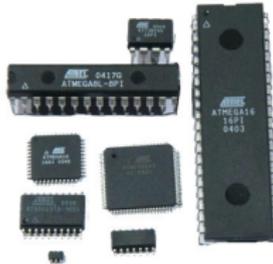


AVRDUDE, avr-libc, AVaRICE

Die Begleiter des AVR-GCC

Jörg Wunsch

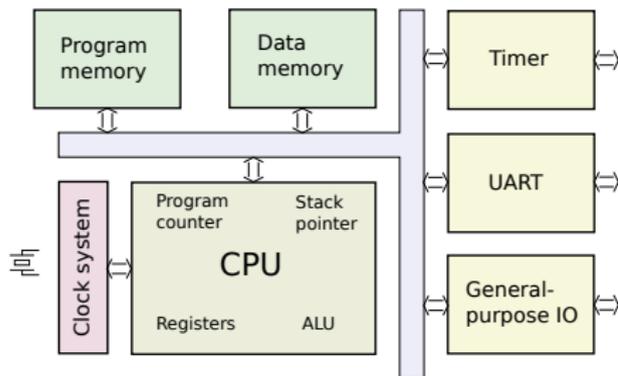
20. März 2016



Agenda

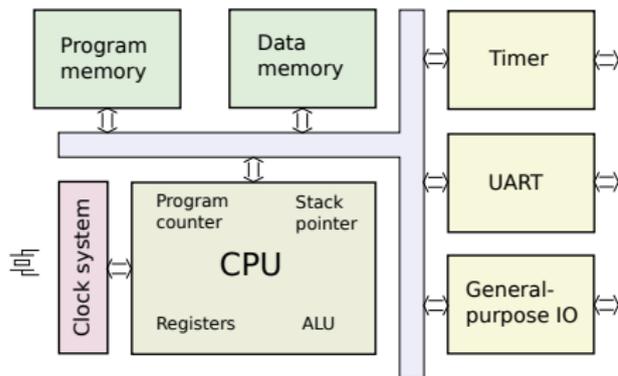
- 1 Hardware-Einführung
- 2 Controller-Programmierung
- 3 AVR-bezogene Opensource-Projekte
 - Technisches
 - Soziales

Mikrocontroller



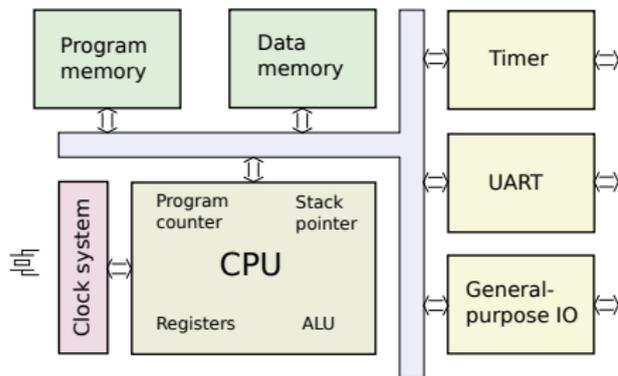
- Kompletter Computer auf einem Chip
- Neben CPU und Speicher vor allem Peripheriebausteine
- Oft für Steuerungen benutzt, daher der Namen *Controller*
- Abkürzung: MCU (*Microcontroller Unit*)

Mikrocontroller



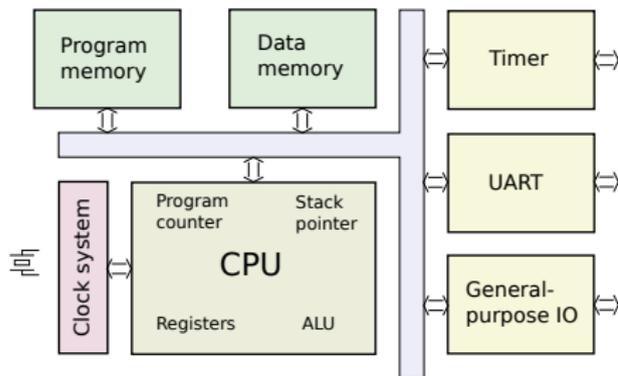
- Kompletter Computer auf einem Chip
- Neben CPU und Speicher vor allem Peripheriebausteine
- Oft für Steuerungen benutzt, daher der Namen *Controller*
- Abkürzung: MCU (*Microcontroller Unit*)

Mikrocontroller



- Kompletter Computer auf einem Chip
- Neben CPU und Speicher vor allem Peripheriebausteine
- Oft für Steuerungen benutzt, daher der Namen *Controller*
- Abkürzung: MCU (*Microcontroller Unit*)

Mikrocontroller



- Kompletter Computer auf einem Chip
- Neben CPU und Speicher vor allem Peripheriebausteine
- Oft für Steuerungen benutzt, daher der Namen *Controller*
- Abkürzung: MCU (*Microcontroller Unit*)

Geschichtliches

- Texas Instruments TMS 1000, 1971 (1974)
- Intel 8048, 1976 (MCS-48)
 - ▶ Keyboardcontroller im IBM PC-AT
- Intel 8051, 1980 (MCS-51)
 - ▶ Viele Lizenznehmer, auch lizenzfrei, „Industriestandard“
- Microchip PIC, 1976
 - ▶ *Peripheral Interface Controller*
 - ▶ Weitläufige Familie, ein Name für viele verschiedene Produkte
- Motorola 68HC05, 1977? (Freescale, NXP)
 - ▶ Controller-Variante des 6800
 - ▶ Nachfolger HC08 und HC11 sehr populär

Geschichtliches

- Texas Instruments TMS 1000, 1971 (1974)
- Intel 8048, 1976 (MCS-48)
 - ▶ Keyboardcontroller im IBM PC-AT
- Intel 8051, 1980 (MCS-51)
 - ▶ Viele Lizenznehmer, auch lizenzfrei, „Industriestandard“
- Microchip PIC, 1976
 - ▶ *Peripheral Interface Controller*
 - ▶ Weitläufige Familie, ein Name für viele verschiedene Produkte
- Motorola 68HC05, 1977? (Freescale, NXP)
 - ▶ Controller-Variante des 6800
 - ▶ Nachfolger HC08 und HC11 sehr populär

Geschichtliches

- Texas Instruments TMS 1000, 1971 (1974)
- Intel 8048, 1976 (MCS-48)
 - ▶ Keyboardcontroller im IBM PC-AT
- Intel 8051, 1980 (MCS-51)
 - ▶ Viele Lizenznehmer, auch lizenzfrei, „Industriestandard“
- Microchip PIC, 1976
 - ▶ *Peripheral Interface Controller*
 - ▶ Weitläufige Familie, ein Name für viele verschiedene Produkte
- Motorola 68HC05, 1977? (Freescale, NXP)
 - ▶ Controller-Variante des 6800
 - ▶ Nachfolger HC08 und HC11 sehr populär

Geschichtliches

- Texas Instruments TMS 1000, 1971 (1974)
- Intel 8048, 1976 (MCS-48)
 - ▶ Keyboardcontroller im IBM PC-AT
- Intel 8051, 1980 (MCS-51)
 - ▶ Viele Lizenznehmer, auch lizenzfrei, „Industriestandard“
- Microchip PIC, 1976
 - ▶ *Peripheral Interface Controller*
 - ▶ Weitläufige Familie, ein Name für viele verschiedene Produkte
- Motorola 68HC05, 1977? (Freescale, NXP)
 - ▶ Controller-Variante des 6800
 - ▶ Nachfolger HC08 und HC11 sehr populär

Geschichtliches

- Texas Instruments TMS 1000, 1971 (1974)
- Intel 8048, 1976 (MCS-48)
 - ▶ Keyboardcontroller im IBM PC-AT
- Intel 8051, 1980 (MCS-51)
 - ▶ Viele Lizenznehmer, auch lizenzfrei, „Industriestandard“
- Microchip PIC, 1976
 - ▶ *Peripheral Interface Controller*
 - ▶ Weitläufige Familie, ein Name für viele verschiedene Produkte
- Motorola 68HC05, 1977? (Freescale, NXP)
 - ▶ Controller-Variante des 6800
 - ▶ Nachfolger HC08 und HC11 sehr populär

Geschichtliches

- Texas Instruments TMS 1000, 1971 (1974)
- Intel 8048, 1976 (MCS-48)
 - ▶ Keyboardcontroller im IBM PC-AT
- Intel 8051, 1980 (MCS-51)
 - ▶ Viele Lizenznehmer, auch lizenzfrei, „Industriestandard“
- Microchip PIC, 1976
 - ▶ *Peripheral Interface Controller*
 - ▶ Weitläufige Familie, ein Name für viele verschiedene Produkte
- Motorola 68HC05, 1977? (Freescale, NXP)
 - ▶ Controller-Variante des 6800
 - ▶ Nachfolger HC08 und HC11 sehr populär

Geschichtliches

- Texas Instruments TMS 1000, 1971 (1974)
- Intel 8048, 1976 (MCS-48)
 - ▶ Keyboardcontroller im IBM PC-AT
- Intel 8051, 1980 (MCS-51)
 - ▶ Viele Lizenznehmer, auch lizenzfrei, „Industriestandard“
- Microchip PIC, 1976
 - ▶ *Peripheral Interface Controller*
 - ▶ Weitläufige Familie, ein Name für viele verschiedene Produkte
- Motorola 68HC05, 1977? (Freescale, NXP)
 - ▶ Controller-Variante des 6800
 - ▶ Nachfolger HC08 und HC11 sehr populär

Geschichtliches

- Texas Instruments TMS 1000, 1971 (1974)
- Intel 8048, 1976 (MCS-48)
 - ▶ Keyboardcontroller im IBM PC-AT
- Intel 8051, 1980 (MCS-51)
 - ▶ Viele Lizenznehmer, auch lizenzfrei, „Industriestandard“
- Microchip PIC, 1976
 - ▶ *Peripheral Interface Controller*
 - ▶ Weitläufige Familie, ein Name für viele verschiedene Produkte
- Motorola 68HC05, 1977? (Freescale, NXP)
 - ▶ Controller-Variante des 6800
 - ▶ Nachfolger HC08 und HC11 sehr populär

Geschichtliches

- Texas Instruments TMS 1000, 1971 (1974)
- Intel 8048, 1976 (MCS-48)
 - ▶ Keyboardcontroller im IBM PC-AT
- Intel 8051, 1980 (MCS-51)
 - ▶ Viele Lizenznehmer, auch lizenzfrei, „Industriestandard“
- Microchip PIC, 1976
 - ▶ *Peripheral Interface Controller*
 - ▶ Weitläufige Familie, ein Name für viele verschiedene Produkte
- Motorola 68HC05, 1977? (Freescale, NXP)
 - ▶ Controller-Variante des 6800
 - ▶ Nachfolger HC08 und HC11 sehr populär

Geschichtliches

- Texas Instruments TMS 1000, 1971 (1974)
- Intel 8048, 1976 (MCS-48)
 - ▶ Keyboardcontroller im IBM PC-AT
- Intel 8051, 1980 (MCS-51)
 - ▶ Viele Lizenznehmer, auch lizenzfrei, „Industriestandard“
- Microchip PIC, 1976
 - ▶ *Peripheral Interface Controller*
 - ▶ Weitläufige Familie, ein Name für viele verschiedene Produkte
- Motorola 68HC05, 1977? (Freescale, NXP)
 - ▶ Controller-Variante des 6800
 - ▶ Nachfolger HC08 und HC11 sehr populär

Geschichtliches

- Texas Instruments TMS 1000, 1971 (1974)
- Intel 8048, 1976 (MCS-48)
 - ▶ Keyboardcontroller im IBM PC-AT
- Intel 8051, 1980 (MCS-51)
 - ▶ Viele Lizenznehmer, auch lizenzfrei, „Industriestandard“
- Microchip PIC, 1976
 - ▶ *Peripheral Interface Controller*
 - ▶ Weitläufige Familie, ein Name für viele verschiedene Produkte
- Motorola 68HC05, 1977? (Freescale, NXP)
 - ▶ Controller-Variante des 6800
 - ▶ Nachfolger HC08 und HC11 sehr populär

Modernisierung

- **Programmspeicher früher:**

- ▶ Masken-ROMs (preiswert für große Stückzahlen)
- ▶ interne EPROMs mit Löschenfenster (flexibel, sehr teuer)
- ▶ interne (E)EPROMs ohne Löschmöglichkeit (OTP, *One-Time Programmable*; preiswert für kleine Stückzahlen)
- ▶ externe EPROMs (kein „echter“ Controller mehr, viele Pins, seltener)

- ⇒ breite Produktpalette nötig

- Anfang der 1990er Jahre: EEPROM — neue Möglichkeiten

- ⇒ Integration dieser neuen Technologie in Controllern

- 1992: Dissertation von Alf-Egil Bogen und Vegard Wollan an der NTH (jetzt NTNU) Trondheim, Norwegen

- Initial als IP-Core und ASIC, Nordic VLSI

- 1997: erster AVR bei Atmel verkaufsfähig: AT90S1200

- einfache Programmierung: ISP, *In-System Programming*

Modernisierung

- Programmspeicher früher:
 - ▶ Masken-ROMs (preiswert für große Stückzahlen)
 - ▶ interne EPROMs mit Löschenfenster (flexibel, sehr teuer)
 - ▶ interne (E)EPROMs ohne Löschmöglichkeit (OTP, *One-Time Programmable*; preiswert für kleine Stückzahlen)
 - ▶ externe EPROMs (kein „echter“ Controller mehr, viele Pins, seltener)
- ⇒ breite Produktpalette nötig
- Anfang der 1990er Jahre: EEPROM — neue Möglichkeiten
- ⇒ Integration dieser neuen Technologie in Controllern
- 1992: Dissertation von Alf-Egil Bogen und Vegard Wollan an der NTH (jetzt NTNU) Trondheim, Norwegen
- Initial als IP-Core und ASIC, Nordic VLSI
- 1997: erster AVR bei Atmel verkaufsfähig: AT90S1200
- einfache Programmierung: ISP, *In-System Programming*

Modernisierung

- Programmspeicher früher:
 - ▶ Masken-ROMs (preiswert für große Stückzahlen)
 - ▶ interne EPROMs mit Löschenfenster (flexibel, sehr teuer)
 - ▶ interne (E)EPROMs ohne Löschmöglichkeit (OTP, *One-Time Programmable*; preiswert für kleine Stückzahlen)
 - ▶ externe EPROMs (kein „echter“ Controller mehr, viele Pins, seltener)
- ⇒ breite Produktpalette nötig
- Anfang der 1990er Jahre: EEPROM — neue Möglichkeiten
- ⇒ Integration dieser neuen Technologie in Controllern
- 1992: Dissertation von Alf-Egil Bogen und Vegard Wollan an der NTH (jetzt NTNU) Trondheim, Norwegen
- Initial als IP-Core und ASIC, Nordic VLSI
- 1997: erster AVR bei Atmel verkaufsfähig: AT90S1200
- einfache Programmierung: ISP, *In-System Programming*

Modernisierung

- Programmspeicher früher:
 - ▶ Masken-ROMs (preiswert für große Stückzahlen)
 - ▶ interne EPROMs mit Löschenfenster (flexibel, sehr teuer)
 - ▶ interne (E)EPROMs ohne Löschmöglichkeit (OTP, *One-Time Programmable*; preiswert für kleine Stückzahlen)
 - ▶ externe EPROMs (kein „echter“ Controller mehr, viele Pins, seltener)
- ⇒ breite Produktpalette nötig
- Anfang der 1990er Jahre: EEPROM — neue Möglichkeiten
- ⇒ Integration dieser neuen Technologie in Controllern
- 1992: Dissertation von Alf-Egil Bogen und Vegard Wollan an der NTH (jetzt NTNU) Trondheim, Norwegen
- Initial als IP-Core und ASIC, Nordic VLSI
- 1997: erster AVR bei Atmel verkaufsfähig: AT90S1200
- einfache Programmierung: ISP, *In-System Programming*

Modernisierung

- Programmspeicher früher:
 - ▶ Masken-ROMs (preiswert für große Stückzahlen)
 - ▶ interne EPROMs mit Löschenfenster (flexibel, sehr teuer)
 - ▶ interne (E)EPROMs ohne Löschmöglichkeit (OTP, *One-Time Programmable*; preiswert für kleine Stückzahlen)
 - ▶ externe EPROMs (kein „echter“ Controller mehr, viele Pins, seltener)
- ⇒ breite Produktpalette nötig
- Anfang der 1990er Jahre: EEPROM — neue Möglichkeiten
- ⇒ Integration dieser neuen Technologie in Controllern
- 1992: Dissertation von Alf-Egil Bogen und Vegard Wollan an der NTH (jetzt NTNU) Trondheim, Norwegen
- Initial als IP-Core und ASIC, Nordic VLSI
- 1997: erster AVR bei Atmel verkaufsfähig: AT90S1200
- einfache Programmierung: ISP, *In-System Programming*

Modernisierung

- Programmspeicher früher:
 - ▶ Masken-ROMs (preiswert für große Stückzahlen)
 - ▶ interne EPROMs mit Löschenfenster (flexibel, sehr teuer)
 - ▶ interne (E)EPROMs ohne Löschmöglichkeit (OTP, *One-Time Programmable*; preiswert für kleine Stückzahlen)
 - ▶ externe EPROMs (kein „echter“ Controller mehr, viele Pins, seltener)
- ⇒ breite Produktpalette nötig
- Anfang der 1990er Jahre: EEPROM — neue Möglichkeiten
- ⇒ Integration dieser neuen Technologie in Controllern
- 1992: Dissertation von Alf-Egil Bogen und Vegard Wollan an der NTH (jetzt NTNU) Trondheim, Norwegen
- Initial als IP-Core und ASIC, Nordic VLSI
- 1997: erster AVR bei Atmel verkaufsfähig: AT90S1200
- einfache Programmierung: ISP, *In-System Programming*

Modernisierung

- Programmspeicher früher:
 - ▶ Masken-ROMs (preiswert für große Stückzahlen)
 - ▶ interne EPROMs mit Löschenfenster (flexibel, sehr teuer)
 - ▶ interne (E)PROMs ohne Löschmöglichkeit (OTP, *One-Time Programmable*; preiswert für kleine Stückzahlen)
 - ▶ externe EPROMs (kein „echter“ Controller mehr, viele Pins, seltener)
- ⇒ breite Produktpalette nötig
- Anfang der 1990er Jahre: EEPROM — neue Möglichkeiten
- ⇒ Integration dieser neuen Technologie in Controllern
- 1992: Dissertation von Alf-Egil Bogen und Vegard Wollan an der NTH (jetzt NTNU) Trondheim, Norwegen
- Initial als IP-Core und ASIC, Nordic VLSI
- 1997: erster AVR bei Atmel verkaufsfähig: AT90S1200
- einfache Programmierung: ISP, *In-System Programming*

Modernisierung

- Programmspeicher früher:
 - ▶ Masken-ROMs (preiswert für große Stückzahlen)
 - ▶ interne EPROMs mit Löschenfenster (flexibel, sehr teuer)
 - ▶ interne (E)EPROMs ohne Löschmöglichkeit (OTP, *One-Time Programmable*; preiswert für kleine Stückzahlen)
 - ▶ externe EPROMs (kein „echter“ Controller mehr, viele Pins, seltener)
- ⇒ breite Produktpalette nötig
- Anfang der 1990er Jahre: EEPROM — neue Möglichkeiten
- ⇒ Integration dieser neuen Technologie in Controllerr
- 1992: Dissertation von Alf-Egil Bogen und Vegard Wollan an der NTH (jetzt NTNU) Trondheim, Norwegen
- Initial als IP-Core und ASIC, Nordic VLSI
- 1997: erster AVR bei Atmel verkaufsfähig: AT90S1200
- einfache Programmierung: ISP, *In-System Programming*

Modernisierung

- Programmspeicher früher:
 - ▶ Masken-ROMs (preiswert für große Stückzahlen)
 - ▶ interne EPROMs mit Löschenfenster (flexibel, sehr teuer)
 - ▶ interne (E)PROMs ohne Löschmöglichkeit (OTP, *One-Time Programmable*; preiswert für kleine Stückzahlen)
 - ▶ externe EPROMs (kein „echter“ Controller mehr, viele Pins, seltener)
- ⇒ breite Produktpalette nötig
- Anfang der 1990er Jahre: EEPROM — neue Möglichkeiten
- ⇒ Integration dieser neuen Technologie in Controllerr
- 1992: Dissertation von Alf-Egil Bogen und Vegard Wollan an der NTH (jetzt NTNU) Trondheim, Norwegen
- Initial als IP-Core und ASIC, Nordic VLSI
- 1997: erster AVR bei Atmel verkaufsfähig: AT90S1200
- einfache Programmierung: ISP, *In-System Programming*

Modernisierung

- Programmspeicher früher:
 - ▶ Masken-ROMs (preiswert für große Stückzahlen)
 - ▶ interne EPROMs mit Löschenfenster (flexibel, sehr teuer)
 - ▶ interne (E)EPROMs ohne Löschmöglichkeit (OTP, *One-Time Programmable*; preiswert für kleine Stückzahlen)
 - ▶ externe EPROMs (kein „echter“ Controller mehr, viele Pins, seltener)
- ⇒ breite Produktpalette nötig
- Anfang der 1990er Jahre: EEPROM — neue Möglichkeiten
- ⇒ Integration dieser neuen Technologie in Controllerr
- 1992: Dissertation von Alf-Egil Bogen und Vegard Wollan an der NTH (jetzt NTNU) Trondheim, Norwegen
- Initial als IP-Core und ASIC, Nordic VLSI
- 1997: erster AVR bei Atmel verkaufsfähig: AT90S1200
- einfache Programmierung: ISP, *In-System Programming*

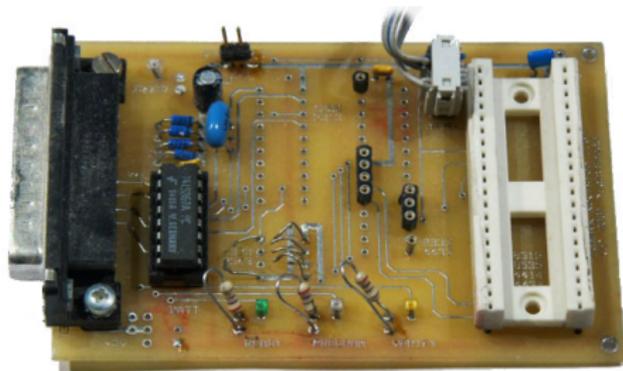
Modernisierung

- Programmspeicher früher:
 - ▶ Masken-ROMs (preiswert für große Stückzahlen)
 - ▶ interne EPROMs mit Löschenfenster (flexibel, sehr teuer)
 - ▶ interne (E)EPROMs ohne Löschmöglichkeit (OTP, *One-Time Programmable*; preiswert für kleine Stückzahlen)
 - ▶ externe EPROMs (kein „echter“ Controller mehr, viele Pins, seltener)
- ⇒ breite Produktpalette nötig
- Anfang der 1990er Jahre: EEPROM — neue Möglichkeiten
- ⇒ Integration dieser neuen Technologie in Controllerr
- 1992: Dissertation von Alf-Egil Bogen und Vegard Wollan an der NTH (jetzt NTNU) Trondheim, Norwegen
- Initial als IP-Core und ASIC, Nordic VLSI
- 1997: erster AVR bei Atmel verkaufsfähig: AT90S1200
- einfache Programmierung: ISP, *In-System Programming*

Modernisierung

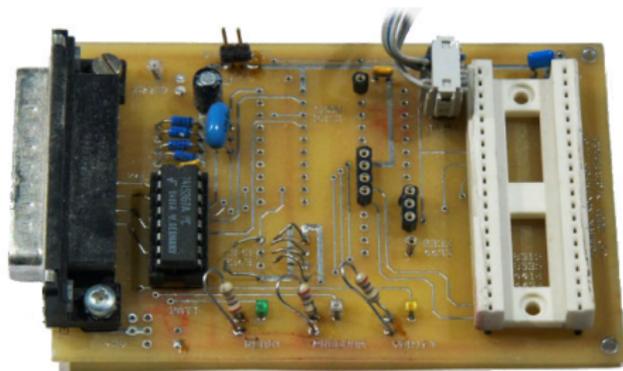
- Programmspeicher früher:
 - ▶ Masken-ROMs (preiswert für große Stückzahlen)
 - ▶ interne EPROMs mit Löschenfenster (flexibel, sehr teuer)
 - ▶ interne (E)EPROMs ohne Löschmöglichkeit (OTP, *One-Time Programmable*; preiswert für kleine Stückzahlen)
 - ▶ externe EPROMs (kein „echter“ Controller mehr, viele Pins, seltener)
- ⇒ breite Produktpalette nötig
- Anfang der 1990er Jahre: EEPROM — neue Möglichkeiten
- ⇒ Integration dieser neuen Technologie in Controllerr
- 1992: Dissertation von Alf-Egil Bogen und Vegard Wollan an der NTH (jetzt NTNU) Trondheim, Norwegen
- Initial als IP-Core und ASIC, Nordic VLSI
- 1997: erster AVR bei Atmel verkaufsfähig: AT90S1200
- einfache Programmierung: ISP, *In-System Programming*

Revolution der Controller



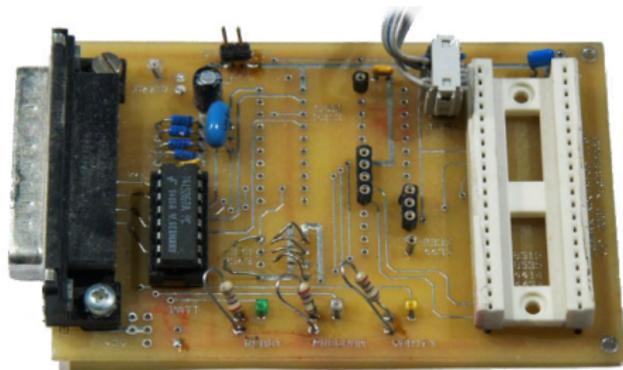
- Flash-Speicher: einer für alle
- ISP: einfach, preiswerte Tools
- Kostenlose
Entwicklungsumgebung
- Schnell: RISC, viele Ein- und
Zwei-Zyklenbefehle

Revolution der Controller



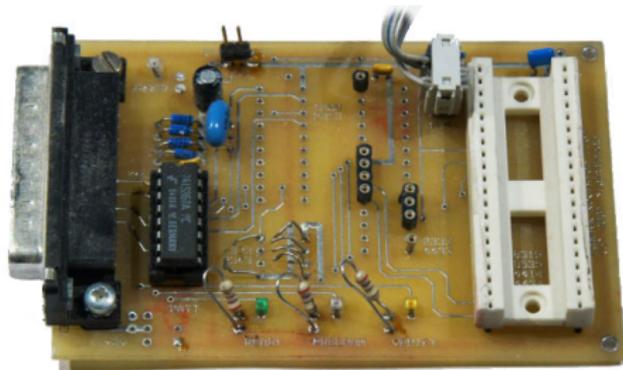
- **Flash-Speicher: einer für alle**
- ISP: einfache, preiswerte Tools
- Kostenlose Entwicklungsumgebung
- Schnell: RISC, viele Ein- und Zwei-Zyklusbefehle

Revolution der Controller



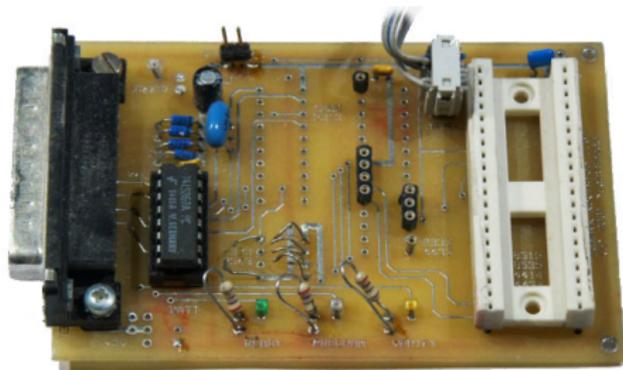
- Flash-Speicher: einer für alle
- ISP: einfach, preiswerte Tools
- Kostenlose
Entwicklungsumgebung
- Schnell: RISC, viele Ein- und
Zwei-Zyklenbefehle

Revolution der Controller



- Flash-Speicher: einer für alle
- ISP: einfach, preiswerte Tools
- Kostenlose Entwicklungsumgebung
- Schnell: RISC, viele Ein- und Zwei-Zyklenbefehle

Revolution der Controller



- Flash-Speicher: einer für alle
- ISP: einfach, preiswerte Tools
- Kostenlose
Entwicklungsumgebung
- Schnell: RISC, viele Ein- und
Zwei-Zyklenbefehle

Toolchains im Vergleich

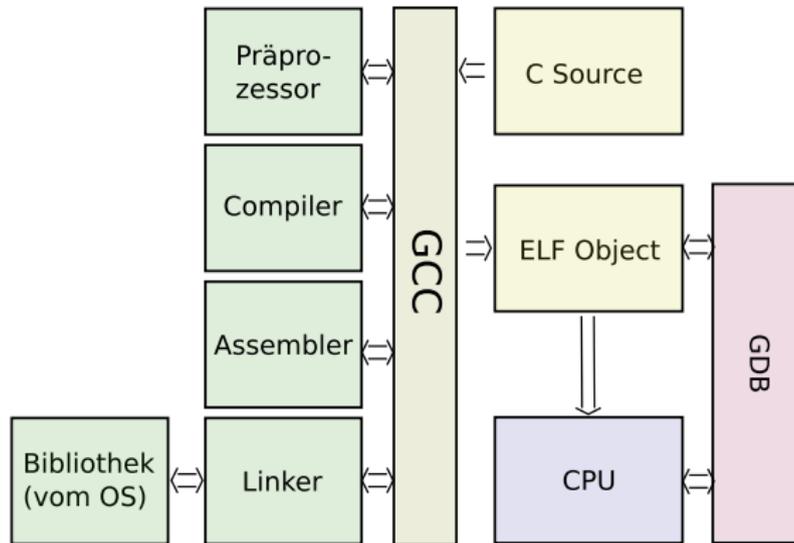


Abbildung : Host-Toolchain (native)

Toolchains im Vergleich

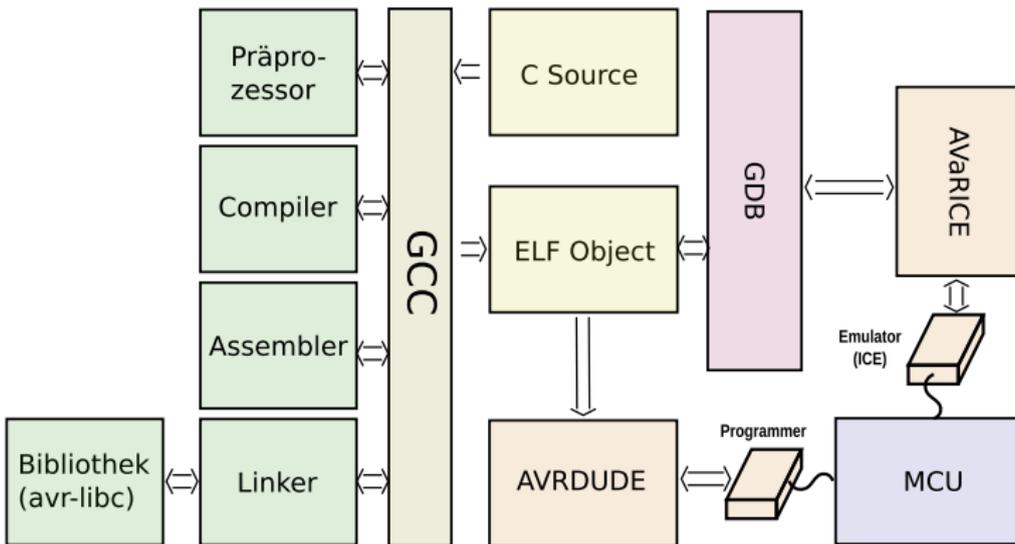


Abbildung : Cross-Toolchain für MCU

Toolchains im Vergleich

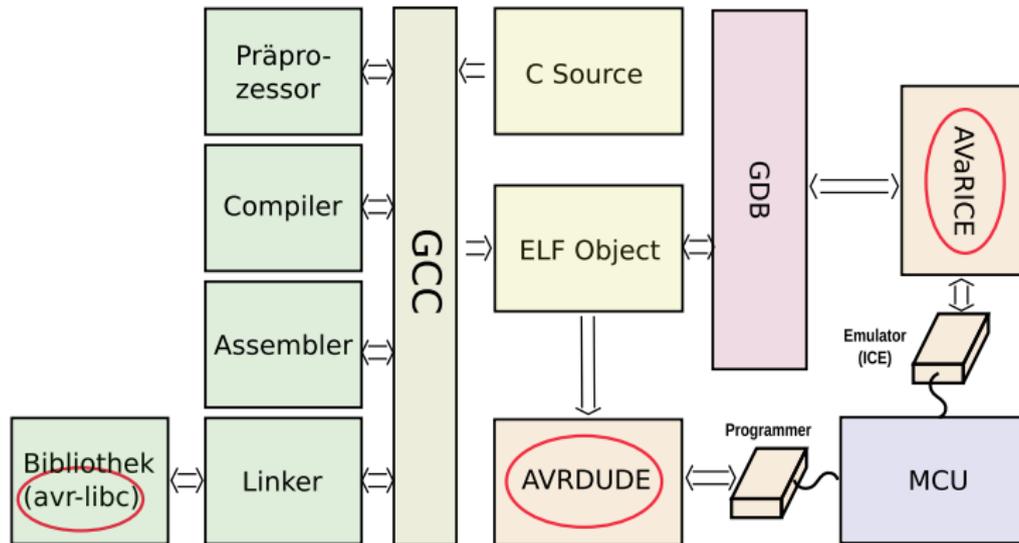


Abbildung : Cross-Toolchain für MCU

Cross-Toolchain

- Compiler übersetzt Code für die Zielplattform statt die eigene CPU
- Assembler benutzt Mnemonics der Ziel-CPU
- Linker und Debugger benutzen ggf. anderes Objektformat
- GNU-Tools werden aus gleichem Sourcecode gebaut; die Unterscheidung erfolgt beim `configure`:

```
configure ... --target=avr
```
- Typisch wird den Tools ein Präfix vorangestellt:
`avr-gcc`, `avr-as`, `avr-ld`, `avr-gdb`
- GCC-Portierung für AVR erfolgte im Jahr 1999/2000 durch Денис Чертыков (Denis Chertykov)
- Initial für `AVA`-Assembler/Linker von Uros Platise
- Wenig später auch AVR-Support in GNU `binutils` (Assembler, Linker)

Cross-Toolchain

- Compiler übersetzt Code für die Zielplattform statt die eigene CPU
- Assembler benutzt Mnemonics der Ziel-CPU
- Linker und Debugger benutzen ggf. anderes Objektformat
- GNU-Tools werden aus gleichem Sourcecode gebaut; die Unterscheidung erfolgt beim `configure`:

```
configure ... --target=avr
```
- Typisch wird den Tools ein Präfix vorangestellt:
`avr-gcc`, `avr-as`, `avr-ld`, `avr-gdb`
- GCC-Portierung für AVR erfolgte im Jahr 1999/2000 durch Денис Чертыков (Denis Chertykov)
- Initial für `AVA`-Assembler/Linker von Uros Platise
- Wenig später auch AVR-Support in GNU `binutils` (Assembler, Linker)

Cross-Toolchain

- Compiler übersetzt Code für die Zielplattform statt die eigene CPU
- Assembler benutzt Mnemonics der Ziel-CPU
- Linker und Debugger benutzen ggf. anderes Objektformat
- GNU-Tools werden aus gleichem Sourcecode gebaut; die Unterscheidung erfolgt beim `configure`:
`configure ... --target=avr`
- Typisch wird den Tools ein Präfix vorangestellt:
`avr-gcc, avr-as, avr-ld, avr-gdb`
- GCC-Portierung für AVR erfolgte im Jahr 1999/2000 durch Денис Чертыков (Denis Chertykov)
- Initial für `AVA`-Assembler/Linker von Uros Platise
- Wenig später auch AVR-Support in GNU `binutils` (Assembler, Linker)

Cross-Toolchain

- Compiler übersetzt Code für die Zielplattform statt die eigene CPU
- Assembler benutzt Mnemonics der Ziel-CPU
- Linker und Debugger benutzen ggf. anderes Objektformat
- GNU-Tools werden aus gleichem Sourcecode gebaut; die Unterscheidung erfolgt beim `configure`:

```
configure ... --target=avr
```

- Typisch wird den Tools ein Präfix vorangestellt:
`avr-gcc`, `avr-as`, `avr-ld`, `avr-gdb`
- GCC-Portierung für AVR erfolgte im Jahr 1999/2000 durch Денис Чертыков (Denis Chertykov)
- Initial für `AVA`-Assembler/Linker von Uros Platise
- Wenig später auch AVR-Support in GNU `binutils` (Assembler, Linker)

Cross-Toolchain

- Compiler übersetzt Code für die Zielplattform statt die eigene CPU
- Assembler benutzt Mnemonics der Ziel-CPU
- Linker und Debugger benutzen ggf. anderes Objektformat
- GNU-Tools werden aus gleichem Sourcecode gebaut; die Unterscheidung erfolgt beim `configure`:

```
configure ... --target=avr
```

- Typisch wird den Tools ein Präfix vorangestellt:
`avr-gcc`, `avr-as`, `avr-ld`, `avr-gdb`
- GCC-Portierung für AVR erfolgte im Jahr 1999/2000 durch Денис Чертыков (Denis Chertykov)
- Initial für `AVA`-Assembler/Linker von Uros Platise
- Wenig später auch AVR-Support in GNU `binutils` (Assembler, Linker)

Cross-Toolchain

- Compiler übersetzt Code für die Zielplattform statt die eigene CPU
- Assembler benutzt Mnemonics der Ziel-CPU
- Linker und Debugger benutzen ggf. anderes Objektformat
- GNU-Tools werden aus gleichem Sourcecode gebaut; die Unterscheidung erfolgt beim `configure`:

```
configure ... --target=avr
```

- Typisch wird den Tools ein Präfix vorangestellt:
`avr-gcc`, `avr-as`, `avr-ld`, `avr-gdb`
- GCC-Portierung für AVR erfolgte im Jahr 1999/2000 durch Денис Чертыков (Denis Chertykov)
- Initial für `AVA`-Assembler/Linker von Uros Platise
- Wenig später auch AVR-Support in GNU `binutils` (Assembler, Linker)

Cross-Toolchain

- Compiler übersetzt Code für die Zielplattform statt die eigene CPU
- Assembler benutzt Mnemonics der Ziel-CPU
- Linker und Debugger benutzen ggf. anderes Objektformat
- GNU-Tools werden aus gleichem Sourcecode gebaut; die Unterscheidung erfolgt beim `configure`:

```
configure ... --target=avr
```

- Typisch wird den Tools ein Präfix vorangestellt:
`avr-gcc`, `avr-as`, `avr-ld`, `avr-gdb`
- GCC-Portierung für AVR erfolgte im Jahr 1999/2000 durch Денис Чертыков (Denis Chertykov)
- Initial für `AVA-Assembler/Linker` von Uros Platise
- Wenig später auch AVR-Support in GNU `binutils` (Assembler, Linker)

Cross-Toolchain

- Compiler übersetzt Code für die Zielplattform statt die eigene CPU
- Assembler benutzt Mnemonics der Ziel-CPU
- Linker und Debugger benutzen ggf. anderes Objektformat
- GNU-Tools werden aus gleichem Sourcecode gebaut; die Unterscheidung erfolgt beim `configure`:

```
configure ... --target=avr
```

- Typisch wird den Tools ein Präfix vorangestellt:
`avr-gcc`, `avr-as`, `avr-ld`, `avr-gdb`
- GCC-Portierung für AVR erfolgte im Jahr 1999/2000 durch Денис Чертыков (Denis Chertykov)
- Initial für `AVA`-Assembler/Linker von Uros Platise
- Wenig später auch AVR-Support in GNU `binutils` (Assembler, Linker)

avr-libc: Systembibliothek „light“

- C-Bibliothek ist inhärenter Bestandteil des Sprachstandards
- Existierende Implementierungen nicht adäquat für 8-Bit-Controller-Umfeld
- Enthält neben den Bibliotheken auch Startup-Code `gcr1.S`
- Viele Teile in Assembler, viel in Headerdateien
- Aufgrund der Harvard-Architektur separater Satz von Funktionen nötig für Daten (Konstanten), die im Flash liegen
- Kein vollständiges C99 (bspw. kein `wchar_t`), aber viele Erweiterungen
- Doxygen-Dokumentation mit derzeit 340 Seiten, die auch weitergehende Erläuterungen zur gesamten Toolchain enthält

avr-libc: Systembibliothek „light“

- C-Bibliothek ist inhärenter Bestandteil des Sprachstandards
- Existierende Implementierungen nicht adäquat für 8-Bit-Controller-Umfeld
- Enthält neben den Bibliotheken auch Startup-Code `gcr1.S`
- Viele Teile in Assembler, viel in Headerdateien
- Aufgrund der Harvard-Architektur separater Satz von Funktionen nötig für Daten (Konstanten), die im Flash liegen
- Kein vollständiges C99 (bspw. kein `wchar_t`), aber viele Erweiterungen
- Doxygen-Dokumentation mit derzeit 340 Seiten, die auch weitergehende Erläuterungen zur gesamten Toolchain enthält

avr-libc: Systembibliothek „light“

- C-Bibliothek ist inhärenter Bestandteil des Sprachstandards
- Existierende Implementierungen nicht adäquat für 8-Bit-Controller-Umfeld
- Enthält neben den Bibliotheken auch Startup-Code `gcrt1.S`
- Viele Teile in Assembler, viel in Headerdateien
- Aufgrund der Harvard-Architektur separater Satz von Funktionen nötig für Daten (Konstanten), die im Flash liegen
- Kein vollständiges C99 (bspw. kein `wchar_t`), aber viele Erweiterungen
- Doxygen-Dokumentation mit derzeit 340 Seiten, die auch weitergehende Erläuterungen zur gesamten Toolchain enthält

avr-libc: Systembibliothek „light“

- C-Bibliothek ist inhärenter Bestandteil des Sprachstandards
- Existierende Implementierungen nicht adäquat für 8-Bit-Controller-Umfeld
- Enthält neben den Bibliotheken auch Startup-Code `gcrt1.S`
- Viele Teile in Assembler, viel in Headerdateien
- Aufgrund der Harvard-Architektur separater Satz von Funktionen nötig für Daten (Konstanten), die im Flash liegen
- Kein vollständiges C99 (bspw. kein `wchar_t`), aber viele Erweiterungen
- Doxygen-Dokumentation mit derzeit 340 Seiten, die auch weitergehende Erläuterungen zur gesamten Toolchain enthält

avr-libc: Systembibliothek „light“

- C-Bibliothek ist inhärenter Bestandteil des Sprachstandards
- Existierende Implementierungen nicht adäquat für 8-Bit-Controller-Umfeld
- Enthält neben den Bibliotheken auch Startup-Code `gcrt1.S`
- Viele Teile in Assembler, viel in Headerdateien
- Aufgrund der Harvard-Architektur separater Satz von Funktionen nötig für Daten (Konstanten), die im Flash liegen
- Kein vollständiges C99 (bspw. kein `wchar_t`), aber viele Erweiterungen
- Doxygen-Dokumentation mit derzeit 340 Seiten, die auch weitergehende Erläuterungen zur gesamten Toolchain enthält

avr-libc: Systembibliothek „light“

- C-Bibliothek ist inhärenter Bestandteil des Sprachstandards
- Existierende Implementierungen nicht adäquat für 8-Bit-Controller-Umfeld
- Enthält neben den Bibliotheken auch Startup-Code `gcrt1.S`
- Viele Teile in Assembler, viel in Headerdateien
- Aufgrund der Harvard-Architektur separater Satz von Funktionen nötig für Daten (Konstanten), die im Flash liegen
- Kein vollständiges C99 (bspw. kein `wchar_t`), aber viele Erweiterungen
- Doxygen-Dokumentation mit derzeit 340 Seiten, die auch weitergehende Erläuterungen zur gesamten Toolchain enthält

avr-libc: Systembibliothek „light“

- C-Bibliothek ist inhärenter Bestandteil des Sprachstandards
- Existierende Implementierungen nicht adäquat für 8-Bit-Controller-Umfeld
- Enthält neben den Bibliotheken auch Startup-Code `gcrt1.S`
- Viele Teile in Assembler, viel in Headerdateien
- Aufgrund der Harvard-Architektur separater Satz von Funktionen nötig für Daten (Konstanten), die im Flash liegen
- Kein vollständiges C99 (bspw. kein `wchar_t`), aber viele Erweiterungen
- Doxygen-Dokumentation mit derzeit 340 Seiten, die auch weitergehende Erläuterungen zur gesamten Toolchain enthält

avr-libc: Details

- Hardware-Zugriff einheitlich als `#include <avr/io.h>`
- Auswahl gesteuert über Compileroption `-mmcu=mcu_type`
- “Hello, world!” auf einem Controller: blinkende LED

```
● #include <avr/io.h>
   #define F_CPU 1E6
   #include <util/delay.h>
```

```
int
main(void)
{
    DDRB = (1 << 1); // PB1 als Ausgang
    for (;;) {
        PORTB |= (1 << 1);
        _delay_ms(100);
        PORTB ^= ~(1 << 1);
        _delay_ms(100);
    }
    return 0; // not reached
}
```

- Compilieren mit:

```
avr-gcc -mmcu=atmega8 -g -Os -o hw.elf hw.c
```

- ⇒ Wie kommt der Code in den Controller?

avr-libc: Details

- Hardware-Zugriff einheitlich als `#include <avr/io.h>`
- Auswahl gesteuert über Compileroption `-mmcu=mcu_type`
- “Hello, world!” auf einem Controller: blinkende LED

```
● #include <avr/io.h>
  #define F_CPU 1E6
  #include <util/delay.h>

int
main(void)
{
  DDRB = (1 << 1); // PB1 als Ausgang
  for (;;) {
    PORTB |= (1 << 1);
    _delay_ms(100);
    PORTB ^= ~(1 << 1);
    _delay_ms(100);
  }
  return 0; // not reached
}
```

- Compilieren mit:
`avr-gcc -mmcu=atmega8 -g -Os -o hw.elf hw.c`
- ⇒ Wie kommt der Code in den Controller?

avr-libc: Details

- Hardware-Zugriff einheitlich als `#include <avr/io.h>`
- Auswahl gesteuert über Compileroption `-mmcu=mcu_type`
- “Hello, world!” auf einem Controller: blinkende LED

```
#include <avr/io.h>
#define F_CPU 1E6
#include <util/delay.h>

int
main(void)
{
    DDRB = (1 << 1); // PB1 als Ausgang
    for (;;) {
        PORTB |= (1 << 1);
        _delay_ms(100);
        PORTB &= ~(1 << 1);
        _delay_ms(100);
    }
    return 0; // not reached
}
```

- Compilieren mit:

```
avr-gcc -mmcu=atmega8 -g -Os -o hw.elf hw.c
```

- ⇒ Wie kommt der Code in den Controller?

avr-libc: Details

- Hardware-Zugriff einheitlich als `#include <avr/io.h>`
- Auswahl gesteuert über Compileroption `-mmcu=mcu_type`
- “Hello, world!” auf einem Controller: blinkende LED

```
● #include <avr/io.h>
  #define F_CPU 1E6
  #include <util/delay.h>

int
main(void)
{
    DDRB = (1 << 1); // PB1 als Ausgang
    for (;;) {
        PORTB |= (1 << 1);
        _delay_ms(100);
        PORTB &= ~(1 << 1);
        _delay_ms(100);
    }
    return 0; // not reached
}
```

- Compilieren mit:

```
avr-gcc -mmcu=atmega8 -g -Os -o hw.elf hw.c
```

- ⇒ Wie kommt der Code in den Controller?

avr-libc: Details

- Hardware-Zugriff einheitlich als `#include <avr/io.h>`
- Auswahl gesteuert über Compileroption `-mmcu=mcu_type`
- “Hello, world!” auf einem Controller: blinkende LED

```
● #include <avr/io.h>
  #define F_CPU 1E6
  #include <util/delay.h>

int
main(void)
{
    DDRB = (1 << 1); // PB1 als Ausgang
    for (;;) {
        PORTB |= (1 << 1);
        _delay_ms(100);
        PORTB &= ~(1 << 1);
        _delay_ms(100);
    }
    return 0; // not reached
}
```

- Compilieren mit:

```
avr-gcc -mmcu=atmega8 -g -Os -o hw.elf hw.c
```

- ⇒ Wie kommt der Code in den Controller?

avr-libc: Details

- Hardware-Zugriff einheitlich als `#include <avr/io.h>`
- Auswahl gesteuert über Compileroption `-mmcu=mcu_type`
- “Hello, world!” auf einem Controller: blinkende LED

```
● #include <avr/io.h>
  #define F_CPU 1E6
  #include <util/delay.h>

int
main(void)
{
    DDRB = (1 << 1); // PB1 als Ausgang
    for (;;) {
        PORTB |= (1 << 1);
        _delay_ms(100);
        PORTB &= ~(1 << 1);
        _delay_ms(100);
    }
    return 0; // not reached
}
```

- Compilieren mit:

```
avr-gcc -mmcu=atmega8 -g -Os -o hw.elf hw.c
```

- ⇒ Wie kommt der Code in den Controller?

AVRDUDE: „Schweizer Taschenmesser“

- Gestartet im Jahr 2000 durch Brian S. Dean als privates Projekt `avrprog`, FreeBSD
- Initial für preiswerte Parallelport-Programmer
- Idee: Daten über Programmer und MCUs in einer Textdatei `avrdude.conf` hinterlegen
- Sukzessiv um andere Programmertypen erweitert, beginnend mit Unterstützung für Atmels STK500 (2002)
- 2003 Umbenennung in `AVRDUDE`, Hosting auf `savannah.nongnu.org`, Umlizensierung auf GPL
- 2003 Linux, 2004 Win32, OSX, 2005 Solaris
- Mehr als 150 AVR-Typen, ca. 100 Programmer



AVRDUDE: „Schweizer Taschenmesser“

- Gestartet im Jahr 2000 durch Brian S. Dean als privates Projekt `avrprog`, FreeBSD
- Initial für preiswerte Parallelport-Programmer
- Idee: Daten über Programmer und MCUs in einer Textdatei `avrdude.conf` hinterlegen
- Sukzessiv um andere Programmertypen erweitert, beginnend mit Unterstützung für Atmels STK500 (2002)
- 2003 Umbenennung in `AVRDUDE`, Hosting auf `savannah.nongnu.org`, Umlizensierung auf GPL
- 2003 Linux, 2004 Win32, OSX, 2005 Solaris
- Mehr als 150 AVR-Typen, ca. 100 Programmer



AVRDUDE: „Schweizer Taschenmesser“

- Gestartet im Jahr 2000 durch Brian S. Dean als privates Projekt `avrprog`, FreeBSD
- Initial für preiswerte Parallelport-Programmer
- Idee: Daten über Programmer und MCUs in einer Textdatei `avrdude.conf` hinterlegen
- Sukzessiv um andere Programmertypen erweitert, beginnend mit Unterstützung für Atmels STK500 (2002)
- 2003 Umbenennung in AVRDUDE, Hosting auf `savannah.nongnu.org`, Umlizensierung auf GPL
- 2003 Linux, 2004 Win32, OSX, 2005 Solaris
- Mehr als 150 AVR-Typen, ca. 100 Programmer



AVRDUDE: „Schweizer Taschenmesser“

- Gestartet im Jahr 2000 durch Brian S. Dean als privates Projekt `avrprog`, FreeBSD
- Initial für preiswerte Parallelport-Programmer
- Idee: Daten über Programmer und MCUs in einer Textdatei `avrdude.conf` hinterlegen
- Sukzessiv um andere Programmertypen erweitert, beginnend mit Unterstützung für Atmels STK500 (2002)
- 2003 Umbenennung in AVRDUDE, Hosting auf `savannah.nongnu.org`, Umlizensierung auf GPL
- 2003 Linux, 2004 Win32, OSX, 2005 Solaris
- Mehr als 150 AVR-Typen, ca. 100 Programmer



AVRDUDE: „Schweizer Taschenmesser“

- Gestartet im Jahr 2000 durch Brian S. Dean als privates Projekt `avrprog`, FreeBSD
- Initial für preiswerte Parallelport-Programmer
- Idee: Daten über Programmer und MCUs in einer Textdatei `avrdude.conf` hinterlegen
- Sukzessiv um andere Programmertypen erweitert, beginnend mit Unterstützung für Atmels STK500 (2002)
- 2003 Umbenennung in `AVRDUDE`, Hosting auf `savannah.nongnu.org`, Umlizensierung auf GPL
- 2003 Linux, 2004 Win32, OSX, 2005 Solaris
- Mehr als 150 AVR-Typen, ca. 100 Programmer



AVRDUDE: „Schweizer Taschenmesser“

- Gestartet im Jahr 2000 durch Brian S. Dean als privates Projekt `avrprog`, FreeBSD
- Initial für preiswerte Parallelport-Programmer
- Idee: Daten über Programmer und MCUs in einer Textdatei `avrdude.conf` hinterlegen
- Sukzessiv um andere Programmertypen erweitert, beginnend mit Unterstützung für Atmels STK500 (2002)
- 2003 Umbenennung in `AVRDUDE`, Hosting auf `savannah.nongnu.org`, Umlizensierung auf GPL
- 2003 Linux, 2004 Win32, OSX, 2005 Solaris
- Mehr als 150 AVR-Typen, ca. 100 Programmer



AVRDUDE: „Schweizer Taschenmesser“

- Gestartet im Jahr 2000 durch Brian S. Dean als privates Projekt `avrprog`, FreeBSD
- Initial für preiswerte Parallelport-Programmer
- Idee: Daten über Programmer und MCUs in einer Textdatei `avrdude.conf` hinterlegen
- Sukzessiv um andere Programmertypen erweitert, beginnend mit Unterstützung für Atmels STK500 (2002)
- 2003 Umbenennung in `AVRDUDE`, Hosting auf `savannah.nongnu.org`, Umlizensierung auf GPL
- 2003 Linux, 2004 Win32, OSX, 2005 Solaris
- Mehr als 150 AVR-Typen, ca. 100 Programmer



AVaRICE

- Emulation: Nachbildung des Controllers im Zielsystem
- ICE — *In-Circuit Emulator*
- Ursprünglich über FPGA + externe Baugruppen; teuer
- JTAG — *Joint Test Action Group*
 - ▶ Ursprünglich Hardwaretest, standardisiert (IEEE 1149)
 - ▶ Erweiterungen für Software-Debugging, herstellerspezifisch
- Atmel: nur über AVR / Atmel Studio
- AVaRICE als GDB Proxy:
 - ▶ Kommunikation mit ICE (seriell, USB)
 - ▶ Kommunikation vom GDB (TCP)



AVaRICE

- Emulation: Nachbildung des Controllers im Zielsystem
- ICE — *In-Circuit Emulator*
- Ursprünglich über FPGA + externe Baugruppen; teuer
- JTAG — *Joint Test Action Group*
 - ▶ Ursprünglich Hardwaretest, standardisiert (IEEE 1149)
 - ▶ Erweiterungen für Software-Debugging, herstellerspezifisch
- Atmel: nur über AVR / Atmel Studio
- AVaRICE als GDB Proxy:
 - ▶ Kommunikation mit ICE (seriell, USB)
 - ▶ Kommunikation vom GDB (TCP)



AVaRICE

- Emulation: Nachbildung des Controllers im Zielsystem
- ICE — *In-Circuit Emulator*
- Ursprünglich über FPGA + externe Baugruppen; teuer
- JTAG — *Joint Test Action Group*
 - ▶ Ursprünglich Hardwaretest, standardisiert (IEEE 1149)
 - ▶ Erweiterungen für Software-Debugging, herstellerspezifisch
- Atmel: nur über AVR / Atmel Studio
- AVaRICE als GDB Proxy:
 - ▶ Kommunikation mit ICE (seriell, USB)
 - ▶ Kommunikation vom GDB (TCP)



AVaRICE

- Emulation: Nachbildung des Controllers im Zielsystem
- ICE — *In-Circuit Emulator*
- Ursprünglich über FPGA + externe Baugruppen; teuer
- JTAG — *Joint Test Action Group*
 - ▶ Ursprünglich Hardwaretest, standardisiert (IEEE 1149)
 - ▶ Erweiterungen für Software-Debugging, herstellerspezifisch
- Atmel: nur über AVR / Atmel Studio
- AVaRICE als GDB Proxy:
 - ▶ Kommunikation mit ICE (seriell, USB)
 - ▶ Kommunikation vom GDB (TCP)



AVaRICE

- Emulation: Nachbildung des Controllers im Zielsystem
- ICE — *In-Circuit Emulator*
- Ursprünglich über FPGA + externe Baugruppen; teuer
- JTAG — *Joint Test Action Group*
 - ▶ Ursprünglich Hardwaretest, standardisiert (IEEE 1149)
 - ▶ Erweiterungen für Software-Debugging, herstellerspezifisch
- Atmel: nur über AVR / Atmel Studio
- AVaRICE als GDB Proxy:
 - ▶ Kommunikation mit ICE (seriell, USB)
 - ▶ Kommunikation vom GDB (TCP)



AVaRICE

- Emulation: Nachbildung des Controllers im Zielsystem
- ICE — *In-Circuit Emulator*
- Ursprünglich über FPGA + externe Baugruppen; teuer
- JTAG — *Joint Test Action Group*
 - ▶ Ursprünglich Hardwaretest, standardisiert (IEEE 1149)
 - ▶ Erweiterungen für Software-Debugging, herstellerspezifisch
- Atmel: nur über AVR / Atmel Studio
- AVaRICE als GDB Proxy:
 - ▶ Kommunikation mit ICE (seriell, USB)
 - ▶ Kommunikation vom GDB (TCP)



AVaRICE

- Emulation: Nachbildung des Controllers im Zielsystem
- ICE — *In-Circuit Emulator*
- Ursprünglich über FPGA + externe Baugruppen; teuer
- JTAG — *Joint Test Action Group*
 - ▶ Ursprünglich Hardwaretest, standardisiert (IEEE 1149)
 - ▶ Erweiterungen für Software-Debugging, herstellerspezifisch
- Atmel: nur über AVR / Atmel Studio
- AVaRICE als GDB Proxy:
 - ▶ Kommunikation mit ICE (seriell, USB)
 - ▶ Kommunikation vom GDB (TCP)



AVaRICE

- Emulation: Nachbildung des Controllers im Zielsystem
- ICE — *In-Circuit Emulator*
- Ursprünglich über FPGA + externe Baugruppen; teuer
- JTAG — *Joint Test Action Group*
 - ▶ Ursprünglich Hardwaretest, standardisiert (IEEE 1149)
 - ▶ Erweiterungen für Software-Debugging, herstellerspezifisch
- Atmel: nur über AVR / Atmel Studio
- AVaRICE als GDB Proxy:
 - ▶ Kommunikation mit ICE (seriell, USB)
 - ▶ Kommunikation vom GDB (TCP)



AVaRICE

- Emulation: Nachbildung des Controllers im Zielsystem
- ICE — *In-Circuit Emulator*
- Ursprünglich über FPGA + externe Baugruppen; teuer
- JTAG — *Joint Test Action Group*
 - ▶ Ursprünglich Hardwaretest, standardisiert (IEEE 1149)
 - ▶ Erweiterungen für Software-Debugging, herstellerspezifisch
- Atmel: nur über AVR / Atmel Studio
- AVaRICE als GDB Proxy:
 - ▶ Kommunikation mit ICE (seriell, USB)
 - ▶ Kommunikation vom GDB (TCP)



AVaRICE

- Emulation: Nachbildung des Controllers im Zielsystem
- ICE — *In-Circuit Emulator*
- Ursprünglich über FPGA + externe Baugruppen; teuer
- JTAG — *Joint Test Action Group*
 - ▶ Ursprünglich Hardwaretest, standardisiert (IEEE 1149)
 - ▶ Erweiterungen für Software-Debugging, herstellerspezifisch
- Atmel: nur über AVR / Atmel Studio
- AVaRICE als GDB Proxy:
 - ▶ Kommunikation mit ICE (seriell, USB)
 - ▶ Kommunikation vom GDB (TCP)



Opensource: Wie teilnehmen?

- AVRDUDE:

- ▶ Zusammenarbeit mit Brian Dean bereits durch früheres FreeBSD-Projekt
- ▶ Diverse Beiträge, Reviews, *man page* etc.
- ▶ Nach kurzer Zeit Ko-Autor



- avr-libc:

- ▶ Durch Marek Michalkiewicz aus Vorgängerprojekt von Denis Chertykov und Uros Platise übernommen
- ▶ Erster eigener Beitrag: `malloc()`



- Theodore (Ted) A. Roth: Umzug der Projekte nach `savannah.nongnu.org`; Administration

- Als öffentliche Opensource-Projekte weiter betrieben

Opensource: Wie teilnehmen?

- AVRDUDE:

- ▶ Zusammenarbeit mit Brian Dean bereits durch früheres FreeBSD-Projekt
- ▶ Diverse Beiträge, Reviews, *man page* etc.
- ▶ Nach kurzer Zeit Ko-Autor



- avr-libc:

- ▶ Durch Marek Michalkiewicz aus Vorgängerprojekt von Denis Chertykov und Uros Platise übernommen
- ▶ Erster eigener Beitrag: `malloc()`



- Theodore (Ted) A. Roth: Umzug der Projekte nach `savannah.nongnu.org`; Administration
- Als öffentliche Opensource-Projekte weiter betrieben

Opensource: Wie teilnehmen?

- AVRDUDE:

- ▶ Zusammenarbeit mit Brian Dean bereits durch früheres FreeBSD-Projekt
- ▶ Diverse Beiträge, Reviews, *man page* etc.
- ▶ Nach kurzer Zeit Ko-Autor



- avr-libc:

- ▶ Durch Marek Michalkiewicz aus Vorgängerprojekt von Denis Chertykov und Uros Platise übernommen
- ▶ Erster eigener Beitrag: `malloc()`

- Theodore (Ted) A. Roth: Umzug der Projekte nach `savannah.nongnu.org`; Administration



- Als öffentliche Opensource-Projekte weiter betrieben

Opensource: Wie teilnehmen?

- AVRDUDE:

- ▶ Zusammenarbeit mit Brian Dean bereits durch früheres FreeBSD-Projekt
- ▶ Diverse Beiträge, Reviews, *man page* etc.
- ▶ Nach kurzer Zeit Ko-Autor



- avr-libc:

- ▶ Durch Marek Michalkiewicz aus Vorgängerprojekt von Denis Chertykov und Uros Platise übernommen
- ▶ Erster eigener Beitrag: `malloc()`

- Theodore (Ted) A. Roth: Umzug der Projekte nach `savannah.nongnu.org`; Administration



- Als öffentliche Opensource-Projekte weiter betrieben

Opensource: Wie teilnehmen?

- AVRDUDE:

- ▶ Zusammenarbeit mit Brian Dean bereits durch früheres FreeBSD-Projekt
- ▶ Diverse Beiträge, Reviews, *man page* etc.
- ▶ Nach kurzer Zeit Ko-Autor

- avr-libc:

- ▶ Durch Marek Michalkiewicz aus Vorgängerprojekt von Denis Chertykov und Uros Platise übernommen
- ▶ Erster eigener Beitrag: `malloc()`

- Theodore (Ted) A. Roth: Umzug der Projekte nach `savannah.nongnu.org`; Administration

- Als öffentliche Opensource-Projekte weiter betrieben



Opensource: Wie teilnehmen?

- AVRDUDE:

- ▶ Zusammenarbeit mit Brian Dean bereits durch früheres FreeBSD-Projekt
- ▶ Diverse Beiträge, Reviews, *man page* etc.
- ▶ Nach kurzer Zeit Ko-Autor

- avr-libc:

- ▶ Durch Marek Michalkiewicz aus Vorgängerprojekt von Denis Chertykov und Uros Platise übernommen
- ▶ Erster eigener Beitrag: `malloc()`

- Theodore (Ted) A. Roth: Umzug der Projekte nach `savannah.nongnu.org`; Administration

- Als öffentliche Opensource-Projekte weiter betrieben



Opensource: Wie teilnehmen?

- AVRDUDE:

- ▶ Zusammenarbeit mit Brian Dean bereits durch früheres FreeBSD-Projekt
- ▶ Diverse Beiträge, Reviews, *man page* etc.
- ▶ Nach kurzer Zeit Ko-Autor

- avr-libc:

- ▶ Durch Marek Michalkiewicz aus Vorgängerprojekt von Denis Chertykov und Uros Platise übernommen
- ▶ Erster eigener Beitrag: `malloc()`

- Theodore (Ted) A. Roth: Umzug der Projekte nach `savannah.nongnu.org`; Administration

- Als öffentliche Opensource-Projekte weiter betrieben



Opensource: Wie teilnehmen?

- AVRDUDE:

- ▶ Zusammenarbeit mit Brian Dean bereits durch früheres FreeBSD-Projekt
- ▶ Diverse Beiträge, Reviews, *man page* etc.
- ▶ Nach kurzer Zeit Ko-Autor



- avr-libc:

- ▶ Durch Marek Michalkiewicz aus Vorgängerprojekt von Denis Chertykov und Uros Platise übernommen
- ▶ Erster eigener Beitrag: `malloc()`



- Theodore (Ted) A. Roth: Umzug der Projekte nach `savannah.nongnu.org`; Administration

- Als öffentliche Opensource-Projekte weiter betrieben

Opensource: Wie teilnehmen?

- AVRDUDE:

- ▶ Zusammenarbeit mit Brian Dean bereits durch früheres FreeBSD-Projekt
- ▶ Diverse Beiträge, Reviews, *man page* etc.
- ▶ Nach kurzer Zeit Ko-Autor

- avr-libc:

- ▶ Durch Marek Michalkiewicz aus Vorgängerprojekt von Denis Chertykov und Uros Platise übernommen
- ▶ Erster eigener Beitrag: `malloc()`

- Theodore (Ted) A. Roth: Umzug der Projekte nach `savannah.nongnu.org`; Administration

- Als öffentliche Opensource-Projekte weiter betreiben



Opensource: Verantwortung übernehmen?

- Ted Roth zog sich Ende 2004 aus persönlichen Gründen zurück
- Neben seiner Administration von `avr-libc` und `AVRDUDE` auch `AVaRICE` betroffen
- Brian Dean's letzter Eintrag im ChangeLog: September 2005
- Marek Michalkiewicz: arbeitsbedingter Ausstieg aus den Opensource-Projekten 2006
- ⇒ Administrative Übernahme der Projekte
- Spagat: Entwickler und „Manager“ zugleich



Opensource: Verantwortung übernehmen?

- Ted Roth zog sich Ende 2004 aus persönlichen Gründen zurück
- Neben seiner Administration von `avr-libc` und `AVRDUDE` auch `AVaRICE` betroffen
- Brian Dean's letzter Eintrag im ChangeLog: September 2005
- Marek Michalkiewicz: arbeitsbedingter Ausstieg aus den Opensource-Projekten 2006
- ⇒ Administrative Übernahme der Projekte
- Spagat: Entwickler und „Manager“ zugleich



Opensource: Verantwortung übernehmen?

- Ted Roth zog sich Ende 2004 aus persönlichen Gründen zurück
- Neben seiner Administration von `avr-libc` und `AVRDUDE` auch `AVaRICE` betroffen
- Brian Dean's letzter Eintrag im ChangeLog: September 2005
- Marek Michalkiewicz: arbeitsbedingter Ausstieg aus den Opensource-Projekten 2006
- ⇒ Administrative Übernahme der Projekte
- Spagat: Entwickler und „Manager“ zugleich

Permissions per member

Projects Admins are always allowed to read private items.

Member	General Rights	On Duty	Bug Tracker	Task Tracker	Ph...
Eric Wedderburn <eric@rump>	<input type="checkbox"/> Admin <input type="checkbox"/> Invoce Item	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/> Techn. & Manager	<input checked="" type="checkbox"/> Techn. & Manager	<input checked="" type="checkbox"/> Techn...
Axel Wächter <awachter>	<input type="checkbox"/> Admin <input type="checkbox"/> Invoce Item	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/> Group Default	<input checked="" type="checkbox"/> Group Default	<input checked="" type="checkbox"/> Group...
Brian Dean <bdolan>	<input type="checkbox"/> Admin	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/> Techn. & Manager	<input checked="" type="checkbox"/> Techn. & Manager	<input checked="" type="checkbox"/> Tech...
Colin O'Flaherty <colofl@rump>	<input type="checkbox"/> Admin <input type="checkbox"/> Invoce Item	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/> Techn. & Manager	<input checked="" type="checkbox"/> Techn. & Manager	<input checked="" type="checkbox"/> Tech...
Thomas Reich <treich>	<input type="checkbox"/> Admin <input type="checkbox"/> Invoce Item	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/> Techn. & Manager	<input checked="" type="checkbox"/> Techn. & Manager	<input checked="" type="checkbox"/> Tech...
Hannes Weibach <hweibach>	<input type="checkbox"/> Admin <input type="checkbox"/> Invoce Item	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/> Group Default	<input checked="" type="checkbox"/> Group Default	<input checked="" type="checkbox"/> Group...
Jörg Wunsch <joerg.wunsch@avrfreaks.org>	<input type="checkbox"/> Admin	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/> Techn. & Manager	<input checked="" type="checkbox"/> Techn. & Manager	<input checked="" type="checkbox"/> Tech...
Berni Leitbacher <leitbacher>	<input type="checkbox"/> Admin <input type="checkbox"/> Invoce Item	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/> Group Default	<input checked="" type="checkbox"/> Group Default	<input checked="" type="checkbox"/> Group...

Opensource: Verantwortung übernehmen?

- Ted Roth zog sich Ende 2004 aus persönlichen Gründen zurück
- Neben seiner Administration von `avr-libc` und `AVRDUDE` auch `AVaRICE` betroffen
- Brian Dean's letzter Eintrag im ChangeLog: September 2005
- Marek Michalkiewicz: arbeitsbedingter Ausstieg aus den Opensource-Projekten 2006
- ⇒ Administrative Übernahme der Projekte
- Spagat: Entwickler und „Manager“ zugleich

Permissions per member

Projects Admins are always allowed to read private items.

Member	General Rights	On Duty	Bug Tracker	Task Tracker	Ph...
Eric Weddington <eric@rump>	<input type="checkbox"/> Admin <input type="checkbox"/> Invoic Items	<input type="checkbox"/>	<input type="checkbox"/> Techn. & Manager	<input type="checkbox"/> Techn. & Manager	<input type="checkbox"/> Techn.
Axel Wächter <awachter>	<input type="checkbox"/> Admin <input type="checkbox"/> Invoic Items	<input type="checkbox"/>	<input type="checkbox"/> Group Default	<input type="checkbox"/> Group Default	<input type="checkbox"/> Group
Brian Dean <bdolan>	<input type="checkbox"/> Admin	<input type="checkbox"/>	<input type="checkbox"/> Techn. & Manager	<input type="checkbox"/> Techn. & Manager	<input type="checkbox"/> Techn.
Colin O'Flaherty <colofl@rump>	<input type="checkbox"/> Admin <input type="checkbox"/> Invoic Items	<input type="checkbox"/>	<input type="checkbox"/> Techn. & Manager	<input type="checkbox"/> Techn. & Manager	<input type="checkbox"/> Techn.
Thomas Fischl <TFischl>	<input type="checkbox"/> Admin <input type="checkbox"/> Invoic Items	<input type="checkbox"/>	<input type="checkbox"/> Techn. & Manager	<input type="checkbox"/> Techn. & Manager	<input type="checkbox"/> Techn.
Hannes Weibach <hweibach>	<input type="checkbox"/> Admin <input type="checkbox"/> Invoic Items	<input type="checkbox"/>	<input type="checkbox"/> Group Default	<input type="checkbox"/> Group Default	<input type="checkbox"/> Group
Jörg Wunsch <joerg.wunsch@avrdude.org>	<input type="checkbox"/> Admin	<input type="checkbox"/>	<input type="checkbox"/> Techn. & Manager	<input type="checkbox"/> Techn. & Manager	<input type="checkbox"/> Techn.
Berni Leitbacher <leitbacher>	<input type="checkbox"/> Admin <input type="checkbox"/> Invoic Items	<input type="checkbox"/>	<input type="checkbox"/> Group Default	<input type="checkbox"/> Group Default	<input type="checkbox"/> Group

Opensource: Beiträge

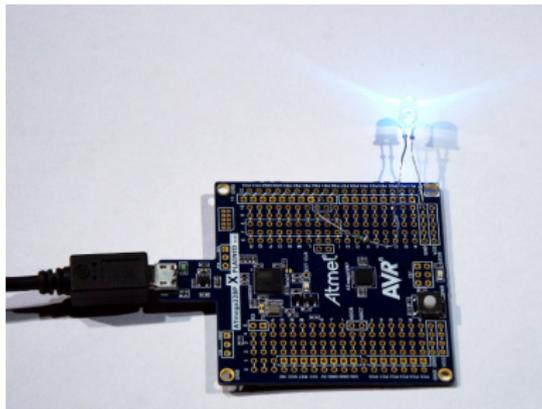
- Bug reports: wichtig; unbedingt in *Bug tracker*, Bug reports in Mails gehen verloren, sofern sie nicht in wenigen Minuten eingearbeitet werden können
- Patches: Hauptarbeit liegt meist nicht im Sourcecode, sondern in der Verifizierung; Dokumentation neuer Features gern „vergessen“
- Hardware: meist unproblematisch zu bekommen
- Finanzielle Beiträge: nie ernsthaft getestet; Hobby-Arbeit muss Spaß machen, möglichst kein Zeitdruck

AVR Downloader/UploadER - Bugs: Browse Items

100 matching items - Items 1 to 50

Item ID	Summary	Status	Assigned To
#42278	AVRISP11: detection of status in configuration file	None	None
#42279	AVRISP11: detection of status in configuration file	None	None
#42280	AVRISP11: detection of status in configuration file	None	None
#42281	AVRISP11: detection of status in configuration file	None	None
#42282	AVRISP11: detection of status in configuration file	None	None
#42283	AVRISP11: detection of status in configuration file	None	None
#42284	AVRISP11: detection of status in configuration file	None	None
#42285	AVRISP11: detection of status in configuration file	None	None
#42286	AVRISP11: detection of status in configuration file	None	None
#42287	AVRISP11: detection of status in configuration file	None	None
#42288	AVRISP11: detection of status in configuration file	None	None
#42289	AVRISP11: detection of status in configuration file	None	None
#42290	AVRISP11: detection of status in configuration file	None	None
#42291	AVRISP11: detection of status in configuration file	None	None
#42292	AVRISP11: detection of status in configuration file	None	None
#42293	AVRISP11: detection of status in configuration file	None	None
#42294	AVRISP11: detection of status in configuration file	None	None
#42295	AVRISP11: detection of status in configuration file	None	None
#42296	AVRISP11: detection of status in configuration file	None	None
#42297	AVRISP11: detection of status in configuration file	None	None
#42298	AVRISP11: detection of status in configuration file	None	None
#42299	AVRISP11: detection of status in configuration file	None	None
#42300	AVRISP11: detection of status in configuration file	None	None
#42301	AVRISP11: detection of status in configuration file	None	None
#42302	AVRISP11: detection of status in configuration file	None	None
#42303	AVRISP11: detection of status in configuration file	None	None
#42304	AVRISP11: detection of status in configuration file	None	None
#42305	AVRISP11: detection of status in configuration file	None	None
#42306	AVRISP11: detection of status in configuration file	None	None
#42307	AVRISP11: detection of status in configuration file	None	None
#42308	AVRISP11: detection of status in configuration file	None	None
#42309	AVRISP11: detection of status in configuration file	None	None
#42310	AVRISP11: detection of status in configuration file	None	None
#42311	AVRISP11: detection of status in configuration file	None	None
#42312	AVRISP11: detection of status in configuration file	None	None
#42313	AVRISP11: detection of status in configuration file	None	None
#42314	AVRISP11: detection of status in configuration file	None	None
#42315	AVRISP11: detection of status in configuration file	None	None
#42316	AVRISP11: detection of status in configuration file	None	None
#42317	AVRISP11: detection of status in configuration file	None	None
#42318	AVRISP11: detection of status in configuration file	None	None
#42319	AVRISP11: detection of status in configuration file	None	None
#42320	AVRISP11: detection of status in configuration file	None	None
#42321	AVRISP11: detection of status in configuration file	None	None
#42322	AVRISP11: detection of status in configuration file	None	None
#42323	AVRISP11: detection of status in configuration file	None	None
#42324	AVRISP11: detection of status in configuration file	None	None
#42325	AVRISP11: detection of status in configuration file	None	None
#42326	AVRISP11: detection of status in configuration file	None	None
#42327	AVRISP11: detection of status in configuration file	None	None
#42328	AVRISP11: detection of status in configuration file	None	None
#42329	AVRISP11: detection of status in configuration file	None	None
#42330	AVRISP11: detection of status in configuration file	None	None
#42331	AVRISP11: detection of status in configuration file	None	None
#42332	AVRISP11: detection of status in configuration file	None	None
#42333	AVRISP11: detection of status in configuration file	None	None
#42334	AVRISP11: detection of status in configuration file	None	None
#42335	AVRISP11: detection of status in configuration file	None	None
#42336	AVRISP11: detection of status in configuration file	None	None
#42337	AVRISP11: detection of status in configuration file	None	None
#42338	AVRISP11: detection of status in configuration file	None	None
#42339	AVRISP11: detection of status in configuration file	None	None
#42340	AVRISP11: detection of status in configuration file	None	None
#42341	AVRISP11: detection of status in configuration file	None	None
#42342	AVRISP11: detection of status in configuration file	None	None
#42343	AVRISP11: detection of status in configuration file	None	None
#42344	AVRISP11: detection of status in configuration file	None	None
#42345	AVRISP11: detection of status in configuration file	None	None
#42346	AVRISP11: detection of status in configuration file	None	None
#42347	AVRISP11: detection of status in configuration file	None	None
#42348	AVRISP11: detection of status in configuration file	None	None
#42349	AVRISP11: detection of status in configuration file	None	None
#42350	AVRISP11: detection of status in configuration file	None	None

Vorführung



Vielen Dank!

