User Guide rev.1 arduFPGA-game-console rev1.0

Description

Introduction

Thank you very much for purchasing our arduFPGA-game-console board.

The schematics, all designs and firmware are open source and licensed under GPLv2.

This board is designed for young developers to learn to develop their own microcontroller designs on a FPGA and for game and portable devices enthusiasts that want to develop games and applications for mobile devices combining the easiest way to develop all kind of mobile applications using arduino IDE and other IDE's and the versatility of an FPGA that each pin can have any desired function and can be implemented almost all interface types.

Resource

There I developed several core versions for each need and are developed such a way to be very human readable:

1. Mega/Xmega
   - MEGA_REDUCED
   - MEGA_MINIMAL
   - MEGA_CLASSIC_8K
   - MEGA_CLASSIC_128K
   - MEGA_ENHANCED_8K
   - MEGA_ENHANCED_128K
   - MEGA_ENHANCED_4M
   - MEGA_XMEGA_1
   - MEGA_XMEGA_2
2. RISCv 32I
   - Single stage single BUS
   - Single stage dual BUS
   - Five stage pipeline single BUS
   - Five stage pipeline dual BUS

Several IO's:

1. PIO
2. SPI
3. UART
4. TWI
5. RTC

These IO's can be used on both architectures.

All schematics, designs and application source code can be found on https://devboard.tech/git

Specifications and features of LATTICE iCE40UP5K

- **5280 LUT's with four inputs.**
- **30 EBR memory blocks with a total of 12KB.**
- **4 SPRAM (single port ram) with a total of 128KB.**
- **One time programmable NVCM.**
- **One PLL.**
- **8 DSP with 16 bit MUL and 32bit ADD.**
- **Two hardened TWI interfaces.**
- **Two hardened SPI interfaces.**
- **One high frequency 48Mhz internal oscillator.**
- **One low frequency 10Khz internal oscillator.**
- **PWM IP block.**
- **All inputs compatible with 1.8, 2.5 and 3.3V.**

Specifications and features of the board

- **Three LDO power supplies 1.2V, 2.5V and 3.3V used as general power for the board.**
- **Onboard uSD socket for on the fly design upgrades, firmware upgrades, application loads and user data.**
- **One onboard 16Mb FLASH memory with endurance greater than 100K erase/write cycles for design storage and GUI boot loader storage.**
- **One 2Mb onboard FLASH memory with an endurance greater than 100K erase/write cycles for user application storage, if user know that will change the application very often, can replace this FLASH IC with a SPI SRAM.**
- **Six PUSH buttons:**
  - **RESET – This button is used to reset the FPGA IC and reload the entire design from the onboard FLASH memory.**
  - **INTERRUPT – This button is used in combination with the default preprogrammed GUI boot loader, the default design is a reduced ATmega32u4 soft core that is able to run arduboy native games without any changes, because the board offer the possibility to change games and load them from onboard uSD the user can short press the INT button and the GUI boot loader will save the emulated EEPROM content to uSD and display the uSD content for user to load another application/game, this button is connected to PORTA, the PORTA does not exist in the original ATmega32u4 device.**
  - **UP, DN – These buttons are used to navigate thru uSD content from GUI explorer, but can be used even by the user application, this button is connected to PORTA, the**

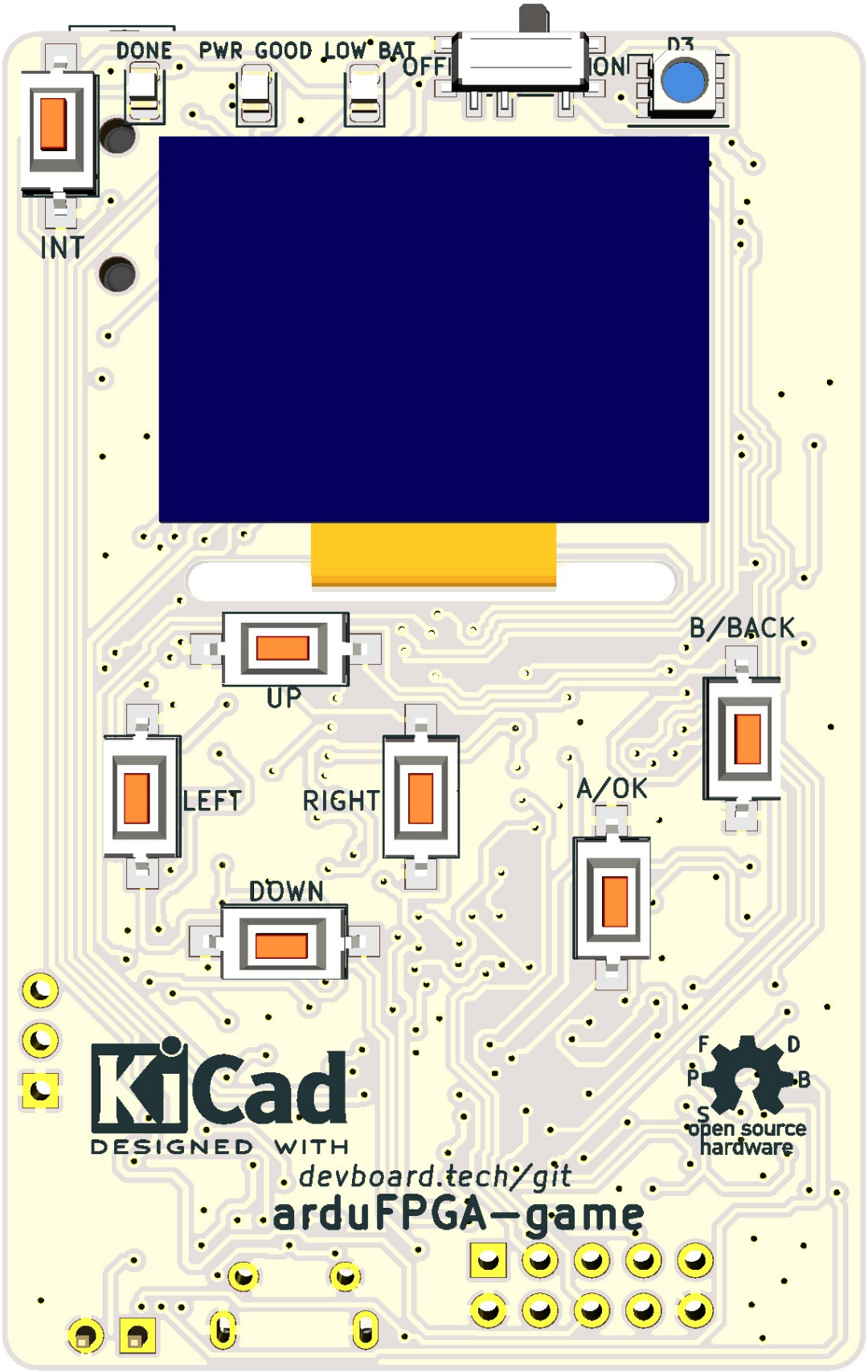
PORTA does not exist in the original ATmega32u4 device, alternatively if the INT button is held press for more than 500mS the keyboard is disconnected from user application and using UP, DN buttons can be increased/decreased the audio volume in four steps.

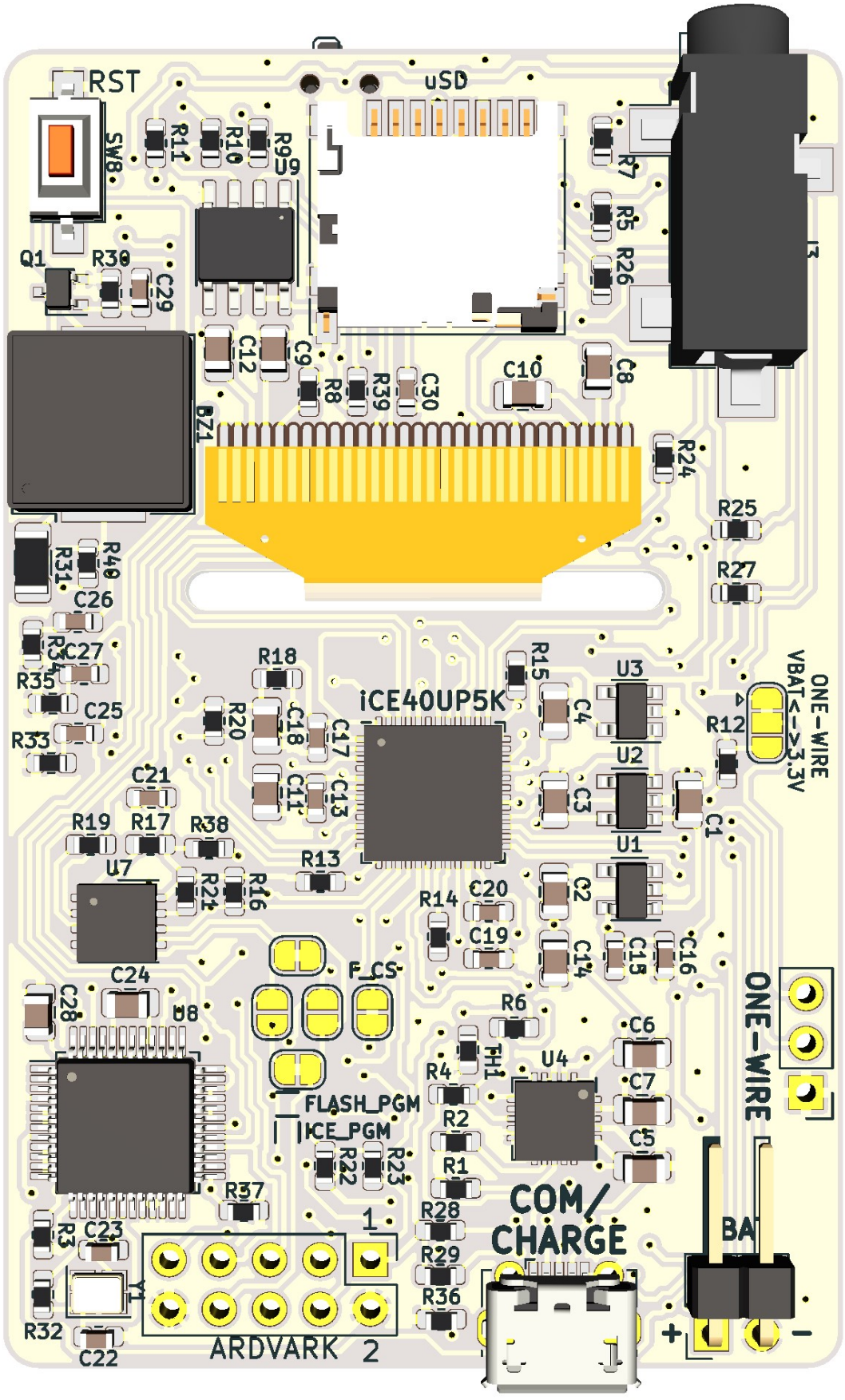
- **BACK – Is used by the GUI boot-loader to go up a directory and alternatively if the INT button is held press more than 500mS the keyboard is disconnected from the user application and will change the RGB LED color in the order: B,G,R.**
- **OK – This button is used to open selected directory or load selected application, if a file with the same name as the application name and with .eep extension is found the GUI boot-loader will load the .eep file content to the internal emulated EEPROM memory and alternatively if the INT button is held press more than 500mS the keyboard is disconnected from the user application and will power/shut down all three colors of the RGB LED acting as a flashlight, this button is connected to PORTA, the PORTA does not exist in the original ATmega32u4 device.**

- **RGB LED:**
  - **The iCE40UP5K device has three dedicated pins with 24mA current capable, in combination with internal hardened PWM modules the user can independently set the luminosity of the RGB LED.**
- **Micro USB connector:**
  - **The micro USB connector is used to charge the battery thru a dedicated onboard charger and has three wires from FPGA to D+, D- and ID pins of the micro USB connector that can have any desired functions.**
- **Audio connector:**

- **This connector is directly wired to onboard VS1053b dedicated audio decoder and three wires of the FPGA that can have any functions when the VS1053b circuitry is held in RESET mode.**

- **ONE-WIRE connector:**
  - **A general purpose three pin connector that can take the power directly from battery or from internal 3.3V power supply and have one FPGA pin connected to it and can have any desired function.**


DONE
PWR GOOD
LOW BAT
OFF
ON
D3
INT
UP
LEFT
RIGHT
DOWN
A/OK
B/BACK
KiCad
DESIGNED WITH
devboard.tech/git
arduFPGA-game
open source hardware



Preloaded design and firmware

This board comes with preinstalled design and firmware out of the box.

The design is a demo reduced version of ATmega32u4 that is capable to play most of unmodified arduboy games.

The firmware is split in two boot loaders:

1. **First stage boot loader.**
   - **First stage boot loader is a boot loader written into the design memory and can be changed only from design.**
   - **This boot loader has only the simple function to load the GUI boot loader or the user application from onboard FLASH memory to internal program memory at power up or when GUI explorer jump to it.**
   - **This boot loader is located at address 0xFC00 and as RAM memory use a RAM memory on top of ATmega32u4 RAM memory located at 0xB00 (the end of ATmega32u4 RAM) and is 512Bytes long, this allow this boot loader to run independently of the user application and in parallel like a service.**
   - **This boot loader accomplish another function, every 1mS a NMI interrupt (this interrupt is an interrupt that does not exist in the original ATmega32u4 device and the vector address is hard written in the design) this interrupt monitor the push of the INT, if the button is press between 100mS and 500mS (short push) will interrupt the running application and load the GUI boot-loader, if press longer will disconnect the keyboard from the user application and execute the alternative push button functions.**

2. Second stage boot-loader.

- **This is a GUI boot loader and executes all the tasks to check on the uSD for design and firmware updates, load applications from uSD, save/restore EEPROM data content change the design and GUI boot loader on the flash if an application made for another design is selected for load.**
- **This boot loader has integrated a FAT 12/16/32 FS to manage uSD memory card content.**

Power UP and application change

With the demo design and firmware the board runs thru several stages:

1. **uSD FS directory structure**
    - **The uSD need to fallow a precise directory tree from root directory, in the root directory need to be platform directory's, this directory's are placed in the root in case of user using multiple designs on the same board, on each platform directory need to be *des.bin* and *app.bin*, design and GUI boot loader and user application images with .app extensions, the *des.bin* and *app.bin* files are loaded to the onboard FLASH memory each time a user application is load from another platform directory.**
2. **Power UP**
    - **At power up the FPGA device will load the design from onboard FLASH memory and start to run it.**
    - **The design will run the first stage boot loader that load the GUI boot loader or the user application from onboard FLASH to internal program memory (The load of GUI boot loader or user application is set up at compilation time of first stage bootloader).**
    - **When GUI boot loader starts will initialize the uSD.**
    - **Before entering explorer mode will check after *current.txt* file, in this file is written the path, the name and extension of last opened application, if this file does not exist, the explorer will begin the navigation from the root of uSD memory card, if *current.txt* exist, the navigation will begin from the path written in *current.txt* file.**

3. Selecting the application

- **When user selects one application to be load, the GUI explorer will check if the content of *des.bin* and *app.bin* in current directory are the same as the design and GUI boot loader written on the onboard FLASH memory, if is different will load the content of this two files in to the design and GUI boot loader sections in to the onboard FLASH memory, after this the GUI boot loader will write the path and the selected application name to the *current.txt* file in the root directory of the uSD memory card.**
- **After *des.bin* and *app.bin* content has been written in to the onboard FLASH memory, the user need to RESET the board (this procedure is needed only if the design is changed), after reset the GUI boot loader will look after *current.txt* file to see where is located the user application to load it.**
- **After GUI boot loader loads the user application in to the application section of the onboard FLASH memory, the GUI boot loader will look for a file with the same name as the application but with *.eep* extension, if the file is found will load the content of that file in to the internal emulated EEPROM.**
- **After application and EEPROM content has been load, the GUI boot loader jump to the first stage boot loader, the first stage boot loader will load the user application in to internal program memory to be executed.**

4. Interrupting the user application

- **If the user push the short press the INT the first stage boot-loader interrupt service routine will load the GUI boot-loader in to the internal program memory and run it, will let the emulated EEPROM memory intact.**

- **When GUI boot loader start will look for *current.txt* file to see what was the location of the application that was interrupted, if *current.txt* is not found, will take no action and enter in normal navigation mode, but if the file is found will save the emulated EEPROM memory content to the uSD in the same directory as the application with the same name as the application but with *.eep* extension, if a file with the same name and extension already exists, it will overwrite it.**

History

Document history

- **2020.05.03: Create the document.**