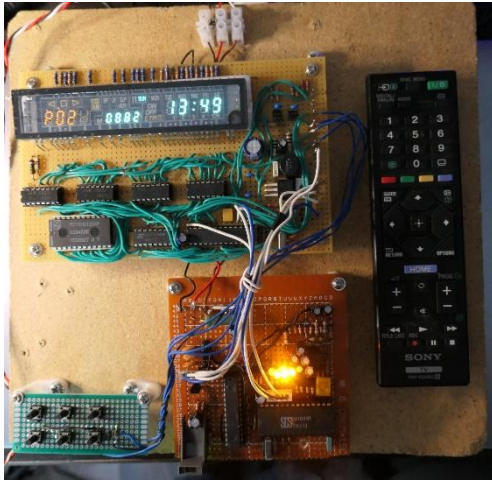


Alte Technik neu erdacht.

Eine VFD-Anzeige bekommt ein neues Leben.



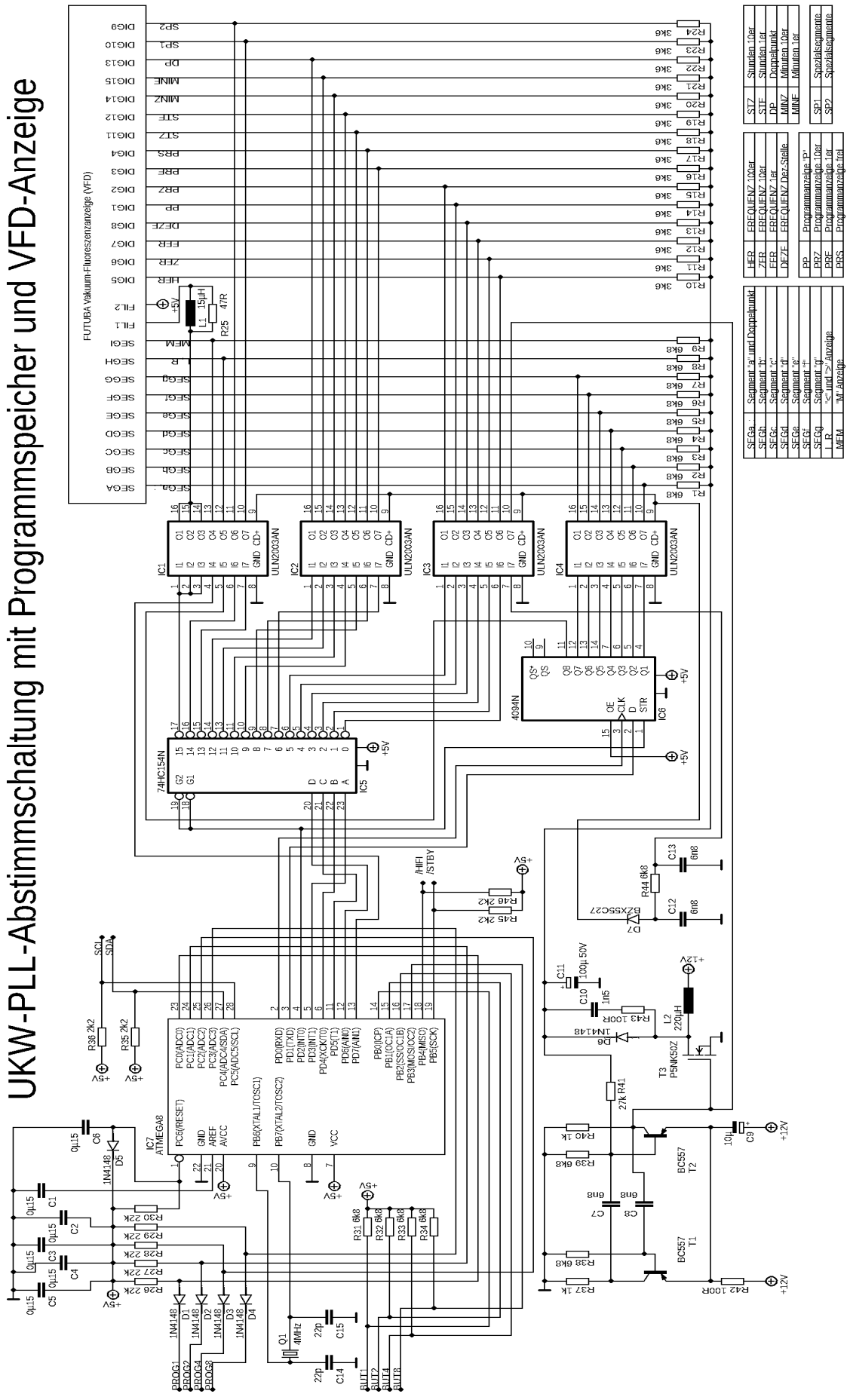
Irgendwann kommt jeder Elektroniker in seinem Leben in den Kontakt von VFD-Anzeigen, entweder weil man einen Videorecorder zerlegt oder einen CD-Player repariert. Eine kleine unscheinbare Glasplatte mit interessanten Innenleben von Symbolen, Heizdraht und jede Mengen an Lötkontakten, welche auf eine stabile Platine gelötet sind. Unverwechselbares Klingen bei jeder Bewegung dieses Bauteils. Das sind die Eigenschaften eines Vakuum-Fluoreszenzdisplays. Erst im Betrieb zeigt sich ein herrlich anzusehendes Schauspiel aus Ästhetik und Vollkommenheit.

Vakuumfluoreszenzanzeigen sind zwischen einer durchsichtigen Glasscheibe und einer rückseitigen Basisplatte, die üblicherweise ebenfalls aus Glas besteht, aufgebaut. Die Platten sind am Rand mit Glaslot verbunden oder miteinander verklebt, das dazwischen liegende Anzeigesystem befindet sich im Vakuum. Vor den die DIGIT-Gittern ist ein dünner, mit Oxiden beschichteter Wolfram-Heizdraht gespannt, welcher als direktbeheizte Kathode arbeitet und von diesem werden thermisch Elektronen emittiert die dann zu den Anoden-Segmenten bei positiver Gitterspannung geleitet werden. Die Leuchtstoffschicht aus Phosphor, mit der die Anoden bedeckt sind, beginnt beim Auftreffen der Elektronen zu leuchten wie im „Magischen Auge“. Ein Segment der Anzeige leuchtet, wenn sowohl das Gitter als auch die Anode elektrisch positiv gegenüber der Kathode sind. Die Spannung zwischen Anode und Kathode liegt zwischen ca. 18 und 40 Volt.

Da die Betriebsspannung noch Basteltauglich ist, so ist klar etwas damit bauen zu müssen. Die 30V kann man leicht aus 12V erzeugen, Die Heizspannung aus den den 5V gewinnen. Mein verwendetes VFD-Display stammt aus einem Videorecorder der 1990er-Jahre und soll mir Programmplatz, Frequenz, Tage und Uhrzeit anzeigen können. Es hat lang gedauert das Konzept zusammen zu programmieren, auch das Herausfinden der PIN-Belegung dauerte etwas. Eines war im Vorhinein klar, es musste ein Anzeigenmultiplex gebaut werden.

Ein Schaltbild schafft Klarheit:

UKW-PLL-Abstimmuschaltung mit Programmspeicher und VFD-Anzeige

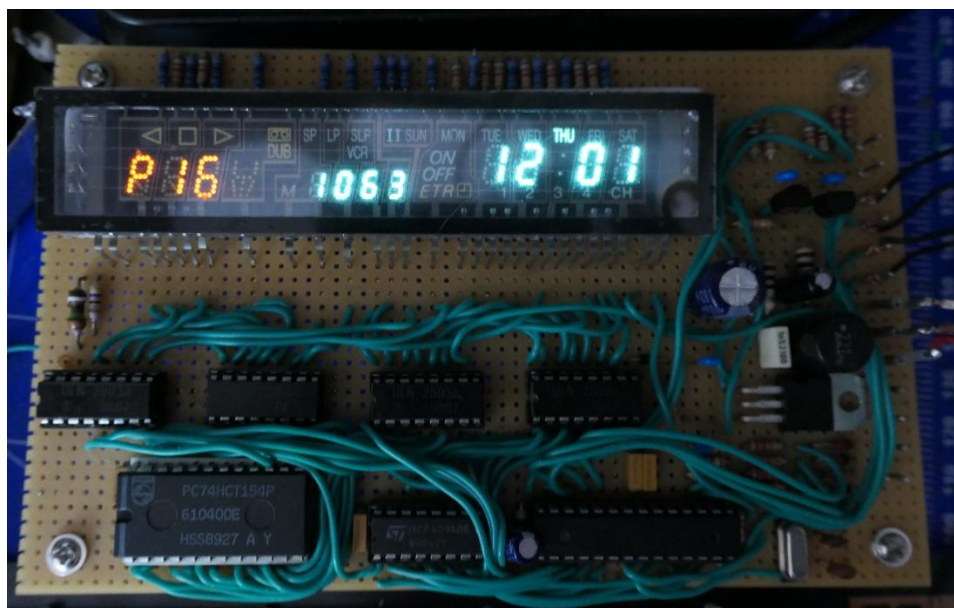


Die Transistoren T1, T2 und T3 erzeugen zusammen mit einigen zusätzlichen Bauteilen die ca. +30V für die VFD-Anzeige, die Schutzschaltung mit der Zenerdiode D2 und einigen Teilgruppen der ULN2003-ICs sorgen dafür, dass Segmente und DIGITs nie mehr als 30V abbekommen und dadurch die Anzeige Helligkeitsstabil bleibt. Hierbei wurde bewusst darauf geachtet, verfügbare Bauteile zu verwenden. Die Darlington-Treiber-ICs ULN2003 die einfachste Weise VFD-Anzeigen ansteuern zu können. Normalerweise steuert man die DIGITs und Segmente aktiv positiv an und verwendet spezielle VFD-Treiber-ICs dafür, sind aber selten zu bekommen oder Maskenprogrammierte ROM-Controller und nicht leicht anzusteuern. Die vielen PINs wollen natürlich betrieben werden und da wir nicht viele PORTs übrighaben, trickst man mit 4-zu-16-Decoder und Schieberegister. So ist es möglich mit 8 Bits 16 Stellen zu je 9 Segmenten leuchten zu lassen. Das Programm am ATMEGA8 ist dann nur noch die Kirsche am Törtchen.

Wegen der Komplexität und Größe des Programms, wurde es im AVR-Assembler geschrieben. Der Vorteil liegt einfach darin, dass es übersichtlicher und leichter zu verstehen ist. Umso schöner ist es, wenn es assembliert ist und diese File für den VFD-Controller ist:

MAKERCLOCK_MEGA8.HEX

Die Datei wird einfach mit einem Programmiergerät, wie den AVR ATMEL STK500 auf den ATMEGA8 geflasht. Der ATMEGA8 muss aber auch noch so programmiert werden, dass der mit einem externen Quartz von 4MHz lauffähig ist. Das Ergebnis sollte STAND-Alone so aussehen:



Nun soll eine Infrarotsteuerbare Kontrolleinheit angeschlossen werden. Diese kann von zwei verschiedenen Fernbedienungen gesteuert werden. Zunächst das Schaltbild:

ATMEG8 & M104B1 - IR-ADAPTER für SONY- und NEC-Protokolle
ALEXANDER ELECTRONICFOX FUCHS 2025(C)

Beide Programme sind in C erstellt worden und sind diesen ARDUINO-Sketch ähnlich:

```
// ATMEGA8 mit internen 8MHz-RC-Oszillator
// PB0=PA-M104B1, PB1=PB-M104B1, PB2=PC-M104B1, PB3=PD-M104B1,
// PB4=HIFI-TO, PB5=STBY-M104B1
// PROGRAMM RUNTER      PC0 vom ATMEGA8
// PROGRAMM RUNTER      PC1 vom ATMEGA8
// LAUTSTÄRKE VERRINGERN PD6 vom ATMEGA8
// LAUTSTÄRKE ERHÖHEN   PD5 vom ATMEGA8
// IN BEREITSCHAFT SCHALTEN PD4 vom ATMEGA8
// HIFI-Funktion        PD3 vom ATMEGA8
// IR-Diode an          PD7 vom ATMEGA8

#define DECODE_NEC
#define DECODE_SONY
#define DISABLE_CODE_FOR_RECEIVE_PROTOCOLS 1
#include <IRremote.hpp>

#define IR_RECEIVE_PIN 7 // IR-Diode an PD7 vom ATMEGA8

// --- Zusätzliche Tastenpins ---
#define BTN_PROG_MINUS A0 // PROGRAMM RUNTER      PC0 vom
ATMEGA8
#define BTN_PROG_PLUS A1 // PROGRAMM RUNTER      PC1 vom
ATMEGA8
#define BTN_VOL_MINUS 6 // LAUTSTÄRKE VERRINGERN PD6 vom
ATMEGA8
#define BTN_VOL_PLUS 5 // LAUTSTÄRKE ERHÖHEN PD5 vom
ATMEGA8
#define BTN_STBY 4 // IN BEREITSCHAFT SCHALTEN PD4 vom
ATMEGA8
#define BTN_HIFI 3 // HIFI-Funktion PD3 vom ATMEGA8

byte portB_state = 0xFF;
unsigned long lastValidCode = 0;
unsigned long lastActionTime = 0;
bool waitingForValidIR = false;

void setup() {
  IrReceiver.begin(IR_RECEIVE_PIN);
  DDRB = 0xFF;

  // Eingänge mit Pullups
  pinMode(BTN_PROG_MINUS, INPUT_PULLUP);
  pinMode(BTN_PROG_PLUS, INPUT_PULLUP);
  pinMode(BTN_VOL_MINUS, INPUT_PULLUP);
  pinMode(BTN_VOL_PLUS, INPUT_PULLUP);
  pinMode(BTN_STBY, INPUT_PULLUP);
  pinMode(BTN_HIFI, INPUT_PULLUP);

  // --- Einschaltimpuls ---
```

```

PORTB = 0xDF;
delay(35);
PORTB = 0xFF;
portB_state = 0xFF;
}

void loop() {
    unsigned long now = millis();

    // === Hardware-Tasten prüfen ===
    if (digitalRead(BTN_PROG_MINUS) == LOW) {
        PORTB = 0xFD; // PROG-
        portB_state = 0xFD;
        lastActionTime = now;
    }
    else if (digitalRead(BTN_PROG_PLUS) == LOW) {
        PORTB = 0xFE; // PROG+
        portB_state = 0xFE;
        lastActionTime = now;
    }
    else if (digitalRead(BTN_VOL_MINUS) == LOW) {
        PORTB = 0xFC; // VOL-
        portB_state = 0xFC;
        lastActionTime = now;
    }
    else if (digitalRead(BTN_VOL_PLUS) == LOW) {
        PORTB = 0xFA; // VOL+
        portB_state = 0xFA;
        lastActionTime = now;
    }
    else if (digitalRead(BTN_STBY) == LOW) {
        PORTB = 0xF0; // STBY
        portB_state = 0xF0;
        lastActionTime = now;
    }
    else if (digitalRead(BTN_HIFI) == LOW) {
        PORTB = 0xEF; // HIFI aktiv
        portB_state = 0xEF;
        lastActionTime = now;
    }

    // === IR-CODE prüfen ===
    if (IrReceiver.decode()) {
        unsigned long code = IrReceiver.decodedIRData.decodedRawData;

        // Fehlerhafte Codes abfangen
        if (code == 0x1821 || code == 0x1020) {
            waitingForValidIR = true;
            IrReceiver.resume();
            return;
        }
    }
}

```

```

if (waitingForValidIR) {
  if (code != 0x0 && code != 0x1821 && code != 0x1020)
    waitingForValidIR = false;
  else {
    IrReceiver.resume();
    return;
  }
}

if (code == 0x0 && lastValidCode != 0)
  code = lastValidCode;

byte newPortB = portB_state;

// --- IR-Befehle --- // PB0=PA-M104B1, PB1=PB-M104B1, PB2=PC-M104B1,
PB3=PD-M104B1, PB4=HIFI-TO, PB5=STBY-M104B1
switch (code) {
  case 0xEA15FF00: case 0x90: newPortB = 0xFE; break; // CH+, P+
  case 0xE718FF00: case 0x91: newPortB = 0xFD; break; // CH-, P-
  case 0xBE41FF00: case 0x92: newPortB = 0xFA; break; // VOL+
  case 0xBD42FF00: case 0x93: newPortB = 0xFC; break; // VOL-
  case 0xF50AFF00: case 0x95: newPortB = 0xF0; break; // STBY
  case 0xBB44FF00: case 0x97: newPortB = 0xEF; break; // HIFI AKTIV
  case 0xAC53FF00: case 0xE5: newPortB = 0xF8; break; // NORMALZUSTAND
}

if (newPortB != portB_state) {
  PORTB = newPortB;
  portB_state = newPortB;
}

if (code != 0x0) lastValidCode = code;
lastActionTime = now;
IrReceiver.resume();
}

// --- Automatisches Reset ---
if (now - lastActionTime > 500 && portB_state != 0xFF) {
  if (portB_state != 0xF0) {
    PORTB = 0xFF;
    portB_state = 0xFF;
  }
}

delay(50);
}

```

Mit Hilfe dieses SKETCHES ist es möglich, die Schaltung anderen Fernbedienungen anzupassen.

PROG-/± lassen den Programmplatz wechseln.

VOL-/± ändern die Lautstärke.

STBY schaltet das Gerät in Bereitschaft.

AUDIO lässt die Schaltung auf einen externen AUDIO-Eingang HIFI umschalten.

Zusätzliche Tasten können zum Regeln der DAC-Ausgänge 1 -3 verwendet werden.

Mit Tasten am Bedienfeld gilt das Gleiche.

DAY ändert den TAG

HOURL ändert die Stunden

MIN ändert die Minuten

FREQ-/± ändert die Frequenz

MEM speichert die Frequenz am aktuellen Programmplatz

Der IC M104B1 stellt folgendes Aufgabengebiet vor:

4 unabhängige DAC-Ausgänge, davon DAC0 = VOL (LAUTSTÄRKE).

32 Programmplätze (16 werden genutzt).

Binäre Bedienung über ATMEGA8.

455kHz-Oszillator für die DACs.

Bereitschaftsschaltung mit Anwurfautomatik.

Die HIFI-Umschaltung erfolgt mit einer bistabilen Kippstufe mit Power-ON-RESET und besteht nur aus zwei Transistoren.

