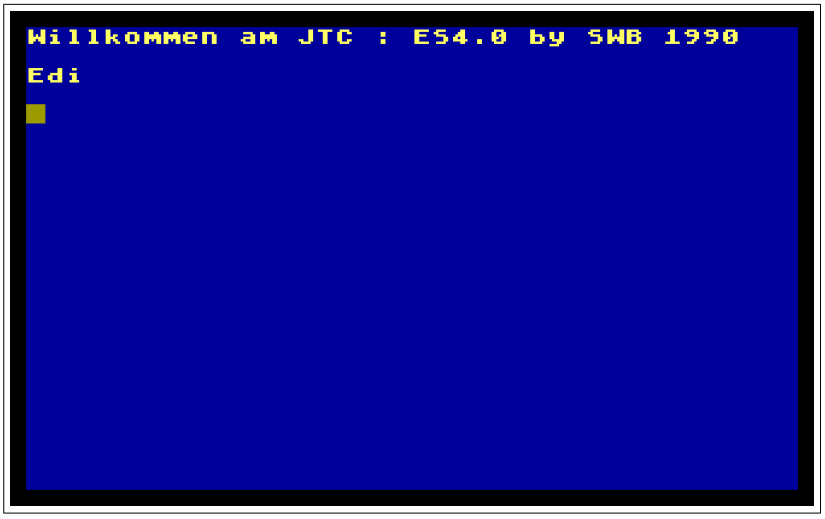
JuTe-6k

Infosammlung

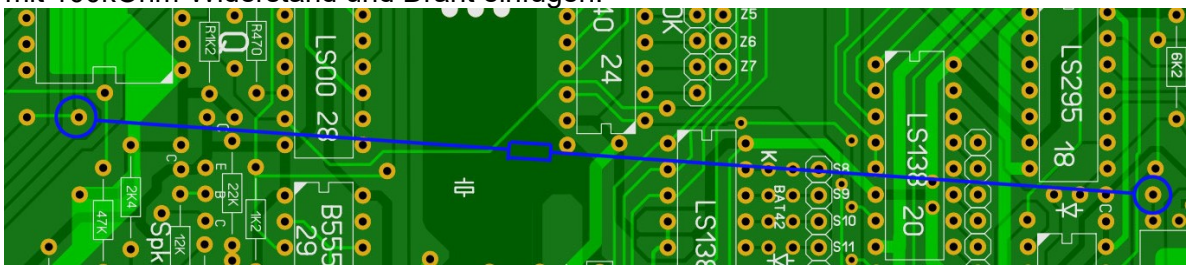
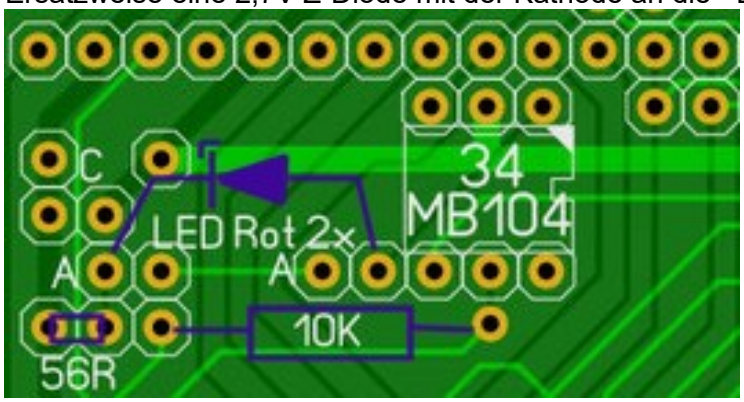
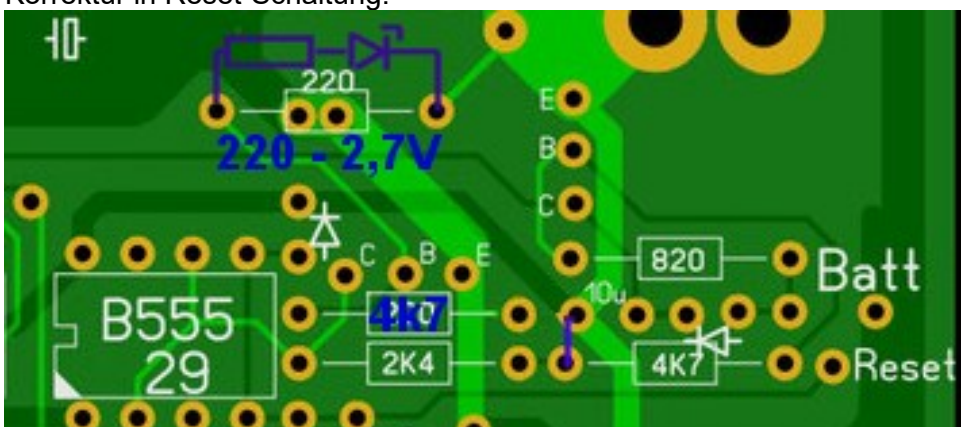


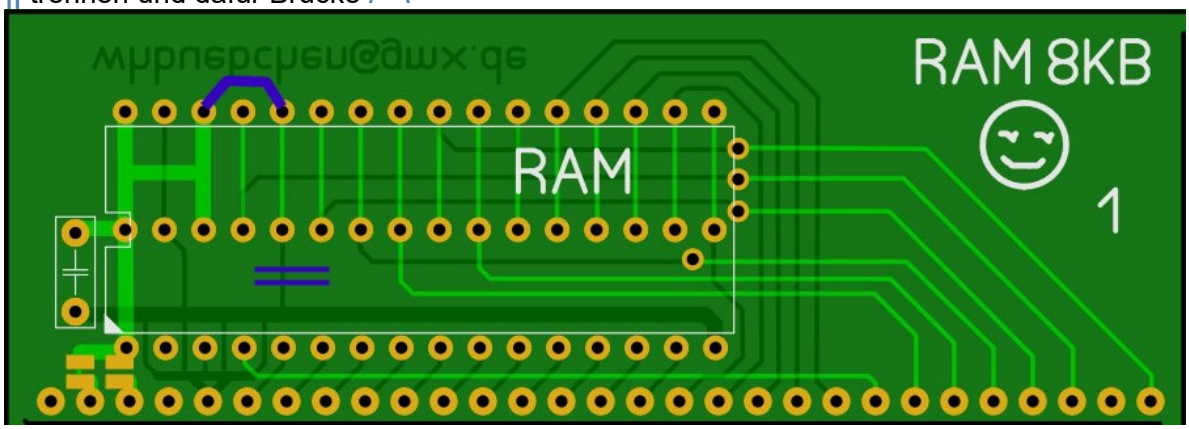
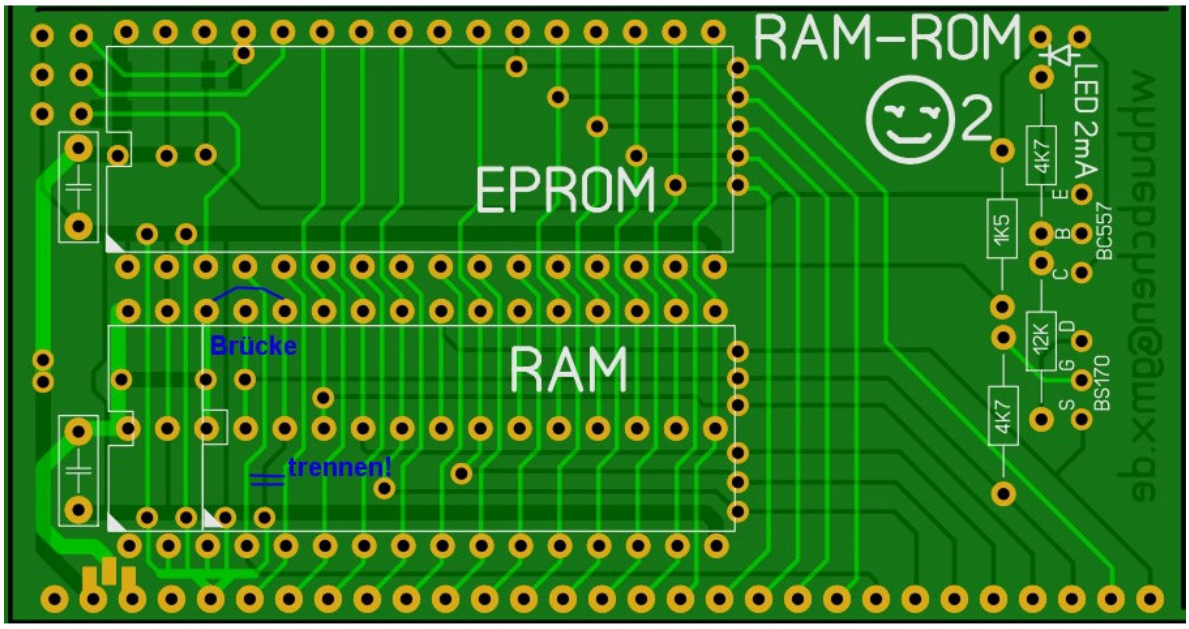
Eckdaten:

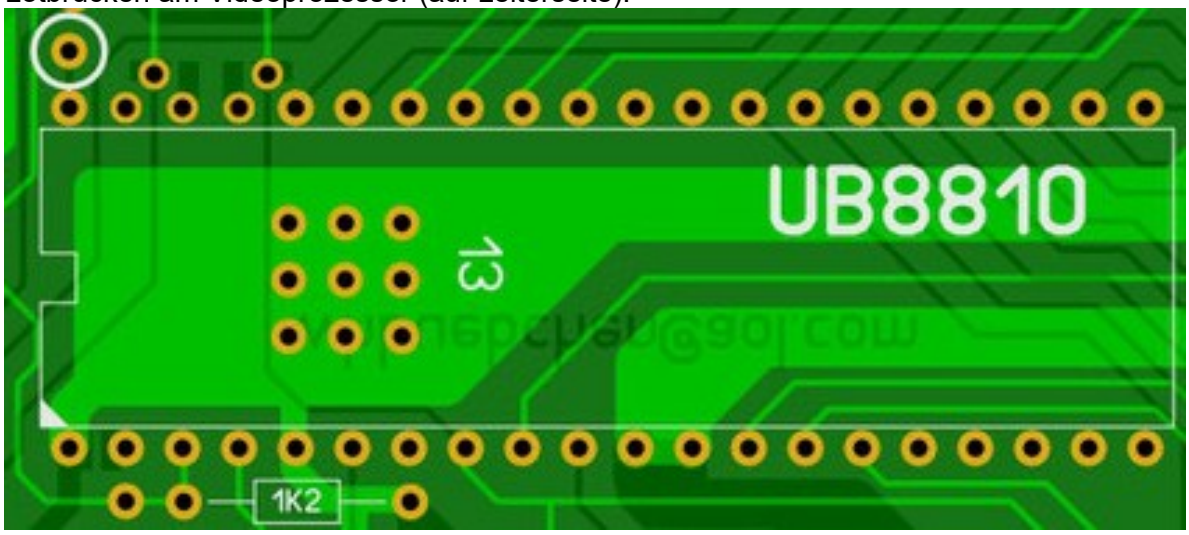
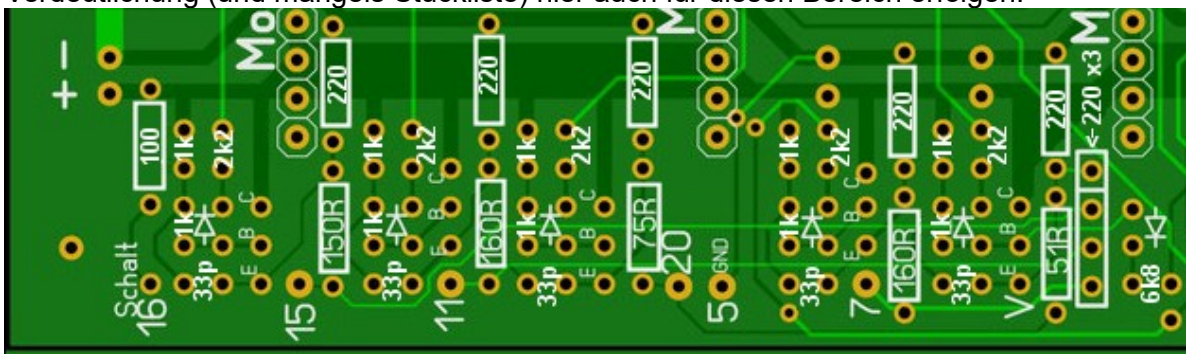
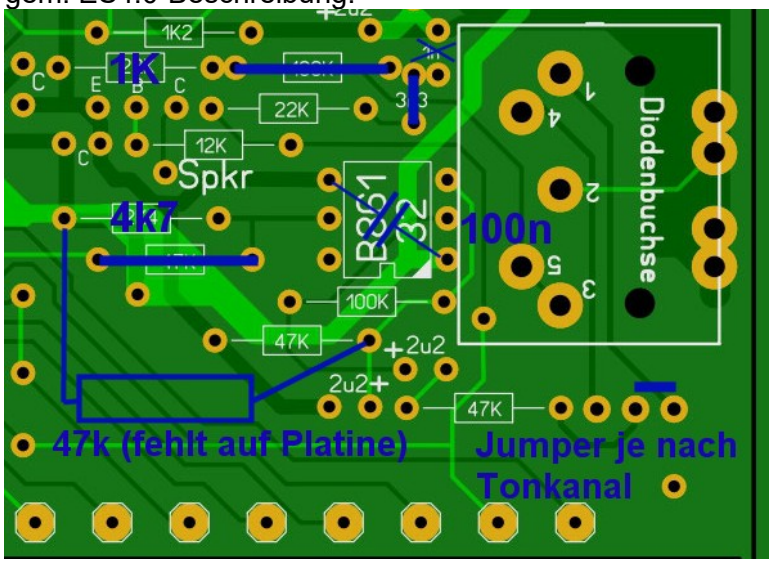
Betriebssystem	ES 4.0 oder 4.5
Bildschirm	40 Zeichen x 24 Zeilen, 320x192 Pixel, 16 Farben für jedes Pixel
Betriebsspannung	5V/ max. 1,5 A (Anschluss Hohlbuchse 5,5x2,1)
...	

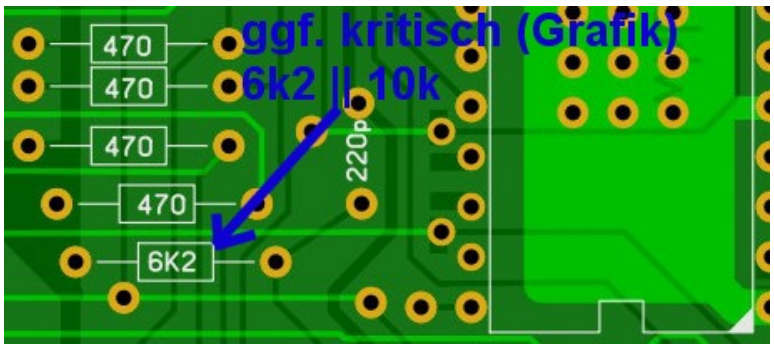
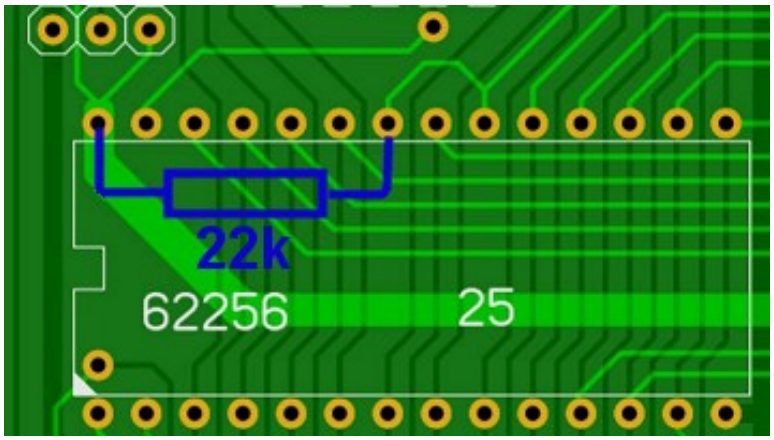
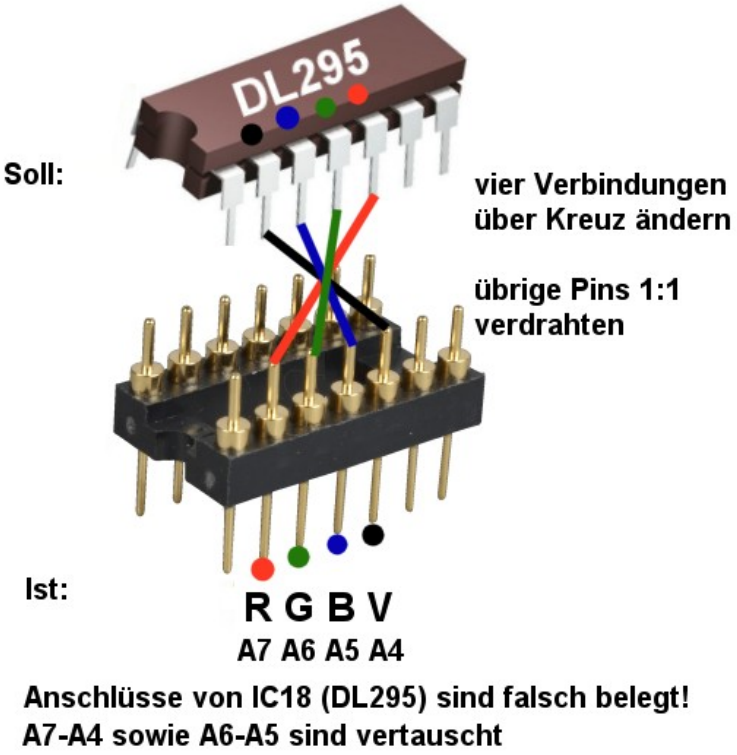
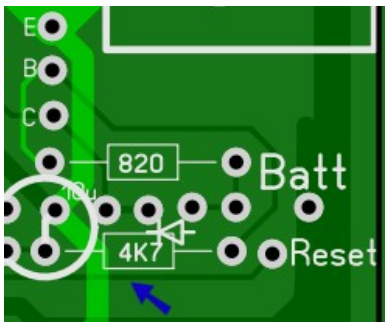
Neuauflage Platine JuTe-6k: 2018/2019/2021, Wolfgang Harward/Wolfgang Kott

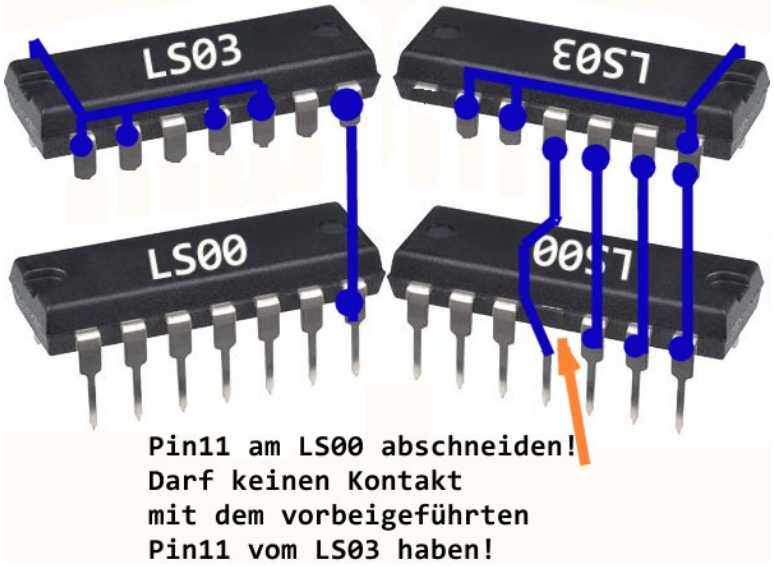
Inhaltsverzeichnis

Korrekturen/Ergänzungen/Hinweise	**Stand: 13.12.2021**
1	Unter IC23 fehlt eine Verbindung auf der Bauelementseite → kratzen + Lötklecks 
2	Signal /BUSY: Verbindung zwischen Hauptprozessor/Pin13 und Videoprozessor/Pin40 mit 100kOhm-Widerstand und Draht einfügen:  Beachte: Trotz dieser Verbindung treten mehr oder weniger starke Störungen bei der Bildausgabe auf, die auch von der Farbgebung abhängen. Das ES4.0 fragt diese BUSY-Leitung nur in wenigen Fällen ab; der Anwender muss das in seinen Programmen tun!
3	Mini-LEDs (2x2mm) vor dem Optokoppler (VQA15, ca. 1.3V bei 10mA) Ersatzweise eine 2,7V Z-Diode mit der Kathode an die +Leitung 
4	falscher Bestückungsaufdruck: - R neben IC13: nicht 62k sondern 6,2k - R für Schaltspannung: nicht 1,5k sondern 100R
5	Korrektur in Reset-Schaltung:  oben: statt 220Ohm → freitragend 220Ohm in Reihe mit Z-Diode 2,7V (Kathode an 5P) unten: statt 220Ohm → 4k7 unten rechts: Verbindungsstelle 2k4 ↔ 4k7 muss an 5P

6	RAM-Module: für '6264 Verbindung an Pin26 ändern (blau): || trennen und dafür Brücke / ‾ \  x Brücke für 5P
7	Brücken auf den universellen RAM-ROM-Modulen: - Fehler für '6264: trennen und Brücke wie bei RAM-Modul - Konfigurationsbrücken je nach EPROM-Größe: **auf Leiterseite:** **Brücke bei 32k*8 (A13=0)** **Brücke bei 32k*8 (Vpp=5P)** **Brücke bei 8k*8 (/PGM=5P)**  **auf Bestückungsseite:** **5P-Brücke für getrennte Versorgung CMOS-RAM** **5P-Brücke normal** Die 5P-Brücken sind im JuTe-6k bei Verwendung als Video-RAM unerheblich (wahlweise). Im Rechnerteil ist bei einer Bestückung mit 62256 bereits die Maximalvariante erreicht, sodass dort kein RAM-Modul eingesetzt werden muss. Die Beschaltung rechts ist optional aufzubauen. Sie enthält eine LED, mit welcher der Zugriff auf das Modul anzeigt wird.
8	Lautsprecheranschluss: Zwei Bauelemente (220 Ohm und 100µF) sind nachzurüsten siehe Anschluss Lautsprecher

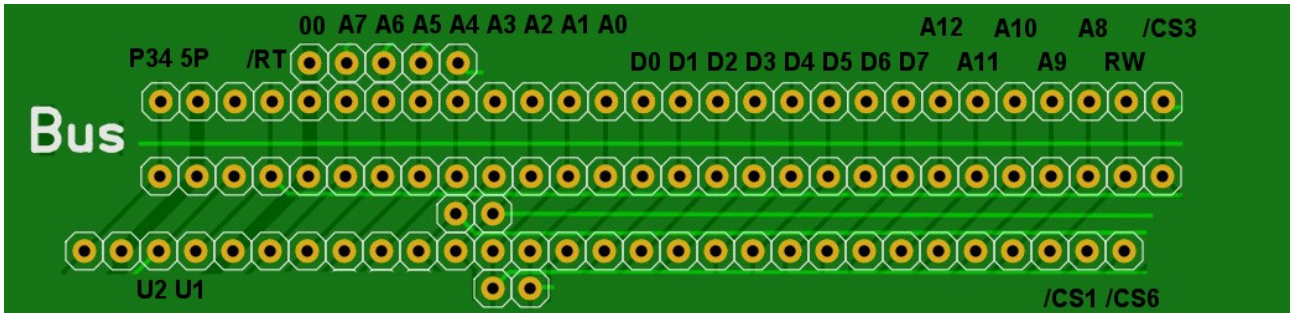
9	Lötbrücken am Videoprozessor (auf Leiterseite):  Bei Verwendung eines 8811, 8831 oder 8861 (low power Versionen) ist die Lötbrücke zwischen Pin 1 und 2 zu setzen. Die beiden anderen Lötbrücken (an Pin37 und Pin38) waren vermutlich für Erweiterungen vorgesehen. Sie haben aktuell für ES4.0 keine Bedeutung und bleiben offen.
10	Video-Ausgangsstufen: Da alle übrigen Bauelemente mit ihren Werten auf der Platine vermerkt sind, soll das zur Verdeutlichung (und mangels Stückliste) hier auch für diesen Bereich erfolgen.  Die Ziffern am unteren Rand sind gleichbedeutend mit den Anschlüssen am Scartstecker.
11	Kassetteninterface: Für einen problemlosen Betrieb empfiehlt sich der Umbau gem. ES4.0-Beschreibung:  - 1nF ersatzlos entfernen - 3,3nF durch Brücke ersetzen - 100k durch Brücke ersetzen - 22k durch 1k ersetzen - 2k4 durch 4k7 ersetzen - 47k durch Brücke ersetzen - 100nF am B861 Leiterseite - fehlenden 47k nachrüsten Jumper je nach vorhandenem TB-Gerät/Kabel-Anschluss (Funktion: Laden vom Band)

12		Bei Grafikproblemen (direkt geschriebene Grafiken werden nicht angezeigt) kann eine geringfügige Änderung des 6k2-Widerstandes helfen: bei mir: Parallelschalten 10k
13		Batteriestütze funktioniert nicht korrekt (zu hohe und zufällige Ruhestromaufnahme): Widerstand zwischen Pin 22 und Pin 28 am SRAM fehlt
14	 **Anschlüsse von IC18 (DL295) sind falsch belegt!** **A7-A4 sowie A6-A5 sind vertauscht**	Im Farbkomplex gibt es leider einen Layoutfehler, der zu falscher Farbdarstellung führt. Mit nebenstehendem Adapter auf der Position von IC18 kann der "Soll-Zustand" hergestellt werden, ohne an der Platine selbst Änderungen vornehmen zu müssen.
15		Der mit blauem Pfeil markierte Widerstand ist nicht nötig. Zweck? Weglassen vermindert die Stromaufnahme aus der Stützbatterie um ca. 800 µA.

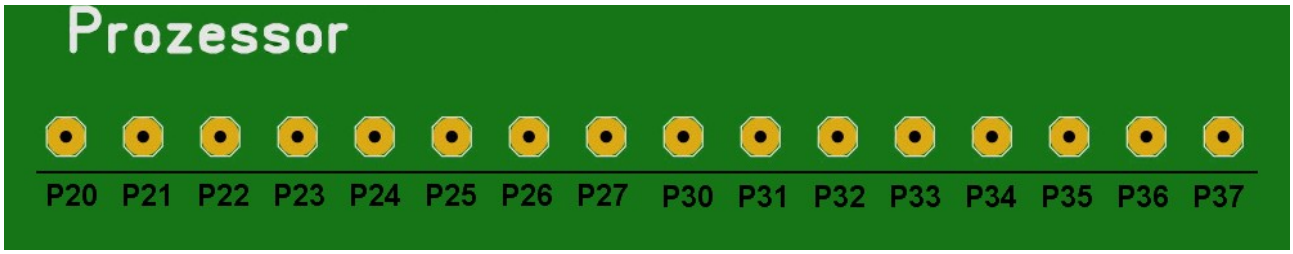
16		Im Zusammenhang mit der Stützbatterie kann der Strom drastisch gesenkt werden, indem das Gatter D28D durch ein open-collector-Gatter (LS03) ersetzt wird. Das ist als Huckepacklösung möglich ohne Eingriff auf der Platine: D28 (LS00) entfernen und durch „Turm“ ersetzen. **Wird auch der auch o.a. 4k7-Widerstand entfernt, dann liegt der Batteriestrom nur noch bei ca.0,5µA, je nach sRAM!**
17	Stromaufnahme gemessen: '8212, '8682: 600 mA +DL540: 690 mA +'LS138: 720mA +LS295/00: 740 mA +RAM & Eprom: 780 mA +U883, U886: ca.1,2 A	Stützbatterie: ca. 0,5 µA

Hardwareinfos

Busbelegung



Portbelegung



Port 2 (P2.0...P2.7) frei für universelle Verwendung!

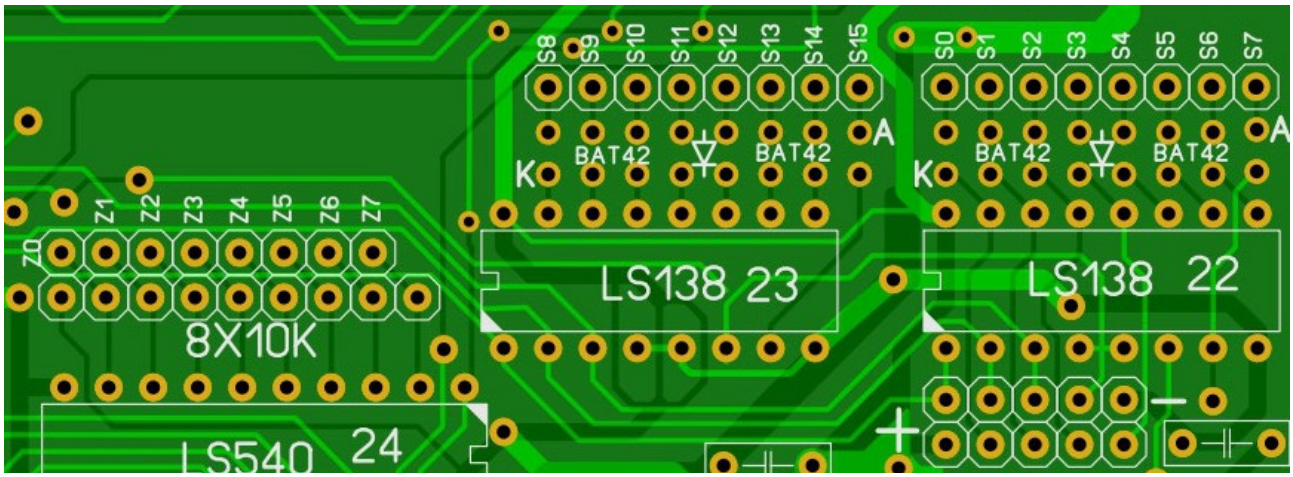
Anschluss SCART-Stecker für Sichtgerät

Signal	Scart-Pin	Anmerkungen
Masse	5	
Blau	7	
Grün	11	
Rot	15	
Schaltspannung	16	AV/RGB-Umschaltung: 0..0,4 V (low = FBAS), 1..3 V (high = RGB)
composit Video (in)	20	Bild-/Austast-/Synchron-Signal

Je nach verwendetem TV-Gerät ist ggf. auch SCART/Pin8 zu beschalten. Damit wird die TV/AV-Umschaltung sowie das Seitenverhältnis gesteuert:
Level 0 = 0..2 V = TV (Standard), → GND
Level 1A = 4,5..7 V = AV (16:9), → 5P
Level 1B = 9,5..12 V = AV (4:3) → externe 12V Einspeisung nötig!

Anschluss Tastatur

S0...S15 Spalten
Z0...Z7 Zeilen



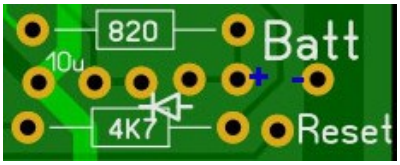
Die Anschlussgruppe (Buchsen-/Stiftleisten) unterhalb von IC22 war für den Anschluss eines PS/2-Tastaturadapters (E. Mueller) vorgesehen. **Dieser (für den normalen JuTe gedacht) ist am JUTE-6k in dieser Form jedoch nicht geeignet!**

Nötige Änderungen:

- IC22 und IC23 nicht bestücken
- Drahtverbindungen zwischen den Pinleisten S0→Z4, S1→Z5, S2→Z6, S3→Z7

Der Adapter ist jedoch auch damit nur bedingt einsetzbar, da nicht alle Tastaturcodes erzeugt werden (z.B. fehlen die Funktionstasten). Es müsste also das PIC-Programm geändert werden…

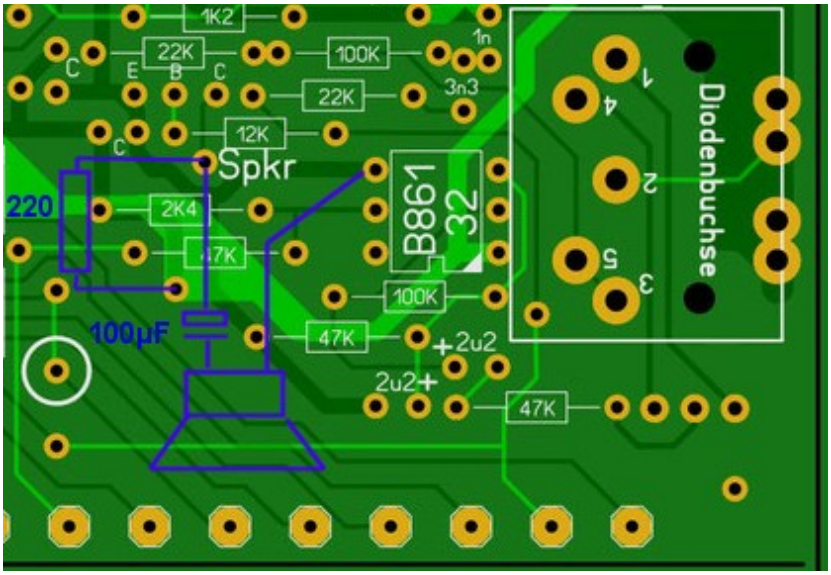
Batterie-Stütze und Reset-Taste



Batterie: 3,6V-Akku (3x1,2V)
RAM '62256 mit geringer Stromaufnahme auswählen!
wenn keine Batterie: ca. 2kOhm von + nach – nötig!

RESET-Taster gegen Masse (-)

Anschluss Lautsprecher



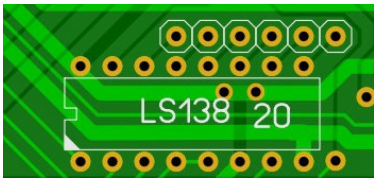
Transistor: SF126 o.ä.

Statt des Lautsprechers (ca. 8 Ohm) kann auch z.B. der Tonkanal des SCART-Anschlusses benutzt werden.

Widerstand 220 Ohm und Elko 100µF dann ebenfalls erforderlich!

Selects

Aus dem Auswahlsignal /CS3 (0 wenn %6000…%7FFF) und den Adressbits A10, A11, A12 werden 8 Steuersignale dekodiert. Diese Signale sind low aktiv, wenn sich die Prozessoradresse im entsprechenden Adressbereich befindet.

%6000...63FF = Videosteuerung (Laden des Farbebenen-Registers)		
%6400...67FF = select 1 **…** **…** **...** **%7800...7BFF = select 6**	1 2 3 4 5 6 	Steuersignal-Leitungen für eigene Erweiterungen
%7C00...7FFF = /KEY (Tastaturabfragesignal)		

EPROM-Erweiterung

Der System-EPROM im Rechnerteil belegt die Adressen (0000...)%0800…%1FFF. Es wird im Normalfall ein 8k*8 EPROM ('2764A) eingesetzt. Das Layout sieht vor, dass auch ein 27C128 zum Einsatz kommen kann. P34 liefert ein Signal zur Bankumschaltung (Adresse A13), womit der obere Teil des 27C128 in den Bereich %0800…%1FFF eingeblendet wird. Da hierdurch jedoch das OS abgeschaltet wird, ist eine universelle Nutzung etwas aufwändiger und ggf. problembehaftet.

Für Erweiterungen ist es einfacher, ein EPROM-Modul mit '2764A im Rechnerteil (einzig freier Platz direkt am Bus) zu stecken. Damit wird der (sonst ungenutzte) Adressbereich %2000… %3FFF belegt. Darin lassen sich nun Anwenderprogramme unterbringen.

Im Falle von BASIC-Programmen werden diese zunächst aus dem EPROM in den RAM kopiert:

M <Enter>	Mon aufrufen
Maaaa E000 bbbb <Enter>	Umkopieren mit aaaa=Anfangsadresse und bbbb=Länge je nach Programm-Daten im EPROM
Q <Enter>	Mon verlassen (zurück zu Edi)
R <Enter>	Basicprogramm starten

Wer es komfortabler haben wil, der schreibt sich eine Routine, die den Programmen im EPROM vorgelagert ist.

- Sie wird aus Mon z.B. mit J2000 gestartet.
- Ein Auswahlmenü wird anzeigt,
- nach der Auswahl der entsprechende EPROM-Bereich automatisch umkopiert und
- Mon verlassen und ein "Auto-Run" vorgenommen.

Software

Adressraum

Adresse	Bedeutung	Anmerkungen
%0000-%07FF	interner ROM	tiny-MP-Basic des UB883
%0800-%1FFF	EPROM 1	Betriebssystem (ES 4.x)
%2000-%3FFF	EPROM 2	/CS1=0 → Steckplatz am Prozessorteil (frei, z.B. für EPROM-Bank)
%4000-%5FFF	Video-RAM	Pixelspeicher 4 x 8kB parallel (R/G/B/V)
%6000-%7FFF	Steuerung	/CS3=0 → Video- u. Tastatursteuerung
%8000-%FFFF	RAM	/CS4=0 → 62256 ist aktiv

RAM-Belegung:

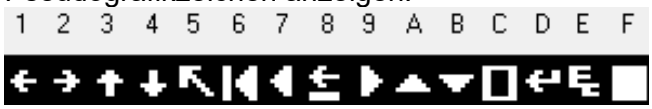
Adresse	Anmerkungen
%8000-%F4FF	Anwenderprogramme, obere Grenze beachten!!!
%B000-%B7FF	2. Zeichensatz
%C000-%CFFF	FLOAT-Variablenspeicher
%C100-%C9FF	ASDIS
%D000-%E4FF	FORTH
%D000-%DFFF	BASIC-Erweiterungen FLOAT (%DAFA), ab DB20 frei: Kreis (%DB...
%E000-%EFFF	BASIC-Quellprogramm (max. 4K)
%F100-%F3FF	KCTRANS
%F500-%FFFF	System-/Arbeitszellen:
%F512-%F579	Druckertreiber (siehe auch %17A6)
%F7A0	Bitmaske Textfarbe
%F7A1	Bitmaske Kursorfarbe
%F800-%FBBF	ASCII-Textspeicher Bild 2
%FC00-%FFBF	ASCII-Textspeicher Bild 1
%FFC0-%FFFF	IO-Vektortabelle

Tastencodes

Die Tastaturabfrage erfolgt durch das ES4.0, indem der Prozessor eine Leseoperation einer Adresse im Bereich %7C00...7FFF tätigt. Je nach konkreter Adresse (Adressbites A0...A3 sind relevant) liegt ein bestimmtes Spaltensignal an der Tastaturmatrix an. Aber nur in o.a. Adressbereich ist das Signal /KEY aktiv und ermöglicht das Einlesen des sich aus der gedrückten Taste ergebenden Zeilenwerts. Nach Vergleich mit der im EPROM enthaltenen Tastaturtabelle wird aus der gedrückten Taste der zugehörige Code ermittelt.

normal	%00…%0F	Steuerzeichen
	%10…%1F	Pseudografik (Großzahlen, deutsche Umlaute)
	%20…%7F	Zeichen lt. ASCII-Tabelle
F-Tasten	%80…%87	
SHT	%C1…%C5	(bei 6 Ebenen)

Steuerzeichen

Code	Name	Bedeutung
00		(nicht belegt)
01		Kursor links
02		Kursor rechts
03		Kursor hoch
04		Kursor runter
05	HOM	Kursor in linke obere Ecke
06	SOL	Kursor an Zeilenanfang
07	DEL	Zeichen unter Cursor löschen
08	DBS	Zeichen links von Kursor löschen ("Backspace")
09	INS	Zeichen einfügen
0A	LDE	Zeile löschen
0B	LIN	Zeile einfügen
0C	CLS	Bildschirm löschen
0D	RET	Eingabe abschließen
0E	ESC	Nächstes Zeichen nicht als Steuerzeichen interpretieren, sondern das zugeordnete Pseudografikzeichen anzeigen:
		
0F		(nicht belegt)

Ablagebereich: Achtung, gilt nur für ES 4.0, in ES4.5 liegen die Adressen woanders!!!???

Adresse	Bedeutung	Anmerkungen
%1B22	Zeilenzahl	4-Zeilen-Tastatur: %04 (max. 6 Ebenen zu je 64 Bytes)
%1D00-1D3F	Grundebene	Die Tastencodes werden, beginnend mit der obersten Zeile, von links nach rechts, (d.h. nach aufsteigender Spalte) abgelegt. Unbelegte Positionen enthalten %FF. Die SHT-Tasten und Kleinbuchstaben sind in der Grundebene (kleine Buchstaben werden in Grundeinstellung in große umgewandelt = Caps aktiv).
%1D40-1D7F	SHT1 → SHIFT	
%1D80-1DBF	SHT2 → Strg	
%1DC0-1DFF	SHT3 → Alt/AltGr	
%1E00-1E3F	SHT4	
%1E40-1E7F	SHT5	

5- bis 8-Zeilen-Tastatur: auf %1B22: %05 bis %08 (Zeilenzahl)
max. 3 Ebenen zu je 128 Byte
Grundebene: %1D00-1D7F
SHT1 %C2 %1D80-1DFF
SHT2 %C4 %1E00-1E7F

Beispiel für EPROM-Belegung (Tastatur 12x4):

Tastaturbelegung

Diese Spalte war beim 2K/4K-System nicht nutzbar.															Taste
	noch nicht belegt	! DBS ! 1	2 INS " 2	3 CLS # 3	4 $ 4	5 ä % 5	6 ö & 6	7 ü ' 7	8 Ä @ 8	9 Ö (9 | 0 Ü) 0	ß ^ > <				←mit SHT3 ←mit SHT2 ←mit SHT1 ←allein
	SHT3	F1 DEL Q q	F2 ↑ W w	F3 SOL E e	F4 LDE R r	F5 T t	F6 Z z	F7 U u	F8 I i	O o	P p	♦ \ - +				
	SHT2	← A a	HOM S s	→ D d	LIN F f	G g	H h	J j	K k	L l	: ;	| / *				
	SHT1	Y y	↓ X x	C c	ESC V v	B b	N n	M m	([, |)] .	_ = Space	~ ? RET				

Space ist die Leertaste

```
         0  1  2  3  4  5  6  7   8  9  A  B  C  D  E  F
1CD0h: DF 8B 1C 3A 09 DF 76 E4  80 6B F6 CF 8B F3 3A 09   ...:..v..k....:.
1CE0h: CF 76 E5 04 6B EB DF 8B  E8 DF 42 55 DB 01 CF 10   .v..k.....BU....
1CF0h: 74 10 75 2C 74 1C A8 3C  04 93 20 3A FC 50 FD AF   t.u,t..<.. :.P..
1D00h: FF FF 31 32 33 34 35 36  37 38 39 30 3C FF FF FF   ..1234567890<...
1D10h: FF C3 71 77 65 72 74 79  75 69 6F 70 2B FF FF FF   ..qwertyuiop+...
1D20h: FF C2 61 73 64 66 67 68  6A 6B 6C 3B 2A FF FF FF   ..asdfghjkl;*...
1D30h: FF C1 7A 78 63 76 62 6E  6D 2C 2E 20 0D FF FF FF   ..zxcvbnm,. ....
1D40h: FF FF 21 22 23 24 25 26  27 40 28 29 3E FF FF FF   ..!"#$%&'@()>...
1D50h: FF FF 51 57 45 52 54 59  55 49 4F 50 2D FF FF FF   ..QWERTYUIOP-...
1D60h: FF FF 41 53 44 46 47 48  4A 4B 4C 3A 2F FF FF FF   ..ASDFGHJKL:/...
1D70h: FF FF 5A 58 43 56 42 4E  4D 5B 5D 3D 3F FF FF FF   ..ZXCVBNM[]=?...
1D80h: FF FF 08 09 0C FF 1A 1B  1C 1D 1E 1F 5E FF FF FF   ............^...
1D90h: FF FF 07 03 06 0A FF FF  FF FF FF FF 5C FF FF FF   ............\...
1DA0h: FF FF 01 05 02 0B FF FF  FF FF FF FF 7C FF FF FF   ............|...
1DB0h: FF FF 0F 04 FF 0E FF FF  FF 7B 7D 5F 7E FF FF FF   .........{} ~...
1DC0h: FF FF 11 12 13 14 15 16  17 18 19 10 7F FF FF FF   ............▯...
1DD0h: FF FF 80 81 82 83 84 85  86 87 FF FF 60 FF FF FF   ............`...
1DE0h: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF   ................
1DF0h: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF   ................
1E00h: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF   ................
1E10h: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF   ................
1E20h: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF   ................
1E30h: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF   ................
1E40h: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF   ................
1E50h: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF   ................
1E60h: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF   ................
1E70h: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF   ................
```

Grundebene

SHT1

SHT2

SHT3

SHT4

SHT5

Funktionstasten

- Die Funktionstasten F1 bis F8 rufen aus der Abfrageroutine heraus jeweils ein Maschinenprogramm auf. Sie sollten deshalb nur betätigt werden, wenn man sicher ist, daß dieses Programm auch existiert.
- Die Funktionstasten sind nicht vom Autorepeat betroffen. Ihre Betätigung wird nach der Ausführung des Funktionstastenprogramms mit einem 0.5 s langen Piep quittiert.
- Die zugeordneten Programme dürfen den Registerpointer %FD nicht verändern.

Eine Standardbelegung für die Funktionstasten ist bereits im ES4.0 ab %EB0 im EPROM gespeichert. Diese kann man vom MON aus mit M 1EB0 F7E0 0020 in den RAM kopieren und enthält folgende Funktionen:

	Ansprung	Inhalt	Funktion
F1	%F7E0	XOR %6C,#%40 RET	Tastaturpiep umschalten (an/aus)
F2	%F7E4	XOR %6C,#%20 RET	"Caps" (klein-)groß Buchstabenwandlung an/aus bei Tastatureingabe
F3	%F7E8	OR %55,#%20 RET	Drucker an
F4	%F7EC	AND %55,#%DF RET	Drucker aus
F5	%F7F0	JP %1E80 RET	Text-Bildschirm wechseln.
F6	%F7F4	XOR %55,#08 RET	Tastencodeverschiebung (an/aus) = "Grafiktaste" Diese ist normalerweise aus und man erhält die normalen Zeichen von der Tastatur. Ist sie eingeschaltet, dann wird zum ASCII-Kode noch %80 addiert und man erhält, solange der Zeichengenerator nicht erweitert ist, sehr merkwürdige Zeichen, die nicht in Basicprogramme eingebaut werden dürfen.
F7	%F7F8	XOR %55,#%10 RET	original nicht belegt. Nutzung z..B. für Rollen ein/aus
F8	%F7FC	JP %0812	original nicht belegt Nutzung z.B. für Soft-Reset

Automatisches Umkopieren der Belegung:

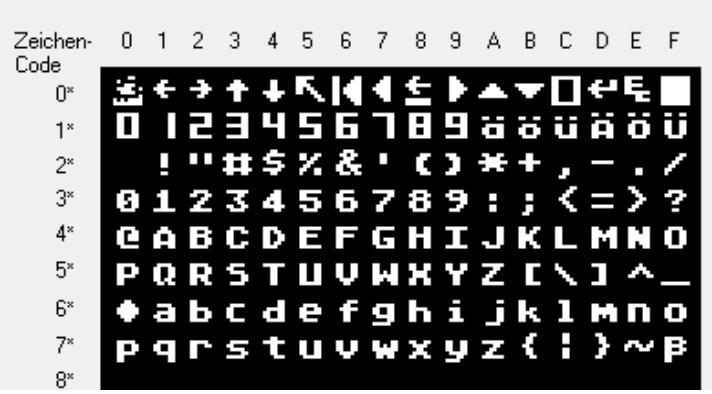
```
            ...
            CALL  #%082D      ;PRISTRI
            DB    %0C
            DB    "Willkommen am JTC : ES4.0 by SWB 1990"
            DEB   %0D,0
            CALL  FCOPY       ;hier F-Tasten umkopieren
            JP    m0949       ;-> EDI
FCOPY:      ...(eigene Routine nötig)
```

Zeichensatz

Der Zeichensatz scheint für einen Bildschirm mit 80x24 Zeichen "optimiert" zu sein (vierte Ausbaustufe). Unter der kleinen Auflösung 320x192 Zeichen sieht er etwas "fett" aus:

Der Standardzeichensatz enthält 127 Zeichen. Auf 00 ist kein "richtiges" Zeichen, sondern das dargestellte Ende des vorher im EPROM liegenden Codes.

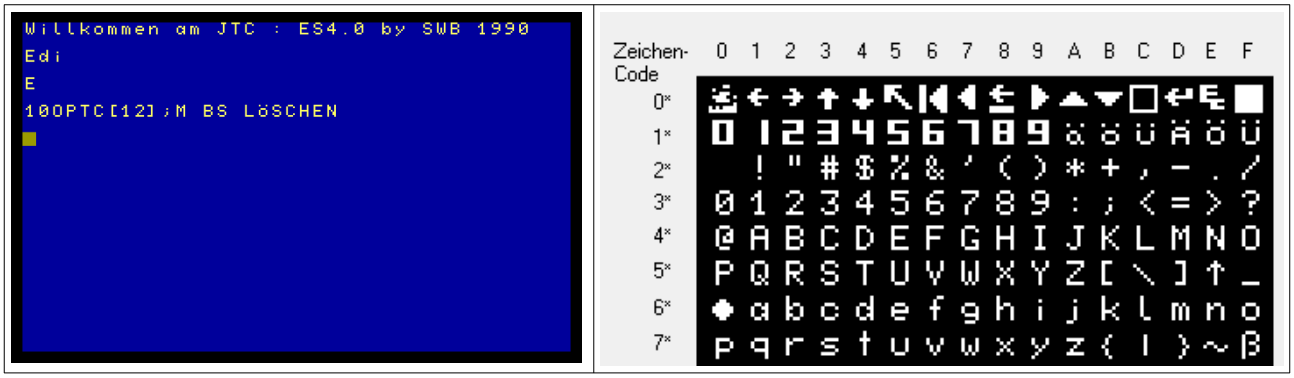
Um die Zeichen mit den Codes 00...0F darzustellen, ist vorher der Code %0E auszugeben.



Lage im Adressraum: %1008…%13FF (offset 0808h im ES4.0-EPROM)

Um einen anderen Zeichensatz auszuwählen und/oder zusätzlich Grafikzeichen (Code > %7F) darzustellen, muss die Zeichensatztabelle an einen anderen Ort verlagert und um weitere Bytes ergänzt werden. In Register %67 ist dem OS die veränderte Lage des Zeichensatzes (H-Byte der Anfangsadresse) mitzuteilen.

Bei Bedarf ließe sich auch der Grundzeichensatz direkt im ES4.0-ROM austauschen, was dann z.B. so aussehen könnte:



BASIC: EDI

EDI ist das Standardbedienprogramm zur Arbeit mit dem Basic des EMR. Folgende Kommandos stehen zur Verfügung (Direktmodus):

N	löscht das Basic-Programm *Wurde dies versehentlich ausgeführt, so lässt sich das Programm wieder restaurieren, sofern keine weiteren Aktionen startfanden:* *M Mon aufrufen* *,E000 80* *Q Mon beenden*
L	lädt ein Programm von Kassette, Ergebnis "0" für fehlerfrei oder "255" für Ladefehler.
S	speichert ein Programm auf Kassette, Ergebnis "0"
?X	schreibt den Wert der Variablen X aus
M	ruft den MON auf
E	listet das Programm zeilenweise auf, zur Fortsetzung mit Leertaste drücken E125 → beginnt mit dem Auflisten bei Zeile 125
R	startet das Basicprogramm
C	setzt das Programm nach STOP-Befehl fort.
10	oder eine andere ein- bis fünfstellige Zahl wird als Zeilennummer angesehen, danach unmittelbare Eingabe der Zeile ohne Leerzeichen. Die Eingabe einer Basiczeile wird mit ENTER (↵) abgeschlossen.
10↵	Zeilennummer gefolgt von ENTER löscht betreffende Zeile

- Ein direktes Kommando wird immer mit der ENTER-Taste abgeschlossen.
- Es werden intern 80 Zeichen als eine "logische Zeile" interpretiert.
- Durch die Ausführung der Kommandos L oder S werden die Basic-Variablen A, B, C verändert: nach LOAD: A → Anfangsadresse
  B → Länge des Programms
  C → Fehlermeldung

Variablen: A…Z (teilweise mit Spezialfunktionen), alles numerisch, keine Stringvariablen!

Zahlen: Es werden ganze Zahlen im Bereich von -32768 bis 32767 (16-Bit-Festkomma-Format) verarbeitet. Die Eingabe erfolgt dezimal mit ggf. Minuszeichen, Ziffernfolge und ENTER oder hexadezimal mit %, Ziffernfolge und ENTER.

Meldungen: O : Ende mit END
S : Ende mit STOP
E : Error:
E0: Syntaxfehler, CONT ohne STOP, Programmende ohne END bzw. STOP
E1: mehr als 15 GOSUBs verschachtelt
E2: RETURN ohne GOSUB
E4: Division durch 0
E8: Überschreitung des zulässigen Zahlenbereichs

Basicbefehle

Für die Basic-Befehle werden keine Schlüsselwörter mehr verwendet, sondern nur Buchstaben. Damit kommt man aber nach kurzer Gewöhnung gut zurecht. Hinter dem Buchstaben (also Basic-Befehl) darf ***kein Leerzeichen*** stehen!

Schlüssel-wort (alt)		Anmerkung	Beispiel
CALL	C	Aufruf Maschinenunterprogramm mit der Startadresse	C%81B
ELSE	>;	Alternativzweig nach Bedingung	>;G100
END	E	Programmende	
G0SUB	S	Aufruf Unterprogramm	S100
GOTO	G	Sprung zu Zeile Nr. Ausdruck	G100
IF...THEN…	F...;...	bedingte Ausführung	FX=0;G100
INPUT	I	Eingabe Zahl	P"X=";IX
LET	L	Zuweisung (Laden der Variablen mit dem Wert des Ausdrucks)	LA=1,B=2
PRINT	P	Anzeige von Zeichenkette/Dezimalwert nachgestelltes Komma verhindert die Zeilenschaltung	P"HALLO"
PROC	O	Ausführen einer Prozedur	
	PTC	Anzeige von Ausdruck gemäß ASCII (12=CLS)	OPTC[12]
	SETEB	Setzen eines Bytes im RAM	OSETEB[A,B]
	SETEW	Setzen eines Doppelbytes im RAM	OSETEW[A,B]
	SETR	Setzen Register mit Wert	OSETR[A,B]
	SETRR	Setzen eines Doppelregisters	OSETRR[A,B]
PTH	H	wie PRINT, aber mit hexadezimaler Ausgabe	
REM	M	Kommentar	MKommentar
RETURN	R	Rückkehr aus Unterprogramm	
STOP	T	Programmunterbrechung, Fortsetzung mit CONT	
TOFF	/	Löschen der Falle	
TRAP...TO …	!...,...	Falle, vor der Ausführung jeder folgenden Zeile wird Vergleich getestet. Falls erfüllt, erfolgt automatisch GOSUB zu Zeile Nr. Ausdruck und Löschen der Falle, sonst keine Wirkung.	
WAIT	W	Warten mit Ausdruck * 1 ms Dauer, Ausdruck muß einen positiven Wert haben.	W100

Standardfunktionen

ABS [x]		absoluter Betrag von x
NOT [x]		logische Negation von x
GTC		Zeicheneingabe vom Terminal
INPUT		Zahleneingabe vom Terminal
RL [x]		x links rotieren
RR [x]		x rechts rotieren
GETR [n]		Inhalt von Register (-nummer) holen
GETRR [nn]		Inhalt von Doppelregister (-nummer) holen
GETEB [adr]		Byte aus externem Datenspeicher holen
GETEW [adr]		Wort aus externem Datenspeicher holen

Verknüpfungsoperatoren: immer zwischen zwei Operanden:

+	plus		LA=1+5
-	minus		LA=B-1
*	mal		LA=B*C
/	geteilt durch		LA=B/C
$M	MODULO	(Divisionsrest)	LA=B$MC
$A	AND		LA=B$AC
$O	OR		LA=B$OC
$X	XOR		LA=B$XC

Vergleichsoperatoren:

=	gleich		
<	kleiner	<=	kleiner oder gleich
>	größer	>=	größer oder gleich
<>	ungleich		

INPUT-Besonderheit

Bei der Verwendung des Basic-Befehls INPUT (Buchstabe I) ist folgendes zu beachten: Die einzugebende Zahl kann mit allen Funktionen des FSE editiert werden; bevor die Eingabe mit RET abgeschlossen wird. Die Zahl muß aber auf der ersten Position der Zeile (ganz links) beginnen und nach der Zahl dürfen in derselben Zeile nur noch Leerzeichen stehen. Dafür kann man notfalls "von Hand" durch Wegfressen unerwünschter Zeichen mit DEL oder DBS sorgen.

Programmbeispiel:

`I"X-Wert="X`	falsch, da vor der Zahl etwas steht
`P"X-Wert:";IX`	richtig
`HX,;OPTC[6];IX`	richtig: Der Kursor steht auf dem Prozentzeichen der Hexzahl. Wird nur RET gedrückt, dann wird die Zahl gleich als neuer Eingabewert genommen. Das geht nicht so einfach für Dezimalzahlen, da diese mit einem Leerzeichen am Anfang gedruckt werden, wenn sie kein negatives Vorzeichen haben. Siehe auch 5.6.

Stringverarbeitung: keine!

Möglich ist nur eine PRINT-Ausgabe von Stringliteralen (z.B. PRINT "HALLO").

Zufallszahl ermitteln:

a) per Zeitgeber:

```
300   OSETRR[%F2,%23];OSETR[%F1,10]      ;Zähler/Zeitgeber setzen
310   LQ=GETR[%F2]$M8+1                  ;auslesen
```

besser/"zufälliger" geht es mit einer Systemfunktion:

b) per Systemfunktion ES4.0

%0836	RND	liefert nach jedem Aufruf in %74/75 eine 16-Bit-Zufallszahl

Anwendung in BASIC:

```
300 C%0836              ;Zahl generieren
310 LX=GETRR[%74]       ;holen: X=-32768...+32767
320 LX=ABS[X]           ;X=0...32768
330 LX=X$M8+1           ;X=1...8
```

kurz:

```
300   C%0836;LQ=ABS[GETRR[%74]]$M8+1     ;Q=1...8
```

Anwendung in Assembler:

```
RND:  EQU   %0836
      LD    R0,#5       ;oberer Grenzwert für Zufallszahl
      INC   R0
      CALL  RND         ;%74/75
Z1:   SUB   %74,R0      ;solange subtrahieren,
      JR    NC,Z1       ;bis Rest < Grenzwert+1
      ADD   %74,R0
```

oder:

```
      LD    R0,#5
Z2:   CALL  RND         ;16-Bit-Wert in %74/75
      CP    %74,R0
      JR    NC,Z2       ;solange wiederholen, bis Wert<=Ziel
```

Basic-Erweiterung für Gleitkomma-Rechnung: "FLOAT FÜR BASIC"

Der Zusatz wird auf %D000 geladen und belegt den Speicher bis %DFFF. Die Aktivierung erfolgt am Anfang eines Basic-Programms durch den Maschinenprogramm-Aufruf C%DAFA. Damit werden neue Befehle hinzugefügt (siehe Tabelle).

Eine **Gleitkommavariable** wird als **[num]** geschrieben[1]. „num" ist dabei die Nummer der Variablen und kann aus den Dezimalzahlen 0 bis 65535, den normalen Basic-Variablen A bis Z und den Operationszeichen + (Plus) und - (Minus) beliebig zusammengesetzt sein.

Die Gleitkommavariablen werden in einem separaten Speicherbereich abgelegt. Jede belegt 6 Byte. Bei Anfang %C000 (…%CFFF) wäre Platz für 682 Float-Variablen…

Kleine Buchstaben für die Kommandos!

Tastatur-Umschaltung auf Kleinbuchstaben: F2 bzw.
JF7E4 aus Mon

<table>
<tr><td>lvar=expr</td><td>"let"
berechnet den Ausdruck expr und übergibt den Wert an die Variable var.</td><td rowspan="3">var steht für eine normale Basic-Variable A bis Z oder eine Gleitkommavariable.

expr ist eine Zusammensetzung aus var, Gleitkommazahlen, den Operationszeichen + - * / und den definierten Funktionen.

Der Ausdruck expr wird von links nach rechts berechnet, ohne irgendwelche Prioritäten zu beachten. Eine andere Berechnungsreihenfolge kann nur durch das Setzen von runden Klammern erreicht werden.
Die Gleitkommazahlen werden in der üblichen Computernotation eingegeben. Das „E" als Exponentenkennzeichen muß groß geschrieben werden.
Beispiele:
234,
-98.07,
5.6E-12,
-34E45</td></tr>
<tr><td>?„text"expr</td><td>druckt den Text „text",
berechnet dann den Ausdruck expr und druckt Ergebnis aus.
„text" und expr können in beliebiger Anzahl und Reihenfolge auftreten.
expr wird ohne führende 0 gedruckt.
Steht nach dieser Folge ein Komma, dann wird nach dem Ausdrucken kein Return-Kode ausgegeben.</td></tr>
<tr><td>ivar</td><td>"input"
wartet auf eine Zahleneingabe und trägt die Zahl in die Variable var ein.</td></tr>
<tr><td colspan="3">Die Funktionen müssen ebenfalls aus kleinen Buchstaben bestehen. Das Argument folgt in runden Klammern. Es sind bereits folgende Funktionen definiert:</td></tr>
<tr><td>abs(expr)</td><td colspan="2">liefert den absoluten Betrag von expr</td></tr>
<tr><td>sgn(expr)</td><td colspan="2">ergibt null, wenn expr null ist und -1 bzw. 1, wenn expr negativ bzw. positiv ist.</td></tr>
<tr><td>int(expr)</td><td colspan="2">setzt bei expr alle Nachkommastellen auf null.</td></tr>
</table>

Folgende Fehler werden erkannt und gemeldet:
E0: unbekannter Befehl am Zeilenanfang oder nach einem Semikolon
E3: Syntaxfehler im expr (z.B. fehlende Klammern)
E12: Zahlenbereichsüberschreitung oder Division durch null

1 Achtung, Fehler in der Beschreibung in Ju+Te Heft 11/90 (dort /num/)

FLOAT-Funktionen:

CALL	Name	Wirkung	
%D000	FSUB	FP0=FP0-FP1	FP0: %70…%75 FP1: %76…%7B
%D003	FADD	FP0=FP0+FP1	
%D0EE	FASC	wandelt Zahl aus FP0 in einen ASCII-String um und legt diesen ab (%76/77) ab, als letztes Zeichen steht ein Byte %00;	
%D1F8	PASC	druckt die Zahl aus FP0 aus, benutzt dazu den Puffer %F750-F76F;	
%D234	ASCF	wandelt den ASCII-String ab (%7C/7D) in eine Zahl in FP0, (%7C/7D)	
%D424	FDIV	FP0=FP0/FP1, wenn FP1 null ist, wird ebenfalls das C-Flag gesetzt, zeigt danach auf das erste nicht umwandelbare Zeichen; Registersatz %4X wird auf Stack zwischengespeichert;	
%D540	FMUL	FP0=FP0*FP1, Registersätze %2X und %4X werden auf Stack zwischengespeichert	
%D578	HEXF	wandelt vorzeichenbehaftete Integerzahl aus %76/77 nach FP0;	
%D5E0	FHEX	wandelt die Zahl aus FP0 vorzeichenbehaftete Integerzahl nach %76/77.	

Maschinensprache: MON

MON ermöglicht die Nutzung und Bearbeitung von Maschinenprogrammen. Er verwendet nur Hexadezimalzahlen, die ohne %-Zeichen geschrieben werden. Dabei müssen A-F große Buchstaben sein. Befehle:

	Parameter	Wirkung
H	aaaa	Hexanzeige 8 aufeinanderfolgen Speicherstellen <spc>
,	aaaa bc bc bc bc bc bc bc bc	Hexschreiben des Speichers
A	aaaa	ASCII-Anzeige von 16 aufeinanderfolgenden Adressen <spc>
;	aaaa ddddddddddddddd	ASCII-Schreiben bis incl. RET-Zeichens, aber maximal 16
Q		Rückkehr zum aufrufenden Programm (meist EDI)
?	aaaa bbbb	Rechnen aaaa + bbbb und aaaa - bbbb
S	aaaa bbbb	speichert auf Kassette ab Adresse aaaa bbbb Bytes<mon>
L	aaaa	lädt von Kassette in Speicher ab Adresse aaaa <mon>
M	aaaa bbbb cccc	verschiebt von Adresse aaaa nach bbbb cccc Bytes <mon>
J	aaaa	startet Maschinenprogramm ab Adresse aaaa mit CALL <mon>
F	aaaa bbbb bc	füllt bbbb Bytes ab Adresse aaaa mit bc <mon>
#	eeee	Anzeige Dezimalzahl eeeee (0 bis 65635) als hex
%	aaaa	Anzeige Hexzahl als dezimal
R	bc	zeigt das Register bc an <spc>
!	bcxx	Register bc wird mit Byte xx beschrieben (ohne Leerzeichen!)

- Die in der Erklärung mit "spc" gekennzeichneten Befehle warten nach Ausgabe einer Zeile auf einen Tastendruck. Wird die Leertaste gedrückt, dann wird die nächste Zeile ausgegeben. Jede andere Taste beendet den Befehl und wird gleich auf den Bildschirm ausgegeben.
- Mit "mon" gekennzeichneten Befehle drucken nach ihrer Ausführung "Mon" aus.
- Wenn die eingegebenen Zahlen bei den Befehlen nicht korrekt sind, dann wird der Befehl nicht ausgeführt, es erfolgt aber keine Fehlermeldung.