asm quelltexte usw.

david vajda

23. Juli 2026

1 lcdh44780m806142026.asm

```
;; david vajda
;; 06/14/2026
;; m8/lcd hd44780

.include "m8def.inc"

.equ HD44780WAITLOOPLONGCOUNT1 = $21
.equ HD44780WAITLOOPLONGCOUNT2 = $C9
.equ HD44780WAITLOOPSHORTCOUNT = 42

ldi r16, HIGH (RAMEND)
out SPH, r16
ldi r16, LOW (RAMEND)
out SPL, r16
ldi r16, 0xff
out DDRB, r16

rcall hd44780init
rcall hd44780cleardisplay
ldi ZH, HIGH (2*str)
ldi ZL, LOW (2*str)
l1:
lpm r16, Z+
rcall hd44780echo
subi r16, 0x00
brne l1

str: .db "https://www.vajda-breadboard.de", 10, 13, 0x00, 0x00

hd44780setup:
sbi PORTB, 5
nop
nop
nop
cbi PORTB, 5
ret
```

```
hd44780waitlooplong:
push r16
push r17
ldi r16, HD44780WAITLOOPLONGCOUNT1
hd44780waitlooplongl1:
ldi r17, HD44780WAITLOOPLONGCOUNT2
hd44780waitlooplongl2:
dec r17
brne hd44780waitlooplongl2
dec r16
brne hd44780waitlooplongl1
pop r17
pop r16
ret


hd44780waitloopshort:
push r16
ldi r16, HD44780WAITLOOPSHORTCOUNT
hd44780waitloopshortl1:
dec r16
brne hd44780waitloopshortl1
pop r16
ret

hd44780init:
;; $3, $3, $3, $2, $20
ldi r16, 50 ;; weiss ich auswendig so lange am anfang warten
hd44780initl1:
rcall hd44780waitlooplong
dec r16
brne hd44780initl1
ldi r16, $3
out PORTB, r16
rcall hd44780setup
rcall hd44780waitlooplong
rcall hd44780setup
rcall hd44780waitlooplong
rcall hd44780setup
rcall hd44780waitlooplong
ldi r16, $2
rcall hd44780setup
rcall hd44780waitlooplong
ldi r16, $20
rcall hd44780setup
rcall hd44780waitlooplong

;; weiss auswendig
;; 0b0000 0001 clear display
;; 0b0000 001x cursor home
```

```
;; 0b0000 01is entry mode
;; 0b0000 1dbc display on, blink, cursor on
;; 0b0001
;; hier folgt etwas zu konfiguration ist wichtig...
;; damit das interface auch 4 bit modus macht...


;;; rein kopiert
           ldi r16, 0b00101000        ; 4Bit / 2 Zeilen / 5x8
           rcall hd44780cmd
           ldi r16, 0b00001100        ; Display ein / Cursor aus / kein Blinken
           rcall hd44780cmd
           ldi r16, 0b00000110        ; Cursor inkrementieren / kein Scrollen
           rcall hd44780cmd
           ret
ret

hd44780cmd:
mov r17, r16
swap r17
andi r17, 0b00001111
out PORTB, r17
rcall hd44780setup
mov r17, r16
andi r17, 0b00001111
out PORTB, r17
rcall hd44780setup
rcall hd44780waitloopshort
ret

hd44780echo:
mov r17, r16
swap r17
andi r17, 0b00001111
ori r17, 0x10
out PORTB, r17
rcall hd44780setup
mov r17, r16
andi r17, 0b00001111
ori r17, 0x10
out PORTB, r17
rcall hd44780setup
rcall hd44780waitloopshort
ret

hd44780cleardisplay:
ldi r16, 0b00000001
rcall hd44780cmd
ret
```

2 lcdh44780m8counter06262026.asm

```
;; david vajda
;; 06/14/2026
;; m8/lcd hd44780 ... to m8/t counter... up...

.include "m8def.inc"

.equ HD44780WAITLOOPLONGCOUNT1 = $21
.equ HD44780WAITLOOPLONGCOUNT2 = $C9
.equ HD44780WAITLOOPSHORTCOUNT = 42

.org 0x0000
rjmp RESET
.org OVF1addr
rjmp OVF1addr

RESET:

ldi r16, HIGH (RAMEND)
out SPH, r16
ldi r16, LOW (RAMEND)
out SPL, r16
ldi r16, 0xff
out DDRB, r16

rcall hd44780init
rcall hd44780cleardisplay
ldi ZH, HIGH (2*str)
ldi ZL, LOW (2*str)
l1:
lpm r16, Z+
rcall hd44780echo
subi r16, 0x00
brne l1

str: .db "https://www.vajda-breadboard.de", 10, 13, 0x00, 0x00

hd44780setup:
sbi PORTB, 5
nop
nop
nop
cbi PORTB, 5
ret

hd44780waitlooplong:
push r16
push r17
ldi r16, HD44780WAITLOOPLONGCOUNT1
```

```
hd44780waitlooplongl1:
ldi r17, HD44780WAITLOOPLONGCOUNT2
hd44780waitlooplongl2:
dec r17
brne hd44780waitlooplongl2
dec r16
brne hd44780waitlooplongl1
pop r17
pop r16
ret


hd44780waitloopshort:
push r16
ldi r16, HD44780WAITLOOPSHORTCOUNT
hd44780waitloopshortl1:
dec r16
brne hd44780waitloopshortl1
pop r16
ret

hd44780init:
;; $3, $3, $3, $2, $20
ldi r16, 50 ;; weiss ich auswendig so lange am anfang warten
hd44780initl1:
rcall hd44780waitlooplong
dec r16
brne hd44780initl1
ldi r16, $3
out PORTB, r16
rcall hd44780setup
rcall hd44780waitlooplong
rcall hd44780setup
rcall hd44780waitlooplong
rcall hd44780setup
rcall hd44780waitlooplong
ldi r16, $2
rcall hd44780setup
rcall hd44780waitlooplong
ldi r16, $20
rcall hd44780setup
rcall hd44780waitlooplong

;; weiss auswendig
;; 0b0000 0001 clear display
;; 0b0000 001x cursor home
;; 0b0000 01is entry mode
;; 0b0000 1dbc display on, blink, cursor on
;; 0b0001
;; hier folgt etwas zu konfiguration ist wichtig...
```

```
;; damit das interface auch 4 bit modus macht...


;;; rein kopiert
           ldi r16, 0b00101000        ; 4Bit / 2 Zeilen / 5x8
           rcall hd44780cmd
           ldi r16, 0b00001100        ; Display ein / Cursor aus / kein Blinken
           rcall hd44780cmd
           ldi r16, 0b00000110        ; Cursor inkrementieren / kein Scrollen
           rcall hd44780cmd
           ret
ret

hd44780cmd:
mov r17, r16
swap r17
andi r17, 0b00001111
out PORTB, r17
rcall hd44780setup
mov r17, r16
andi r17, 0b00001111
out PORTB, r17
rcall hd44780setup
rcall hd44780waitloopshort
ret

hd44780echo:
mov r17, r16
swap r17
andi r17, 0b00001111
ori r17, 0x10
out PORTB, r17
rcall hd44780setup
mov r17, r16
andi r17, 0b00001111
ori r17, 0x10
out PORTB, r17
rcall hd44780setup
rcall hd44780waitloopshort
ret

hd44780cleardisplay:
ldi r16, 0b00000001
rcall hd44780cmd
ret

;;; divide command
;;; max per 16 0b00001111
;;; reg mit 8 Bit
```

```
;;; 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
...

ret

byte2bcdstr:
```

3 m8countled06272026extint.asm

```
;; 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
```

4 m8countled06272026oszi.asm

```
;; 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
```

5 m8countled06272026waitloop.asm

```
;; 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, 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
```

6 m8countledtimer06292026.asm

```
;; david vajda
;; 06/29/2026
;; 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...
;; die nachgucken

ldi r16, (1 << CS11)
out TCCR1B, r16
ldi r16, (1 << TOIE1)
out TIMSK, r16
;; so heissen die richtig, mal gucken...


sei
main001: rjmp main001

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

7 m8div06252026.asm

```
;; 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
and r18, r19
cp r19, r18
breq loop1end
lsr r19
```

```
dec r20
rjmp loop1

loop1end:

out PORTD, r19

end: rjmp end
```

8 m8div06252026b.asm

```
;; 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
```

9 m8div06252026c.asm

```
;; 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

ldi r17, DIVIDEND
ldi r19, DIVISOR
ldi r21, 0x00

.def    dividendr           =   r17
.def    divisorr            =   r19
.def    submaskr            =   r21
.def    dividendsubmaskr    =   r18
.def    quotientr           =   r23
.def    dividendsubmasksub0growingr =   r24

prerun001:
sbrc dividendr, 7
rjmp prerun1end
lsl dividendr

lsl submaskr
ori submaskr, 0b00000001

rjmp prerun001
prerun1end:

ldi r16, 0x00

ldi quotientr, 0x00
div000:

and dividendsubmaskr, submaskr
cp dividendsubmaskr, divisorr
brlt div001
ori quotientr, 0b00000001
```

```
rjmp div002
div001:
ori quotientr, 0b00000000

div002:
lsr submaskr
lsr divisorr
rjmp div000


end: rjmp end
```

10 m8div06252026d.asm

```
;; 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
```

11 m8div06252026e.asm

```
;; 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

ldi r16, DIVIDEND
ldi r17, DIVISOR

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
lsl r22
```

```
ori r22, 0b00000001
rjmp subtraction002end
subtraction001end:
lsr r21
lsl 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
```

12 m8ledcountwaitloop07102026.asm

```
;; (c) david vajda
;; wait loop - m8 - led out
;; 07/10/2026

.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
waitloop01l1:
ldi r17, 0xff
```

```
    waitloop01l2:
    ldi r18, 0xff
    waitloop01l3:
    dec r18
    brne waitloop01l3
    dec r17
    brne waitloop01l2
    dec r16
    brne waitloop01l1
    pop r18
    pop r17
    pop r16
    ret
```

13 bool4tab.csv

```
0;0;0;0
0;0;0;1
0;0;1;0
0;0;1;1
0;1;0;0
0;1;0;1
0;1;1;0
0;1;1;1
1;0;0;0
1;0;0;1
1;0;1;0
1;0;1;1
1;1;0;0
1;1;0;1
1;1;1;0
1;1;1;1
```

14 dff07032026.gal

```
GAL16V8
MODULE rslatch06272026

    CLOCK X1 X0 NC NC NC NC NC NC GND
    /OE Y1 Z1 Y0 Z0 NC NC NC NC VCC

    Y0 = CLOCK & /Z0 & /X0
    Z0 = CLOCK & /Y0 & X0

    Y1 = CLOCK & /Z1 & /X1
    Z1 = CLOCK & /Y1 & X1
```

```
DESCRIPTION
this is rs latch
```

15 gal16v8DNF07122026.gal

```
DEVICE GAL16V8
MODULE GAL16V8QUINE4DNF07122026

CLK X3 X2 X1 X0 NC NC NC NC VCC
/OE Y0 NC NC NC NC NC NC NC GND

Y0 = /X2 & /X1 # /X3 & X2 # /X1 & /X0 # /X3 & X0 # X3 & /X2 & /X0

DESCRIPTION
Eine DNF mit quine mc cluskey fuer den letzten versuch
```

16 gal16v8DNF07122026B.gal

```
;; DEVICE GAL16V8
;; das wort device wird hier wohl nicht verwendet
,, es ist allerdings ein Device GAL16V8
GAL16V8
MODULE GAL16V8QUINE4DNF07122026

;; die frage war, ist im norden CLK
;; und im sueden /OE oder umgekehrt, indem falle
;; stimmte es so, ebenso bei VCC und GND
;; hier wohl verkehrt ...
;; und noch bei input/output, das ist wohl richtig herum
CLK X3 X2 X1 X0 NC NC NC NC GND
/OE Y0 NC NC NC NC NC NC NC VCC

Y0 = /X2 & /X1 # /X3 & X2 # /X1 & /X0 # /X3 & X0 # X3 & /X2 & /X0

DESCRIPTION
Eine DNF mit quine mc cluskey fuer den letzten versuch

;; das ergebnis braucht noch eine neufassung

;; david@www001:~$ ./bin/galasm ./gal16v8DNF07122026.gal
;; GALasm 2.1, Portable GAL Assembler
;; Copyright (c) 1998-2003 Alessandro Zummo. All Rights Reserved
;; Original sources Copyright (c) 1991-96 Christian Habermann

;; Error in line 1: Line  1: type of GAL expected
;; Assembling failed.
```

```
;; david@www001:~$

;; das heisst, wir machen noch einen file
;; und das DEVICE kommt ohne kommentar an den anfang
```

17 gal16v8DNF07122026C.gal

```
GAL16V8
MODULE GAL16V8QUINE4DNF07122026

;; DEVICE GAL16V8
;; das wort device wird hier wohl nicht verwendet
,, es ist allerdings ein Device GAL16V8

;; die frage war, ist im norden CLK
;; und im sueden /OE oder umgekehrt, indem falle
;; stimmte es so, ebenso bei VCC und GND
;; hier wohl verkehrt ...
;; und noch bei input/output, das ist wohl richtig herum
CLK X3 X2 X1 X0 NC NC NC NC GND
/OE Y0 NC NC NC NC NC NC NC VCC

Y0 = /X2 & /X1 # /X3 & X2 # /X1 & /X0 # /X3 & X0 # X3 & /X2 & /X0

DESCRIPTION
Eine DNF mit quine mc cluskey fuer den letzten versuch

;; das ergebnis braucht noch eine neufassung

;; david@www001:~$ ./bin/galasm ./gal16v8DNF07122026.gal
;; GALasm 2.1, Portable GAL Assembler
;; Copyright (c) 1998-2003 Alessandro Zummo. All Rights Reserved
;; Original sources Copyright (c) 1991-96 Christian Habermann

;; Error in line 1: Line  1: type of GAL expected
;; Assembling failed.
;; david@www001:~$

;; das heisst, wir machen noch einen file
;; und das DEVICE kommt ohne kommentar an den anfang

;; so hier sind die kommentare auch fehl am platz, in den ersten zeilen

;; david@www001:~$ ./bin/galasm ./gal16v8DNF07122026C.gal
;; GALasm 2.1, Portable GAL Assembler
;; Copyright (c) 1998-2003 Alessandro Zummo. All Rights Reserved
;; Original sources Copyright (c) 1991-96 Christian Habermann

;; Error in line 6: illegal character in pin declaration
```

```
;; Assembling failed.
;; david@www001:~$
```

18 gal16v8DNF07122026D.chp

```
                    GAL16V8

                 -------\___/-------
             CLK |  1           20 | VCC
                 |                 |
              X3 |  2           19 | NC
                 |                 |
              X2 |  3           18 | NC
                 |                 |
              X1 |  4           17 | NC
                 |                 |
              X0 |  5           16 | NC
                 |                 |
              NC |  6           15 | NC
                 |                 |
              NC |  7           14 | NC
                 |                 |
              NC |  8           13 | NC
                 |                 |
              NC |  9           12 | Y0
                 |                 |
             GND | 10           11 | /OE
                 --------------------
```

19 gal16v8DNF07122026D.gal

```
GAL16V8
MODULE GAL16V8QUINE4DNF07122026

CLK X3 X2 X1 X0 NC NC NC NC GND
/OE Y0 NC NC NC NC NC NC NC VCC

Y0 = /X2 & /X1 # /X3 & X2 # /X1 & /X0 # /X3 & X0 # X3 & /X2 & /X0

DESCRIPTION
Eine DNF mit quine mc cluskey fuer den letzten versuch

;; das ergebnis braucht noch eine neufassung

;; david@www001:~$ ./bin/galasm ./gal16v8DNF07122026.gal
;; GALasm 2.1, Portable GAL Assembler
```

```
;; Copyright (c) 1998-2003 Alessandro Zummo. All Rights Reserved
;; Original sources Copyright (c) 1991-96 Christian Habermann

;; Error in line 1: Line  1: type of GAL expected
;; Assembling failed.
;; david@www001:~$

;; das heisst, wir machen noch einen file
;; und das DEVICE kommt ohne kommentar an den anfang

;; so hier sind die kommentare auch fehl am platz, in den ersten zeilen

;; david@www001:~$ ./bin/galasm ./gal16v8DNF07122026C.gal
;; GALasm 2.1, Portable GAL Assembler
;; Copyright (c) 1998-2003 Alessandro Zummo. All Rights Reserved
;; Original sources Copyright (c) 1991-96 Christian Habermann

;; Error in line 6: illegal character in pin declaration
;; Assembling failed.
;; david@www001:~$

;; so, hat es funktioniert

;; david@www001:~$ ./bin/galasm ./gal16v8DNF07122026D.gal
;; GALasm 2.1, Portable GAL Assembler
;; Copyright (c) 1998-2003 Alessandro Zummo. All Rights Reserved
;; Original sources Copyright (c) 1991-96 Christian Habermann

;; Assembler Phase 1 for "./gal16v8DNF07122026D.gal"
;; Assembler Phase 2 for "./gal16v8DNF07122026D.gal"
;; GAL16V8; Operation mode: simple; Security fuse off
;; Assembling successfully completed.
;; david@www001:~$
```

20 gal16v8DNF07122026D.pin

Pin #	Name	Pin Type
1	CLK	Input
2	X3	Input
3	X2	Input
4	X1	Input
5	X0	Input
6	NC	Input
7	NC	Input
8	NC	Input
9	NC	Input

```
  10   | GND      | GND
  11   | /OE      | Input
  12   | Y0       | Output
  13   | NC       | NC
  14   | NC       | NC
  15   | NC       | NC
  16   | NC       | NC
  17   | NC       | NC
  18   | NC       | NC
  19   | NC       | NC
  20   | VCC      | VCC
```

21 m8rs232transmit07232026.asm

```
;; (c) david vajda
;; 07/13/2026
;; rs232 transmit m8

.include "m8def.inc"

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) | (0 << UCSZ2) | (1 << USBS)
ldi r16, (1 << URSEL)
out UCSRC, r16
ldi r16, (1 << UPM0) | (1 << TXEN)
out UCSRB, r16


main002:
ldi r16, HIGH (msg001*2)
mov ZH, r16
ldi r16, LOW (msg001*2)
mov ZL, r16
main001:
lpm r16, Z+
subi r16, 0x00
breq main002
rcall rs232transmit
rjmp main002
```

```
msg001: .db "david@https://www.vajda-breadboard.de", 10, 13, 0x00

rs232transmit:
sbis UCSRA, UDRE
rjmp rs232transmit
out UDR, r16
ret
```

22 m8texterninterrupt07132026.asm

```
;; (c) david vajda
;; 07/13/2026
;; externes intervall

.include "m8def.inc"

.org 0x0000
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, (1 << ISC11) | (1 << ISC10) | (1 << ISC01) | (1 << ISC00)
out MCUCR, r16

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

ldi r16, 0xff
main001: rjmp main001

INT0hndlr:
dec r16
out PORTB, r16
reti

INT1hndlr:
inc r16
```

```
out PORTB, r16
reti
```

23 m8timercounteroszi07122026.asm

```
;; (c) david vajda
;; 07/12/2026
;; Timer/Counter Intervall, Interrupt, m8 - versuch
;; den timer auf das oszilloskop genau ein zu stellen

.include "m8def.inc"

.org 0x0000
rjmp RESET
.org OVF0Addr
rjmp OVF0timer

RESET:

ldi r16, HIGH (RAMEND)
out SPH, r16
ldi r16, LOW (RAMEND)
out SPL, r16
ldi r16, 0xff
out DDRB, r16

ldi r16, (1 << CS00)
out TCCR0, r16
ldi r16, (1 << TOIE0)
out TIMSK, r16
sei

main001: rjmp main001

OVF0timer:
inc r16
out PORTB, r16
reti
```

24 quine07112026.py

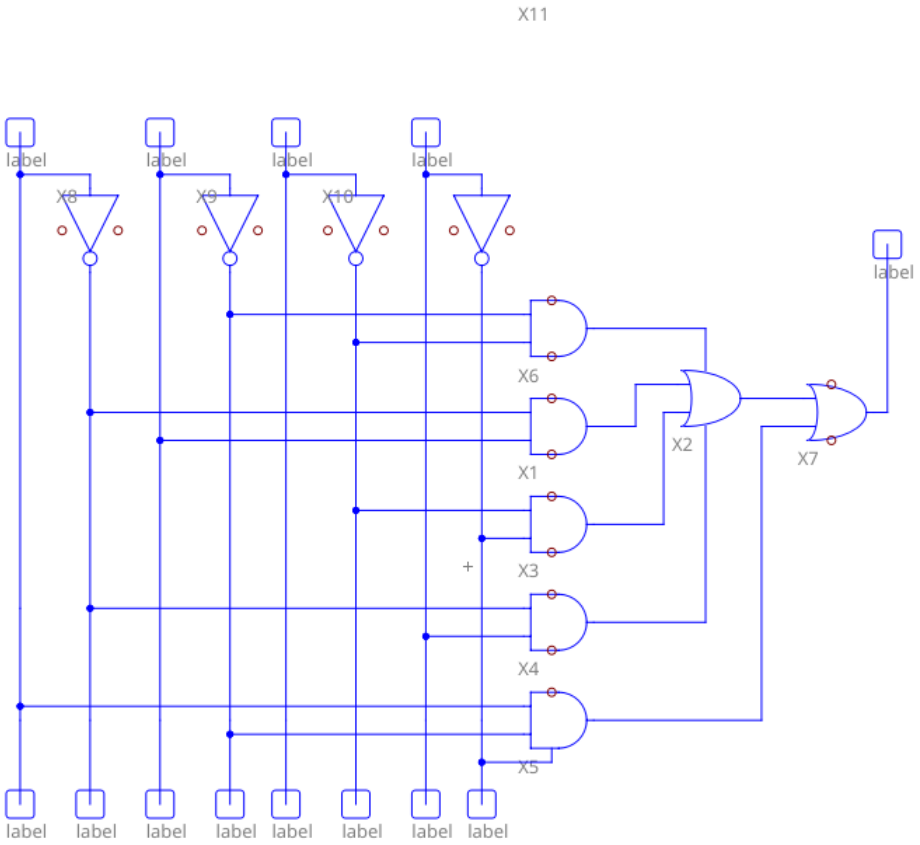
```
import csv
import sys
import sympy as sp
from sympy.logic import boolalg

# (c) david vajda
# 07/11/2026
```

```
# quine .. 4 input var .. dnf ..

# with open('bool4tab.csv', newline='\n') as csvfile:
#    bool4tab = csv.reader (csvfile, delimiter=',')
#    for row in bool4tab:
#        print (', '.join(row))
# galasmformula = "Y = /X2 & /X1 \# /X3 & X2 \# /X1 & /X0 \# /X3 & X0 \# X3 & /X2 & /X0"
galasmformula = sys.argv [1]
print (galasmformula)


python3formula = galasmformula.replace ("/", "not ").replace ("&", " and ").replace ("\#",
print (python3formula)
print ("x3x2x1x0y0")
with open('bool4tab.csv', newline='') as csvfile:
    bool4tab = csv.reader (csvfile,delimiter=';')
    for row in bool4tab:
        X0 = bool(int(row [3]))
        X1 = bool(int(row [2]))
        X2 = bool(int(row [1]))
        X3 = bool(int(row [0]))
        Y = eval(python3formula,{"X3": X3,"X2": X2,"X1": X1,"X0": X0})
        # print (not X2  and  not X1  or  not X3  and  X2  or  not X1  and  not X0  or  no
        # print (Y)
        print (int(X3), int(X2), int(X1), int(X0),int(Y))
```

25 quine07112026LaTeX.py

```
import csv
import sys
import sympy as sp
from sympy.logic import boolalg

# (c) david vajda
# 07/11/2026
# quine .. 4 input var .. dnf ..

# with open('bool4tab.csv', newline='\n') as csvfile:
#    bool4tab = csv.reader (csvfile, delimiter=',')
#    for row in bool4tab:
#        print (', '.join(row))
# galasmformula = "Y = /X2 & /X1 \# /X3 & X2 \# /X1 & /X0 \# /X3 & X0 \# X3 & /X2 & /X0"
galasmformula = sys.argv [1]
print (galasmformula)


python3formula = galasmformula.replace ("/", "not ").replace ("&", " and ").replace ("\#",
```

```
print (python3formula)
print("\\begin{tabular}{|c|c|c|c||c|}")
print("\\hline")
print ("$x_3$ & $x_2$ & $x_1$ & $x_0$ & $y_0$\\\\")
print("\\hline")
with open('bool4tab.csv', newline='') as csvfile:
    bool4tab = csv.reader (csvfile,delimiter=';')
    for row in bool4tab:
        X0 = bool(int(row [3]))
        X1 = bool(int(row [2]))
        X2 = bool(int(row [1]))
        X3 = bool(int(row [0]))
        Y = eval(python3formula,{"X3": X3,"X2": X2,"X1": X1,"X0": X0})
        # print (not X2  and  not X1  or  not X3  and  X2  or  not X1  and  not X0  or  no
        # print (Y)
        print ("$", int(X3), "$ & $", int(X2), "$ & $", int(X1), "$ & $", int(X0), "$ & $"
        print ("\\hline")
print ("\\end{tabular}")

print("\\begin{tikztimingtable}")
X0TikZTimingStr = "X0 & "
X1TikZTimingStr = "X1 & "
X2TikZTimingStr = "X2 & "
X3TikZTimingStr = "X3 & "
YTikZTimingStr = "Y & "
HighLowPegelStr = ["L", "H"]
with open('bool4tab.csv', newline='') as csvfile:
    bool4tab = csv.reader (csvfile,delimiter=';')
    for row in bool4tab:
        X0 = bool(int(row [3]))
        X1 = bool(int(row [2]))
        X2 = bool(int(row [1]))
        X3 = bool(int(row [0]))
        Y = eval(python3formula,{"X3": X3,"X2": X2,"X1": X1,"X0": X0})
        X0TikZTimingStr = X0TikZTimingStr + "1" + str(HighLowPegelStr[X0]) + " "
        X1TikZTimingStr = X1TikZTimingStr + "1" + str(HighLowPegelStr[X1]) + " "
        X2TikZTimingStr = X2TikZTimingStr + "1" + str(HighLowPegelStr[X2]) + " "
        X3TikZTimingStr = X3TikZTimingStr + "1" + str(HighLowPegelStr[X3]) + " "
        YTikZTimingStr = YTikZTimingStr + "1" + str(HighLowPegelStr[Y]) + " "
X0TikZTimingStr = X0TikZTimingStr + "\\\\"
X1TikZTimingStr = X1TikZTimingStr + "\\\\"
X2TikZTimingStr = X2TikZTimingStr + "\\\\"
X3TikZTimingStr = X3TikZTimingStr + "\\\\"
YTikZTimingStr = YTikZTimingStr + "\\\\"
print(X0TikZTimingStr)
print(X1TikZTimingStr)
print(X2TikZTimingStr)
print(X3TikZTimingStr)
print(YTikZTimingStr)
print("\\end{tikztimingtable}")
```



26 bool4tab.csv

```
0;0;0;0
0;0;0;1
0;0;1;0
0;0;1;1
0;1;0;0
0;1;0;1
0;1;1;0
0;1;1;1
1;0;0;0
1;0;0;1
1;0;1;0
1;0;1;1
1;1;0;0
1;1;0;1
1;1;1;0
1;1;1;1
```

27 dff07032026.gal

```
GAL16V8
MODULE rslatch06272026

    CLOCK X1 X0 NC NC NC NC NC NC GND
    /OE Y1 Z1 Y0 Z0 NC NC NC NC VCC

    Y0 = CLOCK & /Z0 & /X0
    Z0 = CLOCK & /Y0 & X0

    Y1 = CLOCK & /Z1 & /X1
    Z1 = CLOCK & /Y1 & X1


DESCRIPTION
this is rs latch
```

28 gal16v8DNF07122026.gal

```
DEVICE GAL16V8
MODULE GAL16V8QUINE4DNF07122026

CLK X3 X2 X1 X0 NC NC NC NC VCC
/OE Y0 NC NC NC NC NC NC NC GND

Y0 = /X2 & /X1 # /X3 & X2 # /X1 & /X0 # /X3 & X0 # X3 & /X2 & /X0

DESCRIPTION
Eine DNF mit quine mc cluskey fuer den letzten versuch
```

29 gal16v8DNF07122026B.gal

```
;; DEVICE GAL16V8
;; das wort device wird hier wohl nicht verwendet
,, es ist allerdings ein Device GAL16V8
GAL16V8
MODULE GAL16V8QUINE4DNF07122026

;; die frage war, ist im norden CLK
;; und im sueden /OE oder umgekehrt, indem falle
;; stimmte es so, ebenso bei VCC und GND
;; hier wohl verkehrt ...
;; und noch bei input/output, das ist wohl richtig herum
CLK X3 X2 X1 X0 NC NC NC NC GND
/OE Y0 NC NC NC NC NC NC NC VCC

Y0 = /X2 & /X1 # /X3 & X2 # /X1 & /X0 # /X3 & X0 # X3 & /X2 & /X0
```

```
DESCRIPTION
Eine DNF mit quine mc cluskey fuer den letzten versuch

;; das ergebnis braucht noch eine neufassung

;; david@www001:~$ ./bin/galasm ./gal16v8DNF07122026.gal
;; GALasm 2.1, Portable GAL Assembler
;; Copyright (c) 1998-2003 Alessandro Zummo. All Rights Reserved
;; Original sources Copyright (c) 1991-96 Christian Habermann

;; Error in line 1: Line  1: type of GAL expected
;; Assembling failed.
;; david@www001:~$

;; das heisst, wir machen noch einen file
;; und das DEVICE kommt ohne kommentar an den anfang
```

30 gal16v8DNF07122026C.gal

```
GAL16V8
MODULE GAL16V8QUINE4DNF07122026

;; DEVICE GAL16V8
;; das wort device wird hier wohl nicht verwendet
,, es ist allerdings ein Device GAL16V8

;; die frage war, ist im norden CLK
;; und im sueden /OE oder umgekehrt, indem falle
;; stimmte es so, ebenso bei VCC und GND
;; hier wohl verkehrt ...
;; und noch bei input/output, das ist wohl richtig herum
CLK X3 X2 X1 X0 NC NC NC NC GND
/OE Y0 NC NC NC NC NC NC NC VCC

Y0 = /X2 & /X1 # /X3 & X2 # /X1 & /X0 # /X3 & X0 # X3 & /X2 & /X0

DESCRIPTION
Eine DNF mit quine mc cluskey fuer den letzten versuch

;; das ergebnis braucht noch eine neufassung

;; david@www001:~$ ./bin/galasm ./gal16v8DNF07122026.gal
;; GALasm 2.1, Portable GAL Assembler
;; Copyright (c) 1998-2003 Alessandro Zummo. All Rights Reserved
;; Original sources Copyright (c) 1991-96 Christian Habermann

;; Error in line 1: Line  1: type of GAL expected
;; Assembling failed.
;; david@www001:~$
```

```
;; das heisst, wir machen noch einen file
;; und das DEVICE kommt ohne kommentar an den anfang

;; so hier sind die kommentare auch fehl am platz, in den ersten zeilen

;; david@www001:~$ ./bin/galasm ./gal16v8DNF07122026C.gal
;; GALasm 2.1, Portable GAL Assembler
;; Copyright (c) 1998-2003 Alessandro Zummo. All Rights Reserved
;; Original sources Copyright (c) 1991-96 Christian Habermann

;; Error in line 6: illegal character in pin declaration
;; Assembling failed.
;; david@www001:~$
```

31 gal16v8DNF07122026D.chp

```
                GAL16V8

            -------\___/-------
        CLK |  1           20 | VCC
            |                 |
         X3 |  2           19 | NC
            |                 |
         X2 |  3           18 | NC
            |                 |
         X1 |  4           17 | NC
            |                 |
         X0 |  5           16 | NC
            |                 |
         NC |  6           15 | NC
            |                 |
         NC |  7           14 | NC
            |                 |
         NC |  8           13 | NC
            |                 |
         NC |  9           12 | Y0
            |                 |
        GND | 10           11 | /OE
            -------------------
```

32 gal16v8DNF07122026D.gal

```
GAL16V8
MODULE GAL16V8QUINE4DNF07122026
```

```
CLK X3 X2 X1 X0 NC NC NC NC GND
/OE Y0 NC NC NC NC NC NC NC VCC

Y0 = /X2 & /X1 # /X3 & X2 # /X1 & /X0 # /X3 & X0 # X3 & /X2 & /X0

DESCRIPTION
Eine DNF mit quine mc cluskey fuer den letzten versuch

;; das ergebnis braucht noch eine neufassung

;; david@www001:~$ ./bin/galasm ./gal16v8DNF07122026.gal
;; GALasm 2.1, Portable GAL Assembler
;; Copyright (c) 1998-2003 Alessandro Zummo. All Rights Reserved
;; Original sources Copyright (c) 1991-96 Christian Habermann

;; Error in line 1: Line  1: type of GAL expected
;; Assembling failed.
;; david@www001:~$

;; das heisst, wir machen noch einen file
;; und das DEVICE kommt ohne kommentar an den anfang

;; so hier sind die kommentare auch fehl am platz, in den ersten zeilen

;; david@www001:~$ ./bin/galasm ./gal16v8DNF07122026C.gal
;; GALasm 2.1, Portable GAL Assembler
;; Copyright (c) 1998-2003 Alessandro Zummo. All Rights Reserved
;; Original sources Copyright (c) 1991-96 Christian Habermann

;; Error in line 6: illegal character in pin declaration
;; Assembling failed.
;; david@www001:~$

;; so, hat es funktioniert

;; david@www001:~$ ./bin/galasm ./gal16v8DNF07122026D.gal
;; GALasm 2.1, Portable GAL Assembler
;; Copyright (c) 1998-2003 Alessandro Zummo. All Rights Reserved
;; Original sources Copyright (c) 1991-96 Christian Habermann

;; Assembler Phase 1 for "./gal16v8DNF07122026D.gal"
;; Assembler Phase 2 for "./gal16v8DNF07122026D.gal"
;; GAL16V8; Operation mode: simple; Security fuse off
;; Assembling successfully completed.
;; david@www001:~$
```

33 gal16v8DNF07122026D.pin

Pin #	Name	Pin Type
1	CLK	Input
2	X3	Input
3	X2	Input
4	X1	Input
5	X0	Input
6	NC	Input
7	NC	Input
8	NC	Input
9	NC	Input
10	GND	GND
11	/OE	Input
12	Y0	Output
13	NC	NC
14	NC	NC
15	NC	NC
16	NC	NC
17	NC	NC
18	NC	NC
19	NC	NC
20	VCC	VCC

34 m8rs232transmit07232026.asm

```
;; (c) david vajda
;; 07/13/2026
;; rs232 transmit m8

.include "m8def.inc"

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) | (0 << UCSZ2) | (1 << USBS)
ldi r16, (1 << URSEL)
out UCSRC, r16
ldi r16, (1 << UPM0) | (1 << TXEN)
out UCSRB, r16
```

```
main002:
ldi r16, HIGH (msg001*2)
mov ZH, r16
ldi r16, LOW (msg001*2)
mov ZL, r16
main001:
lpm r16, Z+
subi r16, 0x00
breq main002
rcall rs232transmit
rjmp main002

msg001: .db "david@https://www.vajda-breadboard.de", 10, 13, 0x00

rs232transmit:
sbis UCSRA, UDRE
rjmp rs232transmit
out UDR, r16
ret
```

35 m8texterninterrupt07132026.asm

```
;; (c) david vajda
;; 07/13/2026
;; externes intervall

.include "m8def.inc"

.org 0x0000
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, (1 << ISC11) | (1 << ISC10) | (1 << ISC01) | (1 << ISC00)
out MCUCR, r16

ldi r16, (1 << INT1) | (1 << INT0)
```

```
out GICR, r16

ldi r16, 0xff
main001: rjmp main001

INT0hndlr:
dec r16
out PORTB, r16
reti

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

36 m8timercounteroszi07122026.asm

```
;; (c) david vajda
;; 07/12/2026
;; Timer/Counter Intervall, Interrupt, m8 - versuch
;; den timer auf das oszilloskop genau ein zu stellen

.include "m8def.inc"

.org 0x0000
rjmp RESET
.org OVF0Addr
rjmp OVF0timer

RESET:

ldi r16, HIGH (RAMEND)
out SPH, r16
ldi r16, LOW (RAMEND)
out SPL, r16
ldi r16, 0xff
out DDRB, r16

ldi r16, (1 << CS00)
out TCCR0, r16
ldi r16, (1 << TOIE0)
out TIMSK, r16
sei

main001: rjmp main001

OVF0timer:
inc r16
out PORTB, r16
```

```
reti
```

37 quine07112026.py

```
import csv
import sys
import sympy as sp
from sympy.logic import boolalg

# (c) david vajda
# 07/11/2026
# quine .. 4 input var .. dnf ..

# with open('bool4tab.csv', newline='\n') as csvfile:
#    bool4tab = csv.reader (csvfile, delimiter=',')
#    for row in bool4tab:
#        print (', '.join(row))
# galasmformula = "Y = /X2 & /X1 \# /X3 & X2 \# /X1 & /X0 \# /X3 & X0 \# X3 & /X2 & /X0"
galasmformula = sys.argv [1]
print (galasmformula)


python3formula = galasmformula.replace ("/", "not ").replace ("&", " and ").replace ("\#",
print (python3formula)
print ("x3x2x1x0y0")
with open('bool4tab.csv', newline='') as csvfile:
    bool4tab = csv.reader (csvfile,delimiter=';')
    for row in bool4tab:
        X0 = bool(int(row [3]))
        X1 = bool(int(row [2]))
        X2 = bool(int(row [1]))
        X3 = bool(int(row [0]))
        Y = eval(python3formula,{"X3": X3,"X2": X2,"X1": X1,"X0": X0})
        # print (not X2  and  not X1  or  not X3  and  X2  or  not X1  and  not X0  or  no
        # print (Y)
        print (int(X3), int(X2), int(X1), int(X0),int(Y))
```

38 quine07112026LaTeX.py

```
import csv
import sys
import sympy as sp
from sympy.logic import boolalg

# (c) david vajda
# 07/11/2026
```

```
# quine .. 4 input var .. dnf ..

# with open('bool4tab.csv', newline='\n') as csvfile:
#    bool4tab = csv.reader (csvfile, delimiter=',')
#    for row in bool4tab:
#        print (', '.join(row))
# galasmformula = "Y = /X2 & /X1 \# /X3 & X2 \# /X1 & /X0 \# /X3 & X0 \# X3 & /X2 & /X0"
galasmformula = sys.argv [1]
print (galasmformula)


python3formula = galasmformula.replace ("/", "not ").replace ("&", " and ").replace ("\#",
print (python3formula)
print("\\begin{tabular}{|c|c|c|c||c|}")
print("\\hline")
print ("$x_3$ & $x_2$ & $x_1$ & $x_0$ & $y_0$\\\\")
print("\\hline")
with open('bool4tab.csv', newline='') as csvfile:
    bool4tab = csv.reader (csvfile,delimiter=';')
    for row in bool4tab:
        X0 = bool(int(row [3]))
        X1 = bool(int(row [2]))
        X2 = bool(int(row [1]))
        X3 = bool(int(row [0]))
        Y = eval(python3formula,{"X3": X3,"X2": X2,"X1": X1,"X0": X0})
        # print (not X2  and  not X1  or  not X3  and  X2  or  not X1  and  not X0  or  no
        # print (Y)
        print ("$", int(X3), "$ & $", int(X2), "$ & $", int(X1), "$ & $", int(X0), "$ & $"
        print ("\\hline")
print ("\\end{tabular}")

print("\\begin{tikztimingtable}")
X0TikZTimingStr = "X0 & "
X1TikZTimingStr = "X1 & "
X2TikZTimingStr = "X2 & "
X3TikZTimingStr = "X3 & "
YTikZTimingStr = "Y & "
HighLowPegelStr = ["L", "H"]
with open('bool4tab.csv', newline='') as csvfile:
    bool4tab = csv.reader (csvfile,delimiter=';')
    for row in bool4tab:
        X0 = bool(int(row [3]))
        X1 = bool(int(row [2]))
        X2 = bool(int(row [1]))
        X3 = bool(int(row [0]))
        Y = eval(python3formula,{"X3": X3,"X2": X2,"X1": X1,"X0": X0})
        X0TikZTimingStr = X0TikZTimingStr + "1" + str(HighLowPegelStr[X0]) + " "
        X1TikZTimingStr = X1TikZTimingStr + "1" + str(HighLowPegelStr[X1]) + " "
        X2TikZTimingStr = X2TikZTimingStr + "1" + str(HighLowPegelStr[X2]) + " "
        X3TikZTimingStr = X3TikZTimingStr + "1" + str(HighLowPegelStr[X3]) + " "
```

```
        YTikZTimingStr = YTikZTimingStr + "1" + str(HighLowPegelStr[Y]) + " "
X0TikZTimingStr = X0TikZTimingStr + "\\\\"
X1TikZTimingStr = X1TikZTimingStr + "\\\\"
X2TikZTimingStr = X2TikZTimingStr + "\\\\"
X3TikZTimingStr = X3TikZTimingStr + "\\\\"
YTikZTimingStr = YTikZTimingStr + "\\\\"
print(X0TikZTimingStr)
print(X1TikZTimingStr)
print(X2TikZTimingStr)
print(X3TikZTimingStr)
print(YTikZTimingStr)
print("\\end{tikztimingtable}")
```