

OpenOCD — Open On-Chip Debugger

Linuxtag an der Hochschule Augsburg

Hubert Högl

`Hubert.Hoegl@hs-augsburg.de`

`http://www.hs-augsburg.de/~hhoegl`

Hochschule Augsburg

27. März 2010

Mikrocontroller programmieren

- Zielhardware (**Target**) ↔ Entwicklungsrechner (**Host**)
- **Crosscompiler** auf Host, z.B. arm-elf-gcc
- **Übertragen** der Programme mit verschiedenen Mechanismen
 - **UV-EPROM** einstecken (prähistorisch)
 - **Monitorprogramm** – lange Tradition, **U-Boot** ganz modern
 - **Bootloader** – modern, meist intern im μC Flash, Beispiel: Atmel SAM-BA
 - Hardware **Debug-Schnittstelle** (JTAG, BDM)

Debuggen auf dem Target

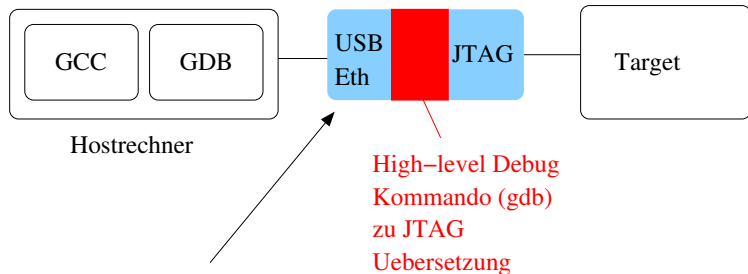
Einfache Mittel:

- Ausgaben auf LEDs bzw. Anzeige
- `uart_send()`, `uart_receive()`
- `printf()`

Mit richtigem **Debugger**, meist `gdb`

- **Debug-Stub** auf Target und `gdb` "remote" Protokoll über UART oder Netzwerk.
- GDB über **JTAG** Schnittstelle des Mikrocontrollers.
 μ C kann völlig "leer" sein.
Frage: Wie kommt GDB auf die Hardware?

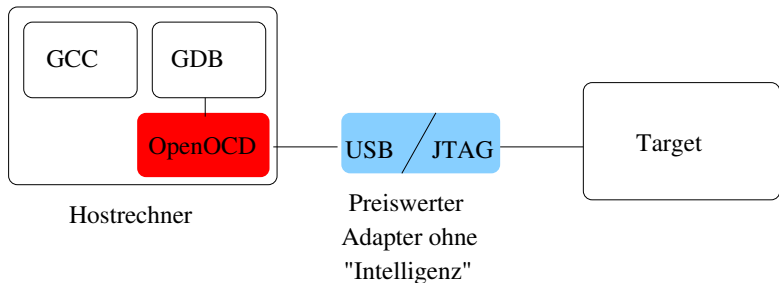
Lücke zwischen GDB und Target



Markt für teure **kommerzielle Geräte**

- Abatron BDI2000
- EPI Jeeni
- Lauterbach
- viele andere

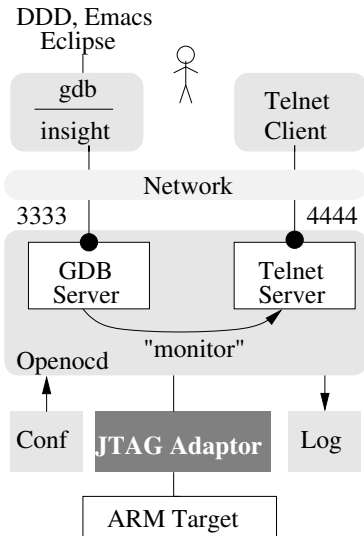
OpenOCD füllt diese Lücke



- **OpenOCD Server** übersetzt Debug Kommando auf JTAG.
- Preiswerter **Adapter** der lediglich JTAG Signale ansteuert.
- **OpenOCD komplettiert die GNU Toolchain für Embedded Systems.**
- Target-Architekturen: **ARM**, MIPS, andere möglich

OpenOCD's Schnittstellen

- Konfiguration
- Telnet port
- GDB remote port
- Logging
- JTAG Interface



Wie gross ist OpenOCD?

sloccount (2010-03-22):

ansic:	100148	(93.71%)
tcl:	6055	(5.67%)
asm:	519	(0.49%)
java:	90	(0.08%)
sh:	62	(0.06%)

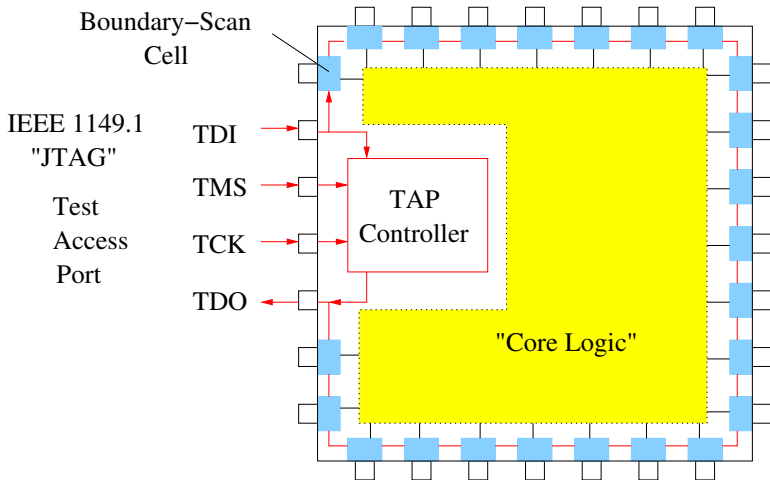
Aufwand: ca. **27** "Mannjahre"

Konfiguration: **47** Interfaces (tcl/interface/), **66** Boards (tcl/board/), **86** Targets (tcl/target/).

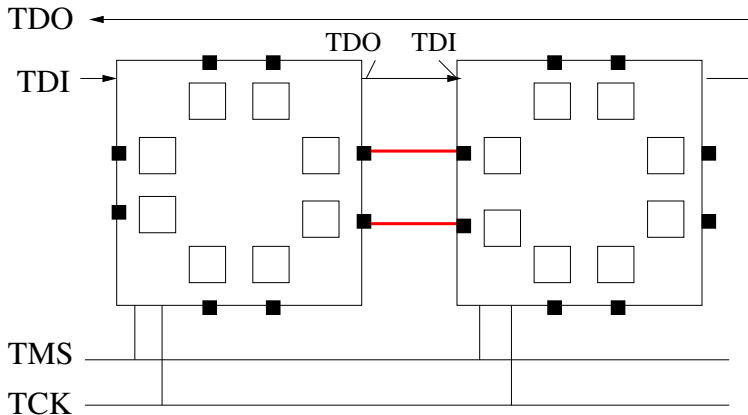
JTAG

- **J**oint **T**est **A**ction **G**roup (80er Jahre).
- Zum Test von Verbindungen auf der Platine konzipiert (**B**oundary **S**can).
- Standard **IEEE 1149.1** (1990, 2001)
- **B**oundary **S**can **C**ells (BSC) umgeben die "Core Logic".
- Die BSCs sind in einer **Schieberegisterkette** (chain) aneinander gereiht.
- **B**oundary **S**can **F**unktionen
 - Trenne Pin vom Logikkern.
 - Signal auf Pin geben (1, 0, Z)
 - Pin-Zustand einlesen.
- **T**est **A**ccess **P**ort: TDI, TDO, TCK, TMS
- TAP Controller

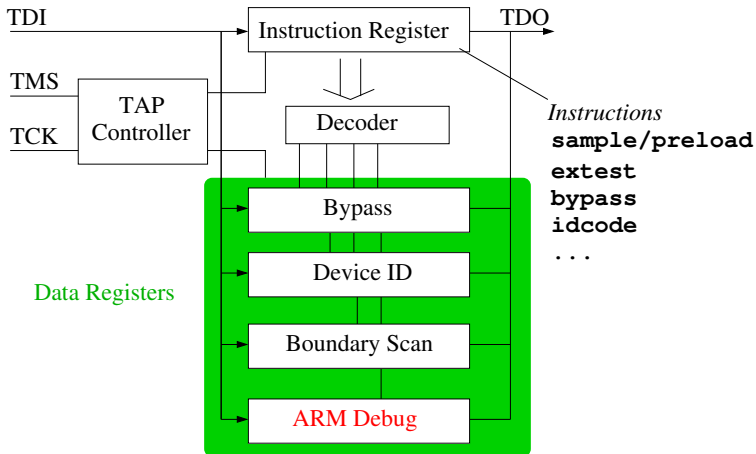
JTAG Port



JTAG Boundary Scan Test

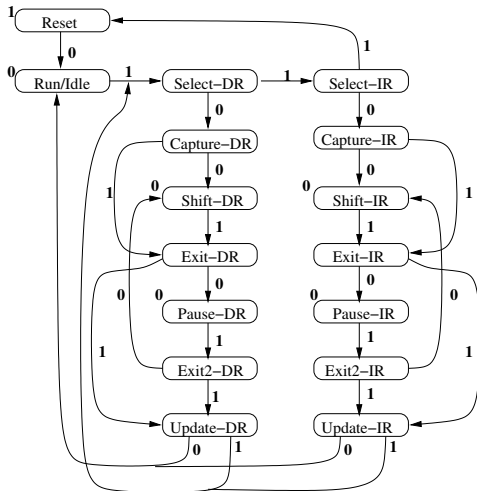


TAP Controller, IR und DR



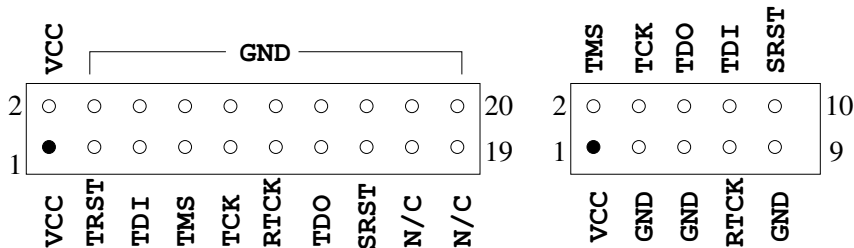
TMS: Die JTAG Zustandsmaschine

- TMS zur "Navigation"
- DR = Data Register
- IR = Instruction Register



JTAG Steckverbinder

Standard ARM JTAG Steckverbinder (links 20-polig, rechts 10-polig):



JTAG Adapter

Mit FT2232

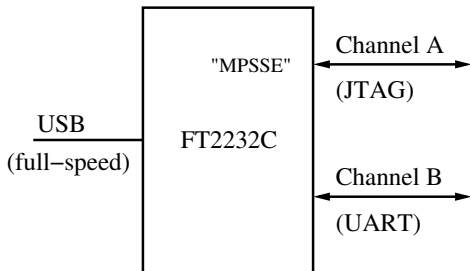
- JTAGkey-Tiny (Amontec)
- ARM-USB-OCD (Olimex)
- ARM-JTAG (Eproo)
- Joern Kaipf, <http://www.joernonline.de>

Ohne FT2232

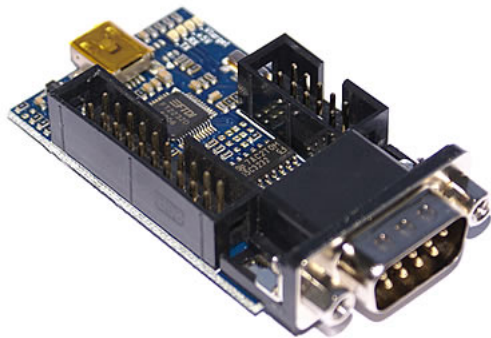
- USBprog <http://www.embedded-projects.net>

FT2232

- FT2232 = 2 × 232
(ca. 2003)
- full-speed device
- 2 Kanäle (UART + JTAG)
- Fast in allen
USB-zu-JTAG
Adapttern
- Neu: FT2232H
(high-speed)
- www.ftdichip.com

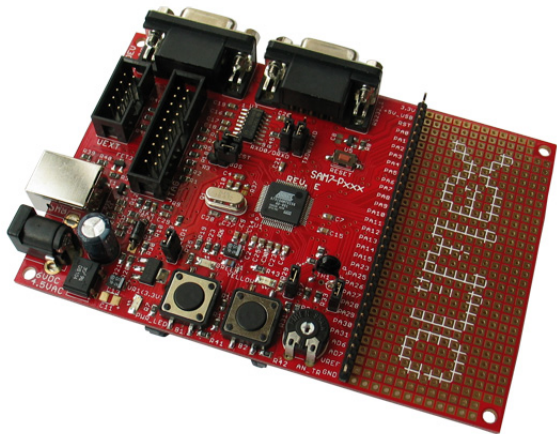


Ein beispielhafter Adapter



OpenOCD-USB von <http://www.eproo.net> (Benedikt Sauter, Augsburg)

Eine Sitzung mit OpenOCD



Olimex SAM7-P256

OpenOCD starten

(1) Aktuellen **Quelltext** mit "git clone" aus Git Repository holen:

```
git://openocd.git.sourceforge.net/gitroot/openocd/openocd
```

(2) **Konfigurieren, kompilieren und installieren:**

```
./bootstrap
```

```
./configure --prefix=/home/hhoegl/local/  
            --enable-ft2232_libftdi
```

```
make
```

```
make install
```

(3) **Wichtige installierte Dateien** kennen:

```
local/bin/openocd <-- OpenOCD Server
```

```
local/share/openocd/scripts/ <-- Konfigurationsfiles
```

```
local/share/info/openocd-info* <-- User's Manual
```

```
local/share/man/man1/openocd.1 <-- Man Page
```

OpenOCD starten (2)

- Target wählen ("Smart ARM" SAM7 von Atmel)
- JTAG Interface wählen (OpenOCD-USB von eproo.net)
- In das **Projekt-Verzeichnis** wechseln (demo1/, demo2/).

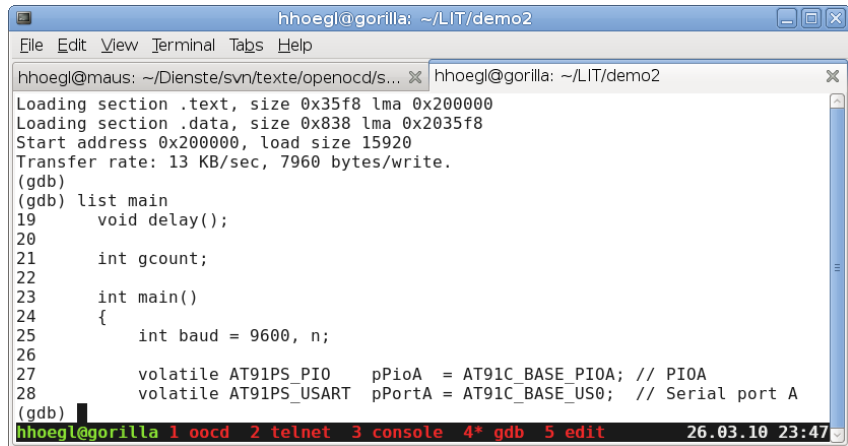
(4) OpenOCD starten

```
sudo openocd \  
-f local/share/openocd/scripts/interface/openocd-usb.cfg \  
-f local/share/openocd/scripts/target/sam7x256.cfg
```

Falls ohne Optionen gestartet, wird **openocd.cfg** gelesen.

Live Demo

Demo: <http://www.hs-augsburg.de/~hhoegl/tmp/LIT/2010/>



```
hhoegl@gorilla: ~/LIT/demo2
File Edit View Terminal Tabs Help
hhoegl@maus: ~/Dienste/svn/texte/openocd/s... x hhoegl@gorilla: ~/LIT/demo2 x
Loading section .text, size 0x35f8 lma 0x200000
Loading section .data, size 0x838 lma 0x2035f8
Start address 0x200000, load size 15920
Transfer rate: 13 KB/sec, 7960 bytes/write.
(gdb)
(gdb) list main
19     void delay();
20
21     int gcount;
22
23     int main()
24     {
25         int baud = 9600, n;
26
27         volatile AT91PS_PIO    pPioA = AT91C_BASE_PIOA; // PIOA
28         volatile AT91PS_USART  pPortA = AT91C_BASE_US0; // Serial port A
(gdb)
hhoegl@gorilla 1 oocd 2 telnet 3 console 4* gdb 5 edit 26.03.10 23:47
```

Lesestoff

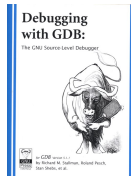
- **OpenOCD Homepage** <http://openocd.berlios.de>
- **OpenOCD User's Guide**
<http://openocd.berlios.de/doc/html/index.html>
- H. Högl, D. Rath, **Open On-Chip Debugger**, Paper für Embedded World 2006, 10 Seiten.
<http://www.hs-augsburg.de/~hhoegl/doc/openocd/ew07.pdf>
- **Development Mailing List**
<https://lists.berlios.de/mailman/listinfo/openocd-development>
- Martin Thomas, **Accessing ARM-Controllers with OpenOCD**
http://www.siwawi.arubi.uni-kl.de/avr_projects/arm_projects/openocd_intro/index.html

Lesestoff (2)

- Michael Fischer, **YAGARTO - Yet another GNU ARM Toolchain**
<http://www.yagarto.de>
- Miro Samek, **Building bare-metal ARM Systems with GNU**
<http://www.state-machine.com/resources/articles.php>
- Wikipedia Eintrag: **Joint Test Action Group**
http://de.wikipedia.org/wiki/Joint_Test_Action_Group

Bücher zum GDB

Das **GDB Manual** – siehe auch “info gdb”.



Norman Matloff, Jay Salzman, **The Art of Debugging with GDB, DDD and Eclipse**, no starch press 2008.

