Kopplung des Rechners KC87 an den Parallelport eines PC ueber das PIO-Interface

---

1.) Das Anliegen der Kopplung

Die Rechner der KC-Typen sind bei Sammlern beliebt und begehrt.
Hier ist die Kopplung eines KC87 mit einem PC ueber das verfuegbare parallele

Z80-PIO-Interface

beschrieben.
Wer keine zugehoerige Peripherie der damaligen Kategorie besitzt oder wenig damit arbeiten möchte/kann, kann mit einem Kabel einen PC anschliessen und damit einen Fileserver usw. damit realisieren.
Mit einem KC87 kann man vom/zum PC Programme und Daten uebertragen und Files anlegen/uebertragen.

(zurueck)

2.) PIO-Interface am KC87

Das Manual dazu ist ueber das Internet per Download, etwa ueber
http://www.sax.de/~zander/index2h.html
zu bekommen.

Pinbelegung des Steckverbinders lt. Bedienungsanleitung...,Anhang 2 :
(- bedeutet: Signal hier nicht benutzt.)

Kontakt	Reihe A	Reihe B	Reihe C
1	00 , Masse	00 , Masse	-
2	B0	B1	B2
3	B3	B4	B5
4	B6	B7	BRDY
5	5P , +5V	BSTB	-

Betriebsmodi , Programmierung
Mit den verfuegbaren Signalen sind Byteeingabe, Byteausgabe und Bit-Operationen möglich.

Die Auswahl des Modus fuer die Kopplung ist durch folgende Praemissen gepraegt:
a.) Die Bedienschritte nach dem Einschalten des KC87 sollen minimal sein, im einfachsten Fall steht kein Magnetbandgeraet zur Verfuegung.
b.) Die Befehlsabfolge zur Kommunikation soll minimal sein.
c.) Keine Bedienerhandlungen am PC
d.) Die Kommunikation muss kollisionsfrei erfolgen.

Zeitaufwand spielt hier bei der Kommunikation bewusst keine Rolle.

Variante 1 (hier im folgenden beschrieben) :
Es werden fuer die Kopplung sowohl der Betriebsmodus Byteausgabe als auch der Betriebsmodus Byteeingabe benutzt.
Vorteil: Es lassen sich a.) und b.) einigermassen einrichten.
Folgende Nachteile des KC-Userinterface sind hier zu beachten und betreffen c.) und d.) :
- Es steht kein Reset-Signal fuer den PC zur Verfuegung.
- Es steht kein Signal fuer die Daten-Richtung fuer den PC zur Verfuegung

Variante 2 (wird extra spaeter getestet) :
Fuer die Kopplung wird der Bit-Modus benutzt.
Vorteil: Es laesst sich ein vollkommener Handshake-Betrieb mit Richtungs-Signal und Reset-Signal realisieren, betrifft c.) und d.)
Nachteil: Serialisierung der Daten, damit hoher KC-Programmaufwand bzw. viele Anfangs-Bedienschritte, betrifft a.) und b.)

Es wird im weiteren nur die Variante 1 ausgefuehrt.

Im einfachsten Fall wird die PIO für unsere Zwecke mit BASIC-Befehlen angesprochen. Diese sind:
Bedeutung:
x Ausdruck mit dem Wert 0...255

	Byte-Ausgabe	Byte-Eingabe
Betriebsmodus einstellen	OUT 139, 15	OUT 139, 79
laufende Operation	OUT 137, x	INP(137)

(zurueck)

3.) Parallelport des PC

Am PC wird der Parallelport, an dem sonst ein Drucker angeschlossen ist, im "**bidirektionalen**" Betrieb benutzt.
Dazu muss im PC-Setup-Programm beim Menuepunkt des Parallelinterface der Betriebsmodus "**EPP**" eingestellt sein.

Pinbelegung des PC-Steckverbinders
Es werden im folgenden die Pins der 25-poligen Buchse des Parallelports und des 25-poligen Steckers am Kabel mit
P1 , P2 , P3 ..... bis P25
bezeichnet.
Die sonst ueblicherweise angegeben Namen fuer den Druckeranschluss werden nicht verwendet, weil diese fuer diesen Fall keine Bedeutungen haben.
(zurueck)

Programmierung des Parallelports
Die Programmierung des PC-Parallelports ist in vielen Fallen im Internet und Buechern beschrieben, leider manchmal etwas diffus und ungenau, weshalb hier eine Zusammenfassung fuer unseren Fall vorgenommen wird.
Es wird der Fall des 1. Parallelports im PC angenommen.
Die Programmierung geschieht ueber folgende Ports/IO-Adressen:

378H	Datenbyte	bidirektional Eingabe und Ausgabe
379H	Status-Bits	Eingang in den PC
37AH	Steuer-(/Control-) Bits	Ausgang aus dem PC

Die Zuordnung der Bits dieser Register zu den Pins am Steckverbinder ist für diese Kopplung wie folgt festgelegt, wobei bedeutet:

Pnn eine 1 entspricht High-Pegel ,
eine 0 entspricht Low -Pegel am Steckverbinder.

/Pnn eine 1 entspricht Low -Pegel ,
eine 0 entspricht High-Pegel am Steckverbinder, d.h. hier ist ein Negator auf dem Mainboard/der Interfacekarte dazwischengeschaltet.

x Das Bit darf nicht veraendert werden .

y Das Bit wird/kann programmiert (werden), Es gibt dazu keinen Kabelanschluss.

\- Wird hier nicht benutzt.

Die hier angegebenen Signalnamen sind der Kopplung PC-KC87 entsprechend festgelegt und entsprechen den Signalnamen des PIO-Interfaces.

Port 378H:

Bit	7	6	5	4	3	2	1	0
Pin-Zuordnung	P9	P8	P7	P6	P5	P4	P3	P2
Bedeutung	B7	B6	B5	B4	B3	B2	B1	B0

Bedeutung:
Bit0 bis Bit7 = B0 bis B7 sind die Datenbits des PIO-Interface.

Port 379H:

Bit	7	6	5	4	3	2	1	0
Pin-Zuordnung	/P11	P10	P12	P13	P15	x	x	x
Bedeutung	-	BRDY	-	-	-	x	x	x

Bedeutung:

Bit6 teilt vom KC87 folgendes mit:
Bei Ausgabe aus dem KC87:
Bit6 = 1 entspricht BRDY = high , die Datenbits sind gueltig
Bit6 = 0 entspricht BRDY = low, Grundzustand
Bei Eingabe in den KC87:
Bit6 = 1 entspricht BRDY = high , d.h. der PC soll die Datenbits belegen
Bit6 = 0 entspricht BRDY = low, Grundzustand

Port 37AH:

Bit	7	6	5	4	3	2	1	0
Pin-Zuordnung	x	x	y	y	/P17	P16	/P14	/P1
Bedeutung	-	-	BIDI	-	-	-	/BSTB	-

Bedeutung:
Bit5 = BIDI legt PC-seitig die Richtung des Datentransportes fuer das Register 378H fest. Es gilt:
Bit5 = BIDI = 0 Ausgabe aus dem PC zum KC87 , es kann auf 378H geschrieben werden
Bit5 = BIDI = 1 Eingabe in den PC vom KC87 , es kann von 378H gelesen werden

Bit1 teilt dem KC87 folgendes mit:
Bei Eingabe in den PC
Bit1 = 0 liefert dem KC87 BSTB = high , Grundzustand bei Eingabe in den PC
Bit1 = 1 liefert dem KC87 BSTB = low , die Daten wurden gerade uebernommen
Bei Ausgabe aus dem PC
Bit1 = 0 liefert dem KC87 BSTB = high , Grundzustand bei Ausgabe aus dem PC
Bit1 = 1 liefert dem KC87 BSTB = low , die Datenbits sind gueltig

(zurueck)

4.) Kabelverdrahtung am Parallelport-Stecker
Entspechend obigen Festlegungen ist das Kabel wie folgt zu beschalten:

PIN am PC-Stecker	Farbe der Leitung (bei meinem Kabel !)	PIO-Signalname	Pin am KC87-Stecker (haengt davon ab, ob der originale Steckverbinder genommen werden kann oder nicht!!!)
2		B0	
3		B1	
4		B2	
5		B3	
6		B4	
7		B5	
8		B6	
9		B7	
10		BRDY	
14		BSTB	
18 bis 24 und 25 (alle Pins 18 bis 24 auch ueberbrueckt)		GND	
(s.Anmerkung a)		+5V	

Anmerkung a:
Diese +5V koennten an einen der freien Eingaenge des Parallelports so geschaltet werden, dass der PC testen kann, ob der KC87 eingeschaltet ist.
Hinzukommt, dass dem PC kein RESET-Signal des KC87 zur Verfuegung steht. Es bestuende zumindest mit Nutzung der +5V die Möglichkeit, durch Aus- und Einschalten des KC87 das PC-Serverprogramm, das i.A. ohne PC-Bedienerhandlungen arbeiten soll, zu synchronisieren.

(zurueck)

4.) Inbetriebnahme, Tests der Kopplung

Es ist generell darauf zu achten, das die Eingabe- /Ausgabe- Operationen beider Partner genau aufeinander abgestimmt erfolgen müssen, nie sollten beide eine Ausgabe gleichzeitig betreiben. (Bem.: Bevor man das Verbindungskabel herstellt, kann man mit 4 Leitungen wenigstens ein Daten-Bit zum Test verwenden.)

Die Inbetriebnahme der Kopplung und der Funktionstest geschieht mit moeglichst einfachen Mitteln, ich habe MS-DOS und den Low-Level-Debugger **DEBUG.EXE**
verwendet.
Aufgrund der **Handshake-Methode** kann man '**statisch**' arbeiten und sich bei der Eingabe der Kommandos sehr viel Zeit lassen.
Man starte am PC das Programm
DEBUG
Eine Uebersicht der Kommandos erhaelt man durch Eingabe des Zeichens ? (Fragezeichen). Es werden nur die Kommandos 'I' und 'O' benoetigt.

Im Internet ist mehrfach das Manual zur Z80-PIO-Programmierung zu bekommen.
Fuer die Durchfuehrung des Handshakeverfahrens sollte man die Diagramme im Manual verwenden.
Am KC87 ist der Datenport über die Adresse 137 und der Steuerport über die Adresse 139 zu erreichen.

A.) Test der Uebertragung vom KC87 zum PC.
Am PC kontrolliert man das Bit BIDI , ob es auf 'Eingabe' gestellt ist:
I 37A (Bem.: Achtung , DEBUG erwartet Hexzahlen und ohne das anhaengende 'H')
In der ausgegebenen Zahl muss das Bit5 gesetzt sein. Sollte dies nicht der Fall sein, muss die ausgegebene Zahl um das Bit5 ergaenzt (xxx) wieder ausgegeben werden (danach kontrollieren!):
O 37A xxx

Nun wird dem KC87 durch den PC ueber Port 37AH das Signal BSTB=high, entspricht Bit1=0 als Grundzustand mitgeteilt.

Am KC87 muss zuerst Port B fuer die Ausgabe zugelassen werden, weil nach dem Einschalten dieser auf Eingabe eingestellt ist.
Man gibt durch direkte Tastatureingabe am KC87 das folgende Kommando ein:
OUT 139, 15

Man pruefe am PC, ob der Grundzustand von BRDY=low , also Bit6=0 ist durch das Kommando
I 379

Jetzt, wenn dies stimmt, kann man am KC87 ein Zeichen z.B. 'A' ausgeben, durch das Kommando:
OUT 137,65

Nun prueft man am PC, ob das Signal BRDY=high ist, das ist in Port 379H, Bit6=1 :
Man gibt am PC das folgende Kommando ein:
I 379
Wenn in der ausgegebenen Zahl zu sehen ist, dass Bit6=1 ist, kann man das Datenbyte am PC einlesen:
I 378
Die angezeigte Zahl muss:
41
lauten, der Hex-Code fuer das Zeichen 'A' .

Die Uebernahme des Zeichens bestaetigt man am PC durch BSTB=low, das ist Port 37AH, Bit1=1.

Nun prueft man (ggf. wiederholt, kommt aber praktisch im Handbetrieb nicht vor) am PC, ob der KC87 das Signal BRDY=low gesetzt hat, das ist in Port 379H, Bit6=0
Ist dies der Fall, so setzt man am PC das Signal BSTB=high, das ist Port 37AH, Bit1=0.
Damit ist der volle Handshake- Zyklus abgeschlossen.

B.) Test der Uebertragung vom PC zum KC87

Zuerst ist der KC-PIO-Port auf Eingabe zu stellen:
OUT 139, 79

Am PC kontrolliert man das Bit BIDI , ob es auf 'Ausgabe' gestellt und gleichzeitig das Signal BSTB=high eingestellt ist:
I 37A
In der ausgegebenen Zahl muss das Bit5=0 und das Bit1=0 sein.
Sollte dies nicht der Fall sein, muss die ausgegebene Zahl ggf. um das Bit5=0 und Bit1=0 geaendert (xxx) wieder ausgegeben werden (danach kontrollieren!):
O 37A xxx

Nun wird der KC87 auf die Eingabe eines Zeichens durch folgendes Kommando eingestellt:
X=INP(137)
Am PC muss nun geprueft werden, ob das Signal BRDY=high gestellt wurde, das ist Port 379H, Bit6=1.
Ist dies der Fall, kann man am PC eine Zahl, z.B. 42 (hexadezimaler Code fuer das Zeichen 'B') ausgeben:
O 378 42
Die Gueltigkeit des Datenbyte wird dem KC87 durch BSTB=low, Port 37A, Bit1=1 angezeigt.
Anschliessend muss vom PC wieder das Signal BSTB=high also Port 37A, Bit1=0 gesetzt werden (Zeitdifferenz im Programm noch feststellen!).

Der KC87 hat das Byte eingelesen, wenn BRDY=low, Port 379H, Bit6=0 gesetzt wurde.
Damit ist der volle Handshake- Zyklus abgeschlossen.
Nun kann man am KC87 sich anzeigen lassen, was eingelesen wurde:
PRINT X
Auf dem Bildschirm muss die Zahl
66 (Code fuer B)
erscheinen.

(zurueck)

5.) PC- Grundprogramme zur Kommunikation mit dem KC87

Die Realisierung der Kommunikation wird durch Programmierung auf der Basis des Betriebssystems Linux durchgefuehrt.
Fuer MSDOS kann man die gleichen Programmquellen vereinfacht verwenden. Eine Windows-Programmierung ist fuer mich nicht relevant.
Es wurden dabei Grundprogramme erstellt, die z.B. bei Skript-Programmierung zu komplexen Ablauefen verwendet werden koennen.

Weil Eingabe-/Ausgabe-Ports programmiert werden, laufen die Programme nur mit root-Rechten.

Diese Grundprogramme sind:

Programmname:	Wirkung:
kcpreset	Einstellung des PC auf Eingabe.
kcpstatus	Feststellung des Statussignals BRDY, Ausgabe des Statuswertes auf stdout
kcbyin	Eingabe eines Zeichens (ein Byte) in den PC, Ausgabe des Zeichens auf stdout
kcbyout	Abfrage des Zeichens von stdin, Ausgabe dieses Zeichens (ein Byte) vom PC
kcstrin	Eingabe einer Zeichenkette (String) in den PC, Ausgabe des Strings auf stdout
kcstrout	Abfrage eines Strings von stdin, Ausgabe dieser Zeichenkette aus dem PC, LF (Code 0AH)als Abschlusszeichen (ist nicht Bestandteil des Strings).
kcfout (filename)	Ausgabe eines codierten BASIC-Files vom PC zum KC87
kcfin (filename)	Eingabe eines codierten BASIC-Files vom KC87 in den PC

Als Returnwert liefern alle Programme den Statuswert nach der Operation .

Die Ausfuehrung und Beschreibung ist den ANSI-C Quellfiles:

kcpreset.c
kcpstatus.c
kcbyin.c
kcbyout.c
kcstrin.c
kcstrout.c
kcfout.c
kcfin.c
pio-par-lib.c

zu entnehmen.
(zurueck)

6.) Festlegung der Kommunikation zwischen KC87 und PC als Server

Der PC soll als Server dienen und **nur auf Anforderung** durch den KC87 taetig werden.
Deshalb muss die Kommunikation des PC im Grundzustand 'Eingabe-Modus' starten und danach kann der KC87 Anforderungen stellen.
Der PC sollte immer so schnell wie möglich ein den 'Eingabe-Modus' übergehen.

Als Basisverzeichnis wird hier
/root/pc-kc87
verwendet, was aber für andere Fälle nicht zwingend ist.

Alle anderen Pfadangaben erfolgen dazu relativ.

Es gibt fuer die Kommunikation folgende Kommandostrings und deren Reihenfolge bei der Kommunikation:

A.) Anfordern des Inhaltsverzeichnissen fuer ein Directory, rekursiv :
KC87 sendet:
"C(rel.Pfad)"

PC sendet:
"(mit 'ls' erzeugte Zeilen)"
"..." als Ende

Es werden nur Strings, abgeschlossen durch LF (Code 0AH) empfangen bzw. gesendet.
Sollte das Zeichen CR (Code 0DH) auftreten wird es vom Empfaenger ignoriert.
Als Endestring fungiert hier der String "..." (3 Punkte)
Sollte das angegebene Verzeichnis relativ zum Basis-Verzeichnis im PC nicht vorhanden sein, so wird es neu erstellt.
Bem: Wenn als relativer Pfad ein Leerstring angegeben wird, so wird das Basisverzeichnis verwendet.

B.) Einstellen des File-Verzeichnisses im PC relativ zum Basisverzeichnis fuer nachfolgende File-Operationen 'G'et und 'P'ut :
KC87 sendet:
"D(rel.Pfad)"
Hier ist kein Endestring vorhanden

Der PC sendet nichts.
Sollte das angegebene Verzeichnis relativ zum Basis-Verzeichnis nicht vorhanden sein, so wird es neu erstellt.

C. Ausgabe eines codierten BASIC-Files vom PC auf den KC87
KC87 sendet:
"G(filename)"

PC sendet:
(Inhalt des codierten BASIC-Files)

Bem.: Unter filename ist die Angabe einschliesslich Endung.

Das codierte BASIC-File ist ein Binaerfile und enthaelt neben der Byteanzahl den Speicherabzug des BASIC-Programmbereiches im KC87 ab
Speicheradresse 400H . Dieser Bereich endet, wenn 3-mal das Null-Byte auftritt.
Das File hat folgenden Aufbau (LP=Low Part und HP=High Part):
1. Byte: LP(Anzahl der Bytes des Speicherabzuges)
2. Byte: HP(Anzahl der Bytes des Speicherabzuges)
... alle Bytes des Speicherabzuges ...

[Spaeter erweitert:
Es wird, wenn kein codiertes File existiert, nach einem gleichnamigen File mit der zusaetzlichen Endung '.bas' gesucht, welches die normale BASIC-Quellcode-Darstellung enthaelt. Existiert dieses, so wird es in ein codiertes File gewandelt und gesendet.
]

D. Eingabe eines codierten BASIC-Files vom KC87 in den PC
KC87 sendet:
"P(filename)"
danach sendet der KC87:
(Inhalt des codierten BASIC-Files)

[Spaeter erweitert:
Es wird nach Abschluss der File-Uebertragung ein File mit der zusaetzlichen Endung '**.bas**' mit dem normalen BASIC-Quellcode im PC erzeugt .
]

E.) Zeilen eines Text-Datenfiles als Strings vom PC auf den KC87 uebertragen:
KC87 sendet:
"A(filename)"
PC sendet:
"(Zeilen des Files)"
"..." als Ende

F.) Strings mit Text/Daten vom KC87 auf den PC uebertragen und in einem File ablegen:
KC87 sendet:
"S(filename)"
"(Strings)"
"..." als ende

Das auf dem PC verwendete Server-Programm ist ein bash-Skript, als zyklisch arbeitendes Programm ohne Bedienerhandlung mit dem Namen

pc-kc87-fileserver

Das Programm laueft wegen der Portzugriffe nur mit einem root-Recht und am besten staendig im Hintergrund.
Dazu gehoert das File **pc-kc87.src**

Ein im Arbeitsverzeichnis '/root/pc-kc87' existierendes File mit dem Namen
fende.txt
mit beliebigen Inhalt beendet das Server-Programm regulaer.

(zurueck)

7.) Die Kommandos/Befehle des KC87 zur Kommunikation mit dem PC als Server

Es gibt z.B. folgende Moeglichkeiten die Kommunikation ablaufen zu lassen:

a) Man gibt alle Befehle in der Kommandophase des KC87 ein.
b) Man integriert den Teil des Programmes, der die Kommunikation regelt, als Bestandteil des Anwenderprogrammes . Dies ist sinnvoll fuer Daten-Eingabe und Daten- Ausgabe. Am guenstigsten arbeitet man mit Strings. Es ist auch moeglich, die Ausgabe des Programmes selbst als codiertes BASIC-File im Anwenderprogramm zu integrieren. Programmierte Programm- Eingabe/-Verkettung geht nicht bzw. waere eine Untersuchung wert, weil PEEK- und POKE-Operationen Überaschungen bescheren koennen..

Zu a.) :

Minimales Beispiel zum Einlesen eines codierten Basic-Programmes vom PC:

```
S$="(filename)":OUT 139,15:OUT 137,71
FOR I=1 TO LEN(S$):OUT 137,ASC(MID$(S$,I,1)):NEXT I:OUT 137,10
OUT 139,79:L=INP(137):H=INP(137):N=256*H+L
FOR I=1 TO N:A=INP(137):POKE 1023+I,A:NEXT I
```

Minimales Beispiel zum Ausgeben eines codierten Basic-Programmes vom KC87:

```
S$="(filename)":OUT 139,15:OUT 137,80
FOR I=1 TO LEN(S$):OUT 137,ASC(MID$(S$,I,1)):NEXT I:OUT 137,10:A=1025
FOR I=0 TO 1:N=256*PEEK(A+1)+PEEK(A):IF N<>0 THEN A=N ELSE I=1:NEXT I
N=A-1018:H=INT(N/256):L=N-256*H:OUT 137,L:OUT 137,H
FOR I=0 TO N:B=PEEK(1024+I):OUT 137,B:NEXT I
```

Zu b.)
Ein Rahmen-Beispiel fuer das Einlesen- /Ausgeben und Verarbeiten von Daten, die auf dem PC als Datenfiles gespeichert werden ist:

kc87pc.bas

(zurueck)

8.) Erweiterungsmoeglichkeiten

Weitere Kommandos für die Kommunikation kann man einfuehren, wenn man im PC-Skript die Stelle weiter ausfuellt, an der

```
#Template fuer Erweiterungen
```

beginnt.
(zurueck)