USB (VDIP-1)

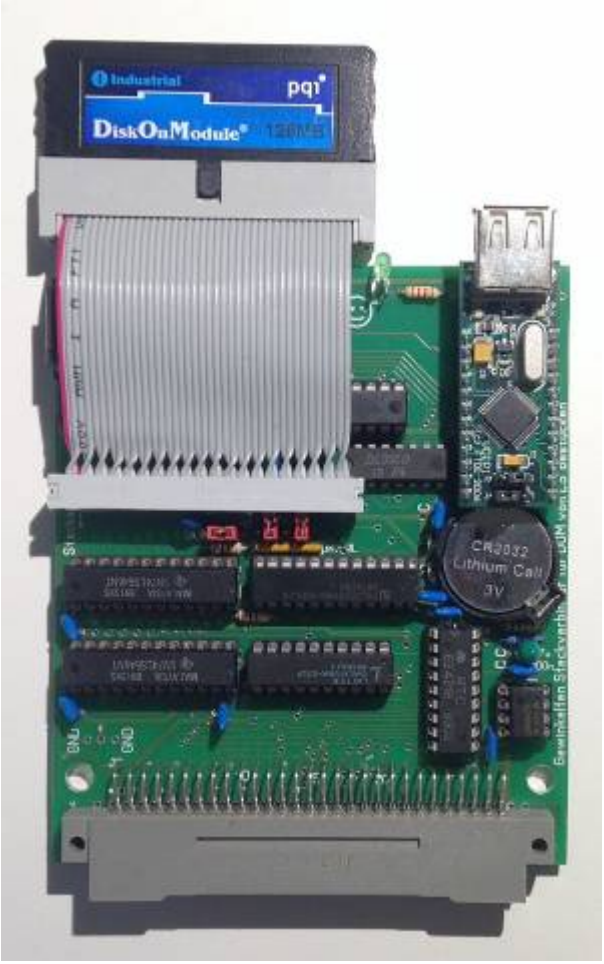
Das GIDE+USB+RTC-Modul von Wolfgang Harwardt erlaubt den **Anschluss von USB-Sticks** und von einer Festplatte, bevorzugt von einem DOM-Modul.

I/O-Adressen	USB	DCh..DFh (empfohlener Portbereich für Z9001 sowie für Z1013)

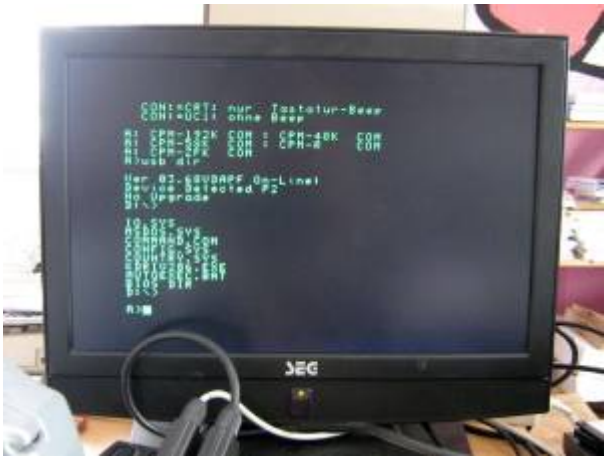
Basis dieser Entwicklung ist der USB-Anschluss von KC85 Labor susowa. Mario Leubner hat die Hardware und die Software USB Tools entwickelt.

Dank fertiger Module wie dem **VDIP1** von Viculum/FTDI DevelopmentModules.htm, DS_VDIP1.pdf ist der Anschluss recht einfach zu realisieren.

An das VDIP1-Modul wird ein USB-Stick angesteckt. Unterstützt werden USB 1.1 und USB 2.0-Sticks. Ein 8GB-Stick wurde erfolgreich getestet. Der Stick muss mit FAT12, FAT16 oder FAT32 formatiert sein.



Unter CP/M stehen die **UTools von Mario Leubner** zur Verfügung. Die UTools sind die Programme USB.COM, UPUT.COM, UGET.COM, UDIR.COM.



am KC mit Mega-Modul und BIC/KC-Floppymodul; Start von USB unter CP/M.

Die 4 Programme werden einfach auf eine Diskette kopiert und direkt unter CP/M von der Kommandozeile aus gestartet. Dabei kann auf den USB-Stick nur über dies Programme zugegriffen werden, der USB-Stick erhält **keinen Laufwerksbuchstaben**!. Programme und Dateien müssen immmer erst vom bzw. zum Stick kopiert werden, ehe sie genutzt werden können!

```
A>USB DIR
A>UDIR
```

zeigen den Inhalt des aktuellen Verzeichnisses des USB-Sticks an.

```
A>UGET SUDOKU.COM
A>SODOKU.COM
```

kopiert die Datei SUDOKU.COM vom USB-Stick auf Diskette. Anschließend wird das Programm von Diskette gestartet.

```
A>UPUT README.TXT
```

kopiert die Datei A:README.TXT auf den USB-Stick.

Es gibt noch mehr Kommando-Parameter, s.u.

Downloads

- utools14.zip UTools V1.4 Mario Leubner, incl. CP/M-SLR-Assembler-Umgebung zum Kompilieren unter Windows.
- utools15.zip UTools V1.5 Mario Leubner, incl. CP/M-SLR-Assembler-Umgebung zum Kompilieren unter Windows.
- Die Programme aus dem Paket zum USB-Modul für den Z1013 können ohne Änderungen auch am Z9001 genutzt werden!!

USB-Tools

Das VDIP1 meldet sich mit

```
Ver03.68VDAPF On-Line:
Device Detected P2
No Upgrade
D:\>
```

Ver03.68VDAPF ist die Firmwareversion (VDAP Disk And Peripheral Firmware Release 3.68)
P2 steht für USB-Port 2 [1)]
D: steht für Drive, gemeint ist der USB-Stick. Das ist **kein** CP/M-Laufwerksbuchstabe!

USB allgemeine Funktionen, Verzeichiswechsel

Kommando	**Bemerkung**
USB	prüft, ob USB-Stick angeschlossen ist
USB CD <verzeichnis>	Verzeichnis wechseln
USB CD /	ins Wurzelverzeichnis wechseln
USB CD ..	ein Verzeichnis zurück
USB DIR	Verzeichnisanzeige
USB DLD <verzeichnis>	Delete Dir, Verzeichnis löschen
USB MKD <verzeichnis>	Make Dir, Verzeichnis anlegen
USB DLF <dateiname>	Delete File, Datei löschen
USB FS	Free Space, Freien Platz anzeigen
USB IDD	Disk-Informationen anzeigen
USB FWV	Firmware-Version anzeigen
USB RD <file>	Read, Textdokument anzeigen
USB REN <alt> <neu>	Rename, Datei umbenennen
USB //	ruft die Hilfe auf !

UDIR Verzeichnisanzeige

UDIR <maske> opt	gefilterte, maskierte Verzeichnisanzeige
UDIR //	Hilfe
UDIR V	vorherige Meldung anzeigen
UDIR W	ausführliche Infoanzeige

UGET Dateien von USB auf CP/M-Laufwerk kopieren

UGET <dir:maske> opt	Datei(en) von USB auf CP/M-Laufwerk kopieren
UGET *.*	holt alle Datein vom vorher auf dem Stick eingestellten Verzeichnis
UGET //	Hilfe
UGET V	vorherige Meldung anzeigen
UGET I	vorhandene Datei ignorieren
UGET O	vorhandene Datei ersetzen
UGET U	vorhandene Datei aktualisieren
UGET M	Fortschrittanzeige wie MTOOLS

UGET <name> /opt	Datei laden
UGET <dir:name> /opt	Datei in angegebenes Verzeichnis laden

UPUT Dateien von CP/M-Laufwerk auf USB schreiben

UPUT <dir:maske> opt	Datei(en) von CP/M-Laufwerk auf USB schreiben
UPUT //	Hilfe
UPUT V	vorherige Meldung anzeigen
UPUT M	Fortschrittanzeige wie MTOOLS
UPUT I	vorhandene Datei ignorieren
UPUT U	vorhandene Datei aktualisieren
UPUT O	vorhandene Datei ersetzen
UPUT T	Textdatei, Abbruch bei EOF (1Ah=^Z)

VDIP1

USB-Sticks

Der VDIP1 unterstützt USB 1.1 und USB 2.0-Sticks. Ein 8GB-Stick wurde erfolgreich getestet. Der Stick muss mit FAT12, FAT16 oder FAT32 formatiert sein.

Achtung: Lange Dateinamen werden nicht unterstützt! Am günstigsten ist es, wenn man nur mit kurzen 8.3-Dateinamen arbeitet.

Flashen einer neuen Firmware

Auf VDIP1 muss die passende Firmware aufgespielt sein (VDAP Version 3.68 oder neuer).

Aktuell ist Version 3.69; die Version 3.68 reicht aber auch. Unter http://www.ftdichip.com/Firmware/Precompiled.htm, Latest Vinculum (VNC1L) Firmware Releases, findet man ggf. eine neue Version. Es wird die **VDAP** Disk And Peripheral Firmware benötigt. Die Reflash (FTD)-Datei wird als FTRFB.FTD ins Root-Verzeichnis des USB-Sticks abgelegt. Beim Starten des Rechners bzw. auch beim Start von USB.COM installiert das VDIP1 automatisch seine neue Software.

Jumper

auf dem VDIP1 muss JP3 1-2 und J4 3-2 gesteckt sein (Parallel FIFO)

LEDs

Die beiden LEDs auf dem VDIP1 signalisieren den aktuellen Zustand:

LED1 (links)	LED2 (rechts)	Bedeutung
blinkt	blinkt	2 Sek. abwechselndes Blinken. Power On
an	aus	USB Stick init.
aus	an	USB Stick ready
aus	aus	kein USB Stick gesteckt
aus	blinkt	Ausführen eines Kommandos

Hardware

Das VDIP-Modul wir an einer PIO angeschlossen. Das erlaubt die Nutzung des parallelen Datenmodus.

Mario Leubner schreibt dazu (http://susowa.homeftp.net/index.php/projekte-mainmenu/usb-mainmenu-131/72-usb-stick-am-kc.html):

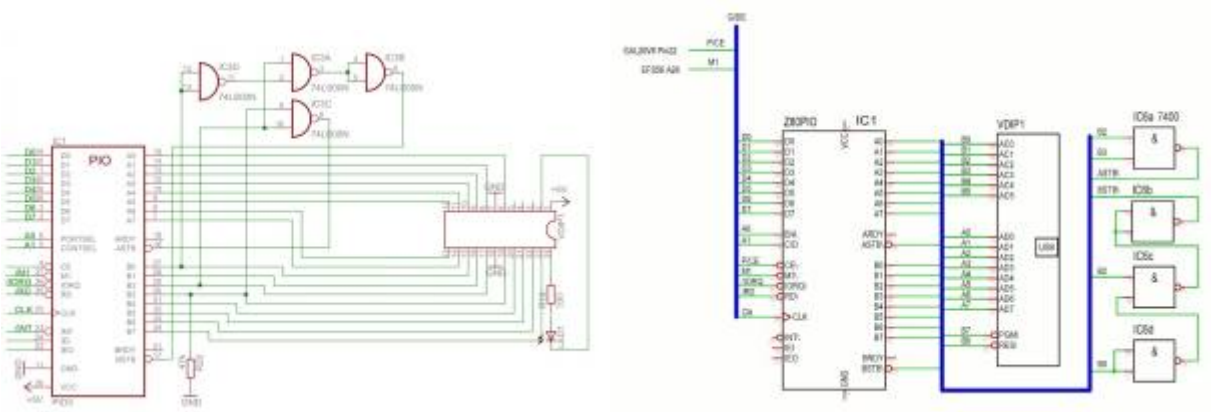
Zur Nutzung der Parallelschnittstelle war zunächst klar, dass der Anschluss am M001 erfolgen wird. Hier stehen zwei PIO-Ports zur Verfügung und Kanal A kann auch bidirektional arbeiten – also Daten empfangen und senden. Kanal B muss dazu im Bitbetrieb arbeiten und kann so für die Bedienung der Statussignale herangezogen werden.

Zunächst galt es die Signalspiele beim Lesen und Schreiben von Daten zu analysieren, also das Handshake zu begreifen. Der Vinculum-Chip zeigt an zwei Pins seinen Status an, zwei weitere Pins dienen als Steuersignale:

RXF# geht auf Low sobald Daten abgeholt werden können
TXE# zeigt mit Low an, dass Daten geschrieben werden können
RD# ein Low-Impuls liest ein Datenbyte
WR ein High-Impuls schreibt Datenbyte ein

Dieses Signalspiel eignet sich leider nicht, um die 4 Signale direkt mit den Strobe- und Ready-Leitungen einer Z80-PIO zu verbinden und diese damit direkt bidirektional zu betreiben. So musste ich die Handshake-Signale an PIO-Port B legen und programmiertechnisch abfragen. Um den bidirektionalen Kanal A von Eingabe auf Ausgabe umzusteuern, ist es erforderlich, die PIO-Anschlüsse A-Strobe und B-Strobe zu beschalten. Dies übernimmt ein zusätzlicher DL000 (74LS00), der von den Leitungen RD# und WR mitgesteuert wird und so automatisch die passende Datenrichtung freigibt.

Dazu ist es ausreichend ASTB zu beschalten - dann wird bereits das PIO- Ein/Ausgaberegister umgeschalten. So könnte BSTB zur Interrupt-Auslösung genutzt werden wenn Daten verfügbar sind. Eine direkte Anschaltung von RXF# auf BSTB funktioniert aber nicht, auch nicht mit Negation. Denn BSTB muss zwingend Low sein, damit die Daten bei der Eingabe auch im Eingaberegister übernommen werden. BSTB kann jedoch auch Interrupts auslösen, und zwar mit der L/H-Flanke (!) mit dem Hintergrund dass die Eingabedaten damit im Eingaberegister verfügbar sind. So habe ich die ursprüngliche Schaltung dahingehend modifiziert, dass sowohl bei der Eingabe BSTB Low ist, als auch das Signal „Daten verfügbar“ einen L/H-Wechsel verursacht wenn neue Daten verfügbar sind, RXF# also auf Low geht (und nicht gerade eine Eingabe läuft).



li: Schaltung von Mario Leubner, rechts: DL. Hier sieht man die Logik besser!

[1)]

Die VDIP-Firmware kann 2 USB-Ports treiben. Das VDIP1-Modul hat nur einen USB-Anschluss (Port2), es ist möglich, einen weiteren zu ergänzen, s. VDIP1 Datasheet