatic how it sounds. Often I forget to touch a grounded device before and till now never something happened. **But you have been warned :)**

1.2. Open source

The project is open source feel free to download the source at:
HTTP://ultraDev.ultrafex.de

Currently the sources aren't on github or bitbucket. Maybe I'll add them in future if somebody does changes.

2. Setup & Install

2.1. Requirements

- As host computer only Windows (XP-W10) is supported. But the command line- and FPGA updater utility are standard C and should be easily compiled on other platforms
- Currently the cartridge needs a 4mb Atari and it is fixed to max. 4mb in memory usage at the moment but this will change of course in future.
- The cartridge should work on:
  - ST (not tested yet)
  - STE
  - Mega STE (not tested yet)
  - Falcon (not tested yet)
  - TT (not tested yet)

2.2. Release folder structure and files

- 68kFirmware

  I decided the Atari 68k sources for the firmware is in the release. Maybe better if you have problems with starting something you can have a look to the sources.

  Also the preassembled firmware.img you can find here. Normally you don't need this. The firmware.img is included in the FPGA firmware.
- 68kLib

  Library to handle the cartridge like printing to screen and stuff. There is a more detailed description a bit below (68k Library)
- 68kLibSamples

  Examples how to use the 68klib there is a more detailed description a bit below (68k Library Examples)
- Cmdline

  Command line tool to upload a prg to the Atari check (Uploading and starting a Prg-File)
- Doc

Usermanual.

- FPGAUpdate

  A small command line tool to update the FPGA firmware. How to do this see (Updating the FPGA firmware)

2.3. Driver install

Go to the Driver directory and execute CDM21228_Setup.exe. Follow the setup instructions.

After Install the USB device „ultraDev" should show up in the "Devices and Printers" panel in Windows.

You can connect ultraDev to one of your USB ports without connecting it to the Atari. Just to see if it's showing up there. Of course upload do not work then...

2.4. Hardware install

Installing the cartridge to your Atari is pretty easy. Just plug it in ;)

Do not wonder if the cartridge screen does not show up directly after booting. The FPGA needs some time to boot so it misses maybe check from the TOS if there is a cartridge. If the Flash Led lights up (on FPGA board) you can press reset and if the cartridge is switched on the cartridge screen will appear.

Installing the Atari reset cable is currently not documented. If you wish the Atari resets each time before a new Prg-file is uploaded have a look to the 68k Library Samples. There is a small inserter code which resets the Atari if needed.

2.5. Connecting a VGA monitor to the Cartridge

Under the cartridge you will find the connector for the VGA screen. ultraDev creates a 1280x1024 @60hz signal.

Have a look to the 68Lib and 68LibExamples how to write stuff to the screen.

2.6. Leds, buttons and switches

Following Leds are on the FPGA board:

- Pwr Led

  Lights up if the FPGA got it's power

- Flash Led

  After power up the Led is off and will light up when the FPGA loaded the Firmware from the flash. If it does not light up you have a problem. This will mean you need a XiLink programmer to fix that.

- Led1-4

  Led 4 blinks slowly if the cartridge is in idle and waiting for data from the host computer. If no USB cable is connected it blinks a lot slower

  Led 3 blinks slowly if the cartridge waits for data during a command. If it is

blinking and the commandline tool hangs there went something wrong. To fix that you can press the config button. This will restart the FPGA.

Led 1-4 blinking fast the cartridge waits for the Atari to be resetted.
If you have a reset cable installed the cartridge recognises if the Atari preformed a reset and continues

If you don't have a reset cable installed you need to press reset by yourself. By the way there is a resetter inserter code for your VBL routine which does the reset for you have a look to the 68kLib and 68kLibExamples how to use it.

All buttons are on the FPGA board:

- Config will restart the FPGA
- Reset is not really used currently

Switch on the cartridge:

- This will switch off or on the cartridge. If you do not want the cartridge boots into the ultraDev screen you can switch it off.
  Have a look to the PCB which position is on and off.

  Not Included yet ;)

2.7. Atari memory used by ultraDev

ultraDev uses some memory from the Atari memory. Currently it's fixed to >=4mb this will change very soon !

3. Uploading and starting a Prg-File

To upload a Prg-File you need to use the command line tool ultraDev.exe which is included in the release in the directory "Cmdline".

3.1. Command line options

- prg (path) Start Prg-file.
- dbg (path) Bugaboo path. If given bugaboo is started before.
- cmd (Bugaboo commands) Bugaboo commands which are executed after loading. -cmd g starts the prg directly after loading. The commands are separated by :
- uc (path) Upload cartridge (68k firmware). Normally you don't need to upload a 68k firmware. The current firmware is included in the FPGA. This is just for testing reasons or if you want to change something to the 68k firmware.

3.2. Changes to Bugaboo

The reset saveness has been removed from Bugaboo. This is needed otherwise the reupload of a new Prg-File don't work.

The le (load executeable) was changed only to load from the cartridge. So do not

use ! The le command is always added to command list after loading bugaboo to load the Prg-File.

4. Updating the FPGA firmware

4.1. Updating with the ultraDevFPGAUpdate

For updating the FPGA with the ultraDevFPGAUpdate tool you need to connect the small cable which was included from the FPGA-board to the ultraDev cartridge PCB.

Then double click ultraDevFPGAUpdate.exe. Then wait. This task tooks about a minute. **DO NOT SWITCH OFF THE ATARI DURING update !**

If the update is done switch off your Atari and switch on.

4.2. Updating with a Xilinx programmer

If you own an Xilinx programmer you can do the FPGA update of course with the programmer. This a lot faster. You can find the mcs-file in the FPGAUpdate folder.

In this case you need to use the 10 pin header cable from your programmer and connect it to the FPGA board. **NOT THE CARTRIDGE connector !** If you do by accident no problem nothing will get broken but it just do not work then.

Steps to Update the firmware:

- Start Impact
- File->Initialise Chain
- No bit-file needed for the FPGA we want to programm the flash
- No Flash attached in the following message box
- Assign the top.mcs file from the FPGAUpdate folder
- In the Device properties dialog for device 2 check "Load FPGA" otherwise the FPGA does not boot.
- Right click on the xcf04s icon and program

It's important that you programm the device 2 xcf04s that's the config rom!. If you programm the FPGA only the data is lost after power off.

But I guess you know ;)

5. 68k Library

ultraDev comes with a small library which helps you to use the cartridge features without coding the new hardware directly. The 68k library is located in the release 68kLib.

There are a few examples how to use the 68k Library check 68k Library Examples.

Have a look to the sources which routines are available I tried to write some more

informations how to use and stuff.

Files included:

- UDAll.s

  Includes all ultraDev includes

- UDBitmap.s

  Service routines for the bitmap mode. The bitmap mode displays a 128x128 image. The resolution is halved to fill the screen.

  How does it work? More or less is the bitmap mode very like on vc20. You just print the chars from 0 - 255 on screen and the bitmap itself is modified by changing the font.

- UDCartridge.s

  Service routines how to detect the cartridge clean and stuff like version numbers etc.

- UDDefines.s

  Includes defines for the new hardware registers

- UDFont.s

  Service routines for uploading a font.

  Keep in mind if you upload a new font into the FPGA this font will remain in the FPGA until you restart it. Means if you fucked up the font it will stay fucked up. But there is a routine which reinitialises the font if you want

- UDResetter.s

  Small inserter code to your vbl routine. If you upload a new version of your Prg-File via the command line tool it resets the Atari to load it. Only needed if you do not have a reset cable installed.

- UDTerminal.s

  Can be used to create a terminal like screen area which means you print text and at the end of the screen the text scrolls up. By setting the start line for example to 16 you can have a scroll area at the bottom of the screen upper area is not touched means you can show stuff here...

  use UDTermPrint to print stuff into the terminal.

- UDVideo.s

  Service routines for the debug screen video "chip"

6. 68k Library examples

Some examples how to use the 68k Library. The samples are located in the 68LibSamples directory.

If you do not want to assemble the examples for your own there are preassembled binaries availble in the bin directory.

You should also have a look to the 68lib sources and check which routines are available.

Examples:

- BGColorRaster.s

  Shows how to set the background colour and how to do some rasters on the background.

- BitmapMode.s

  Shows a 128x128 bitmap image

- CustomChars.s

  Shows a screen with custom chars and a permanent upload means scrolling char. This example also syncs to the VGA VBL.

- PrintText.s

  Simple demo how to print text in different colors to screen.

- Terminal.s
  Shows the usage of a partical terminal screen.

7. Hardware registers

7.1. Reading from and writing to the cartridge

Reading from the cartridge from rom or hardware registers is straight forward just use a move instruction to get the data.

Unfortunately Atari did not add a write signal to the cartridge port. To realise a write to the cartridge you have to use a trick.

So the cartridge are is split into two areas. FAXXXX and FBXXXX. FBXXXX is used for writing.

7.2. Reading

Reading can be done from FAXXXX. Some of the "read" registers act like a switch or a trigger (will be explained a bit below).

7.3. Writing

Writing is a bit confusing I hope I can explain ;)

For writing to the cartridge the area FBXXXX is used. Writing works with a read instruction. The address itself is used for the write value.

XXXX are the bits which are written to the before selected memory destination. So the write value is included in the reading address.

This also means you can not write to a specific address in the destination memory you only can write and the internal address is incremented with each write.

If you want to write to a different memory position you have to set it before.

If you look to the code for writing to the cartridge it does not make sense at the first glance.

Maybe a small example for writing an white "a" to the debug screen:

```
move  UDRegWriteToScreen,d0
move  #'a'+$700,d0
sub.b #$20,d0
lea   $fb0000,a0
move  (a0,d0),d0
```

Looks confusing right? So let's have a look to each line.

```
move  UDRegWriteToScreen,d0
```

Reads from the address $fa4100 the result in d0 does not care. But it does not preform a read inside the FPGA it just switches the write selector to the debug screen memory.

This is what I was talked about earlier. If you read from some memory locations in the FAXXXX area this triggers or switches stuff in the FPGA. In this case it switches the write destination to the debug screen memory.

```
move  #('a'+$700)*2,d0
```

Ascii code for an "a" and +$700 is the color for the char. *2 is because the adress lines in the cartridge has no a0 signal so odd access aren't possible.

sub.b #$20,d0

The font starts with $20 (space) so you need to sub $20 from the ascii code.

```
lea   $fb0000,a0
```

Get the write area address to a0.

```
move  (a0,d0),d0
```

And read from that address. D0 is in this case the value you want to write. If you read from that address the FPGA does following:

- extracts the XXXX bits from the FBXXXX address
- check what is the write destination
- and writes the XXXX bits to the write destination
- increments the internal write address destination address by 1 (for some write destinations you can also set the increment like for font upload which makes sense in the bitmap mode)

To set the internal address the FPGA writes to is working straight forward. For example setting the write address to the debug screen were to write the char on screen is like:

- Reading from UDRegWriteToScreenPos to select the internal address of the

debug screen where to write.

- preform a "write" by reading from the $FBXXXX area to set the address

But ! You do not need to fight with this the 68k library has service routines to handle this.

7.4. Memory map

R/W/S

R = Read

W = Pseudo write via read (explained in 7.3 Writing)

S = Switch memory write destination to...

Address	R/W/S	Description
FA0000-FA4000	R	Rom area
FA4000	R	68k State. Only used to communicate between FPGA and 68k. This is the state what the 68k should for example do during a upload of a program.
FA4002	R	Ready Flag, Only used to communicate between FPGA and 68k. Tells the 68k is for example done with coping the before uploaded block.
FA4004	R	Upload count during upload a program only used for debugging.
FA4006	R	State machine value for the FPGA only used for debugging.
FA4008	R	Bytes received during a upload inside of the block only used for debugging.
FA400A	R	State machine return state value for the FPGA only used for debugging.
FA400C	R	VGA VBL flag. 0 = currently in vbl 1 not. This is the direct VGA signal which goes to the monitor. Keep in mind this signal will stay at 0 during a vbl for some lines on the VGA screen.
FA400E	R	VGA counter x. Current beam x position.
FA4010	R	VGA counter Y. Current beam y position.
FA4012	R	Cartridge detect 1 $dead
FA4014	R	Cartridge detect 2 $beef
FA4016	R	FPGA Version
FA4100	S	Switch write destination to debug screen memory
FA4102	S	Switch write destination to the internal debug screen

		address. Used to set where to write on the debug screen.
FA4104	S	Switch write destination to font memory. Do not forget to set the internal font address before uploading!
FA4106	S	Switch write destination to the internal font address.
FA4108	S	Switch write destination to internal auto increment register for font upload. If you upload a font each time to write a byte to the font memory the font address is incremented by 1. In the bitmap mode it's better to set it to 8 to write a horizontal line. If you don't do you need to set the font address each time because the bitmap mode is char based and stuff.
FA4200	S	Switch write destination to video register 1. 0: 0 = 40x32 resolution 1 = 16*16 double pixel mode 1: unused (maybe later multi colour mode) 2-4: background colour
FA8000-FAC000		Upload memory for program files