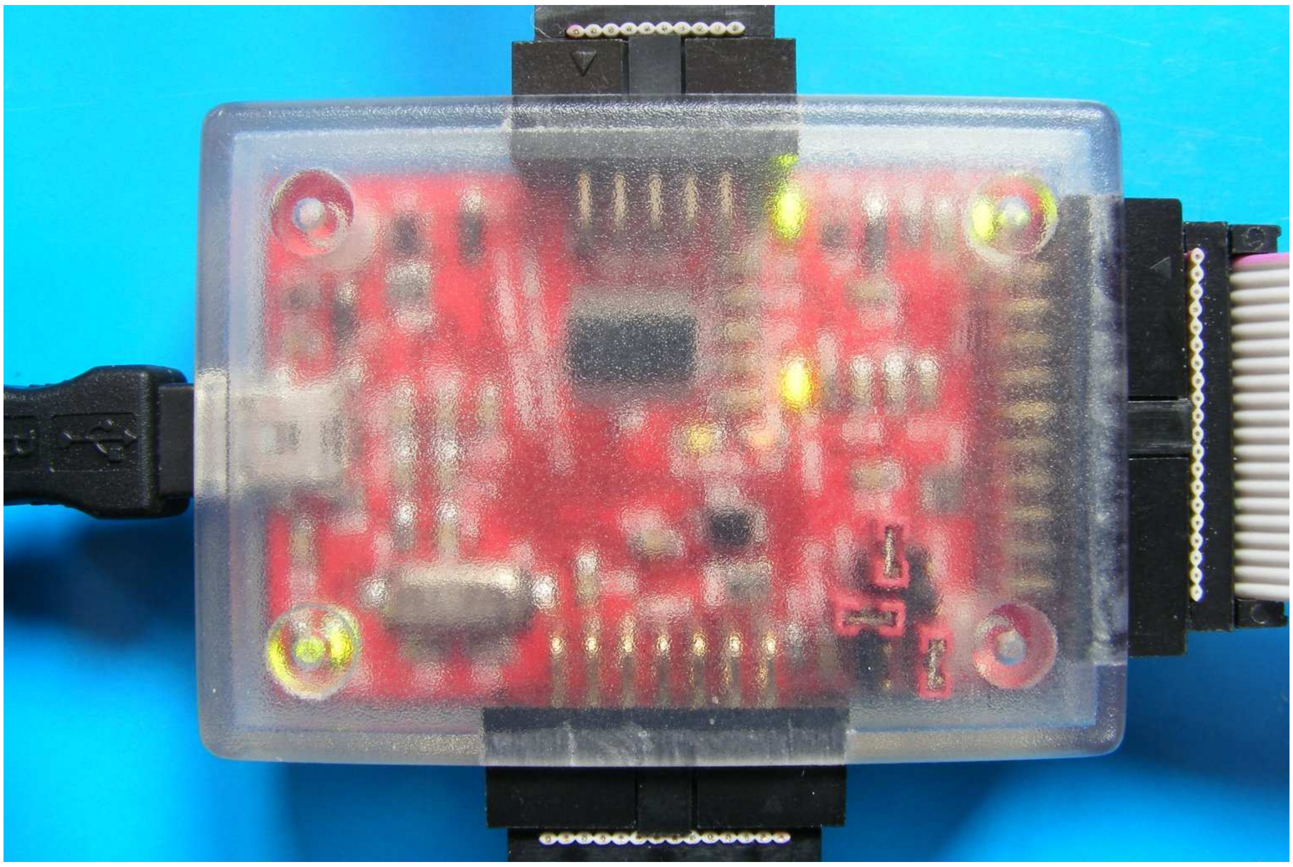
JTAG-lock-pick 1.x.x
Manual EN 2.0 / 120229

www.distortec.com
www.freddiechopin.info

Table of contents

1. Introduction

JTAG-lock-pick is an *ARM* core processors *JTAG* using *USB* bus to connect to *PC*. The device is based on *FTDI FT2232*[1] chip – a dual channel *USB <=> UART/FIFO* converter. By design first channel is connected to *JTAG* interface and the second one has full *UART / RS-232* interface (*RXD*, *TXD*, *RTS*, *CTS*, *DCD*, *DSR*, *DTR*, *RI*) with option to connect to *RS-485* converter. **JTAG-lock-pick** project was created to be an uncompromising and powerful solution, taking maximum advantage of all used *ICs*.



Pic. 1: *JTAG-lock-pick – device in enclosure*

Maximum frequency of *JTAG* interface's clock is *6MHz*. Thanks to *USB* bus the device can be connected to any *PC* on the market – it would not be possible with parallel interface (*LPT*), which is completely obsolete nowadays. Thanks to use of *74LVC* series advanced line buffers, target devices with very wide range of supply voltage – from about *1.6V* up to *5.5V* – can be connected.

One of project's requirements was to have both *UART* and *RS-232* interface. Selection between *RS-232* and *UART* is automatic and based on presence of *UART* interface's line buffers' supply voltage – if this voltage is higher than about *1.7V* then *UART* interface is enabled. Four most important signals of *UART* interface (*RXD*, *TXD*, *RTS*, *CTS*) are also buffered with *74LVC* series level shifters. Other lines are driven directly by *FT2232* chip – output signal's high level voltage is *5V*, input's high level voltage is (typically) about *1.6V*. Thanks to having *TXDEN* signal on *UART* connector, *RS-485* converter can be connected to the device.

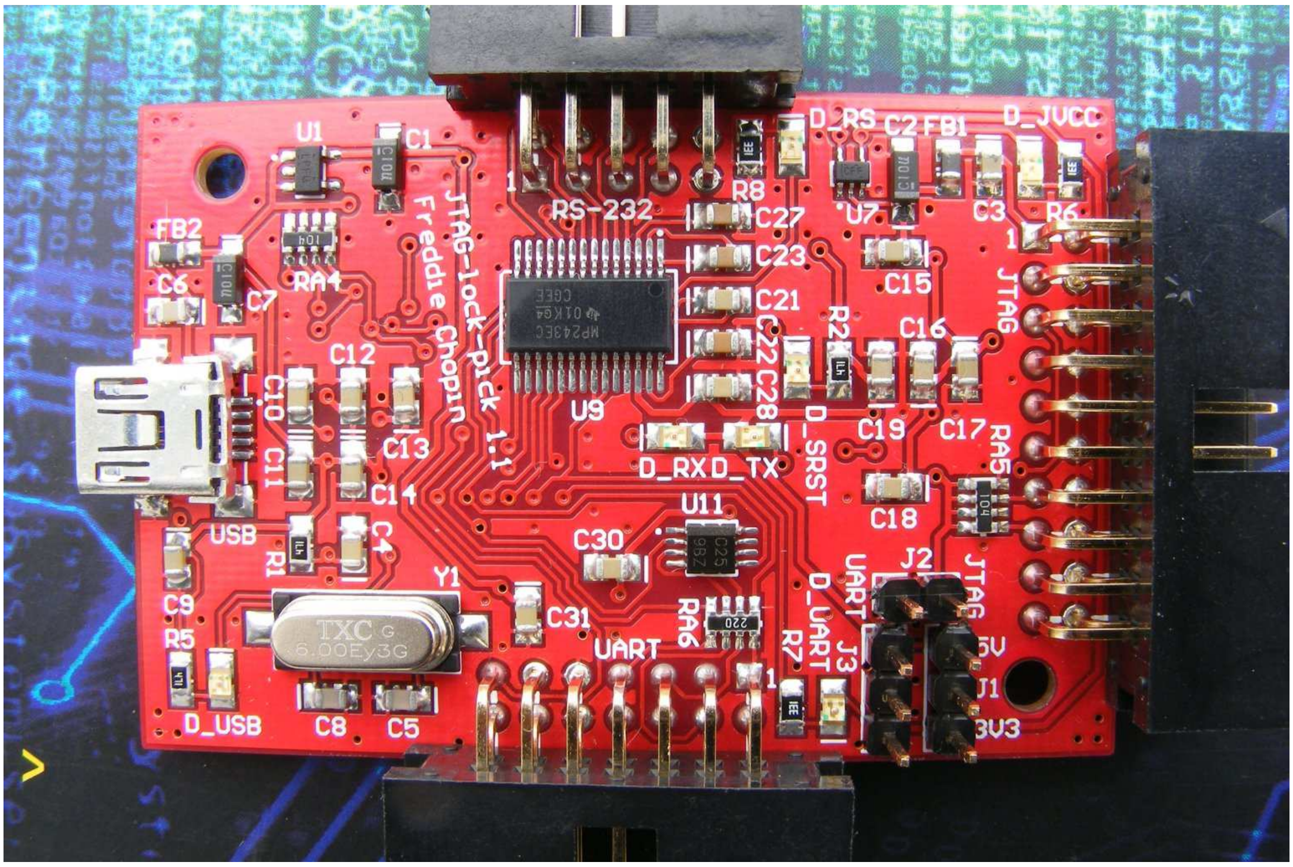
Designed circuit faciliates powering of line buffers – and thus target device (via *JTAG / UART* connectors) – with *3.3V* or *5V* voltage. *UART's* and *JTAG's* supply rails are separated, additional jumper

1 http://www.ftdichip.com/Products/ICs/FT2232D.htm

allows to connect them, so various voltage configurations are possible. *USB* bus is the source of these voltages, maximum current drawn by the target is *500mA* for *5V* voltage (according to specification) or *150mA* for *3.3V* voltage, which is more than enough for complex digital devices without power circuits.

JTAG-lock-pick is fully software compatible with *JTAGkey*[2] interface from *Amontec*, so in many applications available configurations can be used instead of creating them manually. It is also possible to program *FPGA*, *CPLD* and *AVR* chips with *SVF* files using **JTAG-lock-pick** and software provided by *Amontec*.

JTAG-lock-pick has separate *SRST* and *TRST* lines, which can be independently configured to *push-pull* or *open-drain* mode.

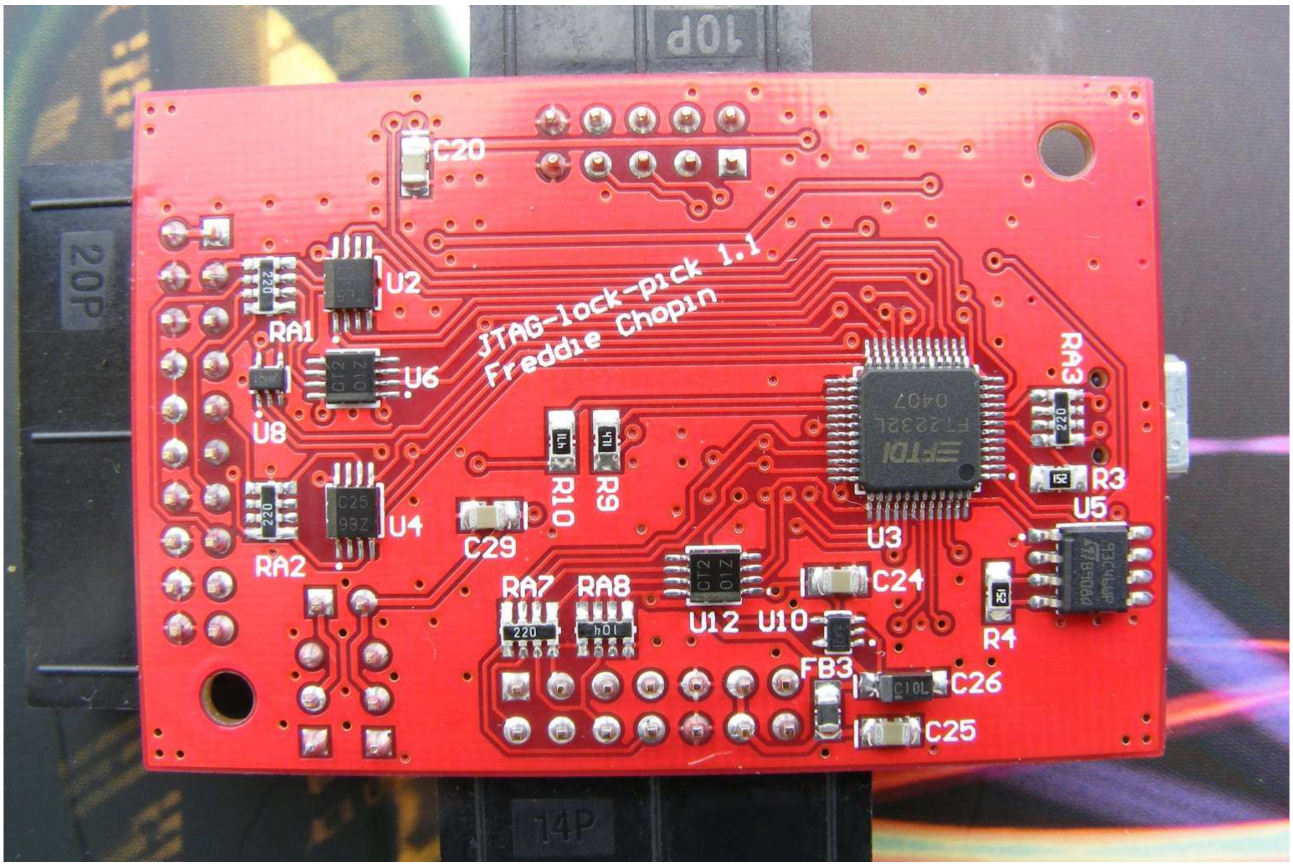


Pic. 2: *JTAG-lock-pick – top side of device's circuit*

JTAG-lock-pick project is a successor to **4R|\/|-JT4G Rev02**[3] project, eliminating it's shortcomings. Most important one – predecessor had *SRST* i *TRST* lines shorted *"by design"*, which limited debugging capabilities. Additionally the buffering *IC* used previously – *74VHC244* – had a significant flaw, which resulted in breakdown of input voltages through clamp diodes on unconnected supply rail, which was very confusing but perfectly safe for *JTAG* and target device. These problems do not occur in **JTAG-lock-pick**.

Picture 1 shows the device in enclosure, **pictures 2** and **3** show assembled **JTAG-lock-pick** circuit.

2 http://www.amontec.com/jtagkey.shtml
3 http://www.freddiechopin.info/index.php/pl/projekty/49-arm-jtag-rev01-rev02



***Pic. 3:** JTAG-lock-pick – bottom side of device's circuit*

1.1. Supported target chips

JTAG-lock-pick interface and *PC* software that uses it (among others: *OpenOCD*, *Atollic TrueSTUDIO*, *CooCox CoIDE*, *Keil MDK-ARM*, *IAR Embedded Workbench for ARM*, *Rowley CrossWorks for ARM*, see **chapter 4**) are able to communicate with almost any existing type of *ARM* processor, including the most popular::

- *ARM7* (*LPC2xxx*, *AT91SAM7*, *STR7xx*, ...),
- *ARM9* (*LPC3xxx*, *AT91SAM9*, *STR9xx*, ...),
- *Cortex-M3* (*STM32*, *LM3S*, *LPC17xx*, *AT91SAM3*, …),
- ...

The only limiting factor is support for specific chip in *PC* software.

1.2. “Strong points” of JTAG-lock-pick project

- safe and reliable communication with target devices with supply voltage in the range from about *1.6V* up to *5.5V*, signals buffered with *74LVC* series advanced level shifters,
- *UART* and *RS-232* interfaces, each with complete set of eight standard signals, four most important lines of *UART* interface are buffered (from about *1.6V* up to *5.5V*),
- ability to power target circuit through *JTAG* with *3.3V* or *5V* voltage,
- separated supply rails of *JTAG* and *UART*,
- safe enclosing, which protects *JTAG*, connected *PC* and target device from accidental damage,

- ability to connect *RS-485* converter,
- …

1.3. Contents of the package

- **JTAG-lock-pick** debugger / programmer (*JTAG* + *RS-232* + *UART*), machine assembled, tested, in transparent enclosure (outer dimensions *24mm x 47mm x 66mm*) with machine milled connector slots and with rubber feet, (**picture 1**, **2** and **3**),
- *JTAG <=> target* ribbon cable, *20cm*, (**picture 4**),
- *IDC-10 <=> DB-9* (*RS-232*) ribbon adapter cable, *20cm*, (**picture 5**),
- two *IDC-14* ribbon crimp connectors + *20cm* of *14-wire* ribbon cable to assemble any *UART* interface cable,
- *USB mini-B* cable, black, *1,8m*,
- *DVD* with manual, drivers, set of useful software – free (*CodeSourcery gcc toolchain*, *Eclipse IDE*, *OpenOCD*, *CooCox CoIDE*) and proprietary in evaluation versions (*Atollic TrueSTUDIO for STM32 Lite*, *Keil MDK-ARM* + *CooCox CoMDKPlugin*, *IAR Embedded Workbench for ARM* + *CooCox CoIARPlugin*, *Rowley CrossWorks for ARM*) – all of them in the most recent versions.

2. Hardware

2.1. Connectors

2.1.1. JTAG

Signals' assignment in *JTAG* connector (shown in **table 1**) is compatible with the so-called *"standard"*.

Table 1: *Pinout of JTAG connector*

JVREF – 1	*2 – JVREF*
nTRST – 3	*4 – GND*
TDI – 5	*6 – GND*
TMS – 7	*8 – GND*
TCK – 9	*10 – GND*
n/c – 11	*12 – GND*
TDO – 13	*14 – GND*
nSRST – 15	*16 – GND*
n/c – 17	*18 – GND*
n/c – 19	*20 – GND*

JVREF lines are supply rails for buffers (or the target circuit – if powering buffers from *PC's* side was selected). **Picture 4** shows typical *20-wire* ribbon connection cable.

2.1.2. RS-232

Signals' assignment in *RS-232* connector (shown in **table 2**) is compatible with standard *DB-9* connector. Use of ribbon cable with *DB-9* crimp connector (shown on **picture 5**) gives standard serial port connected to **JTAG-lock-pick**.

Table 2: *Pinout of RS-232 connector*

DCD – 1	*2 – DSR*
RXD – 3	*4 – RTS*
TXD – 5	*6 – CTS*
DTR – 7	*8 – RI*
GND – 9	*10 – GND*

Pic. 4: *Standard JTAG connection ribbon cable*

Pic. 5: *RS-232 connection ribbon cable*

2.1.3. UART

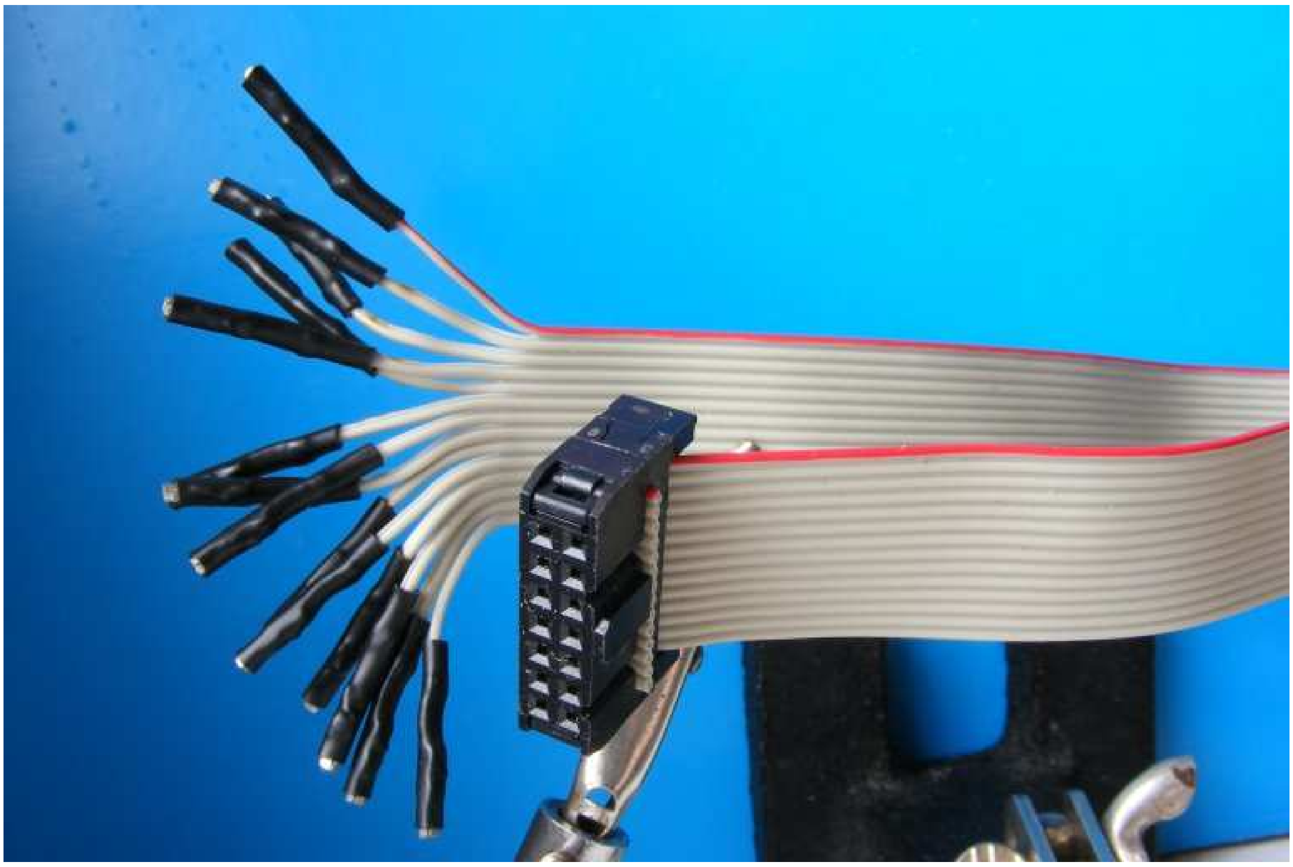
Signals' assignment in *UART* connector (shown in **table 3**) is partially compatible with *RS-232* connector.

Table 3: *Pinout of UART connector*

DCD – 1	*2 – DSR*
RXD – 3	*4 – RTS*
TXD – 5	*6 – CTS*
DTR – 7	*8 – RI*
GND – 9	*10 – GND*
UVREF – 11	*12 – UVREF*
TXDEN – 13	*14 – n/c*

UVREF lines are supply rails for buffers (or the target circuit – if powering buffers from *PC's* side was selected).

JTAG-lock-pick's *UART* interface can be connected with target circuit with straight *14-wire* ribbon cable (similar to the cable shown on **picture 4**), ribbon cable with *"flying-leads"* terminals (example of such cable is shown on **picture 6**) or a different cable – made for specific purpose.



Pic. 6: *Example of UART ribbon cable with "flying-leads" terminals*

Typical *RS-485* transceiver (*i.e. MAX485 IC*) can be connected to *UART* connector, thanks to having *TXDEN* signal – in such case this line controls the direction of transmission.

2.2. LEDs

On **JTAG-lock-pick's** *PCB* there are seven *LEDs*. Their meaning is as follows:

- *D_USB* (green), near *USB* connector – *USB* device enumeration finished, **JTAG-lock-pick** was properly discovered by operating system,
- *D_JVCC* (green), near *JTAG* connector – presence of valid supply voltage for *JTAG* interface's buffers (and target circuit),
- *D_SRST* (yellow), in the center of module – *nSRST* signal is in active (low) state – target device is in reset,
- *D_RS* (green), near *RS-232* connector – lack of valid supply voltage for *UART* interface's buffers, *RS-232* interface is active, *UART* interface is not active,
- *D_UART* (green), near *UART* connector – presence of valid supply voltage for *UART* interface's buffers, *UART* interface is active, *RS-232* interface is not active,
- *D_RX* (yellow), in the center of module – reception of character through *UART / RS-232* interface,
- *D_TX* (red), in the center of module – transmission of character through *UART / RS-232* interface.

2.3. Jumpers

On **JTAG-lock-pick's** *PCB* there are three jumpers used to configure power supplies. **Wrong configuration of jumpers can cause damage to *JTAG*, target circuit or *USB* port in *PC*!** All jumpers are removed by default.

2.3.1. J1 jumper – JVREF

Installing the jumper on position marked as *5V* or *3V3* causes *JTAG* interface's buffers (and target circuit via *JVREF* lines) to be powered with *5V* or *3.3V* respectively. It enables debugging of target circuits that do not provide supply voltage for buffers and / or powering target circuit through *JTAG*. **If target circuit has its own power supply, which is provided via *JVREF* lines, this jumper must not be installed!**

2.3.2. J2 jumper – connecting JVREF and UVREF

Installing this jumper connects *JVREF* and *UVREF* rails, which – for example – enables *UART* interface's buffers to be powered with voltage provided via *JVREF* lines. **If both *JVREF* and *UVREF* voltages are provided externally this jumper must not be installed!**

2.3.3. J3 jumper – UVREF

Role of this jumper is similar to *J1*, but it is responsible for voltage on *UVREF* lines which supplies *UART* interface's buffers.

3. Drivers

First connection of *JTAG* to computer should result in discovery of new *USB* devices, for which chosen drivers should be installed. Basically there are two options of drivers:

– open-source *libusb-win32* drivers,
– *FT2232* manufacturer's drivers based on *ftd2xx* library.

The choice depends on the software that will be used for debugging with **JTAG-lock-pick**. *OpenOCD* (see **chapter 4.1**) versions that are currently available in the internet use *libftdi*[4] library, so they need *libusb-win32* driver. Individual compilation of *OpenOCD*, so that it would use slightly faster (in *Windows*) *ftd2xx* driver, is possible (but complicated). *CooCox CoIDE* environment (see **chapter 4.3**) and *CooCox* plugins: *CoMDKPlugin* (for *Keil MDK-ARM*, see **chapter 4.5**) and *CoIARPlugin* (for *IAR Embedded Workbench for ARM*, see **chapter 4.4.2**) use *ftd2xx* driver. *Rowley CrossWorks for ARM* environment (see **chapter 4.6**) can use both kinds of drivers.

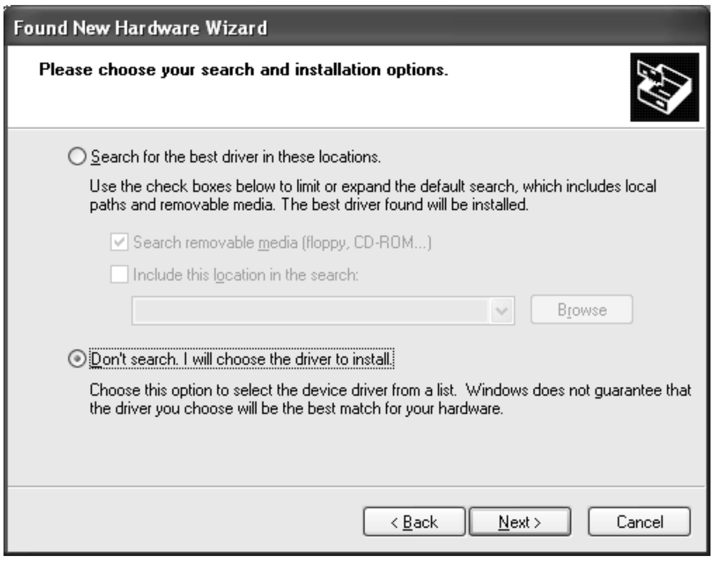
Archive with both types of drivers for **JTAG-lock-pick** is located on the *DVD* included in the package in *Drivers* folder. The most recent versions of drivers can be found on *DISTORTEC's*[5] website in *Download* section and on *Freddie Chopin's*[6] website in *Download → Projects → JTAG-lock-pick* section.

3.1. libusb-win32

This open-source driver is only for the first channel of *FT2232* chip – the one which is connected to *JTAG* interface. For the second channel – used for *UART / RS-232* interface – *ftd2xx* driver must be installed (see **chapter 3.2**).

It is recommended to install this driver in fully manual mode, in which the driver's file is selected directly, not searched by operating system – during installation this option is called *“Don't search. I will choose the driver to install.”* (see **picture 7**). Manual installation is also necessary when changing driver (see **chapter 3.3**).

Correct installation of driver is confirmed by appearance of *JTAG-lock-pick – USB <=> JTAG adapter* in *libusb-win32 devices* group in *Device Manager*. This situation (after installing *ftd2xx* driver for *UART/RS-232* interface) is shown on **picture 8**.



Found New Hardware Wizard

Please choose your search and installation options.

Search for the best driver in these locations.

Use the check boxes below to limit or expand the default search, which includes local paths and removable media. The best driver found will be installed.

☑ Search removable media (floppy, CD-ROM...)

☐ Include this location in the search:

[✓ Don't search. I will choose the driver to install.]

Choose this option to select the device driver from a list. Windows does not guarantee that the driver you choose will be the best match for your hardware.

***Pic. 7:** Manual installation of drivers*

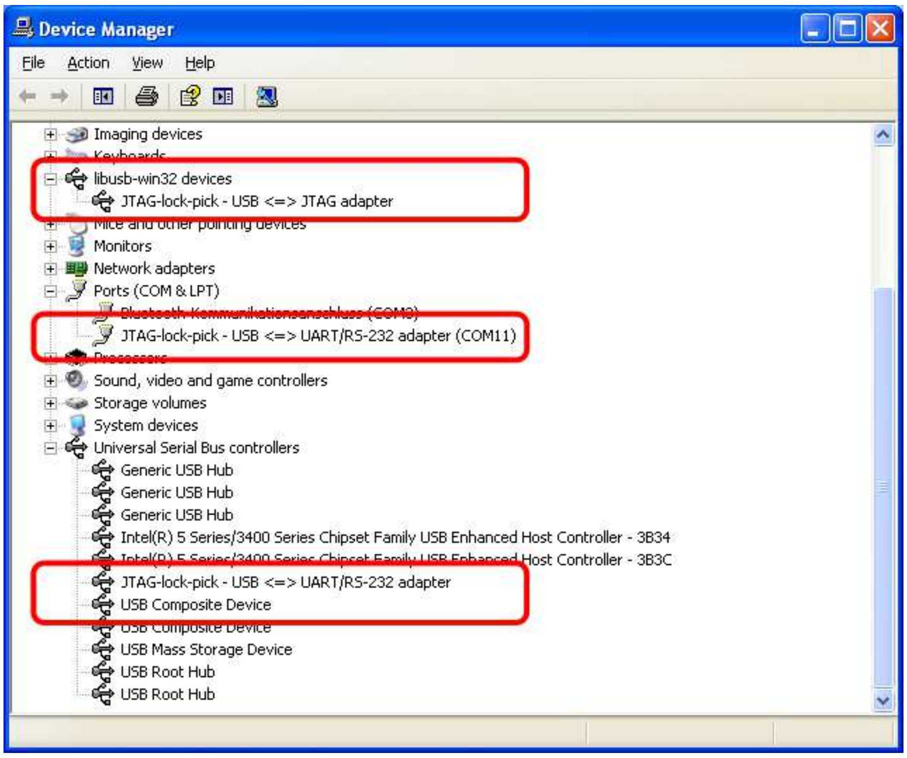
3.2. ftd2xx

After installation of these driver following four elements should appear in system's *Device Manager*:

4 Distribution of *OpenOCD* compiled to use *ftd2xx* proprietary driver allegedly violates *GPLv2* license
5 http://www.distortec.com/
6 http://www.freddiechopin.info/

- in *Universal Serial Bus Controllers* group: *JTAG-lock-pick - USB <=> JTAG adapter* (only if *ftd2xx* driver was installed for *JTAG* channel instead of *libusb-win32*), *JTAG-lock-pick - USB <=> UART/RS-232 adapter* and *USB Composite Device*;
- in *Ports (COM & LPT)* group: *JTAG-lock-pick - USB <=> UART/RS-232 adapter (COM xx)*;

Contents of *Device Manager* after installing whole set of *ftd2xx* drivers is shown on **picture 9**.

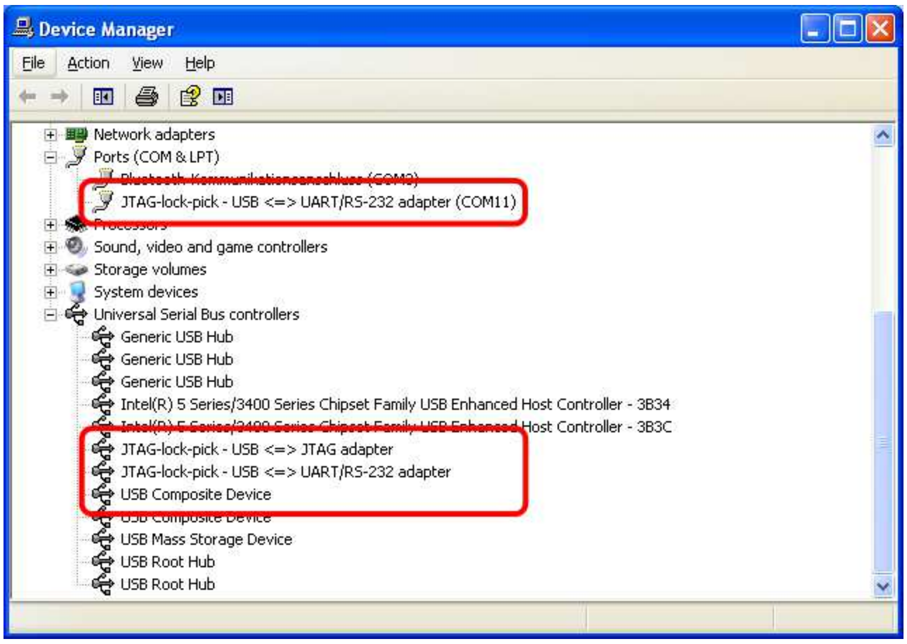


```
Imaging devices
Keyboards
libusb-win32 devices
    JTAG-lock-pick - USB <=> JTAG adapter
Mice and other pointing devices
Monitors
Network adapters
Ports (COM & LPT)
    Bluetooth-Kommunikationsanschluss (COM3)
    JTAG-lock-pick - USB <=> UART/RS-232 adapter (COM11)
Processors
Sound, video and game controllers
Storage volumes
System devices
Universal Serial Bus controllers
    Generic USB Hub
    Generic USB Hub
    Generic USB Hub
    Intel(R) 5 Series/3400 Series Chipset Family USB Enhanced Host Controller - 3B34
    Intel(R) 5 Series/3400 Series Chipset Family USB Enhanced Host Controller - 3B3C
    JTAG-lock-pick - USB <=> UART/RS-232 adapter
    USB Composite Device
    USB Composite Device
    USB Mass Storage Device
    USB Root Hub
    USB Root Hub
```

Pic. 8: *Installed libusb-win32 (JTAG) and ftd2xx (UART/RS-232) drivers in Device Manager*

3.3. Change of driver

If a need to change *JTAG* channel driver occurs (because of – for example – mistake or change of used software), *JTAG-lock-pick - USB <=> JTAG adapter* element in *Device Manager* should be located and double-clicked to open its properties. Then *Driver Update...* button in *Driver* tab should be used, which will open driver update wizard. It is important not to let the system search for the driver automatically – *“Don't search. I will choose the driver to install.”* option should be used (see **picture 7**) – otherwise the system will most likely install exactly the same driver's version again, without making any change.



```
Network adapters
Ports (COM & LPT)
    Bluetooth-Kommunikationsanschluss (COM3)
    JTAG-lock-pick - USB <=> UART/RS-232 adapter (COM11)
Processors
Sound, video and game controllers
Storage volumes
System devices
Universal Serial Bus controllers
    Generic USB Hub
    Generic USB Hub
    Generic USB Hub
    Intel(R) 5 Series/3400 Series Chipset Family USB Enhanced Host Controller - 3B34
    Intel(R) 5 Series/3400 Series Chipset Family USB Enhanced Host Controller - 3B3C
    JTAG-lock-pick - USB <=> JTAG adapter
    JTAG-lock-pick - USB <=> UART/RS-232 adapter
    USB Composite Device
    USB Composite Device
    USB Mass Storage Device
    USB Root Hub
    USB Root Hub
```

Pic. 9: *Installed ftd2xx drivers in Device Manager*

4. Software

List of programs in which **JTAG-lock-pick** can be used contains the most popular tools for ARM core processors, among others: *OpenOCD*, *Atollic TrueSTUDIO*, *CooCox CoIDE*, *Keil MDK-ARM*, *IAR Embedded Workbench for ARM* and *Rowley CrossWorks for ARM*. Usage of **JTAG-lock-pick** in this applications is described in details in following subchapters.

From software's point of view **JTAG-lock-pick** may be used as *Amontec JTAGkey*. Because of that, use of this device is very simple – in most cases used software will have available configurations for this *JTAG*.

4.1. OpenOCD[7]

Basic command to start *OpenOCD* should look like this:

```
> openocd -f interface/jtagkey.cfg -f target/XXX.cfg
```

or

```
> openocd -f interface/jtagkey.cfg -f board/XXX.cfg
```

(depending on where the configuration file for target device / chip is located).

For example – running *OpenOCD* for *STM32* chip (*ARM Cortex-M3*) should result in similar output as shown below (exact messages will – of course – depend on version of *OpenOCD* and used target device):

```
> openocd -f interface/jtagkey.cfg -f target/stm32f1x.cfg
Open On-Chip Debugger 0.5.0 (2011-08-09-23:21)
Licensed under GNU GPL v2
For bug reports, read
        http://openocd.berlios.de/doc/doxygen/bugs.html
Info : only one transport option; autoselect 'jtag'
1000 kHz
adapter_nsrst_delay: 100
jtag_ntrst_delay: 100
cortex_m3 reset_config sysresetreq
Info : clock speed 1000 kHz
Info : JTAG tap: stm32.cpu tap/device found: 0x3ba00477 (mfg: 0x23b,
part: 0xba00, ver: 0x3)
Info : JTAG tap: stm32.bs tap/device found: 0x16410041 (mfg: 0x020,
part: 0x6410, ver: 0x1)
Info : stm32.cpu: hardware has 6 breakpoints, 4 watchpoints
```

OpenOCD can also be used only for programming target from command line (or with batch files / scripts) with following command:

```
> openocd -f interface/jtagkey.cfg -f target/XXX.cfg -c "init; reset
halt; flash write_image erase YYY.EXT; reset run; shutdown"
```

This line consists of several *OpenOCD's* commands:

- *init* – required before following *"executable"* commands (other than configuration) in command line,
- *reset halt* – resets and halts the device, which is required prior to programming the target,
- *flash write_image erase YYY.EXT* – writes contents of *YYY.EXT* file (where *EXT* is *hex*, *bin* or *elf* extension) to target's flash memory, erasing it before programming (only affected

7 http://openocd.sourceforge.net/

range),
- *reset run* – resets and resumes target's operation, which results in execution of programmed firmware,
- *shutdown* – closes *OpenOCD's* session.

Example of programming *STM32* chip with *stm32_blink_led.hex* file is shown below.

```
> openocd -f interface/jtagkey.cfg -f target/stm32f1x.cfg -c "init;
reset halt; flash write_image erase stm32_blink_led.hex; reset run;
shutdown"
Open On-Chip Debugger 0.5.0 (2011-08-09-23:21)
Licensed under GNU GPL v2
For bug reports, read
        http://openocd.berlios.de/doc/doxygen/bugs.html
Info : only one transport option; autoselect 'jtag'
1000 kHz
adapter_nsrst_delay: 100
jtag_ntrst_delay: 100
cortex_m3 reset_config sysresetreq
Info : clock speed 1000 kHz
Info : JTAG tap: stm32.cpu tap/device found: 0x3ba00477 (mfg: 0x23b,
part: 0xba00, ver: 0x3)
Info : JTAG tap: stm32.bs tap/device found: 0x16410041 (mfg: 0x020,
part: 0x6410, ver: 0x1)
Info : stm32.cpu: hardware has 6 breakpoints, 4 watchpoints
Info : JTAG tap: stm32.cpu tap/device found: 0x3ba00477 (mfg: 0x23b,
part: 0xba00, ver: 0x3)
Info : JTAG tap: stm32.bs tap/device found: 0x16410041 (mfg: 0x020,
part: 0x6410, ver: 0x1)
target state: halted
target halted due to debug-request, current mode: Thread
xPSR: 0x01000000 pc: 0x0800024c msp: 0x20000ed0
auto erase enabled
Info : device id = 0x20036410
Info : flash size = 128kbytes
wrote 2048 bytes from file stm32_blink_led.hex in 0.312500s (6.400
KiB/s)
Info : JTAG tap: stm32.cpu tap/device found: 0x3ba00477 (mfg: 0x23b,
part: 0xba00, ver: 0x3)
Info : JTAG tap: stm32.bs tap/device found: 0x16410041 (mfg: 0x020,
part: 0x6410, ver: 0x1)
shutdown command invoked
```

4.2. Atollic TrueSTUDIO[8]

According to information on *Atollic's* website full version of *Atollic TrueSTUDIO* environment has full support for *FTDI FT2232* based interfaces[9] (including **JTAG-lock-pick**, thanks to compatibility with *JTAGkey*). Free *Lite* version for *STM32* chips can still work with **JTAG-lock-pick** via *OpenOCD* (see **chapter 4.1**), but the cooperation is not completely smooth.

Before attempting any configuration the project should be built so that executable file with *.elf* extension has been created. *OpenOCD* should be started in the background – *"externally"* (using operating system's command line) or by configuring it in *Run > External Tools > External Tools Configurations...* menu, example command which starts *OpenOCD* for *STM32* microcontrollers should look like this:

8 http://www.atollic.com/index.php/truestudio
9 http://www.atollic.com/index.php/truestudio/targets/jtagdongles/amontec

```
> openocd -f interface/jtagkey.cfg -f target/stm32f1x.cfg -c
"reset_config trst_and_srst"
```

Debugging configuration can be performed with *Debug Configurations...* option from *Run* menu. In newly opened window new *Embedded C/C++ Application* type configuration should be created and configured in following order:

1. in *Main* tab the name of the project that is meant to be debugged should be typed into (or selected with *Browse...* button) into *Project* field, and then name of executable *.elf* file should by typed into (or selected with *Search Project...* or *Browse...* button) *C/C++ Application* field just above; in most cases these fields will be filled automatically when creating debugging configuration;
2. in *Debugger* tab *Connect to remote GDB server* option should be chosen and *3333* value should be entered into *Port number* field, all other options are ignored – their values do not affect anything and they should be considered nonfunctional;
3. in *Startup Debug* tab whole content of *Initialization Commands* field should be removed and replaced with following lines:

```
# send "reset halt" to OpenOCD
monitor reset halt
# load application to target via GDB
load
# send "reset halt" to OpenOCD
monitor reset halt
# set temporary breakpoint at main() and resume target
tbreak main
continue
```

4. all changes should be confirmed with *Apply* button and debugging session can be started right away with *Debug* button.

Problems that should be expected during cooperation of *Lite* version of *Atollic TrueSTUDIO* environment with *OpenOCD* are (among others):

– target device cannot be *“reset”* during debugging session with *Restart* button;
– operating system informs about improper termination of *GDB* application when debugging session is closed;

4.3. CooCox CoIDE[10]

In this environment available configuration for *Amontec JTAGkey* should be used. All options related to *JTAG* interface configuration can be found in *Debug > Debug Configuration* menu, after selecting the debug configuration for active project that will be available there (typically it will be called *project_name.configuration*). *Amontec-JTAGkey* option should be selected from the *Adapter* list in *Hardware* groupbox in the first tab (*Debugger*) and then click *Apply* button and *Close* button.

4.4. IAR Embedded Workbench for ARM[11]

In *IAR's* environment two approaches to using **JTAG-lock-pick** interface are possible – one can use *OpenOCD* (see **chapter 4.1**) or *CooCox CoIARPlugin*.

10 http://www.coocox.org/CooCox_CoIDE.htm
11 http://www.iar.com/en/Products/IAR-Embedded-Workbench/ARM/

4.4.1. OpenOCD

First step is – obviously - starting *OpenOCD* in background (via operating system's command prompt) with parameters matching target device that is used, usually the call will look more or less like this:

```
openocd -f interface/jtagkey.cfg -f target/XXX.cfg
```

Then in *IAR Embedded Workbench for ARM* environment's project options (*Project > Options*) *Debugger* option should be selected from the side menu. In *Setup* tab *GDB Server* option should be selected from *Driver* list and in *Download* tab *Use flash loader(s)* option should be checked. Then select *GDB Server* option from side menu and in *GDB Server* tab enter *localhost* into *TCP/IP address of hostname [.port]* field.

4.4.2. CooCox CoIARPlugin[12]

After installing the plugin select *Debugger* in side menu of project's options (*Project > Options*). In *Setup* tab *RDI* option should be selected from *Driver* list and in *Download* tab *Use flash loader(s)* option should be checked. Then select *RDI* option from side menu, type path to *CoRDI.dll* file located in *CooCox CoIARPlugin* installation folder (typically it will be *c:\Program Files\CooCox\CoIARPlugin\CoRDI.dll*, in *64-bit* system *Program Files (x86)* folder will be used) into *Manufacturer RDI driver* field and check *Allow hardware reset* option. After closing project's options with *OK* button, new menu – *RDI* – will appear in application. The only active option – *Configure* – should be selected from this menu – in newly opened window target device should be selected from the side list and *Amontec-JTAGkey* option should be selected from the *Adapter* list in *Adapter Config* groupbox.

4.5. Keil MDK-ARM[13]

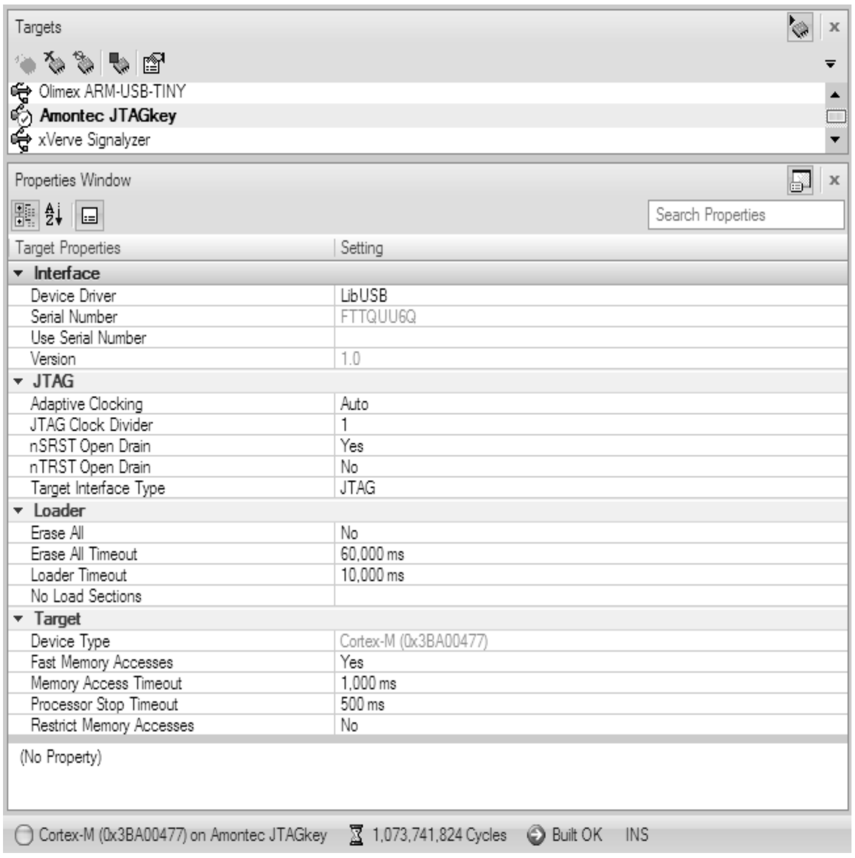
Use of **JTAG-lock-pick** interface in *Keil MDK-ARM* environment (also known as *µVision* or *RealView*) is possible with *CooCox CoMDKPlugin*[14]. After installing it, in project's options (*Project > Options for Target 'project_name'*, option available only when project is selected in *Projects* window) in *Utilities* tab *CooCox Debugger* should be selected from the list under *Use Target Driver for Flash Programming*, *Update Target before Debugging* option should be checked and *Settings* button should be clicked. In newly opened window go to *Debug* tab and select *JTAGkey* from *Adapter* list in *USB Adapter* groupbox. After pressing *OK* button and going back to project's options go to *Debug* tab and select *Use* option on the right side, then – again – select *CooCox Debugger* option from the list next to it – there is no need to configure it again with *Settings* button, because this configuration is shared (the same for *Utilities* and *Debug* tabs). The last options that should be checked are located just below: *Load Application at Startup* and *Run to main()*. Whole configuration should be confirmed with *OK* button.

4.6. Rowley CrossWorks for ARM[15]

Whole configuration of **JTAG-lock-pick** in *Rowley CrossWorks for ARM* environment is limited to double-clicking on *Amontec JTAGkey* option in *Targets* window. Proper connection with target device is indicated on application's status bar and by appearance of *JTAG's* serial number and tar-

12 http://www.coocox.org/CoLinkGuide/CoIARPlugin.html
13 http://www.keil.com/arm/mdk.asp
14 http://www.coocox.org/CoLinkGuide/CoMDKPlugin.html
15 http://www.rowley.co.uk/arm/index.htm

get's *Device ID* in *Properties Window* for *Amontec JTAGkey* – this situation is shown on **picture 10**. In some cases adjustment of *JTAG Clock Divider* value may be necessary.



Target Properties	Setting
Interface	
Device Driver	LibUSB
Serial Number	FTTQUU6Q
Use Serial Number	
Version	1.0
JTAG	
Adaptive Clocking	Auto
JTAG Clock Divider	1
nSRST Open Drain	Yes
nTRST Open Drain	No
Target Interface Type	JTAG
Loader	
Erase All	No
Erase All Timeout	60,000 ms
Loader Timeout	10,000 ms
No Load Sections	
Target	
Device Type	Cortex-M (0x3BA00477)
Fast Memory Accesses	Yes
Memory Access Timeout	1,000 ms
Processor Stop Timeout	500 ms
Restrict Memory Accesses	No

(No Property)

Cortex-M (0x3BA00477) on Amontec JTAGkey   1,073,741,824 Cycles   Built OK   INS

Pic. 10: *Proper connection with target device in Rowley CrossWorks for ARM environment*

5. Sources of additional information

Additional information and support about *JTAG*, software, debugging and *ARM* core processors can be found on many websites:

- *DISTORTEC's* website (http://www.distortec.com/),
- *Freddie Chopin's* website (http://www.freddiechopin.info/),
- *Elektroda's* forum (http://www.elektroda.pl/rtvforum/),
- *SparkFun's* forum (http://forum.sparkfun.com/viewforum.php?f=18),
- *OpenOCD's* website (http://openocd.sourceforge.net/),
- *Yagarto* toolchain's website (http://www.yagarto.de/),
- *WinARM* toolchain's website (information often out of date!) (http://www.siwawi.arubi.uni-kl.de/avr_projects/arm_projects/),
- *google* (http://www.google.com/).

6. Troubleshooting

Problem: After starting *OpenOCD* following message appears:

```
OpenOCD Error: unable to open ftdi device
```

Cause: *OpenOCD* can use two different kinds of drivers to communicate with *FTDI FT2232* chip – proprietary *ftd2xx* or open-source *libusb-win32*. The choice is made at the stage of compilation of *OpenOCD*. This message can point to several issues:

1. *JTAG* is not properly connected to the computer,
2. *JTAG* is *"blocked"* by another application or by another session of *OpenOCD*,
3. Wrong driver was installed for *JTAG*.

Solution: Ad 1. Check *JTAG <=> PC* connection.
Ad 2. Close all other *OpenOCD* sessions or other software connected with *JTAG*.
Ad 3. Uninstall wrong and install correct driver (see **chapter 3.3**).

Problem: Attempt to program / debug target device in *CooCox CoIDE* environment results in following error:

```
Target Chip not found
```

Cause: *CooCox CoIDE* environment in version *1.4.0* (the most recent at the time of writing this document) does not work correctly with *FTDI FT2232* based *JTAGs*.

Solution: Previous version *1.3.0* should be used or higher versions should be tested when they are published.

7. Manual changelog

1.0 (19/08/2009)	
1.1 (29/08/2009)	1. Change of information about location of *libusb-win32* driver – it's no longer bundled with *OpenOCD*, archive with both types of drivers for **JTAG-lock-pick** is available on the website 2. Added information about *OpenOCD* subforum on *SparkFun's* forum
1.2 (15/03/2010)	1. Minor fixes 2. Added information about version *1.1* of the project (Hardware changelog chapter, update of photos, added circuit's schematic diagram and bill of materials) 3. Two print-screens of operating system's *Device Manager* added to chapter about final drivers 4. Update of console output after starting version *0.4.0* of *OpenOCD*
1.3 (26/12/2010)	1. Minor fixes 2. Added information about version *1.1.1* of *PCB* 3. Information about *EEPROM* programming with *FT_Prog* software 4. *Rowley CrossWorks for ARM* environment: information about support for *libusb-win32* driver, update of print-screen 5. Joining of multiple commands passed to *OpenOCD* with *"-c"* option into one string, separating elements with semicolon
1.3.1 (10/12/2011)	1. Minor fixes 2. Detailed description of condition which requires *"base"* drivers to be installed and *EEPROM* to be programmed
2.0 (29/02/12)	1. Fixes to most of descriptions 2. Removal of information about elements and versions of *JTAG* (only assembled *FULL* version is available) 3. Added supported target devices, *"strong points"* and contents of the package 4. Removal of information about *"base"* drivers and *EEPROM* programming (available versions come with programmed *EEPROM*) 5. Added information about change of driver 6. Added information about other software which support **JTAG-lock-pick** 7. Removal of circuit's schematic diagram and bill of materials for version *1.0* (only version *1.1* is available) 8. Added English translation

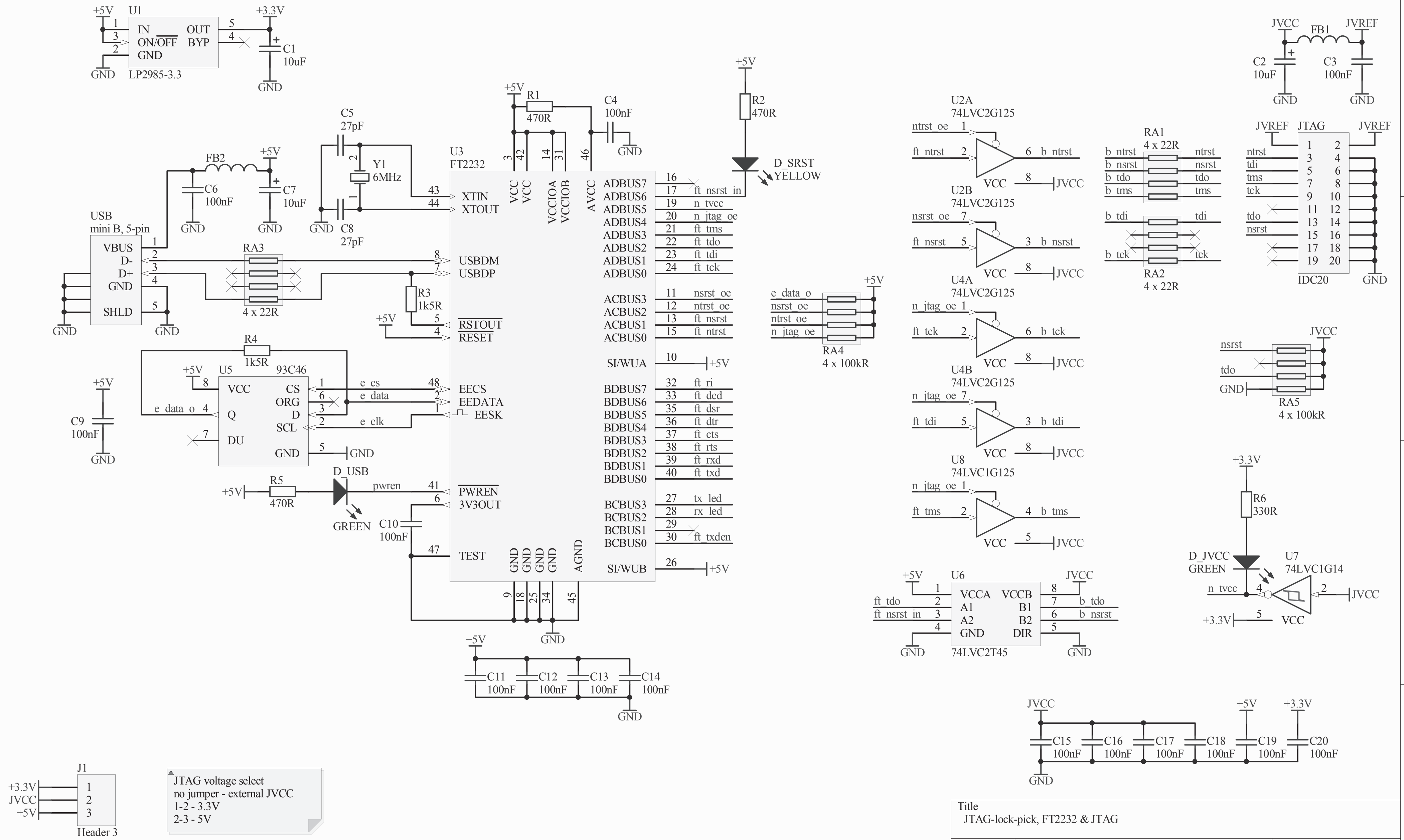
8. Hardware changelog

1.0 (19/02/2009)	
1.1 (31/01/2010)	Schematic: 1. Minor fixes 2. *LP2980 IC* (*50mA max*) replaced with *LP2985 IC* (*150mA max*) (*ICs* are fully compatible) 3. Change of order of *LEDs* and resistors in two places (resistor-diode instead of diode-resistor) – *D_UART* i *D_RS LEDs* can be moved further away from *IDC* connectors, which makes them more visible 4. *RA4* resistor array value changed from *4 x 10kR* to *4 x 100kR* *PCB*: 1. Minor fixes 2. Removal of all tracks *"directly"* joining neighbouring pads (in *MAX3243 IC* and in *4 x 100kR* resistor arrays) 3. Matching of *PCB's* mechanics to measured dimensions of *Z-24A* enclosing (position of mount holes, *IDC-20* connector, *D_USB* and *D_JVCC LEDs*) 4. Moving *IDC* connectors *"deeper"*
1.1.1 (22/11/2010)	*PCB*: 1. Matching mount holes to (finally) published exact drawings of *Z-24A* enclosure

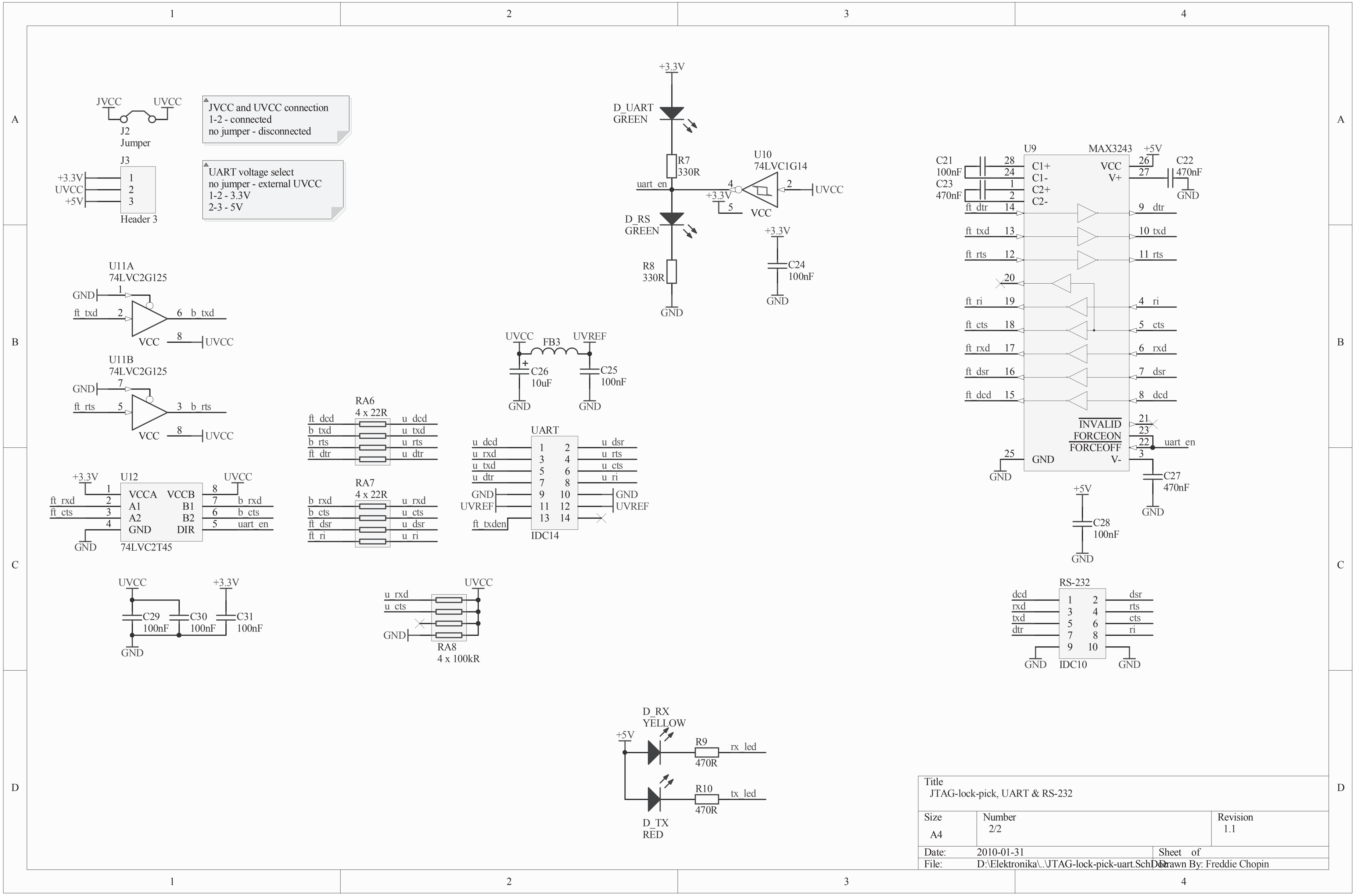
9. Appendix

- circuit's schematic diagram (version *1.1*), page 24
- bill of materials (version *1.1*), page 27

9.1. Circuit's schematic diagram (version 1.1)



Title JTAG-lock-pick, FT2232 & JTAG		
Size A4	Number 1/2	Revision 1.1
Date: 2010-01-31		Sheet of
File: D:\Elektronika\..\JTAG-lock-pick-main.SchDoc		Drawn By: Freddie Chopin



Title
JTAG-lock-pick, UART & RS-232

Size	Number	Revision
A4	2/2	1.1

Date: 2010-01-31 Sheet of
File: D:\Elektronika\..\JTAG-lock-pick-uart.SchDoc Drawn By: Freddie Chopin

9.2. Bill of materials (version 1.1)

JTAG-lock-pick 1.1 - Bill Of Materials

31.01.2010

Designator	Description	Comment	Footprint	Model:Footprint	Value	Quantity
C1, C2, C7, C26	Polarized Tantalum Capacitor	Polarized capacitor	CAPMP3216X18M	Molded Capacitor, 2-Leads, Body 3,2x1,6mm, IPC Low Density	10uF	4
C3, C4, C6, C9, C10, C11, C12, C13, C14, C15, C16, C17, C18, C19, C20, C21, C24, C25, C28, C29, C30, C31	Capacitor	Capacitor	CAPC2012M	Chip Capacitor, Body 2.0x1.3mm, IPC Low Density	100nF	22
C5, C8	Capacitor	Capacitor	CAPC2012M	Chip Capacitor, Body 2.0x1.3mm, IPC Low Density	27pF	2
C22, C23, C27	Capacitor	Capacitor	CAPC2012M	Chip Capacitor, Body 2.0x1.3mm, IPC Low Density	470nF	3
D_JVCC, D_RS, D_UART, D_USB	Typical LED	GREEN	CAPC2012M	Chip Capacitor, Body 2.0x1.3mm, IPC Low Density		4
D_RX, D_SRST	Typical LED	YELLOW	CAPC2012M	Chip Capacitor, Body 2.0x1.3mm, IPC Low Density		2
D_TX	Typical LED	RED	CAPC2012M	Chip Capacitor, Body 2.0x1.3mm, IPC Low Density		1
FB1, FB2, FB3	Choke	Choke	RESC2012M	Chip Resistor, Body 2.0x1.3mm, IPC Low Density		3
J1, J3	Header, 3-Pin	Header 3	HDR1X3	Connector; Header; 3 Position		2
J2	Jumper Wire	Jumper	HDR1X2	Connector; Header; 2 Position		1
JTAG	Header, 10-Pin, Dual row	IDC20	IDC-20, angled	IDC header, angled, 20-pin		1
R1, R2, R5, R9, R10	Resistor	Resistor	RESC2012M	Chip Resistor, Body 2.0x1.3mm, IPC Low Density	470R	5
R3, R4	Resistor	Resistor	RESC2012M	Chip Resistor, Body 2.0x1.3mm, IPC Low Density	1k5R	2
R6, R7, R8	Resistor	Resistor	RESC2012M	Chip Resistor, Body 2.0x1.3mm, IPC Low Density	330R	3
RA1, RA2, RA3, RA6, RA7	Quad chip resistor array, 4D03	Resistor array	RESA3216X06M	Chip Resistor Array, 8-Leads, Body 3,2x1,6mm, IPC Low Density	4 x 22R	5
RA4, RA5, RA8	Quad chip resistor array, 4D03	Resistor array	RESA3216X06M	Chip Resistor Array, 8-Leads, Body 3,2x1,6mm, IPC Low Density	4 x 100kR	3
RS-232	Header, 5-Pin, Dual row	IDC10	IDC-10, angled	IDC header, angled, 10-pin		1
U1	Micropower 150 mA Low Noise Ultra Low-Dropout Regulator	LP2985-3.3	SOT-95P-284X119-5M	SOT23, 5-Leads, Body 3,0x3,0mm (max), Pitch 0,95mm, IPC Low Density		1
U2, U4, U11	Dual bus buffer gate with 3-state outputs	74LVC2G125	SOP-65P-400X130-8M	TSOP, 8-Leads, Body 3,2x2,9mm (max), Pitch 0,65mm, IPC Low Density		3
U3	Dual USB UART / FIFO I.C.	FT2232	TSQFP-50P-900X900X160-48M	TSQFP, 48-Leads, Body 9,0x9,0mm (max), Pitch 0,50mm, IPC Low Density		1
U5	1K (64 x 16 or 128 x 8) Serial Microwire EEPROM	93C46	SOIC127P600X175-8M	SOIC, 8-Leads, Body 5,0x4,0mm (max), Pitch 1,27mm, IPC Low Density		1
U6, U12	Dual-bit dual-supply bus transceiver with configurable voltage translation and 3-state outputs	74LVC2T45	SOP-65P-400X130-8M	TSOP, 8-Leads, Body 3,2x2,9mm (max), Pitch 0,65mm, IPC Low Density		2
U7, U10	Single Schmitt-trigger inverter	74LVC1G14	SOT-65P-212X110-5M	SOT23, 5-Leads, Body 2,3x2,3mm (max), Pitch 0,65mm, IPC Low Density		2
U8	Single bus buffer gate with 3-state output	74LVC1G125	SOT-65P-212X110-5M	SOT23, 5-Leads, Body 2,3x2,3mm (max), Pitch 0,65mm, IPC Low Density		1
U9	±15kV ESD-Protected, 1µA, 3.0V/5.5V, 250kbps, RS-232 Transceiver with AutoShutdown	MAX3243	SOP65P710X200-28M	SOP, 28-Leads, Body 10,3x5,4mm (max), Pitch 0,65mm, IPC Low Density		1
UART	Header, 7-Pin, Dual row	IDC14	IDC-14, angled	IDC header, angled, 14-pin		1
USB	USB 2.0, Right Angle, SMT, B Type, Receptacle, 5 Position, Black	mini B, 5-pin	USB mini-B SMD	Connector; USB2.0, Type B, SM; 5 Position; Right Angle		1
Y1	Crystal Oscillator	6MHz	HC-49S	Quartz Crystal SMD		1