Der Einbau soll exemplarisch in einem Blaupunkt „Salerno“ durchgeführt werden.

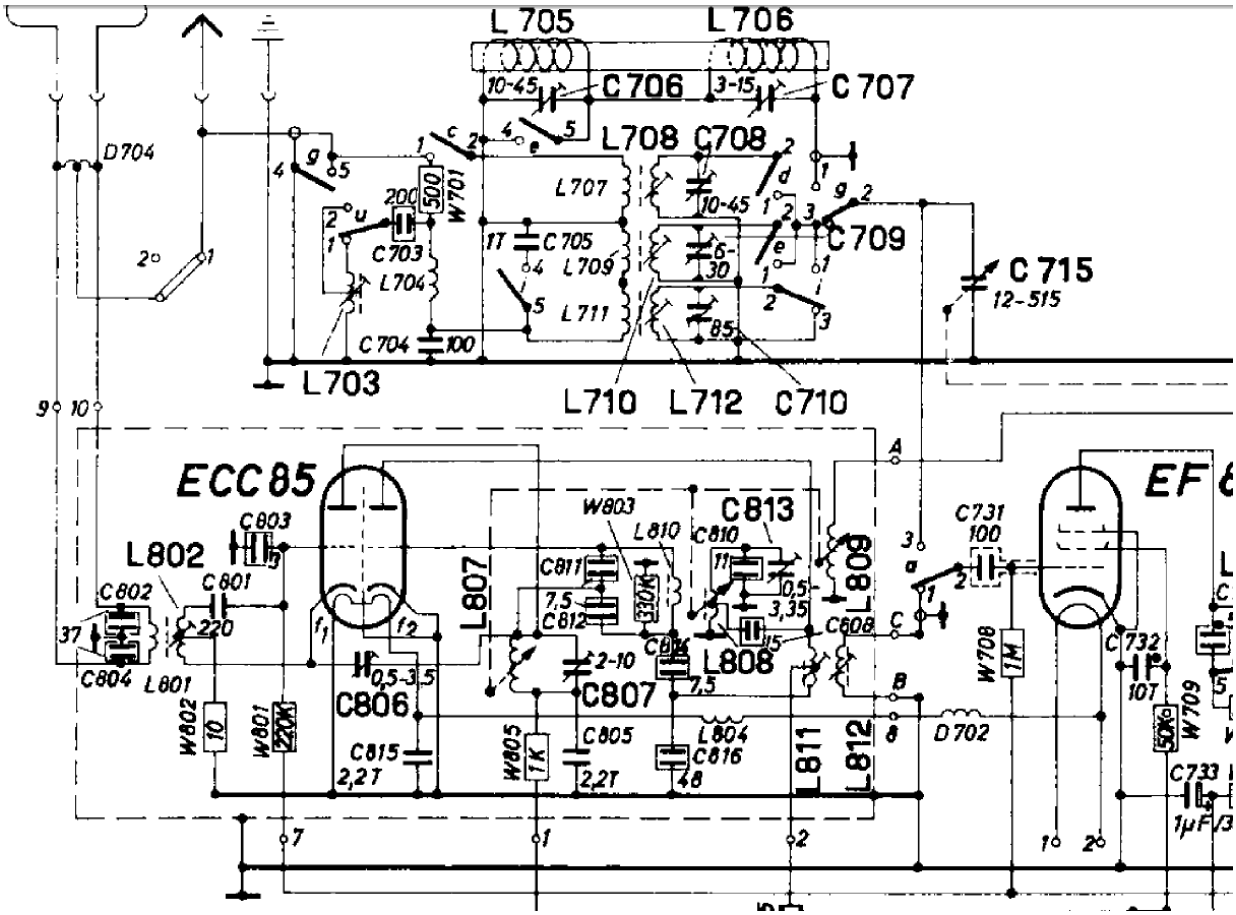

Abbildung 59: Schaltbildauszug Radio

Das Gerät verfügt über einen Netzstecker ohne Schutzkontakt. Das Chassis kann jedoch per durch die Rückwand zugänglicher Buchse geerdet werden. Zugleich wird über den Tuner mit der ECC85 auch der Heizkreis mit geerdet, so dass eine Einschleppung von Brummen in das Schirmgehäuse der Emulation ausgeschlossen ist.

Der Erdungsanschluss der Radio-Rückwand wird mit dem Erdungsanschluss der Netzversorgung oder der Antennenanlage verbunden, um das über das Trafonetzteil eingestreute Brummen zu vermindern.

Probeweise wurde auch der Netzstecker umgedreht. Der interne Aufbau des Trafos erbringt unterschiedlich starke parasitäre Kopplungen.

Das Schirmgehäuse der Emulation muss mit dem Chassis des Radios möglichst niederohmig und niederinduktiv verbunden werden. Diese Maßnahme mindert Brummen und HF-Störungen.

Abbildung 60: Radio Frontansicht

Abbildung 61: Montage Detail

Abbildung 62: Montage Detail

Beobachtungen:

- Es war egal, ob das Chassis mit der Netzerde oder mit der Antennenanlage geerdet wurde.
- Die Polung des Netzsteckers hatte erhebliche Wirkung auf die Beruhigung des Schirmbildes.
- Trotz bestmöglicher Erdung verblieb ein Zittern von 10% des Schirmbildes.
- Die Erdung des Schirmgehäuses ist unverzichtbar für eine gute Brummminderung.
- HF-Störungen waren auch ohne angeschlossene Außenantenne in keinem Frequenzbereich feststellbar. Erst bei Näherung des Antenneneingangs bis auf 10 mm an den ungeerdeten Schirm waren schwache Störungen hörbar. Diese verschwanden restlos nach Erdung des Schirmes.
- Die reflektierende Oberfläche des Schirmgehäuses stört. Hier wäre eine dunkle Abdeckung sinnvoll.
- Im Vergleich zu einer realen EM80 ergibt sich eine geringere Anzeigeempfindlichkeit und eine unruhigere Darstellung.
- Auf das grüne Farbfilter wurde vorerst verzichtet. Es kann jederzeit eingeschoben werden.

Aussicht

Neben der Emulation anderer Röhrentypen bis hin zur Stereo-Anzeige lässt sich das Projekt erweitern, denn der Controller wird erst wenig genutzt. Denkbar wäre die Überwachung der am Sockel angelegten Betriebsspannungen.

Oder die Nutzung der bisher ungenutzten drei Sockelstifte für zusätzliche Messungen und Steuerungen. Zum Beispiel der Messung einer angelegten Frequenz oder einer Funktionsumschaltung.

Interessant ist auch die Beschaffung eines Color-Displays. Und man sollte prüfen, ob TFT oder LCD vom Kontrast her ausreichen.

Alternativen

Die beiden Verfasser hatten 2017 im military-tubes-Forum einen Mini-Inverter zum Betrieb verdunkelter Magischer Augen entworfen.

Die Wirkung beruhte auf der Beobachtung, dass eine angelegte Gleichspannung deutlich weniger Helligkeit erbringt, als eine mit HF überlagerte Gleichspannung gleichen Spannungsmittelwertes.

Es ergaben sich nur geringe Verminderungen der (mittleren) Ablenkwinkel und somit einen Vorteil gegenüber dem bekannten Hochsetzen der Anodengleichspannung. Allerdings wird die Abbildungsschärfe etwas reduziert.

Die geringen Verminderungen des Ablenkwinkels sind subjektiv, da gerade der Leuchtfächer gerade dann besonders gut zu sehen ist, wenn die hohe Amplitude der Leuchtschirmspitzenspannung die Ablenkung reduziert.

Zusätzlich wurde spekuliert, dass der pulsierende Elektronenbeschuss den Leuchtschirm nach und nach regeneriert, denn die Röhren wurden während den Versuchen immer leuchtstärker. Was allerdings auch katodenseitige Ursachen gehabt haben kann.

Trotz des Bemühens zur Erzeugung einer möglichst sinusförmigen HF waren die Funkstörungen des Inverters erheblich. Die Anbringung von Schirmungen und Drosseln wurde nur wenig erprobt und waren auch nur bedingt sinnvoll, da besonders der unschirmbare Leuchtschirm HF abstrahlt.

IMPBMP

Dieses kleine ´C++´-Tool konvertiert monochrome MS-Paint BMP-Dateien in einen `C´-Header. Die Abmessungen des Bildes dürfen die unter COLS und ROWS definierten Werte nicht überschreiten. Das Tool wird z.B. in der Windows DOS-Box aufgerufen mit „IMPBMP em80scrn“ und meldet Fehler bzw. den Wandlungserfolg.

Abbildung 63: Listing „IMPBMP.CPP“

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <stdlib.h>

#define YES       1
#define NO        0
#define ERROR  -1

#define UINT     unsigned int
#define WORD   unsigned int
#define DWORDunsigned long
#define LONG    long

static struct {    /* bmfh */
   UINT    bfType;
   DWORD   bfSize;
   UINT    bfReserved1;
   UINT    bfReserved2;
   DWORD   bfOffBits;
} bmfh;


static struct {    /* bmih */
   DWORD   biSize;
   LONG    biWidth;
   LONG    biHeight;
   WORD    biPlanes;
   WORD    biBitCount;
   DWORD   biCompression;
   DWORD   biSizeImage;
   LONG    biXPelsPerMeter;
   LONG    biYPelsPerMeter;
   DWORD   biClrUsed;
   DWORD   biClrImportant;
} bmih;


#define ROWS    512
#define COLS    512


static int my_write(FILE *dest,char *buf)
{
          if(strlen(buf) != fwrite(buf,sizeof(char),strlen(buf),dest)) {
            printf("\007\n\nSchreibfehler! Abbruch\n\n\n\n");
            return(NO);
          }
          return(YES);
}
```

```
main(int argc,char *argv[])
{
        FILE *source, *dest;
        char buffer[60000L], hbuf[80];
        int col, row_len, rows, cols;
        unsigned char byte;


        clrscr();

        if(argc != 2) {
          printf("\007\n\nBenutzung:\n\n\n");
          printf("\"IMPBMP datei <Return>\"\n\n");
          printf("\"datei\" ist die Bitmap, aus der eine gleichnamige Header-Datei erzeugt wird...\n\n");
          printf("die Bitmap muss einfarbig sein, die Groesse ist egal.\n\n\n\n");
          goto err;
        }

        /* quelldatei öffnen */
        strcpy(buffer,argv[1]);
        strcat(buffer,".BMP");
        source = fopen(buffer,"rb");
        if(!source) {
          printf("\007\n\n...kann die Datei \"%s\" nicht finden!\n\n",buffer);
          printf("Bitte Verzeichnis beachten und Dateinamen genau pruefen\n\n\n\n");
          goto err;
        }

        /* zieldatei öffnen */
        strcpy(buffer,argv[1]);
        strcat(buffer,".H");
        dest = fopen(buffer,"rb");
        if(dest) {
          printf("\007\n\nDie Datei \"%s\" gibt es schon!\n\n",buffer);
          printf("...ich will sie nicht ueberschreiben\n\n\n\n");
          goto err;
        }
        fclose(dest);

        dest = fopen(buffer,"wb");
        if(!dest) {
          printf("\007\n\n...kann die Datei \"%s\" nicht erzeugen!\n\n",buffer);
          printf("Wahrscheinlich ist die Festplatte voll\n\n\n\n");
          goto err;
        }

        /* bitmap-header einlesen */
        if((fread(&bmfh,sizeof(char),sizeof(bmfh),source) != sizeof(bmfh))||(strncmp((char
*)&(bmfh.bfType),"BM",2))) {
          printf("\007\n\nFehler im Bitmap-Header! Abbruch\n\n\n\n");
          goto err;
        }

        /* info-header einlesen */
        if((fread(&bmih,sizeof(char),sizeof(bmih),source) != sizeof(bmih))||(bmih.biBitCount !=
1)||(bmih.biCompression)) {
```

```
            printf("\007\n\nFehler im Info-Header: keine monochrome Bitmap! Abbruch\n\n\n\n");
            goto err;
          }

          /* color-table überspringen */
          col = (size_t)(bmfh.bfOffBits) - sizeof(bmfh) - sizeof(bmih);
          if(col != fread(buffer,sizeof(char),col,source)) {
            printf("\007\n\nFehler in der Farbtabelle! Abbruch\n\n\n\n");
            goto err;
          }

          /* beginn der datei in C-notation schreiben: name, cols, rows */
          rows = min(ROWS,(int)(bmih.biHeight));
          cols = min(COLS,(int)(bmih.biWidth));
          sprintf(buffer,"PROGMEM const unsigned char %s[%d] = {",argv[1],cols * rows / 8);
          if(!my_write(dest,buffer)) goto err;

          /* nun alle daten auf einen schlag einlesen (32 Bit boundaries!) */
          bmih.biWidth /= 8;
          if((size_t)(bmih.biWidth) % 4)
            row_len = (size_t)(bmih.biWidth) + 4 - (size_t)(bmih.biWidth) % 4;
          else row_len = (size_t)(bmih.biWidth);
          fread(buffer,row_len,(size_t)(bmih.biHeight),source);

          for(--rows;rows >= 0;rows--) {
            sprintf(hbuf,"\r\n\t\t\t");
            if(!my_write(dest,hbuf)) goto err;
            for(col = 0;col < cols;col += 8) {
              byte = buffer[(col >> 3) + rows * row_len];
              sprintf(hbuf,"0x%.2X, ",byte);
              if(!my_write(dest,hbuf)) goto err;
            }
          }
          /* ende der Datei in C-Notation schreiben */
          sprintf(buffer,"\r\n};\r\n\n/* ENDE */");
          my_write(dest,buffer);

          printf("\n\nHeader-Datei erfolgreich erstellt\n\n\n\n");

err:      fclose(dest);
          fclose(source);
          return(0);
}


/* ENDE */
```

CNTBMP

Dieses kleine ´c++´-Tool konvertiert monochrome MS-Paint BMP-Dateien in ATTiny `C´-Header. Dazu zählt es in jeder Reihe die Anzahl der ungesetzten (schwarzen) Pixel und gibt diese Anzahl als Zahlenwert aus. Dies wird z.B, genutzt, um die Röhrenkennlinie mit Paint zu konstruieren.

Die Abmessungen des Pixelbildes dürfen die unter COLS und ROWS definierten Werte nicht überschreiten. Das Programm wird z.B. in der Windows DOS-Box aufgerufen mit „CNTBMP em80func“ und meldet Fehler bzw. den Wandlungserfolg.

Abbildung 64: Listing „CNTBMP.CPP"

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <stdlib.h>

#define YES       1
#define NO        0
#define ERROR  -1

#define UINT     unsigned int
#define WORD   unsigned int
#define DWORDunsigned long
#define LONG    long

static struct {    /* bmfh */
   UINT    bfType;
   DWORD   bfSize;
   UINT    bfReserved1;
   UINT    bfReserved2;
   DWORD   bfOffBits;
} bmfh;


static struct {    /* bmih */
   DWORD   biSize;
   LONG    biWidth;
   LONG    biHeight;
   WORD    biPlanes;
   WORD    biBitCount;
   DWORD   biCompression;
   DWORD   biSizeImage;
   LONG    biXPelsPerMeter;
   LONG    biYPelsPerMeter;
   DWORD   biClrUsed;
   DWORD   biClrImportant;
} bmih;

#define ROWS    512
#define COLS     512
```

```
static int my_write(FILE *dest,char *buf)
{
        if(strlen(buf) != fwrite(buf,sizeof(char),strlen(buf),dest)) {
          printf("\007\n\nSchreibfehler! Abbruch\n\n\n\n");
          return(NO);
        }
        return(YES);
}

main(int argc,char *argv[])
{
        FILE *source, *dest;
        char buffer[60000L], hbuf[80];
        int col, row_len, rows, cols, pixel_cnt, bit;
        unsigned char byte;


        clrscr();

        if(argc != 2) {
          printf("\007\n\nBenutzung:\n\n\n");
          printf("\"CNTBMP datei <Return>\"\n\n");
          printf("\"datei\" ist die Bitmap, aus der eine gleichnamige Header-Datei erzeugt wird...\n\n");
          printf("die Bitmap muss einfarbig sein, die Groesse ist egal.\n\n\n\n");
          goto err;
        }

        /* quelldatei öffnen */
        strcpy(buffer,argv[1]);
        strcat(buffer,".BMP");
        source = fopen(buffer,"rb");
        if(!source) {
          printf("\007\n\n...kann die Datei \"%s\" nicht finden!\n\n",buffer);
          printf("Bitte Verzeichnis beachten und Dateinamen genau pruefen\n\n\n\n");
          goto err;
        }

        /* zieldatei öffnen */
        strcpy(buffer,argv[1]);
        strcat(buffer,".H");
        dest = fopen(buffer,"rb");
        if(dest) {
          printf("\007\n\nDie Datei \"%s\" gibt es schon!\n\n",buffer);
          printf("...ich will sie nicht ueberschreiben\n\n\n\n");
          goto err;
        }
        fclose(dest);

        dest = fopen(buffer,"wb");
        if(!dest) {
          printf("\007\n\n...kann die Datei \"%s\" nicht erzeugen!\n\n",buffer);
          printf("Wahrscheinlich ist die Festplatte voll\n\n\n\n");
          goto err;
        }
```

```
        /* bitmap-header einlesen */
        if((fread(&bmfh,sizeof(char),sizeof(bmfh),source) != sizeof(bmfh))||(strncmp((char
*)&(bmfh.bfType),"BM",2))) {
          printf("\007\n\nFehler im Bitmap-Header! Abbruch\n\n\n\n");
          goto err;
        }

        /* info-header einlesen */
        if((fread(&bmih,sizeof(char),sizeof(bmih),source) != sizeof(bmih))||(bmih.biBitCount !=
1)||(bmih.biCompression)) {
          printf("\007\n\nFehler im Info-Header: keine monochrome Bitmap! Abbruch\n\n\n\n");
          goto err;
        }

        /* color-table überspringen */
        col = (size_t)(bmfh.bfOffBits) - sizeof(bmfh) - sizeof(bmih);
        if(col != fread(buffer,sizeof(char),col,source)) {
          printf("\007\n\nFehler in der Farbtabelle! Abbruch\n\n\n\n");
          goto err;
        }

        /* beginn der datei in C-notation schreiben: name, cols, rows */
        rows = min(ROWS,(int)(bmih.biHeight));
        cols = min(COLS,(int)(bmih.biWidth));
        sprintf(buffer,"PROGMEM const unsigned char %s[%d] = {",argv[1],rows);
        if(!my_write(dest,buffer)) goto err;

        /* nun alle daten auf einen schlag einlesen (32 Bit boundaries!) */
        bmih.biWidth /= 8;
        if((size_t)(bmih.biWidth) % 4)
          row_len = (size_t)(bmih.biWidth) + 4 - (size_t)(bmih.biWidth) % 4;
        else row_len = (size_t)(bmih.biWidth);
        fread(buffer,row_len,(size_t)(bmih.biHeight),source);

        for(--rows;rows >= 0;rows--) {
          if((rows % 8) == 7) {
            sprintf(hbuf,"\r\n\t\t\t");
            if(!my_write(dest,hbuf)) goto err;
          }
          pixel_cnt = 0;
          for(col = 0;col < cols;col += 8) {
            byte = buffer[(col >> 3) + rows * row_len];
            for(bit = 7;bit >= 0;bit--) {
              if(!(byte & (1 << bit))) pixel_cnt++;
            }
          }
          sprintf(hbuf,"%3.d, ",pixel_cnt);
          if(!my_write(dest,hbuf)) goto err;
        }
        /* ende der Datei in C-Notation schreiben */
        sprintf(buffer,"\r\n};\r\n\n/* ENDE */");
        my_write(dest,buffer);

        printf("\n\nHeader-Datei erfolgreich erstellt\n\n\n\n");
err:    fclose(dest);
        fclose(source);
        return(0);
}
```