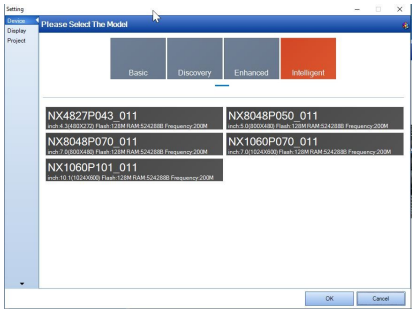
Nextion-Touchscreen Display

Editor und Arduino-Programmierung

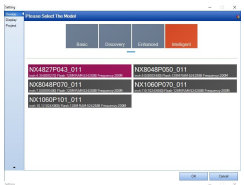
Inhaltsverzeichnis

Editor Start

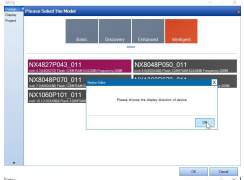


Der Nextion-Editor ist ein installationsfreies Tool.
Der Start erfolgt durch Klick auf NextionEditor.exe

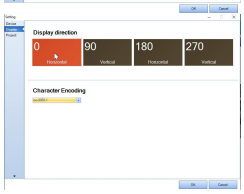
Bei einem Neustart ist zunächst das Display anzugeben.



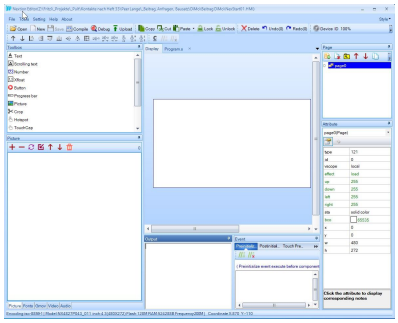
Hier wird der Typ NX4827P043_11 ausgewählt.



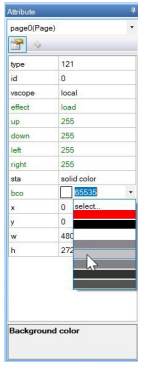
Dann erfolgt die Auswahl der Lage und der Orientierung der Anzeigen



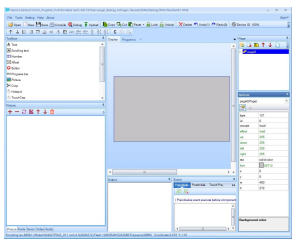
Auswahl Horizontal, 0 Grad
Character Encoding default



Im Editor erscheint das Display-Feld in der Mitte und in der ausgewählten Größe

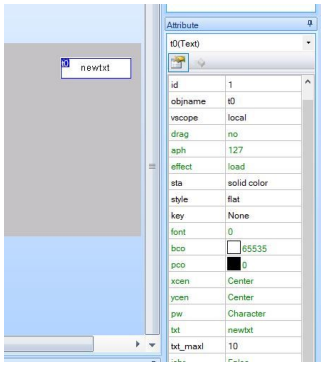


Das Display-Fenster erhält eine Grundfarbe,
Auswahl im Feld bco, Background color



Grundfarbe Display ist jetzt grau.
Jede Auswahl von Objekten erscheint im Display-Fenster

Objekt-Auswahl



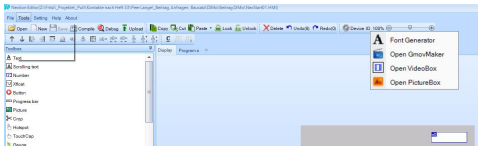
In der Toolbox (oben links) erfolgt die Auswahl des ersten Objekts, Text. Der Editor kennzeichnet dieses Objekt mit „t0“ und vergibt die erste id mit „1“.

Die Lage des Objektes ist nun auf dem Display an die gewünschte Position zu verschieben.

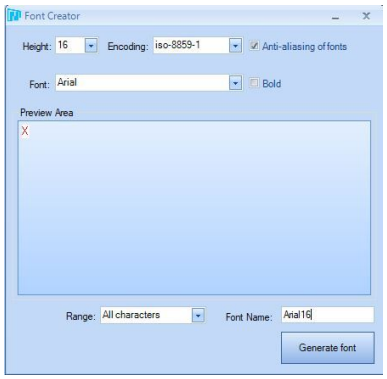
Der initiale Text ist mit dem Attribut „txt“ frei wählbar.

Die grünen Felder sind wärend der Laufzeit des Arduino-Programms veränderbar.

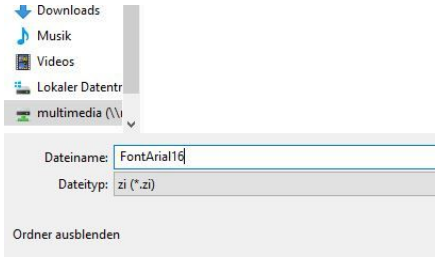
Mit dem Attribut „sta“ und dem Wert „transparency“ wird der Texthintergrund gleich der Display-Grundfarbe.



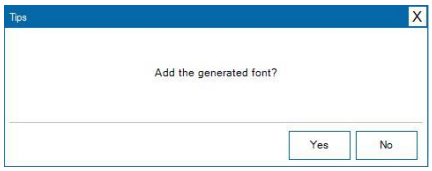
Bevor dieses Objekt ansprechbar ist (und compiliert werden kann) ist ein Font auszuwählen.
Dazu ist unter Tools der Font Generator auszuwählen.



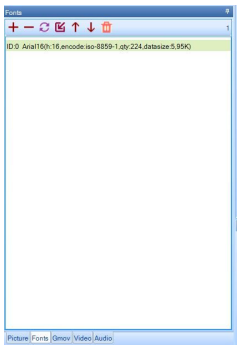
Unter Font wird hier Arial ausgewählt, ein Name vergeben und Generate font geklickt.



Die Speicherung erfolgt am einfachsten im Projekt-Arbeitsverzeichnis



Mit Yes wird der generierte Font in das Nextion-File aufgenommen

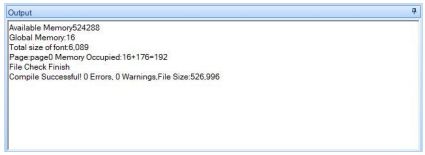


Im Editor-Fenster unten links wird der Font durch die Auswahl „Fonts“ angezeigt.

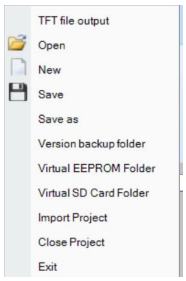
Weitere Fonts werden entsprechend der beschriebenen Vorgehensweise dazugeladen.

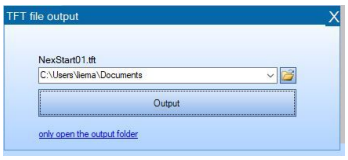
Meist werden verschiedene Schrifthöhen benötigt.

Compilieren und Upload

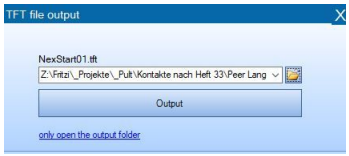


Das Projekt kann jetzt gespeichert und erstmalig compiliert werden.

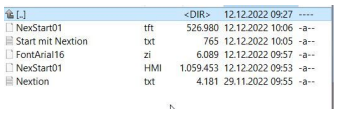




Mit dem Menüpunkt „File“ erfolgt die Auswahl von TFT file output.
Das tft-File ist ein Binär-Format welches alle Objekte enthält und per microSD-Karte in das Display geladen wird.



Auswahl des Verzeichnisses zur Ablage des tft-Files, die Speicherung erfolgt am einfachsten im Projekt-Arbeitsverzeichnis

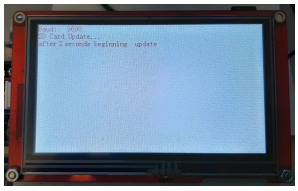


Im Projekt-Arbeitsverzeichnis sind jetzt alle erforderlichen Dateien vorhanden

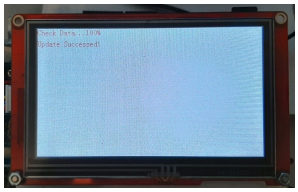
Die Datei NexStart01.tft wird nun auf die microSD-Karte kopiert.

- Kopie in das Hauptverzeichnis der microSD-Karte
- Nur eine tft-Datei ist hier erlaubt
- Kein Schreibschutz für die Datei oder die SD-Karte

Versorgungsspannung des Displays abschalten
microSD Karte (Kontakte nach vorn) in den microSD-Slot einführen
Versorgungsspannung einschalten



Ist alles ok, startet das Update nach kurzer Zeit



Ok, jetzt

- Versorgungsspannung abschalten
- microSD-Karte entfernen

Beim Wiedereinschalten des Displays erscheint der Inhalt mit dem gewähltem Text-Objekt.

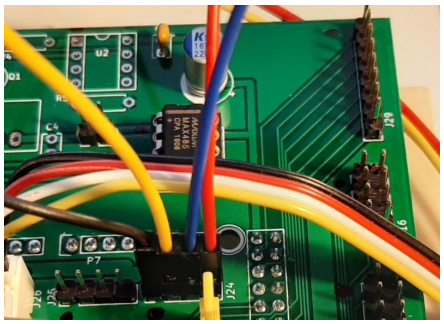
Das Display ist jetzt seriell angesprechbar und der gewünschte Text wird dargestellt.

Die Display-Informationen bleiben dauerhaft gespeichert.

Ausgaben an das Display

Das Nextion-Display ist im Folgenden mit einem Arduino-Mega und dem seriellen Port „Serial2“ verbunden.

Die Versorgung des Displays kann wegen des Strombedarfs von ca. 0,4A nicht vom Arduino übernommen werden. Das Display ist direkt mit 5V z.B. mit einem USB-Netzteil zu versorgen.



Der Anschluss mit der Fahrpult-Platine erfolgt über die Steckverbindung J24.

```
#define nextion Serial2
#define VERSION "NexStart07"

const uint8_t NextiEnd[3] = {0xFF, 0xFF, 0xFF};
const uint8_t NextiSEnd[4] = {0x22, 0xFF, 0xFF, 0xFF};

#define SDIM 3
char *Sarray[SDIM] = {"Text A","Text B","Text C"};
void setup() {

nextion.begin(9600);
delay(2000);
nextion.print("t0.txt=\"");
nextion.print("Text eins\"");
nextion.write(NextiEnd, 3);
delay(2000);

nextion.print("t0.txt=\"");
nextion.print(VERSION);
nextion.write(NextiSEnd, 4);
delay(2000);
}
void loop() {
for (int k=0; k < SDIM; k++) {
  nextion.print("t0.txt=\"");
  nextion.print(Sarray[k]);
  nextion.write(NextiSEnd, 4);
  delay(2000);
  }
}
```

Abschluss Nextion-Befehl
Abschluss String Nextion-Befehl

Schnittstelle Serial2 9600 Bd

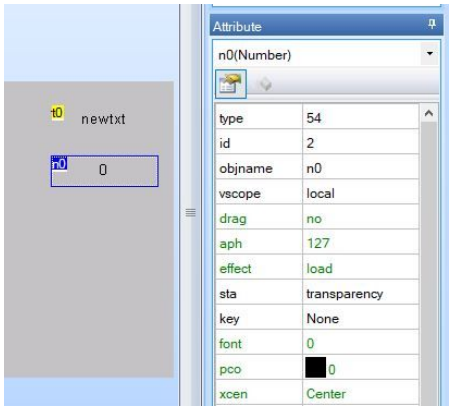
Stringausgabe an Objekt t0
Stringabschluss mit 3x 0xFF

Stringausgabe an Objekt t0
Stringabschluss mit " und 3x 0xFF

Einen String aus `*Sarray[]` ausgeben
Stringabschluss mit " und 3x 0xFF

Hinweis: Der vom Nextion-Editor mitcompilierte Initialtext „newtxt“ erscheint nur nach Stromunterbrechung des Displays.

Ausgeben eines numerischen Wertes



Beide Ausgabe-Fenster sind auf „transparency“ gesetzt.

Der Objektname für die numerische Ausgabe ist „n0“.

Programmergänzungen

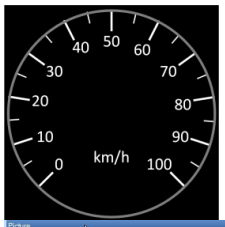
```
#define VERSION "NexStart08"
#define POTI     A11
```

Poti an A11

```
void loop() {
int PotiVal = 0;
for (int k=0; k < SDIM; k++) {
  nextion.print("t0.txt=\"");
  nextion.print(Sarray[k]);
  nextion.write(NextiSEnd, 4);

  PotiVal = analogRead(POTI);
  nextion.print("n0.val=");
  nextion.print(PotiVal);
  nextion.write(NextiEnd, 3);
  delay(2000);
  }
}
```

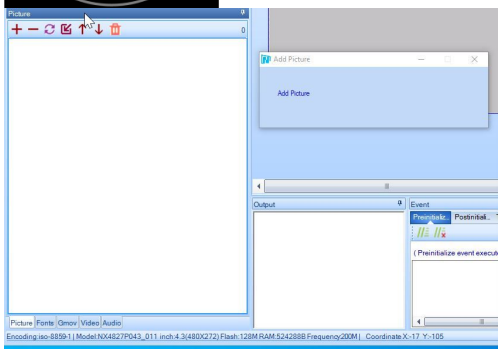
Potiwert lesen
Wert an n0 ausgeben

Skala mit Zeiger

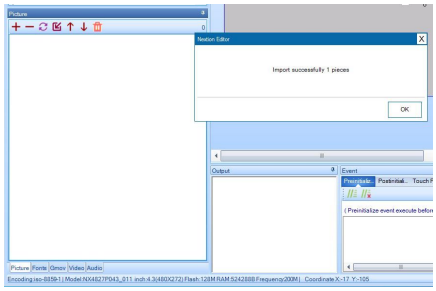


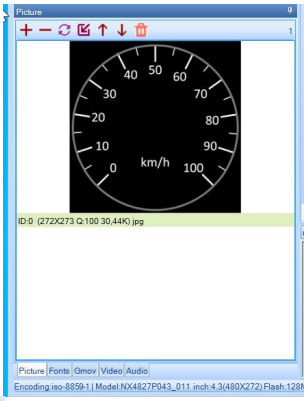
Die Skala ist mit einem Zeichenprogramm zu erstellen. Das Zeichenprogramm ist nicht Bestandteil des Nextion-Editors.

Entsprechend der Bildschirmgröße ist die Skala auf 272x272 Pixel zu verkleinern.

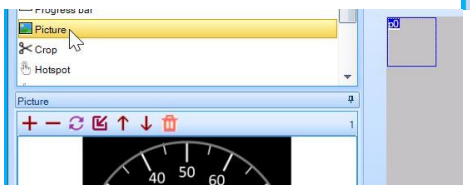


Im Nextion-Editor wird die Skala als Picture abgelegt.
Mit der Auswahl von + erscheint ein Auswahl Fenster,

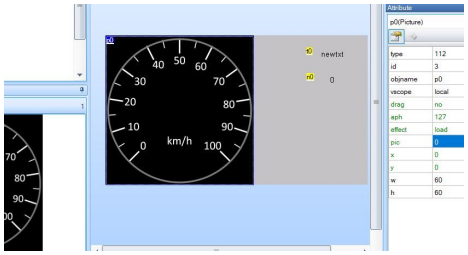




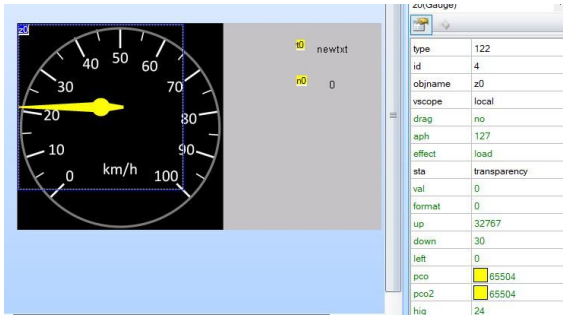
die Grafik wird importiert



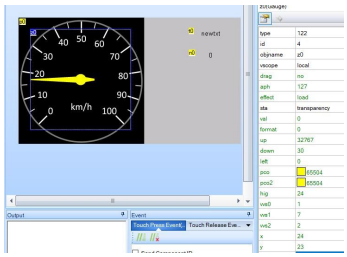
Um das grafische Objekt auf dem Display zu plazieren, wird Picture ausgewählt, es erscheint ein Quadrat mit dem Objektnamen p0



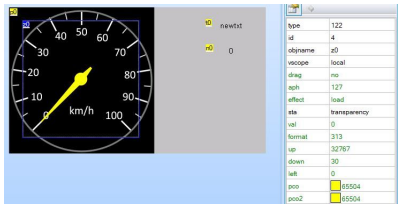
und dem Objekt p0 bei „pic“ die Grafik zugeordnet.



Der Zeiger (Gauge) ist ein eigenes Objekt z0.



Das Zeigerquadrat wird in die Mitte der Skala gezogen



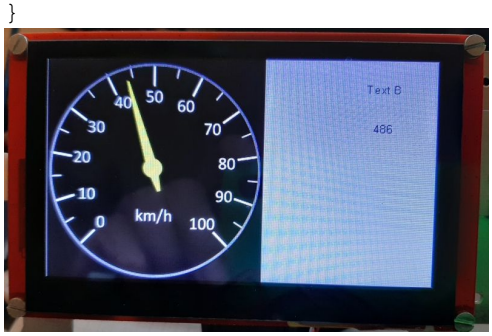
Die Anfangsstellung des Zeigers wird mit „format“ auf die Null der Skala gesetzt.

Zeiger mit Arduino-Programm bewegen

```
void loop() {
int PotiVal = 0;
for (int k=0; k < SDIM; k++) {
  nextion.print("t0.txt=\"");
  nextion.print(Sarray[k]);
  nextion.write(NextiSEnd, 4);

  PotiVal = analogRead(POTI);
  nextion.print("n0.val=");
  nextion.print(PotiVal);
  nextion.write(NextiEnd, 3);

  nextion.print("z0.val=");
  nextion.print(PotiVal/4);
  nextion.write(NextiEnd, 3);

  delay(2000);
  }
}
```

Programmergänzung in loop()

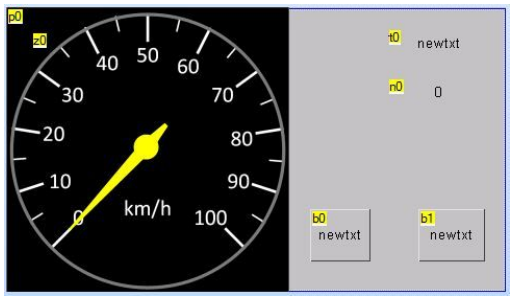
Zeiger ansprechen „val“ ist der Ausschlagwinkel ab der Null-Position



Der Zeiger folgt dem Potiwert

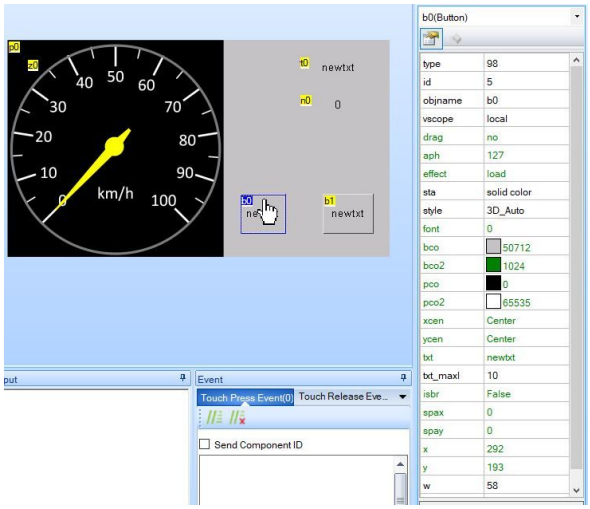
Touchscreen: Werte vom Display lesen

Bei einem Touch auf eine Taste sendet das Display Daten an den Arduino.
Diese Daten sind Strings dessen Werte mit dem Nextion-Editor konfiguriert werden.

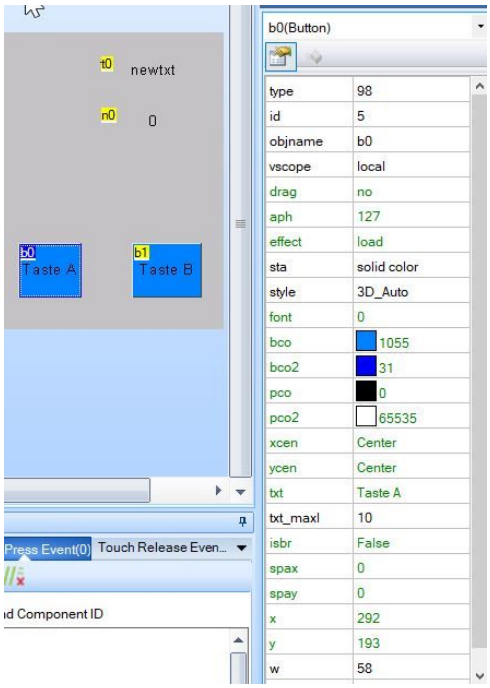


In der Toolbox wird „Button“ ausgewählt und die Tastenobjekte b0 und b1 auf das rechte Feld des Displays gezogen.

Die Größe des Rechteckfensters ist änderbar mit der Maus und durch die Werteeinstellung im Attribut-Fenster der Taste.



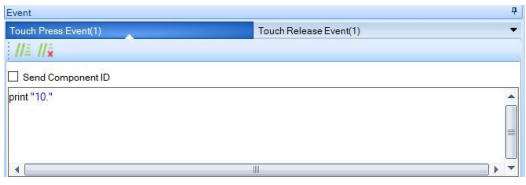
Mit dem Klicken auf eine Taste öffnet sich das dazugehörige Attribut-Fenster



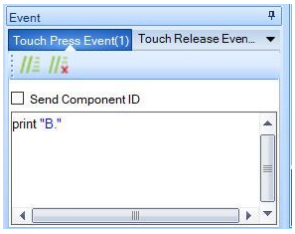
Die grünen Bezeichnungen in Attributen bedeuten, dass die Werte beim Start des Displays und während der Programmlaufzeit geändert werden können.

font	Testgröße und Font wählen
bco	Hintergrundfarbe der Taste in Ruhe
bco2	Hintergrundfarbe Taste gedrückt
pco	Textfarbe Taste in Ruhe
pco2	Textfarbe Taste gedrückt
txt	Tastenbeschriftung
x	Bildschirmposition der Taste
y	Bildschirmposition der Taste

Schwarze Bezeichnungen sind nur mit dem Editor änderbar und nicht zur Programmlaufzeit.



Zu jeder Taste gehört ein Event-Fenster. Sowohl beim Drücken der Taste als auch beim Loslassen kann eine Meldung ausgegeben werden.
Die Auswahl Touch Press Event gibt den String mit print aus, der im unteren Fensterbereich in den Anführungsstrichen steht.



Für die Taste b0 wird `print "A."` eingetragen, für die Taste b1 wird `print "B."` eingetragen.

Touch Release Event bleibt leer, beim Loslassen der Taste wird kein String ausgegeben.

Die String-Auswahl ist frei, zweckmäßig ist ein eindeutiges Ende-Zeichen, hier der Punkt.
Bei mehreren Tasten ist es erforderlich, die Zerlegung des Strings (Parser) zu planen.

Arduino-Programm Tastenabfrage und Zeiger bewegen

```
// Nextion Poti & Touch

#define nextion Serial2

#define VERSION "NexStart16"
#define POTI     A11
#define POTcyc   200

const uint8_t NextiEnd[3] = {0xFF, 0xFF, 0xFF};
const uint8_t NextiSEnd[4] = {0x22, 0xFF, 0xFF, 0xFF};

#define SDIM 3
char *Sarray[SDIM] = {"Text A","Text B","Text C"};
char   myString[10] = {0};


void setup() {
Serial.begin(115200);
Serial.println(VERSION);

nextion.begin(9600);

nextion.print("rest");
nextion.write(NextiEnd, 3);
delay(1000);

nextion.print("t0.txt=\"");
nextion.print(VERSION);
nextion.write(NextiSEnd, 4);
}

void loop() {
PotiRD();
NextionFetch();
}

void PotiRD (void) {
static unsigned long bevT    = 0;
                int PotiVal = 0;
unsigned long aktT = millis();
if (aktT - bevT > POTcyc) {
   bevT = aktT;

  PotiVal = analogRead(POTI);
  nextion.print("n0.val=");
  nextion.print(PotiVal);
  nextion.write(NextiEnd, 3);

  nextion.print("z0.val=");
  nextion.print(PotiVal/4);
  nextion.write(NextiEnd, 3);
  }
} // PotiRD()
```

Für debug-Zwecke

Befehl Nextion-Display: Reset

Zyklische Aufrufe Poti und Nextion-Read

Poti-Abfrage zyklisch alle 200ms

Poti-Wert auf Display-Anzeige

Poti-Wert Umsetzung auf Zeigerausschlag

```
void NextionFetch(void) {
       uint8_t bdata = 0;
static int     i     = 0;

  if (nextion.available() > 0) {
     bdata = nextion.read();
     Serial.print(bdata,HEX); Serial.print(' ');

     if (bdata & 0x80) return;

     switch (bdata) {
     case 0:
            break;
     case 0x1A:
          return;
     case '.':
            myString[i] = 0;
            NextionParser(myString, i);
            myString[0] = 0;
            i=0;
            break;
     default:
            myString[i++] = bdata;
            break;
     } // switch bdata
  } // if nextion.available
} // NextionFetch()
```

Debug-Ausgabe empfangene Bytes

Unterdrückung der Start-Ausgaben des Displays

String-Ende

String zusammenstellen

```
void NextionParser(char *nexStr, uint8_t sEndPtr) {
int k = 2;
switch (nexStr[0]) {
 case 'A':
      k = 0;
      break;
 case 'B':
      k = 1;
      break;
 default:
      break;
 }
 nextion.print("t0.txt=\"");
 nextion.print(Sarray[k]);
 nextion.write(NextiSEnd, 4);
}
```

Einfacher Parser für zwei Tasten

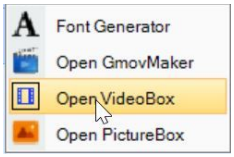
Reaktion auf Tastendruck

Sounds ausgeben

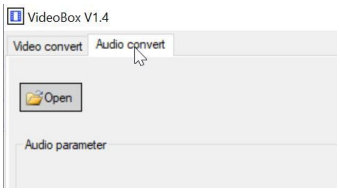
Vorbereitungen

```
526.000       1.wav
429.232       2.wav
355.504       3.wav
526.000       4.wav
475.312       5.wav
530.608       6.wav
456.880       7.wav
424.624       8.wav
```

Verschiedene Sounds müssen als .wav Dateien vorliegen.
Die Beispieldateien enthalten die gesprochen Ziffern der Dateinamen.

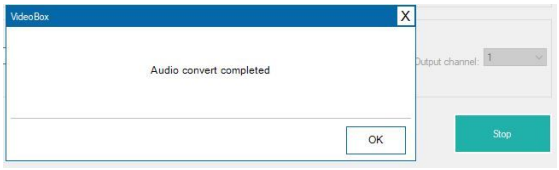


Die Sounds werden durch die VideoBox in der Dateigröße verringert,
Menü Tools, Open VideoBox

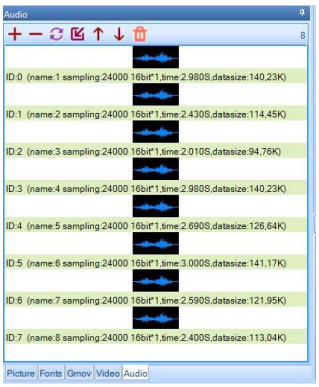


Auswahl Audio convert,

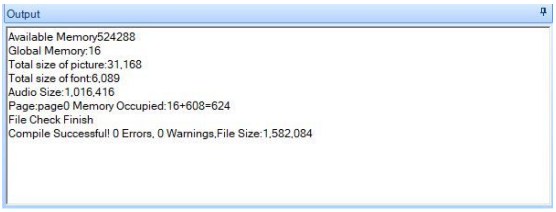
Verzeichnis mit den wav-Audio-Dateien öffnen,
(Verzeichnis suchen, neuen Ordner anlegen (wenn gewünscht)
Button Start to convert klicken



Meldung abwarten, OK



Im Ressourcen-Editor (Fenster links unten) Audio auswählen und die Dateien aufnehmen



Compilieren und tft-File erzeugen,
Upload in das Display

Sound-Ausgabe einleiten vom Arduino

```
void audio(int f) {
char   s[8] = {};
itoa(f,s,10);
nextion.print("play 1,");
nextion.print(s);
nextion.print(",0");
nextion.write(NextiEnd, 3);
}
```

f: Soundnummer

int in String wandeln, Zahlenbasis 10
Nextion Audio-Ausgabe mit `play`

```
void volume(int vol) {
nextion.print("volume=");
nextion.print(vol);
nextion.write(NextiEnd, 3);
}
```

Lautstärke ändern mit `volume`
Wertebereich 0, …, 100

```
volume(25);
audio(0);
```

Ergänzung in setup()

```
void NextionParser(char *nexStr, uint8_t
sEndPtr) {
static int sound = 0;
int k = 2;
switch (nexStr[0]) {
 case 'A':
      k = 0;
      sound++; if (sound > 7) sound = 0;
      break;
 case 'B':
      k = 1;
      sound--; if (sound < 0) sound = 0;
      break;
 default:
      break;
}
 nextion.print("t0.txt=\"");
 nextion.print(Sarray[k]);
 nextion.write(NextiSEnd, 4);

 audio(sound);
}
```

Ergänzung NextionParser()

Taste A, Soundnummer erhöhen

Taste B, Sondnummer vermindern

Sound ausgeben

Literatur

https://nextion.tech/editor_guide/

https://nextion.tech/instruction-set/

https://seithan.com/Easy-Nextion-Library/Use-Nextion-General-View/