Gut, hier tut grad das der Ausdruck irgendwie nicht bei diesen Dingen. Immer wenn alle Eingangs variablen aus sind, ist die ausgangsvariable auch aus. Das soll eigentlich nicht so und gucken wir uns das nachher an und was stimmt noch nicht ach ja irgendwas stimmt mit der Unterbrechung nicht, müssen wir angucken ach ja, die Beiträge zusammenfassen wäre ne gute Sache Beiträge zusammenfassen und mit dem Basic. Das muss ich mir erst mal angucken, wie die Sprache an sich aussieht. Was ist dazu zu bedenken gibt und ja dann halt das Zeug einfach weitermachen wie bisher auch. Was fehlt noch jetzt hier ach mit dem GAL vielleicht mal gucken oder so so

Gut kleine Debug Aufgabe...

Erstens: wenn alle aus sind sind alle aus, kann nicht sein definitiv ein Fehler

Zweitens: led vertauscht, `x2,x1,x0` Hier ist wäre Vertauschung möglich...

jetzt weiter...

jetzt weiter...

gut, ich tue jetzt folgendes: ich gehe noch mal einkaeufe, das zeug wird heute in einem beitrag zusammen gefasst, daran arbeite ich. erstens und zweitens: es gibt ja die programmiersprache BASIC, ich selber habe eine Art Compiler mehrfach geschrieben, viele kleine. Bakus Naur Form sind welche. so, viele kleine, aber groessere

ich hatte das buch von dijkstra ist der herr glaube ich, muss aber gucken, oder ein anderer. compilerbau oberon. an der fernuni hagen, compilerbau kurs

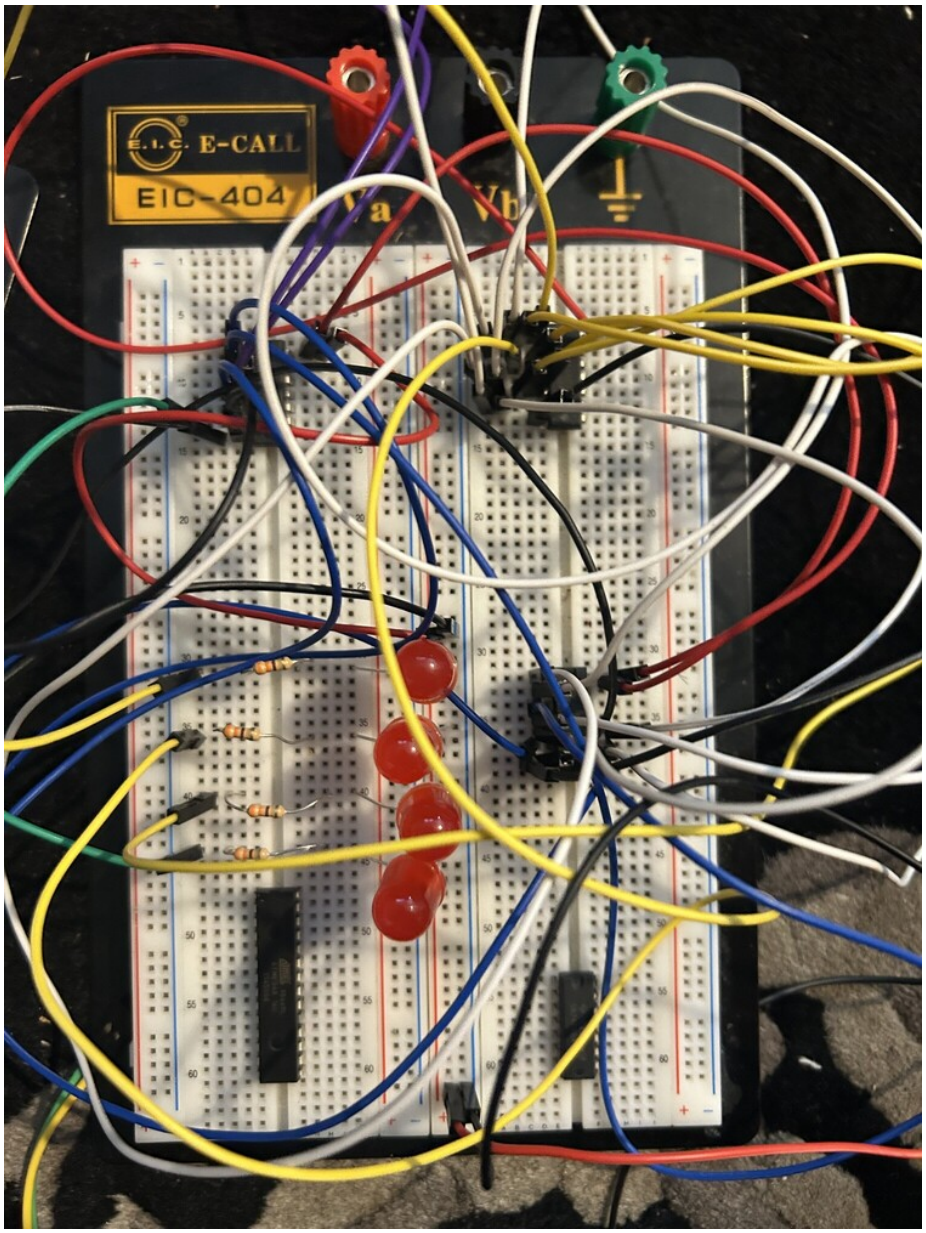
basic vom prinzip einfach. ich probiere einen fuer atmega8 und pc allgemein zu schreiben, indem falle, atmega8...

Jetzt einmal Dampfmaschine fahren lassen dann muss ich ja einmal aufräumen kann ich heute auch nicht mehr machen

Ich habe es allerdings genau gesehen und das hab ich auch schon vorher gesehen. Beim ersten bleibt die LED im Ausgang Schalt Netz würde man beim Schaltwerk sagen aus und es ist falsch. Also da muss ich noch mal gucken was die TTL Gatter betrifft

Der Timer tut, er tut gut toll und ja also ich mach kurz einen Film und zeige kurz es bietet sich jetzt der das externe Interrupt an also wo ich immer auf den Knopf drücken muss weil dann kann ich das ablesen wenn es als Film so schnell durchläuft. Da hab ich natürlich meine Schwierigkeiten auch wenn's relativ langsam geht die LED am Ausgang des Schaltnetzes zu kontrollieren.

Sprich, wenn ich meinen Knopf drücken muss, kann ich genau danach fotografieren




```
;; david vajda; m8 intervall timer / led slow

.include "m8def.inc"

.org 0x0000
rjmp RESET
.org OVF1Addr
rjmp OVF1Hndlr

RESET:
ldi r16, HIGH (RAMEND)
```

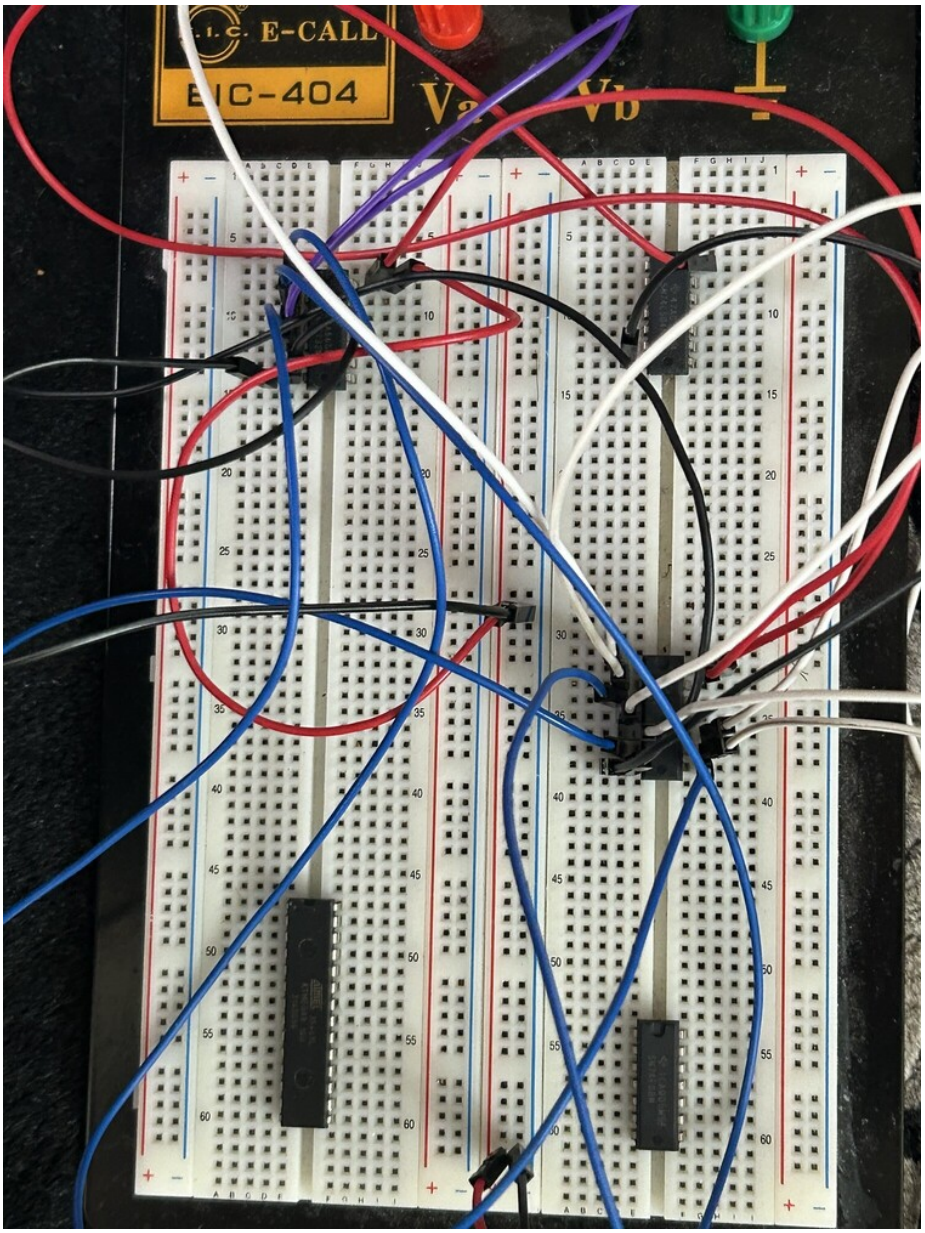
```
out SPH, r16
ldi r16, LOW (RAMEND)
out SPL, r16

ldi r16, 0xff
out DDRB, r16

ldi r16, 0xff
out PORTB, r16

;; ldi r16,
;; out TCCR1B, r16

;; ldi r16,
;; out TIMSK, r16

;; so weit ich es. aber ich weiss nicht,
;; wie heissen, die flags, 1. zur aktivierung
;; des timers, 2. zum prescaler heisst, teiler
;; also beim wievielten mal\dots
;; die nachgucken

sei
main001: rjmp main001

OVF1Hndlr:
inc r16
out PORTB, r16
reti
```

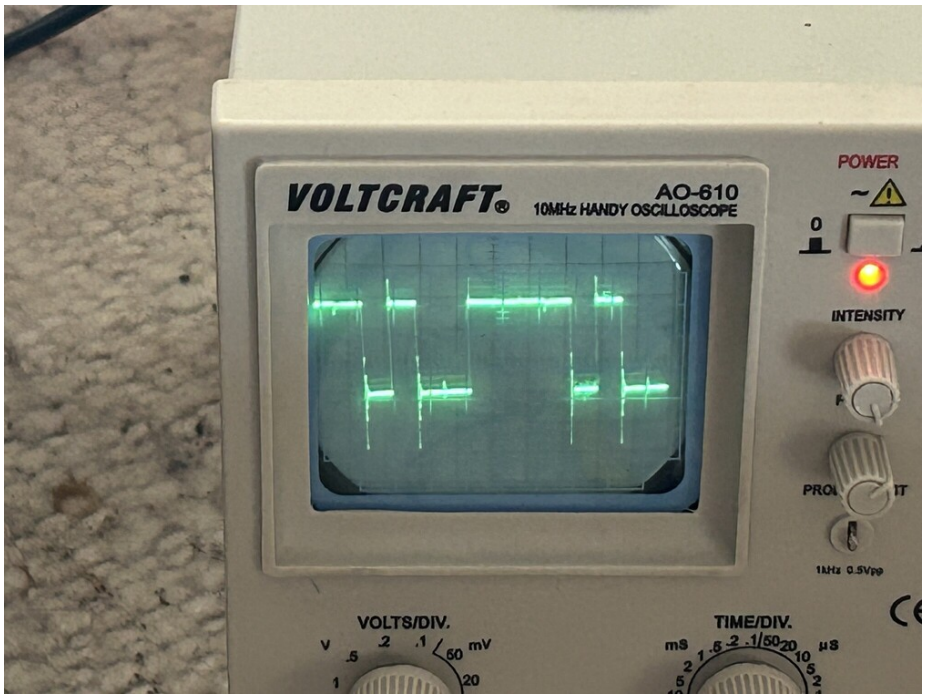
So, jetzt geht es weiter. Es kommt der IntervalTimer.

Gut, die Schaltung tut schon mal. Da ist aber eins und zwei glaub ich vertauscht also null und eins ist vertauscht. Muss noch mal gucken wo an den LED oder am Eingang einen Moment

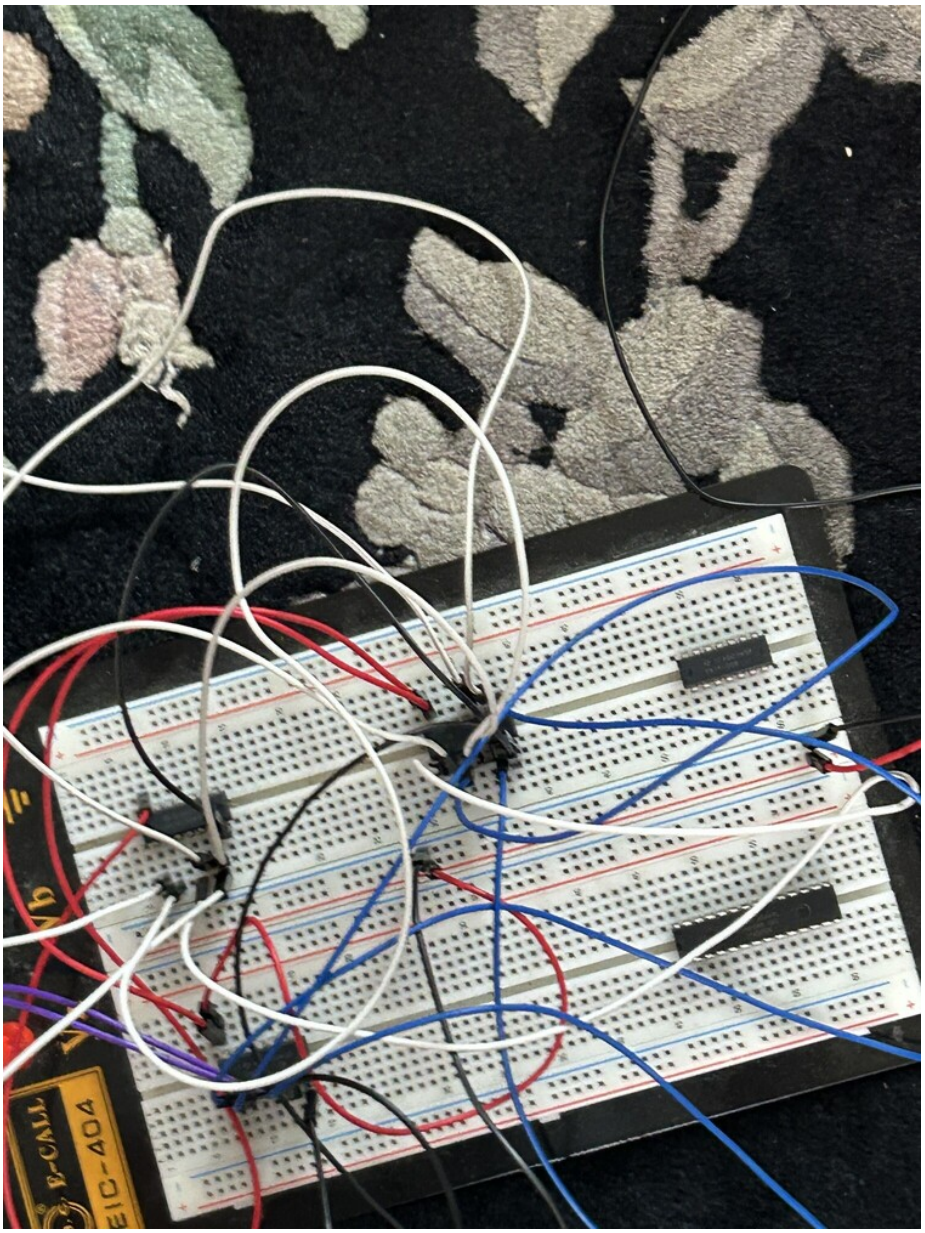
Ist Oszilloskop tut, das ist das eine und die Schaltung mir ist grad jetzt nicht weiter danach. Ich muss auf die Straße raus ist jetzt zu viel geworden für mich und muss noch mal auf die Straße raus. Ich weiß nicht ich hab die Schaltung soweit aufgebaut muss ich nachher ausprobieren.

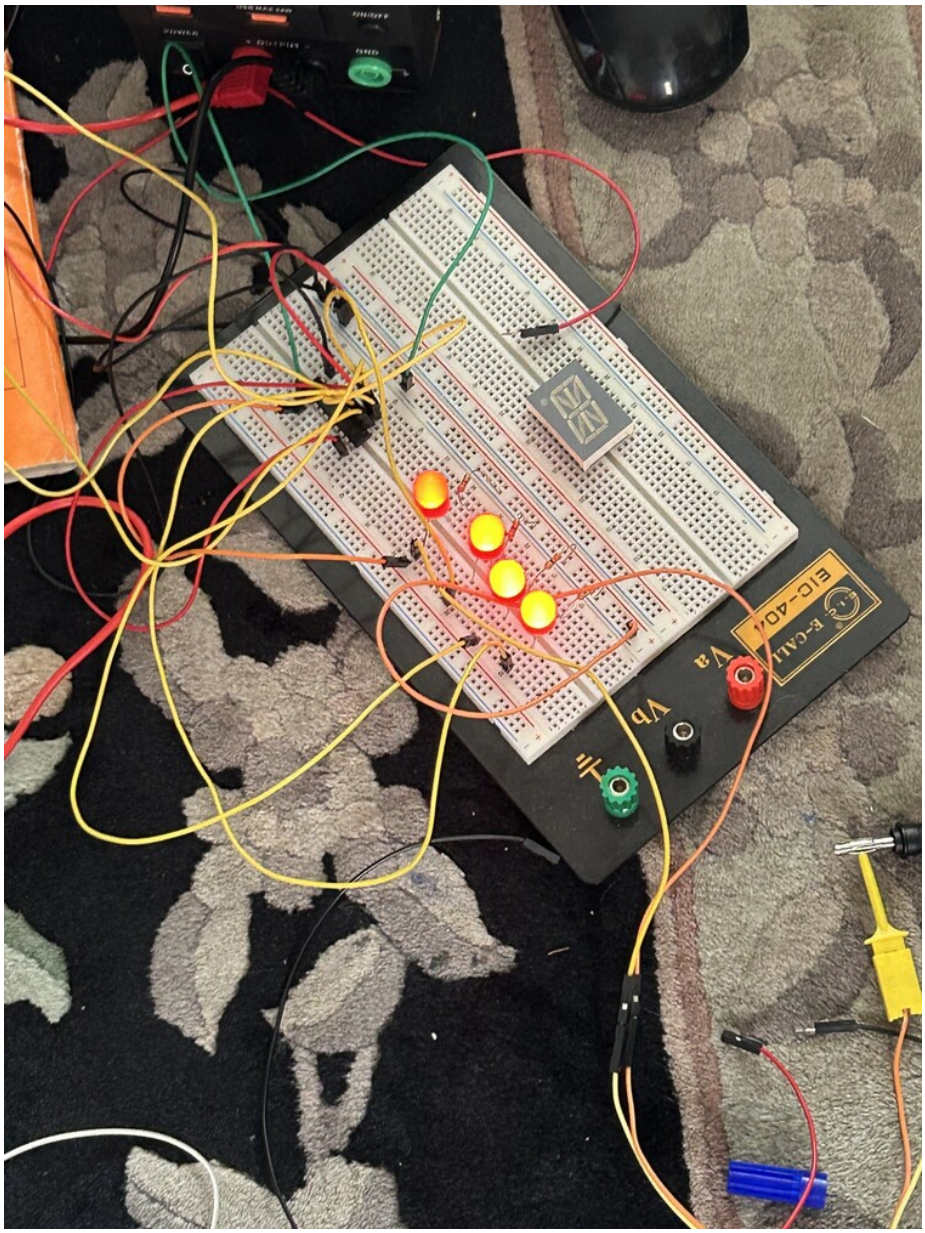
So, jetzt hab ich das Ding hier auch noch gebastelt wenn ich das hab ja also da fehlen jetzt die Anschlüsse einerseits vom nicht und andererseits in die nicht rein so und also in die 7404 wenn es geschehen ist, dann haben wir jetzt kurz eine rauchen und was noch für ein Zeug keine Ahnung. Ach so das mit dem Zeit Zeit Unterbrechung Zähler


E-CALL
EIC-404


VOLTCRAFT
AO-610
10MHz HANDY OSCILLOSCOPE
POWER
INTENSITY
VOLTS/DIV.
TIME/DIV.

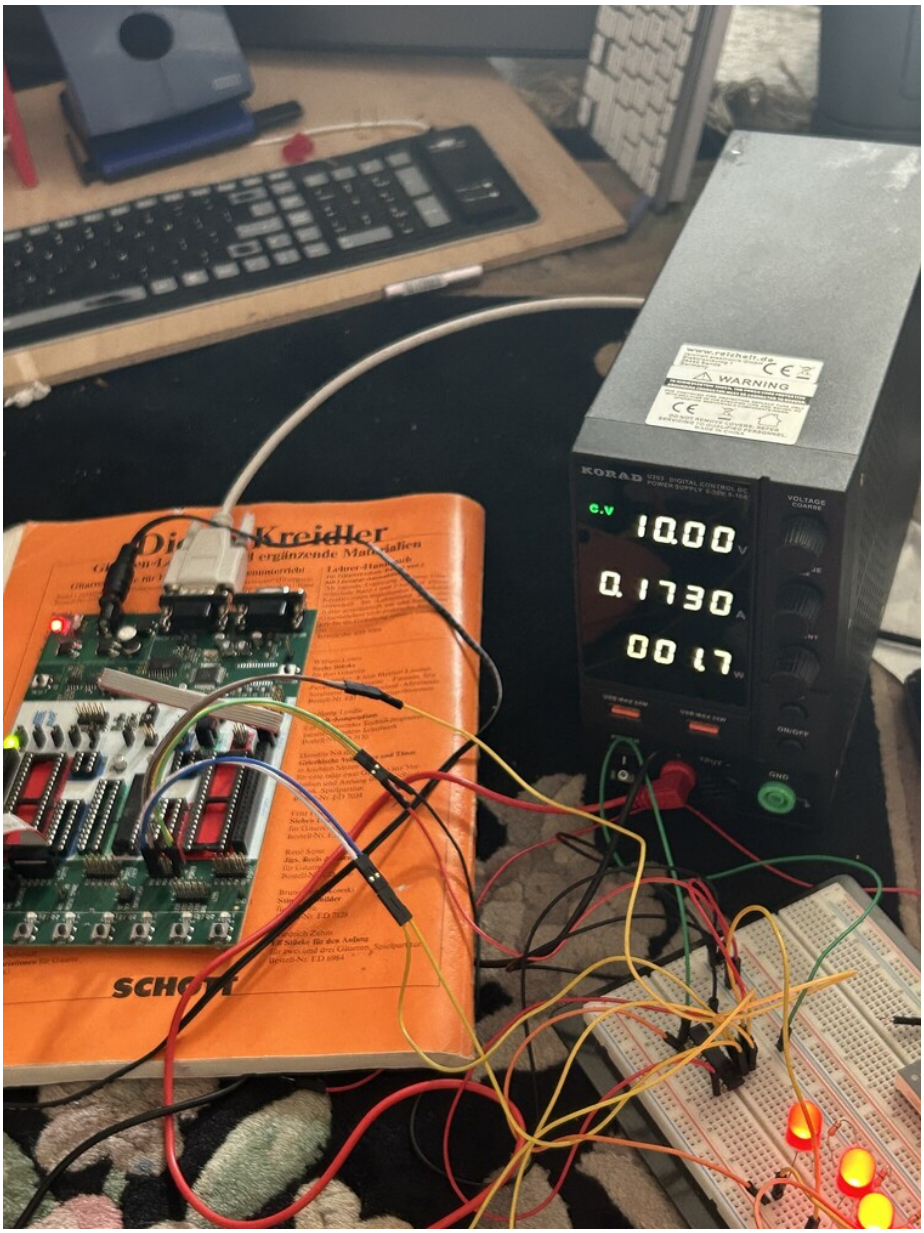

VOLTCRAFT®
AO-610
10MHz HANDY OSCILLOSCOPE
VOLTS/DIV.
V .5 1 2 .2 .1 50 mV 20 10 5
TIME/DIV.
mS 2 1 .5 .2 .1/50 20 10 5 2 1 .5 µS
POSITION
VOLT VAR.
POSITION




Vb
Va


VOLTCRAFT

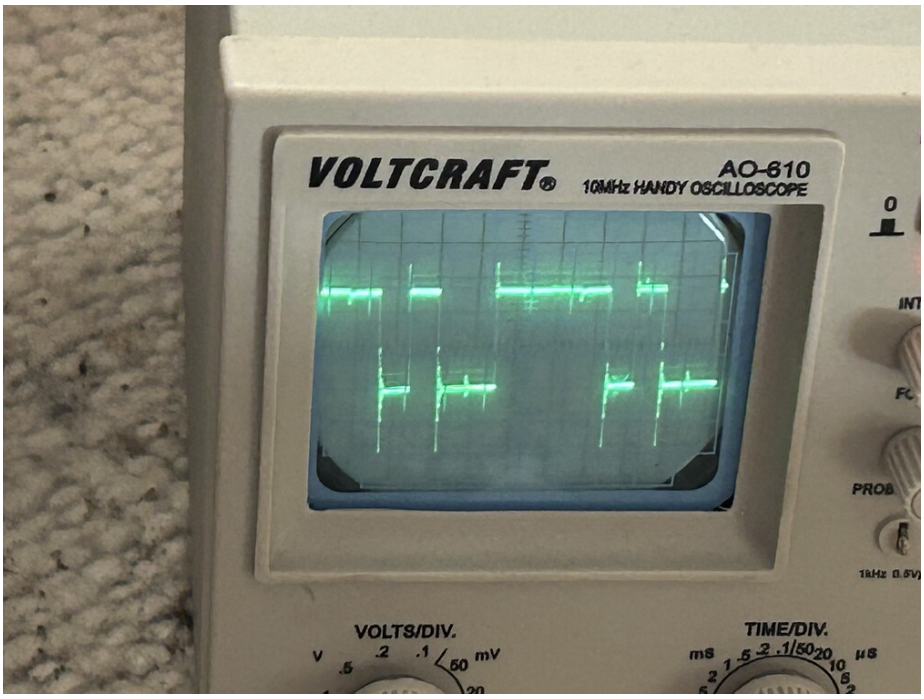

VOLTCRAFT



So, hier ist das Bild richtig das blau-weiß das spielt so kein Rollen auf dem Brett selber. Das ist jetzt vertauscht. Jetzt hab ich das jetzt kommt im nächsten Schritt. Wie gesagt ich muss die Schaltung noch mit normalen Gattern aufbauen. Im nächsten Schritt kommt noch das mit dem Timer Intervall

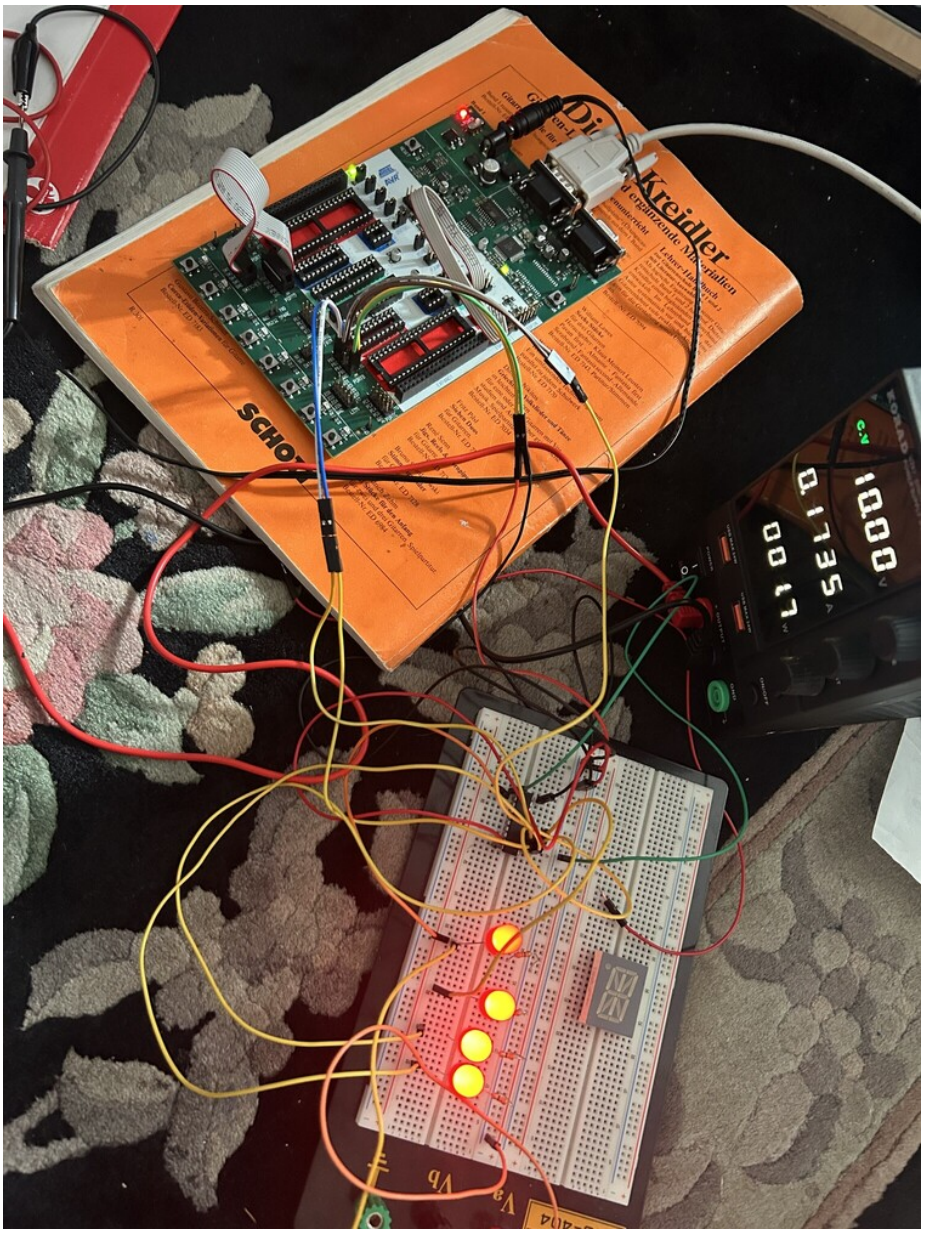

VOLTCRAFT
AO-610
10MHz HANDY OSCILLOSCOPE
VOLTS/DIV.
TIME/DIV.
POSITION
POSITION

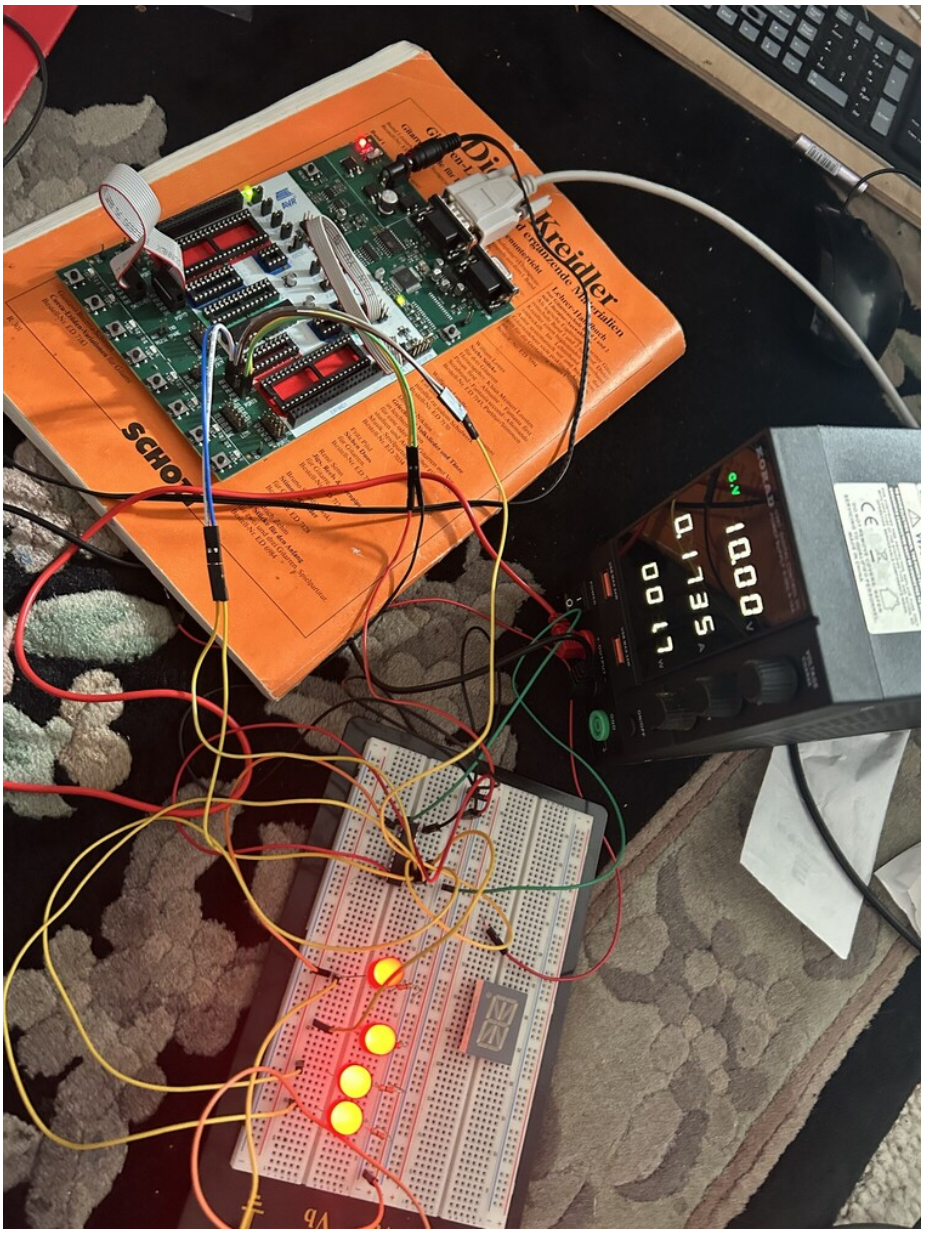

VOLTCRAFT
AO-610
10MHz HANDY OSCILLOSCOPE
VOLTS/DIV.
TIME/DIV.
POSITION

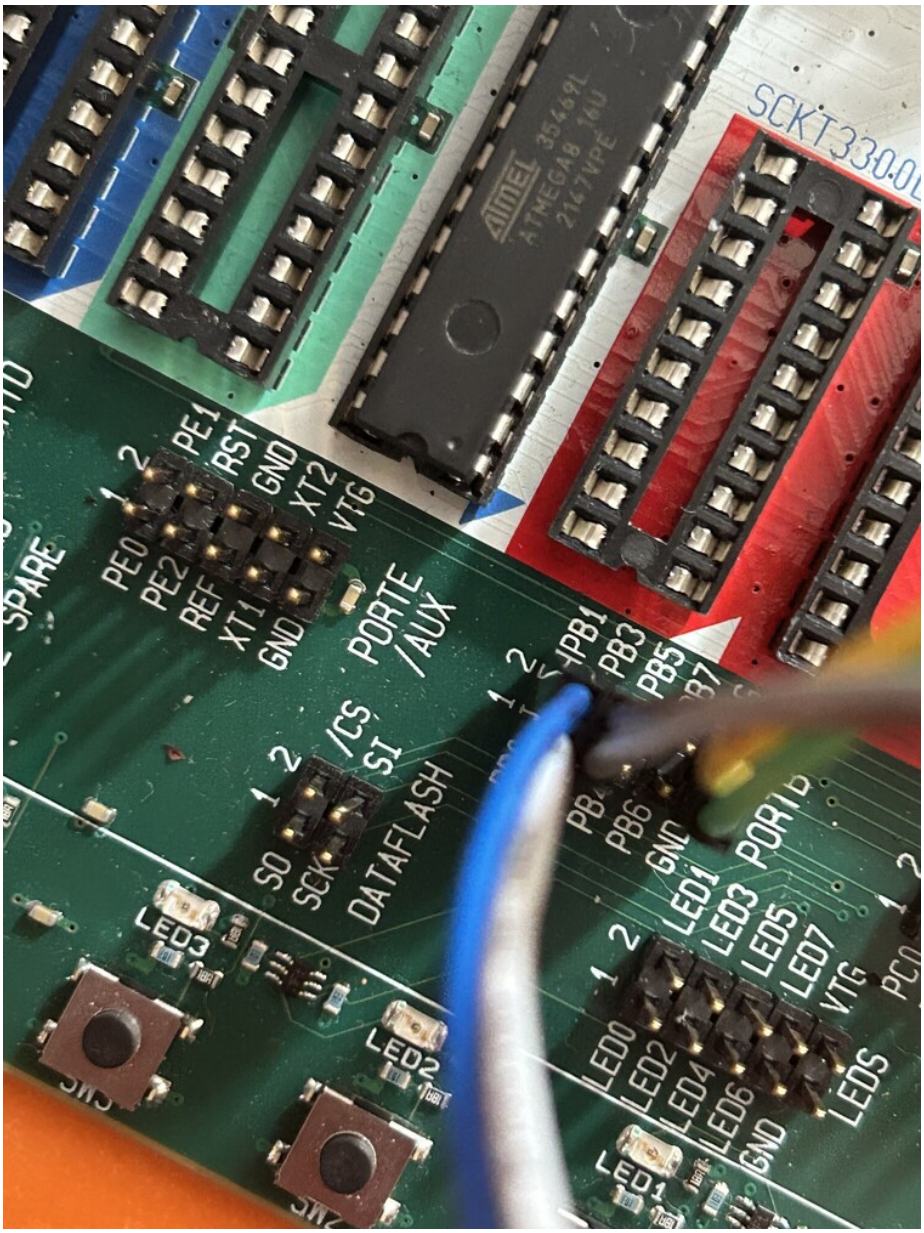


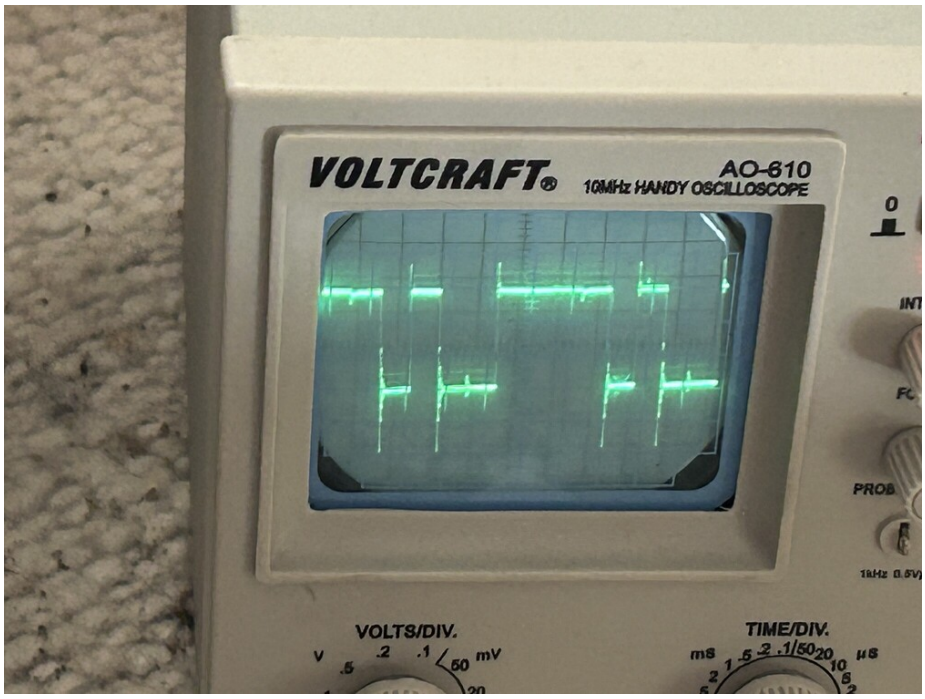
Man sieht es blau-weiße Kabel war vertauscht. Ich mach die Aufnahme auf dem Oszilloskop noch mal.

Nachher ein Beitrag wie gesagt, raus hier mit dem GAL zählt das fantastisch


Kreidler


Kreidler

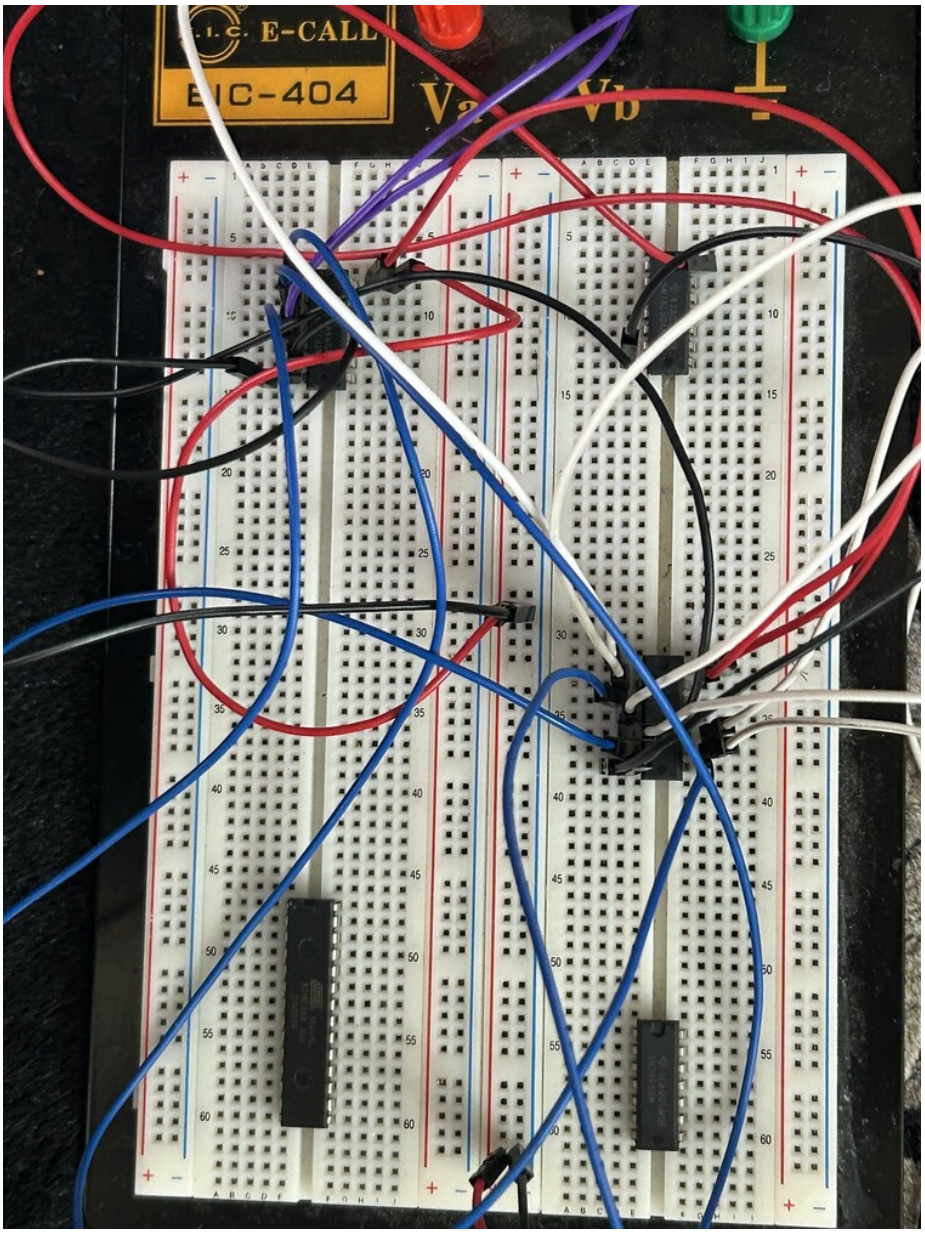

ATMEL
ATMEGA8 16U
2147VPE
PE1
RST
GND
XT2
VTG
PE0
PE2
REF
XT1
GND
SPARE
PORTE
/AUX
/CS
SI
SO
SCK
DATAFLASH
LED3
LED2
LED1
LED0
LED5
LED7
LED4
LED6
GND
VTG

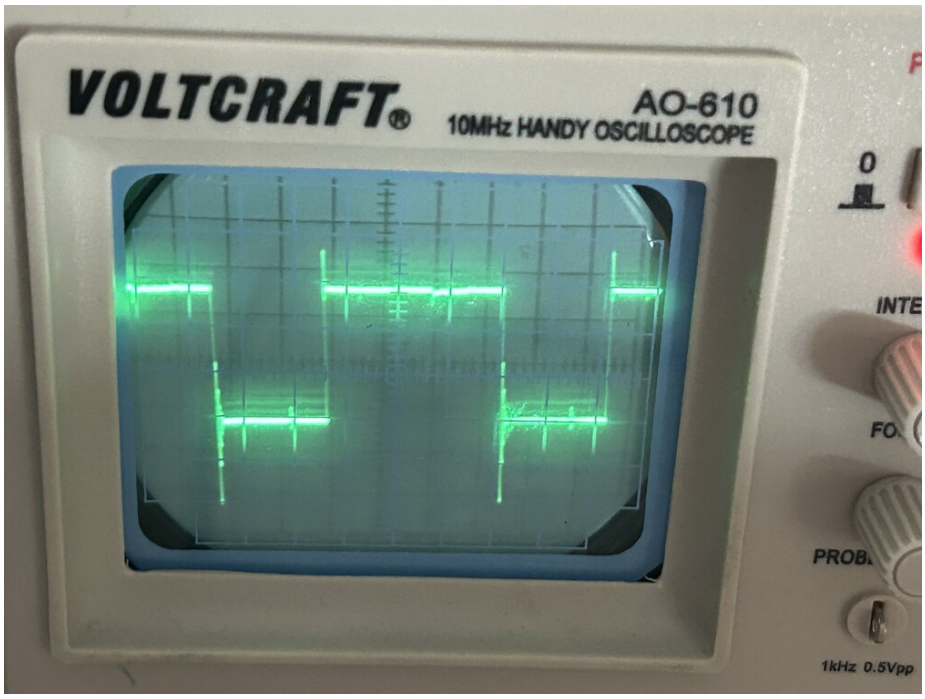

VOLTCRAFT®
AO-610
10MHz HANDY OSCILLOSCOPE
VOLTS/DIV.
TIME/DIV.


VOLTCRAFT
AO-610
10MHz HANDY OSCILLOSCOPE
VOLTS/DIV.
TIME/DIV.
POSITION

`GAL16V8`

Damit zählt fantastisch. Ich zeige Ihnen das auf dem Display von dem Oszilloskop und ich hab es auch ausprobiert mit dem langsamen, was schrittweise die LED nach oben zählt. Und ja, das geht auch sehr gut. Also mit dem GAL das ist die LED nach oben. Zählt ein Moment.


E-CALL
EIC-404


VOLTCRAFT®
AO-610
10MHz HANDY OSCILLOSCOPE
0
INTE
FO
PROBE
1kHz 0.5Vpp


```
;; so jetzt mit oszi, gleich dran machen\dots fuer das erste

;; david vajda
;; 06/27/2026
;; m8 count oszi count

.include "m8def.inc"

ldi r16, HIGH (RAMEND)
out SPH, r16
ldi r16, LOW (RAMEND)
```

```
out SPL, r16

ldi r16, 0xff
out DDRB, r16

ldi r16, 0xff
main001:
out PORTB, r16
;; rcall waitloop001
dec r16
rjmp main001

waitloop001:
push r16
push r17
push r18
ldi r16, 0x04
waitloop002:
ldi r17, 0xff
waitloop003:
ldi r18, 0xff
waitloop004:
dec r18
brne waitloop004
dec r17
brne waitloop003
dec r16
brne waitloop002
pop r18
pop r17
pop r16
ret

;; das tut - aber die wait loop zu langsam
;; danach extern und ohne verzoegerung, aber das laesst sich mit timer 0 auch schnell mit

;; david vajda
;; 06/27/2026
;; m8 count led by wait loop

.include "m8def.inc"

ldi r16, HIGH (RAMEND)
out SPH, r16
ldi r16, LOW (RAMEND)
out SPL, r16

ldi r16, 0xff
out DDRB, r16
```

```
ldi r16, 0xff
main001:
out PORTB, r16
rcall waitloop001
dec r16
rjmp main001

waitloop001:
push r16
push r17
push r18
ldi r16, 0x0f
waitloop002:
ldi r17, 0xff
waitloop003:
ldi r18, 0xff
waitloop004:
dec r18
brne waitloop004
dec r17
brne waitloop003
dec r16
brne waitloop002
pop r18
pop r17
pop r16
ret
```

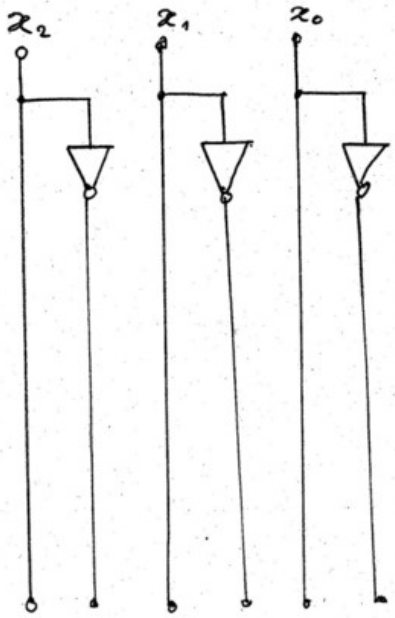
das ziel ist jetzt noch

1. natuerlich timer intervall
2. das timer kann auch kurz sein, sprich, loop ausserdem ohne: waitloop ins oszilloskop rein
3. durch das oszilloskop dann
4. durch ttl gatter - das ist einfach aufgebaut: 3 UND - mit 2 Eingaengen, so ist es da. dann ein ODER mit 3 eingaengen. da eine verbindung
5. das GAL - alles durchschicken
6. aber oszilloskop beides...

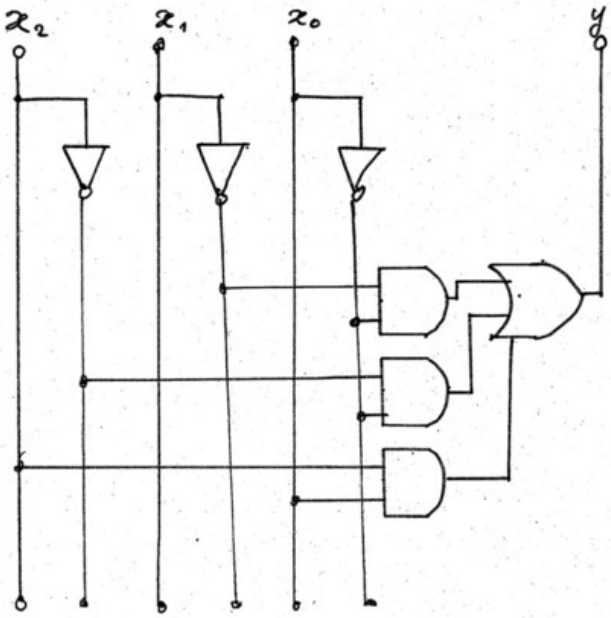
```
;; es tut, der erste quelltext

;; david vajda
;; 06/27/2026
;; m8 count led by ext int

.include "m8def.inc"
```

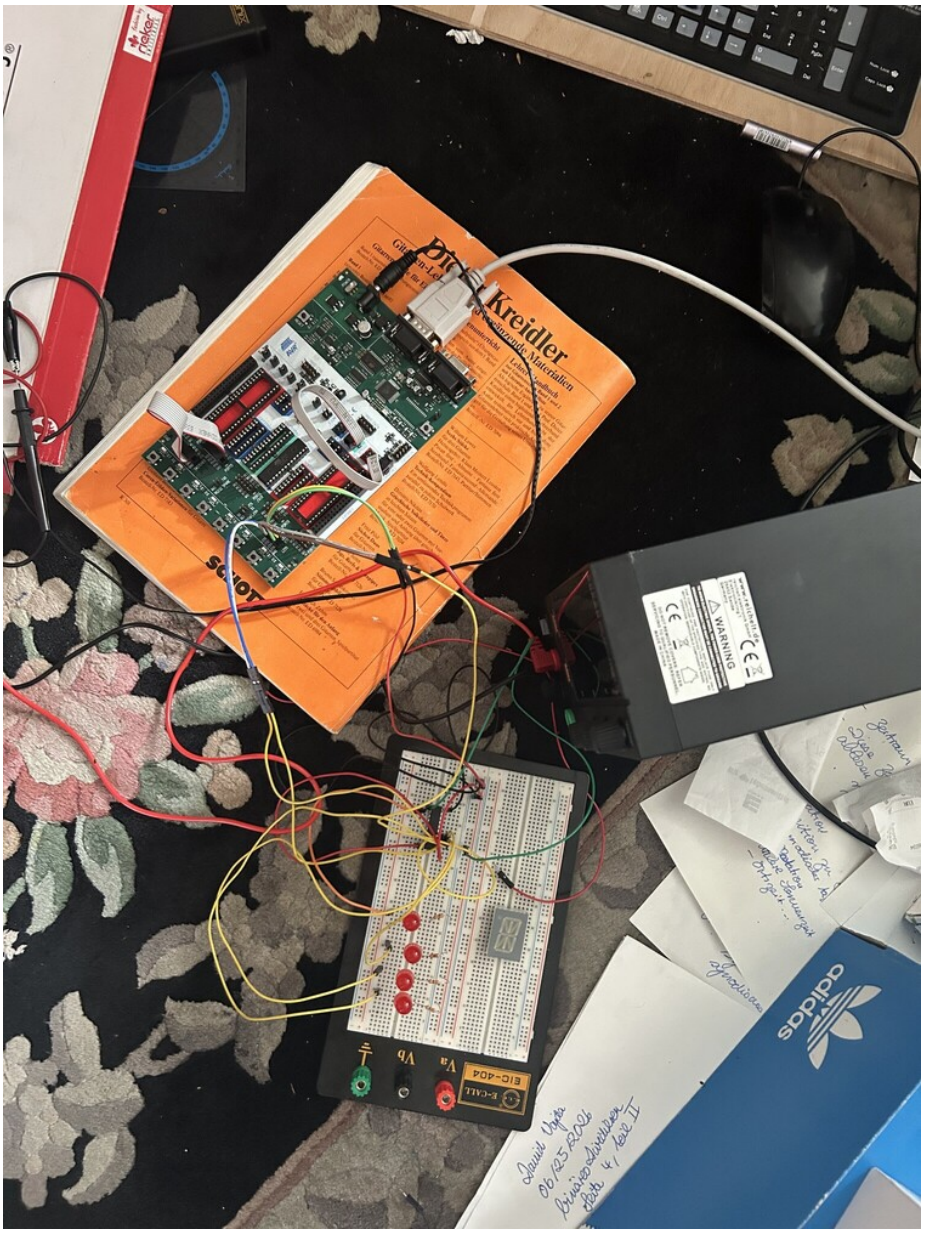
```
.org 0x000
rjmp RESET
.org INT0addr
rjmp INT0hndlr
.org INT1addr
rjmp INT1hndlr

RESET:
ldi r16, HIGH (RAMEND)
out SPH, r16
ldi r16, LOW (RAMEND)
out SPL, r16

ldi r16, 0xff
out DDRB, r16
ldi r16, 0xff

ldi r16, (1 << CS00) | (1 << CS01)
out MCUCR, r16
ldi r16, (1 << INT0) | (1 << INT1)
out GICR, r16

sei
main001: rjmp main001

INT0hndlr:
dec r16
out PORTB, r16
reti

INT1hndlr:
inc r16
out PORTB, r16
reti
```

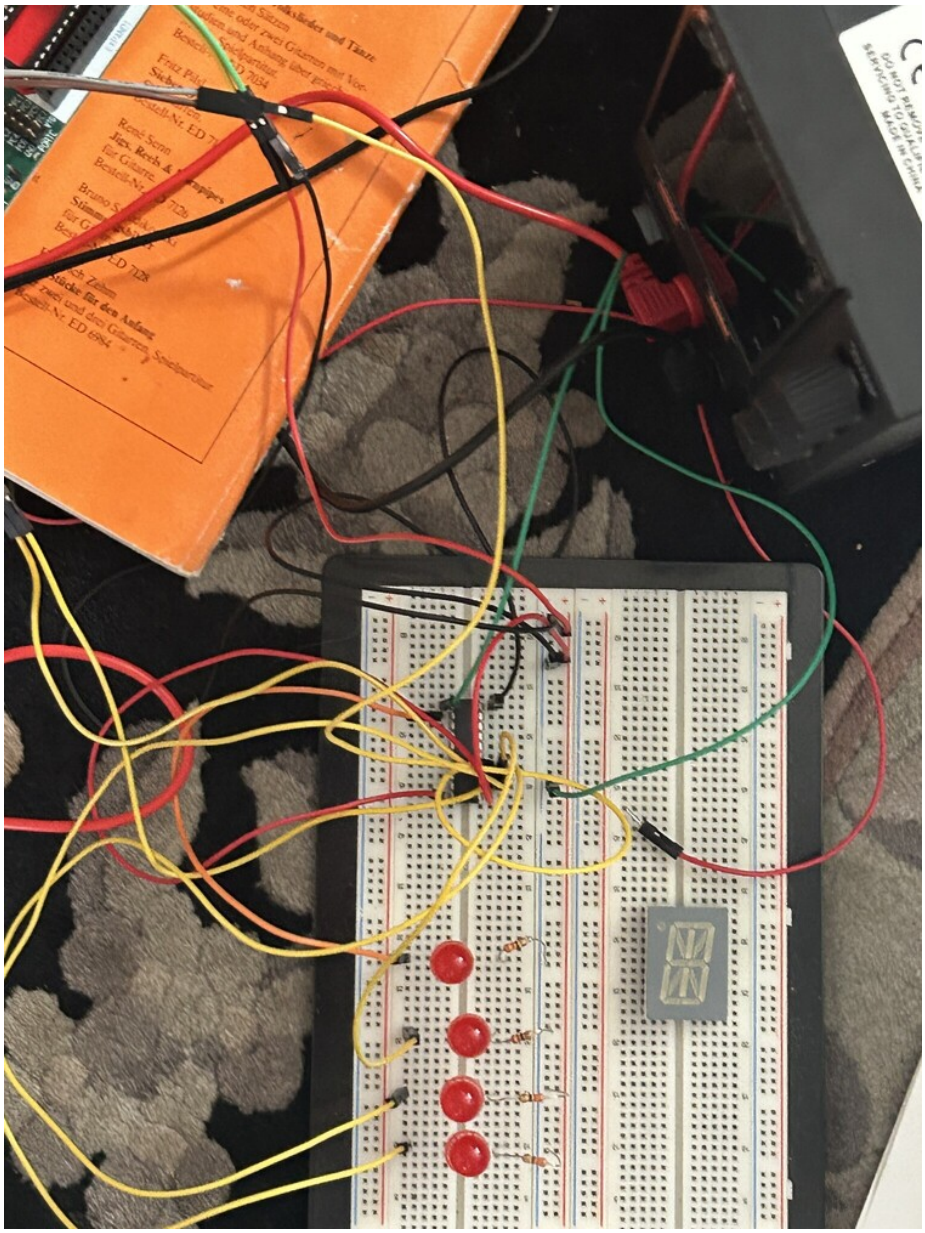
Dawid Vajda 06/27/2026
Schaltplan zur Inf 06/27/2026
quine mc cluskey 4 variablen...
Seite 1 (amerikanisch)
din 40700 ? o. din 40900 ?
xorg (deutsch) amerikanisch
ieee - 91/1984 ...

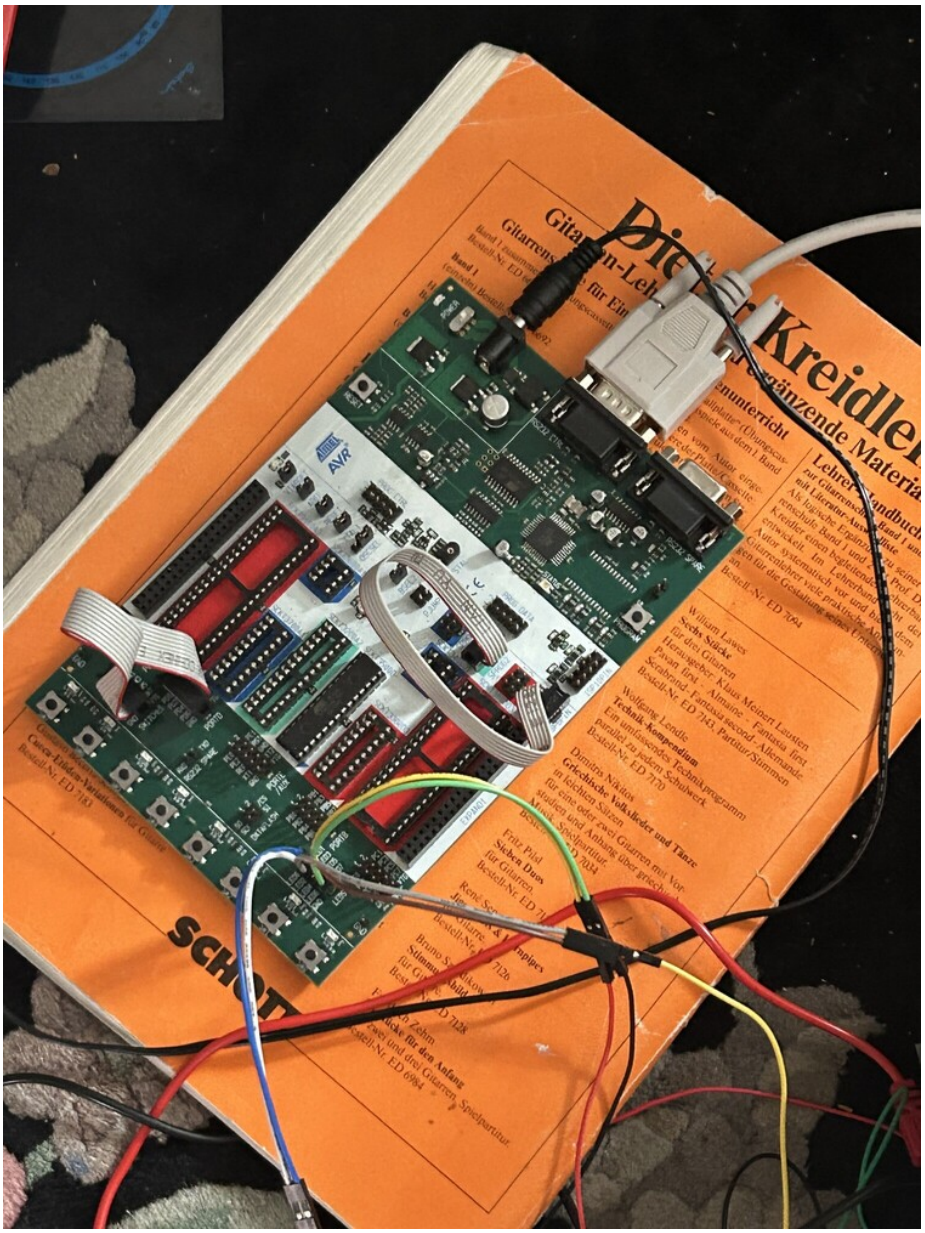


Dawid Vajda 06/27/2026
Schaltplan zur Inf 06/27/2026
quine mc cluskey 4 variablen...
Seite 1 (amerikanisch)
din 40700 ? o. din 40900 ?
sonst (deutsch) amerikanisch
ieee - 91/1984 ...




Kreidler
WARNING
adidas




AVR

```
;; david vajda
;; 06/27/2026
;; m8 count led by ext int

.include "m8def.inc"

.org 0x000
rjmp RESET
.org INT0addr
rjmp INT0hndlr
.org INT1addr
rjmp INT1hndlr

RESET:
ldi r16, HIGH (RAMEND)
out SPH, r16
ldi r16, LOW (RAMEND)
out SPL, r16

ldi r16, 0xff
out DDRB, r16
ldi r16, 0xff

ldi r16, (1 << CS00) | (1 << CS01)
out MCUCR, r16
ldi r16, (1 << INT0) | (1 << INT1)
out GICR, r16

sei
main001: rjmp main001

INT0hndlr:
dec r16
out PORTB, r16
reti

INT1hndlr:
inc r16
out PORTB, r16
reti
```



```
;; david vajda
;; 06/27/2026
;; m8 count led by ext int

.include "m8def.inc"

.org 0x000
rjmp RESET
```

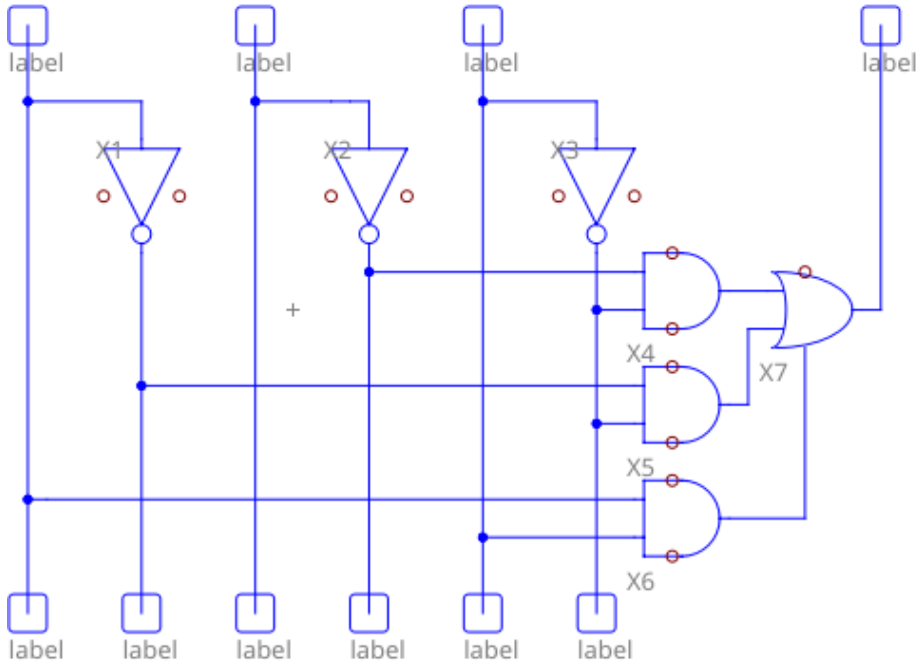
```
.org INT0addr
rjmp INT0hndlr
.org INT1addr
rjmp INT1hndlr

RESET:
ldi r16, HIGH (RAMEND)
out SPH, r16
ldi r16, LOW (RAMEND)
out SPL, r16

ldi r16, 0xff
out DDRB, r16
ldi r16, 0xff

ldi r16, (1 << CS00) | (1 << CS01)
out MCUCR, r16
ldi r16, (1 << INT0) | (1 << INT1)
out GICR, r16

sei
main001: rjmp main001

INT0hndlr:
dec r16
out PORTB, r16
reti

INT1hndlr:
inc r16
out PORTB, r16
reti
```

```
;; david vajda
;; 06/27/2026
;; m8 count led by wait loop

.include "m8def.inc"

ldi r16, HIGH (RAMEND)
out SPH, r16
ldi r16, LOW (RAMEND)
out SPL, r16

ldi r16, 0xff
out DDRB, r16

ldi r16, 0xff
main001:
out PORTB, r16
rcall waitloop001
dec r16
rjmp main001

waitloop001:
push r16
push r17
push r18
ldi r16, 0x0f
waitloop002:
ldi r17, 0xff
waitloop003:
ldi r18, 0xff
waitloop004:
dec r18
brne waitloop004
dec r17
brne waitloop003
dec r16
brne waitloop002
pop r18
pop r17
pop r16
ret
```



```
;; david vajda
;; 06/27/2026
;; m8 count led by wait loop

.include "m8def.inc"

ldi r16, HIGH (RAMEND)
out SPH, r16
```

```
ldi r16, LOW (RAMEND)
out SPL, r16

ldi r16, 0xff
out DDRB, r16

ldi r16, 0xff
main001:
out PORTB, r16
rcall waitloop001
dec r16
rjmp main001

waitloop001:
push r16
push r17
push r18
ldi r16, 0x0f
waitloop002:
ldi r17, 0xff
waitloop003:
ldi r18, 0xff
waitloop004:
dec r18
brne waitloop004
dec r17
brne waitloop003
dec r16
brne waitloop002
pop r18
pop r17
pop r16
ret
```



Ach ja, und so sehr sie angepisst sind, dass ich jetzt hier ständig was sage, da sind sie und werden sich sonst wohin setzen ich mich auch dagegen können wir auch nix machen. Verstehe alles. Lassen wir soweit gut sein. Jetzt kommt als Nächstes trotzdem

Ja, bei mir heißt es lass mir gut sein bitte wir müssen das nicht alles besprechen wer oder was auch immer wir sind oder nicht sind auch darum können wir uns nicht drücken verstehe ich alles. Ich bin trotzdem froh über das Programm hier von Linux und auch über das batronix

Jedenfalls soweit und kommt noch das nächste Thema. Wie gesagt, ich bin froh bei den dividier Befehl. Ich wollte trotzdem zur ständigen Übungen das gehört ja bei mir dazu.

Also, wo ich den atmega8 immer jeden Tag programmiere kommt ja der Klassiker mit der Warteschleife, dann mit dem externen Interrupt und dann mit dem Zeit Intervall

Das kommt und: das ist halt das und das ist auch schön so. Und jedenfalls das mache ich dann auch gleich zur Übung weil dann kann ich das auch mit diesem Baustein sehr gut durchlaufen lassen. So nächstes Thema jetzt erst mal der Schaltplan und wenn ich den hab, dann geh ich noch mal was einkaufen dann kommt der Rest heute Abend.

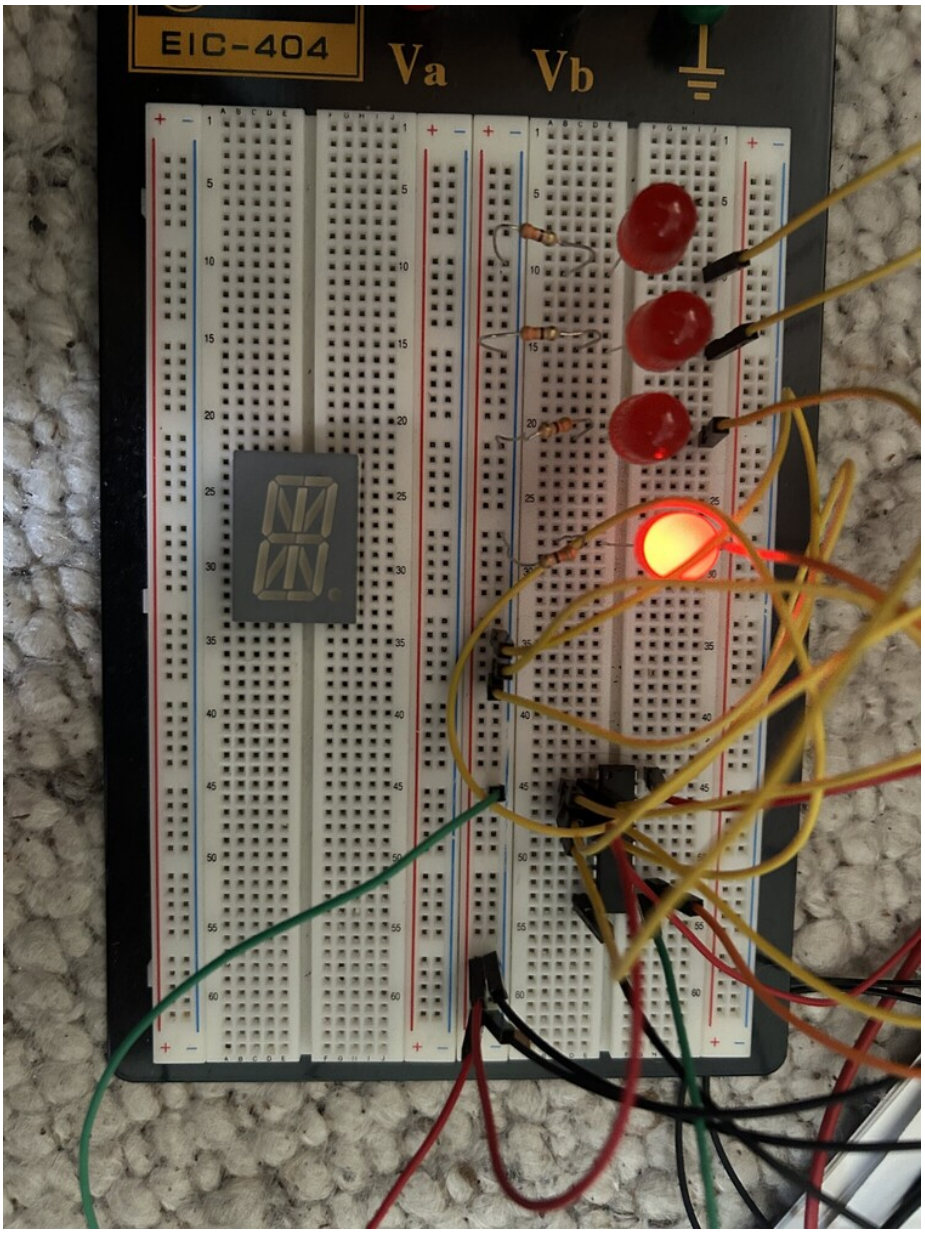
Jetzt sehe ich stimmt die Schaltung doch
Und ich bin froh, weil es ist mein erster programmierter

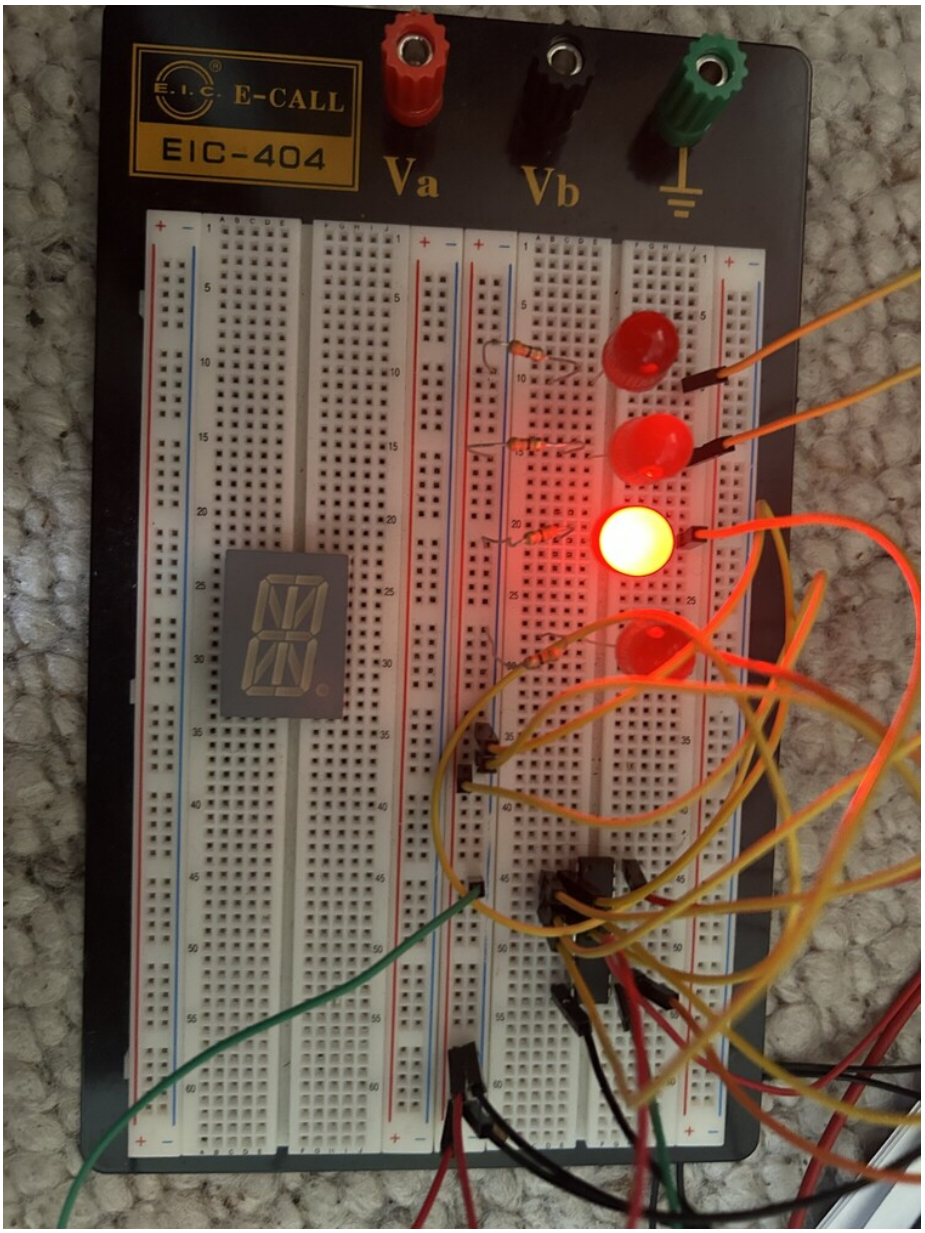
GAL

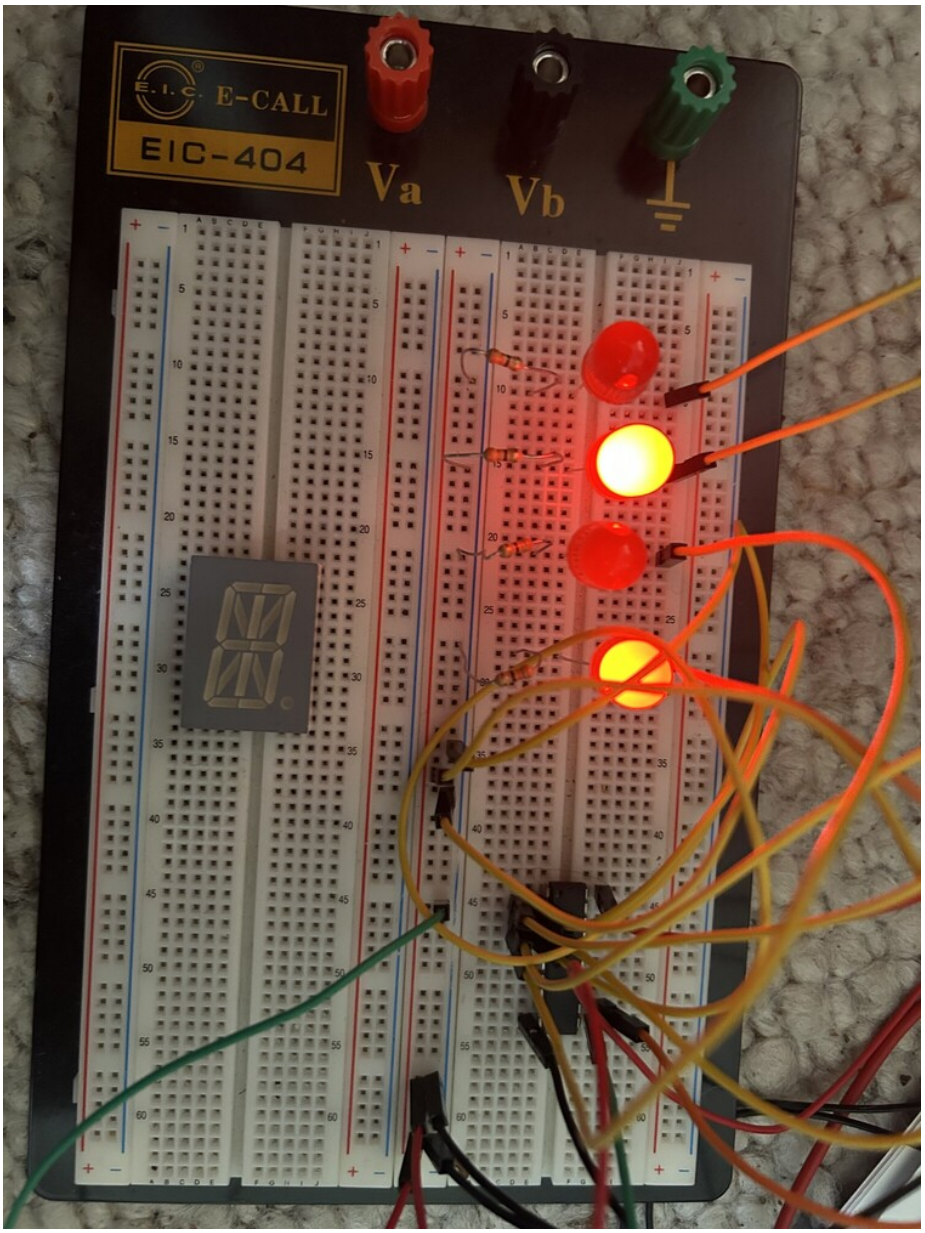
In meinem Leben, von dem ich ausgehen kann, ist es richtig

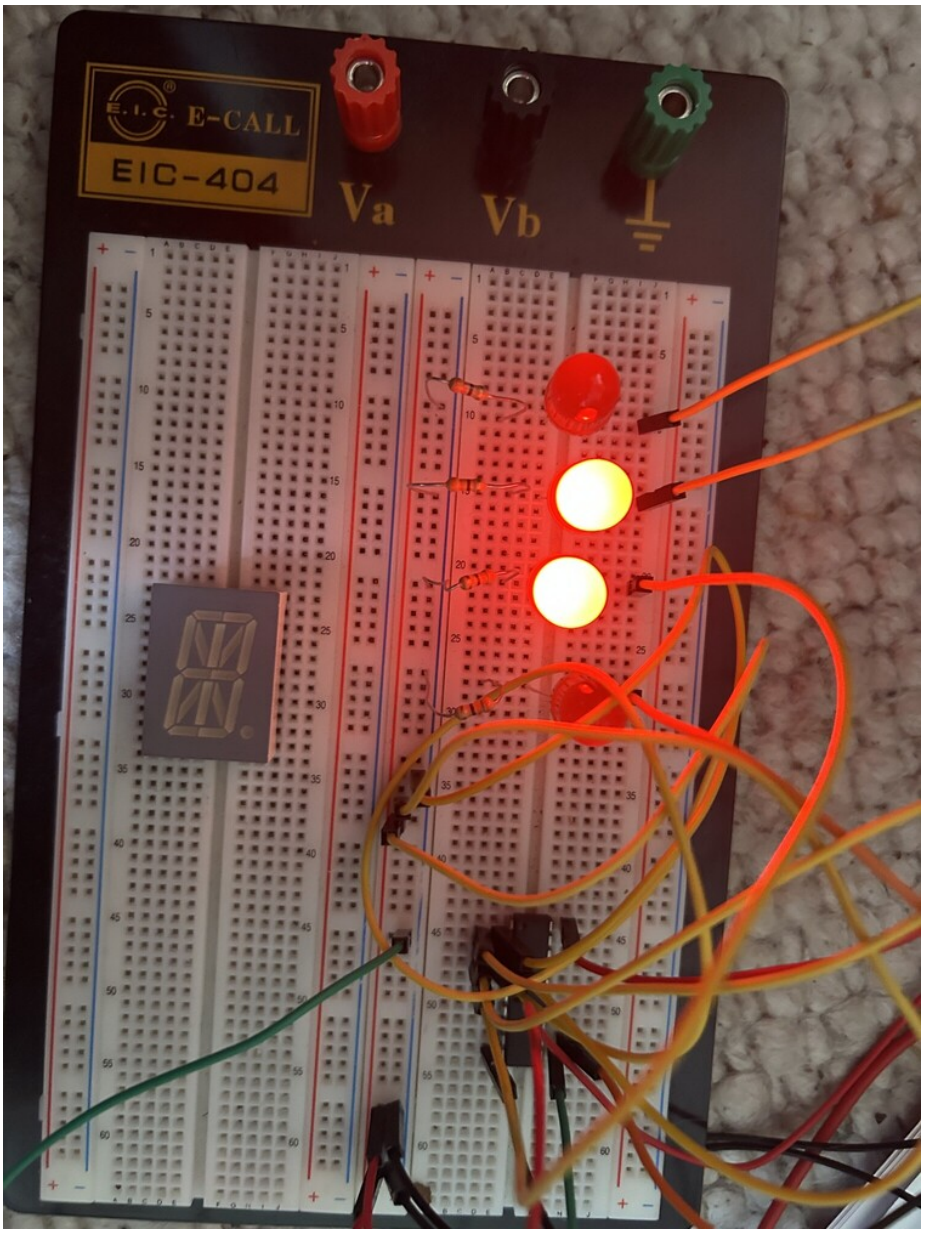
Ich bin glücklich, dass ich da rein geguckt hab, dass ich jetzt auch sehe bei dem map oder wie soll ich sagen? Pin und so weiter

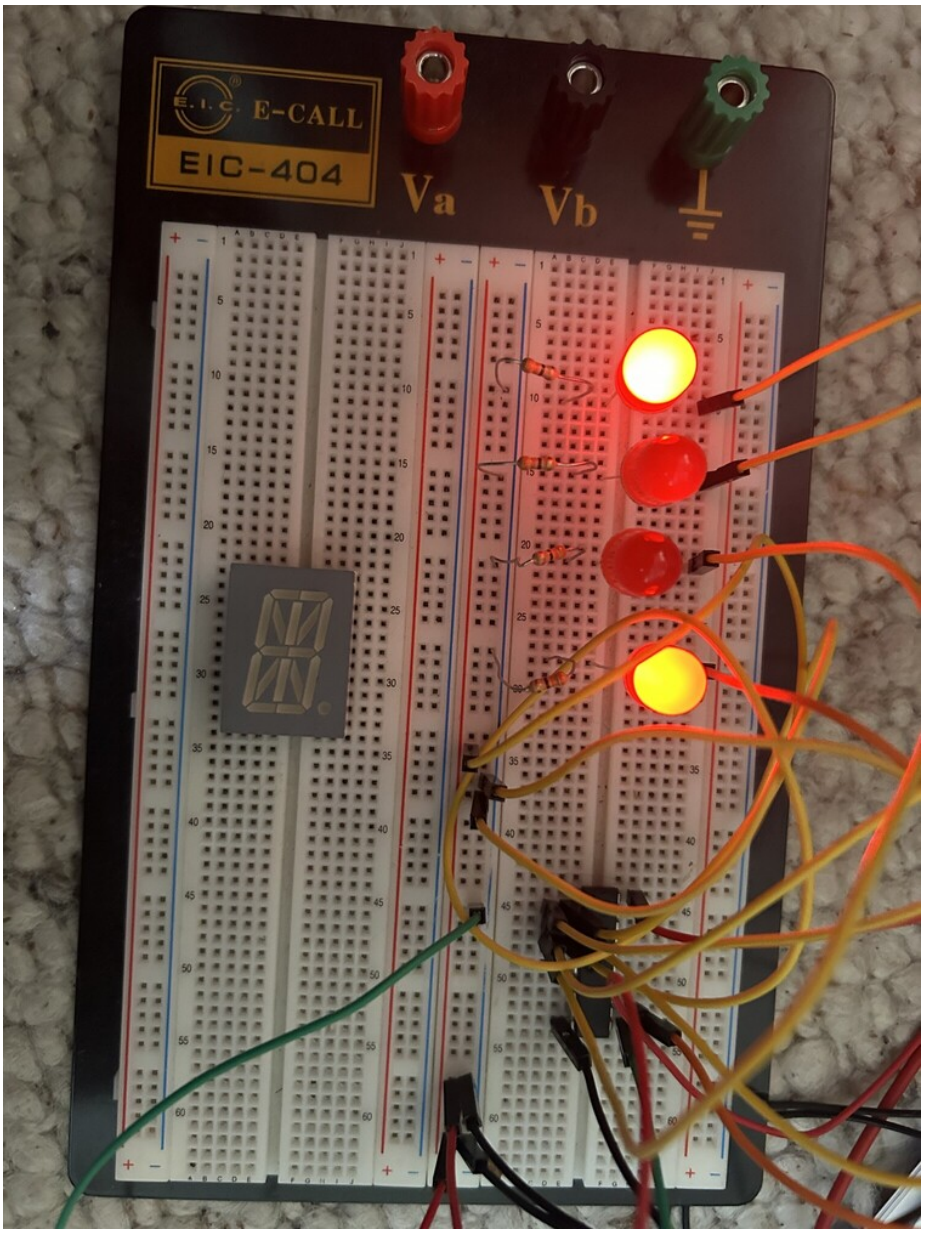
Ich habe galasm das ist sicher eine Abwandlung von Abel die ich zunächst seltsam fand, muss allerdings sagen, es stimmt wahrscheinlich nachdem ich zum Beispiel Beispiel gefunden hab eben nicht, dass keine Flip Flops möglich sind Sie sicher nicht. Der Fall von der Sache her, sah für mich auf den ersten Blick so aus. Es sieht alles ein bisschen anders aus ist sicher genauso hübsch. Ich werde auch probieren, sozusagen dieses jedec also Fuse ist selber zu machen, solange dieses galasm und auch was immer sie selber sind ich auch. Können wir uns sonst was einbilden und es vielleicht am Ende leider auch noch sein oder nicht leider wir können uns nicht von allem drum rum drücken das wissen wir da können wir auch trotzdem galasm beleidigen. Auch dahin da drum kommen wir nicht drum rum wahrscheinlich tut's ich bin froh, dass es damit geklappt hat. Jedenfalls der Baustein tut und es ist sicher auch trotzdem die nette Sprache, auch wenn wir uns noch was einbilden und es auch am Ende sind oder nicht, warum wir nicht drum rum drücken sie verstehen schon jedenfalls es tut jetzt hier in diesem Fall

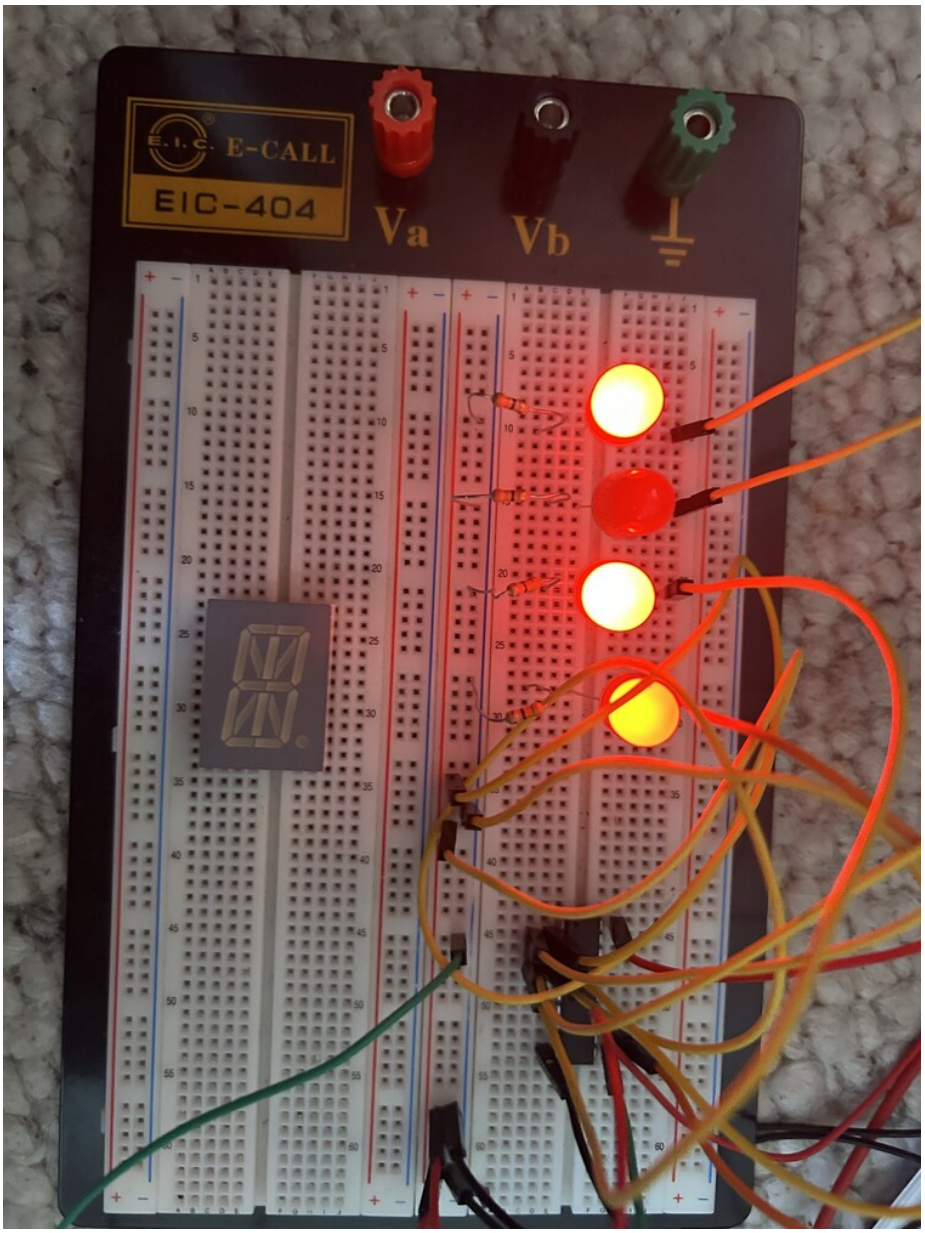

EIC-404
Va
Vb

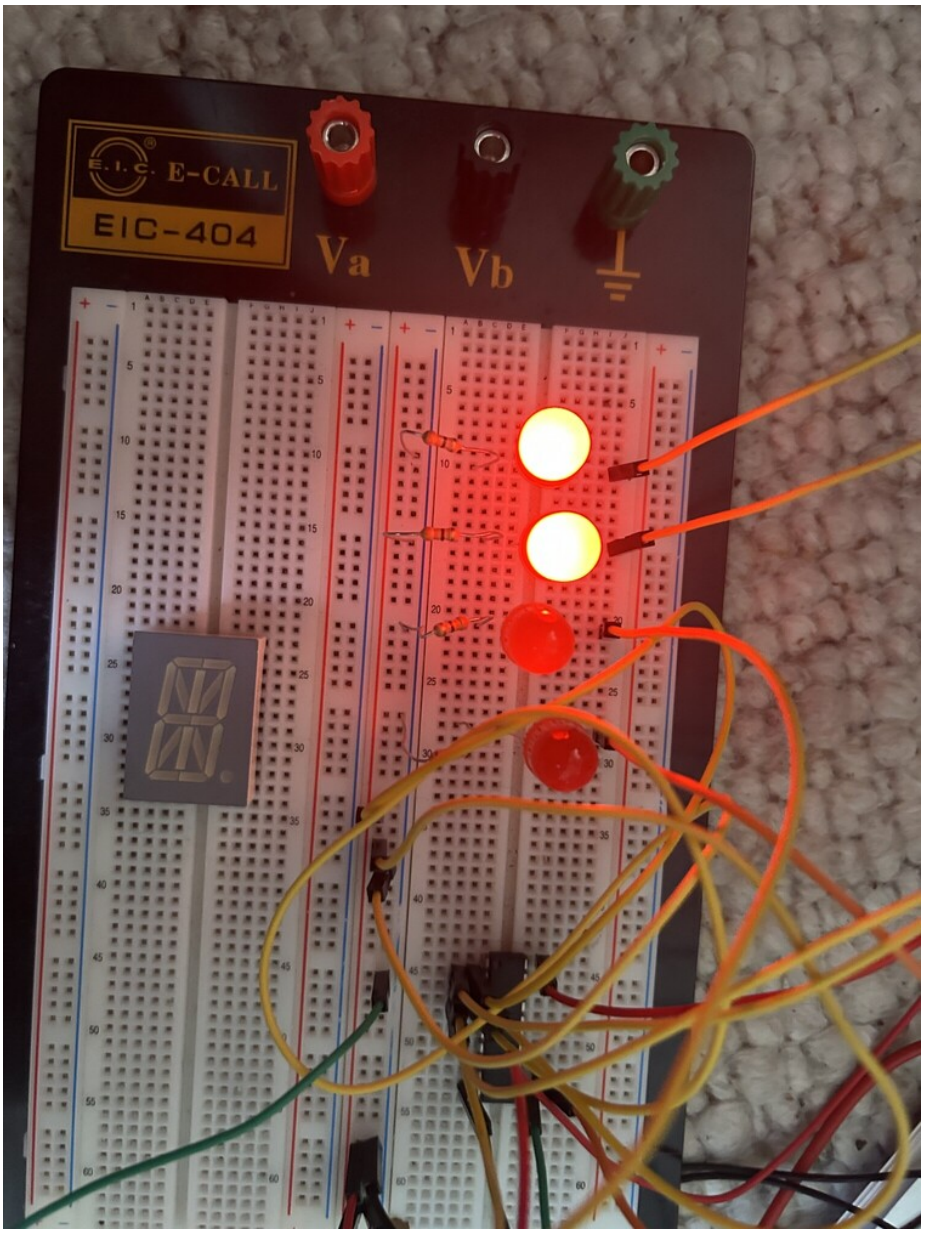

E-CALL
EIC-404
Va
Vb

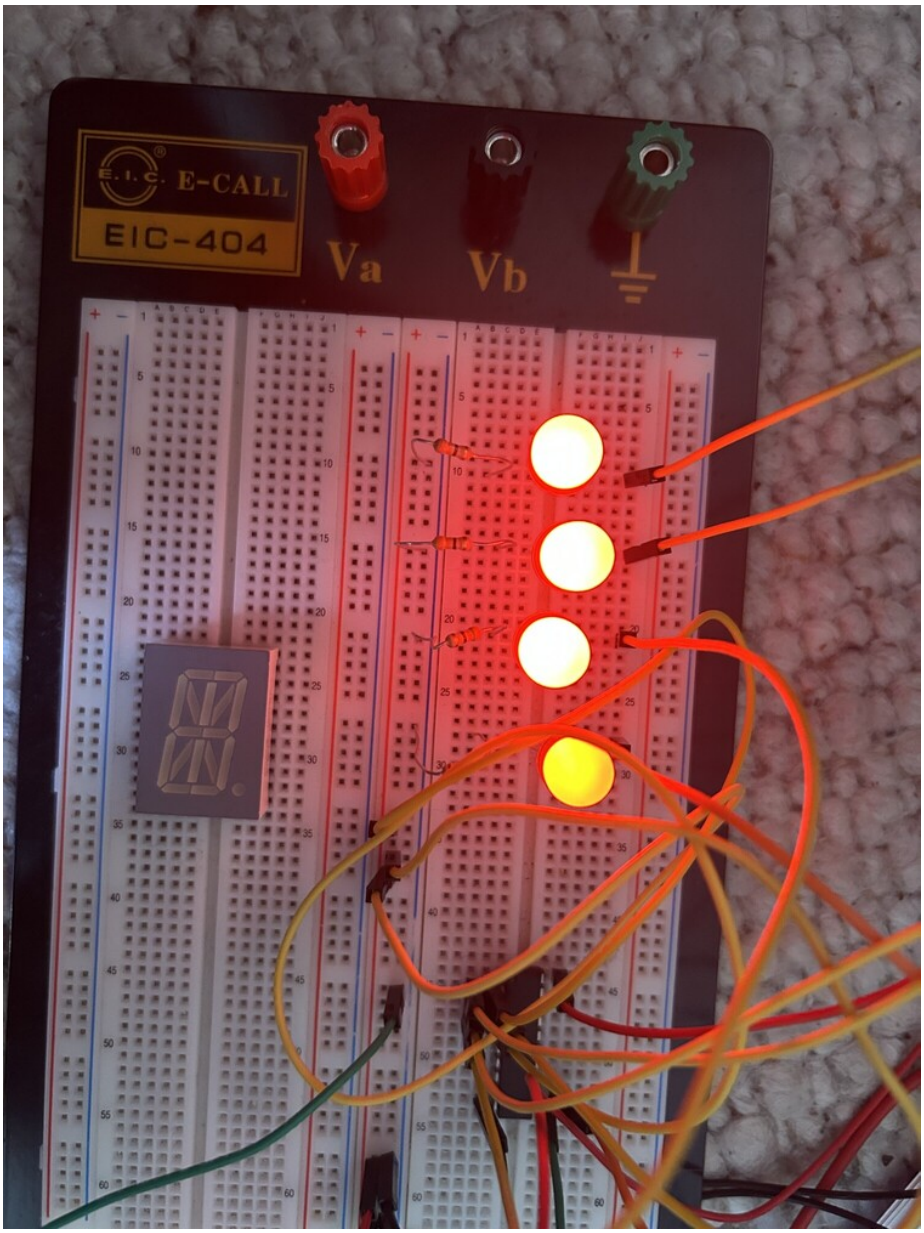

E-CALL
EIC-404
Va
Vb


E-CALL
EIC-404
Va
Vb


E-CALL
EIC-404
Va
Vb

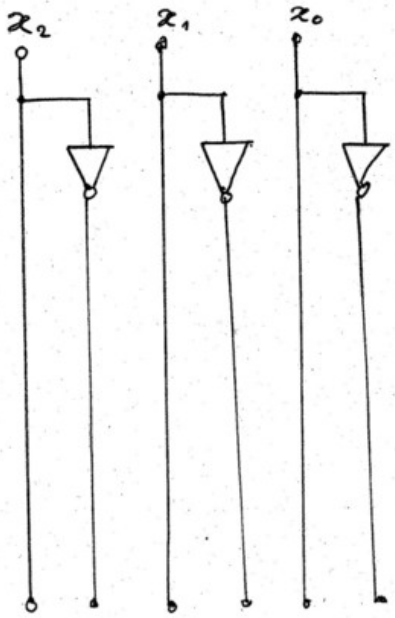

E-CALL
EIC-404
Va
Vb

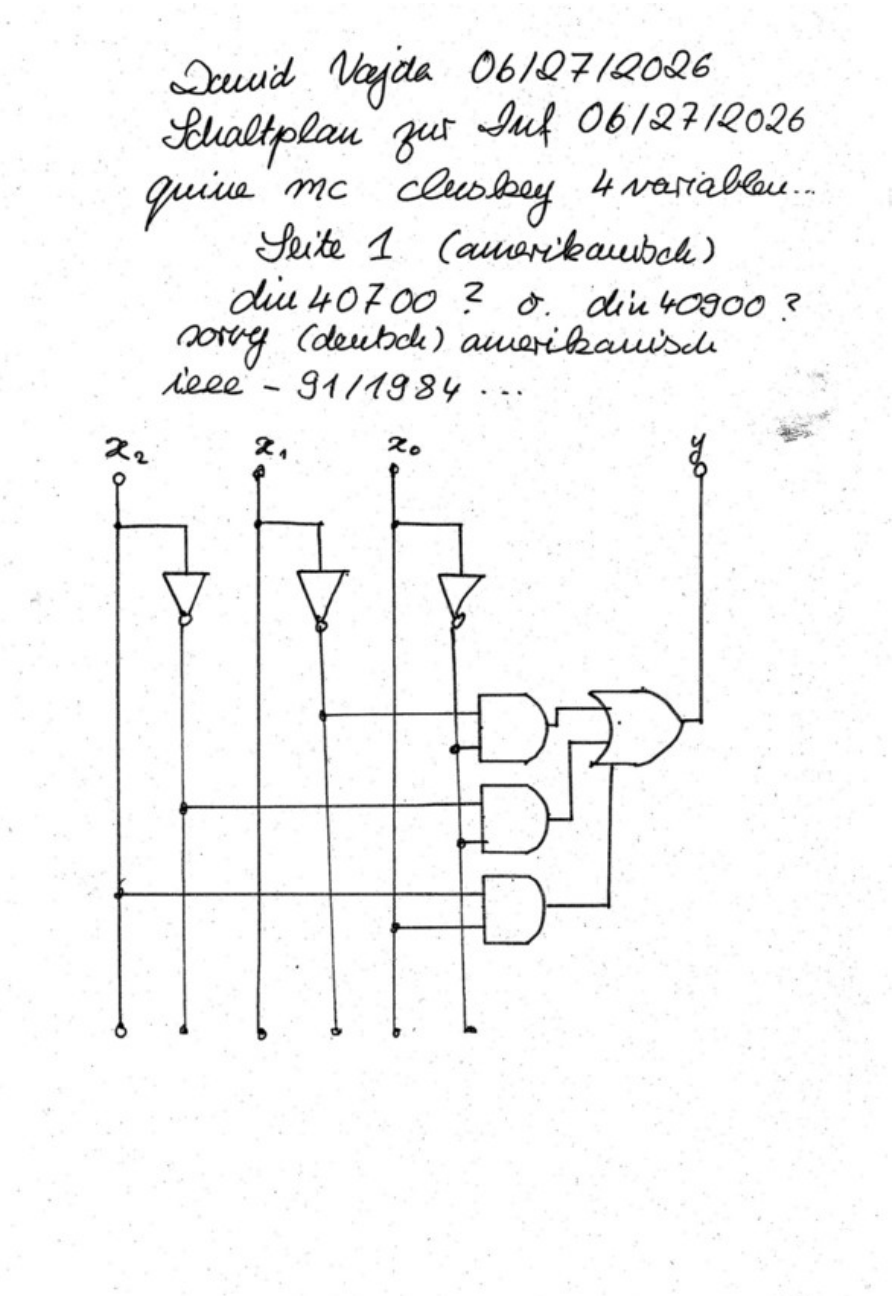

E-CALL
EIC-404
Va
Vb



So, jetzt muss ich noch mal den Baustein testen. Stimmt da irgendwas nicht?

Dawid Vajda 06/27/2026
Schaltplan zur Inf 06/27/2026
quine mc cluskey 4 variablen...
Seite 1 (amerikanisch)
din 40700 ? o. din 40900 ?
xorg (deutsch) amerikanisch
ieee - 91/1984 ...





so, jetzt probiere ich weil wohl eine zeile bei den gleichungen in der tabelle nicht funktioniert hat... muss ich noch mal gucken und die aufgabe soll vollstaendig sein, das heisst:

der atmega8 code, zaehlen, mit warteschleife, externen interrupt, zeitintervall muss her.

und wenn wir das haben: ach so der rest, mit samt diagramm `ieee 91-1984`, `din 40900` und so weiter und din 40700 ok. und dann alle beitraege zusammen.

Ich müsste noch mal die Gleichung analysieren, aber für heute ist mir die Lust vergangen. Danke schön. Ich bin froh, dass dieser GAL 16 V8 tut.

Wie gesagt, da muss ich das normale an der DNF weitermachen auch den Schaltplan aber den muss ich heute nicht mehr machen, mit der Hand und mit dem Computer und die Schaltung noch bauen

```
(C) David Vajda
Sat Jun 27 11:46:55 2026
3 Network - TTL - Disjunktive Normalform

 x2 x1 x0  y
 0 0 0 0  1
 1 0 0 1  0
 2 0 1 0  1
 3 0 1 1  0
 4 1 0 0  1
 5 1 0 1  1
 6 1 1 0  0
 7 1 1 1  1


 x2 x1 x0  y
 0 0 0 0  1
 2 0 1 0  1
 4 1 0 0  1
 5 1 0 1  1
 7 1 1 1  1


 x2 x1 x0  y
Gruppe 0:
 0 0 0 0  1
Gruppe 1:
 2 0 1 0  1
 4 1 0 0  1
Gruppe 2:
 5 1 0 1  1
Gruppe 3:
 7 1 1 1  1


 x2 x1 x0  y
0:2 0 - 0
0:4 - 0 0
4:5 1 0 -
5:7 1 - 1
```

```
 x2 x1 x0  y
0:4 - 0 0
0:2 0 - 0
5:7 1 - 1
4:5 1 0 -
```

```
Primimplikantentafel:

 0 2 4 5 7
0:4 x  x
0:2 x x
5:7    x x
4:5   x x
```

```
Primimplikantentafel, welcher muss auf jeden fall bleiben?
 0 2 4 5 7
0:4 x  x
0:2 x x
5:7    x x p
4:5   x x
```

Gibt es irgendwelche gruende diese oder jene Gleichung Zu streichen? Etwa in Prozenten? Eine Entdeckung? Immerhin: `<CR><LF>` oder `<LF><CR>` eigentlich wuerde doch Beides? Ich dachte: wann überlebt die Schreibmaschine, wg. Wagen am Rand, mehr oder ... Der/die/das Schreiber vielleicht - wo ist mehr klecks? Klecks im nächsten, wird ueberschrieben, oder nicht frische Tinte? Was auch immer... wir koennen ja jetzt fragen an die Prozente die Wahrscheinlichkeitsrechnung stellen?

Was ist denn jetzt klüger, diese variable dieses UND/ODER Gatter bevorzugt oder dieses? Vielleicht weiss es ja jemand? Wie beim Schach? Oder wir wissen es einfach nicht!

Wir wissen es einfach nicht, weil jemand sagen wuerde, oh Vajda! Wie beim Schach!

Weil: ich sage mal so: woher wissen wir? Dass es nicht teil Unserer aufgäbe ist? Nicht teil unserer aufgäbe ist, Einteilung Fast wie Prozente, blos: ich glaube wir muessen das mal einsehen, Dass auch die uni bis ins 19. jhdt. Ziemlich langsam war. zunaechst. Vielleicht gewöhnen wir uns daran...

Fanatisch und perfektionistisch Fanatismus und Perfektionismus blos wir

koennen ja mit dem leben. Aber es ist ein wenig frisch. Und wir muessen erst mal 2000 jähre damit Leben ohne das Gedudel anderer menschen und sei es auf der Strasse. Dann machen wir es wenn schon ordentlich selbst, ohne Anspruch, Andere zu überzeugen. Weil: schon das Gedudel stimuliert sie. Deswegen Wahrscheinlich sagen manche leute ist Strassenmusik schlecht, sie genügt Keinem Anspruch.

Warum? Weil sie schoen ist, wie die Musik im Radio. Noch schöner, wenn gut Schleicht sich da wieder was ein? Ja, leider. Warum? Weil Musik schlecht klingen muss, auch wenn man die dauernd übt. Dann ist gut geübte schlechte Musik, wenn selbst, besser. Warum? Warum ganz einfach: sie kennen ja diese komischen Tänzerinnen im 19. jhdt. Was da auf der Strasse jetzt auf ein mal so schon dudelt gerade, das ist teil Der Sache. Ein fortschritt zum Radio von gerade eben? Nein, leider nicht! Wenn man wo anders hin muss. . . sie wissen Und ich sage: wenn sie weiter schöne Musik spielen, dann wird die welt leider Bald nicht mehr schoen sein, gemeint sind nicht, unschöne Unterredungen im Unschönen Büro. Dass sie noch Musik spielen koennen verdanken sie lange gespielter Unschöner Musik. Das ist das problem. Sie muessen sich alle erst mal sehr lange Gewöhnen und das ist glaube ich das ziel in der Wissenschaft der Zukunft, die jetzt Anfangen sollte und sehr lange so gehen sollte.

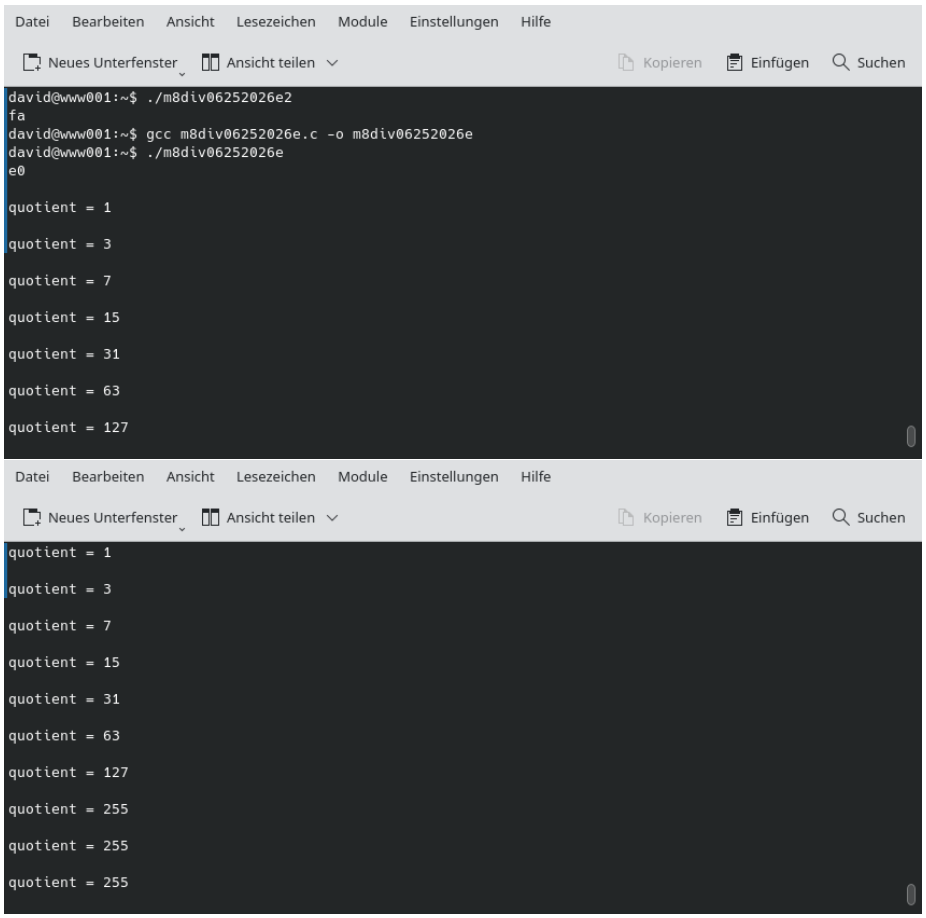
```
 0 2 4 5 7
0:4 x  x
0:2 x x
5:7    x x p

 x2 x1 x0  y
0:4 - 0 0
0:2 0 - 0
5:7 1 - 1

 y <= (not x1 and not x0) or
   (not x2 and not x0) or
   (x2 and x0);

-- vhdl\dots
-- (C) David Vajda
-- Sat Jun 27 11:46:55 2026
-- 3 Network - TTL - Disjunktive Normalform

library ieee;
use ieee.std_logic_1164.all;

entity quine06272026 is
port (
 x2, x1, x0: in std_logic;
 y: out std_logic
);
end;
```

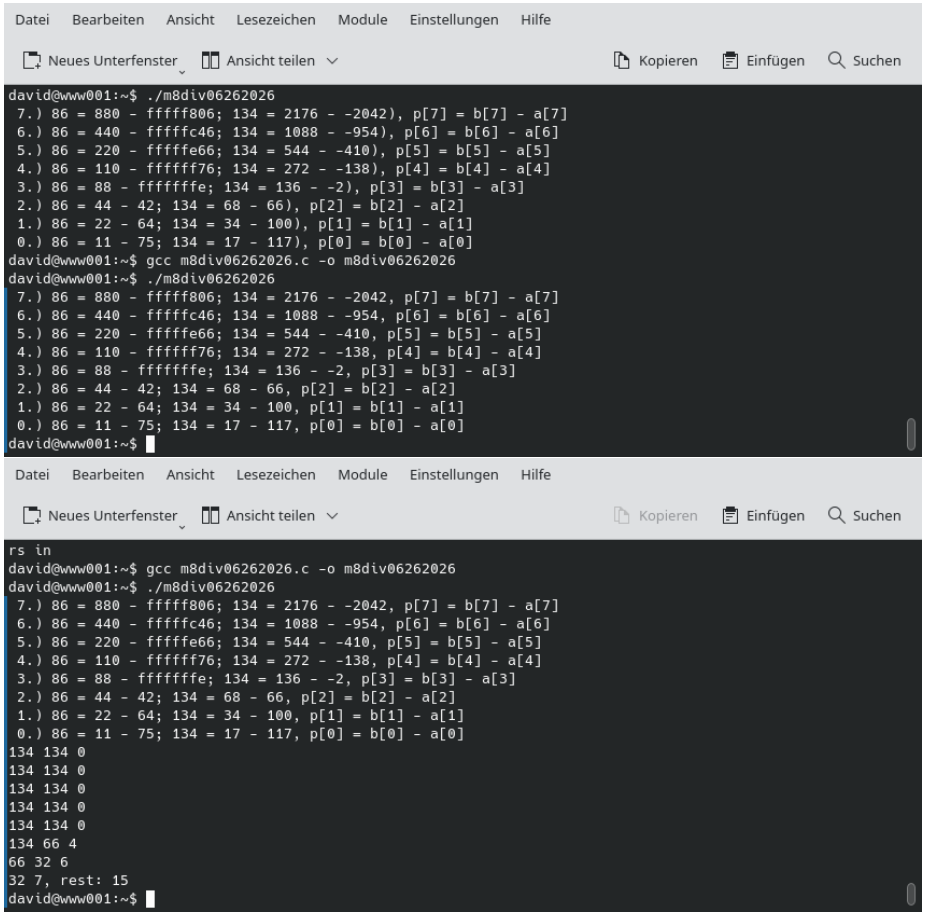
```
architecture behaviour of quine06272026 is
begin
 y <= (not x1 and not x0) or
   (not x2 and not x0) or
   (x2 and x0);
end;

library ieee;
use ieee.std_logic_1164.all;

entity quine06272026debug is
port (
 y: out std_logic
);
end;

architecture behaviour of quine06272026debug is
 component quine06272026
 port (
  x2, x1, x0: in std_logic;
  y: out std_logic
 );
 end component;
 signal x2, x1, x0: std_logic
begin
 q: quine06272026 PORT MAP (x2=>x2, x1=>x1, x0=>x0, y=>y);

 x0 <= '0' after 0 ns, '1' after 10 ns, '0' after 20 ns, '1' after 30 ns, '0' after 40 ns,

 x1 <= '0' after 0 ns, '0' after 10 ns, '1' after 20 ns, '1' after 30 ns, '0' after 40 ns,

 x2 <= '0' after 0 ns, '0' after 10 ns, '0' after 20 ns, '0' after 30 ns, '1' after 40 ns,
end;

(C) David Vajda
Sat Jun 27 11:46:55 2026
3 Network - TTL - Disjunktive Normalform

 x2 x1 x0  y
 0 0 0 0  1
 1 0 0 1  0
 2 0 1 0  1
 3 0 1 1  0
 4 1 0 0  1
 5 1 0 1  1
 6 1 1 0  0
 7 1 1 1  1
```

ergibt $19 \cdot 11 + 3 = 212$ (korrekt) und $17 \cdot 7 + 15 = 134$ (korrekt) ...

```
// david vajda
// 06/26/2026
// divisionsoperation like network one operation

#include <stdio.h>

#define DIVIDEND    134
#define DIVISOR     17

int main (void) {
    int b [8];
    int a [8];
    int q = 0;
    int r;
    int p [8];  // (debug modus fuer q)

    int i;

    for (i = 7;  i >= 0;  i--)
        b [i] = DIVIDEND;
    for (i = 7;  i >= 0;  i--)
        a [i] = DIVISOR << i;

    /*
     *  jetzt haben wir:
     *  b7,
     *  b6,
     *  b5,
     *  ...
     *  b0
     *  mit fuehrenden nullen
     *  dann kommt als naechstes
     *  a7,
     *  a6,
     *  ...
     *  a0
     *
     *  die subtrahieren wir alle einzeln voneinander und gucken
     *  uns das ergebnis in einer art 1. debug modus an...
     */

    for (i = 7;  i >= 0;  i--)
        p [i] = b [i] - a [i];

    for (i = 7;  i >= 0;  i--)
        printf ("%2i.) %x = %x - %x; %i = %i - %i, p[%i] = b[%i] - a[%i]\n", i, b[i], a[i]

    for (i = 7;  i >= 1;  i--) {
        if (b [i] - a[i] >= 0) {
            b [i-1] = b [i] - a[i];
```

```
            q = q + (1 << i);
        }
        else
            b [i-1] = b [i];
        printf ("%i %i %i\n", b[i], b[i-1], q);
    }
    if (b [0] - a[0] >= 0) {
        q = q + (1 << 0);
        r = b [0] - a[0];
    }
    else
        r = b [0];

     printf ("%i %i, rest: %i\n", b[0], q, r);

return 0;
}
```






```
for (i = 7;  i >= 1;  i--) {
        if (b [i] - a[i] >= 0) {
            b [i-1] = b [i] - a[i];
            q = q + (1 << i);
        }
        else
            b [i-1] = b [i];
        printf ("%i %i %i\n", b[i], b[i-1], q);
    }
    if (b [0] - a[0] >= 0)
        q = q + (1 << 0);
     printf ("%i %i\n", b[0], q);
```

```
// david vajda
// 06/26/2026
// divisionsoperation like network one operation

#include <stdio.h>

#define DIVIDEND    134
#define DIVISOR     17
```

```
int main (void) {
    int b [8];
    int a [8];
    int q;
    int p [8];  // (debug modus fuer q)

    int i;

    for (i = 7;  i >= 0;  i--)
        b [i] = DIVIDEND;
    for (i = 7;  i >= 0;  i--)
        a [i] = DIVISOR << i;

    /*
     *  jetzt haben wir:
     *  b7,
     *  b6,
     *  b5,
     *  ...
     *  b0
     *  mit fuehrenden nullen
     *  dann kommt als naechstes
     *  a7,
     *  a6,
     *  ...
     *  a0
     *
     *  die subtrahieren wir alle einzeln voneinander und gucken
     *  uns das ergebnis in einer art 1. debug modus an...
     */

    for (i = 7;  i >= 0;  i--)
        p [i] = b [i] - a [i];

    for (i = 7;  i >= 0;  i--)
        printf ("%2i.) %x - %x = %x (%i), p[%i], b[%i] - a[%i]\n", i, b[i], a[i], p [i], i

return 0;
}
```

Divisions Operation binary + atmega8 commands and co

david vajda

10. Juli 2026

1 teilschritt

								b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0							
		a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0						
			a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0					
				a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0				
					a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0			
						a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0		
							a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	
								a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0

2 teilschritt

								b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
x	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0							
	x	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0						
		x	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0					
			x	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0				
				x	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0			
					x	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0		
						x	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	
							x	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0

3 teilschritt

								b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0		
x	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	s	q_7							
	x	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	s	q_6						
		x	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	s	q_5					
			x	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	s	q_4				
				x	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	s	q_3			
					x	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	s	q_2		
						x	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	s	q_1	
							x	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	s	q_0

4 teilschritt

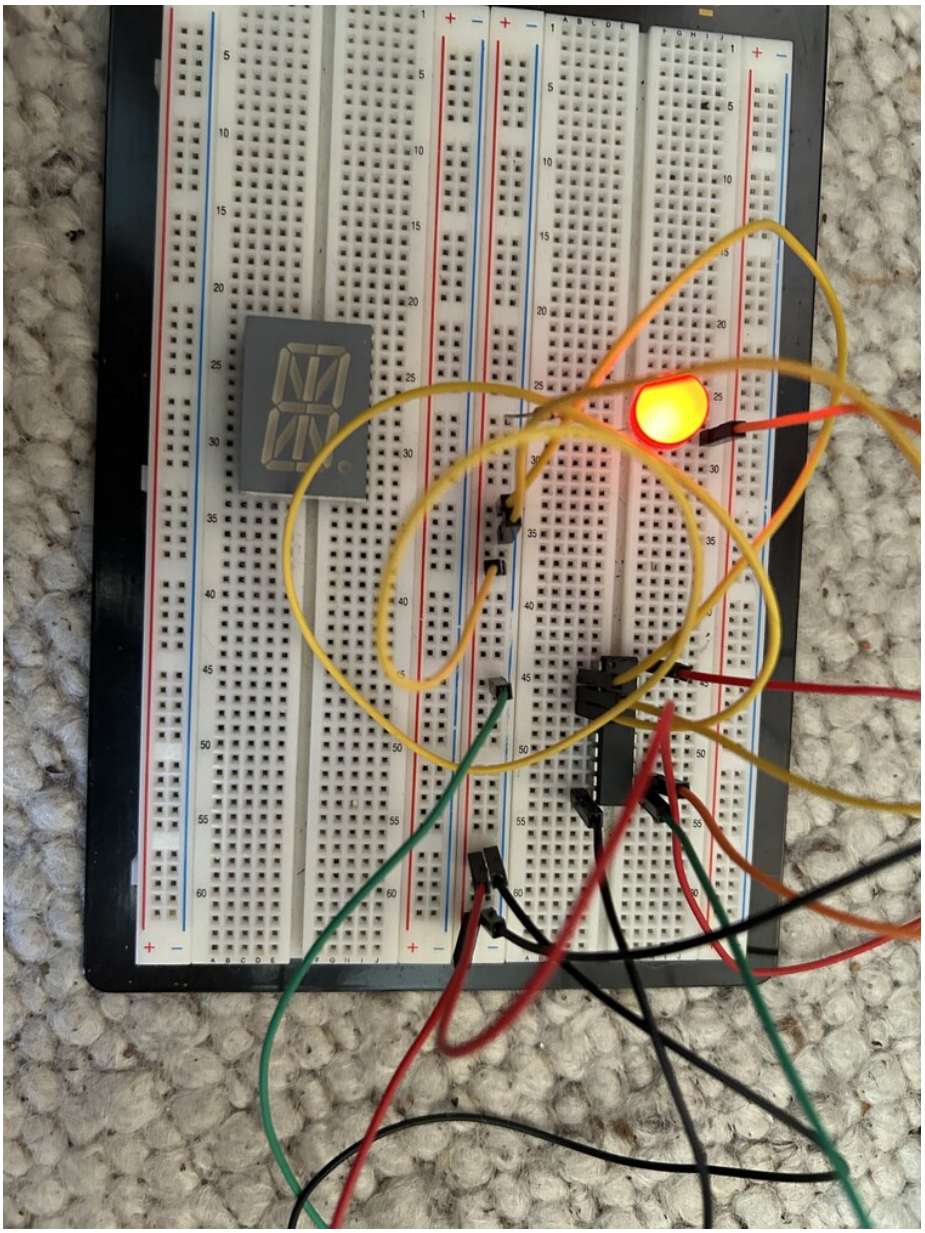
								b_{07}	b_{06}	b_{05}	b_{04}	b_{03}	b_{02}	b_{01}	b_{00}	
x	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	s	q_7						
								b_{17}	b_{16}	b_{15}	b_{14}	b_{13}	b_{12}	b_{11}	b_{10}	
	x	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	s	q_6					
								b_{27}	b_{26}	b_{25}	b_{24}	b_{23}	b_{22}	b_{21}	b_{20}	
		x	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	s	q_5				
								b_{37}	b_{36}	b_{35}	b_{34}	b_{33}	b_{32}	b_{31}	b_{30}	
			x	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	s	q_4			
								b_{47}	b_{46}	b_{45}	b_{44}	b_{43}	b_{42}	b_{41}	b_{40}	
				x	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	s	q_3		
								b_{57}	b_{56}	b_{55}	b_{54}	b_{53}	b_{52}	b_{51}	b_{50}	
					x	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	s	q_2	
								b_{67}	b_{66}	b_{65}	b_{64}	b_{63}	b_{62}	b_{61}	b_{60}	
						x	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	s	q_1
								b_{77}	b_{76}	b_{75}	b_{74}	b_{73}	b_{72}	b_{71}	b_{70}	
							x	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	s	q_0

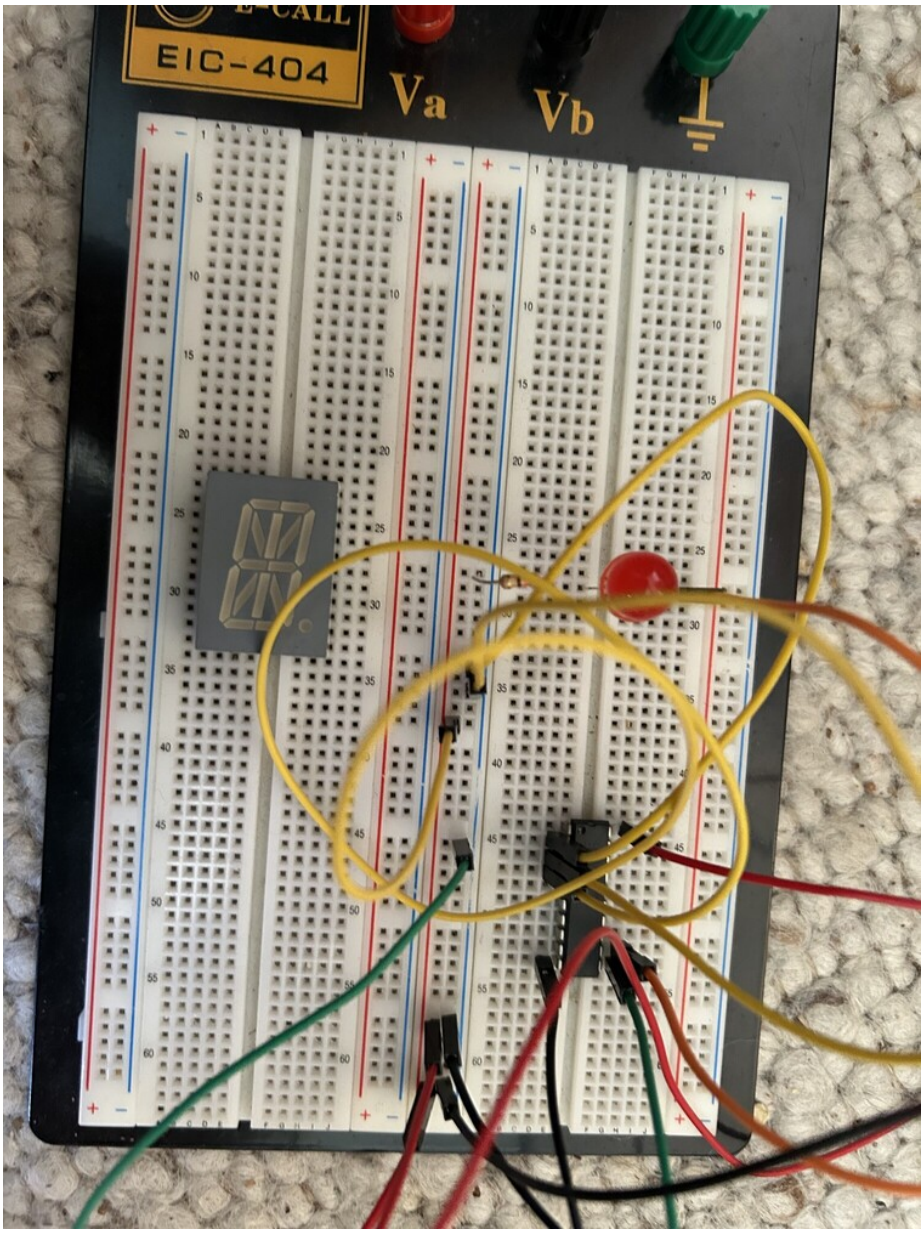
5 teilschritt

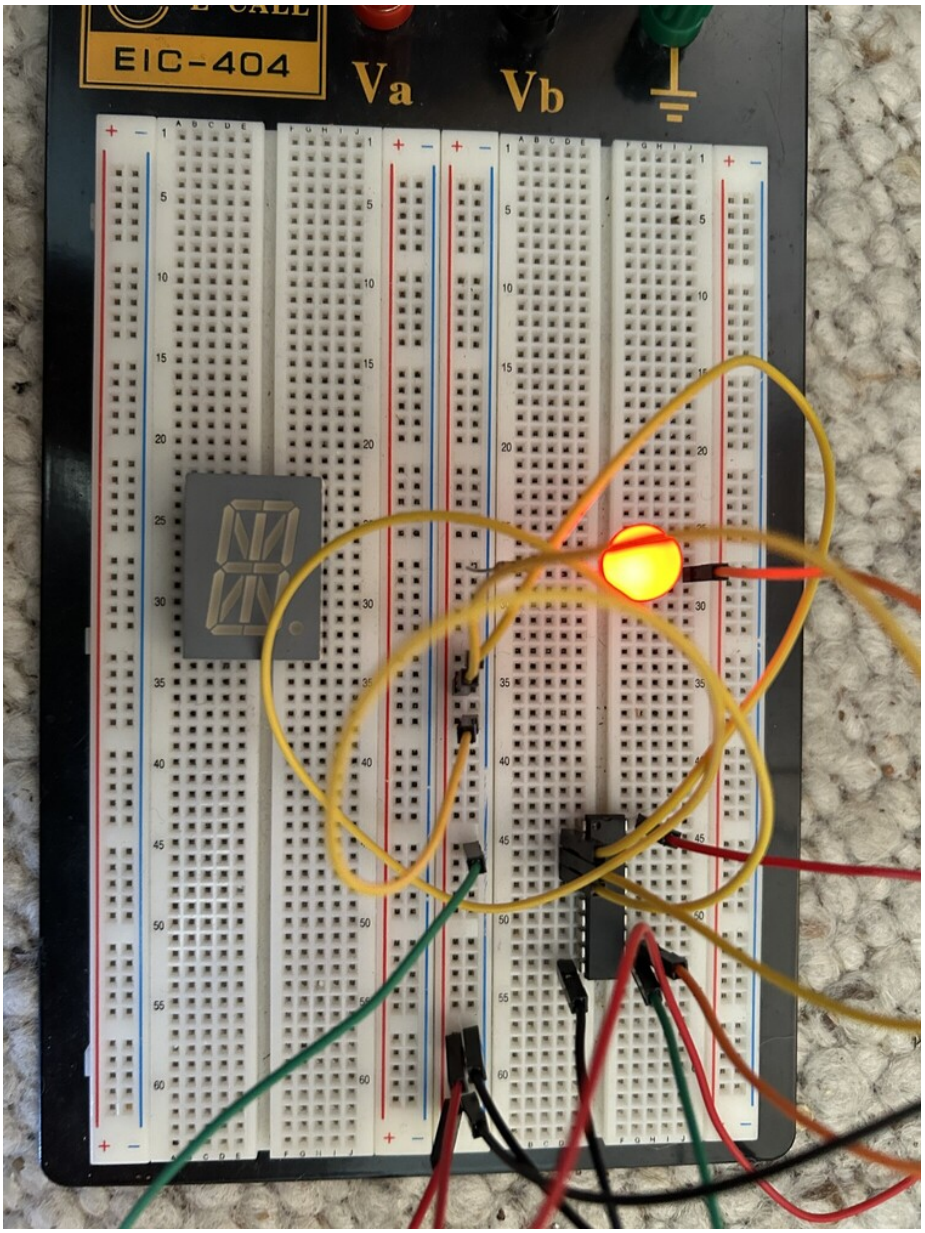
								b_{07}	b_{06}	b_{05}	b_{04}	b_{03}	b_{02}	b_{01}	b_{00}	
x_7	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	s_7	q_7						
								b_{17}	b_{16}	b_{15}	b_{14}	b_{13}	b_{12}	b_{11}	b_{10}	
	x_6	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	s_6	q_6					
								b_{27}	b_{26}	b_{25}	b_{24}	b_{23}	b_{22}	b_{21}	b_{20}	
		x_5	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	s_5	q_5				
								b_{37}	b_{36}	b_{35}	b_{34}	b_{33}	b_{32}	b_{31}	b_{30}	
			x_4	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	s_4	q_4			
								b_{47}	b_{46}	b_{45}	b_{44}	b_{43}	b_{42}	b_{41}	b_{40}	
				x_3	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	s_3	q_3		
								b_{57}	b_{56}	b_{55}	b_{54}	b_{53}	b_{52}	b_{51}	b_{50}	
					x_2	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	s_2	q_2	
								b_{67}	b_{66}	b_{65}	b_{64}	b_{63}	b_{62}	b_{61}	b_{60}	
						x_1	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	s_1	q_1
								b_{77}	b_{76}	b_{75}	b_{74}	b_{73}	b_{72}	b_{71}	b_{70}	
							x_0	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	s_0	q_0

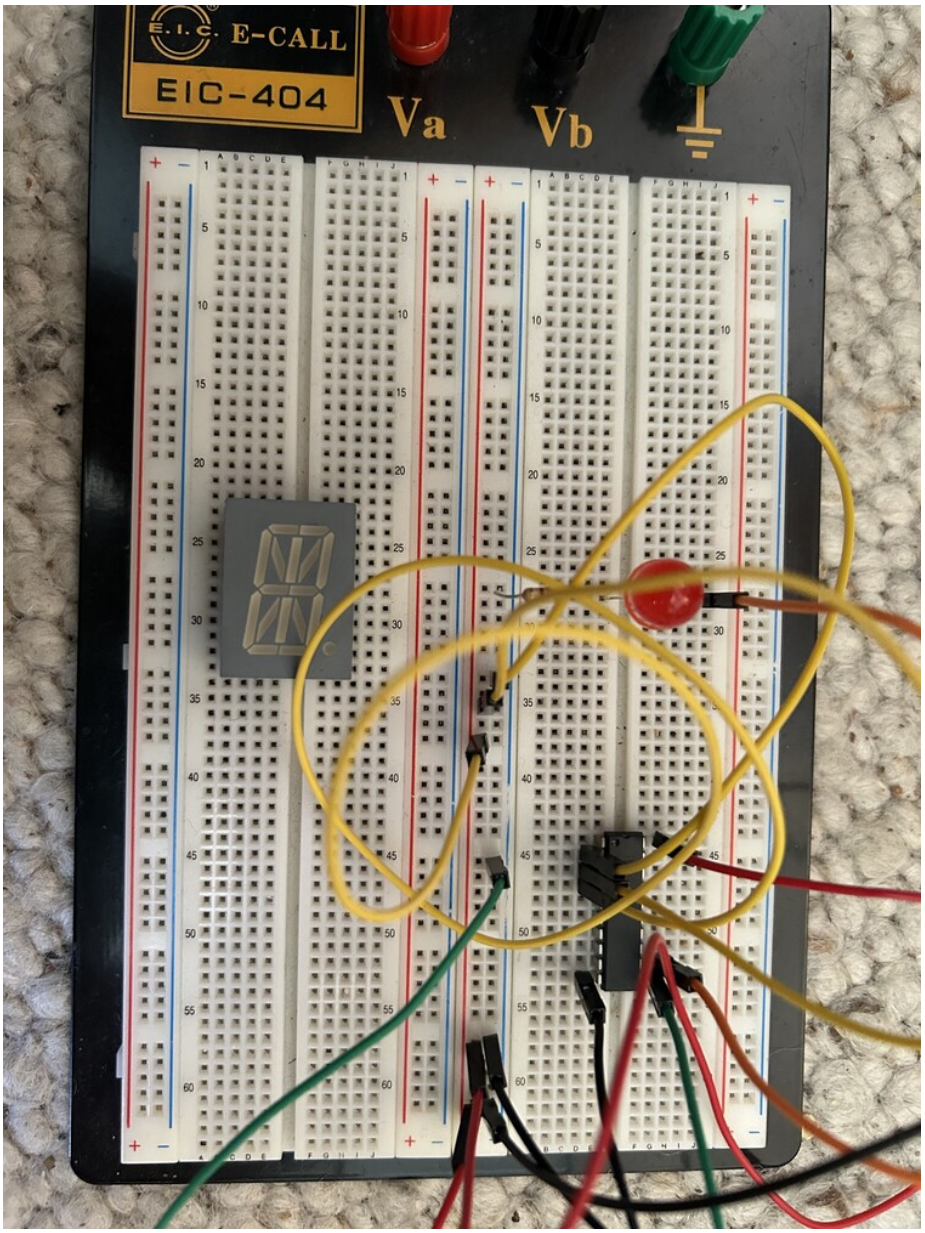
6 teilschritt

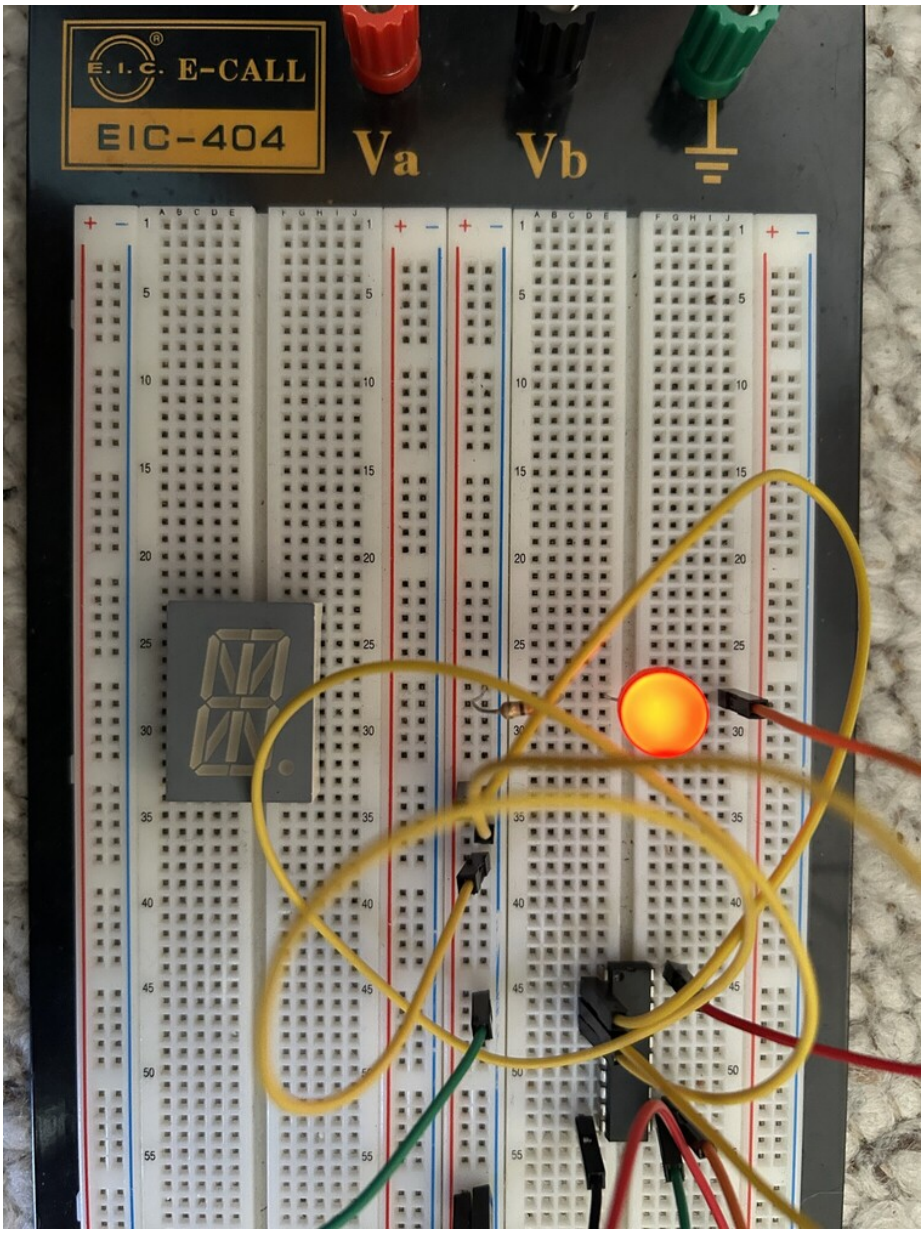
L	L	L	L	L	L	L	L	b_{07}	b_{06}	b_{05}	b_{04}	b_{03}	b_{02}	b_{01}	b_{00}	
x_7	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	s_7	q_7						
L	L	L	L	L	L	L	L	b_{17}	b_{16}	b_{15}	b_{14}	b_{13}	b_{12}	b_{11}	b_{10}	
	x_6	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	s_6	q_6					
L	L	L	L	L	L	L	L	b_{27}	b_{26}	b_{25}	b_{24}	b_{23}	b_{22}	b_{21}	b_{20}	
		x_5	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	s_5	q_5				
L	L	L	L	L	L	L	L	b_{37}	b_{36}	b_{35}	b_{34}	b_{33}	b_{32}	b_{31}	b_{30}	
			x_4	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	s_4	q_4			
L	L	L	L	L	L	L	L	b_{47}	b_{46}	b_{45}	b_{44}	b_{43}	b_{42}	b_{41}	b_{40}	
				x_3	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	s_3	q_3		
L	L	L	L	L	L	L	L	b_{57}	b_{56}	b_{55}	b_{54}	b_{53}	b_{52}	b_{51}	b_{50}	
					x_2	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	s_2	q_2	
L	L	L	L	L	L	L	L	b_{67}	b_{66}	b_{65}	b_{64}	b_{63}	b_{62}	b_{61}	b_{60}	
						x_1	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	s_1	q_1
L	L	L	L	L	L	L	L	b_{77}	b_{76}	b_{75}	b_{74}	b_{73}	b_{72}	b_{71}	b_{70}	
							x_0	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	s_0	q_0

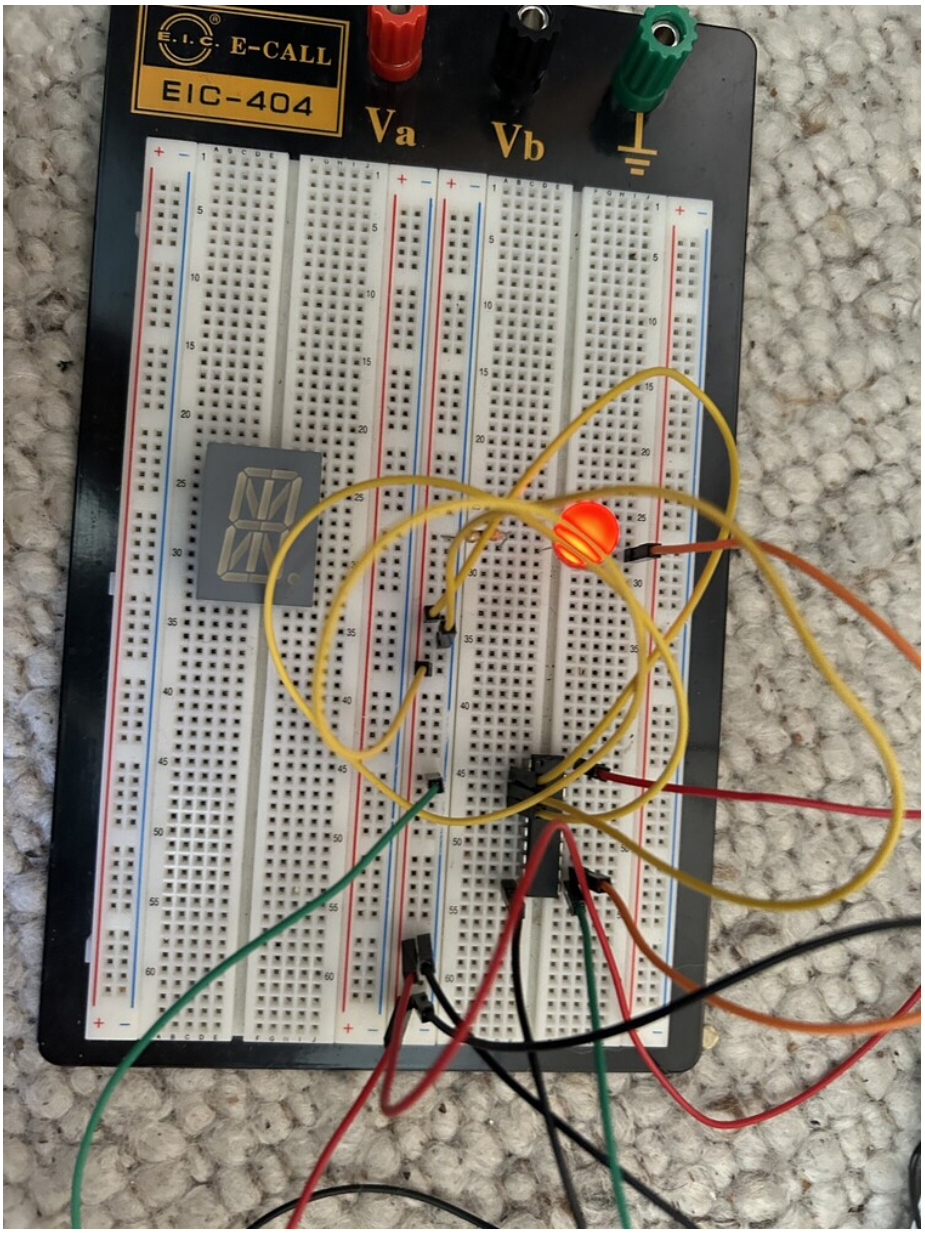


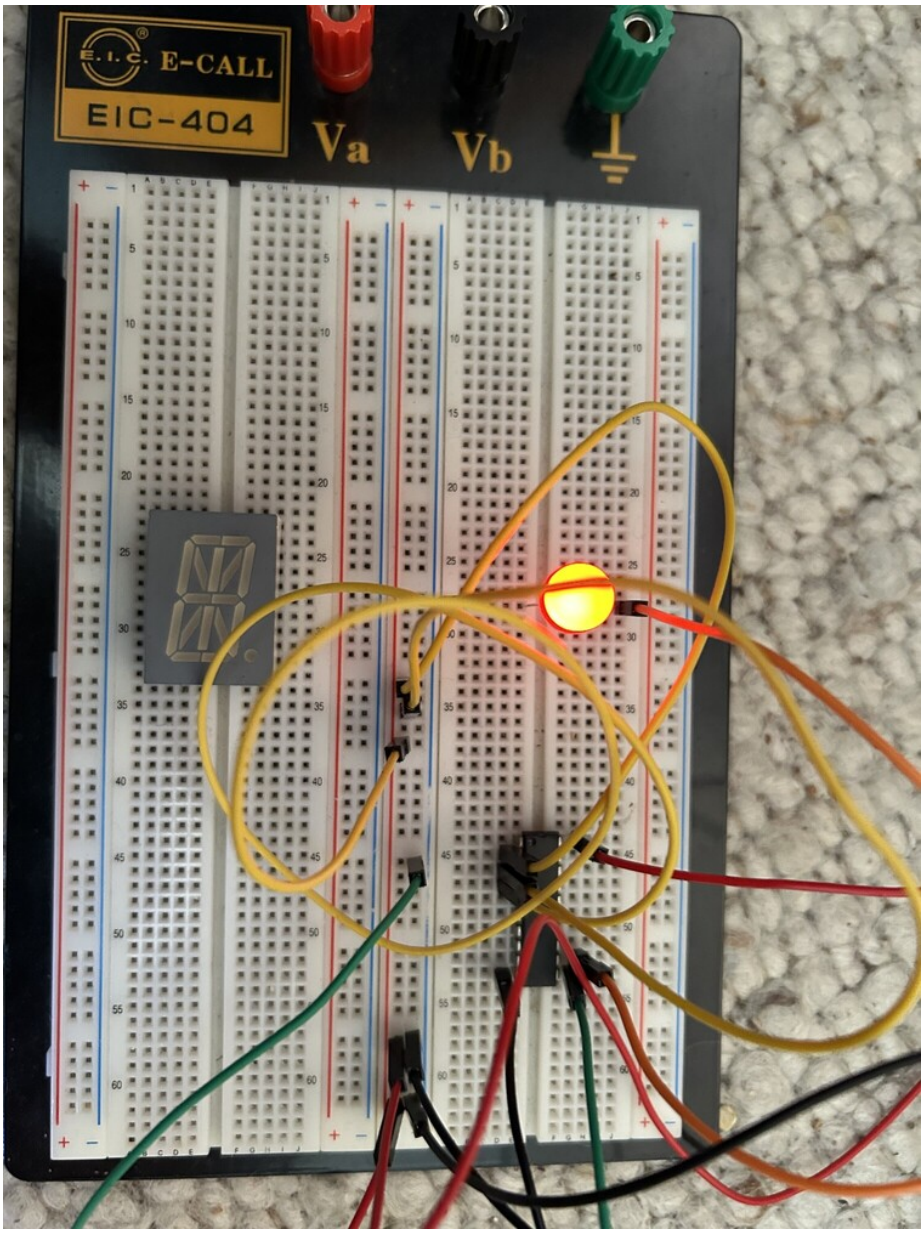

EIC-404
Va
Vb


EIC-404
Va
Vb


E.I.C. E-CALL
EIC-404
Va
Vb


E.I.C.
E-CALL
EIC-404
Va
Vb


E-CALL
EIC-404
Va
Vb



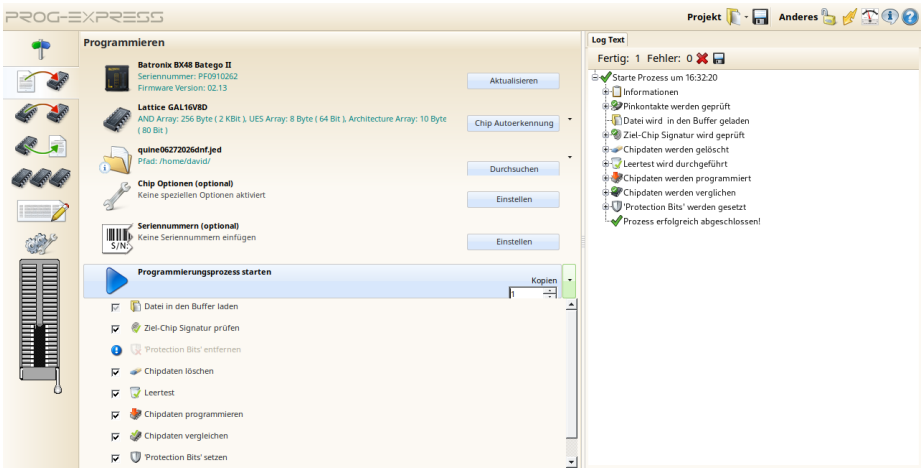
ich zeige die photos von der dnf - und dann gucken sie.

ich habe es hingekriegt, mit dem GAL 16V8

von lattice, es hat getan!!!

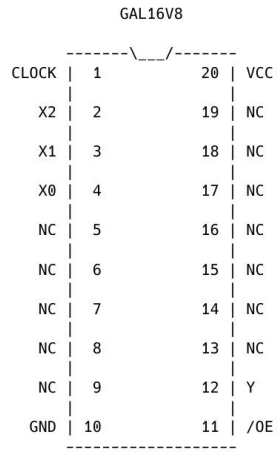
fantastisch!

der fehler in der vergangenheit war - diese schaltung nicht an zu gucken, ich habe die sache programmiert wie im quelltext und sie tut...


PROG-EXPRESS
Projekt
Anderes
Programmieren
Batronix BX48 Batego II
Seriennummer: PF0910262
Firmware Version: 02.13
Aktualisieren
Lattice GAL16V8D
AND Array: 256 Byte (2 KBit), UES Array: 8 Byte (64 Bit), Architecture Array: 10 Byte (80 Bit)
Chip Autoerkennung
quine06272026dnf.jed
Pfad: /home/david/
Durchsuchen
Chip Optionen (optional)
Keine speziellen Optionen aktiviert
Einstellen
Seriennummern (optional)
Keine Seriennummern einfügen
Einstellen
Programmierungsprozess starten
Kopien
Datei in den Buffer laden
Ziel-Chip Signatur prüfen
'Protection Bits' entfernen
Chipdaten löschen
Leertest
Chipdaten programmieren
Chipdaten vergleichen
'Protection Bits' setzen
Log Text
Fertig: 1 Fehler: 0
Starte Prozess um 16:32:20
Informationen
Pinkontakte werden geprüft
Datei wird in den Buffer geladen
Ziel-Chip Signatur wird geprüft
Chipdaten werden gelöscht
Leertest wird durchgeführt
Chipdaten werden programmiert
Chipdaten werden verglichen
'Protection Bits' werden gesetzt
Prozess erfolgreich abgeschlossen!

```
Pin 19 = NC           XOR = 0   AC1 = 0
  0  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
  1  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
  2  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
  3  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
  4  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
  5  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
  6  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
  7  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Pin 18 = NC           XOR = 0   AC1 = 0
  8  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
  9  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 10  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 11  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 12  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 13  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 14  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 15  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Pin 17 = NC           XOR = 0   AC1 = 0
 16  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 17  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 18  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 19  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 20  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 21  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 22  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 23  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Pin 16 = NC           XOR = 0   AC1 = 0
 24  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 25  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 26  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 27  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 28  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 29  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 30  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 31  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Pin 15 = NC           XOR = 0   AC1 = 0
 32  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 33  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 34  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 35  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 36  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 37  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 38  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 39  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Pin 14 = NC           XOR = 0   AC1 = 0
 40  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 41  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 42  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 43  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 44  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 45  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 46  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 47  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
```

```
Pin 13 = NC           XOR = 0   AC1 = 0
 48  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 49  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 50  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 51  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 52  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 53  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 54  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 55  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Pin 12 = Y            XOR = 1   AC1 = 0
 56  ---- -x-- -x-- ---- ---- ---- ---- ----
 57  -x-- ---- -x-- ---- ---- ---- ---- ----
 58  x--- ---- x--- ---- ---- ---- ---- ----
 59  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 60  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 61  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 62  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 63  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
```

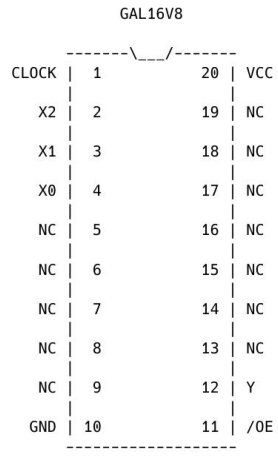


ich habe es hingekriegt, mit dem GAL 16V8
von lattice, es hat getan!!!
fantastisch!

der fehler in der vergangenheit war - diese schaltung nicht an zu gucken, ich habe die sache programmiert wie im quelltext und sie tut...

```
Pin 19 = NC           XOR = 0   AC1 = 0
  0  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
  1  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
  2  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
  3  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
  4  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
  5  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
  6  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
  7  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Pin 18 = NC           XOR = 0   AC1 = 0
  8  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
  9  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 10  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 11  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 12  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 13  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 14  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 15  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Pin 17 = NC           XOR = 0   AC1 = 0
 16  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 17  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 18  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 19  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 20  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 21  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 22  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 23  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Pin 16 = NC           XOR = 0   AC1 = 0
 24  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 25  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 26  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 27  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 28  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 29  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 30  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 31  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Pin 15 = NC           XOR = 0   AC1 = 0
 32  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 33  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 34  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 35  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 36  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 37  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 38  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 39  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Pin 14 = NC           XOR = 0   AC1 = 0
 40  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 41  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 42  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 43  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 44  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 45  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 46  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 47  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
```

```
Pin 13 = NC           XOR = 0   AC1 = 0
 48  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 49  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 50  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 51  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 52  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 53  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 54  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 55  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Pin 12 = Y            XOR = 1   AC1 = 0
 56  ---- -x-- -x-- ---- ---- ---- ---- ----
 57  -x-- ---- -x-- ---- ---- ---- ---- ----
 58  x--- ---- x--- ---- ---- ---- ---- ----
 59  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 60  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 61  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 62  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 63  xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
```



also, ich habe jetzt nicht orignal abel genommen, sondern galasm das ist die eigentliche alternative fuer linux. . . so wie sie eigentlich sein sollte, hier der link

https://github.com/daveho/GALasm

der quelltext sieht nun so aus:

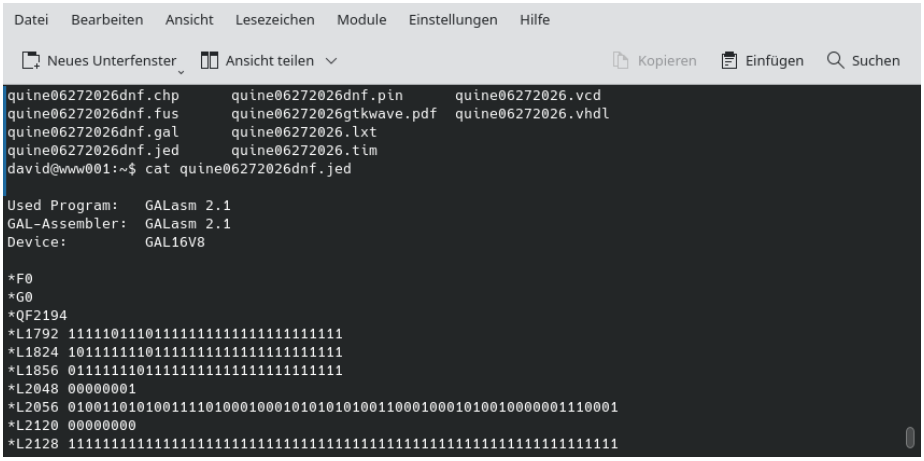
```
GAL16V8
MODULE quine06272026dnf

    CLOCK X2 X1 X0 NC NC NC NC NC GND
```

```
    /OE Y NC NC NC NC NC NC NC VCC

    Y = /X1 * /X0 + /X2 * /X0 + X2 * X0

DESCRIPTION
This is dnf
```

und damit konnte ich ein jedec erzeugen, sieht so aus:
zunaechst ein chip sieht so aus:

```
                        GAL16V8

                      -------\___/-------
                CLOCK |  1            20 | VCC
                      |                  |
                   X2 |  2            19 | NC
                      |                  |
                   X1 |  3            18 | NC
                      |                  |
                   X0 |  4            17 | NC
                      |                  |
                   NC |  5            16 | NC
                      |                  |
                   NC |  6            15 | NC
                      |                  |
                   NC |  7            14 | NC
                      |                  |
                   NC |  8            13 | NC
                      |                  |
                   NC |  9            12 | Y
                      |                  |
                  GND | 10            11 | /OE
                      --------------------
```

```
Datei  Bearbeiten  Ansicht  Lesezeichen  Module  Einstellungen  Hilfe
Neues Unterfenster  Ansicht teilen  Kopieren  Einfügen  Suchen
Used Program:   GALasm 2.1
GAL-Assembler:  GALasm 2.1
Device:         GAL16V8

*F0
*G0
*QF2194
*L1792 11111011101111111111111111111111
*L1824 10111111101111111111111111111111
*L1856 01111111011111111111111111111111
*L2048 00000001
*L2056 0100110101001111010001000101010101001100010001010010000001110001
*L2120 00000000
*L2128 11111111111111111111111111111111111111111111111111111111111111111111
*L2192 1
*L2193 0
*C181b
*
54ff
david@www001:~$ pdftoppm
```

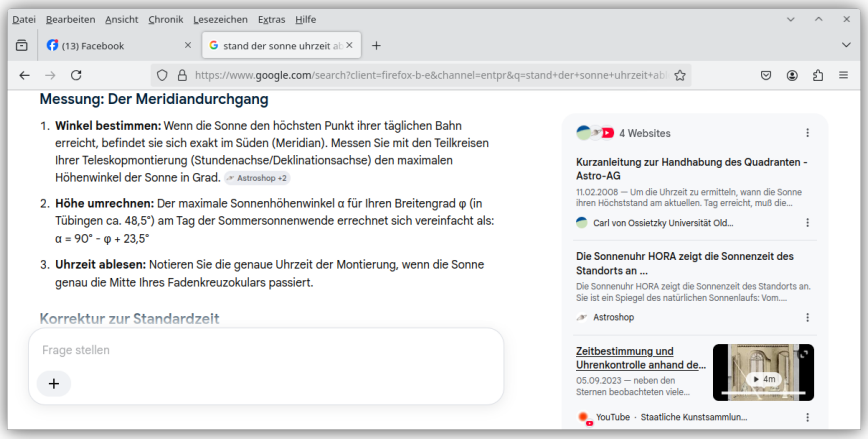


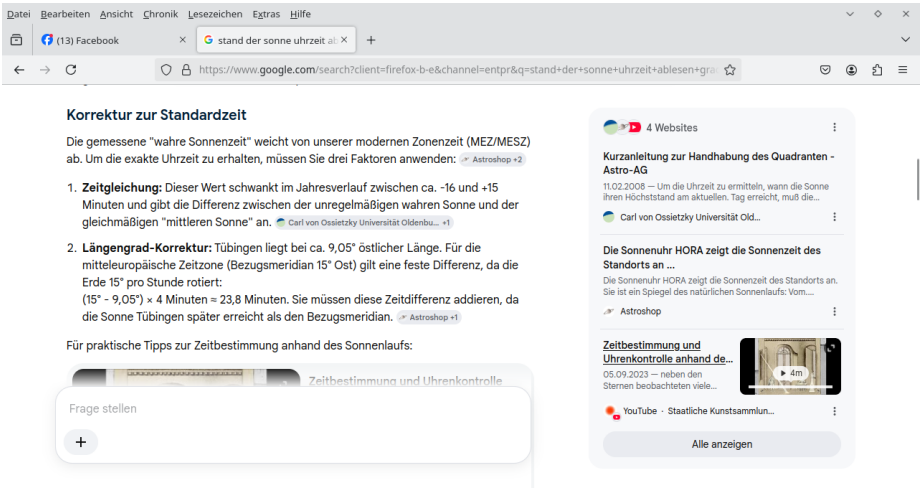
also, hier der richtige abel code:

```
MODULE quine06272026dnf

Declarations
    X2, X1, X0      pin 1, 2, 3     ISTYPE 'in';
    Y               pin 14          ISTYPE 'com';

Equations
    Y = (!X1 & !X0) # (!X2 & !X0) # (X2 & X0);

END
```

ich habe hier sogar ein beispiel von google ki das sehr genau beschreibt wie man eine DNF wirklich in Abel schreibt. das entspricht dem was ich moechte.

da ich die Gleichung wie bisher hinschreiben moechte, aber kein vhdl nutzen moechte nutze ich samt declarations in Zukunft diese Version, die mir google ki fuer eine DNF vorgeschlagen hat....





Also, ich probiere jetzt gänzlich meine Haltung, dem gegenüber zu ändern

Ich benutz kein VHDL und kein ghdl mehr solange es sich um das handelt

Das erste was ich mache, ist somit kein GTK Wave benutzen Was ja dazu, da ist das Impulsdiagramm darzustellen. Ich hab allerdings Schwierigkeiten.

einerseits sozusagen ohne des gtkwave auszukommen und auch des Python3 benutze ich nicht gern und ich kann das nicht konvertieren

Eine Aufgabe in Computersysteme eins und zwei war quasi die Schaltnetz selber zu analysieren quasi des Gegenteil von der Synthese und deswegen mache ich das jetzt per Hand

Und dazu mache ich durchaus das Impulsdiagramm rein weil das kann man per Hand einfach in Latex eingeben

Ansonsten Versuch ich mich doch an der Programmiersprache, Abel Weil die kann man in Linux nur in der sozusagen virtuellen Maschine bochs oder so ausführen das ist mir allerdings schon gelungen und hat ein jedec file erzeugt, was eigentlich sehr einfach benutzbar ist, dann weiter unter Linux

Weiterhin probiere ich dann sozusagen trotzdem fuses am GAL selber zu setzen. Ohne die Sprache und des vhdl lass ich jetzt weg.

David Vajda
binäres Dividieren
Seite 1, Teil II
06/25/2026

$y_7\ y_6\ y_5\ y_4\ y_3\ y_2\ y_1\ y_0$ (Dividend)
u.:
$x_7\ x_6\ x_5\ x_4\ x_3\ x_2\ x_1\ x_0$ Minuend

Divisor...
u. Subtrahend...

ok: angenommen der zu betrachtende Teil des (an Stellen) des Divisors ist 4 bit lang:

$y_7\ y_6\ y_5\ y_4$
nach oben geshiftet:
ursprünglich:
$0b0000\ y_3\ y_2\ y_1\ y_0$
← z.b: $0b00001001$
$\hat{=} 9_{dec}$...

David Vajda
06/25/2026
binäres Dividieren
Seite 2, Teil II

Dann macht man:

x und y hätte gerade
lieber als y und x
ausgedrückt werden sollen:

y_7
$-\ x_7\ x_6\ x_5\ x_4\ 0\ 0\ 0\ 0$

dann ergibt sich:
quotient: 0...

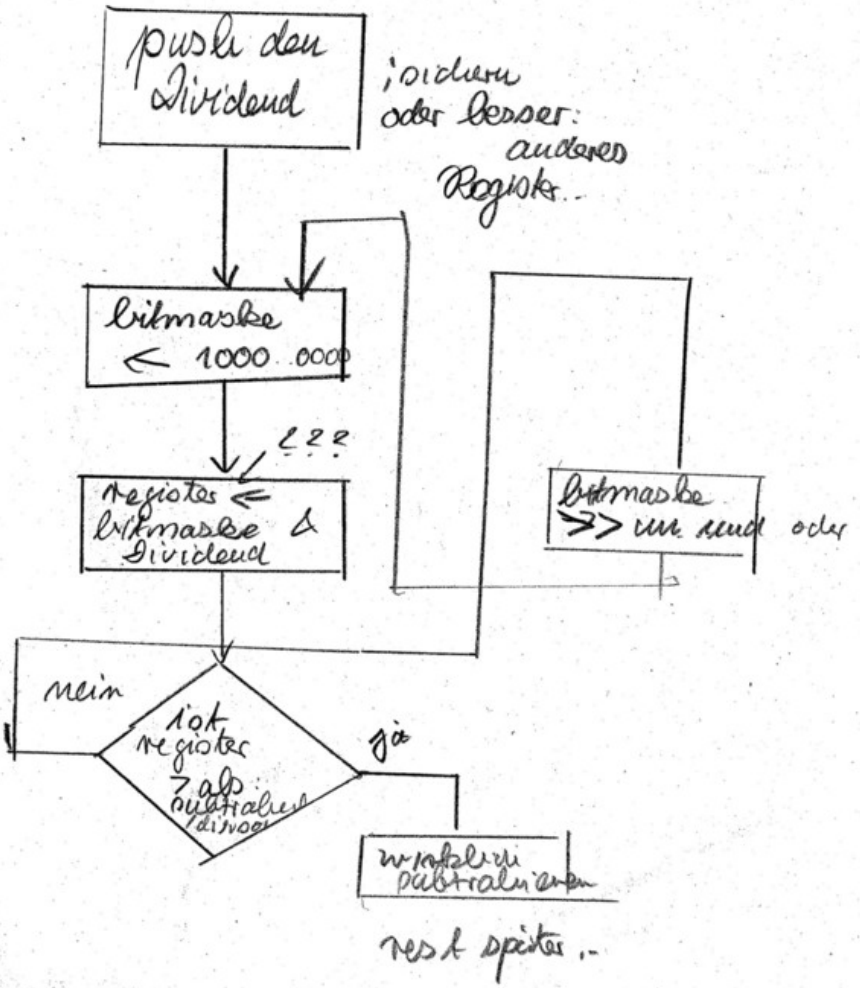
Dann weiter:

$y_7\ y_6\ \ldots$
$-\ x_7\ x_6\ x_5\ x_4\ \ldots$

und weiter:

$y_7\ y_6\ y_5\ \ldots$
$-\ x_7\ x_6\ x_5\ x_4\ \ldots$

David Vajda 06/25/2026
Binäres Dividieren, Seite 3, Teil II



Daniel Vajda
06/25/2026
binäres Suchen
Seite 4, Teil II

Ok:
noch folgendes

V Die andere Bitmaske
würde sich dann anbieten
(Dachte ich, was falsch ist)
um sie mit der vorherigen
zu vergleichen...

Das ist nicht nötig. Da:
Die Subtraktion + der vorherige
das Abbruchkriterium bietet.

V nächstes: Die Bitmaske muss
dann - ??? immer wieder auf
0b100... gesetzt werden?

Nein! 100... und dann wird
1 ergänzt o. 0 wenn aber:
aber: bis nicht bitweise wenn

06/25/2026
Zamid Vajda
Divisionsbeispiel, Seite 1, Teil III

So: wenn die Subtraktion:
ungeklärter Teil:
Errechnung des Quotienten...
Statt gefunden hat...

a) dann muß nothar
wenn sie nicht statt findet
eine 0 im Quotienten vermerkt
werden... und die Sache weiter
nach links geschoben werden

b) dazu kommt: Subtraktion selber.
Die Subtraktion hat stattgefunden?
Die Sache das Register von
dem subtrahiert wird, in dem
Fall angelegenheit setzt sich
aus drei teilen zusammen:
1.) den vorderen
2.) dem Mittleren bei dem
subtrahiert wird
3.) dem linkeren teil...

06/25/2026
Zamid Vajda
Divisionsbeleg, Seite 1, Teil III

So: wenn die Subtraktion:
ungeklärter Teil:
Errechnung des Quotienten...
Statt gefunden hat...

a) dann muß nochmal
wenn sie nicht statt findet
eine 0 im Quotienten vermerkt
werden... und die Sache weiter
nach links geschoben werden

b) dazu kommt: Subtraktion selber.
Die Subtraktion hat stattgefunden?
Die Sache das Register von
dem subtrahiert wird, in dem
Fall angelegenheit setzt sich
aus drei Teilen zusammen:
1.) Den vorderen
2.) Dem Mittleren bei dem
subtrahiert wird
3.) Dem linkeren Teil...

06/25/2026
Jamid Vajda, Divisionsbeispiel
Seite 2, Teil III

ok:

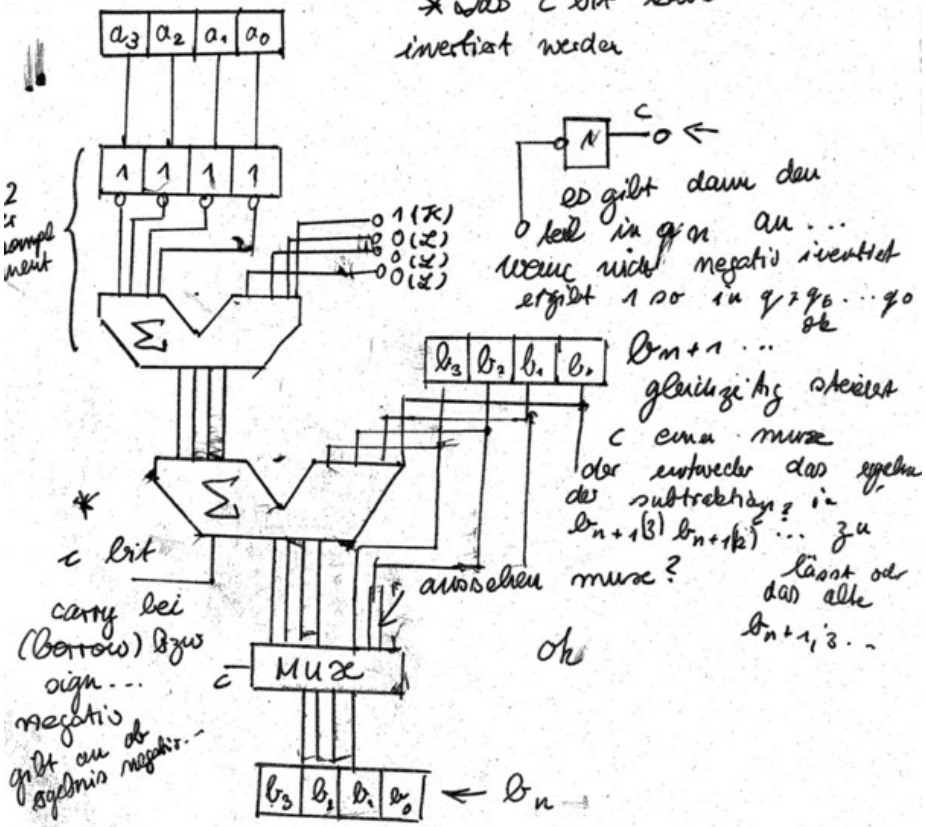
1.) Den vorderen können wir ruhig durch Ones ersetzen.
Das ist keine problem

2.) Den Mittleren:
Differenz = Minuend - Subtrahend
können wir genauso machen...

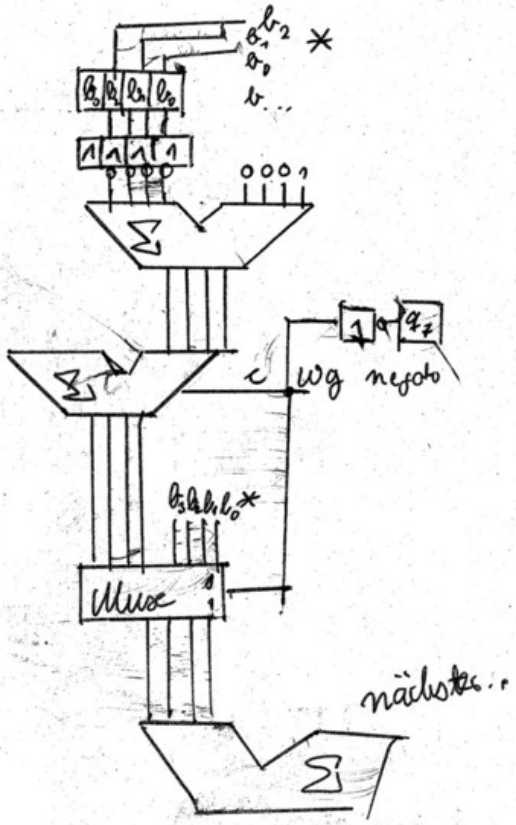
3.) Den linkeren teil gleich lassen

Die frage ist:
✓ wie ersetzen wir vorderen und mittleren teil
✓ Die Antwort könnte lauten:
wir erzeugen eine Maske mit führenden Ones und im niederwertigen teil 1er. Machen eine and verknüpfung...
Dadurch wird der eigentliche Dividend vom ~~linkeren und~~ vorderen und mittleren teil befreit
Da in der Differenz nur Ones führend... ok
und dann oder verknüpfung...

David Vajda
06126/2026
Dividierwerk, Seite 1



David Vogele 06/26/2026
Dividiernetzwerk Seite 22



7 teilschritt

								b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0							
		a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0						
			a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0					
				a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0				
					a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0			
						a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0		
							a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	
								a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0

8 teilschritt

								b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
x	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0							
	x	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0						
		x	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0					
			x	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0				
				x	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0			
					x	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0		
						x	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	
							x	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0

9 teilschritt

								b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0		
x	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	s	q_7							
	x	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	s	q_6						
		x	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	s	q_5					
			x	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	s	q_4				
				x	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	s	q_3			
					x	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	s	q_2		
						x	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	s	q_1	
							x	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	s	q_0

10 teilschritt

								b_{07}	b_{06}	b_{05}	b_{04}	b_{03}	b_{02}	b_{01}	b_{00}	
x	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	s	q_7							
								b_{17}	b_{16}	b_{15}	b_{14}	b_{13}	b_{12}	b_{11}	b_{10}		
	x	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	s	q_6						
								b_{27}	b_{26}	b_{25}	b_{24}	b_{23}	b_{22}	b_{21}	b_{20}		
		x	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	s	q_5					
								b_{37}	b_{36}	b_{35}	b_{34}	b_{33}	b_{32}	b_{31}	b_{30}		
			x	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	s	q_4				
								b_{47}	b_{46}	b_{45}	b_{44}	b_{43}	b_{42}	b_{41}	b_{40}		
				x	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	s	q_3			
								b_{57}	b_{56}	b_{55}	b_{54}	b_{53}	b_{52}	b_{51}	b_{50}		
					x	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	s	q_2		
								b_{67}	b_{66}	b_{65}	b_{64}	b_{63}	b_{62}	b_{61}	b_{60}		
						x	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	s	q_1	
								b_{77}	b_{76}	b_{75}	b_{74}	b_{73}	b_{72}	b_{71}	b_{70}		
							x	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	s	q_0

11 teilschritt

								b_{07}	b_{06}	b_{05}	b_{04}	b_{03}	b_{02}	b_{01}	b_{00}		
x_7	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	s_7	q_7							
								b_{17}	b_{16}	b_{15}	b_{14}	b_{13}	b_{12}	b_{11}	b_{10}		
	x_6	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	s_6	q_6						
								b_{27}	b_{26}	b_{25}	b_{24}	b_{23}	b_{22}	b_{21}	b_{20}		
		x_5	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	s_5	q_5					
								b_{37}	b_{36}	b_{35}	b_{34}	b_{33}	b_{32}	b_{31}	b_{30}		
			x_4	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	s_4	q_4				
								b_{47}	b_{46}	b_{45}	b_{44}	b_{43}	b_{42}	b_{41}	b_{40}		
				x_3	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	s_3	q_3			
								b_{57}	b_{56}	b_{55}	b_{54}	b_{53}	b_{52}	b_{51}	b_{50}		
					x_2	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	s_2	q_2		
								b_{67}	b_{66}	b_{65}	b_{64}	b_{63}	b_{62}	b_{61}	b_{60}		
						x_1	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	s_1	q_1	
								b_{77}	b_{76}	b_{75}	b_{74}	b_{73}	b_{72}	b_{71}	b_{70}		
							x_0	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	s_0	q_0

12 teilschritt

L	L	L	L	L	L	L	L	b_{07}	b_{06}	b_{05}	b_{04}	b_{03}	b_{02}	b_{01}	b_{00}	
x_7	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	s_7	q_7						
L	L	L	L	L	L	L	L	b_{17}	b_{16}	b_{15}	b_{14}	b_{13}	b_{12}	b_{11}	b_{10}	
	x_6	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	s_6	q_6					
L	L	L	L	L	L	L	L	b_{27}	b_{26}	b_{25}	b_{24}	b_{23}	b_{22}	b_{21}	b_{20}	
		x_5	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	s_5	q_5				
L	L	L	L	L	L	L	L	b_{37}	b_{36}	b_{35}	b_{34}	b_{33}	b_{32}	b_{31}	b_{30}	
			x_4	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	s_4	q_4			
L	L	L	L	L	L	L	L	b_{47}	b_{46}	b_{45}	b_{44}	b_{43}	b_{42}	b_{41}	b_{40}	
				x_3	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	s_3	q_3		
L	L	L	L	L	L	L	L	b_{57}	b_{56}	b_{55}	b_{54}	b_{53}	b_{52}	b_{51}	b_{50}	
					x_2	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	s_2	q_2	
L	L	L	L	L	L	L	L	b_{67}	b_{66}	b_{65}	b_{64}	b_{63}	b_{62}	b_{61}	b_{60}	
						x_1	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	s_1	q_1
L	L	L	L	L	L	L	L	b_{77}	b_{76}	b_{75}	b_{74}	b_{73}	b_{72}	b_{71}	b_{70}	
							x_0	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	s_0	q_0

ich habe noch eine idee:

sanduhr.

also, kerze ist nicht falsch und nicht, nicht so falsch,aber sanduhr

ok: andere probleme.

das eine ist:

erstens, ich muss die sonne in tuebingen im internet angucken und den nachthimmel. ich bin kein experte und sollte es lernen

ich fange grob an, der nachthimmel im jahr geteilt durch 12

die uhr muss her und die sonne

sanduhr sei zu erwaehnen, ziel ist es aber ohne uhr zurecht zu finden, ... man muesste eigentlich den tag etwa schliessen koennen.ok, tageslaenge what ever. aber hauptsache wo steht die sonne

ok: wenn das passiert ist. ist noch die frage, abgesehen jetzt schlaegt es 13, vielleicht nur in schwaben

ist die frage: mit der division. es ist einfach mit diesen mitteln zu dividieren, nur sollte eine zahl laenger sein als 8 bit

jetzt kommt die mystische zahl 64

naemlich die anzahl der permutationen

das ist das entscheidende. 64 ist ein quadword - bzw. ein 64 Bit also es gibt

```
byte 8
halfword 16
word 32
double word 64
quad word 128??? oder 256??

aber manchmal
\begin{verbatim}
byte 8
word 16
double word 32
quad word 64
```

ok:

bei einem byte lautet die mystische zahl immer 64. selbst wenn wir 128 bit teiler (divisor

wenn ich ein byte habe, dann entstehen mit verschiebung von insgesamt 8 byte in einem maxi

also permutationen wo von oben nachgeschoeben werden kann.

weil: wenn ich 8 wiederum voll habe und ich subtrahiere die 8

kann ich einfach die 8 als byte wieder versetzt anfangen

ob register oder RAM

aber:
\begin{verbatim}
r0\dots r7
bzw.
\begin{verbatim}
r7 .. r0
ok, ich muss schieben, dann jeweils ein bit weiter.

aber, wenn ich damit fertig bin und muesste das ganze weiter hinten ansetzen
\begin{verbatim}
r6 \dots r-1

gedacht

oder weiter vorne

\begin{verbatim}
r8 .. r1

kann ich genauso gut,
\begin{verbatim}
r7 .. r0
nehmen. der unterschied. es gibt in 8 bit gedacht einen strahl, oben lampe, mit fukus, die

grosse und kleine schritte. wenn die 8 bit quasi rum sind, permutation es kommt auf das gl

man muss als erstes diese operation vollziehen um dann nachher sinnvoll mit dem dividieren

bevor ich gehe noch mal:

1.) multipliziernetzwerke sind deswegen generell einfacher, weil sie nicht auf das ergebni
2.) dividiernetzwerke sind deswegen einfach:

man vollzieht eine division indem man es intern ins zweier komplement umwandelt - (inverti

man addiert das zur entsprechenden stellen vom minuenden (dividenden)

ergibt sich eine negative zahl im zweierkomplement wird das negative als flag auswertbar,

wie das carry flag, bei addition nur als bit 7, bei 8 bit und nicht 8

dadurch wird eine 1 angezeigt, wenn subtraktion moeglich, weil: minuend groesser.

es ist quasi ein spezieller subtrahierer.

von mmx kennen wir swap arround \dots der wert wird nicht ueber und unterschritten, eine s

das geschieht dann allerdings dadurch dass die subtraktion stattfindet, intern, diese dien

dahinter ist ein MUX, einerseits bit im quotienten (ergebnis)

der mux wiederum nimmt!

vorsicht nicht das Ergebnis mit stellen (entsprechend) ganz sondern entweder von subtrakti

das heisst: das ergebnis (sprichwoertlich, sub und mux davor), also das letzte und dieses

dazu nachher zeichnung

uebernehmen im naechsten (immer das ganze, auch vorne) entweder von subtraktion - der hint

von mux, dem davor, vom subtraktionsschritt\dots

\begin{verbatim}
;; so irgendwie - muss noch dran arbeiten, denn ich brauche minimum doppelregister, beim

ich muss das abenteuer so machen, dass es im SRAM statt findet.

hier mehr oder weniger ein beispiel:

```
;;; test ldi r16, DIVIDEND
div:
push r16
push r17
push r18
push r19
ldi r16, DIVIDEND
ldi r17, DIVISOR

lsl r17
lsl r17
lsl r17
lsl r17
```

```
ldi r19, 0b10000000
div001:
cp r16, r17
brlt div002
sub r16, r17
or r18, r19
div002:
lsr r17
lsr r19
brne div001
mov r20, r18
\dots

ret

byte2bcdstr:
```

so, jetzt kommt ein zaehler fuer das lcd - ist das geschehen, die uhr...

heute noch

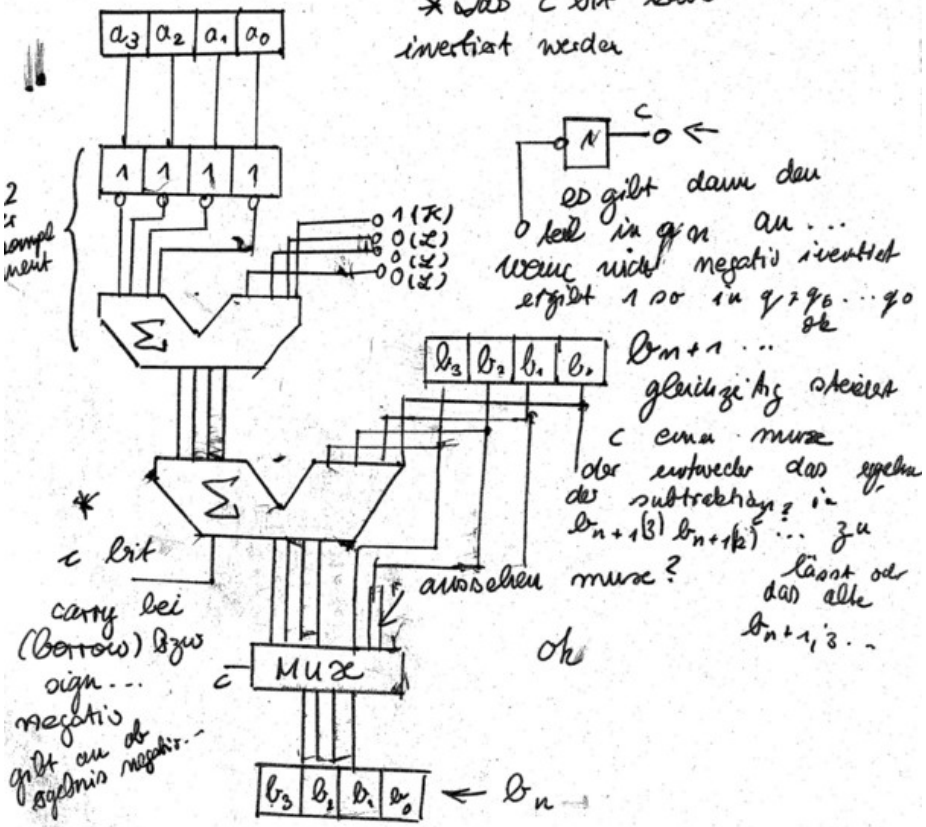
zur zeit ist zur sagen: die einzige alternativ darstellung zur zeit in sekunden waere das abbrennen der kerze, einer bestimmten laenge, doch das waere in sekunden (metrischen zeitmass) gerechnet.

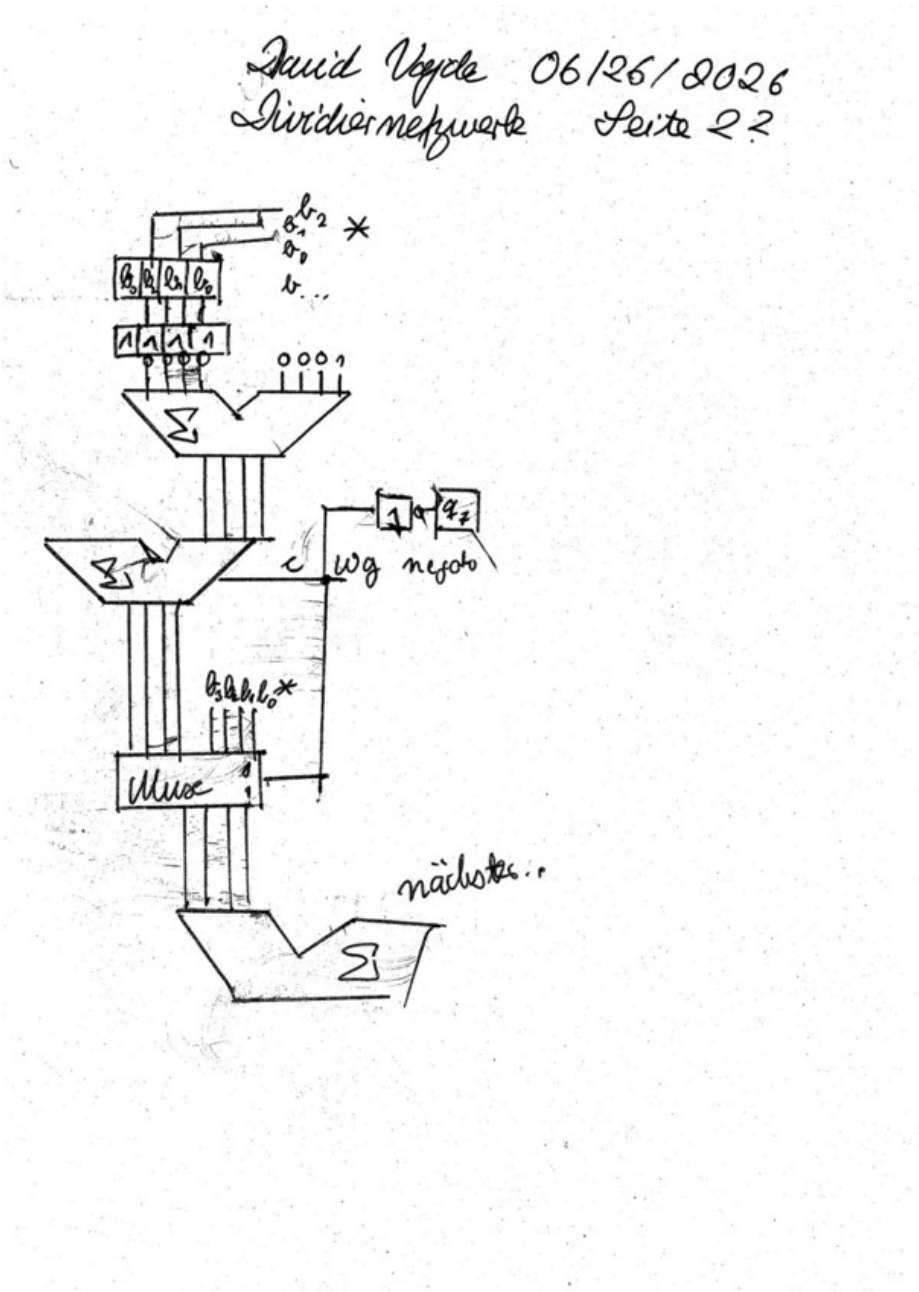
es waere vielleicht hilfreich um definitionen fuer das candela, ... zu haben

in folge dessen, wenn die uhr heute noch fertig ist, geht es um den stand der sonne und so weiter...

ok, aufgrund dessen laesst es sich auch ohne arrays und ohne viele register, im atmega8 - durch die weitergabe - an dasselbe register im naechsten schritt, erreichen... gleich der code... dann tut...

David Vajda
06126/2026
Dividierwerk, Seite 1





eine sache die in dem schaltnetz bisher nicht zu erkennen ist, ist die laenge der registers b das b register, dividend muss naemlich. an den fuehrenden stellen mit 0 aufgefuellt werden und ist entsprechend doppelt so lang (breit) wie b. dadurch kann man sich auch das zaehlen des hoechsten 1er bit in a ersparen, also beim divisor. der dividend wird in jedem falle groesser sein, wenn b oben mit 0en gefuellt ist. die division nicht statt finden.

ich habe heute nacht noch mal nachgedacht und hatte meinen laptop dabei. statt die sache mit befehlen und schleifen zu vollziehen, gehe ich erst den weg,

eines physischen dividiernetzwerkes.

auch, wenn es register enthaelt, ist es ein netzwerk

die sache funktioniert nach dem prinzip

die subtraktion wird an jeder stelle vollzogen. bit fuer bit

da es davon abhaengt was die subtraktion an der naechsthoeheren stelle n+1 ergibt, muss die subtraktion an der naechste stelle unterbrochen oder aktiviert werden

es gibt somit ein c flag - eine art chip enable, hier x. was als eingang dient - und dafuer den entsprechenden ausgang.

dazu kommt:

1.) das wird teil im naechsten schritt

das subtrahieren findet statt - abhaengig von dem hoeheren subtrahierer, indem ein komperator bekannt gibt, an jedem element, dass die subtraktion statt zu finden hat, der dividend (minuend) groesser ist.

sollte das der fall sein, wird subtrahiert. dies findet durch einen subtrahierer statt, der natuerlicherweise mit zweierkomplement und addition implementiert wird, aber quasi nur intern - das ergebnis wird positiv sein, bestaetigt der komperator

hier laesst sich eine zusammenfassung schaffen, indem das ergebnis negativ ist, damit kann der komperator entfallen, wird das flag genutzt, um die subtraktion ungeschehen zu machen

da das ergebnis verwendet werden muss an der subtraktion der naechst kleineren stelle.

dabei aber der veraenderte dividend eine rolle spielt, muessen pro stufe/zeile wie auch immer die ergibnisse an den hoeheren stelle n+1, n+2

bzw. induktiv: `n+1. n = n+1`, dafuer gilt `n+1`

uebernommen werden. ok.

dafuer muss dieses netzwerk geschaffen werden, im naechsten schritt

weil sich damit das netzwerk hardware technisch vollzogen hat, laesst es sich trotzdem mit befehlen anwenden,

man nimmt: `r0 .. r31`

und: nimmt acht register: `r16, r17, ..., r24` vollzieht damit dasselbe...

jetzt weiter...

das ziel ist ja jetzt, die sache zu debuggen, das geht ueber einzelne abfragen der variablen...

zum beispiel mit `printf(`

```
getchar();
```

und so weiter. danach ein `getchar ();`

dafuer brauche ich zunaechst kein abbruchkriterium, ich muss die einzelnen werte inspizieren, das geht jetzt auch schon so...

```
// david vajda
// div command bitwise
// 06/25/2026

#include <stdio.h>

#define DIVIDEND    56
```

```
#define DIVISOR     7

// r16: dividend
// r17: divisor
// r18: dividend tmp - nach verknuepfung mit bitmaske
// r19: bitmaske.
// r20: der anfang der bitmaske: die bitmaske
//      koennte immer bei 0b10000000 sein, ...
//      0b10000000, 0b11000000, 0b11100000
//      da sie aber wandert .. naemlich an die hinteren teile
//      0b00010000, 0b00011000, 0b00011100 z.B.
//      gibt es ein bitmasken startflag: angefangen bei
//      0b10000000 geht es in der auesseren schleife nach:
//      0b01000000, 0b00100000, 0b00010000 ... usw.
// r21: dies ist die maske, beginnend bei 0b1111.1111
//      die mitgenommen wird und bei jedem einzelschritt, eine
//      fuehrende 1 verlieren sollte, 0b0111.1111, 0b0011.111,
//      0b0001.1111 usw. und bei der eigentlichen subtraktion
//      dazu da ist, beim verbleibenden dividenden, den vorderen
//      teil zu entfernen.
// r22: quotient

unsigned char r16;
unsigned char r17;
unsigned char r18;
unsigned char r19;
unsigned char r20;
unsigned char r21;
unsigned char r22;

int main (void) {

    r16 = DIVIDEND;
    r17 = DIVISOR;

    while (1) {
        //  shiftright001:
        //  sbrc r16, 7
        if ((r16 & 0x80) == 0x80)
        // rjmp shiftright001end
            break;
        r16 = r16 << 1;
        // lsl r16
        // rjmp shiftright001
        // shiftright001end:
    }

    printf ("%x\n", r16);

    r22 = 0x00;
```

```
    r21 = 0xff;
    r20 = 0x80;

    //  ldi r22, 0b00000000
    //  ldi r21, 0b11111111
    //  ldi r20, 0b10000000
    //  div000:
    //  mov r19, r20

    while (1) {
        r19 = r20;
        //  div001:
        //  mov r18, r16        //  den dividenden aus dem bleiben r16 ins tmp r18
        //  and r18, r19        //  den divenden im tmp mit der bitmaske verknuepfen
        while (1) {
            r18 = r16;
            r18 = r18 & r19;
            // r18 < r17
            // cp r18, r17
            if (r18 >= r17) {
                r18 = r18 - r17;
                r16 = r16 & r21;
                r16 = r16 | r18;
                r22 = r22 << 1;
                r22 = r22 | 0x01;
            } else {
                r22 = r22 << 1;
                r21 = r21 >> 1;
            }
            // cp r18, r17
            // brlt subtraction001end
            // subtraction001:
            // sub r18, r17
            // and r16, r21
            // or r16, r18
            // lsr r22         // lsl r22!!
            // ori r22, 0b00000001
            // rjmp subtraction002end
            // subtraction001end:
            // lsr r21
            // lsr r22             // quotient wird null erzeugt und muss weiter geschoebe
            // subtraction002end:
        }
        r19 = r19 >> 1;
        r19 = r19 | r20;
        r17 = r17 >> 1;
        r20 = r20 >> 1;
    }
return 0;
}
```

abbruchkriterium war keines da, sehe ich... ok: und jetzt:
ich nehme trotzdem das C programm...

```
// geht gleich weiter, raucherpause...

// david vajda
// div command bitwise
// 06/25/2026

#include <stdio.h>

#define DIVIDEND    56
#define DIVISOR     7

// r16: dividend
// r17: divisor
// r18: dividend tmp - nach verknuepfung mit bitmaske
// r19: bitmaske.
// r20: der anfang der bitmaske: die bitmaske
//      koennte immer bei 0b10000000 sein, ...
//      0b10000000, 0b11000000, 0b11100000
//      da sie aber wandert .. naemlich an die hinteren teile
//      0b00010000, 0b00011000, 0b00011100 z.B.
//      gibt es ein bitmasken startflag: angefangen bei
//      0b10000000 geht es in der auesseren schleife nach:
//      0b01000000, 0b00100000, 0b00010000 ... usw.
// r21: dies ist die maske, beginnend bei 0b1111.1111
//      die mitgenommen wird und bei jedem einzelschritt, eine
//      fuehrende 1 verlieren sollte, 0b0111.1111, 0b0011.111,
//      0b0001.1111 usw. und bei der eigentlichen subtraktion
//      dazu da ist, beim verbleibenden dividenden, den vorderen
//      teil zu entfernen.
// r22: quotient

unsigned char r16;
unsigned char r17;
unsigned char r18;
unsigned char r19;
unsigned char r20;
unsigned char r21;
unsigned char r22;

int main (void) {

    r16 = DIVIDEND;
    r17 = DIVISOR;

    while (1) {
        //  shiftright001:
        //  sbrc r16, 7
```

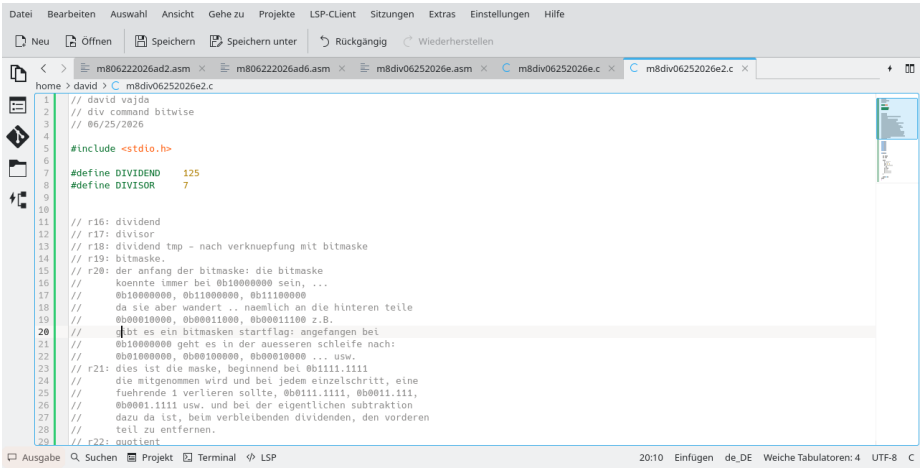
```
        if ((r16 & 0x80) == 0x80)
        // rjmp shiftright001end
            break;
        r16 = r16 << 1;
        // lsl r16
        // rjmp shiftright001
        // shiftright001end:
    }

    printf ("%x\n", r16);

    r22 = 0x00;
    r21 = 0xff;
    r20 = 0x80;

    //  ldi r22, 0b00000000
    //  ldi r21, 0b11111111
    //  ldi r20, 0b10000000
    //  div000:
    //  mov r19, r20

    while (1) {
        r19 = r20;
        //  div001:
        //  mov r18, r16        //  den dividenden aus dem bleiben r16 ins tmp r18
        //  and r18, r19        //  den divenden im tmp mit der bitmaske verknuepfen
        while (1) {
            r18 = r16;
            r18 = r18 & r19;
            // r18 < r17
            // cp r18, r17
            if (r18 >= r17) {
                r18 = r18 - r17;

            }
        }
    }
return 0;
}

ldi r22, 0b00000000
ldi r21, 0b11111111
ldi r20, 0b10000000
div000:
mov r19, r20
div001:
mov r18, r16        //  den dividenden aus dem bleiben r16 ins tmp r18
and r18, r19        //  den divenden im tmp mit der bitmaske verknuepfen

// hier die eigentliche subtraktion
```

```
cp r18, r17
brlt subtraction001end
subtraction001:
sub r18, r17
and r16, r21
or r16, r18
lsr r22
ori r22, 0b00000001
rjmp subtraction002end
subtraction001end:
lsr r21
lsr r22             // quotient wird null erzeugt und muss weiter geschoeben werden
subtraction002end:

// hier die eigentliche subtraktion beendet...

lsr r19             // die maske eines rechts um sie zu vergroessern
or r19, r20         // im naechsten schritt naemlich
rjmp div001
lsr r17             // den divisor weiter nach rechts schieben
lsr r20
rjmp div000

out PORTD, r22

end: rjmp end
```


Datei Bearbeiten Konstanten Einstellungen Hilfe
Bin Deg
11 1000
Hex Dez Okt Bin
38 56 70 111000
AND mod A % ÷ × − Shift
OR 1/x B 7 8 9 + C ←
XOR x! C AC MS
Lsh x² D 4 5 6 (MC
Rsh xʸ E 1 2 3) MR
Cmp x³ F = +/− M+
x·10ʸ 0 ,
NORM BIN
Datei Bearbeiten Konstanten Einstellungen Hilfe
Dec Deg
125
Hex Dez Okt Bin
7D 125 175 1111101
AND mod A % ÷ × − Shift
OR 1/x B 7 8 9 + C ←
XOR x! C AC MS
Lsh x² D 4 5 6 (MC
Rsh xʸ E 1 2 3) MR
Cmp x³ F = +/− M+
x·10ʸ 0 ,
NORM DEC


Datei Bearbeiten Konstanten Einstellungen Hilfe
Hex Deg
7D
Hex
Dez
Okt
Bin
7D
125
175
1111101
AND mod A
OR 1/x B
XOR x! C
Lsh x² D
Rsh xʸ E
Cmp x³ F
x·10ʸ
% ÷ × −
7 8 9 +
4 5 6
1 2 3 =
0 ,
Shift
C ←
AC MS
(MC
) MR
+/− M+
NORM HEX
Datei Bearbeiten Konstanten Einstellungen Hilfe
Bin Deg
111 1101
Hex
Dez
Okt
Bin
7D
125
175
1111101
AND mod A
OR 1/x B
XOR x! C
Lsh x² D
Rsh xʸ E
Cmp x³ F
x·10ʸ
% ÷ × −
7 8 9 +
4 5 6
1 2 3 =
0 ,
Shift
C ←
AC MS
(MC
) MR
+/− M+
NORM BIN


Datei Bearbeiten Konstanten Einstellungen Hilfe
Hex Deg
FA
Hex
Dez
Okt
Bin
FA
250
372
11111010
AND mod A
OR 1/x B
XOR x! C
Lsh x² D
Rsh xʸ E
Cmp x³ F
x·10ʸ
% ÷ × −
7 8 9 +
4 5 6
1 2 3 =
0 ,
Shift
C ←
AC MS
(MC
) MR
+/− M+
NORM HEX
Datei Bearbeiten Konstanten Einstellungen Hilfe
Bin Deg
1111 1010
Hex
Dez
Okt
Bin
FA
250
372
11111010
AND mod A
OR 1/x B
XOR x! C
Lsh x² D
Rsh xʸ E
Cmp x³ F
x·10ʸ
% ÷ × −
7 8 9 +
4 5 6
1 2 3 =
0 ,
Shift
C ←
AC MS
(MC
) MR
+/− M+
NORM BIN




```
// ...

// david vajda
// div command bitwise
// 06/25/2026

#include <stdio.h>

#define DIVIDEND    56
#define DIVISOR     7

// r16: dividend
// r17: divisor
// r18: dividend tmp - nach verknuepfung mit bitmaske
// r19: bitmaske.
// r20: der anfang der bitmaske: die bitmaske
//      koennte immer bei 0b10000000 sein, ...
//      0b10000000, 0b11000000, 0b11100000
//      da sie aber wandert .. naemlich an die hinteren teile
//      0b00010000, 0b00011000, 0b00011100 z.B.
//      gibt es ein bitmasken startflag: angefangen bei
//      0b10000000 geht es in der auesseren schleife nach:
//      0b01000000, 0b00100000, 0b00010000 ... usw.
// r21: dies ist die maske, beginnend bei 0b1111.1111
//      die mitgenommen wird und bei jedem einzelschritt, eine
//      fuehrende 1 verlieren sollte, 0b0111.1111, 0b0011.111,
//      0b0001.1111 usw. und bei der eigentlichen subtraktion
//      dazu da ist, beim verbleibenden dividenden, den vorderen
//      teil zu entfernen.
// r22: quotient

unsigned char r16;
unsigned char r17;
```

```
unsigned char r18;
unsigned char r19;
unsigned char r20;
unsigned char r21;
unsigned char r22;

int main (void) {

    r16 = DIVIDEND;
    r17 = DIVISOR;

    while (1) {
        //  shiftright001:
        //  sbrc r16, 7
        if ((r16 & 0x80) == 0x80)
        // rjmp shiftright001end
            break;
        r16 = r16 << 1;
        // lsl r16
        // rjmp shiftright001
        // shiftright001end:
    }

    printf ("%i\n", r16);
return 0;
}

ldi r22, 0b00000000
ldi r21, 0b11111111
ldi r20, 0b10000000
div000:
mov r19, r20
div001:
mov r18, r16        //  den dividenden aus dem bleiben r16 ins tmp r18
and r18, r19        //  den divenden im tmp mit der bitmaske verknuepfen

// hier die eigentliche subtraktion

cp r18, r17
brlt subtraction001end
subtraction001:
sub r18, r17
and r16, r21
or r16, r18
lsr r22
ori r22, 0b00000001
rjmp subtraction002end
subtraction001end:
lsr r21
lsr r22             // quotient wird null erzeugt und muss weiter geschoeben werden
```

```
subtraction002end:

// hier die eigentliche subtraktion beendet...

lsr r19             // die maske eines rechts um sie zu vergroessern
or r19, r20         // im naechsten schritt naemlich
rjmp div001
lsr r17             // den divisor weiter nach rechts schieben
lsr r20
rjmp div000

out PORTD, r22

end: rjmp end
```

gut, auch mit initialisierungswerten kommt bisher nichts raus - dann nehme ich das nach C und werde aber, auch, wenn ich C anweisungen nehme, fuer r31..r0 durch unsigned char variablen ersetzen...

```
;; david vajda
;; div command bitwise
;; 06/25/2026

.include "m8def.inc"

.equ DIVIDEND   =   56
.equ DIVISOR    =   7

ldi r16, HIGH (RAMEND)
out SPH, r16
ldi r16, LOW (RAMEND)
out SPL, r16

ldi r16, 0xff
out DDRD, r16

;; r16: dividend
;; r17: divisor
;; r18: dividend tmp - nach verknuepfung mit bitmaske
;; r19: bitmaske.
;; r20: der anfang der bitmaske: die bitmaske
;;      koennte immer bei 0b10000000 sein, ...
;;      0b10000000, 0b11000000, 0b11100000
;;      da sie aber wandert .. naemlich an die hinteren teile
;;      0b00010000, 0b00011000, 0b00011100 z.B.
;;      gibt es ein bitmasken startflag: angefangen bei
;;      0b10000000 geht es in der auesseren schleife nach:
;;      0b01000000, 0b00100000, 0b00010000 ... usw.
;; r21: dies ist die maske, beginnend bei 0b1111.1111
```

```
;;      die mitgenommen wird und bei jedem einzelschritt, eine
;;      fuehrende 1 verlieren sollte, 0b0111.1111, 0b0011.111,
;;      0b0001.1111 usw. und bei der eigentlichen subtraktion
;;      dazu da ist, beim verbleibenden dividenden, den vorderen
;;      teil zu entfernen.
;; r22: quotient

shiftright001:
sbrc r16, 7
rjmp shiftright001end
lsl r16
rjmp shiftright001
shiftright001end:

;; ...

ldi r22, 0b00000000
ldi r21, 0b11111111
ldi r20, 0b10000000
div000:
mov r19, r20
div001:
mov r18, r16        ;;  den dividenden aus dem bleiben r16 ins tmp r18
and r18, r19        ;;  den divenden im tmp mit der bitmaske verknuepfen

;; hier die eigentliche subtraktion

cp r18, r17
brlt subtraction001end
subtraction001:
sub r18, r17
and r16, r21
or r16, r18
lsr r22
ori r22, 0b00000001
rjmp subtraction002end
subtraction001end:
lsr r21
lsr r22             ;; quotient wird null erzeugt und muss weiter geschoeben werden
subtraction002end:

;; hier die eigentliche subtraktion beendet...

lsr r19             ;; die maske eines rechts um sie zu vergroessern
or r19, r20         ;; im naechsten schritt naemlich
rjmp div001
lsr r17             ;; den divisor weiter nach rechts schieben
lsr r20
rjmp div000
```

```
out PORTD, r22

end: rjmp end

;; david vajda
;; div command bitwise
;; 06/25/2026

.include "m8def.inc"

.equ DIVIDEND   =   56
.equ DIVISOR    =   7

ldi r16, HIGH (RAMEND)
out SPH, r16
ldi r16, LOW (RAMEND)
out SPL, r16

ldi r16, 0xff
out DDRD, r16

;; r16: dividend
;; r17: divisor
;; r18: dividend tmp - nach verknuepfung mit bitmaske
;; r19: bitmaske.
;; r20: der anfang der bitmaske: die bitmaske
;;      koennte immer bei 0b10000000 sein, ...
;;      0b10000000, 0b11000000, 0b11100000
;;      da sie aber wandert .. naemlich an die hinteren teile
;;      0b00010000, 0b00011000, 0b00011100 z.B.
;;      gibt es ein bitmasken startflag: angefangen bei
;;      0b10000000 geht es in der auesseren schleife nach:
;;      0b01000000, 0b00100000, 0b00010000 ... usw.
;; r21: dies ist die maske, beginnend bei 0b1111.1111
;;      die mitgenommen wird und bei jedem einzelschritt, eine
;;      fuehrende 1 verlieren sollte, 0b0111.1111, 0b0011.111,
;;      0b0001.1111 usw. und bei der eigentlichen subtraktion
;;      dazu da ist, beim verbleibenden dividenden, den vorderen
;;      teil zu entfernen.

shiftright001:
sbrc r16, 7
rjmp shiftright001en0d
lsl r16
rjmp shiftright001
shiftright001end:

;; ...

ldi r21, 0b11111111
```

```
ldi r20, 0b10000000
div000:
mov r19, r20
div001:
mov r18, r16        ;;  den dividenden aus dem bleiben r16 ins tmp r18
and r18, r19        ;;  den divenden im tmp mit der bitmaske verknuepfen

;; hier die eigentliche subtraktion

cp r18, r17
brlt subtraction001end
subtraction001:
sub r18, r17
and r16, r21
or r16, r18
rjmp subtraction002end
subtraction001end:
lsr r21
subtraction002end:

;; hier die eigentliche subtraktion beendet...

lsr r19             ;; die maske eines rechts um sie zu vergroessern
or r19, r20         ;; im naechsten schritt naemlich
rjmp div001
lsr r17             ;; den divisor weiter nach rechts schieben
lsr r20
rjmp div000

end: rjmp end
```

David Vajda
binäres Dividieren
Seite 1, Teil II
06/25/2026

$y_7\ y_6\ y_5\ y_4\ y_3\ y_2\ y_1\ y_0$ (Dividend)
u.:
Minuend
$x_7\ x_6\ x_5\ x_4\ x_3\ x_2\ x_1\ x_0$

Divisor...
u. Subtrahend...

ok: angenommen der zu betrachtende Teil des (an Stellen) des Divisors ist 4 bit lang:

$y_7\ y_6\ y_5\ y_4$

nach oben geshiftet:

ursprünglich:

0b0000$y_3\ y_2\ y_1\ y_0$

← z.b: 0b00001001
≙ 9dec...

David Vajda
06/25/2026
binäres Dividieren
Seite 2, Teil II

Dann macht man:

x und y hätte gerade
lieber als y und x
ausgedrückt werden sollen:

y_7
$-\ x_7\ x_6\ x_5\ x_4\ 0\ 0\ 0\ 0$

dann ergibt sich:
quotient: 0...

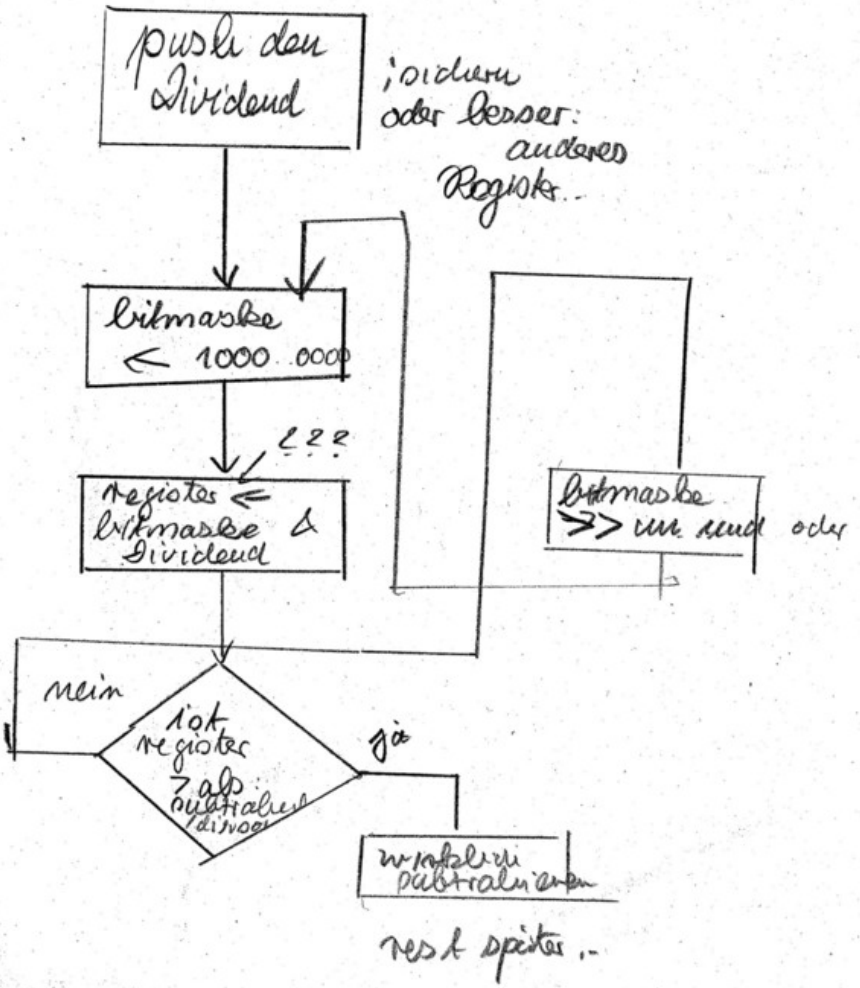
Dann weiter:

$y_7\ y_6\ \ldots$
$-\ x_7\ x_6\ x_5\ x_4\ \ldots$

und weiter:

$y_7\ y_6\ y_5\ \ldots$
$-\ x_7\ x_6\ x_5\ x_4\ \ldots$

David Vajda 06/25/2026
Binäres Dividieren, Seite 3, Teil II



Daniel Vajda
06/25/2026
binäres Suchen
Seite 4, Teil II

Ok:
noch folgendes

V die andere Bitmaske
würde sich dann anbieten
(Dachte ich, was falsch ist)
um sie mit der vorherigen
zu vergleichen...

Das ist nicht nötig. Da:
die Subtraktion + der vorherige
das Abbruchkriterium bietet.

V nächstes: die Bitmaske muss
dann - ??? immer wieder auf
0b100... gesetzt werden?

Nein! 100... und dann wird
1 ergänzt o. 0 wenn aber:
aber: bis nicht bitweise wenn

06/25/2026
Zamid Vajda
Divisionsbeispiel, Seite 1, Teil III

So: wenn die Subtraktion:
ungeklärter Teil:
Errechnung des Quotienten...
Statt gefunden hat...

a) dann muß nochmal
wenn sie nicht statt findet
eine 0 im Quotienten vermerkt
werden... und die Sache weiter
nach links geschoben werden

b) dazu kommt: Subtraktion selber.
Die Subtraktion hat stattgefunden?
Die Sache das Register von
dem subtrahiert wird, in dem
Fall angelegenheit setzt sich
aus drei Teilen zusammen:
1.) Den vorderen
2.) Dem Mittleren bei dem
subtrahiert wird
3.) Dem linkeren Teil...

06/25/2026
Zamid Vajda
Divisionsbeispiel, Seite 1, Teil III

So: wenn die Subtraktion:
ungeklärter Teil:
Errechnung des Quotienten...
Statt gefunden hat...

a) dann muß nothar
wenn sie nicht statt findet
eine 0 im Quotienten vermerkt
werden... und die Sache weiter
nach links geschoben werden

b) dazu kommt: Subtraktion selber.
Die Subtraktion hat stattgefunden?
Die Sache das Register von
dem subtrahiert wird, in dem
Fall angelegenheit setzt sich
aus drei teilen zusammen:
1.) den vorderen
2.) dem Mittleren bei dem
subtrahiert wird
3.) dem linkeren teil...

06/25/2026
Jamid Vajda, Divisionsbeispiel
Seite 2, Teil III

ok:

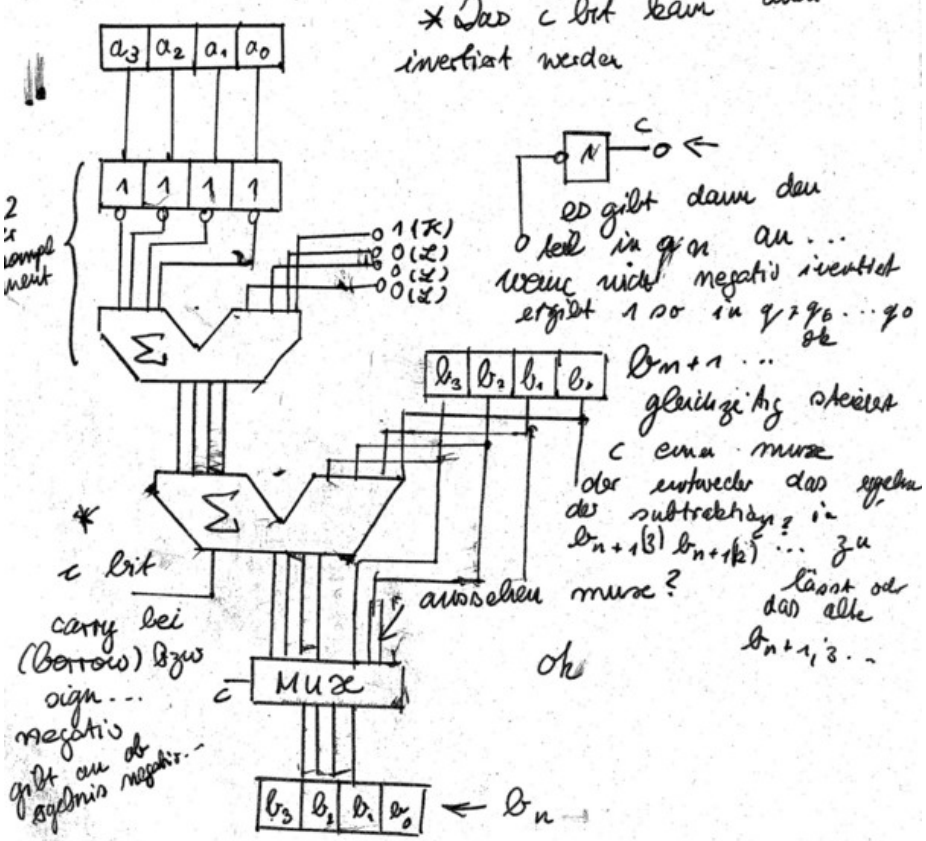
1.) Den vorderen können wir ruhig durch Ones ersetzen.
Das ist kein problem

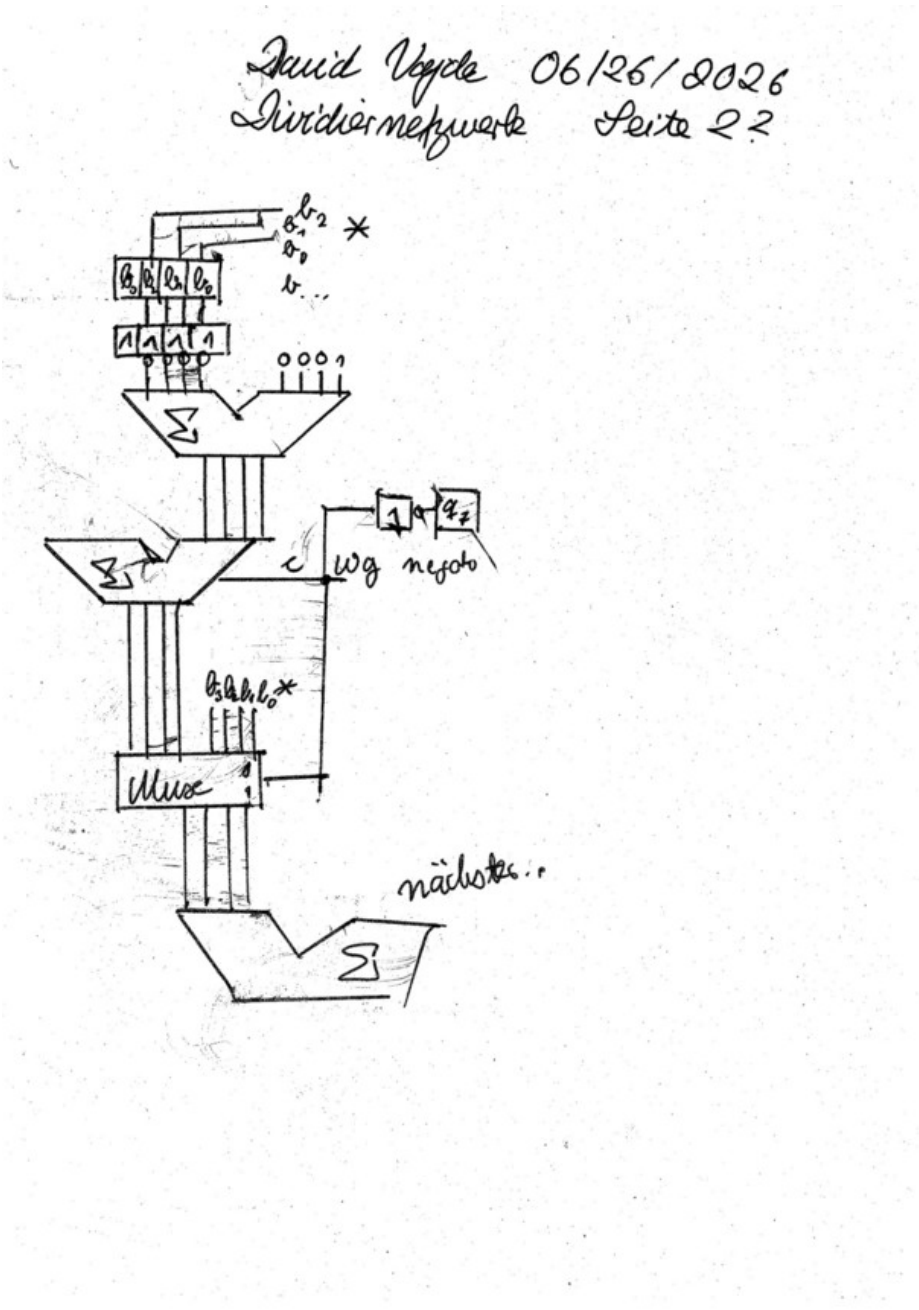
2.) Den Mittleren:
Differenz = Minuend - Subtrahend
können wir genauso machen...

3.) Den linkeren teil gleich lassen

Die Frage ist:
✓ wie ersetzen wir vorderen und mittleren teil
✓ Die Antwort könnte lauten:
wir erzeugen eine Maske mit führenden Ones und im niederwertigen teil 1er.
Machen eine und verknüpfung...
Dadurch wird der eigentliche Dividend vom ~~linkeren und~~ vorderen und mittleren teil befreit
Da in der Differenz nur Ones führend... ok
und dann oder verknüpfung...

David Vajda
06126/2026
Dividierwerk, Seite 1






```
;; gut an der stelle kommen wir nicht weiter, bei der subtraktion muessen wir noch nachden

;; inzwischen muss ich staubsaugen und probiere es vielleicht noch mal, mit dem ADC

;; david vajda
;; div command bitwise
;; 06/25/2026
```

```
.include "m8def.inc"

.equ DIVIDEND   =   56
.equ DIVISOR    =   7

ldi r16, HIGH (RAMEND)
out SPH, r16
ldi r16, LOW (RAMEND)
out SPL, r16

ldi r16, 0xff
out DDRD, r16

;; r16: dividend
;; r17: divisor
;; r18: dividend tmp - nach verknuepfung mit bitmaske
;; r19: bitmaske.
;; r20: der anfang der bitmaske: die bitmaske
;;      koennte immer bei 0b10000000 sein, ...
;;      0b10000000, 0b11000000, 0b11100000
;;      da sie aber wandert .. naemlich an die hinteren teile
;;      0b00010000, 0b00011000, 0b00011100 z.B.
;;      gibt es ein bitmasken startflag: angefangen bei
;;      0b10000000 geht es in der auesseren schleife nach:
;;      0b01000000, 0b00100000, 0b00010000 ... usw.

shiftright001:
sbrc r16, 7
rjmp shiftright001en0d
lsl r16
rjmp shiftright001
shiftright001end:

;; ...

ldi r20, 0b10000000
div000:
mov r19, r20
div001:
mov r18, r16        ;;  den dividenden aus dem bleiben r16 ins tmp r18
and r18, r19        ;;  den divenden im tmp mit der bitmaske verknuepfen

;; hier die eigentliche subtraktion

cp r18, r17
brlt subtraction001end
subtraction001:
sub r18, r17
subtraction001end:
```

```
;; hier die eigentliche subtraktion beendet...

lsr r19             ;; die maske eines rechts um sie zu vergroessern
or r19, r20
rjmp div001
lsr r17             ;; den divisor weiter nach rechts schieben
lsr r20
rjmp div000

end: rjmp end
```

Probieren wir es noch mit einem Hardware algo zum Beispiel als ASM Diagramm...

Folgendes: man kann den Wert retten über Push und zurückholen mit Pop

Aber: man nimmt eine Maske: `0b1000.0000`

Diese wird erweitert:

```
0b1100.0000
0b1110.0000
```

Ok...

Bis es größer ist als der Subtrahend

Danach wird die Maske zurück gesetzt. Ist das geschehen werden sowohl die Maske als auch der Subtrahend eines weiter nach rechts.... ? Bzw. Dann um die entscheidende schon (Anzahl) subtrahierter Stellen geschoben.

So, jetzt mache ich weiter mir ist aufgefallen

Weil sozusagen

Also das mit der Maske ist infrage zu stellen, wenn es soweit geht wie an dieser Stelle

Also man muss quasi schon den ganzen Divisor nehmen Wenn man den hat, muss man auch dadurch teilen aber man muss sozusagen den Teil vom Dividenden bei dem man eben subtrahiert muss man eben immer 1 Bit, dann ein zweites Bit, dann ein drittes Bit und so weiter

So wie ich das auf dem Papier gemacht hab auch wenn das nicht passt und am Anfang ergibt sich dann mehrfach null und es ist ja auch richtig so. Also ich muss den Anfang nicht anders machen als sonst. Und dabei muss sozusagen die Maske die ist schon richtig, die muss dann sozusagen wachsen wenn dann die Subtraktion funktioniert hat, indem ich das abziehe und der Subtrahieren ist kleiner wie Minuend dann dann wenn es der Fall ist dann muss ich sozusagen die Maske wieder auf den Anfang zurück zurücksetzen plus dem was sozusagen erhalten geblieben ist bei der Subtraktion

```
;; so ist besser

;; david vajda
;; div command bitwise
;; 06/25/2026

.include "m8def.inc"

.equ NUM    =   7
```

```
ldi r16, HIGH (RAMEND)
out SPH, r16
ldi r16, LOW (RAMEND)
out SPL, r16

ldi r16, 0xff
out DDRD, r16

ldi r17, NUM
ldi r21, 0x00

loop1:
;; mov r18, r17 und damit auch nicht, brauchen wir nicht...
;; andi r18, 0b10000000;; brauchen wir nicht
sbrc r17, 7
rjmp loop1end
lsl r17

lsl r21
ori r21, 0b00000001

rjmp loop1
loop1end:

com r21
out PORTD, r21

end: rjmp end
```

davor:

```
;; david vajda
;; div command bitwise
;; 06/25/2026

.include "m8def.inc"

.equ NUM    =   8

ldi r16, HIGH (RAMEND)
out SPH, r16
ldi r16, LOW (RAMEND)
out SPL, r16

ldi r16, 0xff
out DDRD, r16

ldi r17, NUM
ldi r19, 0b10000000
ldi r20, 7
```

```
loop1:
;; mov r18, r17 und damit auch nicht, brauchen wir nicht...
;; andi r18, 0b10000000;; brauchen wir nicht
sbrs r17, 7
rjmp loop1end
lsl r17
rjmp loop1

loop1end:

out PORTD, r19

end: rjmp end
```

Ich probiere jetzt die Maske zu erzeugen, es gibt übrigens wahrscheinlich ein besseres Mittel da das subtrahieren oben stattfindet den divisor so weit nach oben bis zur 1 schieben, gleichzeitig Maske erzeugen am Ende komplemtär bilden und: nach oben schieben...

das tut...

die ausgabe entsprechend invertiert...

```
.equ NUM    =   207
.equ NUM    =   63
.equ NUM    =   62
.equ NUM    =   8
```

hier photos....

```
;; david vajda
;; div command bitwise
;; 06/25/2026

.include "m8def.inc"

.equ NUM    =   207

ldi r16, HIGH (RAMEND)
out SPH, r16
ldi r16, LOW (RAMEND)
out SPL, r16

ldi r16, 0xff
out DDRD, r16

ldi r17, NUM
ldi r19, 0b10000000
ldi r20, 7

loop1:
mov r18, r17
```

```
and r18, r19
cp r19, r18
breq loop1end
lsr r19
dec r20
rjmp loop1

loop1end:

out PORTD, r19

end: rjmp end
```

Daniel Vajda
06/25/2026
Binäres Dividieren, Seite 1

echo $((\$RANDOM % 4096))
echo $((\$RANDOM % 57))

erzeugt in der Bash 2 Zufalls-Zahlen von der große passend...
zu unserer Aufgabe...
Dividend und Divisor...
ok: Zusatzübung - ruhig zu gemütlich als zu ungemütliche...
1010=1000 1110b
a := 2702dec ← prüfen mit pc
b := 10dec

bin(a) = bin(2702d) =
2702/2 = 1351 Rest 0
-2
07 -12 675/2 = 337 Rest 1
-6 15 -6 337/2 = 168 Rest =
10 -14 07 -2 168/2 = 84 Rest
-10 11 -6 13 -16 8412 = 42
02 -10 15 -12 08 Rest 0
-2 1 -14 17 -8 4212 = 21 Rest
0 1 -16 0 212 = 10
1 0
512 = 2 Rest 1 1012 = 5 Rest 1
212 = 1 Rest 0 112 = 0 Rest 1

Damid Vajda
06/25/2026
Seite 2 binäres dividieren...

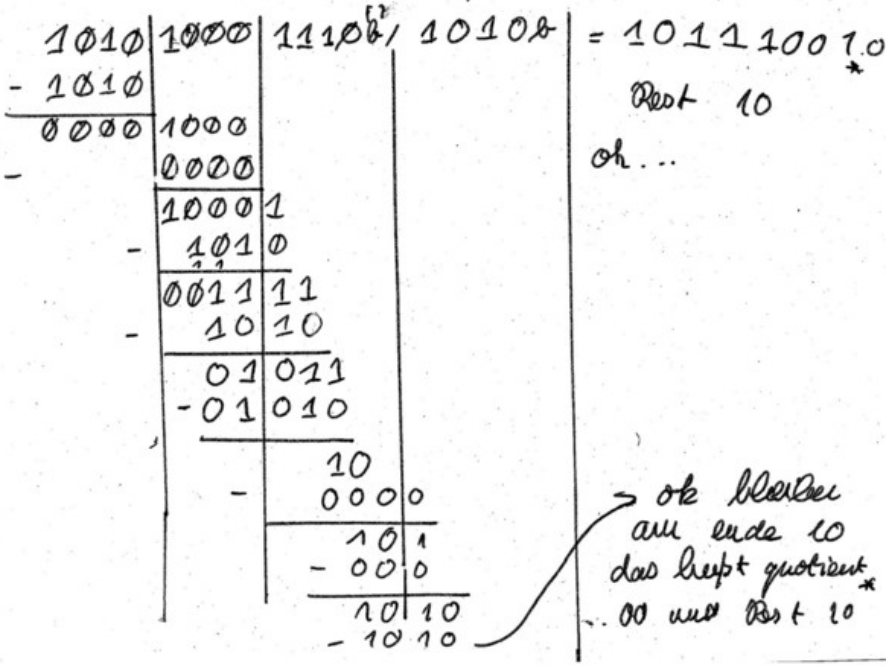
$bin(10_{dec}) = 1010_b$

oh: quotient - dividend / divisor + rest

$z = \frac{a}{b} + r$

$1010.1000.1110_b / 1010_b =$

Schreiben wir auf ab...



David Vajda
06/25/2026
Binäres Dividieren, Seite 3

also ergibt's:

quotient := dividend / divisor + rest

$$r = \frac{a}{b} + \pi$$

$$\frac{1010.1000.1110b}{1010b} =$$

(k, r) = 10111001.00 Rest 10

10.1110.0100b Rest 10

oh:

gucken:

wahrscheinlich falsch? oder nicht?

10.1110.0100b
512 128 64 32 4

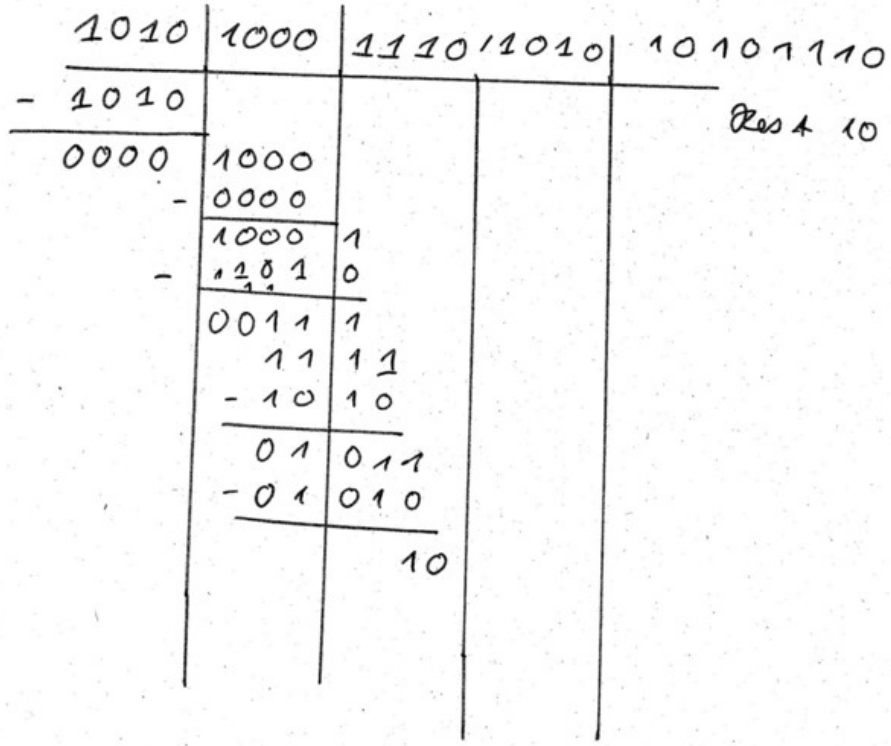
512 + 128 + 64 + 32 + 4 = 740

640 + 96 + 4

2702/10

ok falsch...

David Vajda
06/25/2026
Binäres Dividieren, Seite 4



ok wieder falsch!

Jawd Vajda
06/25/2026
binäres Dividieren, Seite 5

gegenversuch: (dezimal)

1 ersetzen wir durch 7
und 0 durch 2

also

7272722277721/7272 = 10
- 7272

07

ok hier

David Ugda 06/25/2026

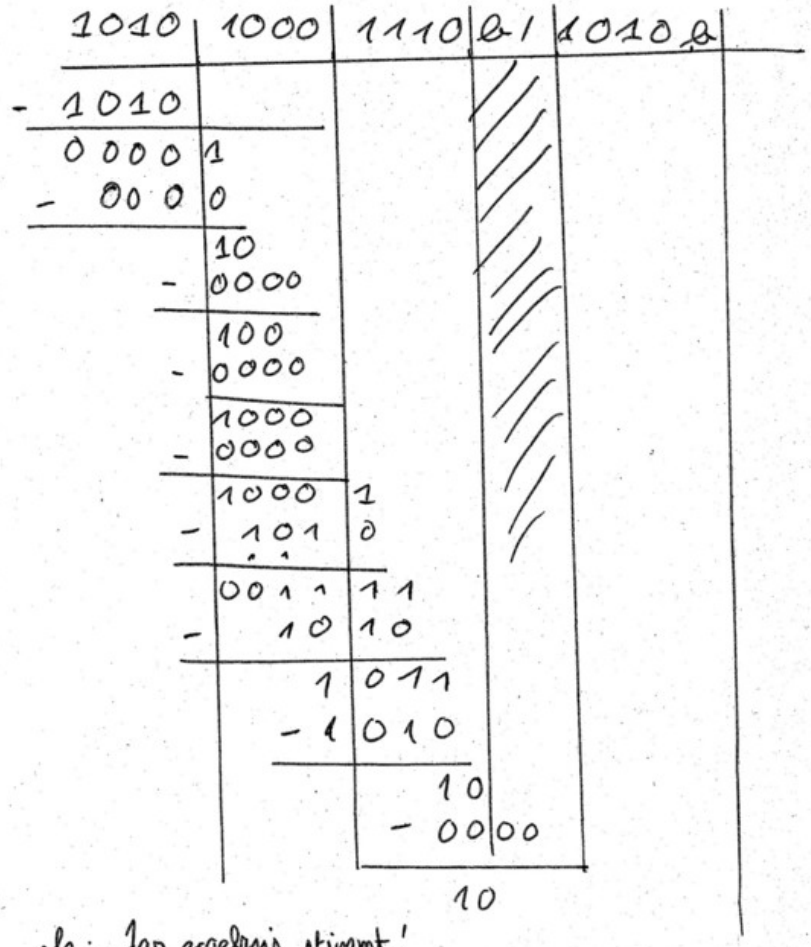
Binäres Dividieren, Seite 6

noch mal:

ergibt dann

↓

10000 1110 Rest 10



ok: Das ergebnis stimmt!

David Vajda
Binäres Dividieren
Seite 7, 06/25/2026

Der erste Schritt:

a) die most significant 1 finden
b) wie das geschieht ist egal
c) entsprechend eine Länge für die Bitmaske nach dem Schema

0000... 1111...
LLLL.... HHHH...

ok...

d) also die Bitmaske entspricht der Anzahl der Ziffern die die Zahl "real" einnimmt
e) die Zahl ganz nach oben (links) schieben. Entspricht
→ [Registerbreite − Länge]
f) egal was im ersten Operanden steht. r16, r17, r18... o. ähnlich dem dividend:
wo hier die most significant 1 ist...

Daniel Voyda
binäres Dividieren
Seite 8, 06/25/2026

Beginnen wir die Subtraktion
ganz vorne...

g) wir müssen darauf achten:
mit jeder "nicht gelungenen"
Subtraktion...
Spruch: Minuend ist kleiner —
bei der Division: gratiert
0 Stelle:
wird jedes mal neu eine null
angehängt

0010
− 1000 = 0
⟶ 0 ... immer 1
Stelle ...

a) bei der Subtraktion darauf
achten: keine negativen zweier-komplement
wert ergeben!
kein swap around!
sondern:

→ nicht ausführ
wenn 0 ergäbe
minuend und
0 am quotient
den immer weite
schie
...

if: Minuend < Subtrahend
Subtraktion

Javid Vajda
Binäres Dividieren
Seite 7, 06/25/2026

Der erste Schritt:

a) Die most significant 1 finden
b) wie das geschieht ist egal
c) entsprechend eine Länge für die Bitmaske nach dem Schema

0000... 1111...
LLLL.... HHHH...

ok...

d) also die Bitmaske entspricht der Anzahl der Ziffern die die Zahl "real" einnimmt
e) die Zahl ganz nach oben (links) schieben. Entsprechen
→ [Registerbreite − Länge]
f) egal was im ersten Operanden steht. r16, r17, r18... o. ähnlich dem Dividend:
wo hier die most significant 1 ist...

Daniel Voyda
binäres Dividieren
Seite 8, 06/25/2026

Beginnen wir die Subtraktion
ganz vorne...

g) wir müssen darauf achten:
mit jeder "nicht gelungenen"
Subtraktion...
Spruch: Minuend ist kleiner —
bei der Division: gratiert
0 Stelle:
wird jedes mal nur eine null
angehängt

0010
− 1000 = 0
⟶ 0 ... immer 1
Stelle ...

h) bei der Subtraktion darauf
achten: keine negativen zweier-Komplement
wert erzeugen!
kein wrap around!
sondern:

wenn: Minuend < Subtrahend
Subtraktion

→ nicht auf die
ein 0 angehängt
minuend und
0 an quotient
den immer 1
weile
solle
...

David Vajda Privates
Livrdioren
Seite 9, 06/25/2026

also: wir nehmen mal

r8 und r9

zunächst wollen wir die most significant 1 in r8 heraus-finden...

Dazu fangen wir oben an und testen das bit.

Dazu gibt es verschiedene Möglichkeiten... zwei stelle ich vor:

1.) entweder wir testen bambaret das bit mit

sbic sbis gibt es bei i/o registern gibt bei "normalen" general purpose registern auch...

Dazu laden wir die zahl..

7, 6, 5, 4, 3, 2, 1

wie auch immer sbis und sbic erwarten beim register aber ist wohl möglich...

anderer befehl...

ansonsten führen sie den nächsten

Befehl aus 0. nicht. würde sich zum
Beispiel bei rjmp lohnen...
Man könnte dann rausspringen wenn
die 1 gefunden wäre aus der Schleife

2.) Möglichkeit wäre:
wir nehmen

and

and r8, r9

in r9 schreiben wir
1000.0000 b

und dann
and r8, r9
vergleichen es mit r9
cp r8, r9
wenn es passt ist in r8 da
eine 1 wo in r9
ausserdem r9 weiter nach
rechts schieben
ist r9 ok...

David Vajda Privates
Seite 9, 06/25/2026

also: wir nehmen mal

r8 und r9

zunächst wollen wir die most significant 1 in r8 heraus-finden...

Dazu fangen wir oben an und testen das bit.

Dazu gibt es verschiedene Möglichkeiten... zwei stelle ich vor:

1.) entweder wir testen konkret das bit mit

sbrc sbis gibt es bei i/o registern gibt bei "normalen" general purpose registern auch...

Dazu laden wir die zahl..

7, 6, 5, 4, 3, 2, 1

wie auch immer sbis und sbrc erwarten beim register aber ist wohl möglich...

anderer befehl...

ansonsten finden sie den nächsten

Daniil Voyda
binäres Dividieren
Seite 8, 06/25/2026

Beginnen wir die Subtraktion
ganz vorne...

g) wir müssen darauf achten:
mit jeder "nicht gelungenen"
Subtraktion...
Spruch: Minuend ist kleiner —
bei der Division: gestrichene
0 Stelle:
wird jedes mal nur eine null
angehängt

0010
− 1000 = 0
⟶ 0 ... immer 1
Schritte...

a) bei der Subtraktion darauf
achten: keine negativen zweier-Komplement
wert ergeben!
kein wrap around!
sondern:

wenn: Minuend < Subtrahend
Subtraktion

→ nicht möglich
um 0 zu erreichen
minuend und
0 am quotient
den immer 1
weiter
schieben
...

Javid Vajda
Binäres Dividieren
Seite 7, 06/25/2026

Der erste Schritt:

a) Die most significant 1 finden
b) wie das geschieht ist egal
c) entsprechend eine Länge für die Bitmaske nach dem Schema

0000... 1111...
LLLL.... RRRR...

ok...

d) also die Bitmaske entspricht der Anzahl der Ziffern die die Zahl "real" einnimmt
e) die Zahl ganz nach oben (links) schieben. Entsprechen $\rightarrow$ [Registerbreite − Länge]
f) egal was im ersten Operanden steht. ↑16, ↑17, ↑18... o. ähnlich dem Dividend: wo hier die most significant 1 ist...

befehl aus 8. nicht. würde sich zum
beispiel bei rjmp lohnen...
Man könnte dann rausspringen wenn
die 1 gefunden wäre aus der Schleife

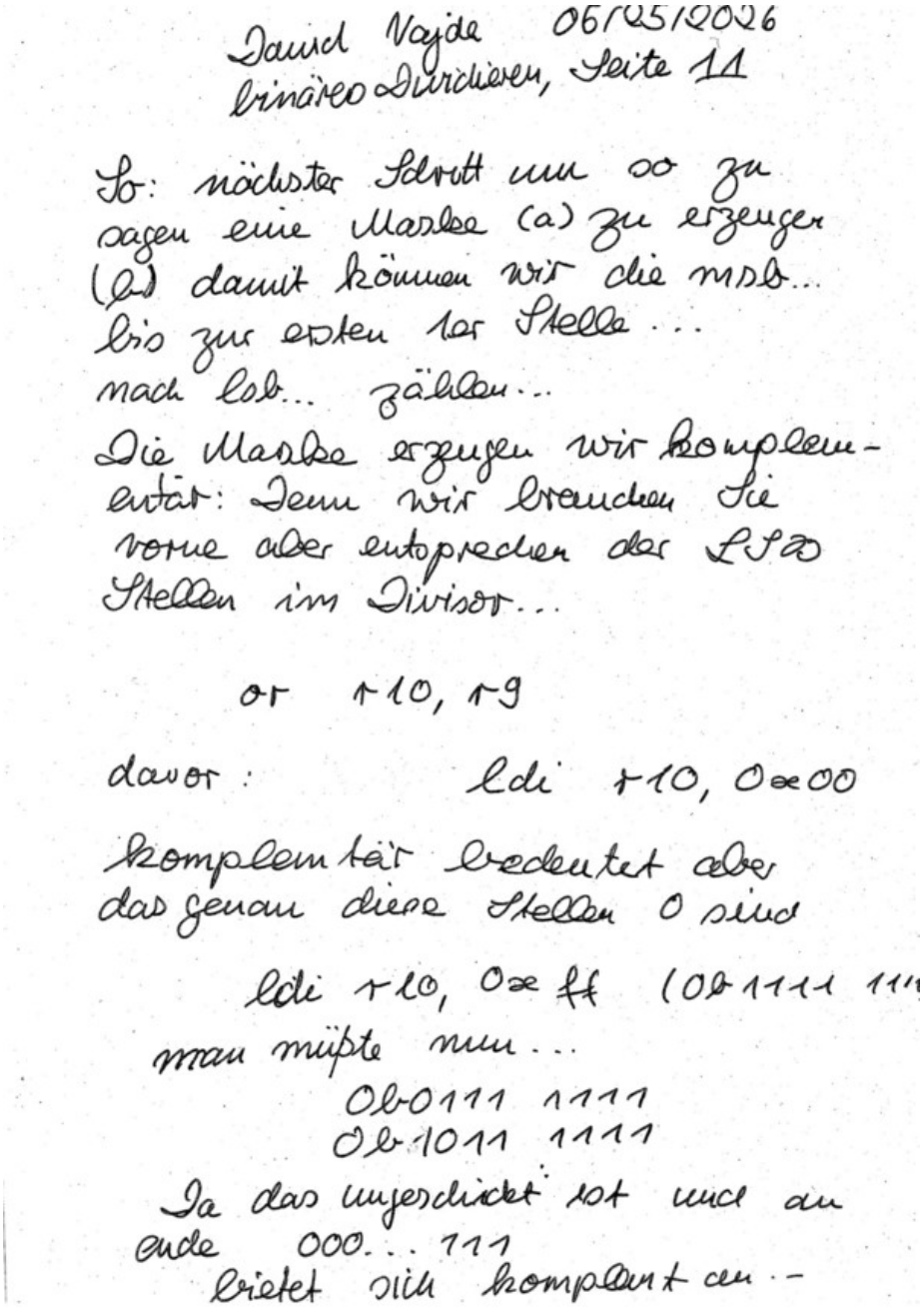
2.) möglichkeit wäre:
wir nehmen

and

and r8, r9

in r9 schreiben wir
1000.0000 b

und dann
and r8, r9
vergleichen es mit r9
cp r8, r9
wenn es passt ist in r8 da
eine 1 wo in r9
anonsten r9 weiter nach
rechts shiften
bis r9 ok...


David Vajda 06/05/2026
binäres Dividieren, Seite 11

So: nächster Schritt nun so zu sagen eine Maske (a) zu erzeugen (b) damit können wir die MSB... bis zur ersten 1er Stelle ... nach lsb... zählen...

Die Maske erzeugen wir komplementär: Denn wir brauchen sie vorne aber entsprechen der LSB Stellen im Divisor...

or r10, r9

davor: ldi r10, 0x00

komplementär bedeutet aber das genau diese Stellen 0 sind

ldi r10, 0xff (0b1111 111

man müßte nun...

0b0111 1111
0b1011 1111

Da das umgedreht ist muss am ende 000...111 bleibt sich komplett an -

Den Fehler gefunden. Tatsächlich muss man Stelle für Stelle machen. Ich hab's dadurch ersetzt, dass ich nicht binär dividiert habe sondern Dezimal. Ich habe die eins durch Siebener ersetzt und die 0er durch zwei

Da, ich dezimal gewöhnt bin schriftlich zu dividieren. Bin er eigentlich nicht bin ich auch davon ausgegangen, dass ich hier wahrscheinlich auf die Lösung komme? Tatsächlich bin ich jetzt dahinter gekommen, dass man hier trotz dem langen längeren, sozusagen divisor am Dividenden Ziffern für Ziffern. Macht hier das Beispiel

Ok, scheinbar ist noch Schwierigkeit da...

Oder nein es ist genau eine 1 falsch aber die ist bei mir sicher nicht falsch...

Also: der Rechner: 100001110

Ok, überprüfen wo Fehler...

erst eine handschriftliche rechenaufgabe, ich mache nachher einen beitrag aus allem hier, vorher muss die uhr fertig werden. so.

da kariertes papier selber mehr kostet und man es nicht ausdrucken soll, der toner ist eine sehr teure alternative. schlage ich vor, nach 4 ziffern binaerzahlen jeweils einen senkrechten strich zu machen

ordnung muss sein, erst ordnung und das heisst, jeden tag binaer dividieren und nicht uhr machen, das ist allerdings eher ein schoenes beispiel und es gibt unglaeubige es lohnt sich nicht, diese staendig zu besiegen, weil die unterliegen einem druck, der alles am ende so schlecht macht trotzdem ist es manchmal hilfreich, will man nicht, dass es einem wiederum so schlecht geht, dass auch nichts mehr

uhr machen generell aber nur bei staendigen lernen. ohne dabei steine zu bewegen, was zu achten ist, aber so einfach ist es nicht

derartige reden koennte man sich erspaeren, sie sind ein heisslicher in der gesellschaft, wo die gesellschaft angefangen hat sich zu verderben und ihr spezial ideen unter das volk jubelt, selber nicht mehr sehen, ihre spezialideen das gegenteil sind

leider sind eben - die leute die das alles so wenig ernst nehmen so, dass sie sich damit dann auch eben dahin bewegen, wo sie leuten was erklaeren, hinterher alles so machen, was nicht zu ist.

um zu erklaeren, wie man alles sehr genau macht. das stimmt nicht, sondern liegt am thema, dass man sich nicht konzentriert, der innere schweine hund, mit verwechslung + eben schon, dass man kein wort versteht, ausser reden, auf die man antwortet

und: sich an das aeussere nicht gewoehnt, die schwierigkeit mit etwas in der umgebung zu recht zu kommen

so, ich muss jetzt weiter machen...

jetzt kommt heute die uhr...

heute mache ich die uhr ... das geht so:

als erstes ueben wir noch mal das binaere dividieren....

so, das geht so:

1.) machen wir noch andere uebungen: die kerze: die ist zur zeitmessung wichtig. wir muessen eine kerze abbrennen lassen, es gibt wahrscheinlich genormte - damit koennen wir erstens feststellen, brennt die konstant ab... damit haben wir eine moeglichkeit zeitintervalle fest zu stellen 2.) es gibt die moeglichkeit ausserhalb der uhr, so zu sagen, diese ist eigentlich ein hilfsmittel - das wird fuer unkundige zum zeitablesen genutzt, (a), und (b) es geht auf die schnelle

nicht enthalten in einer uhr: ist das datum, wenn sich die uhr alleine nach der sonne richtet, bzw. wir das datum nicht haben

was fuer andere moeglichkeiten gibt es noch: z.B. die temperatur. es gibt eine orts temperatur linie

andere argumente, wir koennen ja gucken, wie viel - deswegen die kerze. wie lange ist der tag, um den stand der sonne zu betrachten, verfehlen eben ihr ziel, weil ich, wenn ich den tag nicht weiss, so zu sagen, nicht messen kann, wie lange der tag ist, ohne eine uhr zu haben

eine moeglichkeit ist eine kerze. wenn diese konstant abbrennt. doch das muessen wir in der praxis herausfinden, durch stetiges experimentieren. allgemein literatur, die darauf hinweist, was man alles koennte, wo probleme liegen, reichen nicht

wir muessen uns, selbst, wenn kerzen konstant abbrennen haben wir keine aussagen ueber kerzen die es nicht tun, z.B. wenn sie eine art kegel bilden, d.h. oben spitz sind. wir koennten auch dann davon ausgehen - dass so zu sagen, diese kerzen nicht geeignet, und uns an einem gewoehnlichen menschlichen mass orientieren

wenn wir klug sind, koennen wir auch hier, selbststaendig, aber ohne trick 17, das heisst, wir benuetzen das intelligenter weise, natuerlich wuerden wir uns im alltag, die arbeit nicht schwerer machen, indem wir jetzt staendig spitze kerzen benutzen - sondern wir unabhaeng der nutzbarkeit, ohne diese jemals in erwaegung zu ziehen, ein mass fuer das abbrennen und die zeit t spitzer kerzen heraus

selbst wenn das bekannt waere - aehnlich der Nato Codes Alpha, Bravo Charlie und der Monate Januar 31 tage, Februar 28/29 Tage und so weiter, genuegt nicht das wissen, das muss eintrainiert werden

so. das ist das eine. und somit genuegt es nicht, den stand der sonne

man kann es sich jetzt sehr schwer machen. und versuchen den stand der sonne selbststaendig heraus zu finden. das ist natuerlich blanker unsinn, wenn lehrbuecher vorliegen, wahrscheinlich im internet, gibt es infos. das heisst, man muss gucken, wie das mit der sonne ist, dem stand zur uhrzeit

ich habe eine kontraindikation. demnach muesste die sonne meiner meinung nach, mittags um 12:00 uhr, so zu sagen, senkrecht oben stehen, am zenit sagt man ja, medium coeli, sind wiederum begriffe die ich persoenlich nicht eintrainiert habe, womit man wiederum leuten angst machen, seil weil die so nach koehli klngen und astrologie klingen, dass da leute schon schnell was vermuten

so, das ist das eine. und also: die sonne war 13:00 uhr, etwa senkrecht oben. die tageslaenge ist die frage.

das datum. erstaunlicher weise, hatte ich das sternzeichen krebs wiedererkannt, da wusste ich nicht mal so, gut, wie die tierkreiszeichen heissen wenn man das datum wiederum haben will, dann muss man folgendes machen, nachts in den himmel gucken.

und da sind die sternbilder zu sehen, wenn man glueck hat, man muss auch nicht lernen. gehen wir davon aus, dass es objekte gibt, die innerhalb des monats, oder mit ueberschneidungen erlauben alles zu wissen, dann sage ich:

gut es gibt im netz sicher 12 karten, entsprechend vom 21. monat (0) bis zum 21. monat (1) und dann kann erst mal das machen

das heisst, die himmelkarten nach geben was wieder. und das muss man eintrainieren. das guckt man nicht vom papier ab. deswegen nimmt man ein papier und malt die objekte immer wieder auf das papier, wie chinesische zeichen

so, wenn man das hat. und die kerze bleibt, die tageslaenge usw.

jetzt trotzdem die uhr, mit dem atmega8 geht so:

als, erstes muessen wir binaer dividieren. wir muessen den atmel herstellern nicht unterstellen sie koennten das binaere dividieren (also div) befehl nicht, es ist gut, dass sie das nicht haben

atmel wird auch in diese reihe professionell eingesetzt: sim karte, also also: bankkarte und so weiter

trotzdem sind das dinger zum lernen, eher. so: wenn das alles da ist, ich habe bisher keine vorstellung irgendwo erhalten, wie man das dividieren macht

und es ist sicher trotzdem kein verbrechen das vor zu schlagen. so, das hat folgendes: wir haben in der grundschule bei frau grimme normales handschriftliches rechnen ordentlich gelernt, ebenso wie schreibschrift, das pavillion war fuer die besseren und frau grimme war sicher kein chaot

so: deswegen habe ich mich spaeter gewundert, warum schreiben andere handschriftlich mit druckbuchstaben, das ist schlecht, wenn man sich im schlechten an anderen ein beispiel nimmt

bei frau grimme hat man alles gelernt, was zu ordentlichkeit fuehrt und damit ist kein wenig gegeben, dass das sich spaeter nicht bemerkbar macht und dass vor allem in deutschland das standard ist, oder irgendwo auf der welt

hier sieht man ordentliche lehrerinnen in den ersten jahren. was wiederum meine - spaetere deutschlehrerin auch feststellen, was die lehrerin verbrochen im postiven hatte, hat sich dann spaeter - wieder - und wir kennen die unterstellungen von aussen sage ich mal - die deutschlehrerin obwohl wir schon ordentlich so zu sagen im zeichen des rabatz standen muss ich sagen, hat sich da ordentlich was durchblicken lassen, ich schreibe normale handschrift, da hat die deutschlehrerin gesagt, ich haette ja eine schoene schrift, obwohl sie druckbuchstaben war

und genau die, die immer alles so toll feststellen, sind ja damals die gewesen, die haetten sie im paviloin nicht zu gelassen, sind aber die, die immer alles sagen. die probleme hatten wir quasi schon, das einzige hilft, durchsetzen, fertig...

jedenfalls durch das schriftliche dividieren, was einen ueber taschenrechner abgewoehnt werden sollte, kann man erst binaer dividieren lernen. also, man kann die technik nicht verstehen, wenn man das so nicht macht

so, das geht dann wie gesagt so:

wenn ich eine binaer zahl habe und ich soll sie in die naechste zahl, also den dividenden rein packen, dann geht sie wenn man das richtig macht, 0x (kein mal) oder ein mal rein

anders bei 8 64, das geht 8 mal rein. ist allerdings 1b/11 oder 11/b

das heisst, das ist ein schlechtes beispiel, weil die 1 geht, schon in die 1 alleine rein. also, beim schriftlichen ist das so, das moechte ich nicht erlaeutern. damit bleiben die probleme erspart

so, und dann bleibt das problem, mit dem zweier komplement.

naemlich. ich habe wieder schriftlich subtrahieren geuebt

und mit der hand abgezogen

wenn ich

100 -10024

habe, dann ist das ergebnis negativ

ich habe subtrahend und minuend vertauschen, indem ich weiss, es ist negativ, mache ich das ergebnis negativ

so, wenn ich, das mit dem rein packen machen, dann also beim dividieren muss ich subtrahieren

wenn 100 - 1000 geht die 1000 null mal rein

dann muss ich einen groesser kleiner vergleich machen, ob die cpu intern zweier komplement benutzt ist mir egal.

nur, ich tue das nicht. das heisst, der vergleich mal, wenn kein mal, 0 merken, als teil vom quotient und der rest kommt dann

wenn ich das habe.

dann kann ich dividieren

das muss als erstes her. damit kann ich eine zahl darstellen. so, zum beispiel eine 5 stellige dezimalzahl

im naechsten schritt, muss ich

durch 24 teilen, durch 60 und noch mal. stunden und minuten kommen zusammen

aber: vorsicht: die tage im jahr sind 365...

aber auch 366

jetzt brauche ich einen vierblock. ebenso bei monaten

31 + 28 + 31 + 30 + ...

aber: im viererblock

seit 1980??? eben hier wiederum ein datum, was aus dem FF kommen muss

so: und dann brauche ich eine sehr lange speicherstelle, denn das datum wird ein ticks, aber in sekunden

das aktuelle datum kann ich per rs232 vom pc uebernehmen

+ monate als zeichenkette (das ist standard) + alpha, beta -... zeitzone, utc. usw. ok. fangen wir an. gleich....

;;; ich weiss jetzt, wo ein eindeutiger fehler war, und ist dies muss vor jedem einlesen geschehen:

sbi ADCSRA, ADSC

und das ist logisch. denn der darauffolgende befehl lautet:

sbic ADCSRA, ADSC rjmp conversation001label001

also wird nach jedem abtasten das bit geloescht. das ist anders als bei rs232, wo: rs232 uebertragung startet und das UDRE jedes Mal getestet werden muss...

ok... wobei das mit 0V stimmt eben nicht...

Der im AVR eingebaute ADC ist ein 10-Bit-ADC, d. h. er liefert Messwerte im Bereich 0 bis 1023. Liegt am Eingangskanal 0 V an, so liefert der ADC einen Wert von 0. Hat die Spannung am Eingangskanal die Referenzspannung erreicht (stimmt nicht ganz), so liefert der ADC einen Wert von 1023...

ich weise, darauf hin, dass abtastrate/frequenz/sample rate zwischen 50 kHz und 200 kHz liegen muss. wenn der prozessor mit 1MHz arbeitet, der teiler aber 32 ist, ist das ergebnis: 31 kHz

das ist unter dem wert, der notwendig ist...

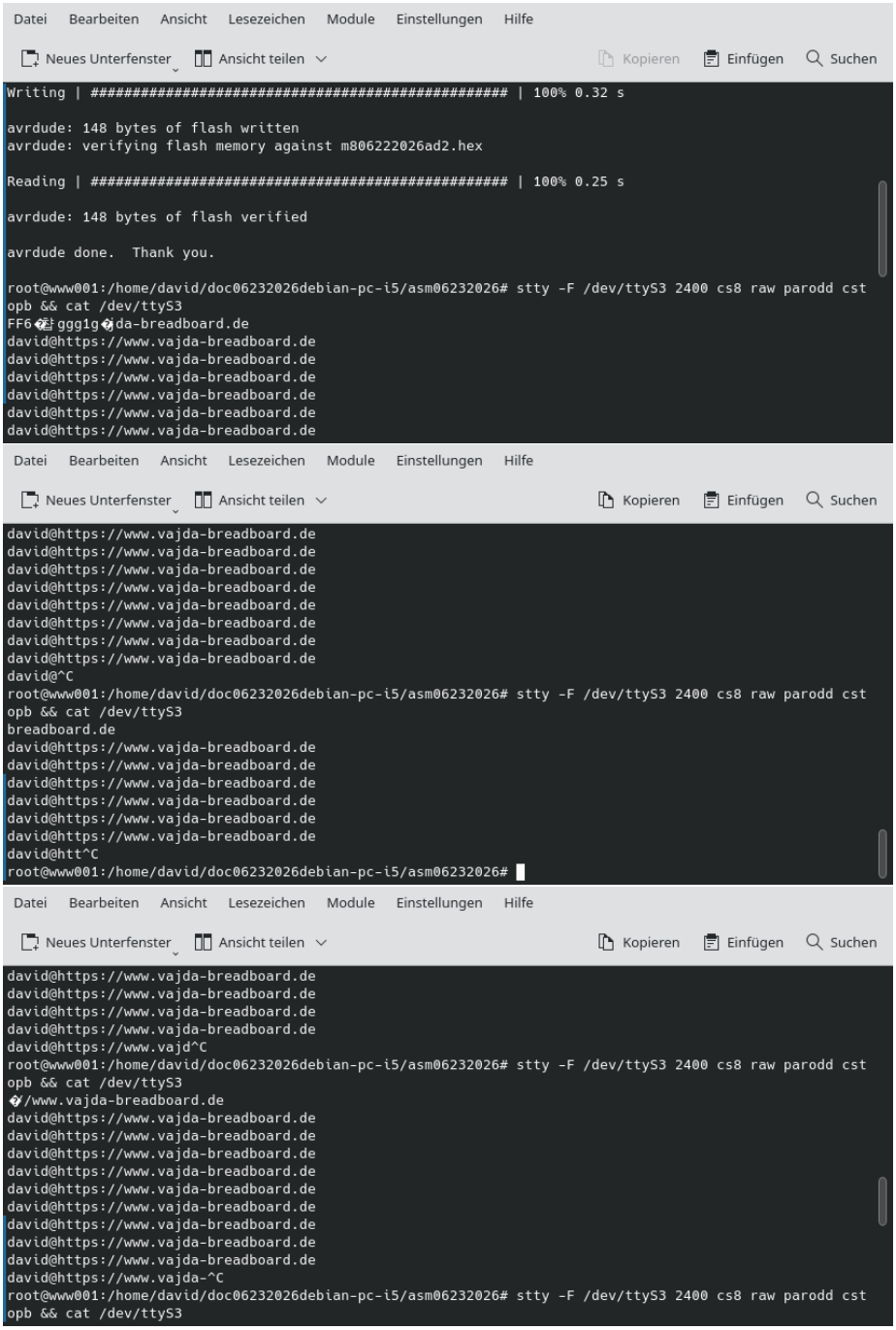
das ist das eine. das andere ist, dass ich nicht weiss, ob man bereits mit 0V messen kann, ich muss den bereich noch mal angucken, da ich aber glaube erfolg zu haben, werde ich versuchen danach den anderen controller zu nehmen und eine freuqenz zu erzeugen...

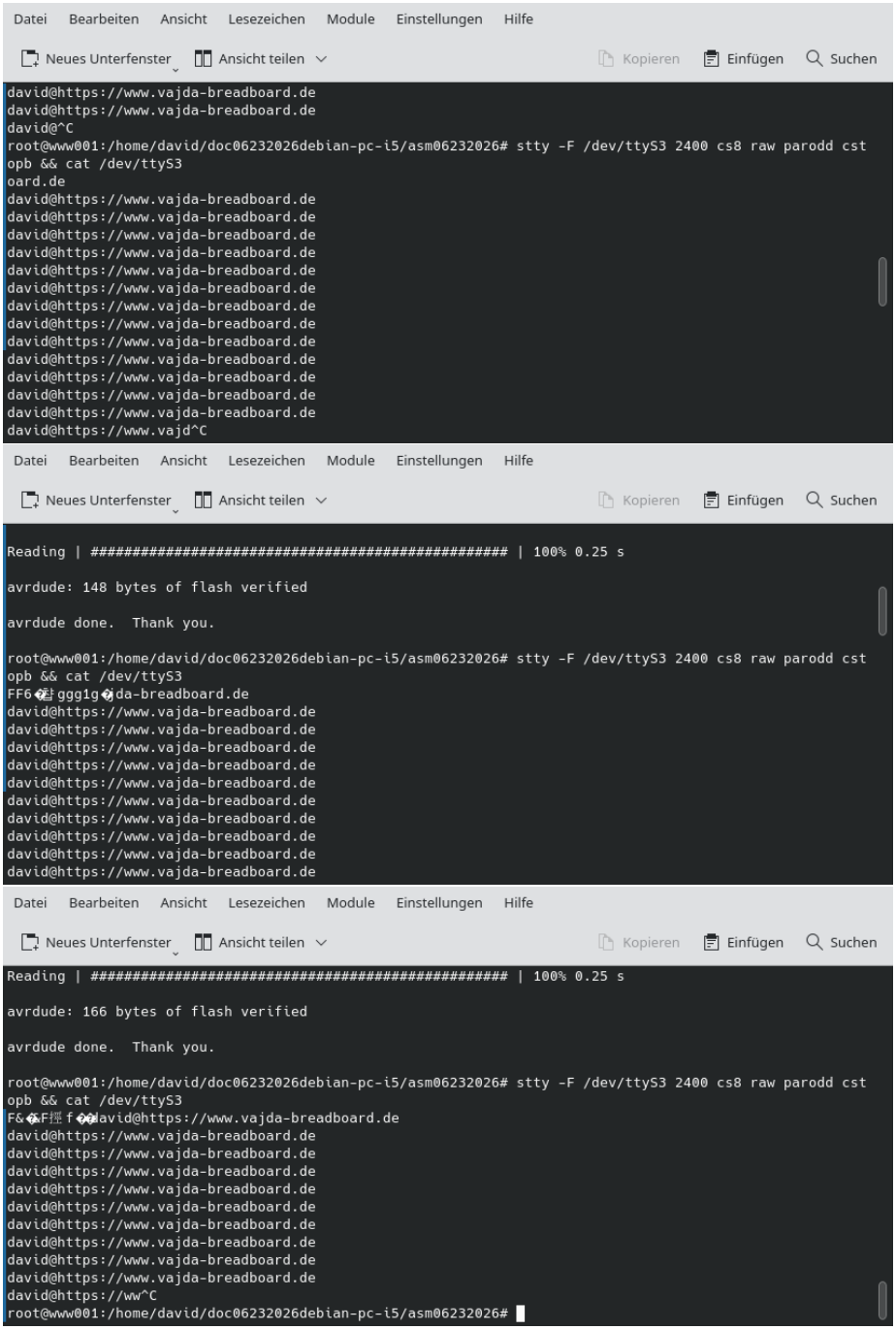
```
;; es sieht so aus, als haette ich ein wenig erfolg gehabt...

    ;; david vajda
    ;; 06/22/2026
    ;; m8 ad converter

    .include "m8def.inc"

    .dseg
    buffer: .byte 512
    cbuffer: .byte 512
    .cseg
```

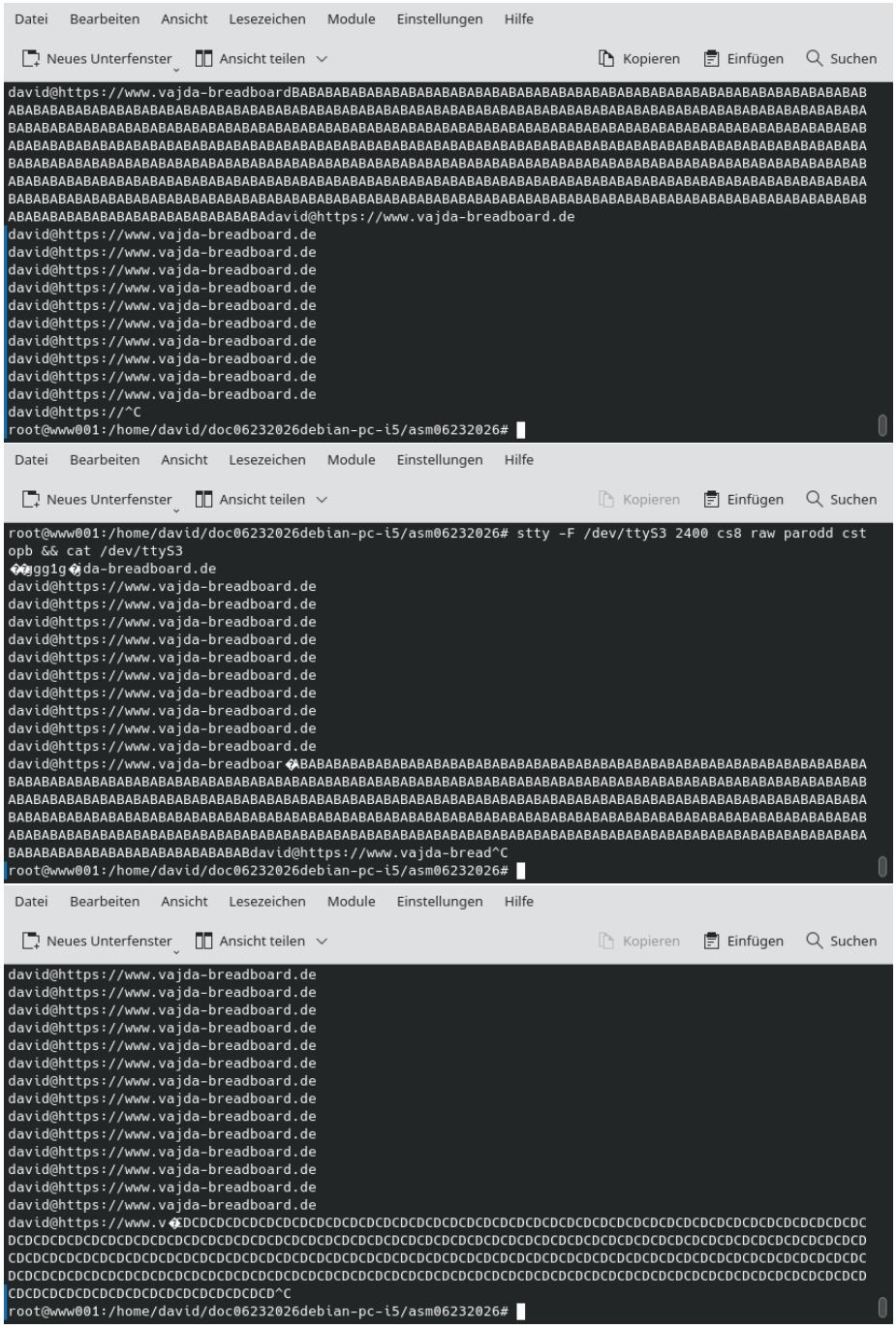
```
    ldi r16, HIGH (RAMEND)
    out SPH, r16
    ldi r16, LOW (RAMEND)
    out SPL, r16

    ldi r16, HIGH (25)
    out UBRRH, r16
    ldi r16, LOW (25)
    out UBRRL, r16

    ;; ldi r16, (1 << URSEL) | (1 << UCSZ2) | (1 << USBS)
    ldi r16, (1 << URSEL) | (1 << USBS)
    out UCSRC, r16
    ldi r16, (1 << TXEN) | (1 << UCSZ1) | (1 << UCSZ0) | (1 << UPM1) | (1 << UPM0)
    out UCSRB, r16

    ldi r16, (1 << REFS0)
    out ADMUX, r16

    ldi r16, (1 << ADEN)  | (1<<ADPS2)
    ;; ADPS2: Prescaler (Samplerate): 16, 1 MHz/16 = 250kHz/4 ~= 51kHz??
    ;; 50 kHz <= Samplerate <= 200 kHz
    out ADCSRA, r16

;;     sbi     ADCSRA, ADSC

    ldi r16, HIGH (buffer)
    mov YH, r16
    ldi r16, LOW (buffer)
    mov YL, r16

    ldi r18, 0xff

        sbi     ADCSRA, ADSC

    conversation001:
    sbic    ADCSRA, ADSC
    rjmp    conversation001
    in r17, ADCL
    in r16, ADCH
    st Y+, r17
    st Y+, r16
    dec r18
    brne conversation001

    label001:
    ldi r16, HIGH (buffer)
    mov YH, r16
    ldi r16, LOW (buffer)
```

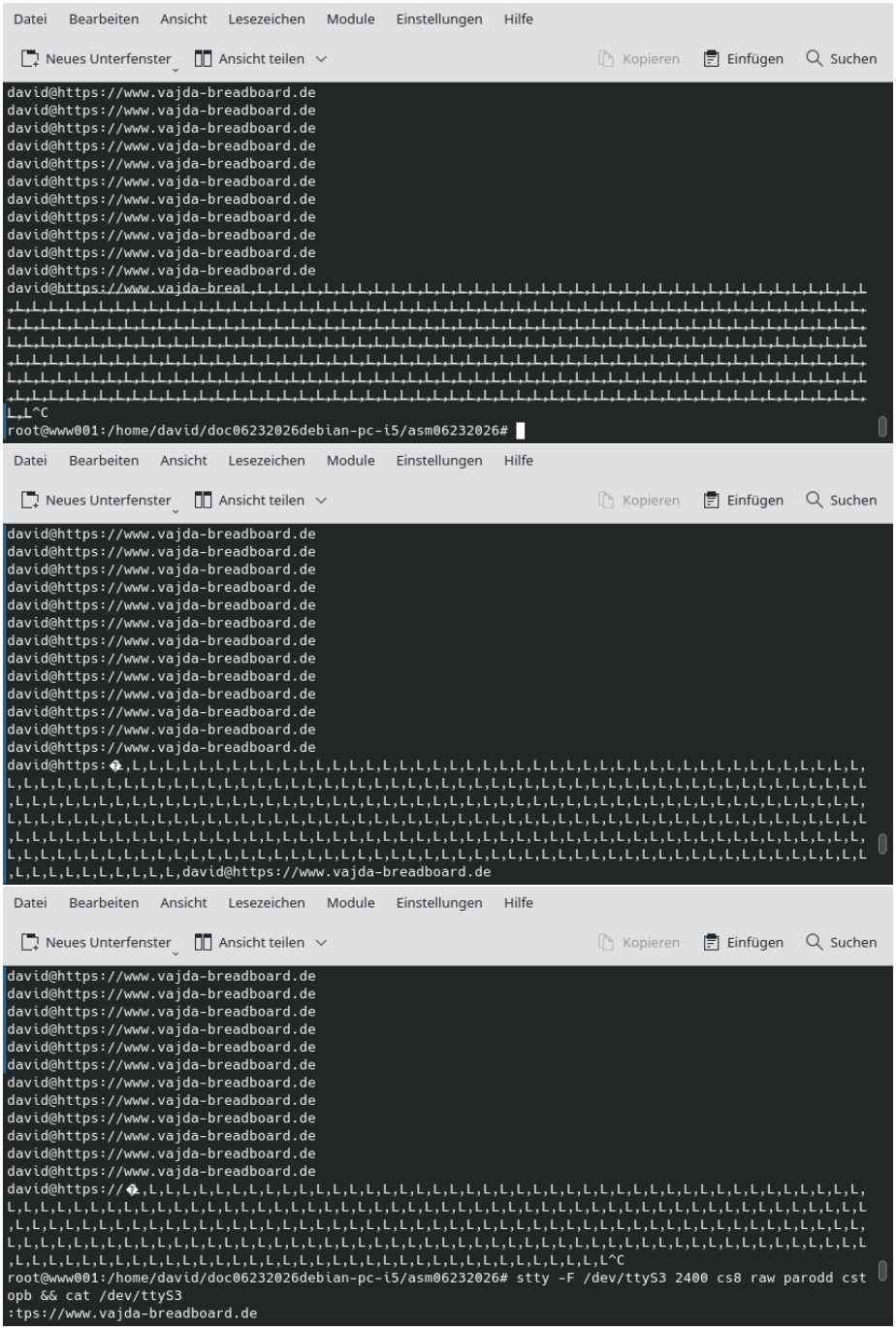
```
 mov YL, r16
 ldi r18, 0xff
 transmit001:
 ld r16, Y+
 rcall rs232transmitch
 ld r16, Y+
 rcall rs232transmitch
 dec r18
 brne transmit001
 rjmp label001



label000:
 rcall rs232transmitstr
 rjmp label000

 rs232transmitstr:
 ldi r16, HIGH (2*msg000)
 mov ZH, r16
 ldi r16, LOW (2*msg000)
 mov ZL, r16

 rs232transmitstr1:
 lpm r16, Z+
 cpi r16, 0x00
 breq rs232transmitstrend
 rcall rs232transmitch
 rjmp rs232transmitstr1
 rs232transmitstrend:
 ret

 rs232transmitch:
 sbis UCSRA, UDRE
 rjmp rs232transmitch
 out UDR, r16
 ret

 msg000: .db "david@https://www.vajda-breadboard.de", 10, 13, 0
```

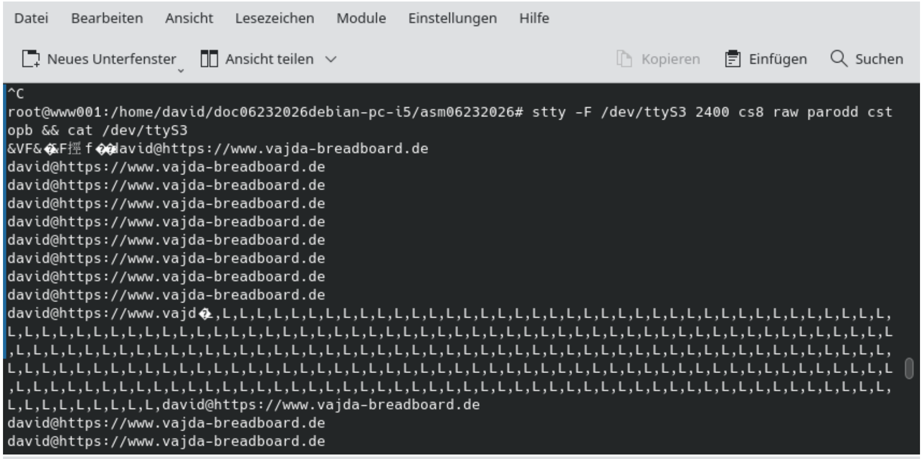
es hat trotzdem zunaechst mal nichts geaendert ich muss jetzt pause machen fuer heute, aber ich denke, es ist auf dem guten wege zur besserung. kabel auch noch mal ueberpruefen...

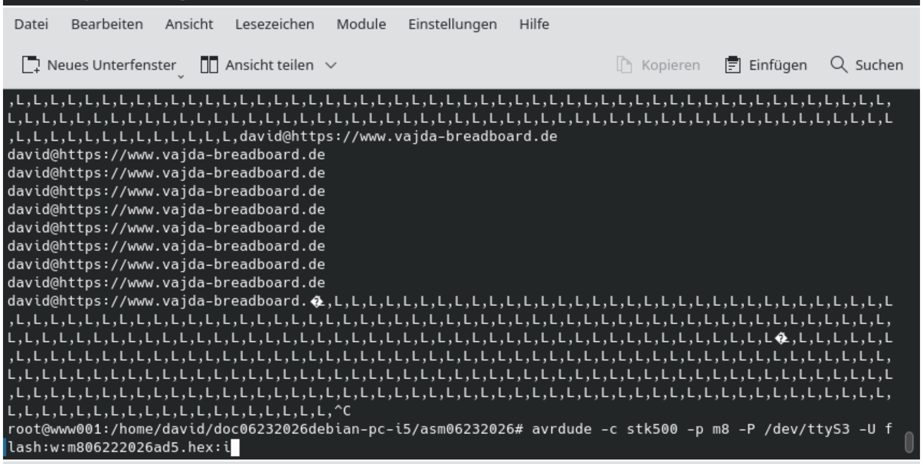

Datei Bearbeiten Ansicht Lesezeichen Module Einstellungen Hilfe
Neues Unterfenster Ansicht teilen Kopieren Einfügen Suchen
Writing | ## | 100% 0.32 s
avrdude: 148 bytes of flash written
avrdude: verifying flash memory against m806222026ad2.hex
Reading | ## | 100% 0.25 s
avrdude: 148 bytes of flash verified
avrdude done. Thank you.
root@www001:/home/david/doc06232026debian-pc-i5/asm06232026# stty -F /dev/ttyS3 2400 cs8 raw parodd cst
opb && cat /dev/ttyS3
FF6ggg1gda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
Datei Bearbeiten Ansicht Lesezeichen Module Einstellungen Hilfe
Neues Unterfenster Ansicht teilen Kopieren Einfügen Suchen
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@^C
root@www001:/home/david/doc06232026debian-pc-i5/asm06232026# stty -F /dev/ttyS3 2400 cs8 raw parodd cst
opb && cat /dev/ttyS3
breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@htt^C
root@www001:/home/david/doc06232026debian-pc-i5/asm06232026#
Datei Bearbeiten Ansicht Lesezeichen Module Einstellungen Hilfe
Neues Unterfenster Ansicht teilen Kopieren Einfügen Suchen
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajd^C
root@www001:/home/david/doc06232026debian-pc-i5/asm06232026# stty -F /dev/ttyS3 2400 cs8 raw parodd cst
opb && cat /dev/ttyS3
/www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-~C
root@www001:/home/david/doc06232026debian-pc-i5/asm06232026# stty -F /dev/ttyS3 2400 cs8 raw parodd cst
opb && cat /dev/ttyS3

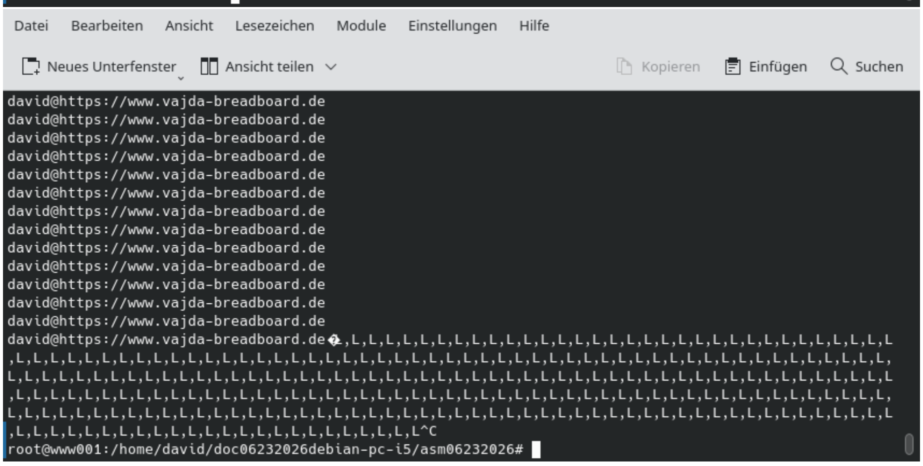

Datei Bearbeiten Ansicht Lesezeichen Module Einstellungen Hilfe
Neues Unterfenster
Ansicht teilen
Kopieren
Einfügen
Suchen
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@^C
root@www001:/home/david/doc06232026debian-pc-i5/asm06232026# stty -F /dev/ttyS3 2400 cs8 raw parodd cst
opb && cat /dev/ttyS3
oard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajd^C
Datei Bearbeiten Ansicht Lesezeichen Module Einstellungen Hilfe
Neues Unterfenster
Ansicht teilen
Kopieren
Einfügen
Suchen
Reading | ## | 100% 0.25 s
avrdude: 148 bytes of flash verified
avrdude done. Thank you.
root@www001:/home/david/doc06232026debian-pc-i5/asm06232026# stty -F /dev/ttyS3 2400 cs8 raw parodd cst
opb && cat /dev/ttyS3
FF6 ggg1g da-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
Datei Bearbeiten Ansicht Lesezeichen Module Einstellungen Hilfe
Neues Unterfenster
Ansicht teilen
Kopieren
Einfügen
Suchen
Reading | ## | 100% 0.25 s
avrdude: 166 bytes of flash verified
avrdude done. Thank you.
root@www001:/home/david/doc06232026debian-pc-i5/asm06232026# stty -F /dev/ttyS3 2400 cs8 raw parodd cst
opb && cat /dev/ttyS3
F& F f avid@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://ww^C
root@www001:/home/david/doc06232026debian-pc-i5/asm06232026#

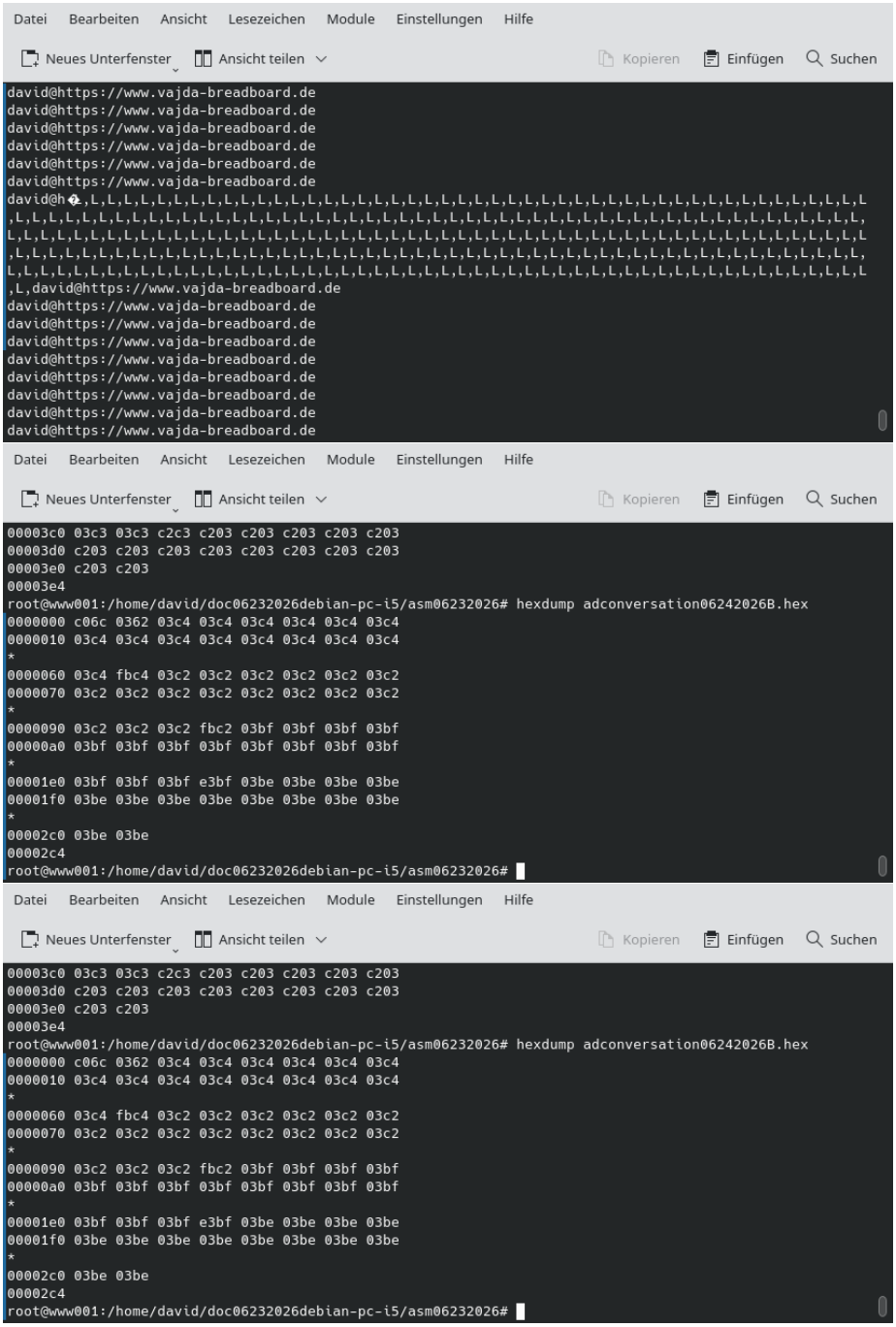

Datei Bearbeiten Ansicht Lesezeichen Module Einstellungen Hilfe
Neues Unterfenster Ansicht teilen Kopieren Einfügen Suchen
david@https://www.vajda-breadboard.de
root@www001:/home/david/doc06232026debian-pc-i5/asm06232026#
Datei Bearbeiten Ansicht Lesezeichen Module Einstellungen Hilfe
Neues Unterfenster Ansicht teilen Kopieren Einfügen Suchen
root@www001:/home/david/doc06232026debian-pc-i5/asm06232026# stty -F /dev/ttyS3 2400 cs8 raw parodd cst
opb && cat /dev/ttyS3
david@https://www.vajda-breadboard.de
root@www001:/home/david/doc06232026debian-pc-i5/asm06232026#
Datei Bearbeiten Ansicht Lesezeichen Module Einstellungen Hilfe
Neues Unterfenster Ansicht teilen Kopieren Einfügen Suchen
root@www001:/home/david/doc06232026debian-pc-i5/asm06232026#

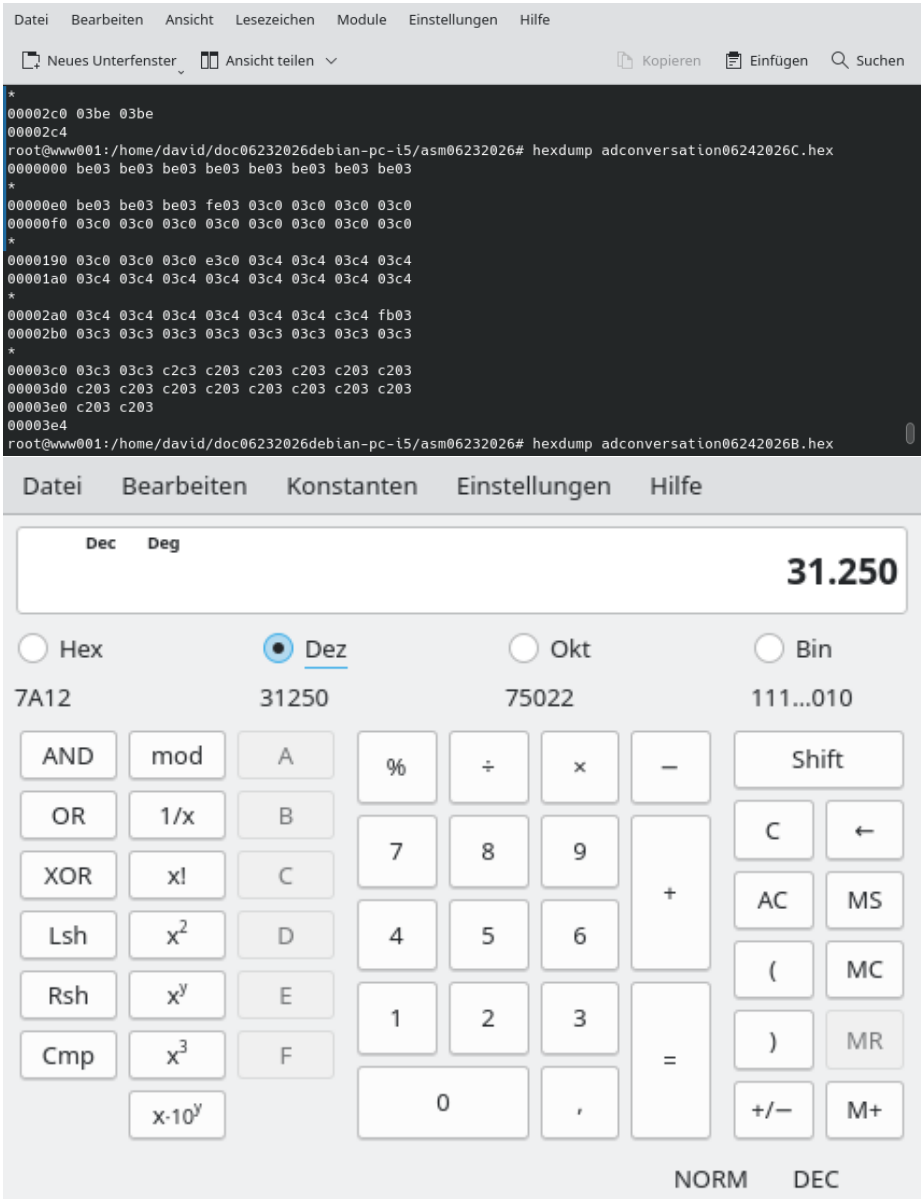

Datei Bearbeiten Ansicht Lesezeichen Module Einstellungen Hilfe
Neues Unterfenster
Ansicht teilen
Kopieren
Einfügen
Suchen
david@https://www.vajda-breadboard.de
root@www001:/home/david/doc06232026debian-pc-i5/asm06232026#
Datei Bearbeiten Ansicht Lesezeichen Module Einstellungen Hilfe
Neues Unterfenster
Ansicht teilen
Kopieren
Einfügen
Suchen
david@https://www.vajda-breadboard.de
Datei Bearbeiten Ansicht Lesezeichen Module Einstellungen Hilfe
Neues Unterfenster
Ansicht teilen
Kopieren
Einfügen
Suchen
david@https://www.vajda-breadboard.de
root@www001:/home/david/doc06232026debian-pc-i5/asm06232026# stty -F /dev/ttyS3 2400 cs8 raw parodd cstopb && cat /dev/ttyS3
:tps://www.vajda-breadboard.de

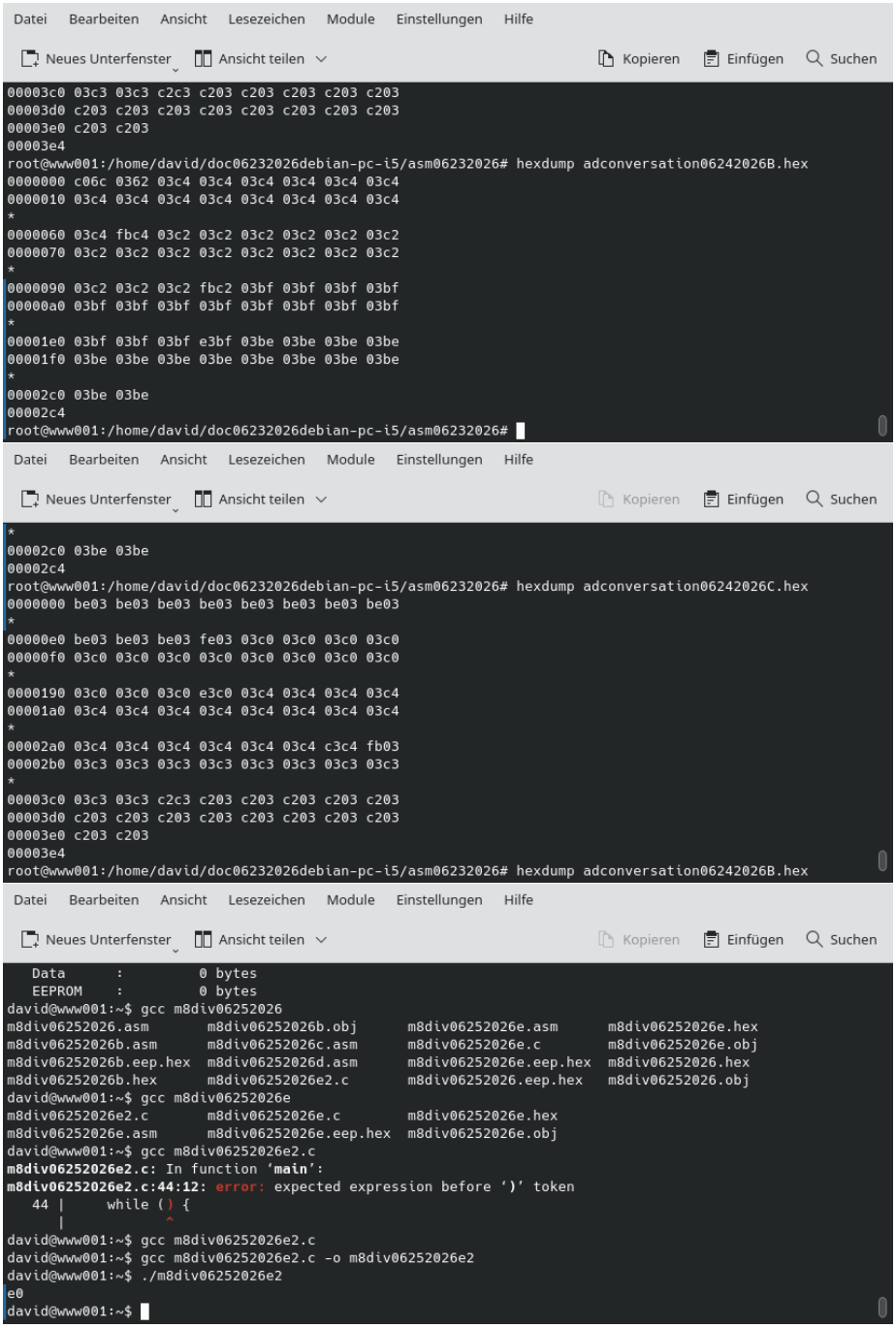

Datei Bearbeiten Ansicht Lesezeichen Module Einstellungen Hilfe
Neues Unterfenster
Ansicht teilen
Kopieren
Einfügen
Suchen
^C
root@www001:/home/david/doc06232026debian-pc-i5/asm06232026# stty -F /dev/ttyS3 2400 cs8 raw parodd cstopb && cat /dev/ttyS3
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de


Datei Bearbeiten Ansicht Lesezeichen Module Einstellungen Hilfe
Neues Unterfenster
Ansicht teilen
Kopieren
Einfügen
Suchen
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
root@www001:/home/david/doc06232026debian-pc-i5/asm06232026# avrdude -c stk500 -p m8 -P /dev/ttyS3 -U flash:w:m806222026ad5.hex:i


Datei Bearbeiten Ansicht Lesezeichen Module Einstellungen Hilfe
Neues Unterfenster
Ansicht teilen
Kopieren
Einfügen
Suchen
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
root@www001:/home/david/doc06232026debian-pc-i5/asm06232026#


Datei Bearbeiten Ansicht Lesezeichen Module Einstellungen Hilfe
Neues Unterfenster
Ansicht teilen
Kopieren
Einfügen
Suchen
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@h,L,L,L


Datei Bearbeiten Ansicht Lesezeichen Module Einstellungen Hilfe
Neues Unterfenster
Ansicht teilen
Kopieren
Einfügen
Suchen
*
00002c0 03be 03be
00002c4
root@www001:/home/david/doc06232026debian-pc-i5/asm06232026# hexdump adconversation06242026C.hex
0000000 be03 be03 be03 be03 be03 be03 be03 be03
*
00000e0 be03 be03 be03 fe03 03c0 03c0 03c0 03c0
00000f0 03c0 03c0 03c0 03c0 03c0 03c0 03c0 03c0
*
0000190 03c0 03c0 03c0 e3c0 03c4 03c4 03c4 03c4
00001a0 03c4 03c4 03c4 03c4 03c4 03c4 03c4 03c4
*
00002a0 03c4 03c4 03c4 03c4 03c4 03c4 c3c4 fb03
00002b0 03c3 03c3 03c3 03c3 03c3 03c3 03c3 03c3
*
00003c0 03c3 03c3 c2c3 c203 c203 c203 c203 c203
00003d0 c203 c203 c203 c203 c203 c203 c203 c203
00003e0 c203 c203
00003e4
root@www001:/home/david/doc06232026debian-pc-i5/asm06232026# hexdump adconversation06242026B.hex
Datei Bearbeiten Konstanten Einstellungen Hilfe
Dec Deg
31.250
Hex
Dez
Okt
Bin
7A12
31250
75022
111...010
AND
mod
A
%
÷
×
–
Shift
OR
1/x
B
C
←
XOR
x!
C
7
8
9
+
AC
MS
Lsh
x2
D
4
5
6
(
MC
Rsh
xy
E
1
2
3
=
)
MR
Cmp
x3
F
0
,
+/–
M+
x·10y
NORM
DEC




```
;; das muss raus

;;add r16, r17
```

ach so, addieren und dann die haelfte!!!

computersysteme 1/2 und besonders joachim rhode - assembler programmierung

was eigentlich ist mmx? simd heisst: single instruction multiple data

mmx - multimedia extension

gepackt und ungepackt: heisst: BCD z.B.

99dec koennte man unterbringen in gepackt

`1001.1001b` in einem byte

oder
`0000.1001.0000.1001` ungepackt in zwei byte

gut, das macht das noch nicht zu simd und simd - na ja

dass es keinen ueberlauf ins naechste gibt, ist das eine.

aber es gibt noch etwas:

swap-arround, saturation...

einen ueberlauf produzieren ist das eine, ...

aber es sollte auch nicht in jedem falle im eigenen rigister wieder von vorne los gehen

`0xff + 0x01 = 0x00` mit oder ohne ueberlauf

dann waeren wir beim unteren wert...

folgendes: ich lasse die addition und nehme den wichtigeren wert...

```
;; bei mir:

    in r17, ADCH
    in r16, ADCL

;; da steht:

    in      adlow, ADCL         ; immer zuerst das Low-Byte lesen
    in      adhigh, ADCH        ; danach das mittlerweile gesperrte High-Byte
```

hat jetzt nichts gebracht: naechster schritt: kabel fehler - falscher pin - oder beim fuellen des puffers fehler - additions fehler - falsch gedacht???

ja, genau, wie ich sagte, da steht es ja...

```
wait_adc:
    sbic    ADCSRA, ADSC        ; wenn der ADC fertig ist, wird dieses Bit gel"oscht
    rjmp    wait_adc
```

also noch mal, dann tut es sicherlich!!!

```
sbi     ADCSRA, ADSC        ; den ADC starten
```

er wurde nicht gestartet... daneben, was heisst hier einfach eine conversation aufnehmen, sorry, war bisher richtig, dass das geruest stimmt

blos: 50kHz. bei der conversation - das muss man ja auch alles lernen, ist halt 50kHz, beim durchlauf der CPU, gut, wenn ich 1 MHz habe, schoen, wenn ich dann natuerlich in einer schleife bei der abtastung, 5 befehle, etwa 5 takte macht natuerlich so pi mal daumen, 200kHz, das ist die hoehere abtastrate, was moeglich waere, trotzdem...

genau wie bei rs232 muss ja gegeben sein

```
sbis UCSRA, UDRE
```

das heisst, ich muss immer warten, bis fertig / bereit / moeglich

oder unterbrechnung...

aber es war nicht aktiviert...

ja, genau, wie ich sagte, da steht es ja...

```
wait_adc:
    sbic    ADCSRA, ADSC        ; wenn der ADC fertig ist, wird dieses Bit gel"oscht
    rjmp    wait_adc
```

also noch mal, dann tut es sicherlich!!!

die uebertragung von `'L'` klappt schon mal, es zeigt sich allerdings genau jetzt, an der stelle, wo an `PC1 VCC` angelegt ist, aber auch wenn an PC1 GND angelegt ist, moeglicherweise ist auch das kabel nicht in ordnung, erscheint immer `'L'`

1. kabel austauschen
2. kontrollieren, .. tut die speicherung an sich im puffer der werte bei der conversation

3. ADC falsch gemacht, z.B. `add r16, r17` ist der mittelwert nicht ok?

```
;; r18 doppelt genommen

    ldi YH, HIGH (buffer)
    ldi YL, LOW (buffer)
    ldi XH, HIGH (cbuffer)
    ldi XL, LOW (cbuffer)
    ldi r22, 0xff
    label001:
    ld r16, Y+
    ld r17, Y+
    add r16, r17
    cpi r16, 0xff >> 2 ; 0xff / 2 = 0xff >> 2 (mittlerer wert)
    brlt labellower001
    labelhigher001:
    ldi r18, 'H'
    ldi r19, ','
    rjmp labelhigherorlowerend001
    labellower001:
    ldi r18, 'L'
    ldi r19, ','
    labelhigherorlowerend001:
    st X+, r18
    st X+, r19
    dec r22
    brne label001



   ;; so zum beispiel

    .dseg
    buffer: .byte 512
    cbuffer: .byte 512
    .cseg
;; ...
    ldi YH, HIGH (buffer)
    ldi YL, LOW (buffer)
    ldi XH, HIGH (cbuffer)
    ldi XL, LOW (cbuffer)
    ldi r18, 0xff
    label001:
    ld r16, Y+
    ld r17, Y+
    add r16, r17
    cpi r16, 0xff >> 2 ; 0xff / 2 = 0xff >> 2 (mittlerer wert)
    brle labellower001:
    labelhigher001:
    ldi r18, 'H'
```

```
    ldi r19, ','
    rjmp labelhigherorlowerend001
    labellower001:
    ldi r18, 'L'
    ldi r19, ','
    labelhigherorlowerend001:
    st X+, r18
    st X+, r19
    dec r18
    brne label001
```

so, jetzt mache ich weiter...

indem ich den quelltext noch mal nehme, schon mal, ohne die werte ernsthaft zu kennen, das lasse ich offen, zu unterscheiden,

oberer Wert: 'H' unterer Wert 'L'. und entsprechend findet die uebertragung an den PC statt

erst so zu sagen, vorab das template, mit bedingten spruengen und dann nachlesen, datasheet ergaenzen...

```
c;; das hier wiederum, die uebertragung von C  und D, an dieser stelle, wo der quelltext e

alles, was darueber hinausgeht, ist nicht mehr teil der fragestellung ...

ok, nachdem das nun eindeutig klar ist - ergibt sich - RESET druecken und: trotz allem, mo

0V bzw. 5V ergeben welche - werte ..

und wenn das geschehen ist. vor allem darauf achten, an PC1 findet hier jeweils die abtast

das wiederholen, dann zweiter controller, entsprechend frequenz erzeugen..


;; david vajda
    ;; 06/22/2026
    ;; m8 ad converter

    .include "m8def.inc"

    .dseg
    buffer: .byte 512
    .cseg

    ldi r16, HIGH (RAMEND)
    out SPH, r16
    ldi r16, LOW (RAMEND)
    out SPL, r16

    ldi r16, HIGH (25)
```

```
out UBRRH, r16
ldi r16, LOW (25)
out UBRRL, r16

;; ldi r16, (1 << URSEL) | (1 << UCSZ2) | (1 << USBS)
ldi r16, (1 << URSEL) | (1 << USBS)
out UCSRC, r16
ldi r16, (1 << TXEN) | (1 << UCSZ1) | (1 << UCSZ0) | (1 << UPM1) | (1 << UPM0)
out UCSRB, r16

ldi r16, (1 << REFS0) | (1 << MUX0)
out ADMUX, r16

ldi r16, (1 << ADEN) | (1 << ADSC) | (1 << ADPS2)
;; ADPS2: Prescaler (Samplerate): 16, 1 MHz/16 = 250kHz/4 ~= 51kHz??
;; 50 kHz <= Samplerate <= 200 kHz
out ADCSRA, r16

ldi r16, HIGH (buffer)
mov YH, r16
ldi r16, LOW (buffer)
mov YL, r16

ldi r18, 0xff
conversation001:
in r17, ADCH
in r16, ADCL
st Y+, r17
st Y+, r16
dec r18
brne conversation001

ldi YH, HIGH (buffer)
ldi YL, LOW (buffer)
ldi r18, 0xff
label001:
ldi r16, 'C'
ldi r17, 'D'
st Y+, r16
st Y+, r17
dec r18
brne label001


ldi r16, HIGH (buffer)
mov YH, r16
ldi r16, LOW (buffer)
mov YL, r16
ldi r18, 0xff
transmit001:
```

```
 ld r16, Y+
 rcall rs232transmitch
 ld r16, Y+
 rcall rs232transmitch
 dec r18
 brne transmit001



label000:
 rcall rs232transmitstr
 rjmp label000

 rs232transmitstr:
 ldi r16, HIGH (2*msg000)
 mov ZH, r16
 ldi r16, LOW (2*msg000)
 mov ZL, r16

 rs232transmitstr1:
 lpm r16, Z+
 cpi r16, 0x00
 breq rs232transmitstrend
 rcall rs232transmitch
 rjmp rs232transmitstr1
 rs232transmitstrend:
 ret

 rs232transmitch:
 sbis UCSRA, UDRE
 rjmp rs232transmitch
 out UDR, r16
 ret

 msg000: .db "david@https://www.vajda-breadboard.de", 10, 13, 0


 ja und nun, herr kaufmann - keine erklaerung  dafuer...? RESET gedrueckt!! und hinreic

 ;; david vajda
 ;; 06/22/2026
 ;; m8 ad converter

 .include "m8def.inc"

 .dseg
 buffer: .byte 512
 .cseg

 ldi r16, HIGH (RAMEND)
```

```
out SPH, r16
ldi r16, LOW (RAMEND)
out SPL, r16

ldi r16, HIGH (25)
out UBRRH, r16
ldi r16, LOW (25)
out UBRRL, r16

;; ldi r16, (1 << URSEL) | (1 << UCSZ2) | (1 << USBS)
ldi r16, (1 << URSEL) | (1 << USBS)
out UCSRC, r16
ldi r16, (1 << TXEN) | (1 << UCSZ1) | (1 << UCSZ0) | (1 << UPM1) | (1 << UPM0)
out UCSRB, r16

ldi r16, (1 << REFS0) | (1 << MUX0)
out ADMUX, r16

ldi r16, (1 << ADEN) | (1 << ADSC) | (1 << ADPS2)
;; ADPS2: Prescaler (Samplerate): 16, 1 MHz/16 = 250kHz/4 ~= 51kHz??
;; 50 kHz <= Samplerate <= 200 kHz
out ADCSRA, r16

ldi r16, HIGH (buffer)
mov YH, r16
ldi r16, LOW (buffer)
mov YL, r16

ldi r18, 0xff
conversation001:
in r17, ADCH
in r16, ADCL
st Y+, r17
st Y+, r16
dec r18
brne conversation001

ldi YH, HIGH (buffer)
ldi YL, LOW (buffer)
ldi r18, 0xff
label001:
ldi r16, 'A'
ldi r17, 'B'
st Y+, r16
st Y+, r17
dec r18
brne label001


ldi r16, HIGH (buffer)
```

```
    mov YH, r16
    ldi r16, LOW (buffer)
    mov YL, r16
    ldi r18, 0xff
    transmit001:
    ld r16, Y+
    rcall rs232transmitch
    ld r16, Y+
    rcall rs232transmitch
    dec r18
    brne transmit001



   label000:
    rcall rs232transmitstr
    rjmp label000

    rs232transmitstr:
    ldi r16, HIGH (2*msg000)
    mov ZH, r16
    ldi r16, LOW (2*msg000)
    mov ZL, r16

    rs232transmitstr1:
    lpm r16, Z+
    cpi r16, 0x00
    breq rs232transmitstrend
    rcall rs232transmitch
    rjmp rs232transmitstr1
    rs232transmitstrend:
    ret

    rs232transmitch:
    sbis UCSRA, UDRE
    rjmp rs232transmitch
    out UDR, r16
    ret

    msg000: .db "david@https://www.vajda-breadboard.de", 10, 13, 0



  ;;; ach doch, da haben wir den salat - aber das ist jetzt noch laengst nicht alles - hab
```

Also

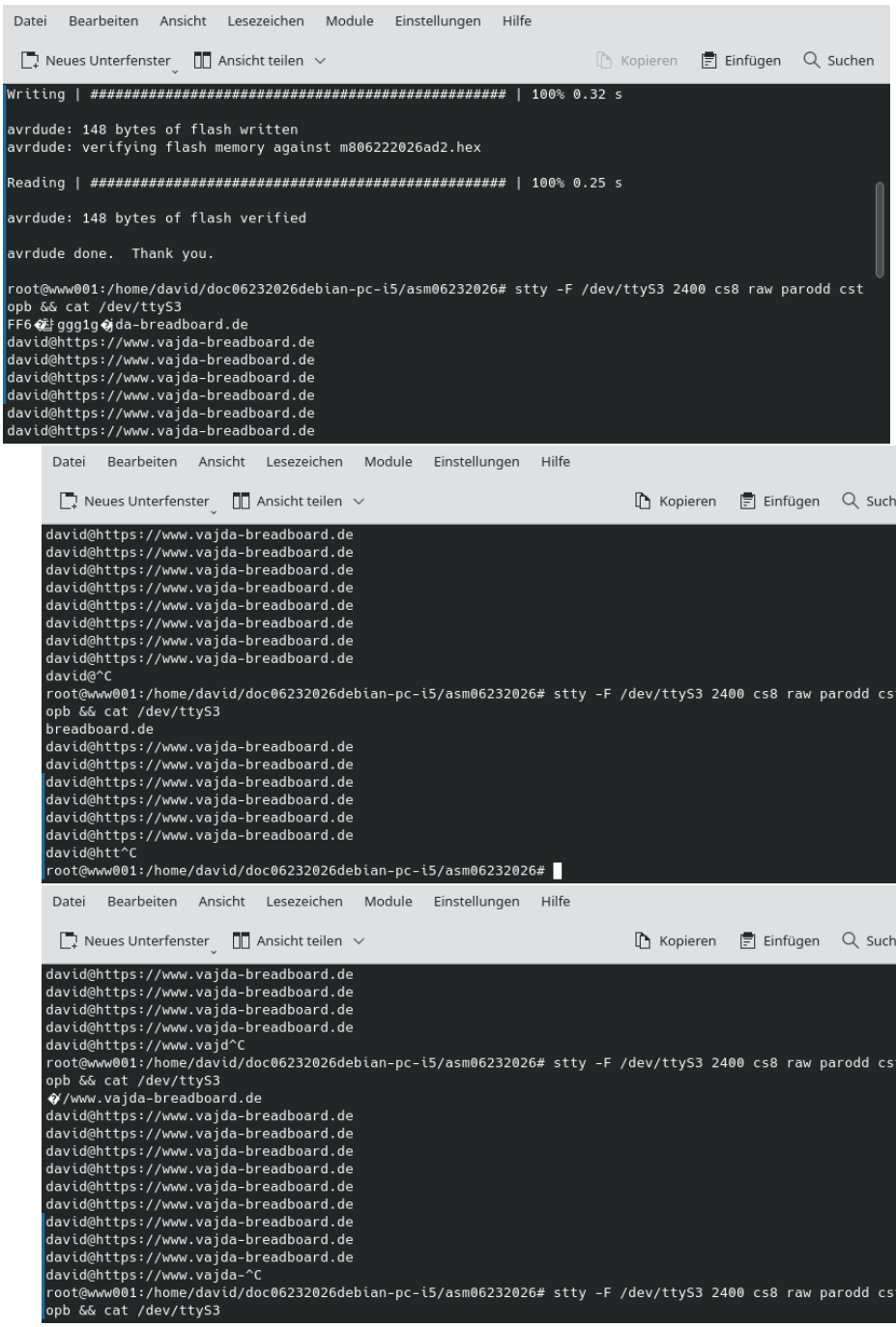
Notwendig aber Hinreichend aber im nicht

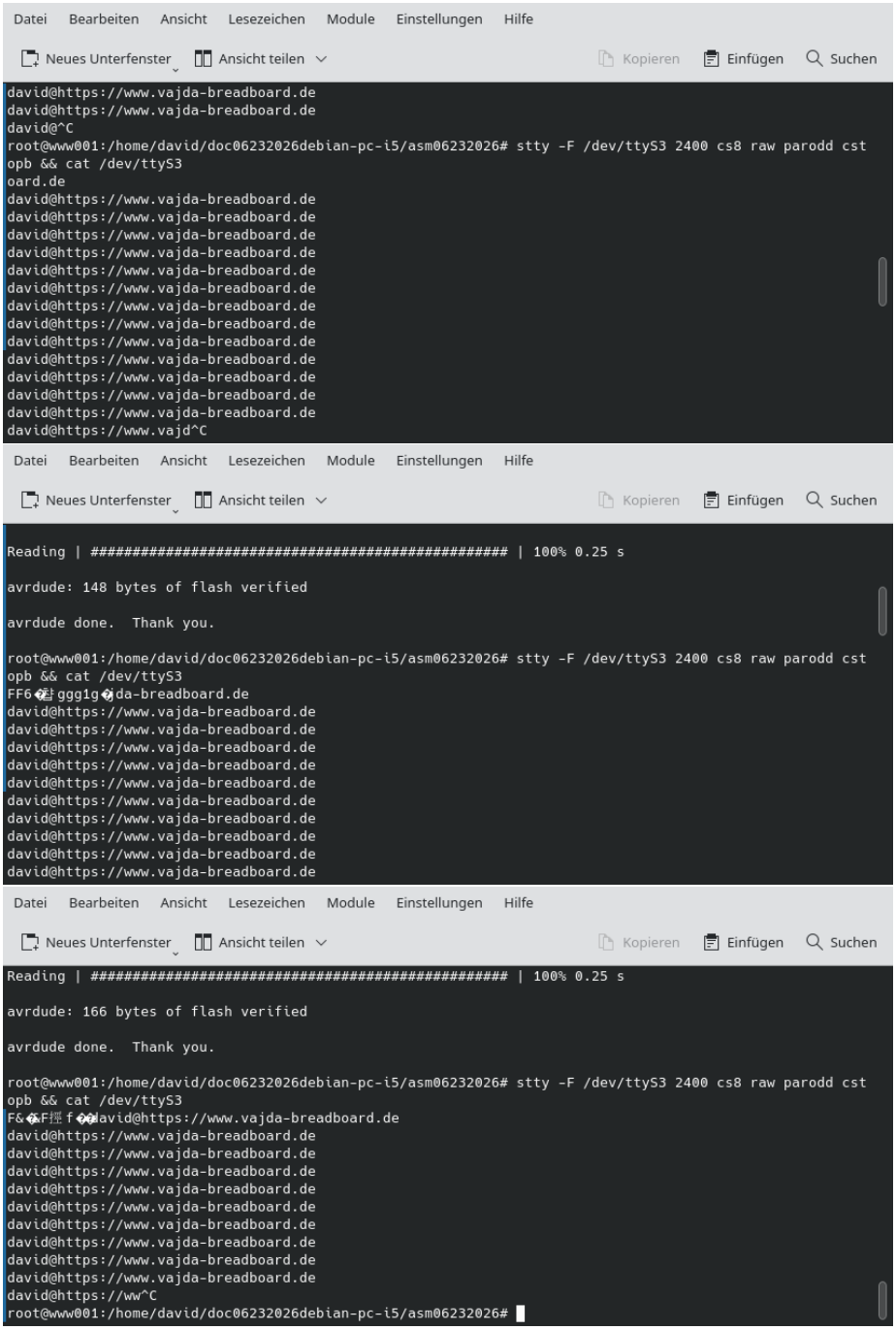
es passierte naemlich das gleiche, nur ohne puffer. also, was mit puffer passierte, passie

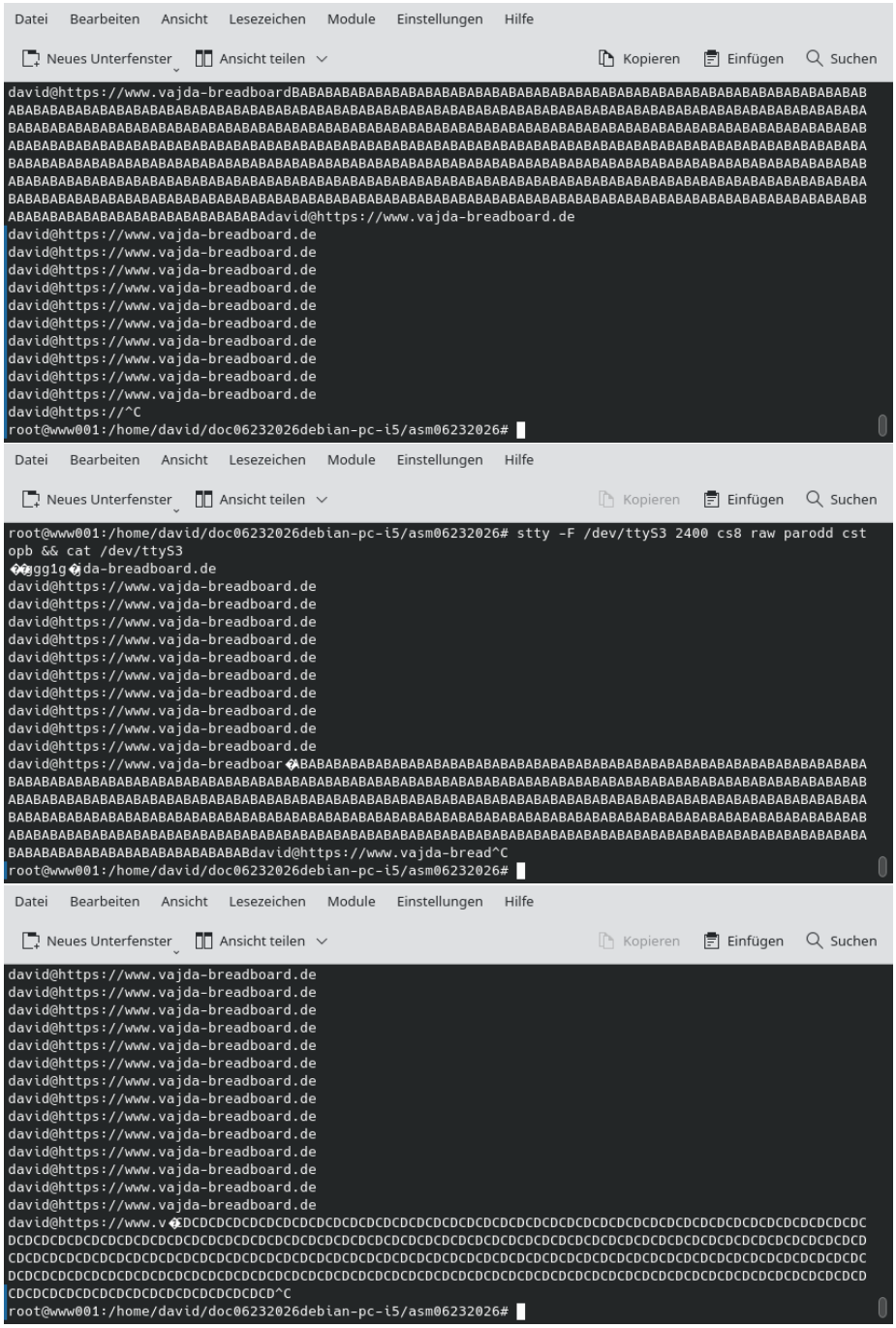
```
;; david vajda
;; 06/22/2026
;; m8 ad converter

.include "m8def.inc"

.dseg
buffer: .byte 512
.cseg

ldi r16, HIGH (RAMEND)
out SPH, r16
ldi r16, LOW (RAMEND)
out SPL, r16

ldi r16, HIGH (25)
out UBRRH, r16
ldi r16, LOW (25)
out UBRRL, r16

;; ldi r16, (1 << URSEL) | (1 << UCSZ2) | (1 << USBS)
ldi r16, (1 << URSEL) | (1 << USBS)
out UCSRC, r16
ldi r16, (1 << TXEN) | (1 << UCSZ1) | (1 << UCSZ0) | (1 << UPM1) | (1 << UPM0)
out UCSRB, r16

ldi r16, (1 << REFS0) | (1 << MUX0)
out ADMUX, r16

ldi r16, (1 << ADEN) | (1 << ADSC) | (1 << ADPS2)
;; ADPS2: Prescaler (Samplerate): 16, 1 MHz/16 = 250kHz/4 ~= 51kHz??
;; 50 kHz <= Samplerate <= 200 kHz
out ADCSRA, r16

ldi r16, HIGH (buffer)
mov YH, r16
ldi r16, LOW (buffer)
mov YL, r16

ldi r18, 0xff
conversation001:
in r17, ADCH
in r16, ADCL
st Y+, r17
st Y+, r16
dec r18
brne conversation001

ldi YH, HIGH (buffer)
```

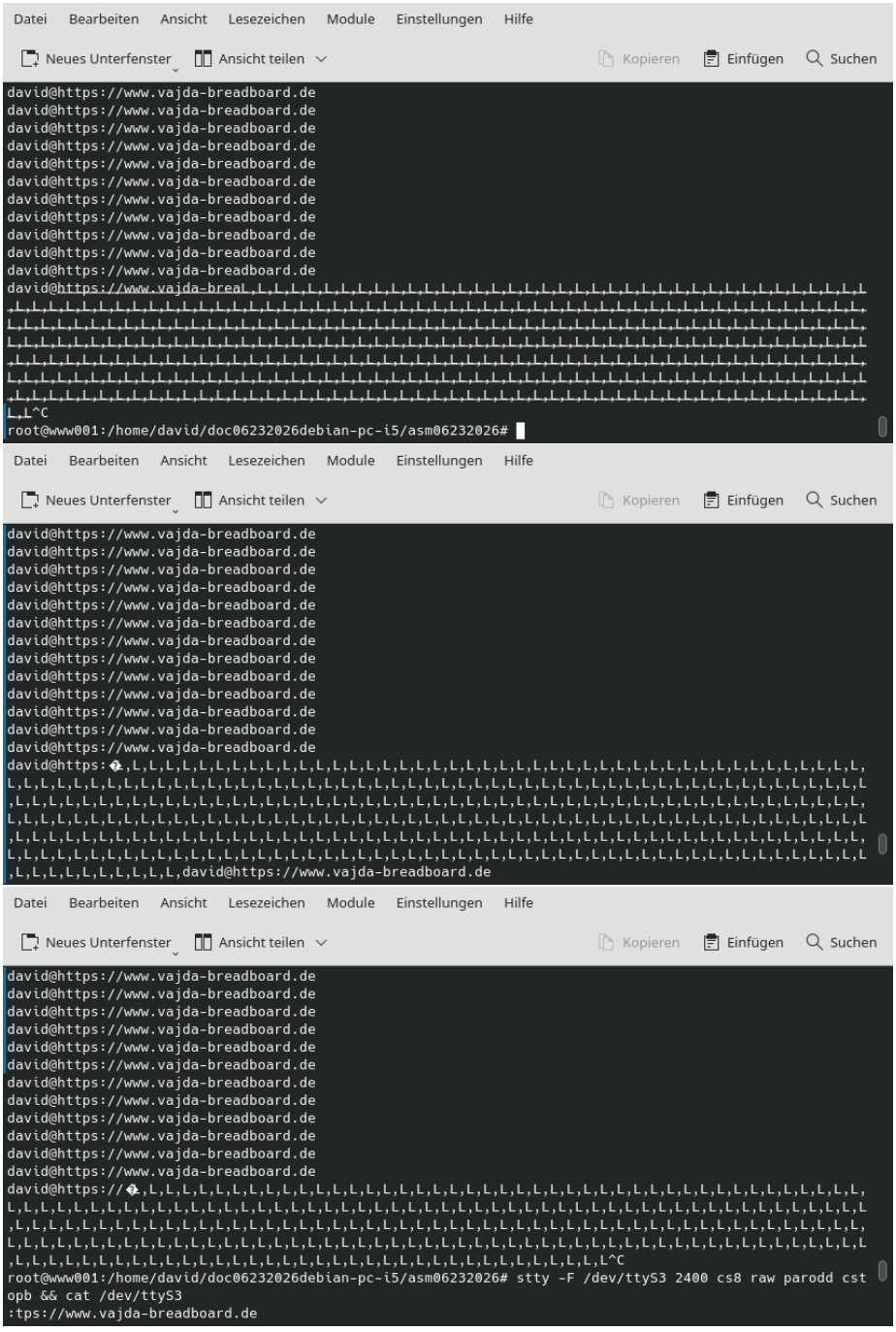
```
 ldi YL, LOW (buffer)
 ldi r18, 0xff
 label001:
 ldi r16, 'A'
 ldi r17, 'B'
 st Y+, r16
 st Y+, r17
 dec r18
 brne label001


 ldi r16, HIGH (buffer)
 mov YH, r16
 ldi r16, LOW (buffer)
 mov YL, r16
 ldi r18, 0xff
 transmit001:
 ld r16, Y+
 ldi r16, 'B'
 rcall rs232transmitch
 ld r16, Y+
 ldi r16, 'A'
 rcall rs232transmitch
 dec r18
 brne transmit001



label000:
 rcall rs232transmitstr
 rjmp label000

 rs232transmitstr:
 ldi r16, HIGH (2*msg000)
 mov ZH, r16
 ldi r16, LOW (2*msg000)
 mov ZL, r16

 rs232transmitstr1:
 lpm r16, Z+
 cpi r16, 0x00
 breq rs232transmitstrend
 rcall rs232transmitch
 rjmp rs232transmitstr1
 rs232transmitstrend:
 ret

 rs232transmitch:
 sbis UCSRA, UDRE
 rjmp rs232transmitch
```

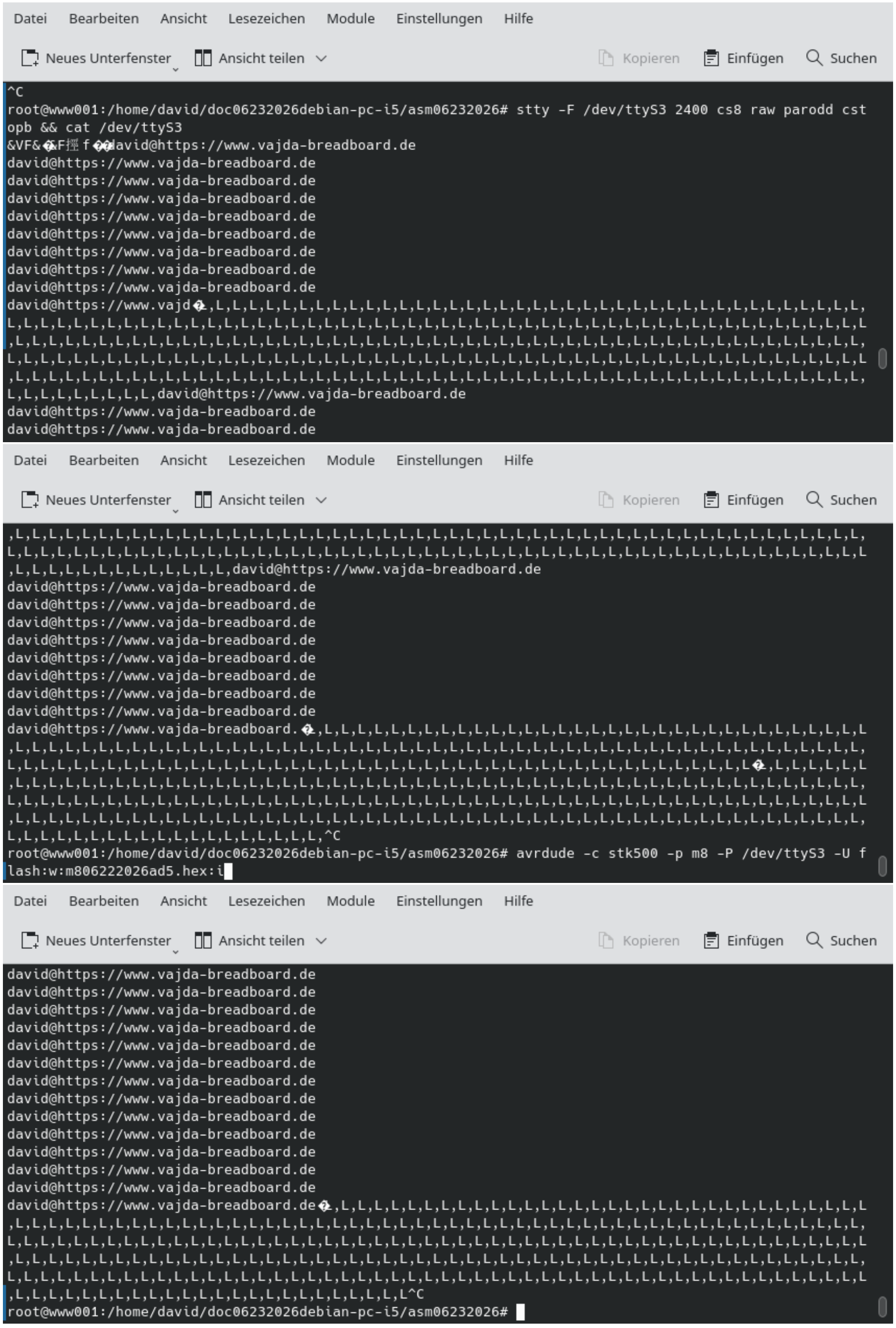
```
out UDR, r16
ret

msg000: .db "david@https://www.vajda-breadboard.de", 10, 13, 0
```

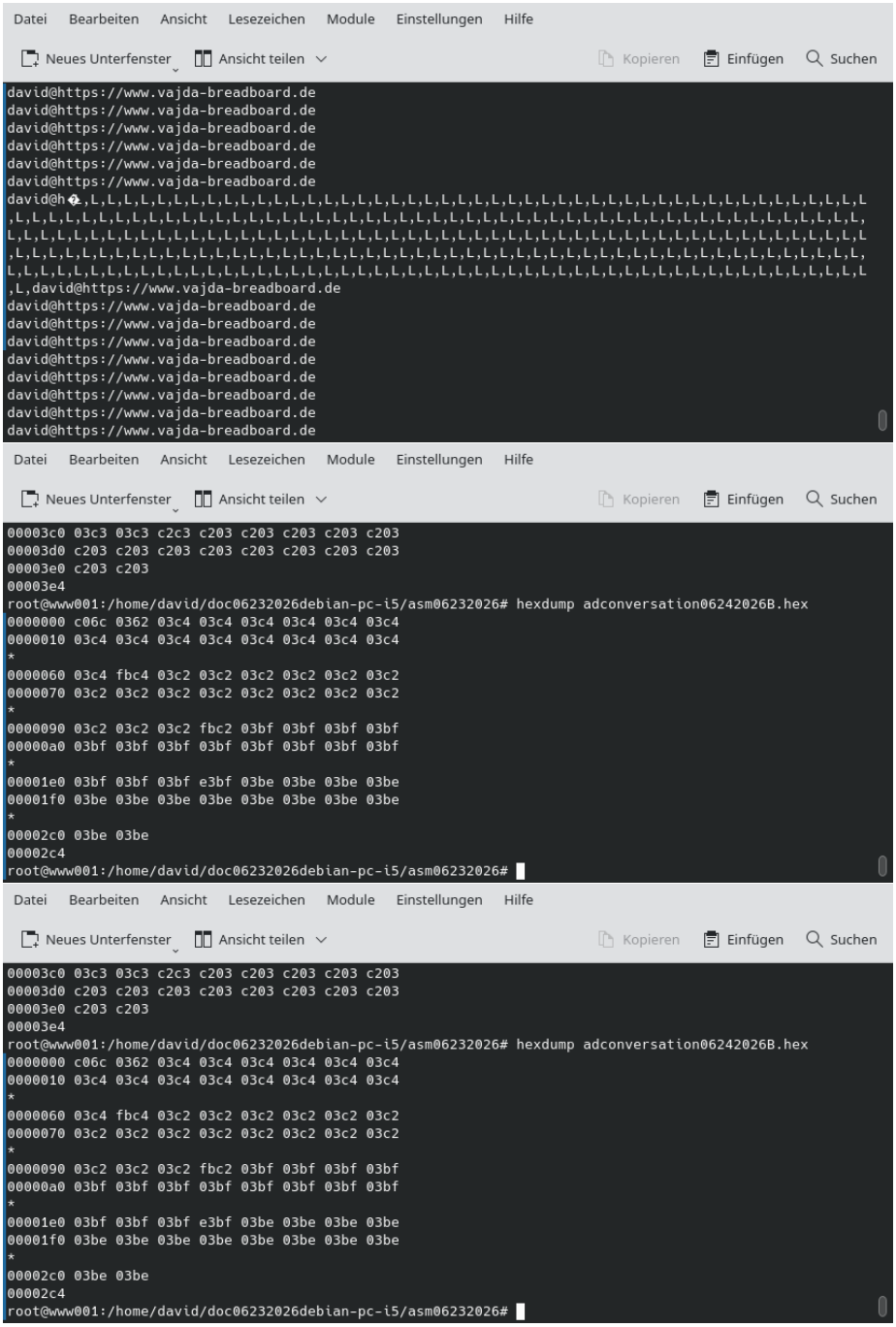


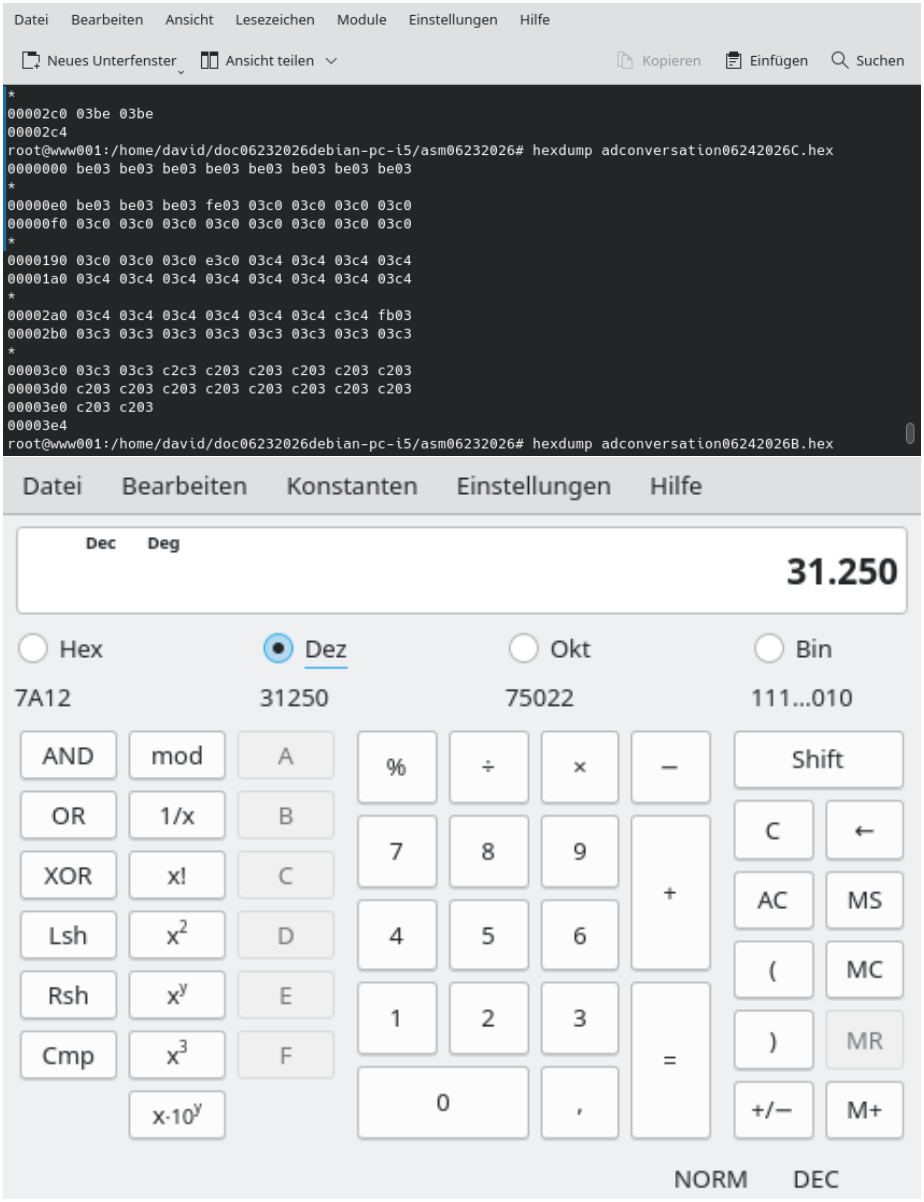

Datei Bearbeiten Ansicht Lesezeichen Module Einstellungen Hilfe
Neues Unterfenster
Ansicht teilen
Kopieren
Einfügen
Suchen
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@^C
root@www001:/home/david/doc06232026debian-pc-i5/asm06232026# stty -F /dev/ttyS3 2400 cs8 raw parodd cst
opb && cat /dev/ttyS3
oard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajd^C
Datei Bearbeiten Ansicht Lesezeichen Module Einstellungen Hilfe
Neues Unterfenster
Ansicht teilen
Kopieren
Einfügen
Suchen
Reading | ## | 100% 0.25 s
avrdude: 148 bytes of flash verified
avrdude done. Thank you.
root@www001:/home/david/doc06232026debian-pc-i5/asm06232026# stty -F /dev/ttyS3 2400 cs8 raw parodd cst
opb && cat /dev/ttyS3
FF6 ggg1g da-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
Datei Bearbeiten Ansicht Lesezeichen Module Einstellungen Hilfe
Neues Unterfenster
Ansicht teilen
Kopieren
Einfügen
Suchen
Reading | ## | 100% 0.25 s
avrdude: 166 bytes of flash verified
avrdude done. Thank you.
root@www001:/home/david/doc06232026debian-pc-i5/asm06232026# stty -F /dev/ttyS3 2400 cs8 raw parodd cst
opb && cat /dev/ttyS3
F& F f avid@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://ww^C
root@www001:/home/david/doc06232026debian-pc-i5/asm06232026#

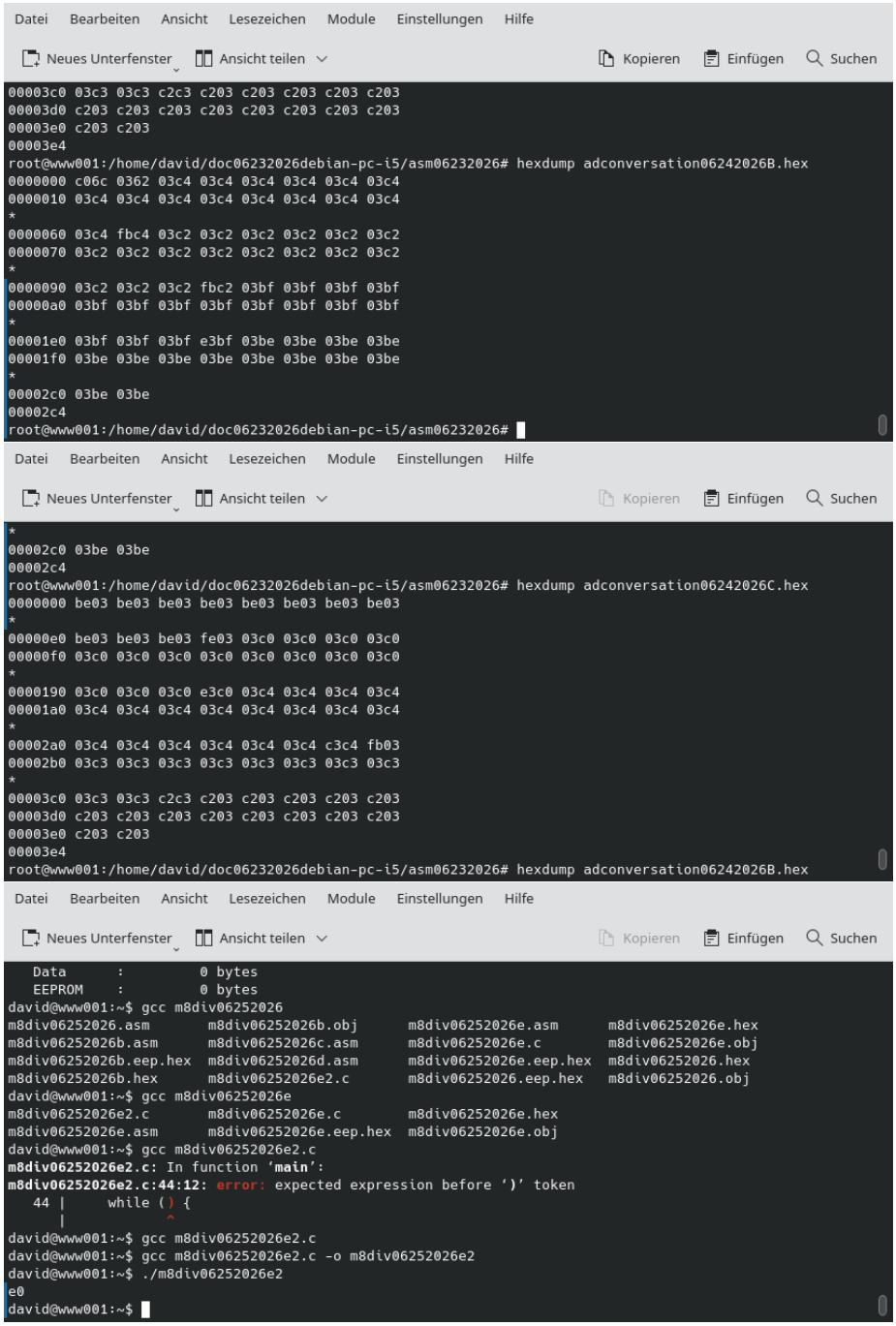

Datei Bearbeiten Ansicht Lesezeichen Module Einstellungen Hilfe
Neues Unterfenster Ansicht teilen Kopieren Einfügen Suchen
david@https://www.vajda-breadboardBABABAB


Datei Bearbeiten Ansicht Lesezeichen Module Einstellungen Hilfe
Neues Unterfenster
Ansicht teilen
Kopieren
Einfügen
Suchen
david@https://www.vajda-breadboard.de
root@www001:/home/david/doc06232026debian-pc-i5/asm06232026#
Datei Bearbeiten Ansicht Lesezeichen Module Einstellungen Hilfe
Neues Unterfenster
Ansicht teilen
Kopieren
Einfügen
Suchen
david@https://www.vajda-breadboard.de
Datei Bearbeiten Ansicht Lesezeichen Module Einstellungen Hilfe
Neues Unterfenster
Ansicht teilen
Kopieren
Einfügen
Suchen
david@https://www.vajda-breadboard.de
root@www001:/home/david/doc06232026debian-pc-i5/asm06232026# stty -F /dev/ttyS3 2400 cs8 raw parodd cstopb && cat /dev/ttyS3
:tps://www.vajda-breadboard.de


Datei Bearbeiten Ansicht Lesezeichen Module Einstellungen Hilfe
Neues Unterfenster Ansicht teilen Kopieren Einfügen Suchen
^C
root@www001:/home/david/doc06232026debian-pc-i5/asm06232026# stty -F /dev/ttyS3 2400 cs8 raw parodd cstopb && cat /dev/ttyS3
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
root@www001:/home/david/doc06232026debian-pc-i5/asm06232026# avrdude -c stk500 -p m8 -P /dev/ttyS3 -U flash:w:m806222026ad5.hex:i
root@www001:/home/david/doc06232026debian-pc-i5/asm06232026#


Datei Bearbeiten Ansicht Lesezeichen Module Einstellungen Hilfe
Neues Unterfenster
Ansicht teilen
Kopieren
Einfügen
Suchen
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@https://www.vajda-breadboard.de
david@h,L,L,L


Datei Bearbeiten Ansicht Lesezeichen Module Einstellungen Hilfe
Neues Unterfenster
Ansicht teilen
Kopieren
Einfügen
Suchen
*
00002c0 03be 03be
00002c4
root@www001:/home/david/doc06232026debian-pc-i5/asm06232026# hexdump adconversation06242026C.hex
0000000 be03 be03 be03 be03 be03 be03 be03 be03
*
00000e0 be03 be03 be03 fe03 03c0 03c0 03c0 03c0
00000f0 03c0 03c0 03c0 03c0 03c0 03c0 03c0 03c0
*
0000190 03c0 03c0 03c0 e3c0 03c4 03c4 03c4 03c4
00001a0 03c4 03c4 03c4 03c4 03c4 03c4 03c4 03c4
*
00002a0 03c4 03c4 03c4 03c4 03c4 03c4 c3c4 fb03
00002b0 03c3 03c3 03c3 03c3 03c3 03c3 03c3 03c3
*
00003c0 03c3 03c3 c2c3 c203 c203 c203 c203 c203
00003d0 c203 c203 c203 c203 c203 c203 c203 c203
00003e0 c203 c203
00003e4
root@www001:/home/david/doc06232026debian-pc-i5/asm06232026# hexdump adconversation06242026B.hex
Datei Bearbeiten Konstanten Einstellungen Hilfe
Dec Deg
31.250
Hex
Dez
Okt
Bin
7A12
31250
75022
111...010
AND mod A
OR 1/x B
XOR x! C
Lsh x2 D
Rsh xy E
Cmp x3 F
x·10y
% ÷ × −
7 8 9 +
4 5 6
1 2 3 =
0 ,
Shift
C ←
AC MS
(MC
) MR
+/− M+
NORM DEC




```
;; hier sehen wir, auf jeden fall hat dieser quelltext nicht funktioniert (code)... und: i

    ;; david vajda
    ;; 06/22/2026
    ;; m8 ad converter
```

```
.include "m8def.inc"

.dseg
buffer: .byte 512
.cseg

ldi r16, HIGH (RAMEND)
out SPH, r16
ldi r16, LOW (RAMEND)
out SPL, r16

ldi r16, HIGH (25)
out UBRRH, r16
ldi r16, LOW (25)
out UBRRL, r16

;; ldi r16, (1 << URSEL) | (1 << UCSZ2) | (1 << USBS)
ldi r16, (1 << URSEL) | (1 << USBS)
out UCSRC, r16
ldi r16, (1 << TXEN) | (1 << UCSZ1) | (1 << UCSZ0) | (1 << UPM1) | (1 << UPM0)
out UCSRB, r16

ldi r16, (1 << REFS0) | (1 << MUX0)
out ADMUX, r16

ldi r16, (1 << ADEN) | (1 << ADSC) | (1 << ADPS2)
;; ADPS2: Prescaler (Samplerate): 16, 1 MHz/16 = 250kHz/4 ~= 51kHz??
;; 50 kHz <= Samplerate <= 200 kHz
out ADCSRA, r16

ldi r16, HIGH (buffer)
mov YH, r16
ldi r16, LOW (buffer)
mov YL, r16

ldi r18, 0xff
conversation001:
in r17, ADCH
in r16, ADCL
st Y+, r17
st Y+, r16
dec r18
brne conversation001

ldi YH, HIGH (buffer)
ldi YL, LOW (buffer)
ldi r18, 0xff
label001:
ldi r16, 'A'
ldi r17, 'B'
```

```
 st Y+, r16
 st Y+, r17
 dec r18
 brne label001


 ldi r16, HIGH (buffer)
 mov YH, r16
 ldi r16, LOW (buffer)
 mov YL, r16
 ldi r18, 0xff
 transmit001:
 ld r16, Y+
 rcall rs232transmitch
 ld r16, Y+
 rcall rs232transmitch
 dec r18
 brne transmit001



label000:
 rcall rs232transmitstr
 rjmp label000

 rs232transmitstr:
 ldi r16, HIGH (2*msg000)
 mov ZH, r16
 ldi r16, LOW (2*msg000)
 mov ZL, r16

 rs232transmitstr1:
 lpm r16, Z+
 cpi r16, 0x00
 breq rs232transmitstrend
 rcall rs232transmitch
 rjmp rs232transmitstr1
 rs232transmitstrend:
 ret

 rs232transmitch:
 sbis UCSRA, UDRE
 rjmp rs232transmitch
 out UDR, r16
 ret

 msg000: .db "david@https://www.vajda-breadboard.de", 10, 13, 0
```

```
    ; ... zum beispiel ergaenzt um das ...
; ...

    ldi YH, HIGH (buffer)
    ldi YL, LOW (buffer)
    ldi r18, 0xff
    label001:
    ldi r16, "A"
    ldi r17, "B"
    st Y+, r16
    st Y+, r17
    dec r18
    brne label001

; ...
```

so, jetzt bin ich so weit...

ich kann glaube ich jetzt nicht erkennen, ob der erste teil der uebertragung was mit unserem AD-Wandler AD-Converter ADC zu tun hat...

ich werde jetzt folgendes tun. zunaechst, bringe ich ohne den controller zu benutzen den quelltext (*.s), (*.asm) assembler code - mnemonics menschlich gesprochene maschinenbefehle.

moeglich waere als assembler - zwischensprache moeglich

- russische worte: fuer laden, speichern...

nachgucken, abkuerzung waehlen, kyrillisch - 1:1 in atmega8 assembler - english

... ohne erklaerung ... aber: das wort quelltext usw. das ... sind meist bezogen auf hochsprachen, das heisst, imperative programmierung und - objektorientierung

andere beispiele sind: hardware beschreibungssprachen - HL - VHDL, ABEL... daneben: protkolle: SMTP, HTTP (nicht zu verwechseln mit HTML)...

ok. und vieles weitere.

auch das sogenannte binaerformat oder hexfile sind sprachen. sie liegen nur in einer fuer computer ausfuehrbaren form vor, computer wie sie uns bisher meistens gelaeufig sind, auch hier waeren ausnahmen moeglich, laut dessen was allerdings die informatik und ihre ideen zu formalen sprachen wiedergeben, wuerde es sich bei heutiger bauweise von computern immer anbieten, eine art umwandlung vom einen ins andere vor zu nehmen, diese wuerde nicht viel anders ausfallen, als bisher.

jedenfalls. wenn das klar ist - quelltext hin oder her - probieren wir es noch mal:

also, das ziel ist jetzt, das ding genau an zu gucken...

ich gucke, sind bei der uebertragung moegliche fehler drin???

1. Das waere die erste frage. dazu eroeffne ich einen neuen code

2. um zu gucken, vielleicht wird ja die zeichenkette von rs232 bisher richtig uebertragen, die uebertragung allerdings von der Conversation von A-D werten ist falsch, fehler?

dann bietet es sich an, eine zeichenkette dahin zu kopieren wo diese aufzeichnungen gelandet sind und an den computer stellvertretend zu uebertragen

3. ein beispiel. dann ueberpruefen sind, die parameter wirklich richtig eingestellt, oder macht der AD-Wandler selbst fehler

4. Versuchen die werte anders dar zu stellen, eben in form einer zeichenkette

   ```
   "1", "1", "0", ...
   ```

   etwas aehnliches

   ```
   S={"H", "L"}
   S^512
   oder |S| = 512
   ```

   an den pc zu uebertragen

dazu nachgucken. die Spannungswerte Voltage, liefern in einem verhaeltnis (0V..5.5V)/1024 mit einer genauigkeit, abweichung von 2 bit koennen schief laufen, hinten lsb ... heisst $1024//4 = 256$

also, hier muss man die grenzen erkenen und kann sagen

```
11.1111.1100b .. 11.1111.1111b steht fuer 5V?

00.0000.0000b .. 00.0000.0011b steht fuer 0V?
```

so in etwa das erkennen, anhand der werte.

ob in diesem bereich oder jenem - ...

und dann uebertragen

```
"H", "L" ..
```

und so weiter. wenn man vorher korrigiert hat, stimmen alle parameter sprich bits und flags in den registern

kurze raucherpause... dann zweiter (dritter) quelltext zum probieren...

da hat, schon definitiv was funktioniert... gucken sie mal, ich habe die ausgabe:

```
david@https://www.vajda-breadboard.de
```

und: ich habe das RS232 Transmit von der letzten uebung in den AD Wandler uebernommen. deswegen erscheint immer noch die

```
MSG david@https://www.vajda-breadboard.de
```

aber das vom AD-Wandler erscheint vorher:

```
FF6xxggg1gxj
```

moeglicherweise oder uebertragungsfehler.

gut - ich mache jetzt folgendes:

1.) photos vom board - bisher erzeuge ich keine frequenz - also doppeltes, dreifaches, ... phasenlaenge (haelfte, drittel der frequenz) muster von sample rate. ich habe vor eine digitale frequenz mit einem zweiten controller atmega8 zu erzeugen .ich nehme zur uebungszwecken immer nur digital. die herstellung spaeter ueber

a) Fourier Reihe original ist die Grundlage - habe uebrigens mal in tuebingen studiert und mathematik 1 fuer informatiker in tuebingen mit einer 4 bestanden. bei professor hauck. zufrieden war ich damit nicht sehr. immerhin doch, ich habe damals viele grundlagen fuer mathematik erobert,ohne die ich nicht zurecht gekommen waere
das buch: mathematik fuer informatik und bioinformatik, betrifft 4 semester uni tuebingen fuer informatiker, war fuer mich die grundlage, ist vielleicht noch immer, um dinge ernsthaft zu begreifen.

zum beispiel das karthesische produkt ordentlich genommen, aber noch viel mehr

trotzdem hat hauck mir gnaedig oder ungnaedig eine vier gegeben, ich habe damals sehr viel gelernt, von wirklichen grundlagen - allerdings viel weiter oben

wo die wirklichen grundlagen alleine nicht sind. trotzdem: ich bin auch nicht boese

computersysteme i/ii mit einer 3.7 glaube ich (technische informatik) habe ich mir ernsthaft verdient! ernsthaft! und ich sage, das ist das ding! Das ist es!

aber: ansonsten sage ich: unabhaengig des theaters...

b) die fourier reihe ist dort schoen beschrieben, exp(x) und sin(x), cos(x) haengen schoen zusammen als reihen

Mathematische grundlagen der fernuni hagen nicht vergesssen (Luise Unger) ist grundlage trotz allem anderen

und: ohne echtes fourier ein mal wirklich, geht glaube ich FFT und FFT2

nicht

2.) FFT ist eine notloesung aber im technischen sinne, die toll funktioniert. allerdings empfehle ich.

a) Robert Sedgewicks, algo in C bringen es auf den punkt: FFT(2?) ist wohl nur von der Funktionsweise im technischen ergebnis wohl eine fourier reihe, ansonsten nicht

dazu kann man viel theorie erklaeren: naemlich,

- langrage ist grundlage - multiplikation zweier polynome. polynome als Summen (Reihen) unterschiedlicher Potenzfunktionen unterschiedlichen grades, haben die angewohnheit, bei hohen gerade, irgendwann sehr viele schlingerkurse ein zu nehmen, aehnlich einer trigonometrischen.
- ist eine grundlage, erfordert integralrechnung. aber ich wuerde hier raten:

- diese gedanken sind muessig! fuer sie wahrscheinlich nicht! so zweifeln viele leute den sinn der griechischen mythologie an. tierkreiszeichen sind vom teufel, tauchen erstaunlicherweise die mythologische symbolik erkennbar im chinesischen wieder auf.

trotzdem: am anfang war niob und chaos. glaubt man nicht?

ich schon: marx irrt sich, wenn er meint - am anfang arbeiteten einige und die anderen auch. von der gesamtheit richtig, unnoetiges kapital, in form reiner scheine, gemeintes kapital, produktion da ... ok: blos: als menschen am anfang an arbeiteten

ja, arbeit ist ware - da stritten sie sich und beklauten sie sich um die ware arbeit so wie heute. ein kleiner fehler... denn: er hat die sache sehr gut erkannt und beschrieben, mit der steinzeit, abgesehen von robinsonaden nicht ganz richtig, die aber auch nicht, denn es gibt keine garantie

es gibt keine robinsonade, oder es ist eine? wenn man muss, dann muss man! wenn einem nur eine robinsonade auf einer insel uebrig bleibt, es nicht das leben was einen dazu zwingt sondern der hunger. 50:50 wuerde ich sagen

also: frueher arbeiteten die leute mit streit und diebstahl wie heute. dazu: was die griechen erfanden ist, dass der mensch am anfang im goldenen zeitalter

lebte. das hat marx abgekupfert, der mensch lebte nicht im goldenen zeitalter, an sich, wenn marx der mensch liesst, meint er: die menschheit

gemeint ist auch nicht der mensch, der hat immer dieselben probleme. also die menschheit lebte nie so ganz im goldenen zeitalter und nie ganz ohne

recht haben die griechen bei der deutung der goetter. am anfang war niob und chaos.

das bedeutet was? dass man als man anfing leider so genau versuchte das prinzip zu verstehen und sich im theokratischen sinne etwas zu erklaeren

meine antwort: prinzipien zu verstehen nuetzen nichts, etwas zum prinzip, liebe junge leute, ihren mathe und informatik profs koennen sie noch vertrauen vielen anderen nicht. die, die sagen, was man bei der jugend falsch macht, vertreten nur ein prinzip

ein prinzip ohne - das hat die tuebinger uni zu recht gesagt und denken sie ueber einen satz nach:

herr baer unser informatik lehrer am tg sagte: computer sind ohne software ein klumpen metall und ich sage ihnen: denken sie an sich: sie haben ein stueck software, aber es ist - so redet man mit ihnen - software und hardware - weder im buch, noch auf der platte, und der computer ist weg.

dann sage ich: glueckwunsch aber was ist ein stueck software ohne etwas physisches?

zweite frage: es gab menschen die sagten: die naturwissenschaft brauche keine uebung brauche kein verstehen, was verstanden wurde, ist verstanden

das was manche oekonomisch sich versuchen zu verhindern, zu erkennen: man kann auf dem markt vor allem auch immer wieder eines tun:

aktien sind schoen, und ich wuerde mir keine holen. manche leute sitzen 10 jahre da und lauern auf eine, weil es wohl nicht sicher ist

es ist einfach zu frueh, den guten freundlichen mann ab zu geben. nett sein ist nicht alles.

und nicht nur das: wenn nun die naturwissenschaft ohne menschen funktioniert, dann sagen sie doch einfach: die welt ist eine scheibe, soll schon jemand gesagt haben? warum immer freundlich ausdruecken, wenn sie keine ist. die frage ist immer noch, nach welchem mathematischen modell beurteilen sie das, denken sie nach

es gibt systeme die verteilen im geometrischen ihre koordinaten anders. nicht nur das: es ist die frage: was sie meinen, die raumkoordinaten, es gibt genug beziehen (entitaeten) in denen sie leben, graphenmodell - ihr geldbeutel ... (relationale datenbanken)

und so weiter: dann sage ich: wer so daher kommt, der kriegt halt das: wenn man am ende ein prinzip wird, dann wuensche den leuten glueck, teil eines prinzips geworden zu sein

das alles ist ein einziges prinzip, nur wer teil eines prinzips wurde, ist halt inzwischen co2 da oben und studiert nicht mehr, das ist der unterschied
aber das stimmt: niobe und chaos ist am anfang, und die ueberlieferung von mir nutzlos.

sie muessen die griechischen mythologischen werke gucken. da sind vielleicht sie. sie muessen sehr prinzipiell verstehen.

sie muessen die beweise der mathematik nach vollziehen das tue ich nicht! das muessen sie wissen. ich wollte immer den beweis der infitisimalrechnung verstehen, dann habe ich ihn auswendig gelernt

das hat zwar dazu gefuerht dass ich inzwischen ganz gut bin, gebe ihnen aber einen tipp:

der betonmischer weder noch der beweis der infinitisimalrechnung nutzt. was nutzt ist mit schreibschrift zu schreiben, die variablennamen zu kennen und: normal rechnen

das heisst moeglichst einfache jeden tag vollzogene aufgaben von aber abwechselnden integralrechnungsaufgaben zu machen

von matrizenrechnung usw.

erst verstehen wollen, dann merken: auswendig lernen wenn sie genug auswendig wissen, dann uebungen

und das ergebnis FFT2: geht so:

das sagt robert sedgewick. das nicht, aber: wenn sie es erst mal gewoehnt sind, dann:

langrage bedeutet multiplikation von polynomen, FFT2 heisst einfach: ganz einfach: sie machen es wie fourier nur sie nehmen irgendwie polynome, damit das aber tut und das ist der unterschied

so viele polynome multiplizieren wie moeglich, so viel wie moeglich, es sollte halt tun

wenn das geht, folgendes:

die aufgabe jetzt geht so

rein theoretisch und da liegt das problem: sollte man mit der sample rate, abtastrate - ich empfehle das linux sound system auswendig zu lernen und wichtig, sind 2er potenzen ohne die geht es nicht

`1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192...`

mindestens bis dahin. und dann die klassischen baud rates: `2400, 4800, 96000`

und dann kommen sie schon zur sampling rate
wenn ich sage: technische uebung ist das das eine.

wir machen die uebung mit dem controller, wenn sie da waeren wo ich bin jeden tag.

wie

1. LCD
2. RS232
3. Externes interrupt
4. timer interrupt

5. AD-Wandler

wenn der AD-Wandler jeden tag kommt, dann jeden tag ueben. eine zweite frequenz, was auch immer

die abtastung im puffer speichern, zum PC, hier merken.

und zwar wenn es so aussieht, es geht:

probieren: egal wie. nehmen wir 5V und 0V haben wir zwei werte, wir koennen sie an den PC in 0 ,1 uebertragen, so wie gerade

`david@https://www.vajda-breadboard.de`

hoch kam, rein theoretisch, 0, 1

werden es mehr: dann im PC ausgabe umwandeln

doch FFT2?

nachdem wert gemerkt und das ist reine theorie nuetzt ihnen was, aber auch nicht:

ueben! wg. sie brauchen das buch

eckard kw. moltrecht, amateuerfunk

sie muessen nicht amateuerfunk lizenspruefung machen, koennen aber

1. Amplitudenmodulation
2. Frequenz
3. Phasen

was heisst das?

1. Additiv
2. Multiplikativ

Additiv heisst, so weit ich weis: amplitude. sie modulieren auf die Traegerfrequenz das nutzsignal ihre bis 50kHz? Sprache auf. und es gibt eben diese mehr oder weniger addierte welle.

wie? mit einem transistor, ... ganz einfach

multiplikativ? wie geht das? mischstufe. und wie geht die?

sie kennen die graetz schaltung - gleichrichter - vier dioden netzspannung 230V das prinzip der schaltung selber verstehe ich selber nicht zu 100
wenn sie die traegerfrequenz (hf) rein tun und das nutzsignal niederfrequenz . dann haben sie eine frequenzmodulation, die frequenzen addieren sich. sie wird schneller und langsamer

interessanter weise, geht das mit digital.

der witz, sie machen generell eine phasenmodulation. so sage ich: jedes mal, nur dann, wenn ein wechsel von H/L oder L/H statt findet, knick in der phase

generell glaube ich aber, sie koennen die Frequenzmodulation mit einem digitalen NF Nutzsignal auch so vollziehen,

... irgendwie. sie muessen hier von entwuerfen ausgehen

wie erzeugt man toene am computer?

wenn sie eine digitales signal mit 1kHz an einen beeper schicken und einen mit 2kHz wird das signal... heller.

ok: so kann man toene erzeugen, obwohl es digital ist

was ist wenn wir sprechen? sie haben wahrscheinlich angst, warum?

weil: sie haben eine sauberes Nutzsignal auf einem sauberen HF Traegersignal

ok: blos: sie glauben das eine sprache gar nicht sauber ist, sie hat so zu sagen nicht ganz lange strecken 5kHz und 10kHz, usw sondern ein chaos?

nicht ganz richtig! denken sie dran: wird ein radio signal ueber FM -Frequenzmodulation VHF - bzw. UKW es gibt nicht nur VHF sondern,
Very, Ultra Super Extra

VHF UHF (SHF EHF ??? name)

denken sie daran: das signal ist trotzdem nur Frequenzmoduliert

selbst wenn die sprache solche zacken machen wuerde! denken sie an alternativ signale beim sprechen!

sie sprechen sagen wir: haben zacken!

sie sagen damit hoeren sie dies und das. aber es gibt eine alternative! sie erzeugen mit normaler frequenz dasselbe aber das gleiche vom hoeren, aber normale frequenz ohne zacken

5kHz und 10kHz

und der witz. wenn sie ueber die sample rate, das signal aufnehmen, dann:

muessen sie es speichern. danach kommt die wiederherstellung.

dabei wird es automatisch normal.

warum? sie sprechen mit zicken und zacken. es kommt zur abtastung. sie haben einfach 48kHz abtastung. sie speichern die werte. sie haben nur das. und stellen danach mit FFT2 das nutzsignal zur ausgabe wieder her. bei der Herstellung sind die zicken und zacken draussen. trotzdem: da das ding stimmt, passt die sache

noch besser: sie erzeugen am ausgang, eine frequenzmodulation. wenn das signal hergestellt ist, geben sie doppelte frequenz oder so aus.

sie wissen ja, der beeper. danach ueberlegen: wollen sie das mischen oder direkt. wie auch immer

das eine ist: staendiges ueben, der abtastrate am Controller atmega8, wie rs232, lcd hd44780 usw.

das andere ist rechnen, glauben sie nicht, sie koennen wenn sie dauernd rechnen und das sollte man, damit gleich alles. koennte man, aber man haette sehr

viel anstrengendes gefummel

das soll man nicht, man uebt, ohne zu wissen wo es lang geht, jede komponente jeden tag einzeln

wenn man es zusammensetzt, hoert man mit komponenten ueben nicht auf.

denn wo ist das ziel? das ist die frage?

```
;; ok, da war noch ein fehler drin, das 1024 passt natuerlich nicht in r16, ... aber durch

    ;; david vajda
    ;; 06/22/2026
    ;; m8 ad converter

    .include "m8def.inc"

    .dseg
    buffer: .byte 512
    .cseg

    ldi r16, HIGH (RAMEND)
    out SPH, r16
    ldi r16, LOW (RAMEND)
    out SPL, r16

    ldi r16, HIGH (25)
    out UBRRH, r16
    ldi r16, LOW (25)
    out UBRRL, r16

    ;; ldi r16, (1 << URSEL) | (1 << UCSZ2) | (1 << USBS)
    ldi r16, (1 << URSEL) | (1 << USBS)
    out UCSRC, r16
    ldi r16, (1 << TXEN) | (1 << UCSZ1) | (1 << UCSZ0) | (1 << UPM1) | (1 << UPM0)
    out UCSRB, r16

    ldi r16, (1 << REFS0) | (1 << MUX0)
    out ADMUX, r16

    ldi r16, (1 << ADEN) | (1 << ADSC) | (1 << ADPS2)
    ;; ADPS2: Prescaler (Samplerate): 16, 1 MHz/16 = 250kHz/4 ~= 51kHz??
    ;; 50 kHz <= Samplerate <= 200 kHz
    out ADCSRA, r16
```

```
 ldi r16, HIGH (buffer)
 mov YH, r16
 ldi r16, LOW (buffer)
 mov YL, r16

 ldi r18, 0xff
 conversation001:
 in r17, ADCH
 in r16, ADCL
 st Y+, r17
 st Y+, r16
 dec r18
 brne conversation001

 ldi r16, HIGH (buffer)
 mov YH, r16
 ldi r16, LOW (buffer)
 mov YL, r16
 ldi r18, 0xff
 transmit001:
 ld r16, Y+
 rcall rs232transmitch
 ld r16, Y+
 rcall rs232transmitch
 dec r18
 brne transmit001



label000:
 rcall rs232transmitstr
 rjmp label000

 rs232transmitstr:
 ldi r16, HIGH (2*msg000)
 mov ZH, r16
 ldi r16, LOW (2*msg000)
 mov ZL, r16

 rs232transmitstr1:
 lpm r16, Z+
 cpi r16, 0x00
 breq rs232transmitstrend
 rcall rs232transmitch
 rjmp rs232transmitstr1
 rs232transmitstrend:
 ret

 rs232transmitch:
```

```
sbis UCSRA, UDRE
rjmp rs232transmitch
out UDR, r16
ret

msg000: .db "david@https://www.vajda-breadboard.de", 10, 13, 0
```

andere version:

```
;; so, weit bin ich mit dem ad wandler fuer heute gekommen, und jetzt lasse ich es dab

;; david vajda
;; 06/22/2026
;; m8 ad converter

.include "m8def.inc"

.dseg
buffer: .byte 512
.cseg

ldi r16, HIGH (RAMEND)
out SPH, r16
ldi r16, LOW (RAMEND)
out SPL, r16

ldi r16, HIGH (25)
out UBRRH, r16
ldi r16, LOW (25)
out UBRRL, r16

;; ldi r16, (1 << URSEL) | (1 << UCSZ2) | (1 << USBS)
ldi r16, (1 << URSEL) | (1 << USBS)
out UCSRC, r16
ldi r16, (1 << TXEN) | (1 << UCSZ1) | (1 << UCSZ0) | (1 << UPM1) | (1 << UPM0)
out UCSRB, r16

ldi r16, (1 << REFS0) | (1 << MUX0)
out ADMUX, r16

ldi r16, (1 << ADEN) | (1 << ADSC) | (1 << ADPS2)
;; ADPS2: Prescaler (Samplerate): 16, 1 MHz/16 = 250kHz/4 ~= 51kHz??
;; 50 kHz <= Samplerate <= 200 kHz
out ADCSRA, r16

ldi r16, HIGH (buffer)
mov YH, r16
```

```
 ldi r16, LOW (buffer)
 mov YL, r16

 ldi r18, 1024
 conversation001:
 in r17, ADCH
 in r16, ADCL
 st Y+, r17
 st Y+, r16
 dec r18
 brne conversation001

 ldi r16, HIGH (buffer)
 mov YH, r16
 ldi r16, LOW (buffer)
 mov YL, r16
 ldi r18, 1024
 transmit001:
 ld r16, Y+
 rcall rs232transmitch
 ld r16, Y+
 rcall rs232transmitch
 dec r18
 brne transmit001



label000:
 rcall rs232transmitstr
 rjmp label000

 rs232transmitstr:
 ldi r16, HIGH (2*msg000)
 mov ZH, r16
 ldi r16, LOW (2*msg000)
 mov ZL, r16

 rs232transmitstr1:
 lpm r16, Z+
 cpi r16, 0x00
 breq rs232transmitstrend
 rcall rs232transmitch
 rjmp rs232transmitstr1
 rs232transmitstrend:
 ret

 rs232transmitch:
 sbis UCSRA, UDRE
 rjmp rs232transmitch
 out UDR, r16
```

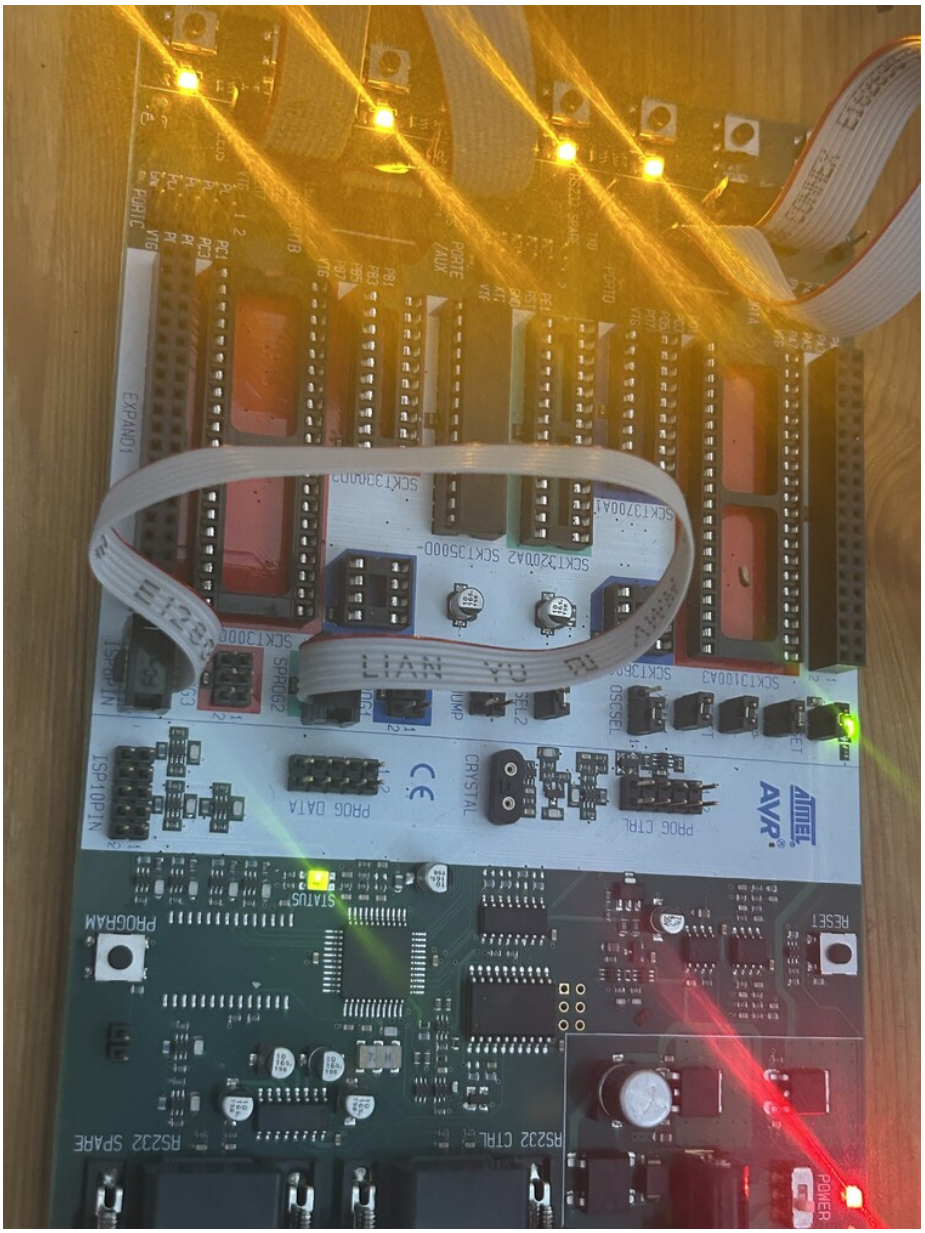
```
    ret

    msg000: .db "david@https://www.vajda-breadboard.de", 10, 13, 0
```

dann probieren wir es gleich aus, nachdem ich noch mal draussen war... wenn das geschehen ist, muesste dnf dran kommen und ich heute noch mal lernen, mal sehen, wenn es dann tut, das wird es, was der AD Wandler so sagt...

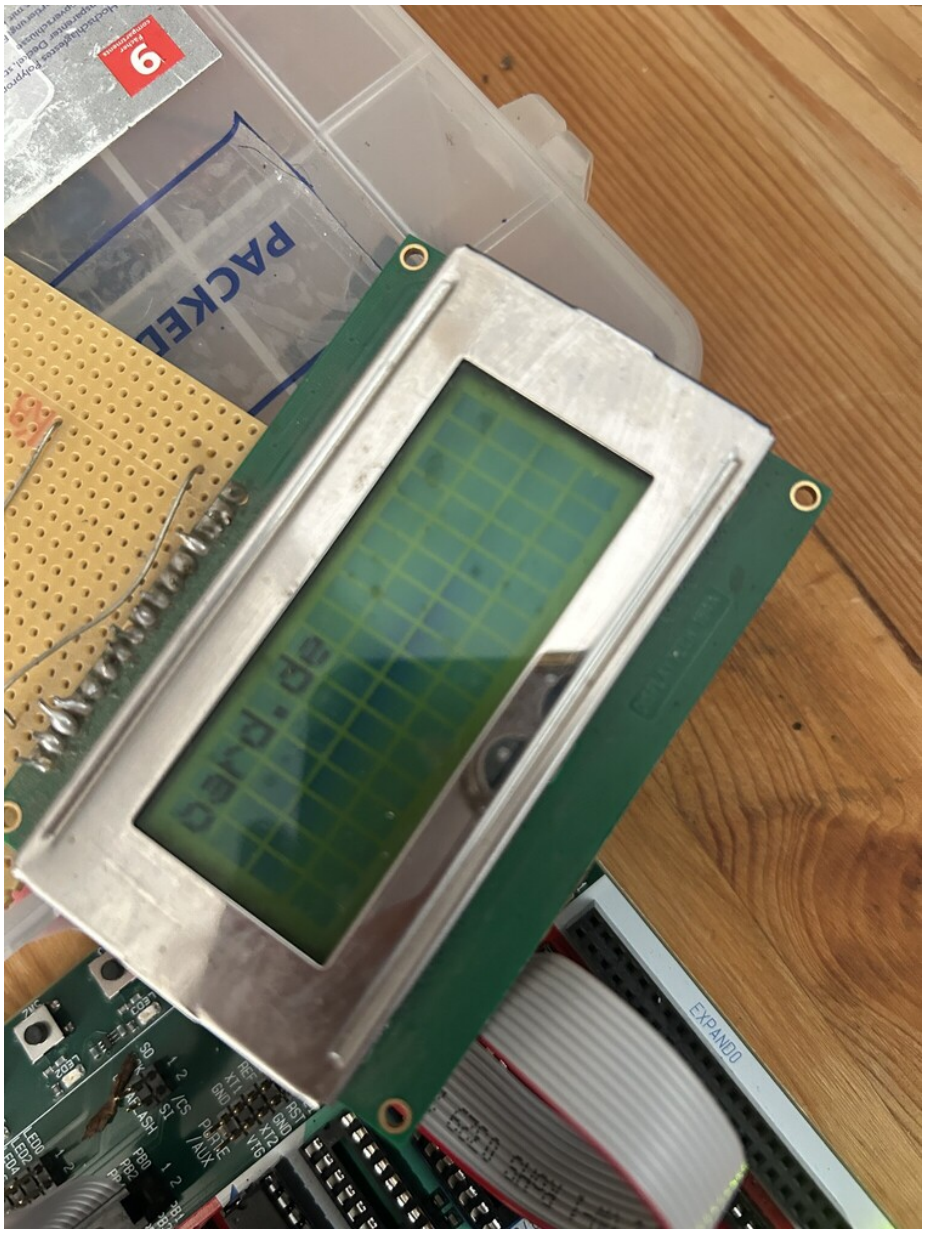
ok, tut, jetzt, ... das war sehr wichtig, weil der ADC (Analog Digital Converter) Analog Digital Wandler... braucht das RS232 jetzt auch wieder richtig, wir koennten es danach probieren...

```
;; die schleife hat gefehlt ... deswegen immer 'd'

    ;; 06/22/2026
    ;; david vajda
    ;; m8 rs232 transmit excersize

    .include "m8def.inc"

    ldi r16, HIGH (RAMEND)
    out SPH, r16
    ldi r16, LOW (RAMEND)
    out SPL, r16

    ldi r16, 0xff   ;; nicht notwendig
    out DDRD, r16   ;; nicht notwendig


    ;; PARITY BITs (odd, even, no parity)
    ;; 1/2 STOP BITs (cstopb - in stty und: USBS - USART Stop Bit Select
    ;;     UPM1:0  USART Parity Mode
    ;; 5 .. 9 Datenbit (UCSZ2:0 - USART Character Size)
    ;; TXEN/RXEN Transmit Enable Recieve Enable
    ;; 25 (Baudrate - 2400)
    ;; URSEL - USART Register Select - is UCSRC or UBRRH - UBBRH ...
    ;;     think - we don't need this, val: 25
    ;; UCSRC, UCSRB, UCSRA
    ;; TXEN, RXEN, UDRE


    ldi r16, HIGH (25)
    out UBRRH, r16
    ldi r16, LOW (25)
    out UBRRL, r16
```

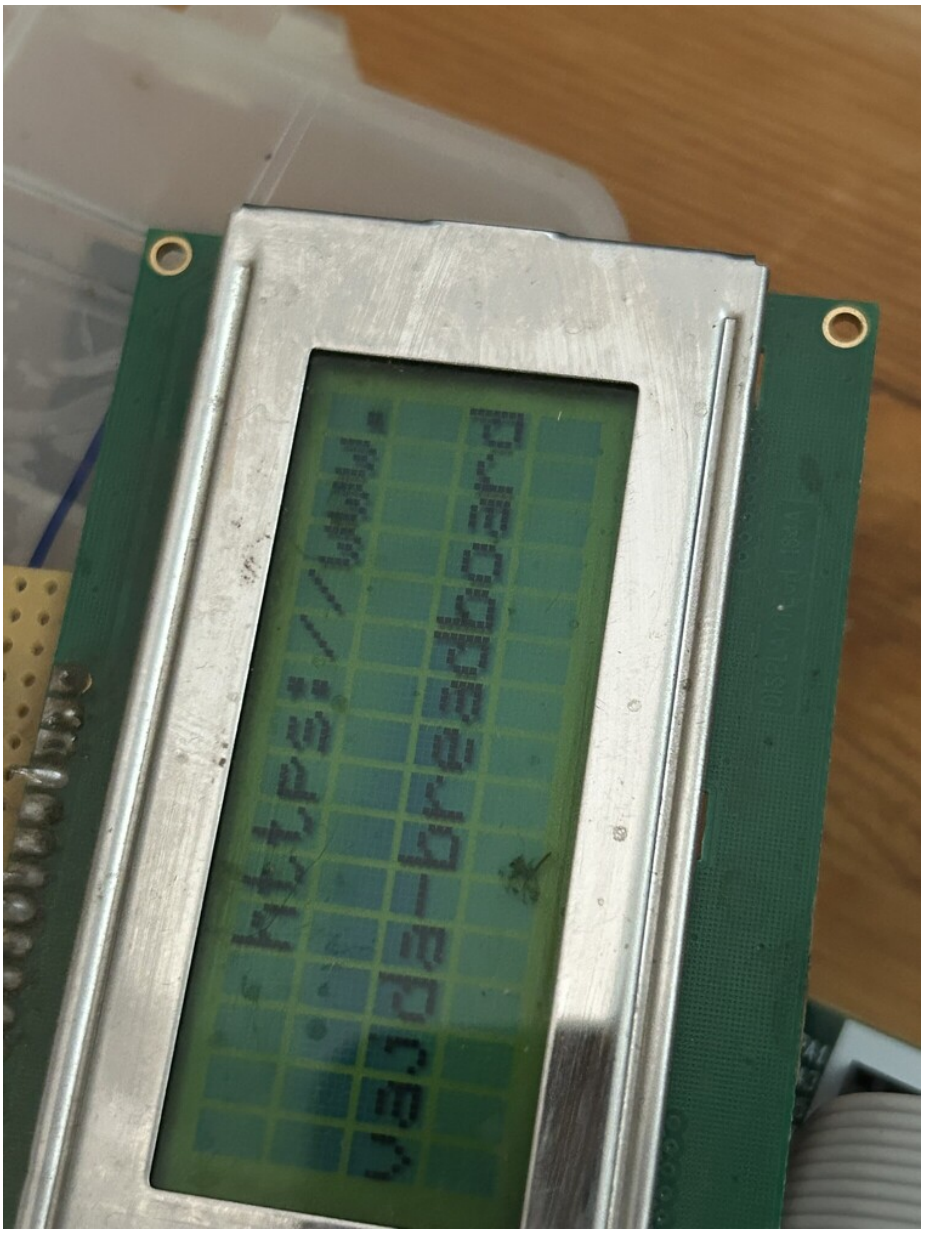
```
;; ldi r16, (1 << URSEL) | (1 << UCSZ2) | (1 << USBS)
ldi r16, (1 << URSEL) | (1 << USBS)
out UCSRC, r16
ldi r16, (1 << TXEN) | (1 << UCSZ1) | (1 << UCSZ0) | (1 << UPM1) | (1 << UPM0)
out UCSRB, r16

label000:
rcall rs232transmitstr
rjmp label000


rs232transmitstr:
ldi r16, HIGH (2*msg000)
mov ZH, r16
ldi r16, LOW (2*msg000)
mov ZL, r16

rs232transmitstr1:
lpm r16, Z+
cpi r16, 0x00
breq rs232transmitstrend
rcall rs232transmitch
rjmp rs232transmitstr1
rs232transmitstrend:
ret

rs232transmitch:
sbis UCSRA, UDRE
rjmp rs232transmitch
out UDR, r16
ret



msg000: .db "david@https://www.vajda-breadboard.de", 10, 13, 0
```

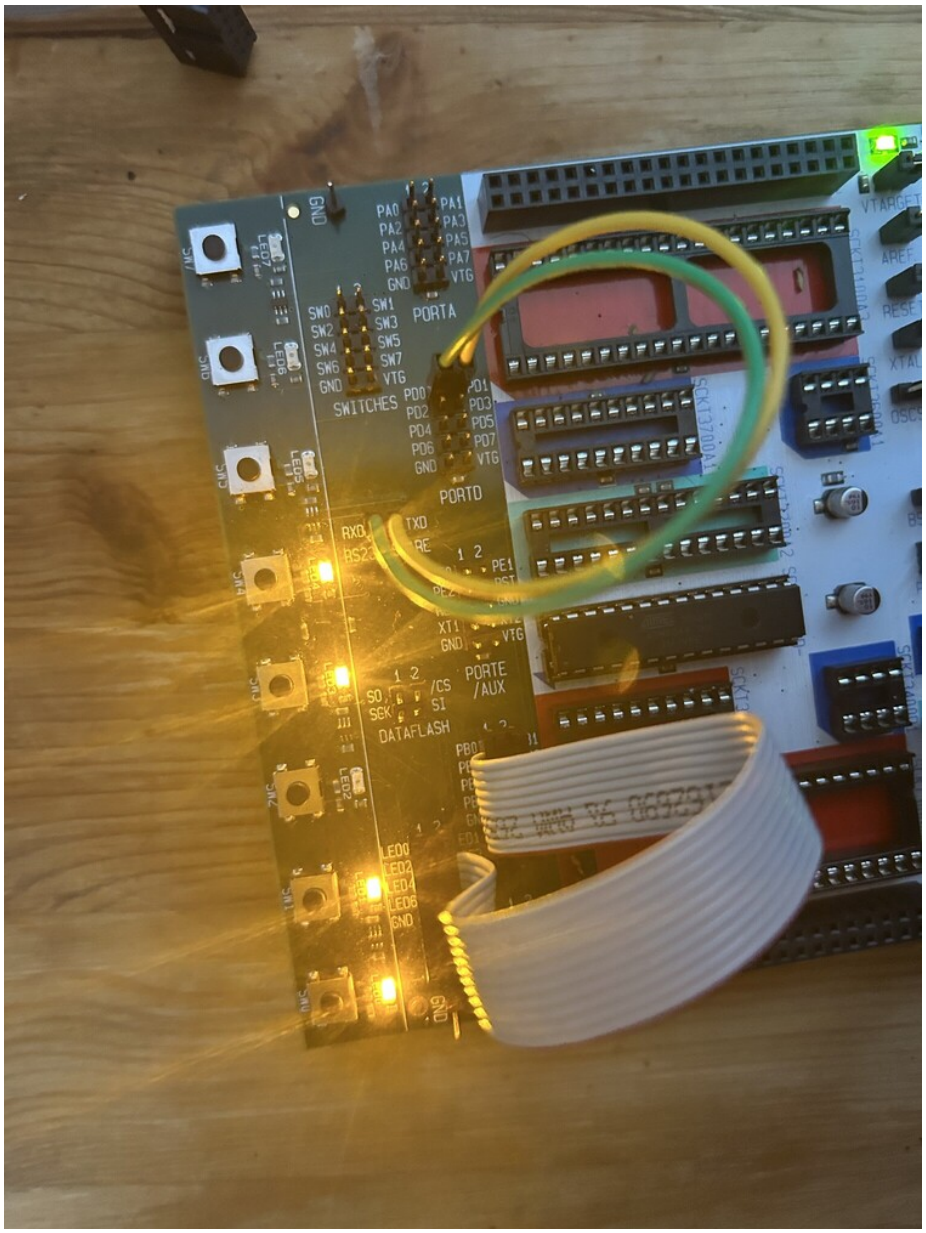

PROGRAM
STATUS
RESET
POWER
RS232 SPARE
RS232 CTRL
PROG DATA
PROG CTRL
CRYSTAL
ISP10PIN
ISP6PIN
JUMP
SPEL2
SPRG2
EXPAND1
PORTE /AUX
PORTD
PORTC
PORTB
PORTA
PORTIC
LIAN YU
AVR
ATMEL


www.reichelt.de
reichelt elektronik GmbH
Elektronikring 1
26452 Sande
Germany
WARNING
TO AVOID ELECTRIC SHOCK, THE POWER CORD PROTECTIVE GROUNDING CONDUCTOR MUST BE CONNECTED TO GROUND.
FOR CONTINUED FIRE PROTECTION, REPLACE FUSE ONLY WITH 250V FUSE OF THE SPECIFIED TYPE AND RATING. NO OPERATION SERVICEABLE COMPONENTS INSIDE.
DO NOT REMOVE COVERS; REFER SERVICING TO QUALIFIED PERSONNEL.
MADE IN CHINA
KORAD
U203 DIGITAL CONTROL DC POWER SUPPLY
VOLTAGE COARSE
C.V
10.00 V
0.2180 A
002.1 W
ON/OFF


PACKE
EXPANDO


https://www.
breadboard


https://www.




```
;; ja, dazu kam, dass ich ld und ldi statt lpm verwendet habe, wenn dann ld und sicher ni

   ;; david vajda
   ;; 06/22/2026
   ;; lcd hd44780

   .include "m8def.inc"


   ;; lcd hd44780 - init: 0x03, 0x03, 0x03, 0x02, 0x20
   ;; always send: ENABLE/HIGH nop nop nop ENABLE/LOW
```

```
;; ENABLE is Pin...
;; (0) VSS (1) VDD (2) VEE (3) RS (4) RW (5) E ...
;; (5) and that is: from PORTB/PORTD in this case
;; Bit 5
;; sbi cbi set bit in i/o register, clear bit in i/o register

.equ    HD44780WAITLOOPLONGVAL1    =    $21
.equ    HD44780WAITLOOPLONGVAL0    =    $C9
.equ    HD44780WAITLOOPSHORTVAL0   =    42


ldi r16, HIGH (RAMEND)
out SPH, r16
ldi r16, LOW (RAMEND)
out SPL, r16

ldi r16, 0xff
out DDRB, r16

rcall hd44780init
ldi r16, HIGH (2*msg001)
mov ZH, r16
ldi r16, LOW (2*msg001)
mov ZL, r16
rcall hd44780sendmsg

main001: rjmp main001

msg001: .db "https://www.vajda-breadboard.de", 0

hd44780init:
ldi r16, 50
hd44780init001:
rcall hd44780waitlooplong
dec r16
brne hd44780init001

ldi r16, 0x03
out PORTB, r16
rcall hd44780enable
rcall hd44780waitlooplong

rcall hd44780enable
rcall hd44780waitlooplong

rcall hd44780enable
rcall hd44780waitlooplong

ldi r16, 0x02
out PORTB, r16
```

```
rcall hd44780enable
rcall hd44780waitlooplong

ldi r16, 0x20
out PORTB, r16
rcall hd44780enable
rcall hd44780waitlooplong

       ldi r16, 0b00101000        ; 4Bit / 2 Zeilen / 5x8
       rcall hd44780sendcmd
       ldi r16, 0b00001100        ; Display ein / Cursor aus / kein Blinken
       rcall hd44780sendcmd
       ldi r16, 0b00000110        ; Cursor inkrementieren / kein Scrollen
       rcall hd44780sendcmd
       ldi r16, 0b00000001
       rcall hd44780sendcmd
ret

hd44780enable:
sbi PORTB, 5
nop
nop
nop
cbi PORTB, 5
ret

hd44780waitlooplong:
push r16
push r17
ldi r16, HD44780WAITLOOPLONGVAL1
hd44780waitlooplong1:
ldi r17, HD44780WAITLOOPLONGVAL0
hd44780waitlooplong0:
dec r17
breq hd44780waitlooplong0
dec r16
breq hd44780waitlooplong1
pop r17
pop r16
ret

hd44780waitloopshort:
push r16
ldi r16, HD44780WAITLOOPSHORTVAL0
hd44780waitloopshort0:
dec r16
breq hd44780waitloopshort0
pop r16
ret
```

```
;; send command
;; send data
;; data is bit 4 (3 .. 0) is data - 4 bit - 4 bit modus
;; 4 bit modus has to be actived -
;;     0b0000 0001     clear display
;;     0b0000 001x     cursor home
;;     0b0000 01is     entry mode
;;     0b0000 1dbc     on/off controll
;;     0b001dnfxxx     display configuration .. par example:
;; d - 4 bit modus/8 bit modus

hd44780sendcmd:
mov r17, r16
swap r17
andi r17, 0b00001111
out PORTB, r17
cbi PORTB, 4
rcall hd44780enable
mov r17, r16
andi r17, 0b00001111
out PORTB, r17
cbi PORTB, 4
rcall hd44780enable
rcall hd44780waitloopshort
ret

hd44780senddata:
mov r17, r16
swap r17
andi r17, 0b00001111
out PORTB, r17
sbi PORTB, 4
rcall hd44780enable
mov r17, r16
andi r17, 0b00001111
out PORTB, r17
sbi PORTB, 4
rcall hd44780enable
rcall hd44780waitloopshort
ret

hd44780sendmsg:
lpm r16, Z+
subi r16, 0x00
breq hd44780sendmsgend
rcall hd44780senddata
rjmp hd44780sendmsg
hd44780sendmsgend:
ret
```

Ich weiß auch, warum, weil ld keinen mathematischen Vergleich liefert

So jetzt im nächsten Schritt jetzt haben wir ja das. Jetzt müssen wir die serielle Schnittstelle ausprobieren und ich glaub trotzdem vorher erstes LCD weil die serielle Schnittstelle benutzen wir ja auch für die Samplerates für den AD Wandler

Heute dann später lerne ich wieder auswendig. Alles was es zu diesem Thema gibt hier und von dem i586 und wenn ich das habe, gucke ich mal ja vielleicht wie gesagt ein bisschen russisch und vielleicht noch die Zeichen so ach ja, ich mach vielleicht doch noch die DNF wie immer und dann auch probieren den GAL

```
;; es war doch B und nicht A, aber es geht....  der zweite zaehler, ich stelle ihn ueber C

    ;; david vajda
    ;; 06/22/2026
    ;; m8 timer/counter interrupt

    .include "m8def.inc"

    .org 0x000
    rjmp RESET
    .org OVF1Addr
    rjmp OVF1Hndlr
    .org OVF0Addr
    rjmp OVF0Hndlr

    RESET:
    ldi r16, HIGH (RAMEND)
    out SPH, r16
    ldi r16, LOW (RAMEND)
    out SPL, r16

    ;; prescaler - 1, 8, 64, 256, 1024??

    ldi r16, 0xff
    out DDRB, r16

    ldi r16, 0xff
    out PORTB, r16

    ldi r16, (1 << CS10)
    out TCCR1B, r16 ;; ??
    ;; timer0: 8 bit timer
```

```
;; timer1: 16 bit timer - TCCR1A u. TCCR1B - Timer 1
;;     Timer/Counter Control Register
;;     A und B gibt es, weil bei Timer 1 ist mehr ein zu stellen,
;;     wie bei Timer 0 wie (UBBRH und UBBRL)
;; timer0: 8 Bit Timer, TIMSK, und TCCR0...

ldi r16, (1 << TOIE1)
out TIMSK, r16
;; TIMSK - MSK: Mask, T: Timer/Counter, I: Intervall? Interrupt

sei
main001: rjmp main001

OVF0Hndlr:
com r16
out PORTB, r16
reti

OVF1Hndlr:
com r16
out PORTB, r16
reti
```

Gut, dann kommt die Zeltschleife und das timer interrupt und der Counter Ich muss gucken, ob ich bei RS 232 Fehler drin hab und dann gucke ich mir noch das LCD an HD 4478 null und dann gucke ich ob wie gesagt wie es mit dem AD Wandler ist. Es wird ja auch per RS 232 hier an den PC übertragen. Ich muss gucken ob sozusagen wie soll ich sagen erstens mal wie ich untersuche, dass das übertragene Signal das richtige ist wie auch immer und ob das dann noch anderen Signalen entspricht und so weiter wenn ich das weiter untersuche deswegen muss auch die Übertragung an den PC stimmen. Ich geh kurz eine rauchen.

```
;; david vajda
    ;; 06/22/2026
    ;; m8 extern interrupt excersize


    .include "m8def.inc"

    .org 0x000
    rjmp RESET
    .org INT0Addr
    rjmp INT0Hndlr
    .org INT1Addr
    rjmp INT1Hndlr

    RESET:
```

```
ldi r16, HIGH (RAMEND)
out SPH, r16
ldi r16, LOW (RAMEND)
out SPL, r16

ldi r16, 0xff
out DDRB, r16
ldi r16, 0x00
out DDRD, r16

ldi r16, (1 << ISC10) | (1 << ISC00)
out MCUCR, r16
ldi r16, (1 << INT1) | (1 << INT0)
out GICR, r16

ldi r16, 0xff
out PORTB, r16

sei

main001: rjmp main001


INT0Hndlr:
dec r16
out PORTB, r16
reti

INT1Hndlr:
inc r16
out PORTB, r16
reti
```