



PC-Parallel-IO-Interface Teil 3

Der dritte und abschließende Teil beschreibt den Datenaustausch mit dem PC und gibt eine komplette Übersicht über den Software-Befehlssatz für die Arbeit mit dem Interface. Ein kleines Windows-Programm macht den Einstieg in die Programmierung besonders einfach.

Datenaustausch

Der Austausch der Daten zwischen Interface und PC erfolgt über eine Standard-RS232-Schnittstelle (COM1 bis COM4), wobei folgende Parameter fest vorgegeben sind: 38400 Baud, 8 Bits, odd Parity, 2 Stopbits.

Für die weitere Beschreibung gelten die folgenden Konventionen:

- Alle Zahlen mit nachgestelltem „h“ sind Hexadezimal-Werte

- Alle Angaben in eckigen Klammern sind Zeichenfolgen
- Alle Bezeichnungen in spitzen Klammern sind einzelne ASCII-Zeichen, wobei folgende Zeichen benutzt werden :
 <SOH> = 01h
 <EOT> = 04h
 <ACK> = 06h
 <DLE> = 10h
 <NAK> = 15h

Jede Datenübertragung hat einen festen Rahmen und wird durch das ASCII-Zeichen <SOH> eingeleitet.

Anschließend folgen die zu übertragenden Daten, danach ein Längsparity-Byte.

Das ASCII-Zeichen <EOT> schließt den Datensatz ab.

Zur Bildung des Längsparity-Bytes werden alle Datenbytes der Nachricht sowie des Startbytes <SOH> XOR-verknüpft.

Da in der Nachricht und im Längsparity-Byte die SteuerCodes <SOH> und <EOT> nicht vorkommen dürfen, müssen vor dem Senden der Daten alle Bytes zwischen

Tabelle 1: Fehlercodes vom Interface

“1” (31h): Parity Error beim Datenempfang
“2” (32h): Überlauf des Empfangspuffers
“3” (33h): Fehler beim Längsparity
“4” (34h): unbekannter Befehl
“5” (35h): falscher Parameter
“6” (36h): Datenbereich überschritten
“7” (37h): Eingang wurde noch nicht getriggert.

<SOH> und <EOT> geprüft und eventuell durch eine besondere Zeichenfolge ersetzt werden. (Die Längsparity-Bildung muß vorher erfolgt sein).

Folgende Zeichen werden durch die angegebene Zeichenfolge ersetzt:

<SOH>	→	<DLE> (81h)
<EOT>	→	<DLE> (84h)
<DLE>	→	<DLE> (90h)

Beim Empfang von Daten sind diese Zeichenfolgen rechtzeitig zu erkennen und durch die zugehörigen Einzelzeichen zu ersetzen. Erst danach kann eine Überprüfung des Längsparity erfolgen.

Soll zum Beispiel auf Kanal A die Bitkombination 0000100 ausgegeben werden, so muß die Nachricht: (41h), (04h) zum Interface gesendet werden. Der Datensatz sieht somit folgendermaßen aus:

(01h) (41h) (04h) (44h) (04h)

Da in der Nachricht das Steuerzeichen <EOT> (04h) vorkommt, ist dieses zu ersetzen und somit folgendes zu senden:

(01h) (41h) (10h) (84h) (44h) (04h)

Vom Interface zum PC werden auf gleiche Weise zwei Arten von Daten gesendet:

1. **[Daten] <ACK>**, wobei bei den meisten Befehlen keine Daten zurückkommen und somit nur <ACK> gesendet wird
2. **(Fehlercode) <NAK>**, wobei „Fehlercode“ ein ASCII-Zeichen im Bereich „1“ bis „7“ ist. (Tabelle 1 gibt Aufschluß über die Bedeutung der Codes).

Tabelle 2: Liste der Befehle

41h („A“)	Ausgabe 8 Bit direkt auf Kanal A
42h („B“)	Ausgabe 8 Bit direkt auf Kanal B
43h („C“)	Ausgabe 16 Bit direkt auf Kanal A-B
44h („D“)	Einlesen 8 Bit direkt von Kanal A
45h („E“)	Einlesen 8 Bit direkt von Kanal B
46h („F“)	Einlesen 16 Bit direkt von Kanal A-B
47h („G“)	Daten Kanal A setzen
48h („H“)	Daten Kanal B setzen
49h („I“)	Daten Kanal A abfragen
4Ah („J“)	Daten Kanal B abfragen
4Bh („K“)	Puffer Kanal A löschen
4Ch („L“)	Puffer Kanal B löschen
50h („P“)	Triggerstatus abfragen
53h („S“)	Betriebsmodus setzen und Zeigerreset
54h („T“)	Auto-Trigger Mode setzen

Befehlsbeschreibung

In Tabelle 2 sind alle Befehle, die das Interface erkennt, aufgelistet.

Nachfolgend werden alle Befehle einzeln mit ihren Parametern erklärt. Der Befehl ist immer das erste Byte der Nachricht.

Falls keine Daten vom Interface erwartet werden, antwortet dieses entweder mit <ACK> oder mit (Fehlercode) <NAK>, wie bereits beschrieben.

Direkte Befehle

Es folgen zunächst die Befehle, die eine direkte Auswirkung auf die Ein- und Ausgänge des Interface haben:

Befehl „A“

Ausgabe 8 Bit direkt auf Kanal A

Das zweite Byte der Nachricht ist die an Kanal A auszugebende Bitkombination.

Befehl „B“

Ausgabe 8 Bit direkt auf Kanal B

Das zweite Byte der Nachricht ist die an Kanal B auszugebende Bitkombination.

Befehl „C“

Ausgabe 16 Bit direkt auf Kanal A-B

Das zweite Byte der Nachricht ist die an Kanal A auszugebende Bitkombination. Das dritte Byte der Nachricht ist die an Kanal B auszugebende Bitkombination.

Befehl „D“

Einlesen 8 Bit direkt von Kanal A

Es ist kein weiteres Byte erforderlich.

Die Antwort vom Interface gibt den momentanen Zustand am Eingang von Kanal A wieder.

Befehl „E“

Einlesen 8 Bit direkt von Kanal B

Es ist kein weiteres Byte erforderlich. Die Antwort vom Interface gibt den momentanen Zustand am Eingang von Kanal B wieder.

Befehl „F“

Einlesen 16 Bit direkt von Kanal A-B

Es ist kein weiteres Byte erforderlich. Das erste Byte der Antwort vom Interface gibt den momentanen Zustand am Eingang von Kanal A und das zweite Byte den Zustand von Kanal B wieder.

Befehle zur Bearbeitung der Datenpuffer

Da das Interface nicht nur die Möglichkeit der direkten Ein- und Ausgabe bietet, sondern auch selbständig Daten einlesen und auslesen kann, gibt es eine weitere Gruppe von Befehlen, die zur Bearbeitung der internen Datenpuffer erforderlich sind.

Befehl „G“

Daten für Ausgabe auf Kanal A setzen

Das zweite Byte der Nachricht gibt die Zieladresse im Puffer für Kanal A des Interfaces an (0 bis 31). Die nachfolgenden Bytes werden ab dieser Adresse im Puffer abgelegt. Die Anzahl der Bytes ist beliebig, allerdings sollte die Endadresse des Puffers nicht überschritten werden.

Befehl „H“

Daten für Ausgabe auf Kanal B setzen

Das zweite Byte der Nachricht gibt die Zieladresse im Puffer für Kanal B des Interfaces an (0 bis 31). Die nachfolgenden Bytes werden ab dieser Adresse im Puffer abgelegt. Die Anzahl der Bytes ist beliebig, allerdings sollte die Endadresse des Puffers nicht überschritten werden.

Befehl „I“

Daten von Kanal A abfragen

Es ist kein weiteres Byte erforderlich. Als Antwort werden die bisher eingelesenen Daten (max. 32) zurückgesendet.

Befehl „J“

Daten von Kanal B abfragen

Es ist kein weiteres Byte erforderlich. Als Antwort werden die bisher eingelesenen Daten (max. 32) zurückgesendet.

Befehl „K“

Datenpuffer Kanal A löschen

Der Datenpuffer von Kanal A wird gelöscht.

Befehl „L“

Datenpuffer Kanal B löschen

Der Datenpuffer von Kanal B wird gelöscht.

Steuerbefehle

Die letzte Gruppe von Befehlen dient zur Steuerung, zur Konfiguration und zur Abfrage des Interfaces-Status. Hiermit können der Betriebsmodus und der Triggermodus eingestellt und der Triggerstatus abgefragt werden.

Befehl „P“

Triggerstatus abfragen

Die Antwort auf diesen Befehl gibt in den unteren 4 Bits des Wertes den aktuellen Triggerstatus des Interfaces wieder, wobei folgende Bits den einzelnen Kanälen zugeordnet sind:

- Bit 0 : Input A getriggert
- Bit 1 : Input B getriggert
- Bit 2 : Output B getriggert
- Bit 3 : Output A getriggert

Befehl „S“

Betriebsmodus setzen

Im zweiten Byte der Nachricht wird der Betriebsmodus des Interfaces eingestellt. Das dritte und das vierte Byte geben die Länge der Ausgabesequenz für Kanal A und Kanal B an.

Bei den Betriebsmodi wird für jeden Kanal zwischen vier Modi unterschieden:

1. Keine Reaktion auf Triggersignal

Die Veränderung der Ausgänge oder das Einlesen der Eingänge ist nur über die direkten Befehle „A“ bis „F“ möglich.

2. Aus-/Eingabe Adresse 0 bei Trigger

Nach dem Eintreffen des Triggersignals werden die Werte des Datenpuffers „Adresse 0“ an den Ausgang gelegt. Ein weiterer Triggerimpuls ändert nichts mehr.

Bei den Eingängen wird bei jedem Triggersignal der Eingangszustand in den Datenpuffer „Adresse 0“ kopiert. Ein im Puffer vorhandener Wert geht dabei verloren.

3. Ausgabe/Einlesen einer Sequenz, mit Stop an Ende

Bei den Ausgängen wird bei jedem Triggersignal das nächste Datenbyte aus dem Datenpuffer ausgegeben. Ist das Ende der Sequenz erreicht, folgen keine weiteren Daten mehr.

An den Eingängen wird bei jedem Triggersignal der Eingangszustand in den Datenpuffer übernommen und der Adresszeiger um 1 erhöht, sofern das Ende des Puffers noch nicht erreicht ist.

4. Ausgabe/Einlesen einer Sequenz, umlaufend

Gleiche Funktion wie Punkt 3, nur daß bei

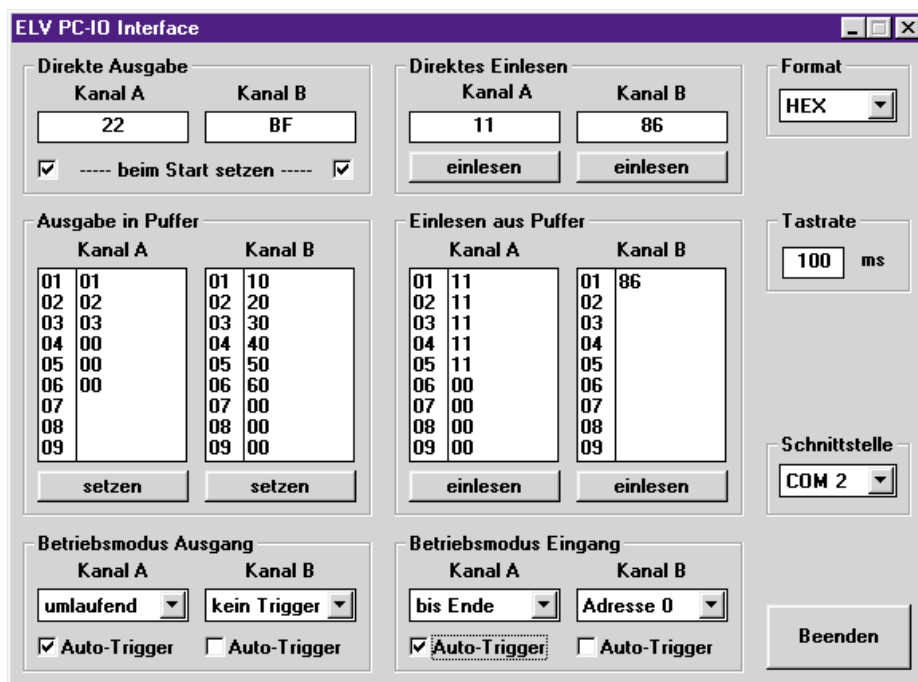


Bild 2: Programmfenster des Windows-Testprogramms. Hier sind alle Steuerungsmöglichkeiten in übersichtlicher Form verfügbar.

Erreichen des Pufferendes wieder am Anfang des Puffers begonnen wird.

Die Bits im Modusbyte haben die in Tabelle 3 beschriebene Zuordnung.

Nach jedem Empfang dieses Befehls werden die Ausgabe- und Eingabezeiger an den Anfang gesetzt.

Befehl „T“

Autotriggermode setzen

Im zweiten Byte der Nachricht wird der Triggermode für jeden Kanal festgelegt. Das dritte Byte gibt den Zeitabstand für automatische Triggerung im 5ms-Raster an (ein Wert von 200 würde einen Triggerabstand von 1s zur Folge haben).

Ist der Autotriggermode für einen Kanal gesetzt, so werden im festgelegten Zeitraster automatisch Triggerimpulse erzeugt, solange das Triggersignal für diesen Kanal High (+5V) ist.

Sofern das Triggersignal Low ist, bleibt diese Funktion gestoppt.

Die Bits im zweiten Byte der Nachricht haben folgende Funktion :

- Bit 0: 1 - Autotrigger für Input Kanal A
- Bit 1: 1 - Autotrigger für Input Kanal B
- Bit 2: 1 - Autotrigger für Output Kanal B
- Bit 3: 1 - Autotrigger für Output Kanal A

Zum Lieferumfang gehört ein kleines Windows-Testprogramm, das die ersten Schritte beim Umgang mit dem Interface erleichtert.


Wie in Abbildung 2 zu sehen, ist es hier nach Auswahl der genutzten Schnittstelle sehr einfach, Ausgänge zu setzen, Eingänge auszulesen und den Inhalt der Datenpuffer zu bearbeiten. Weiter sind ein Setzen des Betriebsmodus sowie die Festlegung des Triggermodus für die einzelnen Kanäle möglich. So kann man sich erst einmal gründlich in den Umgang mit den einzelnen Möglichkeiten des Interfaces einarbeiten, um später eigene Anwendungen zu programmieren. 

Tabelle 3: Zuordnung der Bits im Modusbyte

Bit 1, Bit 0: Betriebsmodus Output Kanal A		
Bit 3, Bit 2: Betriebsmodus Output Kanal B		
0	0:	keine Reaktion auf Trigger
0	1:	Ausgabe Adresse 0 bei Trigger
1	0:	Ausgabe Sequenz, Stop am Ende
1	1:	Ausgabe Sequenz, umlaufend
Bit 5, Bit 4: Betriebsmodus Input Kanal A		
Bit 7, Bit 6: Betriebsmodus Input Kanal B		
0	0:	keine Reaktion auf Trigger
0	1:	Datenbyte nach Adresse 0 bei Trigger
1	0:	Sequenz einlesen, Stop am Ende
1	1:	Sequenz einlesen, umlaufend