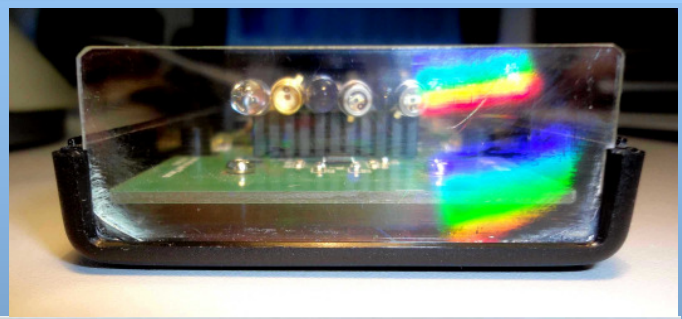
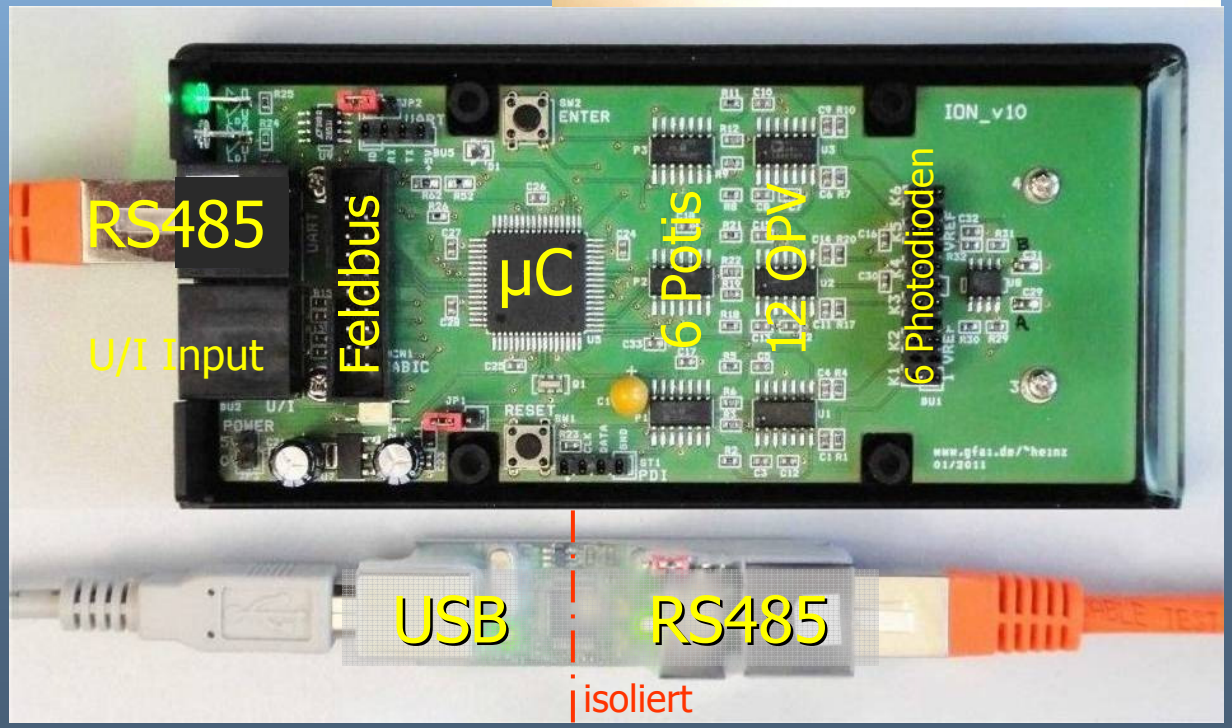
Bedienungsanleitung Spektralregler Ion

- Installation
- Bedienung
- Bestandteile

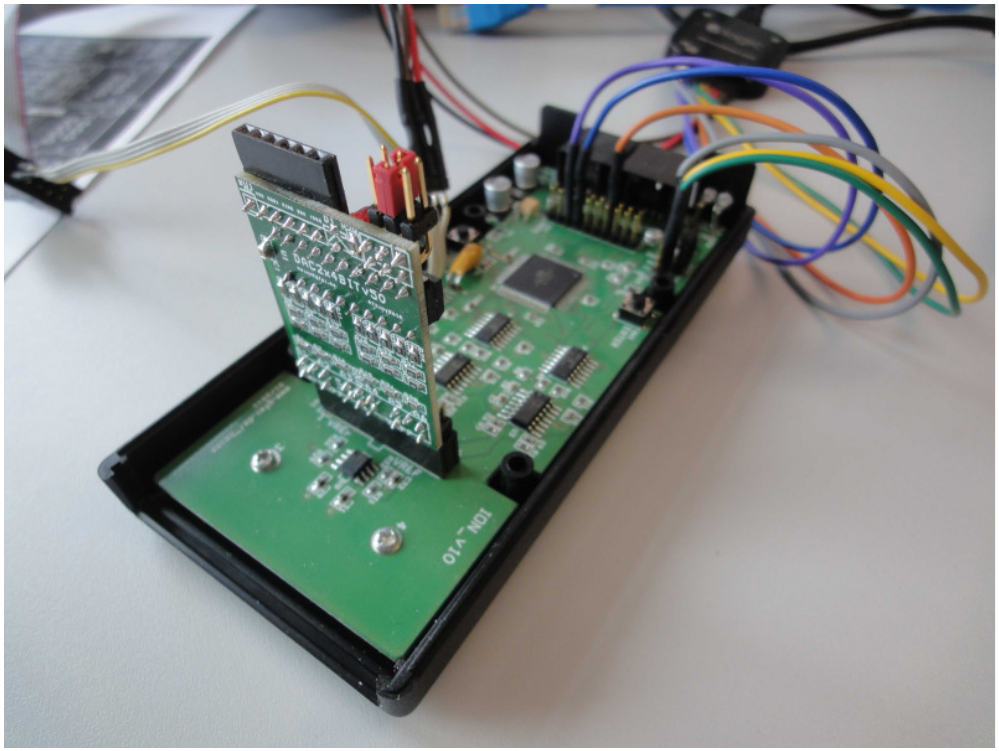




Dr. G. Heinz, GFaI e.V.
Volmerstr.3
12489 Berlin
Tel. +49 (30) 814563-490
Fax. -302
www.gfai.de/~heinz
heinz@gfai.de

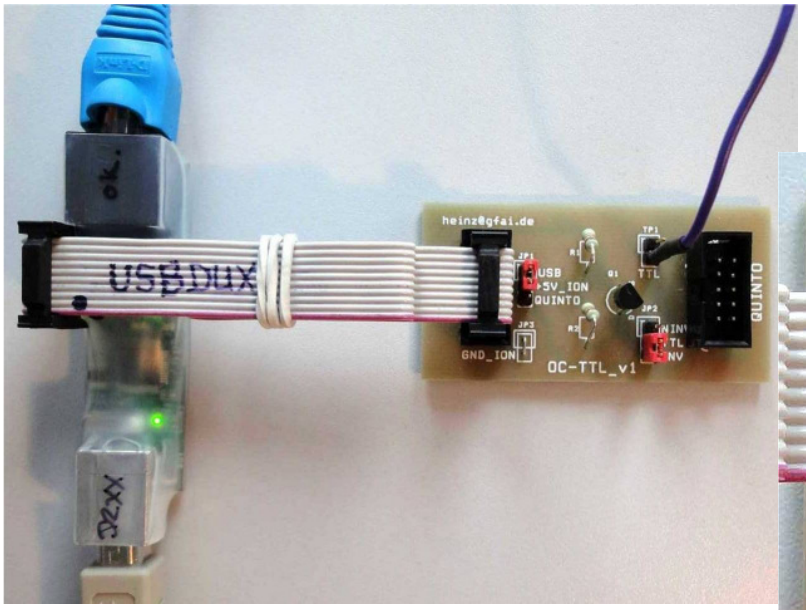
Installation Hardware

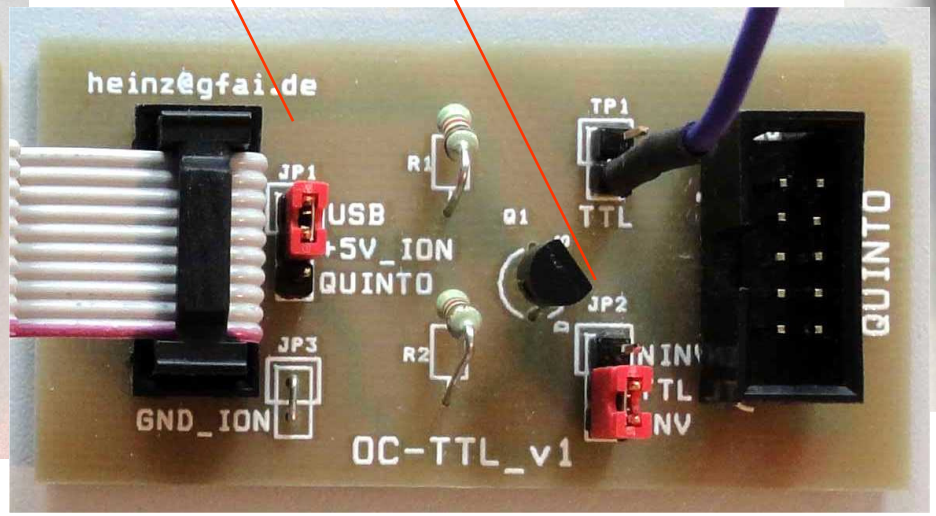
- Für erste Experimente kann der Zeitfunktionsgenerator DAC2x4BIT genutzt werden, dieser wird anstelle von Photodioden auf BU1 des Ion gesteckt (Abbildung)
- Versorgung ist integriert mit 3,3V über den Photodiodenstecker
- Verpolungsschutz: der Stecker des DAC ist asymmetrisch, der DAC kann nur wie dargestellt gesteckt werden
- Installieren Sie den USB-Treiber von FTDI CDM20828_Setup.exe
- Verbinden Sie Ion mit USB-Adapter und PC, Ion wird über USB mit +5V versorgt, rote und grüne LEDs sollten jetzt leuchten (grün stark, rot schwach)



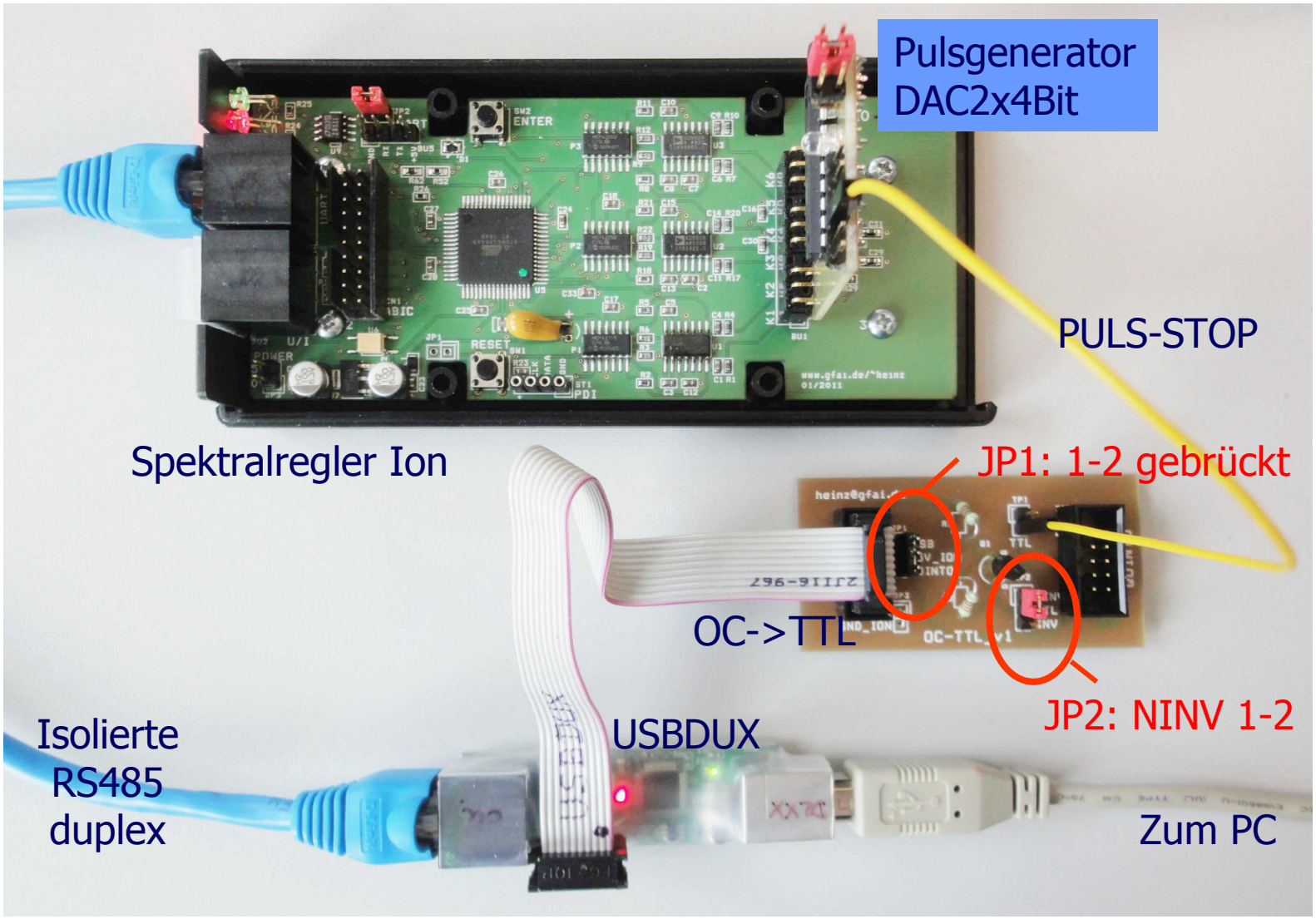
Installation USB-RS485 Adapter (USBDUX)

- Am USBDUX befindet sich Jumper JP2 zur Wahl der Betriebsspannungsquelle des Ion (USB, Brücke auf 2-3)
- Wird ein OC-TTL Adapter angeschlossen, wird JP2 geöffnet, Ion erhält keine Betriebsspannung mehr
- Auf dem OC-TTL ist an JP1 die gewünschte Versorgungsquelle (USB oder Quinto) zu setzen
- OC-TTL JP2 ist für den Anschluß von Simulatoren (Plasmasim, DAC2x4BIT) auf NINV, für Quinto auf INV zu setzen

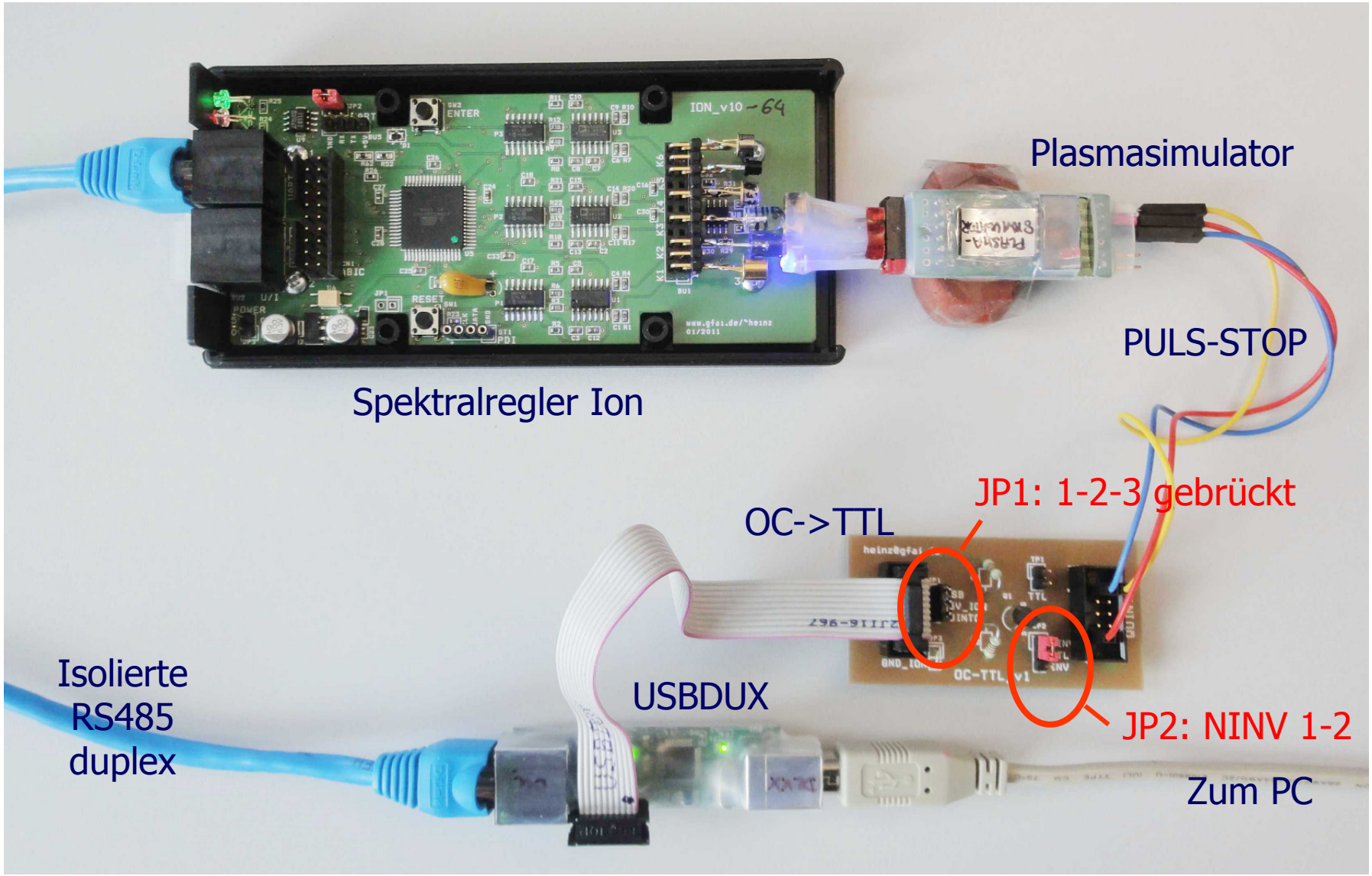




Konfiguration mit DAC

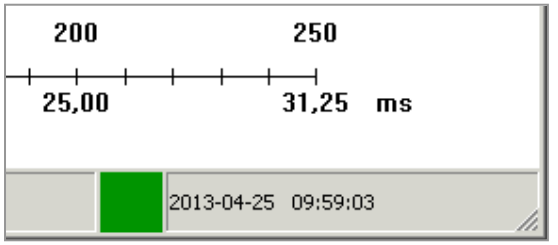


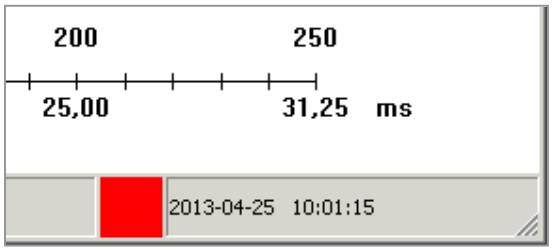
Konfiguration mit Plasmasimulator (optional)

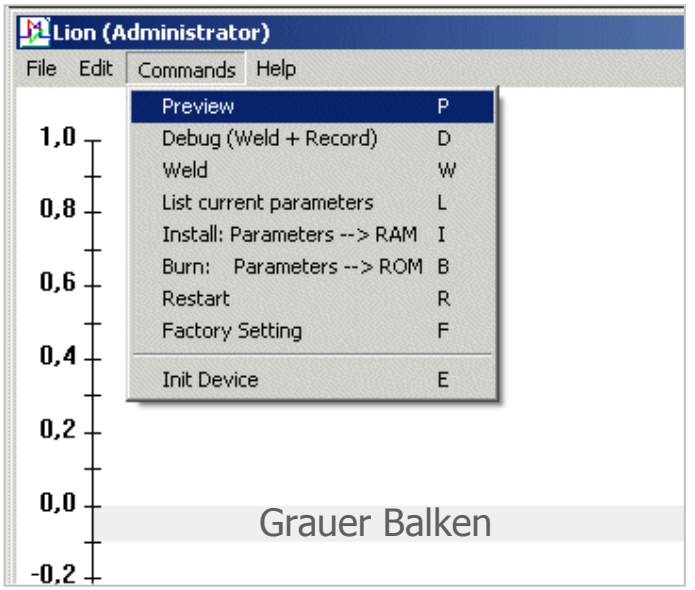


Installation Lion

- Starten Sie die für Ihren PC geeignete Exe:
  - 32 Bit: /i386/Lion32.exe
  - 64 Bit: /amd64/Lion64.exe
- Es erscheint der Eröffnungsbildschirm
- Sofern USBDUX am PC angeschlossen ist, zeigt Lion ein grünes Karo, siehe Bild
- Ist USBDUX nicht angeschlossen oder ist der FTDI-Driver nicht installiert, bleibt das Karo rot, siehe Bild darunter
- Mittig erscheint ein grauer Balken im Bild, das Commands-Menue ist jetzt anklickbar, siehe Bild unten
- Starten Sie z.B. mit Commands/Preview







Preview starten

- Starten Sie die Live-Beobachtung von Zeitfunktionen mit Commands/Preview
- Lion holt nun kontinuierlich kurze Datensätze vom Spektralregler und stellt sie dar
- Die Anzahl geholter Samples a 32 Byte kann mit Edit/Parameter/FT verändert werden
- Beenden Sie den Mode mit dem Befehl Commands/Preview

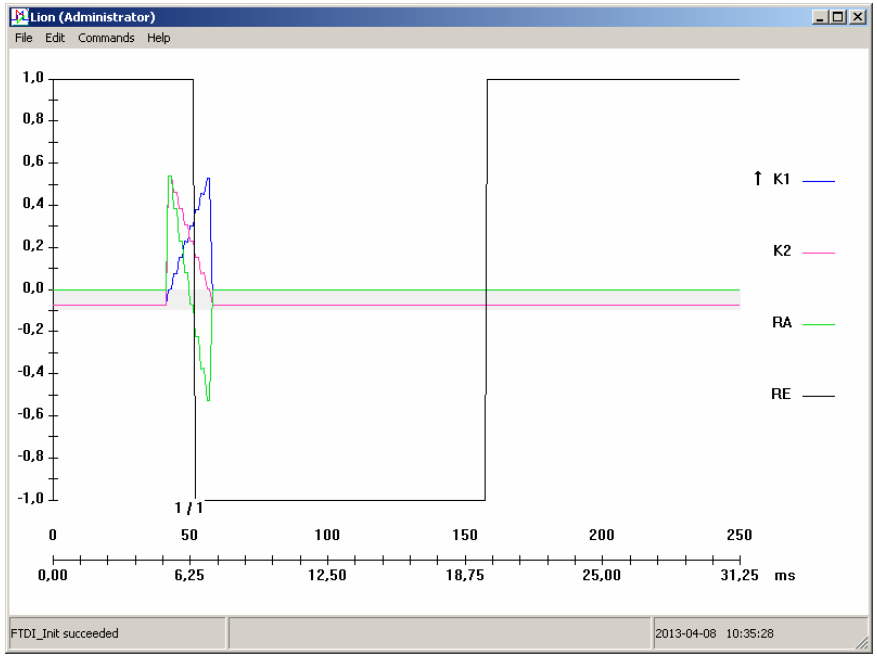
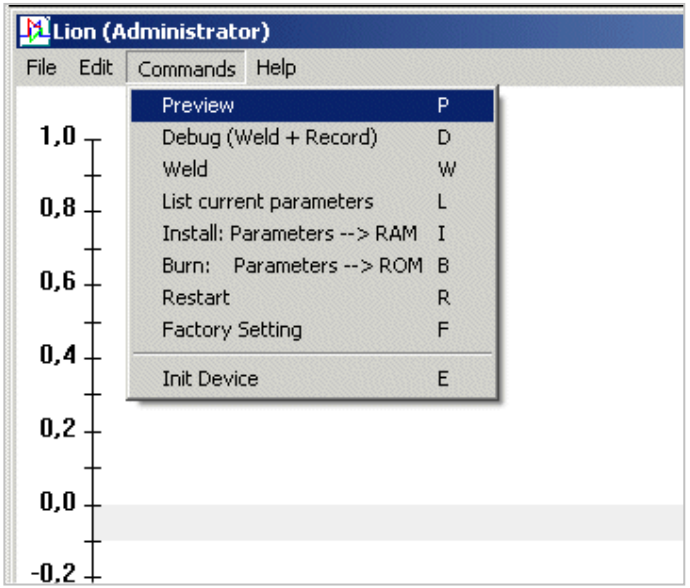


Bild der vom DAC2x4BIT erzeugten Zeitfunktionen K1, K2

RA und RE sind vom Ion berechnet

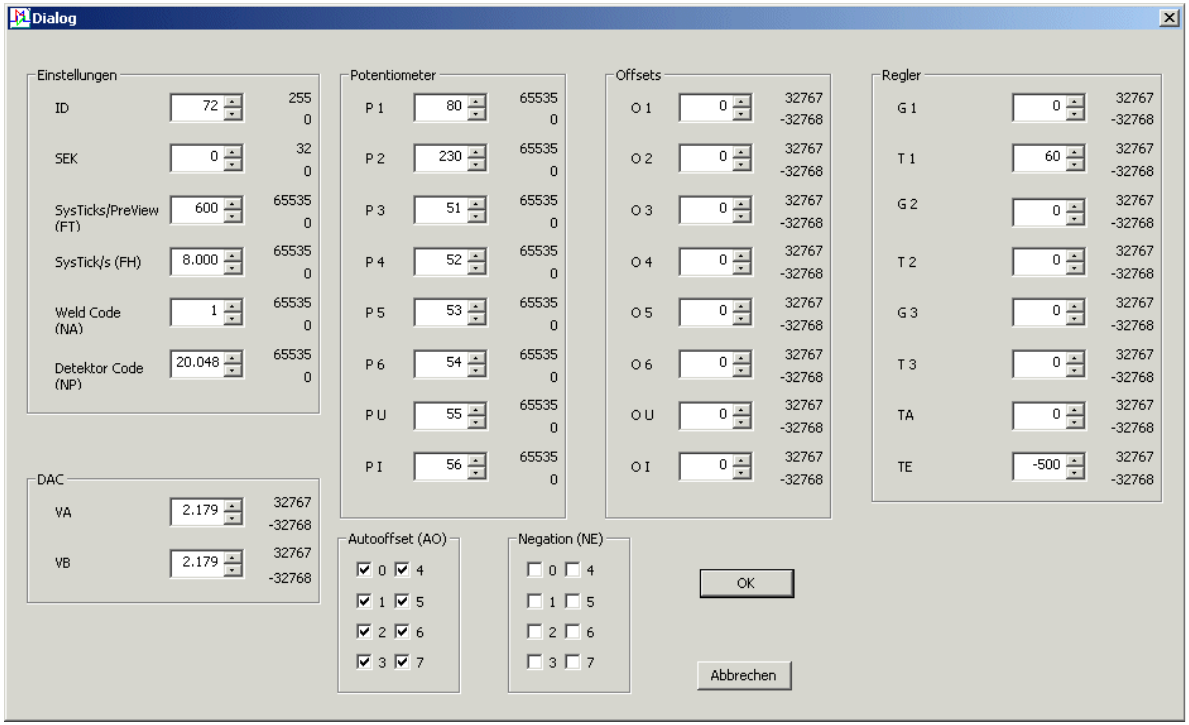
Weitere Befehle im Menue "Commands"



- Daten werden aufgenommen mit Debug, die Aufnahmezeit in Sekunden wird mit dem Parametermenue (SEK) übergeben
- Im Mode Weld steuert Ion nur die Schweißmaschine, ohne Daten zum PC auszugeben
- Ion überträgt die aktuell eingestellten Parameter in das Parametermenue mit List current parameters
- Sind Parameter nur zu testen, können sie mit Install in den RAM des Ion geschrieben werden
- Ist ein Parametersatz korrekt, kann er mit Burn ins ROM des Ion gebrannt werden (maximal 10.000 mal); Ion kommt dann beim Neustart mit diesem Parametersatz hoch
- Sollte die Kommunikation gestört sein, könnte ein Restart helfen
- Nach einer Neuprogrammierung des Ion wird ein Factory Reset empfohlen. Dieses setzt alle Parameter auf Werkseinstellung zurück.

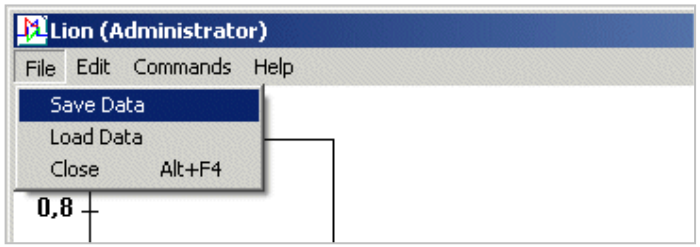
Parameter einstellen

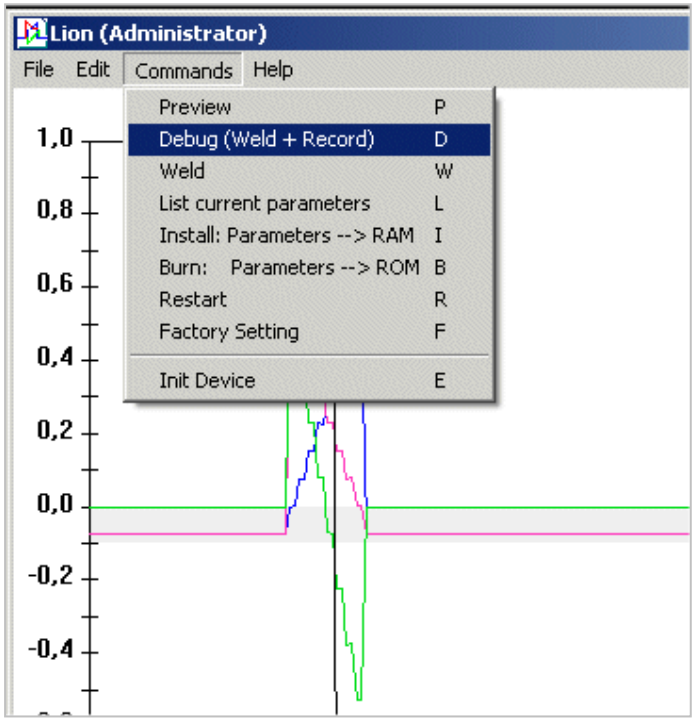
- Verändern Sie Parameter im Menue Edit/Parameter
- Es erscheint das bei Lion-Initialisierung (init device) gelesene Parameterfeld des Ion
- Hinweis: Einige systemrelevante Parameter des Ion sind nicht schreibbar, diese können nur gelesen werden (FH, VA, VB)
- Verstärkungen der Kanäle können mit P1 ...PI verändert werden
- Offsets der Kanäle sind mit O1...OI zu korrigieren
- Der Regler mit NA=1 verwendet die Werte und T1 und TE
- ID ist frei wählbar



Daten aufnehmen (Debug)

- Vergewissern Sie sich im Parametermenue, wieviele Sekunden eingestellt sind, korrigieren Sie gegebenenfalls den Wert
- Starten Sie eine Aufnahme mit Command/Debug
- Die aufgenommenen Daten können abgespeichert werden mit File/Save Data

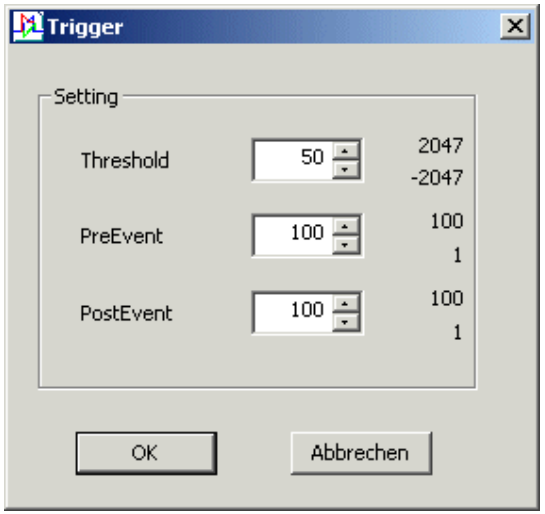




- Laden Sie früher abgespeicherte Daten mit File/Load Data
- Haben Sie keinen Ion angeschlossen, können dennoch abgespeicherte Files geladen und ausgewertet werden

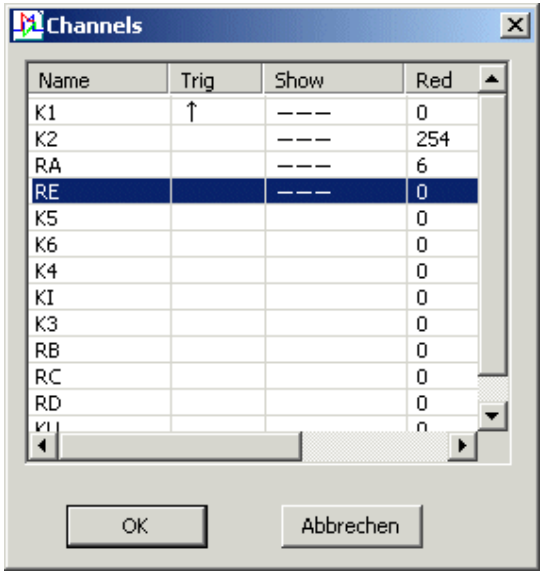
Verändern des Triggers

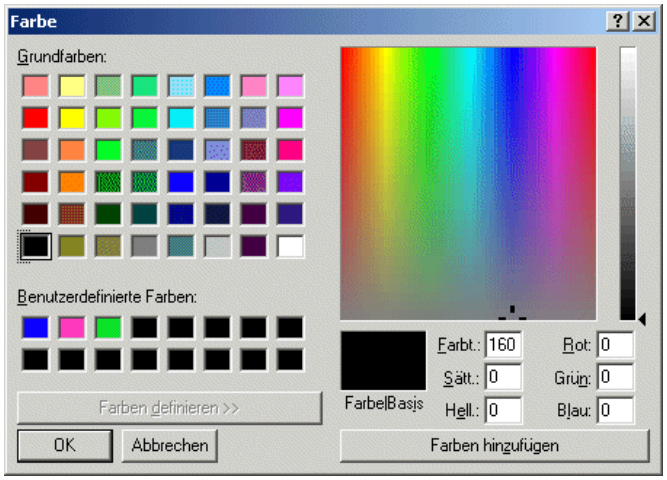
- Lion triggert die aufgenommenen Zeitfunktionen automatisch auf einen im Bild einstellbaren Punkt
- Verschieben Sie den Triggerpunkt entlang der Zeitachse mit Edit/Trigger/PreEvent (Angaben sind in Samples)
- Korrigieren sie die Triggerhöhe mit Edit/Trigger/Threshold (in Digits)
- Falls auf der zu triggernden Zeitfunktion hintereinander schnelle Signalwechsel stattfinden, würde Lion jeweils auf den nächsten Wechsel triggern
- Um solche transienten Wechsel zu vermeiden, stellt Edit/PostEvent eine Pause bereit, innerhalb derer nicht nach der nächsten Triggerflanke gesucht wird



Verändern der Darstellung

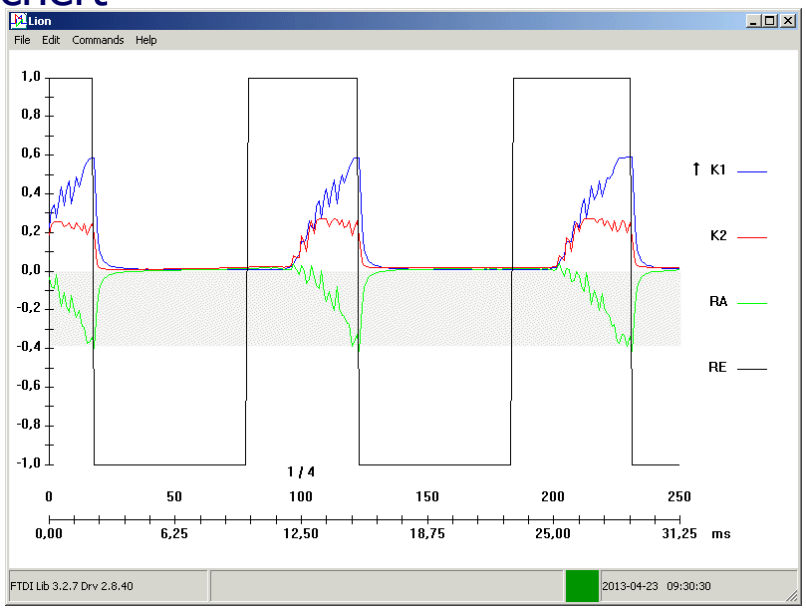
- Ändern Sie die Farbe der Kanäle mit Edit/Channels. Schieben Sie die Maus über den Kanal und Tippen sie ein "c"; das Windows-Farbmenue erscheint
- Ändern Sie den Triggerkanal, indem die Maus über den gewünschten Kanal geschoben wird. Mit "u" (up) oder "d" (down) werden Kanal und Flankenrichtung gesetzt
- Machen Sie Kanäle sichtbar oder unsichtbar mit der Leerzeichen-Taste ("space")





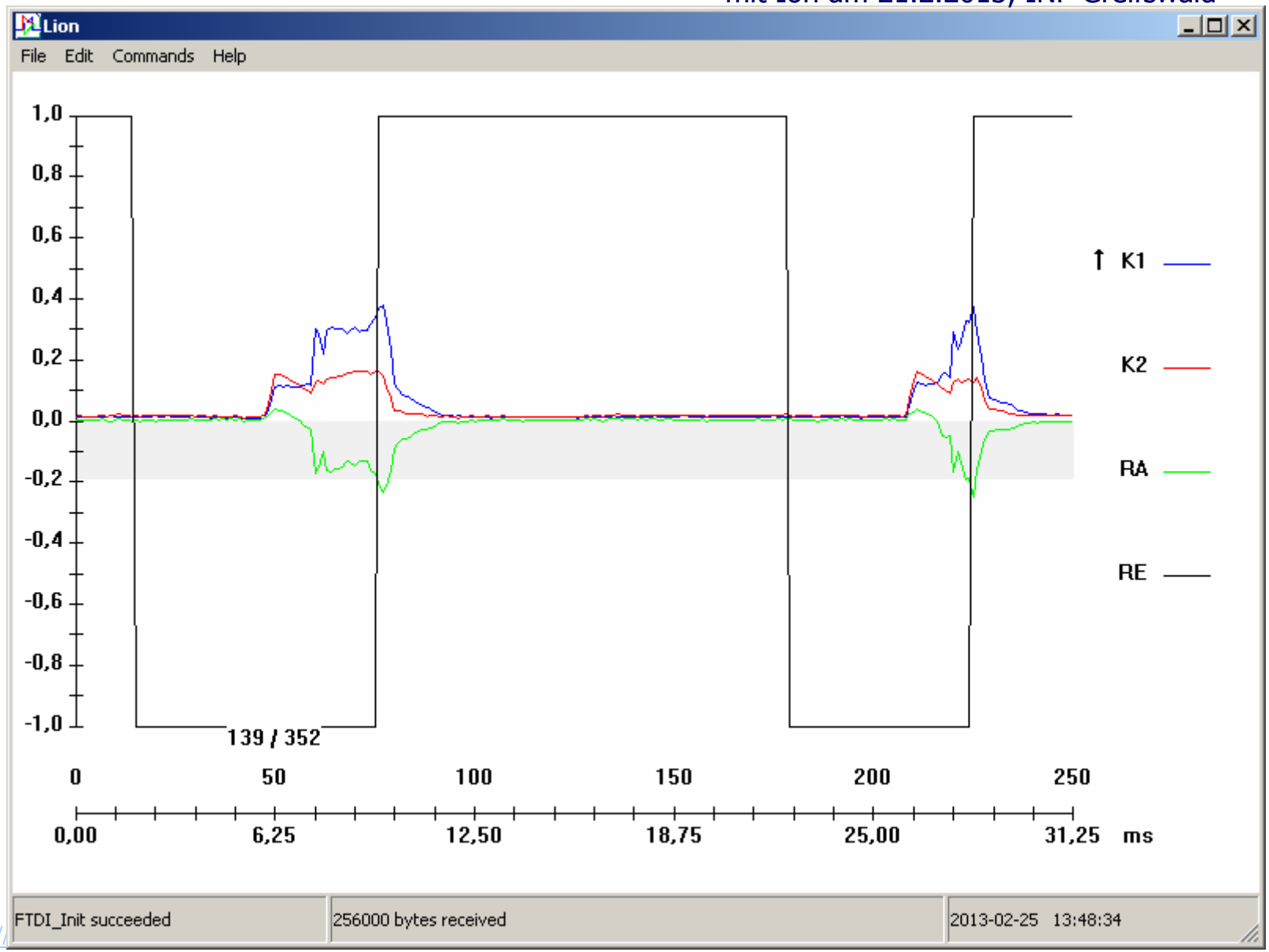
Einstellungen und Daten abspeichern

- Aufgenommene Daten können abgespeichert werden mit File/Save Data
- Einstellungen der Windows-Oberfläche werden im File Lion.ini gespeichert
- Einstellungen von Schweißparametern werden im Ion mit Command/Burn gespeichert
- Schweißparameter und Windows-Parameter werden auch im Record-File gespeichert
- Speichern Sie ein Bild der jeweiligen Windows-Oberfläche über die Windows-DDE Funktion ALT/Print (auf der PC-Tastatur) in der Windows-Zwischenablage
- Wechseln Sie zu einem Photobearbeitungsprogramm und drücken Sie CTRL/V um das Bild zu öffnen



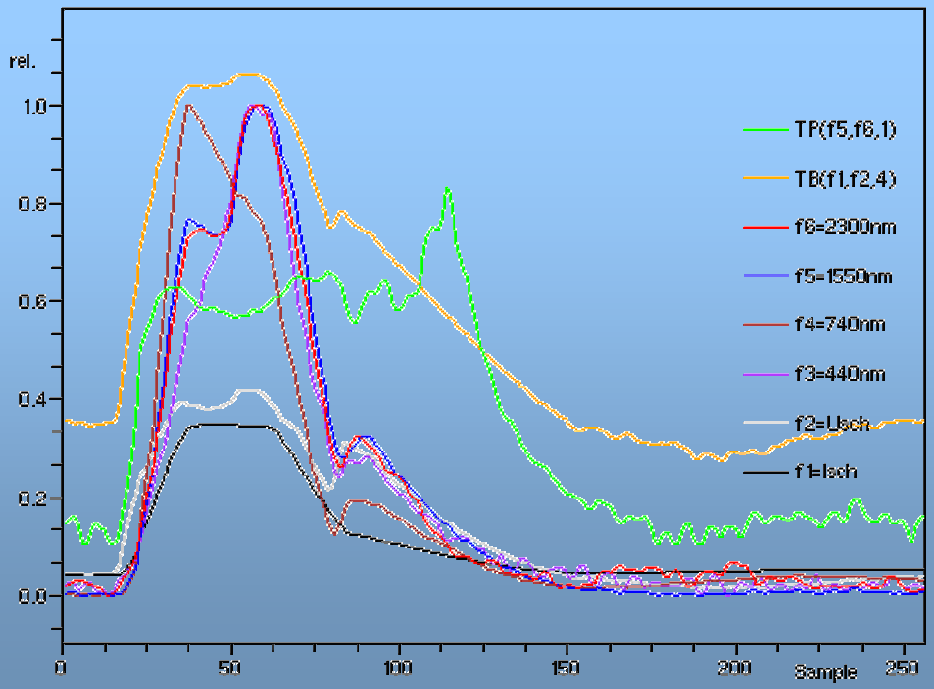
Ergebnis

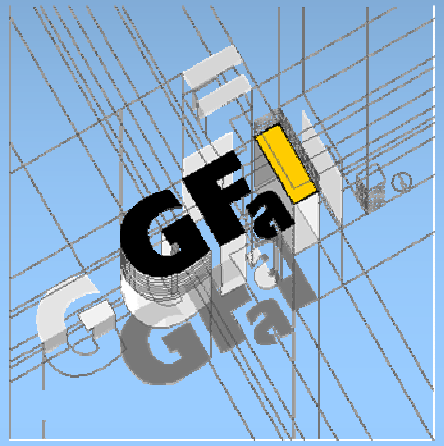
Bild der ersten, spektral geregelten Schweißung
mit Ion am 21.2.2013, INP Greifswald



Hardware

- Baugruppen
- Zubehör





Dr. G. Heinz, GFaI e.V.
Volmerstr.3
12489 Berlin
Tel. +49 (30) 814563-490
Fax. -302
www.gfai.de/~heinz
heinz@gfai.de

Bestandteile Experimentierset ION

Downloads siehe
http://www.gfai.de/~heinz/techdocs/index.htm
und
http://www.gfai.de/spspba/
(User: SPS, Passwort: DAsTg4-j3)

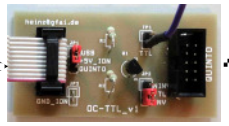
PC

USB

USBdux-Adapter

"Bypass"

OC-TTL-Wandler



Strom
an/aus

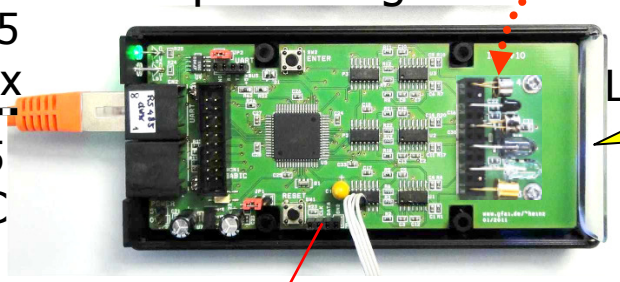
Slave

Simulator
DAC2x4BIT

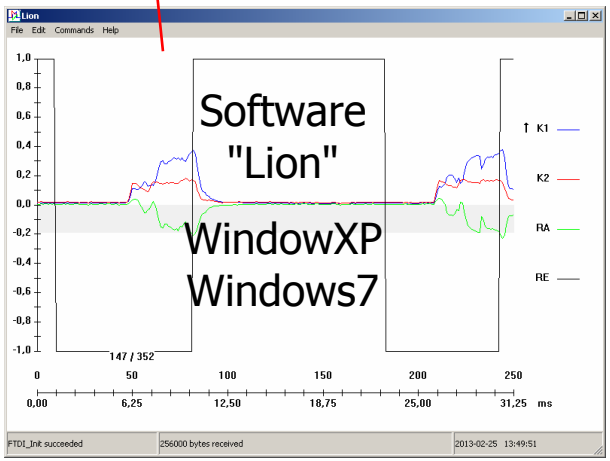
Spektralregler ION

RS485
duplex

RJ45
8P8C

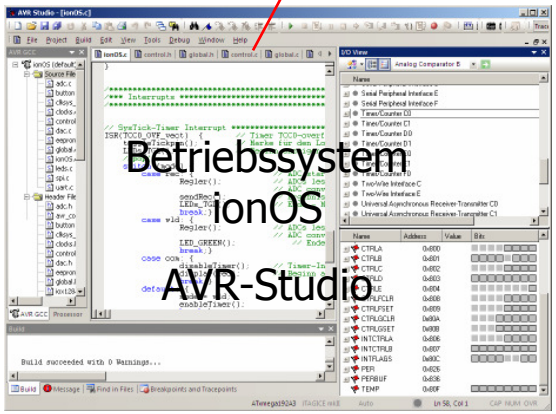


Licht



Software
"Lion"

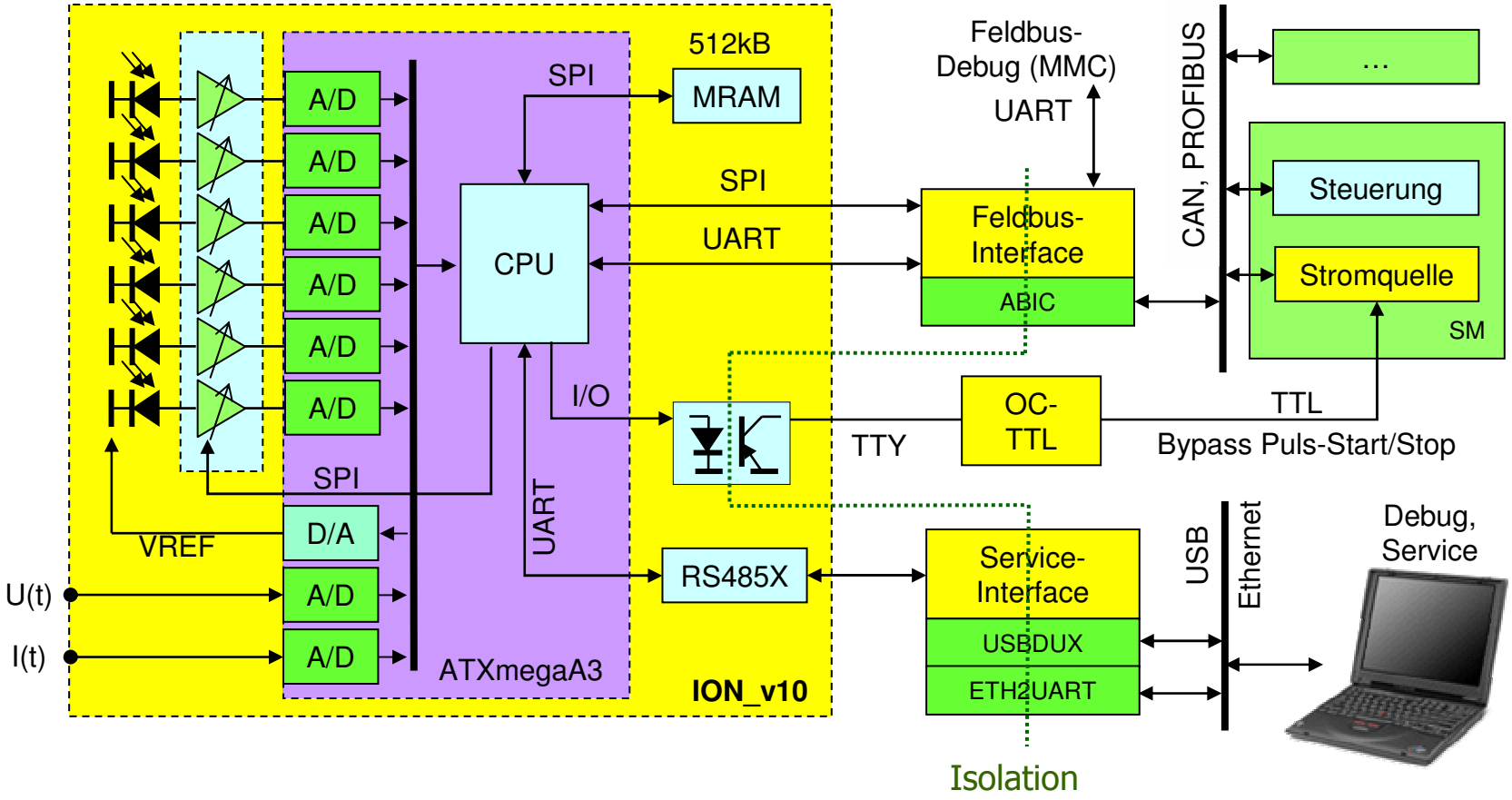
WindowXP
Windows7



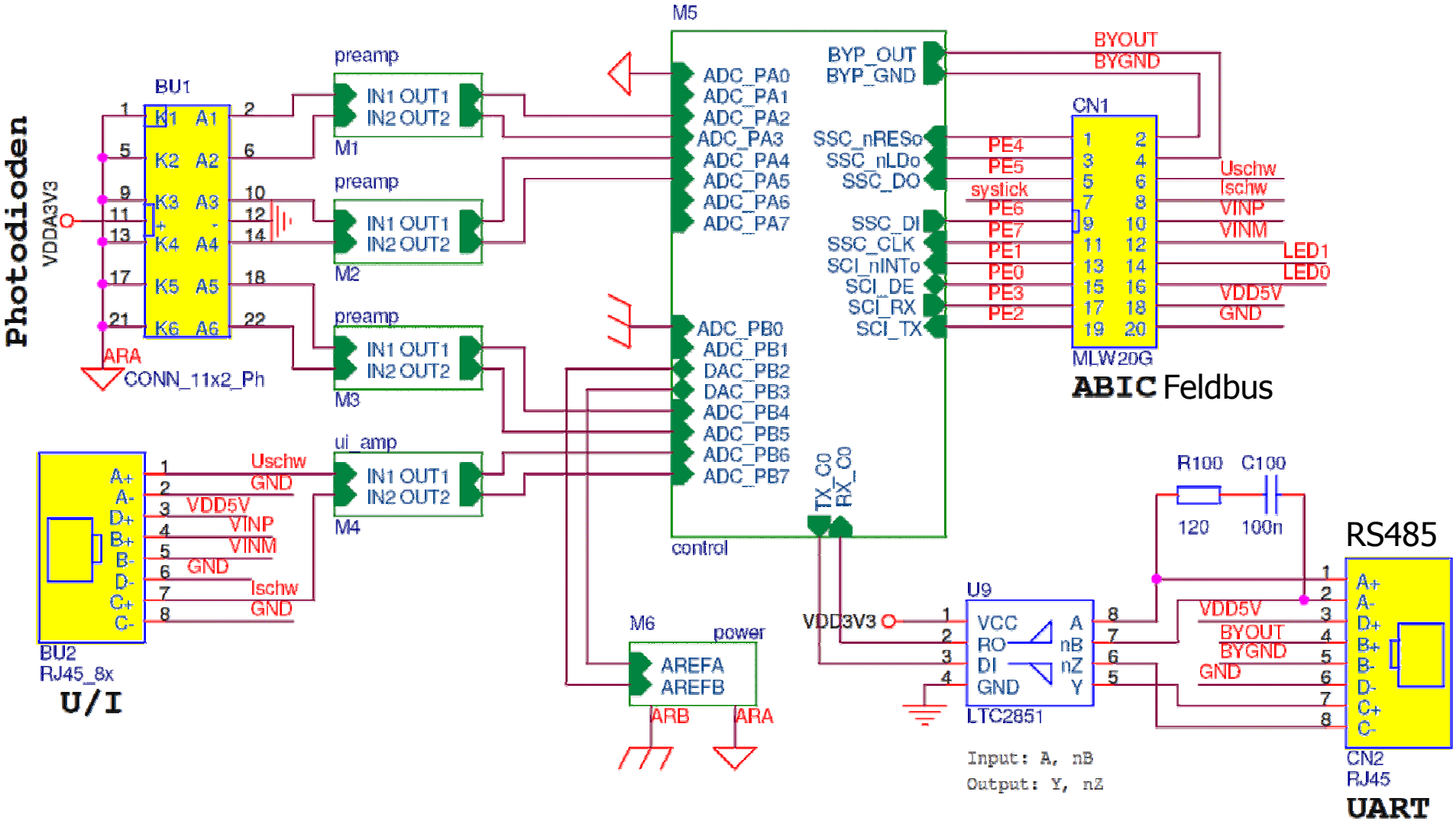
Betriebssystem
ionOS

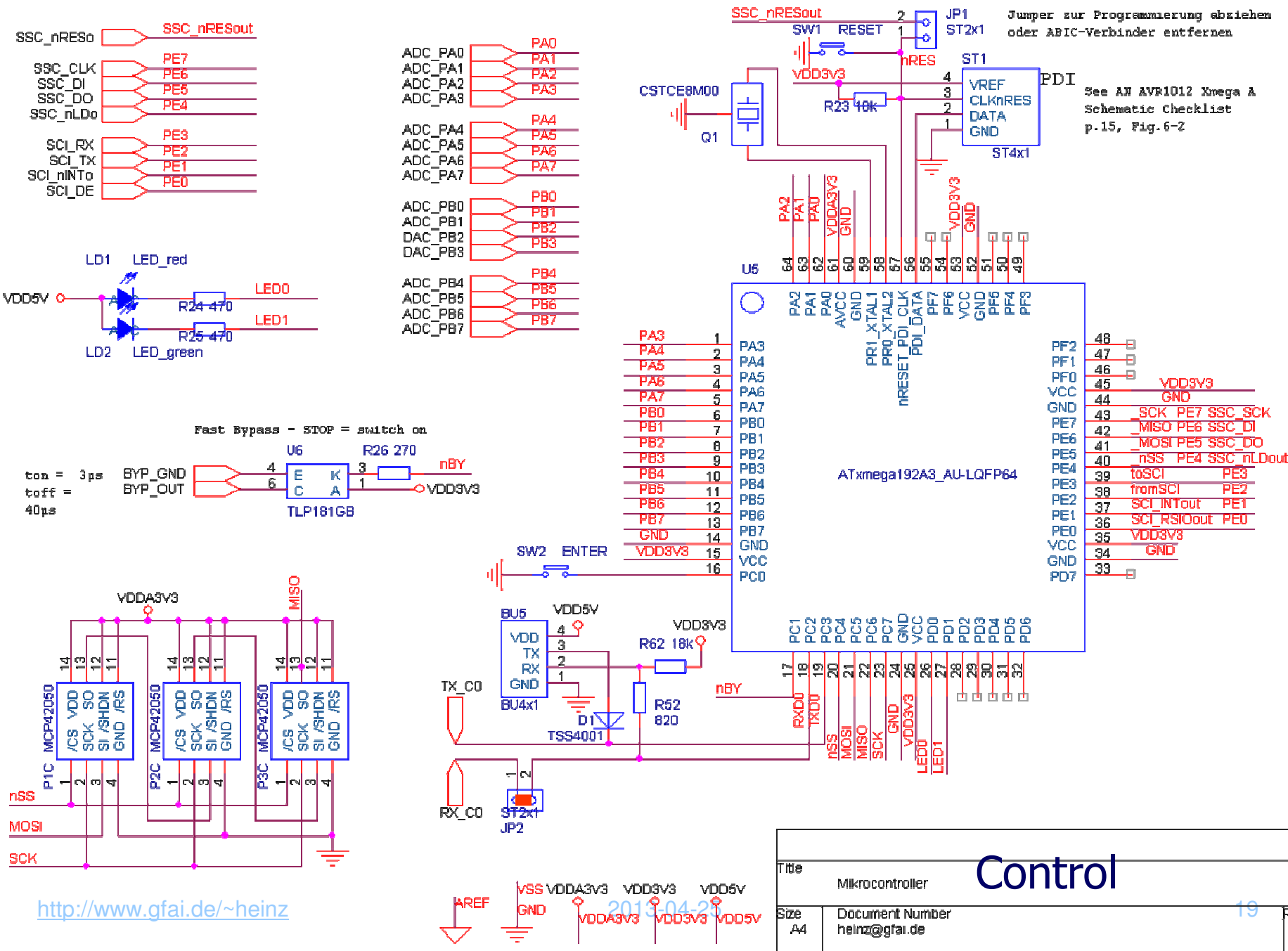
AVR-Studio

Baugruppen

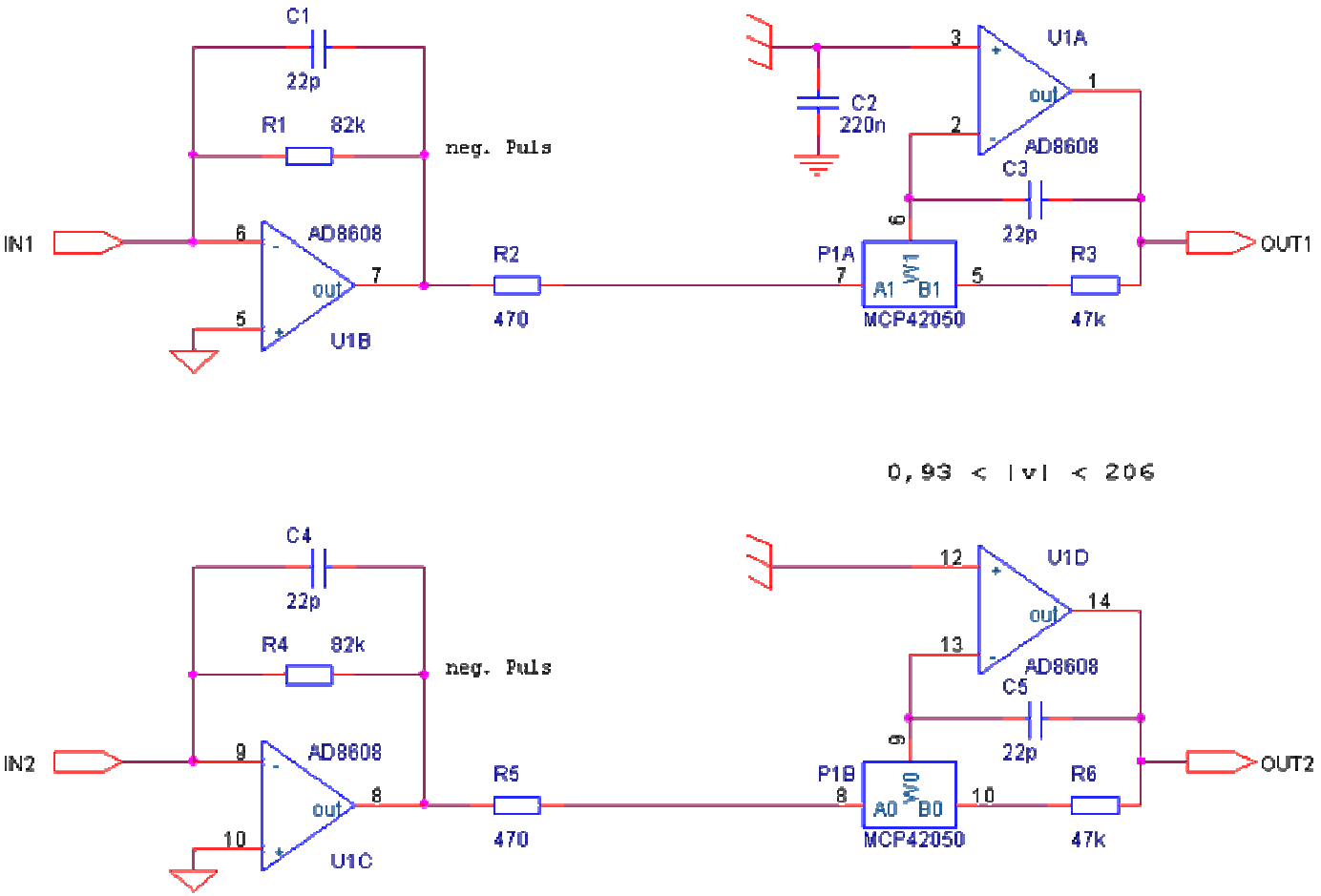

512kB
MRAM
SPI
A/D
A/D
A/D
A/D
A/D
A/D
CPU
SPI
UART
I/O
SPI
UART
D/A
VREF
U(t)
I(t)
A/D
A/D
ATXmegaA3
ION_v10
RS485X
Feldbus-
Debug (MMC)
UART
Feldbus-
Interface
ABIC
CAN, PROFIBUS
…
Steuerung
Stromquelle
SM
OC-
TTL
TTY
TTL
Bypass Puls-Start/Stop
Service-
Interface
USBDUX
ETH2UART
USB
Ethernet
Debug,
Service
Isolation

Ion I/Os

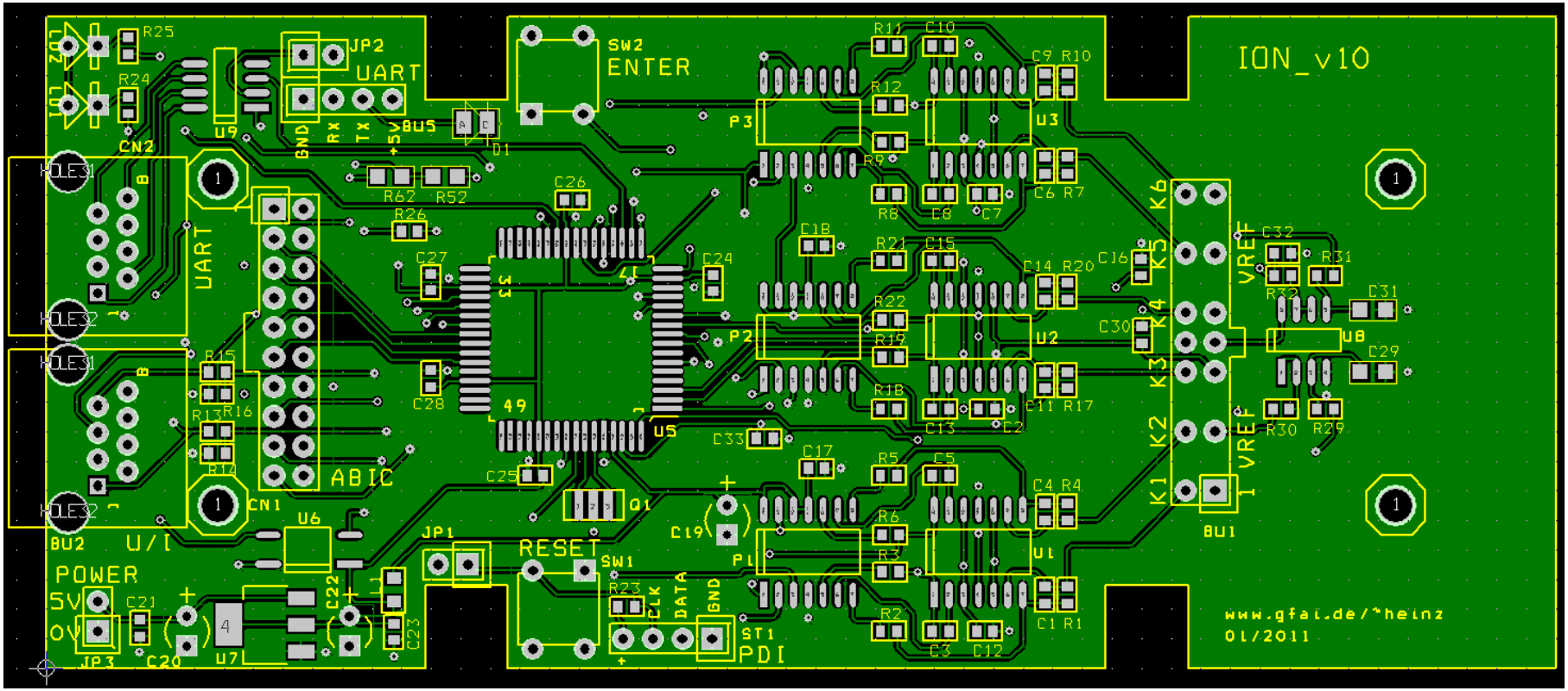

Photodioden
VDDA3V3
BU1
K1 A1
K2 A2
K3 A3
K4 A4
K5 A5
K6 A6
ARA
CONN_11x2_Ph
preamp
IN1 OUT1
IN2 OUT2
M1
M2
M3
M4
ui_amp
M5
ADC_PA0
ADC_PA1
ADC_PA2
ADC_PA3
ADC_PA4
ADC_PA5
ADC_PA6
ADC_PA7
ADC_PB0
ADC_PB1
DAC_PB2
DAC_PB3
ADC_PB4
ADC_PB5
ADC_PB6
ADC_PB7
TX_C0
RX_C0
control
BYP_OUT
BYP_GND
BYOUT
BYGND
SSC_nRESo
SSC_nLDo
SSC_DO
SSC_DI
SSC_CLK
SCI_nINTo
SCI_DE
SCI_RX
SCI_TX
PE4
PE5
systick
PE6
PE7
PE1
PE0
PE3
PE2
CN1
MLW20G
ABIC Feldbus
Uschw
Ischw
VINP
VINM
LED1
LED0
VDD5V
GND
BU2
RJ45_8x
U/I
M6
power
AREFA
AREFB
ARB
ARA
VDD3V3
U9
VCC
RO
DI
GND
A
nB
nZ
Y
LTC2851
Input: A, nB
Output: Y, nZ
R100
C100
120
100n
RS485
CN2
RJ45
UART


SSC_nRESout
SSC_CLK
SSC_DI
SSC_DO
SSC_nLDo
SCI_RX
SCI_Tx
SCI_nINTo
SCI_DE
PE7
PE6
PE5
PE4
PE3
PE2
PE1
PE0
ADC_PA0
ADC_PA1
ADC_PA2
ADC_PA3
ADC_PA4
ADC_PA5
ADC_PA6
ADC_PA7
ADC_PB0
ADC_PB1
DAC_PB2
DAC_PB3
ADC_PB4
ADC_PB5
ADC_PB6
ADC_PB7
PA0
PA1
PA2
PA3
PA4
PA5
PA6
PA7
PB0
PB1
PB2
PB3
PB4
PB5
PB6
PB7
LD1
LED_red
LED0
R24 470
LED1
R25 470
LD2
LED_green
VDD5V
Fast Bypass - STOP = switch on
ton = 3ps
toff = 40ps
BYP_GND
BYP_OUT
U6
TLP181GB
R26 270
nBY
VDD3V3
SW2 ENTER
VDDA3V3
MISO
MCP42050
/CS VDD SCK SO SI /SHDN GND /RS
P1C
P2C
P3C
nSS
MOSI
SCK
BU5
BU4x1
VDD5V
TX
RX
GND
R62 18k
TX_C0
RX_C0
D1
TSS4001
R52
820
JP2
ST2x1
SSC_nRESout
SW1 RESET
nRES
JP1
ST2x1
Jumper zur Programmierung abziehen
oder ABIC-Verbinder entfernen
VDD3V3
R23 10k
CSTCE8M00
Q1
ST1
VREF
CLKnRES
DATA
GND
ST4x1
PDI
See AN AVR1012 Xmega A
Schematic Checklist
p.15, Fig.6-2
U5
ATxmega192A3_AU-LQFP64
PA3 PA4 PA5 PA6 PA7 PB0 PB1 PB2 PB3 PB4 PB5 PB6 PB7 GND VDD3V3
PA2 PA1 PA0 AVCC GND PR1_XTAL1 PR0_XTAL2 nRESET_PDI_CLK PDI_DATA PF7 PF6 VCC GND PF5 PF4 PF3
PF2 PF1 PF0 VCC GND PE7 PE6 PE5 PE4 PE3 PE2 PE1 PE0 VCC GND PD7
PC1 PC2 PC3 PC4 PC5 PC6 PC7 GND VCC PD0 PD1 PD2 PD3 PD4 PD5 PD6
VDD3V3
GND
_SCK PE7 SSC_SCK
_MISO PE6 SSC_DI
_MOSI PE5 SSC_DO
nSS PE4 SSC_nLDout
toSCI PE3
fromSCI PE2
SCI_INTout PE1
SCI_RSIOout PE0
RXD0 TXD0 nSS MOSI MISO SCK GND VDD3V3 LED0 LED1
AREF
VSS VDDA3V3 VDD3V3 VDD5V
GND VDDA3V3 VDD3V3 VDD5V
Title
Mikrocontroller
Control
Size A4
Document Number
heinz@gfai.de
Rev 1.0
Date. Friday, September 23, 2011
Sheet 6 of 7

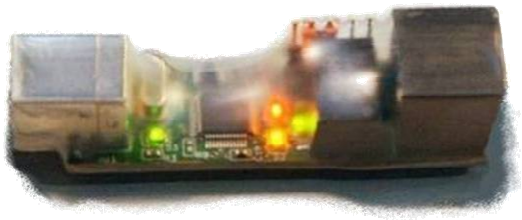
Preamps


C1
22p
R1
82k
neg. Puls
IN1
6
AD8608
7
out
5
U1B
R2
470
C2
220n
3
U1A
1
out
2
AD8608
C3
22p
6
P1A
7
A1
W1
B1
MCP42050
5
R3
47k
OUT1
0,93 < |v| < 206
C4
22p
R4
82k
neg. Puls
IN2
9
AD8608
8
out
10
U1C
R5
470
12
U1D
14
out
13
AD8608
C5
22p
9
P1B
8
A0
W0
B0
MCP42050
10
R6
47k
OUT2
VDDA3V3
VSS
GND
VDD
Title
Preamplifier
Size
A4
Document Number
heinz@gfai.de
Rev
1.0
Date:
Friday, February 25, 2011
Sheet
2
of
7

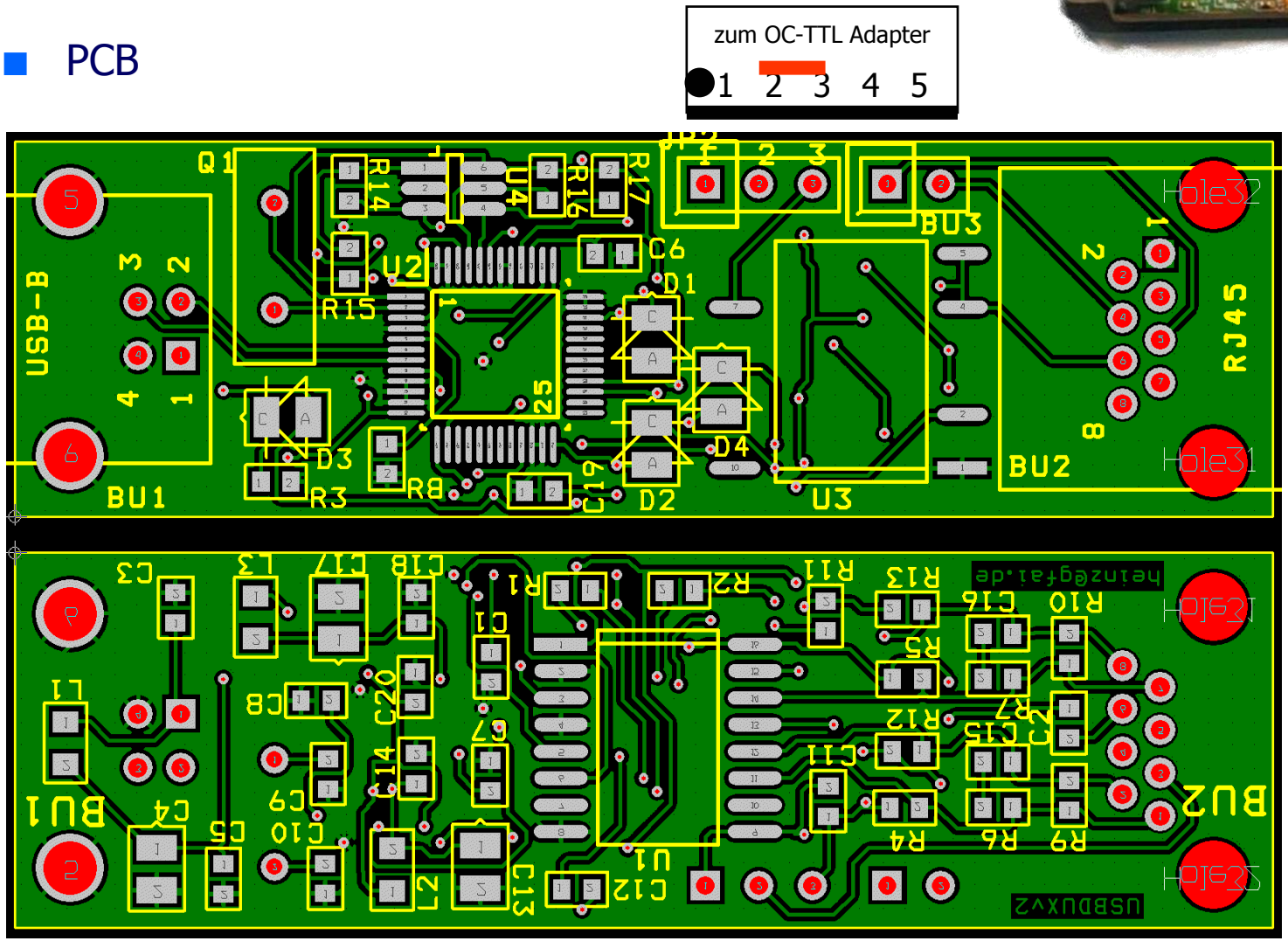
Bauelementelage Ion_v10



USB-RS485 duplex Konverter USBdux



- PCB



LEDs:

D1 Tx

D2 Rx

D3 USB-Pwr

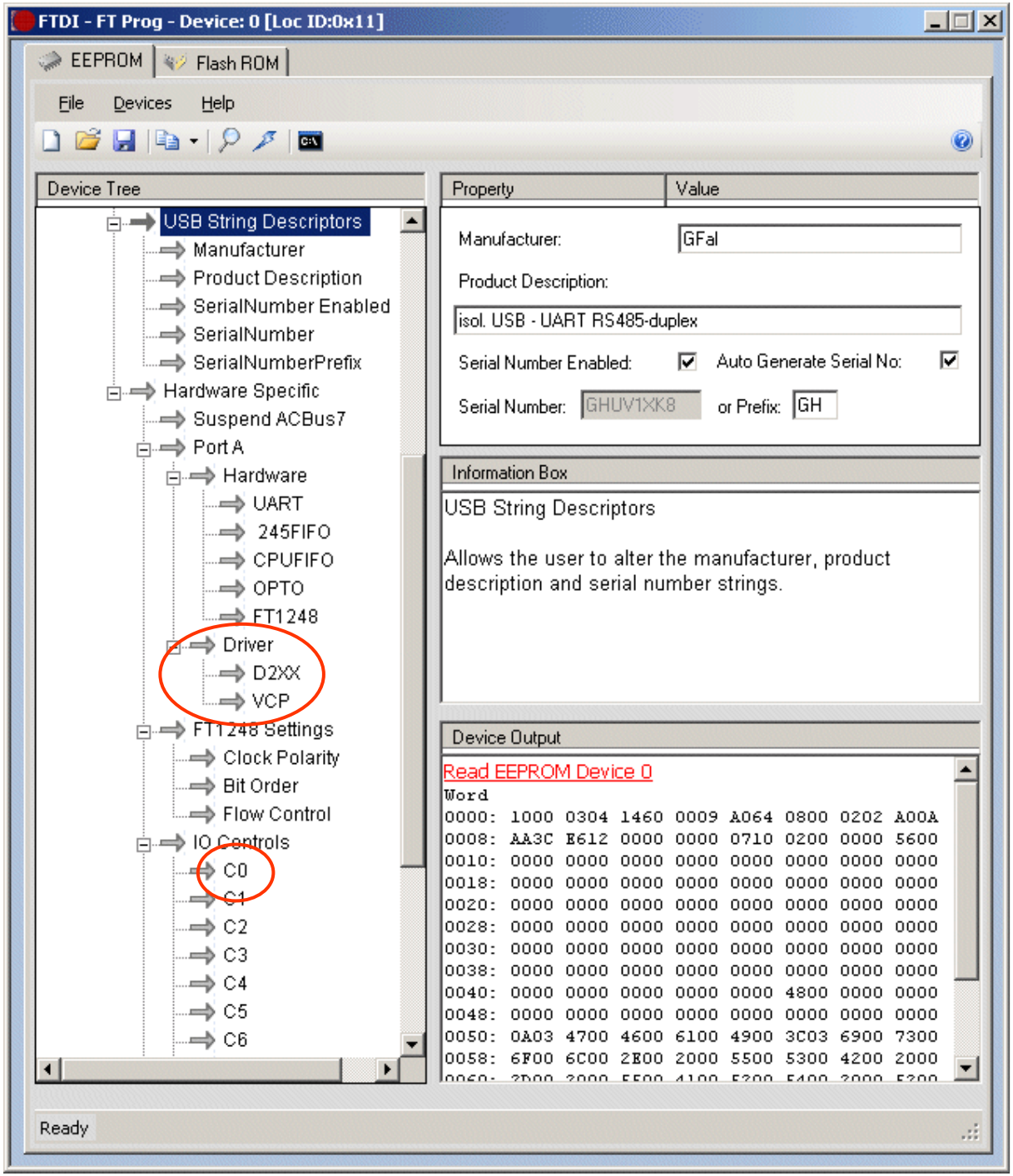
D4 Isol-Pwr

USBDUX Setings

- FT232H max. 12 Mb/s
- RX und TX je eine RS485 (duplex)
- Atxmega leistet max. 4 Mb/s

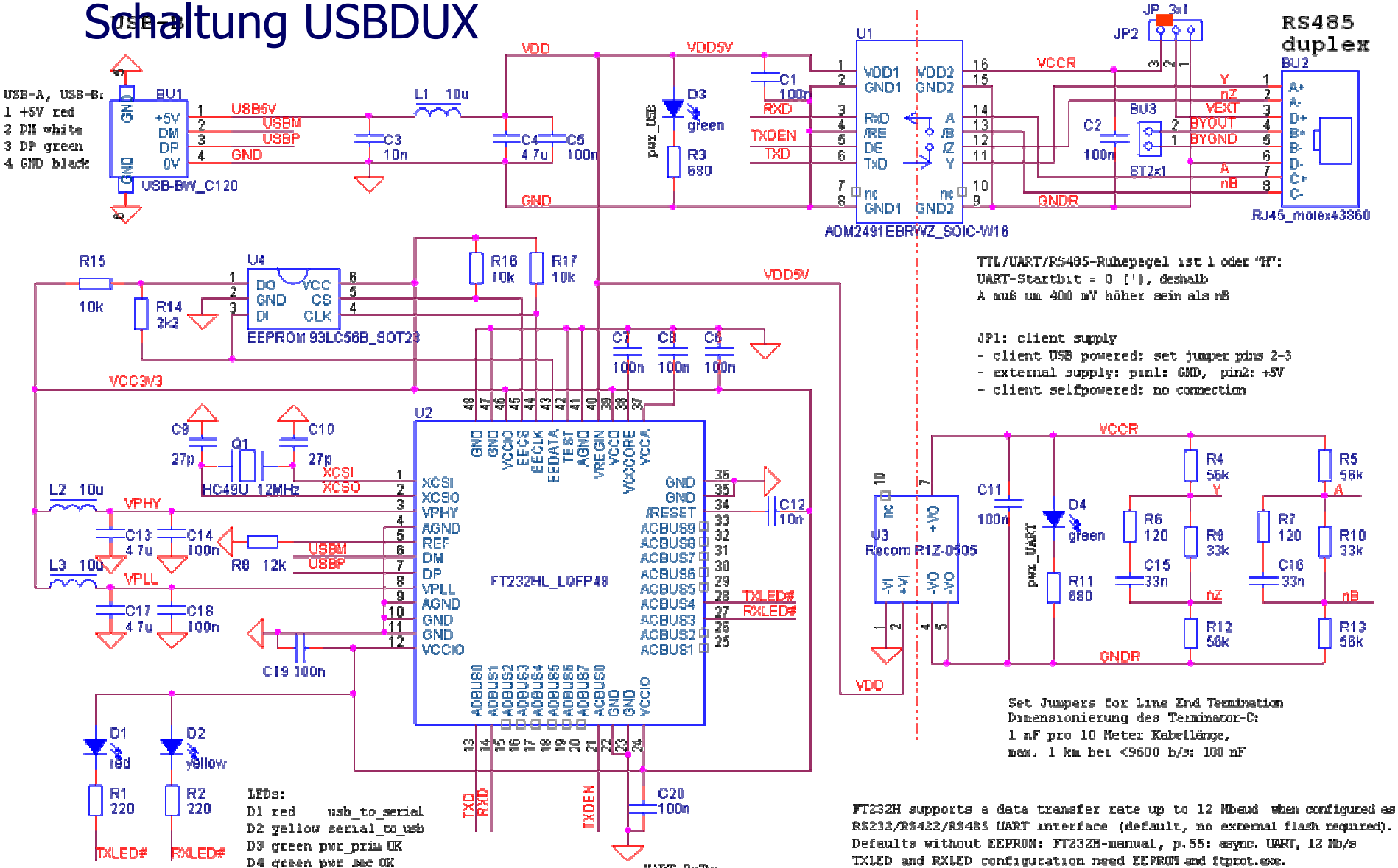
FT Prog:

- Unter Hardware Specific/PortA/Driver muß D2XX gesetzt sein
- Unter IO Controls ist C0 auf Drive1 zu setzen



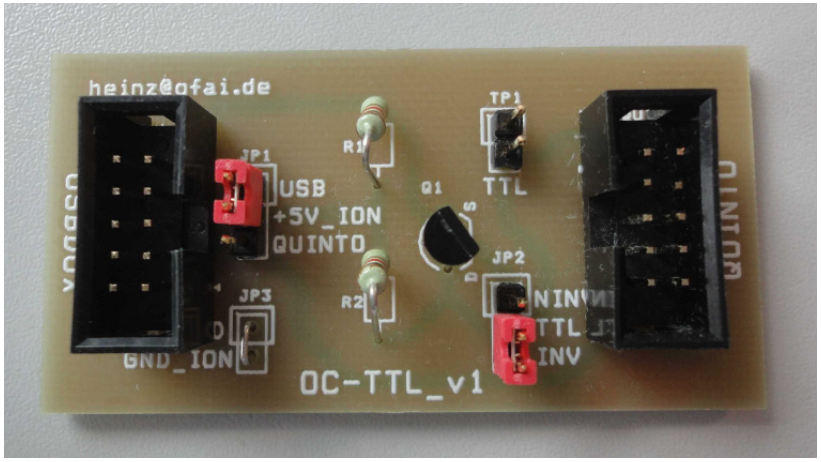
http://www.gfai.de/~heinz

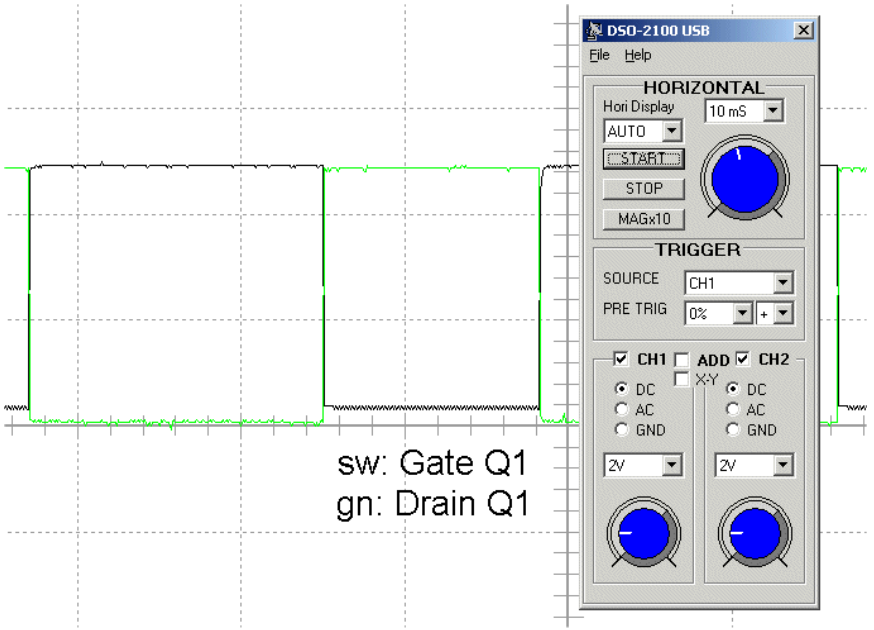
Schaltung USBDUX



Optokoppler/TTL-Interface OC-TTL

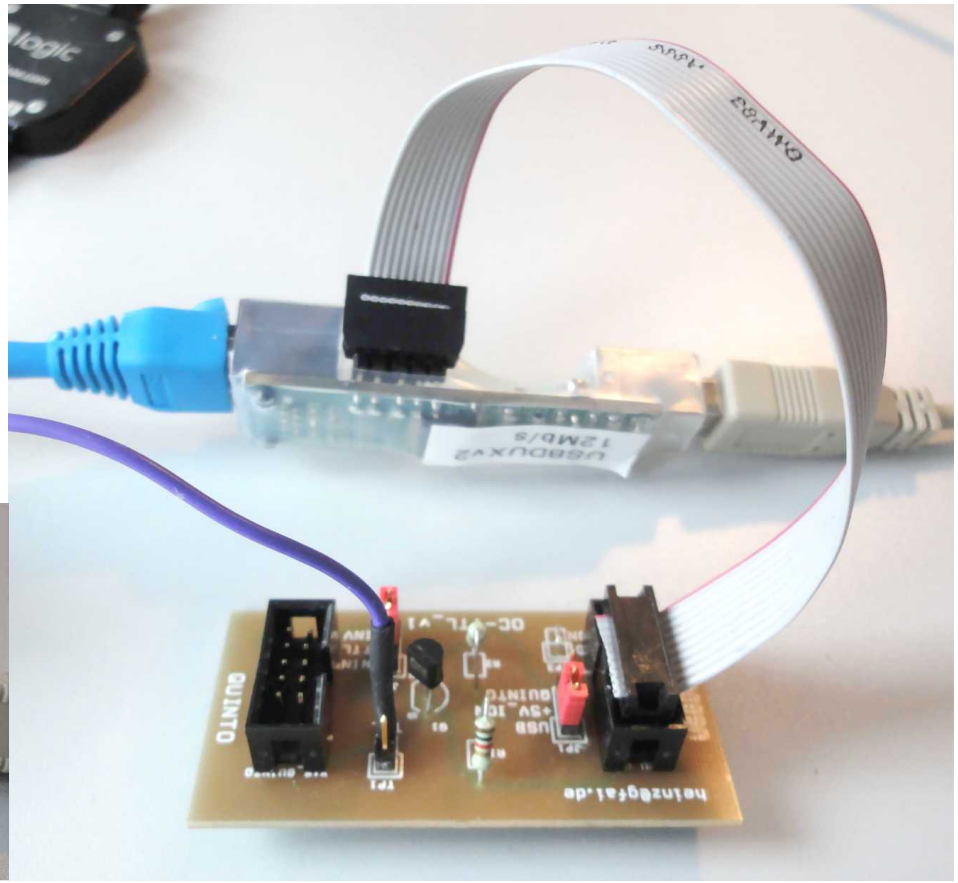
- Das Start/Stop-Signal an die Schweißmaschine wird vom Ion optisch entkoppelt ausgegeben ("Bypass")
- In der Schweißmaschine ist ein Wandler nötig, der das Optokoppler-Signal (OC) auf ein TTL-Signal umsetzt
- In der Cloos Quinto GLC403 wird mit dem TTL-Signal der Eingang X15 angesteuert



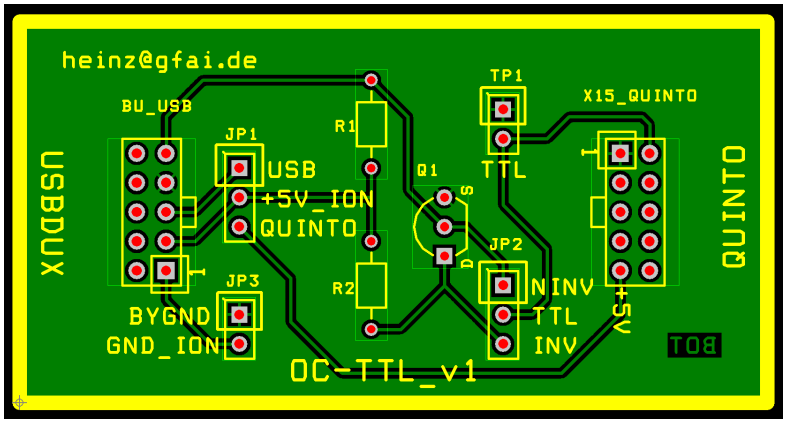


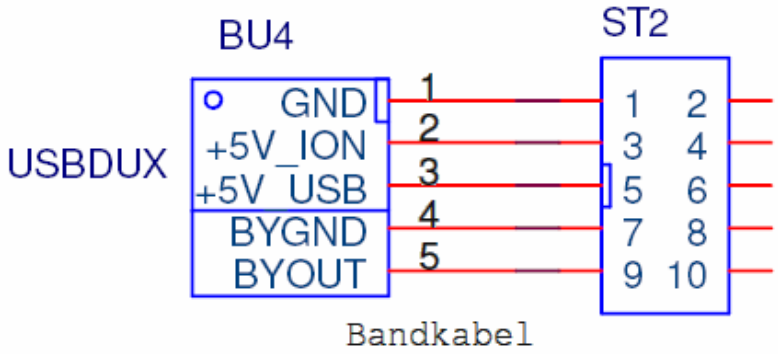
OC-TTL

- An JP1 kann die Versorgungsquelle für Ion eingestellt werden
- An TP1 ist das Start/Stop-Signal zu Testzwecken abgreifbar
- Es wird empfohlen, den Wandler invertiert zu benutzen (JP2 Brücke zwischen Pins 2-3)
- Zur Verbindung mit USBDUX kann ein 10- zu 10-poliges Kabel benutzt werden, siehe Bilder
- Achtung: bei Verbindung zum USBDUX Polung der Verbindung beachten

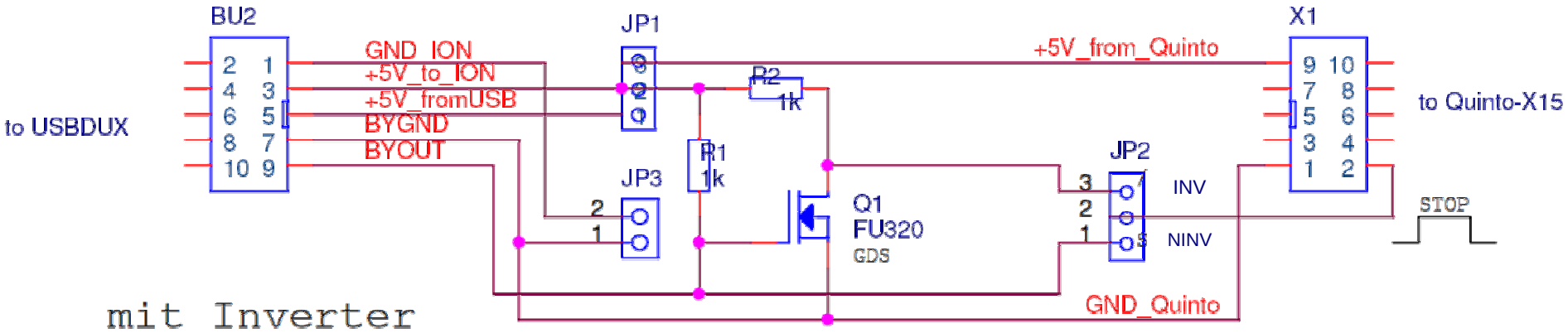


OC-TTL_v1 Schaltung





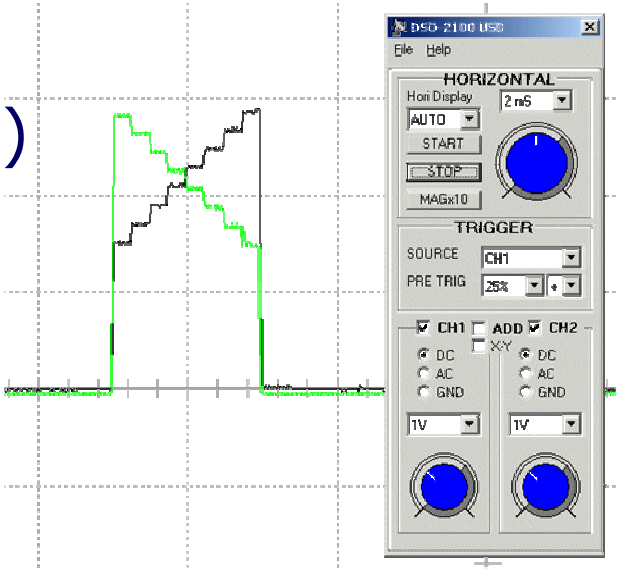
JP1:
2-3 powered by Quinto (default)
1-2 powered by USB (test only)



JP3:
1-2: short GND Quinto <-> ION

JP2:
1-2 noninverted input
2-3 inverted input (prefered)

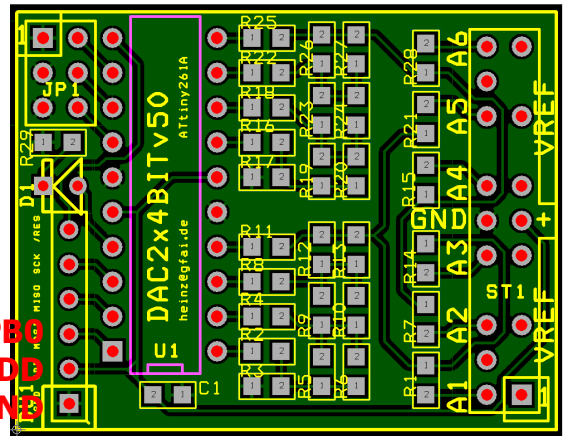
Zeitfunktionsgenerator DAC2X4BIT (2chl)

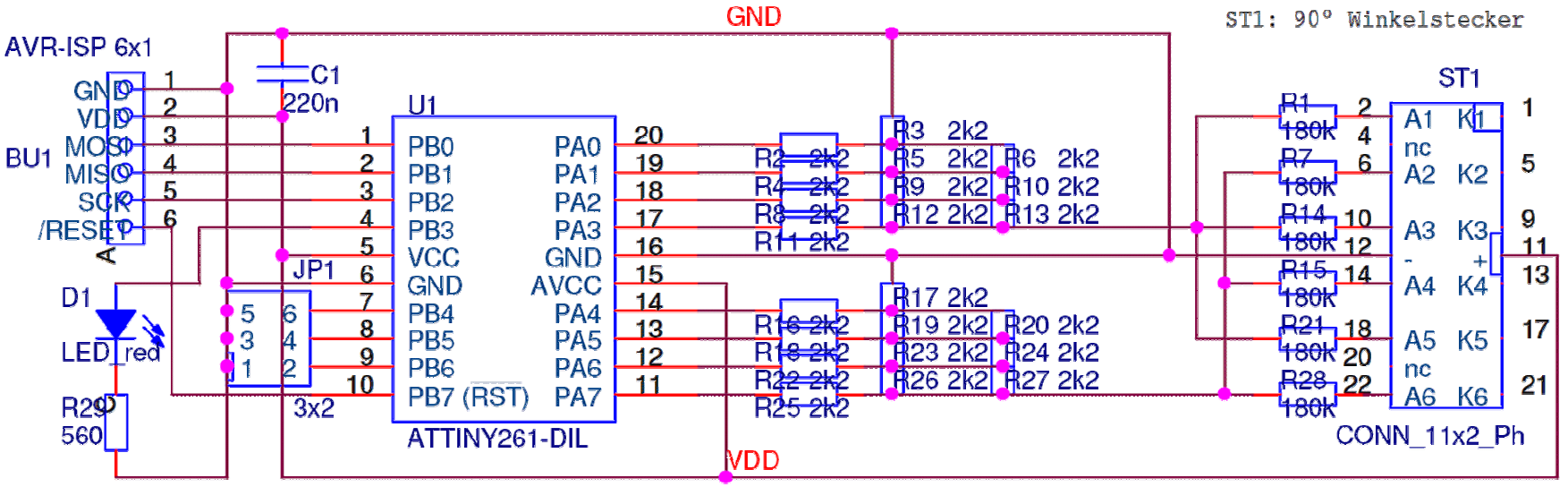


- Ziel: Bereitstellung von zwei Plasma-ähnlichen Zeitfunktionen am Photodiodenstecker
- Von zwei Digital-Analog-Convertern (DAC) werden zwei Zeitfunktionen generiert, die über 180 kOhm Widerstände auf je 3 Kanäle wirken:
- DAC1: K1, K3, K5
- DAC2: K2, K4, K6
- Amplitudenauflösung jedes DAC: 16 Stufen (4 Bit)
- An Jumper JP1 können mit drei Kurzschlußsteckern 2^3 = 8 verschiedene Zeitfunktionen gesetzt werden
- BU1 stellt einen AVR-ISP MKII Programmiereingang dar
- Beschreibung und Quellen können auf der Techdocs-Seite heruntergeladen werden
- Stellen Sie Lion auf "Preview" und testen Sie mit beiliegenden roten Stecker an JP1 andere Pins
- Ausgegebene Zeitfunktionen liegen nicht exakt auf VREF-Mittenpotential, sondern zeigen einheitliches Offset

Simulator DAC2x4BIT

- Der DAC simuliert Photodioden mit einem Strom von maximal je 8,8 µA pro Kanal (180 kOhm)
- Anschluß des STOP-Signals von OC-TTL an PB0
- Stecker ST1 ist identisch zum Photodiodenarray

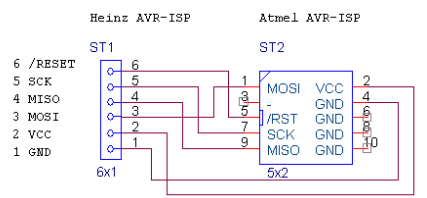




Achtung: Gefahr für ION!
VDD am AVRISP ist 5 Volt, VCC vom ION ist 3,3 Volt! Entweder AVRISP, oder ION stecken!

VDD = 3,3V: ia = +/- 8,84µA pro Kanal

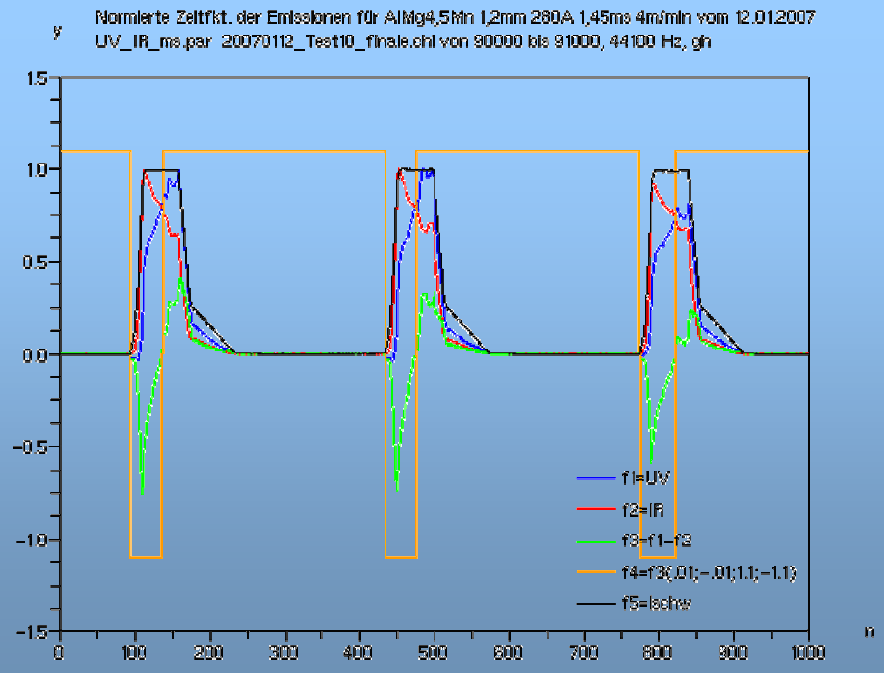
AVR ISP (MKII) Adapter

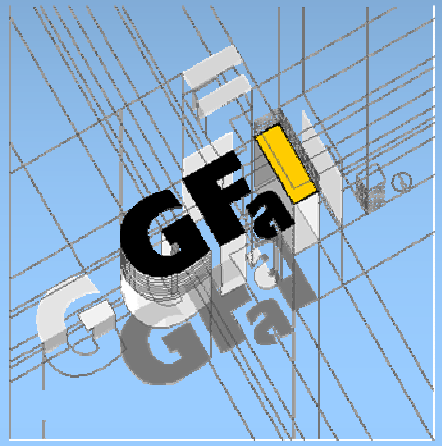


Title	DAC 2x4 Bit - PCB-Version	
Size A4	Document Number heinz@gfai.de	Rev 50

Software

- Arbeitsmoden
- Befehlssyntax
- Regler
- Programmierumgebung

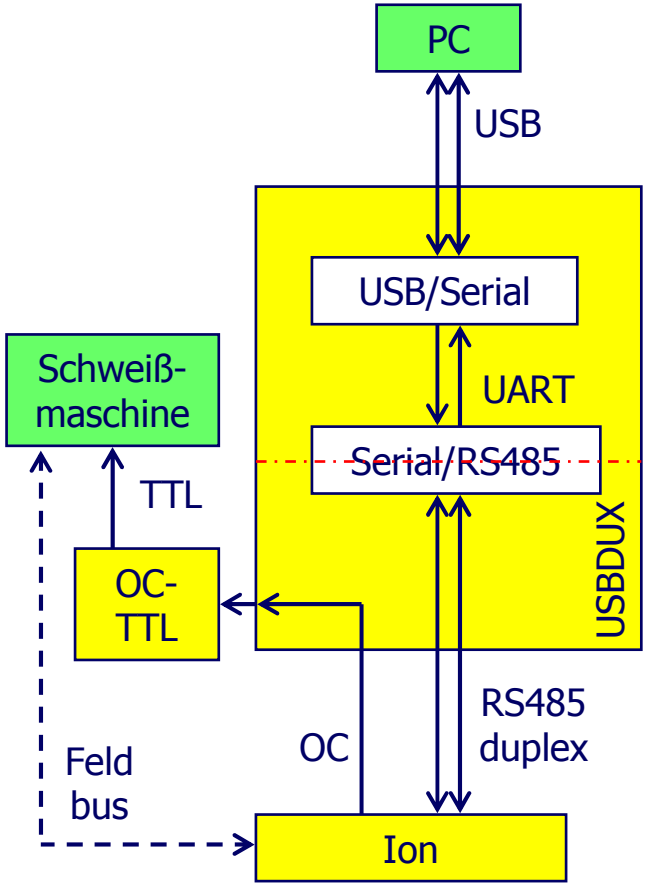




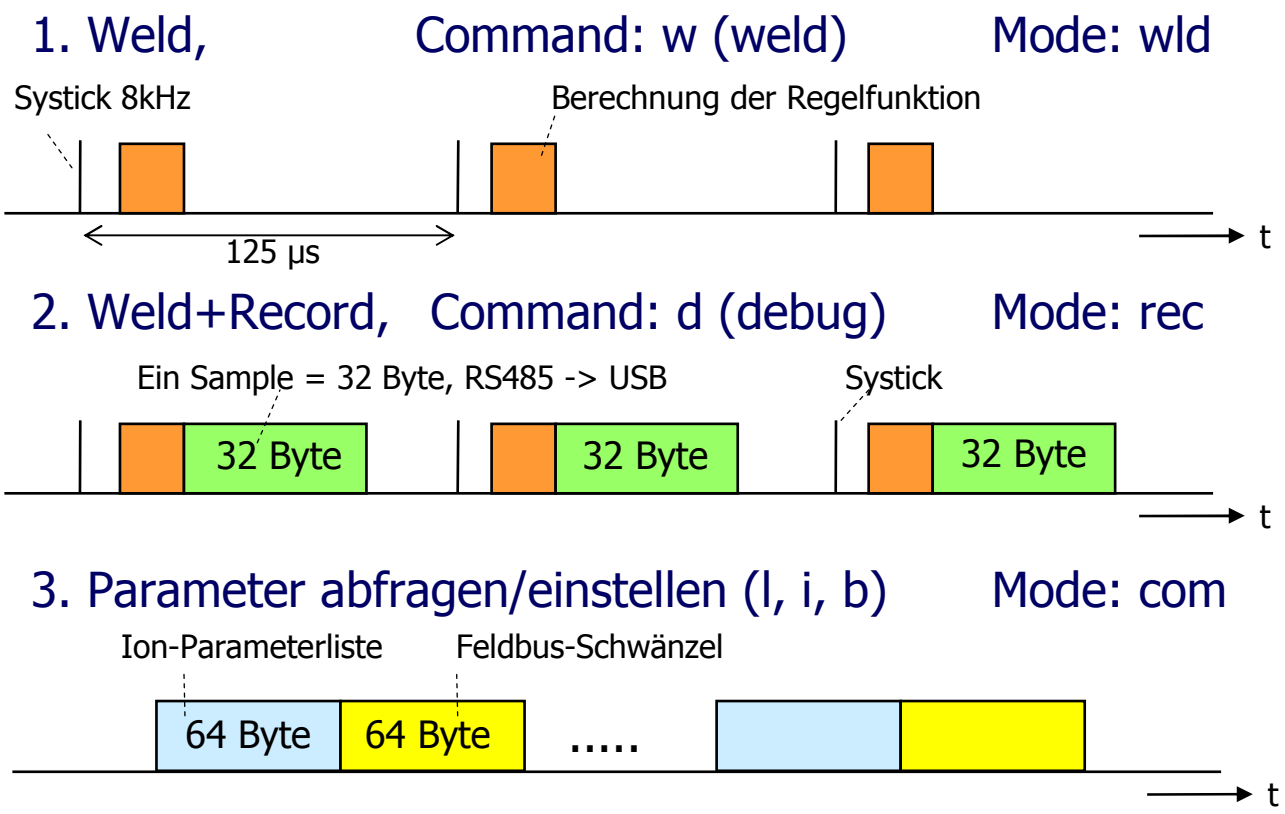
Dr. G. Heinz, GFaI e.V.
Volmerstr.3
12489 Berlin
Tel. +49 (30) 814563-490
Fax. -302
www.gfai.de/~heinz
heinz@gfai.de

Prinzip der Kommunikation

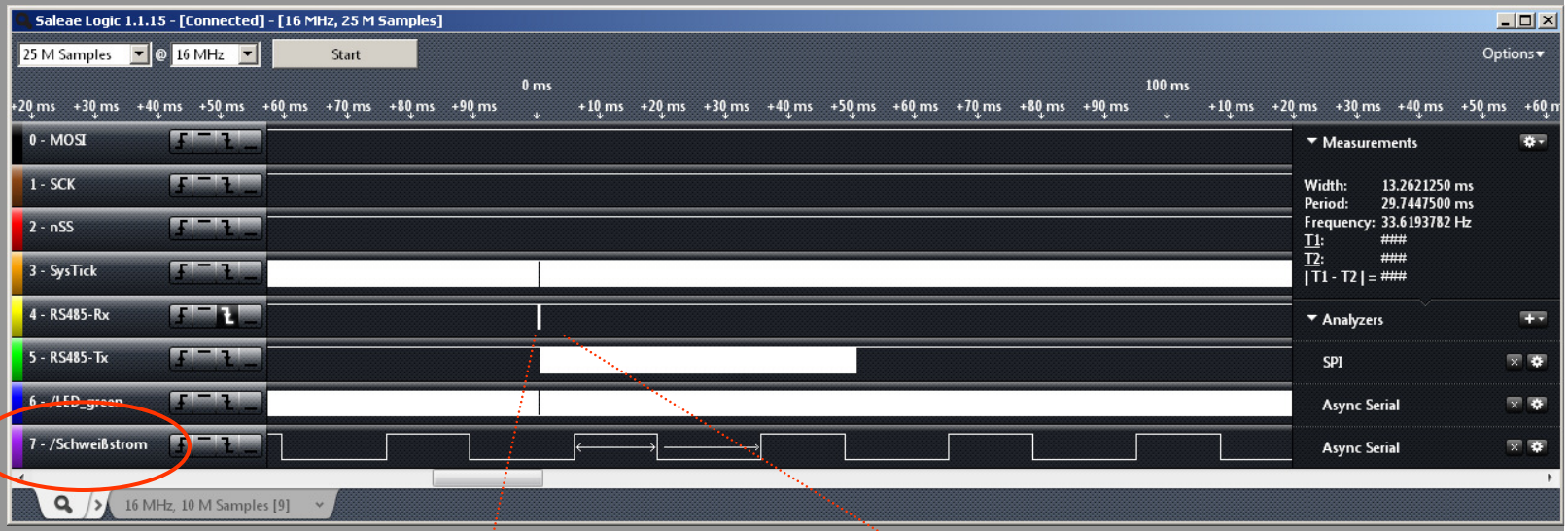
- Zeitfunktionen (REC) und Parameter (PRL) werden über RS485 und USB mit dem PC ausgetauscht
- USB-Driver-Interface ist FTDI D2XX
- Die Steuerung der Schweißmaschine erfolgt frei von Bus-Delays als "Bypass" über Optokoppler (OC)
- Der Isolator-IC nimmt eine Wandlung serial-UART in RS485-duplex vor
- Das Ethernet-Interface (ETH2UART) besitzt dieselbe Schnittstelle



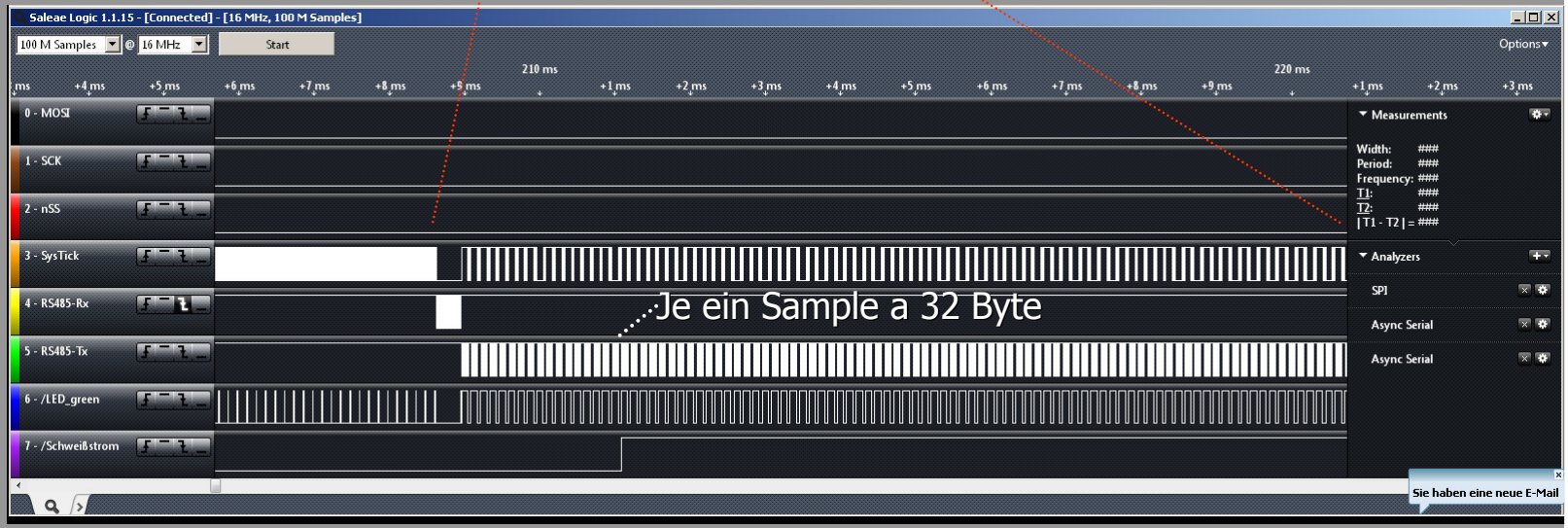
Arbeitsmodi ionOS



Beispiel zur Kommunikation



Preview: 50 ms werden auf den PC hochgeladen, unten: Zoom in



Beispiel WELD→REC

- Records a 32 Byte: Header + 13 Kanäle

Datenformate zum Austausch ionOS - Lion

Recording (pro Sample)

- REC_struct 32 Byte, hex

Parameterliste

- PAR_struct 64 Byte, hex

Befehls-Rahmen (Rx, Tx)

- FR_struct 128 Byte

Kommunikation ionOS - Lion:

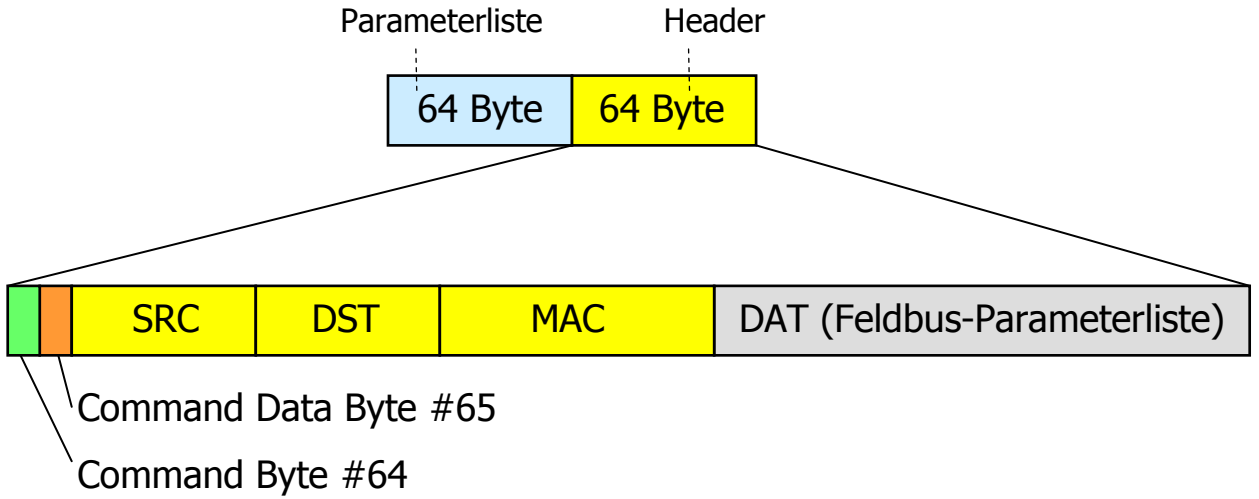
```
enum mode_t  {  // Arbeitsmoden
    wld,        // weld only, systick on, no records out
    rec,        // weld + record (on fast, duplex RS485), systick on
    com         // communicate, systick off
};

enum cmd_t {    // Befehlssatz
    b     = 0x62,  // burn parameter list (save in EEPROM)
    d     = 0x64,  // debug: weld + record, Parameter: uint8_t (Sekunden)
    f     = 0x66,  // factory reset, init default parlist
    i     = 0x69,  // install new parameter list, volatile in RAM
    l     = 0x6c,  // list all current parameters
    r     = 0x72,  // restart
    w     = 0x77   // weld only, no recording
};
```

Parameterliste PRL

```
// Deklaration //////////////////////////////////////////////////////

typedef struct    // Parameter im EEPROM 2x32 Byte = 64 Byte
{                 // insgesamt 34 Parameter

   uint8_t  ID;   // Ident.-Nr.
   uint8_t  SEC;  // Sekunden Recording
   uint8_t  AO;   // Autooffset Channels [8...1]
   uint8_t  NE;   // Negation Channels [8...1]

   uint16_t FT;   // CPU-Clocks pro Systick
   uint16_t FH;   // Systicks pro Sekunde
   uint16_t NA;   // Nummer des Weld-Algorithmus
   uint16_t NP;   // Nummer der Diodenanordnung

   // Potentiometer
   uint16_t P1;   // Gain Potentiometer Channel 1...6
   uint16_t P2;
   uint16_t P3;
   uint16_t P4;
   uint16_t P5;
   uint16_t P6;
   uint16_t PU;   // Gain U
   uint16_t PI;   // Gain I

   // b.w.
```

```
   // ADC Offset
   int16_t O1;    // Offset channel K1
   int16_t O2;
   int16_t O3;
   int16_t O4;
   int16_t O5;
   int16_t O6;
   int16_t OU;    // Offset channel U
   int16_t OI;    // Offset channel I

   // DAC VREFs
   int16_t VA;    // DAC-output VREF A, ARA
   int16_t VB;    // DAC-output VREF B, ARB

   // Reglereinstellungen
   int16_t G1;                  // frei, passend zu Regelalgorithmus
   int16_t T1;                  //
   int16_t G2;                  //
   int16_t T2;                  //
   int16_t G3;                  //
   int16_t T3;                  //
   int16_t TA;                  //
   int16_t TE;                  //
} par_t;

// Speicherzuweisung ///////////////////////////////////////////////
extern par_t PRL;               // Zugriff z.B. mit PRL.P1
```

Befehlsformat Frame (FR)




```
typedef struct {                        // Befehlsstruktur 128 Byte-Format
    par_t           PRL;        // parameter list, EEPROM-saved, 64 Byte
    uint8_t         C;          // command:  d[0xad], l, w, s[par_t]… Byte #64
    uint8_t         D;          // command data, Byte #65
    uint8_t         SRC[4];     // source address
    uint8_t         DST[4];     // destination address
    uint8_t         MAC[6];     // MAC-address of Ion
    uint8_t         DAT[48];    // free for fieldbus parameters
} frame_t;
extern  frame_t FR;     // Zugriff z.B. FR.PRL, FR.DAT …
```

Implementierung von Regler-Algorithmen

- Es können verschiedene Regelalgorithmen in den µC gebrannt werden
- Diese sind mit einer Nummer (PRL.NA) wählbar
- Die Diodenanordnung ist ebenfalls mit einer Nummer (PRL.NP) ansprechbar
- Jeder Perametersatz kann mit einer spezifischen Identifikationsnummer (PRL.ID) versehen werden
- Diese ID wird automatisch in jeden aufgenommenen Datensatz (REC) als REC.ID übernommen, d.h.
  REC.ID ← PRL.ID
- Es ist vorgesehen, den Regelalgorithmus per Bootlader über die USB-Schnittstelle hochzuladen (noch nicht implementiert)
- Bislang sind Regelalgorithmen nur über die Programmierschnittstelle (AVR-PDI) brennbar

Implementierung eines Regelalgorithmus 1

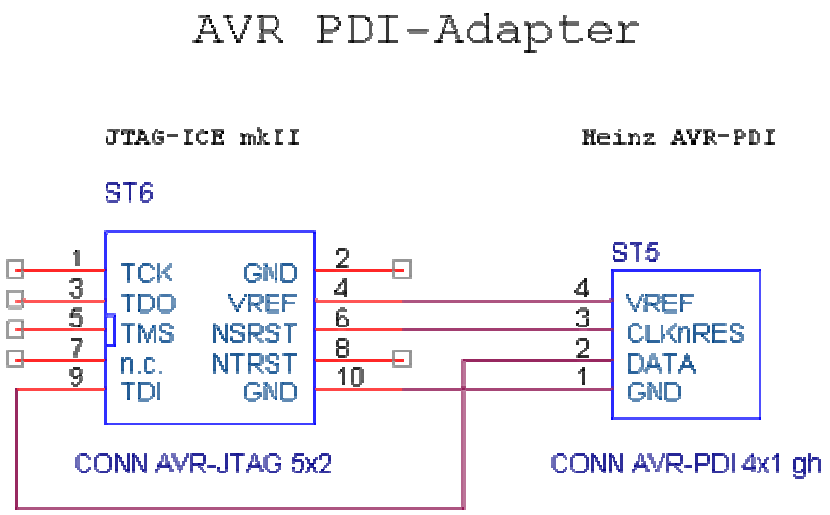
```
void R1(void) {          // Algorithmus Metalldampf-Schweißregler
                         // wird mit jedem Systick-Interrupt ausgeführt
    static uint16_t systick;          // lokaler Systick für Pulsdauer

    // ADC laufen im "Freerun" mit 500 kHz,
    // alle 22µs ist je ein neuer Wert da: 12 bit right adj.
    // ADCs lesen und Offset vorzeichenrichtig korrigieren
            REC.K1 = ADC_K1 + PRL.O1;       // 440nm   EPD440
            REC.K2 = ADC_K2 + PRL.O2;       // 950nm   LD274
            REC.K3 = ADC_K3 + PRL.O3;       // 740nm   EPD740
            REC.K4 = ADC_K4 + PRL.O4;       // 1550nm  8376
            REC.K5 = ADC_K5 + PRL.O5;       // 900nm   SFH228
            REC.K6 = ADC_K6 + PRL.O6;       // 2300nm  8423
            REC.KU = ADC_KU + PRL.OU;       // Schweißspannung
            REC.KI = ADC_KI + PRL.OI;       // Schweißstrom
    // 440nm/740nm: Metalldampf zu Argon: Tropfenexplosion
    // 980nm/1550nm: Fe Dampf;  1950nm/2300 nm: Fe Schmelze

    // Zwischenergebnisse
            REC.RA = REC.K2 - REC.K1;       // Differenzen
            REC.RB = REC.K4 - REC.K3;       // hier nicht verwendet
            REC.RC = REC.K6 - REC.K5;       // hier nicht verwendet
            REC.RD = REC.KI - REC.KU;       // hier nicht verwendet
    //      REC.RE     Strom an/aus -> schweissstrom()          b.w.
```

Implementierung eines Regelalgorithmus 2

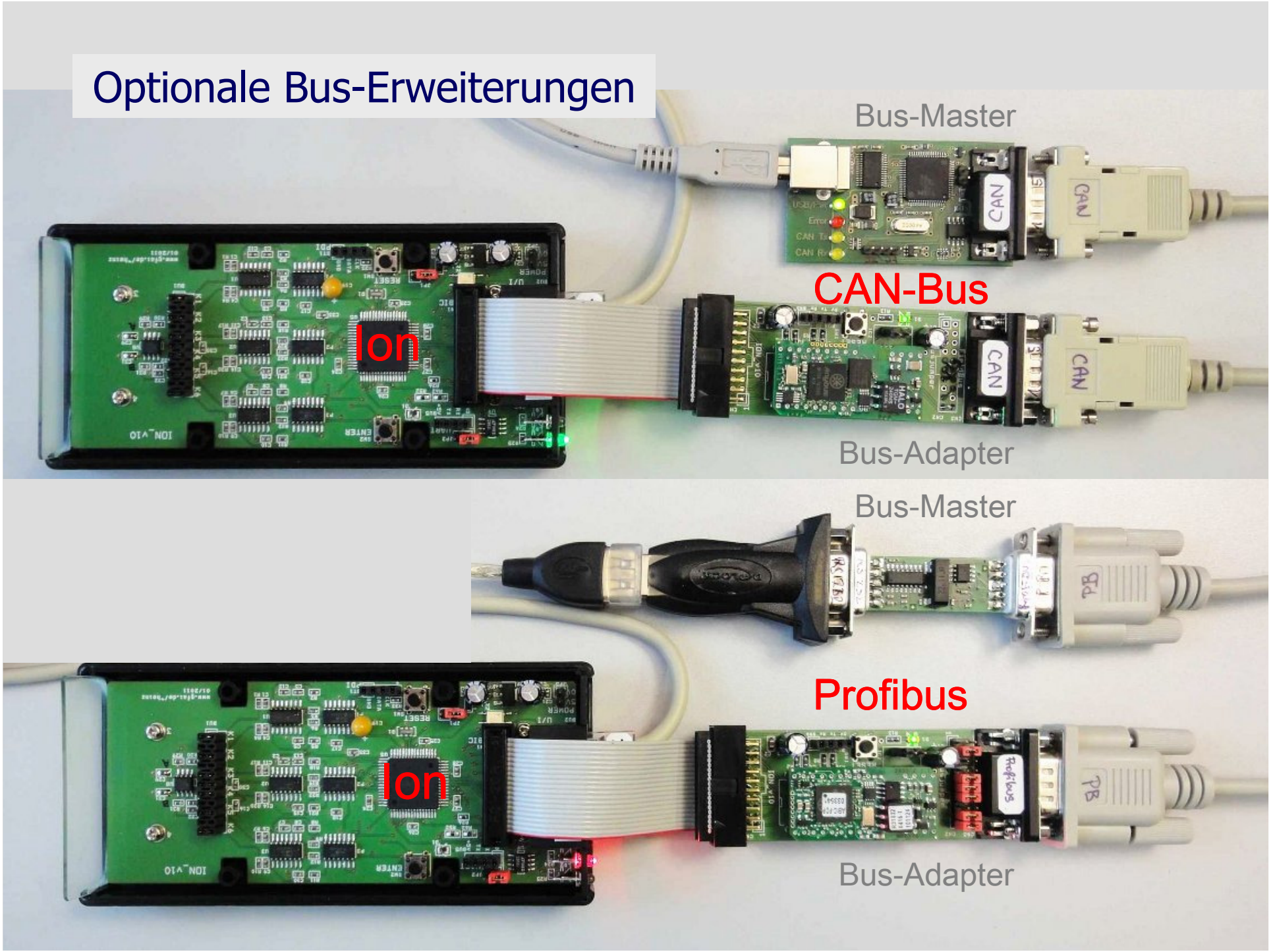
```
    // Fortsetzung…
    // aus der Parameterliste PRL werden verwendet:
            // PRL.T1:  Pulsverlängerung in Systick (8 pro ms)
            // PRL.TA:  Schwellwert für STOP welding
            // PRL.T1:  Berechnetes STOP-Ende
            // PRL.TE:  Schwellwert für Stop z.B. -200 (-2048...+2047)

    // Weld-Stop prüfen mit TE
    if (REC.RA < PRL.TE) {          // TE: Schwellwert für Stop z.B. -200 (-2048...+2047)
            systick = 0;                        // reset, solange Stop
            schweissstrom(stop);
    };

    // Weld Restart nach Pulsverlängerung um T1 (z.B. 80 ~ 10ms)
    if(systick > PRL.T1) {          // Pulsverlängerung in Systicks (8 pro ms)
            schweissstrom(start);   // z.B. 12 ms = 12 x 8; T1 = 96 Systicks
    }

    systick++;                      // dient der Pulsverlängerung
}
```

Programmierung des Ion

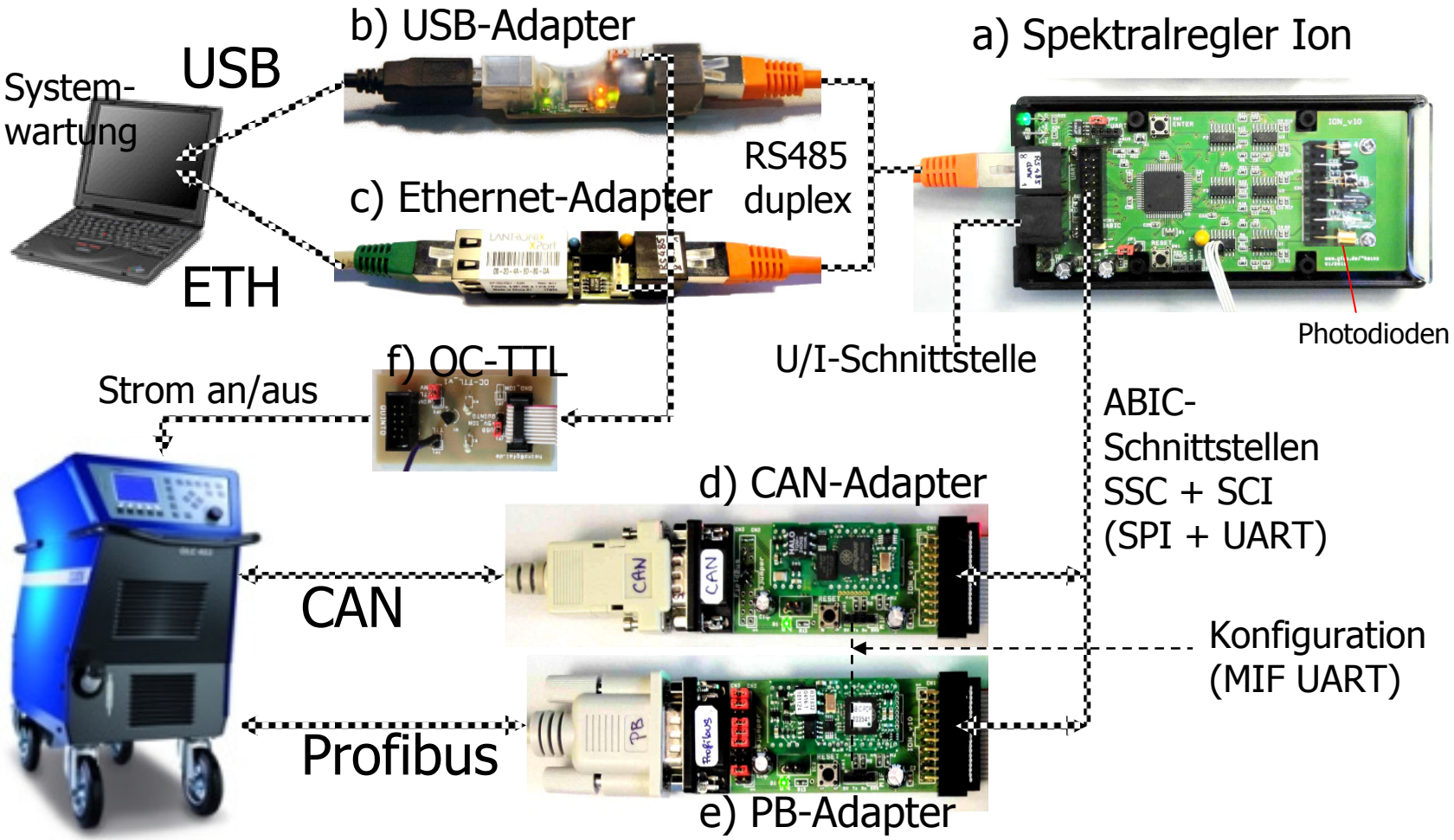
- ionOS wurde unter AVR-Studio 4.19 f.f. compiliert (freie Software)
- Es läuft auf einem Atmel-Controller ATxmega64A3 bzw. ATxmega192A3
- Die Atmel-spezifische Debug- und Programmierschnittstelle PDI (ST6) wird mit einem spezifischen PDI-Stecker verwendet (ST5)
- Als Programmiergerät eignet sich ein "JTAGICE mkII" (~350€) mit Adapter siehe Schema



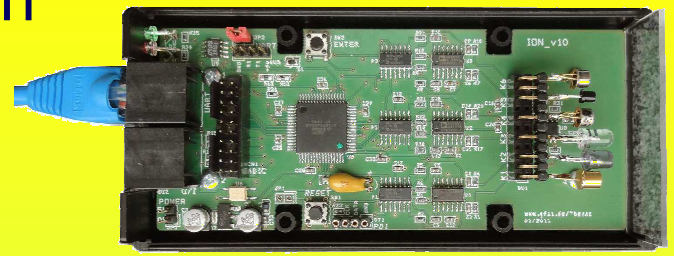
Optionale Bus-Erweiterungen



Übersicht

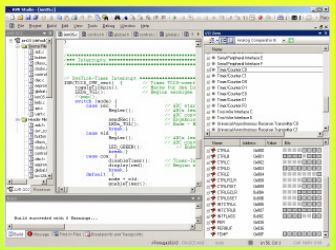

b) USB-Adapter
a) Spektralregler Ion
System-
wartung
USB
ETH
c) Ethernet-Adapter
RS485
duplex
Photodioden
U/I-Schnittstelle
f) OC-TTL
Strom an/aus
ABIC-
Schnittstellen
SSC + SCI
(SPI + UART)
d) CAN-Adapter
CAN
Konfiguration
(MIF UART)
Profibus
e) PB-Adapter

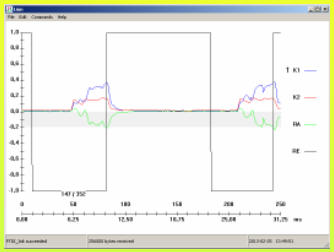
Zusammenstellung verfügbarer Hilfen



Hardware

- Spektralregler Hardware Ion v10 (max. 4 Mb/s)
- spektral selektive Photodioden (nicht über GFaI)



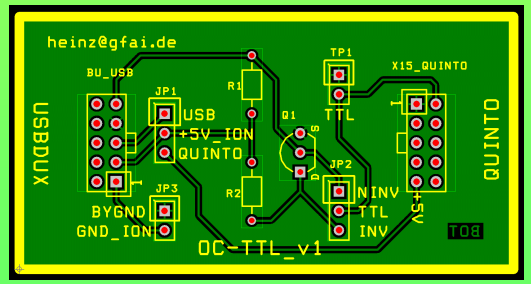


Software

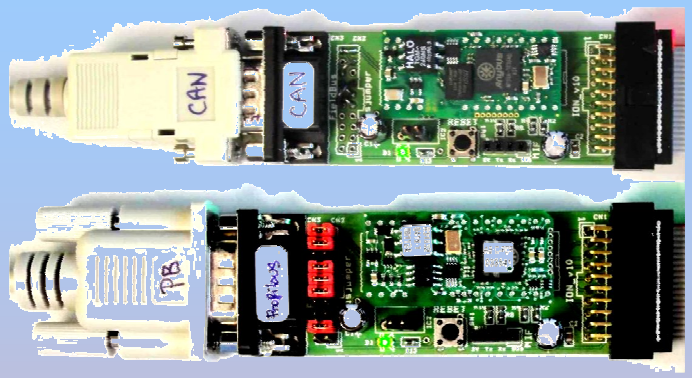
- Betriebssystem ionOS v2
- Windows-Interface Lion v0.1

Zubehör

- USB-Interface USBDUX v2 (RS485 duplex, 12 Mb/s)
- Ethernet to RS485 duplex, 900 kb/s, ETH2UART_v1
- Optokoppler-TTL-Adapter OC-TTL v1



Weitere Hilfsmittel



Hardware

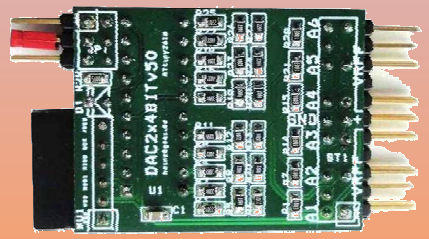
- Anybus-Adapter (SCI + SSC) CAN-Bus
- Anybus-Adapter (SCI + SSC) Profibus

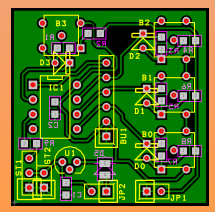
Software

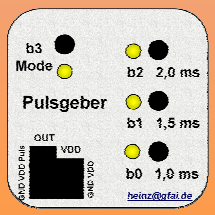
- Scilab-Programme zur Datenauswertung wav2gif.sce
- FTDI-Finder

Meß- und Prüfmittel

- Plasmasimulator 4 Kanäle a 256 Samples, 10 kS/s
- Zeitfunktionsgenerator DAC2X4BIT v50
- Pulsgeber 1…2 ms / 20 ms Pause



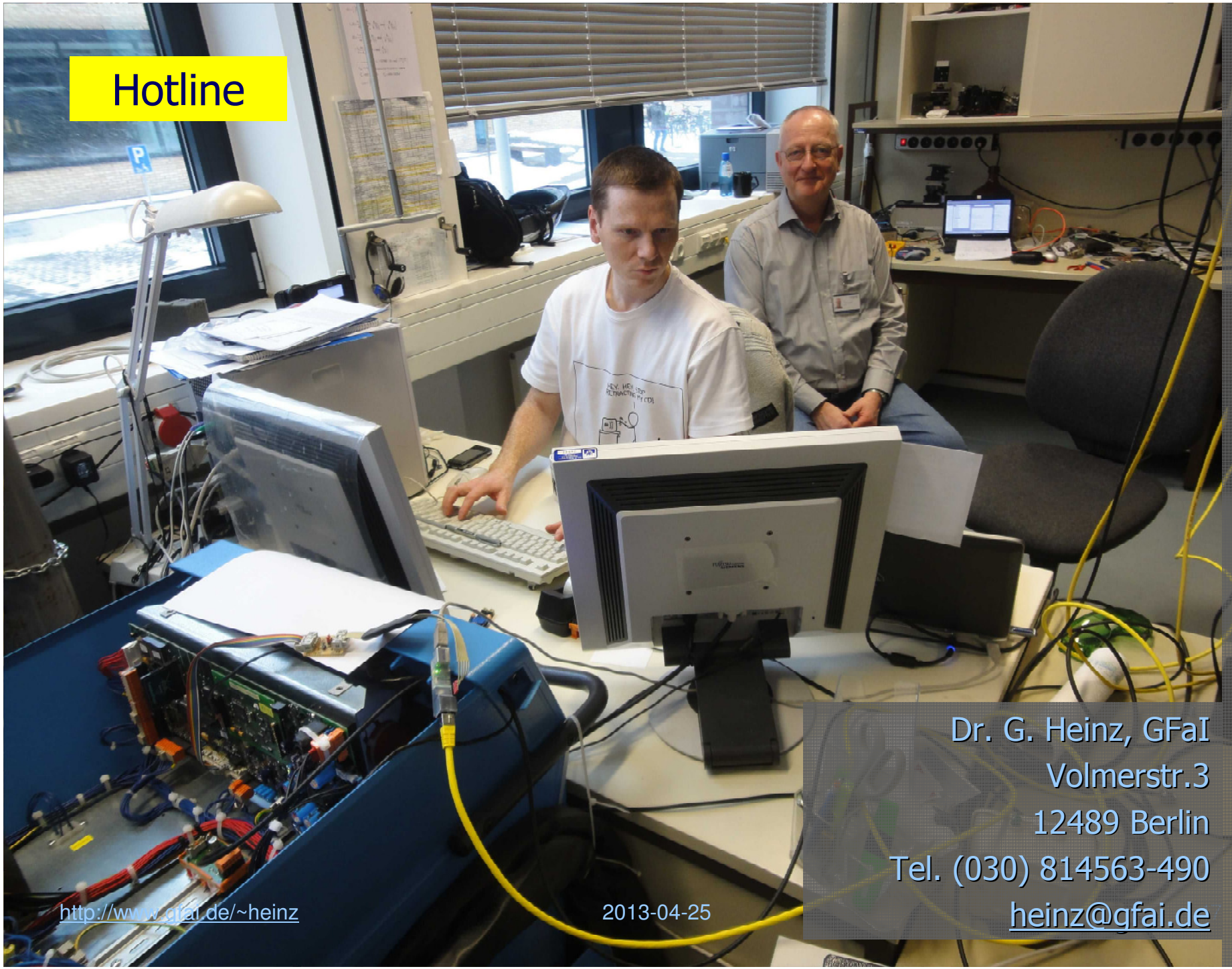




Download Directories

- http://www.gfai.de/~heinz/techdocs/index.htm
- http://www.gfai.de/spspba/ SPS DAsTg4-j3

Hotline



Dr. G. Heinz, GFaI
Volmerstr.3
12489 Berlin
Tel. (030) 814563-490
heinz@gfai.de

http://www.gfai.de/~heinz

2013-04-25

Der erste, erfolgreiche Test des Ion erfolgte im INP Greifswald vom 21.-22.2.2013 an einer CLOOS-QUINTO GLC403.
Danke für die freundliche Unterstützung an Heinz Schoepp und Gregor Goett!