

VHDL - DNF - 3 Input Var - TTL - 74xx - ...

David Vajda

1. Februar 2026

heute erwartet sie, endlich das siebensegment anzeige decoder - bzw. ein kodierer. Kodierung eines zeichen eines zeichen vorrats so ogfähr in das eines anderen. DIN 44300/181 mit EEPROM - kompliziert zu erklären.

02/01/2026, (12) David Vajda
Seite 14,

ült:

zwei gute wege:

1.) Reihe

2.) immer 10er u.
2er Log..

beispiele mit pc...

also:

100, 100.000, 1000...

und dann angeben sowie

2, 8, 2048, 512, 8192, ...

2.) Thema: integral: macht besonders
kompliziert:

bei übungsprog selber
fehlt es:

es wäre schön bei
ganzzahligen Einheiten
Skala auf abzisse

ließe sich Riemann integral
möglichst gut ablesen...

⇒ zu viel perfektion!!!

02/01/2026, (c) David
Vajra Seite 12, ...
macht nichts!

war nur
Spaßsabotage!

noch nicht besonders
schlimm!

thumbs up + smiley
congratulations!
hope you are well!
dazu gut
ok
ming
some
hao = gut !!!

mand
 $\odot + \text{D} = \text{hell} \dots$

jedenfalls:

fußgängerzone:
frau + kind
= gut!

jedenfalls:

jetzt kommt matrizen-
multiplikation.
funktioniert ordentlich

02/01/2026, (c) David Vojda
Seite 10, 1e do..

Verhältnisse:

Kreisparbasse..

✓ Einführung einer real Wirtschaft
im Bereich Rohstoffe
erwerb unter freiem Himmel:
pfandflaschen

→ zukünftiges praktikum
im Bereich:
Rohstoffe...

✓ So wie weiteren Chem
erfahrenen..

Zum ausbau eines
gesamtverständnisses lebendiger
Schaltkreise...

folgt: Wieder zurück zur mathematik...
bin ich persönlich sehr weit...
(pause...)

02/01/2026 (c) David Vajda
Seite 8, 10 do...

a) abhängig: Wetterbericht...

b) abhängig: Rohstoffe:

✓ Regenwetter brauche Holz...

✓ Eis brauche Erzeugnisse Landwirtschaft.

c) Beeinflussung 3. : Medien:

✓ Zuckerrübe tut sich besser an, wir brauchen Fleisch!

✓ Kältige Leute spielen im Regen Fußball und durchqueren den Matsch...

ok: Deswegen: Die p sind teil von: Wirtschaftsprüfung auslösen der Produktion, advertisement... anderer.

ok: Bedeutet:

Die Produktion deutet auf stetigen Wachstum.

02/01/2026 (c) Daniel Vajda
Seite 6, 10 do...

von malte repchant werden wir
aber nie!

Ich möchte mir keine
Anmerkungen über stetiges
Wachstum erlauben.

Besonders im Bereich des Aktien-
marktes - siehe prognosen-
und: Die aufzeichnungen der Wetterdaten
aber das ist jetzt ein neuer
fundamentaler privat Beweis
hoffentlich nicht schon bald
neue geschichte...

nur neben: geschichtlich historischen
feststellungen: Albert Einstein war
physiker... gibt es auch eigentliche
prognosen im Aktienmarkt...

Die darüber aufbauend geben:
wie wird es mit der Aktie weiter-
gehen. Je nachdem ob Daniel
Benz weiterhin in die

02/01/2026, (c) David Nyda
Seite 4, 10 do...

Lösung: Der Code liegt in
4 fader hex version vor.
Lieber zeigten alle varianten
kein gewünschtes verhalten.

Das debuggen dauert mir
so viel zu lange...

ich stopfe es mit 7 seg einfach
in den atmega8.

ist eine der hex codes entsprechend
der ausgänge port A -- I ...
ok... (8 Bit max) für Port.

Dann stimmt der Code invertiert an
nicht.

⇒ Draufbau in eprom.

Da allerdings - zurück zum
Thema: Extremes demux bei
5 Segment Bausteinen...
nämlich

/5 per 5

02/01/2026, (c) David Vajda
Seite 2, to do

muß zum Laufen gebracht werden
als decoder...

⇒ Dies führt auch zur Besserung.
ich habe in der Vergangenheit
darüber nachgedacht: wie
soll ich an den atmega 8
mit ca. 4 ports & bis 8
14 seg anzeigen in Hülle u.
Hülle anschließen?

14 seg bedeutet 14 bit. so gesehen
natürlich kann ich decoder verwenden.
wie (b) 7447 led - zu - 7 seg.
gibt es das bei 14 seg?

ok: Lösung: (26 Buchstaben) sind
< (23) 32 und dafür reichen 5
Bit. Ok: ich denke man braucht
bei Texten, strings minimum
5 Zeichen. Lösung: eeprom:

5 Bit → 14 Bit. Das außer dem
vom außen bezogen (package)...

02/01/2026 (c), David Vajda
to do...

1.) ich wollte den Nutzern auf
facebook ausschließlich die
Duf vorstellen.

... Trotzdem werde ich heute
zu test schreiben.

ich werde wieder atmega8 benutzen

a) mit lcd hd44780

b) mit interrupt (eingabe)
pin 1 u. 2.

(bezogen auf
interrupt)

c) serielle Schnittstelle:

D - 232

1.) Transmit

2.) receive...

2.) ich sage: Der Unterschied:
es gehört in ein pdf...

3.) von dem gal und dem gescheiterten
versuch für das prom, was früher
klappte, aber auf keinen fall
routiniert... So habe ich die
14 seg anzeige nie benutzt.

02/01/2026 (c) David Vajda
Seite 3, 10 do...

kommt mit in das polf von heute...

nach: 10 eprom Bausteine nicht geringer größer hinzukommen ist zu vernachlässigen!

Es geht nicht um physio!

Es geht darum: Mit einem einfachen atmega8(mü) alleine müsste man ein sehr schneller switching...

✓ Kathodenstrahl

✓ demultiplexing

für alle 14 seg zur verfügung stellen.

Anmerkung:

Dies bringt mich auf eine idee:

ich habe ja versucht: Den

codierer / decodierer im eprom

diesmal unter zu bringen.

Nur: Dies mal taten die segmente
früher nicht so wie sie sollten

0210112026 (1) Jand vryda
Seite 5, to do...

würden die anzeigen sehr dunkel...

1/2 per 2 also zwei 7-seg
hatte ich schon. Sie wurden
dunkler, 15 allerdings wird
es sein +

Verwendung von EPV
Zeit.

man braucht einen extra...

ok.
aber weil ausgang:

8 Bit und 5

Zeichen brauche ich 10

Bausteine... habe ich

neben lauter neuen ohne UV-Löschen
eeprom... als eeprom... 27xxx?

ok

02/01/2026, (c) David Veyda
Seite 7, to do...

automobilbranche interessiert:
wie?

ander als viele aktionäre:

mit arbeit...
einem (produktionsmittel...)
prunkotisch - gegenspieler zum geld.

Kurz: Die prognosen für aktionäre
gehen nicht nur vom aktienmarkt
sollt aus: Zitat: Rainer Baule:
erwartungswert:

$$E(X) = \sum p(X=x_i) \cdot x_i$$

bei aktien:

Regenschirm

Spiess eis

Sonne / Kälte : 50:50 yes..

lllo

$p_1, p_2, p_3, \dots, p_n$
könnte extrem lang werden

02/01/2026 (c) David
Vajda, Seite 9, to do...
im Bereich Mathe lin. ab...

und was weiter...
gerade hier beginnt ein voll-
kommen neues interessantes Thema:
an, was schon längst begonnen
hat...

- 1.) assembler im Bereich (mass)
and, mmæ...
 - 2.) matrixmultiplication
 - 3.) integralrechnung...
 - 4.) Wahrscheinlichkeitsrechnung.
- nach dem letzten eingeschränkten
produktionsverhalten in diesem
Bereich, was keines war:

✓Gleichzeitige staatliche Salotage
mit glimpflicher auzauf
womöglich: austauschen
folgenden Beziehungen -
hoffen wir?
Vaubau des wirtschaftlichen

02/01/2026 (c) David
Vajda... Seite 11, 10 do...

möchte ich)

1.) besonders auf aufgaben
matrizenmultiplikation
wieder zurückschreiten...

Es ist das wichtigste Anliegen über-
haupt zielführend.

Wer es auf Zertifikate
im Bereich Mathematik
anlegt.

Mathe. Hilfering und wenn auch
mit 4 Bestandteilen aber ohne viel
Kenntnis - nur mit mit matrize-
sollte ständig üben: mathematische
grundlagen...

ok:

ich habe programme zum wunder-
schönen üben im Bereich matrize-
multiplikation für

pdf...

Das ist perfekt!!!

Zum matrize mul üben!

02/01/2026 (c) Janina Vojta
Seite 13, 10 do...

gut +

✓ mathematisches Thema:

Logarithmus Rechnung
sehr wichtig!

wir brauchen das bei:

dB ... dezibel

generell: Verhältnisgleichung:

in val

out val

o. umgekehrt.

darauf angewandt:

$$\log_{10} = l$$

lg ist entweder
 $10 \log$ oder ld ..

$\log_2$

$\log_{10}$

$\log_e \leftarrow \ln$...

ok: und bei Shannon... nur:
wenn man Logarithmus

01/31/2026

(c) David Uyda
China Zhongguoren

中国人 Seite 1

中 中

女 女 女 女 女 女 女

马 马 马 麻 麻 麻

木 木 木 木 木 木 木

木 木 天 马 马 麻

马 马 马 妈 土 土 土

王 王 王 口 口 口

古 古 土 王 王

你 你 你 你 明 明

月 Mond 月 日 我 你

乚 乚 乚 乚 我 你

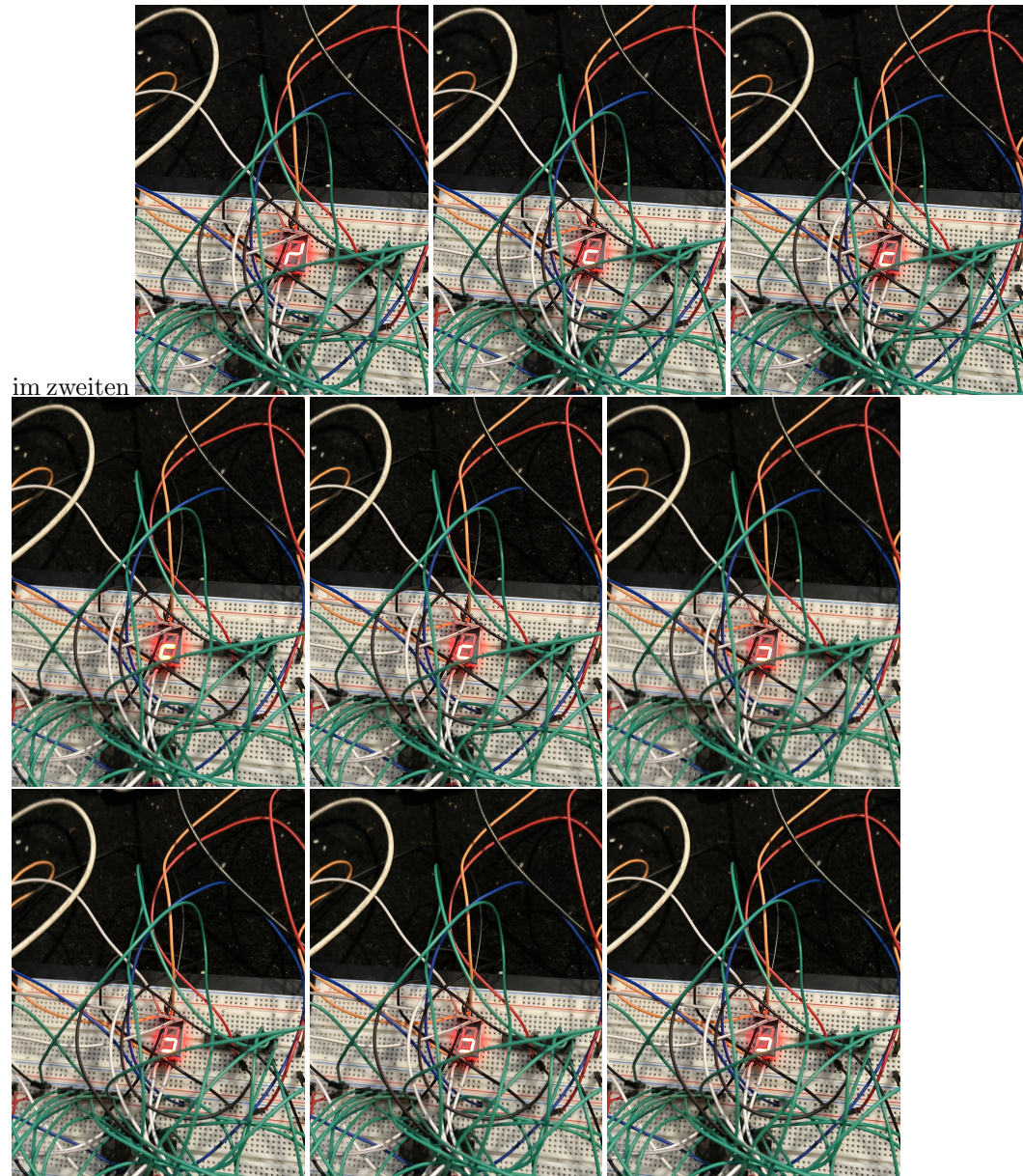
好 好 好

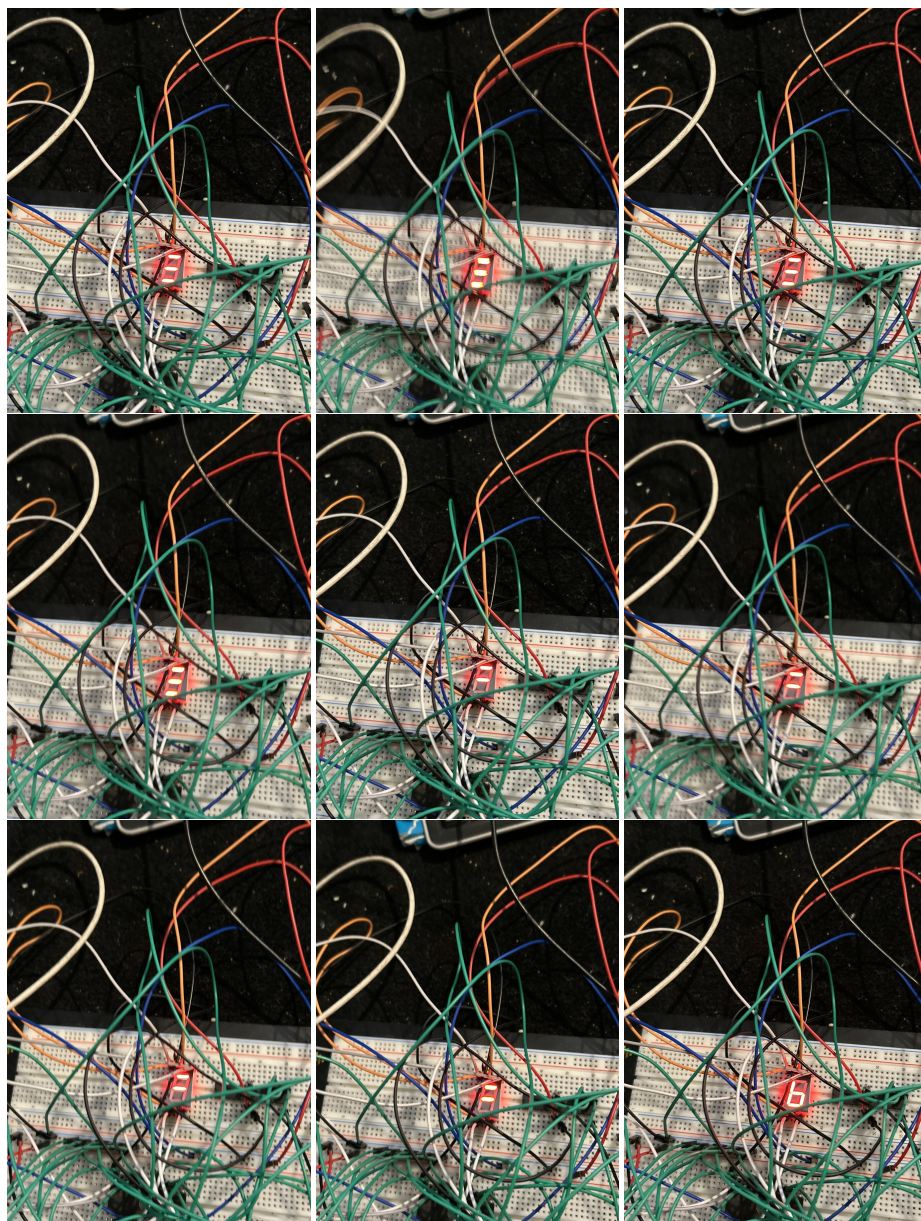
01/37/2026

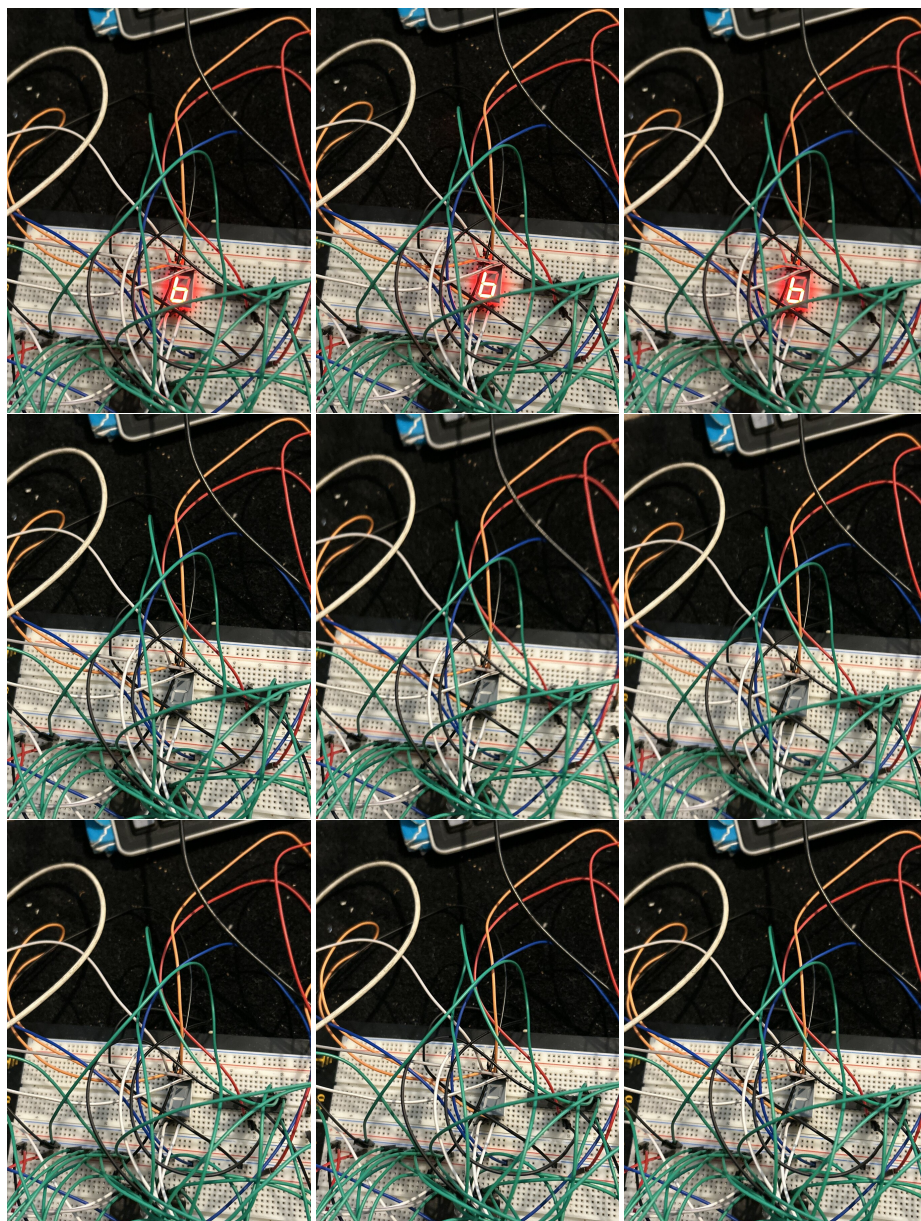
Seite 1 arabisch je 9

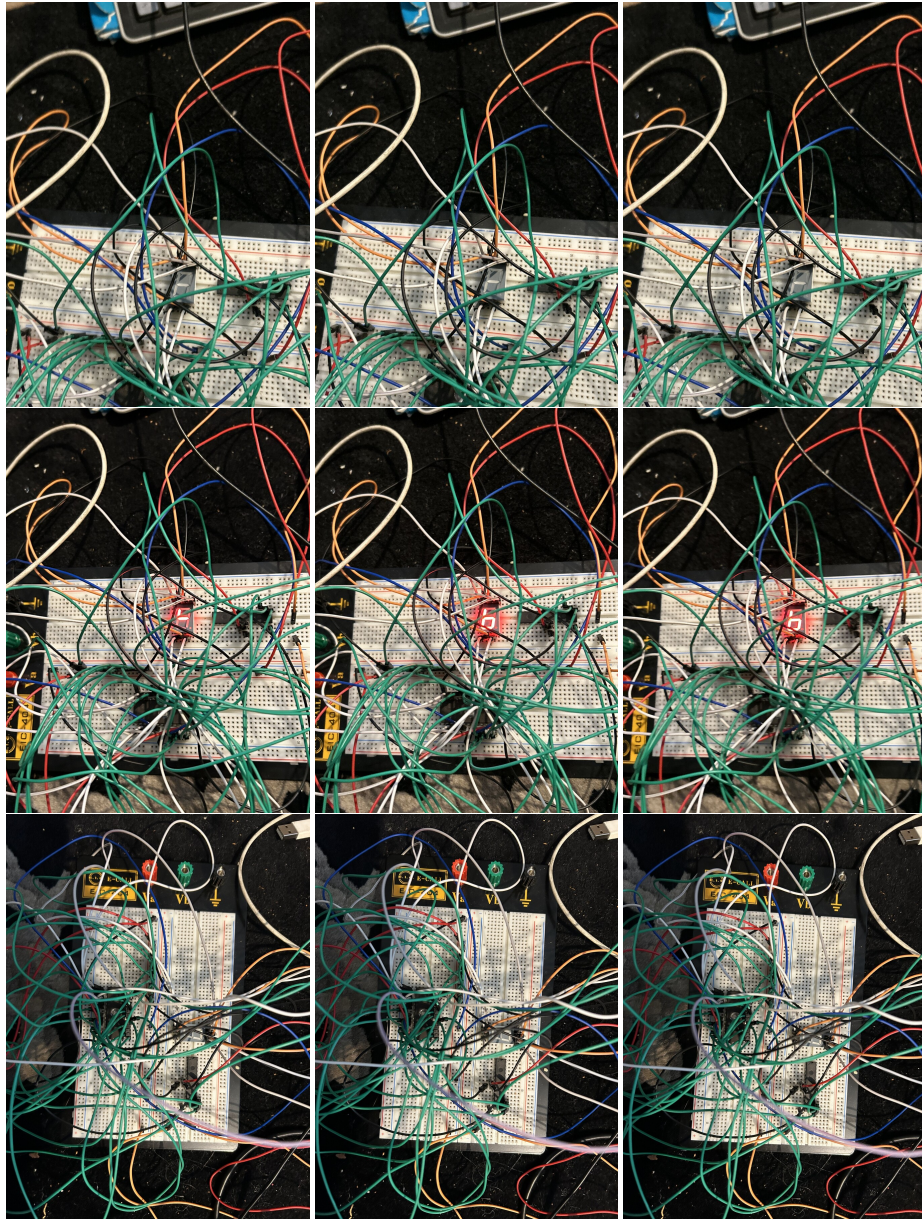
[illegible]

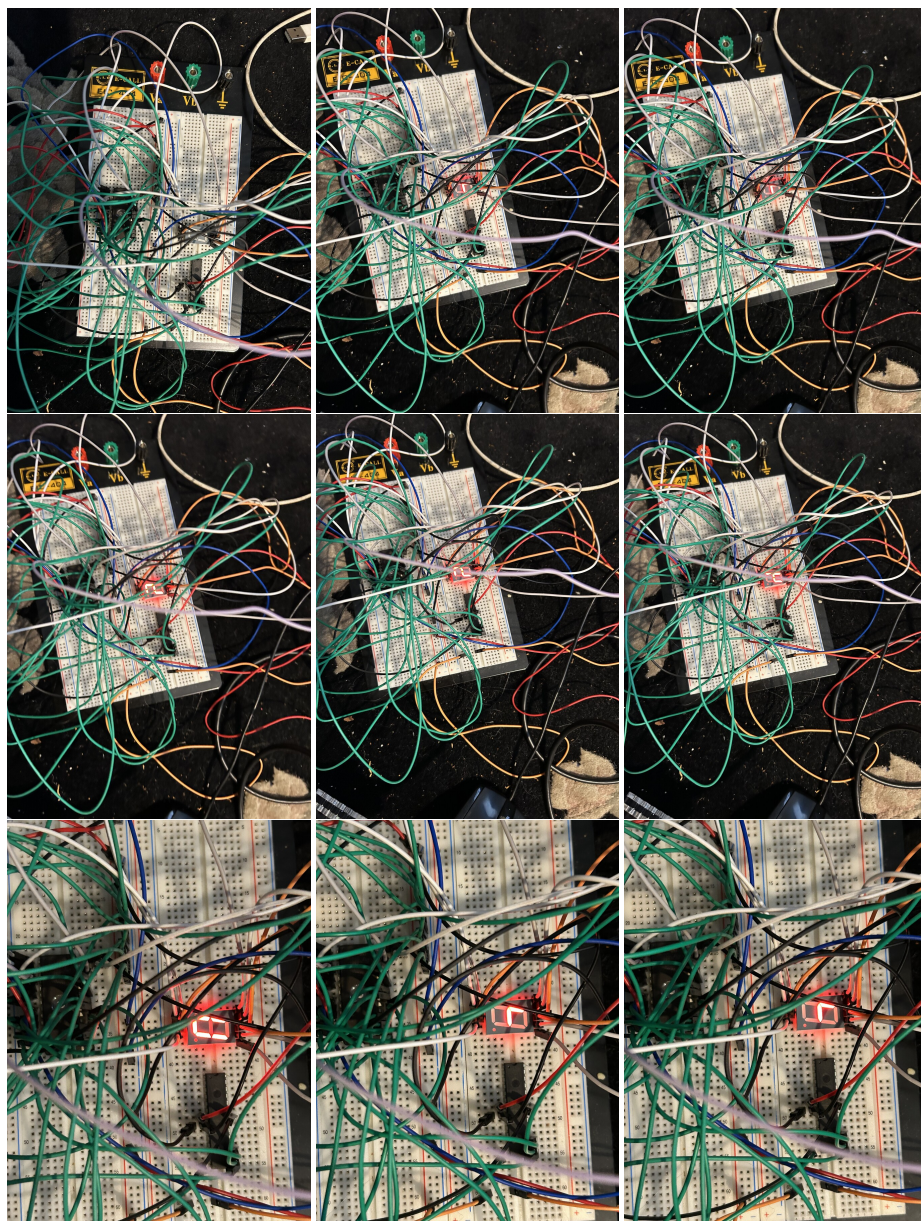
2 zweiter

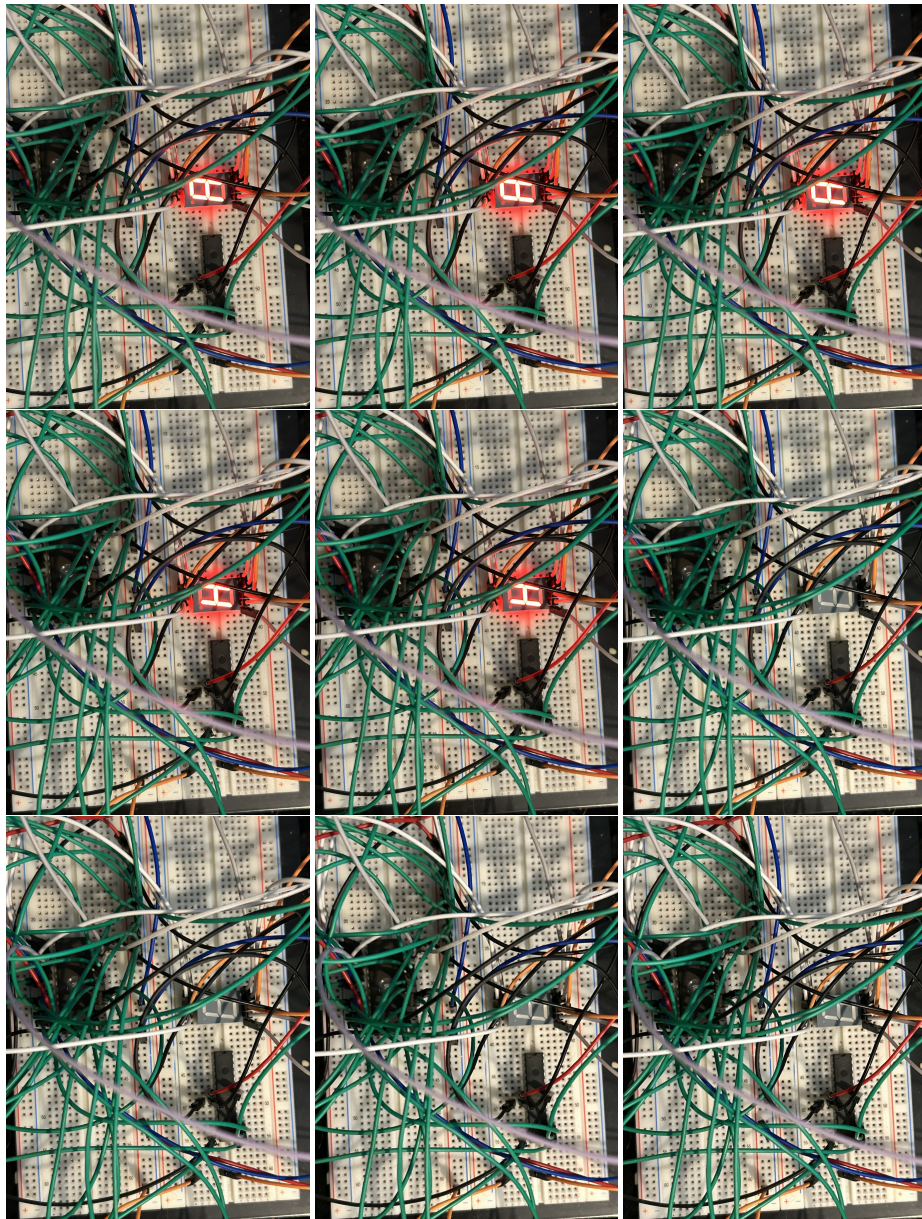


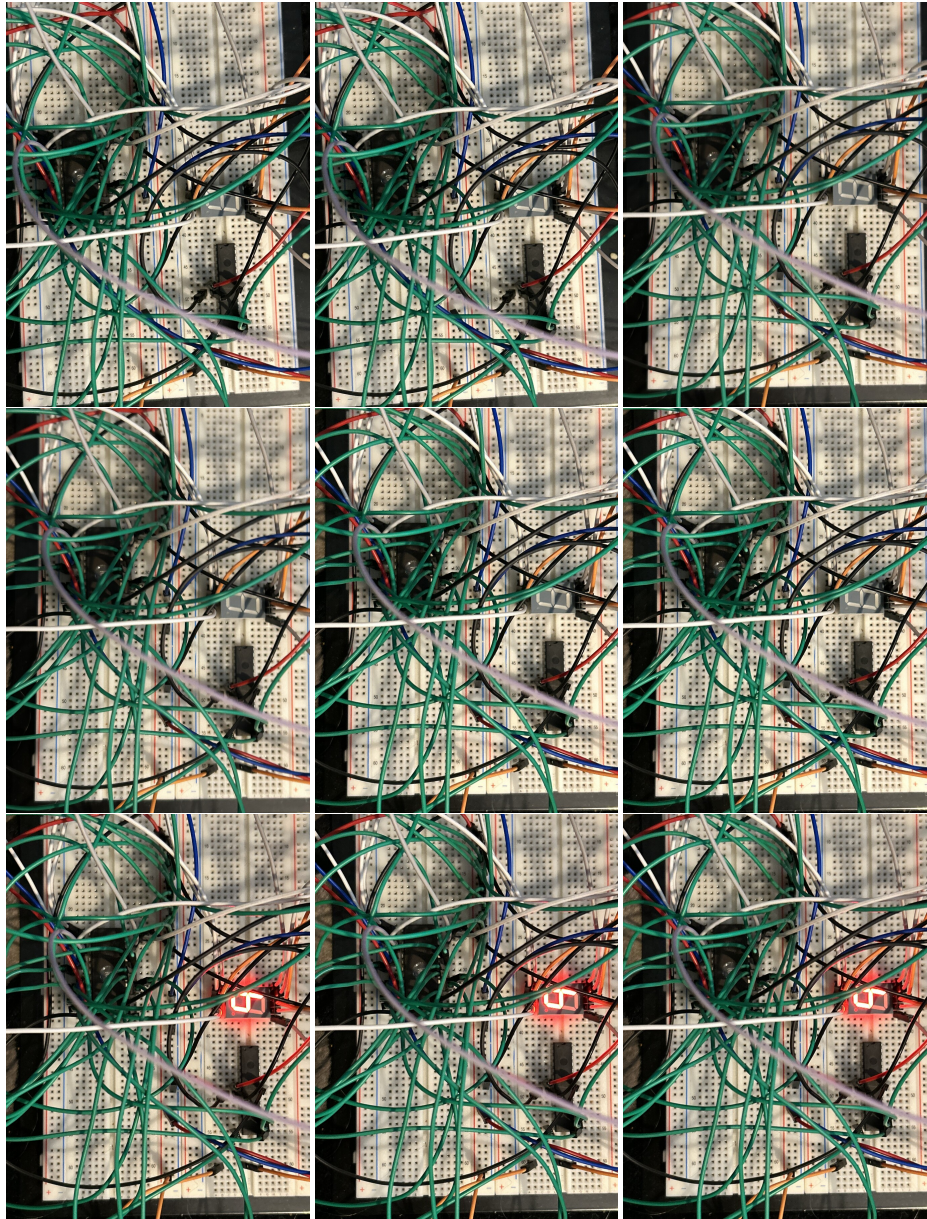


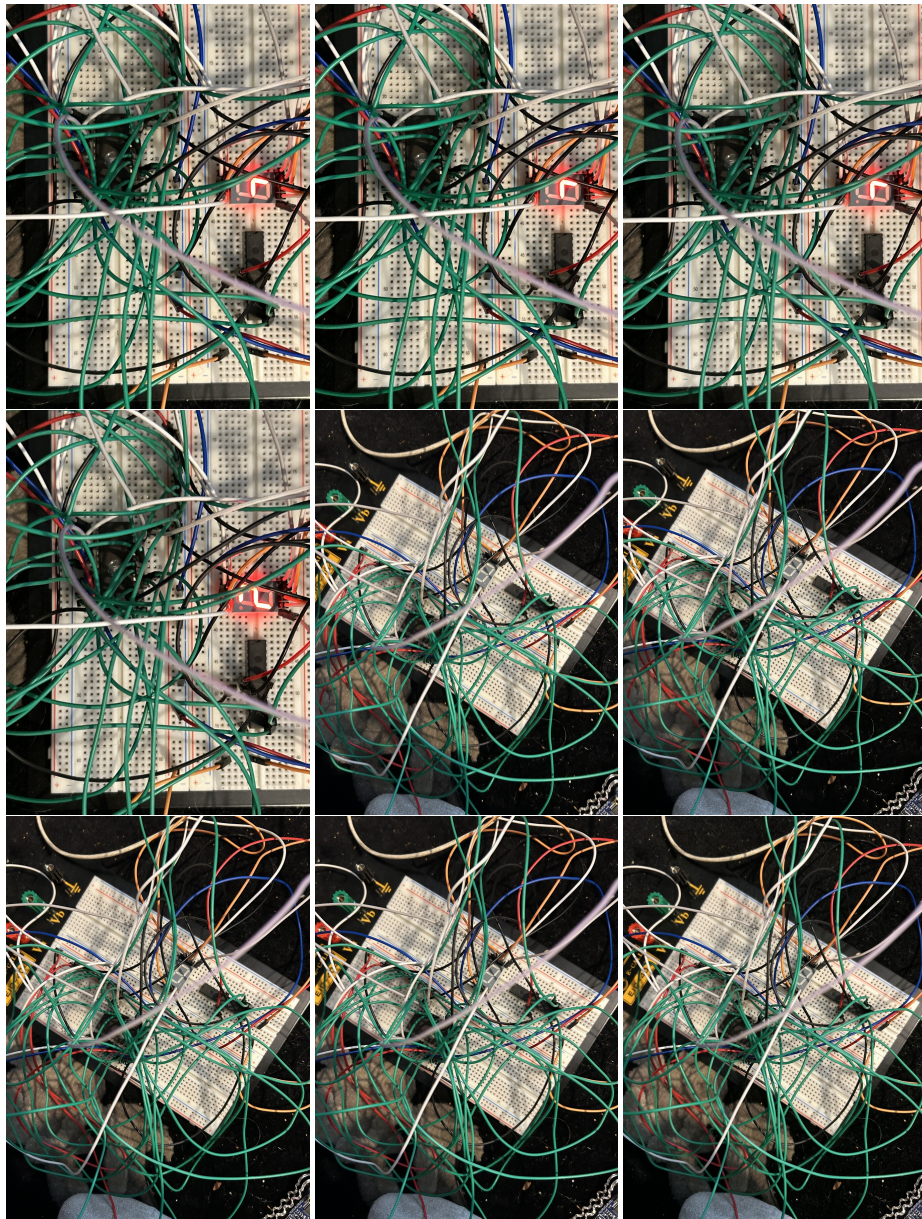


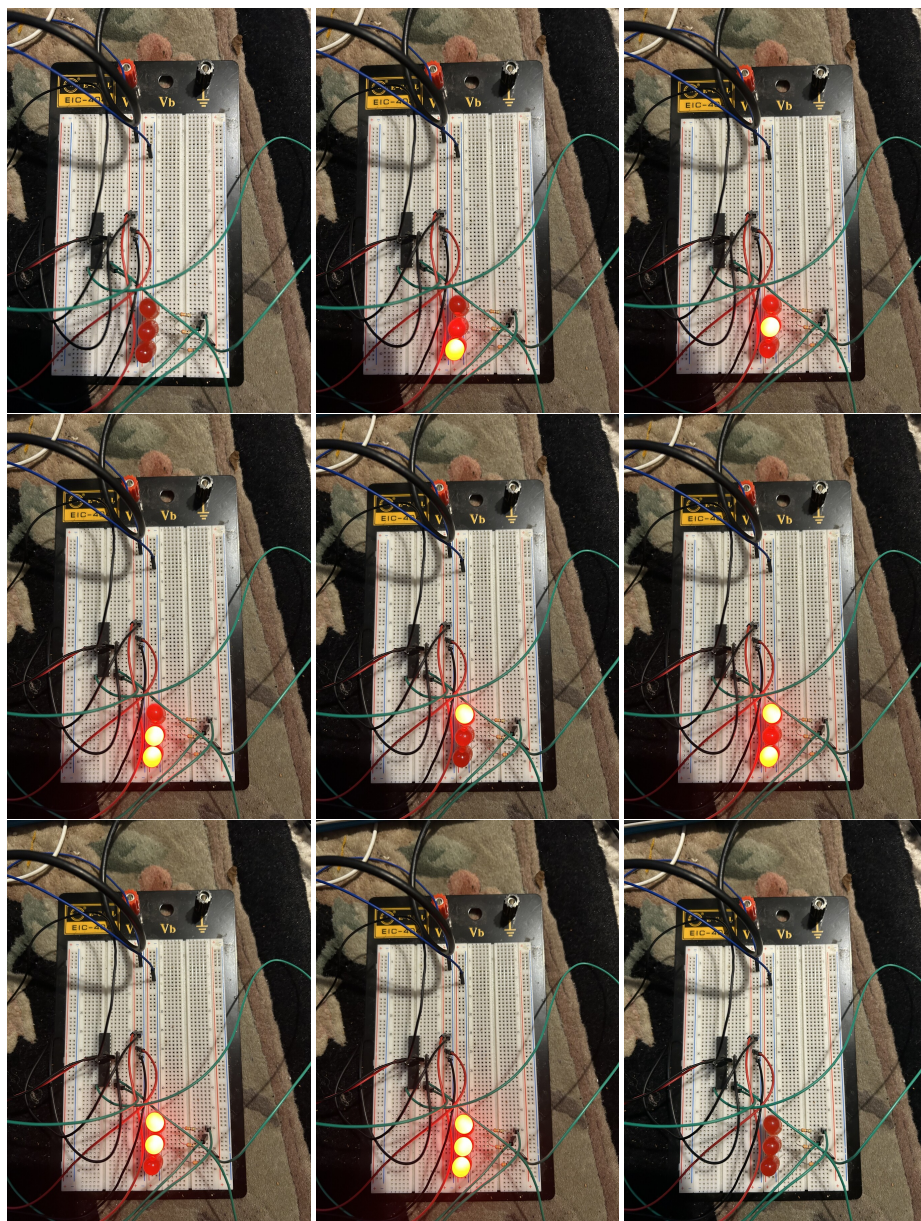


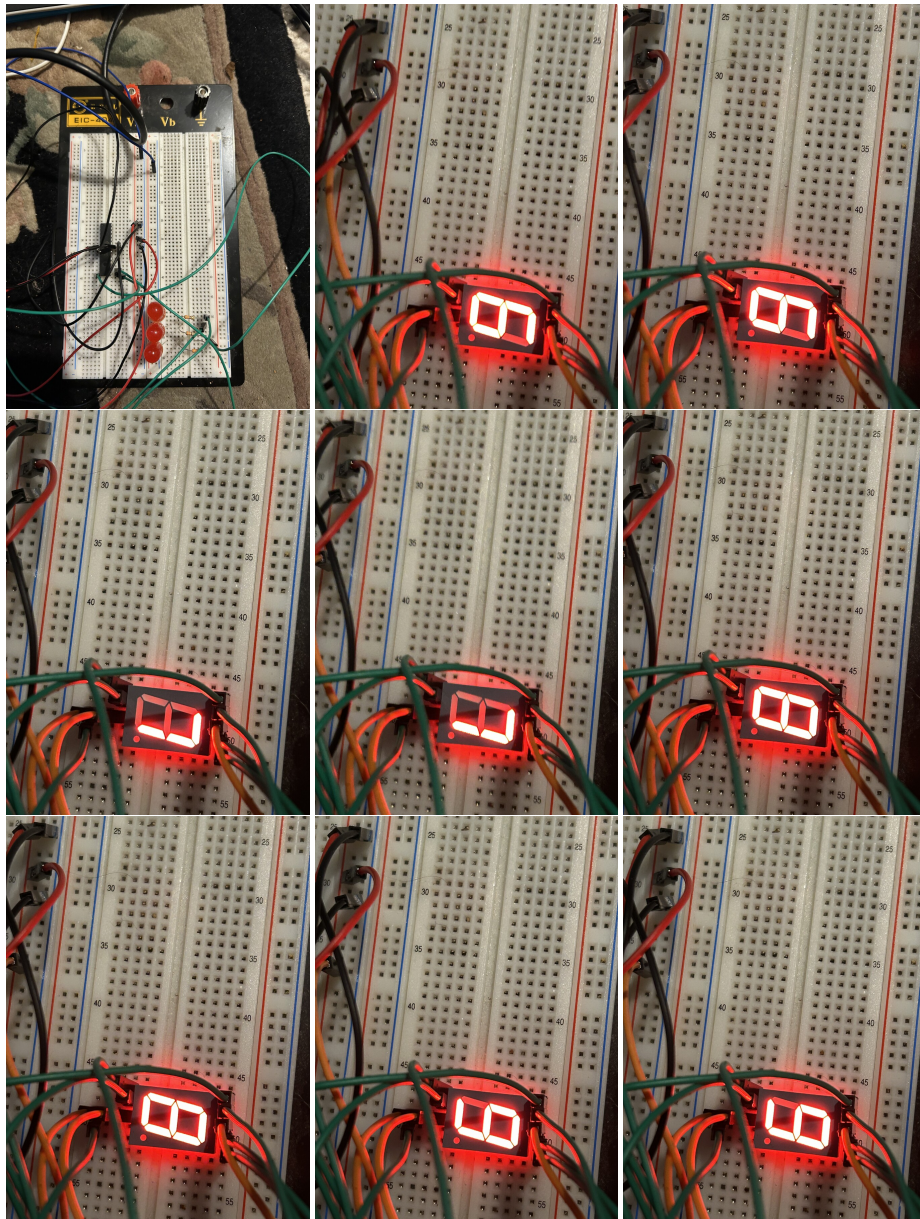


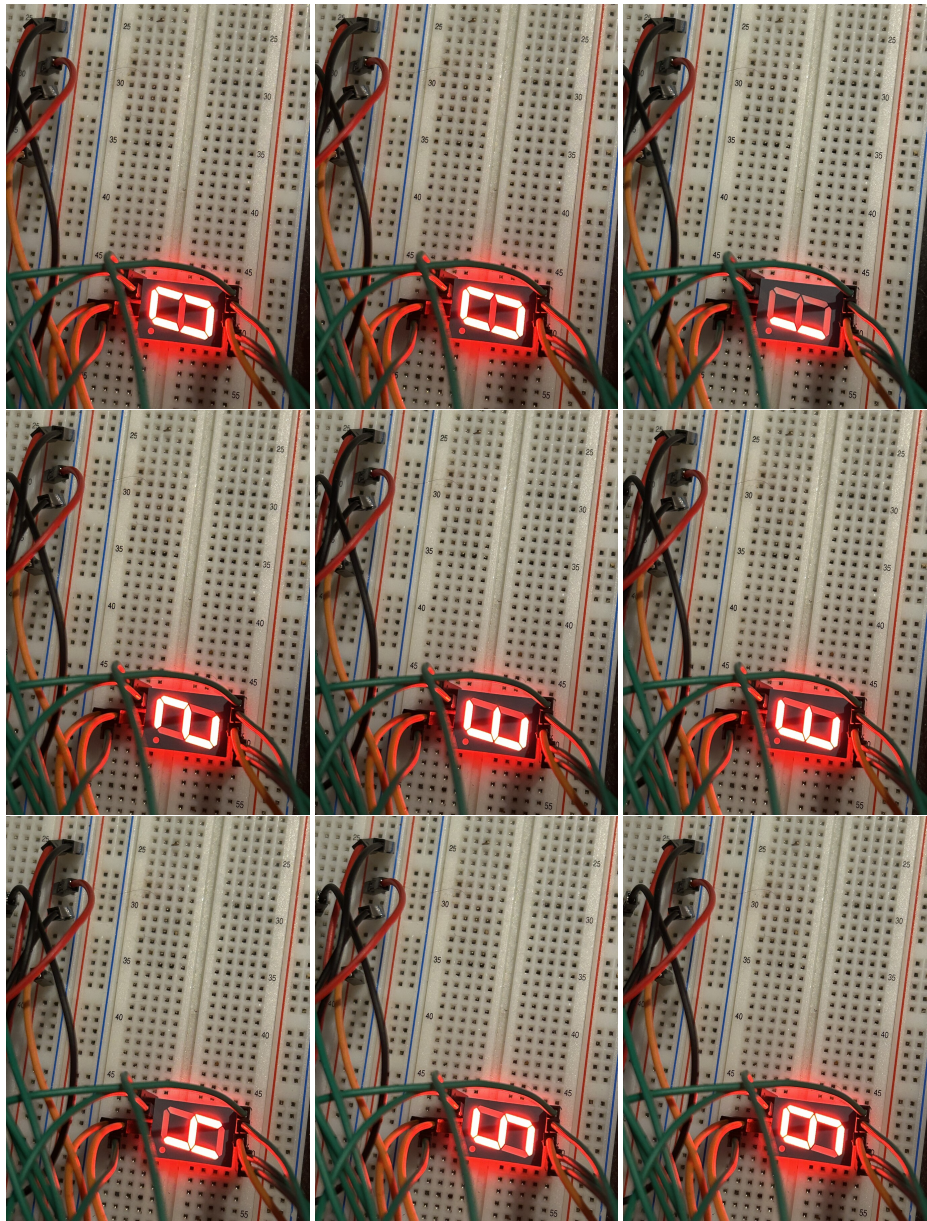


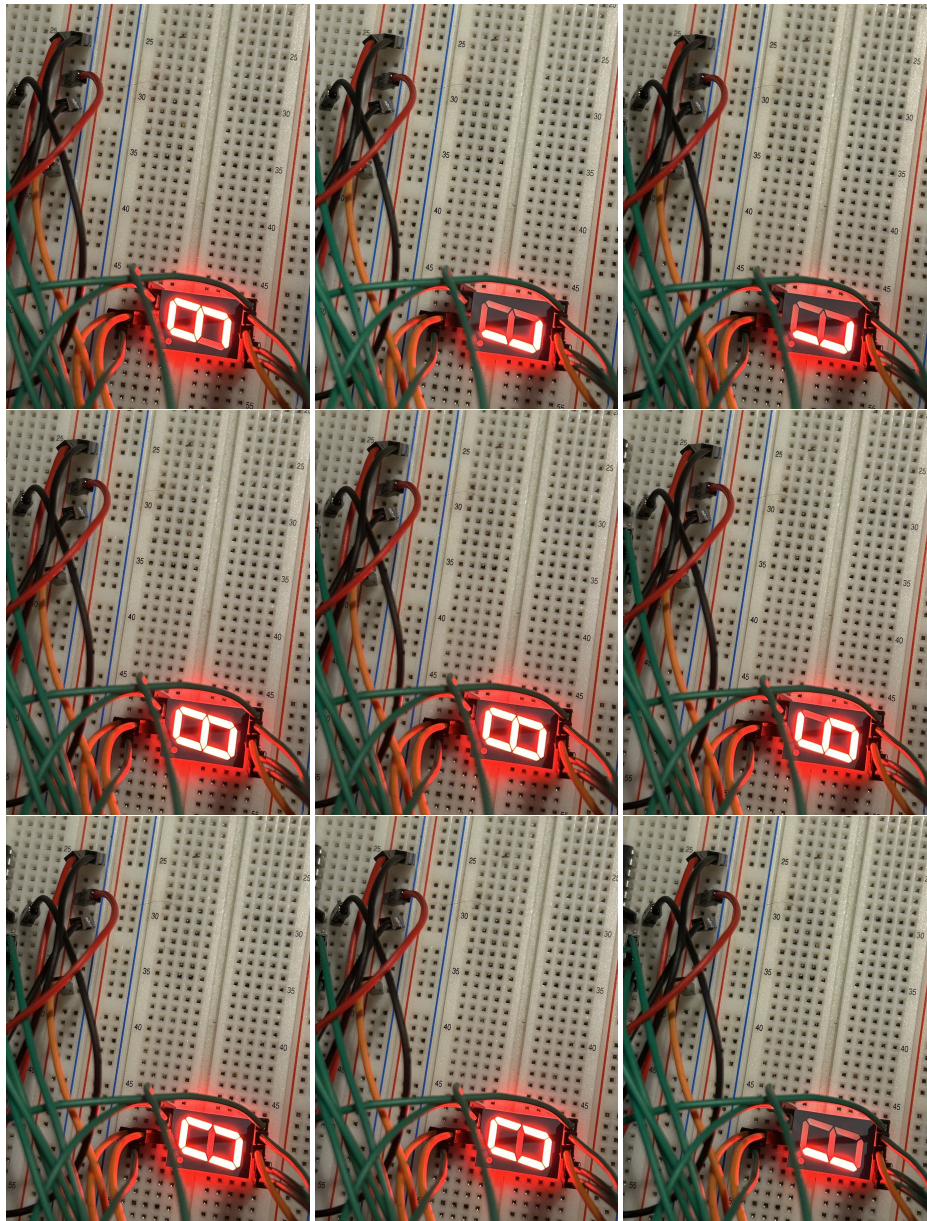


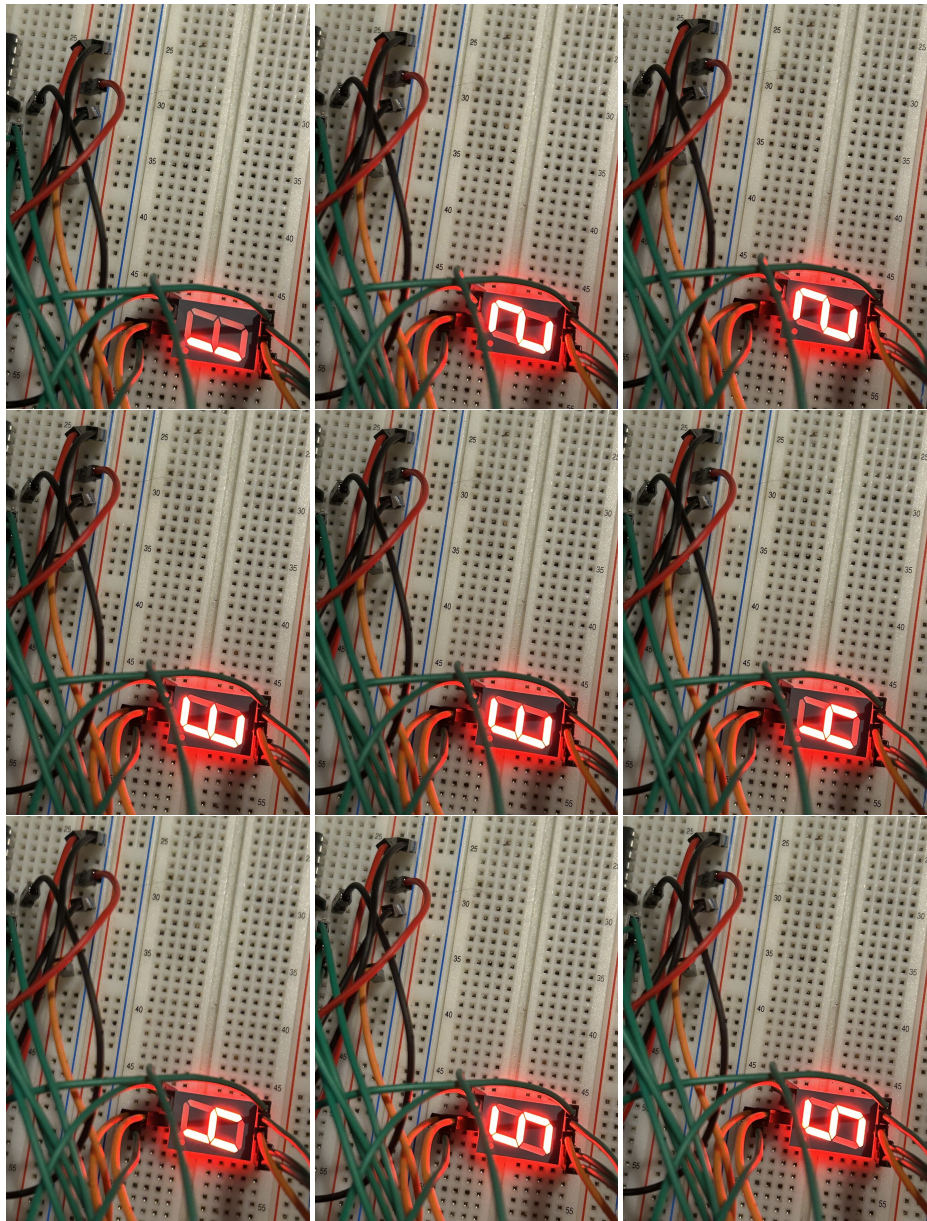


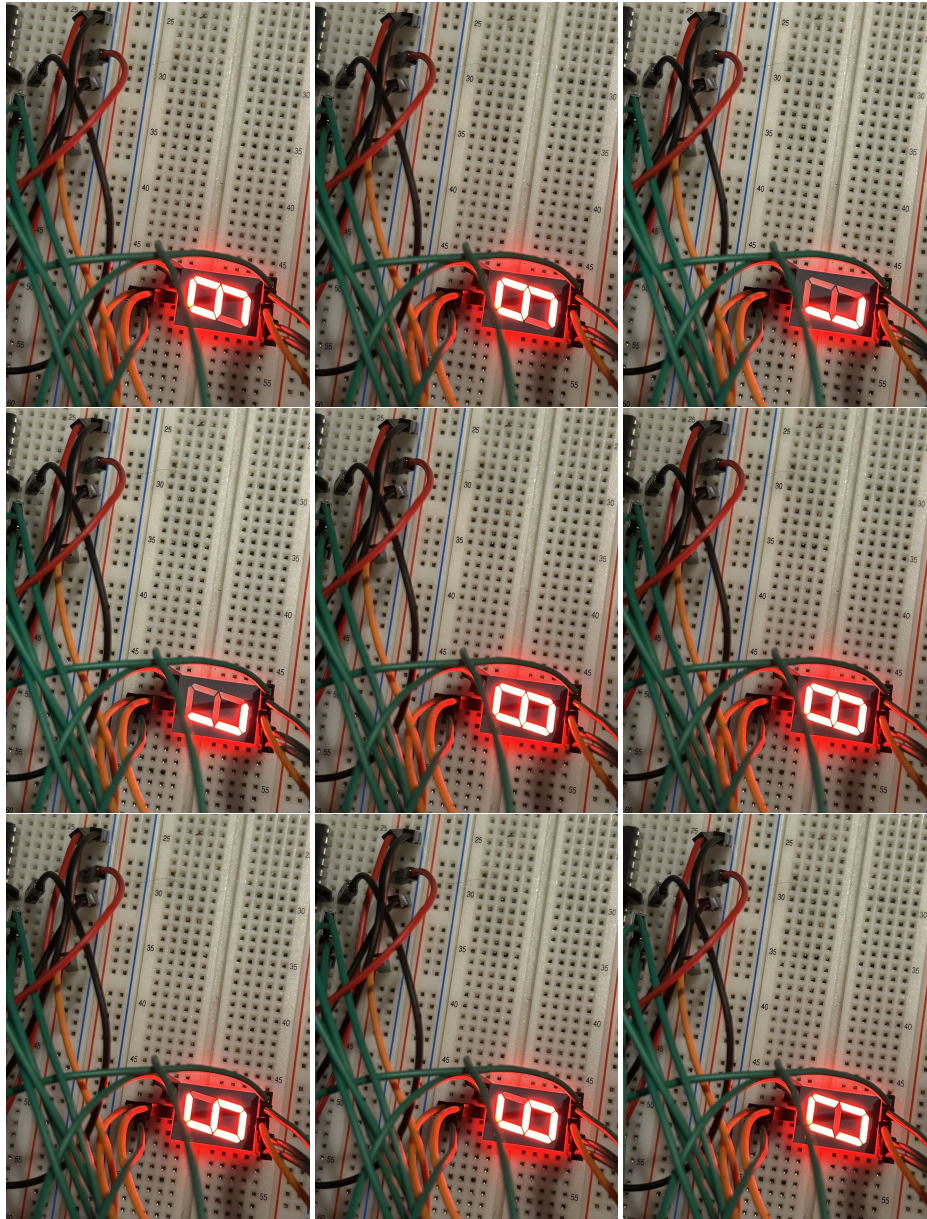


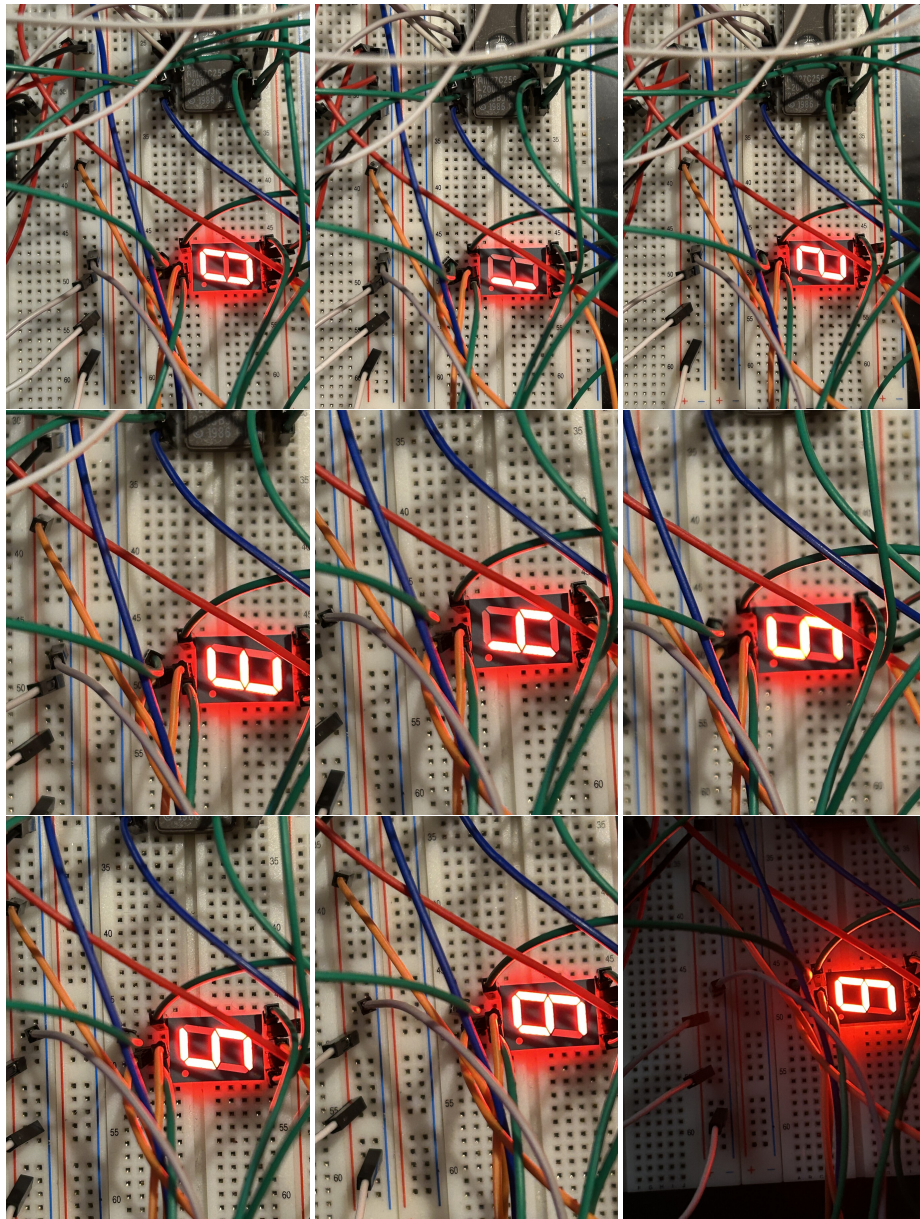














```
;; (c) david vajda  
;; 02/01/2026  
;; 7 seg atmega8 ... testing 4 eeprom
```

```
.include "m8def.inc"
```

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

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

```
l1:  
ldi r17, 0x08  
ldi XH, HIGH (decoded7seg)  
ldi XL, LOW (decoded7seg)  
l2:  
ld r16, X+  
out PORTB, r16  
rcall sleep  
dec r17  
brne l2  
rjmp l1
```

```
sleep:  
push r16  
push r17  
push r18  
ldi r18, 0xff  
s3:
```

```

ldi r17, 0xff
s2:
ldi r16, 0x04
s1:
dec r16
brne s1
dec r17
brne s2
dec r18
brne s3
pop r18
pop r17
pop r16
ret

; led-test

        decoded7seg:      .db 0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07

        ;decoded7seg:      .db    0x3f,0xC0,0x01,0x7E,    0x06,0xF9,0x4F,0x30,    0x5b,0xA4,0x12,0x6D

;; (c) david vajda
;; 02/01/2026
;; 7 seg atmega8 ... testing 4 eeprom

.include "m8def.inc"
.org 0x0000

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

ldi r16, 0xff
out DDRB, r16

l1:
ldi r17, 0x0f
ldi ZH, HIGH (decoded7seg*2)
ldi ZL, LOW (decoded7seg*2)
l2:
lpm r16, Z+
out PORTB, r16
rcall sleep
dec r17
brne l2
rjmp l1

```



```

sleep:
push r16
push r17
push r18
ldi r18, 0xff
s3:
ldi r17, 0xff
s2:
ldi r16, 0x04
s1:
dec r16
brne s1
dec r17
brne s2
dec r18
brne s3
pop r18
pop r17
pop r16
ret

```

```
; led-test
```

```
    decoded7seg:    .dw 0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07
```

```
    ;decoded7seg:    .db    0x3f,0xC0,0x01,0x7E,    0x06,0xF9,0x4F,0x30,    0x5b,0xA4,0x12,0x6D
```

```
;; (c) david vajda
```

```
;; 02/01/2026
```

```
;; 7 seg atmega8 ... testing 4 eeprom
```

```
.include "m8def.inc"
```

```
.org 0x0000
```

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

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

```
l1:
ldi r17, 0x14
ldi ZH, HIGH (decoded7seg*2)
```

```

ldi ZL, LOW (decoded7seg*2)
l2:
lpm r16, Z+
out PORTB, r16
rcall sleep
dec r17
brne l2
rjmp l1

```

```

sleep:
push r16
push r17
push r18
ldi r18, 0xff
s3:
ldi r17, 0xff
s2:
ldi r16, 0x04
s1:
dec r16
brne s1
dec r17
brne s2
dec r18
brne s3
pop r18
pop r17
pop r16
ret

```

```

; led-test

```

```

;decoded7seg: .dw 0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07

```

```

decoded7seg: .dw 0xC0, 0xF9, 0xA4, 0xB0, 0x99, 0x92, 0x82, 0xF8, 0x80, 0x90

```

```

;; (c) david vajda
;; 02/01/2026
;; 7 seg atmega8 ... testing 4 eeprom

```

```

#include "m8def.inc"
.org 0x0000

```

```

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

```

```

ldi r16, 0xff

```



```
out DDRB, r16
```

```
l1:
ldi r17, 0x0a
ldi ZH, HIGH (decoded7seg*2)
ldi ZL, LOW (decoded7seg*2)
l2:
lpm r16, Z+
out PORTB, r16
rcall sleep
dec r17
brne l2
rjmp l1
```

```
sleep:
push r16
push r17
push r18
ldi r18, 0xff
s3:
ldi r17, 0xff
s2:
ldi r16, 0x04
s1:
dec r16
brne s1
dec r17
brne s2
dec r18
brne s3
pop r18
pop r17
pop r16
ret
```

```
; led-test
```

```
    ;decoded7seg:    .dw 0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07
```

```
    decoded7seg:    .db  0xC0, 0xF9, 0xA4, 0xB0, 0x99, 0x92, 0x82, 0xF8, 0x80, 0x90
```

```
;; (c) david vajda
;; 02/01/2026
;; 7 seg atmega8 ... testing 4 eeprom
```

```
.include "m8def.inc"
```

```

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

```

```

ldi r16, 0xff
out DDRB, r16

```

```

l1:
ldi r17, 0x08
ldi XH, HIGH (decoded7seg)
ldi XL, LOW (decoded7seg)
l2:
ld r16, X+
out PORTB, r16
rcall sleep
dec r17
brne l2
rjmp l1

```

```

sleep:
push r16
push r17
push r18
ldi r18, 0xff
s3:
ldi r17, 0xff
s2:
ldi r16, 0x04
s1:
dec r16
brne s1
dec r17
brne s2
dec r18
brne s3
pop r18
pop r17
pop r16
ret

```

```

; led-test

```

```

    decoded7seg:    .db 0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07

```



```

        ;decoded7seg:      .db    0x3f,0xC0,0x01,0x7E,   0x06,0xF9,0x4F,0x30,   0x5b,0xA4,0x12,0x6D

#include <stdio.h>

// (c) david vajda
// 01/29/2026
// eprom 27256, 256k

void main (void) {
    char bytes [] = {0xff, 0x00, 0x00, 0x00, 0x00, 0x0f, 0x00, 0x00};
    int i = 0;

    fprintf (stderr, "(c) david vajda\n");
    fprintf (stderr, "01/29/2026\n");
    fprintf (stderr, "eprom 27256, 256k\n");
    while (i < 8) {
        printf ("%c%c", bytes [i+1], bytes [i]);
        i = i + 2;
    }

    return;
}

#include <stdio.h>

// (c) david vajda
// 01/29/2026
// eprom 27256, 256k

void main (void) {
    char bytes [] = {0xff, 0x00, 0x00, 0x00, 0x00, 0x0f, 0x00, 0x00};
    int i = 0;

    fprintf (stderr, "(c) david vajda\n");
    fprintf (stderr, "01/29/2026\n");
    fprintf (stderr, "eprom 27256, 256k\n");
    while (i < 8) {
        printf ("%c%c", bytes [i+1], bytes [i]);
        i = i + 1;
    }

    return;
}

#include <stdio.h>

// (c) david vajda
// 01/29/2026
// eprom 27256, 256k

void main (void) {

```

```

char bytes [] = {0xff, 0x00, 0x00, 0x00, 0x00, 0x0ff, 0x00, 0x00};
int i = 0;

fprintf (stderr, "(c) david vajda\n");
fprintf (stderr, "01/29/2026\n");
fprintf (stderr, "eprom 27256, 256k\n");
while (i < 8) {
    printf ("%c", bytes [i]);
    i = i + 1;
}

return;
}

#include <stdio.h>

// (c) david vajda
// 01/30/2026
// eeprom - 7 seg -

int main (void) {
#ifdef NOTINVERTED
    int seg7code [] = {0x3f, 0x06, 0x5b, 0x4f, 0x66, 0x6d, 0x7d, 0x07, 0x7f, 0x6f};
#else
    int seg7code [] = {0xC0, 0xF9, 0xA4, 0xB0, 0x99, 0x92, 0x82, 0xF8, 0x80, 0x90};
#endif

    int i = 0;
    while (i < 10) {
        printf ("%c", seg7code[i]);
        i = i + 1;
    }
    return 0;
}

#include <stdio.h>

// (c) david vajda
// 01/30/2026
// eeprom - 7 seg -

int main (void) {
    int seg7code [] = {0x3f, 0x06, 0x5b, 0x4f, 0x66, 0x6d, 0x7d, 0x07, 0x7f, 0x6f};

    int i = 0;
    while (i < 8) {
        printf ("%c", seg7code[i]);
        i = i + 1;
    }
}

```



```

return 0;
}

#include <stdio.h>

// (c) david vajda
// 01/31/2026
// eeprom - 7 seg -

int main (void) {
    int seg7code [] = {0x3f,0xC0,0x01,0x7E, 0x06,0xF9,0x4F,0x30, 0x5b,0xA4,0x12,0x6D, 0x00,0x00,0x00,0x00};
    int i;
    int j;

    j = 0;
    while (j < 4) {
        i = j;
        while (i < 64) {
            printf ("%c", seg7code[i]);
            i = i + 4;
        }
        j = j + 1;
    }
    return 0;
}

#include <stdio.h>

// (c) david vajda
// 02/01/2026
// eeprom - 7 seg -

int main (void) {
    int seg7code [] = {0xC0, 0xF9, 0xA4, 0xB0, 0x99, 0x92, 0x82, 0xF8, 0x80, 0x90, 0xC0, 0x00, 0x00, 0x00, 0x00, 0x00};
    int i;
    int j;

    j = 0;
    while (j < 128) {
        i = 0;
        while (i < 16) {
            printf ("%c", seg7code[i]);
            i = i + 1;}
        j = j + 1;}
    return 0;
}

```

00111111	11000000	0xC0
00000110	11111001	0xF9
01011011	10100100	0xA4
01001111	10110000	0xB0
01100110	10011001	0x99
01101101	10010010	0x92
01111101	10000010	0x82
00000111	11111000	0xF8
01111111	10000000	0x80
01101111	10010000	0x90

00111111	0x3f	01000000	0xC0	00000001	0x01	01111110	0x7E
00000110	0x06	01111001	0xF9	01001111	0x4F	00110000	0x30
01011011	0x5b	00100100	0xA4	00010010	0x12	01101101	0x6D
01001111	0x4f	00110000	0xB0	00000110	0x06	01111001	0x79
01100110	0x66	00011001	0x99	01001100	0x9C	00110011	0x33
01101101	0x6d	00010010	0x92	00100100	0x48	01011011	0x5B
01111101	0x7d	00000010	0x82	00100000	0x20	01011111	0x5F
00000111	0x07	01111000	0xF8	00001111	0x0F	01110000	0x70
01111111	0x7f	00000000	0x80	00000000	0x00	01111111	0x7F
01101111	0x6f	00010000	0x90	00000100	0x04	01111011	0x7B
00000000	0x00	00000000	0x00	00000000	0x00	00000000	0x00
00000000	0x00	00000000	0x00	00000000	0x00	00000000	0x00
00000000	0x00	00000000	0x00	00000000	0x00	00000000	0x00
00000000	0x00	00000000	0x00	00000000	0x00	00000000	0x00
00000000	0x00	00000000	0x00	00000000	0x00	00000000	0x00
00000000	0x00	00000000	0x00	00000000	0x00	00000000	0x00
00000000	0x00	00000000	0x00	00000000	0x00	00000000	0x00

3 m87seg3.eep.hex

```
0000000 303a 3030 3030 3030 4631 0d46 000a
000000d
```

4 m87seg3.hex

```
0000000 303a 3032 3030 3030 3032 3030 4630 0d43
0000010 3a0a 3031 3030 3030 3030 3430 3045 4530
0000020 4642 4630 3545 4430 4642 4630 4645 3730
0000030 4242 4631 3045 3046 3045 3046 0a0d 313a
0000040 3030 3130 3030 4530 4545 3033 3935 3031
0000050 4238 3042 4433 3130 3941 4435 4639 4637
0000060 4337 3046 3946 4633 0d43 3a0a 3031 3030
0000070 3032 3030 4631 3339 4632 3339 4632 4645
0000080 4631 4645 3430 3045 4130 3539 3146 3746
0000090 4131 3539 3631 0a0d 313a 3030 3330 3030
00000a0 4430 4639 3237 3941 4335 4631 3237 3946
```

```

00000b0 3131 3946 3031 3946 3031 3938 3035 3030
00000c0 4330 0d43 3a0a 4530 3030 3034 3030 3130
00000d0 3030 3230 3030 3330 3030 3430 3030 3530
00000e0 3030 3630 3030 3730 3030 3639 0a0d 303a
00000f0 3030 3030 3030 4631 0d46 000a
00000fb

```

5 m87seg4.eep.hex

```

0000000 303a 3030 3030 3030 4631 0d46 000a
000000d

```

6 m87seg4.hex

```

0000000 303a 3032 3030 3030 3032 3030 4630 0d43
0000010 3a0a 3031 3030 3030 3030 3430 3045 4530
0000020 4642 4630 3545 4430 4642 4630 4645 3730
0000030 4242 3431 3145 3046 3045 4146 0a0d 313a
0000040 3030 3130 3030 4530 4545 3033 3935 3031
0000050 4238 3042 4433 3130 3941 4435 4639 4637
0000060 4337 3046 3946 4633 0d43 3a0a 3031 3030
0000070 3032 3030 4631 3339 4632 3339 4632 4645
0000080 4631 4645 3430 3045 4130 3539 3146 3746
0000090 4131 3539 3631 0a0d 313a 3030 3330 3030
00000a0 4430 4639 3237 3941 4335 4631 3237 3946
00000b0 3131 3946 3031 3946 3031 3938 4335 3030
00000c0 3030 0d43 3a0a 3031 3030 3034 3030 3946
00000d0 3030 3441 3030 3042 3030 3939 3030 3239
00000e0 3030 3238 3030 3846 3030 3038 3030 4533
00000f0 0a0d 303a 3032 3530 3030 3930 3030 3130
0000100 0d45 3a0a 3030 3030 3030 3130 4646 0a0d
0000110

```

7 m87seg5.eep.hex

```

0000000 303a 3030 3030 3030 4631 0d46 000a
000000d

```

8 m87seg5.hex

```

0000000 303a 3032 3030 3030 3032 3030 4630 0d43
0000010 3a0a 3031 3030 3030 3030 3430 3045 4530
0000020 4642 4630 3545 4430 4642 4630 4645 3730

```



```

0000030 4242 4131 3045 3046 3045 3546 0a0d 313a
0000040 3030 3130 3030 4530 4545 3033 3935 3031
0000050 4238 3042 4433 3130 3941 4435 4639 4637
0000060 4337 3046 3946 4633 0d43 3a0a 3031 3030
0000070 3032 3030 4631 3339 4632 3339 4632 4645
0000080 4631 4645 3430 3045 4130 3539 3146 3746
0000090 4131 3539 3631 0a0d 313a 3030 3330 3030
00000a0 4430 4639 3237 3941 4335 4631 3237 3946
00000b0 3131 3946 3031 3946 3031 3938 4335 4630
00000c0 3139 0d33 3a0a 3830 3030 3034 3030 3441
00000d0 3042 3939 3239 3238 3846 3038 3039 4641
00000e0 0a0d 303a 3030 3030 3030 4631 0d46 000a
00000ef

```

9 m87seg.eep.hex

```

0000000 303a 3030 3030 3030 4631 0d46 000a
000000d

```

10 m87seg.hex

```

0000000 303a 3032 3030 3030 3032 3030 4630 0d43
0000010 3a0a 3031 3030 3030 3030 3430 3045 4530
0000020 4642 4630 3545 4430 4642 4630 4645 3730
0000030 4242 3831 3045 3042 3045 3733 0a0d 313a
0000040 3030 3130 3030 4130 4546 3031 3944 3033
0000050 4238 3042 4433 3130 3941 4435 4639 4637
0000060 4337 3046 3946 3333 0d33 3a0a 3031 3030
0000070 3032 3030 4631 3339 4632 3339 4632 4645
0000080 4631 4645 3430 3045 4130 3539 3146 3746
0000090 4131 3539 3631 0a0d 313a 3030 3330 3030
00000a0 4430 4639 3237 3941 4335 4631 3237 3946
00000b0 3131 3946 3031 3946 3031 3938 3035 3030
00000c0 4331 0d42 3a0a 3630 3030 3034 3030 3230
00000d0 3330 3430 3530 3630 3730 4639 0a0d 303a
00000e0 3030 3030 3030 4631 0d46 000a
00000eb

```

Im zweiten Teil habe ich versucht, also eigentlich im Ersten also zu sagen direkt gesagt,
 Aus irgendeinem Grund hat das zunächst nicht funktioniert. Ich habe verschiedenes Interess
 Für jede Form von Effekthascherei kommt es LCD dazu. Das ist für mich relativ einfach, wei
 Ich hab jetzt inzwischen bald das mit dem Controller und der Zahlenfolge nicht funktionier

Da allerdings das immer noch nicht funktioniert hat, habe ich beschlossen den Controller s

An der Stelle verstehe ich viele mit dem Intel direkter indirekter und indizierten Adress

Kann passieren. Also direkt die indirekte und indizierte Adressierung, sag ich gleich folg

Das ist jetzt einfach mal der Allgemeinheit auf dem Mund geschaut aber wir wissen nicht we

MOV Befehl, wo man ja sagt, das können wir ja inzwischen alle das Intel geschafft hat die

Typisch ist das, was sie sozusagen im Computersysteme eins und zwei finden und das ist eig

Das mit der indirekten Adressierung des verstehe ich also wie gesagt, Register Register si

Heap Stack Daten code und i/o verhält sich eigentlich ganz anders

Nämlich ungefähr so wie bei dem atmega8

So, der erste Teil die Daten sind generell im Arbeitsspeicher es gibt ein Arbeitsspeicher

Code und Daten sind getrennt und die Daten im Arbeitsspeicher. Wir können auch ganz ohne A

Also drei Speicher der Witz bei der Sache ist, dass sie sozusagen. In dem normalen Speiche

Bei den meisten Architekturen

MOV
LD
ST

Also laden und speichern heißt es und deswegen lade und Speicherarchitektur und laden heiß

Deswegen haben wir dementsprechend sozusagen 3-4 Befehle sag ich mal

ldi haben wir hier d.h. eine immediate Wert Laden d.h. eine Konstante

Aber das ist nicht Hauptziel. Normalerweise laden wir aus dem Arbeitsspeicher und wenn wir

Wir haben direkte Adressierung
Indirekte Adressierung
Indirekte Adressierung mit Auto increment
Indirekte Aggression mit Auto decrement ?

Indirekte Adressierung
Indirekte Adressierung mit Verschiebung Displacement

So, und das ist das Auto increment ist immer Post increment und des dekrement ist immer e

So, und wir haben hier , also drei Register Paare also insgesamt sechs Register, um sozusa

D.h. wenn der ein Label eine Konstante einführen, dann ist die mal an ein 2 Byte sozusagen
Haben wir wie gesagt, Laden ld aber wir haben Laden ld auch mit einem sozusagen Post,inkre
Und dazu benutzen wir die Register Paare XY und ZX lässt sich nicht immer genauso verwande
Um einen Wert zu speichern, das sind zusammen immer 16 Bit in diesen Zeiger müssen wir den

HIGH (Adresse)

Und es übliche

LOW (Adresse)

Dies müssen wir in die Register Paare laden. Dabei haben wir

YH und YL

Okay, und um sozusagen des Post increment hervorzurufen, machen wir Y+

Die Befehle gehen dann so

ldi

Für sozusagen Direktwerte

Dann LD für

X+,Y+,Z+

-X,-Y,-Z

Okay, und wenn wir allerdings mit dis Placement haben, des entspricht der indizierten Adre

Müssen wir darüber nachdenken, dass bei indiziert bei Intel heißt

Register plus Register, plus irgendeinen Wert

Okay, also direkt heißt wir geben die Adresse direkt an das ist eine Konstante

Indirekt heißt über ein Register

Und wenn wir jetzt allerdings indiziert angucken sozusagen, dann heißt das

ldd r16, Y+5

Zum Beispiel also plus einen festen Wert ohne Inkremeng und dekrement des Zeigers , was na

Also, hier haben wir jetzt sozusagen LDI für Direktwerte inmmediate oder konstanten

Was entsprechend Register Register eigentlich keine Adressierung ist

Ld sozusagen für indirekte Werte mit Inkrement oder dekrement oder gar nicht

Dann haben wir LDD sozusagen für indiziert mit Konstante

Dann kommt lds Das wiederum ist direkt d.h. hier geben wir sozusagen die Adresse nicht übe

Wir allerdings aus dem Programm Speicher nehmen das sind unsere konstanten, dann brauchen
lpm und stm

Weil hier haben wir die konstanten in unserem Flash, das ist nicht der Arbeitsspeicher der

Wir müssen nur einen Unterschied machen. Das Label muss auf eine zwei Bytegrenze ausgerich

```
-- (c) david vajda  
-- 01/30/2026  
-- semiconductor
```

```
forward  
reverse
```

```
forward - durchlassen  
reverse - sperren
```

```
forward  
reverse  
reject
```

```
forward - durchlassen  
reverse - sperren  
reject - zurueckweisen
```

```
richtung  
spannung
```

```
richtung  
spannung  
strom
```

```
negativ: reverse  
positiv: forward  
  <= A
```

```
current - strom  
voltage - spannung
```

```
current/currency
```

```
alternating  
direct
```

```
durchlassen
```

sperren
durchbruch

durchlassen
sperren
durchbrechen

Surge forward current - Stoßstrom
Peak forward current - Spitzendurchlassstrom

reverse voltage.
DC: Direct Current - strom

Die englischen Begriffe
currency und current klingen ähnlich, haben aber unterschiedliche Bedeutungen, auch wenn s

VCC: Voltage Current Collector

DC (Direct Current)
AC (Alternating Current) bezeichnet Wechselstrom

Semiconductor component: Halbleiterbauelement

Semiconductor:

Breakdown Voltage: Durchbruchspannung / Sperrspannungsgrenze
Forward Voltage: Durchlassspannung / Vorwärtsspannung
Reverse Voltage: Sperrspannung

Durchbruchspannung
Durchlassspannung
Sperrspannung

Die Durchlassspannung (auch
Flussspannung, Schwellenspannung oder Kniespannung) ist die Mindestspannung, die an einer

Die Vorwärtsspannung ((U_D)), auch Durchlassspannung) ist der Spannungsabfall über eine

Durchlassspannung = Vorwaertsspannung (mindest spannung)
Sperrspannung (maximalspannung)
sperrspannungsgrenze (durchbruch)

ueber sperrspannungsgrenze erfolgt durchbruch...

Reverse Voltage (Sperrspannung) ist die Spannung, die in nicht-leitender Richtung an ein H
Wichtige Fakten zur Sperrspannung (Reverse Voltage):

Funktionsweise: Im Gegensatz zur Durchlassrichtung (Forward Bias) wird die Verarmungss
Anwendung: Diese Eigenschaft wird genutzt, um Wechselstrom (AC) in Gleichstrom (DC) um

Schutzfunktion: Dioden dienen oft als Schutz vor Verpolung (Reverse Polarity Protection)
 Sperrschichtdurchbruch: Wird eine kritische Sperrspannung überschritten, kommt es zum Durchbruch
 Zener-Diode: Eine Ausnahme ist die Zener-Diode, die speziell dafür entwickelt wurde, in der Sperrrichtung zu leiten

Übersetzung/Fachbegriffe:

Reverse Voltage: Sperrspannung
 Reverse Bias: Sperrvorspannung / Sperrrichtung
 Forward Voltage: Durchlassspannung / Vorwärtsspannung
 Breakdown Voltage: Durchbruchspannung / Sperrspannungsgrenze
 Cathode/Anode: Kathode/Anode
 Semiconductor component: Halbleiterbauelement

Reverse voltage is the voltage applied across a semiconductor component (like a diode or LED) in the opposite direction of its forward bias.

-- (c) david vajda
 -- 01/29/2026
 -- 7 seg / prom

0: a, b, c, d, e, f
 1: b, c
 2: a, b, d, e, g
 3: a, b, c, d, g
 4: b, c, f, g
 5: a, c, d, f, g
 6: a, c, d, e, f, g
 7: a, b, c
 8: a, b, c, d, e, f, g
 9: a, b, c, d, f, g

	g	f	e	d	c	b	a
0	0	1	1	1	1	1	1
1	0	0	0	0	1	1	0
2	1	0	1	1	0	1	1
3	1	0	0	1	1	1	1
4	1	1	0	0	1	1	0
5	1	1	0	1	1	0	1
6	1	1	1	1	1	0	1
7	0	0	0	0	1	1	1
8	1	1	1	1	1	1	1
9	1	1	0	1	1	1	1

0	0	0	1	1	1	1	1	1
1	0	0	0	0	0	1	1	0
2	0	1	0	1	1	0	1	1
3	0	1	0	0	1	1	1	1
4	0	1	1	0	0	1	1	0
5	0	1	1	0	1	1	0	1

6	0	1	1	1	1	1	0	1
7	0	0	0	0	0	1	1	1
8	0	1	1	1	1	1	1	1
9	0	1	1	0	1	1	1	1

```

00111111
00000110
01011011
01001111
01100110
01101101
01111101
00000111
01111111
01101111

```

00111111	0x3f
00000110	0x06
01011011	0x5b
01001111	0x4f
01100110	0x66
01101101	0x6d
01111101	0x7d
00000111	0x07
01111111	0x7f
01101111	0x6f

this has to be inverted

00111111	11000000
00000110	11111001
01011011	10100100
01001111	10110000
01100110	10011001
01101101	10010010
01111101	10000010
00000111	11111000
01111111	10000000
01101111	10010000

originale	inverted	hex
00111111	11000000	0xC0
00000110	11111001	0xF9
01011011	10100100	0xA4
01001111	10110000	0xB0
01100110	10011001	0x99
01101101	10010010	0x92
01111101	10000010	0x82
00000111	11111000	0xF8
01111111	10000000	0x80

01101111 10010000 0x90

anmerkung: beim naechsten, bei 0 und 1
(Schaltalgebra), H/L (auch schaltalgebra)
aber unabh"angig von Low...
und: on/off benutzen.

cat tmp3.txt | sed 's/.*\ (0x.*\)/\1, /g' | tr '\n' ' '

0xC0, 0xF9, 0xA4, 0xB0, 0x99, 0x92, 0x82, 0xF8, 0x80, 0x90,

dann machen wir jetzt alle 4 rein, +2 Addressleitungen...

00111111	0x3f	11000000	0xC0	00000011	0x03	11111100	0xFC
00000110	0x06	11111001	0xF9	10011111	0x9F	01100000	0x60
01011011	0x5b	10100100	0xA4	00100101	0x25	11011010	0xDA
01001111	0x4f	10110000	0xB0	00001101	0x0D	11110010	0xF2
01100110	0x66	10011001	0x99	10011001	0x99	01100110	0x66
01101101	0x6d	10010010	0x92	01001001	0x49	10110110	0xB6
01111101	0x7d	10000010	0x82	01000001	0x41	10111110	0xBE
00000111	0x07	11111000	0xF8	00011111	0x1F	11100000	0xE0
01111111	0x7f	10000000	0x80	00000001	0x01	11111110	0xFE
01101111	0x6f	10010000	0x90	00001001	0x09	11110110	0xF6

jetzt mit Dummy - wg. 10 - dann 16, und - End - End

E:

n:

d:

oder: real Hex: A, b, C, d, E, f

00111111	0x3f	11000000	0xC0	00000011	0x03	11111100	0xFC
00000110	0x06	11111001	0xF9	10011111	0x9F	01100000	0x60
01011011	0x5b	10100100	0xA4	00100101	0x25	11011010	0xDA
01001111	0x4f	10110000	0xB0	00001101	0x0D	11110010	0xF2
01100110	0x66	10011001	0x99	10011001	0x99	01100110	0x66
01101101	0x6d	10010010	0x92	01001001	0x49	10110110	0xB6
01111101	0x7d	10000010	0x82	01000001	0x41	10111110	0xBE
00000111	0x07	11111000	0xF8	00011111	0x1F	11100000	0xE0
01111111	0x7f	10000000	0x80	00000001	0x01	11111110	0xFE
01101111	0x6f	10010000	0x90	00001001	0x09	11110110	0xF6

this will follow

fordere 1 - weg:

01111111	0x3f	10000000	0xC0	00000001	0x03	11111100	0xFC
00000110	0x06	11110001	0xF9	10011111	0x9F	01100000	0x60
10110111	0x5b	01001000	0xA4	00100100	0x25	11011010	0xDA

1001111	0x4f	0110000	0xB0	0000110	0x0D	1111001	0xF2
1100110	0x66	0011001	0x99	1001100	0x99	0110011	0x66
1101101	0x6d	0010010	0x92	0100100	0x49	1011011	0xB6
1111101	0x7d	0000010	0x82	0100000	0x41	1011111	0xBE
0000111	0x07	1111000	0xF8	0001111	0x1F	1110000	0xE0
1111111	0x7f	0000000	0x80	0000000	0x01	1111111	0xFE
1101111	0x6f	0010000	0x90	0000100	0x09	1111011	0xF6

vorne eine 0 dran:

00111111	0x3f	01000000	0xC0	00000001	0x03	01111110	0xFC
00000110	0x06	01111001	0xF9	01001111	0x9F	00110000	0x60
01011011	0x5b	00100100	0xA4	00010010	0x25	01101101	0xDA
01001111	0x4f	00110000	0xB0	00000110	0x0D	01111001	0xF2
01100110	0x66	00011001	0x99	01001100	0x99	00110011	0x66
01101101	0x6d	00010010	0x92	00100100	0x49	01011011	0xB6
01111101	0x7d	00000010	0x82	00100000	0x41	01011111	0xBE
00000111	0x07	01111000	0xF8	00001111	0x1F	01110000	0xE0
01111111	0x7f	00000000	0x80	00000000	0x01	01111111	0xFE
01101111	0x6f	00010000	0x90	00000100	0x09	01111011	0xF6

erneutes setzen vom hex code

00111111	0x3f	01000000	0xC0	00000001	0x01	01111110	0x7E
00000110	0x06	01111001	0xF9	01001111	0x4F	00110000	0x30
01011011	0x5b	00100100	0xA4	00010010	0x12	01101101	0x6D
01001111	0x4f	00110000	0xB0	00000110	0x06	01111001	0x79
01100110	0x66	00011001	0x99	01001100	0x9C	00110011	0x33
01101101	0x6d	00010010	0x92	00100100	0x48	01011011	0x5B
01111101	0x7d	00000010	0x82	00100000	0x20	01011111	0x5F
00000111	0x07	01111000	0xF8	00001111	0x0F	01110000	0x70
01111111	0x7f	00000000	0x80	00000000	0x00	01111111	0x7F
01101111	0x6f	00010000	0x90	00000100	0x04	01111011	0x7B

dann jetzt richtige plaetze am ende 000000

00111111	0x3f	01000000	0xC0	00000001	0x01	01111110	0x7E
00000110	0x06	01111001	0xF9	01001111	0x4F	00110000	0x30
01011011	0x5b	00100100	0xA4	00010010	0x12	01101101	0x6D
01001111	0x4f	00110000	0xB0	00000110	0x06	01111001	0x79
01100110	0x66	00011001	0x99	01001100	0x9C	00110011	0x33
01101101	0x6d	00010010	0x92	00100100	0x48	01011011	0x5B
01111101	0x7d	00000010	0x82	00100000	0x20	01011111	0x5F
00000111	0x07	01111000	0xF8	00001111	0x0F	01110000	0x70
01111111	0x7f	00000000	0x80	00000000	0x00	01111111	0x7F
01101111	0x6f	00010000	0x90	00000100	0x04	01111011	0x7B
00000000	0x00	00000000	0x00	00000000	0x00	00000000	0x00
00000000	0x00	00000000	0x00	00000000	0x00	00000000	0x00
00000000	0x00	00000000	0x00	00000000	0x00	00000000	0x00
00000000	0x00	00000000	0x00	00000000	0x00	00000000	0x00

