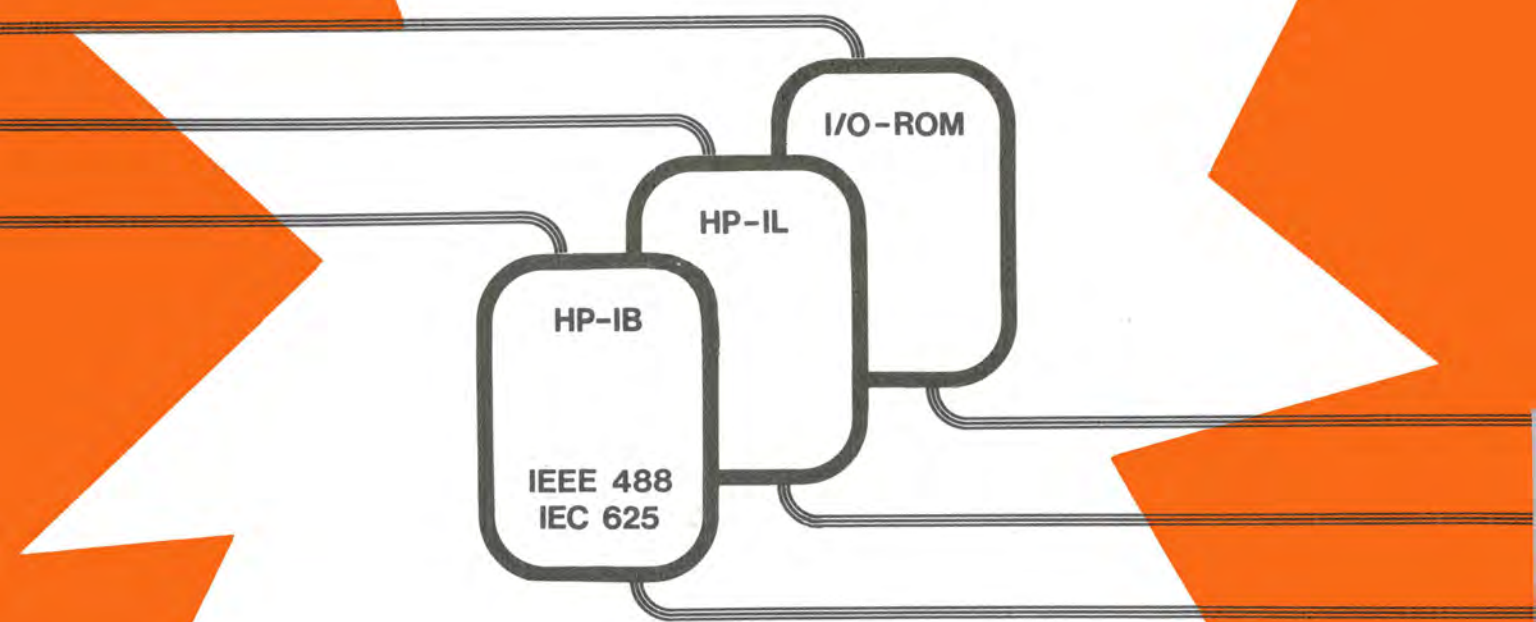
W. Stroinski

Zusammenfassung der Bedienungs- und Programmier-Anleitungen für I/O-ROM, IB- und IL-Interface der HP-Rechner der Serie 80



Heldermann Verlag Berlin

W. Stroinski

Zusammenfassung der Bedienungs- und Programmier-Anleitungen für I/O-ROM, IB- und IL-Interface der HP-Rechner der Serie 80

Heldermann Verlag Berlin

HEWLETT-PACKARD
Personal Computer Division
1010 N.E.Circle Blvd.
Corvallis, OR 97330
U.S.A

Werner Stroinski
Kampweg 7a
D-1000 Berlin 27

CIP-Kurztitelaufnahme der Deutschen Bibliothek

Zusammenfassung der Bedienungs- und Programmier-Anleitungen für I-O-ROM, IB- und Il-Interface der HP-Rechner der Serie 80 / (Hewlett-Packard, Personal Computer Div.). Dt. Bearb. von W. Stroinski. - Berlin : Heldermann, 1986.
ISBN 3-88538-806-5
NE: Stroinski, Werner (Bearb.)

ISBN 3-88538-806-5

In dieser Bearbeitung wurden folgende, zum Lieferumfang der jeweiligen Geräte gehörenden Veröffentlichungen der Firma HEWLETT-PACKARD berücksichtigt:

I/O-Programming Guide,
00085-90142 Rev. D 10/81

I/O-ROM OWNER'S MANUAL,
00087-90121 1/83 (00087-90263)

HP-IB Interface, OWNER'S MANUAL,
82937-90017 1/82 (82937-90026)

HP-IB INSTALLATION AND THEORY OF OPERATION MANUAL,
82937-90007 Rev. B 10/80

HP-IL Interface OWNERS'S MANUAL,
82938-90001 1/82

Die nachfolgend genannten Werke werden im Text erwähnt:

The HP-IL Interface Specification,
82166-90017 12/82

The HP-IL Integrated Circuit User's Manual,
82166-90016 12/82

Gerry Kane, Steve Harper, David Ushijima,
THE HP-IL SYSTEM,
OSBORNE/McGraw-Hill, Berkeley, California U.S.A.

Gerry Kane, Steve Harper, David Ushijima,
Das HP-IL-System,
übersetzt von Ursula Wilk u. Otmar Frey,
McGraw-Hill Book Company GmbH, Hamburg

Inhalts-Verzeichnis

Das IB-Interface

IB-Abschnitt 5

Das IL-Interface

IL-Abschnitt 5

Anhang A

Wie dieses Buch entstand

Zwar wird kaum jemand, der mit Computern umgeht, freiwillig und gern zugeben, daß ihm das Lesen (und Verstehen) des englischen Textes eines MANUALS Schwierigkeiten bereitet, ein Handbuch ist aber auch ihm zweifellos lieber, wenn dieses ohne zusätzliche Fehler in gut lesbarer Form zur Verfügung steht.

Umgekehrt muß man auch die Hersteller verstehen, wenn sie (meist mit Erfolg) versuchen, neue, bessere Geräte auch ohne Beschreibungen in deutscher Sprache an den Kunden zu bringen. Wenn die Geräte erst einmal verkauft sind, dann bringt auch der nachträgliche Hinweis auf deutsche Beschreibungen keinen zusätzlichen Umsatz mehr, weil in der Zeit, die für eine gute Übersetzung notwendig ist, bereits ein "noch moderneres" Gerät auf den Markt drängt.

Ich muß hier ausdrücklich betonen, daß es auch rühmliche Ausnahmen gibt! Die treten aber nur dann auf, wenn sich der Hersteller von Anfang an durch die Beschreibungen in deutscher Sprache einen wesentlich größeren Absatz oder eine länger anhaltende Verkaufsphase erhoffen kann!

Ursprünglich sollten nur einzelne, besonders interessierende Abschnitte für den eigenen privaten und beruflichen Gebrauch übersetzt werden. Dann aber ergab sich, daß das nur Stückwerk wurde, an dem man keine rechte Freude hatte. So wurde der I/O-ROM-Teil, der IB-Teil und die Syntax-Übersicht nach und nach komplett, wobei an diesen Teilen tatsächlich "nur" eine Übersetzung mit leichter Bearbeitung vorgenommen wurde. Weil das eigene Interesse sich auch auf das HP-IL-System ausdehnte, war das IL-Interface als nächstes "fällig". Hier zeigte sich allerdings, daß die Originaltexte wesentlich unergiebiger waren, als die für das IB-Interface. Es wurde daher versucht, auch für das technisch ebenso interessante HP-IL-Interface möglichst viele Fakten zusammenzutragen. Es war damit klar, daß dieser Teil die meiste Arbeit machte und auch nicht mehr als "Übersetzung" zu bezeichnen ist!

Von den auf Seite III in der ersten Gruppe genannten Schriften wurden nur die Kapitel nicht bearbeitet, die sich mit dem Reparatur-Service oder mit sehr internen Vorgängen innerhalb des Interfaces beschäftigen: Der Service-Text ist in vielen deutschen Handbüchern nachzulesen, die internen Schaltvorgänge dürften extrem wenig Leser finden!... Und irgendwann wollte ich ja auch mal zum Ende kommen!

Während der relativ langen Bearbeitungszeit war mir die ideelle und fachliche Unterstützung, die mir von Herrn Jörg Warmuth und Herrn Volkmar Schröder gegeben wurde, eine große Hilfe.

Meiner Frau bin ich ebenfalls zu großem Dank verpflichtet, einmal für die Gelassenheit, mit der sie mein neues Hobby ertrug, aber auch für die Sorgfalt, mit der Sie die Texte auf gute Lesbarkeit prüfte.

Werner Stroinski Juli 1985

Wie man dieses Buch lesen sollte

Bitte, nehmen Sie sich einige Minuten Zeit, um etwas über die Gliederung dieses Buches und seine zweckmäßige Benutzung zu erfahren: Das I/O-ROM und geeignete Interfaces erschließen Ihrem Rechner der Serie 80 ganz neue Anwendungsgebiete.

Im ersten Teil (I/O-ROM) wird der "Zauber-Schlüssel" für diese Gebiete beschrieben. In den I/O-ROM-Abschnitten 1 bis 4 werden, unter Anknüpfen an bereits Bekanntes, einfache aber effektvolle Arbeitstechniken für das I/O-ROM beschrieben. Sie sollten versuchen, diese Abschnitte nach und nach durchzuarbeiten und die vermittelten Verfahren zu erproben, um mit ihnen vertraut zu werden. Auch der versierte I/O-Programmierer wird gelegentlich auf den I/O-ROM-Abschnitt 3 zurückgreifen, da das Gebiet der Formatierung zwar nicht besonders schwierig, aber sehr vielschichtig ist.

Die I/O-ROM-Abschnitte 4 bis 6 beschäftigen sich mit verschiedenen Zahlensystemen und ihrer Darstellung, den Logischen Verknüpfungen, aus denen dann die Logischen Funktionen hergeleitet werden. Im I/O-ROM-Abschnitt 7 werden dann Funktionen zur Umwandlung eines Wertes von einem Zahlensystem in ein anderes beschrieben. Zwar funktioniert das I/O-ROM auch sehr gut mit Zahlen in Dezimal-Darstellung, und es ist im Zusammenhang damit kaum einmal nötig von der gewohnten Dezimal-Darstellung abzuweichen. Manchmal bringt es aber Vorteile, Binär- oder Oktal- oder Hexadezimal-Zahlen zu benutzen, wobei bequeme Umwandlungs-Funktionen sehr nützlich sind.

Wenn Sie mit Binär-, Oktal- und Hexadezimal-Zahlen bereits vertraut sind, können Sie die I/O-ROM-Abschnitte 5 bis 7 überschlagen. Aber auch, wenn Hex(adezimal)- und Oktal-Zahlen für Sie neu sind, schadet es nichts, wenn Sie diese Abschnitte vorerst nur durchblättern. Sie können ja später darauf zurückkommen!

Die I/O-ROM-Abschnitte 8 bis 11 sind dann den "besseren" Verfahren gewidmet, die hauptsächlich zur Beschleunigung und Sicherung des Programm-Ablaufs und zu besserem Zusammenarbeiten von Peripherie und Rechner dienen. Hier müssen Sie alles mit Sorgfalt lesen und auch viel selbst erproben, um alle Vorteile zu erkennen.

Wenn Sie ein Anfänger sind, versuchen Sie, Ihr Programm mit den Methoden zum Laufen zu bringen, die in den Abschnitten 1 bis 3 erklärt werden. Benutzen Sie die Verfahren der Abschnitte 8 bis 11 nur, wenn einfachere Methoden nicht ausreichen!

Der I/O-ROM-Abschnitt 12 erklärt dann noch einige Funktionen, deren Wirkung stark vom Typ des verwendeten Interfaces abhängt.

Im zweiten Teil wird das des HP-IB-Interface (IEEE-488) beschrieben, mit dem sich bequem (meist örtlich begrenzte) Meßsysteme aus Komponenten mit Norm-Schnittstelle aufbauen lassen.

Der dritte Teil vermittelt entsprechende Informationen über das HP-IL-Interface, das in vielen Fällen eine interessante Alternative zum IB-Interface bildet, wenn auch etwas größere Entfernungen zu überbrücken sind.

Im Anhang A sind schließlich die Syntax-Regeln für die Funktionen und Befehle des I/O-ROM's übersichtlich zusammengefaßt und durch eine ausführliche Fehler-Tabelle ergänzt.

I/O-ROM-Abschnitt 1

Was heißt I/O und wozu dient es dem Rechner?

Die Abkürzung "I/O" steht für Input/Output. Das Wort INPUT ist Ihnen sicher schon als BASIC-Anweisung geläufig und wurde in Programmen immer dann benutzt, wenn der Rechner Eingaben über die Tastatur erwartete. Der Sinn dieses Wortes bleibt zwar erhalten, aber sein Geltungsbereich wird erweitert. Mit dem Wort "OUTPUT" dürften Sie wahrscheinlich noch nicht so vertraut sein. Dafür kennen Sie sicherlich schon die BASIC-Worte DISP oder PRINT, die "Sonderfälle" von OUTPUT darstellen.

Bisher hatte der Rechner nur per Bildschirm und/oder Drucker die Möglichkeit, mit Ihnen einen Dialog zu führen. Dabei mußten Sie die Daten über die Tastatur eingeben und konnten die Ergebnisse auf dem Monitor oder auch gedruckt lesen.

Nun sind Bedienen einer Tastatur bzw. Lesen eines Textes Tätigkeiten, die typisch für uns Menschen sind: Es wird kaum ein einfaches Gerät geben, daß diese Aufgaben übernimmt! Damit ist klar, daß der Datenaustausch des Rechners mit weiteren Geräten über andere, einfachere Wege laufen muß: Diese anderen Wege werden durch die verschiedenen "Interfaces" und auch durch das I/O-ROM eröffnet.

Erst durch diese Erweiterungen kann der Rechner in den Geräten seiner Peripherie Ereignisse erkennen und Abläufe steuern.

Wenn der Rechner ein Ereignis erkennen soll, dann muß er von dem betreffenden Gerät eine Nachricht "hören", wir nennen ihn deshalb "Listener". Das "sprechende" Gerät wird als "Talker" bezeichnet. Die Nachricht ist ein "Input", weil sie beim Rechner "eingeht". Die Rollen können auch wechseln: Wenn der Rechner einen Ablauf auslösen oder steuern soll, dann ist er der "Talker" und das angesprochene Gerät der "Listener". Diese Nachricht wird vom Rechner "ausgegeben" und ist deshalb ein "Output".

Der Vorgang der Daten-übermittlung ist im Einzelnen bei den verschiedenen Geräten und Interfaces in seinem Ablauf sehr unterschiedlich, es ist aber glücklicherweise nicht nötig, daß Sie alle technischen Einzelheiten der Interfaces kennen müssen, um mit ihnen arbeiten zu können. Deshalb hat dieses Buch mehrere voneinander relativ unabhängige Teile, was die Durcharbeitung erleichtern soll. Auf Seite XII werden Hinweise für eine zweckmäßige Reihenfolge bei der Lektüre gegeben. Dem IB- und dem IL-Interface sind besondere Beschreibungen gewidmet, das Printer-ROM wird in diesem Abschnitt nur kurz erwähnt, für weitere Informationen über seine Anwendung wird auf das entsprechende Handbuch verwiesen. Der Teil dieses Buches, der das I/O-ROM behandelt und der Anhang A dürften auch für die Benutzer von anderen Interface-Bauarten (RS-232, GPIO und BCD) interessant sein, wenn auch bei den für die Bauart spezifischen Problemen auf die passenden Handbücher verwiesen werden muß.

Dieses Handbuch ist auf die Rechner HP-83/85 (hierzu I/O-ROM 00085-15003) und die Rechner HP-86/87 (hierzu I/O-ROM 00087-15003) zugeschnitten.

Einsetzen des I/O-ROM's

Bei einigen Ausführungen der eben aufgezählten Rechner ist das I/O-ROM (mitunter auch das Mass-Storage-ROM und/oder ein Printer-ROM) bereits fest eingebaut. Zur grundsätzlichen Klärung, ob ein bestimmtes ROM dem Rechner zur Verfügung steht, ist es nur notwendig, vom Rechner die Ausführung einer Anweisung zu verlangen, die ausschließlich von dem gesuchten ROM bereitgestellt wird. Für das I/O-ROM ist dazu IOBUFFER A$ gut geeignet. Wenn nach Eingabe der Anweisung und Betätigung der END LINE-Taste keine Fehlermeldung erfolgt, ist ein I/O-ROM entweder fest im Rechner eingebaut oder bereits in einem ROM-Einschub vorhanden. Falls der Rechner mit der Meldung "BAD STMT" antwortet, fehlt das I/O-ROM. Der Ein- und Ausbau des ROM's ist in der Bedienungsanleitung zum ROM-Einschub HP 82936A ausführlich beschrieben. Hier wird nur auf die Notwendigkeit hingewiesen, den Rechner vor Ein- bzw. Ausbau der ROM- oder RAM-Einschübe unbedingt auszuschalten!.

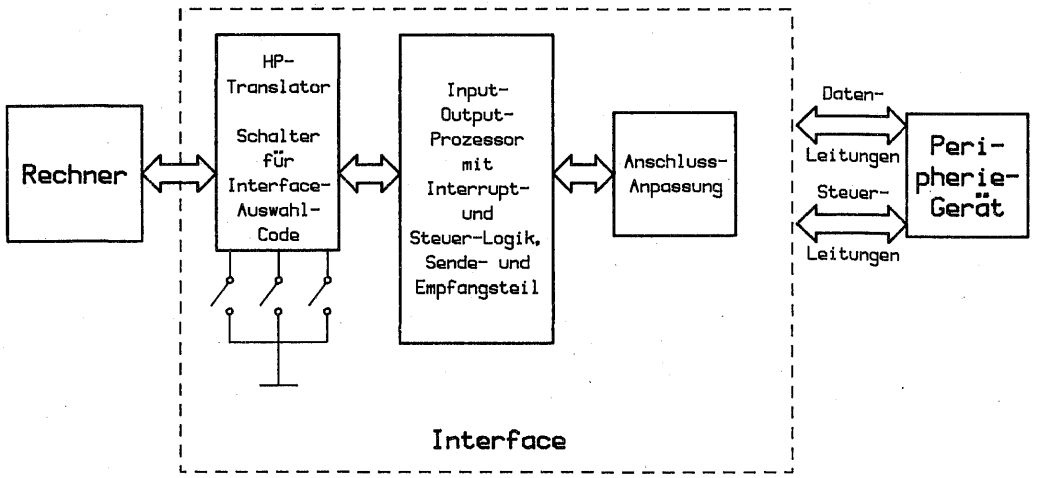
Welche Aufgabe hat ein Interface?

Ein Interface ist ein Gerät, das den ordnungsgemäßen Austausch von Daten mit den Peripherie-Geräten ermöglicht. Dabei müssen vier Gebiete berücksichtigt werden:

Anpassung in mechanischer Hinsicht
Anpassung in elektrischer Hinsicht
Anpassung in der Daten-Codierung
Anpassung in der Übertragungs-Geschwindigkeit

Das folgende Schaubild zeigt die grundsätzlich erforderlichen Elemente, die ein Interface für seine Vermittler-Aufgabe zwischen Rechner und Peripherie benötigt:

Interface-Funktions-Blöcke



Mechanische Kompatibilität

Mechanische Kompatibilität bedeutet letztlich nur, daß Einschübe und Führungen bzw. Stecker- und Buchsenanordnungen zueinander passen. Die Serie 829XX erfüllt diese Bedingungen hinsichtlich der HP-Rechner der Serie 80. Einige Interface-Ausführungen, wie z.B. das HP-IB- und das HP-IL-Interface passen stets auch zu den Peripherie-Geräten des gleichen Systems. Andere Interfaces (z.B. GPIO-Interface) werden ohne äußere Anschlußstecker geliefert. Dann müssen Sie sich die passenden Stecker selbst besorgen und montieren. Ziehen Sie dabei das Handbuch zu Rate, in dem das benutzte Interface beschrieben wird!

Elektrische und Daten-Kompatibilität

Elektrische Kompatibilität bedeutet für das Interface, die Spannungswerte, die im Rechner auftreten, so umzusetzen, daß sie für die Peripherie-Geräte passen. Eng verbunden ist damit meist auch eine Anpassung der Befehle und Meldungen, die zwischen Rechner und Peripherie übermittelt werden. Wenn man die elektrische Anpassung etwa damit vergleichen will, daß sich die Teilnehmer an einem Gespräch auf eine allen zusagende Lautstärke einigen müssen, dann ist die Daten-Kompatibilität eine Anpassung der verwendeten Sprachen. Dazu ist ein Dolmetscher notwendig, wenn die Partner keine gemeinsame Sprache benutzen. Diese Probleme werden im Interface mit elektr(on)ischen Mitteln gelöst, wobei es durchaus vorkommt, daß manche Baugruppen sowohl an der elektrischen Anpassung als auch an der Übersetzung beteiligt sind. Mit der Codierung und Decodierung von Befehlen und Meldungen ist außer dem Interface meist auch noch der Rechner und das I/O-ROM beschäftigt.

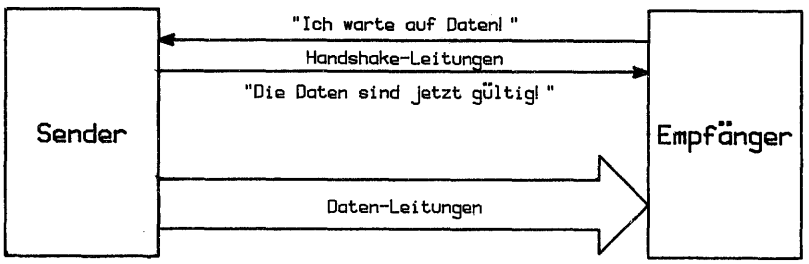
In allen Interface-Bauarten ist ein als 'I/O-Prozessor' bezeichneter Mikro-Computer enthalten, der einige Kurz-Programme für die Datenumsetzung und die Durchführung des Handshake-Verfahrens enthält. Durch die vom I/O-ROM bereitgestellten Befehle läßt sich der I/O-Prozessor auch auf die verschiedenen Handshake-Verfahren umschalten.

Zeit-Kompatibilität

Das Sprech- und Hör-Tempo ist bei den meisten Menschen recht ähnlich, Rechner und Peripherie-Geräte haben dagegen so verschiedene Arbeitsgeschwindigkeiten, daß unbedingt ein besonderes "Ritual" zur erfolgreichen Übertragung von Daten erforderlich ist. Dieses Ritual wird als "Handshake" bezeichnet. Wenn es davon auch verschiedene Durchführungsarten gibt, bleibt doch das folgende Grundschema für alle Verfahren gültig:

1. Der Empfänger gibt bekannt, daß er Daten annehmen kann.
2. Der Sender bietet Daten an und erklärt diese für gültig.
3. Sobald der Empfänger das "Gültig"-Signal erkannt hat, beginnt er mit der Aufnahme der Daten und teilt dem Sender mit, daß er "beschäftigt" ist.
4. Der Sender hält die Daten so lange bereit, bis der Empfänger meldet, daß er für neue Daten empfangsbereit ist. Dann wiederholt sich der Vorgang.

Die Grundidee des "Handshake"-Verfahrens läßt sich so darstellen:



Auswahl von Quelle oder Ziel einer Nachricht

Wenn jemandem eine Nachricht mit der Post übermittelt werden soll, dann muß die Adresse angegeben werden, weil sonst die Zustellung unmöglich ist. Entsprechend ist auch der Umgang mit den Peripherie-Geräten geregelt: Im Programm muß stehen, mit welchem der Peripherie-Geräte der Rechner zusammenarbeiten soll. Dieser Auswahlvorgang wird als "Adressierung" bezeichnet. Die Rechner der Serie 80 wählen die Peripherie-Geräte durch eine Geräteadresse in den I/O-Anweisungen aus. Die Geräteadresse entspricht etwa der Postadresse eines Briefes.

Die Rechner der Serie 80 können zwei unterschiedliche Verfahren zum Adressieren benutzen. Welches der Verfahren zu benutzen ist, hängt vom verwendeten Interface und seiner Betriebsart ab. Die folgenden zwei Absätze beschäftigen sich mit Einzelheiten der beiden Adressierungsarten.

Ausschließliche Benutzung des Interface-Auswahl-Codes

Wenn Sie weder das HP-IB- noch das HP-IL-Interface benutzen und nur ein einziges Peripherie-Gerät am Interface angeschlossen ist, dann entspricht das einer Postsendung, die für die Bewohner eines Einfamilienhauses bestimmt ist: Es genügt die Angabe von Straße und Hausnummer. Diese Daten entsprechen etwa dem 'Interface-Auswahl-Code', für den eine Zahl von 3 bis 10 (einschließlich) einzusetzen ist.

Die verschiedenen Ausführungen der Interfaces werden vom Hersteller bei Auslieferung auf bestimmte Auswahl-Codes eingestellt. Diese Einstellungen sind in der folgenden Tabelle zusammengefaßt:

Listen-Nr.:	Name:		Auswahl-Code eingestellt auf:
82937A	HP-IB	(parallel)	7
82938A	HP-IL	(seriell)	9
82939A	RS-232	(seriell)	10
82940A	GPIO	(parallel)	4
82941A	BCD	(parallel)	3
82949A	Printer	(parallel)	8

Achtung:
Aus elektrischen Gründen darf jeder Auswahl-Code bei den im Rechner eingesetzten Interfaces nur einmal vorkommen!

Die mehrfache Verwendung des gleichen Auswahl-Codes führt unweigerlich zu Störungen: Keines der davon betroffenen Interfaces wird einwandfrei arbeiten! Wie Sie aus der obigen Tabelle ersehen, schützt die vom Hersteller gewählte Voreinstellung so lange vor derartigen Pannen, bis Sie mehr als ein Interface der gleichen Art benutzen. Falls dieser Fall eintritt, muß der Auswahl-Code im Interface umgestellt werden. Die hierzu notwendigen Arbeiten sind in der jeweiligen Beschreibung für das Interface angegeben. Beachten Sie bitte die dort gegebenen Hinweise sorgfältig.

Zusätzliche Benutzung der Primär-Adressen

Wenn Sie ein HP-IB- oder HP-IL-Interface benutzen oder wenn mehrere Peripherie-Geräte an ein Interface angeschlossen sind, ist die Adresse eine Zahl, die 3 oder 4 Stellen aufweist und aus dem Auswahl-Code des Interfaces und der Primär-Adresse eines Peripherie-Gerätes zusammengesetzt ist. Diese Adressier-Methode gleicht der Postadresse für ein Appartement-Haus: die Hausnummer leitet den Brief zum richtigen Haus, und die Appartementnummer stellt sicher, daß er in das richtige Fach des Hausbriefkastens gelangt. Die Appartementnummer entspricht dabei der Primär-Adresse des Peripherie-Gerätes, das die Nachricht erhalten soll. Damit wird dieses Gerät aus der Gruppe aller Peripherie-Geräte ausgewählt, die am selben Interface angeschlossen sind. Hier einige Beispiele:

```
Die Geräteadresse  721 bezeichnet Gerät 21 am Interface  7
Die Geräteadresse  301 bezeichnet Gerät  1 am Interface  3
Die Geräteadresse 1002 bezeichnet Gerät  2 am Interface 10
```

Drucken mit Peripherie-Geräten

Der einfachste Weg, die vom Rechner ausgegebenen Daten einem Peripherie-Gerät zuzuleiten, ist die "PRINTER IS"-Anweisung. Das I/O-ROM schafft nun (ebenso, wie das Printer-ROM oder Plotter/Printer-ROM) die Möglichkeit, mit Anweisungen wie "PRINTER IS 4" oder "PRINTER IS 720" ein geeignetes Peripherie-Gerät zum Drucker zu machen. Dadurch wird das "PRINTER IS"-Gerät zum Ziel für alle durch "PRINT"- oder "PLIST"-Anweisungen verursachten Ausgaben.

Entsprechendes gilt auch für die "CRT IS"-Anweisung. Das "CRT IS"-Gerät wird dann das Ziel aller durch "DISP"-, "LIST"- und "CAT"-Anweisungen verursachten Ausgaben und auch für die "Error"- und "Warning"-Meldungen. Eine Ausnahme bilden Graphik-Anweisungen, die immer an den als Monitor benutzten Bildschirm gehen.

Mit "PRINTER IS" bzw. "CRT IS" kann jede gültige Adressierung verknüpft werden.

Wenn Programm-Listings zu einem Peripherie-Gerät übertragen werden, dann wird jede Programm-Zeile als in sich geschlossene Zeichenkette ausgeben. "PRINT"- und "DISP"-Anweisungen werden dabei aber mit der Standard-Zeilenlänge des Rechners ausgeführt. Speziell für den HP-83/85 bedeutet dies, daß auch das Peripherie-Gerät vorerst nur mit einer Zeilenlänge von 32 Zeichen arbeitet, genau so wie die eingebauten Geräte! Hier hilft nur eine Anweisung in der Form "PRINTER IS 704,80" bzw. "CRT IS 710,80", wobei die Zahl hinter dem Komma die gewünschte Zeichenzahl pro Zeile angibt. Wenn Sie einen HP-83/85 benutzen, werden Sie jetzt auch verstehen, weshalb kein Interface-Auswahl-Code unter "3" zulässig ist: Der eingebaute Bildschirm bzw. Drucker werden mit "1" bzw."2" angesprochen.

Wenn Ihnen die durch den Ersatzwert vorgegebene Zeilenlänge nicht zusagt, dann hilft auch bei den übrigen Rechnern der Serie 80 der für den HP-83/85 angegebene Tip. Näheres darüber und über die Möglichkeit, besser angepaßte Formatierungen zu benutzen, finden Sie in den nächsten zwei I/O-ROM-Abschnitten, denen Sie einige Aufmerksamkeit schenken sollten.

Wenn Sie dagegen das I/O-ROM und das Interface nur dazu benutzen, um Programm-Listings oder "PRINT"- bzw. "DISP"-Anweisungen von Peripherie-Geräten ausführen zu lassen, dann brauchen Sie nicht weiterzulesen. Setzen Sie das geeignete Interface ein, schließen Sie ihren Drucker und/oder Monitor an, fügen Sie, wo nötig, "PRINTER IS"- oder "CRT IS"-Anweisungen in Ihre Programme ein, und Sie sind schon fertig!

Einfache I/O-Operationen

Einführung

Im I/O-ROM-Abschnitt 1 wurde die Durchführung von Daten-Ausgaben mit PRINTER IS- und PRINT-Anweisungen behandelt. Obwohl diese Verfahren recht bequem erscheinen, erkennt man doch sehr schnell deren Schwächen: Störend ist es schon, daß es keine "KEYBOARD IS"-Anweisung gibt, um damit Daten von Peripherie-Geräten zu empfangen. Unbequem ist es aber auch, daß gleiche Daten an mehrere Geräte nur nacheinander ausgegeben werden können, was einmal viel Zeit beansprucht und außerdem jedesmal eine zusätzliche PRINTER IS-Anweisung mit der neuen Adresse nötig macht, da sich in der PRINT-Anweisung immer nur eine Adresse unterbringen läßt.

Die wichtigsten Hilfsmittel beim Einsatz der Interfaces zur übermittlung von Daten von und zum Rechner sind die "OUPUT"- und "ENTER"-Anweisungen. Sie bilden den eigentlichen 'Kern' der I/O-Technik. Mit ihnen kann man einfach und auch relativ schnell Daten von der Quelle zum endgültigen Ziel bringen. Diese beiden Anweisungen reichen für viele Zwecke vollkommen aus.

Die einfachen OUTPUT- und ENTER-Anweisungen (wie in diesem Abschnitt beschrieben) benutzen für alle Daten die ASCII-Darstellung. ASCII steht für

American **S**tandard **C**ode for **I**nformation **I**nterchange.

Diese genormte Codierung für Buchstaben, Zahlen, Satzzeichen und Spezialsymbole ist sehr weit verbreitet. Der ASCII-Code ist also eine übersetzung zwischen dem Binär-Code, wie ihn der Rechner versteht und den alphanumerischen Symbolen, die die Menschen lesen können. Eine komplette Liste der Zeichen des ASCII-Satzes und der entsprechenden Code-Werte findet sich im Anhang A dieses Buches, aber auch im Handbuch für Ihren Rechner.

Die "OUTPUT USING"- und "ENTER USING"-Anweisungen leisten wertvolle Hilfe, wenn eine besondere Formatierung gewünscht wird oder wenn ein Binär-Code einmal nicht als ASCII-Zeichen verstanden werden soll. Diese Form der Aus- und Eingaben wird im I/O-ROM-Abschnitt 3 behandelt.

Die Anwendung einfacher OUTPUT-Anweisungen

Die einfache OUTPUT-Anweisung kann immer da verwendet werden, wo auch eine PRINT-Anweisung korrekte Ergebnisse liefern würde. Im Unterschied zur PRINT-Anweisung, der eine Adressierung (PRINTER IS ...) vorausgehen muß, enthält die OUTPUT-Anweisung die Adresse(n) selbst und kann dadurch die Daten auch an mehrere Geräte zugleich ausgeben. Auf die Adressen folgt dann eine Liste der auszugebenden Daten. Hier nun einige Beispiele von korrekten OUTPUT-Anweisungen:

```
OUTPUT 1 ; "Hallo"
OUTPUT 3 ; X
OUTPUT 51 ; A$,B$
OUTPUT 703,725 ; X;Y;Z
OUTPUT 1000 ; A(1);B(3),N$[2,7]
```

Beachten Sie bitte, daß ein Semikolon die Geräteadresse(n) von der Ausgabeliste trennt, daß aber innerhalb dieser Ausgabeliste Komma oder Semikolon zur Trennung verwendet werden dürfen. Die Ausgabeliste darf numerische Konstanten, Variable, konstante oder variable Zeichenketten enthalten. Ein "Wagenrücklauf" (CR) und ein "Zeilenvorschub" (LF), im folgenden "EOL-Sequenz" (End Of Line) genannt, wird hinter dem letzten Datenblock ausgegeben. Komma bzw. Semikolon innerhalb der Ausgabeliste haben Einfluß darauf, mit welchem Abstand, im folgenden "Feld" genannt, die Datenblöcke gedruckt bzw. angezeigt werden. Die einfache OUTPUT-Anweisung benutzt die gleichen Feldgrößen wie die PRINT-Anweisung: Das Semikolon sorgt für ein "Kompakt-Feld", während das Komma ein "Frei-Feld" schafft. Die Eigenschaften dieser "Felder" sind in der folgenden Tabelle zusammengestellt:

	Numerische Daten	Zeichenketten
Kompakt-Feld (;)	Leerstelle bei positivem, Minus-Zeichen bei negativem Wert vor den Ziffern, eine Leerstelle nach den Ziffern	Ohne führende Leerstelle werden nur die Zeichen der Kette ohne nachfolgende Leerstellen ausgegeben.
Frei-Feld (,)	Leerstelle bei positivem, Minus-Zeichen bei negativem Wert vor den Ziffern. Linksbündig ausgegeben. Das Frei-Feld wird durch nachfolgende Leerstellen auf eine Länge von 11, 21 oder 32 Zeichen ergänzt.	Ohne führende Leerstelle werden die Zeichen der Kette ausgegeben, gefolgt von höchstens 20 Leerstellen.

Die Größe des "Frei-Feldes" wird, wie auch bei PRINT- und DISP-Anweisungen, so gesteuert: Der Rechner setzt immer voraus, daß er die Daten auf einem Bildschirm darstellen muß und daß er deshalb in den Spalten 1 bzw. 22 (des 32-spaltigen Displays vom HP-83/85) bzw. in den Spalten 1, 22, 43, 64 (des 80-spaltigen Displays des HP-87) mit den Daten beginnen muß. Für viele Peripherie-Geräte wird ein derartiges "Frei-Feld" aber unzweckmäßig sein, man denke hier z.B. an das Auflisten von Zahlenwerten mit einem Drucker, der 80 Zeichen in einer Zeile unterbringen kann! Es ist aber leicht, dieses Problem durch die Anwendung des Semikolons in der Ausgabeliste zu umgehen oder man bedient sich gleich der "formatierten" Ausgabe, die im I/O-ROM-Abschnitt 3 beschrieben wird.

Die Anwendung einfacher ENTER-Anweisungen

Eine einfache ENTER-Anweisung kann da verwendet werden, wo auch eine INPUT-Anweisung korrekte Ergebnisse liefern würde. Die ENTER-Anweisung enthält die Adresse des als Quelle benutzten Gerätes und eine Liste der aufzunehmenden Daten. Sie erinnern sich, daß INPUT-Anweisungen nur das Tastenfeld als Quelle hatten und deshalb keine Geräteadresse brauchten, während ENTER-Anweisungen sich immer auf ein Peripherie-Gerät als Quelle beziehen und deshalb dessen Adresse enthalten müssen. Hier nun einige Beispiele von korrekten ENTER-Anweisungen:

```
ENTER 3 ; X
ENTER 51 ; A$,B$,C$
ENTER 703 ; X,Y,Z
ENTER 1000 ; A(1),B(3),N$
```

Beachten Sie, daß ein Semikolon die Adresse von der Eingabeliste trennt, daß aber innerhalb der Eingabeliste nur Kommata verwendet werden dürfen. Die Eingabeliste darf numerische Variable und Variable für Zeichenketten enthalten.

Für einen erfolgreichen Einsatz der ENTER-Anweisung muß man unbedingt wissen, was Beginn und Ende einer Daten-Aufnahme im Rechner auslöst. Die einfache ENTER-Anweisung in den obigen Beispielen, bedeutet für den Rechner "Frei-Feld" beim Formatieren der eintreffenden Daten. Dieses Format ist nun für numerische Daten und Zeichenketten unterschiedlich.

Die Aufnahme numerischer Daten

Der Rechner nimmt numerische Daten durch Lesen der ASCII-Darstellung dieser Werte auf. Wenn der Rechner z.B. eine ASCII "1", danach eine ASCII "2" und schließlich eine ASCII "5" liest, ordnet er den Wert 125 der entsprechenden Variablen zu.

Wenn man verstanden hat, wie der Rechner eine Zahl im Frei-Feld-Format aufnimmt, klären sich auch andere "Geheimnisse" des I/O. Nehmen wir einmal an, daß die folgende Anweisung in unserem Programm steht:

```
ENTER 3 ; X,Y
```

Nehmen wir weiter an, daß der Rechner (über das Interface mit dem Auswahl-Code 3) folgende Zeichen empfangen hat:

D	I	E	N	S	T	A	G		D	E	Z		1	1	,		1	9	7	9	CR	LF

Der Rechner nimmt keines der nicht-numerischen Zeichen zur Kenntnis, die Zeichen "DIENSTAG DEZ" haben keine Wirkung. Danach aber wird die "11" gelesen. Sobald der Rechner begonnen hat, eine Zahl zu bilden, beendet ein nicht-numerisches Zeichen diesen Lesevorgang. Hier ist es das Komma hinter der "11", das den Rechner veranlaßt, der Variablen X den Wert 11 zuzuordnen und auf die nächsten numerischen Daten zu warten. Das Leerzeichen vor "1979" wird ignoriert und der Rechner liest dann "1979". Das Wagenrücklauf-Symbol "CR" bewirkt, daß der Wert 1979 der Variablen Y zugeordnet wird. Dann liest der Rechner weiter und findet das Zeilenvorschub-Symbol "LF". Dieses schließt die ENTER-Anweisung ab, und der Rechner wird, mit X=11 und Y=1979, zur Bearbeitung der nächsten Programmzeile weitergehen.

Der eben beschriebene Vorgang der Aufnahme von numerischen Daten, die als Frei-Feld formatiert sind, läßt sich so zusammenfassen:

1. Voranstehende nicht-numerische Zeichen werden überlesen.
2. Voranstehende, eingefügte und nachfolgende Leerzeichen werden überlesen.
3. Die numerischen Zeichen werden zum Aufbau æiner Zahl benutzt.
4. Der Aufbau einer Zahl wird durch ein folgendes nicht-numerisches Zeichen abgeschlossen.
5. Die Zeichen werden gelesen, bis ein Zeilenvorschub-Symbol (LF) auftritt.

In den bisherigen Erläuterungen sprachen wir ohne Angabe von Einzelheiten von numerischen und nicht-numerischen Zeichen. Nun sind die Ziffern "0" bis "9" immer numerische Zeichen, aber auch der Dezimalpunkt, das Plus- bzw. Minus-Zeichen und der Buchstabe "E" können numerisch interpretiert werden, wenn sie innerhalb einer Zahl an einer sinnvollen Stelle auftreten. Wenn die folgenden Zeichen

|-|-|T|E|S|T| |1|2|.|5|E|-|3|

von einer ENTER-Anweisung gelesen wird, die einen numerischen Wert erwartet, dann spielt sich folgendes ab:

Die führenden Minus-Zeichen und das "E" in "TEST" werden überlesen, weil sie keinen sinnvollen Bezug zu einem Zahlenwert haben. Aber die Zeichen "12.5E-3" werden in $12.5 * 10^{-3}$ umgesetzt. Hier treten das Minuszeichen und das "E" in einer sinnvollen Verknüpfung mit Ziffern auf und werden deshalb als Bestandteile einer Zahl erkannt.

Die Aufnahme von Zeichenketten

Der Rechner nimmt Zeichenketten auf, indem er die ASCII-Zeichen der Reihe nach in einer String-Variablen ablegt. Für einen String im Frei-Feld-Format läuft der Vorgang so lange weiter bis:

1. die vorhandene String-Variable gefüllt ist oder
2. ein Zeilenvorschub-Symbol (LF) oder
3. Wagenrücklauf- und Zeilenvorschub-Symbol(e) (EOL) gelesen werden.

Wenn im Programm steht:

```
ENTER 4 ; A$,B$,C$
```

Dann wird der Rechner auf die Zeichenkette

|H|A|L|L|O|LF|CR|LF|D|O|R|T|CR|LF|

in folgender Weise reagieren: Er beginnt mit dem Lesen der Buchstaben "HALLO", die in der String-Variablen A$ abgelegt werden. Dies endet beim Lesen des ersten Zeilenvorschub-Symbols. Beachten Sie, daß dieses Symbol selbst nicht in A$ abgelegt wird, es beendet nur das Ablegen in A$! Jetzt beginnt das Lesen für B$: Hier folgen nur Wagenrücklauf-Symbol und ein Zeilenvorschub-Symbol, wodurch auch das Lesen für B$ beendet wird. Weil keines der beiden Symbole nach B$ gelangt, wird B$ ein Nullstring. Nun beginnt das Lesen für C$, wodurch die folgenden Buchstaben in C$ abgelegt werden. Dieser Vorgang wird wieder beim Lesen der Zeichen "CR" und "LF" beendet. Da jetzt alle Variablen besetzt sind und der Rechner ein "LF" als letztes Zeichen erkannt hat, geht er mit A$="HALLO", B$="" und C$="DORT" an die Bearbeitung der nächsten Programmzeile.

Beachten Sie, daß ein Wagenrücklauf-Symbol nur dann überlesen wird, wenn ihm ein Zeilenvorschub-Symbol unmittelbar folgt. Ein "CR" (ohne folgendes "LF") werden Sie deshalb im aufgenommenen String wiederfinden.

Ein anderes Beispiel soll zeigen, was ein "voller" String ist. Diesmal setzen wir folgende Anweisungen voraus:

```
DIM X$[3]
ENTER 4 ; X$
```

Und die folgenden Daten sollen beim Rechner eintreffen:

B	O	Y	C	O	T	T	CR	LF

Der Rechner liest die Zeichen, kann aber nur die ersten drei ("BOY") in X$ ablegen, da diese Stringvariable damit voll besetzt ist. Der Rechner liest aber die gesamten Daten, bis er das Zeilenvorschub-Symbol findet. Erst dann ist die ENTER-Anweisung ausgeführt und der Rechner geht nun mit X$="BOY" zur Bearbeitung der nächsten Programmzeile über.

Notizen:

I/O-Operationen mit Formatierung

Einführung

Obwohl die Formatierung mit dem "Frei-Feld" für einige I/O-Aktivitäten recht gut brauchbar ist, gibt es doch viele Fälle, bei denen eine exaktere Steuerung des Formates nötig ist: Womöglich liegen Daten als Binär-Muster vor, für die es keine Beziehung zum ASCII-Code gibt, oder ein Zeilenvorschub-Symbol ist nicht erwünscht bzw. nicht zu erwarten. Oder eine Zahlenkolonne soll nach Dezimalzeichen ausgerichtet werden, oder es werden nur 2-stellige Exponenten gewünscht oder...oder... Kurzum, die Liste der Gründe für eine Formatierung ist sehr, sehr lang!

Die Formatierung der auszugebenden oder eintreffenden Daten wird durch die IMAGE-Spezifikatoren bewirkt. Diese Spezifikatoren können entweder in einer besonderen IMAGE-Anweisung stehen, auf die im Bedarfsfall zurückgegriffen wird, es ist aber auch möglich, die Spezifikatoren direkt in die betroffenen OUTPUT- bzw. ENTER-Anweisungen einzufügen. Der I/O-ROM-Abschnitt 3 dieses Buches beschreibt Einzelheiten über Anwendung und Wirkung der IMAGE-Spezifikatoren.

Formatierte Ausgabe von Daten

Mit der Steuerung des Ausgabe-Formates lassen sich Formatierungen, Abstände und Raumaufteilung der auszugebenden Daten exakt steuern. Außerdem kann damit auch über die Ausgabe oder Unterdrückung der EOL-Sequenz entschieden werden. Der Rechner formatiert die auszugebenden Daten gemäß der OUTPUT-Anweisung, die entweder selbst eine Formatierungsangabe enthält oder die passende IMAGE-Anweisung nennt. Hierfür gibt es drei Formen, wobei die unter 3. gezeigte Label-Technik für den HP-83/85 nicht anwendbar ist:

1. 10 IMAGE<Output-Format>
   20 OUTPUT<Ziel>USING 10;<Ausgabe-Liste>
2. OUTPUT<Ziel>USING<Output-Format>;<Ausgabe-Liste>
3. 10 LBL: IMAGE<Output-Format>
   20 OUTPUT<Ziel>USING LBL;<Ausgabe-Liste>

Die obige Darstellung zeigt die allgemeine Form der OUTPUT USING-Anweisungen. Nun einige konkrete Beispiele (wobei die Zeilen 10 und 60 wegen des Aufrufs durch Label beim HP-83/85 nicht möglich sind!):

```
10 TOT: IMAGE "Total=",ZZ.D
20 IMAGE 5A,2X,17A
.
.
60 OUTPUT 4 USING TOT ; C1,C2,C3
70 OUTPUT 701 USING 20 ; A$,B$
80 OUTPUT 9 USING "#,B" ; X
90 OUTPUT 53 USING "MDDD.DD" ; T(1),T(2)
100 OUTPUT 710,711 USING I$ ; N$,A
```

In der allgemeinen Form steht "Ziel" für "Geräteadresse". Deren Aufbau ist im I/O-ROM-Abschnitt 1 erläutert. "Output-Format" steht an Stelle einer korrekten Zusammenstellung von Image-Spezifikatoren. Die Liste der Image-Spezifikatoren kann eine in Anführungszeichen eingeschlossene Zeichenkette oder eine String-Variable sein, die ihrerseits die Liste der Image-Spezifikatoren enthält. Die Spezifikatoren müssen innerhalb der Liste durch Kommata getrennt sein. Die "Ausgabe-Liste" enthält die Aufzählung der auszugebenden Daten, wobei es hier gleichgültig ist, ob die einzelnen Datenverkörperungen durch Komma oder Semikolon voneinander getrennt sind, weil das endgültige Ausgabeformat und auch die Abstände durch die Image-Spezifikatoren bestimmt werden. In der Ausgabeliste sind nun Semikolon und Komma "gleichberechtigt".

Formatierung von numerischen Daten

Die Image-Spezifikatoren dieser Gruppe beeinflussen das Format der auszugebenden Daten. Die meisten dieser Spezifikatoren werden Ihnen schon von den PRINT USING- bzw. DISP USING-Anweisungen her bekannt sein. Diese Spezifikatoren lassen sich nach ihren Aufgaben in drei Gruppen aufteilen:

1. Spezifikatoren für die Stellenzahl
2. Spezifikatoren für das Vorzeichen
3. Spezifikatoren für das Dezimalzeichen und die Stellengruppierung

1. Spezifikatoren für die Stellenzahl:

Diese Spezifikatoren entscheiden über die Zahl der ausgegebenen Stellen vor und hinter dem Dezimalzeichen, die Ausgabe oder Unterdrückung führender Nullen und die Form des ausgegebenen Exponenten.

Image-Spezifikator:	Wirkung auf das Ausgabe-Format:
D	Veranlaßt die Ausgabe einer Stelle einer Zahl. Wenn es sich um eine führende Null handelt, wird nur eine Leerstelle ausgegeben. Bei negativen Zahlen und fehlendem Vorzeichen-Spezifikator besetzt das Minus-Zeichen den Platz einer Stelle. Wenn ein Vorzeichen auszugeben ist. "schwimmt" dieses links neben der am weitesten links ausgegebenen Ziffer.
Z	Gleiche Wirkung wie "D", aber führende Nullen werden mit ausgegeben.
*	Gleiche Wirkung wie "D", aber führende Nullen werden durch Sternchen dargestellt.
E	Veranlaßt die Ausgabe des Exponenten der Zahl als Folge von 5 Zeichen, bestehend aus "E", dem Vorzeichen des Exponenten und dem 3-stelligen Exponenten selbst.
e	Gleiche Wirkung wie "E", aber nur 2-stelliger Exponent.
K	Veranlaßt die Ausgabe der Zahl in kompakter Form ohne führende und folgende Leerzeichen.

2. Spezifikatoren für die Vorzeichen

Diese Spezifikatoren steuern die Ausgabe des Vorzeichens. Denken Sie bitte daran, daß auch bei fehlendem Vorzeichenspezifikator negative Zahlen eine Stelle vor den Ziffern für das Vorzeichen benötigen.

Image-Spezifikator:	Wirkung auf das Ausgabe-Format:
S	Veranlaßt die Ausgabe eines führenden Vorzeichens.
M	Veranlaßt die Ausgabe einer Leerstelle bei positiven und die Ausgabe eines Minus-Zeichens bei negativen Zahlen.

3. Spezifikatoren für Dezimalzeichen und Zifferngruppierungen

Diese Spezifikatoren entscheiden, welches Symbol als Dezimalzeichen zu verwenden ist und ob eine Gruppierung der Stellen in 3-er-Gruppen vorzunehmen ist, und welches Symbol dafür benutzt wird.

Image-Spezifikator:	Wirkung auf das Ausgabe-Format:
.	Verwendung des Punktes als Dezimalzeichen. (anglo-amerikanische Schreibweise).
R	Verwendung des Kommas als Dezimalzeichen. (europäische Schreibweise).
C	Verwendet Kommata zur Einteilung der Zahl in Gruppen zu je 3 Ziffern (anglo-amerikanische Schreibweise).
P	Gleiche Gruppierung wie mit "C", jedoch Punkte an Stelle der Kommata (europäische Schreibweise).

Es wäre unrealistisch, hier Beispiele für alle möglichen Kombinationen der auf Zahlen anwendbaren Spezifikatoren zu erwarten. Die folgenden Beispiele zeigen nur einige der Möglichkeiten, wie man diese Spezifikatoren kombinieren kann und wie die formatierten Zahlen dann von einem Drucker dargestellt werden. Beispiele finden Sie auch im "Bedienungs- und Programmier-Handbuch" ("Formatierung in Ausdruck und Anzeige") ihres Rechners.

Programm-Anweisung:		Ausdruck:
OUTPUT 701 USING "ZZZZ.DD" ; 30.336		0030.34
OUTPUT 701 USING "4Z.2D" ; 30.336		0030.34
OUTPUT 701 USING "4Z.2D" ; -30.336		-030.34
OUTPUT 701 USING "3DC3DC3D" ; 1E6		1,000,000
OUTPUT 701 USING "3DC3DC3D" ; 1.2345E4		12,345
OUTPUT 701 USING "3DC3DC3D" ; 1.2E9	(Überlauf!)	, ,
OUTPUT 701 USING "SZ.DDD" ; .5		+0.500
OUTPUT 701 USING "M.DDD" ; .5		0.500
OUTPUT 701 USING "MD.DDD" ; .5		.500
OUTPUT 701 USING "Z.DDE" ; .00456		4.56E-003
OUTPUT 701 USING "Z.DDe" ; .00456		4.56E-03
OUTPUT 701 USING "Z.DDe" ; -.00456		-.45E-02
OUTPUT 701 USING "MZ.DDe" ; -.00456		-4.56E-03

Beachten Sie bei diesen Beispielen, daß die Form "ZZZZ" das gleiche bewirkt wie die Form "4Z". Entsprechendes gilt auch für die Spezifikatoren "D" und "*". Sie können die Zahl der gewünschten Stellen einfach dadurch festlegen, daß Sie diese Zahl vor den entsprechenden Spezifikator setzen. Die Wirkung von Klammern kann, je nach Anordnung, verschieden sein. Die Form "3D" bedeutet "Ausgabe einer Zahl mit 3 Stellen", die Form "3(D)"jedoch "Ausgabe von 3 Zahlen mit jeweils 1 Stelle.

Bei der Verwendung der Spezifikatoren für die Stellenzahl sollte man das Auslösen von überlaufmeldungen möglichst vermeiden, die immer dann auftreten, wenn für die sinngemäße Darstellung einer Zahl mehr Stellen nötig sind, als der Spezifikator erlaubt. Die Meldung unterbricht weder das Programm, noch verhindert sie die verstümmelte Ausgabe! Es ist verständlicherweise sehr schwierig, die Art der Verstümmelung exakt vorherzusagen und es besteht nur sehr geringe Wahrscheinlichkeit dafür, daß die ausgegebene Zahl der Zahl ähnlich sieht, die die Überlaufwarnung verursachte!

Formatierung von Zeichenketten (Strings)

Die Image-Spezifikatoren dieser Gruppe beeinflussen das Format der ausgegebenen Zeichenketten. Diese Spezifikatoren können mit denen zur Formatierung von Zahlen kombiniert werden, wodurch sich gute Möglichkeiten für Beschriftungen und Bezifferungen ergeben. Die vier nachfolgend beschriebenen Spezifikatoren müßten Ihnen von den PRINT USING- bzw. DISP USING-Anweisungen her vertraut sein.

Image-Spezifikator:	Wirkung auf das Ausgabe-Format:
A	Veranlaßt die Ausgabe eines Zeichens. Sind bereits alle Zeichen des betroffenen Strings ausgegeben, wird ein Leerzeichen ausgegeben.
"Literal"	Ein "Literal" ist eine Zeichenkette (String-Konstante), die in Anführungszeichen stehenden Klartext, aber auch mit der CHR$-Funktion aufgerufene Zeichen enthält. Diese festgelegte Zeichenfolge wird ausgegeben, wenn die Ausgabeliste dazu auffordert. Die Anführungszeichen, die vor und hinter dem Klartext stehen, werden nicht mit ausgegeben. Literale dienen meist zur Bezeichnung anderer Ausgabe-Werte. Literale können nicht direkt in OUTPUT-Anweisungen eingefügt werden, sie benötigen stets eine IMAGE-Anweisung.
X	Veranlaßt die Ausgabe eines Leerzeichens.
K	Veranlaßt die Ausgabe der Zeichenkette als 'Kompakt-Feld' ohne führende und folgende Leerzeichen.

Die folgenden Beispiele zeigen einige der vielen Möglichkeiten, wie diese Spezifikatoren anzuwenden sind und die daraus folgenden Ausdrucke. Weitere Beispiele finden Sie im "Bedienungs-und Programmier-Handbuch" ihres Rechners ("Formatierung in Ausdruck und Anzeige").

Programm-Anweisung:	Ausdruck:
`OUTPUT 701 USING "5A,A" ; "X","Y"`	`X Y`
`OUTPUT 701 USING "K,3X,K" ; "UNCLE","SAM"`	`UNCLE SAM`
`OUTPUT 701 USING "K,3X,K" ; 98.6,99.9`	`98.6 99.9`

```
10 IMAGE "TOTAL = ",3D,X,K
20 T=125 @ A$="CARS"
30 OUTPUT 701 USING 10 ; T,A$
```

TOTAL = 125 CARS

Beachten Sie, daß auch vor die Spezifikatoren "X" und "A" Zahlen gesetzt werden dürfen, so wie dies bei "D", "Z" und "*" zulässig war. Der Spezifikator "K" wirkt bei Zeichenketten und numerischen Daten gleich. Image-Spezifikatoren für Zahlen und Zeichenketten können in ein- und derselben Anweisung kombiniert werden. Wenn aber ein Literal benötigt wird, muß dieses in einer IMAGE-Anweisung stehen.

Formatierung von Binär-Ausgaben

Diese beiden Image-Spezifikatoren werden erst durch das I/O-ROM möglich und sind deshalb neu für Sie. Mit diesen Spezifikatoren kann erreicht werden, daß Daten nicht als Verkörperung von ASCII-Zeichen sondern als ein oder zwei Byte umfassende Bit-Muster ausgegeben werden. Die I/O-ROM-Abschnitte 5 und 6 dieses Buches erklären die Einzelheiten der binären Darstellungsart. Wenn Ihnen der Umgang mit Binär-Zahlen ungewohnt sein sollte, wird geraten, diese Abschnitte durchzuarbeiten, bevor Sie die Binär-Spezifikatoren praktisch anwenden.

Unter Benutzung dieser Spezifikatoren können nur Zahlen in einem bestimmten Bereich ausgegeben werden. Wenn es sich dabei nicht um ganzzahlige Werte handelt, werden diese vor der Ausgabe auf den nächsten ganzzahligen Wert gerundet, und erst dann in binärer Darstellung ausgegeben.

Image-Spezifikator:	Wirkung auf das Ausgabe-Format:
B	Gibt einen Wert als einzelnes Byte (also 8 Bits) aus. Der Wert muß im Bereich von 0 bis 255 liegen. Werte außerhalb dieser Grenzen werden durch die Funktion MOD(256) reduziert und dann ausgegeben.
W	Gibt einen Wert in Form von zwei Bytes (also 16 Bits) aus, die als ein 16-Bit-Wort behandelt werden. Das höherwertige Byte wird zuerst ausgegeben, gefolgt von dem geringerwertigen. Der auszugebende Wert muß im Bereich von -32768 bis 32767 liegen. Negative Werte werden als 2-er-Komplement mit 16 Bits ausgegeben. Bei Unter- bzw. Überschreitung der obigen Grenzwerte werden eben diese Grenzwerte als Bitmuster ausgegeben.

Programm-Anweisung:	Bit-Muster:
`OUTPUT 3 USING "B" ; 127`	01111111
`OUTPUT 3 USING "B" ; 3`	00000011
`OUTPUT 3 USING "W" ; 3`	00000000 00000011
`OUTPUT 3 USING "W" ; -1`	11111111 11111111

Beachten Sie bitte, daß die Binär-Spezifikatoren die als Abschluß übliche EOL-Sequenz nicht automatisch unterdrücken! In den gezeigten Beispielen folgt also auf das Bitmuster jeweils noch das Wagenrücklauf- und das Zeilenvorschub-Symbol (CR und LF).

Auf das BCD-Interface (HP 82941A) können die Spezifikatoren "B" und "W" nicht angewendet werden! Schauen Sie deshalb wegen dieser speziellen Formatierung in das zum BCD-Interface gehörende Handbuch!

Beeinflussung des Zeilen-End-Signals (EOL,"End Of Line")

Diese beiden Image-Spezifikatoren steuern das "Zeilen-End-Signal" (von nun an als "EOL-Sequenz" bezeichnet), das aus einem String besteht, der dem letzten Posten der Datenliste nachfolgt und/oder einer anderen Maßnahme, mit der das letzte Byte einer Datenliste besonders markiert wird. Welche(s) Zeichen oder welches Signal im Einzelnen dafür benutzt wird, hängt vom betroffenen Interface und dem Programm ab. Für HP-IB- und HP-IL-Interface sind diese Dinge im IB- bzw. IL-Teil dieses Handbuches ausführlich beschrieben. Wenn das Programm die EOL-Sequenz nicht verändert, besteht diese aus zwei Zeichen: Wagenrücklauf-Symbol (CR) und Zeilenvorschub-Symbol (LF).

Die nachfolgend beschriebenen Spezifikatoren verändern nicht die Art der EOL-Sequenz, sie entscheiden nur darüber, ob sie ausgegeben wird oder nicht.

Image-Spezifikator:	Wirkung auf die Ausgabe:
/	Veranlaßt die Ausgabe einer EOL-Sequenz. Oft benutzt, um ganze Leerzeilen zu erzeugen.
#	Verhindert die Ausgabe der letzten EOL-Sequenz. Dieser Spezifikator wird häufig zusammen mit den Binär-Spezifikatoren benutzt, um zu verhindern, daß die angesprochene Peripherie zusätzlich die EOL-Sequenz erhält und auch diese noch als Binär-Daten wertet.

Das "/" kann innerhalb der Spezifikatoren-Liste mehrmals vorkommen, an beliebiger Stelle stehen und auch mit vorangestellter Zahl auftreten, wenn die EOL-Sequenz mehrmals unmittelbar nacheinander ausgegeben werden soll. Das "#" darf dagegen in der Formatierung nur einmal enthalten sein, und zwar an erster Stelle! Denken Sie bitte auch daran, daß das "#" nur die EOL-Sequenz am Ende der Ausgabeliste unterdrückt, nicht aber die zusätzlich durch den "/"-Spezifikator innerhalb der Liste angeordneten EOL-Ausgaben.

Der "#"-Spezifikator wird gern zur isolierten Ausgabe eines Bytes benutzt:

```
OUTPUT 6 USING "#,B" ; X
```

Diese Anweisung gibt die binäre Darstellung des Wertes von X ohne EOL-Sequenz und ohne irgendwelche anderen möglicherweise unerwünschten Bitmuster aus.

Eine typische Anwendung des "/"-Spezifikators zeigt die folgende Anweisung:

```
OUTPUT 701 USING "K,4/,K" ; A$,B$
```

Wenn das angesprochene Gerät ein Drucker ist, wird A$ gedruckt, gefolgt von vier Leerzeilen, erst danach wird B$ gedruckt. Wenn A$="HI" und B$="JOE" ist, sieht die ausgegebene Zeichenfolge so aus:

H	I	CR	LF	CR	LF	CR	LF	CR	LF	J	O	E	CR	LF

Der für das I/O-ROM typische "#"-Spezifikator zeigt seine Wirkung auch beim Druck wiederholter PRINT-Anweisungen in einer einzigen Zeile:

```
10 for I=1 to 3
20 PRINT USING "#,8A" ; "Hoppla,"
30 NEXT I
40 PRINT USING "5A" ; "Hopp!"
50 END

Hoppla, Hoppla, Hoppla, Hopp!
```

Formatierte Aufnahme von Daten

Die Kombination der ENTER-Anweisung mit den Image-Spezifikatoren räumt Ihnen auf zwei Gebieten großen Einfluß ein:

1. Sie können dem Rechner mitteilen, wie die eingehenden Daten aussehen werden und was mit ihnen geschehen soll.
2. Sie können genau festlegen, wann die Aufnahme für die einzelne(n) Variable(n) abzuschließen ist und wodurch die ENTER-Anweisung selbst beendet werden soll.

Die folgenden Ausführungen beschäftigen sich zuerst mit der Formatierung eintreffender Daten und dann mit den Abschluß-Spezifikatoren.

Der Rechner formatiert die eintreffenden Daten gemäß der ENTER-Anweisung, die entweder selbst eine Formatierungsangabe enthält oder auf die passende IMAGE-Anweisung verweist. Hierfür gibt es drei Formen, wobei die unter 3. gezeigte Label-Technik für den HP-83/85 nicht anwendbar ist:

1.
```
10 IMAGE<Enter-Format>
20 ENTER<Quelle>USING 10;<Eingabe-Liste>
```
2.
```
ENTER<Quelle>USING<Enter-Format>;<Eingabe-Liste>
```
3.
```
10 LBL: IMAGAGE<Enter-Format>
20 ENTER<Quelle>USING LBL ;<Eingabe-Liste>
```

Die obige Darstellung zeigt die allgemeine Form der ENTER USING-Anweisungen. Hier nun einige konkrete Beispiele (wobei die Zeilen 20 und 70 wegen des Aufrufs durch Label beim HP-83/85 nicht möglich sind!):

```
10 IMAGE 2(A),K
20 FMT: IMAGE 5D,2X,3De
.
.
60 ENTER 4 USING 10 ; A$,B$,X
70 ENTER 711 USING FMT ; I,J
80 ENTER 9 USING "#,B" ; A(1),A(2)
90 ENTER 52 USING "%,8A,/,K" ; Q$,R$
100 ENTER 712 USING I$ ; N$,A
```

Die allgemeine Form benutzt fast die gleichen Symbole, die uns schon von der Beschreibung der OUTPUT-Anweisungen bekannt sind: In der allgemeinen Form steht "Quelle" für "Geräteadresse". "Enter-Format" steht an Stelle einer korrekten Zusammenstellung von Image-Spezifikatoren. Die "Eingabe-Liste" enthält eine Aufzählung von Variablen, für die Daten erwartet werden. Wie bei der einfachen ENTER-Anweisung darf diese Liste nur Variable für numerische Daten oder für Strings enthalten. Es ist widersinnig, die "Eingabe" einer "Konstanten" zuzuordnen!

Formatierungs-Methoden

Die hier behandelten Image-Spezifikatoren sagen dem Rechner, was mit den eintreffenden Daten geschehen soll. Es gibt da folgende Möglichkeiten:

1. Die Daten sollen einer numerischen Variablen zugeordnet werden.
2. Die Daten sollen einer String-Variablen zugeordnet werden.
3. Die Daten sollen als Binär-Zahl angesehen werden.
4. Eine bestimmte Zahl von Zeichen soll nicht beachtet (also übersprungen) werden.

Formatierung zu numerischen Werten

Die Image-Spezifikatoren dieser Gruppe nehmen (mit Ausnahme von "K") nur numerische Zeichen an, also Ziffern, Vorzeichen, Exponenten sowie Dezimal- und Ziffern-Gruppierungs-Zeichen. Mit den angenommenen Zeichen werden numerische Variable besetzt.

Image-Spezifikator:	Wirkung auf das Eingabe-Format:
D Z * . S M	Diese sechs Spezifikatoren wirken alle gleich: Sie veranlassen den Rechner, ein Zeichen anzunehmen und es zum Aufbau eines Zahlenwertes zu verwenden. Die eingehenden Zeichen brauchen dabei nicht dem gewünschten Format zu entsprechen,es reicht, wenn die richtige Anzahl von Zeichen eintrifft. Die sechs verschiedenen Spezifikatoren sollen ermöglichen, daß das gewünschte Format in der Anweisung erkennbar ist und für OUTPUT- und ENTER-Anweisungen, wenn gewünscht, gemeinsame IMAGE-Anweisungen gelten können.
E	Veranlaßt den Rechner, 5 Zeichen anzunehmen, aus denen eine Zahl gebildet werden kann. Es ist möglich, aber nicht notwendig, daß diese Zahl ein Exponent ist.
e	Gleiche Wirkung wie "E", es werden aber nur 4 Zeichen angenommen und verarbeitet.
C	Veranlaßt den Rechner, ein Zeichen anzunehmen und zum Aufbau eines Zahlenwertes zu verwenden. Wenn "C" in der Formatierung irgendwo auftritt, werden während der Datenaufnahme für diese Variable alle Kommata ignoriert.
K	Veranlaßt den Rechner, numerische Daten oder eine Zeichenkette aufzunehmen und damit die momentan bearbeitete Variable zu besetzen, die als "Frei-Feld" (erklärt im I/O-ROM-Abschnitt 2) formatiert wird.
R P	Diese beiden Spezifikatoren wurden bei OUTPUT-Anweisungen benutzt, um Zahlen in europäischer Schreibweise ausgeben zu können. Diese Spezifikatoren sind in ENTER-Anweisungen nicht erlaubt. Wenn Zahlen in europäischer Schreibweise aufzunehmen sind, ist die CONVERT-Anweisung zu benutzen, die auf Seite ROM-28 erklärt wird.

Wenn die mit dem Spezifikator "K" aufgenommenen Daten numerischen Variablen zugeordnet werden sollen, muß beachtet werden, daß dieser Spezifikator numerische Daten nur im Zahlenbereich des Rechners ohne Fehlermeldung annehmen kann: Von den beliebig vielen Stellen der Mantisse werden nur die ersten 12 (ohne Rundung!) angenommen, die Stellung des Dezimalpunktes unter Wahrung der Größenordnung auf wissenschaftliche Schreibweise umgestellt und ein eventuell eingelesener Exponent entsprechend korrigiert. Ein Exponent unter -499 oder über +499 führt zur Fehlermeldung. Die Zuordnung endet beim Erkennen des ersten nicht-numerischen Zeichens bzw. mit dem Lesen von "LF".

Mit "K" lassen sich aber auch String-Variable besetzen! Diese Anwendung von "K" wird auf der nächsten Seite behandelt!

Formatierung zu Zeichenketten (Strings)

Die Image-Spezifikatoren dieser Gruppe ordnen die eintreffenden Zeichen den angegebenen String-Variablen zu.

Image-Spezifikator:	Wirkung auf das Eingabe-Format:
A	Veranlaßt den Rechner, ein alphanumerisches Zeichen aufzunehmen und es der momentan bearbeiteten String-Variablen zuzuordnen.
K	Veranlaßt den Rechner, eine Zeichenkette oder numerische Daten aufzunehmen und damit die momentan bearbeitete Variable im 'Frei-Feld'-Format zu besetzen (erklärt im I/O-ROM-Abschnitt 2).

Im allgemeinen lassen sich nur Strings bekannter Länge mit "nA" einwandfrei zuordnen, weil erst nach dem Eintreffen von "n" Zeichen die Zuordnung beendet wird. "CR" und/oder "LF" brechen diesen Vorgang nicht vorzeitig ab, sondern werden in der String-Variablen mit abgelegt!

Bei unbekannter Zahl der Zeichen ist "K" besser geeignet, weil dadurch bei Auftreten von "LF" die Zuordnung beendet wird, auch wenn die String-Variable noch nicht voll besetzt ist.

Mit "K" lassen sich aber auch numerische Variable besetzen. Diese Anwendung von "K" wurde auf der vorigen Seite behandelt!

Es folgen jetzt einige Beispiele für passende Formatierungen. Dazu nehmen wir an, daß der Rechner die nachstehende Zeichenfolge empfängt:

1	2	3	4	H	A	L	L	O	CR	LF

Jede der folgenden ENTER-Anweisungen nimmt diese Zeichen einwandfrei auf und besetzt damit die numerische und die String-Variable:

```
ENTER 720 USING "4D,5A" ; X,Y$
ENTER 720 USING "Z.DD,5A" ; X,Y$
ENTER 720 USING "e,K" ; X,Y$
```

Beachten Sie bitte, daß jedes numerische Format, das Platz für vier Stellen bietet, die Zeichen "1234" korrekt der numerischen Variablen zuweist.

Wenn wir jetzt eine etwas andere Zeichenfolge aufzunehmen hätten

1	,	2	3	4	H	A	L	L	O	CR	LF

dann muss eine passende ENTER-Anweisung als Image-Spezifikator unbedingt ein "C" enthalten, weil die Zeichenfolge sonst in "1" und "234HALLO" aufgeteilt wird:

```
ENTER 720 USING "C4D,K" ; X,Y$  oder
ENTER 720 USING "DDDDC,5A" ; X,Y$
```

Beachten Sie bitte, daß das "C" nicht an der Stelle stehen muß, an der das Komma erwartet wird. Das Komma wird aber immer als Zeichen gezählt und muß deshalb in der Formatierung mit berücksichtigt sein.

Formatierung zu Binär-Daten

Diese beiden Image-Spezifikatoren formatieren die aufgenommenen Daten zu Binär-Zahlen.

Image-Spezifikator:	Wirkung auf die Eingabe:
B	Veranlaßt den Rechner zur Annahme eines Bytes, dessen Dezimalwert (0 bis 255) einer Variablen zugeordnet wird.
W	Veranlaßt den Rechner zur Annahme von zwei Bytes, die zum Aufbau eines 16-Bit-Wortes benutzt werden. Das Wort wird als Binär-Zahl (mit 2-er-Komplement) gewertet und einer numerischen Variablen zugeordnet. Das zuerst empfangene Byte hat die höhere Wertigkeit.

Überspringen unerwünschter Zeichen

Diese beiden Spezifikatoren werden benutzt, um einige der eingehenden Zeichen zu überspringen, weil diese z.B. unerwünscht sind.

Image-Spezifikator:	Wirkung auf die Eingabe:
X	Veranlaßt den Rechner, ein Zeichen zu überspringen.
/	Veranlaßt den Rechner, zum nächsten Zeilenvorschub-Symbol zu springen. Bis zum Erreichen dieses Spezifikators ordnet der Rechner die einlaufenden Daten den Variablen zu. Wenn "/" erreicht ist, "übersieht" der Rechner alle Zeichen, bis er wieder ein Zeilenvorschub-Symbol empfängt.

Der "X"-Spezifikator kann nur benutzt werden, wenn man über die Struktur der eingehenden Daten genau informiert ist. Er ist dann aber sehr effektiv und bequem. Wenn z.B. ein Text von einem anderen Rechner eintrifft, der die Zeichenketten mit Zeilennummern ausgibt, die nun aber nicht mehr erwünscht sind, können diese Zeilennummern unterdrückt werden, sofern diese immer die gleiche, bekannte Stellenzahl haben. Die folgende Anweisung geht davon aus, daß stets die ersten 8 Zeichen des Strings die Zeilennummer enthalten:

```
ENTER 720 USING"8X,K";A$
```

Der "/"-Spezifikator wird benutzt, um ein Zeilenvorschub-Symbol abzuwarten, bevor die nächste Variable besetzt wird. Um die Wirkung dieses Spezifikators zu erkennen, soll angenommen werden, daß folgende Daten empfangen wurden:

1	2	3	H	I	LF	B	Y	E	CR	LF

Bei Benutzung der Anweisung

```
ENTER 720 USING "3D,K" ; X,Y$
```

wird X mit dem Wert 123 und Y$ mit dem Text "HI" besetzt. Auch von der Anweisung

```
ENTER 720 USING "3,/,K" ; X,Y$
```

wird X wieder mit dem Wert 123 besetzt, Y$ nimmt nun aber "BYE" als Text auf. Der "/"-Spezifikator veranlaßt nämlich den Rechner, nach Aufnahme des numerischen Wertes alle Zeichen zu überspringen, bis er wieder ein Zeilenvorschub-Symbol findet. Erst dann beginnt die Besetzung von Y$ mit den Zeichen, die auf das Zeilenvorschub-Symbol folgen. Ohne den "/"-Spezifikator hätte dagegen die Besetzung von Y$ unmittelbar nach Abarbeitung des "3D"-Feldes begonnen.

Daten-Annahme ohne Zeilenvorschub-Symbol

Die ENTER-Anweisung benötigt für gewöhnlich das Zeilenvorschub-Symbol (LF) als Abschluß der eintreffenden Daten, damit der Rechner zur nächsten Programmzeile weitergehen kann. Wenn dieses Symbol fehlt, tritt das "Aufhängen" des Rechners ein, der auf dieses Zeichen wartet. Wenn bekannt ist, daß die einlaufenden Daten nicht mit einem Zeilenvorschub-Symbol enden, können Sie die ENTER-Anweisung mit einem besonderen Spezifikator zu einwandfreiem Arbeiten bringen.

Image-Spezifikator:	Wirkung auf die Eingabe:
#	Macht das Zeilenvorschub-Symbol zum Abschluß einer ENTER-Anweisung unnötig. Mit diesem Spezifikator gilt die Anweisung als erfüllt, sobald die letzte Variable besetzt wurde.

Wenn der "#"-Spezifikator diesen Effekt haben soll, muß er an erster Stelle in der IMAGE-Liste stehen, wie in diesen Beispielen gezeigt wird:

```
ENTER 3 USING "#,K" ; A$
ENTER 720 USING "#,4D,6D" ; X,Y
```

Das erste Beispiel bewirkt die Belegung der String-Variablen im Frei-Feld-Format mit eintreffenden Zeichen, ohne daß ein Zeilenvorschub-Symbol nachfolgen muß. Diese Anweisung ist abgearbeitet, wenn die String-Variable voll besetzt ist. Das zweite Beispiel zeigt die formatierte Belegung von numerischen Variablen mit numerischen Daten. Diese Anweisung endet nach der Aufnahme von 4 bzw. 6 Zeichen.

Weitere Anwendungsmöglichkeiten der Abschlußzeichen

Die bisher beschriebenen Image-Spezifiktoren, lösen fast alle Formatierungs-Probleme. Nur in wenigen Spezialfällen braucht man noch mehr Flexibilität. Diese Fälle treten meist im Zusammenhang mit der EOI-Leitung vom HP-IB oder den END-Bytes beim HP-IL auf. Die folgenden Erläuterungen sind nur wichtig, wenn Sie diese Möglichkeiten benutzen oder mit ungewöhnlichen Zeilenvorschub-Problemen Ärger haben; nur dann müßten Sie diesen Abschnitt aufmerksam lesen, der wohl der schwierigste dieser Grundlagen ist. EOI und END wirken auf sehr ähnliche Weise.

Feld- und Zeilen-Abschlüsse

Zweck der ENTER-Anweisung ist es, "Nachrichten" zu lesen. Für den Programmierer sind "Nachrichten" eine Folge von Einzeldaten. Für den Rechner sind "Nachrichten" Daten-Felder, die durch ein vereinbartes Zeichen gegeneinander abgegrenzt sind. Das sind die "Feld-Abschlüsse". Weil eine ENTER-Anweisung erst als ausgeführt gilt, wenn die Eingabeliste erfüllt und der Abschluß des letzten Daten-Feldes erkannt wurde, wird dieser Abschluß wegen seiner besonderen Aufgabe "Anweisungs-Abschluß" genannt. (Etwas anders sieht die Sache aus, wenn "Buffer" benutzt werden. Darüber steht im I/O-ROM-Abschnitt 8 mehr!) Wenn nichts anderes festgelegt wird, ist das Zeilenvorschub-Symbol (LF) Universal-Zeichen für die Abschlüsse. Auch die Kombination Wagenrücklauf- und Zeilenvorschub-Symbol (CR,LF) ist zulässig, weil das I/O-ROM das Wagenrücklauf-Symbol ignoriert, wenn ein Zeilen-Vorschub-Symbol unmittelbar folgt.

Eintreffende Nachrichten bestehen meist aus mehreren Feldern. Ein Feld ist dabei eine Gruppe von Zeichen, die einer einzigen Variablen der Eingabeliste zuzuordnen sind. Eine ENTER-Anweisung, die z.B. zur Aufnahme des Namens und des Alters dienen soll, könnte so aussehen:

```
ENTER 720 ; N$,A
```

Diese Anweisung liest die Nachrichten Namen und Alter. Das sind zwei Felder: ein String-Feld (Name), und ein numerisches Feld (Alter). Eine korrekte ENTER-Anweisung wird das String-Feld nach N$ und das numerische Feld nach A leiten. Meist ist es unnötig, besondere Abschluß-Bedingungen zu vereinbaren, weil die vorhandenen Abschluß-Bedingungen des Systems für Felder und Anweisungen in den meisten Fällen genügen:

Ein Stringfeld wird beendet, wenn die String-Variable gefüllt ist (Prüfen Sie die DIM-Anweisungen!) oder die vom Image-Spezifikator erwartete Zeichenzahl oder ein Zeilenvorschub-Symbol eingetroffen ist.

Ein numerisches Feld endet mit dem ersten nicht-numerischen Zeichen (Ausnahme: Leerzeichen) oder mit dem Erreichen der von der IMAGE-Anweisung erwarteten Stellenzahl.

Eine ENTER-Anweisung wird beendet, wenn nach Besetzen aller Variablen das Zeilenvorschub-Symbol oder eine Kombination aus Wagenrücklauf- und Zeilenvorschub-Symbol eingetroffen ist. Das kann der Abschluß sein, der auch das letzte Feld beendet.

Unter diesen Bedingungen kann die obige ENTER-Anweisung Namen und Alter nur in zwei Fällen sauber trennen: Entweder muß hinter dem Stringfeld ein "LF" eingefügt werden oder der String mit dem Namen muß immer die gleiche Länge haben, für die N$ entsprechend zu dimensionieren ist.

Spezifikatoren für Abschluß-Bedingungen

Wenn in einigen Fällen mit den normalen Feld- und Anweisungs-Abschlußbedingungen das gewünschte Verhalten nicht zu erreichen ist, können die Spezifikatoren für Abschluß-Bedingungen hilfreich sein. Die nachstehend beschriebenen Spezifikatoren können sowohl für Feld- als auch für Anweisungs-Abschlüsse benutzt werden. Der Feld-Abschluß beendet die Zuordnung der aufgenommenen Zeichen zu einer Variablen, während der Anweisungs-Abschluß nach Besetzung der letzten Variablen die Beendigung der ENTER-Anweisung bewirkt.

Image-Spezifikator:	als Feld-Abschluß:	als Anweisungs-Abschluß:
#	Für Frei-Feld-Eingabe ist Zeilenvorschub-Symbol unwirksam. Zeilenvorschub-Symbole werden in die Variable mit aufgenommen.	Zeilenvorschub-Symbol als Anweisungs-Abschluß unnötig. Anweisung gilt als ausgeführt, wenn auch die letzte Variable besetzt wurde.
%	Auch EOI kann Abschluß sein.	EOI oder Zeilenvorschub-Symbol wirken auch einzeln als Abschluß.
#% oder %#	Für Frei-Feld-Eingabe ist Zeilenvorschub-Symbol unwirksam, Zeilenvorschub-Symbole werden in die Variable mit aufgenommen, nur EOI ist als Abschluß wirksam!	Lediglich EOI wirkt als Abschluß. Das Zeilenvorschub-Symbol ist unwirksam!

Ob diese Image-Spezifikatoren als Feld- oder Anweisungs-Abschlüsse wirksam sind, hängt von der Stelle ab, an der sie innerhalb der Image-Liste stehen. Schauen wir uns das folgende Beispiel an:

```
ENTER 720 USING "%,%K" ; X
```

Nur wenn einer der obigen Spezifikatoren allein an erster Stelle steht, so wie hier das erste "%", dann wirkt er als Anweisungs-Abschluß; verbunden mit anderen Spezifikatoren, wie hier bei"%K", ist er nur als Feld-Abschluß wirksam. Das gilt für "#", "%" und "#%" in gleicher Weise.

Weil die normalen Abschlußbedingungen immer wirksam bleiben, verändern diese Spezial-Abschlüsse die Systemfunktionen nur in wenigen Fällen. Wegen ihres tiefgreifenden Einflusses wollen wir uns aber die Wirkung in jedem Fall einzeln ansehen:

Aufnahme von Zeilenvorschub-Symbolen in Zeichenketten.

Der Spezifikator "#K" veranlaßt den Rechner, alle ankommenden Zeichen (auch die Zeilenvorschub-Symbole) in einen String zu setzen, bis dieser gefüllt ist. Das erste danach eintreffende Zeilenvorschub-Symbol bewirkt dann den Abschluß der Anweisung. Wenn Sie also den Abschluß einer Anweisung wünschen, sobald die String-Variable gefüllt ist, und Sie nicht erst den Anweisungs-Abschluß abwarten wollen, dann ist "#,#K" die richtige Form der speziellen Abschluß-Bedingung.

EOI (bzw. END-Byte) als Abschlußbedingung für String-Eingaben

Die Kombination "%K" erlaubt dem Rechner, einen im Frei-Feld-Format aufgenommenen String durch das EOI-Signal abzuschließen. Es gibt nämlich Geräte, die EOI als Zeilen-End-Signal verwenden, weil ihnen keine anderen Signale, wie z.B. das Zeilenvorschub-Symbol, dafür zur Verfügung stehen. In diesen Fällen ist "%,%K" die passende Kombination. Damit kann ein EOI-Signal eine ENTER-Anweisung beenden. Wenn Sie auch die Zeilenvorschub-Symbole im String wiederfinden wollen, läßt sich mit "#%K" auch die Anweisung durch EOI abschließen. Auch eine Erweiterung auf "%,#%K" ist denkbar, wenn kein Zeilenvorschub-Symbol zum Abschluß zu erwarten ist. Eine noch längere Form, "#%,#%K" gestattet dem EOI nicht nur den Abschluß der ENTER-Anweisung, sondern macht ihn nur von diesem Signal abhängig. Fest-Feld-Eingaben können auf ein erwartetes EOI geprüft werden. Die Form "%7A" z.B. setzt sieben Zeichen in den String und erwartet ein EOI-Signal zusammen mit dem siebenten Zeichen. Behalten Sie im Gedächtnis, daß man aus den Spezifikatoren sehr viele gültige Kombinationen bilden kann. Die eben gezeigten sind von der einfachsten Art!

EOI als Abschlußbedingung bei numerischen und binären Eingaben

Die Kombination "%K" erlaubt dem Rechner, einen im Frei-Feld-Format aufgenommenen numerischen Wert durch das EOI-Signal abzuschließen. Wie wir schon im vorigen Absatz gesehen haben, kann auch die Form "%,%K" notwendig sein, wenn das EOI-Signal auch als Abschluß für die ENTER-Anweisung dienen soll. Fest-Feld-Eingaben können auch hier auf ein erwartetes EOI-Signal geprüft werden. Die Form "%7D" baut aus sieben numerischen Zeichen eine Zahl auf und erwartet EOI zusammen mit dem siebenten Zeichen. Binäre Felder werden ähnlich bearbeitet: Die Form "%W" nimmt zwei Bytes auf, macht daraus eine 16-Bit-Ganzzahl und erwartet EOI beim zweiten Byte.

Immer gibt es eine Ausnahme

Mit den Spezifikatoren für die Abschluß-Bedingungen lassen sich aber nicht alle Probleme lösen! Schauen wir uns nochmals das alte Beispiel mit dem Namensfeld und der Altersangabe an: Wir müssen voraussetzen, daß die Namen in ihrer Länge variieren und von der Altersangabe durch ein Komma getrennt sind. Wenn das Alter vorn stände, wäre alles problemlos, weil das Komma die Aufnahme des numerischen Wertes abschließt. Weil aber der Name zuerst kommt, ist die Lösung etwas komplizierter:

Sie könnten mit der CONVERT-Anweisung (am Schluß des Abschnittes erklärt!) jedes Komma in ein Zeilenvorschub-Symbol wandeln und dadurch die Eingaben trennen. (Das betrifft dann aber alle Kommata, was auch nicht so gut ist!) Es ist dann besser, beide Felder erst in einen gemeinsamen String einzulesen, um mit der POS-Funktion darin die Stellung des Kommas zu ermitteln und dann die Angaben zu trennen. Damit wird wohl deutlich, wie wichtig es für den Programmierer ist, die Art der einlaufenden Daten möglichst genau zu kennen. Bei einigen Problemen sind die Spezifikatoren für die Abschluß-Bedingungen "die Lösung", bei anderen Problemen sind sie wirkungslos! Der Programmierer muß aber immer eine Lösung finden!

Ein ehrliches Wort über Formatierungen.

Die Wahl der richtigen Formatierung entscheidet oft über Erfolg oder Mißerfolg eines Programms. Wenn man an die große Zahl der anzuschließenden Peripherie-Geräte denkt und die nahezu unbegrenzten Formatierungsmöglichkeiten berücksichtigt, dann ist verständlich, warum es so schwer ist, auf Anhieb eine optimale Formatierung zu finden. Auch erfahrene Programmierer erleben da bei ihren Versuchen Fehlschläge, bis sie die perfekte Kombination der Spezifikatoren gefunden haben.

Es ist zwar nicht neu, aber immer noch wahr, wenn gesagt wird: "Du kannst keinen Rechner dazu zwingen, etwas richtig zu tun, von dem Du nicht selbst weißt, wie es getan werden muß!" Das trifft besonders auf die formatierten I/O-Operationen zu. Wenn Sie nicht genau wissen, welche Zeichenfolgen ausgegeben werden sollen oder was in einer eingehenden Folge steht, dann ist es sehr unwahrscheinlich, daß Sie die richtige Formatierung wählen.

Das Ausgeben einer korrekten Zeichenfolge ist nur eine Frage der Überlegung: Sie wissen, welche Daten Ihr Programm erzeugt und in welcher Form die Daten ausgegeben werden sollen. Dabei muß man sich nur vor einem Überlauf hüten, den ein falsches Format auslösen kann.

Wie kann aber ein Programmierer die genaue Struktur einlaufender Daten bestimmen, ohne sie im Rechner analysieren zu können? Wenn es Image-Spezifikatoren nur für Strings und numerische Eingaben gäbe, wäre das ein äußerst schwieriges Problem. Es gibt aber glücklicherweise einen Weg, um auch völlig unbekannte Zeichenfolgen zu untersuchen: Jede eintreffende Zeichenfolge kann einschließlich der Abschluß-Zeichen mit der Kombination "#,B" aufgenommen werden. Der Dezimalwert jedes einzelnen Bytes kann angezeigt oder ausgedruckt werden. Das ist wahrlich nicht die bequemste Art der Datenuntersuchung, Sie können aber mit einer ASCII-Tabelle oder der CHR$-Funktion jede eingegangene Zeichenfolge auch darstellen. Kennt man aber erst die Art der Daten, ist die Wahl des passenden Spezifikators schon wesentlich leichter. Das folgende Programm ist dafür typisch:

```
100 ! Programm zur Untersuchung eingehender Daten
110 S1=3 ! Interface Select Code
120 ! Festlegen einer Abschluß-Bedingung
130 SET TIMEOUT S1;3000
140 ON TIMEOUT S1 GOTO 230
150 !
160 I=1 ! Zaehler initialisieren
170 ! Input: 1 Byte, Bildschirm-Analyse
180 ENTER S1 USING "#,B" ; X
190 DISP "BYTE";I;TAB(11);"WERT =";X;TAB(24);"Zeichen= ";CHR$(X)
200 I=I+1 ! Byte-Zaehler
210 GOTO 180
220 !
230 DISP "Daten-Eingabe zu Ende"
240 RESET S1 ! Stoppt die I/O-Operation
250 END
```

Konvertieren von I/O-Daten

Die letzte Art des "Formatierens" besteht darin, die ein- oder ausgehenden Daten einer "Übersetzung" zu unterziehen. Ein bereits früher erwähntes Beispiel waren eingehende numerische Daten in europäischer Schreibweise (Komma als Dezimalzeichen, Punkt(e) zur Gruppierung). Kein Spezifikator kann solche Daten direkt annehmen. Punkt(e) und Komma(ta) müssen in für den Rechner passende Zeichen geändert werden. Das Hilfsmittel dafür ist die CONVERT-Anweisung, die in allgemeiner Form so aussieht:

CONVERT<Richtung><Auswahl-Code><Methode>;<Stringvariable>

Die Parameter bedeuten:

<Richtung>
zeigt an, ob die Übersetzung für eingehende Daten (dann "IN" einsetzen) oder für ausgehende Daten (dann "OUT" einsetzen) gelten soll.

<Auswahl-Code>
legt fest, welches Interface die Übersetzung benutzt. Beachten Sie, daß sich die CONVERT-Anweisung stets auf alle Geräte bezieht, die mit diesem Interface verbunden sind. Geräte-Adressen sind hier nicht zulässig. Der Parameter muß also ganzzahlige Werte zwischen 3 und 10 haben.

<Methode>
enthält Angaben über die Art der Übersetzung. Die Übersetzungs-Tabelle ist in einer Stringvariablen untergebracht. Wenn "PAIRS" angegeben ist, enthält dieser nur Zeichen-Paare. Von diesen Paaren wird immer das jeweils zweite Zeichen verwendet, wenn das jeweils erste Zeichen in ein- oder ausgehenden Daten auftritt. Diese Methode ist gut, wenn nur wenige Zeichen umgesetzt werden müssen.

Wenn "INDEX" eingesetzt wird, muß der String eine komplette Übersetzungs-Tabelle enthalten. Der numerische Wert jedes ein- oder ausgehenden Zeichens wird dann als Indexziffer für diese Tabelle benutzt. Das erste Element des Übersetzungs-Strings entspricht dann dem Zeichen mit dem numerischen Wert "0". Wenn der numerische Wert des zu übersetzenden Zeichens größer ist als die Zahl der zur Tabelle gehörenden Zeichen, dann wird keine Übersetzung ausgeführt. Diese Methode ist bequem, wenn sehr viele Zeichen umgesetzt werden müssen.

<Stringvariable>
legt fest, welche Stringvariable die Übersetzungsangaben enthält. Hier muß eine Stringvariable stehen, eine Stringkonstante (Literal) ist nicht zulässig!

Die Wirkung der CONVERT-Anweisung wird durch Beispiele deutlicher. Als erstes wieder das Problem der europäischen Schreibweise von Zahlen. Hier müssen eingehende Daten umgesetzt werden: Das Komma muß durch den Dezimal-Punkt und der Gruppierungspunkt muß durch ein Leerzeichen ersetzt werden. Für ein Interface mit dem Auswahl-Code 7 kann das mit den folgenden Anweisungen erreicht werden:

```
A$=",.. "
CONVERT IN 7 PAIRS ; A$
```

Die Umsetzung hat folgende Wirkung:

```
Daten vor der Umsetzung:      12.345,6
Daten nach der Umsetzung:     12 345.6
```

Weil beim Frei-Feld-Format Leerzeichen innerhalb einer Zahlenfolge ignoriert werden und der Punkt als Dezimalzeichen erkannt wird, brauchen Sie nicht einmal ein ENTER-Format anzugeben, damit der Rechner die Daten als Zahl annimmt. Es ist aber wichtig, daran zu denken, daß diese CONVERT-Anweisung alle Kommata in Punkte und alle Punkte in Leerzeichen wandelt, gleichgültig ob diese Zeichen sich in einer europäisch geschriebenen Zahl oder in einem Textblock befinden! Das kann sehr unerwünschte Effekte haben. Deshalb ist es gut zu wissen, wie man die "Übersetzung" abschalten kann, wenn man sie nicht mehr braucht. In diesem Beispiel geht das so:

```
CONVERT IN 7
```

Man gibt also nur den Interface-Auswahl-Code und die Richtung an, ohne "PAIRS" oder "INDEX" oder andere Parameter und hat damit die vorher wirksame Übersetzung wieder ausgeschaltet.

Auch Steuer-Zeichen, wie z.B. das Wagenrücklauf-Symbol oder das Zeilenvorschub-Symbol können "übersetzt" werden. Das folgende Beispiel zeigt die Anweisungen, die die Umwandlung eines Wagenrücklauf-Symbols (CR) in ein Zeilenvorschub-Symbol (LF) bewirken. Diese Umsetzung ist nötig, wenn von einem Peripherie-Gerät Daten aufgenommen werden sollen, das lediglich das Wagenrücklauf-Symbol ausgibt, wo wir eigentlich ein Zeilenvorschub-Symbol als Anweisungs-Abschluß brauchen.

```
A$=CHR$(13)&CHR$(10)
CONVERT IN 7 PAIRS ; A$
```

Ein anderes "übersetzungsbeispiel" ist die Ausgabe des EBDIC-Codes an Stelle des ASCII-Codes. EBDIC ist eine Art der Zeichendarstellung, die von bestimmten Rechnern benutzt wird. Weil alle ASCII-Zeichen auch im EBDIC-Code vorkommen, ist es sinnvoll, hier die INDEX-Methode zu wählen, die einen String von 128 Zeichen erfordert. Im folgenden Beispiel ist es wichtiger, das Prinzip zu verstehen, als zu wissen, welches EBDIC-Zeichen zu welchem Dezimalwert gehört. Die Dezimalwerte der 128 EBDIC-Zeichen werden aus einer DATA-Anweisung gelesen und mit der CHR$-Funktion zu einem String geformt. Dieser String ist dann die übersetzungs-Tabelle, die in der CONVERT-Anweisung für das Interface mit dem Auswahl-Code 7 aufgerufen wird. Der Hinweis "INDEX" sagt dem Rechner, daß die auszugebenden ASCII-Zeichen mit ihrem numerischen Wert als Index zum Auffinden des richtigen EBDIC-Zeichens zu verwenden sind. So wird z.B. eine rechte, geschweifte Klammer, die im ASCII-Code den Dezimalwert 125 hat (dem entspricht die zweit-vorletzte Position in der unten angegebenen DATA-Aufstellung), in ein CHR$(155) gewandelt, weil mit diesem Code in EBDIC die gleiche Klammer definiert ist.

```
 10 DIM A$[128]
 20 A$=""
 30 DATA 0,1,2,3,55,45,46,47,22,
    5,37,11,12,13,14,15,16,17,18
    ,19,53,61,50,38,24,25,63,36
 40 DATA 28,29,30,31,64,90,127,1
    23,91,108,80,125,77,93,92,78
    ,107,96,75,97,240,241,242
 50 DATA 243,244,245,246,247,248
    ,249,122,94,76,126,110,111,1
    24,193,194,195,196,197,198
 60 DATA 199,200,201,209,210,211
    ,212,213,214,215,216,217,226
    ,227,228,229,230,231,232,233
 70 DATA 173,21,189,95,109,20,12
    9,130,131,132,133,134,135,13
    6,137,145,146,147,148,149
 80 DATA 150,151,152,153,162,163
    ,164,165,166,167,168,169,139
    ,79,155,74,7
 90 FOR I=1 TO 128
100 READ X
110 A$=A$&CHR$(X)
120 NEXT I
130 CONVERT OUT 7 INDEX ; A$
```

Fehlerbehandlung

Fehler während des Programmablaufs lassen sich bei den Rechnern der Serie 80 mit der ON ERROR-Anweisung festhalten und behandeln. Die ERRN- und die ERRL-Funktion, mit denen Sie wichtige Fehlermeldungen in Ihren Programmen anzeigen können, werden Sie sicher schon gut kennen. Diese Funktionen arbeiten auch im Zusammenhang mit dem I/O-ROM. Weil aber alle Zusatz-ROM's und Interfaces die Fehlernummern ab 101 aufwärts gemeinsam benutzen, ist eine weitere Unterscheidung nötig, wenn der Rechner mehr als ein Zusatz-ROM und mehr als ein Interface enthält, um den Anlaß für die Störung eingrenzen zu können.

Das I/O-ROM bietet zwei zusätzliche Funktionen zur Fehler-Eingrenzung, durch die der Programmierer die notwendigen Informationen zur Isolierung der Ursache des Fehlers erhält:

ERROM ermittelt die Kennzahl des ROM's, das den zuletzt aufgetretenen Fehler verursachte. Wenn dieser durch das I/O-ROM ausgelöst wurde, gibt die ERROM-Funktion den Wert 192 zurück. Beachten Sie bitte, daß diese Funktion nur durch ROM-Fehler aktualisiert wird. Die ERROM-Funktion "erinnert sich" also bei jedem neu auftretenden Fehler an das zuletzt "aufgefallene" ROM, auch wenn dieses an dem neuen Fehler nicht beteiligt ist!

ERRSC ermittelt den Auswahl-Code vom Interface, das den zuletzt aufgetretenen Interface-abhängigen Fehler verursachte. Beachten Sie bitte, daß diese Funktion nur durch Interface-abhängige Fehler aktualisiert wird. Die ERRSC-Funktion "erinnert sich" also bei jedem neu auftretenden Fehler an das zuletzt "aufgefallene" Interface, auch wenn dieses an dem neuen Fehler nicht beteiligt ist!

Weil auch andere ROM's einige der Fehlernummern des I/O-ROM's benutzen und weil die beiden obigen Funktionen grundsätzlich bei jedem Systemfehler aktualisiert werden, ist es wichtig, die verschiedenen Error-Funktionen in geeigneter Reihenfolge abzufragen. Wenn Sie in einer Fehler-Routine nach Fehlern des I/O-ROM's suchen, sollten Sie zuerst den Test "ERRN>100" ausführen. Nur wenn diese Bedingung erfüllt ist, steht fest, daß ein ROM oder ein Interface den Fehler auslöste. Nun kann mit "ERROM" das betroffene ROM ermittelt werden. Wenn das I/O-ROM der Auslöser war, sollte man mit "ERRN" die Fehlernummer bestimmen, bevor man mit "ERRSC" nach dem auslösenden Interface fragt: Die das I/O-ROM betreffende Fehlernummer 112 kann während des Programmablaufs nie auftreten und nur die Fehlernummern unter 123 betreffen Interface-abhängige Fehler. Die Abfrage "ERRN<123" sagt Ihnen dann, ob ein Interface-Fehler vorliegt.

Der folgende Programm-Ausschnitt zeigt eine zweckmäßige Reihenfolge von Prüfungen, um I/O-Fehler einzugrenzen. Das Beispiel zeigt nur die Fehlermeldung: Zu einer guten Fehler-Routine gehören aber auch noch Anweisungen, wie in der speziellen Situation zu verfahren ist, damit der Fehler überwunden wird.

```
 10 ON ERROR GOSUB 100
 20 !
 30 !
100 !TEST, OB KEIN I/O-FEHLER
110 IF ERRN <101 THEN GOTO 350
120 IF ERROM#192 THEN GOTO 350
130 !
140 ! TEST AUF INTERFACE-FEHLER
150 IF ERRN<123 THEN GOTO 260
160 !
170 !
180 ! FEHLER-ANZEIGE FUER DAS I/O-ROM
190 DISP "I/O-ROM-FEHLER";ERRN;"IN ZEILE";ERRL
200 !
210 ! HIER FEHLERBEHEBUNG ANGEBEN!
220 !
230 STOP @ RETURN
240 !
250 !
260 ! INTERFACE-FEHLER-ANZEIGE
270 DISP "INTERFACE-FEHLER";ERRN;"IN ZEILE";ERRL
280 DISP "VERURSACHT DURCH INTERFACE";ERRSC
290 !
300 ! HIER FEHLERBEHEBUNG ANGEBEN!
310 !
320 STOP @ RETURN
330 !
340 !
350 ! ANZEIGE SONSTIGER FEHLER
360 !
370 ! WEITERE FEHLER-UNTERSCHEIDUNG
380 ! MIT ABHILFE-TIPS HIER!
390 !
400 !
410 !
500 STOP @ RETURN
```

Im Anhang A dieses Buches gibt es eine komplette Liste aller Fehlermeldungen des I/O-ROM's und der verschiedenen Interfaces mit ihren Ursachen und Wirkungen und Hinweisen zur Behebung dieser Störungen.

Warum reden wir so viel von den Bits?

Die Menschheit hat die Zahlen erdacht. Sie sind angenommene Punkte auf einer kontinuierlichen Zahlen-Geraden, denen man sichtbare Zeichen, Töne oder irgend etwas anderes zuordnet, das der Mensch wahrnehmen kann. Die meisten Menschen sind es gewohnt, in einem Zahlensystem mit der Basis 10 zu denken. Im Gegensatz dazu sind Zahlen in einem Rechner elektronische Muster aus Einsen und Nullen. Die Computer führen viele Operationen in diesem Zahlensystem mit der Basis 2 durch. Für viele I/O-Anwendungen ist es typisch, daß diese elektronischen Muster aus Einsen und Nullen direkt als Steuersignale für andere elektronische Geräte geeignet sind.

Einige I/O-Operationen stellen mit Bit-Gruppen Zeichen und Werte dar, ein Verfahren, das die meisten Programmierer schon sehr zeitig gelernt haben. Ein Beispiel dafür ist der ASCII-Code. Andererseits werden bei den I/O-Operationen die Bits auch mit sehr "individuellen" Methoden "bearbeitet". Dieses Herumbasteln an den Bits kommt recht häufig vor, wenn man in das Gebiet der I/O-Steuerung einsteigt, und es kann dem Programmierer nur empfohlen werden, mit Bit-Mustern und binären Operationen beizeiten vertraut zu werden!

Dieser Abschnitt gibt einen überblick über das Zahlensystem mit der Basis 2, die verschiedenen Arten seiner Darstellung und die logischen Operatoren. Die beiden Abschnitte danach beschreiben die vom I/O-ROM gebotenen Funktionen zur Bearbeitung von Bitmustern und zum Umrechnen von Zahlen auf Systeme mit anderer Basis.

Ein Überblick über das System mit der Basis 2

Bevor wir uns mit der Basis 2 beschäftigem, sollten wir nochmal die uns vertraute Basis 10 anschauen: Die Zahl "Hunderfünfundzwanzig" wird mit Ziffernsymbolen so geschrieben:

125

Die einzelnen Ziffern haben dabei "Stellenwerte", die sich als Potenzen von 10 darstellen lassen:

$$1 \times 10^2 + 2 \times 10^1 + 5 \times 10^0$$

Auch im System mit der Basis 2 gibt es den Begriff des Stellenwertes, nur treten hier die Potenzen von 2 an die Stelle der Potenzen von 10.
Der Wert "Hundertfünfundzwanzig" stellt sich deshalb so dar:

1111101

Die Basis 2 benötigt nur die Ziffern "1" und "0", wobei die 1 anzeigt, daß der Stellenwert gebraucht wird, während eine 0 bedeutet, daß der Stellenwert nicht benötigt wird. Die obige binäre Darstellung läßt sich also auch so schreiben:

$$2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^0$$

Oder, was das gleiche ist:

$$64 + 32 + 16 + 8 + 4 + 1$$

Der Begriff "Bit" ist eine Kombination der englischen Worte "binary digit". Als Bit wird eine Ziffer des Zahlensystems mit der Basis 2 bezeichnet. Diese Ziffer kann nur die Form "0" oder "1" haben. Damit der Umgang mit den Bits bei Speicherung, Übermittlung und Zeichendarstellung einfacher abläuft, werden mehrere Bits zu einer Gruppe zusammenfaßt. Eine Gruppe von 8 Bits wird als "Byte" bezeichnet. Die Gruppierung zu 8 Bits ergab sich aus der Möglichkeit, diese Zahl von Bits jeweils gleichzeitig zu bearbeiten.

Beachten Sie, daß in beiden vorangegangenen Beispielen die am weitesten rechts stehende Ziffer den niedrigsten Stellenwert (mit dem Exponenten 0) hatte. Deswegen beginnt auch die Zählung der Bits für gewöhnlich mit einer 0 an Stelle einer 1, damit die Stellenbezeichnung des Bits mit dem Exponenten übereinstimmt, der zu dieser Stelle gehört. Die folgende Tabelle zeigt die Stellung der Bits eines Bytes mit ihrer Bedeutung und ihren Werten:

Bit-Position:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Bedeutung:	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Wert:	128	64	32	16	8	4	2	1

Beispiele: 130 als Dezimalzahl ist 10000010 als Binärzahl
3 als Dezimalzahl ist 00000011 als Binärzahl
25 als Dezimalzahl ist 00011001 als Binärzahl

Auch der Begriff "Wort" wird häufig im Zusammenhang mit dem Computer benutzt. Als "Wort" wird die Zahl von Bits bezeichnet, die der Computer wegen seiner inneren Struktur gleichzeitig bearbeiten kann. Obwohl die Rechner der Serie 80 intern eine 8-Bit-Struktur aufweisen, sind sie auch zu Operationen mit 16-Bit-Worten und Fließkomma-Zahlen fähig. Auch "Ganzzahlen", d.h. mit 16 Bits darstellbare Werte, werden im Zusammenhang mit dem Rechner als "Worte" bezeichnet, weil das System sie als Grundeinheit verarbeiten kann.

Damit hängt eine andere Gemeinsamkeit der Rechner der Serie 80 zusammen: Die Benutzung des 2-er-Komplements zur Zahlendarstellung. Dadurch ist für Betrag und Vorzeichen nur ein Wort nötig:

Ist Bit 15 eine Null, handelt es sich um eine positive Zahl, deren Betrag durch die restlichen Bits binär dargestellt ist.

Ist Bit 15 eine Eins, liegt eine negative Zahl in 2-er-Komplement-Darstellung vor, deren Betrag sich nach Invertieren des Wortes und Addieren von 1 ergibt.

Frage:
Welcher Zahlenwert gehört zu 11111111 11110000 ?

Lösung:
Bit 15 (links) ist eine 1, also ist es eine negative Zahl!
Wir invertieren alle Bits: 00000000 00001111
Addieren eine 1: 00000000 00000001
Das Ergebnis ist: 00000000 00010000
Der Zahlenwert des gegebenen Bitmusters ist -16

Übersicht über die verschiedenen Darstellungsarten

Wenn in einem Text Zahlen der unterschiedlichen Zahlensystemen auftreten, ist es unabdingbar, zwischen dem Zahlenwert und der Zahlendarstellung zu unterscheiden. Nehmen wir einmal die Zahl "einhundert". Der Zahlenwert kann z.B. durch hundert Bohnen verkörpert werden. Die Darstellung ist im Dezimal-System eine "1", gefolgt von "0" und "0". Der gleiche Wert kann aber auch als "64" (Basis 16) oder als "01100100" (Basis 2) oder als "10" (Basis 100) dargestellt werden.

Zur Darstellung einer Zahl werden verabredete Symbole verwendet, die den Zahlenwert verkörpern. Zahlen werden oft auch in nicht-dezimaler Form dargestellt, wenn die Benutzung einer anderen Basis anschaulichere Ergebnisse verspricht. Stellen wir uns ein kleines System mit 8 Pumpen und 8 Rohrleitungen vor, deren Zustand durch ein 16-Bit-Wort vom Rechner überwacht werden soll. Wenn diese Meldung dezimal ausgegeben wird, werden wir große Mühe haben, die Zahl der aktiven Pumpen und Leitungen zu erkennen. Wenn wir die Meldung aber binär ausgeben, mit der Absprache, daß das erste Byte die Pumpen und das zweite Byte die Leitungen betrifft, dann sagt die Meldung "00100000 10000000" sehr klar, daß eine Pumpe und eine Leitung in Betrieb sind. Die gleiche Meldung hätte dezimal "8320" gelautet. Wie lange hätten Sie wohl gebraucht, um diese Meldung zu deuten?

Wenn man eine Binärzahl dezimal darstellt, liegt das Problem darin, daß die einzelne Stelle des Dezimal-System keine ganze Zahl von Bits verkörpert. Ein Zahlenmuster der Basis 10 läßt nicht erkennen, welche Bits "1" oder "0" sind. Wenn man die binäre Darstellung benutzt, ist die Stellenzahl unbequem groß. Um diese Klippe zu umschiffen, benutzt man beim Umgang mit internen Vorgängen des Rechners, der ja nur Bits und Bytes kennt, gern die Basen 8 bzw. 16 zur Darstellung von Binär-Zahlen. Diese Basen haben Stellenwerte, die Potenzen von 2 sind, wodurch die Umrechnung reine Übungssache ist. Die Zahl der Stellen ist aber drei- bis viermal geringer als bei binärer Darstellung, was den Umgang mit den Zahlen vereinfacht. So sind bei Basis 16 nur zwei Stellen zur Darstellung eines Bytes nötig.

Mit der Basis 8, also mit "Oktalzahlen", lassen sich drei binäre Stellen als eine oktale Stelle darstellen. Mit der Basis 16, also mit "Hexadezimalzahlen" braucht man nur ein Hex-Zeichen, um vier binäre Stellen darzustellen. Die Tabelle stellt die dezimale (Basis 10), die binäre (Basis 2), die oktale (Basis 8) und die hexadezimale (Basis 16) Darstellungsart für die Zahlenwerte von 0 bis 16 gegenüber:

Dezimal	Binär	Oktal	Dezimal	Binär	Hex.
0	000	0	0	0000	0
1	001	1	1	0000	1
2	010	2	2	0010	2
3	011	3	3	0010	3
4	100	4	4	0100	4
5	101	5	5	0101	5
6	110	6	6	0110	6
7	111	7	7	0111	7
8	1 000	10	8	1000	8
9	1 001	11	9	1001	9
10	1 010	12	10	1010	A
11	1 011	13	11	1011	B
12	1 100	14	12	1100	C
13	1 101	15	13	1101	D
14	1 110	16	14	1100	E
15	1 111	17	15	1111	F
16	10 000	20	16	1 0000	10

Das Darstellen aller Werte unterhalb der Basis in einer "Stelle" ist die Idee der "Stellenwert-Technik". Für Darstellungen mit der Basis 16 muß man die Buchstaben A bis F als Symbole für die Werte 10 bis 15 "ausleihen". Das folgende Beispiel zeigt, wie sich lange Binärzahlen durch Aufteilen in kleinere Blöcke in gut lesbare Oktal- oder Hex-Zahlen übersetzen lassen:

Oktal-Zahlen:	1 2 5	3 6 0	0 7 0	2 0 1	2 0 2	
Oktal-Gruppen:	01 010 101	11 110 000	00 111 000	10 000 001	10 000 010	Immer das
Byte-Gruppen:	01010101	11110000	00111000	10000001	10000010	gleiche
Hex-Gruppen:	0101 0101	1111 0000	0011 1000	1000 0001	1000 0010	Bitmuster
Hex-Zahlen:	5 5	F 0	3 8	8 1	8 2	

Übersicht über die logischen Operationen

Bis zum Ende des I/O-ROM-Abschnitts 6 wird unter "logischer Operation" die Anwendung der Boole'schen Operatoren, wie z.B. AND und OR verstanden. Im Zusammenhang mit den I/O-Operationen lassen sich die logischen Operationen gut zur individuellen Änderung einzelner Bits benutzen, ohne die benachbarten Bits zu stören. Diese logischen Operationen unterscheiden sich dadurch deutlich von den arithmetischen Operationen Addition und Subtraktion: Bei den logischen Operationen gibt es nämlich in keiner Richtung irgenwelche Überträge, die sich bei arithmetischen Operationen manchmal von der letzten bis zur ersten Stelle durch eine Zahl fortsetzen, obwohl nur in der letzten Stelle etwas geändert wurde. So praktisch die arithmetischen Operationen für Berechnungen sein mögen, für nicht-numerische Dinge sind sie unbrauchbar! Kehren wir zum Beispiel mit den Pumpen und Leitungen zurück: Die durch eine Addition erzeugten Überträge würden die Übersicht über das ganze System stören! Wenn man einzelne Bits als individuelle Steuersignale benutzen will, dann muß die Möglichkeit bestehen, diese Bits einzeln zu beeinflussen. Die von den Rechnern der Serie 80 hierzu gebotenen Mittel werden im I/O-ROM-Abschnitt 6 beschrieben. Hier wird nur die Wirkung der üblichen logischen Operatoren erklärt.

Die Operatoren AND, OR, EXOR und NOT gehören bereits zum Standard-Anweisungssatz. Diese Operatoren prüfen aber eine Variable als Ganzes auf ihren Wert. Der Wert 0 gilt als "falsch", jeder andere, von Null verschiedene Wert gilt als "wahr". Für diese Operatoren gilt zwar auch die Boole'sche Algebra, sie bearbeiten aber nicht die einzelnen Bits. Die durch das I/O-ROM möglichen binären logischen Operationen verarbeiten dagegen die gegebenen "Worte" Bit für Bit. Ohne diese Binär-Funktionen wäre eine Isolierung eines einzelnen Bits nur mit umständlichen Prüfungen, Verzweigungen und Rechenoperationen möglich.

Die logischen Operatoren AND, OR und EXOR verknüpfen jeweils zwei Werte zu einem Ergebnis.

Schauen wir uns zuerst den AND-Operator an. Das bedeutet z.B.: Eis kann in einem Gefäß nur entstehen, wenn es Wasser enthält und die Temperatur unter dem Schmelzpunkt von Wasser liegt. Für den Rechner bedeutet dies, daß das Ergebnis nur dann eine "1" ergibt, wenn beide Eingaben den Wert "1" haben.
Das Symbol der UND-Verknüpfung ist ^ , mitunter wird auch * benutzt.
Die Wahrheitstabelle für das AND sieht so aus:

A	B	A^B
0	0	0
0	1	0
1	0	0
1	1	1

Wichtig ist eine zweite Überlegung: Das Ergebnis ist 0 für A=0 und B für A=1. Das binäre UND ist also ein bequemes Verfahren zur Löschung bestimmter Bits. Wenn Sie z.B. die beiden Bits mit dem geringsten Wert löschen wollen, brauchen Sie nur das Original-Byte und eine passende Maske der AND-Operation zu unterwerfen:

Original-Byte:	10011101
Masken - Byte:	11111100
Ergebnis-Byte:	10011100

Diese Operation löscht die beiden niedrigsten Bits, unabhängig von deren Zustand zuvor, und erhält den Zustand der 6 oberen Bits. Und das alles ohne lange Prüfungen und Verzweigungen!

Der nächste Operator ist das binäre ODER, oder vollständig "Inklusiv-ODER". Das bedeutet z.B.: Sie können Eiskrem oder Kuchen als Dessert haben, aber auch (wenn Ihr Magen das verträgt) beides! In der technischen Literatur (auch in Patent-Ansprüchen) führt das zu der eigenartigen Wortform "und/oder", um auf den inklusiven Charakter hinzuweisen. Für den Rechner bedeutet dies, daß das Ergebnis eine "1" ist, sobald mindestens eine der beide Eingaben den Wert "1" hat. Das Symbol der Inklusiv-OR-Verknüpfung ist v , mitunter wird auch + benutzt.
Die Wahrheitstabelle für das Inklusiv-OR sieht so aus:

A	B	AvB
0	0	0
0	1	1
1	0	1
1	1	1

Wichtig ist eine zweite Überlegung: Das Ergebnis ist 1 für A=1 und B für A=0. Das binäre Inklusiv-ODER ist also ein bequemes Verfahren zum Setzen bestimmter Bits. Wenn Sie z.B. die beiden Bits mit dem geringsten Wert setzen wollen, brauchen Sie nur das Original-Byte und eine passende Maske der OR-Operation zu unterwerfen:

Original-Byte:	10011101
Masken - Byte:	00000011
Ergebnis-Byte:	10011111

Diese Operation setzt die beiden niedrigsten Bits, gleichgültig wie deren Zustand vorher war, und erhält den Zustand der 6 oberen Bits.

Die einfachste logische Operation ist die Bildung des Komplements. Zu dieser Operation ist nur ein Argument nötig. Die Operation läßt sich etwa mit einem sehr eigensinnigen Menschen vergleichen, der immer genau das Gegenteil von dem tut, was man ihm sagt. Für den Rechner bedeutet dies, das er in einer Binär-Zahl jede "0" in eine "1" und jede "1" in eine "0" wandeln muß. Das wird auch "1-er-Komplement" oder "Invertierung" genannt. Das Symbol dafür ist in der Boole'schen Algebra ein waagerechter Strich über der Variablen. Die Wahrheitstabelle dafür sieht so aus:

A	$\overline{A}$
0	1
1	0

Wenn man ein ganzes Byte der Komplementbildung unterzieht, wird jedes Bit einzeln komplementiert:

Binär-Darstellung von A: 10011101
Binär-Komplement von A: 01100010

Der nächste Operator, den wir kennenlernen müssen, ist EXOR, das 'Exklusiv-ODER'. Das bedeutet z.B.: Sie können mit dem Zug oder mit dem Flugzeug von Hamburg nach Stuttgart kommen; Sie müssen sich aber für eines von beiden entscheiden! Für den Rechner bedeutet dies, daß das Ergebnis eine "1" ist, sobald eine und nur eine der beiden Eingaben den Wert "1" hat (Wenn beide Eingaben den Wert "1" haben, ist das Ergebnis genau so, als wenn beide Eingaben den Wert "0" haben). Das Symbol der EXOR-Verknüpfung ist $\veebar$, mitunter wird auch $\oplus$ benutzt.
Die Wahrheitstabelle für das EXOR sieht so aus:

A	B	A$\veebar$B
0	0	0
0	1	1
1	0	1
1	1	0

Wichtig ist eine zweite Überlegung: Das Ergebnis ist B für A=0 und das Komplement von B für A=1. Das binäre Exklusiv-ODER ist also ein bequemes Verfahren zum Komplementieren bestimmter Bits. Wenn Sie z.B. die beiden Bits mit dem geringsten Wert komplementieren wollen, brauchen Sie nur das Original-Byte und eine passende Maske der EXOR-Operation zu unterwerfen:

Original-Byte: 10011110
Masken - Byte: 00000011
Ergebnis-Byte: 10011101

Diese Operation komplementiert die beiden niedrigsten Bits und erhält den Zustand der 6 oberen Bits.

Binär-Funktionen

Wie schon in den voranstehenden Abschnitten erwähnt wurde, ist es bei Programmen mit I/O-Operationen oftmals notwendig, einzelne Bits auf ihren Zustand abzufragen oder auch im gewünschten Sinn zu verändern. Das I/O-ROM stellt für diese Zwecke einige sehr bequem anwendbare Funktionen bereit. Die Binär-Funktionen werden benötigt, um mit den Status- und Steuer-Registern der Interfaces sinnvoll arbeiten zu können, sie sind aber auch zur Bearbeitung von I/O-Daten bei der Steuerung der angeschlossenen Geräte recht nützlich. Dieser Abschnitt beschreibt die mit den Rechnern der Serie 80 ausführbaren Binär-Funktionen.

Dabei ist es wichtig, sich daran zu erinnern, daß alle in Verbindung mit dem I/O-ROM benutzten Binär-Funktionen mit Worten arbeiten, die aus 16 Bits aufgebaut sind. So ist z.B. das binäre Komplement für den Wert Null '11111111 11111111' (zur Basis 2). Die einzelnen Positionen der Bits sind von 0 bis 15 numeriert, der Werte-Bereich für binäre Argumente liegt zwischen -32 768 und +32 767. Die Binärfunktionen können ohne weiteres auch auf Binärzahlen mit weniger als 16 Bits angewendet werden, für die fehlenden (höherwertigen) Bits werden dann einfach Nullen angenommen.

In den folgenden Erklärungen wird der Begriff "Ganzzahl" häufig gebraucht, um damit die Argumente der Binär-Funktionen zu beschreiben. Für das I/O-ROM ist eine "Ganzzahl" eine Binärzahl mit 16 Bits aus dem Werte-Bereich von -32 768 bis +32 767. Dies ist eine andere Festlegung als die aus dem "Bedienungs- und Programmier-Handbuch": Dort war eine "Ganzzahl" (Integer) eine 5-stellige Zahl im Bereich von -99 999 bis +99 999. Beachten Sie bitte diesen Unterschied, um Mißverständnisse mit dem Begriff "Ganzzahl" zu vermeiden!

Wenn Ihnen Binär-Operationen wie AND und EXOR nicht vertraut sind, sollten Sie dem I/O-ROM-Abschnitt 5 einige Zeit widmen, bevor Sie hier weiterlesen!

Die binäre UND-Funktion (AND)

Die binäre UND-Funktion unterwirft die entsprechenden Bits von 2 "Ganzzahlen" einzeln der UND-Verknüpfung und gibt das Ergebnis dieser Verknüpfungen wiederum als "Ganzzahl" aus. Hier einige Anwendungs-Beispiele der BINAND-Funktion:

```
X=BINAND(Y,15)
A=C+BINAND(D,E)
PRINT BINAND(255,Z-32)
```

Beachten Sie bitte, daß die Argumente in der Klammer durch ein Komma zu trennen sind. Als Argumente sind zulässig: numerische Konstanten, numerische Variable, numerische Ausdrücke oder Kombinationen aus diesen drei Möglichkeiten. Es wird stillschweigend vorausgesetzt, daß die Argumente dekadisch zu bewerten sind. Wenn Sie die Argumente auf eine andere Basis bezogen wissen wollen, schauen Sie bitte in den I/O-ROM-Abschnitt 7! Jedes Bit des Ergebnisses wird nach folgender Wahrheitstabelle ermittelt:

1. Argument	2. Argument	Ergebnis
0	0	0
0	1	0
1	0	0
1	1	1

Die binäre Inklusiv-ODER-Funktion (OR)

Die binäre Inklusiv-ODER-Funktion unterwirft die entsprechenden Bits von 2 "Ganzzahlen" einzeln der Inklusiv-ODER-Verknüpfung und gibt das Ergebnis dieser Verknüpfungen wiederum als "Ganzzahl" aus. Hier einige Anwendungs-Beispiele der BINIOR-Funktion:

```
X=BINIOR(Y,15)
A=C+BINIOR(D,E)
PRINT BINIOR(255,Z-32)
```

Beachten Sie bitte, daß die Argumente in der Klammer durch ein Komma zu trennen sind. Als Argumente sind zulässig: numerische Konstanten, numerische Variable, numerische Ausdrücke oder Kombinationen aus diesen drei Möglichkeiten. Es wird stillschweigend vorausgesetzt, daß die Argumente dekadisch zu bewerten sind. Wenn Sie die Argumente auf eine andere Basis bezogen wissen wollen, schauen Sie bitte in den I/O-ROM-Abschnitt 7! Jedes Bit des Ergebnisses wird nach folgender Wahrheitstabelle ermittelt:

1. Argument	2. Argument	Ergebnis
0	0	0
0	1	1
1	0	1
1	1	1

Die binäre Exklusiv-ODER-Funktion (EXOR)

Die binäre Exklusiv-ODER-Funktion unterwirft die entsprechenden Bits von 2 "Ganzzahlen" einzeln der Exklusiv-ODER-Verknüpfung und gibt das Ergebnis dieser Verknüpfungen wiederum als "Ganzzahl" aus. Hier einige Anwendungs-Beispiele für die BINEOR-Funktion:

```
X=BINEOR(Y,15)
A=C+BINEOR(D,E)
PRINT BINEOR(255,Z-32)
```

Beachten Sie bitte, daß die Argumente in der Klammer durch ein Komma zu trennen sind. Als Argumente sind zugelassen: numerische Konstanten, numerische Variable, numerische Ausdrücke oder Kombinationen aus diesen drei Möglichkeiten. Es wird stillschweigend vorausgesetzt, daß die Argumente dekadisch zu bewerten sind. Wenn Sie die Argumente auf eine andere Basis bezogen wissen wollen, schauen Sie bitte in den I/O-ROM-Abschnitt 7! Jedes Bit des Ergebnisses wird nach folgender Wahrheitstabelle ermittelt:

1. Argument	2. Argument	Ergebnis
0	0	0
0	1	1
1	0	1
1	1	0

Die binäre Komplement-Funktion

Die binäre Komplement-Funktion invertiert alle Bits einer "Ganzzahl" und gibt das Ergebnis wiederum als "Ganzzahl" aus. Hier einige Anweisungs-Beispiele für die BINCMP-Funktion:

```
A=BINCMP(B)
X=BINCMP(Y-Z)
PRINT BINCMP(N*8)
```

Beachten Sie bitte, daß das Argument in Klammern einzuschließen ist. Als Argument ist zulässig: numerische Konstante, numerische Variable, numerischer Ausdruck und Kombinationen aus diesen drei Möglichkeiten. Stillschweigend wird vorausgesetzt, daß das Argument dekadisch zu werten ist. Wenn Sie das Argument auf eine andere Basis bezogen wissen wollen, schauen Sie bitte in den I/O-ROM-Abschnitt 7! Jedes Bit des Ergebnisses wird nach folgender Wahrheitstabelle ermittelt:

Argument	Ergebnis
0	1
1	0

Sie sollten immer daran denken, daß die binäre Komplement-Funktion vollständige 16-Bit-Worte voraussetzt. Das kann zu unerwarteten Ergebnissen führen, wenn Sie ausschließlich mit Bytes (also mit 8-Bit-Worten) arbeiten: Das 16-Bit-Komplement für ein 8-Bit-Wort ist stets eine negative Zahl! 8-Bit-Komplemente lassen sich erzeugen, indem man das 8-Bit-Wort und die Zahl 255 der BINEOR-Funktion unterwirft: dann werden die 8 niedrigsten Bits invertiert und die 8 höchsten Bits behalten den Wert Null. Das Verfahren läßt sich leicht auf jede Wortlänge zwischen 1 und 15 Bits anpassen.

Die Bit-Test-Funktion

Die Bit-Test-Funktion stellt fest, ob ein bestimmtes Bit einer "Ganzzahl" gesetzt oder gelöscht ist. Als Ergebnis wird eine 0 (falls gelöscht) oder eine 1 (falls gesetzt) ausgegeben. Die allgemeine Form der Bit-Test-Anweisung lautet:

```
BIT (Ganzzahl,Bitposition)
```

Das ganzzahlige Argument muß als erstes in der Klammer stehen und von der Bitposition durch ein Komma getrennt sein. Als Bitposition dürfen nur Werte im Bereich von 0 bis 15 (einschließlich) auftreten, wobei die Position 0 das Bit mit der geringsten und die Position 15 das Bit mit der höchsten Wertigkeit bezeichnet. Als Argumente sind numerische Konstanten, numerische Variable und numerische Ausdrücke oder Kombinationen aus diesen drei Möglichkeiten zugelassen. Hier einige Anwendungs-Beispiele der Bit-Test-Funktion:

```
A=BIT(B,3)
X=BIT(Y,Z-1)
IF BIT(N,1) THEN GOSUB 220
```

Die Bit-Test-Funktion ist für Entscheidungen und Programmverzweigungen sehr nützlich. Durch Kombination mit der IF-Anweisung wird damit der Programmablauf vom Wert bestimmter Bits beeinflußt. Die Funktion liefert den Wert 0, wenn das abgefragte Bit gelöscht ist und den Wert 1, wenn dieses Bit gesetzt ist.

Funktionen zur Basis-Konvertierung

Programmierer, die häufig mit Bits und Bytes umgehen müssen, entwickeln eine Vorliebe für Zahlensysteme, die sich auch gut zur Darstellung von Bytes und 'Worten' eignen. Für die Darstellung von Bit-Mustern ist zweifellos die Benutzung der Basis 2 die beste Lösung. Die Basis 8 war zu einer Zeit sehr beliebt, als die Rechner noch Schwierigkeiten mit der Annahme oder Darstellung nicht-numerischer Zeichen hatten. Erst danach wurde die 16 als Basis sehr populär, weil die meisten Rechner mit Wortlängen arbeiten, die ganzzahlige Vielfache von 4 sind, und moderne Systeme die Zeichen A bis F leicht in hexadezimale Werte umsetzen können.

Um die jeweils anschaulichste und bequemste Zahlendarstellung benutzen zu können, stellen die Rechner der Serie 80 Umwandlungs-Funktionen bereit, mit denen Eingaben und Ausgaben in jedem der oben genannten Zahlensysteme vorgenommen werden können. Die vorhandenen Funktionen zur Basis-Konvertierung haben einiges gemein:

Alle Konvertierungs-Funktionen haben die Basis 10 als Ausgangs- oder Zielpunkt. Eine Umwandlung einer Zahl mit der Basis 2, 8 oder 16 in eine Zahl mit einer dieser Basen ist nur über die Basis 10 möglich.

Die Basis-10-Seite der Konvertierung hat immer numerischen Charakter, die Seite mit der von 10 abweichenden Basis ist immer ein String.

Weil die Darstellung eines Wertes mit einer von 10 abweichenden Basis immer ein String ist, kann diese Darstellung, wie jeder String, ein- bzw. ausgegeben, verglichen, gespeichert und auch in einem gewissen Umfang verändert werden. Es sind aber mit diesen Strings keine arithmetischen Operationen durchführbar.

Alle Argumente für Konvertierungsfunktionen müssen im Bereich derjenigen "Ganzzahlen" liegen, ders ich mit 16 Bits beherrschen läßt. Diese Einschränkung gilt für beide Seiten der Konvertierung, also auch für die Dezimaldarstellung.

Konvertierung von der Basis 10 zur Basis 2, 8 oder 16

Diese Funktionen verarbeiten eine Zahl des Dezimalsystems zu einem String, der entsprechend der gewählten Basis geformt wird. Hauptzweck dieser Funktionen ist Anzeige, Druck oder Ausgabe von Werten in einer Darstellung mit der gewünschten Basis. Selbstverständlich sind diese Funktionen auch allgemein anwendbar!

Als Argument sind dabei numerische Konstante, numerische Variable und numerischer Ausdruck oder eine Kombinbation aus diesen Möglichkeiten zugelassen. Vom Argument wird nach der Rundung der ganzzahlige Teil verwendet, der im Bereich von -32768 bis +32767 liegen muß.
Funktionen für die Konvertierung zu den Basen 2, 8 und 16 stehen zur Verfügung.

Von der Basis 10 zur Basis 2

Es handelt sich um die "Dezimal-zu-Binär-String"-Funktion. Sie wandelt den ganzzahligen Teil des gerundeten Arguments in einen String von 16 Einsen und Nullen um. Dieser String ist die Verkörperung der ursprünglich dezimalen Ganzzahl als Zahl zur Basis 2. Wenn das Argument oberhalb der obigen Grenzen lag, liefert die Funktion den String "0111111111111111". War das Argument unterhalb der obigen Grenzen, liefert die Funktion "1000000000000000". Es folgen einige Anwendungs-Beispiele:

```
PRINT DTB$(X)
A$=DTB$(N*2)
OUTPUT 701;DTB$(32+Y)
DISP DTB$(255)
```

Von der Basis 10 zur Basis 8

Es handelt sich um die "Dezimal-zu-Oktal-String"-Funktion. Sie wandelt den ganzzahligen Teil des gerundeten Arguments in einen String von 6 Zeichen um. Dieser String ist die Verkörperung der ursprünglich dezimalen Ganzzahl als Zahl zur Basis 8. Wenn das Argument oberhalb der obigen Grenzen lag, liefert die Funktion den String "077777". War das Argument unterhalb der obigen Grenzen, liefert die Funktion "100000". Es folgen einige Anwendungs-Beispiel:

```
PRINT DTO$(X)
A$=DTO$(N*2)
OUTPUT 701;DTO$(32-Y)
DISP DTO$(255)
```

Von der Basis 10 zur Basis 16

Es handelt sich um die "Dezimal-zu-Hex-String"-Funktion. Sie wandelt den ganzzahligen Teil des gerundeten Arguments in einen String von 4 Zeichen um. Dieser String ist die Verkörperung der ursprünglich dezimalen Ganzzahl als Zahl zur Basis 16. Wenn das Argument oberhalb der obigen Grenzen lag, liefert die Funktion den String "7FFF". Wenn das Argument unterhalb der obigen Grenzen lag, liefert die Funktion "8000". Es folgen einige Anwendungs-Beispiele:

```
PRINT DTH$(X)
A$=DTH$(N*2)
OUTPUT 701;DTH$(32-Y)
DISP DTH$(255)
```

Konvertierung von den Basen 2, 8 oder 16 zur Basis 10

Diese Funktionen verarbeiten einen String und erzeugen einen numerischen Wert. Hauptzweck dieser Funktionen ist die Annahme von Daten, die in binärer, oktaler oder hexadezimaler Form vorliegen. Selbstverständlich sind diese Funktionen auch allgemein anwendbar!

Als Argument sind dabei String-Konstante, String-Variable und String-Ausdruck oder eine Kombinbation aus diesen Möglichkeiten zugelassen. Das Argument muß eine Ganzzahl verkörpern, die sich mit 16 Bits darstellen läßt.
Funktionen für Konvertierungen von den Basen 2,8 und 16 stehen zur Verfügung.

Von der Basis 2 zur Basis 10

Es handelt sich um die "Binär-zu-Dezimal"-Funktion. Sie wandelt einen String, der die binäre Verkörperung des Wertes darstellt, in die Dezimaldarstellung um. Das ursprüngliche Argument darf aus maximal 16 Einsen und Nullen bestehen. Weil das Resultat numerisch ist, kann es mit numerischen Funktionen und Operationen weiter bearbeitet werden. Es folgen einige Anwendungs-Beispie:

```
PRINT BTD("100101")
X=BTD(A$)
Y=255-BTD(N$)
```

Von der Basis 8 zur Basis 10

Es handelt sich um die "Oktal-zu-Dezimal"-Funktion. Sie wandelt einen String, der die oktale Verkörperung des Wertes darstellt, in die Dezimaldarstellung um. Das ursprüngliche Argument darf aus maximal 6 Zeichen bestehen. Es sind nur die Ziffern "0" bis "7" zugelassen. Wenn das Argument 6-stellig ist, darf in der ersten Stelle nur eine "0" oder "1" stehen. Weil das Resultat numerisch ist, kann es mit numerischen Funktionen und Operationen weiter bearbeitet werden. Es folgen einige Anwendungs-Beispiele:

```
PRINT OTD("371")
X=OTD(A$)
Y=255-OTD(N$)
```

Von der Basis 16 zur Basis 10

Es handelt sich um die "Hex-zu-Dezimal"-Funktion. Sie wandelt einen String, der die hexadezimale Verkörperung des Wertes darstellt, in die Dezimaldarstellung um. Das ursprüngliche Argument darf aus maximal 4 Zeichen bestehen. Es sind nur die Ziffern "0" bis "9" und die Buchstaben "A" bis "F" zugelassen. Weil das Resultat numerisch ist, kann es mit numerischen Funktionen und Operationen weiter bearbeitet werden. Es folgen einige Anwendungs-Beispiele:

```
PRINT HTD("1F4")
X=HTD(A$)
Y=255-HTD(N$)
```

Beliebige Konvertierungen zwischen den Basen 2, 8 und 16

Derartige Konvertierungen lassen sich sehr leicht verwirklichen, indem man zwei Konvertierungsfunktionen ineinander verschachtelt. Das folgende kurze Programm ist ein Beispiel für diese Technik: Es stellt hexadezimale Eingaben in binärer Form dar.

```
10 DISP "Welche Hex-Zahl ist zu wandeln";
20 INPUT A$
30 CLEAR
40 DISP A$;"(hex)= ";DTB$(HTD(A$));"(binär)"
50 GOTO 10
```

Weitere Übertragungsverfahren

Einführung

Die I/O-ROM-Abschnitte 8 bis 11 beschreiben Verfahren, die dem erfahrenen Programmierer neue Wege zur Lösung seiner oft komplexen Probleme zeigen. Vielleicht benötigen Sie diese Verfahren nie, es ist aber gut zu wissen, daß es sie gibt!

Hier wird erklärt, was ein "I/O-Buffer", was "Transfer", was eine "Zeilen-End-Verzweigung" und was eine "Tasten-Maske" ist. Auch der Zugang zu den Status- und Steuer-Registern der Interfaces wird hier gezeigt.

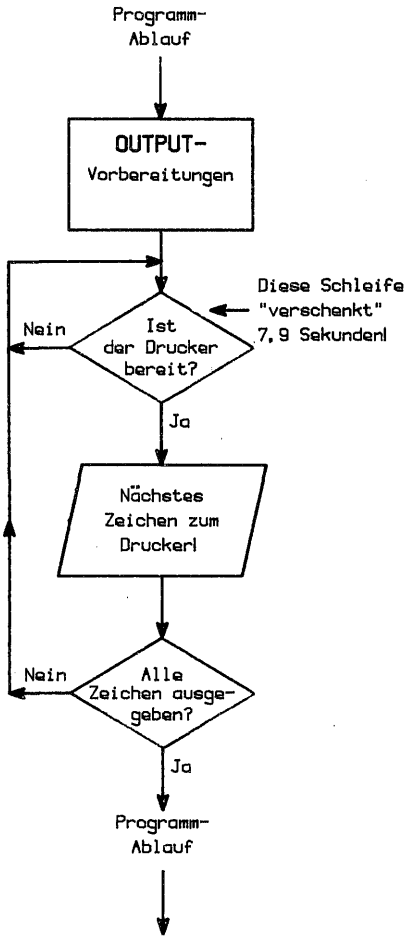
Lesen Sie bei Bedarf die entsprechenden Abschnitte sorgfältig durch, um die dort beschriebenen Möglichkeiten voll zu nutzen!

Dieser Abschnitt beschäftigt sich mit der TRANSFER-Anweisung, mit der sich Daten auf sehr flexible Weise übertragen lassen. Eindrucksvoll ist die Anpassungsfähigkeit der TRANSFER-Methode. Sie erlaubt es uns, die Geschwindigkeit des Rechners wesentlich besser zu nutzen, wenn er mit der Peripherie zusammenarbeitet. Stellen wir uns einmal ein sehr langsames Gerät (z.B. Drucker) vor, das pro Sekunde nur 8 Zeichen schafft. Er braucht also für 80 Zeichen 10 Sekunden. Der Rechner könnte diese 80 Zeichen aber in weniger als 0,1 Sekunden an den Drucker geben! Der Rechner ist also gezwungen, auf den Drucker zu warten und man verschenkt somit jedesmal 7,9 Sekunden von 8 Sekunden. Der Rechner ließe sich viel besser nutzen, wenn es gelingt, die Wartezeit zu vermeiden. Wir wollen sehen wie!

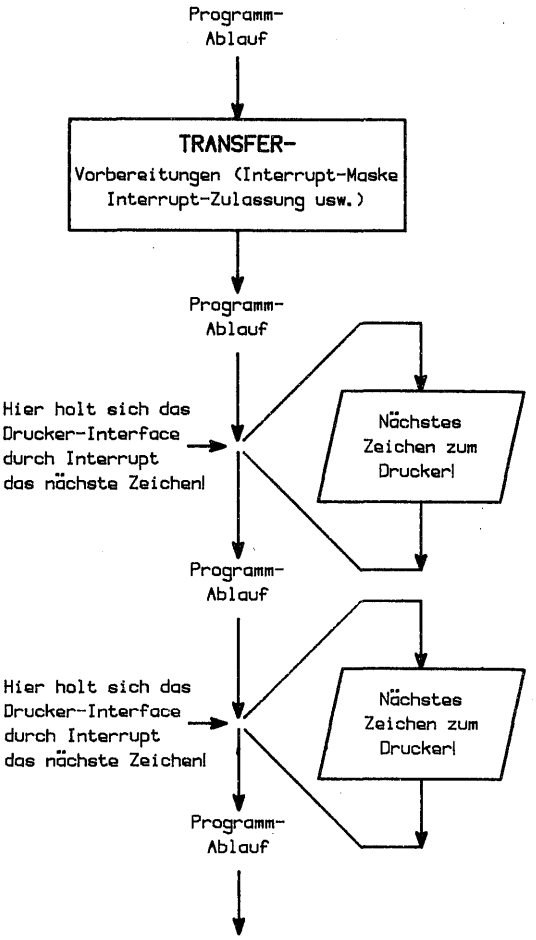
Das Bild auf Seite ROM-49 stellt die bisher bekannte "Handshake"-Methode, die mit OUTPUT- und ENTER-Anweisungen arbeitete, der "Interrupt"-Methode gegenüber, die die TRANSFER-Anweisung benutzt. Wenn der Rechner eine OUTPUT-Anweisung ausführt, ist er gezwungen, jedes Zeichen einzeln per "Handshake" zu übergeben, bis alle Daten ausgegeben sind. Erst dann kann der Rechner die nächste Anweisung in Angriff nehmen. Dagegen benutzt die TRANSFER-Anweisung bei der Interrupt-Methode einige Spezial-Zeiger (Pointer) zur Steuerung der Ausgabe und räumt dem Drucker-Interface die Möglichkeit ein, den Rechner zu "unterbrechen". Dadurch kann der Rechner schon 10 ms nach der Ausgabe eines Zeichens die nächste Programmzeile bearbeiten! (Das Zulassen eines "Interrupts" gleicht hier dem Auflegen des Telefonhörers: man muß seine Arbeit "unterbrechen", wenn jemand anruft!)

Der Rechner arbeitet nun in seinem Programm weiter, bis der Drucker zur Aufnahme des nächsten Zeichens bereit ist. Der Rechner wird dann vom Drucker-Interface unterbrochen, gibt schnellstens das nächste Zeichen aus, stellt den Zeiger um eine Stelle weiter, schaut nach, ob es sich etwa um das letzte Zeichen gehandelt hat und geht dann wieder zu der Arbeit über, bei der er unterbrochen wurde. Wenn alle Zeichen ausgegeben wurden, sperrt der Rechner die Möglichkeit, ihn bei der Arbeit zu stören und geht wieder ausschließlich seiner Haupttätigkeit nach. (Das ist wie das Abnehmen des Telefonhörers ohne Grund, damit der Anschluß 'besetzt' erscheint und man deswegen nicht mehr gestört werden kann!)

Handshake-Methode

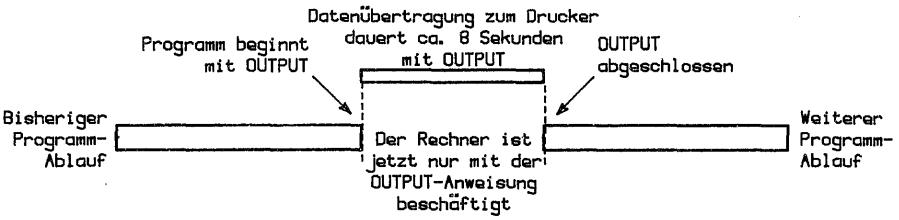


Interrupt-Methode

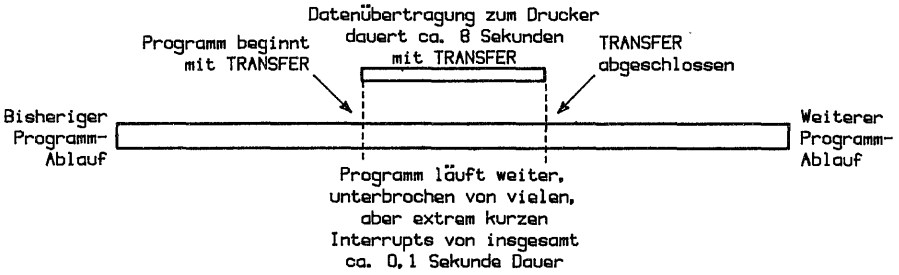


Wenn man sich bei beiden Verfahren den allgemeinen Ablauf ansieht, erkennt man den entscheidenden Unterschied: die Handshake-Methode ist ein lineares oder **sequentielles** Verfahren, während die Interrupt-Methode, zumindest zeitweise, parallel oder **überlappend** arbeitet. Beschäftigen wir uns deshalb auch mit dem Schaubild auf der nächsten Seite:

Sequentiell:



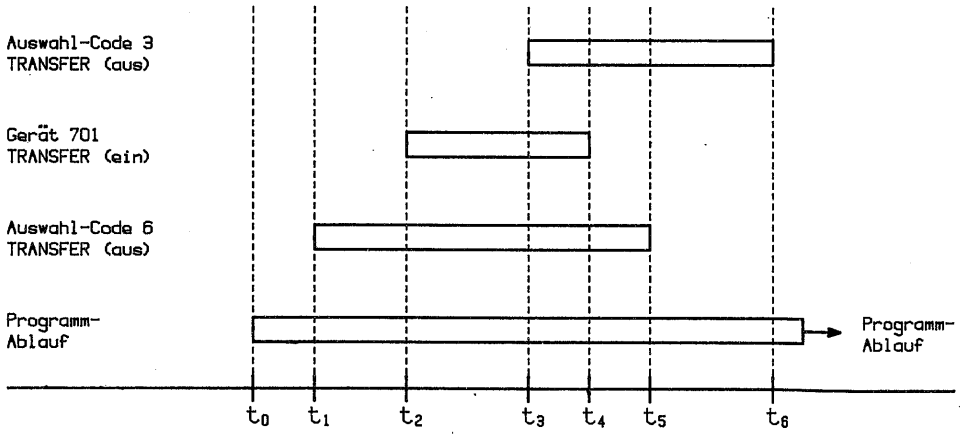
Überlappend:



Beim sequentiellen Ablauf wird bei Erreichen der OUTPUT-Anweisung nur diese bearbeitet, wobei die Bearbeitungsdauer von der Arbeitsgeschwindigkeit des aufnehmenden Gerätes und der Datenmenge abhängt. Dagegen beeinflußt bei einer TRANSFER-Anweisung die Arbeitsgeschwindigkeit des aufnehmenden Gerätes und die Menge der Daten nur die Dauer des (scheinbaren) Parallel-Betriebs von Programm und Datenübermittlung. Eine tatsächliche Verzögerung des laufenden Programms ist dagegen kaum bemerkbar, weil während der Interrupts die Information nur von einem Speicher im Rechner an einen Speicher des Interfaces übergegeben wird.

Noch günstiger wird dieser überlappende Ablauf, wenn mehrere I/O-Operationen zur gleichen Zeit auftreten: Nehmen wir an, daß auf die erste TRANSFER-Anweisung eine weitere TRANSFER-Anweisung für ein anderes Gerät folgt (z.B. ein weiterer Monitor), dann geschehen sogar drei Dinge gleichzeitig: das Programm läuft weiter, der Drucker ist in Aktion und auch auf dem Monitor geschieht etwas.

Das folgende Schaubild zeigt nun das Prinzip, wie sich durch zeitlich überlappende TRANSFER-Anweisungen die Leistungsfähigkeit eines Systems enorm steigern läßt. (Bitte, beachten Sie, daß dies nur für das "Interrupt"-TRANSFER gilt, wie später noch erklärt wird.)



t_0 : Die Ausführung des Programms beginnt.

t_1 : Das Interface 6 beginnt eine TRANSFER-Ausgabe.

t_2 : Das Interface 7 beginnt eine TRANSFER-Eingabe.

t_3 : Interface 3 beginnt eine TRANSFER-Ausgabe.

Jetzt laufen drei Vorgänge neben dem Programm!

t_4 : Interface 7 beendet die TRANSFER-Eingabe

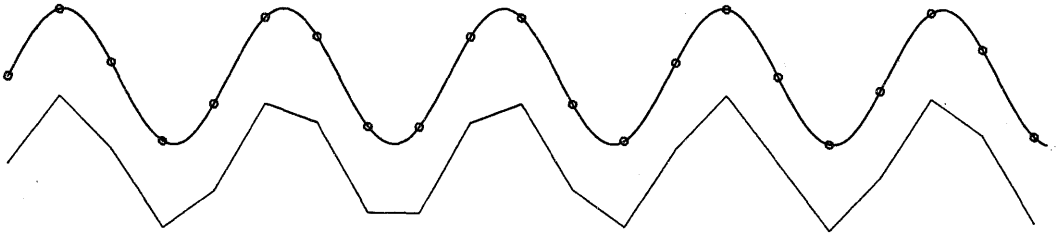
t_5 : Interface 6 beendet die TRANSFER-Ausgabe.

t_6 : Interface 3 beendet die TRANSFER-Ausgabe.

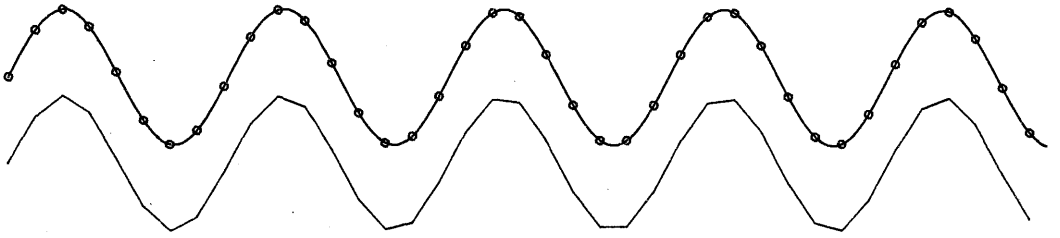
Das Programm lief während der Ein- und Ausgaben (kaum verzögert!) weiter, womit wohl deutlich wird, wie leistungsfähig die Interrupt-Methode das System macht!

Die TRANSFER-Anweisung ist aber auch noch in einer zweiten Hinsicht sehr anpassungsfähig: es lassen sich damit nicht nur mehrere (relativ langsame) Aufgaben zeitlich praktisch parallel ausführen, es ist auch möglich, eine einzige Angelegenheit mit sehr großer Geschwindigkeit zu erledigen. Es gibt nämlich Dinge, die sind sinnlos oder unmöglich, wenn sie nur mit niedriger oder mittlerer Geschwindigkeit ausgeführt werden können: Das Schaubild auf Seite ROM-51 zeigt, den Einfluß der Abtasthäufigkeit auf die Güte der Rekonstruktion eines Vorganges.

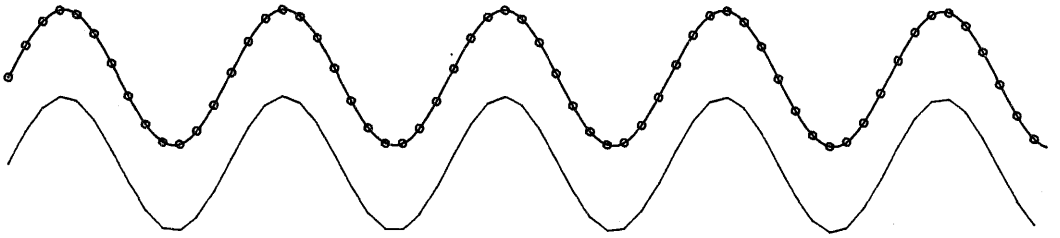
Es ist nun ohne Schwierigkeiten möglich, den zeitlichen Verlauf dieses Signals mit dem Digitalvoltmeter HP3437A aufzunehmen, weil dieses Gerät bis zu 3600 Messungen (3 1/2-stellig) in jeder Sekunde durchführen kann, was eine Daten-Flut von 25 000 Zeichen pro Sekunde ergibt! Die Rechner der Serie 80 sind zwar nicht ganz so schnell, aber etwa 20 000 Zeichen pro Sekunde können sie schon aufnehmen, wenn es mal schnell gehen muß!



Rekonstruktion aus ca. 4 Abtastungen pro Periode



Rekonstruktion aus ca. 8 Abtastungen pro Periode



Rekonstruktion aus ca. 13 Abtastungen pro Periode

Die Abtastung mit geringer Häufigkeit ergibt nur wenige Meßwerte für den gesamten Verlauf des Vorganges. Die Rekonstruktion ist auch nur ein recht grob angenähertes Bild des ursprünglichen Verlaufs. Mit Erhöhung der Zahl der Abtastungen wird der Unterschied zwischen Original und Rekonstruktion immer geringer. Es ist wohl klar, daß eine derartige Datenflut nur mit einem sehr schnellen Übertragungs-Verfahren bewältigt werden kann. Hier heißt die Lösung "Fast-Handshake".

Eine TRANSFER-Anweisung nach der "Fast-Handshake"-Methode läßt sich gut mit einem Formel-1-Rennwagen vergleichen: Nur ein Sitz, Handschaltung, winzige Windschutzscheibe, knochenharte Federung und kein Koffer-Raum, dafür aber über 300 km/h.

Die bereits erwähnte Interrupt-Methode entspricht dann einem Wohnmobil, das von einer Person gefahren wird, während die restlichen Mitreisenden anderweitig mit Lesen, Kochen oder Spielen die Zeit verbringen und nur hin und wieder den Fahrer nach dem Ablauf der Reise fragen.

Die schon vertrauten OUTPUT- und ENTER-Anweisungen mit ihren vielfältigen Formatierung- und Konvertierungsmöglichkeiten für die Daten sind dann der Rolls-Royce mit automatischem Getriebe, elektrischen Fensterhebern, Fernseher und Bar und anderen höchst individuellen Annehmlichkeiten!

Es gibt immer mehrere Richtungen, in die sich die Perfektion eines technischen Gegenstandes entwickeln kann!

Die Fast-Handshake-Methode ist die schnellste TRANSFER-Ausführung, die die Rechner der Serie 80 ausführen können. Wenn eine derartige Anweisung abgearbeitet wird, dann sind **alle anderen** Tätigkeiten des Rechners blockiert. Sogar die RESET-Taste des Rechners ist ohne Einfluß, bis das Fast-Handshake-TRANSFER beendet ist.

Wie beim Formel-1-Rennwagen müssen Sie auch hier Perfektion in einer bestimmten Richtung mit Einbußen auf anderen Gebieten erkaufen!

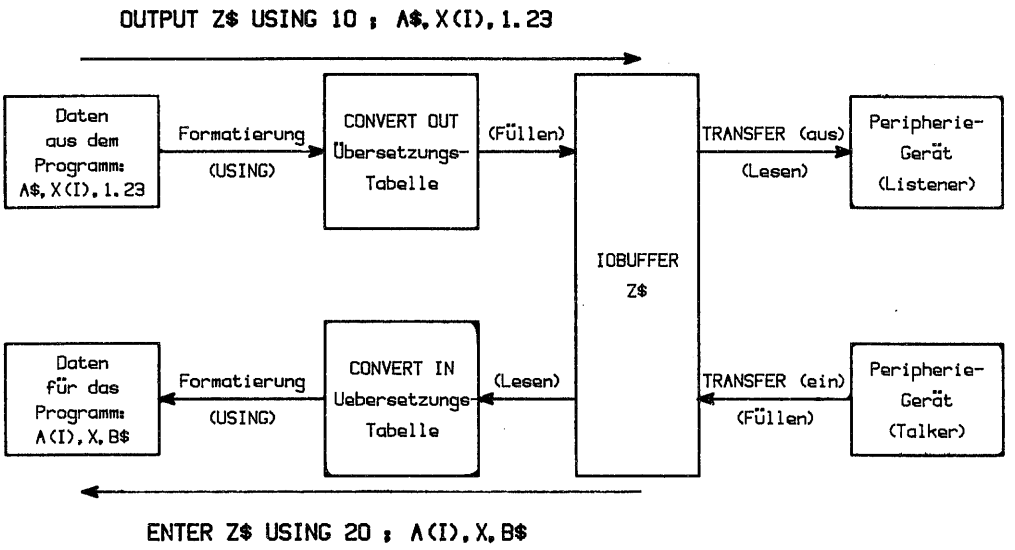
Zweck und Funktion der Buffer

Bei allen TRANSFER-Anweisungen kann es dem Rechner nicht zugemutet werden, die auszugebenden Daten erst zu ermitteln, bzw. die aufzunehmenden Daten gleich den entsprechenden Variablen zuzuweisen. Das Ausgeben bzw. Aufnehmen muß entweder sehr einfach (Interrupt-Methode) oder sehr schnell (Fast-Handshake-Methode) für den Rechner möglich sein.

Aus diesen Gründen verlangen die TRANSFER-Anweisungen die Zwischenspeicherung in einem Buffer. Ein Buffer ist ein Teil des Hauptspeichers, der für die vorübergehende Speicherung auszugebender oder aufzunehmender Daten benutzt wird, wobei es durchaus möglich ist, den gleichen Buffer für ein- und ausgehende Daten gleichzeitig zu nutzen. Darüber entscheidet die Form der TRANSFER-Anweisung.

In einer TRANSFER-Anweisung läßt sich weder eine Formatierung noch eine Konvertierung der Daten unterbringen. Es ist aber sehr wohl möglich, die für die Ausgabe per TRANSFER bestimmten Daten beim Laden in den Buffer zu formatieren und/oder zu konvertieren. Ebenso ist es möglich, die im Buffer aufgenommenen Daten vor der weiteren Verarbeitung im Rechner zu konvertieren und/oder zu formatieren.

Zur Darstellung der Zusammenarbeit von OUTPUT, ENTER, Formatierung, Konvertierung, IOBUFFER und TRANSFER soll das folgende Bild dienen.



Die OUTPUT-Anweisung holt sich die angegebenen Daten aus dem Programm, nimmt gegebenenfalls eine Formatierung und/oder Konvertierung vor und legt die als ASCII-Zeichen dargestellten Daten anschließend im I/O-Buffer an der Stelle ab, die der Füll-Zeiger angibt. Der I/O-Buffer ist voll, wenn der Füll-Zeiger am Ende des Buffer-Strings (in diesem Fall Z$) angekommen ist. Sie sollten sich bei dieser Gelegenheit daran erinnern, daß die normale OUTPUT-Anweisung an das Ende der Daten eine EOL-Sequenz (üblich sind CR und LF) anfügt, die auch Platz im I/O-Buffer beansprucht. Die für die Daten erforderliche Buffergröße muß diesem Umstand Rechnung tragen.

Die TRANSFER-Ausgabe holt sich dann die Zeichen, beginnend an der durch den Lese-Zeiger bestimmten Stelle, aus dem I/O-Buffer und schickt sie über das angegebene Interface zu den daran angeschlossenen Peripherie-Geräten. Ob dies mit der Interrupt- oder Fast-Handshake-Methode erfolgt, bestimmt der Programmierer. Wenn die TRANSFER-Anweisung ausgeführt ist, gibt auch das Interface noch die ihm vorgeschriebene EOL-Sequenz aus.

Die TRANSFER-Eingabe nimmt Zeichen vom angegebenen Interface (und von dem daran angeschlossenen Peripherie-Gerät) entgegen und setzt sie in den I/O-Buffer an die durch den Füll-Zeiger bestimmte Stelle. Auch hier hat der Programmierer zu entscheiden, ob Interrupt oder Fast-Handshake benutzt werden soll.

Die ENTER-Anweisung holt die Daten dann aus dem I/O-Buffer, beginnend an der durch den Lese-Zeiger bestimmten Stelle, und ordnet sie den Variablen zu, wobei eine Konvertierung und/oder Formatierung möglich ist. Die Daten können dann vom Programm weiter verarbeitet werden. Wenn Sie aus einem Buffer Daten abrufen müssen, können Sie mit der Anweisung

```
ENTER Z$ USING "#,#K" ; A$
```

Lesefehler vermeiden, weil diese Form ohne Feld- und Anweisungs-Terminatoren auskommt, die eventuell im Buffer fehlen können.

Die Zeiger

Damit eine String-Variable als I/O-Buffer verwendet werden kann, müssen ihr durch die IOBUFFER-Anweisung vier "Zeiger" zugeordnet werden: Das sind "Füll-Zeiger", "Lese-Zeiger", "Ausgabe-Zeiger" und "Eingabe-Zeiger".

Beim HP-83/85 besetzen diese vier Zeiger acht Bytes der definierten Bufferlänge, wodurch sich der mit Daten besetzbare Platz entsprechend verringert. Bei diesen Rechnern lassen sich beliebig viele I/O-Buffer einrichten.

Beim HP-86/87 kann dagegen die definierte Bufferlänge voll genutzt werden, weil für die Bufferzeiger ein bestimmter Teil des Hauptspeicher fest reserviert ist. Bei diesen Rechnern lassen sich deshalb maximal nur 10 I/O-Buffer einrichten.

Der Füll-Zeiger wird bei Einrichtung des Buffers auf 0 gesetzt. Dieser Zeiger hat immer den Wert, den auch die LEN-Funktion für diese String-Variable ergeben würde. Das Einbringen eines Zeichens in den Buffer läuft so ab:

1. Der Füll-Zeiger wird um 1 erhöht.
2. Das Zeichen wird gespeichert.

Dieser Vorgang läuft bei der Zuweisung (z.B. Z$=Z$&A$), der OUTPUT-Anweisung und der TRANSFER-Eingabe automatisch ab. In den meisten Fällen erübrigt sich also eine Wert-Zuweisung für den Füll-Zeiger.

Der Lese-Zeiger wird beim Einrichten des Buffers auf 1 gesetzt. Die Ausgabe eines Zeichens aus dem Buffer läuft so ab:

1. Das Zeichen wird ausgegeben.
2. Der Lese-Zeiger wird um 1 erhöht.

Dieser Vorgang geschieht automatisch entweder durch eine ENTER-Anweisung oder durch eine TRANSFER-Ausgabe.

Ein Buffer ist "voll", wenn der Füll-Zeiger auf das Ende des für Daten nutzbaren Bereiches zeigt. Jede Füll-Operation, die einen vollen Buffer anspricht, führt zu einer Fehlermeldung.

Ein Buffer ist "ausgelesen", wenn der Lese-Zeiger einen Wert hat, der um 1 über dem Wert des Füll-Zeigers liegt. Das braucht nicht der vollen Länge des Buffers zu entsprechen! Wenn ein Buffer ausgelesen ist, wird automatisch der Füll-Zeiger auf 0 und der Lese-Zeiger auf 1 gesetzt. Das ist der gleiche Zustand, der auch bei Ausführung der IOBUFFER-Anweisung eintrat, nur mit dem Unterschied, daß die IOBUFFER-Anweisung auch noch die Zeiger für die Übersetzungstabellen auf den Startwert setzt (löscht).

Bei den Rechnern HP-83/85 ist die für Daten nutzbare Länge um 8 kleiner, als der definierte Umfang der String-Variablen. Für diese Rechner ist also eine nur 8 Bytes umfassende String-Variable als I/O-Buffer ungeeignet, weil dieser Buffer schon "voll" ist, bevor auch nur ein Zeichen in ihm untergebracht wurde. Gleichzeitig wäre dieser Buffer aber auch "ausgelesen", weil er ja keinen Inhalt hat!

Die Buffer-Aktivitäten

Nur wenn eine TRANSFER-Anweisung ausgeführt wird, gilt der angesprochene Buffer als "aktiv". Der Buffer kann "ausgeben" und/oder "aufnehmen" (auch gleichzeitig). Der Wert von Ausgabe- und Eingabe-Zeiger wird über die TRANSFER-Anweisung(en) gesteuert: Wenn der Buffer aktiv ist, ist der jeweilige Zeiger auf den Auswahl-Code des zugeordneten Interfaces gesetzt. Bei inaktivem Zustand hat der entsprechende Zeiger den Wert 0. Wenn z.B. folgende Interrupt-TRANSFER-Ausgabe zum Interface 6 ausgeführt wird,

```
TRANSFER Z$ TO 6 INTR
```

dann nimmt der Ausgabe-Zeiger für die Dauer dieser Operation den Wert 6 an.

Ein Buffer wird erst mit Beendigung der TRANSFER-Anweisung wieder inaktiv. Auch dies erfolgt für beide Richtungen unabhängig voneinander. Die Abfrage der jeweiligen Zeiger gibt jederzeit die nötigen Informationen über Zustand und Aktivität des I/O-Buffers.

Zustand und Steuerung des Buffers

Die vier Buffer-Zeiger können durch die STATUS-Anweisung abgefragt werden. Es handelt sich dabei um die nachstehend angegebenen Register:

Status-Register:	Start-Wert:	Register-Funktion:	Anweisung zum Lesen der Register des Buffers Z$:
SR0	1	Lese-Zeiger	STATUS Z$,0;T0
SR1	0	Füll-Zeiger	STATUS Z$,1;T1
SR2	0	Eingabe-Zeiger	STATUS Z$,2;T2
SR3	0	Ausgabe-Zeiger	STATUS Z$,3;T3

Diese Register können jederzeit gelesen werden, egal ob der entsprechende Buffer aktiv ist oder nicht. Wird jedoch versucht, diese Daten für eine gewöhnliche String-Variable zu ermitteln, die nicht zuvor mit der IOBUFFER-Anweisung deklariert wurde, dann erfolgt eine Fehlermeldung.

Es folgt jetzt ein Beispiel, bei dem die Status-Register des I/O-Buffers zur Steuerung des Programms benutzt werden. In diesem Beispiel wird zuerst eine String-Variable dimensioniert, um später für 80 Zeichen als Buffer zu dienen und anschließend zum I/O-Buffer erklärt. Die Daten werden dann mit OUTPUT in den Buffer gesetzt und von dort wird mit TRANSFER an einen HP-IB-Drucker gegeben. Dabei wird laufend der Ausgabe-Zeiger geprüft, um festzustellen, wann die TRANSFER-Anweisung beendet ist. Das Programm zeigt dann das Ende an.

```
 10 DIM Z$[88]  (für HP-83/85, bei HP-86/87 reicht [80])
 20 IOBUFFER Z$ ! Nutzbare Laenge von Z$ sind 80 Zeichen
 30 FOR I=0 TO 1 STEP .1
 40 OUTPUT Z$ USING "#,D.DDD,X" ; SIN(I)
 50 NEXT I
 60 ! Zeigt die Werte der Buffer-Register
 70 STATUS Z$,0 ; T0,T1,T2,T3
 80 PRINT "Register vor TRANSFER ",T0,T1,T2,T3
 90 TRANSFER Z$ TO 701 INTR
100 STATUS Z$,0 ; T0,T1,T2,T3
110 DISP T0,T1,T2,T3
120 IF T3#0 THEN GOTO 100
130 PRINT "TRANSFER beendet!"
140 END
```

Die Variable T0 zeigt an, wieviel Zeichen bereits aus dem Buffer ausgelesen wurden. T1 zeigt die Gesamtzahl der Zeichen im Buffer, und T3 zeigt an, zu welchem Interface die Daten übertragen werden.

Für ein anderes Beispiel setzen wir voraus, daß ein Gerät X drei numerische Werte, gefolgt von einer CR-LF-Sequenz übermitteln will. Das folgende Programm zeigt die Buffer-Register an, bis die TRANSFER-Anweisung beendet ist. Danach wird ein ENTER ausgeführt, um die Werte aus dem Buffer zu holen und sie anschließend auszudrucken:

```
 10 ! Unser Geraet X hat die HP-IB-Adresse 721
 20 DIM Z$[88]  (für HP-83/85, bei HP-86/87 reicht [80])
 30 IOBUFFER Z$
 40 TRANSFER 721 TO Z$ INTR ; DELIM 10
 50 ! 10 ist das Dezimalaequivalent fuer Line-Feed
 60 STATUS Z$,0 ; T0,T1,T2,T3
 70 DISP T0,T1,T2,T3
 80 IF T3#0 THEN GOTO 60
 90 ENTER Z$ ; X,Y,Z
100 DISP X,Y,Z
110 END
```

Wenn das Gerät X sehr langsam ist, können Sie an der Variablen T1 "überwachen", wie die Zeichen in den Buffer einlaufen. T0 wird nicht geändert, bis das ENTER ausgeführt wurde und T2 zeigt die Aktivität während des TRANSFER-Ablaufes an. Beim Beginn wird T2=7 gesetzt und nach Beendigung wieder auf T2=0 zurückgesetzt.

Lese- und Füll-Zeiger des Buffers können mit CONTROL-Anweisungen beeinflußt werden. Das eröffnet z.B. die Möglichkeit, bestimmte Daten wieder und wieder auszugeben, ohne diese jedesmal neu errechnen zu müssen. Die folgende Tabelle zeigt die Zuordnung der betreffenden Register und ein Beispiel für den Zugang zu ihnen:

Steuer-Register:	Start-Wert:	Register-Funktion:	Anweisung zum Setzen der Register des Buffers Z$:
CR0	1	Lese-Zeiger	CONTROL Z$,0;V0
CR1	0	Füll-Zeiger	CONTROL Z$,1;V1

Diese Register können jederzeit mit neuen Werten besetzt werden. Wird jedoch versucht, diese Register für eine String-Variable zu setzen, die nicht zuvor mit der IOBUFFER-Anweisung deklariert wurde, dann erfolgt eine Fehlermeldung.

Im folgenden Beispiel werden 360 Werte von SIN(X) errechnet, um einen kompletten Zyklus der Sinuskurve darzustellen. Diese Werte werden an den Buffer Z$ gegeben und anschließend zu einem Gerät X (einem Digital-Analog-Wandler) geschickt, und zwar durch Fast-Handshake-TRANSFER.

Wenn das FHS-TRANSFER erledigt ist, wird der Lese-Zeiger automatisch auf 1 gesetzt, der Füll-Zeiger aber wird willkürlich auf 360 gebracht, damit das TRANSFER von vorn beginnen kann. Dies wird endlos wiederholt, bis das Programm mit der 'PAUSE'-Taste gestoppt wird. Der Effekt dieses Programms ist die Erzeugung einer beinahe kontinuierlichen Sinus-Spannung durch das Gerät X. (Einzelheiten über das Gerät X sind hier weggelassen, um Sie nicht zu verwirren).

```
 10 ! Platz schaffen fuer 360 8-Bit-Werte und die Zeiger
 20 DIM Z$[368]  (für HP-83/85, bei HP-86/87 reicht [360])
 30 IOBUFFER Z$
 40 DEG
 50 FOR I=0 TO 359
 60 OUTPUT Z$ USING "#,B" ; SIN(I)*255
 70 DISP I;@ NEXT I
 80 TRANSFER Z$ TO 5 FHS
 90 CONTROL Z$,1 ; 360
100 GOTO 90
110 END
```

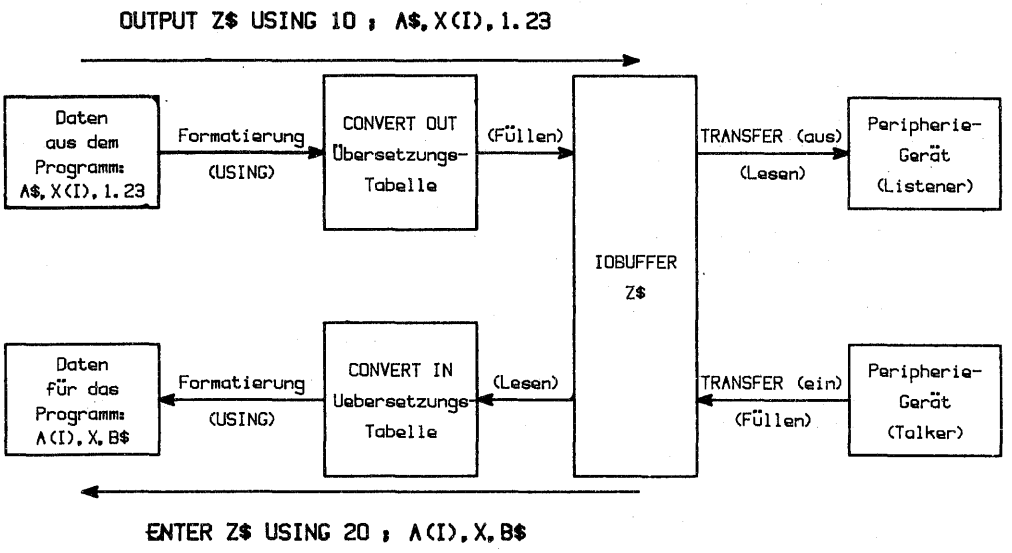
Die Zeilen 10 bis 40 definieren und initialisieren den Buffer und versetzen den Rechner in den DEG-Modus. Die Zeilen 50 bis 80 füllen den Buffer mit den 8-Bit-Werten für jedes der 360 Argumente (also eine komplette Sinus-Schwingung), und Zeile 90 startet dann die TRANSFER-Anweisung. Die Zeile 100 wird ausgeführt, wenn das TRANSFER beendet ist. Der Füll-Zeiger wird auf 360 gesetzt, damit der Buffer wieder "voll" erscheint und das TRANSFER erneut begonnen.

Mit der Steuerung über den Füll- und Lese-Zeiger ist es möglich, Daten-Ausgaben zu wiederholen, nur Teile der Gesamt-Daten auszugeben, Daten in jeden gewünschten Abschnitt des Buffers zu schreiben, Daten aus jedem Teil des Buffers auszulesen usw. Diese Operationen werden nicht die einzigen sein, die Sie für Ihre Anwendungen brauchen, die Flexibilität aber, mit der diese Dinge getan werden können, wird Sie davon überzeugen, daß bei den I/O-Operationen fast nichts unmöglich ist.

Datenübertragung mit TRANSFER-Anweisungen

Bis zu diesem Kapitel ist die TRANSFER-Anweisung schon einigemal erwähnt worden. Mit dieser Anweisung und der damit verbundenen IOBUFFER-Anweisung können Sie nun Programme gestalten, die für den Daten-Austausch mit Peripherie-Geräten optimal arbeiten, weil der Rechner, auch während des Ablaufs der Daten-Übergabe, fast ununterbrochen für andere Aufgaben verwendbar ist.

Das folgende Diagramm zeigt die Beziehungen der TRANSFER-Anweisung zu den Programm-Variablen, den Übersetzungstabellen und dem I/O-Buffer.



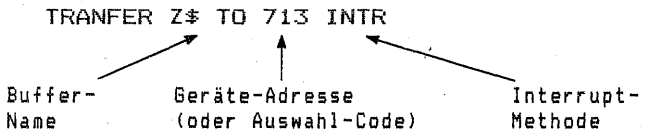
Vom gesamten Vorgang ist der Datentransport selbst am leichtesten zu verstehen:

Die TRANSFER-Ausgabe nimmt, vereinfacht dargestellt, an der durch den Lese-Zeiger markierten Stelle Bytes aus dem Buffer und schickt diese an das Peripherie-Gerät.

Entsprechend werden bei der TRANSFER-Eingabe Zeichen vom Peripherie-Gerät angenommen und an der durch den Füll-Zeiger markierten Stelle im Buffer abgelegt.

TRANSFER-Ausgabe

Für ein Ausgabe-TRANSFER brauchen Sie nur anzugeben, ob die Datenübermittlung mit "INTerRupt" oder "Fast-HandShake" erfolgen soll:



Eine für die Interrupt-Methode programmierte TRANSFER-Ausgabe läuft wie folgt ab:

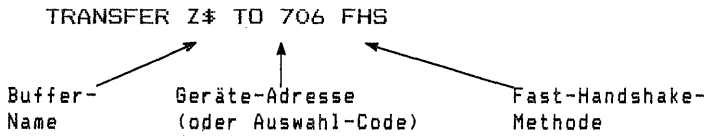
Mit Ausführungsbeginn der Anweisung erhält das angesprochene Gerät (hier das Gerät 13 am HP-IB-Interface mit Auswahl-Code 7) die Berechtigung, den Rechner über das Interface zu "unterbrechen" (interrupt), wenn das Gerät bereit ist, ein neues Zeichen anzunehmen. (Das ist ein durch Hardware bedingter Interrupt. Die durch Software auszulösenden Interrupts werden im I/O-ROM-Abschnitt 9 beschrieben.)

Der Rechner muß, immer wenn ihn das Interface unterbrechen möchte, den Programmablauf so lange anhalten, bis das nächste Zeichen aus dem Buffer an das Interface übergeben wurde. (Das Interface ist danach allein für die einwandfreie Übermittlung an das Peripherie-Gerät zuständig.)

Wenn der Buffer restlos ausgelesen ist (siehe Seite ROM-52 und folgende), sperrt der Rechner dem Interface die Möglichkeit ihn zu "unterbrechen". Die TRANSFER-Ausgabe ist damit im wesentlichen beendet, auch wenn das Interface von sich aus noch die vereinbarte EOL-Sequenz senden muß, die normalerweise aus Wagenrücklauf-und Zeilenvorschub-Symbol (CR + LF) besteht. (Hierzu wird auf die entsprechenden Abschnitte der Interface-Beschreibungen verwiesen).

Wenn die TRANSFER-Ausgabe abgeschlossen ist, prüft der Rechner, ob eine vom Benutzer gewünschte Programm-Verzweigung nach der TRANSFER-Ausgabe auszuführen ist, die dann unmittelbar nach Beendigen der TRANSFER-Anweisung beginnt. Mehr hierüber finden Sie im I/O-ROM-Abschnitt 9.

Für eine nach der Fast-Handshake-Methode auszuführende TRANSFER-Ausgabe kann die Anweisung so aussehen:



Eine für Fast-Handshake programmierte TRANSFER-Ausgabe läuft in einigen Punkten anders ab, als bei der Interrupt-Methode geschildert:

Der Rechner entscheidet bei Erreichen einer derartigen Anweisung selbst, daß er jetzt alle Zeichen aus dem Buffer ins Interface übertragen muß. Der Rechner widmet sich ausschließlich dieser einen Aufgabe, bis der Buffer leer ist. In dieser Zeit werden keinerlei andere Unterbrechungen beachtet, nicht einmal die RESET-Taste ist jetzt wirksam! Für den Rechner hat das Fast-Handshake-TRANSFER unbedingten Vorrang vor allen anderen Aufgaben.

Wenn der Buffer ausgelesen ist, ist die TRANSFER-Ausgabe im wesentlichen beendet, nur das Interface kann danach noch mit der Ausgabe der EOL-Sequenz beschäftigt sein. Wenn eine Programm-Verzweigung nach der TRANSFER-Anweisung vorgesehen ist, wird diese unmittelbar danach ausgeführt.

TRANSFER-Eingabe

Die TRANSFER-Eingabe gleicht im wesentlichen der TRANSFER-Ausgabe, nur daß hier Daten von dem Peripherie-Gerät in den Rechner übertragen werden. Zusätzlich zur Entscheidung zwischen Interrupt- oder Fast-Handshake-Methode ist hier anzugeben, welche Bedingungen die TRANSFER-Anweisung zur Erfüllung benötigt. Die folgenden Verfahren lassen sich für INTR- und FHS-Methode benutzen:

COUNT –

Es ist die Zahl der Zeichen oder Bytes anzugeben, nach deren Eintreffen die TRANSFER-Eingabe als erfüllt gilt. COUNT läßt sich dann verwenden, wenn die Zahl der erwarteten Zeichen oder Bytes bekannt ist.

EOI –

Das ist eine vom Interface abhängige Abschlußbedingung. Beim HP-IB-System setzen einige Geräte bei der Ausgabe des letzten Daten-Bytes die EOI-Leitung auf "wahr", beim HP-IL-System kann das letzte Byte als END-Byte ausgegeben werden. Wenn dies als Eingabe-Ende vereinbart wurde, wird beim Erkennen des vereinbarten Signals die Eingabe abgeschlossen.

Standard-Bedingung –

Hier wird die Eingabe abgeschlossen, wenn der Buffer gefüllt ist. Wie leicht einzusehen ist, kann ein voller I/O-Buffer keine Daten mehr aufnehmen, weshalb die TRANSFER-Eingabe abgebrochen wird. Diese Abschlußbedingung braucht nicht besonders programmiert zu werden.

Eine TRANSFER-Eingabe wird beendet, sobald eine der vorstehend angegebenen Bedingungen eintritt.

Für die INTR-Methode sind noch weitere Bedingungen für die Beendigung zulässig:

DELIM –

Es wird der numerische Wert des Daten-Bytes angegeben, das an letzter Stelle eintrifft. Dieser Abschluß läßt sich gut verwenden, wenn die Zahl der erwarteten Zeichen entweder nicht bekannt ist, oder von Fall zu Fall wechselt. Das als Abschluß verabredete Zeichen darf natürlich nicht innerhalb der Daten vorkommen, weil dann die TRANSFER-Eingabe zu früh beendet wird!

Interface-Abschluß –

Das ist eine Interface-abhängige Schluß-Bedingung. Bestimmte Interfaces erlauben es, zusätzliche Abschluß-Bedingungen in die Steuer-Register zu setzen. Diese Abschluß-Bedingungen können entweder durch bestimmte vereinbarte Zeichen-Kombinationen oder auch durch den Schaltzustand einzelner Leitungen ausgelöst werden. Lesen Sie das zu Ihrem Interface gehörende Handbuch, um auch diese Möglichkeiten richtig nutzen zu können!

Das Programmieren mit der TRANSFER-Anweisung

Jede TRANSFER-Anweisung setzt einen I/O-Buffer voraus, der vorher als String ausreichender Größe definiert und mit IOBUFFER deklariert sein muß. Nach der Ausführung der IOBUFFER-Anweisung kann auch jede CONVERT-Anweisung für diesen Buffer durchgeführt werden. Diese Reihenfolge muß eingehalten werden, da die IOBUFFER-Anweisung unter anderem auch die Zeiger für die Übersetzungstabelle auf feste Anfangswerte setzt.

Wenn es nötig ist, ein Interface zu initialisieren, dann muß das geschehen, bevor eine TRANSFER-Anweisung ausgeführt wird. Wenn Ihr Programm "Verzweigungen am Ende der Programmzeile" benutzt (siehe Seite ROM-66), dann müssen auch diese Anweisungen dem Rechner vor der ersten TRANSFER-Anweisung bekanntgegeben worden sein.

Eine TRANSFER-Eingabe kann an jeder Stelle des Programms ausgeführt werden, wenn die eben beschriebenen Anweisungen eingesetzt wurden. Eine TRANSFER-Ausgabe setzt dagegen voraus, daß Daten im Buffer sind, weil sonst nichts auszugeben ist! Deshalb ist eine OUTPUT-Anweisung oder die Zuordnung eines Strings notwendig, bevor man daran geht, eine TRANSFER-Ausgabe zu veranlassen.

Diese Forderung schießt jedoch über das Ziel hinaus: es ist nämlich nicht nötig, daß bereits alle Daten im Buffer stehen, bevor man eine mit der Interrupt-Methode arbeitende TRANSFER-Ausgabe startet! Ebenso ist es unnötig, auf das Ende einer mit der Interrupt-Methode arbeitenden TRANSFER-Eingabe zu warten, bevor man Daten aus dem Buffer in das Programm übernimmt. (Für die rein sequentielle FHS-Methode gelten die folgenden überlegungen nicht!)

Wenn ein Peripherie-Gerät nennenswert langsamer arbeitet als der Rechner, darf die TRANSFER-Ausgabe bereits beginnen, wenn der Buffer nur zum Teil gefüllt ist. Wenn dann weitere Daten zur Verfügung stehen, die auch an die Peripherie gegeben werden sollen, kann man diese Daten mit der OUTPUT-Anweisung in den Buffer bringen, während die TRANSFER-Ausgabe läuft. Dieser Vorgang kann unbegrenzt fortgesetzt werden bis entweder

1. alle Daten ausgegeben sind oder
2. der Buffer voll ist (dann muss das Programm warten, bis wieder Platz im Buffer ist!) oder
3. bis der Buffer leer ist (dann wird die TRANSFER-Ausgabe beendet und muß neu gestartet werden!)

Der Fall 1 ist trivial und erfordert keine überwachung. Fall 2 ergibt sich aus dem Buffer-Status, und das Programm erfährt durch das Abfragen von S1, wann es weitergehen kann. Das Auftreten von Fall 3 beweist, daß das Peripherie-Gerät zumindest im Augenblick schneller als der Rechner ist. (Das mag an einem langwierigen Rechengang oder anderen Verzögerungen im Programm liegen.) Eine "Verzweigung am Ende der Programmzeile" oder eine Prüfung des Buffer-Status zeigt dann das Ende der TRANSFER-Ausgabe an. Es kann dann eine Entscheidung getroffen werden, die TRANSFER-Ausgabe fortzusetzen. Das folgende Programm-Beispiel zeigt, wie diese Bedingungen sich bemerkbar machen:

```
 10 ! Dieses Programm kann Fall 1, 2 u. 3 unterscheiden
 20 ! Die Buffergroesse wurde ausprobiert
 30 ! um moeglichst wenig Speicherplatz zu belegen
 40 DEG @ DIM B$[160]
 50 DEF FNA(X) = INT(SIN(X)*35+36.5)
 60 CONTROL 7,16 ; 0 ! Sperrt CR-LF-Sequenz im Interface
 70 IOBUFFER B$
 80 FOR X=0 TO 359 STEP 10 ! FOR-NEXT-Schleife fuer Fall 1
 90 STATUS B$,1 ; S1,S2,S3,S4
100 ! Testet auf leeren Buffer (Fall 2)
101 Z=FNA(X)
110 IF S1+Z>145 THEN GOTO 90
115 I$=VAL$(Z)&"X,A"
120 OUTPUT B$ USING I$ ; "*"
130 ! Prueft Ausgabe-Zeiger, bevor TRANSFER gestartet wird
140 IF S4=0 THEN TRANSFER B$ TO 720 INTR ! Fall 3
150 NEXT X
160 !
170 GOTO 80
180 END
```

Die Zeilen-End-Verzweigung

Einiges über Interrupts

Wenn Sie den Ausdruck "Interrupt" im Zusammenhang mit dem Rechner noch nie gehört haben, sollten Sie darunter "Unterbrechung" verstehen. Denken Sie bitte an ein Telefon, das während ihrer Arbeit läutet: Sie müssen dann Ihre Arbeit unterbrechen und erst den Anruf erledigen. Danach wenden Sie sich wieder der Arbeit zu, bei der man Sie "unterbrochen" hat.

Wenn Sie als "1-Mann-Abteilung" arbeiten und technisch findig sind, bauen Sie einen Schalter in die Telefonleitung und sind dann, nach ihrem Wunsch, erreichbar oder nicht (der daneben abgelegte Hörer fiel mehr auf!). Leider wirkt diese Sperre für alle Anrufe, auch die Ihnen genehmen Partner können Sie nicht erreichen. Wenn Sie aber schon über eine Sekretärin verfügen, geht es komfortabler: Diese Dame wird dann nach Ihren Weisungen entscheiden, welcher Anruf für Sie wichtig ist. Auch der Rechner hat Möglichkeiten, auf Störungen zu reagieren oder diese zu ignorieren. Jede "Störung" im bisherigen Sinn ist von nun an ein "Interrupt".

Der Rechner unterteilt die ihn betreffenden Interrupts in zwei Gruppen:

1. Hardware-Interrupts (durch Geräte ausgelöst) und
2. Software-Interrupts (durch den Programm-Ablauf ausgelöst)

Hardware-Interrupts braucht der Rechner für komplizierte aber reguläre Programmabläufe. Sie sind meist gut zu verstehen und machen viele Dinge, wie z.B. den Interrupt-TRANSFER, überhaupt erst möglich. Denn hierbei stehen Programmablauf und Datenübermittlung in Konkurrenz. Der Rechner muß dabei entscheiden. Aber auch extern oder intern zufällig eintretende Dinge können einen Hardware-Interrupt auslösen. Der Rechner braucht dann besondere Routinen zur Bewältigung dieser Probleme. Weil es aber viele Gründe gibt, die einen Interrupt rechtfertigen, muß auch der Programmierer festlegen können, wann ein besonderer Grund für einen Interrupt vorliegt, der spezielle Lösungen erfordert. Hierzu dienen die Zeilen-End-Verzweigungen, die man Software-Interrupts nennt.

Stellen Sie sich am Ende jeder Programmzeile ein (unsichtbares) "IF - THEN" vor:

```
50 PRINT "Ergebnis=";X @ IF<wenn XXXX passiert>THEN
   GOSUB<Spezial-Routine>
60 X=X+Y @ IF<wenn XXXX passiert>THEN
   GOSUB<Spezial-Routine>
```

Dann wird nach Ausführung jeder Programmzeile geprüft, ob XXXX eingetreten ist, und dann die Konsequenz gezogen. XXXX kann nun ein ERROR, ein Hardware-Interrupt, eine Meldung über das Ende eines TRANSFER-Vorganges, die Betätigung einer Taste oder die Überschreitung eines Termins sein. Für drei dieser Fälle enthält der Rechner bereits fertige Routinen: Error-Meldungen, Spezial-Funktions-Tasten und Timer. Wenn eines dieser Ereignisse eintritt, kann eine besondere Routine angesteuert werden. Der Programmierer hat dabei auch die Möglichkeit, die einzelnen Kriterien für die Zeilen-End-Verzweigung nach Bedarf ein- und auszuschalten. Das ist der Inhalt der nächsten Kapitel.

Programmieren mit der Zeilen-End-Verzweigung

Einführung

Dieser Abschnitt beschreibt Programmiertechniken, DIE mit Zeilen-End-Verzweigungen Service-Routinen ansteuern, um damit Ereignisse zu verarbeiten, die vom I/O-ROM ausgelöst wurden. Eine Betrachtung über die Rangordnung der einzelnen Interrupt-Ursachen und die Zweckmäßigkeit von GOTO- oder GOSUB-Service-Routinen soll Ihnen Hinweise geben, wie Sie Ihre Spezialprobleme lösen können.

Interrupts von den Interfaces

In jedem Interface (HP-IB, HP-IL, RS-232, BCD usw.) können im Steuer-Register CR1 die Bedingungen festgelegt werden, die zu einem Interrupt führen sollen. CR1 wird deshalb auch "Interrupt-Masken-Register" genannt. Wenn Sie in diesem Register ein bestimmtes Bit setzen, dann darf das Interface den Rechner "unterbrechen", wenn das zu diesem Bit gehörende Ereignis eingetreten ist. Die Zuordnung der Bits zu den Ereignissen finden Sie in der Beschreibung, die zu jedem Interface gehört. So ist z.B. beim HP-IB-Interface ein "Service Request"-Interrupt vorgesehen (kurz "SRQ" genannt). Dieser Interrupt ist nur möglich, wenn das Bit 3 (numerisch: 8) vom CR1 gesetzt ist. Um diese Voraussetzung für das Interface mit dem Auswahl-Code 7 zu erreichen, ist folgende Anweisung nötig:

```
ENABLE INTR 7 ; 8
```

Den gleichen Effekt hat auch die Anweisung

```
CONTROL 7,1 ; 8
```

weil beide Anweisungen das Bit 3 des Steuer-Registers 1 vom Interface 7 setzen.

Wir nehmen jetzt der Einfachheit wegen an, daß nur ein Gerät am Interface 7 angeschlossen ist und daß dieses Gerät SRQ nur dazu verwendet, um seine Bereitschaft zur Ausgabe eines numerischen Wertes anzuzeigen. Eine Anweisung zum Lesen dieses Wertes kann die folgende Form haben:

```
ENTER 701 ; V1
```

Das ist in unserem Beispiel die ganze "Routine", die wir jetzt noch in ein Programm einbauen müssen.

Sobald der Rechner den neuen Wert gelesen hat, muß er nämlich für den nächsten SRQ-Interrupt wieder ansprechbar sein, weil sonst das Peripherie-Gerät den Rechner nicht mehr "unterbrechen" kann. Unsere Service-Routine nimmt Gestalt an:

```
1000 ! SRQ-Service-Routine fuer Auswahl-Code 7
1010 STATUS 7,1 ; A ! Interruptregister wird gelesen!
1020 ENTER 701 ; V1
1030 ! Setzt Interrupt-Moeglichkeit neu
1040 ENABLE INTR 7 ; 8 @ RETURN
```

Beachten Sie bitte, daß die ENABLE- und die RETURN-Anweisung in einer einzigen Zeile stehen! In einigen Fällen ist das unbedingt nötig! Die Gründe hierfür werden auf den Seiten ROM-67 bis ROM-70 ausführlich erklärt.

Nun muß noch der Sprung zur Service-Routine programmiert werden, wobei zu entscheiden ist, ob das als Subroutine (GOSUB) oder als Programm-Segment (GOTO) ausgeführt werden soll. Darüber entscheidet die ON INTR-Anweisung. Es ist klar, daß der Rechner bereits vor Eintritt des Ereignisses wissen muß, was er tun soll und wo er die genauen Instruktionen findet, denn sonst passiert überhaupt nichts!

Deshalb muß die ON INTR-Anweisung vom Programm bereits bearbeitet sein, bevor die ENABLE INTR-Anweisung erreicht wird, so wie das hier im Beispiel zu sehen ist:

```
10 ON INTR 7 GOSUB 1000
20 OUTPUT 722 ;"F1R7T2T3D1"
30 V1,X=0
40 LOCAL 722! Manuelle Einstellung!
50 ENABLE INTR 7;8
60 ! Hier folgt irgendeine Programmfortsetzung
70 X=X+1 @ DISP X,V1 !! Warte-Schleife
80 WAIT 100 @ GOTO 70 ! im Programm!
   .
   .
1000 ! SRQ-Service-Routine fuer Auswahl-Code 7
1010 STATUS 7,1 ; A ! Interruptregister SR1 wird
1020 ENTER 722 ; V1 ! gelesen und dabei gelöscht!
1030 ! Interrupt-Moeglichkeit wird erneuert!
1040 ENABLE INTR 7;8 @ RETURN
1050 END
```

Zeile 10 nennt die Programmstelle (1000) und die Art (GOSUB, nicht GOTO) der Service-Routine für den Auswahl-Code 7. Das Voltmeter HP-3455A wird mit Zeile 20 zum Messen nach Triggerung initialisiert. Die LOCAL-Anweisung ermöglicht es, diese Triggerung am Gerät von Hand vorzunehmen. Die ENABLE-Anweisung bereitet das Interface auf einen SRQ-Interrupt vor (Bit 3 der Interrupt-Maske). Die Zeilen 70 und 80 enthalten nur eine Zählgröße und zeigen den vom Voltmeter zuletzt gemessenen Wert im Abstand von 100 Millisekunden immer wieder an.

Jedesmal, wenn ein SRQ eintrifft, springt das Programm aus der Schleife (Zeilen 70 und 80) zur Subroutine in Zeile 1000. Hier wird die ENTER-Anweisung ausgeführt, die den neuen Meßwert des Voltmeters in den Rechner bringt. In Zeile 1040 wird dann das SRQ erneut zugelassen. Dann kehrt das Programm durch das RETURN in den normalen Ablauf zurück und zeigt den neuen Spannungswert so lange an, bis wegen eines aktuelleren Wertes vom Voltmeter erneut SRQ ausgegeben wird.

Was würde nun geschehen, wenn man die Zeile 1040 mit der ENABLE INTR-Anweisung "vergessen" hätte? Sehr einfach! Beim ersten SRQ läuft der Vorgang ab, wie oben beschrieben. Die nächsten SRQ-Interrupts hätten aber keinerlei Wirkung mehr, auch wenn diese vom Voltmeter noch so oft wiederholt werden! Ohne das richtige Bit im CR1 interessiert sich das Interface überhaupt nicht dafür, egal, wie wichtig das SRQ auch genommen werden müßte!

Die mit der ON INTR-Anweisung festgelegten Bedingungen für Zeilen-End-Verzweigungen werden durch eine OFF INTR-Anweisung mit Angabe des Interface-Auswahl-Codes aufgehoben. Dieses Interface kann dann keine Zeilen-End-Verzweigungen mehr auslösen, bis es durch eine ON INTR-Anweisung erneut dazu wieder ermächtigt wird.

Es zeugt von einem guten Programmier-Stil, wenn das Status-Register jedesmal nach einem Interrupt gelesen wird, weil die übernahme des Register-Inhalts in eine Variable zum einen dieses Register löscht und damit für neu auftretende Interrupts frei macht, andererseits in diesem Register mehrere Interrupt-Gründe ihre Spuren hinterlassen haben können und man dieses Register, wie gesagt, nur einmal abfragen kann. Die weitere Untersuchung muß dann an der zugeordneten Variablen erfolgen (A in unserem Beispiel).

Interrupts wegen Zeit-Überschreitung

Bei Peripherie-Geräten ist es möglich, daß sie zu nicht vorhersehbaren Zeitpunkten ihre Dienste ganz oder teilweise aufkündigen. Das kann durch innere Fehler in den Geräten, aber auch durch Unachtsamkeit der Benutzer hervorgerufen werden (z.B. Netz ausgeschaltet, Batterie leer o.ä.). Für uns ist es unwichtig, warum diese Ausfälle auftreten, uns stört nur, daß es solche Pannen überhaupt gibt, weil dadurch unsere Programme empfindlich gestört werden.

Wenn nämlich ein Gerät ausfällt, kann es mit dem Rechner keinen Datenaustausch per "Handshake" mehr vornehmen (Handshake ist bekanntlich ein Verfahren zur Übergabe von Daten zwischen Geräten, bei denen der "Sender" Daten als "gültig" bezeichnet, deren Entgegennahme von den "Empfängern" bestätigt wird). Bei der Bearbeitung von ENTER-, SEND- oder OUTPUT-Anweisungen führt eine Störung zu einer Unterbrechung des "Handshake"-Protokolls, die übertragung der Daten stockt, die Anweisung kann also nicht beendet werden und das Programm bleibt so lange "hängen", bis der Benutzer endlich merkt, daß etwas schief gegangen sein muß. Das ist aber für die meisten I/O-Anwendungen besonders dann unzumutbar, wenn solche Pannen häufiger auftreten.

Mit der SET TIMEOUT-Anweisung läßt sich nun für Handshake-Operationen eine Maximaldauer festlegen. Wenn ein Interface diese Zeitgrenzen überschreitet, kann zwischen zwei verschiedenen Verfahren zur Behebung der Störung gewählt werden. Die einfachste Maßnahme besteht aus dem Abbruch des Datenaustauschs und Übergang zur nächsten Programmzeile. Die andere Maßnahme sieht eine Service-Routine für Zeitüberschreitungen vor, wenn deshalb eine I/O-Operation abgebrochen werden mußte.

Die einfache Methode zeigt dieses Beispiel einer SET TIMEOUT-Anweisung:

```
10 SET TIMEOUT 7 ; 10000
20 OUTPUT 706 ;"Einfacher Daten-Test"
30 DISP "Bei Zeile 30 angekommen!"
40 END
```

Wenn das Gerät 706 das Handshake unterbricht, bevor die Übertragung fertig ist, wartet der Rechner 10 Sekunden (gerechnet vom Beginn der übertragung) und zeigt dann den Text der Zeile 30 an. Allerdings wird diese Zeile auch angezeigt, wenn die Übertragung störungsfrei beendet wurde!

Man kann aber auch raffinierter vorgehen, wenn eine I/O-Operation wegen Zeitüberschreitung abgebrochen wurde: Eine getrennte Service-Routine für jeden Auswahl-Code kann die spezifischen Eigenschaften der angeschlossenen Geräte berücksichtigen. Das soll im nächsten Beispiel deutlich werden:

```
10 SET TIMEOUT 7 ; 2000
20 SET TIMEOUT 5 ; 4000
30 ON TIMEOUT 7 GOSUB 1000
40 ON TIMEOUT 5 GOSUB 1100
50 ENTER 706 ; V1,X
60 OUTPUT 5 ; V1,X
70 GOTO 50
   .
   .
```

```
      .
      .
      .
      .
      .
1000 ! Service für Auswahl-Code 7
1010 PRINT "Zeitfehler im HP-IB!"
1020 RESET 7 ! Erneuter Versuch!
1030 RETURN
1100 ! Service für Auswahl-Code 5
1110 PRINT "Zeitfehler im Interface 5!"
1120 RESET 5 ! Erneuter Versuch!
1130 RETURN
```

Offensichtlich kann man also für ein fest umrissenes System beim Auftreten einer Zeitüberschreitung viel mehr auslösen als eine simple Störungsmeldung mit Abbruch der begonnenen Operation, RESET und erneutem Versuch, die gestörte Operation doch noch ausführen zu lassen: Beim ersten Auftreten einer Zeitüberschreitung wird ein Flag gesetzt, das vor der erneuten Ausführung der gestörten I/O-Operation abgefragt wird. Über dieses Flag kann dann z.B. angezeigt werden, daß der Drucker kein Papier mehr hat oder ein Speichermedium bereits alle Daten geliefert hat. Die Wiederholung der I/O-Operation ist in diesen Fällen ja sinnlos. Das folgende Beispiel zeigt dieses Verfahren:

```
10 SET TIMEOUT 6 ; 1500
20 ON TIMEOUT 6 GOSUB 1000
30 F1=0 @ DIM V(50) ! Zeitfehler-Flag loeschen!
40 FOR I=1 TO 50
50 IF F1#0 THEN GOTO 80 ! Flag-Test vor dem ENTER
60 ENTER 6 ; V(I)
70 NEXT I
80 PRINT I-1;"Werte empfangen" @ STOP
     .
     .
     .
     .
1000 DISP "Zeitfehler : keine Daten mehr!"
1010 F1=1 ! Setzt Zeitfehler-Flag!
1020 RETURN
1030 END
```

Wenn ein Zeitfehler aufgetreten ist, wird Flag 1 gesetzt. Die Abfrage in Zeile 50 läßt dann keine weiteren ENTER-Operationen zu, sondern veranlaßt den Sprung aus der Schleife, wobei (als Beispiel) die Zahl der bisher aufgenommenen Daten ausgegeben und das Programm angehalten wird. Diese Zeile wird zwar auch bei ungestörter Aufnahme aller Daten erreicht, dann ist aber das Ende der Daten-Übermittlung erreicht, wie aus der Zahl der Daten leicht zu erkennen ist.

Natürlich gibt es auch eine Anweisung, die die Zeilen-End-Verzweigungen beim Auftreten von Zeitüberschreitungen verhindert. Eine OFF TIMEOUT-Anweisung mit dem Auswahl-Code des betroffenen Interfaces bewirkt dies. Den gleichen Effekt hat auch eine SET TIMEOUT-Anweisung mit der Zeit 0, weil dann die Überwachungsdauer (praktisch) sofort verstrichen ist, der zu überwachende Vorgang also erst danach beginnt und somit keiner Zeitbegrenzung mehr unterliegt. Die zu beachtenden Syntax-Regeln finden Sie im Anhang A.

Interrupts bei TRANFER-Abschlüssen

Das Ende von TRANSFER-Operationen läßt sich mit zwei verschiedenen Methoden überwachen, die in ihrer Flexibilität aber sehr unterschiedlich sind. Die eine fragt dazu die Status-Register SR2 und SR3 des betroffenen Buffers ab, während die andere den Abschluß der Datenübertragung selbst als Kriterium benutzt.

Wie schon im Kapitel "Zweck und Funktion der Buffer" angegeben wurde, enthalten diese Register während der TRANSFER-Operationen mit Peripherie-Geräten die Auswahl-Codes der daran beteiligten Interfaces und werden erst am Ende dieser Operationen wieder auf 0 gesetzt. Dieser Wechsel läßt sich durch wiederholtes Abfragen der Buffer-Status-Register ermitteln, wodurch sich aber der Programmablauf verzögert. Nur wenn der Rechner in der Zwischenzeit noch etwas anderes zu erledigen hat, kann man mit einer reichlich verzwickten Programmierung so die TRANSFER-Operationen überwachen und, falls vorteilhaft, wieder zur Bearbeitung der anderen Aufgabe zurückspringen. Es gibt aber einen bequemeren Weg:

Wenn eine ON EOT-Anweisung (ON End Of Transfer, "bei Ende der Übertragung") ins Programm gesetzt wird, läßt sich das Ende der Übertragung im angegebenen Interface selbst zur Auslösung einer Zeilen-End-Verzweigung verwenden. ON EOT veranlaßt dann den Sprung zur passenden Service-Routine. In dieser Routine kann man dann den Bufferinhalt den Variablen zuordnen oder neue Daten in den Buffer bringen oder die Buffer-Zeiger neu setzen, damit ein erneutes Füllen oder Lesen des Buffers möglich wird.

Um die Wirkung einer ON EOT-Anweisung vorzuführen, beschäftigen wir uns mit dem folgenden Beispiel. Ein Voltmeter (HP-3437A) wird so programmiert, daß es 100 mal in Abständen von 0,9 Sekunden eine Spannung mißt. Das Ergebnis der Messungen (bestehend aus je 7 Zeichen) soll anschließend mit der Interrupt-Transfer-Methode zum Rechner übertragen werden. Da nach dem letzten Wert jeder Serie das Voltmeter zusätzlich eine EOL-Sequenz (CR und LF) ausgibt, muß also der I/O-Buffer für die 100 Daten einen Umfang von 100 * 7 + 2 + 8 = 710 Bytes beim HP-83/85 (bzw. 702 beim HP-86/87) haben. Nach Abschluß des Transfers der Daten von 100 Messungen soll das Programm zur Zeile 1000 verzweigen, mit der Service-Routine die Daten speichern und danach eine neue Meß-Serie starten.

```
  10 ! Beispiel fuer Transfer-Abschluss mit Service-Routine
  20 ON EOT 7 GOSUB 1000
  30 OUTPUT 724 ;"D.9SN100SE0SR3T1F1"
  40 I=1 @ X=0 @ DIM D$[710],D(100)
  50 IOBUFFER D$
  60 ! Es wird angenommen, dass bereits 10 Daten-Files
  70 ! angelegt wurden, deren Namen in der folgenden
  80 ! DATA-Anweisung stehen!
  90 DATA "DF1","DF2","DF3","DF4","DF5","DF6","DF7",
     "DF8","DF9","DF10"
 100 TRANSFER 724 TO D$ INTR
 110 X=X+1 ! Warteschleife, die nur das
 120 DISP X ! das Weiterarbeiten des
 130 GOTO 110 ! Rechners zeigen soll!
   .
   .
1000 ! EOT-Service-Routine
1010 ! File-Namen lesen, File oeffnen!
1020 READ F$ @ ASSIGN# 1 TO F$
1030 ! Holt 100 Werte aus dem Buffer
1040 FOR N=1 TO 100
1050 ENTER D$ ; D(N)
1060 NEXT N
```

```
1070 ! Neues TRANSFER starten
1080 TRANSFER 724 TO D$ INTR
1090 ! Daten speichern, dabei haelt
     Transfer zeitweise an!
1100 PRINT# 1 ; D() @ ASSIGN# 1 TO *
1110 ! Pruefen, ob alle Files besetzt sind
1120 I=I+1 @ IF I>10 THEN GOTO 1140
1130 RETURN
1140 ! Wenn fertig, I/O stoppen, Programm abschliessen
1145 ! ABORTIO bewirkt autmatisch EOT,
     deshalb zuvor OFF EOT noetig!
1150 OFF EOT 7 @ ABORTIO 7
1160 PRINT "Alle Daten-Files sind besetzt!"
1170 END
```

In Zeile 1050 ist deshalb keine Formatierung nötig, weil die vom Voltmeter gelieferten Meßdaten auch nicht-numerische Zeichen enthalten, bei deren Erreichen jedesmal die Zuordnung auf die nächste Variable D(N) übergeht. Die logische Verbindung von TRANSFER-Anweisung und Service-Routine innerhalb des Programms ist durch die markierten Zeilen leicht zu erkennen. Das gezeigte Programm ist nur ein Beispiel für die vielen Möglichkeiten, die sich mit der EOT-Verzweigung ausführen lassen. Ähnlich wie oben gezeigt, könnten die aufgenommenen Daten auch mit einer weiteren Transfer-Ausgabe an einen Zentral-Rechner weitergereicht werden.

Die EOT-Service-Routine läßt sich mit einer OFF EOT-Anweisung mit Angabe des Interface-Auswahl-Codes auch wieder abschalten. Die zu beachtenden Syntax-Regeln finden Sie im Anhang A.

Rangordnung der Ursachen von Zeilen-End-Verzweigungen

In den folgenden Absätzen wird beschrieben, welche Dringlichkeit die einzelnen Ursachen für Zeilen-End-Verzweigungen haben. Damit können Sie gut beurteilen, wie sich ein Programm verhalten wird, wenn mehrere Ursachen für Verzweigungen gleichzeitig auftreten.

Wenn am Ende einer Programmzeile mehrere Gründe für Verzweigungen vorliegen, dann werden diese, eine nach der anderen, vom Rechner abgearbeitet. Die folgende Tabelle enthält alle Ursachen für Zeilen-End-Verzweigungen, kombiniert mit den Auswahl-Codes und gibt die daraus folgende Rangordnung an.

Rangordnung der Ursachen

	Auswahl-Code							
Ursache	3	4	5	6	7	8	9	10
ON ERROR	----------				1		----------	
ON INTR	2	3	4	5	6	7	8	9
ON TIMEOUT	10	11	12	13	14	15	16	17
ON EOT	18	19	20	21	22	23	24	25
ON KEY	----------				26		----------	
ON TIMER	----------				27		----------	

So haben z.B. alle ON INTR-Verzweigungen Vorrang vor den ON TIMEOUT-Verzweigungen, diese wiederum vor ON EOT-Verzweigungen usw. Verzweigungen gleicher Art von verschiedenen Interfaces herrührend, werden entsprechend der "Rangnummer" nacheinander bearbeitet. Von geringster Dringlichkeit sind die ON TIMER- von höchster die ON ERROR-Verzweigungen.

Sie sollten bemerken, daß hier von einem Vor-"Rang" gesprochen wird, nicht von einem Vor-"Recht". Das bedeutet einmal, daß auch ohne besondere Maßnahmen beim Auftreten mehrerer Ursachen für Zeilen-End-Verzweigungen alle entsprechend vereinbarten Service-Routinen gemäß der Dringlichkeit abgearbeitet werden, andererseits aber auch, daß während einer Service-Routine auch neue Zeilen-End-Verzweigungen ausgelöst werden können! Wenn dagegen nichts unternommen wird, können darmit recht unerwartete Programmabläufe ausgelöst werden!

Damit jede begonnene Service-Routine ungestört ablaufen kann, muß für diese Zeit sichergestellt sein, daß keine weitere Zeilen-End-Verzweigung ausgelöst wird. Das kann durch OFF XXXXXX-Anweisungen in der jeweils ersten Zeile der Service-Routine geschehen. Ein Beispiel (für Interrupt-Verzweigung) soll dies erläutern:

```
 10 ON INTR 7 GOSUB 100
 20 ON INTR 9 GOSUB 200
 30 ENABLE INTR 7;8
 40 ENABLE INTR 9;64
 50 !
 60 !
 70 ! Hier steht das Rumpf-Programm
 80 !
 90 !
100 OFF INTR 9 ! Auswahl-Code 9 darf nicht verzweigen!
  .
  .
130 ! Hier steht die Service-Routine
140 ! fuer Auswahl-Code 7
  .
  .
  .
180 ! Auswahl-Code 9 darf wieder verzweigen:
190 ON INTR 9 GOSUB 200 @ ENABLE INTR 7;8 @ RETURN
200 OFF INTR 7 ! Jetzt ist Auswahl-Code 7 blockiert!
  .
  .
240 ! Hier steht die Service-Routine
250 ! fuer Auswahl-Code 9
  .
  .
  .
280 ! Auswahl-Code 7 darf wieder verzweigen:
290 ON INTR 7 GOSUB 100 @ ENABLE INTR 9;64 @ RETURN
300 END
```

Nehmen wir an, daß während der Ausführung der Zeile 50 beide Verzweigungen ausgelöst wurden. Sie sind auszuführen, wenn Zeile 50 fertig ist. Das Interface 7 hat Vorrang, deshalb erfolgt der Sprung zur Zeile 100. Dem Interface 9 wird hier die Berechtigung genommen, wegen eines (bereits anstehenden) Interrupts eine Zeilen-End-Verzweigung einzuleiten. Erst unmittelbar am Ende der Service-Routine (Zeile 190) wird diese Sperre wieder aufgehoben. Deshalb erfolgt durch das RETURN sofort ein Sprung zur Zeile 200, in der jetzt das Interface 7 für Verzweigungen gesperrt wird. Erst in Zeile 290 wird es wieder zur Zeilen-End-Verzweigung zugelassen. Nur wenn inzwischen im Interface 7 kein neuer Interrupt aufgetreten ist, erfolgt ein Rücksprung zur Zeile 60, sonst muß die Service-Routine bei Zeile 100 erneut abgearbeitet werden!

Beachten Sie bitte, daß mit der OFF INTR-Anweisung die Verzweigung also nur aufgeschoben wird, daß aber die bereits "vorgemerkten" Verzweigungen unmittelbar nach einer späteren ON INTR-Anweisung ausgeführt werden.

Der Rechner "merkt" sich Ereignisse wie TRANSFER-Abschlüsse, Interrupts, Zeitüberschreitungen und die üblichen Error-Meldungen für unbegrenzte Zeit, es sei denn, es wird RESET oder STOP ausgeführt. Deshalb erfolgt bei ON EOT-, ON INTR-, oder ON TIMEOUT-Anweisungen sofort eine Verzweigung, wenn noch ein "unerledigter" Fall vorliegt.

Die Art der Verzweigung kann die Programm-Ausführung beeinflussen. Eine relativ sichere Vorhersage über den Ablauf läßt sich machen, wenn GOSUB-Verzweigungen benutzt werden. Die GOTO-Form sollte man nur wählen, wenn das Programm sehr einfach ist und nur eine oder zwei Verzweigungen auftreten können. Ein sehr einfaches Beispiel dafür ist:

```
ON INTR 7 GOTO 9999      (99999 beim HP-86/87!)
ENABLE INTR 7;128
```

Diese beiden Anweisungen verlangen den Sprung zur letzten Zeile (meist steht dort die END-Anweisung), wenn beim HP-IB- oder HP-IL-Interface ein IFC (Interface-Clear-Befehl) eintrifft. Dieser "Ausstieg" kann bei der Entwicklung von solchen Programmen nützlich sein, die "Tasten-Maskierungen" benutzen (siehe I/O-ROM-Abschnitt 10). Wenn dann nämlich irgend etwas schief geht, können Sie das Programm dadurch "retten", daß Sie durch irgend ein anderes Gerät oder einen Analysator ein IFC-Signal auf das Interface wirken lassen.

Das folgende Programm soll Ihnen zeigen, wie einige der verschiedenen Arten von ON XXXXX-Ereignis-Verzweigungen zusammenwirken. Das Listing und die Ergebnisse werden zwar mit einer kurzen Erklärung angegeben, Sie können aber noch viel mehr aus dem Beispiel lernen, wenn Sie selbst mit den Timern, der Zeitbegrenzung und den Sonder-Funktions-Tasten experimentieren. Starten Sie das Programm und warten Sie ab, was dann passiert und was nicht passiert! Betätigen Sie die Sonder-Funktions-Tasten, wenn der Rechner anhält, und warten Sie dann die Anzeigen ab!

```
 10 ! Dieses Programm dient zur Vorfuehrung der
 20 ! Zeilen-End-Verzweigung. Setzen Sie den bei Ihnen
 30 ! gueltigen Interface-Auswahl-Code und die Geraete-
 40 ! Adresse in den Zeilen 80 und 90 ein, damit das
 50 ! Programm erfolgreich laufen kann! Denken Sie daran,
 60 ! dass ENTER und TRANSFER-Eingabe nicht gleichzeitig
 70 ! fuer das gleiche Interface zugelassen sind!
 80 S1=10 ! Auswahl-Code fuer TRANSFER-Methode
 90 S2=705 ! Geraete-Adresse fuer ENTER-Anweisung
100 I=0 @ DIM A$[80],B$[88]
110 IOBUFFER B$
120 !
130 !
140 ON KEY# 1 GOSUB 390 ! Stellt Zaehler zurueck
150 ON KEY# 2 GOSUB 450 ! Beendet TRANSFER
160 ON KEY# 3 GOSUB 510 ! Haelt das Programm an
170 ON KEY# 4 GOSUB 770 ! Startet weiteren ENTER-Vorgang
180 !
190 !
200 ON TIMER# 1,1300 GOSUB 560
210 ON TIMER# 2,1400 GOSUB 610
220 !
230 !
240 SET TIMEOUT S2 ; 1000 ! Nicht den Auswahl-Code vergessen!
250 ON TIMEOUT S2 GOSUB 660
260 1
270 !
280 ON EOT S1 GOSUB 710 ! Nicht den Auswahl-Code vergessen!
290 TRANSFER S1 TO B$ INTR
```

```
300 ENTER S2 ; A$
310 !
320 !
330 BEEP 100,20 @ WAIT 500 @ BEEP 20,20
340 DISP "Durchlauf ";I
350 I=I+1 @ GOTO 330
360 !
370 !
380 !
390 ! Service fuer k1 loescht den Zaehler
400 I=0
410 DISP "Taste 1 fertig"
420 RETURN
430 !
440 !
450 ! Service fuer k2 beendet TRANSFER
460 RESET S1
470 DISP "TRANSFER beendet"
480 RETURN
490 !
500 !
510 ! Service fuer k3 beendet Programm
520 DISP "Ende des Programms"
530 END
540 !
550 !
560 ! Service fuer Timer 1
570 DISP "Timer 1 ist abgelaufen!"
580 RETURN
590 !
600 !
610 ! Service fuer Timer 2
620 DISP "Timer 2 ist abgelaufen!"
630 RETURN
640 !
650 !
660 ! Zeitkontrolle fuer ENTER S2
670 DISP "ENTER dauerte zu lange!"
680 RETURN
690 !
700 !
710 ! TRANFER-Ende fuer S2
720 DISP "TRANSFER zu Ende!"
730 TRANSFER S1 TO B$ INTR
740 RETURN
750 !
760 !
770 ! Service fuer k4 startet neue ENTER-Aktivitaet
780 ENTER S2 ; A$
790 RETURN
800 END
```

Beeinflussung des Tasten-Feldes

Einführung

Fast immer wird ein Programm empfindlich gestört, wenn im unpassenden Moment zufällig eine Taste des Rechners berührt wird. Bei anderen Gelegenheiten soll der Benutzer nur ganz bestimmte Tasten betätigen, weil die Benutzung aller übrigen Tasten im Augenblick sinnlos oder unerwünscht ist.

Es war daher früher üblich, das Tastenfeld der Rechner mit "Masken" aus Plastik oder Metall so abzudecken, daß alle im Moment nicht benötigten Tasten verdeckt waren. Nur die erlaubten Tasten waren durch Ausschnitte in den Masken zugänglich. Die Erstellung dieser Masken war recht kostspielig, ihr Erfolg nur bei konsequenter Anwendung durch den Benutzer gewährleistet.

Mit dem I/O-ROM lassen sich solche "Masken" sehr leicht in das Programm einbauen, womit ein zwangsläufiger Schutz gegen Fehlbedienungen erreicht wird, der nicht "vergessen" werden kann. Diese "Software"-Masken sind auch sehr schnell und einfach an die wechselnden Forderungen anzupassen.

Die programmierte Tasten-Feld-Maske

Es gibt grundsätzlich drei verschiedene Arbeits-Zustände des Rechners:

Wartezustand

Programmablauf

Warten auf Tastenfeld-Eingabe(n)

In den Wartezustand geht der Rechner nach dem Einschalten oder nach Ausführung einer STOP-, PAUSE- oder END-Anweisung, also immer dann, wenn kein Programm abläuft. Der Wartezustand wird also auch für die Programm-Entwicklung oder Fehlersuche benutzt, eine "Maske" ist dabei sinnlos bis störend und deshalb nicht vorgesehen.

Beim Programmablauf bewirkt die Betätigung der meisten Tasten eine Programm-Unterbrechung. Hiervon ausgenommen ist nur die TR/NORM-Taste (nur beim HP-86/87 vorhanden) und alle Sonder-Funktions-Tasten, sofern ihnen eine Funktion zugeordnet ist.

Für Tastenfeld-Eingaben innerhalb eines Programms hält der Rechner das Programm an und fordert durch eine Anzeige den Benutzer zur Eingabe auf. So arbeitet z.B. die INPUT-Anweisung.

Für die Festlegung der Maske sind die Tasten in vier Klassen eingeteilt:

1. RESET-Taste
2. PAUSE-Taste
3. KEY LABEL- und Spezial-Funktions-Tasten
4. Alle übrigen Tasten (außer den unter 1. bis 3. genannten).

Jede dieser Klassen kann für sich oder auch gemeinsam mit anderen Klassen während des Programmablaufs und/oder beim Warten auf Eingaben gegen Einwirkung von außen gesperrt ("maskiert") werden.

Dabei ist es mit einer einzigen Anweisung im Programm möglich, für die beiden Betriebszustände verschiedene Masken zu definieren.

Die ENABLE KBD-Anweisung enthält als Parameter einen numerischen Wert zwischen 0 und 255, dessen Bitmuster zwei an die Rechnerzustände angepaßte Masken ergibt. Dabei verkörpern die vier höchsten Bits die während des Programmablaufs gültige Maskierung, während die vier niedrigsten Bits die gewünschte Maske für die Tastenfeld-Eingabe festlegen. Die Bedeutung der Bits ist in der folgenden Tabelle zusammengefaßt. Das Setzen der Bits aktiviert die entsprechende(n) Taste(n), ein gelöschtes Bit blockiert die Benutzung. Das bedeutet, daß nur die Tasten wirksam sind, für die in der ENABLE KBD-Anweisung die passenden Bits gesetzt wurden.

Bit-Nummer	Dezimal-Wert	Betriebs-Art	Beeinflußte Tasten
7	128		RESET
6	64	Programm-	PAUSE
5	32	Ablauf	KEY LABEL u. k1 bis k8/14
4	16		Alle übrigen Tasten
3	8		RESET
2	4	Tastenfeld-	PAUSE
1	2	Eingabe	KEY LABEL u. k1 bis k8/14
0	1		Alle übrigen Tasten

Das folgende Beispiel zeigt die Blockierung aller Tasten mit Ausnahme der Spezial-Funktions-Tasten während des Programm-Ablaufs. Anschließend werden dann für eine Tastenfeld-Eingabe RESET-, PAUSE- und die Spezial-Funktions-Tasten maskiert. Damit kann der Benutzer zwar Eingaben vornehmen, er hat aber keine Möglichkeit, das Programm zu stören.

```
 10 ENABLE KBD 1+32 ! Bezeichnet die erlaubten Tasten
 20 REMOTE 706,710 @ LOCAL LOCKOUT 7
 30 ON KEY# 1 GOTO 1000
 40 ON KEY# 2 GOTO 2000
 50 ON KEY# 3 GOTO 3000
  .
  .
  .
100 PRINT "Betriebsart waehlen"
110 GOTO 110 ! Schleife, Programm "laeuft"!
  .
1000 DISP "Datum (Tag,Monat,Jahr) eingeben" @ INPUT D,M,Y
1010 ! usw.!
```

Der Benutzer kann nun mit den Spezial-Funktions-Tasten den gewünschten Programm-Abschnitt wählen, aber nichts anderes tun, denn nur die Spezial-Funktions-Tasten (Klasse 3) reagieren. Wenn z.B. die Taste k1 gedrückt wird, springt das Programm zur Zeile 1000. Wenn der Rechner nun auf die Eingabe wartet, sind nur die Tasten der Klasse 4 wirksam. Wenn das Programm dann weiterläuft, sind wieder nur die Spezial-Funktions-Tasten in Betrieb.

Das folgende Programm-Beispiel zeigt noch eindrucksvoller die Wirkung aber auch die "Gefahren", die von der ENABLE KBD-Anweisung ausgehen: Der Benutzer hat nach dem Starten des Programms nur 10 Sekunden Zeit, um den Rechner während der Text-Anzeige mit der PAUSE-Taste anzuhalten oder mit der RESET-Taste zu erneutem Start zu veranlassen:

```
 10 ! Zeile 50 ermoeglicht nur STOP oder RESET,
 20 ! wenn der Rechner auf die Eingabe in Zeile 100
 30 ! wartet. Nach Ablauf von 10 Sekunden (TIMER# 1)
 40 ! gibt es kein Halten mehr....!
 50 ENABLE KBD 12
 60 !
 70 !
 80 !
 90 BEEP @ DISP "Nur 10 Sekunden, um anzuhalten
    oder zurueckzusetzen"
100 ON TIMER# 1,10000 GOTO 110 @ INPUT X9$
110 OFF TIMER# 1 @ DISP "Jetzt startet das Programm!"
120 !
130 ! Jetzt ist das Programm nicht mehr anzuhalten!
140 !
150 ! usw....
```

Hier hilft "nur noch" der Netzschalter, um den Rechner wieder unter Kontrolle zu bekommen!

Notizen:

Unmittelbarer Zugriff auf das Interface

Einführung

Dieser Abschnitt behandelt Anweisungen, mit denen der Programmierer das Interface in seiner Arbeitsweise an spezielle Bedingungen anpassen kann.

Der Ablauf der Vorgänge im Interface kann mit der STATUS-Anweisung überwacht werden. Aus diesen Zustands-Meldungen läßt sich der Pegel der externen I/O-Leitungen, aber auch der interne Zustand des Interfaces erkennen, es ist dazu nur das passende Status-Register abzufragen.

Mit der CONTROL-Anweisung läßt sich dagegen festlegen, auf welche Art das Interface bestimmte Dinge tun soll. Das Interface steuert zwar grundsätzlich die I/O-Leitungen automatisch so, wie es die gewählte Betriebsart erfordert, es besteht aber auch die Möglichkeit, auf bestimmten I/O-Leitungen Signale in der vom Benutzer gewünschten Art auszugeben.

Abfragen des Zustandes

Der Zustand von Interfaces (und auch von Buffern) wird mit der STATUS-Anweisung ermittelt. Diese ist sehr einfach aufgebaut, sie enthält nur den Interface-Auswahl-Code, die Nummer des gewünschten Registers und eine Variable. Die folgende Anweisung ermittelt den Wert des Identifizierungs-Registers SR0 von einem Interface mit dem Auswahl-Code 7:

```
STATUS 7,0 ; X
```

Auswahl-Code	Register-Nummer	Numerische Variable

Im obigen Fall wird X mit dem Wert 1 besetzt, dem Kennzeichen für ein HP-IB-Interface. Jede Interface-Bauart gibt nämlich bei dieser Abfrage eine andere Zahl zurück, die für die Bauart typisch ist. Die nötigen Angaben über die von Ihnen verwendete Interface-Bauart finden Sie im entsprechenden Handbuch.

Mit einer einzigen STATUS-Anweisung lassen sich auch mehrere Status-Register abfragen: Es ist dazu nur nötig, das niedrigste Register anzugeben und eine ausreichende Zahl von Variablen in die Anweisung zu setzen. Das folgende Beispiel liest aus dem Interface mit Auswahl-Code 7 die Status-Register SR1 bis SR6:

```
STATUS 7,1 ; X1,X2,X3,X4,X5,X6
```

Auswahl-Code	Start-Reg.	Wert SR1	Wert SR2	Wert SR3	Wert SR4	Wert SR5	Wert SR6

Die weitere Benutzung der so ermittelten Daten hängt vom Interface und vom Programm ab. Bei dem Beispiel (HP-IB-Interface) haben die Register folgende Zuordnung:

SR1 : Ursache des Interrupts.

SR2 : Zustand der HP-IB-Steuerleitungen.

SR3 : Zustand der HP-IB-Datenleitungen.

SR4 : HP-IB-Adresse des Interfaces.

SR5 : Zustand des HP-IB-Interfaces.

SR6 : Sekundär-Befehl.

Bei anderen Interface-Bauarten können diese Register natürlich ganz andere Bedeutungen haben!

Die meisten Interface-Ausführungen sind in der Lage, eine Interrupt-Meldung an den Rechner zu geben, wenn die vorher vereinbarten Bedingungen dafür eintreten. Wenn aber mehrere Ursachen zu einem Interrupt führen können, muß die auslösende Ursache erst einmal festgestellt werden, um danach die passende Service-Routine auszusuchen. Das folgende Programm-Beispiel benutzt die STATUS-Anweisung und die BIT-Funktion zum Feststellen der Ursache und zur Wahl der passenden Routine:

```
10 ON INTR 7 GOSUB 1000
15 ! Interrupt soll erfolgen wenn Interface
16 ! Aktiver Controller/Talker/Listener wird!
20 ENABLE INTR 7;112
   .
1000 ! Zeilen-End-Verzweigung für HP-IB-Interrupts
1010 STATUS 7,1 ; S1 ! Liest Interrupt-Ursache
1020 IF BIT(S1,4)=1 THEN GOTO 1100 ! Test, ob akt. Talker
1030 IF BIT(S1,5)=1 THEN GOTO 1200 ! Test, ob akt. Controller
1040 IF BIT(S1,6)=1 THEN GOTO 1300 ! Test, ob akt. Listener
1050 !
1060 ! Die folgenden Zeilen sind für unerwartete Fehler!
1070 PRINT "Dieser Interrupt ist beim HP-IB unzulässig!"
1080 PRINT "HP-IB-Interrupt-Ursache = ";S1
1090 STOP
1100 ! Service fuer akt. Talker
   .
1200 ! Service fuer akt. Controller
   .
1300 ! Service fuer akt. Listener
```

Zeile 1010 ermittelt bei Auftreten eines Interrupts den Wert vom Register SR1. In den Zeilen 1020 bis 1040 wird dann der Anlaß für den Interrupt analysiert und die passende Service-Routine ausgewählt. Die Zeilen 1050 bis 1090 dienen der Aufdeckung möglicher Programmfehler oder unzulässiger Interrupts. Kein Gerät und kein Programm ist nämlich so perfekt, daß man unvorhergesehene Fehler grundsätzlich ausschließen kann!

Interface-Beeinflussung

Das Interface läßt sich grundsätzlich in seiner Arbeitsweise an das vorhandene System und dessen Bedingungen genau anpassen. Das bedeutet meistens "nur" das Festlegen der Bedingungen, die bei allen Betriebs- und auch Störungs-Zuständen einen Interrupt auslösen sollen oder dürfen. Es kann sich aber auch um die Ausarbeitung eines speziellen Handshake-Verfahrens für ein besonders "widerborstiges" Peripherie-Gerät handeln. In all diesen Fällen kann man mit passend formulierten CONTROL-Anweisungen die Steuer-Register vom Interface in die gewünschten Zustände versetzen.

Drei Gründe sprechen für eine eingehende Beschäftigung mit den Steuer-Registern und der CONTROL-Anweisung. Da ist zuerst die "Interrupt-Maske", die beim HP-IB- bzw. HP-IL-Interface im Steuer-Register CR1 angelegt wird. Die beiden folgenden Anweisungen
haben (für ein HP-IB-Interface mit Auswahl-Code 7) gleiche Wirkung:

```
ENABLE INTR 7 ; 8    oder    CONTROL 7,1 ; 8
```

Beide Anweisungen gestatten einen Interrupt, wenn eine SRQ-Meldung (Bedienungs-Ruf, Service ReQuest) vorliegt. Meist wird der ENABL INTR-Anweisung der Vorzug gegeben, die ausschließlich das Interrupt-Masken-Register beeinflußt und deshalb die bessere Dokumentation bietet. Eine CONTROL-Anweisung kann dagegen auch mehrere aufeinander folgende Steuer-Register ansprechen, was zwar im Programm Platz spart, dafür aber sehr viel Aufmerksamkeit bei der Analyse erfordert.

Auch die externen I/O-Leitungen lassen sich gezielt über das im Interface befindliche Steuer-Register CR2 beeinflussen. Sie können jede dieser Leitungen durch Setzen oder Löschen des passenden Bits in CR2 in den gewünschten Zustand bringen. Widmen Sie auch hier wieder dem zum Interface gehörenden Handbuch ausreichende Aufmerksamkeit! Die beiden folgenden Anweisungen haben (bei einem HP-IB-Interface mit Auswahl-Code 7) auf das Steuer-Register CR2 (fast) die gleiche Wirkung:

```
ASSERT 7 ; 32      oder      CONTROL 7,2 ; 32
```

Beide Anweisungen setzen die Steuer-Leitung für SRQ auf "wahr", trotzdem wirken beide Anweisungen unterschiedlich: Die CONTROL-Anweisung wird erst ausgeführt, wenn die momentan laufende I/O-Operation (z.B. TRANSFER mit Interrupt-Methode) beendet ist. Anders bei der ASSERT-Anweisung: sie wird sofort ausgeführt, ohne Rücksicht auf laufende I/O-Operationen. Auf alle Fälle sollte man Vorsicht walten lassen, wenn man in die Interface-Steuer-Register eingreift: als Konsequenz einer ungeschickten oder falschen Steuerungs-Anweisung kann im Interface oder im Peripherie-Gerät eine Fehlfunktion ausgelöst werden. Für den Fall, daß Sie sich mit "SRQ" herumplagen, sollten Sie lieber die REQUEST-Anweisung verwenden, weil dann ein korrekter Ablauf der Vorgänge auf den Bus-Leitungen gewährleistet ist.

Achtung: Durch ungeschickte Maßnahmen an den Steuerleitungen können Daten verstümmelt und Fehlfunktionen ausgelöst werden!

Schließlich kann man durch Beeinflussung der Steuer-Register die vom Interface als EOL abgegebene Zeichenfolge verändern. Diese Zeichenfolge ist in der Wirkung ein "Ende-der-Daten"-Signal, das in den vergangenen Zeiten bei Lochkarten anzeigte, daß eine Karte vollständig gelocht (oder gelesen!) war. Für alle Interfaces besteht die Standard-EOL-Sequenz aus Wagenrücklauf- und Zeilenvorschub-Symbol (CR/LF) und wird nach jeder PRINT- oder OUTPUT-Anweisung ausgegeben (wenn sie nicht vom Programmierer ausgeschaltet wurde). Die EOL-Sequenz wird auch ausgegeben, wenn ein "/"-IMAGE-Spezifikator auftritt, oder wenn das Schlüsselwort "END" innerhalb der SEND-Anweisung auftritt.

Manchmal muß die EOL-Sequenz auf die Peripherie-Geräte Rücksicht nehmen, sei es, daß ein Drucker den Zeilenvorschub doppelt braucht, sei es, daß ein Monitor ihn nicht verträgt, weil bei ihm schon das CR-Signal den Zeilenvorschub verursacht. Eine doppelte Ausgabe des letzten Zeilenvorschubs läßt sich mit der nachstehenden Anweisung erzielen:[1])

```
CONTROL 7,16;128+3,13,10,10
```

EOI möglich	EOL-Zähler	CR-Symbol	2mal LF-Symbol

Mit zunehmender Erfahrung werden Sie selbst merken, wie anpassungsfähig diese Steuer-Register das System machen. Der Umgang mit diesen Registern sollte aber mit Überlegung geschehen, damit Ihnen unangenehme "Überraschungen" erspart bleiben. Schauen Sie sich die für ihr Interface gültigen Beschreibungen sehr genau an, um zu erfahren, welche Möglichkeiten bestehen und wie man mit diesen umgeht. Das Sammeln von "Erfahrungen" wird nötig sein, bis das System so läuft, wie Sie es wünschen. Aber auch die Programm-"Profis" von heute mußten das mal lernen!

---

1)Sie sollten dazu wissen, daß OUTPUT- und SEND-Anweisungen EOI als Merkmal des letzten Datenbytes nicht ausgeben, EOI ist also nur zusammen mit PRINT-, DISP und TRANSFER-Anweisungen möglich!

Notizen:

Weitere I/O-Anweisungen

Einführung

Das I/O-ROM hält noch weitere Anweisungen bereit, die in den vorangegangenen Abschnitten noch nicht erwähnt wurden. Von diesen Anweisungen sind einige nur für bestimmte Interface-Bauarten brauchbar, und für alle diese Anweisungen gilt, daß sie auf jede Interface-Art anders wirken können. Der Sinn dieser Anweisungen soll durch die jeweiligen Beispiele gezeigt werden. Ausführlichere Angaben finden Sie in den zu den Interfaces gehörenden Beschreibungen.

Interface-abhängige Anweisungen:

Die SEND-Anweisung

Mitunter besteht der Wunsch oder die Notwendigkeit, eine ganz speziell zusammengestellte Datenfolge an ein oder mehrere Peripherie-Geräte zu übermitteln. Hierzu eignet sich die SEND-Anweisung ausgezeichnet, wie die beiden Beispiele zeigen:

```
SEND 705 ; DATA "Willkommen im I/O-Land!"
SEND 9 ; DATA "9. Oktober","13. November"
```

Anschließend an die Daten wird noch die EOL-Sequenz ausgegeben, falls dies vorher vereinbart wurde.

Mit der SEND-Anweisung lassen sich beim RS-232-Interface nur Daten ausgeben, bei den anderen Bauarten sind sowohl Daten als auch Befehle mit der SEND-Anweisung übertragbar. Damit besteht die Möglichkeit, Byte-Folgen auszugeben, deren Zusammenstellung völlig auf die Belange der angesprochenen Geräte abgestimmt werden kann. Das gleiche kann im Prinzip auch durch OUTPUT-Anweisungen erreicht werden, die SEND-Anweisung bietet aber die bessere Dokumentation innerhalb des Programms. Hier einige Beispiele von korrekt formulierten SEND-Anweisungen:

```
SEND 7 ; CMD B$
SEND 9 ; MTA MLA UNL LISTEN 4,5 CMD F,H
SEND 4 ; CMD X$
```

Sie werden in diesen Fällen die zu den Peripherie-Geräten gehörenden Handbücher befragen müssen, um sich über die notwendigen Byte-Folgen zu informieren.

Die Anweisungen HALT, ABORTIO und RESET

Mit der HALT-Anweisung läßt sich fast jede laufende Datenübertragung (Ausnahme: FHS-Transfer) abbrechen. Das ist eine gute Methode, um aus einer Situation herauszukommen, die z.B. wegen einer Störung des Handshakes zum "Aufhängen" des Rechners führte. Beim FHS-Transfer sind allerdings andere Verfahren nötig!

Die nächste häufig benutzte Anweisung ist ABORTIO. Auch sie bricht die Datenübertragung ab, setzt bei allen Interface-Bauarten die Steuer-Leitungen, bei einigen Interface-Arten auch die Daten-Leitungen auf ihren Einschaltzustand zurück. ABORTIO eignet sich deshalb gut als Abschluß einer I/O-Operation, weil alle Geräte in definierte Zustände versetzt werden.

Die RESET-Anweisung ist in dieser Gruppe die radikalste Anweisung, weil sie nicht nur jeglichen Datenaustausch abbricht, sondern auch alle vorher vereinbarten Einstellungen der einzelnen Glieder des System auf die Einschaltwerte zurücksetzt, die an den Geräten z.T. durch Schalter definiert wurden. Das ist aber nur nötig, wenn ein Programm erstmalig gestartet werden soll oder wenn ein Fehler überwunden werden muß. Hier nun Beispiele, die diese Anweisungen benutzen:

```
HALT 7
ABORTIO 7
RESET 7
```

Die Anweisungen RESUME und CLEAR

Wenn bei der Ausführung von HALT- oder SEND-Anweisungen eine Zeilen-End-Verzweigung zugelassen ist, dann wird diese auch ausgeführt. Auf eine HALT- bzw. SEND-Anweisung muß mitunter eine RESUME-Anweisung folgen, damit Eingabe- und Ausgabe-Funktionen wieder ausführbar sind. Die CLEAR-Anweisung wird beim HP-IB-, HP-IL- und GPIO-Interface zum Rücksetzen der Peripherie-Geräte benutzt. Das ausführende Interface selbst wird von der CLEAR-Anweisung nicht beeinflußt. Auch hier wieder Beispiele:

```
RESUME 7
CLEAR 705
```

Funktionen zur Bus-Steuerung:

Bei den speziell zur Steuerung von Meß-Systemen entworfenen Interfaces (HP-IB und HP-IL) gibt es noch die Anweisungen REMOTE und LOCAL, die darüber entscheiden, ob eine Meßeinrichtung per Programm vom Rechner oder manuell über die Einstellorgane auf das gewünschte Verfahren bzw. auf den gewünschten Bereich eingestellt werden soll.

Die REMOTE-Anweisung bereitet dabei die Ferneinstellung vor, während durch die LOCAL-Anweisung (oder einen besonderen Schalter am Gerät) die Einstellorgane dort wieder aktiviert werden.

Für extreme Sicherheitsanforderungen kann mit der LOCAL LOCKOUT-Anweisung sogar jegliche manuelle Beeinflussung am Gerät solange unterbunden werden, bis eine spätere LOCAL-Anweisung diese Sperre wieder aufhebt. Ein Beispiel für die Anwendung der LOCAL-Anweisung finden Sie im I/O-ROM-Abschnitt 9 (Seite 63).

Die jetzt noch folgenden Anweisungen werden an dieser Stelle nicht besprochen, da ihre Wirkungen zu sehr von der Bauart des Interfaces abhängen:

REQUEST SPOLL PPOLL PASS CONTROL TRIGGER

In der Syntax-Übersicht (Anhang A) finden Sie eine Übersicht der Anwendungen in Verbindung mit den verschiedenen Interface-Bauarten. Für weitere Informationen wird auf die entsprechenden Interface-Beschreibungen verwiesen.

Allgemeine Informationen

Einführung

Das HP-IB-Interface verbindet die Rechner der Serie 80 mit dem Hewlett-Packard-Interface-Bus. Dieses Interface entspricht voll dem IEEE STANDARD 488-1978 und, mit Ausnahme der Steckverbindung, auch voll der Norm IEC 625. Es ermöglicht eine große Zahl von Operationen, mit denen Sie einen Rechner der Serie 80 zur Steuerung von Peripherie-Geräten und zur Auswertung, Bearbeitung und Weitergabe der dort anfallenden Daten verwenden können.

Das HP-IB-Interface wird zwar mit sehr unterschiedlichen System-Zusammenstellungen eingesetzt, gemeinsames Kennzeichen ist aber die schnelle Übermittlung von Daten und Befehlen über kurze Distanzen. Hauptanwendungsgebiet ist einmal die Meßtechnik in der Forschung, in der bio-elektronischen Medizin und in der Prozeß-Steuerung und -überwachung, wofür das Input/Output-ROM erforderlich ist, zum anderen das Zusammenwirken des Rechners mit Geräten zur Ausgabe bzw. Speicherung von Daten, wie Drucker, Plotter, Massenspeicher und ähnlichen Geräten, wobei das Plotter/Printer-ROM und/oder das Mass-Storage-ROM verwendet wird. Da diese Anwendungen einfacher sind, werden sie im IB-Abschnitt 3 zuerst beschrieben, der Umgang mit dem I/O-ROM schließt sich daran an.

Die späteren Abschnitte enthalten weitere Informationen über das HP-IB-Interface mit Programmbeispielen und theoretische Grundlagen seiner Funktion. Diese Abhandlungen gelten sowohl für die einsteckbare Interface-Ausführung, als auch für die fest im Rechner installierte Version. Auf Unterschiede der Bauformen wird von Fall zu Fall ausdrücklich hingewiesen.

Benötigte ROM's

Zur Durchführung von Input/Output-Operationen braucht das IB-Interface mindestens ein Input/Output- oder ein Mass Storage- oder ein Printer-ROM. Der Typ des ROM's entscheidet dabei, in welchem Umfang I/O-Operationen möglich sind. Da bei einigen Ausführungen der Rechner der Serie 80 diese ROM's bereits im Grundgerät fest eingebaut sind, beziehen sich die angegebenen Bestellnummern auf die Steck-ROM's:

Für HP-83/85:

Input/Output-ROM	(Bestellnummer 00085-15003)
Plotter/Printer-ROM	(Bestellnummer 00085-15002)
Mass-Storage-ROM	(Bestellnummer 00085-15001)

Für HP-86/87:

Input/Output-ROM	(Bestellnummer 00087-15003)
Plotter-ROM	(Bestellnummer 00087-15002)

Ob ein bestimmtes ROM dem System zur Verfügung steht, läßt sich einfach prüfen: Wird eine für das entsprechende ROM typische Anweisung vom Rechner ohne Fehlermeldung ausgeführt, dann ist dieses ROM vorhanden. Antwortet der Rechner aber mit 'BAD STMT', dann muß das fehlende ROM in den eventuell schon vorhandenen ROM-Einschub eingebaut werden. Der Ein- und Ausbau von ROM's ist in der Bedienungsanleitung zum ROM-Einschub HP 82936A ausführlich beschrieben. Hier wird nur daran erinnert, daß der Rechner bei diesen Arbeiten unbedingt ausgeschaltet sein muß!

Das Mass-Storage-ROM ermöglicht den Zugriff auf Platten-Laufwerke, während das Plotter/Printer-ROM den Umgang mit Druckern und Plottern stark vereinfacht. Das I/O-ROM stellt die universellen Funktionen bereit, die für alle erdenklichen Peripherie-Geräte benötigt werden.

Technische Daten

Die wichtigsten technischen Daten sind in der folgenden Aufstellung zusammengefaßt. Eine komplette Beschreibung aller elektrischen und mechanischen Einzelheiten sowie Angaben über das Übertragungsprotokoll findet man im
IEEE-Standard 488-1978

Abmessungen:	167 mm x 127 mm x 15 mm (o. Kabel)
Gewicht:	0,5 kg
Länge des fest montierten Kabels:	ca. 2 m (Angaben über Maximal-Länge aller Kabel in IB-Abschnitt 2)
Arbeitstemperatur:	0 bis 55 °C
Lagertemperatur:	-40 bis 65 °C
Stromversorgung:	aus dem Hauptgerät (Rechner)
Anzahl der Peripherie-Geräte:	Für ein HP-IB-Interface sind maximal 14 Peripherie-Geräte zugelassen

IB-Abschnitt 2

Einbau, Anschluß und Voreinstellungen

Wenn Sie einen Rechner der Serie 80 zur Verfügung haben, der ein fest eingebautes HP-IB-Interface enthält, sind die Hinweise über Ein- und Ausbau des Interfaces für Sie natürlich unwichtig. Für das An- und Abschalten von Peripherie-Geräten gelten dagegen für alle Interface-Bauarten die gleichen Regeln, die Sie gründlich lesen sollten.

Auspacken und Prüfen auf Beschädigungen

Falls der Versandkarton beschädigt ist, bitten Sie einen Beauftragten des Spediteurs, beim Auspacken anwesend zu sein. Falls Sie bei unbeschädigtem Versandkarton nach dem Auspacken mechanische Schäden am Interface feststellen oder das Interface den Funktionstest nicht besteht, informieren Sie sofort den Spediteur und die nächstgelegene HP-Vertretung. Heben Sie den Versandkarton für die Prüfung durch den Spediteur auf. Die HP-Vertretung wird dann Reparatur oder Austausch des Gerätes veranlassen, ohne die Abwicklung der Ersatzansprüche gegen den Spediteur abzuwarten.

Interface-Einbau

Sie sollten den gesamten Einbau-Vorgang besonders hinsichtlich der Vorsichts-Maßnahmen verstanden haben, bevor Sie mit dieser Arbeit beginnen oder Peripherie-Geräte anschließen:

Die Hersteller von Peripherie-Geräten benutzen (leider) keine einheitliche Technik zur Lösung der Probleme von Erdung bzw. Schirmung. Häufig hat das Chassis des Gerätes keine leitende Verbindung mit der 'logischen Masse', weil man 'Erdschleifen' und die dadurch möglichen Störungen der Daten-Leitungen vermeiden möchte.

Beim Einsetzen in den Rechner der Serie 80 wird das HP-IB-Interface mit seiner 'logischen Masse' zwangsläufig mit dem inneren Rechner-Chassis leitend verbunden. Die möglicherweise nicht mit dem Geräte-Chassis verbundene 'logische Masse' eines Peripherie-Gerätes kann nun, besonders bei einem defekten Peripherie-Gerät, Spannung gegen die 'logische Masse' des IB-Interfaces führen. Diese Spannung kann für den Benutzer gefährlich werden, solange das Peripherie-Gerät nicht fest mit dem Bus-System verbunden ist.

Wenn Sie nicht wissen, welche Schaltungs- bzw. Schutzmaßnahmen im Peripherie-Gerät angewendet wurden, sollten Sie den Hersteller befragen. Erst wenn Ihnen bestätigt wurde, daß eine zum Interface verträgliche Technik vorliegt, dürfen Sie mit den nächsten Schritten Ihr HP-IB-Interface mit den Peripherie-Geräten zum System zusammenschließen.

Warnung:
Um weder Personen noch Geräte zu gefährden, muß der Aufbau der anzuschließenden Geräte bekannt sein. Auch muß die nachstehend beschriebene Reihenfolge beim Einbau des Interfaces und beim Anschluß peripherer Geräte genau eingehalten werden!

Achtung:
Schalten Sie den Rechner und auch alle zur Peripherie gehörenden Geräte ab, bevor Sie das IB-Interface einsetzen bzw. entfernen wollen! Sie vermeiden damit die Gefahr, eines oder mehrere der Geräte zu beschädigen!

1. Schalten Sie den Netzschalter an der Rückseite des Rechners aus (Stellung "0" bzw. "OFF"), und sorgen Sie dafür, daß der Rechner über ein passendes 3-adriges Kabel mit einer (intakten) Schuko-Steckdose verbunden ist.

Warnung:
Stecken Sie weder Finger, noch Werkzeuge oder andere leitende Gegenstände in die Einschuböffnungen, da die Gefahr von geringfügigen elektrischen Schlägen besteht, die Sie unnötig erschrecken oder unter Umständen auch Störungen an Herzschrittmachern auslösen können. Außerdem besteht die Gefahr, die Kontakte und auch innere Teile des Rechners zu beschädigen.

2. Entfernen Sie die Schutzkappe einer noch unbesetzten Einschuböffnung. Andere, noch unbesetzte Einschuböffnungen sollten auch weiterhin abgedeckt bleiben!

Achtung:
Sie dürfen das Interface nicht mit Gewalt in die Einschuböffnung drücken, da es sonst mechanisch beschädigt wird! Das Einsetzen ist nur möglich, wenn die Beschriftung des Gehäuses von oben sichtbar ist.

3. Führen Sie das Interface in dieser Lage, mit der Steckleiste (Printplatte) voran, in die Einschuböffnung, was ohne großen Widerstand möglich sein muß. Nur für das letzte Stück (ca. 7 mm) bis zum Anliegen der Anschläge ist einige Kraft nötig, da jetzt die Steckanschlüsse hergestellt werden. Das Einsetzen geht leichter, wenn das Gehäuse abwechselnd links und rechts hineingedrückt wird. Die unterschiedliche Form der Führungsschienen auf der linken und rechten Seite verhindert einen seitenverkehrten Einbau.

Anschließen von Peripherie-Geräten

1. Überzeugen Sie sich, daß die anzuschließenden Geräte (aber auch die schon zum System gehörenden!) mit ihrem jeweiligen Netzschalter ausgeschaltet sind (die Geräte dürfen aber bereits an das Netz angeschlossen sein!).

2. Prüfen Sie die Adress-Schalter der anzuschließenden Geräte auf richtige Einstellung. Denken Sie daran, daß als Adressen Dezimalzahlen zwischen 0 und 30 (einschließlich) zugelassen sind und jedes Gerät eine individuelle Adresse erhalten muß. Auch darf keines der Peripherie-Geräte die Adresse des Rechners haben (diese wurde im Werk auf 21 eingestellt!).

3. Schließen Sie jetzt das am Interface vorhandene Kabel an das erste Peripherie-Gerät an. Mit weiteren Kabeln, die sich "Huckepack" verbinden lassen, können jetzt weitere Geräte angeschlossen werden. Auf der Seite IB-6 sind lieferbare Kabelausführungen und auch die zu beachtende Einschränkung der Gesamt-Kalänge angegeben.

4. Nachdem alle Peripherie-Geräte an den Bus angeschlossen sind, dürfen Sie die momentan benötigten Geräte einschalten und danach auch den Rechner. Es ist zulässig, die nicht benötigten Geräte ausgeschaltet zu lassen, sofern mehr als die Hälfte der Peripherie-Geräte eingeschaltet ist: bei 3 Geräten reicht es aus, wenn davon 2 eingeschaltet sind, von 5 Geräten sollten aber mindestens 3 eingeschaltet sein. Wenn der Rechner vor den Geräten eingeschaltet wurde, kann er nicht die vorhandenen Geräte erkennen. In diesem Fall muß als Starthilfe die RESET-Taste einmal gedrückt werden.

Abtrennen von Peripherie-Geräten

Das Abtrennen der Peripherie-Geräte vom Bus-System muß mit folgenden Schritten durchgeführt werden:

Warnung:
Entfernen Sie das Interface nicht aus dem Rechner, solange dieser eingeschaltet ist! Beschädigungen am Interface und/oder am Rechner könnten die Folge sein!

1. Überzeugen Sie sich, daß alle zum Bus-System gehörenden Geräte und auch der Rechner ausgeschaltet sind (alle Geräte dürfen dabei durch ihre Kabel weiter mit dem Netz verbunden bleiben!).

2. Entfernen Sie die IB-Verbindungskabel von allen Geräten, die zukünftig nicht mehr zum System gehören sollen. Verringern Sie dabei, falls möglich, die Gesamt-Kabellänge. Wenn alle nicht benötigten Peripherie-Geräte abgetrennt sind, sollten Sie zuerst die verbliebenen Geräte und anschließend den Rechner wieder einschalten.

Interface-Ausbau

Für den ungefährdeten Ausbau des HP-IB-Interfaces 82937A sind folgende Schritte nötig:

1. Überzeugen Sie sich, daß alle zum Bus-System gehörenden Geräte und auch der Rechner ausgeschaltet sind (alle Geräte dürfen dabei durch ihre Kabel weiter mit dem Netz verbunden bleiben!).

2. Lösen Sie das am IB-Interface angebrachte Kabel vom Peripherie-Gerät und ziehen Sie dann das Interface vorsichtig aus der Einschuböffnung. Dabei ist anfangs (ca. 7 mm) relativ viel Kraft nötig, um die Steckverbindung zu trennen. Die Trennung geht leichter, wenn am Gehäuse abwechselnd links und rechts gezogen wird. Danach muß sich das Interface praktisch ohne Widerstand aus der Einschuböffnung herausnehmen lassen. Das Interface soll in seinem Original-Karton oder auf andere Weise gut geschützt aufbewahrt werden.

3. Die nunmehr leere Einschuböffnung ist mit einer Schutzkappe zu verschließen.

Verbindungskabel für den System-Aufbau

Warnung:
Bevor Sie zusätzliche Kabel zum Aufbau eines Bus-Systems benutzen, sollten Sie die Seite IB-3 gelesen und die dort geschilderten Probleme voll verstanden haben!

Das Einschub-Interface wird mit einem fest montierten, etwa 2 m langen Kabel geliefert, das in einer "Huckepack"-Garnitur endet, die Stecker- und Buchsen-Seite aufweist. Damit läßt sich - meist ohne weiteres - das erste Peripherie-Gerät anschließen. In den seltenen Fällen, in denen das Gegenstück im Gerät tief versenkt eingebaut ist, muß ein Adapter verwendet werden, um einen ordnungsgemäßen Anschluß zu erreichen.

Weitere Peripherie-Geräte können dann mit zusätzlichen Kabeln an das System angefügt werden, wobei die Steckgarnituren aufeinander gesteckt und mit den Rändelschrauben gegen Trennung gesichert werden.

Bei den nach Europa ausgelieferten Geräten und Kabeln, die dem Standard IEEE 488 entsprechen, haben diese Befestigungsteile metrisches Gewinde (M 3,5) und sind schwarz (vernickelt). Es gibt aber auch Geräte und Kabel (kenntlich an den metallisch blanken Befestigungsteilen), die Gewinde nach amerikanischer Norm haben. Eine "Kombination" dieser beiden Gewindearten führt zwangsläufig zu Beschädigungen! Die Anpassung ist aber durch einen Umrüst-Satz leicht möglich (HP-Bestellnummer 5060-0138).

Da es außer dem Standard IEEE 488-1978 auch noch die Norm IEC 625 gibt, existieren auch Adapter, die die Verbindung der unterschiedlichen Stecker ermöglichen. Man sollte jedoch den in IEC 625 genormten Stecker innerhalb des IB-Systems nach Möglichkeit vermeiden, da dieser Stecker auch von anderen Bus-Systemen benutzt wird, die elektrisch nicht zum IB-System passen und wegen wesentlich höherer Betriebsspannungen die IB-Geräte beschädigen können!

Lieferbare Kabel-Längen

Beschreibung	Bestellnummer
Kabel, 0,5 m lang	HP 10833D
Kabel, 1,0 m lang	HP 10833A
Kabel, 2,0 m lang	HP 10833B
Kabel, 4,0 m lang	HP 10833C
Adapter	HP 10834A

Maximal zulässige Kabellängen

Für eine einwandfreie Funktion des Bus-Systems sind die folgenden Bedingungen einzuhalten:

1. Die größte zulässige Länge eines Einzelkabels als Verbindung zwischen zwei Geräten ist 4 m.
2. Die Summe der Längen aller zum Bus-System gehörenden Kabel darf nur so groß sein, daß auf jedes Gerät (einschließlich dem Interface im Rechner) im Schnitt nur 2 m entfallen, insgesamt jedoch 20 m nicht überschritten werden.

Es gibt hinsichtlich der Zusammenschaltung der geeigneten Peripherie-Geräte keine weiteren Vorschriften, nur sollten an einem Gerät nie mehr als 3 bis höchstens 4 Stecker-Garnituren im "Huckepack" zusammentreffen, da die Verbindung zwischen Geräte-Chassis und eingebauter Buchse dann mechanisch zu stark beansprucht wird.

Auswahl-Code, Adresse, Controller-Funktion und Parallel-Abfrage

Die in der Überschrift genannten Kenndaten lassen sich am Interface auf die gewünschten Werte einstellen. Zur Vereinfachung der Benutzung verlassen die Geräte das Werk mit festgelegten Grundeinstellungen:

* Interface-Auswahl-Code 7
* Talker/Listener-Adresse 21
* System-Controller-Funktion vorhanden
* Parallel-Abfrage-Leitung DIO 1

Beim fest im Rechner eingebauten IB-Interface lassen sich die gewünschten Werte an zwei Schaltern an der Rückseite des Rechners ohne Montage-Arbeiten einstellen, der folgende, die Montage-Arbeiten beschreibende Absatz ist nur für das einsteckbare IB-Interface gültig.

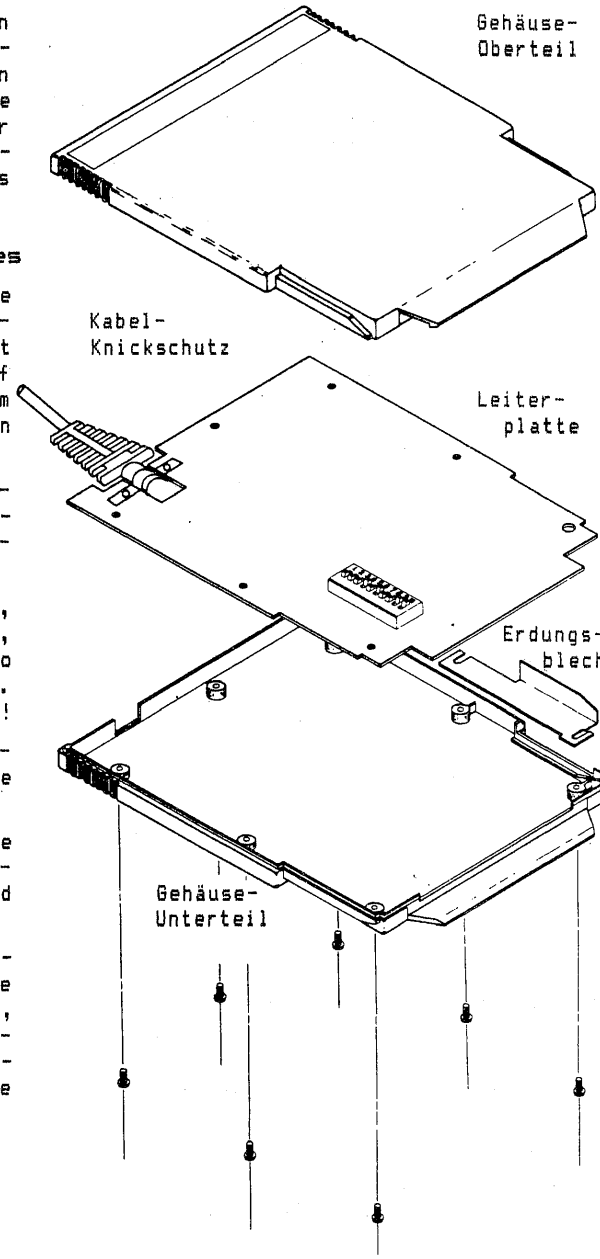
öffnen des Interface-Gehäuses

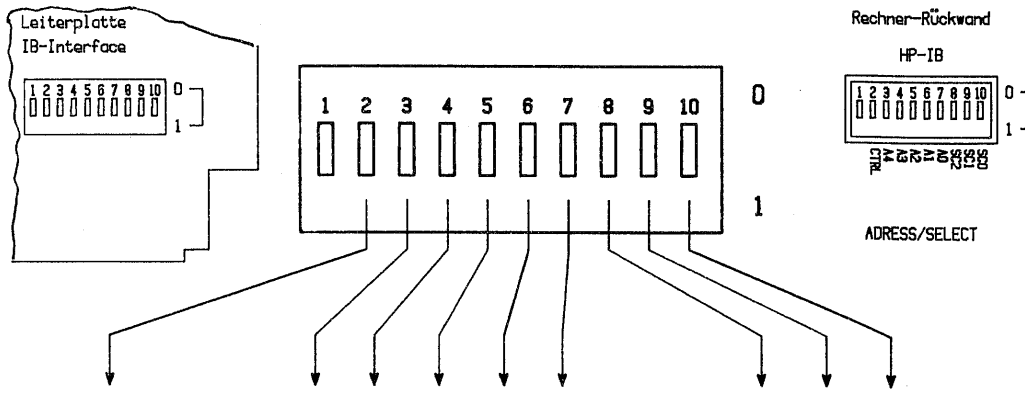
Das nebenstehende Bild zeigt die einzelnen Interface-Teile. Zur Demontage muß das Interface (Schrift nach unten, Kabel nach links!) auf eine ebene Fläche gelegt werden,um die folgenden Arbeiten durchführen zu können:

1. Lösen Sie mit passendem Kreuz-Schlitzschraubendreher die sieben sichtbaren Schrauben vollständig.
2. Greifen Sie unter das Gehäuse, halten Sie alle Teile zusammen, und drehen Sie das Interface so um, daß das Kabel links bleibt. Dabei auf die Schrauben achten!
3. Halten Sie das Kabel am Knickschutz fest und heben Sie die obere Gehäusehälfte ab.

Überzeugen sie sich, daß die Lage der Einzelteile der Abbildung entspricht. Wichtig ist, den im Bild dargestellten Schalter zu finden.

Beim späteren Zusammenbau der Teile ist in umgekehrter Reihenfolge zu verfahren und darauf zu achten, daß das Erdungsblech zwischen Lötseite der Leiterplatte und Gehäuseunterteil an der richtigen Seite liegt.





: 2 :	: 3	4	5	6	7 :	: 8	9	10 :
System- :	Talker/Listener-				:	Interface-		:
Contr. :	Adresse				:	Auswahl-Code		:
Nein: 0 :	0 : 0	0	0	0	0 :	3 : 0	0	0 :
Ja : 1 :	1 : 0	0	0	0	1 :	4 : 0	0	1 :
	2 : 0	0	0	1	0 :	5 : 0	1	0 :
	3 : 0	0	0	1	1 :	6 : 0	1	1 :
	4 : 0	0	1	0	0 :	7 : 1	0	0 :
	5 : 0	0	1	0	1 :	8 : 1	0	1 :
	6 : 0	0	1	1	0 :	9 : 1	1	0 :
	7 : 0	0	1	1	1 :	10 : 1	1	1 :
	8 : 0	1	0	0	0 :			
	9 : 0	1	0	0	1 :			
	10 : 0	1	0	1	0 :			
	11 : 0	1	0	1	1 :			
	12 : 0	1	1	0	0 :			
	13 : 0	1	1	0	1 :			
	14 : 0	1	1	1	0 :			
	15 : 0	1	1	1	1 :			
	16 : 1	0	0	0	0 :			
	17 : 1	0	0	0	1 :			
	18 : 1	0	0	1	0 :			
	19 : 1	0	0	1	1 :			
	20 : 1	0	1	0	0 :			
	21 : 1	0	1	0	1 :	---Voreinstellungen ab Werk!		
	22 : 1	0	1	1	0 :			
	23 : 1	0	1	1	1 :			
	24 : 1	1	0	0	0 :			
	25 : 1	1	0	0	1 :			
	26 : 1	1	0	1	0 :			
	27 : 1	1	0	1	1 :			
	28 : 1	1	1	0	0 :			
	29 : 1	1	1	0	1 :			
	30 : 1	1	1	1	0 :			

Hinweis:
Die Auswahl-Codes 1 und 2 sind für den (internen) Bildschirm bzw. Drucker reserviert!

Die Bedeutung der Codier-Schalter-Elemente

Wie bereits eingangs erwähnt, wird die Eigenschaft, System-Controller zu sein, die Talk/Listen-Adresse und der Auswahl-Code bei der Lieferung des Interfaces im Herstellerwerk durch Einstellen bestimmter Schalter festgelegt, deren Bedeutung hier im einzelnen beschrieben wird.

Es handelt sich hierbei um den Schalterblock, der an der Rückseite des Rechners (beim fest eingebauten IB-Interface) oder auf der Leiterplatte (beim Einschub-Interface) zu finden ist. Für beide Ausführungen gilt die folgende Beschreibung.

Der Schalterblock enthält 10, von 1 bis 10 numerierte Kipp- oder Schiebeschalter mit jeweils zwei Stellungen, die auf der Platine mit "0" und "1" bzw. auf der Rückseite des Rechners noch zusätzlich bezeichnet sind, wie die Abbildung auf Seite IB-8 oben rechts zeigt.

Von diesen Schalterelementen ist das Element 1 ohne Wirkung. Mit dem Element 2 läßt sich entscheiden, ob der Rechner System-Controller sein soll. Die Elemente 3 bis 7 dienen zur Festlegung der Talk/Listen-Adresse des Rechners. Für den Auswahl-Code sind die Elemente 8 bis 10 zuständig.

Auf Seite IB-8 sind die für die einzelnen Werte notwendigen Schalterstellungen tabellarisch aufgeführt, wobei die vom Hersteller gewählten Standard-Einstellungen durch Unterstreichung hervorgehoben sind.

Achtung:
Wenn Sie die Stellung einzelner Schalter-Elemente verändern wollen, achten Sie darauf, daß nicht -unbeabsichtigt- auch die benachbarten Elemente mit verstellt werden! Eine Bleistiftspitze oder ein ähnlich schlanker Gegenstand eignet sich gut zum Verstellen. Falsche Einstellung der Schalter beschädigt zwar nichts, stört aber die System-Funktion empfindlich!

Der System-Controller-Schalter

Das Element 2 ist bei Lieferung auf "1" gestellt, wodurch der Rechner System-Controller ist. Diese Einstellung braucht nur verändert zu werden, wenn im anzuschließenden System bereits ein System-Controller existiert. Es darf nämlich in jedem HP-IB-System nur ein einziges Gerät diesen Vorrang besitzen, weil sonst Störungen in der Funktion des Systems unvermeidlich sind. Während des Arbeitens eines System kann zwar jedes dazu geeignete Gerät durch entsprechende Befehle zum Controller aktiviert werden, beim Einschalten oder Starten des Systems muß aber, auch ohne Befehl, bereits ein Gerät Controller sein. Diese Aufgabe (neben einigen anderen) erfüllt der System-Controller.

Die Schalter für die Talk/Listen-Adresse

Wie jedes andere Gerät des IB-Systems hat auch der Rechner eine individuelle Geräte-Nummer, die als Talk/Listen-Adresse bezeichnet wird. Durch Angabe dieser Adresse kann sich der Rechner auch selbst zum Talker ernennen, wenn er Daten ausgeben muß oder er macht sich zum Listener, wenn er Daten aufnehmen will.

Die Talk/Listen-Adresse müssen Sie nur dann ändern, wenn die Adresse 21 bereits an ein anderes Gerät vergeben ist und dies nicht mehr geändert werden kann. Das tritt z.B. ein, wenn mehr als ein Rechner der Serie 80 im Anlieferungszustand in einem System arbeiten soll. Zur Festlegung der Talk/Listen-Adresse dienen die Elemente 3 bis 7 am Codier-Schalter.

Die Schalter für den Interface-Auswahl-Code

Die Elemente 8 bis 10 des Codier-Schalters bestimmen den Auswahl-Code des Interfaces. Bei der Auslieferung ist der Auswahl-Code "7" eingestellt, der typisch ist für das HP-IB-Interface. Eine Änderung dieses Zustandes ist nur nötig, wenn ein anderes, bereits mit dem Rechner verbundenes Interface diesen Code schon belegt. Da der Rechner die Interfaces mit ihren Auswahl-Codes anspricht, dürfen in einem Rechner niemals zwei Interfaces den gleichen Auswahl-Code belegen.

Antwort auf eine Parallel-Abfrage (Parallel Poll)

Mit Ausnahme des Controllers kann jedes Gerät des HP-IB-Systems mit der SRQ-Leitung eine Abfrage durch den Controller provozieren, die dieser meist auf serielle Weise (seriell=nacheinander!) durchführt. Daneben gibt es für Systeme mit nur wenigen Geräten noch die Möglichkeit der Parallel-Abfrage (parallel=gleichzeitig!). Dazu wird jedem Gerät eine der Datenleitungen DIO 1 bis DIO 8 als Antwort-Leitung zugeordnet. Auch der Rechner kann auf eine Abfrage antworten, wenn er darauf vorbereitet wurde und nicht die Funktion des aktiven Controllers ausübt.

Die hierzu notwendigen Einstellungen bzw. Arbeiten werden nachstehend für beide Ausführungsformen des Interfaces beschrieben:

Fest eingebautes IB-Interface:

Beim fest eingebauten IB-Interface erfolgt die Auswahl der Antwort-Leitung durch einen an der Rückwand des Rechners angebrachten Drehschalter, der durch die Beschriftung "PARALLEL POLL" gekennzeichnet ist und der sich mit einem Schraubendreher mit flacher Klinge betätigen läßt.

Einschub-IB-Interface:

Beim Einschub-Interface wird die Antwort-Leitung durch eine eingelötete Drahtbrücke festgelegt. Diese Drahtbrücke liegt verdeckt unter den Einzeladern, in die sich das IB-Kabel im Innern des Interface-Gehäuses aufspaltet. Diese Einzeladern führen zu einer vielpoligen Steckverbindung, deren Gegenstück in der Printplatte eingelötet ist. Nach Abziehen des Buchsenteils von den Steckerstiften (parallel zur Printplatte nur am Gehäuse ziehen, keinesfalls an den Einzeladern!) läßt sich das Kabel beiseite drücken, wodurch das Umschaltfeld für die Antwort-Leitung voll sichtbar wird.

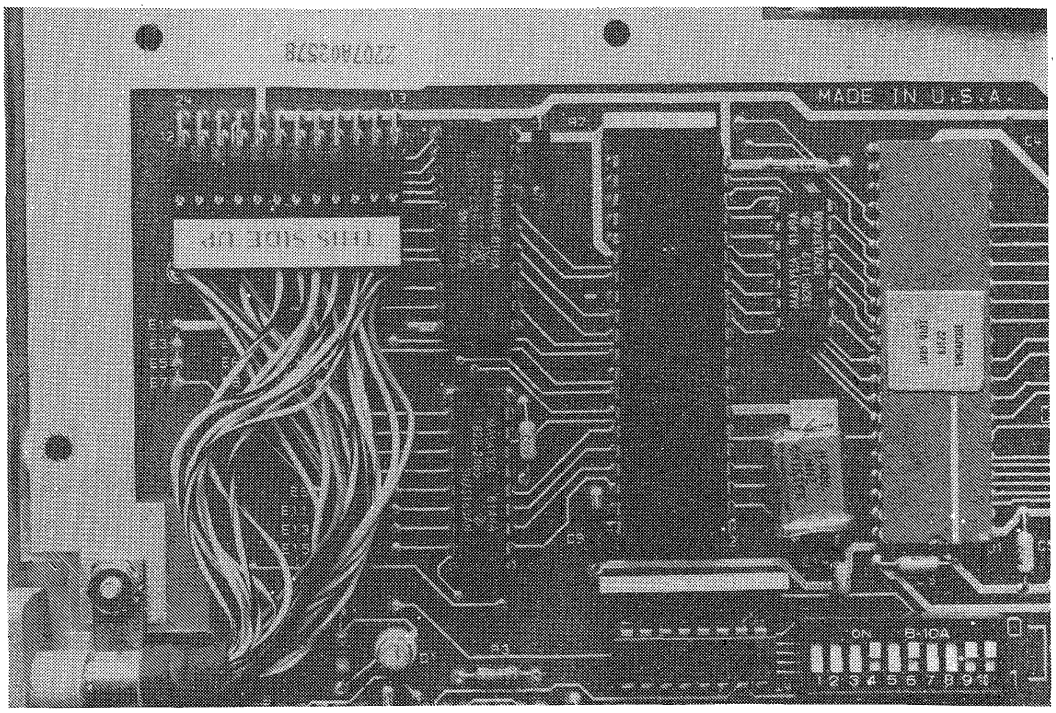
Zwei bzw. vier Reihen von durchkontaktierten Bohrungen tragen Beschriftungen, wie sie auch in der unten angegebenen Tabelle zu finden sind (bei einigen Ausführungen sind nur die Löcher mit ungeraden Nummern auf der Bestückungsseite bezeichnet, die Angaben für die übrigen Löcher finden sich auf der Lötseite!).

Prüfen Sie jetzt, ob sich die Drahtbrücke an der gewünschten Stelle befindet und suchen Sie eventuell die Bohrungen, in die die Brücke umzusetzen ist. Zum Umlöten benutzen Sie bitte einen Lötkolben mit höchstens 25 W Leistung, bei dem sicher ist, daß er keine für die Halbleiterschaltung gefährlichen Spannungen führt! Benutzen Sie das ausgelötete Drahtstück auch für die neue Verbindung und prüfen Sie nach den Lötarbeiten, daß die Printplatte auf beiden Seiten ohne Zinnspritzer geblieben ist! Danach stellen Sie die Steckverbindung wieder her.

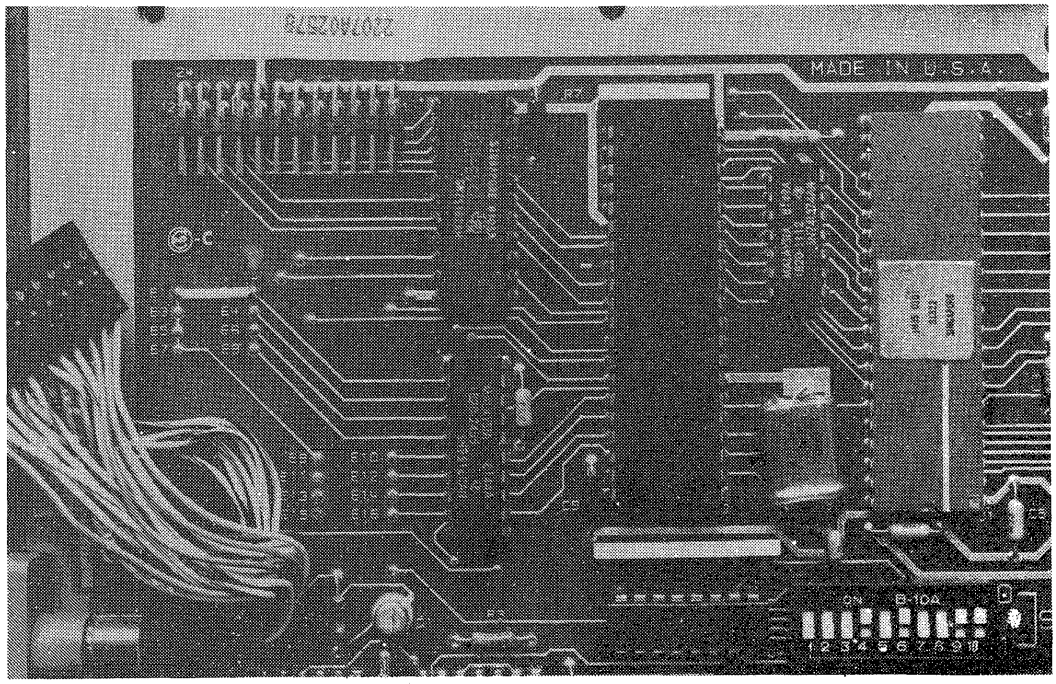
Antwort-Leitung	Brücke von	nach	
DIO 1	E1	E2	Lieferzustand!
DIO 2	E3	E4	
DIO 3	E5	E6	
DIO 4	E7	E8	
DIO 5	E9	E10	
DIO 6	E11	E12	
DIO 7	E13	E14	
DIO 8	E15	E16	

Nach Abschluß dieser Einstell- und Schaltarbeiten können Sie das Interface-Gehäuse wieder zusammenbauen, zuschrauben und das Interface in den Rechner einsetzen, wie auf den Seiten IB-7 und IB-3 beschrieben.

Bei Auslieferung ist bei allen IB-Interfaces DIO 1 die Antwort-Leitung für die Parallel-Abfrage!



Geöffnetes Einschub-IB-Interface



Umschaltfeld unter dem Kabel

Notizen:

Umgang mit dem HP-IB-Interface

Kurz-Einführung

Dieser Abschnitt ist in zwei Unterabschnitte aufgeteilt:

1. Ein sehr kurzes 'Rezept' für ein System, das ohne I/O-ROM auskommt.
2. Beschreibung des HP-IB-Systems mit den Anweisungen des I/O-ROM's.

Eine ins Detail gehende Beschreibung finden Sie dann im IB-Abschnitt 4. Für die Benutzung des HP-IB-Systems ist es aber nicht erforderlich, daß Sie alles vorher lesen. Wenn Sie einen externen Drucker nur für das Ausgeben von Programm-Listings und Ergebnissen brauchen, sind Sie schon an der nächsten Überschrift 'informiert' und können die Lektüre (vorerst) beenden!

Ein einfaches HP-IB-System ohne I/O-ROM

Wir glauben, daß Sie die IB-Abschnitte 1 u. 2 durchgelesen haben und deren Inhalt in groben Zügen kennen. Wir nehmen weiter an, daß Sie ein IB-Interface und einen Drucker besitzen. Um den Drucker benutzen zu können, müssen Sie folgendes tun:

1. Das Interface in den **ausgeschalteten** Rechner einbauen!
2. Das Verbindungskabel an Ihren Drucker anschließen.
3. Die Adresse Ihres Druckers feststellen (siehe Drucker-Handbuch!).
4. Den Auswahl-Code vom Interface feststellen (bei Lieferung "7"!).

Wenn wir voraussetzen, daß der Drucker die Adresse 01 und das Interface den Auswahl-Code 7 hat, sollten Sie jetzt die Geräte einschalten und die folgende Anweisung eingeben:

```
PRINTER IS 701     (END LINE)
```

Wenn Sie nun, wie gewohnt, PLIST ausführen, erscheint der Druck auf dem angegebenen Drucker und auch alle weiteren PRINT-Anweisungen gehen von nun an zu dem von Ihnen 'adressierten' IB-Drucker. Dieses Verfahren läuft für andere Geräte (Plotter und Massenspeicher) entsprechend. Schauen Sie nur in die Handbücher für das Printer/Plotter- und das Mass-Storage-ROM, um die gegebenen Möglichkeiten durch Beispiele kennenzulernen! Auch im Handbuch für den Rechner finden sich interessante Anwendungen für HP-Peripherie-Geräte.

Damit ist die erste Runde geschafft, nur wenn Sie neugierig sind, müssen Sie noch weiter lesen!

Vorstellung des HP-IB-Systems

Dieser Abschnitt ist für den bestimmt, der erstmalig mit dem HP-IB-System Kontakt bekommt. Darum wird hier eine leichtverständliche Einführung gegeben, um das Konzept klar darzustellen.

Was ist 'HP-IB' überhaupt?

HP-IB bedeutet: **H**ewlett-**P**ackard-**I**nterface-**B**us.

Das ist die Hewlett-Packard-Version der Norm IEEE-488-1978, die für die Zusammenschaltung von Geräten aufgestellt wurde. Die Grund-Idee des HP-IB-Systems ist es, Einzelgeräte so zu normen, daß sie hinsichtlich des Daten-Austauschs ohne weiteres elektrisch und mechanisch zueinander passen. Man kann dann seine überlegungen darauf konzentrieren, welche Daten zwischen den Geräten zu übertragen sind. Nach diesen Vorklärungen über die Kompatibilität dürfen wir uns jetzt mit der Struktur des HP-IB und mit dem dafür typischen 'Bus' beschäftigen, der die Verbindung zwischen den Geräten herstellt.

Struktur-Übersicht des HP-IB-Systems

Das HP-IB-System hat eine genau festgelegte Struktur, die man auch als sehr sorgfältig durchdachte Organisation bezeichnen kann. Diese allein sorgt für einen geordneten Ablauf im System. Diese Organisation ist leichter zu verstehen, wenn man das HP-IB-System mit einem Komitee vergleicht, das für den Ablauf seiner Beratungen eine Reihe von festen Regeln als zweckmäßig erkannt hat. Dies ist die für das HP-IB-System geltende Norm IEEE-488-1978.

Ein Mitglied des Komitees, üblicherweise als 'Vorsitzender' bezeichnet, hat die Aufgabe, die Sitzung zu leiten und die Verhandlungen zu organisieren. Der Vorsitzende ist für die Taten und Ergebnisse des Komitees verantwortlich und wendet die vereinbarten Regeln an, um einen geordneten Verlauf sicherzustellen. Wenn er, was manchmal vorkommt, die Sitzung nicht leiten mag oder kann, bestimmt er einen Vertreter zum 'geschäftsführenden Vorsitzenden'.

Vorsitzender im HP-IB-System ist der 'System-Controller'. Seine 'Wahl' geschieht durch die Einstellung eines Schalters am Interface: Diese Wahl kann per Programm nie geändert werden. Auch im HP-IB-System ist es möglich, einen 'geschäftsführenden Vorsitzenden' zu bestimmen: dies ist dann der 'Aktive Controller'. Im System darf jedes Gerät mit dieser Funktion betraut werden, das den Ablauf der Operationen steuern kann (z.B. ein Rechner).

Wenn der System-Controller durch Einschalten in Betrieb genommen oder auf seinen Startzustand zurückgesetzt wird, übernimmt er automatisch die Rolle des Aktiven Controllers. Diese Aufgabe kann aber unmittelbar danach an ein anderes Gerät abgegeben werden, damit sich der System-Controller anderen Aufgaben zuwenden kann: Ein HP-IB-System kann also durchaus auch mehrere Rechner enthalten.

Zwecks besserer Verständigung sollten in einem Komitee die einzelnen Redner immer nur nacheinander sprechen. Dafür sorgt der jeweilige Vorsitzende, der den Rednern nach bestimmten Grundsätzen nacheinander 'das Wort erteilt'.

Dem Redner entspricht im HP-IB der 'Aktive Talker'. Auch hier darf immer nur ein Gerät so bezeichnet werden. Die Worterteilung heißt 'Adressierung zum Talker' und wird vom Aktiven Controller vorgenommen. Die 'Rede' des aktiven Talkers ist eine 'Datenmeldung', womit wir zur eigentlichen Aufgabe des HP-IB kommen.

In einem Komitee lauschen (im Idealfall?) alle Anwesenden den Worten des Redners, beim HP-IB ist das aber anders: Der aktive Controller legt auch fest, wann welche Geräte 'zuhören' müssen und sagt allen anderen Geräten, daß die augenblicklich gebotenen Daten nicht für sie bestimmt sind. Die Aufforderung zum 'Zuhören' heißt 'Adressierung zum Listener', und wird vom aktiven Controller vorgenommen. Die angesprochenen Geräte werden dadurch zu 'Aktiven Listenern'. Diese Adressierung hat auf die übrigen Geräte die Wirkung einer 'Ent-Adressierung', wodurch diese Geräte die gebotenen Daten nicht zur Kenntnis nehmen können.

Das Verfahren, viele Geräte 'weghören' zu lassen ist nur auf den ersten Blick unverständlich: Wenn man sich aber vorstellt, daß alle Zuhörer eines Komitees alles vollständig mitschreiben wollten, wird der Sinn klar: Da es schnelle und langsame Schreiber gibt, würde hier der langsamste auch bei den Passagen, von denen er gar nicht betroffen ist, das Tempo bestimmen. Durch zeitweises 'Beurlauben' der langsamen Schreiber kann erheblich Zeit gespart werden, wodurch sich mehr Dinge erledigen lassen, ohne die Sitzung zu verlängern.

Auch beim HP-IB-System kann es solche Situationen geben: Wenn Drucker und Massenspeicher z.B. zur gleichen Zeit aktive Listener sind, dann wird der relativ langsame Drucker bestimmen, wie schnell die angebotenen Daten verarbeitet werden. Das Speichern eines Programms dauert dann so lange, wie der Druck des Listings. Viel schneller kommt man voran, wenn man für die Dauer der Speicherung den Drucker vom Datenempfang ausschließt.

Nachdem wir jetzt die Struktur des Systems in großen Zügen kennen, wollen wir nun sehen, wie es seine Hauptaufgabe, die Datenübermittlung ausführt!

Die Datenübertragung im HP-IB-System

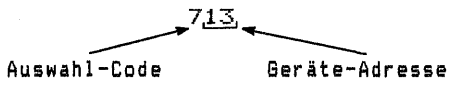
Die Datenübertragung dient dem Austausch von Informationen zwischen den einzelnen Geräten des HP-IB-Systems. (Unser Komitee führt seine Geschäfte, indem es die Ansichten der einzelnen Redner anhört und daraus seine Schlußfolgerungen zieht.) Im HP-IB-System gehen die Daten vom aktiven Talker nur zu den aktiven Listenern, wobei auch hier der langsamste das Übertragungs-Tempo bestimmt. Diese Einschränkung ist nötig, um eine fehlerfreie Übertragung der Daten sicherzustellen. Der technische Ausdruck für diese Temposteuerung bei der Übertragung ist 'Handshake': Stellen Sie sich ein Komitee vor, bei dem diesmal nur die interessierten Zuhörer mitschreiben und während des Schreibens eine Hand heben. Der Sprecher kann nur dann mit seiner Rede beginnen oder sie fortsetzen, wenn alle Hände unten sind.

Ein ähnlicher Ritus wird auch beim HP-IB benutzt. Es darf Sie aber nicht wundern, daß das Verfahren wegen der Zahl der sehr unterschiedlich arbeitenden Geräte viel komplexer ist. Sie brauchen die Einzelheiten auch nicht zu kennen, um das HP-IB-System erfolgreich benutzen zu können.

Wie adressiert man ein Gerät? Die einzelnen Geräte müssen ansprechbar sein, damit Ihnen den Talker- oder Listener-Status gegeben werden kann. Der erste Teil einer Adresse ist der Auswahl-Code des HP-IB-Interfaces (z.B. 7). Das läßt sich mit der Anschrift eines Appartementhauses vergleichen. Der zweite Teil (13 für unser Beispiel) entspricht dann der Wohnungs-Nummer innerhalb des Hauses. Wenn wir für einen Bewohner dieses Hauses eine Sendung adressieren wollen, müssen wir beides angeben: Anschrift und Wohnungsnummer, etwa so:

```
Ringstraße 7     (Straße und Hausnummer)
Wohnung 13       (Wohnungsnummer)
```

Eine vollständige Geräte-Adresse für ein mit dem Bus verbundenes Gerät sieht dann so aus:



Jedes am Bus angeschlossene Gerät muß eine individuelle Adresse haben, sonst gibt es ein großes Durcheinander, wie man sich leicht vorstellen kann.

Zur Durchführung einer Datenübermittlung muß der aktive Controller folgendes tun:

1. Allen Geräten die Hörerlaubnis entziehen, um 'lahme Lauscher' auszuschalten.
2. Durch Adressierung festlegen welches Gerät Talker werden soll.
3. Durch Adressierung festlegen, welche(s) Gerät(e) Listener sein soll(en).
4. Bei allen betroffenen Geräten die Bereitschaft für die Übertragung abwarten.

Glücklicherweise wird dies alles vom Rechner meistens automatisch erledigt. Hiervon gibt es zwar einige Ausnahmen, aber das sind Spezialfälle, die nur den Experten interessieren. Wir behandeln deshalb diese Fälle hier nicht.

Normalerweise ist der Rechner als aktiver Controller in jede Datenübertragung mit verwickelt. Wenn der Rechner Daten an den Drucker senden soll, wie z.B. folgenden Text: "Jetzt ist es so weit!", dann kann das mit der folgenden Anweisung geschehen:

```
OUTPUT 701 ; "Jetzt ist es so weit!"
```

Diese Anweisung muß folgende Dinge tun, die alle in dem Schlüsselwort OUTPUT einschlossen sind:

1. Allen Geräten den Listener-Status entziehen.
2. Den Rechner als Talker deklarieren ("Ich bin Talker" ausgeben).
3. Den Drucker (Geräte-Nr. 01) als Listener deklarieren.
4. Die Datenübertragung durchführen.

Genau so einfach, wie der Rechner Daten ausgeben kann, läuft auch der Empfang von Daten ab. Wir wollen annehmen, daß der Rechner das Meßergebnis eines Voltmeters übernehmen soll, das eben eine Spannung zu +1.075 V ermittelt hat. Wenn das Voltmeter die Adresse 03 hat, ergibt sich die Anweisung zur Aufnahme der Daten als:

```
ENTER 703 ; V
```

Auch hier enthält das Schlüsselwort ENTER eine ganze Serie an Tätigkeiten, die zu tun sind:

1. Allen Geräten den Listener-Status entziehen.
2. Den Rechner als Listener deklarieren ("Ich bin Listener" senden).
3. Das Voltmeter (Geräte-Nr. 03) als Talker deklarieren.
4. Die Datenübertragung durchführen.

Beachten Sie bitte, daß der Rechner nur die Adresse des anzusprechenden Gerätes benötigt, seine eigene Adresse kennt er selbstverständlich!

Auch wenn das HP-IB-System nur Daten übertragen könnte, dann wäre der einfache Anschluß der Geräte an den Bus schon ein großer Vorteil gegenüber anderen Systemen. Das HP-IB-System kann aber mehr! Wieviel mehr, das werden Sie in den beiden nächsten Kapiteln erfahren.

Steuerung und Überwachung des HP-IB-Systems

Ebenso wie der Vorsitzende eines Komitees bestimmte Rechte gegenüber den Komitee-Mitgliedern haben muß, braucht auch der System-Controller unabdingbare Vorrechte zum Eingriff in das Bus-System.

Wenn eine Versammlung zu 'platzen' droht, weil z.B. mehrere Leute zugleich reden, oder weil sich ein Redner nicht das Wort entziehen lassen will, dann muß der Vorsitzende die Versammlung zur Ordnung rufen. Meistens wird dann eine große Glocke heftig geschwungen. Weil nicht auszuschließen ist, daß auch das HP-IB-System sich 'regelwidrig' aufführt, kann der System-Controller (als einziges Gerät!) die Neuordnung des Systems veranlassen: Er gibt ein **Interface Clear** aus.
Dadurch werden alle Aktivitäten auf dem Bus beendet, alle Geräte verlieren ihre Rechte als Talker oder Listener und die vorher evtl. abgegebene Controller-Funktion wird vom System-Controller wieder übernommen, der nun selbst aktiv wird. Bei den üblichen Bus-Operationen wendet man Interface Clear (IFC) nur an, wenn unvorhergesehene Zustände oder Störungen den Bus-Betrieb beeinflussen.

Mit dem **Device-Clear**-Befehl lassen sich mehrere, aber auch alle Geräte auf den Zustand zurücksetzen, den sie beim Einschalten hatten. Diesen DCL-Befehl kann nur der aktive Controller ausgeben.

Die Übergabe der Controller-Funktion an ein anderes Gerät muß gut bedacht sein, weil dieses Gerät gewisse Eigenschaften haben muß, die es als Controller braucht. Auch der 'stellvertretende Vorsitzende' muß Format haben, wenn er den Vorsitz gut ausüben soll. Mit dem **Take Control**-Befehl kann der aktive Controller seine Aufgabe an ein anderes dazu fähiges Gerät abgeben, wenn er sich momentan um andere Dinge kümmern muß. Der TCT-Befehl wird über den Bus vom (noch) aktiven Controller zu dem Gerät geschickt, das die Conroller-Funktion von nun an übernehmen soll. Es ist dabei klar, daß sich manche Geräte, wie z.B. ein Drucker, nicht als aktiver Controller eignen. Der Controller muß wenigstens einige auf andere Geräte gerichtete Aktivitäten entfalten können.

Manchmal ist es wichtig, daß bestimmte Geräte mit ihren Aktivitäten gleichzeitig beginnen: Denken Sie z.B. an Messungen an mehreren Punkten einer Schaltung. Die betroffenen Instrumente müssen gleichzeitig ausgelöst ('getriggert') werden, wozu der Controller den **Group Execute Trigger**-Befehl senden muß.

Eine andere Art der Geräte-Beeinflussung bezieht sich auf die Fernbedienung. Normalerweise hat ein Voltmeter (als Beispiel) auf seiner Frontplatte Schalter, Skalen und drehknöpfe, mit denen der Benutzer Betriebsart und Meßbereich einstellen kann. Müßte man solche Einstellungen nun am Gerät selbst vornehmen, dann wäre das Ziel einer Automatisierung nicht zu erreichen. Hier muß eine Fernbedienung ermöglicht werden, wozu der **Remote Enable**-Befehl gut geeignet ist.

Die Umschaltung auf Ferneinstellung nimmt der aktive Controller vor, indem er den **Remote Enable**-Befehl ausgibt.

Es gibt dabei aber noch ein Problem: Was passiert, wenn ein zufällig 'Vorbeikommender' die Ferneinstellung wieder ausschaltet? Das ist bei vielen Geräten über einen Schalter auf der Frontplatte möglich. Die Ausgabe zweifelhafter Daten könnte die Folge sein. Es ist also nötig, die direkte Betätigung zeitweise zu sperren. Dies geschieht mit dem **Local Lockout**-Befehl.

Natürlich muß diese Sperre auch aufzuheben sein, damit die Geräte auch wieder manuell eingestellt werden können. Dann wiederum muß aber die Ferneinstellung blokkiert werden, damit die am Gerät direkt vorgenommenen Einstellungen nicht mit der Ferneinstellung verändert werden können. Dazu dient der **Go To Local**-Befehl.

Damit ist die Aufzählung der vom Controller ausgehenden Aktivitäten beendet. Es fehlt lediglich noch ein Punkt, um alle Steuer-Möglichkeiten des HP-IB-Systems beschrieben zu haben: Die zum System zusammengefaßten Geräte müssen auch von sich aus den Controller ansprechen können. Damit beschäftigt sich das nächste Kapitel.

Die Behandlung von Geräte-Meldungen

Wenn bei einem der Geräte ein Problem entsteht oder wenn es eine Meldung an den Controller ausgeben will, muß er diesen Umstand erfahren, wie auch Komitee-Mitglieder dem Vorsitzenden durch Handzeichen anzeigen, daß sie etwas sagen möchten. Der Vorsitzende kann dann zum passenden Zeitpunkt auf diese Meldungen reagieren.

Dem Handaufheben entspricht im HP-IB-System die **Service Request** -Meldung. Ein Drucker kann z.B. diese Meldung bei Störungen des Druckbetriebes ausgeben, um dem Verlust von Daten vorzubeugen. Vorsitzender wie Controller müssen ab und zu auf solche Meldungen reagieren und dabei zwei Dinge herausfinden:

1. Wer hat sich (alles) gemeldet?

2. Welche Anliegen bringt jeder Einzelne vor?

Wenn ein guter Vorsitzender alle Komitee-Mitglieder mit Namen kennt, kann er bei passender Gelegenheit jeden einzeln nach seinen Wünschen fragen; und wenn zuviel Wortmeldungen auf einmal erfolgen, notiert er sich eben die einzelnen Namen. Ein entsprechendes Verfahren läßt sich beim HP-IB-System nur für wenige Geräte innerhalb des Bus-Systems anwenden: Bis zu 8 Geräte kann der Controller gleichzeitig auf 'Wortmeldungen' abfragen und dabei auch erkennen, welche Geräte sich gemeldet haben. Für diese **Parallel Poll** genannte Abfrage sind aber an den Geräten gewisse Voraussetzungen und Vorbereitungen nötig (wie sie für den Rechner auf den Seiten IB-10/11 beschrieben sind).

Da dieses Verfahren nicht auf die Zahl von Geräten ausgedehnt werden kann, die in einem Bus-System maximal vereinigt sein können und außerdem die notwendigen Voraussetzungen vom Standard IEEE-488 nur empfohlen und nicht gefordert werden, besitzt das HP-IB-System noch ein anderes, etwas langsameres Abfrage-Verfahren, bei dem grundsätzlich die Zahl der abgefragten Geräte nur durch die Bus-Struktur begrenzt ist. Dabei löst bereits eine einzige 'Wortmeldung' eine serielle Abfrage aller Geräte aus, die bis zum Auffinden aller 'Wortmeldungen' fortgesetzt wird.

Die Abfrage der einzelnen Geräte geschieht zeitlich nacheinander, wodurch sich die Bezeichnung **Serial Poll** erklärt. Der Controller fragt bei diesem Verfahren die einzelnen Geräte nicht danach, ob sie eine Meldung abgegeben haben, sondern gleich nach den Gründen der Meldung. Diese Frage löscht beim einzelnen Gerät die Meldung, weil es diese ja an den Controller abgeben konnte und jetzt von ihm eine Reaktion erwartet.

Das Programm im Controller entscheidet dann darüber, wann und in welcher Reihenfolge auf die einzelnen Meldungen reagiert wird.

Die Ursachen für eine Meldung können bei jedem Gerät völlig verschieden sein. Ein Drucker kann z.B. außer dem Ende des Papiers auch anzeigen, daß irgendein Deckel nicht geschlossen wurde oder daß das Farbband klemmt. Für ein Voltmeter kann eine Bereichsüberschreitung oder ein unzulässiger Befehl Grund für eine Meldung sein.

Durch entsprechende Ausarbeitung der ausgelösten Routinen läßt sich so eine sehr eingehende Überwachung aller Funktionen der Peripherie-Geräte durchführen, obwohl der Aufbau des Systems sehr einfach und praktisch ohne Probleme ermöglicht wird:

Mechanische und elektrische Anschluß-Probleme sind durch die Normung ausgeräumt, genau so die Abstimmung der Datenübermittlung hinsichtlich Codierung und Tempo. Der nächste Abschnitt zeigt, wie ein HP-IB-System einzurichten ist und in welcher Reihenfolge die dafür typischen Operationen auszuführen sind.

Notizen:

HP-IB-Operationen mit dem I/O-ROM

Einführung

Das HP-IB-System ist zum Betrieb weniger Geräte (Controller und 1 Peripherie-Gerät) genau so gut geeignet, wie für komplexe Steuerfunktionen in Systemen von beträchtlichem Umfang (mehrere Controller mit einer Vielzahl von Geräten). Dazu ist in den meisten Fällen das I/O-ROM nötig, weshalb von nun an angenommen wird, daß dieses ROM dem System zur Verfügung steht.

Die folgenden Kapitel sind auf Benutzer zugeschnitten, denen die Grundlagen des HP-IB-Systems bekannt sind, die aber Informationen über die praktische Anwendung brauchen. Deshalb stehen hier keine langen Aufzählungen der Anweisungen mit Syntax, sondern Beispiele, die die richtige Reihenfolge der notwendigen Operationen der System-Funktionen vom HP-IB verdeutlichen. Benutzen Sie die folgenden Kapitel zum Entwurf Ihrer Programme und schauen Sie bei Bedarf in die Syntax-Übersicht im Anhang A dieses Buches, um Ihre Programmzeilen exakt zu formulieren.

Einschalten und Prüfen

Einzelheiten über den Anschluß der Geräte finden Sie im IB-Abschnitt 2, weshalb wir uns hier auf das Wichtigste beschränken können.

Wenn das Interface eingesetzt ist und die einzelnen Geräte mit den HP-IB-Kabeln angeschlossen sind, sollte man sich mit einer kurzen Routine über den Status der einzelnen Komponenten Gewißheit verschaffen. Wenn Sie von den angeschlossenen Geräten die Adressen kennen, läßt sich dafür ein Programm dieser Art benutzen:

```
10 STATUS 7,0 ; A,B,C,D,E
20 PRINT "Interface-Status = ";A,B,C,D,E
30 A=SPOLL(722)
40 B=SPOLL(703)
50 C=SPOLL(713)
60 PRINT "Geraet Nr. 22, Status = ";A
70 PRINT "Geraet Nr. 03, Status = ";B
80 PRINT "Geraet Nr. 13, Status = ";C
90 END
```

In Zeile 10 wird der Status vom IB-Interface selbst ermittelt und in Zeile 20 angezeigt. Hier sollten sich mit 1, 0, 64, n und 53 die Standard-Werte ergeben. In den Zeilen 30 bis 50 werden mit SPOLL seriell die Status-Bytes der 3 Geräte abgefragt. Die Ergebnisse werden mit den Geräte-Adressen in den Zeilen 60 bis 80 ausgedruckt. Wenn die Adressen der angeschlossenen Geräte nicht bekannt sind, hilft das folgende Programm-Beispiel weiter:

```
10 S=7 ! Variable S ist der Auswahl-Code
20 SET TIMEOUT S ; 500
30 ON TIMEOUT S GOTO 100
40 FOR I=0 TO 31
50 DISP "Abfrage Geraet Nr. ";I
60 S1=SPOLL(S*100+I)
70 PRINT "Geraet ";I;"STATUS = ";S1
80 NEXT I
90 STOP
100 ABORTIO 7
110 PRINT "Geraet ";I;" nicht vorhanden!"
120 GOTO 80
130 END
```

Alle vorhandenen Geräte werden die serielle Abfrage mit ihren Status-Bytes beantworten. Diese werden ausgedruckt. Beachten Sie bitte, daß jedes Gerät einen anderen Status melden kann. Zur Deutung dieser Meldungen müssen Sie die Geräte-Handbücher befragen. Sie werden unter dem Stichwort 'Serial Poll Response Byte' oder 'Reaktion auf serielle Abfrage' die nötigen Angaben finden. Nachdem Sie jetzt den Zustand des Systems kennen (hoffentlich arbeitet alles!), können wir nun mit der Steuerung der zum System gehörenden Geräte beginnen.

Steuerung des Bus-Systems

Alle in diesem Abschnitt erwähnten Operationen setzen voraus, daß der Rechner mit seinem HP-IB-Interface System-Controller ist (mit dieser Einstellung werden die Geräte ausgeliefert). Eine typische Operation des HP-IB-Systems ist die Umschaltung der angeschlossenen Geräte auf Ferneinstellung: Die meisten Geräte können entweder direkt von Hand (über die Tasten, Knöpfe und Schalter) oder indirekt (über den Bus) eingestellt werden. Um den Fernbedienungs-Modus einzuschalten, muß die 'Remote Enable'-Leitung (REN) auf 'wahr' gesetzt und das Gerät zum Listener erklärt werden.
Die REN-Leitung wird jedesmal auf 'wahr' gesetzt, wenn der Rechner eingeschaltet wird oder den RESET-Befehl ausführt. Das gleiche bewirkt eine REMOTE-Anweisung. Ein Gerät wird Listener, sobald eine Anweisung mit seiner Listener-Adresse ausgeführt wurde. Mit der folgenden Anweisung werden die Geräte 22 und 10 auf Ferneinstellung geschaltet:

```
10 REMOTE 722,710
```

Um auszuschließen, daß keines der Geräte durch direkte Umschaltungen von Hand dem Einfluß der Ferneinstellung wieder entzogen werden kann, sollte die REMOTE-Anweisung durch einen LOCAL LOCKOUT-Anweisung (LLO) ergänzt werden:

```
10 REMOTE 713,722
20 LOCAL LOCKOUT 7
```

Jetzt ist das System auf Ferneinstellung festgelegt und alle Schalter an den Peripherie-Geräten sind wirkungslos. Als nächstes müssen jetzt die Geräte auf die gewünschten Betriebsarten eingestellt werden. Das geschieht mit einer einfachen OUTPUT-Anweisung für das jeweilige Gerät, wie es das nächste Beispiel zeigt. Wir nehmen dazu an, daß unter Auswahl-Code 7 und Gerätenummer 22 ein HP-Digital-Voltmeter 3455A auf den Gleichspannungsmeßbereich 0,1 V und Dauermessung eingestellt werden soll. Die Betriebsanweisung verlangt dazu die Ausgabe der ASCII-Sequenz "F1R1T1" an das Gerät. Die Anweisung kann dann so aussehen:

```
50 OUTPUT 722 ; "F1R1T1"
```

Beachten Sie, daß nur die ASCII-Zeichen in der OUTPUT-Anweisung die Einstellung bewirken! Für andere Betriebszustände sind nur andere ASCII-Sequenzen nötig. Die Wirkung solcher Sequenzen nennt man 'geräteabhängig', weil ihre Wirkung davon abhängt, welches Gerät die Sequenz empfängt.

Wenn das System noch ein zweites Gerät enthält, das ebenfalls eingestellt werden muß, kann das Programm so aussehen:

```
10 REMOTE 722,713
20 LOCAL LOCKOUT 7
30 ! Auswahl Gs-Bereich 0,1 V an Gerät 22
40 ! (Digital-Voltmeter HP 3455A)
50 OUTPUT 722 ; "F1R1T1"
60 ! Auswahl 1kHz Auflösung an Gerät 13
70 ! (Frequenzmesser HP 5328A)
80 OUTPUT 713 ; "PF4G3S1S5T"
```

Sobald die Geräte für ihre Anwendung programmiert sind, kann man auch Ergebnisse erwarten. Jetzt können wir die ENTER-Anweisung mit entsprechender Adressierung zum Abruf der Daten von den Geräten einsetzen. Dazu müssen wir unser bisheriges Programm so ergänzen:

```
 90 ! Übernimmt Spannung nach V
100 ENTER 722 ; V
110 ! Übernimmt Frequenz nach F
120 ENTER 713 ; F
130 PRINT "Spannung : ";V;" Volt"
140 PRINT "Frequenz : ";F;" Hz"
150 END
```

Das gegebene Beispiel war insofern einfach, als vom Voltmeter und vom Frequenzmesser immer die momentan ermittelten Meßwerte abgerufen wurden. Wir wollen jetzt annehmen, daß uns die Meßwerte für Spannung und Frequenz nur dann interessieren, wenn beide in einer Meßschaltung zwar an verschiedenen Stellen aber zum gleichen Zeitpunkt ermittelt wurden. Als Beispiel sei hier eine Spannungs-Frequenz-Wandlung erwähnt, die mit der Meßschaltung untersucht werden soll. Es ist also nötig, die beiden Meßgeräte zu 'triggern'.

Um das zu erreichen, muß bei beiden Geräten die Betriebsart von Dauermessung auf Triggerung geändert werden. Wir brauchen dazu andere OUTPUT-Anweisungen. Außerdem muß vor den ENTER-Anweisungen die TRIGGER-Anweisung eingefügt werden. TRIGGER- und ENTER-Anweisungen stehen in einer Schleife, damit nach Abschluß der Meßreihe ein Gesamteindruck vom Übertragungsverhalten des Prüflings vorliegt:

```
 10 REMOTE 713,722
 20 LOCAL LOCKOUT 7
 30 ! Auswahl Gs-Bereich 0,1 V an Gerät 22
 40 ! (Digital-Voltmeter HP 3455A)
 50 OUTPUT 722 ; "F2R2T2T3"
 60 ! Auswahl 1kHz Auflösung bei Gerät 13
 70 ! (Frequenzmesser HP 5328A)
 80 OUTPUT 713 ; "PF4G3R"
 90 FOR I=1 TO 100
100 TRIGGER 713,722 @ RESUME 7 ! ATN wieder auf 'falsch'!
110 ! Übernimmt Spannung nach V
120 ENTER 722 ; V
130 ! Übernimmt Frequenz nach F
140 ENTER 713 ; F
150 PRINT "Spannung : ";V;" Volt"
160 PRINT "Frequenz : ";F;" Hz"
170 NEXT I
160 END
```

Die ENTER- und OUTPUT-Anweisungen waren hier Mittel zur Datenabfrage und zur Einstellung der Betriebsarten der Geräte. Das nächste Kapitel behandelt das Gebiet der Datenübertragung innerhalb des HP-IB-Systems aus allgemeinerer Sicht.

Allgemeines über die Datenübertragung im HP-IB-System

Einführung

Die Übertragung von Daten läßt sich mitunter sehr einfach durchführen, manchmal wirft diese Operation aber unerwartet große Probleme auf. Dieses Kapitel erklärt die einfachsten Verfahren zuerst und geht erst danach zu den leistungsfähigeren, aber auch komplizierteren Methoden über. Gemeint ist hier die TRANSFER-Anweisung, die für das HP-IB-System typisch ist.

Einfache Ausgabe-Operationen

Um Daten vom Rechner an ein Gerät des HP-IB-Systems zu übergeben, wird grundsätzlich die OUTPUT-Anweisung benutzt. Es ist aber auch möglich mit der PRINT-Anweisung Daten an ein Gerät des Systems zu geben. Es muß dann aber das anzusprechende Gerät vorher mit einer PRINTER IS-Anweisung als System-Drucker definiert werden. Diese PRINTER IS-Anweisung kann am Anfang eines BASIC-Programms stehen, wodurch alle PRINT-Anweisungen, die danach folgen, zum angegebenen Gerät gelangen, bis ein anderes Gerät per Programm die Rolle des System-Druckers zugewiesen bekommt. Wenn nur ein einziges Gerät Daten erhalten muß, ist das recht bequem, wie das folgende Beispiel zeigt:

```
10 ! Der HP-IB-Drucker hat Adresse 01
20 PRINTER IS 701
30 FOR I=0 TO 1 STEP .01
40 PRINT SIN(I)
50 NEXT I
60 END
```

Die Methode wird aber um so mühsamer, je mehr Geräte anzusprechen sind. Die Wahl der Betriebsart von 2 Meßgeräten sieht dann so aus:

```
10 PRINTER IS 722
20 ! Weist Daten an DVM
30 PRINT "Programmier-Code"
40 PRINTER IS 713
50 ! Weist Daten an Freq.-Messer
60 PRINT "Programmier-Code"
70 END
```

Diese Methode braucht (ohne die Erklärungen) vier Anweisungen. Mit zwei OUTPUT-Anweisungen hat man den gleichen Erfolg:

```
10 OUTPUT 722 ; "Programmier-Code"
20 OUTPUT 713 ; "Programmier-Code"
```

Die OUTPUT-Anweisung läßt sich auch formatiert ins Programm setzen, wie das ausführlich im I/O-ROM-Abschnitt 3 erklärt wurde. Die HP-IB-Version der Anweisung sieht dann so aus:

```
OUTPUT 722 USING "nA" ; "Programmier-Code"
```

Eine für das HP-IB-System formulierte Ausgabe-Anweisung unterscheidet sich von den sonst üblichen Formulierungen nur durch die Adresse, die darüber informiert, wer die Daten bekommen soll. Wenn Sie mehr Einzelheiten über den Umgang mit der OUTPUT-Anweisung erfahren wollen, schlagen Sie im I/O-ROM-Abschnitt 2 dieses Buches nach. Weiter entwickelte Techniken finden Sie im IB-Abschnitt 5 (HP-IB für den Experten) und im I/O-ROM-Abschnitt 8, der die TRANSFER-Anweisung ausführlich beschreibt.

Einfache Eingabe-Operationen

Mit der ENTER-Anweisung lassen sich am einfachsten Daten von einem Gerät des HP-IB-Systems in den Rechner bringen. Die Daten von einem Universal-Intervall-Geber (HP 5307A) werden durch die folgende Anweisung vom Rechner angenommen:

```
ENTER 716 ; A$
```

Die Anweisung wird durch das Zeilenvorschub-Symbol beendet, das der Intervall-Geber als letztes Zeichen ausgibt. Andere Geräte markieren das letzte Zeichen dadurch, daß die EOI-Leitung gleichzeitig auf 'wahr' gesetzt wird. Für diesen Fall muß die ENTER-Anweisung (z.B. für das Gerät 06) wie folgt abgeändert werden:

```
ENTER 706 USING "%,K" ; A$
```

Der Spezifikator "%" sorgt zwar dafür, daß das EOI-Signal als Ende der Datenübertragung gewertet wird, wenn nun aber inmitten der Daten ein Zeilenvorschub-Symbol enthalten ist, dann schließt dieses mit Vorrang die (noch unvollständige) Datenzuweisung für die String-Variable A$ ab. Diese Unsicherheit wird mit der folgenden Anweisung beseitigt, weil diese Form nur EOI als Abschluß anerkennt und auch die im String eingestreuten Zeilenvorschub-Symbole (wie die übrigen Zeichen) der String-Variablen zuweist:

```
ENTER 706 USING "#%,K"; A$
```

Näheres über die Anwendung der ENTER-Anweisung steht in den I/O-ROM-Abschnitten 2 und 3 dieses Handbuches, wo die ENTER-Anweisung in allen Einzelheiten besprochen wird. Weiter entwickelte Techniken, die flexibler sind, sich der TRANSFER-Anweisung bedienen oder Daten in den Rechner holen, auch wenn er nicht die Controller-Funktion hat, sind im folgenden Kapitel beschrieben.

Erweiterte I/O-Operationen

Einführung

Wenn Sie diesen Abschnitt beginnen, dann tun Sie es wahrscheinlich deshalb, weil Sie ein Problem haben, das mit den einfachen OUTPUT- oder ENTER-Anweisungen nicht zu lösen ist. Dieser Abschnitt beschäftigt sich mit vier verschiedenen Problemen der Datenübertragung:

1. Datenübertragung mit einem Rechner ohne Controller-Funktion.
2. ENTER- und OUTPUT-Anweisungen, die mehrere Listener betreffen.
3. Vom Rechner ausgelöste Übertragung von Daten, die ihn selbst aber nicht betreffen.
4. Benutzer-spezifische TRANSFER-Verfahren, unter Einschluß von Interrupt und Fast-Handshake.

Selbst-Adressierung ohne Controller-Funktion

Diese Eingabe-/Ausgabe-Technik wird verwendet, wenn der Rechner (auch als System-Controller) die Controller-Funktion an ein anderes Gerät abgegeben hat oder wenn der Rechner (ohne System-Controller zu sein) auch von keinem anderen Gerät zum aktiven Controller bestimmt wurde.

Der Rechner (und auch jedes andere Gerät im System) muß mit der Ausgabe bzw. Aufnahme von Daten warten, bis er als Talker bzw. Listener adressiert ist. Es gibt drei Verfahren, diese Adressierung zu bemerken:

Die erste Methode besteht im einfachen Warten, das ist bequem, blockiert aber den Rechner während der Wartezeit für jede andere Tätigkeit und ist deshalb nur für kleine Systeme anzuraten, bei denen die Wartezeiten kurz sind. OUTPUT-, ENTER- oder TRANSFER-Anweisungen enthalten dabei nur den Auswahl-Code aber keine Geräte-Adresse. Die Anweisungen laufen automatisch, sobald die passende Adressierung erfolgt ist und könnten so aussehen:

```
OUTPUT 7 ; A$...    oder    TRANSFER A$ TO 7
```

Es wird die Adressierung als Talker abgewartet, dann werden die in der Ausgabe-Liste definierten Daten oder der Inhalt des I/O-Buffers ausgegeben.

```
ENTER 7 ; A$...     oder    TRANSFER 7 TO A$
```

Es wird die Adressierung als Listener abgewartet, dann werden die angebotenen Daten den Variablen der Eingabe-Liste oder dem I/O-Buffer zugewiesen.

Bei der zweiten Methode prüft der Rechner in gewissen Abständen den Interface-Status, um dadurch eine Adressierung zu bemerken. Im Status-Register SR5 wird das Bit 4 gesetzt, wenn das Interface als Talker adressiert wurde. Entsprechend wird bei der Adressierung zum Listener im gleichen Register das Bit 6 gesetzt. Bis zur Adressierung kann der Rechner zwischen den Prüfungen andere Aufgaben erledigen. Sobald eine Adressierung erkannt wird, führt der Rechner die dann mögliche Ausgabe oder Eingabe aus und kehrt dann, unter Beibehaltung der periodischen Abfrage, zu seiner üblichen Arbeit zurück. Im folgenden Programmbeispiel wird der Stand eines Zählers weitergeschaltet, angezeigt und jedesmal geprüft, ob schon eine Adressierung erfolgte. Wenn nicht, wird weitergezählt und angezeigt, wenn ja, wird der Zählerstand an das HP-IB ausgegeben.

```
10 ! Zeilen 50/60 prüfen, ob als Talker adressiert
20 I=0
30 I=I+1
40 DISP I
50 STATUS 7,5 ; S ! SR5 lesen!
60 IF BIT(S,4)=0 THEN GOTO 30 ! "TA"-Bit gesetzt?
70 OUTPUT 7 ; I
80 GOTO 20
90 END
```

Beim nächsten Beispiel wird geprüft, ob inzwischen eine Adressierung als Listener (für eine ENTER-Anweisung) erfolgte. Der Unterschied zur vorigen Anweisung liegt nur darin, daß hier ein anderes Bit geprüft wird.

```
10 STATUS 7,5 ; S! SR5 lesen!
20 IF BIT(S,6)=0 THEN DISP "Kein Listener!"
```

Sehen Sie bitte in den IB-Abschnitt 5 (HP-IB für den Experten) dieses Handbuches, wenn Sie weitere Einzelheiten über die Status-Register erfahren wollen!

Die dritte Methode, auf eine Adressierung zu reagieren, ohne dabei die anderen Tätigkeiten einschränken zu müssen, benutzt die Interrupt-Technik. Dabei löst das Interface im Rechner einen Interrupt aus, sobald eine Adressierung erfolgt. Das hat für den Programmierer den Vorteil, daß er sein Programm nicht kunstvoll um die periodisch durchzuführende Status-Abfrage herumbauen muß. Das Programm kann ganz normal ablaufen, bis eine Adressierung als Talker oder Listener vorliegt. Das folgende Beispiel soll das verdeutlichen:

```
10 ! Bereitet Interrupt und Zeilen-End-Verzweigung
20 ! bei Listener-Adressierung vor:
30 ON INTR 7 GOSUB 100
40 ENABLE INTR 7 ; 64
50 ! Zeilen 30/40 ermöglichen Zeilen-End-Verzweigung
60 I=0
70 I=I+1
80 DISP I
90 GOTO 70
100 ! Service-Routine für Zeilen-End-Verzweigung
110 ! Status-Register SR1 wird durch Abfrage gelöscht:
120 STATUS 7,1 ; S
130 ENTER 7 ; A$
140 PRINT "Empfangen wurde: ";A$
150 ! Interrupt-Maske wird erneuert:
170 ENABLE INTR 7 ; 64
180 RETURN
190 END
```

In den Zeilen 30/40 wird festgelegt, daß durch die Adressierung als Listener ein Interrupt ausgelöst werden soll. Zeile 30 enthält dafür die Sprungadresse. In den Zeilen 60 bis 80 steht hier als ganz einfaches Beispiel wieder eine Zählschleife, die der Rechner ausführt, bis der Interrupt auftritt. In Zeile 100 beginnt die Routine, die nach der Adressierung als Listener ablaufen soll. Die ENTER-Anweisung enthält auch hier nur den Auswahl-Code und keine Geräte-Adresse! Das bedeutet etwa: "Ich warte, bis ich als Listener adressiert bin und werde dann die angebotenen Daten aufnehmen". Die Zeile 170 erneuert dann die Interrupt-Genehmigung für den Fall, daß erneut eine Adressierung zum Listener erfolgen sollte.

OUTPUT-Anweisungen lassen sich auf ähnliche Weise schreiben nur, daß dann ein anderer Wert (16) für die ENABLE INTR-Maske einzusetzen ist und die Anweisung OUTPUT 7 ; <Wert> an die Stelle der ENTER-Anweisung im obigen Beispiel tritt. Schauen Sie bitte in den IB-Abschnitt 5 (HP-IB für den Experten), wenn Sie alle Einzelheiten über Status- und Steuer-Register erfahren wollen, über die das Interface in seinem Interrupt-Verhalten beeinflußt werden kann.

Jetzt wollen wir einmal annehmen, daß unser Rechner zwar kein aktiver Controller ist, trotzdem aber Daten sowohl aufnehmen, als auch ausgeben soll. Was ist dann zu tun? Die grundsätzliche Folge der Operationen bleibt die gleiche, wir müssen nur zusätzlich prüfen, welche Art der Adressierung vorliegt. Das folgende Programm enthält diese Prüfung. Zeile 100 ermittelt den Wert des Status-Registers SR1 und in den Zeilen 110 bzw. 120 wird entschieden, ob eine Verzweigung zur Ausgabe oder zur Aufnahme von Daten vorzunehmen ist:

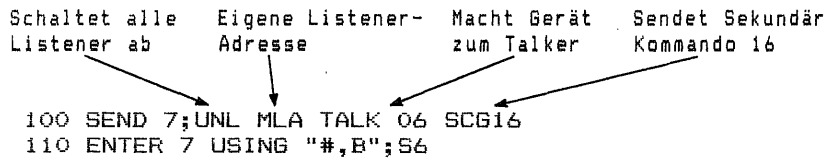
```
 10 ! Interrupt bei Talker- u. Listener-Adressierung
 20 ENABLE INTR 7 ; (64+16)
 30 ON INTR 7 GOSUB 100
 40 I=0
 50 I=I+1
 60 DISP I
 70 GOTO 50
 80 ! Es folgen der Status-Test und
 90 ! die Entscheidung über die Routine!
100 STATUS 7,1 ; S ! Liest und löscht SR1
110 IF BIT(S,4) THEN GOTO 150 ! OUTPUT-Routine
120 IF BIT(S,6) THEN GOTO 180 ! ENTER-Routine!
130 PRINT "Fehler im Status-Register!" @ GOTO 210
150 ! Routine für OUTPUT-Operation!
160 OUTPUT 7 ; I
170 GOTO 210
180 ! Routine für ENTER-Operation!
190 ENTER 7 ; A$
200 PRINT "Empfangen wurde: ";A$
210 ENABLE INTR 7 ; (64 + 16)
220 RETURN
```

Selbstverständlich dürfen bei den drei angegebenen Verfahren die üblichen IMAGE-Anweisungen und auch Festlegungen über die Abschlußbedingungen für Felder und Anweisungen verwendet werden!

Maßgeschneiderte Bus-Sequenzen

Hin und wieder kann es nötig werden, über den Bus zu einem bestimmten Gerät eine besonders zugeschnittene Sequenz zu schicken, die sich von den üblichen Sequenzen stark unterscheidet. Mit der SEND-Anweisung lassen sich solche Spezial-Botschaften übermitteln, durch die man 'Sonderabsprachen' mit den Geräten treffen kann. Als Preis für diese Möglichkeit müssen Sie aber jedes Zeichen der Sequenz selber formulieren: automatisch passiert da nichts mehr!

Als Beispiel für eine 'maßgeschneiderte' Sequenz soll mit der folgenden Anweisung ein Sekundär-Befehl an einen Digitalisierer (HP 9874) übermittelt werden, der die Adresse 06 hat und über Auswahl-Code 7 zu erreichen ist. Der Sekundär-Befehl 16 veranlaßt den Digitalisierer, seinen Status (1 Byte) auszugeben, der mit der folgenden ENTER-Anweisung gelesen wird. Beachten Sie, daß hierbei auch die ENTER-Anweisung keine Geräte-Adresse enthalten darf!



Auch andere Operationen, einschließlich des An- und Abschaltens der Parallel-Abfrage (PPC und PPU) lassen sich auf diese Weise durchführen.

Sie können die im IB-Abschnitt 5 (HP-IB für den Experten) befindlichen Tabellen benutzen, um die nötigen Kodierungen für Ihre Pläne zu erfahren.

Gleichzeitige Datenübermittlung an mehrere Listener

Wenn mehrere Geräte die ausgegebenen Daten empfangen sollen, müssen deren Adressen zur Listener-Gruppe zusammengefaßt werden. Das muß bei ENTER- und OUTPUT-Anweisungen auf verschiedene Weise geschehen, wie in diesem Kapitel gezeigt wird.

Sowohl bei OUTPUT- als auch bei Ausgabe-TRANSFER-Anweisungen ist die Möglichkeit vorgesehen, mehrere Listener gleichzeitig anzusprechen, wie das im nachfolgenden Beispiel gezeigt wird: Wir wollen annehmen, daß ein String (B1$) an einen Drucker (Adresse 04) und an einen Rechner der Serie 80, der nicht Controller ist und die Adresse 20 hat, übermittelt werden soll. Als OUTPUT-Anweisung sieht das dann so aus:

```
OUTPUT 704,720 ; B1$
```

Für eine TRANSFER-Ausgabe muß man dagegen so formulieren:

```
TRANSFER B1$ TO 704,720 INTR
```

Für beide Anweisungen gilt lediglich die Einschränkung, daß die gemeinsam adressierten Geräte den gleichen Auswahl-Code (in diesem Fall 7) haben müssen.

Auch durch eine ENTER-Anweisung ist es möglich, die aufzunehmenden Daten gleichzeitig mehreren Geräten zugänglich zu machen. Das geht aber nicht so einfach, da die ENTER-Anweisung nur eine einzige Adresse, nämlich die des Talkers, enthält und das HP-IB-System immer nur einen Talker zuläßt. Der Bus muß also besonders dazu vorbereitet werden, um dann eine ENTER-Anweisung folgen zu lassen, die nur den Auswahl-Code enthält. Dabei behält der Bus den zuvor geschaffenen Zustand.

Wenn z.B. das Gerät 07 (Voltmeter) seine Daten sowohl zum Gerät 13 (Drucker) als auch zum Gerät 04 (Massenspeicher) und außerdem zum aktiven Controller übertragen soll, muß der Bus gemäß der SEND-Anweisung vorbereitet werden: Es werden mit UNL alle Listener abgeschaltet, auf TALK bzw. LISTEN folgen die nötigen Adressen, der aktive Controller muß zusätzlich mit MLA (My Listen Address) adressiert werden. Die ENTER-Anweisung wickelt dann die Übertragung ab:

```
10 SEND 7 ; UNL TALK 7 LISTEN 13,4 MLA
20 ENTER 7 ; V
```

Für eine TRANSFER-Eingabe läßt sich eine ähnliche Sequenz verwenden, in diesem Falle zur Übertragung von Daten per Interrupt in den I/O-Buffer T$ des als Controller aktiven Rechners, wobei wegen der Dimensionierung des I/O-Buffers auf den I/O-ROM-Abschnitt 8 (Zweck und Benutzen der Buffer) hingewiesen wird:

```
10 DIM T$[88]
20 IOBUFFER T$
30 SEND 7 ; UNL TALK 7 LISTEN 13,4 MLA
40 TRANSFER 7 TO T$ INTR
```

Eine andere Spielart der Datenübertragung an mehrere Listener liegt vor, wenn der Rechner dabei weder Daten sendet noch Daten empfängt: Es werden dann im Bus Daten zwischen den Geräten ausgetauscht. Um z.B. Daten vom Gerät 11 zu den Geräten 23, 04 und 07 zu übermitteln, lassen sich die folgenden Anweisungen benutzen:

```
10 SEND 7 ; UNL TALK 11 LISTEN 23,04,07
20 RESUME 7
```

Die Übertragung beginnt automatisch, sobald durch die RESUME-Anweisung die ATN-Leitung auf 'falsch' gesetzt wird, als Zeichen dafür, daß alle Befehle gesendet wurden und nun Daten folgen. Weil der Rechner hier nicht als Listener adressiert ist, sind weder ENTER- noch OUTPUT-Anweisung zum Auslösen der Übertragung nötig. Selbstverständlich ist dieses Verfahren auch anwendbar, wenn nur ein einziges Gerät die Daten erhalten soll.

Es gibt hier aber noch ein Problem: Woran erkennt der Rechner den Abschluß der Übertragung? Am besten dadurch, daß wir ihn in Zeile 10 zum Listener machen (MLA) und ein Eingabe-Transfer per Interrupt vereinbaren (wie im Beispiel davor), weil dann das Ende der Übertragung eine Zeilen-End-Verzweigung auslösen kann. Hier ein Beispiel dafür:

```
 5 DIM T$[1008]
 6 IOBUFFER T$ @ ON EOT 7 GOSUB 1000
10 SEND 7 ; UNL TALK 11 LISTEN 23,4,7 MLA
20 ! Hier wird EOI als Abschluß vereinbart:
30 TRANSFER 7 TO T$ INTR ; EOI
40 RESUME 7
50 ! RESUME setzt ATN auf 'falsch': Übertragung beginnt!
```

Wie aus den Beispielen zu ersehen ist, dürfen die einstelligen Geräte-Adressen mit oder ohne führende Null eingegeben werden.

Alternative Übertragungsverfahren

Wie bereits mehrmals erwähnt wurde, besteht bei der TRANSFER-Anweisung die Wahl zwischen dem Interrupt-Verfahren (übertragung und Programm laufen gleichzeitig nebeneinander) und dem Fast-Handshake-Verfahren, das bei Unterbindung aller anderen Tätigkeiten größte übertragungsgeschwindigkeit bietet. In der TRANSFER-Anweisung ist anzugeben, welches Verfahren (INTR oder FHS) benutzt werden soll. Dieses Thema ist im I/O-Abschnitt 8 (Weitere übertragungsverfahren) ausführlich erklärt. Die wichtigsten Einzelheiten dieses Abschnitts sollten zum Verstehen des nächsten Beispiels geläufig sein:

```
 10 ! Daten per Interrupt von Gerät 04 zum Buffer T$
 20 ! Zeilen-Vorschub-Signal zeigt Ende an!
 30 DIM T$[88]
 40 I=0
 50 IOBUFFER T$
 60 ON EOT 7 GOSUB 120
 70 ! Transfer vorbereiten mit LF als Abschluß:
 80 TRANSFER 704 TO T$ INTR ; DELIM 10
 90 I=I+1
100 DISP "Ich zähle: ";I
110 GOTO 90
120 PRINT "Empfangen wurde: ";T$
130 ! Vorbereitung für neuen TRANSFER:
140 IOBUFFER T$ ! Löscht Bufferinhalt
150 TRANSFER 704 TO T$ INTR ; DELIM 10 @ RETURN
160 END
```

Behandlung von Störungs-Meldungen

Einführung

Dieses Kapitel beschäftigt sich mit Störungs-Meldungen, die im System zu beliebigen Zeitpunkten auftreten können. Der Anlaß für eine Störungs-Meldung ist 'geräteabhängig', d.h. die verschiedenen Geräte haben individuelle Gründe um derartige Meldungen abzugeben. Bei einem Drucker kann z.B. das Ende des Papiers ein Grund für eine solche Meldung sein, bei einem Digitalisierer kann es die Fehlbedienung durch den Benutzer sein, usw. Beim Auftreten einer Meldung müssen immer zwei Dinge ermittelt werden:

1. Welches Gerät hat die Meldung abgegeben?
2. Was will das Gerät melden?

Im Folgenden wird gezeigt, wie dies erreicht wird.

Das Bemerken der Störungs-Meldung

Im HP-IB-Interface wird der Inhalt des Status-Registers SR2 (auch) durch die Störungs-Meldung (SRQ) beeinflußt. Dieses Register läßt sich so abfragen:

```
STATUS 7,2 ; S
```

Wenn das Bit 5 der Variablen S gesetzt ist, bedeutet dies eine 'wahre' SRQ-Leitung, d.h. eine Störung im System liegt vor. Durch das Programm kann dann entschieden werden, wie auf das SRQ zu reagieren ist (Siehe hierzu IB-Abschnitt 5 'HP-IB für den Experten', wo eine vollständige Beschreibung der Status-Register zu finden ist). Wenn die Möglichkeit zum Interrupt eingeräumt wurde, kann auch durch Abfrage von SR1 festgestellt werden, ob ein SRQ auftrat:

```
STATUS 7,1 ; S
```

Hier weist das gesetzte Bit 3 von S auf eine SRQ-Meldung hin. Als Alternative zum wiederholten Abfragen der Status-Registers SR2 oder SR1 hat man auch die Möglichkeit, eine Zeilen-End-Verzweigung zu programmieren, die durch SRQ ausgelöst wird. Das folgende Beispiel schafft die Voraussetzungen dazu:

```
10 ON INTR 7 GOSUB 200
20 ! Bit 3 des CR1 wird gesetzt:SRQ-Interrupt möglich!
30 ENABLE 7 ; 8
```

Das Programm wird dann bei jeder Störungsmeldung zu dem Unterprogramm verzweigen, das bei Zeile 200 beginnt. Als nächstes wollen wir jetzt überlegen, was in diesem Unterprogramm stehen sollte.

Das Feststellen der Störungs-Ursache

Der Hauptzweck der seriellen Abfrage (Serial Poll) ist es, dem aktiven Controller konkrete Informationen über den Zustand des abgefragten Gerätes zu geben. Das abgefragte Gerät antwortet mit einem Byte, von dem 7 Bits beliebige Bedeutungen haben können, die natürlich vorher festzulegen sind. Eine Ausnahme macht das Bit 6: Es zeigt an, ob das abgefragte Gerät momentan eine SRQ-Meldung abgibt.

Wir beziehen uns schon hier auf das nachfolgende Beispiel, bei dem das Programm zur Zeile 300 verzweigt, sobald das Gerät 15 eine Störungsmeldung abgibt. Zuerst wird in dieser Service-Routine das Status-Byte vom Gerät 15 abgefragt. Dann werden die einzelnen Bits untersucht, zuerst, um auf SRQ zu kontrollieren, und dann, um die Störung einzugrenzen.

Wir nehmen an, daß die einzelnen Bits folgende Bedeutungen haben sollen, wenn sie gesetzt sind:

Bit 0 : Papier fehlt

Bit 1 : Deckel offen

Bit 2 : Parameter außer Bereich

Bit 3 : Fehlerhafte Escape-Sequenz

Bit 4 : nicht benutzt

Bit 5 : nicht benutzt

Bit 6 : SRQ wird ausgegeben

Bit 7 : nicht benutzt

Dann könnte eine Routine für das Gerät 15 z.B. so aussehen:

```
300 ! Service fuer Geraet 15
310 S=SPOLL(715)
320 ! Prüfung, ob SRQ vorliegt:
330 IF BIT(S,6) THEN GOTO 350
340 ENABLE INTR 7 ; 8 @ RETURN
350 IF NOT BIT(S,0) THEN GOTO 370
360 PRINT "Drucker ohne Papier!"
370 IF NOT BIT(S,1) THEN GOTO 400
380 PRINT "Drucker-Deckel offen!"
400 IF NOT BIT(S,2) THEN GOTO 420
410 PRINT "Druck-Parameter außer Bereich!"
420 IF NOT BIT(S,3) THEN PRINT "Unbekannter Drucker-Fehler!"
430 PRINT "Escape-Sequenz fehlerhaft!" @ GOTO 340
```

In einem größeren System sind serielle Abfragen für jedes Gerät nötig, um zuerst das betroffene Gerät zu ermitteln und danach dessen Probleme zu erfahren. Hierzu ein kurzes Beispiel:

```
100 S=SPOLL(715)
110 IF NOT BIT(S,6) THEN GOTO 200
120 ! Zeilen 120 bis 190 betreffen Geraet 15
200 S=SPOLL(723)
210 IF NOT BIT(S,6) THEN GOTO 300
     .
     .
```

Aktivitäten als "Nicht-Controller"

Wenn der Rechner nicht aktiver Controller ist, existieren für den Programmierer einige Einschränkungen, die er zur Vermeidung von Störungen im System beachten muß: Bestimmte Operationen kann eben nur der System-Controller auslösen, andere wiederum nur der aktive Controller. Ein "Nicht-Controller" kann eben nur um die Erteilung des Talker- bzw. Listener-Status ersuchen oder mit SRQ um Aufmerksamkeit bitten.

Abgeben der Controller-Funktion

Der Rechner kann nur als aktiver Controller diese Funktion an ein dazu geignetes anderes Gerät durch die PASS CONTROL-Anweisung übergeben. Danach kann der Rechner seine Aktivität beispielsweise einem anderen HP-IB-System zuwenden oder er kann seine ganze Kapazität für die schnellste Übermittlung von Daten an einen übergeordneten Rechner verwenden. Die folgende Anweisung übergibt die Funktion des aktiven Controllers an das Gerät 20, einen anderen Rechner der Serie 80, der bisher nicht aktiver Controller war:

```
680 PASS CONTROL 720
```

Häufig wird der erste Rechner die Controller-Funktion später wiederum übernehmen wollen, es muß dazu schon vor der Abgabe der Controller-Funktion festgelegt werden, woran dieser Rechner die Rückgabe der Controller-Funktion erkennt und wie er sich dann verhalten soll. Es wird hier ausdrücklich darauf hingewiesen, daß auch der System-Controller nach Abgabe der Controller-Funktion so lange "Nicht-Controller" bleibt, bis ihm diese Funktion wieder zurückgegeben wird oder er sich selbst durch ein RESET des Systems sein Vorrecht verschafft. Wenn man diesen Sonderfall ausschließt, gelten die folgenden Überlegungen für alle "Nicht -Controller", auch wenn eines der Geräte als System-Controller definiert ist.

Annehmen der Controller-Funktion

Obwohl eine einfache Abfrage des Interface-Status darüber Auskunft geben kann, ob der Rechner aktiver Controller (geworden) ist, gelingt diese Überwachung doch besser mit einer Zeilen-End-Verzweigung, die automatisch bei Erteilung der Controller-Funktion ausgelöst wird. Im folgenden Programmausschnitt wird das Gerät 20 aktiver Controller, vorher wird aber schon die Möglichkeit für eine Zeilen-End-Verzweigung zur Zeile 700 geschaffen, um zu prüfen, ob die Controller-Funktion inzwischen schon wieder zurückgegeben wurde.

```
650 ! Subroutine zur Abgabe der Controller-Funktion
660 ON INTR 7 GOSUB 710
670 ! Bereitet Verzweigung bei Rückgabe vor!
680 ENABLE INTR 7 ; 32
690 PASS CONTROL 720
700 RETURN
710 ! Ab hier Routine für Annahme der Controller-Funktion
720 STATUS 7,1 ; S
730 IF BIT(S,5) THEN GOTO 760 ! Bit 5: Akt. Controller
740 ! Verzweigung, aber kein aktiver Controller!
750 RETURN
760 ! Ab hier wieder akt. Controller!
770 ! Es muß etwas geschehen!
 .
 .
```

Dieses Verfahren kann leicht so erweitert werden, daß es mit der gleichen Routine auch möglich wird, auf eine Adressierung des Rechners zum Talker oder Listener mit einer passenden Programmverzweigung zu reagieren.

Im folgenden Beispiel wird die einfachste Technik zur übergabe der Controller-Funktion zwischen zwei Rechnern der Serie 80 demonstriert, von denen Rechner 1 bei Start seines Programms aktiver Controller sein muß. Keiner der Rechner muß im Prinzip dabei den Rang des System-Controllers haben!

Programm des Rechners 1:
(zu Beginn und am Schluß aktiver Controller!)

```
100 ! Adressierung als Controller:
110 OUTPUT 720 ; "Da ist sie!"
120 PASS CONTROL 720
130 ! Adressierung nun als Nicht-Controller:
140 ENTER 7 ; A$
150 DISP A$
160 ! Adressierung wieder als Controller:
170 OUTPUT 720 ; "Gern geschehen!"
180 END
```

Programm des Rechners 2:
(nur vorübergehend aktiver Controller!)

```
200 ! Adressierung als Nicht-Controller:
210 ENTER 7 ; A$
220 DISP A$
230 ! Adressierung als Controller:
240 OUTPUT 721 ; "Mit Dank zurück!"
250 PASS CONTROL 721
260 ! Wieder als Nicht-Controller:
270 ENTER 7 ; B$
280 DISP B$
290 END
```

Die beiden Programme sind einzugeben, wobei Rechner 1 die Adresse 21 und Rechner 2 die Adresse 20 haben muß (sonst Programm ändern!).

Erst ist Rechner 2 zu starten, er wartet dann in Zeile 210 auf Daten. Wenn jetzt auch Rechner 1 gestartet wird, gelangt mit Zeile 110 der erste Text (Da ist sie!) an Rechner 2 und wird dort angezeigt. Unmittelbar danach gibt Rechner 1 mit Zeile 120 die Controller-Funktion an den Rechner 2 und wartet selbst in Zeile 140 auf Daten vom Rechner 2. Dieser, nunmehr Controller, kündigt in Zeile 240 durch Text (Mit Dank zurück!) die Rückgabe der Controller-Funktion in Zeile 250 an. Die anstehende ENTER-Anweisung in Zeile 140 ist damit erfüllt, Rechner 1 zeigt den Text an und ist unmittelbar danach nun auch wieder aktiver Controller. Er verabschiedet sich mit neuem Text (Gern geschehen!) in Zeile 170, der in Zeile 270 vom Rechner 2 bereits erwartet wird. Mit der Anzeige dieses Textes sind beide Programme abgelaufen und halten bei den Zeilen 180 bzw. 290 an.

Diese einfache Technik ist zwar sehr übersichtlich, aber nur für kleinere Systeme zu empfehlen, weil bereits geringfügige Verzögerungen im Programmablauf (im Beispiel zwischen den Zeilen 110/120 bzw. 240/250) zu Fehlermeldungen führen müssen, da der jeweilige Rechner dann Anweisungen in Angriff nimmt, für die die Controller-Funktion (die er dann noch nicht erhalten hat) unabdingbar ist.

Korrekte Programme für "Gelegenheits-Controller"

Schon auf Seite IB-27 wurde gezeigt, WIE ein Rechner der Serie 80 Daten senden und empfangen kann, auch wenn er nicht aktiver Controller ist - dazu brauchte nur der Interface-Auswahl-Code angegeben zu werden; Geräte-Adressen waren dabei nicht zulässig! Der Rechner konnte auch, wie auf Seite IB-33 gezeigt wurde, die übergabe der Controller-Funktion erkennen und entsprechend verfahren. Hier werden nun diese beiden Beispiele so zusammengefaßt, daß mit Interrupt, Zeilen-End-Verzweigung und Analyse nicht nur die übergabe der Controller-Funktion, sondern auch die Adressierungen zum Talker bzw. Listener über Programmverzweigungen zu passenden Routinen führen. Nach Abarbeiten der jeweiligen Routine wird die Interrupt-Maske erneuert und, falls möglich, die Controller-Funktion wieder zurückgegeben.

```
      .
      .
      .
650 ! Subroutine zur Abgabe der Controller-Funktion
660 ON INTR 7 GOSUB 700
670 PASS CONTROL 720
680 ENABLE INTR 7 ; (16+32+64) @ RETURN
690 !
700 ! Routine fuer 'Nicht-Controller'
710 STATUS 7,1 ; S
720 IF BIT(S,5) THEN GOTO 810
730 IF BIT(S,4) THEN GOTO 780
740 IF NOT BIT(S,6) THEN PRINT "Fehler" @ STOP
750 ! Bit 6: als Listener aktiv:
760 ENTER 7 ; A$
770 GOTO 860
780 ! Bit 4: als Talker aktiv:
790 OUTPUT 7 ; X$
800 GOTO 860
810 ! Bit 5: als Controller aktiv:
820 ! Ab hier Controller-Tätigkeit, danach Rückgabe!
      .
      .
      .
850 PASS CONTROL 720
860 ENABLE INTR 7 ; (16+32+64) @ RETURN
```

Zusätzliche Informationen über Interrupts finden sich im IB-Abschnitt 5 (HP-IB für den Experten), wo auch die Steuer- und Status-Register beschrieben sind.

Ausgeben von Störungsmeldungen

Im HP-IB-System kann jedes einzelne Gerät (Ausnahme: aktiver Controller!) die SRQ-Leitung auf 'wahr' setzen, um damit den aktiven Controller auf einen Zustand hinzuweisen, der 'Beachtung' verdient. Da der Controller außer dieser Tatsache als solcher auch erfahren muß, was ihm mitgeteilt werden soll, hält jedes Gerät, das eine SRQ-Meldung abgegeben hat, ein speziell aufgebautes Byte bereit, das dem Controller beim Abfragen des Geräte-Zustandes übermittelt wird.

Bei den einfacheren Geräten des Systems (Drucker, Plotter, Massenspeicher, Digitalisierer und ähnliche) erfolgt die Ausgabe der SRQ-Meldung mit Formulierung des Antwort-Bytes über fest eingebaute Programme. Anders ist es, wenn ein Rechner eine SRQ-Meldung abgeben will: Weil es sich hier um ein sehr universell benutzbares Gerät handelt, ist es unmöglich, vorfabrizierte Antwort-Bytes für alle nur denkbaren Fälle als Firmware einzubauen. Die Rechner der Serie 80 stellen aber zur bequemen Handhabung der SRQ-Meldungen die REQUEST-Anweisung bereit. Sie setzt nur ein Mindestmaß an Kenntnissen über den Aufbau des Antwort-Bytes voraus:

Von den 8 Bits (gemäß ihrer Wertigkeit mit 7 bis 0 bezeichnet) hat nur das Bit 6 eine Sonderstellung. Es bestätigt, wenn gesetzt, dem analysierenden Controller, daß tatsächlich eine SRQ-Meldung vorliegt. Die übrigen Bits (0 bis 5 und 7) kann der Programmierer einzeln und auch kombiniert zur Kennzeichnung von (maximal 127) verschiedenen Zuständen benutzen.

Für den Fall, daß ein Rechner (ohne Controller-Status) langwierige Berechnungen ausführt, kann z.B. vereinbart werden, daß beim Vorliegen eines Ergebnisses der Rechner ein Antwort-Byte vorbereitet, bei dem Bit 0 (Rechnung fertig) und Bit 6 (SRQ liegt vor) gesetzt sind. Dieses und auch das 'Wahr'-Setzen der SRQ-Leitung besorgt die folgende Anweisung:

```
900 REQUEST 7 ; 1+64
```

Hinter dem Semikolon folgt ein numerischer Ausdruck, durch den der Zustand der einzelnen Bits des Antwort-Bytes festgelegt wird. Zwecks besserer Übersicht kann man, wie im Beispiel, die Formulierung mit einzelnen Summanden vornehmen, um die (kurze aber unanschauliche) Angabe als Summe zu vermeiden. Die REQUEST-Anweisung muß auch einen Auswahl-Code enthalten: Hiermit wird das Interface angesprochen, das zum Rechner gehört, der die SRQ-Meldung ausgibt! Dieser Auswahl-Code braucht nicht mit dem übereinzustimmen, den das HP-IB-Interface im aktiven Controller besetzt!

Der Umgang mit Störungen im IB-Interface und im Bus

Dieser Abschnitt beschreibt einige Verfahren, die Sie zur Vermeidung oder Untersuchung von Störungen kennen sollten, wenn Sie das HP-IB-Interface benutzen. Das bedeutet nicht, daß Sie unbedingt auf solche Probleme stoßen werden, aber es ist ein guter Grundsatz, im Programm den "Problemen" nach Möglichkeit zuvor zu kommen und sich schon vorher Gedanken über sie zu machen.

Wie vermeidet man das "Hängenbleiben" bei Bus-Operationen?

Allgemein läßt sich sagen, daß eine Störung in einem HP-IB-Gerät entweder die Datenübertragung anhält und/oder eine SRQ-Abfrage durch den aktiven Controller auslöst. Wir wissen bereits, wie der Controller auf den Eingang von Störungsmeldungen reagiert (Seite IB-31 und folgende), was passiert aber, wenn ein Gerät während einer Datenübertragung das Handshake abbricht und dabei eine SRQ-Abfrage auslöst? Dieses Zusammentreffen schafft ein Problem, denn der Rechner kommt nicht zum Zeilen-Ende, kann also die SRQ-Routine nicht beginnen. Der Grund liegt in Definition und Namen der Zeilen-End-Verzweigung: Diese ist nie während der Ausführung, sondern erst nach Beendigung einer Programmzeile möglich. Und das gilt auch für Zeilen mit OUTPUT- oder ENTER-Anweisungen! Das Gerät hat jetzt die Datenübertragung eingestellt, der Rechner wartet auf die Fortsetzung, er kann das Ende der Zeile nicht erreichen und deshalb auch nicht die Zeilen-End-Verzweigung beginnen, die vielleicht eine Lösung bietet! Der Rechner hängt also fest. Eine Rettung aus dieser "Fallgrube" ist möglich, wenn Sie die dafür vorgesehene SET TIMEOUT-Anweisung in das Programm einbauen.

Das folgende Beispiel zeigt die richtige Reihenfolge der notwendigen Operationen, um ein Programm vor dem 'Hängenbleiben' zu schützen.

```
 10 DIM S(6)
 20 ! Zuerst wird die TIME OUT-Routine vorbereitet
 30 ON TIMEOUT 7 GOSUB 170
 40 ! Jetzt wird Handshake auf 1200 ms begrenzt
 50 SET TIMEOUT 7 ; 1200
 60 ! Jetzt wird der HP 3455 auf Triggerung programmiert
 70 OUTPUT 722 ; "F1R1T3T3"
 80 ! Hier wird das DVM getriggert und abgelesen
 90 TRIGGER 722
100 ENTER 722 ; X
110 DISP X;"Volt"
120 GOTO 90
130 !
140 ! Ab hier TIME OUT-Routine, zuerst
150 ! wird der Interface-Status geprüft
160 ON TIMEOUT 7 GOTO 270 ! Falls Interface defekt
170 FOR I=0 TO 6
180 STATUS 7,I ; S(I)
190 PRINT "Status-Register ";I;" =";S(I)
200 NEXT I
210 ! Hier gehören die vom Ergebnis der Status-
220 ! Abfrage abhaengigen Service-Routinen hin
    .
250 GOTO 290 ! Ende der TIME OUT-Routine
260 !
270 PRINT "Interface-Fehler"
280 RESET 7 ! Versuch zu helfen
290 ON TIMEOUT 7 GOSUB 170 ! Erneuert alte Service-Routine
300 RETURN
```

Der Programmierer muß also festlegen, wie das Programm auf das Ergebnis der Statusabfrage reagieren soll. Meistens ist für das Interface ein RESET nötig, um den Datenaustausch mit dem gestörten Gerät abzubrechen, danach sollte der Benutzer über die Störung informiert werden. Diese Dinge werden im nächsten Abschnitt behandelt.

Der Umgang mit "Problemen"

Wahrscheinlich lesen Sie diesen Abschnitt, wenn Ihr HP-IB-System nicht korrekt arbeitet oder, weil Sie sich informieren wollen, wie man "Problemen" zuvorkommt. Sie können hier zuerst sehen, was zu tun ist, wenn "überhaupt nichts mehr" läuft, weil dann schnelle Hilfe nötig ist. Anschließend werden Tips zur Eingrenzung der Störungen gegeben.

Wenn ein HP-IB-System festgefahren scheint (es läuft irgendetwas, aber nichts geschieht!), dann sollte Ihnen die RESET-Taste wieder Einfluß auf den Rechner verschaffen. Nur, wenn der Rechner in einem FHS-Transfer hängenbleibt, hilft Ihnen das nichts! Die beste Möglichkeit für diesen Fall ist die Ausführung eines INTERFACE-CLEAR (IFC) auf irgendeine Weise (auch ein Bus-Analysator kann das!). Als zweit-"bestes" bleibt nur Aus- und wieder Einschalten des Rechners!

Danach und auch bei den sonstigen Störungen sollten Sie dann die einzelnen Geräte seriell auf ihren Status prüfen. Das setzt aber voraus, daß Sie wissen, wie man diese Information korrekt bekommt. Manchmal hilft auch hier einfaches Aus- und Einschalten, um ein Peripherie-Gerät wieder zur "Vernunft" zu bringen.

Der folgende Abschnitt zeigt die verschiedenen Möglichkeiten (und die Auswahlkriterien dafür), die dem Programmierer beim Entwerfen der Fehler-Routinen für die Behebung von Störungen im HP-IB-System zur Verfügung stehen.

Handlung:	Absicht bzw. Ergebnis:
FOR I=1 TO 6 STATUS 7,I ; S(I) NEXT I	Ermittelt den momentanen Status vom Interface um Zustand und Ursachen erklären zu können.
S=SPOLL(<jedes Gerät>)	Ermittelt momentanen Status der Geräte im System.(z.B. Drucker ohne Papier)
CLEAR <Einzelgerät>	Setzt das bezeichnete Gerät auf seinen teilweise geräteabhängigen "Clear"-Status zurück.(RESET-ähnlich!)
CLEAR 7	Setzt alle Geräte auf ihren geräteabhängigen "Clear"-Status zurück.
ABORTIO 7	Anwendbar nur, wenn der Rechner der Serie 80 System-Controller ist! Beendet alle Bus-Aktivitäten, der System-Controller wird wieder aktiver Controller.

Es ist zweckmäßig, die ermittelten Zustände von Interface und Geräten ausdrucken zu lassen. Ebenso sollte auch jede Aktion zur Behebung der Störung dokumentiert werden. Diese Informationen sind die Grundlage für die Ermittlung der Störungsursache und für den Programmierer nützlich, um entsprechende Hinweise für die Fehlerbeseitigung zu geben (z.B. den Drucker mit Papier zu versorgen!).

HP-IB für den Experten

Dieser Abschnitt ist für Benutzer bestimmt, die mit der Norm IEEE-488-1978 vertraut sind und über das Interface B2937A vollständige Informationen in kompakter Form benötigen. Hier werden aber nur die grundsätzlichen Möglichkeiten gezeigt, die das Interface dem Experten bietet, detaillierte Programmbeispiele sind hier nicht zu erwarten!

Die HP-IB-I/O-Anweisungen

ABORTIO: Verursacht Interface-Clear (IFC), macht Fernsteuerung (REN) möglich, sofern das Interface System-Controller ist. Wenn das Interface nur aktiver Controller ist, gibt es seine Talk-Adresse bekannt, wodurch jedes andere Gerät im System den Talker-Status verliert. Wenn das Interface nicht Controller ist, verbleibt der Bus im vorhandenen Zustand und ist zur nächsten Operation bereit (siehe auch HALT).

ASSERT: ASSERT beeinflußt sofort das Steuer-Register 2 und damit die HP-IB-Steuerleitungen, auch bei 'beschäftigtem' Interface. Die Beeinflussung von Steuer-Register 2 mittels CONTROL hat den gleichen Effekt, CONTROL wartet aber ab, bis das Interface seine laufende Operation beendet hat. Der Benutzer darf nur solche Werte in das Register 2 bringen, die der Interface-Logik genügen, sonst kann das Interface blockiert werden, was eine RESET-Operation erfordert.

CLEAR: Nur dem aktiven Controller gestattet. Die adressierten, bzw. alle Geräte (Auswahl-Code allein!) des Systems erhalten DCL-Befehl (Device CLear). Die ATN-Leitung bleibt auf 'wahr'. Sie läßt sich mit RESUME auf 'falsch' bringen.

CONTROL: Beeinflußt die Steuer-Register vom Interface. Fehler 111 wird ausgelöst, wenn die Beeinflussung eines nicht vorhandenen Registers versucht wird. Zugänglich sind die Register 0...3 und 16...23 (sowie Register 0 und 1 der I/O-Buffer).

ENABLE INTR: Beeinflußt Steuer-Register 1 (auch mit CONTROL möglich), erzeugt die Maske für die Zeilen-End-Verzweigungen.

ENTER: Mit Adresse nur dem aktiven Controller gestattet. Mit Auswahl-Code beginnt die Aufnahme erst nach der Adressierung des Interfaces zum Listener. Die Wirkung von ENTER-Anweisungen mit und ohne Adressierung zeigen die Beispiele:

`ENTER 305 ; A$` Nur für den aktiven Controller zugelassen, führt erst Adressierung aus, nimmt dann Daten für die Variable A$ auf.

`ENTER 3 ; A$` Falls Interface nicht aktiver Controller, wird abgewartet, bis Listener-Adressierung erfolgt. Falls das Interface selbst aktiver Controller ist, muß sich dieses zuvor selber als Listener deklariert haben, sonst erfolgt Fehlermeldung 116.

HALT: Veranlaßt das Interface, die laufende Tätigkeit abzubrechen und für die nächste I/O-Operation bereit zu sein. HALT beeinflußt nicht die Bus-Signale. Beachten Sie, daß ABORTIO ebenso wirkt, wenn das Interface weder System-Controller noch aktiver Controller ist.

LOCAL: Mit Adresse(n) nur dem aktiven Controller gestattet. Mit Auswahl-Code und REN-Leitung auf 'wahr' nur dem System-Controller gestattet. Den adressierten, bzw. allen Geräten des Systems wird der GTL-Befehl (Go To Local) gegeben. Die ATN-Leitung bleibt auf 'wahr'. Sie läßt sich mit RESUME auf 'falsch' bringen.

LOCAL LOCKOUT: Nur dem aktiven Controller gestattet. Veranlaßt das Interface zur Ausgabe des LLO-Befehls (Local Lock Out). Die ATN-Leitung bleibt auf 'wahr'. Sie läßt sich mit RESUME auf 'falsch' bringen.

OUTPUT: Mit Adresse(n) nur dem aktiven Controller gestattet. Mit Auswahl-Code beginnt die Ausgabe erst, wenn das Interface aktiver Talker geworden ist. Die Wirkung von OUTPUT-Anweisungen mit und ohne Adressierung zeigen die Beispiele:

OUTPUT 305,306 ; A$ Nur dem aktiven Controller gestattet, führt zuerst Adressierung aus, gibt dann Inhalt der Variablen A$ aus.

OUTPUT 3 ; A$ Falls Interface nicht aktiver Controller, wird die Adressierung zum Talker abgewartet. Falls Interface aktiver Controller, muß sich dieses zuvor selbst als Talker deklarieren, sonst Fehlermeldung 115.

PASS CONTROL: Nur dem aktiven Controller gestattet. Nach Adressierung wird die Controller-Funktion übergeben. Der aktive Controller kann seine Funktion an jedes dafür geeignete Gerät (auch an sich selbst) übergeben.

PPOLL: Nur dem aktiven Controller gestattet. Holt das Byte von der Parallel-Abfrage (PPOLL) aus dem Interface.

REMOTE: Nur dem System-Controller gestattet. Setzt die REN-Leitung (Remote Enable) auf 'wahr' und schaltet ohne Adressen alle, bzw. nur die adressierten Geräte des Systems auf Fernbedienung um. Mit Adressen bleibt die ATN-Leitung auf 'wahr'. Sie läßt sich mit RESUME auf 'falsch' bringen.

REQUEST: Dem aktiven Controller nicht gestattet. Bringt durch Setzen von Bit 6 (der Bits 0...7) gültiges SRQ ins Interface. Übrige Bits beliebig. Dieses Byte ist für aktiven Controller als Antwort auf serielle Abfrage bestimmt, wobei SRQ gelöscht wird. SRQ kann auch mit neuer REQUEST-Anweisung zurückgenommen werden, falls diese Bit 6 wieder löscht.

RESET: Setzt das Interface bedingungslos auf den Einschaltzustand zurück und löst damit Selbst-Test für das Interface aus, der bei gestörtem Ablauf die Fehlermeldung 110 auslöst. Bei erfolgreichem Test erfolgt keine Meldung.

RESUME: Nur dem aktiven Controller gestattet. Setzt ATN-Leitung auf 'falsch'. Ist nach einem CLEAR zweckmäßig, um die 'wahr' gebliebene ATN-Leitung wieder auf 'falsch' zu bringen. Nach I/O-Operationen wie ENTER, OUTPUT und TRANSFER ist RESUME unnötig, da diese Anweisungen von sich aus ATN auf 'falsch' halten.

SEND: Nur vom aktiven Controller meist zu speziellen Ausgaben benutzt. Die ATN-Leitung wird bei Ausgabe von Daten 'falsch', bei Ausgabe von Befehlen 'wahr' und behält den zuletzt von SEND verursachten Zustand. Sie läßt sich mit RESUME bei Bedarf auf 'falsch' bringen.

SPOLL: Nur dem aktiven Controller gestattet. Ermittelt mit serieller Abfrage den Wert des Status-Bytes. Dieses zeigt an, ob das Gerät ein SRQ ausgelöst hat, und kann auch zusätzliche Informationen geben.

STATUS: Liest die Status-Register vom Buffer bzw. Interface. STATUS wird sofort ausgeführt, ohne den Interface-Zustand zu berücksichtigen, es sei denn, die IFC-Leitung ist 'wahr'. Die Status-Register haben die Nummern 0...3 für Buffer bzw. 0...6 für Interfaces. Der Versuch, andere Register mit der STATUS-Anweisung zu lesen, führt zur Fehlermeldung 111.

TRANSFER: Für Interrupt- oder Fast-Handshake-Transfer I/O-Operationen benutzt. TRANSFER setzt definierten I/O-Buffer voraus (siehe I/O-ROM-Abschnitt 8, wo auch Informationen über die Buffer-Zeiger zu finden sind). Zusammen mit TRANSFER können drei verschiedene Schluß-Merkmale benutzt werden:

1. DELIM mit dem Dezimalwert eines verabredeten Zeichens.
2. COUNT mit einem Byte-Zähler.
3. Beendigung der übertragung mit der EOI-Leitung.

Nachstehend eine Tabelle über die Anwendungsmöglichkeiten:

TRANSFER-Methode	**DELIM**	**COUNT**	**EOI**
TRANSFER(ein)FHS	Nicht zulässig	Anwendung bei Bedarf, sonst Transferablauf durch Füllzeiger und Bufferende gesteuert.	EOI-Abschluß wirksam
TRANSFER(aus)FHS	Nicht zulässig	Nicht zulässig, Transferablauf durch Lese- und Füllzeiger bestimmt	Nicht zulässig: EOL-Sequenz wird ausgegeben
TRANSFER(ein)INTR	Zulässig	Anwendung bei Bedarf, sonst Eingabe durch DELIM oder EOI oder Bufferlänge beendet	EOI-Abschluß wirksam
TRANSFER(aus)INTR	Nicht zulässig	Nicht zulässig, Transferablauf durch Lese- und Füllzeiger bestimmt	Nicht zulässig: EOL-Sequenz wird ausgegeben

TRIGGER: Nur dem aktiven Controller gestattet. An adressierte, bzw. alle Geräte (Auswahl-Code allein!) des Systems ergeht der GET-Befehl (Group Execute Trigger). Die ATN-Leitung bleibt auf 'wahr'. Sie läßt sich mit RESUME auf 'falsch' bringen.

Typische HP-IB-Output-Sequenzen

Die folgende Tabelle zeigt die an den Bus gegebene Befehlsfolge, die durch diese Anweisung ausgelöst wird (Steuerbefehle im Fettdruck, Daten im Normaldruck):

```
10 OUTPUT 705;"HEWLETT-PACKARD INTERFACE BUS"
```

Befehl	Binär	Dezimal	Kennkürzel
U	**01010101**	**85**	**ATN,MTA**
?	**00111111**	**63**	**UNL**
%	**00100101**	**37**	**LAG,ATN**
H	01001000	72	(Daten)
E	01000101	69	(Daten)
W	01010111	87	(Daten)
*	*	*	*
*	*	*	*
*	*	*	*
*	*	*	*
*	*	*	*
*	*	*	*
B	01000010	66	(Daten)
U	01010101	85	(Daten)
S	01010011	83	(Daten)
	00001101	13	(CR)
	00001010	10	(LF)

Typische HP-IB-Enter-Sequenzen

Die folgende Tabelle zeigt die an den Bus gegebene Befehlsfolge, die durch diese Anweisung ausgelöst wird (Steuerbefehle im Fettdruck, Daten im Normaldruck):

```
10 ENTER 705;A$
```

Befehl	Binär	Dezimal	Kennkürzel
?	**00111111**	**63**	**ATN,UNL**
5	**00110101**	**53**	**MLA**
E	**01000101**	**69**	**TAG,ATN**
H	01001000	72	(Daten)
E	01000101	69	(Daten)
W	01010111	87	(Daten)
*	*	*	*
*	*	*	*
*	*	*	*
*	*	*	*
*	*	*	*
*	*	*	*
B	01000010	66	(Daten)
U	01010101	85	(Daten)
S	01010011	83	(Daten)
	00001101	13	(CR)
	00001010	10	(LF)

Vereinbarungen über Kennkürzel

<ATN>	Achtung, ATN-Leitung wahr! (es folgen Befehle)
<$\overline{\text{ATN}}$>	Achtung, ATN-Leitung falsch! (es folgen Daten)
<CA>	Aktiver-Controller-Status
<CR>	Wagen-Rücklauf-Befehl
<Daten>	ein oder mehrere Byte(s)
<DCL>	Alle Geräte des Systems auf Einschaltzustand zurücksetzen
<GET>	an adressierten Geräten Triggerung auslösen
<GTL>	ermöglicht Geräte-Einstellung durch lokale Stellorgane
<LA>	Aktiver Listener-Status
<LAG>	Adressierung einer Gruppe von Listenern
<LF>	Zeilenvorschub
<LLO>	sperrt Gerätebeeinflussung durch lokale Stellorgane
<MLA>	My Listen Address, (Listener-Adresse des Rechners)
<MTA>	My Talk Address, (Talker-Adresse des Rechners)
<PPC>	Parallel-Abfrage eingeschaltet
<PPU>	Parallel-Abfrage ausgeschaltet
<SC>	System-Controller
<SCG>	Gruppe von Sekundär-Befehlen
<SDC>	bestimmte Geräte des Systems auf Einschaltzustand zurücksetzen
<SPD>	Serielle Abfrage gesperrt
<SPE>	Serielle Abfrage zugelassen
<TA>	Aktiver Talker-Status
<TAD>	erklärt Gerät zum Talker
<TCT>	übergibt Controller-Funktion
<UNT>	hebt Talker-Status auf
<ca. 6 µs>	Zeitverzögerung von etwas mehr als 6 Mikrosekunden.

Aufbau der Meldungen

Die durch das Interface-Bus-System verbundenen Geräte tauschen untereinander in geordneter Weise Informationen aus. Dieser Austausch geht stets von einem einzigen Gerät aus und ist an eines oder mehrere der vorhandenen Geräte gerichtet. Die einzelnen Teile der Information werden als 'Meldungen' bezeichnet.

Es lassen sich zwölf Arten von Meldungen unterscheiden. Alle nachstehend angegebenen Arten können von den Rechnern der Serie 80 bearbeitet werden.

1. Daten-Meldung. Das ist die Information selbst, die von einem Talker ausgegeben wird und für einen oder mehrere Listener bestimmt ist.
2. Trigger-Meldung. Veranlaßt den oder die Listener oder auch alle Geräte gleichzeitig zu geräte-spezifischen Tätigkeiten.
3. Clear-Meldung. Veranlaßt den oder die Listener oder auch alle Geräte, sich auf bestimmte, vorher vereinbarte Zustände einzustellen.
4. Remote-Meldung. Macht den oder die Listener oder auch alle Geräte für Ferneinstellung sensibel.
5. Local-Meldung. Macht den oder die Listener oder auch alle Geräte für Ferneinstellung unsensibel und gibt die manuelle Einstellung an den Geräten wieder frei.

6. Local-Lockout-Meldung. Sperrt die Beeinflussung der Geräte durch die von außen zugänglichen Einstellorgane an den Geräten.

7. Clear-Lockout/Local-Meldung. Gibt die Sperrung der Einstellorgane an den Geräten selbst wieder frei, schaltet die Ferneinstellung ab. Löscht auch die Remote-Meldung für alle Geräte des Systems.

8. Request-Service-Meldung. Jedes Gerät kann diese Meldung jederzeit ausgeben, um damit den Wunsch nach Abfrage durch den Controller anzuzeigen. Die Meldung wird bei Ausgabe des Status-Bytes wieder gelöscht, wenn das Gerät keine weitere Abfrage mehr wünscht.

9. Status-Byte-Meldung. Dieses Byte beschreibt den aktuellen Zustand eines Gerätes im Bus-System. Dabei zeigt Bit 6 den Wunsch nach Abfrage an, die restlichen Bits beschreiben den Arbeitszustand des Gerätes entsprechend einer vorherigen Definition. Dieses Byte wird als Antwort auf eine vom Controller durchgeführte serielle Abfrage ausgegeben, sobald das Gerät zum Talker deklariert wurde.

10. Status-Bit-Meldung. Dieses Byte beschreibt die Zustände mehrerer Geräte im System gleichzeitig (parallel!). Dabei kann jedes der Geräte das ihm zugeordnete Bit beeinflussen, um damit eine geräte-abhängige Information zu übermitteln. Die Geräte geben den Zustand des ihnen zugeteilten Bits für gewöhnlich als Antwort auf eine Parallel-Abfrage (Parallel Poll) aus. Die Status-Bit-Meldung kann vom Controller auch abgerufen werden, um zu erfahren, mit welchem Bit und mit welchem Zustand ein bestimmtes Gerät auf die Parallel-Abfrage antwortet. Es können deshalb mehrere Geräte mit dem gleichen Bit auf die Abfrage antworten.

11. Pass-Control-Meldung. überträgt die Steuerung des Bus-Systems vom momentan aktiven Controller auf seinen Nachfolger.

12. Abort-Meldung. Der System-Controller gibt diese Meldung aus und wird damit unter allen Umständen wieder aktiver Controller im Bus-System. Es werden dadurch alle Aktivitäten auf dem Bus abgebrochen (Die Wirkung ist aber nicht mit der Clear-Meldung gleichzusetzen).

Mit diesen Meldungen lassen sich alle durch das HP-IB-System gegebenen Möglichkeiten nutzen. Es ist aber üblich, daß manche Geräte nur die Meldungen annehmen, die sie selbst benötigen bzw. verarbeiten können. Für einen reibungslosen Ablauf der Operationen innerhalb des Systems ist unbedingt nötig, daß Ihnen bekannt ist, welche Meldungen von den einzelnen Geräten angenommen und verarbeitet werden.

Die HP-IB-Steuer-Leitungen

Die nebenstehende Darstellung zeigt neben den Daten-Leitungen (DIO...DIO 8) noch acht weitere Leitungen, mit denen das Bus-System in der für das HP-IB-System typischen Weise gesteuert wird.

Drei dieser Leitungen werden als 'Handshake'-Leitungen bezeichnet und dienen der zeitlichen Abstimmung bei der Daten-Übertragung, damit der Talker für keinen der Listener zu schnell 'spricht'. Diese Leitungen haben folgende Bezeichnungen bzw. Aussagewirkungen:

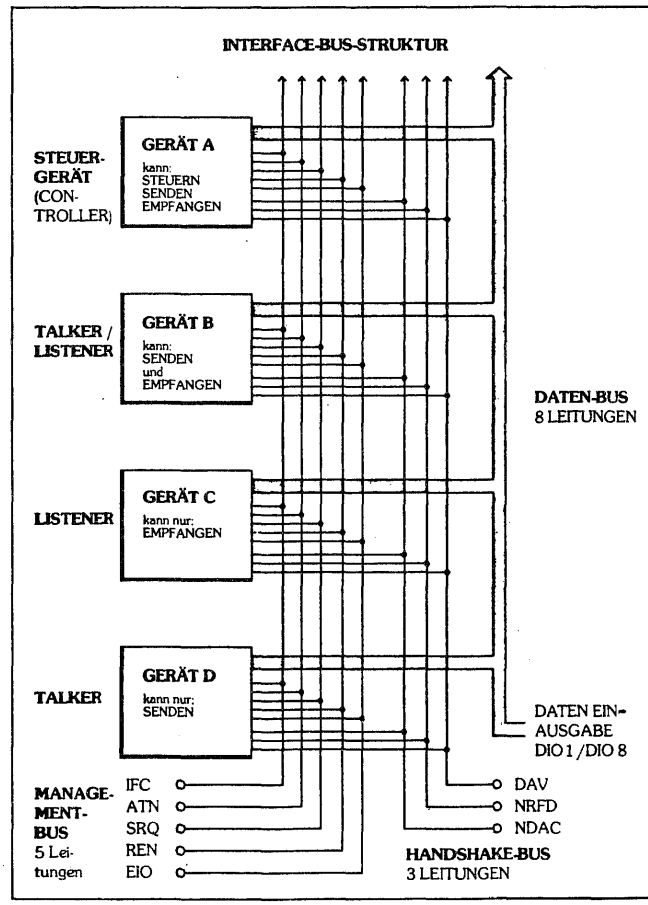
DAV Daten sind gültig! (DAta Valid)
NRFD Nicht bereit zur Daten-Aufnahme! (Not Ready For Data)
NDAC Noch keine Daten angenommen! (Not Data ACcepted)

Die "negative" Festlegung der beiden zuletzt genannten Leitungen hat (bei Vorrang des 'wahren' Zustandes) den Sinn, daß auch ein einziges Gerät die Ausgabe neuer Daten so lange blockieren kann, bis es zur Aufnahme wieder bereit ist.

Die Übertragung eines Bytes geschieht dann so: Der Talker muß warten, bis alle Listener erklärt haben, daß sie noch keine Daten angenommen haben (NDAC 'wahr') und auch zum Daten-Empfang bereit sind (NRFD 'falsch'). Der Talker kann dann das den Datenleitungen zugeordnete Byte als gültig erklären (DAV 'wahr').

Die Listener nehmen das Byte auf, setzen sofort NRFD auf 'wahr' und geben, entsprechend ihren individuellen Geschwindigkeiten, nacheinander die NDAC-Leitung frei, die schließlich 'falsch' wird. Damit ist sicher, daß alle Listener die Meldung aufgenommen haben, der Talker setzt dann DAV auf 'falsch', wodurch die Daten ungültig werden.

Darauf setzen alle Listener die NDAC-Leitung auf 'wahr' und geben erst nach der Verarbeitung des eben empfangenen Bytes, jeder für sich die NRFD-Leitung wieder frei, die aber erst wieder 'falsch' wird, wenn alle Listener wieder zur Daten-Aufnahme bereit sind. Der Talker konnte inzwischen das nächste Byte den Datenleitungen zuordnen und das Spiel beginnt von vorn.



Die ATN-(ATtentioN)-Leitung

über die Daten-Leitungen DIO 1...8 werden auch die Befehls-Meldungen übertragen, die sämtlich durch Bytes mit einer führenden Null verkörpert werden. Sie gleichen damit den ersten 128 Daten-Bytes des gesamten Zeichenvorrats. Zur Unterscheidung dient die ATN-Leitung: Bei Daten ist ATN 'falsch', bei Befehlen 'wahr'. Die IB-Befehle sind auf den Seiten IB-48/49 aufgeführt. Es werden nicht alle 128 ASCII-Zeichen als Befehle benutzt.

Die IFC-(InterFace Clear)-Leitung

Die IFC-Leitung kann nur durch den System-Controller auf 'wahr' gesetzt werden. Damit wird der System-Controller bedingungslos aktiver Controller, falls er diese Funktion nicht bereits hatte, und unterbindet jede Aktivität im Bus-System. Der Talker und alle Listener verlieren ihren Status. Im Normalfall wird die IFC-Leitung nur benutzt, um unerwartete Operationen abzubrechen oder, um dem System-Controller wieder die Steuerung des Systems zu übertragen, weil etwas Ungewöhnliches auf dem Bus passierte. IFC hat absoluten Vorrang gegenüber allen anderen Dingen, die im System im Moment geschehen.

Die REN-(Remote ENable)-Leitung

Mit dieser Leitung ist es möglich, zum System gehörende Geräte auf Fernbedienung zu schalten, um sie von anderen Geräten des Systems steuern zu lassen. Meistens (aber nicht ausschließlich) besorgt das der aktive Controller. Jedes Gerät, das bei 'wahrer' REN-Leitung Listener ist (oder wird), geht in den REMOTE-Status und wird damit fernbedienbar.

Die EOI-(End Or Identify)-Leitung

Die Datenübertragung innerhalb des HP-IB-Systems benutzt den Standard-ASCII-Code und endet meist mit dem ASCII-Zeichen für den Zeilenvorschub (CHR$(10):LF). Einige Geräte (z.B. Massen-Speicher) empfangen oder senden aber Datenblöcke, in denen alle 256 möglichen Byte-Muster enthalten sein können. Dann läßt sich keines dieser Muster als Schluß-Zeichen deklarieren, weil es ja auch irgendwo innerhalb des Blockes auftreten kann. Das Ende der Daten muß dann mit der EOI-Leitung markiert werden, die bei Ausgabe des letzten Bytes 'wahr' wird. Der (oder die) Listener erkennen daran das Ende der Übertragung.

Die EOI-Leitung dient auch bei der Parallel-Abfrage (PPOLL) zur Identifizierung.

Die SRQ-(Service ReQuest)-Leitung

Der aktive Controller muß das HP-IB-System dauernd auf fehlerfreie Funktion überwachen können. Wenn eines der Geräte die Aufmerksamkeit des Controllers auf sich ziehen will, kann es die SRQ-Leitung auf 'wahr' setzen. So muß sich ein Drucker z.B. bemerkbar machen, wenn das Papier zu Ende geht, oder ein Digitalisierer muß melden, daß neue Werte vorliegen, die der Controller übernehmen soll. Dies alles sind Meldungen (keine Befehle!), auf die der Controller mit Service-Routinen reagieren kann, wann und wie es (dem Programm) paßt. Das Gerät wird aber diese Meldung aufrechterhalten, bis sie beachtet wurde. Wie auf die Meldungen zu reagieren ist, hängt von den Eigenschaften der Geräte ab. Hier müssen die Geräte-Handbücher befragt werden.

Die Reaktionen des HP-IB-Interfaces auf Steuerbefehle

Die folgende Aufstellung beschreibt die Reaktionen auf die verschiedenen Befehle:

ATN
Die nachfolgenden Meldungen werden als Befehle gewertet. Das Interface kann während des Eintreffens der Befehle ungestört mit dem Rechner weiter zusammenarbeiten.

IFC
Alle Listener und der Talker werden in den Zustand versetzt, den sie beim Einschalten annehmen. Die Controller-Funktion geht zum System-Controller zurück, falls sie nicht schon vorher vom System-Controller ausgeübt wurde.

REN
Das Bit 1 im Status-Register SR5 wird gesetzt.

EOI
Die Übermittlung von Daten in einen Buffer wird damit beendet. Auch ENTER- und TRANSFER-Anweisungen können damit zum Abschluß gebracht werden.

SRQ
Das Bit 3 im Status-Register SR1 wird gesetzt. Falls das Bit 3 im Steuer-Register CR1 gesetzt ist, wird ein Interrupt ausgelöst.

DCL, SDC
Ein Interrupt wird ausgelöst, wenn Bit 2 im Steuer-Register CR1 gesetzt ist.

GTL, LLO
GTL löscht die Bits 0 und 1 im Status-Register SR5. LLO setzt Bit 0 im Status-Register SR5.

GET
Ein Interrupt wird ausgelöst, wenn Bit 1 im Steuer-Register CR1 gesetzt ist.

Serielle Abfrage
Der momentane Inhalt des Bytes für die serielle Abfrage (REQUEST) wird ohne Zutun des Rechners ausgegeben.

Parallele Abfrage
Auf eine Parallel-Abfrage reagiert das Interface mit der Ausgabe des SRQ-Bits auf der durch Drahtbrücke oder Schalter ausgewählten Datenleitung.

PPU,PPC
Die Reaktion auf eine Parallel-Abfrage ist nur per Schalter zu setzen und nicht programmierbar. Deshalb keine Reaktion.

TCT
Die Controller-Funktion für das HP-IB-System wird übernommen.

HP-IB-Universal-Befehle

Die folgende Aufstellung enthält alle HP-IB-Universal-Befehle und deren Zuordnung zu den ASCII-Zeichen mit den jeweiligenen Dezimalwerten. Außerdem werden die numerischen Bereiche für Adressen und Befehle angegeben. Bei der Ausgabe von Befehlen ist die ATN-Leitung 'wahr'.

Dezimal-Wert	ASCII-Zeichen	Interface-Meldung	Beschreibung der Wirkung
0	NUL		
1	SOH	GTL	Lokale Stellorgane wirksam
2	STX		
3	ETX		
4	EOT	SDC	Einzelgeräte zurücksetzen
5	ENQ	PPC	Parallel-Abfrage vorbereiten
6	ACK		
7	BEL		
8	BS	GET	Gruppentriggerung auslösen
9	HT	TCT	Controller-Fktn. übergeben
10	LF		
11	VT		
12	FF		
13	CR		
14	SO		
15	SI		
16	DLE		
17	DC1	LLO	Lokale Stellorgane sperren
18	DC2		
19	DC3		
20	DC4	DCL	alle Geräte zurücksetzen
21	NAK	PPU	Parallel-Abfrage abschalten
22	SYN		
23	ETB		
24	CAN	SPE	Serielle Abfrage zulassen
25	EM	SPD	Serielle Abfrage sperren
26	SUB		
27	ESC		
28	FS		
29	GS		
30	RS		
31	US		
32..62	SP bis > (Zahlen und Spezialzeichen)	LAG	Listener-Adreß-Gruppe
63	?	UNL	Listener abschalten
64..94	@ bis ^ (große Buchstaben)	TAG	Talker-Adreß-Gruppe
95	_	UNT	Talker abschalten
96..126	(kleine Buchstaben)	SCG	Sekundär-Befehls-Gruppe
127	DEL		

Zulässige Bus-Adressen und ihre Codierungen

Adressierungszeichen als Listener:	Talker:	Stellung der Adress-Schalter (5)	(4)	(3)	(2)	(1)	Adressierungs-Code dezimal	oktal
SP	@	0	0	0	0	0	0	0
!	A	0	0	0	0	1	1	1
"	B	0	0	0	1	0	2	2
#	C	0	0	0	1	1	3	3
$	D	0	0	1	0	0	4	4
%	E	0	0	1	0	1	5	5
&	F	0	0	1	1	0	6	6
'	G	0	0	1	1	1	7	7
(	H	0	1	0	0	0	8	10
)	I	0	1	0	0	1	9	11
*	J	0	1	0	0	0	10	12
+	K	0	1	0	1	1	11	13
,	L	0	1	1	0	0	12	14
-	M	0	1	1	0	1	13	15
.	N	0	1	1	1	0	14	16
/	O	0	1	1	1	1	15	17
0	P	1	0	0	0	0	16	20
1	Q	1	0	0	0	1	17	21
2	R	1	0	0	1	0	18	22
3	S	1	0	0	1	1	19	23
4	T	1	0	1	0	0	20	24
Einstellung bei Lieferung!								
5	**U**	**1**	**0**	**1**	**0**	**1**	**21**	**25**
6	V	1	0	1	1	0	22	26
7	W	1	0	1	1	1	23	27
8	X	1	1	0	0	0	24	30
9	Y	1	1	0	1	1	25	31
:	Z	1	1	0	1	0	26	32
;	[	1	1	0	1	1	27	33
<	/	1	1	1	0	0	28	34
=	]	1	1	1	0	1	29	35
>	^	1	1	1	1	0	30	36

Zustands-Abfrage (Polling)

Die Abfrage der im System befindlichen Geräte gibt dem Controller Aufschluß über deren Betriebszustand. Die beiden hierfür verwendeten Verfahren sind die serielle (SPOLL) und die parallele (PPOLL) Abfrage. Beide sind in ihrem Ablauf recht kompliziert. Weil Sie aber per Programm entscheiden können, ob Sie seriell und/oder parallel abfragen wollen, werden beide Verfahren getrennt erläutert, was das Verständnis erleichtert.

Serielle Abfrage (Serial Poll)

Die serielle Abfrage liefert dem aktiven Controller von jedem Gerät, das auf diese Abfrage reagiert, ein Status-Byte. Die Abfrage wird erst nötig, wenn die SRQ-Leitung auf 'wahr' geht. Der Controller wird dann nacheinander die Geräte abfragen, von denen eine SRQ-Meldung zu erwarten ist. Das abgefragte Gerät wird mit seinem Status-Byte antworten, in dem das Bit 6 (der Bits 0...7) nur dann gesetzt ist, wenn das Gerät die SRQ-Meldung (mit-)verursacht hat. Der Grund für die SRQ-Meldung kann mit den übrigen Bits genauer erklärt werden. Ihre Zuordnung zu bestimmten Zuständen wird für jedes Gerät anders sein. Es gibt zwei Wege, die SRQ-Leitung zu überwachen:

1. Periodische Abfrage, ob Bit 5 im Status-Register SR2 gesetzt ist.
2. Zulassen eines Interrupts mit anschließender Zeilen-End-Verzweigung, ausgelöst durch Bit 3 im Status-Register SR1.

In einem System, das außer dem Rechner nur ein Gerät enthält, das seinerseits nur eine Störungsursache hat, ist eine serielle Abfrage sinnlos! Wenn aber dieses Gerät mehrere Gründe für eine SRQ-Meldung haben kann, und erst recht bei mehreren Geräten, die SRQ-Meldungen auslösen können, muß seriell abgefragt werden, damit festgestellt werden kann, wer (alles) die Meldung auslöste und was das einzelne Gerät zu melden hat.

Die serielle Abfrage wird vom aktiven Controller mit der SPOLL-Anweisung mit kompletter Adresse durchgeführt. Das HP-IB-Interface veranlaßt dann folgende Aktivitäten:

1. Die ATN-Leitung wird auf 'wahr' gesetzt.
2. Alle Listener werden mit UNL abgeschaltet.
3. Das Interface adressiert sich selbst als Listener und das abzufragende Gerät als Talker.
4. Der SPE-Befehl (Serial Poll Enable) wird ausgegeben, danach geht die ATN-Leitung auf 'falsch', damit das abgefragte Gerät sein Status-Byte senden kann. Nur wenn dieses Gerät als einziges die SRQ-Meldung auslöste, geht die SRQ-Leitung wieder auf 'falsch'. Wenn das abgefragte Gerät die SRQ-Meldung nicht oder gemeinsam mit anderen Geräten auslöste, verbleibt die SRQ-Leitung weiterhin 'wahr', um anzuzeigen, daß (mindestens noch) ein anderes Gerät ebenfalls SRQ gemeldet hat.
5. Die ATN-Leitung wird nochmals 'wahr' und es folgt der SPD-Befehl (Serial Poll Disable).
6. Zum Schluß wird mit dem UNT-Befehl der Talker wieder abgeschaltet.

Achten Sie bitte darauf, daß die SRQ-Leitung nach Abfrage eines Gerätes nur dann wieder auf 'falsch' geht, wenn das abgefragte Gerät als einziges (noch) die SRQ-Meldung auslöste! Deshalb muß das Programm festlegen, in welcher Reihenfolge die Abfrage der Geräte abläuft.

Wenn der Rechner nicht aktiver Controller ist, darf er keine Abfrage durchführen, dafür aber selber eine SRQ-Meldung abgeben. Dazu ist nur eine REQUEST-Anweisung nötig, mit der das Status-Byte vorbereitet wird. In diesem Byte muß das Bit 6 unbedingt gesetzt sein, da nur dann die SRQ-Leitung des Systems sofort 'wahr' wird. Die übrigen Bits können zur Beschreibung des Zustandes oder der Wünsche des Rechners frei definiert und entsprechend benutzt werden. Der Rechner übergibt das anstehende Status-Byte bei der seriellen Abfrage an den aktiven Controller. Nach der Abfrage durch den aktiven Controller wird die SRQ-Leitung wieder 'falsch', wenn der Rechner als einziges Gerät SRQ melden wollte. Unabhängig vom späteren Zustand der SRQ-Leitung bleibt aber das Status-Byte auch nach der Abfrage so lange bestehen, bis es durch eine neue REQUEST-Anweisung geändert wird.

Parallele Abfrage (Parallel Poll)

Bei der (zeitlich) parallelen Abfrage wird im System jedem Gerät, das abgefragt werden soll, eine der Daten-Leitungen (DIO 1 bis DIO 8) als Antwortkanal zugeordnet. Das geschieht am einzelnen Gerät durch entsprechendes Vorbereiten der Schaltung (beim IB-Interface geschah das, je nach Ausführung, mit einem Schalter oder einer Draht-Brücke).

Die Parallele Abfrage kann viel schneller arbeiten als eine serielle, da der aktive Controller die acht Daten-Leitungen praktisch gleichzeitig lesen kann und aus dem Zustand der acht Bits sofort erkennt, welche der Geräte ungewöhnliche Zustände melden wollen.

Vorteilhaft läßt sich die Parallel-Abfrage einsetzen, wenn höchstens 8 Geräte zu betreuen sind, aber auch, wenn mehrere Geräte gleicher Art in dem System ähnliche Aufgaben haben, also im System eine Gruppe bilden. Das könnten z.B. mehrere Plotter, einige Drucker und verschiedene Generatoren und Meßgeräte sein: Man kann dabei den Geräten gleicher Art (also der ganzen Gruppe!) den gleichen Antwort-Kanal zuweisen und erfährt dann durch eine einzige Parallel-Abfrage, bei welcher Geräte-Art Hilfe benötigt wird, wodurch sich die Gesamtabfrage beschleunigen läßt.

Notizen:

Die Steuer-Register und die Status-Register

Einleitung

Das Interface soll in erster Linie den Daten-Austausch zwischen dem Rechner und den mit ihm verbundenen Peripherie-Geräten ermöglichen. Die Baugruppen, die zu einem Interface benötigt werden, sind im allgemeinen sehr anpassungsfähig und zum Teil in ihrem Verhalten programmierbar, sie können also durch bestimmte Anweisungen vom Rechner auf verschiedene Arbeitsweisen eingestellt werden. Das HP-IB-Interface besitzt dazu 12 Steuer-Register, in die sich bestimmte Daten einschreiben lassen. Diese Register sind über die SET I/O-Anweisung (des Plotter-Printer-ROM's) bzw. die CONTROL-Anweisung (des I/O-ROM's) erreichbar.

Mit diesen beiden Anweisungen lassen sich die Steuer-Register mit den Nummern 0 bis 3 und 16 bis 23 ansprechen. Die Numerierung der Register erscheint etwas eigenartig, ermöglicht aber eine gewisse Kompatibilität zu anderen, älteren Systemen (wie RS 232 und GPIO), bei denen bestimmte, auch im HP-IB-System nötige Angaben schon immer in den Registern 16 bis 23 festgelegt wurden. Der Versuch, Daten in Register einzubringen, die außerhalb der zugelassenen Bereiche liegen, führt zur Fehlermeldung 111.

In der praktischen Anwendung mit den Steuer-Registern 1, 2 und 3 eng verbunden sind die Status-Register gleicher Nummer. Das HP-IB-Interface besitzt insgesamt 7 Status-Register, die mit den Nummern 0 bis 6 bezeichnet werden. Diese Register können nur abgefragt, also gelesen werden, und beschreiben den internen Zustand des Interfaces recht eingehend.

In den folgenden Ausführungen werden auch verschiedene Interrupt-Gründe erwähnt, die als "durch Zustand bedingt", "durch Ereignis bedingt" und "durch Bitte um Hilfe bedingt" unterschieden werden. Dazu ist zu bemerken, daß von keinem dieser Interrupt-Gründe exakt vorhergesagt werden kann, wann er eintritt! Die Unterscheidung bezieht sich vielmehr darauf, welche Auswirkungen die zeitliche Abfolge von Interrupt-Erwartung und Interrupt-Auftreten hat. Es ist wie im Leben: gegen manche Dinge kann man sich nur durch "Vor"-sichtsmaßnahmen sichern, bei anderen läßt sich auch "nach"-her noch etwas erreichen! Zur textlichen Vereinfachung wird das von nun an als "Zustands"-, "Ereignis"- und "Hilfe"-Interrupt bezeichnet.

Die HP-IB-Steuer-Register CR0 bis CR3

Die Steuer-Register CR0 bis CR3 haben sehr großen Einfluß auf das Verhalten des IB-Interfaces, wodurch auch die Erklärungen recht umfangreich ausfallen. Die später behandelten Steuer-Register CR16 bis CR23 legen dagegen nur fest, auf welche Art das Ende einer Datensequenz gekennzeichnet wird.

Die Steuer-Register lassen sich alle (numerisch aufeinanderfolgende auch gemeinsam) mit der CONTROL-Anweisung des I/O-ROM's in die jeweils gewünschten Zustände versetzen. Auch die SET I/O-Anweisung des Plotter-Printer-ROM's erreicht alle Steuer-Register, aber nur einzeln! Daneben werden vom I/O-ROM weitere spezielle Anweisungen geboten, die jeweils nur ein bestimmtes Steuer-Register ansprechen: So leisten trotz unterschiedlicher Form und Syntax die beiden Anweisungen

```
ENABLE INTR 7 ; 8   und   CONTROL 7,1 ; 8
```

exakt das gleiche, wobei ENABLE INTR nur auf CR1 wirken kann. Bei den Anweisungen

```
ASSERT 7 ; 1   und   CONTROL 7,2 ; 1
```

ist die Übereinstimmung dagegen nicht vollkommen, weil die gewünschte Einstellung des Registers CR2 mit verschiedener Dringlichkeit vorgenommen wird: Die CONTROL-Anweisung, formuliert um CR2 auf den Wert 1 zu bringen, beginnt erst, wenn die momentan im IB-System laufende Operation beendet ist. Die ASSERT-Anweisung, die nur CR2 erreichen kann, wird sofort ausgeführt, ohne Rücksicht auf die laufende Operation.

==========

Warnung!

Die Steuer-Register 2 und 3 (CR2 u. CR3) ermöglichen den direkten Zugriff auf die HP-IB-Steuer- und -Daten-Leitungen. Hier ist genaue Kenntnis über das HP-IB-Protokoll nötig und Vorsicht angeraten! Ungeschickte Handhabung kann hier der Anlaß zu Störungen der Bus-Operationen aber auch zu Schäden an den Geräten sein!

==========

Die folgende Tabelle gibt eine Übersicht über diese Steuer-Register, danach folgen Erklärungen für die einzelnen Register.

HP-IB-Steuer-Register CR0 bis CR3

Reg. Nr.:	Bit-Nummer: 7	6	5	4	3	2	1	0	Start-Wert:	Register-Funktion:
CR0	X	X	X	X	unge-rade	gera-de	Pari-tät 1	Pari-tät 0	0	Paritäts-Modus
CR1	IFC	LA	CA	TA	SRQ	DCL oder SDC	GET	SCG	0	Interrupt-Maske
CR2	X	REN	SRQ	ATN	EOI	DAV	NDAC	NRFD	Sinn-los	IB-Steuer-Leitungen
CR3	DIO8	DIO7	DIO6	DIO5	DIO4	DIO3	DIO2	DIO1	Sinn-los	IB-Daten-Leitungen

CR0: Paritäts-Festlegung
Es werden nur die Bits 0 bis 3 benutzt. Dabei hat jedes gesetzte Bit Vorrang vor allen benutzten höheren Bits! Es wird der Paritäts-Modus für ein- und ausgehende Daten festgelegt. Für Befehle wird keine Parität festgelegt.

Bit 0:
Falls gesetzt, werden Daten mit Paritäts-Bit "0" ausgegeben und erwartet. Falls gelöscht, hängt der Modus vom Bit 1 in CR0 ab.

Bit 1:
Falls gesetzt, werden Daten mit Paritäts-Bit "1" ausgegeben und erwartet. Falls gelöscht, hängt der Modus vom Bit 2 in CR0 ab.

Bit 2:
Falls gesetzt, werden Daten auf eine gerade Anzahl von "1'sen" ergänzt und auch so erwartet. Falls gelöscht, hängt der Modus vom Bit 3 in CR0 ab.

Bit 3:
Falls gesetzt, werden Daten auf eine ungerade Anzahl von "1'sen" ergänzt und auch so erwartet. Falls gelöscht, wird das Paritäts-Bit weder ausgegeben noch erwartet und auch nicht geprüft.

Beim Einschalten und nach RESET wird CR0 auf 0 gesetzt, das Paritäts-Bit also unterdrückt. CONTROL 7,0 ; 4 ergibt gerade Parität, CONTROL 7,0 ; 4+2 jedoch ein stets gesetztes Paritäts-Bit, weil Bit 1 von CR0 Vorrang gegenüber Bit 2 von CR0 hat!

CR1: Interrupt-Maske
Das Setzen eines Bits gestattet der dazugehörenden Interrupt-Bedingung, eine Zeilen-End-Verzweigung auszulösen.

Bit 0: SCG
Es muß während des Anliegens eines Sekundär-Befehls (SCG) gesetzt sein, wenn dadurch ein Interrupt ausgelöst werden soll. Ereignis-Interrupt!

Bit 1: GET
Es muß während des Anliegens eines Trigger-Befehls (GET) gesetzt sein, wenn dadurch ein Interrupt ausgelöst werden soll. Ereignis-Interrupt!

Bit 2: DCL oder **SDC**
Es muß während des Anliegens eines DCL- oder SDC-Befehls gesetzt sein, wenn dadurch ein Interrupt ausgelöst werden soll. Ereignis-Interrupt!

Bit 3: SRQ
Es ist zu setzen, wenn ein Interrupt durch eine 'wahre' SRQ-Leitung ausgelöst werden soll. Hilfe-Interrupt!

Bit 4: TA
Es ist zu setzen, wenn jede Adressierung zum aktiven Talker einen Interrupt auslösen soll. Zustands-Interrupt!

Bit 5: CA
Es ist zu setzen, wenn die Adressierung zum aktiven Controller einen Interrupt auslösen soll. Zustands-Interrupt!

Bit 6: LA
Es ist zu setzen, wenn jede Adressierung zum aktiven Listener einen Interrupt auslösen soll. Zustands-Interrupt!

Bit 7: IFC
Es muß während des Anliegens eines IFC-Befehls gesetzt sein, wenn dadurch ein Interrupt ausgelöst werden soll. Ein von außen (aus dem System) kommender IFC-Befehl kann diesen Interrupt auch auslösen, wenn das Interface System-Controller ist. Ereignis-Interrupt!

CR2:HP-IB-Steuerleitungen:
über dieses Register sind die 8 HP-IB-Steuerleitungen direkt zu beeinflussen. Sobald (und solange) ein Bit dieses Registers gesetzt ist, liegt die ihm zugeordnete Steuerleitung auf 'wahr'. Dem Programmierer wird die genaue Einhaltung des HP-IB-Bus-Protokolls angeraten, wenn er dieses Register benutzen will: Nur wenn das Interface aktiver Controller ist, darf die ATN-Leitung auf 'wahr' gesetzt und die REN-Leitung auf 'wahr' gehalten werden.

CR3: HP-IB-Datenleitungen
Mit diesem Register sind die HP-IB-Datenleitungen DIO1 bis DIO8 direkt steuerbar. Sobald (und solange) ein Bit dieses Registers gesetzt ist, liegt die ihm zugeordnete Datenleitung auf 'wahr'. Dem Programmierer wird die genaue Einhaltung des HP-IB-Bus-Protokolls angeraten, wenn er dieses Register benutzen will: Störungen im Interface, aber auch Beschädigungen sind möglich, wenn auf CR3 Einfluß genommen wird, solange das Interface aktiver Listener ist. Beachten Sie auch, daß die Zählung der Bits (wie üblich) von 0 bis 7 reicht, die Datenleitungen aber durch Ziffern von 1 bis 8 unterschieden werden!

Die HP-IB-Status-Register SR0 bis SR6

Diese Register lassen sich mit der STATUS-Anweisung lesen. Um z.B. den Inhalt des Statusregisters SR3 der Variablen S3 zuzuordnen, kann folgende Anweisung dienen:

```
STATUS 7,3 ; S3
```

Die folgende Tabelle gibt Übersicht über diese Register und ihren Zweck. In den folgenden Abschnitten werden die Einzelheiten erklärt.

HP-IB-Status-Register SR0 bis SR6

Reg. Nr.:	Bit-Nummer: 7	6	5	4	3	2	1	0	Start-Wert:	Register-Funktion:
SR0	0	0	0	0	0	0	0	1	1	Interface-Kennung
SR1	IFC	LA	CA	TA	SRQ	DCL oder SDC	GET	SCG	0	Interrupt-Grund
SR2	0	REN	SRQ	ATN	EOI	DAV	NDAC	NRFD	64	IB-Steuer-Leitungen
SR3	DIO8	DIO7	DIO6	DIO5	DIO4	DIO3	DIO2	DIO1	Sinn-los	IB-Daten-Leitungen
SR4	0	0	SC	A4	A3	A2	A1	A0	53	System-Controller +Adresse
SR5	SC	LA	CA	TA	SPE	Pari-tät:F	REN	LLO	160	Interface-Status
SR6	0	0	0	SC4	SC3	SC2	SC1	SC0	0	Sekundär-Befehl

SR0: Interface-Identifizierung
Abfrage liefert den Wert 1, das Kennzeichen für ein HP-IB-Interface.

SR1: Interrupt-Gründe
Dieses Register wird bei jeder Abfrage durch die STATUS-Anweisung gelöscht. Der abgefragte Wert zeigt durch den Zustand der einzelnen Bits an, welche von den im Register CR1 zugelassenen Interrupt-Ursachen inzwischen eingetreten sind. Weitere Einzel- und Besonderheiten sind aus den Ablaufdiagrammen auf den Seiten IB-60 und IB-61 ersichtlich. Die einzelnen Bits reagieren auf folgende Ursachen nur, wenn das gleichwertige Bit in CR1 gesetzt ist:

Bit 0: SCG
Anzeige für das Eintreffen eines Sekundär-Befehls. Der Wert dieses Befehls ist in SR6 abgelegt. Ereignis-Interrupt!

Bit 1: GET
Anzeige für das Eintreffen eines GET-Befehls (Triggerauslösung), wobei das Interface aktiver Listener ist. Ereignis-Interrupt!

Bit 2: DCL oder SDC
Anzeige für das Eintreffen eines DCL- oder SDC-Befehls (Device CLear oder Selected Device Clear), wobei das Interface aktiver Listener ist. Ereignis-Interrupt!

Bit 3: SRQ
Anzeige für das Eintreffen einer SRQ-Meldung (Service ReQuest), mit der ein Gerät des Systems die Aufmerksamkeit des Controllers auf sich ziehen will. Hilfe-Interrupt!

Bit 4: TA
Anzeige dafür, daß das Interface aktiver Talker wurde oder weiter bleiben soll. Zustands-Interrupt!

Bit 5: CA
Anzeige dafür, daß das Interface aktiver Controller wurde. Zustands-Interrupt!

Bit 6: LA
Anzeige dafür, daß das Interface aktiver Listener wurde oder weiter bleiben soll. Zustands-Interrupt!

Bit 7: IFC
Anzeige für das Eintreffen eines IFC-Befehls (InterFace Clear). Ereignis-Interrupt!

SR2: HP-IB-Steuerleitungen
Ein gesetztes Bit zeigt an, daß die ihm zugeordnete Steuerleitung 'wahr' ist.

SR3: HP-IB-Datenleitungen
Ein gesetztes Bit zeigt an, daß die ihm zugeordnete Datenleitung 'wahr' ist.

SR4: HP-IB-Adresse und System-Controller-Funktion

Bit 0 bis 4: Interface-Adresse, Bit 5: System-Controller
Diese Bits richten sich nach der Stellung des zum Interface gehörenden Schalters. Bit 0 bis 4 verkörpern die Adresse (Werkeinstellung: 21), Bit 5 richtet sich gleichfalls nach diesem Schalter und ist nur gesetzt, wenn das Interface System-Controller ist.

Bit 6 und 7: Sind stets gelöscht.

SR5: IB-Interface-Zustand
Aus diesem Register ist der momentane Zustand des IB-Interfaces zu ersehen.

Bit 0: LLO
Bit 0 ist gesetzt, wenn sich das Interface im LLO-Zustand (Local LockOut) befindet, also nicht vom Rechner beeinflußt werden kann.

Bit 1: REN
Bit 1 ist gesetzt, wenn sich das Interface im REN-Zustand (Remote ENable) befindet, also von außen beeinflußt werden kann.

Bit 2: Paritäts-Fehler
Bit 2 wird (nur bei eingeschalteter Paritäts-Bedingung) gesetzt, wenn einlaufende Daten nicht dem vereinbarten Paritäts-Modus entsprechen.

Bit 3: SPE
Bit 3 wird nach Empfang des SPE-Befehls (Serial Poll Enable) gesetzt, der die serielle Abfrage durch den Controller vorbereitet. Der SPD-Befehl (Serial Poll Disable) löscht das Bit 3 wieder.

Bit 4: TA
Bit 4 wird durch die Adressierung zum Talker gesetzt und mit dem UNT-Befehl, der jeder (neuen) Talker-Adressierung vorausgeht, wieder gelöscht.

Bit 6: CA
Bit 6 wird durch die Adressierung zum aktiven Controller gesetzt. Wenn das Interface System-Controller ist, wird Bit 6 bei Abgabe der Controller-Funktion an ein anderes Gerät gelöscht. Wenn das Interface nicht System-Controller ist, wird das Bit 6 bei der Weitergabe der Controller-Funktion oder bei Empfang von IFC (Inter-Face Clear) gelöscht.

Bit 7: SC
Bit 7 wird (gemeinsam mit Bit 5 von SR4) gesetzt, wenn das Interface System-Controller ist.

SR6: Register für Sekundär-Befehl

Bit 0 bis 4: Sekundär-Befehl
Die Bits 0 bis 4 verkörpern nur dann einen empfangenen Sekundär-Befehl, wenn auch gleichzeitig das Bit 0 in CR0 gesetzt ist. Sonst enthalten diese Bits als 'Sekundär'-Befehl den Befehl, der auf die Adressierung des Interfaces zu Talker oder Listener folgte.

Bit 5 bis 7:
Diese Bits werden vom Interface immer gelöscht gehalten.

Die HP-IB-Steuer-Register CR16 bis CR23

Die Steuer-Register CR16 bis CR23 geben dem Benutzer die Möglichkeit, die Zeilen-End-Sequenz (EOL) nach Wunsch zu gestalten. Diese wird jedesmal ausgegeben, wenn der "/"-Spezifikator in einer PRINT- bzw. OUTPUT-Anweisung auftritt oder wenn die Anweisung für die Datenübertragung beendet wird. Ebenso ist es aber auch möglich, bei Ausgabe des letzten Zeichens die EOI-Leitung (End Or Identify) kurzzeitig auf 'wahr' zu setzen und damit das Ende der Daten anzuzeigen.

Die folgende Tabelle gibt Übersicht über diese Register und deren Wirkungen. In den folgenden Absätzen werden die Einzelheiten erklärt.

HP-IB-Zeilen-End-Sequenz-Register CR16 bis CR23

Reg. Nr.:	Bit-Nummer: 7	6	5	4	3	2	1	0	Start-Wert:	Register-Funktion:
CR16	EOI ein	X	X	X	X	EOL2	EOL1	EOL0	2	EOI,EOL-Umfang
CR17	----------Startwert : CHR$(13) :					Wagenrücklauf---------			13	Zeichen 1
CR18	----------Startwert : CHR$(10) :					Zeilenvorschub--------			10	Zeichen 2
CR19									0	Zeichen 3
CR20									0	Zeichen 4
CR21									0	Zeichen 5
CR22									0	Zeichen 6
CR23									0	Zeichen 7

CR16: Form der EOL-Sequenz und EOI-Option

Hier wird die Zeichenzahl der EOL-Sequenz festgelegt und außerdem vereinbart, ob die EOI-Leitung zur Kennzeichnung des Abschlusses dienen soll.

Bit 0 bis 2: Zeichenanzahl
Die Summe der Wertigkeiten der gesetzten Bits entscheidet über die Zahl der ausgegebenen Zeichen.

Bit 3 bis 6: bleiben wirkungslos

Bit 7: EOI-Option
Wenn Bit 7 gesetzt ist, wird die Steuerleitung für die Dauer der Übertragung des letzten Bytes 'wahr'. Das ist das letzte Zeichen der EOL-Sequenz, wenn wenigstens eines der Bits 0 bis 2 gesetzt ist, sonst das letzte Zeichen der Meldung (was nur bei PRINT-, DISP- und TRANSFER-Anweisungen zulässig ist).

CR17 bis 23: Zeichenregister

In diesen Registern können bis zu sieben Bytes als EOL-Sequenz abgelegt werden. Als Ersatzwert stehen in CR17 und CR18 Wagenrücklauf und Zeilenvorschub, die gemäß dem Ersatzwert 2 im CR16 ausgegeben werden.

Um z.B. mit der EOL-Sequenz einen doppelten Zeilenvorschub auszulösen, muß der EOL-Umfang mit 3 und die Sequenz mit 'CR, LF, LF' eingegeben werden:

```
CONTROL 7,16 ; 3,13,10,10
```

Steuer-Register-Beeinflussung mit dem Plotter-Printer-ROM

Auch wenn dem Rechner nur ein Plotter-Printer-ROM zur Verfügung steht, können Sie alle Steuer-Register von CR0 bis CR3 und CR16 bis CR23 in die von ihnen gewünschten Zustände versetzen.

Weil das Plotter-Printer-ROM jedoch keine Abfrage der Status-Register zuläßt (dazu ist die STATUS-Anweisung nötig, die nur das I/O-ROM bietet), sollten Sie Eingriffe in die Register CR0 bis CR3 unterlassen, damit Störungen im IB-Bus-System vermieden werden.

Sinnvoll kann dagegen auch hier ein Einwirken auf die Zeilen-End-Sequenz-Register CR16 bis CR23 sein. Um mit der SET I/O-Anweisung eine von der Norm abweichende EOL-Sequenz zu definieren, müssen Sie jedes der betroffenen Register einzeln ansprechen. Wenn, wie auch im letzten Beispiel angenommen, eine EOL-Sequenz mit doppeltem Zeilenvorschub (also insgesamt 3 Zeichen!) ausgegeben werden soll, benötigen grundsätzlich die Register CR16/17/18/19 neue Werte:

```
SET I/O 7,16,3        CR16: 3 Zeichen
SET I/O 7,17,13       CR17: Wagenrücklauf       CHR$(13)
SET I/O 7,18,10       CR18: Zeilenvorschub 1    CHR$(10)
SET I/O 7,19,10       CR19: Zeilenvorschub 2    CHR$(10)
```

Nur wenn Sie sicher sein können, daß die EOL-Register noch ihre Start-Werte enthalten, dürfen Sie sich die 2. und 3. Anweisung ersparen.

Ablauf-Diagramme für die Interrupt-Register CR1 und SR1

Stark vereinfacht kann man sich die Register CR1 und SR1 als ein Formular mit bestimmten Fragen und vorbereiteten Antwortfeldern vorstellen. Dieses Formular ist für einen Parkplatz-Wächter bestimmt. Da der Parkplatz an einer engen Einbahn-Straße liegt, kann der Wächter, wenn er will, alle Fahrzeuge bemerken. Durch unsere Fragen verlangen wir nun von ihm, auf bestimmte Dinge zu achten und diese Beobachtungen durch Ankreuzen zu 'protokollieren'. Dabei ist klar, daß ein mehrmaliges gleiches Vorkommnis nur zu einem einzigen Kreuz führt! Die STATUS-Abfrage gleicht dann der Wegnahme des Formulars, wodurch der Wächter vorübergehend seine 'Aufgabe' verliert und sich nur noch um seinen Parkplatz kümmert. Erst wenn er ein neues Formular mit neuen Fragen bekommt, kann er wieder tätig sein und uns dann sogar noch sagen, wer alles in der Zwischenzeit auf den Parkplatz gefahren ist. Nur die Vorgänge auf der Straße sind ihm entgangen.

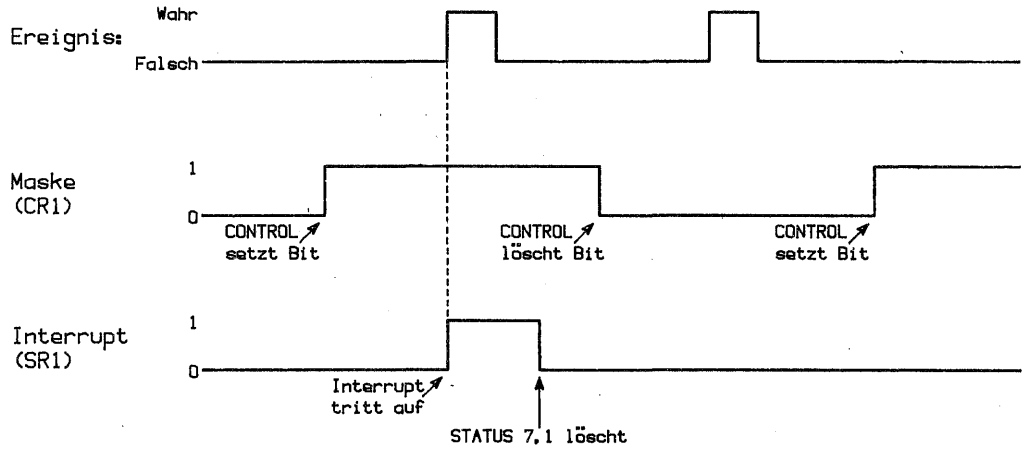
In der Formulierung unserer Fragen liegen da erstaunliche Möglichkeiten, wie die folgende Gegenüberstellung zeigen soll:

a) Machen Sie ein Kreuz, wenn der erste rote (blaue, gelb-schwarze, grüne) Sportwagen auf der Straße vorbeifährt!
b) Machen Sie ein Kreuz, wenn seit der Abholung des vorigen Formulars ein blauer (roter, grüner) Pkw geparkt hat!
c) Machen Sie ein Kreuz, wenn sich ein Lkw auf dem Parkplatz befindet!

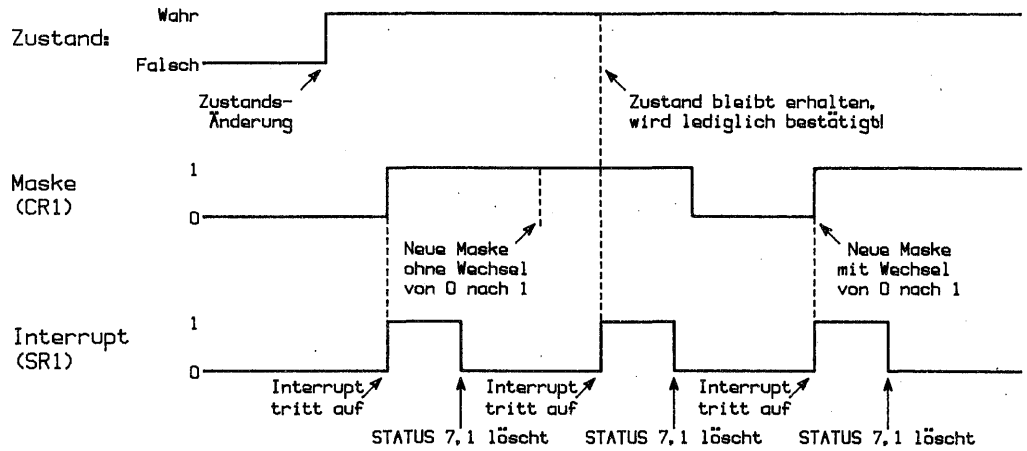
Den schnellen Sportwagen in verschiedener Farbe im Fall a entsprechen die Befehle SCG, GET, SDC/DCL und IFC. Die Pkw's im Fall b verkörpern die Adressierungen zum Talker, Controller oder Listener, und der Lkw im Fall c entspricht der SRQ-Meldung, die, genau wie der Lkw, nach und nach geprüft und entladen werden muß.

Die folgenden Ablauf-Diagramme zeigen die Reaktionen des Interfaces beim Auftreten von Interrupt-Ursachen der drei verschiedenen Arten.

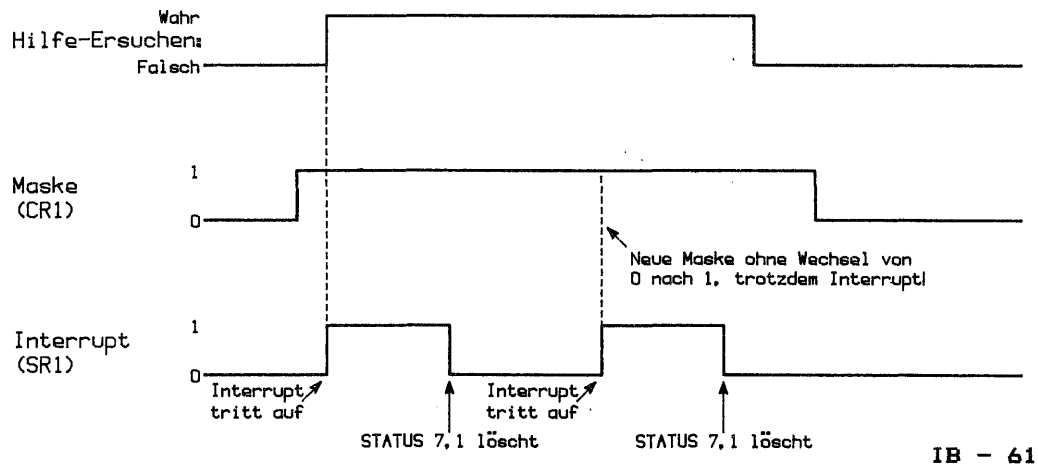
Ereignis als Interrupt-Ursache:
(SCG, GET, SDC/DCL, IFC)



Zustandsänderung als Interrupt-Ursache:
(TA, CA, LA)



Hilfe-Ersuchen als Interrupt-Ursache:
(SRQ)



SC (System-Controller)

In jedem Bus-System darf nur ein einziges Gerät System-Controller sein. Diese Sonderstellung wird dem betreffenden Gerät durch eine besondere Einstellung am Gerät selbst erteilt (für gewöhnlich durch Einstellen eines Kipp- oder Schiebeschalters). Dadurch wird dieses Gerät beim Einschalten, aber auch bei jedem Rücksetzen des Bus-Systems auf seinen Anfangszustand zum aktiven Controller. Das momentan als aktiver Controller definierte Gerät kann diese Funktion jederzeit an ein anderes Gerät übergeben, sofern dieses technisch in der Lage ist die Controller-Funktionen auszuüben. (Was nicht für alle Geräte selbstverständlich ist!).

Die Funktion des System-Controllers verbleibt dagegen immer bei dem Gerät, das vorher durch eine mechanische Einstellung dazu bestimmt wurde: Diese Funktion kann nicht an ein anderes Gerät übertragen werden. Jedesmal, wenn der System-Controller feststellt, daß irgendetwas an den Bus-Operationen nicht einwandfrei abläuft, kann er das gesamte Bus-System (mit IFC) zurücksetzen und die Funktion des aktiven Controllers wieder an sich ziehen.

Grenzen des HP-IB-Systems

Bevor wir diese Kurzdarstellung des HP-IB-Systems abschließen, sollen auch die Grenzen des Systems erwähnt werden: Die erste Einschränkung bezieht sich darauf, daß maximal 15 Geräte zu einem Bus-System zusammengefaßt werden können: Diese Beschränkung ist durch die Auslegung und Belastungsfähigkeit der Leitungstreiber gegeben. Eine weitere Einschränkung ist auch die Vorschrift, daß die zur Verbindung der Geräte benötigten Kabel insgesamt nur 20 m lang sein dürfen: Die Spannungen auf den verschiedenen Leitungen können sich nämlich nicht schlagartig ändern, sondern benötigen dazu eine gewisse (kurze) Zeit, die der gesamten Kabellänge (Kapazität!) proportional ist. Die Begrenzung der Kabellänge stellt sicher, daß das System auch bei seiner höchsten zugelassenen Übertragungsgeschwindigkeit noch einwandfrei arbeitet. Es ist zu bedenken, daß das HP-IB-System hauptsächlich entwickelt wurde, um räumlich eng benachbarte Geräte auf einfache Weise miteinander zu verbinden. Für den Datenaustausch zwischen Geräten, die weit voneinander entfernt aufgestellt sind, eignen sich andere (vor allem die seriell arbeitenden) Interface-Ausführungen besser.

Zusammenfassung der HP-IB-(I/O-ROM-)Anweisungen

Die folgende Tabelle faßt die Bedingungen zusammen, die das Interface erfüllen muß, damit bestimmte Programm-Anweisungen zulässig sind und nennt die hierdurch ausgelösten Vorgänge. Bedenken Sie, daß diese Anweisungen nicht durch das Interface sondern das I/O-ROM ermöglicht werden.

Form der Anweisung	zulässig bei Status	Zusatz-Bedingungen	ausgelöste Aktionen (Bus-Sequenzen in Fettdruck)
ABORTIO 7	SC	keine	IFC, wird aktiver Controller **[IFC]**
	$\overline{SC}$ & CA	keine	MTA, beläßt ATN wahr **[ATN*MTA]**
	$\overline{SC}$ & $\overline{CA}$	keine	Beendet I/O-Operation (Es erfolgt keine Ausgabe!)
ASSERT 7;X	alle	keine	setzt Wert sofort in CR2 IFC-Bit bleibt unverändert!
CLEAR 701	CA	keine	Gerät 701 wird adressiert, erhält dann SDC **[ATN*UNL,MTA,LAG,SDC]**
CLEAR 7	CA	keine	Keine Adressierung. Sendet DCL. Weil ATN wahr bleibt, RESUME 7 günstiger, wenn ATN auf falsch gehen soll!) **[ATN*DCL]**
CONTROL 7,n;X	alle	keine	Setzt X nach CRn, sobald Interface laufenden Vorgang beendet hat
ENABLE INTR 7;X	alle	keine	Setzt X nach CR1, sobald Interface laufenden Vorgang beendet hat
ENTER 705;X	CA	keine	Gerät 05 wird Talker, Rechner wird Listener. Eingehende Daten werden nach X gesetzt (Siehe Seite IB-42)
ENTER 7;X	CA	LA	Daten werden nach X gesetzt. (es erfolgt keine Ausgabe!)
	$\overline{CA}$	LA-Status abwarten	Wartet auf Listener-Status, setzt dann eingehende Daten nach X (Es erfolgt keine Ausgabe!)
HALT	alle	keine	Beendet I/O-Operation (Es erfolgt keine Ausgabe!)
LOCAL 7	SC	keine	$\overline{REN}$ wird auf falsch gesetzt **[REN]**
LOCAL 701	CA	keine	Nach Adressierung wird GTL ausgegeben, (Achtung: ATN bleibt wahr, läßt sich mit RESUME 7 auf falsch setzen) **[ATN*UNL,MTA,LAG,GTL]**
LOCAL LOCKOUT 7	CA	keine	LLO wird ausgegeben. **[ATN*LLO]**

Form der Anweisung	zulässig bei Status	Zusatz-Bedingungen	ausgelöste Aktionen (Bus-Sequenzen in Fettdruck)
OUTPUT 705;X	CA	keine	Rechner wird Talker Gerät 05 wird Listener, die Daten X werden ausgegeben. (Siehe Seite IB-42)
OUTPUT 7;X	CA	TA	Die Daten X werden ausgegeben (Es erfolgt keine weitere Ausgabe!)
	$\overline{CA}$	TA-Status abwarten	Wartet auf Talker-Status, gibt dann Daten X aus. (Es erfolgt keine weitere Ausgabe!)
PASS CONTROL 715	CA	keine	Gerät 15 wird Talker, dann wird TCT ausgegeben. **[ATN*UNL,MLA,TAG, UNL,TCT,$\overline{ATN}$]**
PASS CONTROL 7	CA	keine	Keine Adressierung, sendet TCT **[ATN*UNL,TCT,$\overline{ATN}$]**
PPOLL(7)	CA	keine	Sendet IDY (Identifizierung) **[ATN*EOI,(6 µs),$\overline{ATN}$*$\overline{EOI}$]**
REMOTE 7	SC	keine	REN wird auf wahr gesetzt **[REN]**
REMOTE 701	SC	keine	REN wird auf wahr gesetzt, Gerät 01 wird adressiert. Achtung: ATN bleibt wahr! **[REN,ATN*UNL,MTA,LAG]**
REQUEST 7;X	$\overline{CA}$	keine	Falls Bit 6 von X auf 1, wird SRQ wahr. Der Rechner antwortet mit X auf serielle Abfrage setzt danach SRQ auf falsch.
RESET 7	alle	keine	Interface auf Einschaltzustand. Falls Interface SC ist, wird IFC wahr, danach REN falsch und kurz danach REN wieder wahr.
RESUME 7	CA	keine	ATN wird auf falsch gesetzt. **[$\overline{ATN}$]**
SEND 7;Befehle	CA	keine	Gibt die angegebenen Befehle aus, ATN wird und bleibt wahr.
SEND 7;Daten	alle	TA	Gibt die angegebenen Daten aus, ATN wird und bleibt falsch.
SPOLL(7)	CA	LA	Führt serielle Abfrage durch **[ATN*SPE,$\overline{ATN}$,<Daten>, ATN*SPD,UNT]**
SPOLL(724)	CA	keine	Gerät 724 wird Talker, dann Abfrage. **[$\overline{ATN}$*UNL,MLA,TAG,SPE, $\overline{ATN}$,(Daten),ATN*SPD,UNT]**
STATUS 7,n;X	beliebig	keine	Besetzt X mit dem Wert von SRn.
TRIGGER 7	CA	keine	Sendet GET aus. **[ATN*GET]**
TRIGGER 701	CA	keine	Nach Adressierung wird GET ausgegeben. (Achtung: ATN bleibt wahr!) **[ATN*UNL,MTA,LAG,GET]**

Funktions-Prüfung

Das nachstehend aufgelistete Programm kann in den Rechner eingegeben und zum Prüfen des IB-Interfaces benutzt werden. In den meisten Fällen wird der Ablauf des Test-Programms Aufklärung darüber geben können, ob das IB-Interface einwandfrei oder fehlerhaft arbeitet.

Achtung!
Vor Durchführen dieser Prüfung sollten Sie sich vergewissern, daß der Rechner der Serie 80 durch entsprechende Schalter-Einstellung System-Controller ist! (Hierzu IB-Abschnit 2!)

Die Peripherie-Geräte dürfen beim Test am Rechner angeschlossen bleiben.

Schalten sie den Rechner ein und prüfen Sie durch Eingabe von

```
IOBUFFER A$ (END LINE),
```

ob dem Rechner ein I/O-ROM zur Verfügung steht (es darf keine Fehlermeldung auftreten). Falls die Fehlermeldung 'BAD STMT' auftritt, schalten Sie den Rechner wieder aus, bauen Sie ein I/O-ROM ein, wie es in der Bedienungsanleitung zum ROM-Einschub HP 82936A beschrieben ist, und schalten dann den Rechner wieder ein!

Geben Sie dann das folgende Programm in gewohnter Weise ein!

```
  10 ! IB-INTERFACE-TEST
  20 CLEAR @ OPTION BASE 0
  30 INTEGER A1,A2,B(7),I,J,S,N
  40 DISP "Test für HP-IB-Interface!" @ DISP "Mit IB-Interface besetzt:"
     @ GOSUB 180
  50 IF J#0 THEN 60 ELSE DISP "Kein HP-IB-Interface vorhanden!" @ GOTO 170
  60 DISP "Welchen Auswahl-Code hat das zu testende Interface";
  70 INPUT S
  80 GOSUB 270
  90 DISP "Wieviel Testläufe";
 100 INPUT N
 110 RESET S
 120 GOSUB 300
 130 GOSUB 540
 140 N=N-1
 150 IF N>0 THEN 120
 160 DISP "Test fehlerfrei beendet für" @ DISP "IB-Interface ";S
 170 END
 180 J=0
 190 FOR I=0 TO 7 @ B(I)=0 @ NEXT I
 200 FOR I=3 TO 10
 210 IF RIO(I,0)>0 THEN 250
 220 STATUS I,0 ; B(I-3)
 230 IF B(I-3)#1 THEN 250
 240 J=1 @ DISP I;
 250 NEXT I
 260 RETURN
 270 IF S<3 OR S>10 THEN 290
 280 IF B(S-3)#1 THEN 290 ELSE RETURN
 290 DISP "Kein Auswahl-Code für HP-IB-Interface!" @ GOTO 40
```

```
300 DISP "Stellung der Interfaceschalter  und'HANDSHAKE' im Test!"
310 A2=0
320 A1=53
330 WIO S,0;2
340 WIO S,1;230
350 IF RIO(S,0)<128 THEN 390
360 A2=A2+1 @ IF A2>2 THEN GOSUB 600
370 IF A2=3 THEN 390
380 GOTO 350
390 WIO S,0;0
400 WIO S,1;3
410 IF RIO(S,0)<128 THEN 450
420 A2=A2+1 @ IF A2>4 THEN GOSUB 600
430 IF A2=4 THEN 450
440 GOTO 410
450 WIO S,0;2
560 WIO S,1;120
470 WIO S,0;0
480 IF RIO(S,0)<1 THEN 480
490 A1=RIO(S,1)
500 IF RIO(S,0)#0 THEN GOSUB 610
510 IF A1>192 THEN A1=A1-192
520 IF A1#53 THEN GOSUB 640
530 RETURN
540 DISP "Status-Register werden getestet!"
550 FOR I=0 TO 5
560 STATUS S,I ; B(I)
570 NEXT I
580 IF B(0)#1 OR B(1) OR B(2)#64 OR B(3) OR B(4)#A1 OR B(5)#160 THEN GOSUB 680
590 RETURN
600 DISP "Fehler beim 'HANDSHAKE'!" @ GOTO 620
610 DISP "Zum erwarteten Zeitpunkt ist der Xlator nicht gelöscht!"
620 DISP "Wahrscheinlich Prozessor oder   Xlator fehlerhaft!"
630 RETURN
640 DISP "Schalter stehen anders als bei  Auslieferung!"
650 A$=DTB$(A1)
660 DISP "Elemente 2-7 stehen  "&A$[11,16] @ DISP "Lieferzustand war:   110101"
670 RETURN
680 DISP "Status-Register 0 bis 5 haben   falsche Inhalte:"
690 FOR I=0 TO 2
700 DISP "SR"&VAL$(I)&"=";B(I),"SR"&VAL$(I+3)&"=";B(I+3)
710 NEXT I
720 RETURN
```

Das Programm fragt alle Interface-Auswahl-Codes nach HP-IB-Interfaces ab und gibt die damit besetzten Auswahl-Codes bekannt. Danach muß entschieden werden, welches IB-Interface wie oft getestet werden soll. Der Ablauf des Testes kann an (wiederholten) Anzeigen verfolgt werden. Der erfolgreiche Abschluß wird angezeigt.

Der Hinweis auf die vom Lieferzustand abweichende Stellung der Schalter bedeutet im Prinzip keinen Fehler, da diese Schalter vom Benutzer für seine Zwecke passend eingestellt sein können.

Von den Status-Registern werden SR0 bis SR 5 gelesen (SR6 ist unwichtig!). Dabei sollten sich die Werte

1 0 64 0 53 160

ergeben, wobei lediglich der Inhalt von SR4 vom oben gezeigten Wert (53) abweichen darf. In diesem Register bildet sich die Stellung der Schaltelemente 2 bis 7 ab, mit denen die Festlegung als System-Controller und die Talk-Listen-Adresse festgelegt wurden. Der kleinste Wert, der bei diesem Test in SR4 auftreten kann, ist 32. Dann hat der Rechner die Adresse 0. Als größter Wert ist 62 möglich, dann ist die Rechner-Adresse auf 30 gesetzt. Werte unter 32 sind unmöglich, weil dann der Rechner kein System-Controller wäre und das Programm nicht ablaufen würde.

HP-IB-Fehlermeldungen

Nummer:	Meldung:	Bedeutung bzw.Ursache:
101	XFR	Nur Warnung! Es wurde eine laufende TRANSFER-Operation unterbrochen!
110	I/O CARD	Interface-Test lief nicht ohne Fehler ab. Kann Hinweis auf Geräte-Fehler sein!
111	I/O OPER	Die geforderte I/O-Operation ist wegen falscher Parameter oder grundsätzlich mit dem IB-Interface nicht möglich!
112	I/O-ROM	Das I/O-ROM hat den Test nicht bestanden, Geräte-Fehler, der nicht vom IB-Interface verursacht wurde!
113		Das Ausführen dieser Anweisung setzt voraus, daß der Rechner System-Controller ist.
114		Das Ausführen dieser Anweisung setzt voraus, daß der Rechner aktiver Controller ist.
115		Das Ausführen dieser Anweisung setzt voraus, daß der Rechner aktiver Talker ist. Anlaß für diesen Fehler kann eine PRINT-Anweisung sein, vor der lediglich der Auswahl-Code festgelegt wurde. Im IB-System müssen vor Ausführung jeder Ausgabe-Anweisung Talker und Listener festgelegt sein.
116		Das Ausführen dieser Anweisung setzt voraus, daß der Rechner aktiver Listener ist. Anlaß kann eine nur mit dem Auswahl-Code versehene PLOTTER IS- oder eine PRINTER IS-Anweisung sein, durch die der Rechner Talker wurde. Im IB-System müssen Talker und Listener vor Ausführung jeder Eingabe-Anweisung festgelegt sein.
117		Das Ausführen dieser Anweisung setzt voraus, daß der Rechner nicht aktiver Controller ist.
123	NO ";"	Syntax-Fehler: Es fehlt ein Semikolon in der Anweisung.
124	ISC	Es gibt kein Interface mit dem angegebenen Auswahl-Code!
125	ADDR	Geräte-Adresse fehlerhaft oder kein Gerät vorhanden, daß diese Adresse hat!
126	BUFFER	Buffer-Fehler! Angegebene String-Variable wurde nicht mit IOBUFFER deklariert oder Buffer ist schon voll (vor Beginn einer ENTER-Anweisung) bzw. leer (vor Beginn einer OUTPUT- oder TRANSFER-Anweisung).
127	NUMBER	Tritt auf, wenn bei Aufnahme von Daten die Zeichenfolge keinen Zahlenwert ergibt oder wenn ein Zahlenwert ausgegeben werden soll, der die Grenzen des mit "e" definierten Bereichs überschreitet.
128	EARLY TERM	Vorzeitiger Abbruch! Der Buffer ist ausgelesen, bevor die COUNT-Bedingung bzw. die ENTER-Liste erfüllt war.
129	VAR TYPE	Die Art der in der Eingabe-Liste genannten Variablen paßt nicht zum dafür vorgesehenen IMAGE-Spezifikator.
130	NO TERM	Das Interface bzw. der Buffer wartet auf das vereinbarte Abschluß-Zeichen für die ENTER-Anweisung. Als Abschluß wirkt noch das Zeilenvorschub-Symbol.

Beispiele für Fehler-Meldungen

Anweisung:	Wirkung:
SET I/O 7,10,0	Fehler 111, weil das Steuer-Register 10 im HP-IB-Interface nicht existiert!
REMOTE 7	Fehler 113, falls Rechner nicht System-Controller ist!
OUTPUT 702 ; A$	Fehler 114, falls Rechner nicht aktiver Controller ist!
OUTPUT 7 ; A$	Fehler 115, falls Rechner nicht aktiver Talker ist! Die Funktion des aktiven Controllers ist hierfür belanglos!
ENTER 7 ;A$	Fehler 116, falls Rechner nicht aktiver Listener ist! Die Funktion des aktiven Controllers ist hierfür belanglos!
REQUEST 7 ; <Byte>	Fehler 117, falls Rechner aktiver Controller ist! REQUEST-Meldungen dürfen nur von Geräten ausgegeben werden, die nicht als Controller aktiv sind!

Notizen:

Allgemeine Informationen

Einführung

Das HP-IL-Interface HP 82938A ermöglicht die Verbindung der Rechner der Serie 80 mit dem HP-IL-System. In diesem System sind die Rechner der Serie 80 in gewissem Sinne Außenseiter, weil die meisten der HP-IL-Geräte, wegen ihres geringen Energiebedarfs aus Batterien versorgt werden.

Hier wird beschrieben, wie das HP-IL-Interface anzuschließen und zu benutzen ist. Eingehendere Informationen über die einzelnen Peripherie-Geräte sind in den Handbüchern für diese Geräte enthalten. Ebenso ist auch die Beschreibung für das I/O-ROM eine vorzügliche Quelle für weitere Informationen bei speziellen Fragen.

Wenn das HP-IL-Interface den Rechner der Serie 80 mit einem IL-System verbindet, kann er auf alle in diesem System vereinigten Geräte Einfluß nehmen und sich am allgemeinen Datenaustausch beteiligen. Diese Übertragung über eine Zweidraht-Leitung geschieht seriell und asynchron, wobei die Daten stets in einer Richtung von einem Gerät zum nächsten weitergereicht werden. Alle Geräte müssen deshalb gewisse mechanische, elektrische und funktionelle Eigenschaften aufweisen, um "HP-IL-fähig" zu sein.

Bild 1-1. Typische Interface-Schleife

Als Vorteile des Systems sind die geringen Abmessungen und der niedrige Energiebedarf der Geräte, die Preiswürdigkeit und die einfache und leichte Verbindungs-Technik zu nennen, die es auch zum Überbrücken mittlerer Entfernungen zwischen den Geräten geeignet macht.

Das HP-IL-System ist offensichtlich eine Weiterentwicklung des inzwischen genormten HP-IB-Systems (IEEE 488 bzw. IEC 625), dessen universelle Möglichkeiten weitgehend beibehalten werden sollten, obwohl dieses neue System für ganz andere Anwendungsgebiete mit geringerer maximaler Übertragungsgeschwindigkeiten konzipiert ist. Das Verstehen des IL-Systems setzt zwar keine Kenntnisse voraus, die das HP-IB-System betreffen, man wird also viele gut bewährte Techniken aus dem IB-System wiedererkennen, aber auch einige neue Ideen, die das System für den Benutzer noch bequemer machen!

Benötigte ROM's

Zur Durchführung von Input/Output-Funktionen braucht das IL-Interface mindestens eines der nachstehend angegebenen ROM's. Der Typ des ROM's entscheidet dabei, in welchem Umfang Schleifen-Operationen möglich sind:

Für HP-83/85:

Input/Output-ROM (Bestellnummer 00085-15003)

Plotter/Printer-ROM (Bestellnummer 00085-15002)

Für HP-86/87:

Input/Output-ROM (Bestellnummer 00087-15003)

Plotter-ROM (Bestellnummer 00087-15002)

Das I/O-ROM ist für universelle Input/Output-Operationen mit einem sehr weiten Anwendungsbereich bestimmt, während das Plotter/Printer-ROM speziell die Durchführung von Druck- und Plott-Vorgängen sehr vereinfacht.

Setzen Sie (mindestens) eines dieser ROM's in den ROM-Einschub HP 82936A ein, und stecken Sie den ROM-Einschub in einen der vier Steckplätze an der Rückseite des Rechners, bzw. überzeugen Sie sich durch Eingabe eines für das entsprechende ROM typischen Befehls, daß eine geeignete ROM-Bestückung vorliegt.

Auswahl-Code

Der Interface-Auswahl-Code ist notwendig, um in den Programmen festzulegen, über welches Interface Daten vom Rechner ausgegeben oder aufgenommen werden sollen. Der Auswahl-Code für das HP-IL-Interface wurde bei der Auslieferung vom Hersteller auf "9" eingestellt. Jedes mit dem Rechner verbundene Interface muß einen besonderen Auswahl-Code erhalten. Der Auswahl-Code kann durch Verstellen von Schaltern (auf der Platine des IL-Interfaces) geändert werden. Die Durchführung dieser Änderung ist im IL-Abschnitt 2 ausführlich beschrieben.

Technische Daten

Abmessungen:	167 mm x 127 mm x 15 mm
Zugelassene Kabellänge:	bis zu 10 m Standard-Kabel von Gerät zu Gerät
Arbeitstemperatur:	0 bis 55 °C
Lagertemperatur:	-40 bis 65 °C
Steck-Verbinder:	2-polig (2 Stück)
Signal-Pegel:	1,5 V Wechselspannung
Spannungsversorgung:	über die I/O-Steckverbinder des Rechners

Schleifen-Funktionen

Jedes in die Schleife einbezogene Gerät kann grundsätzlich Daten senden, empfangen und bearbeiten, sofern es die hierzu notwendigen Eigenschaften aufweist. Die wichtigsten sind nachstehend hier aufgeführt:

Listener
Ein Gerät mit Listener-Eigenschaft ist als aktiver Listener fähig, die vom aktiven Talker ausgegebenen Daten aufzunehmen.

Talker
Ein Gerät mit der Talker-Eigenschaft ist als aktiver Talker in der Lage, Daten an einen oder mehrere Listener auszugeben. Es darf immer nur ein Gerät aktiver Talker sein.

System-Controller
Beim Einschalten muß eines (und nur eines!) der Geräte System-Controller in dem HP-IL-Systems sein. Bei den Rechnern der Serie 80 kann durch einen Schalter festgelegt werden, daß der Rechner diese Aufgabe an sich zieht.

Der System-Controller setzt die IL-Schleife auf den Einschaltzustand und wird dadurch aktiver Controller. Durch sein Sonderrecht kann der System-Controller die IL-Schleife jederzeit auf den Einschaltzustand zurücksetzen und dadurch wieder aktiver Controller werden, auch wenn er diesen Status in der Zwischenzeit an ein anderes Gerät der Schleife abgegeben hatte.

Aktiver Controller
Der aktive Controller bereitet den Daten-Austausch vor, indem er festlegt, welches Gerät als Talker und welche(s) als Listener arbeiten soll(en). Der aktive Controller kann auch spezielle Befehle, wie Trigger- und Rücksetz-Anweisungen für bestimmte Geräte ausgeben. Schließlich kann der aktive Controller diese Funktion an jedes andere dazu befähigte Gerät weitergeben.

Serielle Abfrage
Der aktive Controller kann nacheinander jedes Gerät nach seinem Zustand befragen, wobei das Gerät die Möglichkeit bekommt, auf eigene "Probleme" hinzuweisen.

Parallele Abfrage
Der aktive Controller kann mehrere Geräte gleichzeitig über bestimmte Zustände befragen, wenn diese Geräte vorher geeignet vorbereitet wurden.

Für alle diese Funktionen finden sich in den nächsten Abschnitten ausführliche Erklärungen und Anwendungsbeispiele.

Inbetriebnahme

Auspacken und Prüfen auf Beschädigungen

Falls der Versandkarton beschädigt ist, bitten Sie einen Beauftragten des Spediteurs, beim Auspacken anwesend zu sein. Falls Sie bei unbeschädigtem Versandkarton nach dem Auspacken mechanische Schäden am Interface feststellen oder das Interface den Funktionstest nicht besteht, informieren Sie sofort den Spediteur und die nächstgelegene HP-Vertretung. Heben Sie den Versandkarton für die Prüfung durch den Spediteur auf. Die HP-Vertretung wird dann Reparatur oder Austausch des Gerätes veranlassen, ohne die Abwicklung der Ersatzansprüche gegen den Spediteur abzuwarten.

Schalter-Einstellungen

Die Schalter für den Auswahl-Code und für die Übernahme der Funktion als System-Controller wurden bei Auslieferung wie folgt eingestellt:

Auswahl-Code: 9

System-Controller-Funktion: möglich

Wenn es nicht nötig ist, sich von der Stellung der Schalter zu überzeugen oder diese zu ändern, können Sie auf Seite IL-XX weiterlesen, wo der Einbau des Interfaces beschrieben wird.

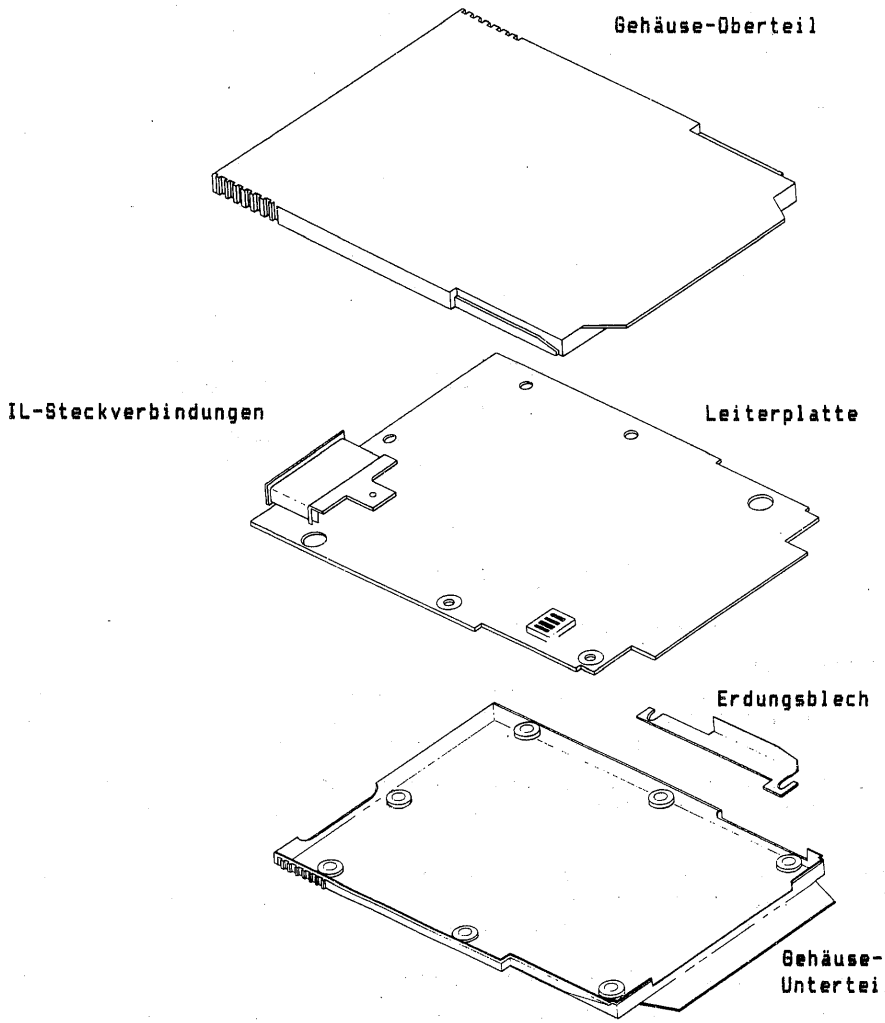
Zur Kontrolle oder Veränderung der Schalterstellungen muß das Gehäuse des Interfaces geöffnet werden. Die hierzu nötigen Arbeiten sind nachstehend beschrieben.

öffnen des Interface-Gehäuses

Das Bild auf Seite IL-5 zeigt, wie die einzelnen Teile zusammengehören. Legen Sie das Interface so auf eine waagerechte, ebene Fläche, daß die Schrauben sichtbar sind und die IL-Steckverbinder nach links zeigen. Führen Sie dann die folgenden Arbeiten durch:

1. Lösen Sie mit einem Kreuzschlitz-Schraubendreher (möglichst "Pozidriv") die sieben jetzt sichtbaren Schrauben vollständig.
2. Greifen Sie unter das Gehäuse, halten Sie alle Teile zusammen, und drehen Sie nun das Gehäuse so um, daß die Steckverbinder immer noch nach links zeigen. Achten Sie dabei auf den Verbleib der Schrauben!
3. Heben Sie nun die obere Gehäusehälfte ab. Überzeugen Sie sich, daß die Lage der Einzelteile dem Bild auf Seite IL-5 entspricht. Wichtig ist, daß Sie den im Bild hervorgehobenen Schalter finden.

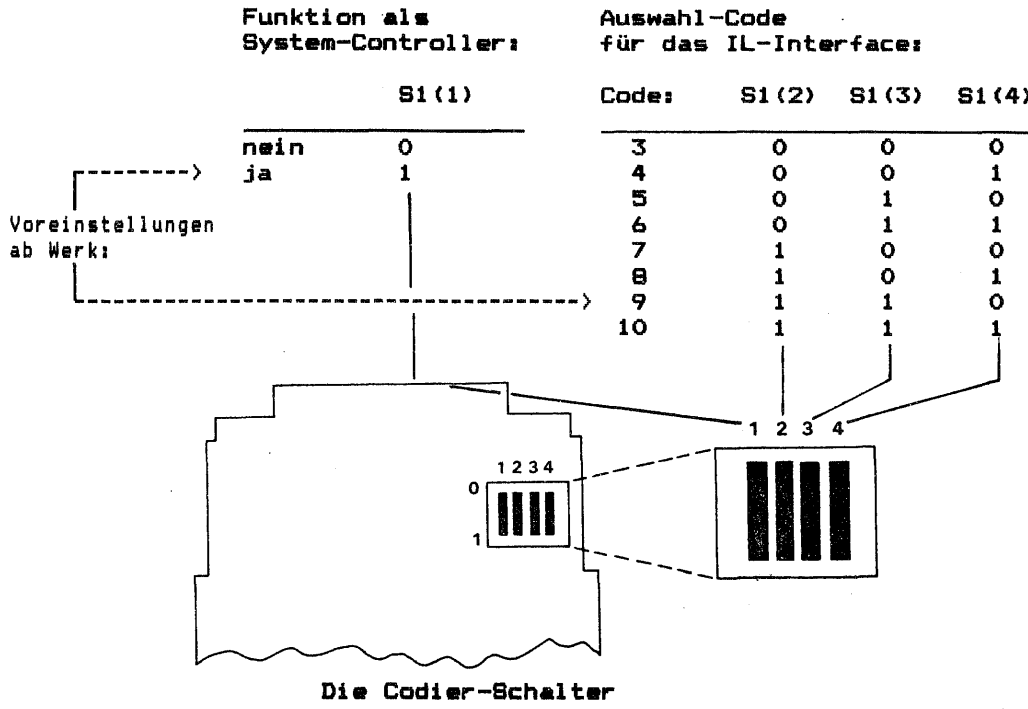
Beim späteren Zusammenbau der Teile ist in umgekehrter Reihenfolge zu verfahren und darauf zu achten, daß das Erdungsblech zwischen Lötseite der Leiterplatte und Gehäuseunterteil an der richtigen Seite liegt.



Montage-Zeichnung

Für eine eventuelle Verstellung der Schalter-Elemente beziehen wir uns auf das Bild auf dieser Seite. Die Elemente 2, 3 und 4 legen den Interface-Auswahl-Code fest, das Segment 1 entscheidet darüber, ob der Rechner die Rolle des System-Controllers übernimmt.

Achtung! Wenn Sie die Werks-Einstellung ändern wollen, achten Sie bitte darauf, daß Sie die richtigen Elemente verstellen! Am besten eignet sich dafür die Spitze eines Bleistiftes oder ein ähnliches, schlankes Werkzeug.



Die Codier-Schalter

Die Schalter für den Auswahl-Code

Mit den Schaltelementen S1(2) bis S1(4) lassen sich die Auswahl-Codes von 3 bis 10 einstellen. Drehen Sie das Interface so, wie es das Bild zeigt. Orientieren Sie sich an den Markierungen "0" und "1" auf der Printplatte. Jetzt können Sie prüfen, welcher Code eingestellt ist (im Anlieferungszustand z.B. "9").

Um einen anderen gewünschten Auswahl-Code einzustellen, bedienen Sie sich der Tabelle rechts oben. Für den Code "3" müßten z.B. alle drei Schalterelemente auf "0" stehen.

Denken Sie daran, daß jeder Auswahl-Code nur einmal vorkommen darf, wenn mehr als ein Interface im Rechner vorhanden ist. Bei mehrfacher Vergabe des gleichen Auswahl-Codes wird es sonst zu Störungen kommen.

Der Schalter für die System-Controller-Funktion

Der System-Controller wird beim Einschalten und bei jedem RESET automatisch aktiver Controller. Das Element S1(1) ist bei Lieferung auf "1" gestellt, wodurch dem Rechner die System-Controller-Funktion gegeben wird. Während des Arbeitens des Systems kann der System-Controller jedes dazu geeignete Gerät in der Schleife zum aktiven Controller machen. Die System-Controller-Funktion kann dagegen nicht per Programm vergeben werden.

Wenn ihr Rechner die System-Controller-Funktion ausüben soll, dann belassen Sie das Schalterelement S1(1) in der Voreinstellung auf "1". Durch Vergleich mit dem Bild auf Seite IL-6 können Sie diesen Zustand leicht kontrollieren.

Wenn ihr Rechner die System-Controller-Funktion nicht ausüben soll (oder darf!), schalten Sie das Element S1(1) in die Stellung "0". Es ist wichtig, daß nur ein einziges Gerät im IL-System als System-Controller definiert ist. Verstöße gegen diesen Grundsatz haben gestörtes Arbeiten der Schleife und/oder die Ausgabe der Fehlermeldung 118 zur Folge.

Interface-Einbau

Bevor Sie das Interface einbauen, sollten Sie den folgenden Text bis zur nächsten Überschrift aufmerksam durchlesen!

Achtung:
Schalten Sie den Rechner und auch alle zur Peripherie gehörenden Geräte ab, bevor Sie das IL-Interface einsetzen bzw. entfernen wollen! Sie vermeiden damit die Gefahr, eines oder mehrere der Geräte zu beschädigen!

Die folgenden Tätigkeiten sollten dann der Reihe nach durchgeführt werden:

1. Schalten Sie den Netzschalter an der Rückseite des Rechners aus (Stellung "0" bzw. "OFF"). Schalten Sie auch alle Peripherie-Geräte aus, die an einem schon früher eingebauten Interface angeschlossen sind.

   Warnung:
   Stecken Sie weder Finger, noch Werkzeuge oder andere leitende Gegenstände in die Einschuböffnungen, da die Gefahr von geringfügigen elektrischen Schlägen besteht, die Sie unnötig erschrecken oder, unter Umständen, auch Störungen an Herzschrittmachern auslösen können. Außerdem besteht die Gefahr, die Kontakte und auch innere Teile des Rechners zu beschädigen.

2. Entfernen Sie die Schutzkappe einer noch unbesetzten Einschuböffnung. Andere, noch unbesetzte Einschuböffnungen sollten auch weiterhin abgedeckt bleiben.

   Achtung:
   Sie dürfen das Interface nicht mit Gewalt in die Einschuböffnung drücken, da es sonst mechanisch beschädigt wird! Das Einsetzen ist nur möglich, wenn die Beschriftung des Gehäuses von oben sichtbar ist.

3. Führen Sie das Interface in dieser Lage, mit der Steckleiste (Printplatte) voran, in die Einschuböffnung, was ohne großen Widerstand möglich sein muß. Nur für das letzte Stück (ca. 7 mm) bis zum Anliegen der Anschläge ist einige Kraft nötig, da jetzt die Steckanschlüsse hergestellt werden. Das Einsetzen geht leichter, wenn das Gehäuse abwechselnd links und rechts hineingedrückt wird. Die unterschiedliche Form der Führungsschienen auf der linken und rechten Seite verhindert einen seitenverkehrten Einbau.

Anschließen von Peripherie-Geräten

Zum IL-System gehört der Rechner der Serie 80, das IL-Interface und bis zu 30 Peripherie-Geräte. Interface und Peripherie-Geräte müssen eine geschlossene Schleife bilden, wobei die Reihenfolge der Geräte in dieser Schleife (meist) beliebig ist. Die Kabellänge zwischen zwei benachbarten Geräten darf bis zu 10 m betragen.

Zu jedem HP-IL-Gerät wird ein passendes Verbindungskabel mitgeliefert. Diese Kabel ermöglichen Aufbau oder Erweiterung der gewünschten Schleife durch unverwechselbare Steckverbindungen, die fühlbar einrasten.

Wegen der "Schleifen"-Struktur des HP-IL-Systems müssen zu jedem Gerät, auch zum IL-Interface, zwei Leitungen führen. Da an der Datenübertragung immer alle Geräte beteiligt sind, darf keines der Geräte während des Arbeitens der Schleife ausgeschaltet sein!

Abtrennen von Peripherie-Geräten

Wenn eines der Peripherie-Geräte aus der Schleife entfernt werden soll, darf man beide Steckverbindungen vom Gerät einfach abziehen und direkt miteinander verbinden, um die Schleife wieder zu schließen. Sinngemäß ist zu verfahren, wenn mehrere Geräte aus der Schleife herausgenommen werden sollen. Wenn diese Manipulation allerdings während einer laufenden Schleifen-Operation ausgeführt wird, löst dies den Abbruch der Operation und/oder eine Fehlermeldung aus!

Interface-Ausbau

Zum Ausbau des HP-IL-Interfaces 82938A sind folgende Arbeiten durchzuführen:

1. Schalten Sie den Netzschalter auf der Rückseite des Rechners aus (Stellung "0" bzw."OFF").
2. Ziehen Sie das IL-Interface vorsichtig aus der Einschuböffnung. Dabei ist zuerst (ca. 7 mm) relativ viel Kraft nötig, um die Steckverbindungen zu trennen. Die Trennung geht leichter, wenn am Gehäuse abwechselnd links und rechts gezogen wird. Danach muß sich das Interface praktisch ohne Widerstand aus der Einschuböffnung herausnehmen lassen. Das Interface soll in seinem Original-Karton oder auf andere Weise gut geschützt aufbewahrt werden.
3. Die nunmehr leere Einschuböffnung ist mit einer Schutzkappe zu verschließen.

IL-Abschnitt 3

Kurz-Einführung in das HP-IL-System

Die Grund-Idee für das Hewlett-Packard-Schleifen-Interface (HP-IL) ist leicht zu verstehen und in der Praxis bequem anzuwenden: Ein Rechner der Serie 80 und alle zu dem System gehörenden Geräte bilden durch die Kabel eine in sich geschlossene Schleife, in der Daten (stets in einer Richtung) transportiert werden können. Die Information, die zwischen beliebigen Geräten der Schleife auszutauschen ist, kann vom sendenden Gerät nur an das nächste weitergegeben werden. Dieses (und die folgenden) reicht die Nachricht weiter, bis sie das Ziel erreicht hat. Die Nachricht wird aber auch vom Empfänger weitergereicht und kommt schließlich wieder beim Absender an, der jetzt prüfen kann, ob nichts verstümmelt wurde.

In manchen Gegenden gibt es ein lustiges Kinderspiel mit dem Namen "Stille Post": Dabei sitzen die Kinder im Kreis nebeneinander, und eines beginnt das Spiel, indem es seinem Nachbarn ein (meist schwieriges) Wort ins Ohr flüstert. Dieses Wort macht die Runde und kommt zum Anfang zurück, 'natürlich' mehr oder weniger verändert! Darin liegt das Vergnügen! Die 'innere Verwandtschaft' zum HP-IL-System ist offensichtlich, nur der Zweck 'etwas' anders!

Es kommen also alle Geräte der Schleife mit allen Informationen in Kontakt, aber nur die Geräte, für die die Information bestimmt ist, nehmen diese auf und reagieren dann in gewünschter Weise. So kann der Rechner alle Geräte der Schleife im Rahmen ihrer Möglichkeiten beeinflussen.

Fast alle Geräte der Schleife können sowohl senden als auch empfangen (aber nie gleichzeitig!). Welche Tätigkeit überwiegt, hängt von der Art des Gerätes ab: Ein Massenspeicher wird beide Tätigkeiten etwa gleich häufig ausüben, Drucker werden vorwiegend empfangen, Meßgeräte hauptsächlich senden.

Die verwendeten Fachausdrücke sind der akustischen übermittlung entlehnt: Der Sender wird "Talker" (Sprecher), genannt, der Empfänger "Listener" (Zuhörer).

Die übertragung sollte so ablaufen, daß immer nur eine Information innerhalb der Schleife weitergegeben wird, um Konfusionen zu vermeiden: Es darf nur einer sprechen, während die übrigen mehr oder weniger interessiert zuhören. Für diese Ordnung ist ein Moderator nötig, der im HP-IL-System "aktiver Controller" heißt. Er bestimmt ein Gerät zum aktiven Talker. Der aktive Controller bestimmt aber auch, welche Geräte in der Schleife die umlaufenden Informationen aufnehmen dürfen, indem er sie zu aktiven Listenern macht. Ein Rechner der Serie 80 kann aktiver Controller sein, selbstverständlich aber auch aktiver Talker oder aktiver Listener.

In der Schleife dürfen zu einem bestimmten Zeitpunkt wohl mehrere Geräte aktive Listener sein, es darf aber immer nur einen aktiven Talker und auch nur einen aktiven Controller geben.

Alle Geräte der Schleife müssen eingeschaltet sein, auch wenn sie im Augenblick weder aktiver Talker noch aktiver Listener sind, weil die Weitergabe der Informationen sonst nicht erfolgen kann.

Struktur-Übersicht des HP-IL-Systems

Das HP-IL-System hat eine genau festgelegte Struktur, die man auch als sehr sorgfältig durchdachte Organisation bezeichnen kann. Diese allein sorgt für einen geordneten Ablauf im System. Diese Organisation ist leichter zu verstehen, wenn man das HP-IL-System mit einem Verein vergleicht, dessen Mitglieder an verschiedenen Orten wohnen und für den Kontakt untereinander eine Reihe von festen Regeln als zweckmäßig erkannt haben. Außerdem existiert eine Art Einbahn-Ring-Rohrpost, die alle Mitglieder berührt: Man informiert sich also gegenseitig mit "Rundschreiben" oder "Umläufen".

Ein Mitglied des Vereins, häufig als "Vorsitzender" bezeichnet, hat die Aufgabe, die Verhandlungen zu organisieren. Der Vorsitzende ist für den Verein verantwortlich und wendet die vereinbarten Regeln an, um die geordnete Vereinsarbeit zu erreichen. Wenn er die Leitung nicht ausüben kann oder ausüben mag, bestimmt er als Vertreter einen "geschäftsführenden Vorsitzenden".

Vorsitzender im HP-IL-System ist der "System-Controller". Seine "Wahl" geschieht durch die Einstellung eines Schalters: Diese Wahl kann per Programm nie geändert werden. Auch im HP-IL-System ist es möglich, einen "geschäftsführenden Vorsitzenden" zu bestimmen: dies ist dann der "aktive Controller". Jedes Gerät des Systems kann mit dieser Funktion betraut werden, wenn es den Ablauf der Operationen steuern kann (z.B. ein Rechner).

Wenn der System-Controller durch Einschalten in Betrieb genommen oder auf seinen Startzustand zurückgesetzt wird, übernimmt er automatisch die Rolle des aktiven Controllers. Diese Aufgabe kann aber unmittelbar danach an ein anderes Gerät abgegeben werden, damit sich der System-Controller anderen Aufgaben zuwenden kann: Ein HP-IL-System kann also durchaus auch mehrere Rechner enthalten. Es gibt aber auch Rechner, die die Controller-Funktion nie abgeben können (z.B. der HP-41).

Zwecks besserer Verständigung sollten schon in einer Versammlung niemals mehrere Redner gleichzeitig sprechen, bei der eigenartigen Verbindung der Mitglieder untereinander muß noch mehr Ordnung gehalten werden. Dafür sorgt der jeweilige Vorsitzende, der den Mitgliedern nur nacheinander "Rundschreiben" erlaubt.

Das Recht zu "Rundschreiben" hat im HP-IL nur der "Aktive Talker". Es darf immer nur ein Gerät so bezeichnet werden. Das geschieht durch "Adressierung zum Talker" und wird vom aktiven Controller vorgenommen. Das "Rundschreiben" des aktiven Talkers ist die "Meldung", womit wir zur eigentlichen Aufgabe des HP-IL kommen.

Nun interessiert nicht jedes "Rundschreiben" alle Mitglieder, der aktive Controller legt deshalb fest, wer das "Rundschreiben" lesen muß. Diese Aufforderung zum "Lesen" heißt "Adressierung zum Listener" und kann mehrere Geräte betreffen. Diese Auswahl beschleunigt den Umlauf der Rundschreiben sehr, da alle Mitglieder angewiesen sind, jede Information ungelesen weiterzugeben, wenn diese nicht für sie bestimmt sind. Nur die aktiven Listener sollen sich sofort eine Kopie anfertigen, die sie (nach Weitergeben des Originals) in Ruhe lesen können.

Der Kopier-Trick macht es überhaupt erst möglich, daß auch mehrere, relativ langsam arbeitende Geräte die Meldung fast gleichzeitig erhalten und mit der Bearbeitung beginnen können. Damit laufen die Bearbeitungszeiten auch bei mehreren langsamen Geräten praktisch parallel ab und addieren sich nicht. Die Gesamt-Bearbeitungszeit ist nur unmerklich länger als die des langsamsten beteiligten Gerätes.

Nachdem wir jetzt die Struktur des Systems in großen Zügen kennen, wollen wir nun sehen, wie die verschiedenen Aufgaben gelöst werden.

Die Datenübertragung im HP-IL-System

Es ist ein Prinzip im HP-IL-System, daß jede Meldung nach Durchlaufen der ganzen Schleife wieder beim aktiven Talker ankommt, der dann prüfen kann, ob sein "Rundschreiben" unterwegs nicht beschädigt oder verändert wurde. Von diesem Grundsatz wird nur ganz selten abgewichen.

Wenn ein "Rundschreiben" immer nur ein Wort enthalten darf, ist die Zahl der verschiedenen Meldungen durch die Zahl der Worte festgelegt. Dieses Problem hat auch das IL-System, denn es gibt ja nur 256 verschiedene 8-Bit-Kombinationen (Bytes). Beim "Rundschreiben" könnte man die Zahl der möglichen Meldungen auch bei kleinem Wortschatz dadurch erweitern, daß man die "Rundschreiben" auf Papier unterschiedlicher Farbe verschickt. Ähnlich verfährt das IL-System, indem es vor das zu ausgebende 8-Bit-Wort drei zusätzliche Bits setzt, deren Bedeutung vorher verabredet wurde. Mit dem höchsten dieser "Rahmen"-Bits (C2) wird festgelegt, ob es sich um Daten oder Befehle handelt. Bei den Daten läßt sich mit dem mittleren Bit (C1) anzeigen, daß die Daten-Sequenz beendet ist. Schließlich kann das niedrigste Bit (C0) des Rahmens von jedem Gerät der Schleife gesetzt werden, um Aufmerksamkeit zu erregen. Bei den Befehlen haben C1 und C0 andere Bedeutungen deren Erklärung hier zu umständlich ist. Dem Setzen von C0 entspräche beim "Rundschreiben" etwa die Lochung des Originals durch eines der Mitglieder, was der Absender sicher bemerken würde. Die Sicherung der Übertragung ist durch die Prüfung der wieder eintreffenden Meldung gegeben und durch die Tatsache, daß keine weitere Meldung ausgegeben werden kann, bevor die vorige unverändert wieder empfangen wurde.

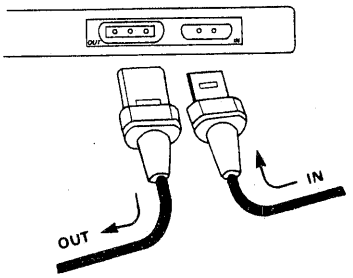
Das Bemerken von Geräte-Meldungen

Durch das Setzen eines bestimmten Rahmen-Bits kann jedes Gerät Aufmersamkeit erregen. Der aktive Controller erfährt das aber nur dann unmittelbar, wenn das meldende Gerät in der Schleife hinter dem Talker aber vor dem Controller liegt. Dann kann der Controller bei entsprechender Vorbereitung, die Meldung sofort erkennen. Liegt das Gerät dagegen hinter dem Controller aber vor dem Talker, dann wird das Rahmen-Bit erst gesetzt, nachdem die Meldung den Controller passiert hat. Hierbei ist es der Talker, der auf jeden Fall die Änderung bemerkt und darauf mit einer Störungsmeldung reagiert, die den Controller auf jeden Fall aufmerksam macht. Die Feststellung, welches Gerät die Meldung auslöste und welche Ursache zur Meldung Anlaß gab geschieht durch den Controller mit einer mehr oder weniger aufwendigen Abfrage. Außerdem hat der aktive Controller noch die Möglichkeit, "Rundschreiben ohne Inhalt" über die Schleife zu schicken, die dann von den Geräten entsprechend markiert werden können.

Geräte-Adressen

Um die in der Schleife zusammengefaßten Geräte unterscheiden zu können, muß jedes Gerät eine Adresse (eine Zahl zwischen 0 und 30) haben. Die Vergabe der Adressen geschieht grundsätzlich durch den aktiven Controller. Beim Einschalten wird immer der System-Controller automatisch aktiver Controller. Der aktive Controller beansprucht stets die Adresse "0" für sich und teilt dies dem benachbarten ersten Gerät in der Schleife mit. Das nimmt nun die Adresse "1" an und gibt dies dem nächsten Gerät bekannt und so fort...

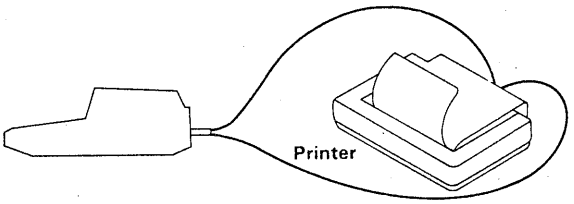
Da das letzte Gerät der Schleife den aktiven Controller anspricht, erkennt dieser nun, daß die Adressierung abgeschlossen ist. Die mitgeteilte Adresse des letzten Gerätes informiert ihn, wieviel Geräte er in der Schleife ansprechen kann. Jedes Gerät speichert die eigene Adresse intern bis auf weiteres. Die Flußrichtung (und damit auch die Zählrichtung der Adressen) läßt sich aus der Form der Kabelstecker erkennen: Das Prinzip des "Trichters" sorgt dafür, daß keine Information "daneben geht"! Der dünne Stecker gibt die Information an den dickeren Stecker! An den Geräten sind die Bezeichnungen OUT und IN ein weiterer Hinweis.



IL-Steck-Verbindungen

Drucker-Operationen

Das folgende Bild zeigt einen HP-Tischrechner der Serie 80 (mit I/O-ROM und/oder Printer/Plotter-ROM), der über das HP-IL-Interface und die Schleife einen passenden Drucker steuert (z.B. HP 82162A). Hier ist der Rechner der System-Controller und wird beim Einschalten automatisch aktiver Controller.



Der Drucker in der Schleife kann nun zum System-Drucker bestimmt werden. Wenn der Drucker als erstes Gerät in der Schleife liegt und das IL-Interface den Auswahl-Code "9" hat, heißt diese Anweisung:

```
PRINTER IS 901
```

Auswahl-Code ↗ ↖ Geräte-Adresse

Damit gehen alle Druckanweisungen über die Schleife an den Drucker. Wenn nun dieser Drucker eine andere Zeilenlänge als 32 Zeichen hat, können Sie die Ausgabe an dessen Eigenschaften anpassen: Das Printer- bzw. Plotter/Printer-ROM erlaubt die Anpassung der Zeichenzahl pro Zeile an die Möglichkeiten des Druckers.

Der Ersatzwert für die Zeilenlänge ist 32, es ist aber möglich, 1 bis 140 Zeichen pro Zeile vorzuschreiben. Das folgende Beispiel setzt voraus, daß der Drucker in der Schleife an zweiter Stelle liegt und 80 Zeichen pro Zeile drucken soll:

```
PRINTER IS 902, 80
```

Bei Benutzung des I/O-ROM's läßt sich die Zeilenlänge mit den OUTPUT USING- oder PRINT USING-Anweisungen beeinflussen. Die folgenden Anweisungen sind ein Beispiel für einen Druck mit 80 Zeichen pro Zeile, sie lassen sich leicht für andere Geräte und Zeilenlängen abwandeln:

```
10 DIM A$[80]
20 A$="Vergrößern Sie Ihr Zelt und dehnen Sie Ihren
Lebensraum aus!"
30 OUTPUT 901 USING "80A" ; A$
40 END
```

Wenn die bisher gegebenen Informationen ausreichen, um Ihre Geräte in Funktion zu setzen, können Sie die Lektüre des IL-Teils dieses Buches abbrechen! Nur wenn Sie die vielen Möglichkeiten kennenlernen wollen, die Ihnen das IL-System zur Lösung auch größerer Probleme bietet, müssen Sie weiterlesen!

Hinweis:
Das Plotter/Printer-ROM erleichtert die Benutzung von Plottern und Druckern in der IL-Schleife, darüber hinausgehende Operationen erfordern aber das I/O-ROM, mit dem sich aber auch Plotter und Drucker ansteuern lassen! Von nun an wird angenommen, daß dem Rechner ein I/O-ROM zur Verfügung steht.

Beenden von I/O-Operationen

Für den Fall, daß Störungen im Ablauf eines Programms auftreten, sollten Sie wissen, wie man eine nicht korrekt ablaufende I/O-Operation unterbrechen kann: Falls die Tasten noch reagieren, kann man eine der folgenden Anweisungen ausführen lassen (Eingeben und danach END LINE):

```
ABORTIO 9   oder:   HALT 9
```

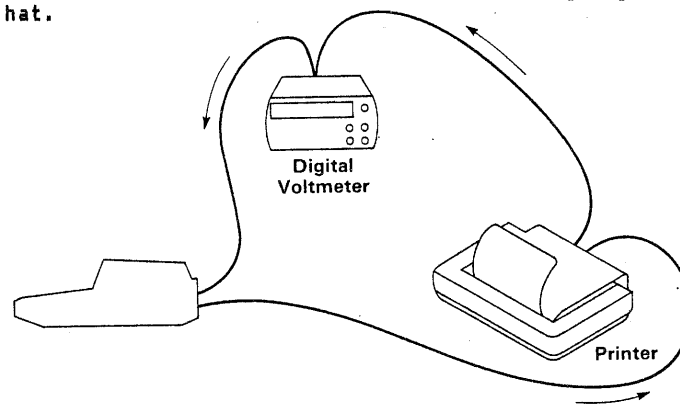
Wenn das Tastenfeld keinerlei Reaktion mehr zeigt, läßt sich meistens durch mehrfaches Drücken der RESET-Taste der Rechner wieder unter Kontrolle bringen. Wenn der Rechner wieder normal reagiert, dann ist auch das HP-IL-Interface wieder auf seinen Einschaltzustand zurückgesetzt. Wenn allerdings keine dieser Maßnahmen zum Erfolg führt, hilft nur noch der "drittbeste" Weg: Der Rechner muß ausgeschaltet werden, wobei aber Programm und Daten verloren gehen. Es kann zusätzlich notwendig werden, auch die übrigen Geräte der Schleife vorübergehend auszuschalten, um einen definierten Zustand des Systems zu erzwingen.

Senden und Empfangen über die Schleife

Es gibt mehrere Verfahren, um Daten über die Schleife zu übertragen: Wenn das HP-IL-Interface im Rechner aktiver Controller ist und selbst Daten aufnehmen will, reicht es, mit den ENTER- bzw. TRANSFER(ein)-Anweisungen den gewünschten Talker zu adressieren. Entsprechend enthalten die OUTPUT- bzw. TRANSFER(aus)-Anweisungen die Listener-Adresse(n), wenn das IL-Interface Daten ausgeben will.

Als Beispiel für diese Technik soll ein System betrachtet werden, das außer dem Rechner noch ein Digital-Voltmeter (als Talker) und einen Drucker (als Listener) enthält. Wenn Ihnen die dabei verwendeten I/O-Anweisungen nicht geläufig sind, sollten Sie sich im I/O-ROM-Teil dieses Buches informieren.

Für ein Digitalvoltmeter (z.B. ein Gerät HP 3468A) ist es typisch, daß es vorwiegend als Talker arbeitet. Zuvor muß es aber durch den aktiven Controller erst auf Fernbedienung (Zeile 10) und dann auf die passende Betriebsart (Zeile 20) eingestellt werden. Danach kann es (als Talker) den Meßwert an den Rechner geben, der ihn als Listener aufnimmt und anschließend (mit Kommentar) anzeigt. Hierbei wird vorausgesetzt, daß das Voltmeter in der Schleife hinter dem Drucker eingefügt ist und deshalb die Adresse "2" hat.



Das nachstehende Programm kann nach Start mit 'RUN' nur ablaufen, wenn alle Geräte der Schleife eingeschaltet sind.

```
10 REMOTE 9
20 OUTPUT 902 ; "F1RAT1" ! Setzt Voltmeter auf DC-Messung
30 ENTER 902 ; A
40 DISP "Spannung =";A;" Volt"
50 END
```

Mit einer einzigen Anweisung, die den Meßwert an den in der Schleife vorhandenen Drucker (als Listener) weitergibt, wird die Dokumentation ausgeführt:

```
35 OUTPUT 901 ; A
```

Eine automatische Wiederholung des Vorganges läßt sich mit einer Programmschleife erreichen, die sich so in das bestehende Programm einfügen läßt:

```
10 REMOTE 9
20 OUTPUT 902 ; "F1RAT1" ! Setzt Voltmeter auf DC-Messung
25 FOR I=1 TO 10000
30 ENTER 902 ; A
35 OUTPUT 901 ; A
40 DISP "Spannung =";A;" Volt"
45 NEXT I
50 END
```

Voltmeter und Drucker werden jetzt 10 000 Meßwerte aufnehmen und drucken, und der Rechner wird in dieser Zeit als Zwischenstation für die Daten benutzt und dadurch leider für andere Arbeiten blockiert. Ein anderes Verfahren, bei dem der Rechner nur bei der Einleitung der Datenübertragung beschäftigt ist, sollen Sie als nächstes kennenlernen. Dieses Verfahren benutzt die sehr flexible SEND-Anweisung, um fast den gleichen Effekt wie das vorige Programmbeispiel zu erzielen:

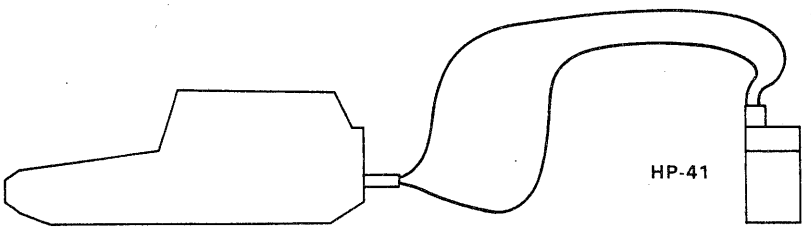
```
10 REMOTE 9
20 OUTPUT 902 ; "F1RAT1"
30 SEND 9 ; LISTEN 1 TALK 2
40 RESUME 9
50 END
```

Starten Sie nun das Programm mit 'RUN'. Der Rechner bereitet nun mit den Zeilen 10 bis 30 die Geräte vor und leitet mit der RESUME-Anweisung die Übertragung ein. Die in der Schleife umlaufenden Daten werden jetzt vom IL-Interface nur weitergereicht und gelangen garnicht erst in den Rechner. Wenn Sie kurz danach die PAUSE-Taste betätigen, hat dies auf die Datenübertragung keinen Einfluß, ja Sie dürfen das Programm mit 'SCRATCH' auch löschen, ohne daß die Operation beendet wird! Nun können Sie den Rechner im Programm- oder Rechen-Modus für andere Dinge benutzen, die Datenübertragung läuft trotzdem so lange weiter, bis Sie das HP-IL-Interface erneut ansprechen. Zum Abbruch der übertragung führt die folgende Anweisung:

```
HALT 9
```

Betrieb ohne System-Controller-Status

Bei manchen Geräte-Zusammenstellungen darf der Rechner der Serie 80 (von nun an als 'HP-80' bezeichnet) nicht aktiver Controller sein. Manchmal ist es sogar notwendig, dem HP-80 auch die Vorrechte als System-Controller zu nehmen. So z.B. bei Anschluß eines Taschenrechners HP-41, dessen IL-Modul (HP82160A) selbst System-Controller sein muß. Da in der Schleife nur ein einziges Gerät System-Controller sein darf, ist es bei der im nächsten Bild gezeigten Anordnung unbedingt nötig, das IL-Interface im HP-80 intern umzuschalten. Im IL-Abschnitt 2 steht, was dafür zu tun ist!



Wir können dann HP-80 und HP-41 zur dargestellten Schleife zusammenschalten. Der HP-41 (hier als System-Controller) wird beim Einschalten automatisch auch aktiver Controller. Das IL-Interface im HP-80 wird dagegen beim Einschalten nicht automatisch aktiver Listener, sondern benötigt ein Programm, um die eintreffenden Daten aufnehmen zu können:

```
10 CLEAR
20 DISP "Ich warte auf Daten!"
30 ENTER 9 ; A$
40 DISP A$
50 END
```

Wenn Sie jetzt die Worte "TOTAL KOMPATIBEL" in das ALPHA-Register des HP-41 setzen, können Sie diese Daten mit der OUTA-Funktion des HP-41 über die Schleife in den HP-80 bringen, der sie in A$ ablegt. Wenn die übertragene Zeichenkette über 18 Zeichen enthält, muß A$ mit DIM passend definiert werden, weil sonst der HP-80 nur die ersten 18 Zeichen der Kette aufnehmen kann.

Um Daten vom HP-80 zum HP-41 zu übertragen, benutzen wir ein anderes Programm:

```
10 DIM Z$[24]
20 Z$="HALLO, HIER DIE SERIE 80"
30 OUTPUT 9 ; Z$
40 END
```

Wenn Sie jetzt am HP-41 AUTOID und INA ausführen, wird der Text im ALPHA-Register angezeigt. Beachten Sie bitte, daß ein Gerät, das nicht aktiver Controller ist, nur den Auswahl-Code nennt. Die Benutzung von Geräte-Adressen bleibt dem aktiven Controller vorbehalten!

Programm-Listing für den HP-41

Wenn zum HP-80 ein System-Drucker gehört oder wenn dieser Rechner einen eingebauten Drucker aufweist (HP-85), können Sie nun auch die im HP-41 gespeicherten Programme von diesem Drucker auflisten lassen. Hierzu braucht der HP-80 das folgende Programm:

```
 10 CLEAR
 20 DISP "Ich warte!"
 30 ENTER 9 ; A$ ! Nimmt Daten aus der Schleife auf!
 40 ! Anzeige- und Druck-Anweisung:
 50 PRINT A$
 60 DISP A$
 70 ! Prüfung, ob Programmende erreicht:
 80 IF POS(A$,"END") THEN 110
 90 GOTO 30
100 ! "Fertig!"-Anzeige, wenn 'END' erreicht!
110 ENTER 9 USING "#,"B# ; A
120 DISP "Fertig!"
130 END
```

Nach Start dieses Programms benötigt der HP-41 eine 1 im X-Register, dann sind am HP-41 die Funktionen AUTOIO, MANIO und SELECT auszuführen und schließlich (in gewohnter Weise) PRP. Der (eingebaute) Systemdrucker des HP-80 gibt dann das erwartete Listing aus. Bei dieser Betriebsart muß der Schiebeschalter am IL-Interface des HP-41 auf 'Printer Enable' stehen. Außerdem sollte man auf (geringfügige) Unterschiede im Zeichensatz der Drucker gefaßt sein!

Übertragung von Daten aus dem HP-41

Wir nehmen einmal an, daß Sie unterwegs in Ihren HP-41 zehn wichtige Meßdaten in die Register R01 bis R10 eingegeben haben, die Sie nach der Rückkehr in das Büro an den HP-80 zur Speicherung übergeben wollen. Mit dem folgenden Programm für den HP-41 können Sie diese Daten der IL-Schleife übergeben:

```
01♦LBL "XFER"
02 CF 29          Schaltet das Komma-Flag aus
03 1.010          Bereitet ISG-Zählschleife vor
04 STO 00
05 1              Wählt den HP-80 als Ziel für die Daten
06 SELECT
07♦LBL 00
08 CLA            Löscht ALPHA-Register
09 ARCL IND 00    Holt Wert ins ALPHA-Register
10 OUTA           Sendet Wert an den HP-80
11 ISG 00         Erhöht Zählschleife und prüft die Abbruchbedingung
12 GTO 00
13 END
```

Natürlich muß auch hier der HP-80 für die Ereignisse gerüstet sein: Es muß zuerst nach den üblichen Regeln ein Daten-File (Namen z.B. "FILE") auf der Kassette oder Diskette angelegt werden:

```
CREATE "FILE",10,10
```

Der HP-80 benötigt folgendes Programm:

```
 10 OPTION BASE 1
 20 DIM X(10) ! Variablenfeld fuer 10 Werte!
 30 ! "FILE" wird eroeffnet:
 40 ASSIGN# 1 TO "FILE"
 50 FOR I=1 TO 10
 60 ! Die einzelnen Werte aus dem HP-41 werden empfangen:
 70 ENTER 9 ; X
 80 ! Die Werte werden den Records in "FILE" zugeordnet:
 90 PRINT# 1 ; I,X
100 DISP "Reg.-Inhalt von";I;" ist";X ! Anzeige!
110 NEXT I
120 ASSIGN# 1 TO *
130 END
```

Starten Sie erst das Programm im HP-80 und danach das im HP-41. Sie sollten dann die 10 Werte mit ihren Registernummern auf dem Bildschirm sehen. Die Aufzeichnung "FILE" enthält dann auch die 10 Werte, auf die Sie jederzeit wieder zurückgreifen können, z.B. mit diesem Programm:

```
 10 ASSIG# 1 TO "FILE" ! Oeffnet File "FILE"
 20 FOR I=1 TO 10 ! Es sollen 10 Werte gelesen werden!
 30 READ# 1 ; I,X ! Liest Wert aus richtigem Record
 40 ! Jeder Wert wird mit Reg.-Nummer gedruckt:
 50 PRINT "Reg.-Inhalt von";I;" ist";X
 60 NEXT I
 70 ASSIGN# 1 TO *
 80 END ! Programm endet nach dem 10. Wert
```

Diese Beispiele zeigen nur die grundsätzlichen Verfahren, die beim HP-IL-System zum Senden und Empfangen von Informationen verwendet werden können. Schon die Benutzung dieser einfachsten Techniken kann eine wesentliche Erweiterung der Anwendungsmöglichkeiten Ihres Tischrechners bedeuten.

Vorbemerkung des Herausgebers zu den IL-Abschnitten 4 und 5

Bei der Bearbeitung der verschiedenen Handbücher, die zu dieser Zusammenfassung führte, wurde es immer klarer, daß das IL-Interface in vieler Hinsicht eine Weiterentwicklung des schon langbewährten (und als IEEE-488 bzw. IEC-625 genormten) IB-Systems ist. Diese Weiterentwicklung zielte auf ein Verfahren, bei dem der materielle Aufwand für die Verbindungs-Kabel deutlich verringert werden sollte, obwohl die ohne Sondergeräte zu bewältigenden Entfernungen wesentlich größer angesetzt wurden. Damit war entschieden, daß nur eine serielle Übertragung verwendet werden konnte.

Um möglichst viel der bereits bewährten Systematik des IB-Systems übernehmen zu können, mußte auch bei der seriellen Übertragung der einzelnen Bits ein Weg gefunden werden, Daten-Bytes von Befehlen zu unterscheiden. Außerdem mußte für alle in der Schleife vorhandenen Geräte auch während einer Datenübermittlung die Möglichkeit zu einer Meldung an den aktiven Controller bestehen.

Die gewählte Lösung benutzt dazu drei weitere Bits, die dem jeweils übertragenen Byte vorangestellt werden und diesem verschiedene Bedeutung geben. Man kann diese drei Bits (mit ihren 8 Zuständen) in mancher Hinsicht mit den acht Steuerleitungen vergleichen, die das IB-System außer den 8 Datenleitungen nötig hat.

Auch im Hinblick auf den Anwender ist das IL-System eine Weiterentwicklung zu bequemerer Benutzung. Es ist meist nicht mehr nötig, sich um die Adressen der Peripherie-Geräte zu kümmern, da deren Erteilung und Verwaltung durch den jeweiligen aktiven Controller erfolgt. Damit ist in vielen Fällen auch das Erkennen der Eigenschaften der Peripherie-Geräte verbunden. Mit anderen Worten: Der Controller kennt ohne Zutun des Anwenders die Adressen der Drucker, Massenspeicher und der verschiedenen Meßgeräte.

Auf den folgenden Seiten sind die Angaben zusammengetragen, die über das IL-Interface zum Rechner der Serie 80 vorlagen. Diese Zusammenstellung muß lückenhaft bleiben, weil das IL-System eine große Zahl von Befehlen kennt, die nicht bei allen zu Controller-Aktivitäten befähigten Geräten mit den gleichen Kennkürzeln bezeichnet werden. Außerdem ist das IL-System, was seine Anwendungsbreite betrifft, noch sehr jung....

Wer sich eingehend mit dem IL-System beschäftigen will, um dieses in jeder Beziehung vollkommen und optimal zu nutzen, kommt nicht ohne die grundlegenden Veröffentlichungen aus, die hier angegeben sind:

The HP-IL Interface Specifikation
HP-Bestellnummer 82166-90017

The HP-IL Integrated Circuit User's Manual
HP-Bestellnummer 82166-90016

Gerry Kane, Steve Harper, David Ushijima
THE HP-IL SYSTEM
An Introductory Guide to the Hewlett-Packard Interface Loop
OSBORNE/McGraw-Hill
Berkeley, California

bzw.

Gerry Kane, Steve Harper, David Ushijima
Das HP-IL-System
Einführung in die Hewlett-Packard Interface-Schleife
übersetzt von Ursula Wilk u. Otmar Frey
McGraw-Hill Book Company GmbH
Hamburg

HP-IL-Operationen mit dem I/O-ROM

Einführung

Das HP-IL-System ist zum Betrieb weniger Geräte (Controller und 1 Peripherie-Gerät) genau so gut geeignet, wie für komplexe Steuerfunktionen in Systemen von beträchtlichem Umfang (mehrere Controller mit einer Vielzahl von Geräten). Dazu ist in den meisten Fällen das I/O-ROM nötig, weshalb von nun an angenommen wird, daß dieses ROM dem System zur Verfügung steht.

Die folgenden Kapitel sind auf Benutzer zugeschnitten, denen die Grundlagen des HP-IL-Systems bekannt sind, die aber Informationen über die praktische Anwendung brauchen. Deshalb stehen hier keine langen Aufzählungen der Anweisungen mit Syntax, sondern Beispiele, die die richtige Reihenfolge der notwendigen Operationen der System-Funktionen vom HP-IL verdeutlichen. Benutzen Sie die folgenden Kapitel zum Entwurf Ihrer Programme und schauen Sie bei Bedarf in die Syntax-übersicht im Anhang A dieses Buches, um Ihre Programmzeilen exakt zu formulieren.

Einschalten und Prüfen

Einzelheiten über den Anschluß der Geräte finden Sie im IL-Abschnitt 2, weshalb wir uns hier auf das Wichtigste beschränken können.

Wenn das Interface eingesetzt ist und die einzelnen Geräte mit den HP-IL-Kabeln eine geschlossene Schleife bilden, sollte man sich mit einer kurzen Routine über Lage und Eigenschaften der einzelnen Komponenten Gewißheit verschaffen.

Da Sie sich im HP-IL-System nicht um die Adressen der Peripherie-Geräte zu kümmern brauchen, läuft das nachstehende Programm praktisch automatisch ab:

```
  10 ! **********************
  20 ! @ SCHLEIFEN-STATUS @
  30 ! **********************
  40 CLEAR @ DISP "Auswahl-Code vom IL-Interface";
  50 INPUT S ! Interface-Auswahl-Code
  60 STATUS S,7 ; S7
  70 CLEAR @ DISP "Adresse:   Geraete-Typ:   Code:"
  80 FOR I=1 TO S7 ! Wieviel Geraete?
  90 A=0 @ D$=""
 100 SEND S ; UNL MLA TALK I
 110 ON ERROR GOTO 140
 120 SEND S ; CMD 254 ! Abfrage Zusatz-ID:
 130 ENTER S USING "B" ; A
 140 ON ERROR GOTO 170
 150 SEND S ; CMD 255 ! Abfrage der Geraete-Bezeichnung:
 160 ENTER S ; D$
 170 OFF ERROR
 180 IF A=0 AND D$="" THEN DISP USING 320 ;
     "Geraet ",I," gibt keine Meldung!" @ GOTO 300
 190 IF D$#"" THEN 290
 200 ON A\16+1 GOTO 210,220,230,240,250,260,270,280,280,280
     280,280,280,280,280,280
 210 D$="Controller?" @ GOTO 290
 220 D$="Speicher?" @ GOTO 290
 230 D$="Drucker?" @ GOTO 290
 240 D$="Display?" @ GOTO 290
 250 D$="Interface?" @ GOTO 290
 260 D$="Instrument?" @ GOTO 290
 270 D$="Graphik?" @ GOTO 290
 280 D$="???????????"
```

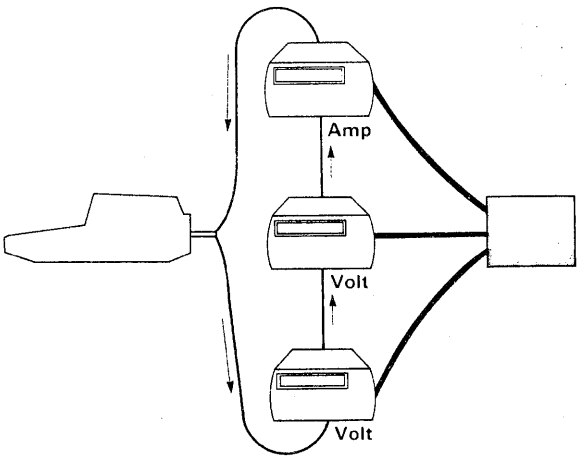
```
290 DISP USING 330 ; "Geraet ",I,D$,"SAI: ",A
300 NEXT I
310 SEND S ; UNL UNT
320 IMAGE 7A,2Z,21A
330 IMAGE 7A,2Z,14A,4A,3Z
340 END
```

Das Programm erfragt in Zeile 40/50 den Auswahl-Code des IL-Interfaces und ermittelt in Zeile 80 die Zahl der vorhandenen Geräte. In einer Programm-Schleife wird von jedem Gerät des IL-Systems die Typ-Kennung (SAI) und/oder die HP-Nummer abgefragt. Die Antwort wird dann angezeigt, wobei aus der Typ-Kennung zuvor der Geräte-Typ bestimmt wird. Da für das HP-IL-System bisher nur eine sehr 'lockere' Normung existiert, werden die errechneten 'Erkenntnisse' durch Fragezeichen als Vermutungen ausgewiesen.

Mit den ON ERROR- und OFF ERROR-Anweisungen wird erreicht, daß auch beim Ausbleiben der Antworten das Programm bis zum Ende weiterläuft.

Steuerung eines Meßsystems

In der folgenden Skizze sollen drei Multimeter (HP 3468 A in IL-Version) in einer Schaltung gleichzeitig einen Strom und zwei Spannungen messen:



Dieses Programm sorgt für das Umschalten der drei Multimeter auf Ferneinstellung, Sperren der Einstellorgane auf den Frontplatten und Wahl der Meßbereiche und Meßverfahren (u.a. den Triggerbetrieb):

```
 10 REMOTE 901,902,903
 20 LOCAL LOCKOUT 9
 30 OUTPUT 901 ; "F1RAT2" ! Legt Spgs.-Bereich vom MM 1 fest
 40 OUTPUT 902 ; "F1RAT2" ! Legt Spgs.-Bereich vom MM 2 fest
 50 OUTPUT 903 ; "F3RAT2" ! Legt Strom-Bereich vom MM 3 fest
 60 TRIGGER 901,902,903 ! Auslösen der Messungen
 70 ENTER 901 ; V1 ! Holt Spgs.-Wert vom MM1
 80 ENTER 902 ; V2 ! Holt Spgs.-Wert vom MM2
 90 ENTER 903 ; A !  Holt Strom-Wert vom MM3
100 DISP "V1=";V1,"V2=";V2,"A =";A ! Ergebnis-Anzeige
110 WAIT 1000 @ GOTO 60
120 END
```

Nun werden zeitlich zusammengehörende Spannungs- und Stromwerte in Abständen von etwa 1 Sekunde ermittelt, bis das Programm in irgendeiner Weise angehalten wird. Selbstverständlich läßt sich die einfache Pause auch durch ein anderes Kriterium ersetzen, das die Aufnahme eines weiteren Werte-Tripels auslösen soll.

Störungsmeldungen

Wenn ein Gerät der Schleife eine Störung gemeldet hat, weil es nicht mehr weiterarbeiten kann (dem Drucker ging z.B. das Papier aus!), kann der aktive Controller das gestörte Gerät ermitteln. Der folgende Programm-Ausschnitt veranlaßt eine serielle Abfrage aller Geräte, sobald eine Störungsmeldung eingetroffen ist:

```
1000 S=9 ! Interface-Auswahl-Code
1010 STATUS S,7 ; S7 ! Wieviel Geraete?
1020 FOR I=1 TO S7
1030 ON ERROR GOTO 1060
1040 Q=SPOLL(S*100+I) ! Fragt jedes Geraet ab
1050 IF BIT(Q,6) THEN 1100 ! Brauchte dieses Geraet Hilfe?
1060 NEXT I
1070 RETURN
1100 DISP "Hilferuf von Adresse ";I
1110 PAUSE
```

Der Rechner kann beim Auftreten der Störung diese Routine nur erreichen, wenn bereits vorher eine Verzweigung zugelassen und passend definiert wurde. Das kann am Anfang des Programms mit den folgenden Anweisungen geschehen:

```
100 ENABLE INTR 9 ; 8 ! SRQ-Interrupt im HP-IL
110 ON INTR 9 GOSUB 1000
```

Im I/O-ROM-Abschnitt 9 ist dieses Gebiet ausführlich behandelt. Einige nützliche Hinweise über diese Technik finden Sie auch ab Seite IL-29.

Einfache Maßnahmen gegen Fehler

Bei Störungen eines laufenden Programms ist es praktisch, wenn durch Hinweise auf die Ursachen die Fehlersuche vereinfacht wird. Dazu sind Routinen nötig, die von den Ursachen selbst in Gang gesetzt werden. Eine häufige Störungs-Ursache ist die Unterbrechung der Schleife (z.B. durch unbemerktes Abschalten eines Gerätes) oder das Unterbrechen der Datenübermittlung. Das folgende Beispiel löst in beiden Fällen passende Anzeigen aus:

```
  10 ON ERROR GOTO 200
  20 SET TIMEOUT 9 ; 5000
  30 ON TIMEOUT 9 GOTO 300
  40 PRINTER IS 901 !     ! Das hier ist nur
  50 PRINT "Das ist es!" ! ein Beispiel für das
  60 DISP "Er druckt!" ! ! Programm, das hier
  70 GOTO 50 !           ! stehen könnte!
 200 ! Falle für Fehler 118:
 210 IF ERRN=118 AND ERRSC=9 THEN DISP
     "Fehler 118 in Zeile ";ERRL @ GOTO 330
 220 ! Platz für
 230 ! weitere
 240 ! Fehler-
 250 ! Routinen
 260 GOTO 40
 300 ! Zeitfehler-Routine
 310 DISP "Zeit-Fehler im HP-IL!"
 330 RESET 9 @ BEEP @ GOTO 40
 340 END
```

Entsprechende Routinen sind in allen Programmen empfehlenswert, die I/O-Operationen enthalten, weil dadurch das Auftreten von Fehlern keine unüberblickbaren Probleme schafft.

Während diese Beschreibung bisher nur einfache Beispiele für die Anwendungen des IL-Interfaces auf bestimmte Geräte-Zusammenstellungen brachte, werden nun die dabei verwendeten Anweisungen einzeln ausführlich behandelt:

Einfache Ausgabe-Operationen

Um Daten vom Rechner an ein Gerät des HP-IL-Systems zu übergeben, wird grundsätzlich die OUTPUT-Anweisung benutzt. Es ist aber auch möglich mit der PRINT-Anweisung Daten an ein Gerät des Systems zu geben. Es muß dann aber das anzusprechende Gerät vorher mit einer PRINTER IS-Anweisung als System-Drucker definiert werden. Diese PRINTER IS-Anweisung kann am Anfang eines BASIC-Programms stehen, wodurch alle PRINT-Anweisungen, die danach folgen, zum angegebenen Gerät gelangen, bis ein anderes Gerät per Programm die Rolle des System-Druckers zugewiesen bekommt. Wenn nur ein einziges Gerät Daten erhalten muß, ist das recht bequem, wie das folgende Beispiel zeigt (Der HP-IL-Drucker hat hier Adresse 01!):

```
20 PRINTER IS 901
30 FOR I=0 TO 1 STEP .01
40 PRINT SIN(I)
50 NEXT I
60 END
```

Die Methode wird aber um so mühsamer, je mehr Geräte anzusprechen sind. Die Wahl der Betriebsart von 2 Meßgeräten sieht dann so aus:

```
10 PRINTER IS 907
20 ! Adressiert Geraet 07 (DVM)
30 PRINT "Programmier-Code"
40 PRINTER IS 905
50 ! Adressiert Geraet 05 (Frequenzmesser)
60 PRINT "Programmier-Code"
70 END
```

Diese Methode braucht (ohne die Erklärungen!) vier Anweisungen. Mit zwei OUTPUT-Anweisungen hat man den gleichen Erfolg:

```
10 OUTPUT 907 ; "Programmier-Code"
20 OUTPUT 905 ; "Programmier-Code"
```

Die OUTPUT-Anweisung läßt sich auch formatiert ins Programm setzen, wie das ausführlich im I/O-ROM-Abschnitt 3 erklärt wurde. Die HP-IL-Version der Anweisung sieht dann so aus:

```
OUTPUT 907 USING "nA" ; "Programmier-Code"
```

Eine für das HP-IL-System formulierte Ausgabe-Anweisung unterscheidet sich also kaum von den für HP-IB nötigen Anweisungen und von den bei anderen Systeme üblichen Formulierungen nur durch die Adresse, die darüber informiert, wer die Daten bekommen soll. Einzelheiten über die OUTPUT-Anweisung (und deren Zusammenhang mit der TRANSFER-Anweisung) stehen in den I/O-Abschnitten 2 und 8 dieses Buches.

Einfache Eingabe-Operationen

Mit der ENTER-Anweisung lassen sich am einfachsten Daten von einem Gerät des HP-IL-Systems in den Rechner bringen. Die Daten von einem Meßgerät werden durch die folgende Anweisung vom Rechner angenommen:

```
ENTER 908 ; A$
```

Die Anweisung wird durch das Zeilenvorschub-Symbol beendet, das das Meßgerät als letztes Zeichen ausgibt. Manche für das HP-IL-System passende Geräte kennzeichnen das letzte Zeichen auch durch Ausgabe mit einem 'END'-Rahmen (siehe Seite IL-55). Für diesen Fall muß die ENTER-Anweisung (z.B. für das Gerät 06) so aussehen:

```
ENTER 906 USING "%,K" ; A$
```

Der Spezifikator "%" sorgt dann dafür, daß auch das END-Byte als Ende der Datenübertragung gewertet wird. Wenn nun aber ein Zeilenvorschub-Symbol mitten in den Daten enthalten ist, dann schließt dieses mit Vorrang die (noch unvollständige) Datenzuweisung für die String-Variable A$ ab. Diese Unsicherheit wird mit folgender Anweisung beseitigt, weil diese Form nur END-Bytes als Abschluß anerkennt und auch die im String eingestreuten Zeilenvorschub-Symbole (wie die übrigen Zeichen) der String-Variablen zuweist:

```
ENTER 906 USING "#%,K"; A$
```

Näheres über die Anwendung der ENTER-Anweisung steht in den I/O-ROM-Abschnitten 2 und 3 dieses Handbuches, wo die ENTER-Anweisung in allen Einzelheiten besprochen wird. Weiter entwickelte Techniken, die flexibler sind, sich der TRANSFER-Anweisung bedienen oder Daten in den Rechner holen, auch wenn er nicht die Controller-Funktion hat, sind im folgenden Kapitel beschrieben.

Erweiterte I/O-Operationen

Einführung

Wenn Sie diesen Abschnitt beginnen, dann tun Sie es wahrscheinlich deshalb, weil Sie ein Problem haben, das mit den einfachen OUTPUT- oder ENTER-Anweisungen nicht zu lösen ist. Dieser Abschnitt beschäftigt sich mit vier verschiedenen Problemen der Datenübertragung:

1. Datenübertragung mit einem Rechner ohne Controller-Funktion.
2. ENTER- und OUTPUT-Anweisungen, die mehrere Listener betreffen.
3. Vom Rechner ausgelöste Übertragung von Daten, die ihn selbst aber nicht betreffen.
4. Benutzer-spezifische TRANSFER-Verfahren, unter Einschluß von Interrupt und Fast-Handshake.

Selbst-Adressierung ohne Controller-Funktion

Diese Eingabe-/Ausgabe-Technik wird verwendet, wenn der Rechner (auch als System-Controller) die Controller-Funktion an ein anderes Gerät abgegeben hat oder wenn der Rechner (ohne System-Controller zu sein) auch von keinem anderen Gerät zum aktiven Controller bestimmt wurde.

Der Rechner (und auch jedes andere Gerät im System) muß mit der Ausgabe bzw. Aufnahme von Daten warten, bis er als Talker bzw. Listener adressiert ist. Es gibt drei nachfolgend beschriebene Verfahren, um diese Adressierung zu bemerken.

Die erste Methode besteht im einfachen Warten. Das ist bequem, blockiert aber den Rechner während der Wartezeit für jede andere Tätigkeit und ist deshalb nur für kleine Systeme anzuraten, bei denen die Wartezeiten kurz sind. OUTPUT-, ENTER- oder TRANSFER-Anweisungen enthalten dabei nur den Auswahl-Code aber keine Geräte-Adresse. Die Anweisungen sind also recht einfach aufgebaut und laufen automatisch ab, sobald die passende Adressierung erfolgt.

OUTPUT 9 ; A$... oder TRANSFER A$ TO 9

Es wird die Adressierung als Talker abgewartet, dann werden die in der Ausgabe-Liste definierten Daten oder der Inhalt des I/O-Buffers ausgegeben.

ENTER 9 ; A$... oder TRANSFER 9 TO A$

Es wird die Adressierung als Listener abgewartet, dann werden die angebotenen Daten den Variablen der Eingabe-Liste oder dem I/O-Buffer zugewiesen.

Bei der zweiten Methode prüft der Rechner in gewissen Abständen den Interface-Status, um dadurch eine Adressierung zu bemerken. Im Status-Register SR5 wird das Bit 4 gesetzt, wenn das Interface als Talker adressiert wurde. Entsprechend wird bei der Adressierung zum Listener im gleichen Register das Bit 6 gesetzt. Bis zur Adressierung kann der Rechner zwischen den Prüfungen andere Aufgaben erledigen. Sobald eine Adressierung erkannt wird, führt der Rechner die dann mögliche Ausgabe oder Eingabe aus und kehrt dann, unter Beibehaltung der periodischen Abfrage, zu seiner üblichen Arbeit zurück. Im folgenden Programmbeispiel wird der Stand eines Zählers weitergeschaltet, angezeigt und jedesmal geprüft, ob schon eine Adressierung erfolgte. Wenn nicht, wird weitergezählt und angezeigt, wenn ja, wird der Zählerstand an das HP-IL ausgegeben.

```
10 ! Zeilen 50/60 prüfen, ob als Talker adressiert
20 I=0
30 I=I+1
40 DISP I
50 STATUS 9,5 ; S ! SR5 lesen!
60 IF BIT(S,4)=0 THEN GOTO 30 ! "TA"-Bit gesetzt?
70 OUTPUT 7 ; I
80 GOTO 20
90 END
```

Beim nächsten Beispiel wird geprüft, ob inzwischen eine Adressierung als Listener (für eine ENTER-Anweisung) erfolgte. Der Unterschied zur vorigen Anweisung liegt nur darin, daß hier ein anderes Bit geprüft wird.

```
10 STATUS 9,5 ; S! SR5 lesen!
20 IF BIT(S,6)=0 THEN DISP "Kein Listener!"
```

Sehen Sie bitte in den IL-Abschnitt 5 (HP-IL für den Experten) dieses Handbuches, wenn Sie weitere Einzelheiten über die Status-Register erfahren wollen!

Die dritte Methode, auf eine Adressierung zu reagieren, ohne dabei die anderen Tätigkeiten einschränken zu müssen, benutzt die Interrupt-Technik. Dabei löst das Interface im Rechner einen Interrupt aus, sobald eine Adressierung erfolgt. Das hat für den Programmierer den Vorteil, daß er sein Programm nicht kunstvoll um die periodisch durchzuführende Status-Abfrage herumbauen muß. Das Programm kann ganz normal ablaufen, bis eine Adressierung als Talker oder Listener vorliegt. Das folgende Beispiel soll das verdeutlichen:

```
10 ! Bereitet Interrupt und Zeilen-End-Verzweigung
20 ! bei Listener-Adressierung vor:
30 ON INTR 9 GOSUB 100
40 ENABLE INTR 9 ; 64
50 ! Zeilen 30/40 ermöglichen Zeilen-End-Verzweigung
60 I=0
70 I=I+1
80 DISP I
90 GOTO 70
```

```
100 ! Service-Routine für Zeilen-End-Verzweigung
110 ! Status-Register SR1 wird durch Abfrage gelöscht:
120 STATUS 9,1 ; S
 130 ENTER 9 ; A$
140 PRINT "Empfangen wurde: ";A$
150 ! Interrupt-Maske wird erneuert:
160 ENABLE INTR 9 ; 64
170 RETURN
180 END
```

In den Zeilen 30/40 wird festgelegt, daß durch die Adressierung als Listener ein Interrupt ausgelöst werden soll. Zeile 30 enthält dafür die Sprungadresse. In den Zeilen 60 bis 80 steht hier als ganz einfaches Beispiel wieder eine Zählschleife, die der Rechner ausführt, bis der Interrupt auftritt. In Zeile 100 beginnt die Routine, die nach der Adressierung als Listener ablaufen soll. Die ENTER-Anweisung enthält auch hier nur den Auswahl-Code und keine Geräte-Adresse! Das bedeutet etwa: "Ich warte, bis ich als Listener adressiert bin und werde dann die angebotenen Daten aufnehmen". Die Zeile 160 erneuert dann die Interrupt-Genehmigung für den Fall, daß erneut eine Adressierung zum Listener erfolgen sollte.

OUTPUT-Anweisungen lassen sich auf ähnliche Weise schreiben nur, daß dann ein anderer Wert (16) für die ENABLE INTR-Maske einzusetzen ist und die Anweisung OUTPUT 7 ; <Wert> an die Stelle der ENTER-Anweisung im obigen Beispiel tritt. Schauen Sie bitte in den IL-Abschnitt 5 (HP-IL für den Experten), wenn Sie alle Einzelheiten über Status- und Steuer-Register erfahren wollen, über die das Interface in seinem Interrupt-Verhalten beeinflußt werden kann.

Jetzt wollen wir einmal annehmen, daß unser Rechner zwar kein aktiver Controller ist, trotzdem aber Daten sowohl aufnehmen, als auch ausgeben soll. Was ist dann zu tun? Die grundsätzliche Folge der Operationen bleibt die gleiche, wir müssen nur zusätzlich prüfen, welche Art der Adressierung vorliegt. Das folgende Programm enthält diese Prüfung. Zeile 100 ermittelt den Wert des Status-Registers SR1 und in den Zeilen 110 bzw. 120 wird entschieden, ob eine Verzweigung zur Ausgabe oder zur Aufnahme von Daten vorzunehmen ist:

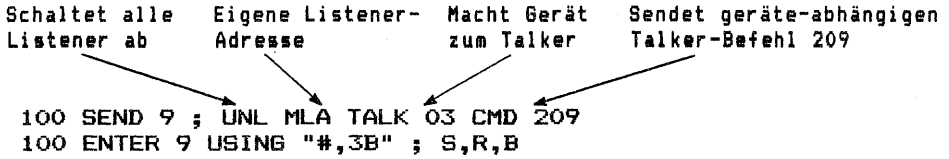
```
 10 ! Interrupt bei Talker- u. Listener-Adressierung
 20 ENABLE INTR 9 ; (64+16)
 30 ON INTR 9 GOSUB 100
 40 I=0
 50 I=I+1
 60 DISP I
 70 GOTO 50
 80 ! Es folgen der Status-Test und
 90 ! die Entscheidung über die Routine!
100 STATUS 9,1 ; S ! Liest und löscht SR1
110 IF BIT(S,4) THEN GOTO 150 ! OUTPUT-Routine
120 IF BIT(S,6) THEN GOTO 180 ! ENTER-Routine!
130 PRINT "Fehler im Status-Register!" @ GOTO 210
150 ! Routine für OUTPUT-Operation!
160 OUTPUT 9 ; I
170 GOTO 210
180 ! Routine für ENTER-Operation!
190 ENTER 9 ; A$
200 PRINT "Empfangen wurde: ";A$
210 ENABLE INTR 9 ; (64 + 16)
220 RETURN
```

Selbstverständlich dürfen bei den drei angegebenen Verfahren die üblichen IMAGE-Anweisungen und auch Festlegungen über die Abschlußbedingungen für Felder und Anweisungen verwendet werden!

Maßgeschneiderte Bus-Sequenzen

Hin und wieder kann es nötig werden, zu einem bestimmten Gerät der Schleife eine besonders zugeschnittene Sequenz zu schicken, die sich von den üblichen Sequenzen stark unterscheidet. Mit der SEND-Anweisung lassen sich solche Spezial-Botschaften übermitteln, durch die man "Sonderabsprachen" mit den Geräten treffen kann. Als Preis für diese Möglichkeit müssen Sie aber jedes Zeichen der Sequenz selber formulieren: automatisch passiert da nichts mehr!

Als Beispiel für eine "maßgeschneiderte" Sequenz soll mit der folgenden Anweisung ein geräte-abhängiger Befehl an ein Kassetten-Laufwerk (HP82161A) gegeben werden, das die Adresse 3 hat und über Auswahl-Code 9 zu erreichen ist. Der geräte-abhängige Befehl 195 veranlaßt das Gerät, die momentane Position des Bandes anzugeben, die mit der folgenden ENTER-Anweisung (S=Spur, R=Record und B=Byte) gelesen wird. Beachten Sie, daß beide Anweisungen keine Geräte-Adresse enthalten dürfen!



Auch andere Operationen, einschließlich des An- und Abschaltens der Parallel-Abfrage (PPC und PPU) lassen sich auf diese Weise durchführen.

Sie können die im IL-Abschnitt 5 (HP-IL für den Experten) befindlichen Tabellen benutzen, um die nötigen Kodierungen für Ihre Pläne zu erfahren.

Gleichzeitige Datenübermittlung an mehrere Listener

Wenn mehrere Geräte die ausgegebenen Daten empfangen sollen, müssen deren Adressen zur Listener-Gruppe zusammengefaßt werden. Das muß bei ENTER- und OUTPUT-Anweisungen auf verschiedene Weise geschehen, wie in diesem Kapitel gezeigt wird.

Sowohl bei OUTPUT- als auch bei Ausgabe-TRANSFER-Anweisungen ist die Möglichkeit vorgesehen, mehrere Listener gleichzeitig anzusprechen, wie das im nachfolgenden Beispiel gezeigt wird: Wir wollen annehmen, daß ein String (B1$) an einen Drucker (Adresse 04) und an einen Rechner der Serie 80, der nicht Controller ist und die Adresse 20 hat, übermittelt werden soll. Als OUTPUT-Anweisung sieht das dann so aus:

```
OUTPUT 904,920 ; B1$
```

Für eine TRANSFER-Ausgabe muß man dagegen so formulieren:

```
TRANSFER B1$ TO 904,920 INTR
```

Für beide Anweisungen gilt lediglich die Einschränkung, daß die gemeinsam adressierten Geräte den gleichen Auswahl-Code (in diesem Fall 9) haben müssen.

Auch durch eine ENTER-Anweisung ist es möglich, die aufzunehmenden Daten gleichzeitig mehreren Geräten zugänglich zu machen. Das geht aber nicht so einfach, da die ENTER-Anweisung nur eine einzige Adresse, nämlich die des Talkers, enthält und das HP-IL-System immer nur einen Talker zuläßt. Der Bus muß also besonders dazu vorbereitet werden, um dann eine ENTER-Anweisung folgen zu lassen, die nur den Auswahl-Code enthält. Dabei behält der Bus den zuvor geschaffenen Zustand.

Wenn z.B. das Gerät 07 (Voltmeter) seine Daten sowohl zum Gerät 13 (Drucker) als auch zum Gerät 04 (Massenspeicher) und außerdem zum aktiven Controller übertragen soll, muß der Bus gemäß der SEND-Anweisung vorbereitet werden: Es werden mit UNL alle Listener abgeschaltet, auf TALK bzw. LISTEN folgen die nötigen Adressen, der aktive Controller muß zusätzlich mit MLA (My Listen Address) adressiert werden. Die ENTER-Anweisung wickelt dann die Übertragung ab:

```
10 SEND 9 ; UNL TALK 7 LISTEN 13,4 MLA
20 ENTER 9 ; V
```

Für eine TRANSFER-Eingabe läßt sich eine ähnliche Sequenz verwenden, in diesem Falle zur Übertragung von Daten per Interrupt in den I/O-Buffer T$ des als Controller aktiven Rechners, wobei wegen der Dimensionierung des I/O-Buffers auf den I/O-ROM-Abschnitt 8 (Zweck und Benutzen der Buffer) hingewiesen wird:

```
10 DIM T$[88]
20 IOBUFFER T$
30 SEND 9 ; UNL TALK 7 LISTEN 13,4 MLA
40 TRANSFER 9 TO T$ INTR
```

Eine andere Spielart der Datenübertragung an mehrere Listener liegt vor, wenn der Rechner dabei weder Daten sendet noch Daten empfängt: Die Schleife transportiert Daten zwischen den Geräten. Um z.B. Daten vom Gerät 11 zu den Geräten 23, 4 und 7 zu übermitteln, lassen sich die folgenden Anweisungen benutzen:

```
10 SEND 9 ; UNL TALK 11 LISTEN 23,04,07
20 RESUME 9
```

Die RESUME-Anweisung fordert den Talker zur Ausgabe der Daten auf. Weil der Rechner hier nicht als Listener adressiert ist, sind weder ENTER- noch OUTPUT-Anweisung zum Auslösen der Übertragung nötig. Selbstverständlich ist dieses Verfahren auch anwendbar, wenn nur ein einziges Gerät die Daten erhalten soll.

Es gibt hier aber noch ein Problem: Woran erkennt der Rechner den Abschluß der Übertragung? Am besten dadurch, daß wir ihn in Zeile 10 zum Listener machen (MLA) und ein Eingabe-Transfer per Interrupt vereinbaren (wie im Beispiel davor), weil dann das Ende der Übertragung eine Zeilen-End-Verzweigung auslösen kann. Hier ein Beispiel dafür:

```
 5 DIM T$[1008]
 6 IOBUFFER T$ @ ON EOT 9 GOSUB 1000
10 SEND 9 ; UNL TALK 11 LISTEN 23,4,7 MLA
20 ! Hier wird END-Byte als Abschluß vereinbart:
30 TRANSFER 9 TO T$ INTR ; EOI
40 RESUME 9
50 ! RESUME startet die Übertragung!
```

Wie aus den Beispielen zu ersehen ist, dürfen die einstelligen Geräte-Adressen mit oder ohne führende Null eingegeben werden.

Alternative Übertragungsverfahren

Wie bereits mehrmals erwähnt wurde, besteht bei der TRANSFER-Anweisung die Wahl zwischen dem Interrupt-Verfahren (übertragung und Programm laufen gleichzeitig nebeneinander) und dem Fast-Handshake-Verfahren, das bei Unterbindung aller anderen Tätigkeiten größte übertragungsgeschwindigkeit bietet. In der TRANSFER-Anweisung ist anzugeben, welches Verfahren (INTR oder FHS) benutzt werden soll. Dieses Thema ist im I/O-Abschnitt 8 (Weitere übertragungsverfahren) ausführlich erklärt. Die wichtigsten Einzelheiten dieses Abschnitts sollten zum Verstehen des nächsten Beispiels geläufig sein:

```
 10 ! Daten per Interrupt von Gerät 04 zum Buffer T$
 20 ! Zeilen-Vorschub-Signal zeigt Ende an!
 30 DIM T$[88]
 40 I=0
 50 IOBUFFER T$
 60 ON EOT 9 GOSUB 120
 70 ! Transfer vorbereiten mit LF als Abschluß:
 80 TRANSFER 904 TO T$ INTR ; DELIM 10
 90 I=I+1
100 DISP "Ich zähle: ";I
110 GOTO 90
120 PRINT "Empfangen wurde: ";T$
130 ! Vorbereitung für neuen TRANSFER:
140 IOBUFFER T$ ! Löscht Bufferinhalt
150 TRANSFER 904 TO T$ INTR ; DELIM 10 @ RETURN
160 END
```

Behandlung von Störungs-Meldungen

Dieses Kapitel beschäftigt sich mit Störungs-Meldungen, die im System zu beliebigen Zeitpunkten auftreten können. Der Anlaß für eine Störungs-Meldung ist "geräteabhängig", d.h. die verschiedenen Geräte haben individuelle Gründe um derartige Meldungen abzugeben. Bei einem Drucker kann z.B. das Ende des Papiers ein Grund für eine solche Meldung sein, bei einem Digitalisierer kann es die Fehlbedienung durch den Benutzer sein, usw. Die in der Schleife vorhandenen Geräte können nur durch Veränderung des Rahmens von umlaufenden Daten- oder Kennungs-Meldungen eine Störungsmeldung abgeben. Deshalb gibt das Interface des aktiven Controllers auch in den Zeiten scheinbarer "Ruhe" in der Schleife dauernd Kennungs-Meldungen (IDY) aus. Beim Auftreten von Störungs-Meldungen müssen zwei Dinge ermittelt werden:

1. Welches Gerät hat die Meldung abgegeben?
2. Was will das Gerät melden?

Im Folgenden wird gezeigt, wie dies erreicht wird.

Das Bemerken der Störungs-Meldung

Im HP-IL-Interface wird der Inhalt des Status-Registers SR5 (auch) durch die Störungs-Meldung (SRQ) beeinflußt: In diesem Register ist das Bit 3 gesetzt, solange Daten- oder Kennungs-Meldungen mit gesetztem SRQ-Bit beim Controller eintreffen. Dieses Register läßt sich so abfragen:

```
STATUS 9,5 ; S
```

Das Bit 3 der Variablen S zeigt also an, daß zuvor eine SRQ-Meldung eingetroffen ist, d.h. eine Störung in der Schleife vorliegt. Mit dem Programm kann dann entschieden werden, wie auf das SRQ zu reagieren ist. Siehe hierzu IL-Abschnitt 5 (HP-IL für den Experten), wo eine vollständige Beschreibung der Status-Register zu finden ist. Wenn zuvor das Bit 3 im Steuer-Register CR1 gesetzt war, kann auch durch Abfrage von SR1 festgestellt werden, ob ein SRQ auftrat:

```
STATUS 9,1 ; S
```

Hier weist das gesetzte Bit 3 von S auf eine noch unbeachtet gebliebene SRQ-Meldung hin. Als Alternative zum wiederholten Abfragen der Status-Registers SR5 oder SR1 hat man auch die Möglichkeit, eine Zeilen-End-Verzweigung zu programmieren, die durch SRQ ausgelöst wird. Das folgende Beispiel schafft die Voraussetzungen:

```
10 ON INTR 9 GOSUB 200
20 ! Bit 3 des CR1 wird gesetzt:SRQ-Interrupt möglich!
30 ENABLE 9 ; 8
```

Das Programm wird dann bei jeder Störungsmeldung zu dem Unterprogramm verzweigen, das bei Zeile 200 beginnt. Als nächstes wollen wir jetzt überlegen, was in diesem Unterprogramm stehen sollte.

Das Feststellen der Störungs-Ursache

Der Hauptzweck der seriellen Abfrage (Serial Poll) ist es, dem aktiven Controller konkrete Informationen über den Zustand des abgefragten Gerätes zu geben. Das abgefragte Gerät antwortet mit einem Byte, von dem 7 Bits beliebige Bedeutungen haben können, die natürlich vorher festzulegen sind. Eine Ausnahme macht das Bit 6: Es zeigt an, ob das abgefragte Gerät momentan eine SRQ-Meldung abgibt.

Wir beziehen uns schon hier auf das nachfolgende Beispiel, bei dem das Programm zur Zeile 300 verzweigt, wenn ein Gerät eine Störungs-Meldung abgibt. Zuerst wird in der Service-Routine bei jedem Gerät geprüft, ob es SRQ gemeldet hat, dann werden die restlichen Bits untersucht, um die Störung einzugrenzen.

Wir nehmen an, daß die einzelnen Bits folgende Bedeutungen haben sollen, wenn sie gesetzt sind:

```
Bit 0 : Papier fehlt
Bit 1 : Deckel offen
Bit 2 : Parameter außer Bereich
Bit 3 : Fehlerhafte Escape-Sequenz
Bit 4 : nicht benutzt
Bit 5 : nicht benutzt
Bit 6 : SRQ wird ausgegeben
Bit 7 : nicht benutzt
```

Dann könnte eine Routine für das Gerät 5 z.B. so aussehen:

```
300 ! Service fuer Geraet 5
310 S=SPOLL(705)
320 ! Prüfung, ob SRQ vorliegt:
330 IF BIT(S,6) THEN GOTO 350
340 ENABLE INTR 9 ; 8 @ RETURN
350 IF NOT BIT(S,0) THEN GOTO 370
360 PRINT "Drucker ohne Papier!"
370 IF NOT BIT(S,1) THEN GOTO 400
380 PRINT "Drucker-Deckel offen!"
400 IF NOT BIT(S,2) THEN GOTO 420
410 PRINT "Druck-Parameter außer Bereich!"
420 IF NOT BIT(S,3) THEN PRINT "Unbekannter Drucker-Fehler!"
430 PRINT "Escape-Sequenz fehlerhaft!" @ GOTO 340
```

In einem größeren System sind serielle Abfragen für jedes Gerät nötig, um zuerst das betroffene Gerät zu ermitteln und danach dessen Probleme zu erfahren. Hierzu ein kurzes Beispiel:

```
100 S=SPOLL(905)
110 IF NOT BIT(S,6) THEN GOTO 200
120 ! Zeilen 120 bis 190 betreffen Geraet 5
200 S=SPOLL(706)
210 IF NOT BIT(S,6) THEN GOTO 300
     .
     .
```

Es muß hier auf einige Besonderheiten der Peripherie-Geräte des IL-Systems hingewiesen werden, die im Zusammenhang mit der SPOLL-Anweisung zu Verwunderung Anlaß geben können:

Die Status-Meldung der IL-Geräte kann mehrere Bytes umfassen, die SPOLL-Anweisung nimmt stets nur das erste Byte der Status-Meldung auf. Weder das I/O-ROM noch das IL-Interface der Rechner der Serie 80 bieten eine direkte Möglichkeit, die gesamte Status-Meldung aufzunehmen, wie Sie das eventuell von den Rechnern der Serien 40 und 70 kennen! Dazu ist ein Programm nötig, wie auf Seite IL-41 beschrieben!

Die obigen Beispiele der Statusabfrage geht davon aus, daß die Adressen der Geräte bekannt sind. Beim IL-System sind nun die Adressen von der (elektrischen) Reihenfolge abhängig und deshalb keineswegs unveränderlich! Es muß deshalb bei jeder Status-Abfrage genau bekannt sein, welche Geräte-Art sich hinter der (zufälligen) Adresse verbirgt, damit die richtige Routine aufgerufen werden kann. Die Technik dazu ist auf Seite IL-19 in den Programmzeilen 100 bis 130 und 200 zu finden. Besonders bei räumlich ausgedehnten Systemen ist diese Abfrage bequemer (und sicherer) als die Kontrolle durch Augenschein.

Außerdem gibt es auch IL-Geräte, die zwar keine SRQ-Meldung ausgeben können, die ihren Zustand aber sehr wohl mit ihrem Status-Byte beschreiben!

Aktivitäten als 'Nicht-Controller'

Wenn der Rechner nicht aktiver Controller ist, existieren für den Programmierer einige Einschränkungen, die er zur Vermeidung von Störungen im System beachten muß: Bestimmte Operationen kann eben nur der System-Controller auslösen, andere wiederum nur der aktive Controller. Ein "Nicht-Controller" kann eben nur um die Erteilung des Talker- bzw. Listener-Status ersuchen oder mit SRQ um Aufmerksamkeit bitten.

Abgeben der Controller-Funktion

Der Rechner kann nur als aktiver Controller diese Funktion an ein dazu geignetes anderes Gerät durch die PASS CONTROL-Anweisung übergeben. Danach kann der Rechner seine Aktivität z.B. einem anderen Interface zuwenden oder er kann seine ganze Kapazität für die schnellste Übermittlung von Daten an einen übergeordneten Rechner verwenden. Mit der folgenden Anweisung wird dem Gerät 6, einem anderen Rechner in der Schleife, die Funktion des aktiven Controllers angetragen

```
680 PASS CONTROL 906
```

Häufig wird der erste Rechner die Controller-Funktion später wiederum übernehmen wollen, es muß dazu schon vor der Abgabe der Controller-Funktion festgelegt werden, woran dieser Rechner die Rückgabe der Controller-Funktion erkennt und wie er sich dann verhalten soll. Es wird hier ausdrücklich darauf hingewiesen, daß auch der System-Controller nach Abgabe der Controller-Funktion so lange "Nicht-Controller" bleibt, bis ihm diese Funktion wieder zurückgegeben wird oder er sich selbst durch ein RESET des Systems sein Vorrecht verschafft. Wenn man diesen Sonderfall ausschließt, gelten die folgenden Überlegungen für alle "Nicht -Controller", auch wenn eines der Geräte als System-Controller definiert ist.

Annehmen der Controller-Funktion

Obwohl eine einfache Abfrage des Interface-Status darüber Auskunft geben kann, ob der Rechner aktiver Controller (geworden) ist, gelingt diese Überwachung doch besser mit einer Zeilen-End-Verzweigung, die automatisch bei Erteilung der Controller-Funktion ausgelöst wird. Im folgenden Programmausschnitt wird das Gerät 06 aktiver Controller, vorher wird aber schon die Möglichkeit für eine Zeilen-End-Verzweigung zur Zeile 700 geschaffen, um zu prüfen, ob die Controller-Funktion inzwischen schon wieder zurückgegeben wurde.

```
650 ! Subroutine zur Abgabe der Controller-Funktion
660 ON INTR 9 GOSUB 710
670 ! Bereitet Verzweigung bei Rückgabe vor!
680 ENABLE INTR 9 ; 32
690 PASS CONTROL 906
700 RETURN
710 ! Ab hier Routine für Annahme der Controller-Funktion
720 STATUS 9,1 ; S
730 IF BIT(S,5) THEN GOTO 760 ! Bit 5: Akt. Controller
740 ! Verzweigung, aber kein aktiver Controller!
750 RETURN
760 ! Ab hier wieder akt. Controller!
770 ! Es muß etwas geschehen!
 .
 .
```

Dieses Verfahren kann leicht so erweitert werden, daß es mit der gleichen Routine auch möglich wird, auf eine Adressierung des Rechners zum Talker oder Listener mit einer passenden Programmverzweigung zu reagieren.

Im folgenden Beispiel wird die einfachste Technik zur übergabe der Controller-Funktion zwischen zwei Rechnern der Serie 80 demonstriert, von denen Rechner A bei Start seines Programms aktiver Controller sein muß. Keiner der Rechner muß im Prinzip dabei den Rang des System-Controllers haben!

Programm des Rechners A:
(zu Beginn und am Schluß aktiver Controller!)

```
100 ! Adressierung als Controller:
110 OUTPUT 903 ; "Da ist sie!"
120 PASS CONTROL 903
130 ! Adressierung nun als Nicht-Controller:
140 ENTER 9 ; A$
150 DISP A$
160 ! Adressierung wieder als Controller:
170 OUTPUT 903 ; "Gern geschehen!"
180 END
```

Programm des Rechners B:
(nur vorübergehend aktiver Controller!)

```
200 ! Adressierung als Nicht-Controller:
210 ENTER 9 ; A$
220 DISP A$
230 ! Adressierung als Controller:
240 OUTPUT 902 ; "Mit Dank zurück!"
250 PASS CONTROL 902
260 ! Wieder als Nicht-Controller:
270 ENTER 9 ; B$
280 DISP B$
290 END
```

Die beiden Programme sind einzugeben, wobei angenommen wird, daß hinter dem Rechner A (in Umlaufrichtung der Meldungen) zwei andere Geräte folgen und deshalb der Rechner B für den Rechner A (als aktiver Controller) unter der Adresse "3" zu erreichen ist. Wenn (wieder in Umlaufrichtung) zwischen Rechner B und Rechner A ein weiteres Gerät eingefügt ist, kann der Rechner B (als aktiver Controller) dagegen den Rechner A unter Adresse "2" erreichen.

Erst ist Rechner B zu starten, er wartet dann in Zeile 210 auf Daten. Wenn jetzt auch Rechner A gestartet wird, gelangt mit Zeile 110 der erste Text (Da ist sie!) an Rechner B und wird dort angezeigt. Unmittelbar danach gibt Rechner A mit Zeile 120 die Controller-Funktion an den Rechner B und wartet selbst in Zeile 140 auf Daten vom Rechner B. Dieser, nunmehr Controller, kündigt in Zeile 240 durch Text (Mit Dank zurück!) die Rückgabe der Controller-Funktion in Zeile 250 an. Die anstehende ENTER-Anweisung in Zeile 140 ist damit erfüllt, Rechner A zeigt den Text an und ist unmittelbar danach nun auch wieder aktiver Controller. Er verabschiedet sich mit neuem Text (Gern geschehen!) in Zeile 170, der in Zeile 270 vom Rechner B bereits erwartet wird. Mit der Anzeige dieses Textes sind beide Programme abgelaufen und halten bei den Zeilen 180 bzw. 290 an.

Diese einfache Technik ist zwar sehr übersichtlich, aber nur für kleinere Systeme zu empfehlen, weil bereits geringfügige Verzögerungen im Programmablauf (im Beispiel zwischen den Zeilen 110/120 bzw. 240/250) zu Fehlermeldungen führen müssen, da der jeweilige Rechner dann Anweisungen in Angriff nimmt, für die die Controller-Funktion (die er dann noch nicht erhalten hat) unabdingbar ist.

Hinsichtlich der Adressen soll noch darauf hingewiesen werden, daß beide Rechner sich gegenseitig mit Adresse "1" ansprechen, wenn sich kein weiteres Gerät außer den Rechnern in der Schleife befindet. Dann muß aber einer der beiden Rechner unbedingt System-Controller sein, da sonst nach dem Einschalten ein betriebsfähiger Zustand der IL-Schleife nicht erreicht wird!

Korrekte Programme für "Gelegenheits-Controller"

Schon auf Seite IL-23 wurde gezeigt, wie ein Rechner der Serie 80 Daten senden und empfangen kann, auch wenn er nicht aktiver Controller ist - dazu brauchte nur der Interface-Auswahl-Code angegeben zu werden; Geräte-Adressen waren dabei nicht zulässig! Der Rechner konnte auch, wie auf Seite IL-31 gezeigt wurde, die Übergabe der Controller-Funktion erkennen und entsprechend verfahren. Hier werden nun diese beiden Beispiele so zusammengefaßt, daß mit Interrupt, Zeilen-End-Verzweigung und Analyse nicht nur die Übergabe der Controller-Funktion, sondern auch die Adressierungen zum Talker bzw. Listener über Programmverzweigungen zu passenden Routinen führen. Nach Abarbeiten der jeweiligen Routine wird die Interrupt-Maske erneuert und, falls möglich, die Controller-Funktion wieder zurückgegeben.

```
      .
      .
      .
  650 ! Subroutine zur Abgabe der Controller-Funktion
  660 ON INTR 9 GOSUB 700
  670 PASS CONTROL 906
  680 ENABLE INTR 9 ; (16+32+64) @ RETURN
  690 !
  700 ! Routine fuer 'Nicht-Controller'
  710 STATUS 9,1 ; S
  720 IF BIT(S,5) THEN GOTO 810
  730 IF BIT(S,4) THEN GOTO 780
  740 IF NOT BIT(S,6) THEN PRINT "Fehler" @ STOP
  750 ! Bit 6: als Listener aktiv:
  760 ENTER 9 ; A$
  770 GOTO 860
  780 ! Bit 4: als Talker aktiv:
  790 OUTPUT 9 ; X$
  800 GOTO 860
  810 ! Bit 5: als Controller aktiv:
  820 ! Ab hier Controller-Tätigkeit, danach Rückgabe!
      .
      .
      .
  850 PASS CONTROL 906
  860 ENABLE INTR 7 ; (16+32+64) @ RETURN
```

Zusätzliche Informationen über Interrupts finden sich im IL-Abschnitt 5 (HP-IL für den Experten), wo auch die Steuer- und Status-Register beschrieben sind.

Ausgeben von Störungsmeldungen

Im HP-IL-System kann jedes einzelne Gerät (Ausnahme: aktiver Controller!) die in der Schleife umlaufenden Daten-, END- und IDENTIFY-Meldungen mit den Werten 0, 2 und 6 in gleiche Meldungen mit den Werten 1, 3 und 7 umwandeln, um den aktiven Controller damit auf einen Zustand hinzuweisen, der "Beachtung" verdient (hierzu Seite IL-51 und IL-53). Da der Controller außer dieser Tatsache als solcher auch erfahren muß, was ihm mitgeteilt werden soll, hält jedes Gerät, das eine SRQ-Meldung abgegeben hat, eine wenigstens ein Byte umfassende Zustands-Beschreibung bereit, von der dem aktiven Controller bei der Abfrage des Zustandes wenigstens das erste Byte übermittelt wird.

Bei den einfacheren Geräten des Systems (Drucker, Plotter, Massenspeicher, Digitalisierer und ähnliche) erfolgt die Ausgabe der SRQ-Meldung mit Formulierung des Antwort-Bytes über fest eingebaute Programme. Anders ist es, wenn ein Rechner eine SRQ-Meldung abgeben will: Weil es sich hier um ein sehr universell benutzbares Gerät handelt, ist es unmöglich, vorfabrizierte Antwort-Bytes für alle nur denkbaren Fälle als Firmware einzubauen. Die Rechner der Serie 80 stellen aber zur bequemen Handhabung der SRQ-Meldungen die REQUEST-Anweisung bereit. Sie setzt nur ein Mindestmaß an Kenntnissen über den Aufbau des Antwort-Bytes voraus:

Von den 8 Bits (gemäß ihrer Wertigkeit mit 7 bis 0 bezeichnet) hat nur das Bit 6 eine Sonderstellung: Es bestätigt, wenn gesetzt, dem analysierenden Controller, daß tatsächlich eine SRQ-Meldung vorliegt. Die übrigen Bits (0 bis 5 und 7) kann der Programmierer einzeln und auch kombiniert zur Kennzeichnung von (maximal 127) verschiedenen Zuständen benutzen.

Für den Fall, daß ein Rechner (ohne Controller-Status) langwierige Berechnungen ausführt, kann z.B. vereinbart werden, daß beim Vorliegen eines Ergebnisses der Rechner ein Antwort-Byte vorbereitet, bei dem Bit 0 (Rechnung fertig) und Bit 6 (SRQ liegt vor) gesetzt sind. Dieses und auch das Beeinflussen des Wertes der umlaufenden Daten-, END- oder IDENTIFY-Meldungen besorgt die folgende Anweisung:

```
900 REQUEST 9 ; 1+64
```

Hinter dem Semikolon folgt ein numerischer Ausdruck, durch den der Zustand der einzelnen Bits des Antwort-Bytes festgelegt wird. Zwecks besserer Übersicht kann man, wie im Beispiel, die Formulierung mit einzelnen Summanden vornehmen, um die (kurze aber unanschauliche) Angabe als Summe zu vermeiden. Die REQUEST-Anweisung muß auch einen Auswahl-Code enthalten: Hiermit wird das Interface angesprochen, das zum Rechner gehört, der die SRQ-Meldung ausgibt! Dieser Auswahl-Code braucht nicht mit dem übereinzustimmen, den das HP-IL-Interface im aktiven Controller besetzt!

Der Umgang mit Störungen in IL-Schleife und -Interface

Dieser Abschnitt beschreibt einige Verfahren, die Sie zur Vermeidung oder Untersuchung von Störungen kennen sollten, wenn Sie das HP-IL-Interface benutzen. Das bedeutet nicht, daß Sie unbedingt auf solche Probleme stoßen werden, aber es ist ein guter Grundsatz, im Programm den "Problemen" nach Möglichkeit zuvor zu kommen und sich schon vorher Gedanken über sie zu machen.

Wie vermeidet man das 'Hängenbleiben' während einer Operation?

Allgemein läßt sich sagen, daß eine Störung in einem HP-IL-Gerät entweder die Datenübertragung anhält und/oder eine SRQ-Abfrage durch den aktiven Controller auslöst. Wir wissen bereits, wie der Controller auf den Eingang von Störungsmeldungen reagiert (Seite IL-29 und folgende), was passiert aber, wenn ein Gerät eine Datenübertragung unvollendet abbricht und deswegen eine SRQ-Abfrage auslöst? Dieses Zusammentreffen schafft ein Problem, denn der Rechner kommt nicht zum Zeilen-Ende, kann also die SRQ-Routine nicht beginnen. Der Grund liegt in Definition und Namen der Zeilen-End-Verzweigung: Diese ist nie während der Ausführung, sondern erst nach Beendigung einer Programmzeile möglich. Und das gilt natürlich auch für Zeilen mit OUTPUT- oder ENTER-Anweisungen! Das Gerät hat jetzt die Datenübertragung eingestellt, der Rechner wartet auf die Fortsetzung, er kann das Zeilenende nicht erreichen und deshalb auch nicht die Zeilen-End-Verzweigung beginnen, die vielleicht eine Lösung bietet! Der Rechner hängt also fest. Eine Rettung aus dieser 'Fallgrube' ist möglich, wenn Sie die dafür vorgesehene SET TIMEOUT-Anweisung in das Programm einbauen.

Das folgende Beispiel zeigt die richtige Reihenfolge der notwendigen Operationen, um ein Programm vor dem 'Hängenbleiben' zu schützen.

```
 10 DIM S(8)
 20 ! Es soll 10-mal gemessen werden, hoechstens aber 10 Sek.
 30 ON TIMEOUT 9 GOSUB 160
 40 ! Hier wird die Zeitgrenze von 10 000 ms eingestellt:
 50 SET TIMEOUT 9 ; 10000
 60 ! Jetzt wird ein DVM auf Triggerung programmiert:
 70 OUTPUT 902 ; "F1R1T3T3"
 80 FOR J=1 TO 10 ! 10-mal messen:
 90 TRIGGER 902
100 ENTER 902 ; X
110 DISP X;"Volt"
120 NEXT J
130 DISP "Ende!"
140 ! Ab 160 TIME OUT-Routine, zuerst
150 ! Vorkehrung gegen defektes IL-Interface!
160 ON TIMEOUT 9 GOTO 270 ! Falls Test zu lange dauert!
170 FOR I=0 TO 7
180 STATUS 9,I ; S(I)
190 PRINT "Status-Register ";I;" =";S(I)
200 NEXT I
210 ! Hier gehören die vom Ergebnis der Status-
220 ! Abfrage abhaengigen Service-Routinen hin
    .
250 GOTO 290 ! Ende der TIME OUT-Routine
260 !
270 PRINT "Interface-Fehler"
280 RESET 9 ! Versuch zu helfen
290 ON TIMEOUT 9 GOSUB 160 ! Erneuert alte Service-Routine
300 RETURN
```

Der Programmierer muß also festlegen, wie das Programm auf das Ergebnis der Statusabfrage reagieren soll. Meistens ist für das Interface ein RESET nötig, um den Datenaustausch mit dem gestörten Gerät abzubrechen, danach sollte der Benutzer über die Störung informiert werden. Diese Dinge werden im nächsten Abschnitt behandelt.

Der Umgang mit "Problemen"

Wahrscheinlich lesen Sie diesen Abschnitt, wenn Ihr HP-IL-System nicht korrekt arbeitet oder, weil Sie sich informieren wollen, wie man "Problemen" zuvorkommt. Sie können hier zuerst sehen, was zu tun ist, wenn "überhaupt nichts mehr" läuft, weil dann schnelle Hilfe nötig ist. Anschließend werden Tips zur Eingrenzung der Störungen gegeben.

Wenn ein HP-IL-System festgefahren scheint (es läuft irgendetwas, aber nichts geschieht!), dann sollte Ihnen die RESET-Taste wieder Einfluß auf den Rechner verschaffen. Nur, wenn der Rechner in einem FHS-Transfer hängenbleibt, hilft Ihnen das nichts! Die beste Möglichkeit für diesen Fall ist die Ausführung eines INTERFACE-CLEAR (IFC) auf irgendeine Weise (auch ein Schleifen-Analysator kann das!). Als zweit-"bestes" bleibt nur Aus- und wieder Einschalten des Rechners!

Danach und auch bei den sonstigen Störungen sollten Sie dann die einzelnen Geräte seriell auf ihren Status prüfen. Das setzt aber voraus, daß Sie wissen, wie man diese Information korrekt bekommt. Manchmal hilft auch hier einfaches Aus- und Einschalten, um ein Peripherie-Gerät wieder zur "Vernunft" zu bringen.

Der folgende Abschnitt zeigt die verschiedenen Möglichkeiten (und die Auswahlkriterien dafür), die dem Programmierer beim Entwerfen der Fehler-Routinen für die Behebung von Störungen im HP-IL-System zur Verfügung stehen.

Handlung:	**Absicht bzw. Ergebnis:**
FOR I=1 TO 7 STATUS 9,I ; S(I) NEXT I	Ermittelt den momentanen Status vom Interface um Zustand und Ursachen erklären zu können.
S=SPOLL(<jedes Gerät>)	Ermittelt momentanen Status der Geräte im System.(z.B. Drucker ohne Papier)
CLEAR <Einzelgerät>	Setzt das bezeichnete Gerät auf seinen teilweise geräteabhängigen "Clear"-Status zurück.(RESET-ähnlich!)
CLEAR 9	Setzt alle Geräte auf ihren geräteabhängigen "Clear"-Status zurück.
ABORTIO 9	Zweckmäßig nur, wenn der Rechner der Serie 80 System-Controller ist! Beendet alle Bus-Aktivitäten, der System-Controller wird wieder aktiver Controller.

Es ist zweckmäßig, die ermittelten Zustände von Interface und Geräten ausdrucken zu lassen. Ebenso sollte auch jede Aktion zur Behebung der Störung dokumentiert werden. Diese Informationen sind die Grundlage für die Ermittlung der Störungsursache und für den Programmierer nützlich, um entsprechende Hinweise für die Fehlerbeseitigung zu geben (z.B. den Drucker mit Papier zu versorgen!).

HP-IL für den Experten

Dieser Abschnitt ist für Benutzer bestimmt, die das HP-IL-System in seinen Grundzügen bereits gut kennen und jetzt alle gegebenen Möglichkeiten kennenlernen wollen, um auch neuartige Anwendungstechniken erproben zu können. Hier werden aber nur die grundsätzlichen Möglichkeiten gezeigt, die das HP-IL-System bietet, ausführliche Programmbeispiele sind hier nicht zu erwarten!

Die HP-IL-I/O-Anweisungen

ABORTIO: Falls das Interface System-Controller ist, wird IFC (Interface Clear) ausgeführt und REN (Fernsteuerung) für alle Geräte der Schleife möglich. Als aktiver Controller wird das Interface Talker. Wenn das Interface nicht aktiver Controller ist, wird die laufende Operation abgebrochen, ohne den Zustand der Schleife zu ändern. Die nächste Operation kann beginnen (siehe auch HALT).

ASSERT: ASSERT beeinflußt sofort das Steuer-Register CR3 und damit das sofort auszugebende Byte, auch beim "beschäftigtem" Interface. Die Beeinflussung von CR3 durch CONTROL hat ähnlichen Effekt, wartet aber vor Ausführung die Beendigung der laufenden Operation ab und gibt das neue Byte nicht automatisch aus, sondern braucht dazu SEND-Anweisung, die nur den Auswahl-Code angibt.

CLEAR: Nur dem aktiven Controller gestattet. Die adressierten Geräte (bei Auswahl-Code aber alle) Geräte des Systems erhalten den DCL-Befehl (Device CLear), der die Geräte in zuvor festgelegte Zustände versetzt.

CONTROL: Beeinflußt die Steuer-Register vom Interface. Fehler 111 wird ausgelöst, wenn die Beeinflussung eines nicht vorhandenen Registers versucht wird. Zugänglich sind die Register 0...5 und 16...23 (sowie Register 0 und 1 der I/O-Buffer).

ENABLE INTR: Beeinflußt Steuer-Register CR1 (auch mit CONTROL möglich), erzeugt die Maske für die Zeilen-End-Verzweigungen.

ENTER: Mit Adresse nur dem aktiven Controller gestattet. Mit Auswahl-Code beginnt die Aufnahme erst nach der Adressierung des Interfaces zum Listener. Die Wirkung von ENTER-Anweisungen mit und ohne Adressierung zeigen die Beispiele:

`ENTER 905 ; A$` Nur für den aktiven Controller zugelassen, führt erst Adressierung aus, nimmt dann Daten für die Variable A$ auf.

`ENTER 9 ; A$` Falls Interface nicht aktiver Controller, wird abgewartet, bis Listener-Adressierung erfolgt. Falls das Interface selbst aktiver Controller ist, muß sich dieses zuvor selber als Listener deklariert haben, sonst erfolgt Fehlermeldung 116.

HALT: Veranlaßt das Interface, die laufende Tätigkeit abzubrechen und für die nächste I/O-Operation bereit zu sein. HALT beeinflußt nicht die Bus-Signale. Beachten Sie, daß ABORTIO ebenso wirkt, wenn das Interface weder System-Controller noch aktiver Controller ist.

LOCAL: Nur dem aktiven Controller gestattet. An die (gleichzeitig oder zuvor) adressierten Listener des Systems wird der NRE-Befehl (Not Remote Enable) gegeben, der die Fernsteuerung sperrt.

LOCAL LOCKOUT: Nur dem aktiven Controller gestattet. Veranlaßt das Interface zur Ausgabe des LLO-Befehls (Local Lock Out), der die manuelle Beeinflussung der Geräte verhindert.

OUTPUT: Mit Adresse(n) nur dem aktiven Controller gestattet. Mit Auswahl-Code beginnt die Ausgabe erst, wenn das Interface aktiver Talker geworden ist. Die Wirkung von OUTPUT-Anweisungen mit und ohne Adressierung zeigen die Beispiele:

`OUTPUT 905,906 ; A$` Nur dem aktiven Controller gestattet, führt zuerst Adressierung aus, gibt dann Inhalt der Variablen A$ aus.

`OUTPUT 9 ; A$` Falls Interface nicht aktiver Controller, wird die Adressierung zum Talker abgewartet. Falls Interface aktiver Controller, muß sich dieses zuvor selbst als Talker deklarieren, sonst Fehlermeldung 115.

PASS CONTROL: Nur dem aktiven Controller gestattet. Nach (gleichzeitiger oder vorheriger) Adressierung zum Talker wird die Controller-Funktion übergeben. Der aktive Controller kann seine Funktion an jedes dazu geeignete Gerät (auch an sich selbst) übergeben.

PPOLL: Nur dem aktiven Controller gestattet. Es wird IDY über die Schleife geschickt, damit die vorbereiteten Geräte das zuwiesene Bit in dieser Meldung verändern können.

REMOTE: Nur dem aktiven Controller gestattet. An die (gleichzeitig oder zuvor) adressierten Listener des Systems wird der REN-Befehl (Remote ENable) ausgegeben, der Fernsteuerung ermöglicht.

REQUEST: Dem aktiven Controller nicht gestattet. Bringt durch Setzen von Bit 6 (der Bits 0...7) gültiges SRQ ins Interface. übrige Bits beliebig. Dieses Byte ist für aktiven Controller als Antwort auf serielle Abfrage bestimmt, wobei SRQ gelöscht wird. SQR kann auch mit neuer REQUEST-Anweisung zurückgenommen werden, falls diese Bit 6 wieder löscht.

RESET: Setzt das Interface bedingungslos auf den Einschaltzustand zurück und löst damit Selbst-Test für das Interface aus, der bei gestörtem Ablauf die Fehlermeldung 110 auslöst. Bei erfolgreichem Test erfolgt keine Meldung. Falls das Interface System-Controller ist, wird die Schleife neu geordnet und Fernsteuerung möglich.

RESUME: Nur dem aktiven Controller gestattet. Veranlaßt Beginn der zuvor vereinbarten Datenübertragung, falls diese noch nicht begonnen wurde, sonst keine Wirkung.

SEND: Zur Ausgabe von speziell zusammengestellten Meldungen angewendet. Datenausgabe setzt Talker-Status voraus, sonst wird Fehlermeldung 115 ausgelöst. Nur der aktive Controller darf damit auch Befehle ausgeben. Bei Befehlen braucht der aktive Controller nicht aktiver Talker zu sein.

SPOLL: Nur dem aktiven Controller gestattet. Ermittelt mit serieller Abfrage den Wert des Status-Bytes. Dieses zeigt an, ob das Gerät ein SRQ ausgelöst hat, und kann auch zusätzliche Informationen geben.

STATUS: Liest die Status-Register vom Buffer bzw. Interface. STATUS wird sofort ausgeführt, ohne den Interface-Zustand zu berücksichtigen und stellt damit den momentanen Interface-Zustand dar. Die Status-Register haben die Nummern 0...3 für Buffer bzw. 0...7 für IL-Interfaces. Der Versuch, andere Register mit der STATUS-Anweisung zu lesen, löst Fehlermeldung 111 aus.

TRANSFER: Für Interrupt- oder Fast-Handshake-Transfer I/O-Operationen benutzt. TRANSFER setzt definierten I/O-Buffer voraus (siehe I/O-ROM-Abschnitt 8, wo auch Informationen über die Buffer-Zeiger zu finden sind). Zusammen mit TRANSFER können drei verschiedene Schluß-Merkmale benutzt werden:

1. DELIM: Ende bei Erkennen des verabredeten Zeichens.
2. COUNT: Ende nach festgelegter Menge von Bytes.
3. EOI: Beenden der übertragung durch END-Byte.

Nachstehend eine Tabelle über die Anwendungsmöglichkeiten:

TRANSFER-Methode	**DELIM**	**COUNT**	**EOI**
TRANSFER(ein)FHS	Nicht zulässig	Anwendung bei Bedarf, sonst Transferablauf durch Füllzeiger und Bufferende gesteuert.	END-Byte wirksam
TRANSFER(aus)FHS	Nicht zulässig	Nicht zulässig, Transferablauf durch Lese- und Füllzeiger bestimmt	Nicht zulässig: EOL-Sequenz wird ausgegeben
TRANSFER(ein)INTR	Zulässig	Anwendung bei Bedarf, sonst Eingabe durch DELIM oder EOI oder Bufferlänge beendet	END-Byte wirksam
TRANSFER(aus)INTR	Nicht zulässig	Nicht zulässig, Transferablauf durch Lese- und Füllzeiger bestimmt	Nicht zulässig: EOL-Sequenz wird ausgegeben

TRIGGER: Nur dem aktiven Controller gestattet. An die (gleichzeitig oder zuvor) adressierten Listener des Systems wird der GET-Befehl (Group Execute Trigger) ausgegeben.

Aufbau der Meldungen

Die durch die IL-Schleife miteinander verbundenen Geräte tauschen untereinander in geordneter Weise Informationen aus. Dieser Austausch geht stets von einem einzigen Gerät aus und ist an eines oder mehrere Geräte gerichtet. Die Information wird in einzelnen Bytes übertragen. Jedem Byte wird ein "Rahmen" vorgesetzt, der aus 3 Bits besteht. Jede dieser Gruppen ist eine "Meldung", deren 11 Bits seriell ausgegeben werden.

Der vorgesetzte Rahmen ermöglicht die Unterscheidung der verschiedenen Meldungs-Arten durch den Wert der drei vorangestellten Bits (siehe hierzu die Tabellen auf Seite IL-51 und IL-53).

1. Datenmeldungen werden vom Talker ausgegeben und sind für einen oder mehrere Listener bestimmt. Diese Meldungen können mit zwei verschiedenen Rahmen ("0" und "2") ausgegeben werden, wobei der Rahmen "2" das letzte Byte einer mehr Bytes umfassenden Information gekennzeichnet werden kann.

   Jedes Gerät der Schleife, das diese Datenmeldungen weitergibt, kann den Rahmen von "0" auf "1" bzw. von "2" auf "3" verändern, um damit auf einen SRQ-Fall hinzuweisen.

   Die Umlauf-Zeit einer Daten-Meldung wird nicht automatisch überwacht, es ist deshalb zweckmäßig, im Programm eine Zeitbegrenzung vorzusehen, um ein "Hängenbleiben" rechtzeitig zu bemerken.

2. Die Befehls-Meldung wird nur vom aktiven Controller ausgegeben und ist durch den Rahmen "4" gekennzeichnet. Der aktive Controller wartet nur kurze Zeit auf das Wiedereintreffen der unveränderten Meldung, da die Geräte den Befehl sofort weitergeben und ihn erst dann abarbeiten.

   Zur Gruppe der Befehls-Meldungen, die 256 Befehle ermöglicht, gehören z.B. die Adressierungen zum Talker oder Listener, die Umschaltung auf Fernbedienung, die Löschung der automatischen Adressen, sowie Trigger- und Reset-Befehl (siehe hierzu Seite IL-42).

3. Ready-Meldungen dienen zur allgemeinen Überwachung und Steuerung des Schleifenzustandes. Meldungen dieser Art werden mit dem Rahmen "5" vom Controller oder Talker ausgeben. Dabei lassen sich drei Arten unterscheiden:

   Am einfachsten ist die RFC-Meldung zu verstehen, die von jedem Gerät in der Schleife weitergegeben wird, wenn es keine zuvor erhaltenen Befehle mehr abzuarbeiten hat: Ein beim aktiven Controller wieder eintreffender RFC-Befehl bestätigt die Bereitschaft der Schleife für neue Operationen.

   Die zweite Gruppe der Ready-Meldungen umfaßt Befehle und Quittungen, die an bestimmte Geräte gerichtet sind: Wenn der aktive Controller Ready-Meldungen dieser Art ausgibt, erwartet er eine Reaktion des angesprochenen Gerätes. Wenn der aktive Talker eine der ihm gestatteten Ready-Meldungen ausgibt, erwartet er eine Reaktion des aktiven Controllers: Keine dieser Meldungen darf unverändert beim sendenden Gerät wieder eintreffen!

   Die dritte Gruppe der Ready-Meldungen beschäftigt sich mit den Aufgaben der Adressierung.

4. Der Rahmen "6" kennzeichnet eine IDY-Meldung, durch die sich die Geräte der Schleife von "STAND BY" in den Arbeitsmodus bringen lassen. Die Rechner der Serie 80 benutzen diese Meldung zur Dauerüberwachung der Schleife in Zeiten fehlender sonstiger Aktivität: Dabei können Geräte mit SRQ-Wünschen den Rahmen der IDY-Meldung in "7" ändern.

Ausgeben von READY-Meldungen

Die Rechner der Serie 80 benutzen die Befehls- und Ready-Meldungen implizit bei der Ausführung der auf das IL-System gerichteten I/O-Anweisungen. Mit Hilfe der SEND-Anweisung lassen sich auch alle Befehls-Meldungen (Seite IL-44) explizit benutzen. Von den Ready-Meldungen (Seite IL-45) sind aber nur SDI (dezimal 98) als CMD 255 und SAI (dezimal 99) als CMD 254 in einer SEND-Anweisung ausführbar (Beispiel hierzu auf Seite IL-19). Die restlichen Meldungen der READY-Gruppe sind zum größten Teil nur für ausgefallene Spezialfälle interessant, lediglich die Abfrage aller Status-Bytes mit SST kann häufig nützlich sein.

Durch die kombinierte Anwendung der CONTROL-, ASSERT- und STATUS-Anweisungen lassen sich auch alle READY-Meldungen ausgeben, es ist dabei jedoch nötig, auf die eintreffende Meldungen so zu reagieren, wie es das HP-IL-Protokoll vorschreibt.

Folgende Schritte sind dazu notwendig: Das abzufragende Gerät muß als Talker, der Rechner als Listener deklariert werden. Mit einer CONTROL-Anweisung wird der Rahmen "5" ins Register CR2 gesetzt. Die gewünschte Codierung der READY-Meldung wird danach mit der ASSERT-Meldung in das Register CR3 gebracht, wodurch Rahmen (aus CR2) und Meldung (aus CR3) sofort an die Schleife übergeben werden. Die Antwort des Talkers gelangt in die Status-Register SR2 und SR3. Abfrage und weitere Speicherung sind damit möglich. Außerdem muß aber die komplette empfangene Meldung in der Schleife weitergegeben werden, damit der Talker die Ausgabe der nächsten Meldung vornehmen kann: die Inhalte von SR2 und SR3 werden deshalb mit CONTROL- bzw. ASSERT-Anweisungen nach CR2 bzw. CR3 geschrieben (und damit automatisch weitergegeben. Am Ende der übermittelten Meldung geht eine ETO-Meldung ein, die im aktiven Controller den Beginn der nächsten IL-Operation auslösen kann.

Am Beispiel der ausführlichen Statusabfrage soll das Verfahren erläutert werden:

```
 10 RESET 9
 20 SEND 9 ; UNL MLA TALK 1 ! Geraet 1 soll abgefragt werden!
 30 S2=5 @ S3=97 ! Code fuer READY und SST !
 40 FOR I=1 TO 9 ! Es werden bis zu 8 Status-Bytes erwartet!
 60 CONTROL 9,2 ;S2 ! Setzt Rahmen in CR2 !
 70 ASSERT 9;S3 ! Setzt SST-Code in CR3 !
 70 STATUS 9,2 ; S2,S3 ! Fragt eingehende Meldung ab !
 80 IF NOT S2 THEN S(I)=S3 ELSE 100 ! Daten-Byte speichern,
 90 NEXT I !sonst Schleife verlassen!
100 IF S2#5 OR S3#64 THEN DISP "Fehler bei Uebermittlung!"
110 SEND 9 UNL UNT CMD 155 ! Abschalten des Systems!
120 DISP "Es wurde empfangen:"
130 FOR J=1 TO I-1 ! Das ETO-Byte wird nicht angezeigt!
140 DISP J;". Byte:";S(J)
150 NEXT J
160 END
```

Das gezeigte Beispiel muß noch um die Auswertung der Status-Bytes erweitert werden, eine Anwendung auf mehrere Geräte ist durch den Aufbau einer übergeordneten Schleife leicht möglich.

Ähnliche Möglichkeiten bestehen auch, wenn das zuletzt ausgegebene Daten-Byte ein END-Byte (mit Rahmen "2" oder "3") ist. Auch hier ist lediglich eine protokollgerechte Reaktion nötig, um den Ablauf ohne Störung zu beenden. Bei unbekannter Geräte-Konfiguration in der Schleife empfiehlt es sich, die Art des abgefragten Gerätes mit der auf den Seiten IL-19/20 gezeigten Technik zuvor zu ermitteln.

Notizen:

Zusammenstellung der Befehls- und READY-Meldungen

Zum besseren Verständnis der auf den folgenden Seiten tabellarisch dargestellten Befehls- und Ready-Meldungen sind hier die darin verwendeten Abkürzungen und ihre Bedeutungen angegeben:

AAD/AAG	Automatisch erteilte ADresse -speziell/Gruppenbezeichnung
AAU	Automatische Adressen Ungültig
ACG	Adressierte Befehlsmeldung (Gruppenbezeichnung)
AEP/AES/AMP	Automatische -Erweiterte Primäradresse/Sekundäradresse/ Mehrfach-Primäradresse
ARG	Adressierte Ready-Meldung (Gruppenbezeichnung)
DCL	Device Clear, Rücksetzen aller Geräte
DDC/DDL/DDT	Geräteabhängiger Befehl -allgemein/für Listener/für Talker (Gruppenbezeichnungen)
EAR	Nur gelegentliche SRQ-Abfrage mit IDY-Meldung
EOT/ETE/ETO	Ende einer Übertragung -allgemein/fehlerhaft/fehlerfrei
GET	Trigger-Befehl
GTL	Freigeben der manuellen Einstellung
IAA/IEP/IES/IMP	Illegale -Automatische Adresse/erweiterte Primäradresse /erweiterte Sekundäradresse/Mehrfach Primäradresse
IFC	InterFace Clear
LAD/LAG	ListenerADresse -speziell/Gruppenbezeichnung
LLO	Local LockOut, Sperren der manuellen Betätigung
LPD	Loop Power Down, Umschalten auf Stand-By-Betrieb
NRD	Not Ready for Data, Daten können nicht mehr aufgenommen werden
NRE	Not Remote Enable, Fernsteuerung nicht möglich
NUL	Null-Befehl (ohne Wirkung)
PPD/PPE/PPU	Parallel-Abfrage -sperren/zuordnen/Festlegungen löschen
REN	Remote ENable, Fernsteuerung möglich
RFC	Ready For Command, zur nächsten Operation bereit
SAD/SAG	SekundärADresse -speziell/Gruppenbezeichnung
SOT	Start einer Datenübertragung (Gruppenbezeichnung)
SAI/SDI/SDA/SST	Abruf von -Zusatzidentifizierung/Gerätebezeichnung/Daten/Status
SDC	Rücksetzen der adressierten Geräte
TAD/TAG	TalkerADresse -speziell/Gruppenbezeichnung
TCT	Take ConTrol, übergabe der Controller-Funktion
UCG	Universelle Befehlsmeldung (Gruppenbezeichnung)
UNL/UNT	Abschalten -aller Listener/des Talkers

Die Codierungen der Befehls-Meldungen

Rahmen: 100 Wert: 4

	x0 0000	x1 0001	x2 0010	x3 0011	x4 0100	x5 0101	x6 0110	x7 0111	x8 1000	x9 1001	xA 1010	xB 1011	xC 1100	xD 1101	xE 1110	xF 1111	
0x 0000	NUL	GTL			SDC	PPD			GET							ELN	ACG
1x 0001	NOP	LLO			DCL	PPU			EAR								UCG
2x 0010	LAD 0	LAD 1	LAD 2	LAD 3	LAD 4	LAD 5	LAD 6	LAD 7	LAD 8	LAD 9	LAD 10	LAD 11	LAD 12	LAD 13	LAD 14	LAD 15	LAG
3x 0011	LAD 16	LAD 17	LAD 18	LAD 19	LAD 20	LAD 21	LAD 22	LAD 23	LAD 24	LAD 25	LAD 26	LAD 27	LAD 28	LAD 29	LAD 30	UNL	LAG
4x 0100	TAD 0	TAD 1	TAD 2	TAD 3	TAD 4	TAD 5	TAD 6	TAD 7	TAD 8	TAD 9	TAD 10	TAD 11	TAD 12	TAD 13	TAD 14	TAD 15	TAG
5x 0101	TAD 16	TAD 17	TAD 18	TAD 19	TAD 20	TAD 21	TAD 22	TAD 23	TAD 24	TAD 25	TAD 26	TAD 27	TAD 28	TAD 29	TAD 30	UNT	TAG
6x 0110	SAD 0	SAD 1	SAD 2	SAD 3	SAD 4	SAD 5	SAD 6	SAD 7	SAD 8	SAD 9	SAD 10	SAD 11	SAD 12	SAD 13	SAD 14	SAD 15	SAG
7x 0111	SAD 16	SAD 17	SAD 18	SAD 19	SAD 20	SAD 21	SAD 22	SAD 23	SAD 24	SAD 25	SAD 26	SAD 27	SAD 28	SAD 29	SAD 30		SAG
8x 1000	PPE0 0	PPE0 1	PPE0 2	PPE0 3	PPE0 4	PPE0 5	PPE0 6	PPE0 7	PPE1 0	PPE1 1	PPE1 2	PPE1 3	PPE1 4	PPE1 5	PPE1 6	PPE1 7	ACG
9x 1001	IFC		REN	NRE							AAU	LPD					UCG
Ax 1010	DDL 0	DDL 1	DDL 2	DDL 3	DDL 4	DDL 5	DDL 6	DDL 7	DDL 8	DDL 9	DDL 10	DDL 11	DDL 12	DDL 13	DDL 14	DDL 15	ACG
Bx 1011	DDL 16	DDL 17	DDL 18	DDL 19	DDL 20	DDL 21	DDL 22	DDL 23	DDL 24	DDL 25	DDL 26	DDL 27	DDL 28	DDL 29	DDL 30	DDL 31	ACG
Cx 1100	DDT 0	DDT 1	DDT 2	DDT 3	DDT 4	DDT 5	DDT 6	DDT 7	DDT 8	DDT 9	DDT 10	DDT 11	DDT 12	DDT 13	DDT 14	DDT 15	ACG
Dx 1101	DDT 15	DDT 16	DDT 18	DDT 19	DDT 20	DDT 21	DDT 22	DDT 23	DDT 24	DDT 25	DDT 26	DDT 27	DDT 28	DDT 29	DDT 30	DDT 31	ACG
Ex 1110																	
Fx 1111																	

Übernommen aus "The HP-IL Interface Specifikation"

Die Codierungen der READY-Meldungen

Rahmen: 101 Wert: 5

	x0 0000	x1 0001	x2 0010	x3 0011	x4 0100	x5 0101	x6 0110	x7 0111	x8 1000	x9 1001	xA 1010	xB 1011	xC 1100	xD 1101	xE 1110	xF 1111	
0x 0000	RFC																
1x 0001																	
2x 0010																	
3x 0011																	
4x 0100	ETO	ETE	NRD														ARG
5x 0101																	ARG
6x 0110	SDA	SST	SDI	SAI	TCT												ARG
7x 0111																	ARG
8x 1000	AAD 0	AAD 1	AAD 2	AAD 3	AAD 4	AAD 5	AAD 6	AAD 7	AAD 8	AAD 9	AAD 10	AAD 11	AAD 12	AAD 13	AAD 14	AAD 15	AAG
9x 1001	AAD 16	AAD 17	AAD 18	AAD 19	AAD 20	AAD 21	AAD 22	AAD 23	AAD 24	AAD 25	AAD 26	AAD 27	AAD 28	AAD 29	AAD 30	IAA	AAG
Ax 1010	AEP 0	AEP 1	AEP 2	AEP 3	AEP 4	AEP 5	AEP 6	AEP 7	AEP 8	AEP 9	AEP 10	AEP 11	AEP 12	AEP 13	AEP 14	AEP 15	AAG
Bx 1011	AEP 16	AEP 17	AEP 18	AEP 19	AEP 20	AEP 21	AEP 22	AEP 23	AEP 24	AEP 25	AEP 26	AEP 27	AEP 28	AEP 29	AEP 30	IEP	AAG
Cx 1100	AES 0	AES 1	AES 2	AES 3	AES 4	AES 5	AES 6	AES 7	AES 8	AES 9	AES 10	AES 11	AES 12	AES 13	AES 14	AES 15	AAG
Dx 1101	AES 16	AES 17	AES 18	AES 19	AES 20	AES 21	AES 22	AES 23	AES 24	AES 25	AES 26	AES 27	AES 28	AES 29	AES 30	IES	AAG
Ex 1110	AMP 0	AMP 1	AMP 2	AMP 3	AMP 4	AMP 5	AMP 6	AMP 7	AMP 8	AMP 9	AMP 10	AMP 11	AMP 12	AMP 13	AMP 14	AMP 15	AAG
Fx 1111	AMP 16	AMP 17	AMP 18	AMP 19	AMP 20	AMP 21	AMP 22	AMP 23	AMP 24	AMP 25	AMP 26	AMP 27	AMP 28	AMP 29	AMP 30	IMP	AAG

Übernommen aus "The HP-IL Interface Specifikation"

Zustands-Abfrage (Send Status, SST)

Der Zustand der zum System gehörenden Geräte kann vom Controller jederzeit abgefragt werden. Das geschieht (oft unbemerkt) bei vielen der I/O-Anweisungen, wenn sie das IL-Interface betreffen. Die gleiche Abfrage gibt aber auch beim Auftreten von SRQ-Fällen die nötigen Hinweise für das weitere Vorgehen. Für die Behandlung von SRQ-Fällen sind aber 2 Verfahren möglich, die nun getrennt behandelt werden.

Serielle Abfrage (Serial Poll)

Die serielle Abfrage liefert dem aktiven Controller von jedem Gerät, das auf diese Abfrage reagiert, ein Status-Byte. Die Abfrage wird erst nötig, wenn eine SRQ-Meldung bemerkt wurde. Der Controller wird dann nacheinander die Geräte abfragen, von denen eine SRQ-Meldung erwartet wird. Das Gerät wird nun mit seinem (ersten) Status-Byte antworten, in dem das Bit 6 (der Bits 0..7) nur gesetzt ist, wenn das Gerät die SRQ-Meldung (mit-)verursacht hat. Die Ursache für die SRQ-Meldung kann mit den übrigen Bits genauer erklärt werden. Ihre Zuordnung zu bestimmten Zuständen wird für jedes Gerät anders sein. Es gibt 2 Wege, den SRQ-Fall zu bemerken:

1. Periodische Abfrage, ob Bit 3 im Status-Register SR5 gesetzt ist.
2. Zulassen eines Interrupts mit anschließender Zeilen-End-Verzweigung, ausgelöst durch Bit 3 im Status-Register SR1.

In einem System, das außer dem Rechner nur ein Gerät enthält, das seinerseits nur eine Störungsursache hat, ist eine serielle Abfrage sinnlos! Wenn aber dieses Gerät mehrere Gründe für eine SRQ-Meldung haben kann, und erst recht bei mehreren Geräten, die SRQ-Meldungen auslösen können, muß seriell abgefragt werden, damit festgestellt werden kann, wer (alles) die Meldung auslöste und was das einzelne Gerät zu melden hat.

Die serielle Abfrage wird vom aktiven Controller mit der SPOLL-Anweisung mit kompletter Adresse durchgeführt. Das HP-IL-Interface veranlaßt dann folgende Aktivitäten:

1. Alle Listener werden abgeschaltet. Das Interface adressiert sich selbst als Listener und das abzufragende Gerät als Talker.
2. Der SPE-Befehl (Serial Poll Enable) wird ausgegeben, gefolgt von der SST-Anweisung (Send Status), wodurch das abgefragte Gerät sein (erstes) Status-Byte ausgibt.
3. Danach folgt der SPD-Befehl (Serial Poll Disable) womit die Abfrage beendet ist und das (erste) Status-Byte nun intern untersucht werden kann.
4. Unmittelbar nach dem Abschluß der Abfrage beginnt der Rechner der Serie 80 wieder mit der dauernden Ausgabe von IDY-Meldungen, was beim Weiterbestehen von SRQ sofort erneut die entsprechenden Status-Register setzen würde.

Im Programm muß festgelegt werden, welche Geräte in welcher Reihenfolge abzufragen sind.

Wenn der Rechner nicht aktiver Controller ist, darf er keine Abfrage durchführen, dafür aber selber eine SRQ-Meldung abgeben. Dazu ist nur eine REQUEST-Anweisung nötig, mit der das Status-Byte vorbereitet wird. In diesem Byte muß das Bit 6 unbedingt gesetzt sein, da nur dann der Rahmen der nächsten dazu geeigneten Meldung von "0" auf "1" bzw. "2" auf "3" bzw. "6" auf "7" geändert wird. Die übrigen Bits können zur Beschreibung des Zustandes oder der Wünsche des Rechners frei benutzt und entsprechend definiert werden. Der Rechner übergibt dann sein Status-Byte bei der seriellen Abfrage an den aktiven Controller. Nach der Abfrage durch den aktiven Controller durchlaufen die IDY-Meldungen die IL-Schleife wieder unverändert, wenn der Rechner als einziges Gerät SRQ melden wollte. Im Status-Byte wird durch die Abfrage nur das Bit 6 gelöscht, alle restlichen Bits bleiben bis zur nächsten REQUEST-Anweisung unverändert.

Parallele Abfrage (Parallel Poll)

Bei der parallelen (gleichzeitigen) Abfrage wird jedem Gerät des Systems ein Bit der IDY-Meldung zugeordnet, das im SRQ-Fall in verabredeter Weise geändert wird. Das erfordert bei den einzelnen Geräten zwar einige Vorbereitungen (Adressierung als Listener und Ausgabe des entsprechenden PPE-Befehls), die jedoch nur einmal nötig sind. Die Reaktion auf die Abfrage ist dafür um so schneller: Danach verändern die so vorbereiteten Geräte, auch ohne als Listener oder Talker aktiviert zu sein, eine umlaufende IDY-Meldung in der vereinbarten Weise, wenn bei ihnen ein SRQ-Fall auftritt.

Die Parallele Abfrage kann viel schneller arbeiten als eine serielle, da der aktive Controller die acht Bits der IDY-Meldung fast gleichzeitig erhält und daraus sofort erkennt, welche der Geräte ungewöhnliche Zustände melden wollen.

Vorteilhaft läßt sich die Parallel-Abfrage einsetzen, wenn höchstens 8 Geräte zu überwachen sind. Eine Erweiterung auf 16 Geräte ist möglich, kompliziert aber die Auswertung erheblich. Falls mehrere Geräte gleicher Art im System ähnliche Aufgaben haben, also im System eine Gruppe bilden, ist ein anderes Vorgehen günstiger: Alle Geräte einer Gruppe bekommen das gleiche Bit zugeordnet. Die Parallelabfrage zeigt dann sofort, welche Geräte-Gruppe Hilfe benötigt, wodurch die gezielte Abfrage der Geräte schneller durchzuführen ist.

Notizen:

Die Steuer-Register und die Status-Register

Einleitung

Das Interface soll in erster Linie den Daten-Austausch zwischen dem Rechner und den mit ihm verbundenen Peripherie-Geräten ermöglichen. Die Baugruppen, die zu einem Interface benötigt werden, sind im allgemeinen sehr anpassungsfähig und zum Teil in ihrem Verhalten programmierbar, sie können also durch bestimmte Anweisungen vom Rechner auf verschiedene Arbeitsweisen eingestellt werden. Das HP-IL-Interface besitzt dazu 13 Steuer-Register, in die sich bestimmte Daten einschreiben lassen. Diese Register sind über die SET I/O-Anweisung (des Plotter-Printer-ROM's) bzw. die CONTROL-Anweisung (des I/O-ROM's) erreichbar.

Mit diesen beiden Anweisungen lassen sich die Steuer-Register mit den Nummern 1 bis 5 und 16 bis 23 ansprechen. Die Numerierung der Register erscheint etwas eigenartig, ermöglicht aber eine gewisse Kompatibilität zu anderen, älteren Systemen (wie RS 232 und GPIO), bei denen bestimmte, auch im HP-IL-System nötige Angaben schon immer in den Registern 16 bis 23 festgelegt wurden. Der Versuch, Daten in Register einzubringen, die außerhalb der zugelassenen Bereiche liegen, führt zur Fehlermeldung 111.

In der praktischen Anwendung sind mit den Steuer-Registern 1 bis 4 die Status-Register gleicher Nummer eng verbunden. Das HP-IL-Interface verfügt über 8 Status-Register, die mit Nummern von 0 bis 7 bezeichnet sind. Diese Register können nur abgefragt, also gelesen werden, und beschreiben den internen Zustand des Interfaces recht eingehend.

In den folgenden Ausführungen werden auch verschiedene Interrupt-Gründe erwähnt, die als "durch Zustand bedingt", "durch Ereignis bedingt" und "durch Bitte um Hilfe bedingt" unterschieden werden. Dazu ist zu bemerken, daß von keinem dieser Interrupt-Gründe exakt vorhergesagt werden kann, wann er eintritt! Die Unterscheidung bezieht sich vielmehr darauf, welche Auswirkungen die zeitliche Abfolge von Interrupt-Erwartung und Interrupt-Auftreten hat. Es ist wie im Leben: gegen manche Dinge kann man sich nur durch "Vor"-sichtsmaßnahmen sichern, bei anderen läßt sich auch "nach"-her noch etwas erreichen! Zur textlichen Vereinfachung wird das von nun an als "Zustands"-, "Ereignis"- und "Hilfe"-Interrupt bezeichnet.

Die HP-IL-Steuer-Register CR1 bis CR5

Die Steuer-Register CR1 bis CR5 haben sehr großen Einfluß auf das Verhalten des IL-Interfaces, wodurch auch die Erklärungen recht umfangreich ausfallen. Die später behandelten Steuer-Register CR16 bis CR23 legen dagegen nur fest, auf welche Art das Ende einer Datensequenz gekennzeichnet wird.

Die Steuer-Register lassen sich alle (numerisch aufeinanderfolgende auch gemeinsam) mit der CONTROL-Anweisung des I/O-ROM's in die jeweils gewünschten Zustände versetzen. Auch die SET I/O-Anweisung des Plotter-Printer-ROM's erreicht alle Steuer-Register, aber nur einzeln! Daneben werden vom I/O-ROM weitere spezielle Anweisungen geboten, die jeweils nur ein bestimmtes Steuer-Register ansprechen: So leisten trotz unterschiedlicher Form und Syntax die beiden Anweisungen

```
ENABLE INTR 9 ; 8   und   CONTROL 9,1 ; 8
```

exakt das gleiche, wobei ENABLE INTR nur auf CR1 wirken kann. Bei den Anweisungen

```
ASSERT 9 ; 1   und   CONTROL 9,3 ; 1
```

ist die Übereinstimmung dagegen nicht vollkommen, weil die gewünschte Einstellung des Registers CR3 mit verschiedener Dringlichkeit vorgenommen wird: Die CONTROL-Anweisung, formuliert, um CR3 auf den Wert 1 zu bringen, beginnt erst, wenn die momentan im IL-System laufende Operation beendet ist. Die ASSERT-Anweisung, die nur CR3 erreichen kann, wird sofort ausgeführt, ohne Rücksicht auf die laufende Operation.

==========

Warnung!

Die Steuer-Register 2 und 3 (CR2 u. CR3) ermöglichen den direkten Zugriff auf die HP-IL-Steuer- und -Daten-Funktionen. Hier ist genaue Kenntnis über das HP-IL-Protokoll nötig und Vorsicht angeraten! Ungeschickte Handhabung kann hier der Anlaß zu Störungen in der Schleife aber auch zu Schäden an den Geräten sein!

==========

Die folgende Tabelle gibt eine Übersicht über diese Steuer-Register, danach folgen Erklärungen für die einzelnen Register.

HP-IL-Steuer-Register CR1 bis CR5

Reg. Nr.:	Bit-Nummer: 7	6	5	4	3	2	1	0	Start-Wert:	Register-Funktion:
CR1	IFC	LA	CA	TA	SRQ	DCL oder SDC	GET	DDC	0	Interrupt-Maske
CR2	X	X	X	X	X	C2	C1	C0	4	Nächster Rahmen
CR3	D7	D6	D5	D4	D3	D2	D1	D0	Sinn-los	Nächstes Byte
CR4	AAD	X	X	A4	A3	A2	A1	A0	Sinn-los	Geräte-Adresse
CR5	X	X	X	X	X	X	X	EAR	0	IDY-Modus

CR1: Interrupt-Maske
Das Setzen eines Bits gestattet der dazugehörenden Interrupt-Bedingung, eine Zeilen-End-Verzweigung auszulösen.

Bit 0: DDC (DDL oder DDT)
Es muß beim Empfangen des gerätespezifischen Befehls (DDL oder DDT) gesetzt sein, wenn dadurch ein Interrupt ausgelöst werden soll. Ereignis-Interrupt!

Bit 1: GET
Es muß beim Empfangen eines Trigger-Befehls (GET) gesetzt sein, wenn dadurch ein Interrupt ausgelöst werden soll. Ereignis-Interrupt!

Bit 2: DCL oder SDC
Es muß beim Empfangen eines DCL- oder SDC-Befehls gesetzt sein, wenn dadurch ein Interrupt ausgelöst werden soll. Ereignis-Interrupt!

Bit 3: SRQ
Es ist zu setzen, wenn ein Interrupt durch eine SRQ-Meldung ausgelöst werden soll. Hilfe-Interrupt!

Bit 4: TA
Es ist zu setzen, wenn jede Adressierung zum aktiven Talker einen Interrupt auslösen soll. Zustands-Interrupt!

Bit 5: CA
Es ist zu setzen, wenn die Adressierung zum aktiven Controller einen Interrupt auslösen soll. Zustands-Interrupt!

Bit 6: LA
Es ist zu setzen, wenn jede Adressierung zum aktiven Listener einen Interrupt auslösen soll. Zustands-Interrupt!

Bit 7: IFC
Es muß beim Empfangen eines IFC-Befehls gesetzt sein, wenn dadurch ein Interrupt ausgelöst werden soll. Ein von außen (aus der Schleife) kommender IFC-Befehl kann diesen Interrupt auch auslösen, wenn das Interface System-Controller ist. Ereignis-Interrupt!

CR2: Schleifen-Steuerung (Rahmen-Bits)
Die 3 Rahmen-Bits dieses Registers werden dem Byte aus Register CR3 vorangestellt und als nächste Meldung (Frame) gemeinsam ausgegeben. Die Rahmen-Bits haben die in der folgenden Tabelle aufgeführten Bedeutungen. Der Startwert für das Register CR2 ist "4" (Befehl).

Wert:	Zuordnung:	Achtung:
0	Daten	SRQ kann nur
1	Daten	mit dem Wert
2	END (letztes Daten-Byte)	"7" ausgegeben
3	END (letztes Daten-Byte)	werden!
4	CMD (Befehl)	
5	READY (Fertigmeldung)	
6	IDENTIFY (Kennung)	
7	IDENTIFY (Kennung mit SRQ)	

CR3: Nächstes Byte
Das Byte aus diesem Register wird als nächstes unmittelbar nach den Rahmen-Bits aus CR2 als Meldung ausgegeben. Für einen Wagen-Rücklauf muß z.B. in CR2 vermerkt werden, daß es sich um Daten handelt und CR3 muß den Dezimalwert des Symbols enthalten:

```
10 CONTROL 9,2 ; 0,13 (Schreibt 0 in CR2 und 13 in CR3)
```

Diese Daten finden Sie in den Status-Registern SR2 und SR3 nach Durchlaufen der Schleife wieder.

CR4:Geräte-Adresse
Die Bits 0 bis 4 verkörpern die dem Interface zugeordnete Adresse, die Bits 5 und 6 sind unbenutzt, Bit 7 ist ein Flag, das nur bei gültiger Adresse gesetzt ist. Der Startwert ist "31", also eine ungültige Adresse. Sobald aber das Interface aktiver Controller wird, wechselt der Wert auf "128" (Adresse "0"). Bei ungültiger Adresse muß der Wert unter "128" liegen.

CR5: IDY-Modus
Von diesem Register wird nur das Bit 0 benutzt, das sich nur beeinflussen läßt, wenn das Interface aktiver Controller ist. Der Startwert ist 0. In diesem Zustand gibt der aktive Controller in jeder durch das Programm gegebenen Pause IDY-Meldungen in die Schleife, um damit SRQ-Meldungen zu ermöglichen.

Wenn das Bit 0 gesetzt ist, wird die automatische Abfrage unterbrochen, bis durch die nächste Aktivität des Controllers in der Schleife dieses Bit wieder gelöscht wird und die Dauer-Abfrage erneut aufgenommen wird.

Die HP-IL-Status-Register SR0 bis SR7

Diese Register lassen sich mit der STATUS-Anweisung lesen. Um z.B. den Inhalt des Statusregisters SR3 der Variablen S3 zuzuordnen, kann folgende Anweisung dienen:

```
STATUS 9,3 ; S3
```

Die folgende Tabelle gibt Übersicht über diese Register und ihren Zweck. In den folgenden Abschnitten werden die Einzelheiten erklärt.

HP-IL-Status-Register SR0 bis SR7

Reg. Nr.:	Bit-Nummer: 7	6	5	4	3	2	1	0	Start-Wert:	Register-Funktion:
SR0	0	0	0	0	0	1	0	1	5	Interface-Kennung
SR1	IFC	LA	CA	TA	SRQ	DCL oder SDC	GET	DDC	0	Interrupt-Grund
SR2	X	X	X	X	X	C2	C1	C0	4	Letzter Rahmen
SR3	D7	D6	D5	D4	D3	D2	D1	D0	0	Letztes Byte
SR4	AAD	X	X	A4	A3	A2	A1	A0	Sinn-los	Geräte-Adresse
SR5	Tx	LA	CA	TA	SRQ	EAR	REN	LLO	Sinn-los	Interface-Status
SR6	X	X	T/L	C4	C3	C2	C1	C0	0	Geräteabhängiger Befehl
SR7	X	X	X	A4	A3	A2	A1	A0	0	Geräte-Anzahl

SR0: Interface-Identifizierung
Abfrage liefert den Wert 5, das Kennzeichen für ein HP-IL-Interface.

SR1: Interrupt-Gründe
Dieses Register wird bei jeder Abfrage durch die STATUS-Anweisung gelöscht. Der abgefragte Wert zeigt durch den Zustand der einzelnen Bits an, welche von den im Register CR1 zugelassenen Interrupt-Ursachen inzwischen eingetreten sind. Weitere Einzel- und Besonderheiten sind aus den Ablaufdiagrammen auf den Seiten IL-56 und IL-57 ersichtlich. Die einzelnen Bits reagieren auf folgende Ursachen nur, wenn das gleichwertige Bit in CR1 gesetzt ist:

Bit 0: DDC (DDL oder DDT)
Anzeige für das Eintreffen eines geräte-abhängigen Befehls. Der Befehl selbst ist in SR6 abgelegt. Ereignis-Interrupt!

Bit 1: GET
Anzeige für das Eintreffen eines GET-Befehls (Triggerauslösung), wobei das Interface aktiver Listener ist. Ereignis-Interrupt!

Bit 2: DCL oder SDC
Anzeige für das Eintreffen eines DCL- oder SDC-Befehls (Device CLear oder Selected Device Clear), wobei das Interface aktiver Listener ist. Ereignis-Interrupt!

Bit 3: SRQ
Anzeige für das Eintreffen einer SRQ-Meldung (Service ReQuest), mit der ein Gerät des Systems die Aufmerksamkeit des Controllers auf sich ziehen will. Hilfe-Interrupt!

Bit 4: TA
Anzeige dafür, daß das Interface aktiver Talker wurde oder weiter bleiben soll. Zustands-Interrupt!

Bit 5: CA
Anzeige dafür, daß das Interface aktiver Controller wurde. Zustands-Interrupt!

Bit 6: LA
Anzeige dafür, daß das Interface aktiver Listener wurde oder weiter bleiben soll. Zustands-Interrupt!

Bit 7: IFC
Anzeige für das Eintreffen eines IFC-Befehls (InterFace Clear). Ereignis-Interrupt!

SR2: Schleifen-Steuerung (Rahmen-Bits)
Die Bits 0 bis 2 dieses Registers sind der Rahmen der zuletzt eingegangenen Meldung. Der Wert dieser Bits hat folgende Bedeutungen:

Wert:	Zuordnung:
0	**Daten**
1	**Daten** (mit SRQ)
2	**END** (letztes Daten-Byte)
3	**END** (letztes Daten-Byte mit SRQ)
4	**CMD** (Befehl)
5	**READY** (Fertigmeldung)
6	**IDENTIFY** (Kennung)
7	**IDENTIFY** (Kennung mit SRQ)

SR3: Letztes Byte
Das Byte in diesem Register wurde als letztes empfangen. Die Rahmen-Bits aus SR2 beschreiben die Art der empfangenen Meldung.

SR4: Geräte-Adresse

Bit 0 bis 4:
Die Bits 0 bis 4 verkörpern die dem Interface zugeordnete Adresse. Die Bits 5 und 6 sind unbenutzt.

Bit 7: Gültigkeit der Adresse
Dieses Bit ist gelöscht, wenn die Adresse ungültig ist. Es wird gesetzt, wenn dem Interface vom aktiven Controller (auch wenn es das selber ist!) eine Adresse erteilt wird.

SR5: Interface-Status
Aus diesem Register ist der momentane Zustand des IL-Interfaces zu ersehen.

Bit 0: LLO
Bit 0 ist gesetzt, wenn sich das Interface im LLO-Zustand (Local LockOut) befindet, also nicht vom Rechner beeinflußt werden kann.

Bit 1: REN
Bit 1 ist gesetzt, wenn sich das Interface im REN-Zustand (Remote ENable) befindet, also von außen beeinflußt werden kann.

Bit 2: EAR
Bit 2 ist gesetzt, wenn in der Schleife die willkürliche, gelegentliche SRQ-Abfrage (Enable Asynchronous Request) durchgeführt werden kann. Wenn Bit 2 gelöscht ist, fragt das Interface (natürlich nur als aktiver Controller!) bei jeder Gelegenheit automatisch dauernd alle Geräte der Schleife auf SRQ-Meldungen ab.

Bit 3: SRQ
Bit 3 ist gesetzt, wenn aus der Schleife eine SRQ-Meldung empfangen wurde.

Bit 4: TA
Bit 4 ist gesetzt, sobald das Interface aktiver Talker ist und schon einen Befehl zur Ausgabe von Daten erhalten hat.

Bit 5: CA
Bit 5 ist gesetzt, wenn das Interface aktiver Controller ist.

Bit 6: LA
Bit 6 ist gesetzt, wenn das Interface aktiver Listener ist.

Bit 7: Tx
Wenn das Interface aktiver Controller ist, wird das Bit 7 während der Daten-Übermittlung in der Schleife gesetzt, auch dann, wenn das Interface von den Meldungen nicht betroffen ist.

SR6: Register für den geräte-abhängigen Befehl
Der Wert dieses Register verkörpert einen Befehl mit geräte-spezifischer Wirkung, wenn gleichzeitig in SR1 das Bit 0 (DDC) gesetzt ist.

Bit 0 bis 4: DDC
Diese Bits verkörpern den geräte-abhängigen Befehl, auf den in SR1 mit dem Bit 0 hingewiesen wird.

Bit 5: DDL/DDT
Bit 5 ist gesetzt, wenn es sich um einen Listener-Befehl (DDL) handelt, bei einem Talker-Befehl (DDT) ist das Bit 5 gelöscht. Die Bits 6 und 7 sind unbenutzt.

SR7: Geräte-Anzahl
Die Abfrage dieses Registers ergibt die Zahl der Geräte, die vom IL-Interface betreut werden (das Interface wird dabei nicht mitgezählt). Dafür werden die Bits 0 bis 4 benutzt. Der Wert dieser Bits entspricht der Zahl der adressierbaren Geräte in der Schleife ohne das Interface. Die Bits 5 bis 7 sind unbenutzt.

Die HP-IL-Steuer-Register CR16 bis CR23

Die Steuer-Register CR16 bis CR23 geben dem Benutzer die Möglichkeit, die Zeilen-End-Sequenz (EOL) nach Wunsch zu gestalten. Diese wird jedesmal ausgegeben, wenn der "/"-Spezifikator in einer PRINT- bzw. OUTPUT-Anweisung auftritt oder wenn die Anweisung für die Datenübertragung beendet wird. Ebenso ist es aber auch möglich, das letzte Zeichen als END-Byte auszugeben und damit die Enden von Zeichengruppen zu markieren, ohne aber damit die Übermittlung endgültig abzuschließen.

Die folgende Tabelle gibt Übersicht über diese Register und deren Wirkungen. In den folgenden Absätzen werden die Einzelheiten erklärt.

HP-IL-Zeilen-End-Sequenz-Register CR16 bis CR23

Reg. Nr.:	Bit-Nummer: 7	6	5	4	3	2	1	0	Start-Wert:	Register-Funktion:
CR16	END ein	X	X	X	X	EOL2	EOL1	EOL0	2	END,EOL-Umfang
CR17	----------Startwert : CHR$(13) :					Wagenrücklauf---------			13	Zeichen 1
CR18	----------Startwert : CHR$(10) :					Zeilenvorschub--------			10	Zeichen 2
CR19									0	Zeichen 3
CR20									0	Zeichen 4
CR21									0	Zeichen 5
CR22									0	Zeichen 6
CR23									0	Zeichen 7

CR16: Form der EOL-Sequenz und END-Option

Hier wird die Zeichenzahl der EOL-Sequenz festgelegt und außerdem vereinbart, ob das letzte Zeichen einer Gruppe als END-Byte ausgegeben werden soll.

Bit 0 bis 2: Zeichenanzahl
Die Summe der Wertigkeiten der gesetzten Bits entscheidet über die Zahl der ausgegebenen Zeichen.

Bit 3 bis 6: bleiben wirkungslos

Bit 7: END-Option
Wenn Bit 7 gesetzt ist, wird das letzte Byte der Nachricht mit dem Rahmen "2" als END-Byte markiert. Das "letzte" Byte ist das letzte Zeichen der EOL-Sequenz, wenn wenigstens eines der Bits 0 bis 2 gesetzt ist, sonst das letzte Zeichen der Nachricht. Die END-Option ist nur bei PRINT-, DISP- und TRANSFER-Anweisungen wirksam, OUTPUT- und SEND-Anweisungen geben keine END-Bytes aus!

CR17 bis 23: Zeichenregister

In diesen Registern können bis zu sieben Bytes als EOL-Sequenz abgelegt werden. Als Startwert stehen in CR17 und CR18 Wagenrücklauf und Zeilenvorschub, die gemäß dem Startwert 2 im CR16 ausgegeben werden.

Um z.B. mit der EOL-Sequenz einen doppelten Zeilenvorschub auszulösen, muß der EOL-Umfang mit 3 und die Sequenz mit 'CR, LF, LF' eingegeben werden:

```
CONTROL 9,16 ; 3,13,10,10
```

Steuer-Register-Beeinflussung mit dem Plotter-Printer-ROM

Auch wenn dem Rechner nur ein Plotter-Printer-ROM zur Verfügung steht, können Sie alle Steuer-Register von CR1 bis CR5 und CR16 bis CR23 in die von ihnen gewünschten Zustände versetzen.

Weil das Plotter-Printer-ROM jedoch keine Abfrage der Status-Register zuläßt (dazu ist die STATUS-Anweisung nötig, die nur das I/O-ROM bietet), sollten Sie Eingriffe in die Register CR1 bis CR5 unterlassen, damit Störungen des IL-Protokolls vermieden werden.

Sinnvoll kann dagegen auch hier ein Einwirken auf die Zeilen-End-Sequenz-Register CR16 bis CR23 sein. Um mit der SET I/O-Anweisung eine von der Norm abweichende EOL-Sequenz zu definieren, müssen Sie jedes der betroffenen Register einzeln ansprechen. Wenn, wie auch im letzten Beispiel angenommen, eine EOL-Sequenz mit doppeltem Zeilenvorschub (also insgesamt 3 Zeichen!) ausgegeben werden soll, benötigen grundsätzlich die Register CR16/17/18/19 neue Werte:

```
SET I/O 9,16,3        CR16: 3 Zeichen
SET I/O 9,17,13       CR17: Wagenrücklauf        CHR$(13)
SET I/O 9,18,10       CR18: Zeilenvorschub 1     CHR$(10)
SET I/O 9,19,10       CR19: Zeilenvorschub 2     CHR$(10)
```

Nur wenn Sie sicher sein können, daß die EOL-Register noch ihre Start-Werte enthalten, dürfen Sie sich die 2. und 3. Anweisung ersparen.

Ablauf-Diagramme für die Interrupt-Register CR1 und SR1

Stark vereinfacht kann man sich die Register CR1 und SR1 als ein Formular mit bestimmten Fragen und vorbereiteten Antwortfeldern vorstellen. Dieses Formular ist für einen Parkplatz-Wächter bestimmt. Da der Parkplatz an einer engen Einbahn-Straße liegt, kann der Wächter, wenn er will, alle Fahrzeuge bemerken. Durch unsere Fragen verlangen wir nun von ihm, auf bestimmte Dinge zu achten und diese Beobachtungen durch Ankreuzen zu 'protokollieren'. Dabei ist klar, daß ein mehrmaliges gleiches Vorkommnis nur zu einem einzigen Kreuz führt! Die STATUS-Abfrage gleicht dann der Wegnahme des Formulars, wodurch der Wächter vorübergehend seine 'Aufgabe' verliert und sich nur noch um seinen Parkplatz kümmert. Erst wenn er ein neues Formular mit (neuen) Fragen bekommt, kann er wieder tätig werden und uns dann sogar noch sagen, wer alles in der Zwischenzeit auf den Parkplatz gefahren ist. Nur die Vorgänge auf der Straße sind ihm entgangen.

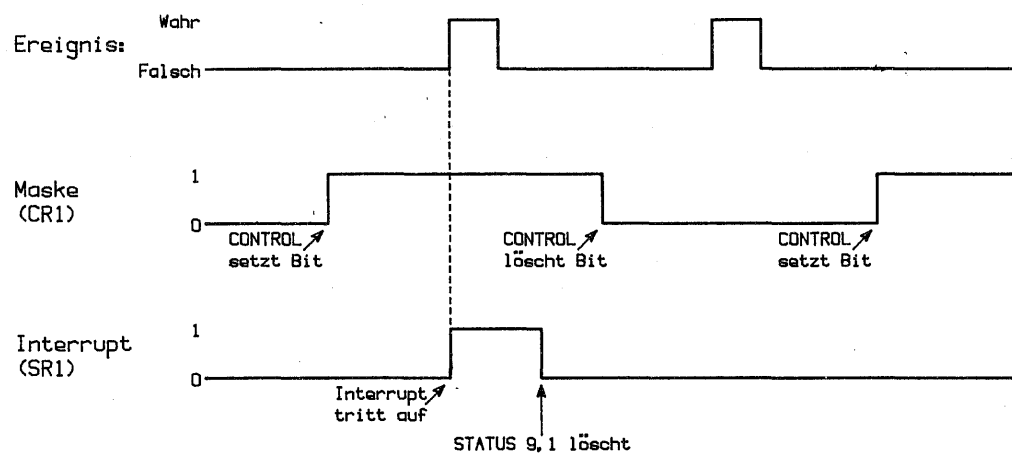
In der Formulierung unserer Fragen liegen da erstaunliche Möglichkeiten, wie die folgende Gegenüberstellung zeigen soll:

a) Machen Sie ein Kreuz, wenn der erste rote (blaue, gelb-schwarze, grüne) Sportwagen auf der Straße vorbeifährt!
b) Machen Sie ein Kreuz, wenn seit der Abholung des vorigen Formulars ein blauer (roter, grüner) Pkw geparkt hat!
c) Machen Sie ein Kreuz, wenn sich ein Lkw auf dem Parkplatz befindet!

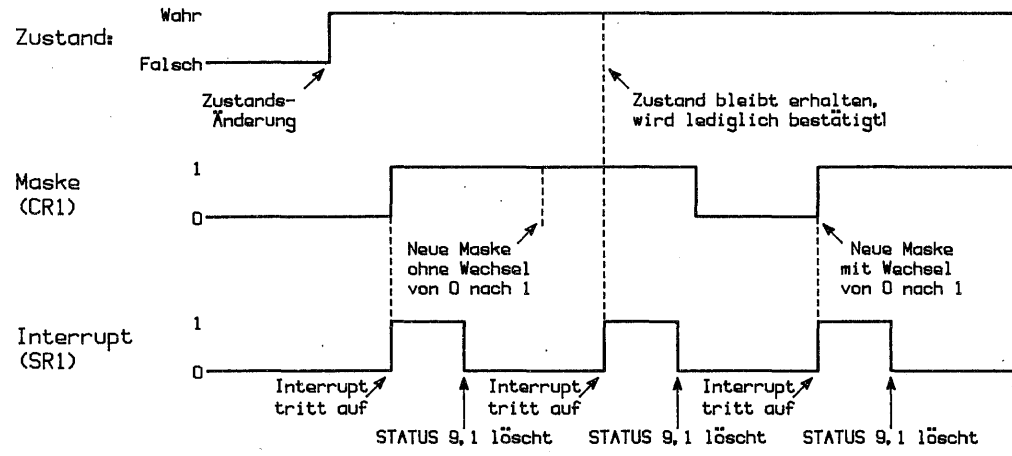
Den schnellen Sportwagen in verschiedener Farbe im Fall a entsprechen die Befehle DDC, GET, SDC/DCL und IFC. Die Pkw's im Fall b verkörpern die Adressierungen zum Talker, Controller oder Listener, und der Lkw im Fall c entspricht der SRQ-Meldung, die, genau wie der Lkw, nach und nach geprüft und entladen werden muß.

Die folgenden Ablauf-Diagramme zeigen die Reaktionen des Interfaces beim Auftreten von Interrupt-Ursachen der drei verschiedenen Arten.

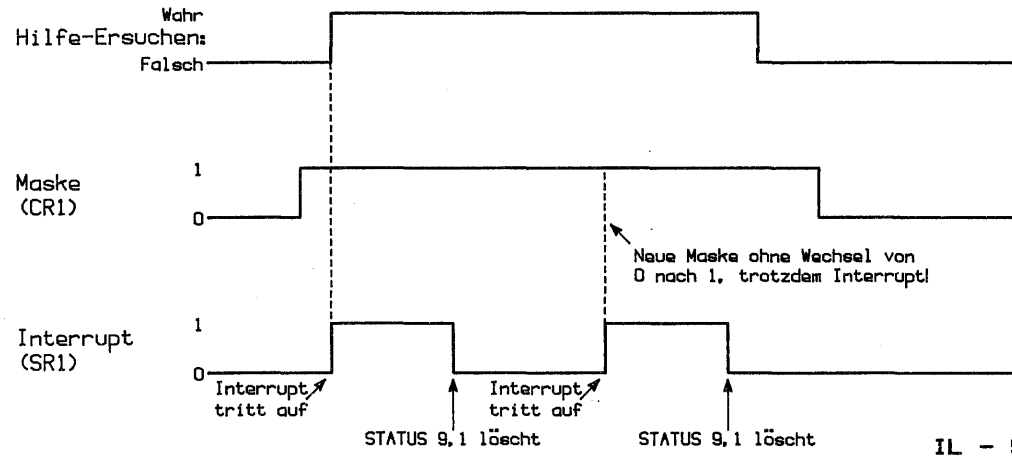
Ereignis als Interrupt-Ursache:
(DDC, GET, SDC/DCL, IFC)



Zustandsänderung als Interrupt-Ursache:
(TA, CA, LA)



Hilfe-Ersuchen als Interrupt-Ursache:
(SRQ)



Zusammenfassung der HP-IL-(I/O-ROM-)Anweisungen

Die folgende Tabelle faßt die Bedingungen zusammen, die das Interface erfüllen muß, damit bestimmte Programm-Anweisungen zulässig sind und nennt die hierdurch ausgelösten Vorgänge. Bedenken Sie, daß diese Anweisungen nicht durch das Interface sondern durch das I/O-ROM ermöglicht werden.

Form der Anweisung	zulässig bei Status	Zusatz-Bedingungen	ausgelöste Aktionen (Befehlsfolgen in Fettdruck)
ABORTIO 9	CA	keine	IFC, wird aktiver Controller. **[IFC]**
	$\overline{CA}$	keine	Beendet I/O-Operation.
ASSERT 9;X	alle	keine	Setzt X sofort nach CR3 und gibt X mit dem Rahmen von CR2 aus.
CLEAR 901	CA	keine	Gerät 01 wird adressiert, erhält dann SDC. **[UNL, MTA, LAG, SDC]**
CLEAR 9	CA	keine	Sendet ohne Adressierung DCL. **[DCL]**
CONTROL 9,n;X	alle	keine	Setzt X nach CRn, sobald Interface laufenden Vorgang beendet hat.
ENABLE INTR 9;X	alle	keine	Setzt X nach CR1, sobald Interface laufenden Vorgang beendet hat.
ENTER 905;X	CA	keine	Gerät 05 wird Talker, Rechner wird Listener. Eingehende Daten werden nach X gesetzt. **[UNL, TAG, SDA]**
ENTER 9;X	CA	LA	Eingehende Daten werden nach X gesetzt.
	$\overline{CA}$	LA-Status abwarten	Wartet auf Listener-Status, setzt dann eingehende Daten nach X.
HALT 9	CA	keine	Beendet I/O-Operation (empfängt Daten, gibt NRD aus, bestätigt dann letzte Daten).
HALT 9	$\overline{CA}$	keine	Beendet I/O-Operation (Sendet **[ETO]** falls Talker).
LOCAL 9	CA	keine	Sperrt Fernsteuerung. **[NRE]**
LOCAL 901	CA	keine	Gerät 01 wird Listener und mit GTL auf Handbedienung gesetzt. **[UNL, MTA, LAG, GTL]**
LOCAL LOCKOUT 9	CA	keine	LLO wird ausgegeben. **[LLO]**

Form der Anweisung	zulässig bei Status	Zusatz-Bedingungen	ausgelöste Aktionen (Befehlsfolgen in Fettdruck)
OUTPUT 905;X	CA	keine	Rechner wird Talker, Gerät 05 wird Listener, die Daten X werden ausgegeben. **[UNL, MTA, LAG]**
OUTPUT 9;X	CA	TA	Die Daten X werden ausgegeben.
	$\overline{CA}$	TA-Status abwarten	Wartet auf Talker-Status, gibt dann Daten X aus.
PASS CONTROL 915	CA	keine	Gerät 15 wird Talker, dann wird TCT ausgegeben. **[UNL, MLA, TAG, UNL, TCT]**
PASS CONTROL 9	CA	keine	Keine Adressierung, sendet TCT. **[UNL, TCT]**
PPOLL(9)	CA	keine	Sendet IDY (Identifizierung). **[IDY]**
REMOTE 9	CA	keine	Fernsteuerung wird möglich. **[REN]**
REMOTE 901	CA	keine	Fernsteuerung wird möglich, Gerät 01 wird adressiert. **[REN, UNL,MTA,LAG]**
REQUEST 9;X	$\overline{CA}$	keine	Falls X Bit 6 setzt, wird SRQ ausgegeben. Das Interface antwortet mit X auf die serielle Abfrage, die SRQ auf jeden Fall löscht.
RESET 9	SC	keine	Interface auf Einschaltzustand. IFC ordnet Schleife, Fernsteuerung wird nach Sperrung wieder möglich. **[IFC, NRE, REN]**
RESET 9	$\overline{SC}$	keine	Interface auf Einschaltzustand.
RESUME 9	CA	keine	Startet vorbereitete Übertragung mit **[SDA]**, aber wirkungslos, wenn Übertragung bereits läuft.
SEND 9;<Befehle>	CA	keine	Gibt die angegebenen Befehle aus.
SEND 9;<Daten>	alle	TA	Gibt die angegebenen Daten aus.
SPOLL(9)	CA	LA	Erfragt 1. Status-Byte des Talkers. **[SST, <Daten>, UNT]**
SPOLL(924)	CA	keine	Gerät 24 wird Talker, gibt 1. Status-Byte aus und wird entadressiert. **[UNL,MLA,TAG,SST,<Daten>,UNT]**
STATUS 9,n;X	alle	keine	Besetzt X mit dem Wert von SRn.
TRANSFER<out>	alle	keine	Wie bei OUTPUT.
TRANSFER<in>	alle	keine	Wie bei ENTER.
TRIGGER 9	CA	keine	Sendet Trigger-Befehl **[GET]**.
TRIGGER 901	CA	keine	Gerät 01 wird Listener, danach folgt Trigger-Befehl. **[UNL, MTA, LAG, GET]**

Notizen:

Funktions-Prüfung

Das nachstehend aufgelistete Programm kann in den Rechner eingegeben und zum Prüfen des IL-Interfaces benutzt werden. In den meisten Fällen wird der Ablauf des Test-Programms Aufklärung darüber geben können, ob das IL-Interface einwandfrei oder fehlerhaft arbeitet.

Achtung!
Vor Durchführen dieser Prüfung sollten Sie sich vergewissern, daß der Rechner der Serie 80 durch entsprechende Schalter-Einstellung System-Controller ist! (Hierzu IL-Abschnitt 2!)

Die Schleife muß geschlossen sein. Falls die Schleife Peripherie-Geräte enthält, müssen diese alle eingeschaltet sein.

Schalten sie den Rechner ein und prüfen Sie durch Eingabe von

IOBUFFER A$ (END LINE),

ob dem Rechner ein I/O-ROM zur Verfügung steht (es darf keine Fehlermeldung auftreten). Falls die Fehlermeldung 'BAD STMT' auftritt, schalten Sie den Rechner wieder aus, bauen Sie ein I/O-ROM ein, wie es in der Bedienungsanleitung zum ROM-Einschub HP 82936A beschrieben ist, und schalten dann den Rechner wieder ein!

Geben Sie dann das folgende Programm in gewohnter Weise ein!

```
 10 ! IL-INTERFACE-TEST
 20 OPTION BASE 0 @ INTEGER B(7),F,I,J,N,S,X,Y,Y2,Z
 30 CLEAR
 40 DISP "Test für HP-IL-Interface!" @ DISP "Mit IL-Interface besetzt:"
    @ GOSUB 210
 50 IF J#0 THEN 60 ELSE DISP "Kein HP-IL-Interface vorhanden!" @ GOTO 200
 60 DISP "Welchen Auswahl-Code hat das zu testende Interface";
 70 INPUT S
 80 GOSUB 300
 90 DISP "Wieviel Testläufe";
100 INPUT N
110 SET TIMEOUT S;500
120 ON TIMEOUT S GOTO 1030
130 RESET S
140 GOSUB 330 ! STATUS-TEST
150 GOSUB 540 ! DATEN-TEST
160 GOSUB 660 ! SRQ-TEST
170 N=N-1
180 IF N>0 THEN 130
190 DISP "Test fehlerfrei beendet!" @ OFF TIMEOUT S
200 STOP
210 J=0
220 FOR I=0 TO 7 @ B(I)=0 @ NEXT I
230 FOR I=3 TO 10
240 IF RIO(I,0)>2 THEN 280
250 STATUS I,0 ; B(I-3)
260 IF B(I-3)#5 THEN 280
270 J=I @ DISP I;
280 NEXT I
290 RETURN
300 IF S<3 OR S>10 THEN 320
310 IF B(S-3)#5 THEN 320 ELSE RETURN
320 DISP "Kein Auswahl-Code für     HP-IL-  Interface" @ GOTO 40
330 DISP "Statusregister werden gestestet!"
340 WAIT 500 ! CONTROLLER ORDNET DIE SCHLEIFE!
350 IF RIO(S,1)#3 THEN 810 ! RESET-FEHLER!
```

```
360 FOR I=0 TO 7
370 B(I) =FNS(I) ! STATUS-REGISTER WERDEN GELESEN
380 NEXT I
390 F=0
400 IF B(0)#5 OR B(1)#0 OR B(6)#0 OR B(2)>7 THEN F=1
410 IF B(5)=0 THEN GOSUB 840 @ GOTO430 ! PRUEFT, OB INTERFACE CONTROLLER IST!
420 IF B(5)=32 THEN 460 ELSE GOSUB 820 ! FEHLER IN DER SCHLEIFE!
430 IF B(7)#0 OR BIT(B(4),5) OR BIT(B(4),6) THEN F=1
440 IF F THEN 860
450 GOTO 1250 ! PROBLEM IN DER SCHLEIFE, ABER NICHT IM CONTROLLER!
460 ! AB HIER IST DAS INTERFACE CONTROLLER IN EINER GEORDNETEN SCHLEIFE!
470 IF B(2)#4 OR B(4)#128 OR B(7)>-1 THEN F=1
480 IF B(5)#32 AND B(5)#40 THEN F=1
490 IF B(2)#4 THEN F=1
500 IF B(7)<32 THEN DISP "Geräte in der Schleife: ";B(7)
510 IF F THEN 860
520 RETURN
530 !
540 DISP "Die Datenübermittlung wird      getestet!"
550 SEND S ; MLA
560 FOR J=0 TO 17
570 I=INT(2^(J-10))+(J<9)*255-INT(2^(J-1)))
580 ! GIBT 255,254,253,251,247,239,223,191,127,0,1,2,4,8,16,32,64,128 AUS!
590 CONTROL S,2 ; 0,I @ WAIT 5 ! AUSGABE EINES BYTES MIT RAHMEN!
600 X=FNS(2) @ Y=FNS(3)
610 IF X#0 AND X#1 THEN 910 ! FALSCHER RAHMEN EMPFANGEN!
620 IF Y#1 THEN 930 ! FALSCHES BYTE EMPFANGEN!
630 NEXT J
640 RETURN
650 !
660 DISP "SRQ-Interrupt wird getestet!"
670 F=0
680 ENABLE INTR S;8 ! SRQ-INTR WIRD ZUGELASSEN!
690 ON INTR D GOTO 770
700 CONTROL S,5 ; 1 ! SRQ ZU BELIEBIGEM ZEITPUNKT MOEGLICH!
710 ON TIMER# 1,30 GOTO 740
720 F=1 @ CONTROL S,2 ; 7,170 ! SENDET 170 (IDY MIT SRQ)!
730 GOTO 730 ! WARTESCHLEIFE!
740 OFF INTR S @ OFF TIMER#1
750 GOTO 950 ! INTERRUPT BLIEB AUS!
760 !
770 OFF INTR S @ OFF TIMER #1 ! IDY TRAF EIN!
780 IF NOT F THEN 970 ! INTERRUPT ZU FRUEH!
790 RETURN
800 !
810 DISP "RESET-Fehler" @ GOTO 1260
820 DISP "Schleife ist unterbrochen oder  hat mehr als einen Controller!"
830 RETURN
840 DISP "Das Interface muß für den Rest  des Testes Controller sein!"
850 RETURN
860 DISP "Die Status-Register 0-7 haben   nach RESET falsche Inhalte:"
870 FOR I=1 TO 3
880 DISP "SR"&VAL$(I)&"=";B(I),"SR"&VAL$(I+4)&"=";B(I+4),
890 NEXT I
900 GOTO 1260
910 DISP "Daten-Rahmen traf nicht ein oderwurde verändert!"
920 GOTO 1260
930 DISP "Daten trafen verändert ein:","gesendet:";I,"empfangen:";Y
940 GOTO 1260
```

```
 950 DISP "Schleife unterbrochen oder      Interrupt unwirksam!"
 960 GOTO 1260
 970 DISP "Unerwarteter Interrupt. Ein     Gerät der Schleife könnte Hilfe
 brauchen!"
 980 DISP "Test mit leerer Schleife wieder-holen, um Ursache einzugrenzen!"
 990 GOTO 1260
1000 DISP "IOP-Handshake-Fehler!"
1010 DISP "Erwartet wurde:";X,"Empfangen wurde:";Y2
1020 GOTO 1260
1030 OFF INTR S @ OFF TIMER#1 ! INTERFACE ZEITUEBERSCHREITUNG!
1040 DISP "Zeitüberschreitung beim         Interface ";S
1050 GOTO 1260
1060 ! RIO UND WIO SIND SPEZIELLE TEST-FUNKTIONEN!
1070 DEF FNW(D) ! EINGABE IN DEN XLATOR!
1080 WIO S,BIT(D,8);(D MOD 256) @ WAIT 1
1090 Y2=RIO(S,0)
1100 FNW=-5
1110 IF Y2#X THEN 1000 ! HANDSHAKE-FEHLER!
1120 IF BIT(Y2,0) THEN FNW=RIO(S,1)
1130 FN END
1140 !
1150 DEF FNS(D) ! STATUS
1160 X=8 @ Z=FNW(1)
1170 Z=RIO(S,1)
1180 Z=FNW(2)
1190 X=11 @ FNS=FNW(256+D)
1200 X=8 @ Z=FNW(4)
1210 X=0 @ Z=FNW(1)
1220 Z=FNW(0)
1230 FN END
1240 !
1250 DISP "Test abgebrochen!" @ GOTO 1270
1260 DISP "Test nicht bestanden!"
1270 BEEP @ RESET S
1280 OFF TIMEOUT S
1290 END
```

Das Programm fragt alle Interface-Auswahl-Codes nach HP-IL-Interfaces ab und gibt die damit besetzten Auswahl-Codes bekannt. Danach muß entschieden werden, welches IL-Interface wie oft getestet werden soll. Der Ablauf des Testes kann an (wiederholten) Anzeigen verfolgt werden. Der erfolgreiche Abschluß wird angezeigt.

Alle Status-Register von SR0 bis SR7 werden gelesen. Dabei sollten sich die Werte

5 0 4 0 128 32 0 X

ergeben, wobei lediglich der Inhalt von SR7 (entsprechend der Zahl der Geräte in der Schleife) zwischen 0 und 30 variieren kann.

HP-IL-Fehlermeldungen

Nummer:	Meldung:	Bedeutung bzw.Ursache:
101	XFR	Nur Warnung! Es wurde eine laufende TRANSFER-Operation unterbrochen!
110	I/O CARD	Interface-Test fehlerhaft! Geräte-Fehler möglich!
111	I/O OPER	Die gewünschte I/O-Operation ist wegen falscher Parameter oder grundsätzlich mit dem IL-Interface nicht möglich!
112	I/O ROM	Das I/O-ROM hat den Test nicht bestanden, Geräte-Fehler, der nicht vom IL-Interface verursacht wurde!
113		Der aktive Talker übernahm nicht die Controller-Funktion!
114		Die gewünschte Anweisung kann nur ausgeführt werden, wenn das Interface aktiver Controller ist!
115		Die gewünschte Anweisung kann nur ausgeführt werden, wenn das Interface aktiver Talker ist. Anlaß für diesen Fehler kann eine PRINT-Anweisung sein, die lediglich den Auswahl-Code festlegte. Beim IL-System müssen Talker und Listener vor Beginn der Datenübertragung adressiert sein!
116		Die gewünschte Anweisung kann nur ausgeführt werden, wenn das Interface aktiver Listener ist. Anlaß für den Fehler kann eine nur den Auswahl-Code angebende PLOTTER IS- bzw. PRINTER IS-Anweisung sein, durch die das Interface aktiver Talker wurde. Im IL-System müssen Talker und Listener vor Beginn der Datenübertragung adressiert sein!
117		Die gewünschte Anweisung kann nur ausgeführt werden, wenn das Interface nicht aktiver Controller ist!
118		Ein übertragungs- oder Protokoll-Fehler trat auf!
119		Der aktive Talker hat die SOT-Meldung nicht beachtet und mit der Datenübertragung nicht begonnen!
123	NO ";"	Syntax-Fehler: Es fehlt ein Semikolon in der Anweisung!
124	ISC	Es gibt kein Interface mit dem angegebenen Auswahl-Code!
125	ADDR	Geräte-Adresse fehlerhaft oder unbesetzt!
126	BUFFER	Buffer-Fehler! Angegebene String-Variable wurde nicht mit IOBUFFER deklariert oder es wurde versucht, den Buffer zu überladen oder über den Füllzeiger hinaus auszulesen!
127	NUMBER	Tritt auf, wenn bei Aufnahme von Daten die Zeichenfolge keinen Zahlenwert ergibt oder wenn ein Zahlenwert ausgegeben werden soll, der nicht zu dem mit "e" formatierten Bereich paßt!
128	EARLY TERM	Vorzeitiger Abbruch! Der Buffer ist ausgelesen, bevor die ENTER-Anweisung erfüllt ist oder es traf vor Erreichen der vereinbarten Byte-Menge ein gültiges Abschlußzeichen ein!
129	VAR TYPE	Die Art der in der Eingabe-Liste genannten Variablen paßt nicht zum dafür vorgesehenen IMAGE-Spezifikator!
130	NO TERM	Das Interface bzw. der Buffer wartet auf den vereinbarten Abschluß für die ENTER-Anweisung. Das ist (für gewöhnlich) das Zeilenvorschub-Symbol.

Beispiele für Fehler-Meldungen

Anweisung:	Wirkung:
SET I/O 9,0,0	Fehler 111, weil das Steuer-Register 0 im HP-IL-Interface nicht existiert!
REMOTE 9	Fehler 113, falls Rechner nicht aktiver Controller ist!
OUTPUT 902 ; A$	Fehler 114, falls Rechner nicht aktiver Controller ist!
OUTPUT 9 ; A$	Fehler 115, falls Rechner nicht aktiver Talker ist! Die Eigenschaft 'aktiver Controller' ist dabei belanglos!
ENTER 9 ;A$	Fehler 116, falls Rechner nicht aktiver Listener ist! Die Eigenschaft 'aktiven Controller' ist dabei belanglos!
REQUEST 9 ; <Byte>	Fehler 117, falls Rechner aktiver Controller ist! REQUEST-Meldungen dürfen nur von Geräten ausgegeben werden, die nicht als Controller aktiv sind!

Notizen:

Anhang A

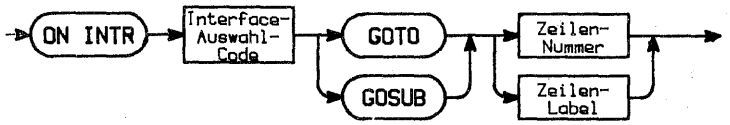
Syntax-Übersicht

Vereinbarungen zur Darstellung der Syntax

Es werden auf den folgenden Seiten zwei verschiedene Verfahren nebeneinander verwendet, um den Aufbau der I/O-Anweisungen darzustellen. Nachstehend sind die beiden Verfahren und die hierzu notwendigen Verabredungen aufgeführt:

Bildliche Darstellung

Die in abgerundete Felder eingeschlossenen Zeichen müssen genau in dieser Form verwendet werden. In den rechteckigen Feldern stehen dagegen die Namen der für die Anweisung notwendigen Parameter. Genaue Angaben über jeden der Parameter finden sich im nachfolgenden Text. Die einzelnen Elemente der Anweisung sind durch Linien verbunden. Diese dürfen nur in der durch den Pfeil am Ende jeder Linie festgelegten Richtung durchlaufen werden. Jede Kombination von Anweisungselementen, die unter Beachtung der Linien und Pfeile entstanden ist, hat eine korrekte Syntax. Wenn es für ein Anweisungselement mehrere Möglichkeiten gibt, existieren auch mehrere, quasi parallele Wege in der bildlichen Darstellung.
Diese Art der Syntax-Darstellung ist sehr anschaulich und in einigen Fällen auch besser geeignet, alle Möglichkeiten korrekt anzugeben:

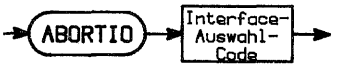


Text-Darstellung

Die Text-Darstellung ist zusätzlich angegeben und knüpft an die Darstellungsart an, wie sie in den Geräte-Beschreibungen üblich ist. Vielen Benutzern dürfte diese Darstellung recht vertraut sein:

PUNKT MATRIX	Alle Angaben als Punktmatrix müssen wie vorgeschrieben eingegeben werden.
[ ]	Alle Angaben in eckigen Klammern sind zugelassen, aber nicht zwingend vorgeschrieben.
Parameter	Angaben in kursiver Schrift sind Bezeichnungen von Parametern deren Werte einzugeben sind.
Parameter 1 *Parameter 2*	Von übereinander angegebenen Parametern ist nur einer nötig.

ABORTIO



ABORTIO *Interface-Auswahl-Code*

Beispiele für Programm-Zeilen:

```
100 ABORTIO 7
250 IF S<128 THEN ABORTIO 50
```

Parameter:

Interface-Auswahl-Code - ein numerischer Ausdruck, der einen ganzzahligen Wert zwischen 3 und 10 ergibt.

Auswirkungen:

Bei allen Interface-Arten: Unterbricht jede im Interrupt-Modus ablaufende TRANSFER-Anweisung. Wenn ABORTIO ein Interface betrifft, für das eine Zeilen-End-Verzweigung für EOT (Ende der übertragung) vereinbart ist, wird durch die Unterbrechung der laufenden TRANSFER-Anweisung auch die Verzweigung ausgelöst.

HP-IB:

System-Controller: Setzt IFC-Leitung (Interface Clear) und REN-Leitung (Remote Enable) auf 'wahr'.

Aktiver Controller (aber nicht System-Controller): Setzt ATN-Leitung auf 'wahr' und gibt MTA (Meine Talker-Adresse) aus.

Übrige Geräte: Datenübermittlung per Handshake wird unterbrochen, Geräte sind für nächste Operation bereit.

RS-232:

Alle Modem-Steuerleitungen werden abgeschaltet (Steuer-Register 2).

BCD:

Datenübermittlung per Handshake wird unterbrochen, CTL-Leitung wird auf 'falsch' gesetzt, externe Daten-Leitungen bekommen hohe Impedanz.

GPIO:

Datenübermittlung per Handshake wird unterbrochen, Steuer-Leitungen gehen auf 'falsch', die Ports A und B bekommen hohe Impedanz, die Leitungen der Ports C und D werden auf 'falsch' gesetzt.

HP-IL:

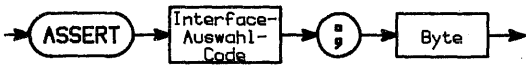
Aktiver oder System-Controller: IFC (Interface Clear) wird ausgegeben.

Übrige Geräte: Momentane Tätigkeit wird unterbrochen, Geräte sind für die nächste Operationen bereit.

Damit zusammenhängende Anweisungen:

HALT
ON EOT
RESET

ASSERT



ASSERT *Interface-Auswahl-Code* ; *Byte*

Beispiele für Programm-Zeilen:

```
100 ASSERT 7 ; 12
210 IF A1=128 THEN ASSERT S ; X
```

Parameter:

Interface-Auswahl-Code - ein numerischer Ausdruck, der einen ganzzahligen Wert zwischen 3 und 10 ergibt.

Byte - ein numerischer Ausdruck, der einen ganzzahligen Wert zwischen 0 und 255 ergibt.

Auswirkungen:

HP-IB:

Setzt sofort den Wert des Bytes in das Steuer-Register CR2 des Interfaces. Dabei bleibt aber das Bit 7 (IFC) auf jeden Fall unverändert! (Dazu ABORTIO verwenden!)

RS-232, BCD und GPIO:

Setzt sofort den Wert des Bytes in das Steuer-Register CR2.

HP-IL:

Schickt eine Meldung in die Schleife, bestehend aus den im Steuer-Register CR2 befindlichen Bits (C2, C1 und C0) und dem in der Anweisung stehenden Byte.
Diese Anweisung wirkt ähnlich wie

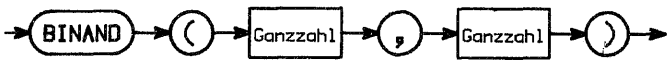
CONTROL *Interface-Auswahl-Code* , 3 ; *Byte*

mit dem Unterschied, daß ASSERT das Interface unterbricht und die Meldung ohne Rücksicht auf den Zustand der Schleife ausgibt.

Damit zusammenhängende Anweisungen:

ABORTIO
CONTROL

B I N A N D



BINAND (*Ganzzahl* , *Ganzzahl*)

Beispiele für Programm-Zeilen:

```
10 B1=BINAND(X1,15)
100 PRINT BINAND(I,N*2^3)
```

Parameter:

Ganzzahl - ein numerischer Ausdruck, der einen ganzzahligen Wert zwischen -32 768 und +32 767 ergibt.

Auswirkungen:

BINAND ist eine Funktion, die die entsprechenden Bits von zwei Ganzzahlen einzeln der UND-Verknüpfung unterwirft und das Ergebnis aller Verknüpfungen wieder als Ganzzahl ausgibt. Die Verknüpfung erfolgt gemäß der folgenden Tabelle:

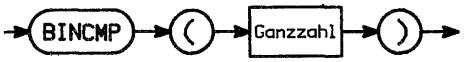
1. Argument	2. Argument	Ergebnis
0	0	0
0	1	0
1	0	0
1	1	1

Der Begriff 'Ganzzahl' wird häufig als Bezeichnung für die Argumente der Binär-Funktionen benutzt. Für das I/O-ROM ist 'Ganzzahl' eine Binärzahl mit 16 Bits aus dem Werte-Bereich von -32 768 bis +32 767.
Dies ist eine andere Festlegung als die aus dem 'Bedienungs- und Programmier-Handbuch': Dort war eine 'Ganzzahl' (Integer) eine maximal 5-stellige Zahl im Bereich von -99 999 bis +99 999. Beachten Sie bitte diesen Unterschied, um Mißverständnisse mit dem Begriff 'Ganzzahl' zu vermeiden!

Damit zusammenhängende Anweisungen:

BINCMP
BINEOR
BINIOR
BIT

B I N C M P



BINCMP *(Ganzzahl)*

Beispiele für Programm-Zeilen:

```
100 C=BINCMP(X1)
120 PRINT BINCMP(N*2^3)
```

Parameter:

Ganzzahl - ein numerischer Ausdruck, der einen ganzzahligen Wert zwischen -32 768 und +32 767 ergibt.

Auswirkungen:

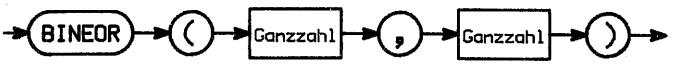
BINCMP ist eine Funktion, die die Bits einer Ganzzahl einzeln invertiert und das Ergebnis dieser Invertierungen wieder als Ganzzahl ausgibt. Wenn das Argument weniger als 16 Bits aufweist, wird ihm eine passende Zahl von Null-Bits vor der Invertierung vorangestellt.

Der Begriff 'Ganzzahl' wird häufig als Bezeichnung für die Argumente der Binär-Funktionen benutzt. Für das I/O-ROM ist 'Ganzzahl' eine Binärzahl mit 16 Bits aus dem Werte-Bereich von -32 768 bis +32 767.
Dies ist eine andere Festlegung als die aus dem 'Bedienungs- und Programmier-Handbuch': Dort war eine 'Ganzzahl' (Integer) eine maximal 5-stellige Zahl im Bereich von -99 999 bis +99 999. Beachten Sie bitte diesen Unterschied, um Mißverständnisse mit dem Begriff 'Ganzzahl' zu vermeiden!

Damit zusammenhängende Anweisungen:

BINAND
BINEOR
BINIOR
BIT

BINEOR



BINEOR *(Ganzzahl , Ganzzahl)*

Beispiele für Programm-Zeilen:

```
20 B1=BINEOR(X1,15)
100 PRINT BINEOR(I,2^N)
```

Parameter:

Ganzzahl - ein numerischer Ausdruck, der einen ganzzahligen Wert zwischen -32 768 und +32 767 ergibt.

Auswirkungen:

BINEOR ist eine Funktion, die die entsprechenden Bits von zwei Ganzzahlen einzeln der Exklusis-ODER-Verknüpfung unterwirft und das Ergebnis dieser Verknüpfungen wieder als Ganzzahl ausgibt. Die Verknüpfung erfolgt gemäß der folgenden Tabelle:

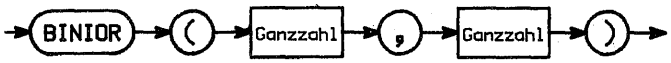
1. Argument	2. Argument	Ergebnis
0	0	0
0	1	1
1	0	1
1	1	0

Der Begriff 'Ganzzahl' wird häufig als Bezeichnung für die Argumente der Binär-Funktionen benutzt. Für das I/O-ROM ist 'Ganzzahl' eine Binärzahl mit 16 Bits aus dem Werte-Bereich von -32 768 bis +32 767.
Dies ist eine andere Festlegung als die aus dem 'Bedienungs- und Programmier-Handbuch': Dort war eine 'Ganzzahl' (Integer) eine maximal 5-stellige Zahl im Bereich von -99 999 bis +99 999. Beachten Sie bitte diesen Unterschied, um Mißverständnisse mit dem Begriff 'Ganzzahl' zu vermeiden!

Damit zusammenhängende Anweisungen:

BINAND
BINCMP
BINIOR
BIT

B I N I O R



BINIOR (*Ganzzahl* , *Ganzzahl*)

Beispiele für Programm-Zeilen:

```
20 B1=BINIOR(X1,255)
100 PRINT BINIOR(I,2^N)
```

Parameter:

Ganzzahl - ein numerischer Ausdruck, der einen ganzzahligen Wert zwischen -32 768 und +32 767 ergibt.

Auswirkungen:

BINIOR ist eine Funktion, die die entsprechenden Bits von zwei Ganzzahlen einzeln der Inklusis-ODER-Verknüpfung unterwirft und das Ergebnis dieser Verknüpfungen wieder als Ganzzahl ausgibt. Die Verknüpfung erfolgt gemäß der folgenden Tabelle:

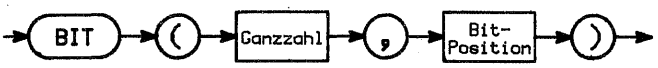
1. Argument	2. Argument	Ergebnis
0	0	0
0	1	1
1	0	1
1	1	1

Der Begriff 'Ganzzahl' wird häufig als Bezeichnung für die Argumente der Binär-Funktionen benutzt. Für das I/O-ROM ist 'Ganzzahl' eine Binärzahl mit 16 Bits aus dem Werte-Bereich von -32 768 bis +32 767.
Dies ist eine andere Festlegung als die aus dem 'Bedienungs- und Programmier-Handbuch': Dort war eine 'Ganzzahl' (Integer) eine maximal 5-stellige Zahl im Bereich von -99 999 bis +99 999. Beachten Sie bitte diesen Unterschied, um Mißverständnisse mit dem Begriff 'Ganzzahl' zu vermeiden!

Damit zusammenhängende Anweisungen:

BINAND
BINCMP
BINEOR
BIT

B I T



BIT (*Ganzzahl* , *Bit-Position*)

Beispiele für Programm-Zeilen:

```
40 Y=BIT(X3,7)
180 IF BIT(N,2^I) THEN GOTO 220
```

Parameter:

Ganzzahl - ein numerischer Ausdruck, der einen ganzzahligen Wert zwischen -32 768 und +32 767 ergibt.

Bit-Position - ein numerischer Ausdruck, der einen ganzzahligen Wert zwischen 0 und 15 ergibt. Das Bit mit der niedrigsten Wertigkeit hat die Position 0, das mit der höchsten die Position 15.

Auswirkungen:

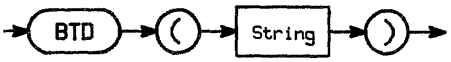
BIT ist eine Funktion, die den Zustand eines einzelnen Bits einer Ganzzahl feststellt. Das Ergebnis ist "1" ('wahr'), falls das untersuchte Bit gesetzt und "0" ('falsch'), falls es gelöscht angetroffen wurde.

Der Begriff 'Ganzzahl' wird häufig als Bezeichnung für die Argumente der Binär-Funktionen benutzt. Für das I/O-ROM ist 'Ganzzahl' eine Binärzahl mit 16 Bits aus dem Werte-Bereich von -32 768 bis +32 767.
Dies ist eine andere Festlegung als die aus dem 'Bedienungs- und Programmier-Handbuch': Dort war eine 'Ganzzahl' (Integer) eine maximal 5-stellige Zahl im Bereich von -99 999 bis +99 999. Beachten Sie bitte diesen Unterschied, um Mißverständnisse mit dem Begriff 'Ganzzahl' zu vermeiden!

Damit zusammenhängende Anweisungen:

BINAND
BINCMP
BINEOR
BINIOR

BTD



BTD (*String*)

Beispiele für Programm-Zeilen:

```
20 X=BTD(H$&L$)+A1
130 DISP BTD("11000001")
```

Parameter:

String - ein String-Ausdruck, der eine Zeichenkette mit höchstens 16 Zeichen ergibt, in der nur die Zeichen "0" und "1" zulässig sind. Diese Zeichenkette ist die binäre Darstellung einer Ganzzahl.

Auswirkungen:

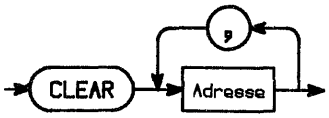
BTD ist eine Funktion, die den dezimalen Wert der als String binär dargestellten Ganzzahl ermittelt. Dabei ist das Argument immer ein String, das Resultat aber ein numerischer Wert.

Der Begriff 'Ganzzahl' wird häufig als Bezeichnung für die Argumente der Binär-Funktionen benutzt. Für das I/O-ROM ist 'Ganzzahl' eine Binärzahl mit 16 Bits aus dem Werte-Bereich von -32 768 bis +32 767.
Dies ist eine andere Festlegung als die aus dem 'Bedienungs- und Programmier-Handbuch': Dort war eine 'Ganzzahl' (Integer) eine maximal 5-stellige Zahl im Bereich von -99 999 bis +99 999. Beachten Sie bitte diesen Unterschied, um Mißverständnisse mit dem Begriff 'Ganzzahl' zu vermeiden!

Damit zusammenhängende Anweisungen:

DTB$
DTH$
DTO$
HTD
OTD

CLEAR



CLEAR *Adresse*[,*Adresse*]...

Beispiele für Programm-Zeilen:

```
60 CLEAR 3
250 CLEAR S*100+D1,S*100+D2
```

Parameter:

Adresse - ein gültiger Interface-Auswahl-Code oder eine gültige Kombination von Interface-Auswahl-Code und Primär-Adresse (I/O-ROM-Abschnitt 1: Seite ROM -4/5). Gemeinsam anzusprechen sind nur die Geräte, die über dasselbe Interface erreichbar sind.

Auswirkungen:

HP-IB und HP-IL:

Kann nur vom aktiven Controller veranlaßt werden. (HP-IB beläßt die ATN-Leitung auf 'wahr'. Wenn AN auf 'falsch' gehen soll, wird die Benutzung von RESUME empfohlen.)

Interface-Auswahl-Code allein: bewirkt Ausgabe von Device Clear (DCL).

Bei Angabe der Geräte-Adresse(n) wird UNL (alle Listener abschalten), LAD (neue Listener-Adresse(n)) und anschließend SDC (als Listener erklärte Geräte zurücksetzen) ausgegeben.

RS-232 und BCD:

Error-Meldung.

GPIO:

Interface-Auswahl-Code allein, veranlaßt das Interface RESA und RESB pulsieren zu lassen.

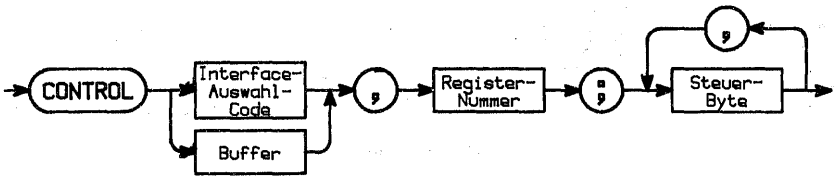
Bei gerader Geräte-Adresse, läßt das Interface RESA pulsieren.

Bei ungerader Geräte-Adresse, läßt das Interface RESB pulsieren.

Damit zusammenhängende Anweisungen:

CONTROL
SEND

CONTROL



CONTROL *Interface-Auswahl-Code* | *Buffer* , *Register-Nummer* ; *Steuer-Byte* [, *Steuer-Byte*]...

Beispiele für Programm-Zeilen:

```
10 CONTROL 9,1 ; C1,C2,C3,C4,C5,C6
50 CONTROL S,R ; C(R)
```

Parameter:

Interface-Auswahl-Code - ein numerischer Ausdruck, der einen ganzzahligen Wert zwischen 3 und 10 ergibt.

Buffer - der Name einer String-Variablen, die zuvor zum Buffer erklärt wurde.

Register-Nummer - ein numerischer Ausdruck, der einen ganzzahligen Wert zwischen 0 und 3 (für I/O-Buffer) bzw. 0 bis 23 (für Interfaces) ergibt. Es dürfen nur Register angesprochen werden, die im gewählten Interface zugänglich sind, sonst erfolgt Fehlermeldung.

Steuer-Byte - ein numerischer Ausdruck, der einen ganzzahligen Wert zwischen 0 und 255 ergibt. Die binäre Darstellung dieses Wertes wird zum Einstellen der 8 Bits des jeweiligen Steuer-Registers benutzt.

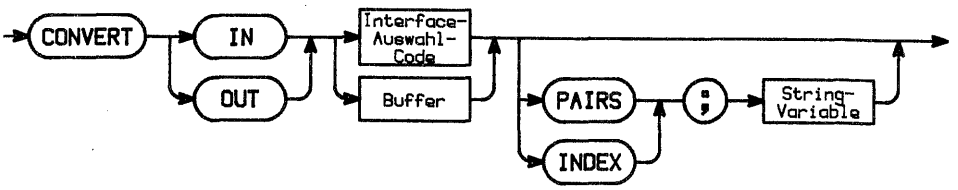
Auswirkungen:

CONTROL schreibt ein oder mehrere Bytes in die Steuer-Register des Interfaces oder des Buffers. Die Register-Nummer gibt an, in welches Register das erste Byte geschrieben wird. Wenn mehrere Steuer-Bytes vorhanden sind, werden damit aufeinander folgenden Register besetzt, beginnend mit dem angegebenen.

Damit zusammenhängende Anweisungen:

ABORTIO
ASSERT
ENABLE INTR
IOBUFFER
STATUS

CONVERT



CONVERT IN *Buffer* / OUT *Interface-Auswahl-Code* [PAIRS / INDEX ; *String-Variable*]

Beispiele für Programm-Zeilen:

```
20 CONVERT OUT 4 PAIRS ; A$
50 CONVERT IN 10 INDEX ; C$
110 CONVERT IN 7 ! Schaltet Konvertierung ab!
```

Parameter:

Interface-Auswahl-Code - ein numerischer Ausdruck, der einen ganzzahligen Wert zwischen 3 und 10 ergibt.

Buffer - der Name einer String-Variablen, die zuvor zum Buffer erklärt wurde.

String-Variable - der Name eines Strings, in dem zuvor die Übersetzungstabelle abgelegt wurde.

Auswirkungen:

Für ein bestimmtes Interface oder einen bestimmten Buffer kann für die angegebene Richtung eine Übersetzung von Zeichen ein- oder ausgeschaltet werden. Obwohl die die CONVERT-Anweisung dauernd einem Buffer oder einem Interface zugeordnet ist, kann die Übersetzung aber nur während der Ausführung einer OUTPUT- oder ENTER-Anweisung vorgenommen werden. Während der Bearbeitung einer TRANSFER- oder SEND-Anweisung ist eine Übersetzung unmöglich.

Wenn die wahlfreien Parameter fehlen, (wie im 3. Beispiel), wird die Übersetzung in der genannten Richtung für das bezeichnete Interface ausgeschaltet.

Beim Empfang von Daten (Richtung IN) werden alle eintreffenden Bytes als erstes gemäß der Übersetzungstabelle verändert. Bei der Ausgabe (Richtung OUT) werden die Bytes erst unmittelbar vor Übergabe an das empfangende Gerät oder den Buffer gemäß der Übersetzungstabelle verändert. Für ein bestimmtes Interface oder einen bestimmten Buffer können IN- und OUT-Konvertierungen gleichzeitig gelten.

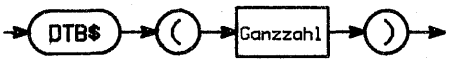
Wenn als Konvertierungs-Methode PAIRS angegeben ist, wird die Übersetzungstabelle als eine Zusammenstellung von Zeichenpaaren gewertet, von denen das jeweils erste Zeichen durch das zweite Zeichen des Paares ersetzt wird. Zeichen, die nicht als erste innerhalb eines Paares zu finden sind, durchlaufen die Konvertierung unverändert. Diese Methode wird empfohlen, wenn nur wenige Zeichen übersetzt werden müssen.

Wenn als Konvertierungs-Methode INDEX angegeben ist, wird der numerische Wert des zu übersetzenden Bytes als Zeiger für eine Übersetzungstabelle benutzt. Das durch den Zeiger markierte Byte tritt dann an die Stelle des zu übersetzenden Bytes. Wenn der numerische Wert über das Ende der Tabelle hinausweist, unterbleibt die Übersetzung. Die Zählung der Bytes in der Übersetzungstabelle beginnt mit "0". Diese Methode ist angezeigt, wenn viele Zeichen zu übersetzen sind.

Damit zusammenhängende Anweisungen:

ENTER
IOBUFFER
OUTPUT

D T B $



DTB$ (*Ganzzahl*)

Beispiele für Programm-Zeilen:

```
100 A$=DTB$(16+2*N)
200 PRINT DTB$(X1)
```

Parameter:

Ganzzahl - ein numerischer Ausdruck, der einen ganzzahligen Wert zwischen -32 768 und +32 767 ergibt.

Auswirkungen:

DTB$ist eine Funktion, die eine Ganzzahl binär mit 16 Bits darstellt. Das Argument ist ein numerischer Wert, das Resultat aber immer ein String.

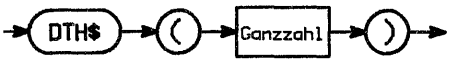
Wenn das Argument unterhalb bzw. oberhalb der obigen Bereiches liegt, liefert die Funktion "1000000000000000" bzw. "0111111111111111".

Der Begriff 'Ganzzahl' wird häufig als Bezeichnung für die Argumente der Binär-Funktionen benutzt. Für das I/O-ROM ist 'Ganzzahl' eine Binärzahl mit 16 Bits aus dem Werte-Bereich von -32 768 bis +32 767.
Dies ist eine andere Festlegung als die aus dem 'Bedienungs- und Programmier-Handbuch': Dort war eine 'Ganzzahl' (Integer) eine maximal 5-stellige Zahl im Bereich von -99 999 bis +99 999. Beachten Sie bitte diesen Unterschied, um Mißverständnisse mit dem Begriff 'Ganzzahl' zu vermeiden!

Damit zusammenhängende Anweisungen:

BTD
DTH$
DTO$
HTD
OTD

DTH$



DTH$ (*Ganzzahl*)

Beispiele für Programm-Zeilen:

```
110 B$=DTH$(32+2^N)
210 PRINT DTH$(X2)
```

Parameter:
Ganzzahl - ein numerischer Ausdruck, der einen ganzzahligen Wert zwischen -32 768 und +32 767 ergibt.

Auswirkungen:
DTH$ist eine Funktion, die eine Ganzzahl hexadezimal mit 4 Zeichen darstellt. Das Argument ist ein numerischer Wert, das Resultat aber immer ein String.

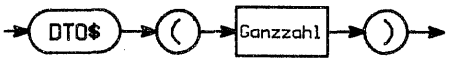
Wenn das Argument unterhalb bzw. oberhalb der obigen Bereiches liegt, liefert die Funktion "8000" bzw. "7FFF".

Der Begriff 'Ganzzahl' wird häufig als Bezeichnung für die Argumente der Binär-Funktionen benutzt. Für das I/O-ROM ist 'Ganzzahl' eine Binärzahl mit 16 Bits aus dem Werte-Bereich von -32 768 bis +32 767.
Dies ist eine andere Festlegung als die aus dem 'Bedienungs- und Programmier-Handbuch': Dort war eine 'Ganzzahl' (Integer) eine maximal 5-stellige Zahl im Bereich von -99 999 bis +99 999. Beachten Sie bitte diesen Unterschied, um Mißverständnisse mit dem Begriff 'Ganzzahl' zu vermeiden!

Damit zusammenhängende Anweisungen:

BTD
DTB$
DTO$
HTD
OTD

DTO$



DTO$ (*Ganzzahl*)

Beispiele für Programm-Zeilen:

```
120 C$=DTO$(64+2^N)
220 PRINT DTO$(X3)
```

Parameter:

Ganzzahl - ein numerischer Ausdruck, der einen ganzzahligen Wert zwischen -32 768 und +32 767 ergibt.

Auswirkungen:

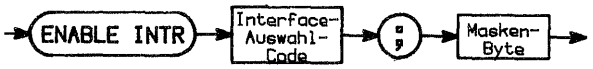
DTO$ ist eine Funktion, die eine Ganzzahl oktal darstellt. Das Argument ist immer ein numerischer Wert, das Resultat aber ein String.

Der Begriff 'Ganzzahl' wird häufig als Bezeichnung für die Argumente der Binär-Funktionen benutzt. Für das I/O-ROM ist eine 'Ganzzahl' eine Binärzahl mit 16 Bits aus dem Werte-Bereich von -32 768 bis +32 767.
Dies ist eine andere Festlegung als die aus dem 'Bedienungs- und Programmier-Handbuch': Dort war eine 'Ganzzahl' (Integer) eine maximal 5-stellige Zahl im Bereich von -99 999 bis +99 999. Beachten Sie bitte diesen Unterschied, um Mißverständnisse mit dem Begriff 'Ganzzahl' zu vermeiden!

Damit zusammenhängende Anweisungen:

BTD
DTB$
DTH$
HTD
OTD

ENABLE INTR



ENABLE INTR *Interface-Auswahl-Code* ; *Masken-Byte*

Beispiele für Programm-Zeilen:

```
10 ENABLE INTR 7 ; 8 ! SRQ-Interrupt fuer HP-IB
50 IF S>2 AND S<11 THEN ENABLE INTR S ;X
```

Parameter:

Interface-Auswahl-Code - ein numerischer Ausdruck, der einen ganzzahligen Wert zwischen 3 und 10 ergibt.

Masken-Byte - ein numerischer Ausdruck, der einen ganzzahligen Wert zwischen 0 und 255 ergibt. Die binäre Darstellung dieses Wertes wird zum Einstellen der 8 Bits des Steuer-Registers CR1 benutzt.

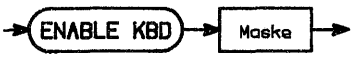
Auswirkungen:

Für das angegebene Interface führen die durch die einzelnen Bits bezeichneten Anlässe zu einem Interrupt, wenn das entsprechende Bit gesetzt ist, bzw. gesetzt wird. Das Masken-Byte wird in das Steuer-Register CR1 geschrieben. Die Bedeutung der einzelnen Bits ist von Interface zu Interface verschieden; schauen Sie bitte wegen weiterer Einzelheiten in das zum Interface gehörende Handbuch! Die gleiche Wirkung läßt sich auch mit der CONTROL-Anweisung erreichen, wenn diese auf das Steuer-Register CR1 Einfluß nimmt.

Damit zusammenhängende Anweisungen:

CONTROL
ON INTR
STATUS

ENABLE KBD



ENABLE KBD *Maske*

Beispiele für Programm-Zeilen:

```
30 ENABLE KBD 33
180 IF X THEN ENABLE KBD K1
```

Parameter:

Maske - ein numerischer Ausdruck, der einen ganzzahligen Wert zwischen 0 und 255 ergibt. Die binäre Darstellung dieses Wertes legt fest, wann bestimmte Tasten oder Tastengruppen freigegeben oder gesperrt sind.

Auswirkungen:

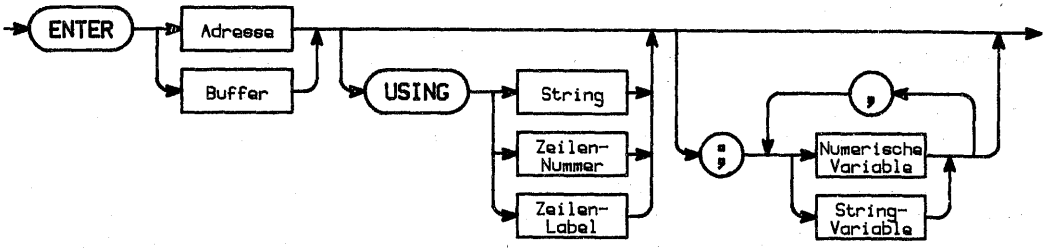
Die einzelnen Bits der Maske sind bestimmten Tasten bzw. Tastengruppen bei zwei verschiedenen Betriebsarten des Rechners zugeordnet. Wenn das entsprechende Bit gesetzt ist, sind die zugeordneten Tasten wirksam, wenn es gelöscht ist, sind die Tasten gesperrt.

Bit-Nummer	Dezimal-Wert	Betriebs-Art	Beeinflußte Tasten
7	128		RESET
6	64	Programm-	PAUSE
5	32	Ablauf	KEY LABEL u. k1 bis k8/14
4	16		Alle übrigen Tasten
3	8		RESET
2	4	Tastenfeld-	PAUSE
1	2	Eingabe	KEY LABEL u. k1 bis k8/14
0	1		Alle übrigen Tasten

Damit zusammenhängende Anweisungen:

```
INPUT
ON KEY#
```

ENTER



ENTER *Adresse* / *Buffer* [USING *String* / *Zeilen-Nummer* / *Zeilen-Label* ! Nur bei HP-86/87!][; *Variable*][, *Variable*]...]

Beispiele für Programm-Zeilen:

```
70 ENTER 701 USING LBL: ; X,Y,Z
90 ENTER C$ ; N(I),Z$
120 ENTER 3 USING 30 ; A$
250 ENTER 100*S+A USING "#,B" ; N
```

Parameter:

Adresse - ein gültiger Interface-Auswahl-Code oder eine gültige Kombination von Interface-Auswahl-Code und Primär-Adresse (I/O-ROM-Abschnitt 1: Seiten ROM-4/5).

Buffer - der Name einer String-Variablen, die zuvor zum Buffer erklärt wurde.

String - ein String-Ausdruck, der einen gültigen Satz von Image-Spezifikatoren ergibt.

Zeilen-Nummer - eine ganze Zahl zwischen 1 und 9999 (bzw. 99999 bei HP-86/87), die eine Programmzeile mit einer passend formulierten IMAGE-Anweisung bezeichnet.

Zeilen-Label - ein Name, aufgebaut nach den Regeln für 'LABEL', der zu einer passend formulierten IMAGE-Anweisung führt.
Zeilen-Label sind beim HP-83/85 nicht möglich!

Variable - numerische oder String-Variable als Ziel für die mit der ENTER-Operation aufgenommenen Daten.

Auswirkungen:

Die vom angegebenen Buffer oder Gerät eintreffenden Bytes werden aufgenommen, zu einer Zahl oder einem String zusammengesetzt und dann der angegebenen Variablen zugeordnet. Bei Benutzung einer CONVERT-Anweisung wird die Übersetzung unmittelbar nach Annahme der Bytes vom Buffer bzw. Gerät vorgenommen.

Wenn keine Formatierung angegeben ist (der USING-Teil also fehlt), wird ein Frei-Feld besetzt. Das bedeutet für eine String-Variable, daß sie die eintreffenden Bytes so lange aufnimmt, bis entweder ein Zeilenvorschub (LF) oder eine Zeilenvorschub-Wagenrücklauf-Sequenz (LF,CR) eintrifft, oder die String-Variable voll besetzt ist. Die eben genannten Abschlußzeichen werden nicht in die String-Variable gesetzt. Für eine numerische Variable bleiben bis zu 256 vorher eintreffende nicht-numerische Zeichen unberücksichtigt, Leerzeichen bleiben auch während des Aufbaus der Zahl wirkungslos. Der Aufbau der Zahl wird mit dem Eintreffen des eines Zeichens beendet, das weder ein Leerzeichen noch ein numerisches Zeichen ist.

Wenn mit USING eine Formatierung angegeben ist, werden die eintreffenden Daten in Anwendung der Image-Spezifikatoren aufgenommen. Diese können dort entweder in Anführungszeichen als String-Konstante (auch 'Literal' genannt) stehen oder es muß der Hinweis 'IMAGE' mit einer Zeilen-Nummer oder einem Zeilen-Label folgen (die letzte Form ist beim HP-83/85 nicht möglich). Unter der genannten Zeilen-Nummer oder dem Zeilen-Label ist dann die IMAGE-Anweisung zu formulieren. Weitere Informationen über Image-Spezifikatoren finden sich bei der IMAGE-Anweisung in diesem Anhang A und im I/O-ROM-Abschnitt 3.

Die ENTER-Anweisung benötigt ein Zeilen-Vorschub-Zeichen, um nach Besetzung aller Variablen der Liste die Programmzeile abschließen zu können. Hierzu genügt das Zeilenvorschub-Zeichen, mit dem die Datenzuordnung für die letzte Variable in der Liste abgeschlossen wurde. Wenn die Daten von einem Gerät kommen, das kein Zeilenvorschub-Zeichen ausgibt, bleibt das Programm in der ENTER-Zeile 'hängen', stammen die Daten dagegen aus einem Buffer, wird beim Fehlen des Zeilenvorschub-Zeichens die Fehlermeldung 'NO TERM' ausgegeben. Diese Schwierigkeit kann dadurch behoben werden, daß als erster Image-Spezifikator ein "#" verwendet wird. Für weitere Informationen über Abschlußzeichen wird auf den I/O-ROM-Abschnitt 3 verwiesen. Das 'Hängenbleiben' des Rechners kann mit den Anweisungen SET TIMEOUT und ON TIMEOUT verhindert werden.

Damit zusammenhängende Anweisungen:

CONVERT
IMAGE
IOBUFFER
ON TIMEOUT
SET TIMEOUT
TRANSFER

ERROM

ERROM

Beispiele für Programm-Zeilen:

```
30 X=ERROM
70 IF ERROM=192 THEN GOTO 100
```

Parameter:
Keine.

Auswirkungen:
ERROM ist eine Funktion, die die Identifikations-Nummer desjenigen Zusatz-ROM's ermittelt, das zuletzt an der Auslösung einer Fehler-Meldung beteiligt war. Alle Zusatz-ROM's benutzen gemeinsam die Fehler-Nummern über 100. Die Identifikations-Nummer des I/O-ROM's ist 192. Beachten Sie, daß der mit ERROM ermittelte Wert nur geändert wird, wenn ein anderes ROM eine neue Fehlermeldung auslöst.

Damit zusammenhängende Anweisungen:

ERRL
ERRN
ERRSC

ERRSC

ERRSC

Beispiele für Programm-Zeilen:

```
40 Y=ERRSC
90 IF ERRSC=7 THEN GOSUB 200
```

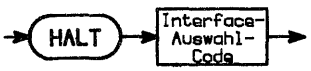
Parameter:
Keine.

Auswirkungen:
ERRSC ist eine Funktion, die den Interface-Auswahl-Code desjenigen Interfaces ermittelt, das zuletzt an der Auslösung einer I/O-Fehler-Meldung beteiligt war. Beachten Sie, daß der mit ERRSC ermittelte Wert auch beim Rücksetzen (RESET) des Interfaces nicht gelöscht wird und sich nur ändert, wenn danach in einem anderen Interface ein neuer I/O-Fehler ausgelöst wird.

Damit zusammenhängende Anweisungen:

ERRL
ERRN
ERROM

HALT



HALT *Interface-Auswahl-Code*

Beispiele für Programm-Zeilen:
```
100 HALT 7
200 HALT S1
```

Parameter:
Interface-Auswahl-Code - ein numerischer Ausdruck, der einen ganzzahligen Wert zwischen 3 und 10 ergibt.

Auswirkungen:
Bei allen Interface-Arten wird die momentan laufende I/O-Operation abgebrochen. Wenn HALT ein Interface betrifft, für das eine Zeilen-End-Verzweigung für EOT (Ende der Übertragung) vereinbart wurde, wird durch HALT eine eventuell laufende TRANSFER-Operation unterbrochen und dadurch die Verzweigung ausgelöst.

HP-IB:
Beläßt den Bus im vorgefundenen Zustand.

RS 232, BCD und GPIO:
Beeinflußt nicht die äußeren Leitungen, deren Zustand deshalb danach mit der STATUS-Anweisung abgefragt werden kann. Nach HALT kann RESET oder ABORTIO nötig sein, um die Handshake-Leitungen für die nächste Operation in einen geeigneten Zustand zu versetzen.

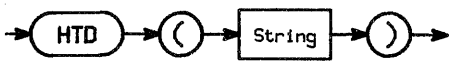
HP-IL:
Falls das Interface aktiver Controller ist und eine Datenübermittlung stattfindet, löst HALT die Ausgabe von NRFD (Not Ready For Data) durch das Interface aus. Wenn die übertragenen Daten nicht für das Interface selbst bestimmt waren, kann die Übertragung mit RESUME wieder aufgenommen werden.

Falls das Interface nicht aktiver Controller ist, verbleibt die Schleife im vorgefundenen Zustand.

Damit zusammenhängende Anweisungen:

ABORTIO
ON EOT
RESET

HTD



HTD (*String*)

Beispiele für Programm-Zeilen:

```
20 Y=HTD(H$&L$)+A2
40 DISP HTD("F73A")
```

Parameter:
String - ein String-Ausdruck, der eine Zeichenkette mit höchstens 4 Zeichen ergibt, in der nur die Zeichen "0" bis "9" und "A" bis "F" zulässig sind. Diese Zeichenkette ist die hexadezimale Darstellung einer Ganzzahl.

Auswirkungen:
HTD ist eine Funktion, die den dezimalen Wert der als String hexadezimal dargestellten Ganzzahl ermittelt. Dabei ist das Argument immer ein String, das Resultat aber ein numerischer Wert.

Der Begriff 'Ganzzahl' wird häufig als Bezeichnung für die Argumente der Binär-Funktionen benutzt. Für das I/O-ROM ist 'Ganzzahl' eine Binärzahl mit 16 Bits aus dem Werte-Bereich von -32 768 bis +32 767.
Dies ist eine andere Festlegung als die aus dem 'Bedienungs- und Programmier-Handbuch': Dort war eine 'Ganzzahl' (Integer) eine maximal 5-stellige Zahl im Bereich von -99 999 bis +99 999. Beachten Sie bitte diesen Unterschied, um Mißverständnisse mit dem Begriff 'Ganzzahl' zu vermeiden!

Damit zusammenhängende Anweisungen:

BTD
DTB$
DTH$
DTO$
OTD

IMAGE

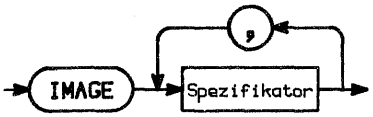


IMAGE *Spezifikator* [, *Spezifikator*]...

Beispiele für Programm-Zeilen:

```
10 LBL: IMAGE "Total =",4D.DD
100 IMAGE #,%K,2X,#K
```

Zusammenstellung der OUTPUT-Image-Spezifikatoren:

Spezifikator:	Wirkung auf die Ausgabe:
A	Ein Zeichen eines Strings wird ausgegeben
B	Eine Ganzzahl (0...255) wird als einzelnes Byte ausgegeben
C	Als Gruppierungszeichen werden Kommata ausgegeben (amerik. Stil)
D	Eine Stelle einer Zahl wird ausgegeben, führende Nullen erscheinen als Leerzeichen
E	Ein Exponent (insgesamt 5 Zeichen) wird ausgegeben
e	Ein Exponent (insgesamt 4 Zeichen) wird ausgegeben
K	Eine Variable wird im Frei-Feld-Format ausgegeben
M	Das positive Vorzeichen einer Zahl wird als Leerstelle, das negative als Minuszeichen ausgegeben
P	Als Gruppierungszeichen werden Punkte ausgegeben (europ. Stil)
R	Ein Komma wird als Dezimal-Trennzeichen ausgegebenn (europ. Stil)
S	Das Vorzeichen einer Zahl wird immer mit ausgegeben
W	Eine Ganzzahl (-32768...+32767) wird in zwei Bytes (als 16-Bit-Wort) ausgegeben
X	Eine Leerstelle wird ausgegeben
Z	Eine Stelle einer Zahl wird ausgegeben, führende Nullen werden mit ausgegeben
"..."	Ein Literal (String-Konstante) wird ausgegeben
#	Die Zeilen-End-Sequenz (üblich 'LF,CR') wird nach der letzten Position der Ausgabeliste unterdrückt
*	Eine Stelle einer Zahl wird ausgegeben, führende Nullen erscheinen als Sterne (**)
.	Ein Punkt wird als Dezimal-Trennzeichen ausgegeben (amerik. Stil)
/	Eine Zeilen-End-Sequenz (üblich 'LF,CR') wird ausgegeben.

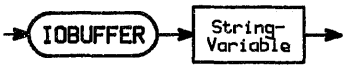
Zusammenstellung der ENTER-Image-Spezifikatoren:

Spezifikator:	Wirkung auf die Eingabe:
A	Ein Zeichen eines Strings wird erwartet
B	Ein Byte wird aufgenommen und als Ganzzahl (0...255) gewertet
C	Ein Zeichen für eine Zahl wird erwartet, Kommata werden ignoriert
D	Ein Zeichen für eine Zahl wird erwartet
E	Fünf Zeichen für eine Zahl werden erwartet
e	Vier Zeichen für eine Zahl werden erwartet
K	Eine Variable wird im Frei-Feld-Format aufgenommen
M	Ein Zeichen für eine Zahl wird erwartet
S	Ein Zeichen für eine Zahl wird erwartet
W	Zwei Bytes werden als 16-Bit-Wort aufgenommen und als Ganzzahl (-32768...+32767) gewertet
X	Ein Zeichen wird übersprungen
Z	Ein Zeichen für eine Zahl wird erwartet
#	Nach Abschluß der Eingaben wird kein Zeilen-Vorschub-Zeichen (üblich 'LF') erwartet
%	Das EOI-Signal darf das Ende der Daten anzeigen
*	Ein Zeichen für eine Zahl wird erwartet
.	Ein Zeichen für eine Zahl wird erwartet
/	Ein Zeilen-Vorschub-Zeichen (üblich 'LF') wird erwartet.

Damit zusammenhängende Anweisungen:

```
CONVERT
ENTER...USING
OUTPUT...USING
```

IOBUFFER



IOBUFFER *String-Variable*

Beispiele für Programm-Zeilen:

```
10 DIM A$[88]
20 IOBUFFER A$
30 DIM B$[80]
```

Parameter:

String-Variable - der Name einer String-Variablen, die nun zum I/O-Buffer erklärt werden soll. Die Dimensionierung muß beim HP-83/85 um 8 Bytes größer sein, als die benötigte Buffer-Größe. Beim HP-86/87 ist eine Dimensionierung genau in Buffergröße ausreichend.

Auswirkungen:

Beim HP-83/85 werden 8 Bytes der dimensionierten Länge des Strings für die Speicherung des Bufferzustandes reserviert. Es können, nach Maßgabe der freien Speicher-Kapazität beliebig viele IO-Buffer angelegt werden.
Beim HP-86/87 wird eine der 10 Buffer-Tabellen für diesen Buffer reserviert. Es können daher gleichzeitig höchstens 10 I/O-Buffer existieren.

Buffer-Lese-Zeiger:

Startwert=1. Zugänglich über die Steuer/Status-Register CR0/SR0. Die Zeichen werden bei ENTER- oder TRANSFER(aus)-Anweisungen nach folgendem Schema aus dem Buffer genommen:

1. Zeichen lesen (bleibt aber erhalten!)
2. Lese-Zeiger um 1 erhöhen.

Buffer-Füll-Zeiger:

Startwert=0. Zugänglich über die Steuer/Status-Register CR1/SR1. Die Zeichen werden bei OUTPUT oder TRANSFER(ein)-Anweisungen nach folgendem Schema in den Buffer gesetzt:

1. Füll-Zeiger um 1 erhöhen
2. Zeichen speichern.

Ausgabe-Zeiger:

Startwert=0. Abfragbar über das Status-Register SR3. Dieses Register enthält den Auswahl-Code des Interfaces, das mit einer TRANFER-Anweisung Daten aus dem Buffer entnimmt und ausgibt. SR3 nimmt den Wert 0 an, wenn keine Ausgabe mehr stattfidet.

Eingabe-Zeiger:

Startwert=0. Abfragbar über das Status-Register SR2. dieses Register enthält den Auswahl-Code des Interfaces, das mit einer TRANSFER-Anweisung Daten annimmt und in den Buffer einschreibt. SR2 nimmt den Wert 0 an, wenn von keinem Interface Daten in diesen Buffer eingeschrieben werden.

Konvertierungs-Zeiger:

Diese Zeiger sind durch Abfragen nicht zugänglich. Wenn eine CONVERT-Anweisung im Zusammenhang mit IOBUFFER ausgeführt wird, stellt das System Zeiger bereit, die auf die richtige Übersetzungs-Tabelle weisen: Diese Zeiger entstehen bei der Abarbeitung der IOBUFFER-Anweisung. Deshalb muß die CONVERT-Anweisung im Programm hinter der IOBUFFER-Anweisung stehen.

Kennzeichen eines vollen Buffers:

Ein Buffer ist voll, wenn der Füll-Zeiger einen Wert annimmt, der (beim HP-86/87) der dimensionierten Länge, bzw. (beim HP-83/85) der um 8 Bytes verringerten dimensionierten Länge entspricht. Der Versuch, darüber hinaus Daten einzulesen, erzeugt die Fehler-Meldung 'BUFFER'.

Kennzeichen eines leeren Buffers:

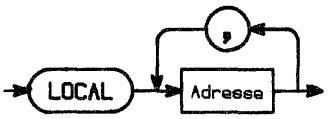
Ein Buffer ist ausgelesen (ohne daß er danach ohne Inhalt ist!), wenn der Lese-Zeiger einen Wert annimmt, der um 1 über dem Wert des Füll-Zeigers liegt. Wenn der Buffer ausgelesen ist, wird der Füll-Zeiger automatisch auf 0 und der Lese-Zeiger auf 1 gesetzt. Ausgabe- und Eingabe-Zeiger werden nach Auslesen des Buffers jedoch nicht verändert. Auch der Konvertierungszeiger wird dadurch nicht beeinflußt. Die 'alten', eben gelesenen Daten stehen unverändert im Buffer. Eine Versetzung des Füll-Zeigers auf einen passenden Wert (>0) läßt den Buffer wieder gefüllt erscheinen und sein Inhalt kann, beginnend bei der durch den Lese-Zeiger bestimmten Position, erneut gelesen werden.

Buffer-Status-Register:		**Buffer-Steuer-Register:**	
Lese-Zeiger	SR0	Lese-Zeiger	CR0
Füll-Zeiger	SR1	Füll-Zeiger	CR1
Eingabe-Zeiger	SR2		
Ausgabe-Zeiger	SR3		

Damit zusammenhängende Anweisungen:

CONTROL
CONVERT
ENTER
OUTPUT
STATUS
TRANSFER

LOCAL



LOCAL *Adresse*[,*Adresse*]...

Beispiele für Programm-Zeilen:
```
220 LOCAL 7
330 LOCAL 100*S+D1 @ RESUME 7
```

Parameter:
Adresse - ein gültiger Interface-Auswahl-Code oder eine gültige Kombination von Interface-Auswahl-Code und Primär-Adresse (I/O-ROM-Abschnitt 1: Seite ROM-4/5). Gemeinsam anzusprechen sind nur die Geräte, die über dasselbe Interface erreichbar sind.

Auswirkungen:

HP-IB:
Wenn das Interface aktiver Controller ist und die Adresse Interface-Auswahl-Code und Primär-Adresse enthält, werden diese Geräte Listener und erhalten den GTL-Befehl (Go To Local).

Wenn die Adresse nur den Interface-Auswahl-Code enthält, muß das Interface System-Controller sein. Es setzt dann die REN-Leitung auf 'falsch'.

In beiden Fällen verbleibt die ATN-Leitung auf 'wahr' und läßt sich bei Bedarf mit RESUME auf 'falsch' setzen.

Wenn ein Gerät auf Fernbedienung (REMOTE) gesetzt ist und seine äußeren Einstellorgane mit LOCAL LOCKOUT gesperrt sind, muß es die GTL-Meldung empfangen oder es muß die REN-Leitung auf 'falsch' gesetzt werden, damit eine direkte Betätigung dieses Gerätes wieder möglich wird.

RS-232, BCD und GPIO:
Error-Meldung.

HP-IL:
Kann nur vom aktiven Controller veranlaßt werden: Interface-Auswahl-Code allein veranlaßt die Ausgabe von NRE (Not Remote Enable).

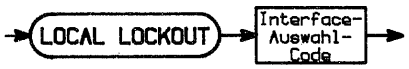
Mit Geräte-Adresse(n) wird UNL (Listener abschalten), LADn (neue Listener-Adresse(n)) und GTL (Go To Local) ausgegeben.

Wenn ein Gerät auf Fernbedienung (REMOTE) gesetzt ist und seine äußeren Einstellorgane mit LOCAL LOCKUT gesperrt sind, muß es die GTL- oder die NRE-Meldung (Not Remote Enable) empfangen, damit eine direkte Betätigung dieses Gerätes wieder möglich wird.

Damit zusammenhängende Anweisungen:

LOCAL LOCKOUT
REMOTE

LOCAL LOCKOUT



LOCAL LOCKOUT *Interface-Auswahl-Code*

Beispiele für Programm-Zeilen:

```
50 LOCAL LOCKOUT 50 @ RESUME 50
100 REMOTE 706,712 @ LOCAL LOCKOUT 7
```

Parameter:

Interface-Auswahl-Code - ein numerischer Ausdruck, der einen ganzzahligen Wert zwischen 3 und 10 ergibt.

Auswirkungen:

HP-IB und HP-IL:

Der LLO-Befehl (Local Lockout) kann nur vom aktiven Controller ausgegeben werden. Bei HP-IB verbleibt die ATN-Leitung auf 'wahr' und läßt sich bei Bedarf mit RESUME auf 'falsch' setzen.

LOCAL LOCKOUT bleibt so lange wirksam, bis entweder:
(HP-IB) die REN-Leitung (Remote Enable) auf 'falsch' gesetzt oder
(HP-IL) die NRE-Meldung (Not Remote Enable) ausgegeben wird.

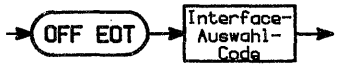
RS-232, BCD und GPIO:

Error-Meldung.

Damit zusammenhängende Anweisungen:

LOCAL
REMOTE
RESUME

OFF EOT



OFF EOT *Interface-Auswahl-Code*

Beispiele für Programm-Zeilen:

```
40 OFF EOT 3
120 IF S>128 THEN OFF EOT S1
```

Parameter:

Interface-Auswahl-Code - ein numerischer Ausdruck, der einen ganzzahligen Wert zwischen 3 und 10 ergibt.

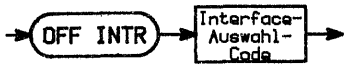
Auswirkungen:

Für das angegebene Interface werden Zeilen-End-Verzweigungen wegen EOT (Ende der Übertragung) nicht mehr ausgeführt. OFF EOT verhindert diese Verzweigungen aber nicht für dauernd. Eine erneute Aktivierung mit ON EOT bewirkt sofort eine Verzweigung, auch wenn die Übertragung bereits vorher beendet wurde.

Damit zusammenhängende Anweisungen:

OFF INTR
OFF TIMEOUT
ON EOT
ON INTR
ON TIMEOUT

OFF INTR



OFF INTR *Interface-Auswahl-Code*

Beispiele für Programm-Zeilen:

```
110 OFF INTR 7
180 IF X THEN OFF INTR S2
```

Parameter:

Interface-Auswahl-Code - ein numerischer Ausdruck, der einen ganzzahligen Wert zwischen 3 und 10 ergibt.

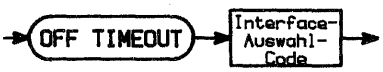
Auswirkungen:

Für das angegebene Interface werden Zeilen-End-Verzweigungen wegen INTR (Interrupt) nicht mehr ausgeführt. OFF INTR verhindert diese Verzweigungen aber nicht für dauernd. Eine erneute Aktivierung mit ON INTR bewirkt sofort eine Verzweigung, auch wenn der Interrupt bereits vorher auftrat.

Damit zusammenhängende Anweisungen:

CONTROL
ENABLE INTR
OFF EOT
OFF TIMEOUT
ON EOT
ON INTR
ON TIMEOUT

OFF TIMEOUT



OFF TIMEOUT *Interface-Auswahl-Code*

Beispiele für Programm-Zeilen:

```
50 OFF TIMEOUT S DIV 100
120 OFF TIMEOUT 7
```

Parameter:

Interface-Auswahl-Code - ein numerischer Ausdruck, der einen ganzzahligen Wert zwischen 3 und 10 ergibt.

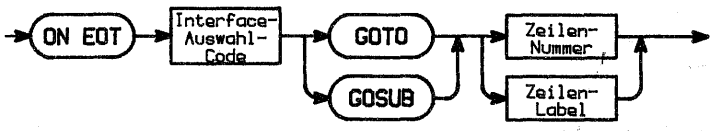
Auswirkungen:

Für das angegebene Interface werden Zeilen-End-Verzweigungen wegen Zeitüberschreitungen nicht mehr ausgeführt. OFF TIMEOUT verhindert diese Verzweigungen aber nicht für dauernd. Eine erneute Aktivierung mit ON TIMEOUT bewirkt sofort eine Verzweigung, auch wenn die Zeitüberschreitung schon vorher auftrat.

Damit zusammenhängende Anweisungen:

OFF EOT
OFF INTR
ON EOT
ON INTR
ON TIMEOUT
SET TIMEOUT

ON EOT



ON EOT *Interface-Auswahl-Code* GOTO *Zeilen-Nummer*
GOSUB *Zeilen-Label* ! Nur bei HP-86/87!

Beispiele für Programm-Zeilen:

```
40 ON EOT 7 GOTO SERVICE 7
120 ON EOT S4 GOSUB 1000
```

Parameter:

Interface-Auswahl-Code - ein numerischer Ausdruck, der einen ganzzahligen Wert zwischen 3 und 10 ergibt.

Zeilen-Nummer - eine ganze Zahl
- zwischen 1 und 9999 beim HP-83/85 -
- zwischen 1 und 99999 beim HP-86/87 -,

die eine gültige Zeile im Programm bezeichnet, an der eine passend formulierte Routine beginnt.

Zeilen-Label - ein Name, aufgebaut nach den Regeln für 'LABEL', der im Programm zu einer passend formulierten Routine führt.
Zeilen-Label sind beim HP-83/85 nicht möglich!

Auswirkungen:

ON EOT bereitet eine Zeilen-End-Verzweigung zur bezeichneten Programmstelle vor, die bei Beendigung einer Datenübertragung vom oder zum angegebenen Interface ausgeführt wird. Auch eine bereits vorher abgeschlossene Datenübertragung durch das betroffene Interface löst diesen Sprung bei Ausführung von ON EOT sofort aus. Pro Interface kann immer nur eine Beendigung der Datenübertragung für eine nachträgliche Programmverzweigung gespeichert werden.

Jedes Interface kann andere Bedingungen für das Ende der Datenübertragung haben, diese sind programmierbar. Nähere Einzelheiten darüber finden sich im Handbuch für das entsprechende Interface.

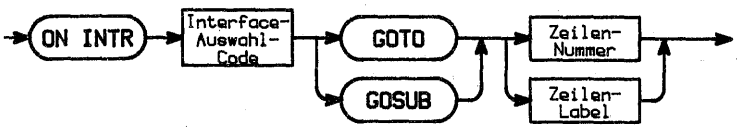
Jedes ON EOT überschreibt ältere ON EOT-Anweisungen für das gleiche Interface.

Damit zusammenhängende Anweisungen:

CONTROL
OFF EOT
OFF INTR
OFF TIMEOUT
ON INTR
ON TIMEOUT
STATUS
TRANSFER

Die Rangfolge der Verzweigungen ist im I/O-ROM-Abschnitt 9 auf Seite ROM-67 angegeben.

ON INTR



ON INTR *Interface-Auswahl-Code* GOTO *Zeilen-Nummer*
GOSUB *Zeilen-Label* ! Nur bei HP-86/87!

Beispiele für Programm-Zeilen:

```
30 ON INTR 51 GOSUB 2000
60 ON INTR 3 GOTO EXIT
150 EXIT: ABORTIO
```

Parameter:

Interface-Auswahl-Code - ein numerischer Ausdruck, der einen ganzzahligen Wert zwischen 3 und 10 ergibt.

Zeilen-Nummer - eine ganze Zahl
- zwischen 1 und 9999 beim HP-83/85 -
- zwischen 1 und 99999 beim HP-86/87 -,
die eine gültige Zeile im Programm bezeichnet, an der eine passend formulierte Routine beginnt.

Zeilen-Label - ein Name, aufgebaut nach den Regeln für 'LABEL', der im Programm zu einer passend formulierten Routine führt.
Zeilen-Label sind beim HP-83/85 nicht möglich!

Auswirkungen:

ON INTR bereitet eine Zeilen-End-Verzweigung zur bezeichneten Programmstelle vor, die beim Auftreten eines Interrupts für das angegebene Interface ausgeführt wird (siehe auch ENABLE INTR). Auch ein bereits vorher aufgetretener Interrupt für das betroffene Interface löst diesen Sprung bei Ausführen von ON INTR sofort aus. Pro Interface kann immer nur ein Interrupt für eine nachträgliche Programmverzweigung gespeichert werden.

Die Interrupt-Bedingungen können mit ENABLE INTR- oder CONTROL-Anweisungen festgelegt werden. Diese Bedingungen sind vom jeweiligen Interface abhängig. Nähere Einzelheiten darüber finden sich im Handbuch für das entsprechende Interface.

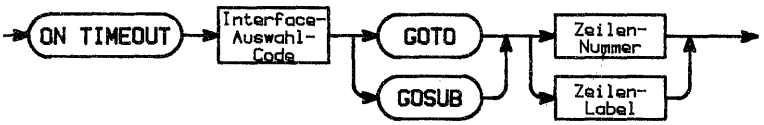
Jedes ON INTR überschreibt ältere ON INTR-Anweisungen für das gleiche Interface.

Damit zusammenhängende Anweisungen:

CONTROL
ENABLE INTR
OFF EOT
OFF INTR
OFF TIMEOUT
ON EOT
ON TIMEOUT

Die Rangfolge der Verzweigungen ist im I/O-ROM-Abschnitt 9 auf Seite ROM-67 angegeben.

ON TIMEOUT



ON TIMEOUT *Interface-Auswahl-Code* GOTO *Zeilen-Nummer*
GOSUB *Zeilen-Label* ! Nur bei HP-86/87!

Beispiele für Programm-Zeilen:

```
20 ON TIMEOUT 53 GOSUB 2500
50 ON TIMEOUT 7 GOTO INIT
200 INIT: RESET
```

Parameter:

Interface-Auswahl-Code - ein numerischer Ausdruck, der einen ganzzahligen Wert zwischen 3 und 10 ergibt.

Zeilen-Nummer - eine ganze Zahl
- zwischen 1 und 9999 beim HP-83/85 -
- zwischen 1 und 99999 beim HP-86/87 -,
die eine gültige Zeile im Programm bezeichnet, an der eine passend formulierte Routine beginnt.

Zeilen-Label - ein Name, aufgebaut nach den Regeln für 'LABEL', der im Programm zu einer passend formulierten Routine führt.
Zeilen-Label sind beim HP-83/85 nicht möglich!

Auswirkungen:

ON TIMEOUT bereitet eine Zeilen-End-Verzweigung zur bezeichneten Programmstelle vor, die beim Auftreten einer Zeitüberschreitung im angegebenen Interface ausgeführt wird (siehe auch SET TIMEOUT). Auch eine bereits vorher aufgetretene Zeitüberschreitung für das betroffene Interface löst diesen Sprung bei Ausführung von ON TIMEOUT sofort aus. Pro Interface kann immer nur eine Zeitüberschreitung für eine nachträgliche Programmverzweigung gespeichert werden.

Der Programm-Sprung wegen Zeitüberschreitung unterbleibt während der Übertragung von Daten durch eine TRANSFER-Anweisung (INTR oder FHS). ON TIMEOUT kann nur vor Beginn des Datenaustauschs mit TRANSFER wirksam werden, wenn z.B. das gewünschte Interface oder Gerät nicht adressiert werden konnte oder die Datenausgabe nicht begonnen wurde. Eine Verzweigung wegen Zeitüberschreitung wird auch beim Handshake-Verfahren nicht ausgeführt, wenn während der Übertragung eine Verzögerung auftritt.

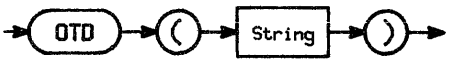
Jedes ON TIMEOUT überschreibt ältere ON TIMEOUT-Anweisungen für das gleiche Interface.

Damit zusammenhängende Anweisungen:

```
OFF EOT
OFF INTR
OFF TIMEOUT
ON EOT
ON INTR
SET TIMEOUT
```

Die Rangfolge der Verzweigungen ist im I/O-ROM-Abschnitt 9 auf Seite ROM-67 angegeben.

OTD



OTD (*String*)

Beispiele für Programm-Zeilen:

```
20 X=OTD(H$&L$)+A3
40 DISP OTD("177345")
```

Parameter:

String - ein String-Ausdruck, der eine Zeichenkette mit höchstens 6 Zeichen ergibt, in der für die letzten 5 Zeichen nur die Zeichen "0" bis "7" und für das erste von 6 Zeichen nur "0" oder "1" zulässig sind. Diese Zeichenkette ist die oktale Darstellung einer Ganzzahl.

Auswirkungen:

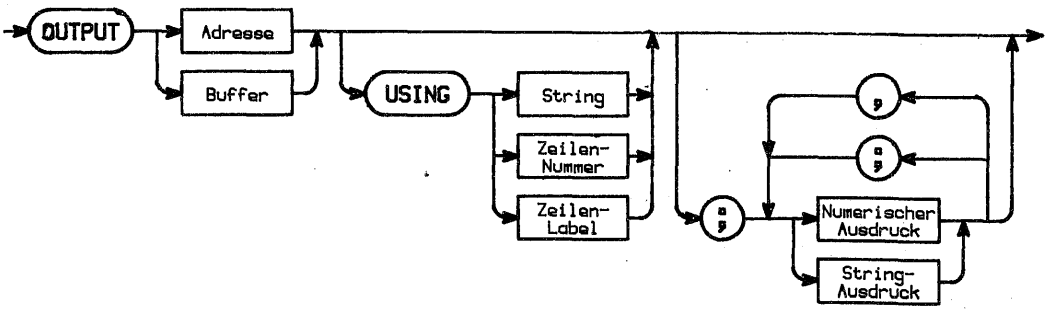
OTD ist eine Funktion, die den dezimalen Wert der als String oktal dargestellten Ganzzahl ermittelt. Dabei ist das Argument immer ein String, das Resultat aber ein numerischer Wert.

Der Begriff 'Ganzzahl' wird häufig als Bezeichnung für die Argumente der Binär-Funktionen benutzt. Für das I/O-ROM ist 'Ganzzahl' eine Binärzahl mit 16 Bits aus dem Werte-Bereich von -32 768 bis +32 767.
Dies ist eine andere Festlegung als die aus dem 'Bedienungs- und Programmier-Handbuch': Dort war eine 'Ganzzahl' (Integer) eine maximal 5-stellige Zahl im Bereich von -99 999 bis +99 999. Beachten Sie bitte diesen Unterschied, um Mißverständnisse mit dem Begriff 'Ganzzahl' zu vermeiden!

Damit zusammenhängende Anweisungen:

BTD
DTB$
DTH$
DTO$
OTD

OUTPUT



OUTPUT *Adr.,Buffer* [, *Adr.,...*][USING *String, Zeilen-N.*][;*Ausdr.*[, *Ausdr.*][;*Ausdr.*]...]
Zeilen-L. ! Nur bei HP-86/87!

Beispiele für Programm-Zeilen:

```
70 OUTPUT 701 USING Ausformat ; X,Y,Z
90 OUTPUT C$ ; N(I) ;Z$
120 OUTPUT 3 USING 30 ; A$
250 OUTPUT 100*S+A USING "#,B" ; N
```

Parameter:

Adr(esse) - ein gültiger Interface-Auswahl-Code oder eine gültige Kombination von Interface-Auswahl-Code und Primär-Adresse (I/O-ROM-Abschnitt 1: Seiten ROM-4/5). Mehrere Geräte können nur dann gemeinsam angesprochen werden, wenn sie alle über das gleiche Interface erreichbar sind.

In der OUTPUT-Anweisung sind auch die Adressen "1" und "2" zugelassen (mit "1" wird beim HP-85/87 der interne Bildschirm, mit "2" wird beim HP-85 der interne Drucker angesprochen).

Buffer - der Name einer String-Variablen, die mit IOBUFFER bereits zum Buffer erklärt wurde.

String - ein String-Ausdruck, der einen gültigen Satz von Image-Spezifikatoren ergibt.

Zeilen-N(ummer) - eine ganze Zahl
- zwischen 1 und 9999 beim HP-83/85 -
- zwischen 1 und 99999 beim HP-86/87 -,
die eine Programmzeile mit einer passend formulierten IMAGE-Anweisung bezeichnet.

Zeilen-L(abel) - ein Name, aufgebaut nach den Regeln für 'LABEL', der zu einer passend formulierten IMAGE-Anweisung führt.
Zeilen-Label sind beim HP-83/85 nicht möglich!

Ausdr(uck) - Numerischer oder String-Ausdruck, der ausgegeben werden soll. Konstanten und Variable sind als Ausdrücke zugelassen, zur Trennung können Kommata oder Semikolons gemischt benutzt werden.

Auswirkungen:

Bytes, die numerische Werte oder Strings verkörpern, werden an den bezeichneten Buffer oder an eine oder mehrere Geräte-Adressen ausgegeben. Wenn dabei eine CONVERT-Anweisung abzuarbeiten ist, wird die Übersetzung unmittelbar vor der Ausgabe der Bytes an das Interface oder den Buffer vorgenommen.

Bei fehlender USING-Anweisung und Trennung der einzelnen Variablen der Ausgabe-Liste durch Kommata, erfolgt die Ausgabe im Frei-Feld-Format. Das bedeutet für einen String, daß er linksbündig mit höchstens 20 nachfolgenden Leerstellen ausgegeben wird. Auch ein numerischer Wert wird linksbündig mit führender Leerstelle (bzw. Vorzeichen) in einem Feld ausgegeben, das einen Umfang von 11 bzw. 21 bzw. 32 Zeichen hat.

Bei fehlender USING-Anweisung und Trennung der einzelnen Variablen der Ausgabe-Liste durch Semikolon, erfolgt die Ausgabe im Kompakt-Feld-Format. Das bedeutet für einen String, daß er ohne führende und folgende Leerstellen ausgegeben wird. Numerische Werte werden mit führender Leerstelle (falls positiv) oder Minus-Zeichen (falls negativ) und weiterer Leerstelle hinter den Ziffern ausgegeben.

Mit einer USING-Anweisung erfolgt die Ausgabe unter Beachtung der vorhandenen Image-Spezifikatoren. Image-Spezifikatoren können, in Anführungszeichen eingeschlossen, oder als String-Variable in der OUTPUT-Anweisung oder in einer IMAGE-Anweisung stehen, auf die in der OUTPUT-Anweisung hingewiesen wird. Weitere Einzelheiten über die Image-Spezifikatoren sind in diesem Anhang A bei der IMAGE-Anweisung und auch im I/O-ROM-Abschnitt 3 auf den Seiten ROM-13...18 zu finden.

OUTPUT setzt eine Zeilen-End-Sequenz hinter die letzte Ausgabe. Diese Sequenz ist von der Art des Interfaces abhängig, läßt sich durch die CONTROL-Anweisung beeinflussen und besteht für gewöhnlich aus der Wagenrücklauf/Zeilenvorschub-Sequenz (CR/LF). Diese Sequenz wird unterdrückt, wenn "#" als erster Image-Spezifikator angegeben ist. Weitere Einzelheiten über die Abschluß-Spezifikatoren finden sich im I/O-ROM-Abschnitt 3 auf Seite ROM-19.

Wenn mit der OUTPUT-Anweisung ein Buffer angesprochen wird, wird eine Wagenrücklauf/Zeilenvorschub-Sequenz nach dem letzten Daten-Byte in den Buffer gesetzt, wenn dies nicht mit dem "#"-Spezifikator verhindert wird.

Damit zusammenhängende Anweisungen:

CONTROL
CONVERT
IMAGE
IOBUFFER
TRANSFER

PASS CONTROL

PASS CONTROL *Adresse*

Beispiele für Programm-Zeilen:

```
100 PASS CONTROL 100*S+D
250 PASS CONTROL 721 @ ENABLE INTR 7;32
```

Parameter:

Adresse - ein gültiger Interface-Auswahl-Code oder eine gültige Kombination von Interface-Auswahl-Code und Primär-Adresse (I/O-ROM-Abschnitt 1: Seiten ROM-4/5).

Auswirkungen:

HP-IB und HP-IL:

Das Interface muß aktiver Controller sein! PASS CONTROL übergibt die Aufgaben des aktiven Controllers an die bezeichnete Adresse.

Wenn Interface-Auswahl-Code und Primär-Adresse angegeben sind, gibt das Interface dem bezeichneten Gerät den Talker-Status und sendet anschließend TCT (Take Control) als Aufforderung an das bezeichnete Gerät. (Beim HP-IB wird anschließend die ATN-Leitung auf 'falsch' gesetzt).

Wenn nur der Interface-Auswahl-Code angegeben ist, wird die Adressierung ausgelassen und sofort TCT ausgegeben. Das übernehmende Gerät muß also schon als Talker deklariert sein, bevor die PASS CONTROL-Anweisung ausgeführt werden kann.

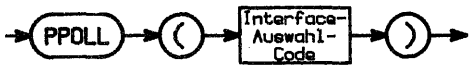
RS-232, BCD und GPIO:

Error-Meldung.

Damit zusammenhängende Anweisungen:

ABORTIO
ENABLE INTR
ON INTR
REQUEST
RESET

PPOLL



PPOLL *Interface-Auswahl-Code*

Beispiele für Programm-Zeilen:

```
310 X=PPOLL(7)
620 P9=PPOLL(S0)
```

Parameter:

Interface-Auswahl-Code - ein numerischer Ausdruck, der einen ganzzahligen Wert zwischen 3 und 10 ergibt.

Auswirkungen:

HP-IB und HP-IL:

Das Interface muß aktiver Controller sein! PPOLL holt das Ergebnis einer Parallelabfrage in den Rechner. Die IDY (Identifizierungsaufforderung) wird ausgegeben. Jedes Gerät, das auf die PPOLL-Anweisung reagieren kann, bringt das ihm zugewiesene Bit in den durch vorherige Vereinbarungen festgelegten Zustand.

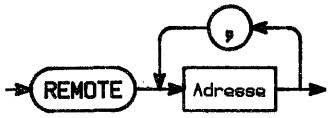
RS-232, BCD und GPIO:

Error-Meldung.

Damit zusammenhängende Anweisungen:

SPOLL

REMOTE



REMOTE *Adresse* [, *Adresse*]...

Beispiele für Programm-Zeilen:

```
50 REMOTE 720
130 REMOTE 100*S+D @ RESUME S
```

Parameter:

Adresse - ein gültiger Interface-Auswahl-Code oder eine gültige Kombination von Interface-Auswahl-Code und Primär-Adresse (I/O-ROM-Abschnitt 1: Seiten ROM-4/5). Mehrere Geräte können nur dann gemeinsam angesprochen werden, wenn sie alle über das gleiche Interface erreichbar sind.

Auswirkungen:

HP-IB:

Kann nur vom System-Controller veranlaßt werden:
Falls lediglich der Interface-Auswahl-Code angegeben ist, setzt das Interface die REN-Leitung (Remote Enable) auf 'wahr'. Die Geräte sind jedoch erst dann fernbedienbar, wenn sie als Listener angesprochen werden.

Wenn die Anweisung Geräte-Adresse(n) enthält, setzt das Interface die REN-Leitung auf 'wahr', gibt UNL (Listener abschalten) und dann die neue(n) Adresse(n) aus. Die ATN-Leitung verbleibt auf 'wahr' und läßt sich bei Bedarf mit RESUME auf 'falsch' setzen.

RS-232 und GPIO:

Error-Meldung.

BCD:

Setzt den Teil-Feld-Separator. Näheres im Handbuch für das BCD-Interface.

HP-IL:

Kann nur vom aktiven Controller veranlaßt werden:
Interface-Auswahl-Code allein veranlaßt die Ausgabe von REN (Remote Enable).

Mit Geräte-Adresse(n) wird REN (Remote Enable), UNL (Listener abschalten), und LADn (neue Listener-Adresse(n)) ausgegeben.

Damit zusammenhängende Anweisungen:

LOCAL
LOCAL LOCKOUT
RESUME

REQUEST

REQUEST *Interface-Auswahl-Code* ; *Antwort-Byte*

Beispiele für Programm-Zeilen:

```
50 REQUEST 7 ; 64+4
260 IF T>40 THEN REQUEST S ; 64+X
```

Parameter:

Interface-Auswahl-Code - ein numerischer Ausdruck, der einen ganzzahligen Wert zwischen 3 und 10 ergibt.

Antwort-Byte - ein numerischer Ausdruck, der einen ganzzahligen Wert zwischen 0 und 255 ergibt.

Auswirkungen:

HP-IB und HP-IL:

Das Interface darf nicht Controller sein! Das Antwort-Byte für eine Serienabfrage wird in der gewünschten Form gesetzt. Nur wenn das Bit 6 (Dezimalwert=64) gesetzt ist, wird das Hilfeersuchen (SRQ) vom aktiven Controller erkannt. Das Antwort-Byte wird erst nach Zulassung einer Serienabfrage (SPE) an den aktiven Controller gegeben. Die Serienabfrage des Controllers erkennt das Hilfeersuchen und löscht deshalb das Antwort-Byte im anfordernden Gerät. Der gleiche Effekt läßt sich mit REQUEST erreichen, wenn damit Bit 6 auf Null gesetzt wird.

RS-232:

Ein BREAK wird ausgegeben. Der Inhalt des BREAK ist durch das Antwort-Byte gegeben. Der Space-Zustand (0) wird entsprechend der Zahl der im Antwort-Byte genannten Zeichen gehalten, danach folgt für die Dauer von 5 Zeichen der Mark-Zustand (1).

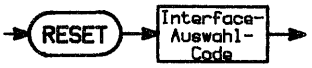
BCD und GPIO:

Error-Meldung.

Damit zusammenhängende Anweisungen:

PASS CONTROL
SPOLL

RESET



RESET *Interface-Auswahl-Code*

Beispiele für Programm-Zeilen:

```
30 RESET 7
300 IF S>128 THEN RESET S3
```

Parameter:

Interface-Auswahl-Code - ein numerischer Ausdruck, der einen ganzzahligen Wert zwischen 3 und 10 ergibt.

Auswirkungen:

RESET setzt das angegebene Interface bedingungslos auf den Einschalt-Zustand zurück. Danach läuft ein Selbst-Test ab, der beim Auftreten von Fehlern die Meldung 'ERROR 110' auslöst. Dann werden die Steuer-Register des Interfaces an den Zustand angeglichen, der der Stellung der internen Schalter auf der Interface-Platine entspricht. Wenn für ein Interface eine Zeilen-End-Verzweigung für EOT (Ende der übertragung) vereinbart wurde, löst ein Rücksetzen während einer TRANSFER-Operation diesen Sprung aus.

HP-IB:

Das Interface gibt als System-Controller IFC (Interface Clear) und REN (Remote Enable) aus.

RS-232:

Die Modem-Steuerleitungen werden abgeschaltet.

BCD:

Die Daten-Leitungen werden hochohmig, die Handshake-Leitungen werden auf 'falsch' und die I/O-Leitungen auf den Eingabe-Status gesetzt.

GPIO:

Die Ports A und B werden hochohmig, die Ports C und D werden abgeschaltet, die CTL-Leitungen gehen auf 'falsch' und OUTA und OUTB werden gesetzt, um ausgehende Daten anzuzeigen.

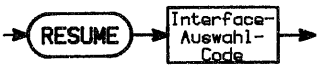
HP-IL:

Das Interface gibt als System-Controller IFC (Interface Clear), AAU (Adressen ungültig) und AAD1 (Automatische Adressierung) aus. Danach folgt NRE (Not Remote Enable) und REN (Remote Enable).

Damit zusammenhängende Anweisungen:

ABORTIO
HALT
ON EOT

RESUME



RESUME *Interface-Auswahl-Code*

Beispiele für Programm-Zeilen:

```
30 RESET 7
300 IF S>128 THEN RESET S3
```

Parameter:

Interface-Auswahl-Code - ein numerischer Ausdruck, der einen ganzzahligen Wert zwischen 3 und 10 ergibt.

Auswirkungen:

HP-IB:

Das Interface muß aktiver Controller sein (CA=1)! Die ATN-Leitung wird dann auf 'falsch' gesetzt, die nachfolgend genannten Anweisungen halten jedoch die ATN-Leitung auf 'wahr':
CLEAR, LOCAL, LOCAL LOCKOUT, REMOTE, SEND und TRIGGER.

RS-232:

Der Transmitter ist freigegeben. Weitere Einzelheiten sind im Handbuch für das Serielle Interface zu finden.

BCD und GPIO:

Error-Meldung.

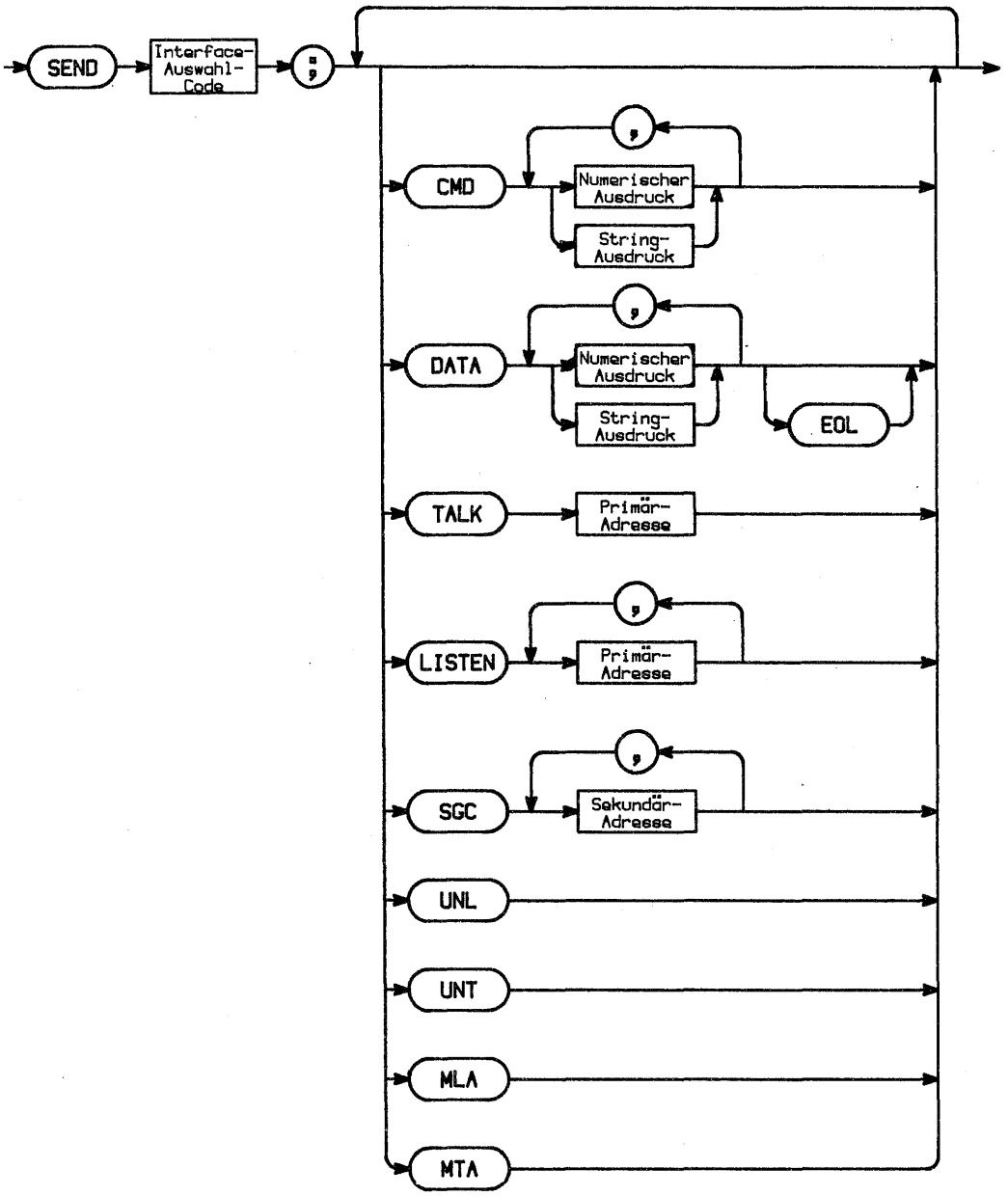
HP-IL:

Das Interface muß aktiver Controller sein. SDA (Send Data) wird ausgegeben, wenn noch keine Datenübertragung stattfindet.

Damit zusammenhängende Anweisungen:

CONTROL
HALT
SEND

SEND


SEND
Interface-
Auswahl-
Code
;
,
CMD
Numerischer
Ausdruck
String-
Ausdruck
DATA
EOL
TALK
Primär-
Adresse
LISTEN
SGC
Sekundär-
Adresse
UNL
UNT
MLA
MTA

SEND *Interface-Auswahl-Code* ;[[CMD *Liste*][DATA *Liste* [EOL]]
[TALK *Primär-Adresse*][LISTEN *Primär-Adresse* [,*Primär-Adresse*]...]
[SCG *Sekundär-Adresse* [, *Sekundär-Adresse*]...][UNL][UNT]
[MLA][MTA]...]

Beispiele für Programm-Zeilen:

```
100 SEND 7 ; CMD "U?%" DATA "Hallo"
200 SEND 7 ; CMD A$ SCG 14,18 DATA X$
300 SEND S ; MTA UNL LISTEN 6,14 CMD P,R SCG 6
```

Parameter:

Interface-Auswahl-Code - ein numerischer Ausdruck, der einen ganzzahligen Wert zwischen 3 und 10 ergibt.

Liste - eine Aufzählung von numerischen oder String-Ausdrücken, durch Kommata getrennt.

Primär-Adresse - ein numerischer Ausdruck, der einen Wert von 0 bis 31 ergibt.

Sekundär-Adresse - ein numerischer Ausdruck, der einen Wert von 0 bis 31 ergibt.

Auswirkungen:

HP-IB:

Die Befehle CMD, TALK, LISTEN, SCG, UNL, UNT, MLA und MTA können nur ausgegeben werden, wenn das Interface aktiver Controller ist. Die ATN-Leitung wird während der Ausgabe von Befehlen auf 'wahr' gehalten.
Während der Ausgabe von Daten (mit DATA) ist die ATN-Leitung auf 'falsch', auch dann, wenn im Augenblick keine Daten ausgegeben werden.

CMD: Die aus 8-Bit-Worten bestehenden aufgelisteten Befehle werden mit auf 'wahr' gesetzter ATN-Leitung ausgegeben. Primär-Befehle haben das Bit-Muster XOOCCCCC, wobei der Zustand des ersten Bits beliebig sein kann. Die fünf mit 'C' gekennzeichneten Bits verkörpern den Befehl (Dezimalwert 0 bis 31). Mit SEND CMD läßt sich, falls nötig, immer ungerade Parität bei Befehlen herstellen.

DATA: Die Werte der aufgelisteten numerischen oder String-Ausdrücke werden ausgegeben, während die ATN-Leitung auf 'falsch' liegt. Jedes gewünschte Muster aus 8 Bits kann ausgegeben werden. Wenn EOL angegeben ist, wird die im Interface in den Steuer-Registern 17 bis 23 festgelegte Zeilen-End-Sequenz nach den Daten ausgegeben.

TALK: Das Gerät mit der angegebenen Adresse (Dezimal 0 bis 31) wird zum Talker bestimmt.

LISTEN: Die Geräte mit den angegebenen Adressen (Dezimal 0 bis 31) werden zu Listenern bestimmt.

SCG: Die angegebenen Sekundär-Adressen werden ausgegeben (Gruppe der Sekundär-Befehle).

UNL: Der Befehl zum Abschalten aller Listener wird ausgegeben. (Dezimal 63); die ATN-Leitung bleibt 'wahr'!

UNT: Der Befehl zum Abschalten des Talkers wird ausgegeben. (Dezimal 95); die ATN-Leitung bleibt 'wahr'!

MLA: Die Listener-Adresse des Interfaces (My Listen Address) wird ausgegeben (im Werk auf dezimal 53 eingestellt).

MTA: Die Talker-Adresse des Interfaces (My Talk Adress) wird ausgegeben (im
Werk auf dezimal 85 eingestellt).

RS-232:

Ausgaben können nur mit DATA erfolgen:

DATA: Die Werte der aufgelisteten numerischen oder String-Ausdrücke werden ausgegeben. Wenn EOL angegeben ist, wird die im Interface in den Steuer-Registern 17 bis 23 vereinbarte Zeilen-End-Sequenz nach den Daten ausgegeben.

BCD:

Einzelheiten sind im Handbuch für das BCD-Interface zu finden!

CMD: Die Primär-Adressen 0 bis 6 setzen den Teil-Feld-Spezifikator.

DATA: Die niedrigsten vier Bits eines jeden Daten-Bytes werden ausgegeben; Steuerzeichen, Leerfelder und Kommata bleiben unberücksichtigt. Wenn EOL angegeben ist, wird das Daten-Format geprüft.

LISTEN, TALK: Die Primär-Adressen 0 bis 6 setzen den Teil-Feld-Spezifikator.

SCG: Error-Meldung.

UNL, UNT, MLA und MTA: Ohne Wirkung!

GPIO:

Einzelheiten sind im Handbuch für das GPIO-Interface zu finden!

CMD: Die Primär-Adressen 0 bis 15 bestimmen die Port-Aufteilung. Der DCL-Befehl (Device Clear) läßt RESA und RESB takten. Der SDC-Befehl (Selected Device Clear läßt, festgelegt durch die zuletzt benutzte Adresse, RESA oder RESB takten.

DATA: Die Werte der aufgelisteten numerischen oder String-Ausdrücke werden ausgegeben. Die Daten werden als Bytes (8-Bit-Worte) ausgegeben. Wenn EOL angegeben ist, wird die im Interface in den Steuer-Registern 17 bis 23 vereinbarte Zeilen-End-Sequenz nach den Daten ausgegeben.

LISTEN, TALK: Die Primär-Adressen bestimmen die Port-Aufteilung.

SCG: Error-Meldung.

UNL, UNT, MLA und MTA: Ohne Wirkung!

HP-IL:

Die Befehle CMD, TALK, LISTEN, SCG, UNL, UNT, MLA und MTA können nur ausgegeben werden, wenn das Interface aktiver Controller ist.

CMD: Die Befehle werden als 8-Bit-Worte mit Befehlsrahmen ausgegeben.

DATA: Die Bytes der aufgelisteten numerischen oder String-Ausdrücke werden als 8-Bit-Worte mit Datenrahmen ausgegeben. Wenn EOL angegeben ist, wird die im Interface in den Steuer-Registern 17 bis 23 festgelegte Zeilen-End-Sequenz nach den Daten ausgegeben.

TALK: Das Gerät mit der angegebenen Adresse (Dezimal 0 bis 31) wird zum Talker bestimmt.

LISTEN: Die Geräte mit den angegebenen Adressen (Dezimal 0 bis 31) werden zu Listenern bestimmt.

SCG: Die angegebenen Sekundär-Adressen werden mit Adreßrahmen ausgegeben.

UNL: Die Meldung zum Abschalten aller Listener wird mit Befehlsrahmen ausgegeben.

UNT: Die Meldung zum Abschalten des Talker wird mit Befehlsrahmen ausgegeben.

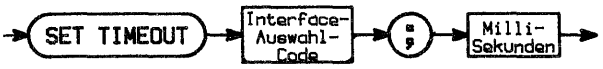
MLA: Das Interface teilt mit, daß es selbst Listener wird.

MTA: Das Interface teilt mit, daß es selbst Talker wird.

Damit zusammenhängende Anweisungen:

OUTPUT

SET TIMEOUT



SET TIMEOUT *Interface-Auswahl-Code* ; *Millisekunden*

Beispiele für Programm-Zeilen:

```
100 SET TIMEOUT 50 ; X*1000
280 ON TIMEOUT 7 GOTO 550 @ SET TIMEOUT 7 ; 4500
```

Parameter:

Interface-Auswahl-Code - ein numerischer Ausdruck, der einen ganzzahligen Wert zwischen 3 und 10 ergibt.

Millisekunden - ein numerischer Ausdruck, der einen ganzzahligen Wert zwischen 0 und 32 767 ergibt.

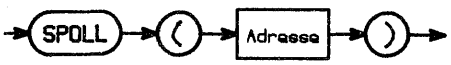
Auswirkungen:

Die in Millisekunden angegebene Zeit ist wird dem genannten Interface zur Abwicklung eines Datenaustausches im Handshake-Verfahren mit einem Peripherie-Gerät eingeräumt. Wenn eine ON TIMEOUT-Zeilen-Endverzweigung vereinbart ist, wird der Programmsprung nach Ablauf der vorgegebenen Zeit ausgeführt. So lange dieser Verzweigungs-Grund noch nicht vereinbart ist, erfolgt bei Zeotüberschreitungen keine Anzeige, jedoch bewirkt auch eine nachträgliche Aktivierung mit ON TIMEOUT bei bereits vorliegender Zeitüberschreitung einen sofortigen Programmsprung.

Damit zusammenhängende Anweisungen:

OFF TIMEOUT
ON TIMEOUT

SPOLL



SPOLL(*Adresse*)

Beispiele für Programm-Zeilen:

```
50 P=SPOLL(S4)
250 IF SPOLL(701)>63 THEN GOTO 750
```

Parameter:
Adresse - ein gültiger Interface-Auswahl-Code oder eine gültige Kombination von Interface-Auswahl-Code und Primär-Adresse (I/O-ROM-Abschnitt 1: Seiten ROM-4/5).

Auswirkungen:

HP-IB:
Das Interface (als aktiver Controller!) ermittelt den Status von einem Gerät des Bus-Systems (Serienabfrage). Das Status-Byte kann dann im Rechner untersucht werden. Nur wenn im Status-Byte das Bit 6 gesetzt ist (dezimal 64), benötigt das abgefragte Gerät Hilfe (SRQ gesetzt).

Wenn Interface-Auswahl-Code und Primär-Adresse angegeben sind, setzt das Interface die ATN-Leitung auf 'wahr', schaltet mit UNL alle Listener ab, sendet MLA (die eigene Listener-Adresse), TAD (die Talker-Adresse des abzufragenden Gerätes), gibt mit SPE die Serienabfrage frei, setzt dann die ATN-Leitung auf 'falsch', nimmt das Status-Byte auf, setzt die ATN-Leitung auf 'wahr', sperrt mit SPD die Serienabfrage und löscht mit UNT den Talker-Status.

Wenn nur der Interface-Auswahl-Code angegeben ist, wird die Adressierung ausgelassen. Das Interface setzt sofort die ATN-Leitung auf 'wahr' und gibt dann mit SPE sofort die Serienabfrage frei und verfährt dann weiter, wie im vorigen Absatz beschrieben.

RS-232, BCD und GPIO:
Fehler-Meldung.

HP-IL:
Das Interface (als aktiver Controller!) ermittelt das erste Status-Byte von einem Gerätes der Schleife (Serienabfrage).

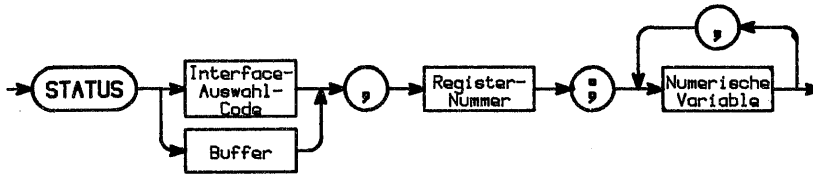
Wenn Interface-Auswahl-Code und Primär-Adresse angegeben sind, schaltet das Interface mit UNL alle Listener ab, sendet MLA (eigene Listener-Adresse), TAD (die Talker-Adresse des abzufragenden Gerätes) und SST (Send Status). Das Interface wartet dann den Eingang des Daten-Bytes, gefolgt von EOT (Ende der übertragung) ab und löscht dann mit UNT den Talker-Status.

Wenn nur der Interface-Auswahl-Code angegeben ist, wird die Adressierung ausgelassen. Das Interface gibt dann sofort SST aus und verfährt dann weiter, wie im vorigen Absatz beschrieben.

Damit zusammenhängende Anweisungen:

PPOLL

STATUS



STATUS *Interface-Auswahl-Code* , *Register-Nummer* ; *Num.Variable* [,*Num.Variable*]...
Buffer

Beispiele für Programm-Zeilen:

```
20 STATUS 7,0 ; C0,C1,C2,C3,C4
70 STATUS S1,5 ; .S1-5   (Zweites Beispiel nur für HP-86/87 zulässig!
                          Variable!)
```

Parameter:

Interface-Auswahl-Code - ein numerischer Ausdruck, der einen ganzzahligen Wert zwischen 3 und 10 ergibt.

Buffer - der Name einer String-Variablen, die zuvor zum Buffer erklärt wurde.

Register-Nummer - ein numerischer Ausdruck, der einen ganzzahligen Wert
- zwischen 0 und 15 bei *Interface-Auswahl-Code* -
- zwischen 0 und 3 bei *Buffer* -
ergibt. Das damit angesprochene Register muß existieren, sonst erfolgt Fehlermeldung 111. (Die verschiedenen Interface-Bauarten sind unterschiedlich mit Status-Registern ausgerüstet!)

Num. Variable - eine numerische Variable, die den Wert eines Status-Registers aufnimmt.

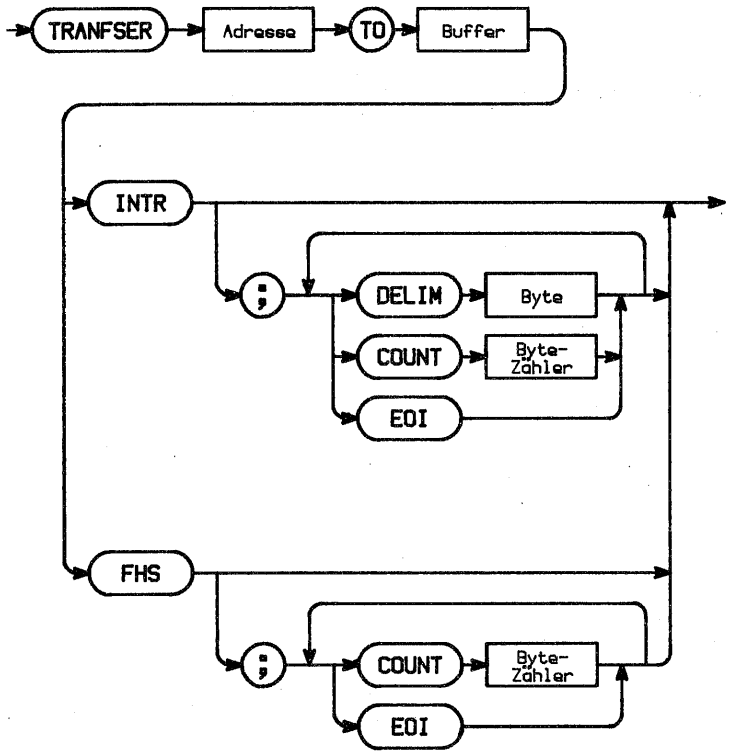
Auswirkungen:

STATUS liest ein oder mehrere Register und ordnet den oder die ermittelten Werte der oder den angegebenen numerischen Variablen zu. Die Register-Nummer nennt das Register, dessen Inhalt der ersten Variablen zugeordnet wird.
Wenn mehrere Variable bereitgestellt sind, werden diese mit den Inhalten aufeinander folgender Register besetzt, beginnend mit dem angegebenen.

Damit zusammenhängende Anweisungen:

ASSERT
CONTROL
ENABLE INTR
IOBUFFER

TRANSFER (ein)



TRANSFER *Adresse* TO *Buffer* INTR[;[COUNT *Byte-Zähler*][DELIM *Byte*][EOI]]

TRANSFER *Adresse* TO *Buffer* FHS[;[COUNT *Byte-Zähler*][EOI]]

Beispiele für Programm-Zeilen:

```
100 TRANSFER 706 to B$ INTR
200 TRANSFER 100*S+D TO B$ INTR ; COUNT 80 DELIM 10 EOI
300 TRANSFER 3 TO A$ FHS ; COUNT 16
```

Parameter:
Adresse - ein gültiger Interface-Auswahl-Code oder eine gültige Kombination von Interface-Auswahl-Code und Primär-Adresse (I/O-ROM-Abschnitt 1: Seite ROM-4/5).

Buffer - der Name einer String-Variablen, die mit IOBUFFER bereits zum Buffer erklärt wurde.

Byte-Zähler - ein numerischer Ausdruck, der einen ganzzahligen Wert zwischen 0 und 32 767 ergibt. Festlegung der maximal aufzunehmenden Bytes.

Byte - ein numerischer Ausdruck, der einen ganzzahligen Wert zwischen 0 und 255 ergibt. Bezeichnet den Dezimalwert des ASCII-Zeichens, das den Transfer beendet.

Auswirkungen:
Daten werden von der bezeichneten Adresse angenommen und unter Benutzung des Füll-Zeigers in den bezeichneten Buffer gesetzt. Die Transfer- Operation wird beendet, wenn der Buffer voll ist oder die vereinbarten Abschlußbedingungen eintreten. Auch das Interface kann eine eigene Abschlußbedingung haben: Schauen Sie deshalb bitte in das zum Interface gehörende Handbuch. Die Vereinbarung COUNT begrenzt die Zahl der aufzunehmenden Bytes. Die Vereinbarung DELIM nennt das Dezimal-Äquivalent des ASCII-Zeichens, bei dessen Auftreten die Transfer-Operation endet. Die Vereinbarung EOI löst das Ende der Transfer-Operation aus, wenn ein (vom Interface abhängiges) END-Signal auftritt (z.B. die EOI-Leitung des HP-IB-Systems). Der Abschluß bei vollem Buffer ist immer wirksam. Wenn eine Zeilen-End-Verzweigung ON EOT vereinbart ist, erfolgt der Programmsprung mit Beendigung der Übertragung.

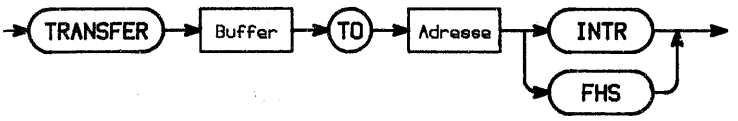
Wenn die INTR-Methode (Interrupt) vereinbart ist, darf das Interface den Rechner jedesmal unterbrechen, wenn ein neues Zeichen gelesen werden kann. Die Transfer-Operation setzt sich bis zum Abschluß fort, auch wenn das Programm inzwischen abgelaufen ist. Die Meldung 'WARNING 101' wird ausgegeben, wenn das Programm während einer noch ablaufenden Transfer-Operation anhält. Stellen Sie sicher (evtl. durch die Benutzung von RESET, HALT oder ABORTIO), daß die Transfer-Operation beendet ist, bevor Sie Änderungen im Programm vornehmen.
Mit dieser Transfer-Methode können unter idealen Bedingungen bis zu 400 Bytes pro Sekunde übertragen werden.

Wenn die FHS-Methode (Fast Handshake) vereinbart ist, sind Interface und Rechner bis zur Beendigung der Transfer-Operation für keine anderen Befehle zugänglich: nicht einmal die RESET-Taste ist in dieser Zeit wirksam! Wenn der Rechner während einer FHS-Transfer-Operation 'hängenbleibt', läßt er sich nur durch Aus- und wieder Einschalten oder durch eine besondere Interface-abhängige Operation (z.B. IFC beim HP-IB) wieder in den normalen Zustand versetzen.
Mit dieser Transfer-Methode können unter idealen Bedingungen bis zu 20 000 Bytes pro Sekunde übertragen werden.

Damit zusammenhängende Anweisungen:

ABORTIO
CONTROL
ENTER
HALT
IOBUFFER
ON EOT
RESET
STATUS

TRANSFER (aus)



TRANSFER *Buffer* TO *Adresse* INTR / FHS

Beispiele für Programm-Zeilen:

```
150 TRANSFER B$ TO 721 INTR
280 OUTPUT B$ USING "#,K" ; D$ @ TRANSFER B$ TO 9 FHS
400 ON EOT S1 GOSUB 660 @ TRANSFER X$ TO S1 INTR
```

Parameter:

Buffer - der Name einer String-Variablen, die mit IOBUFFER bereits zum Buffer erklärt wurde.

Adresse - ein gültiger Interface-Auswahl-Code oder eine gültige Kombination von Interface-Auswahl-Code und Primär-Adresse (I/O-ROM-Abschnitt 1: Seiten ROM-4/5).

Auswirkungen:

Daten werden unter Benutzung des Lese-Zeigers aus dem bezeichneten Buffer genommen und an die gewünschte Adresse übertragen. Nur wenn die Adresse Interface-Auswahl-Code und Primär-Adresse enthält, wird vor Ausgabe der Daten die Adressierung vorgenommen. Die für jedes Interface individuell programmierbare Zeilen-End-Sequenz wird nach dem letzten Byte aus dem Buffer ausgegeben. Beachten Sie, daß der Buffer durch OUTPUT-Anweisungen zusätzliche Wagenrücklauf/Zeilenvorschub-Zeichen enthalten kann. Die Transfer-Operation wird beendet, wenn der Buffer ausgelesen ist. Wenn eine Zeilen-End-Verzweigung ON EOT vereinbart ist, erfolgt der Programmsprung mit Beendigung der Übertragung.

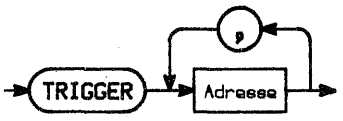
Wenn die INTR-Methode (Interrupt) vereinbart ist, darf das Interface den Rechner jedesmal unterbrechen, wenn ein neues Zeichen gesendet werden kann. Die Transfer-Operation setzt sich bis zum Abschluß fort, auch wenn das Programm inzwischen abgelaufen ist. Die Meldung 'WARNING 101' wird ausgegeben, wenn das Programm während einer noch ablaufenden Transfer-Operation anhält. Stellen Sie sicher (evtl. durch die Benutzung von RESET, HALT oder ABORTIO), daß die Transfer-Operation beendet ist, bevor Sie Änderungen im Programm vornehmen.
Mit dieser Transfer-Methode können unter idealen Bedingungen bis zu 400 Bytes pro Sekunde übertragen werden.

Wenn die FHS-Methode (Fast Handshake) vereinbart ist, sind Interface und Rechner bis zur Beendigung der Transfer-Operation für keine anderen Befehle zugänglich: nicht einmal die RESET-Taste ist in dieser Zeit wirksam! Wenn der Rechner während einer FHS-Transfer-Operation 'hängenbleibt', läßt er sich nur durch Aus- und wieder Einschalten oder durch eine besondere Interface-abhängige Operation (z.B. IFC beim HP-IB) wieder in den normalen Zustand versetzen.
Mit dieser Transfer-Methode können unter idealen Bedingungen bis zu 20 000 Bytes pro Sekunde übertragen werden.

Damit zusammenhängende Anweisungen:

ABORTIO
CONTROL
HALT
IOBUFFER
ON EOT
OUTPUT
RESET
STATUS

TRIGGER



TRIGGER *Adresse*[,*Adresse*]...

Beispiele für Programm-Zeilen:

```
70 TRIGGER 706,715 @ ENTER 706 ; V1
190 TRIGGER S1
```

Parameter:

Adresse - ein gültiger Interface-Auswahl-Code oder eine gültige Kombination von Interface-Auswahl-Code und Primär-Adresse (I/O-ROM-Abschnitt 1: Seiten ROM-4/5). Gemeinsam anzusprechen sind nur die Geräte, die über dasselbe Interface erreichbar sind.

Auswirkungen:

HP-IB:

Das Interface muß aktiver Controller sein.

Wenn die Adresse(n) Interface-Auswahl-Code und Primär-Adresse(n) enthalten, werden nur diese Geräte Listener und erhalten den GET-Befehl (Group Execute Trigger).

Wenn die Adresse nur den Interface-Auswahl-Code enthält, gibt das Interface sofort den GET-Befehl aus. Die Adressierung muß schon vorher erfolgt sein.

In beiden Fällen verbleibt die ATN-Leitung auf 'wahr' und läßt sich bei Bedarf mit RESUME auf 'falsch' setzen.

RS-232, BCD und GPIO:

Error-Meldung.

HP-IL:

Das Interface muß aktiver Controller sein.

Wenn die Adresse(n) Interface-Auswahl-Code und Primär-Adresse(n) enthalten, werden nur diese Geräte Listener und erhalten dann die GET-Meldung (Group Execute Trigger).

Wenn die Adresse nur den Interface-Auswahl-Code enthält, gibt das Interface sofort GET aus. Die Adressierung muß schon vorher erfolgt sein.

Damit zusammenhängende Anweisungen:

RESUME
SEND

I/O-ROM-Fehlermeldungen

Nummer:	Meldung:	Bedeutung bzw. Ursache:
101	XFR	Nur Warnung! Eine laufende TRANSFER-Operation wurde unterbrochen! Ein Programm sollte nie geändert geändert werden, solange noch eine Datenübertragung läuft!

Bevor Sie ein Programm ändern oder erneut starten, sollten Sie alle I/O-Aktivitäten mit einer RESET-, HALT- oder ABORTIO-Anweisung oder über das Tastenfeld mit der RESET-Taste beenden.

110	I/O CARD	Das betroffene Interface hat den Selbst-Test nicht bestanden. Das kann ein Hinweis auf einen Geräte-Fehler sein.

Mit der ERRSC-Anweisung läßt sich das fehlerhafte Interface ermitteln. Wenn die Fehlermeldung bei erneutem Einschalten des Rechners wieder erscheint, sollten Sie das Interface über Ihren Lieferanten an HP zur Prüfung geben.

111	I/O OPER	Die gewünschte I/O-Operation läßt sich mit dem betroffenen Interface nicht ausführen. Beispiele dafür sind: Abfrage bzw. Setzen von Status- bzw. Steuer-Registern, die nicht existieren. Benutzen von Primär-Adressen beim RS-232-Interface. Benutzen von Anweisungen, die für das betroffene Interface nicht zugelassen sind.

Mit der ERRL-Anweisung läßt sich die Zeile mit der störenden Anweisung ermitteln. Informieren Sie sich in der Syntax-übersicht, ob und wie diese Anweisung für das betroffene Interface anzuwenden ist. Schauen Sie auch in das zum Interface gehörende Handbuch, um weitere Einzelheiten über den richtigen Gebrauch der Anweisungen zu erfahren!

112	I/O ROM	Das I/O-ROM hat den Test nicht bestanden, Geräte-Fehler, der nicht von den beteiligten Interfaces verursacht wurde!

Wenn nach erneutem Einschalten des Rechners die gleiche Fehlermeldung wieder auftritt, sollten Sie das I/O-ROM über ihren Lieferanten an HP zur Prüfung geben.

113	<ohne>	Die Bedeutung dieser Fehlermeldung hängt davon ab, welche Interface-Bauart sie auslöste:
	HP-IB:	Die gewünschte Anweisung kann nur ausgeführt werden, wenn das IB-Interface der System-Controller ist!
	HP-IL:	Der aktive Talker übernahm nicht die Controller-Funktion!
	RS-232:	Der UART-Empfänger ist überfordert, es gingen Daten verloren, weil sie zu schnell übertragen wurden.
	BDC:	Es wurde versucht, das BCD-Interface in einen unzulässigen Zustand zu bringen.
	GPIO:	Es wurde eine ungerade Zahl von Bytes übertragen, während das GPIO-Interface 16-Bit-Worte erwartete.

Mit der ERRSC-Anweisung läßt sich das auslösende Interface ermitteln. Schauen Sie in das zum Interface gehörende Handbuch, um Einzelheiten über Ursachen der Fehler und Maßnahmen zur Abstellung zu erfahren!

Nummer:	Meldung:	Bedeutung bzw.Ursache:
114	<ohne>	Die Bedeutung dieser Fehlermeldung hängt davon ab, welche Interface-Bauart sie auslöste:
	HP-IB und HP-IL:	Die gewünschte Anweisung kann nur ausgeführt werden, wenn das Interface aktiver Controller ist!
	RS-232:	Der aufnehmende Buffer ist überfordert, Daten gingen verloren.
	BDC:	Port 10 momentan nicht verfügbar.
	GPIO:	Ein FHS-Transfer wurde durch STO abgebrochen.

Mit der ERRSC-Anweisung läßt sich das auslösende Interface ermitteln. Schauen Sie in das zum Interface gehörende Handbuch, um Einzelheiten über Ursachen der Fehler und Maßnahmen zur Abstellung zu erfahren!

115	<ohne>	Die Bedeutung dieser Fehlermeldung hängt davon ab, welche Interface-Bauart sie auslöste:
	HP-IB und HP-IL:	Die gewünschte Anweisung kann nur ausgeführt werden, wenn das Interface aktiver Talker ist!
	RS-232:	Die automatische Abschaltung wurde ausgelöst.
	BDC:	Ein FHS-Transfer wurde durch FLGB abgebrochen.
	GPIO:	Das GPIO-Interface erlaubt keine OUTPUT-Operation über die Ports A oder B.

Mit der ERRSC-Anweisung läßt sich das auslösende Interface ermitteln. Schauen Sie in das zum Interface gehörende Handbuch, um Einzelheiten über Ursachen der Fehler und Maßnahmen zur Abstellung zu erfahren!

116	<ohne>	Die Bedeutung dieser Fehlermeldung hängt davon ab, welche Interface-Bauart sie auslöste:
	HP-IB und HP-IL:	Die gewünschte Anweisung kann nur ausgeführt werden, wenn das Interface aktiver Listener ist!
	RS-232:	Benutzt diese Fehlermeldung nicht.
	BDC:	Die übertragungsrichtung ist nicht klar definiert.
	GPIO:	Die übertragung kann wegen gestörter Handshake-Leitungen nicht beginnen.

Mit der ERRSC-Anweisung läßt sich das auslösende Interface ermitteln. Schauen Sie in das zum Interface gehörende Handbuch, um Einzelheiten über Ursachen der Fehler und Maßnahmen zur Abstellung zu erfahren!

117	<ohne>	Die Bedeutung dieser Fehlermeldung hängt davon ab, welche Interface-Bauart sie auslöste:
	HP-IB und HP-IL:	Die gewünschte Anweisung kann nur ausgeführt werden, wenn das Interface nicht aktiver Controller ist!
	RS-232:	Benutzt diese Fehlermeldung nicht!
	BDC:	Der Interface-Befehl betrifft ein nicht vorhandenes Feld.
	GPIO:	Benutzt diese Fehlermeldung nicht.

Mit der ERRSC-Anweisung läßt sich das auslösende Interface ermitteln. Schauen Sie in das zum Interface gehörende Handbuch, um Einzelheiten über Ursachen der Fehler und Maßnahmen zur Abstellung zu erfahren!

Nummer:	Meldung:	Bedeutung bzw.Ursache:
118	<ohne>	Die Bedeutung dieser Fehlermeldung hängt davon ab, welche Interface-Bauart sie auslöste:
	HP-IB:	Benutzt diese Fehlermeldung nicht!
	HP-IL:	Ein Protokoll- oder Übertragungsfehler ist aufgetreten!
	RS-232:	Benutzt diese Fehlermeldung nicht!
	BDC:	Die CTL-Leitung ist gestört, Operations-Beginn unmöglich!
	GPIO:	Benutzt diese Fehlermeldung nicht!

Mit der ERRSC-Anweisung läßt sich das auslösende Interface ermitteln. Schauen Sie in das zum Interface gehörende Handbuch, um Einzelheiten über Ursachen der Fehler und Maßnahmen zur Abstellung zu erfahren!

119	<ohne>	Die Bedeutung dieser Fehlermeldung hängt davon ab, welche Interface-Bauart sie auslöste:
	HP-IB:	Benutzt diese Fehlermeldung nicht!
	HP-IL:	Der aktive Talker hat die SDT-Meldung nicht beachtet und mit der Datenübertragung nicht begonnen!
	RS-232:	Benutzt diese Fehlermeldung nicht!
	BDC:	Das Datenformat paßt nicht zur Betriebsart des Interfaces!
	GPIO:	Benutzt diese Fehlermeldung nicht!

Mit der ERRSC-Anweisung läßt sich das auslösende Interface ermitteln. Schauen Sie in das zum Interface gehörende Handbuch, um Einzelheiten über Ursachen der Fehler und Maßnahmen zur Abstellung zu erfahren!

123	NO ";"	Syntax-Fehler! Es fehlt ein Semikolon in der Anweisung!

Ermitteln Sie mit der ERRL-Anweisung die fehlerhafte Programmzeile und setzen Sie das Semikolon an der richtigen Stelle ein!

124	ISC	Der angegebene Interface-Auswahl-Code ist entweder nicht zulässig oder nicht mit einem Interface besetzt. Auswahl-Codes dürfen allgemein Werte von 3 bis 10 haben. Die Codes 1 (Bildschirm) und 2 (interner Drucker) sind nur in OUTPUT-Anweisungen zulässig!

Achten Sie auf korrekte Interface-Auswahl-Codes. Besondere Aufmerksamkeit erfordern Variable, mit denen Auswahl-Codes ausgedrückt werden! Schauen Sie auch nach, ob das benötigte Interface korrekt eingesetzt ist und ob der Auswahl-Code inzwischen nicht geändert wurde!

125	ADDR	Die angegebene Primär-Adresse ist nicht zulässig. Erlaubt sind Adressen im Bereich zwischen 0 und 31, jedoch können nicht alle Interfaces den ganzen Bereich benutzen.

Achten Sie auf korrekte Primär-Adressen. Besondere Aufmerksamkeit erfordern Variable, mit denen Primär-Adressen ausgedrückt werden!

Nummer:	Meldung:	Bedeutung bzw.Ursache:
126	BUFFER	Diese Fehlermeldung kann vier Ursachen haben: Die Deklarierung der Stringvariablen mit IOBUFFER fehlt! Der mit ENTER angesprochene BUFFER ist bereits ausgelesen! Der mit OUTPUT angesprochene Buffer ist bereits voll! TRANSFER (aus) betrifft einen bereits ausgelesenen Buffer!

Überzeugen Sie sich, daß die notwendige IOBUFFER-Anweisung im Programm hinter der DIM-Anweisung der Stringvariablen, aber vor der Erst-Benutzung des Buffers steht! Der Inhalt eines Buffers kann immer mit einfachen PRINT/DISP-Anweisungen (mit dem Buffer-Namen) ermittelt werden, weitere Informationen geben Füll- und Lesezeiger, deren Stellung mit der STATUS-Anweisung zu erfahren sind.

127	NUMBER	Für den auszugebenden Zahlenwert ist ein zweistelliger Exponent (Format "e") vorgeschrieben, nötig ist aber ein Exponent mit drei Stellen, oder die aufgenommene Zeichen ergeben keinen gültigen Zahlenwert.

Wenn der Fehler durch eine Ausgabe ausgelöst wurde, passen einfach Zahlenwert und Formatierung nicht zusammen. Aufgenommene Daten lösen die Fehlermeldung aus,

wenn mehr als 255 nichtnumerische Daten vor dem Zahlenwert empfangen wurden oder,

wenn innerhalb einer auf begrenzte Zeichenzahl formatierten Zahlendarstellung unerwartete Leerstellen stehen oder,

wenn Satz-Zeichen und Zahlen-Symbolen so vermischt sind, daß sich numerisch kein Sinn ergibt.

128	EARLY TERM	Die einen Buffer auslesende ENTER-Anweisung konnte nicht abgeschlossen werden, weil vor Erfüllung der Eingabe-Liste der Buffer ausgelesen war. Bei der Aufnahme von Daten ist vor Erreichen der verabredeten Zeichenanzahl ein gültiges Abschlußzeichen eingetroffen.

Die Fehler-Ursache läßt sich durch Prüfen der ENTER-Liste, der IMAGE-Anweisungen bzw. durch Analyse der einlaufenden Daten finden.

129	VAR TYPE	Die Art der in der ENTER-Liste stehenden Variablen (String bzw. numerisch) paßt nicht zur IMAGE-Anweisung.

Die Fehler-Ursache läßt sich durch Prüfen der ENTER-Liste und der IMAGE-Anweisungen ermitteln.

130	NO TERM	Das für den Abschluß einer ENTER-Anweisung notwendige Abschluß-Zeichen wurde vom Interface bzw. Buffer nicht gegeben. Für gewöhnlich wird das Zeilenvorschub-Symbol als Abschluß-Zeichen benutzt.

Die Fehler-Ursache läßt sich durch Prüfen der ENTER-Liste, der IMAGE-Anweisungen bzw. durch Analyse der einlaufenden Daten finden.

Notizen:

Der Zeichensatz der Rechner der Serie 80

dez.	CHR	okt.	hex.	binär	dez.	CHR	okt.	hex.	binär
0	◀	000	00	00000000	64	@	100	40	01000000
1	¿	001	01	00000001	65	A	101	41	01000001
2	¾	002	02	00000010	66	B	102	42	01000010
3	¼	003	03	00000011	67	C	103	43	01000011
4	α	004	04	00000100	68	D	104	44	01000100
5	ß	005	05	00000101	69	E	105	45	01000101
6	Γ	006	06	00000110	70	F	106	46	01000110
7	π	007	07	00000111	71	G	107	47	01000111
8	Δ	010	08	00001000	72	H	110	48	01001000
9	σ	011	09	00001001	73	I	111	49	01001001
10	↑	012	0A	00001010	74	J	112	4A	01001010
11	↓	013	0B	00001011	75	K	113	4B	01001011
12	µ	014	0C	00001100	76	L	114	4C	01001100
13		015	0D	00001101	77	M	115	4D	01001101
14	τ	016	0E	00001110	78	N	116	4E	01001110
15	Φ	017	0F	00001111	79	O	117	4F	01001111
16	Θ	020	10	00010000	80	P	120	50	01010000
17	Ω	021	11	00010001	81	Q	121	51	01010001
18	δ	022	12	00010010	82	R	122	52	01010010
19	Ä	023	13	00010011	83	S	123	53	01010011
20	ä	024	14	00010100	84	T	124	54	01010100
21	Ö	025	15	00010101	85	U	125	55	01010101
22	ö	026	16	00010110	86	V	126	56	01010110
23	Ü	027	17	00010111	87	W	127	57	01010111
24	ü	030	18	00011000	88	X	130	58	01011000
25	Ü	031	19	00011001	89	Y	131	59	01011001
26	Ü	032	1A	00011010	90	Z	132	5A	01011010
27	Æ	033	1B	00011011	91	Ä	133	5B	01011011
28	æ	034	1C	00011100	92	Ö	134	5C	01011100
29	ø	035	1D	00011101	93	Å	135	5D	01011101
30	£	036	1E	00011110	94	Ü	136	5E	01011110
31	▒	037	1F	00011111	95	_	137	5F	01011111
32		040	20	00100000	96	`	140	60	01100000
33	!	041	21	00100001	97	a	141	61	01100001
34	"	042	22	00100010	98	b	142	62	01100010
35	#	043	23	00100011	99	c	143	63	01100011
36	$	044	24	00100100	100	d	144	64	01100100
37	%	045	25	00100101	101	e	145	65	01100101
38	&	046	26	00100110	102	f	146	66	01100110
39	'	047	27	00100111	103	g	147	67	01100111
40	(	050	28	00101000	104	h	150	68	01101000
41	)	051	29	00101001	105	i	151	69	01101001
42	*	052	2A	00101010	106	j	152	6A	01101010
43	+	053	2B	00101011	107	k	153	6B	01101011
44	,	054	2C	00101100	108	l	154	6C	01101100
45	-	055	2D	00101101	109	m	155	6D	01101101
46	.	056	2E	00101110	110	n	156	6E	01101110
47	/	057	2F	00101111	111	o	157	6F	01101111
48	0	060	30	00110000	112	p	160	70	01110000
49	1	061	31	00110001	113	q	161	71	01110001
50	2	062	32	00110010	114	r	162	72	01110010
51	3	063	33	00110011	115	s	163	73	01110011
52	4	064	34	00110100	116	t	164	74	01110100
53	5	065	35	00110101	117	u	165	75	01110101
54	6	066	36	00110110	118	v	166	76	01110110
55	7	067	37	00110111	119	w	167	77	01110111
56	8	070	38	00111000	120	x	170	78	01111000
57	9	071	39	00111001	121	y	171	79	01111001
58	:	072	3A	00111010	122	z	172	7A	01111010
59	;	073	3B	00111011	123	ä	173	7B	01111011
60	<	074	3C	00111100	124	ö	174	7C	01111100
61	=	075	3D	00111101	125	→	175	7D	01111101
62	>	076	3E	00111110	126	ü	176	7E	01111110
63	?	077	3F	00111111	127	▒	177	7F	01111111

dez.	CHR	okt.	hex.	binär	dez.	CHR	okt.	hex.	binär
128		200	80	10000000	192		300	C0	11000000
129		201	81	10000001	193		301	C1	11000001
130		202	82	10000010	194		302	C2	11000010
131		203	83	10000011	195		303	C3	11000011
132		204	84	10000100	196		304	C4	11000100
133		205	85	10000101	197		305	C5	11000101
134		206	86	10000110	198		306	C6	11000110
135		207	87	10000111	199		307	C7	11000111
136		210	88	10001000	200		310	C8	11001000
137		211	89	10001001	201		311	C9	11001001
138		212	8A	10001010	202		312	CA	11001010
139		213	8B	10001011	203		313	CB	11001011
140		214	8C	10001100	204		314	CC	11001100
141		215	8D	10001101	205		315	CD	11001101
142		216	8E	10001110	206		316	CE	11001110
143		217	8F	10001111	207		317	CF	11001111
144		220	90	10010000	208		320	D0	11010000
145		221	91	10010001	209		321	D1	11010001
146		222	92	10010010	210		322	D2	11010010
147		223	93	10010011	211		323	D3	11010011
148		224	94	10010100	212		324	D4	11010100
149		225	95	10010101	213		325	D5	11010101
150		226	96	10010110	214		326	D6	11010110
151		227	97	10010111	215		327	D7	11010111
152		230	98	10011000	216		330	D8	11011000
153		231	99	10011001	217		331	D9	11011001
154		232	9A	10011010	218		332	DA	11011010
155		233	9B	10011011	219		333	DB	11011011
156		234	9C	10011100	220		334	DC	11011100
157		235	9D	10011101	221		335	DD	11011101
158		236	9E	10011110	222		336	DE	11011110
159		237	9F	10011111	223		337	DF	11011111
160		240	A0	10100000	224		340	E0	11100000
161		241	A1	10100001	225		341	E1	11100001
162		242	A2	10100010	226		342	E2	11100010
163		243	A3	10100011	227		343	E3	11100011
164		244	A4	10100100	228		344	E4	11100100
165		245	A5	10100101	229		345	E5	11100101
166		246	A6	10100110	230		346	E6	11100110
167		247	A7	10100111	231		347	E7	11100111
168		250	A8	10101000	232		350	E8	11101000
169		251	A9	10101001	233		351	E9	11101001
170		252	AA	10101010	234		352	EA	11101010
171		253	AB	10101011	235		353	EB	11101011
172		254	AC	10101100	236		354	EC	11101100
173		255	AD	10101101	237		355	ED	11101101
174		256	AE	10101110	238		356	EE	11101110
175		257	AF	10101111	239		357	EF	11101111
176		260	B0	10110000	240		360	F0	11110000
177		261	B1	10110001	241		361	F1	11110001
178		262	B2	10110010	242		362	F2	11110010
179		263	B3	10110011	243		363	F3	11110011
180		264	B4	10110100	244		364	F4	11110100
181		265	B5	10110101	245		365	F5	11110101
182		266	B6	10110110	246		366	F6	11110110
183		267	B7	10110111	247		367	F7	11110111
184		270	B8	10111000	248		370	F8	11111000
185		271	B9	10111001	249		371	F9	11111001
186		272	BA	10111010	250		372	FA	11111010
187		273	BB	10111011	251		373	FB	11111011
188		274	BC	10111100	252		374	FC	11111100
189		275	BD	10111101	253		375	FD	11111101
190		276	BE	10111110	254		376	FE	11111110
191		277	BF	10111111	255		377	FF	11111111

COMPUTERBÜCHER IM HELDERMANN VERLAG

Albers, K.: HP-41 Barcodes mit dem HP-IL-System.
1986, 320 Seiten, ISBN 3-88538-804-9, DM 44.-
Der Barcodelesestift ist ein preiswertes, dabei jedoch sehr effizientes Zubehörteil für das Einlesen von Daten oder zum Programmieren. Barcodes (BCs) sind die kostengünstigste Methode der externen Datenspeicherung und deren Massenverbreitung. Über HP-41 BCs, ihre oft verblüffend ergiebige Anwendung, Herstellung und ihren Aufbau ist wenig dokumentiert. Mit diesem Buch schließt der Autor die Lücke.
Das Buch behandelt ausführlich Vor- und Nachteile von BCs und Lesestift, gibt eine Übersicht über alle BC-Typen des 41-er Systems und erläutert den grundsätzlichen Aufbau und das Kodierungsschema. Die Herstellung von BCs auf dem IL-Thermodrucker, dem ThinkJet-Drucker und dem Plotter 7470A mit vielen nützlichen Hinweisen für einwandfreie Ergebnisse, sowie die dafür notwendigen oder wünschenswerten Geräte werden eingehend beschrieben. 2-Byte-BCs der ASCII-Zeichen 0-127, 3-Byte-Ausführung 0-255, Alpha- und synthetische Textzeilen-BCs zum Programmieren, überraschend einfache und schnelle Verfahren für 'load bytes' ohne Hilfsprogramm und BCs von einfachen oder längeren 'BLDSPEC'-Sonderzeichen geben rationelle Arbeitshilfen. Die Drucker-Modi 8-Bit/ESCAPE werden genau erklärt. BCs von Zahlen, trickreiches Programmieren großer Zahlen, Kurzformexponenten, numerische und Alphafolgedaten für sequentielle Registereingabe werden ausführlich beschrieben. BCs von 1- und 2-Byte-Tastenfeldfunktionen und von nicht programmierbaren Rechnerfunktionen, INDirekt, besondere Funktionen und deren BCs werden erschöpfend behandelt. Aus einer speziellen Art BCs bewirkt 'SNAP 2', der BC-Byteschnapper die verblüffend problemlose Lesestifteingabe jedes synthetischen Befehls an beliebiger Programmstelle ohne Tastenbelegungen. XROM-Funktionen-BCs, Mehrbyte-BCs von Tastenfeldfunktionen mit und ohne Argument, alle nicht programmierbaren Rechner- und XROM-Funktionen mit Argument (SIZE, DEL, ASN, PRP usw.) sowie BCs von LBL/GTO/XEQ-"Alpha" und für ein simuliertes synthetisches Tastenfeld für Sofortausführung jeder Art synthetischer Befehle ohne Tastenzuweisung erweitern die Arbeitsmöglichkeiten ganz erheblich. Der Aufbau von Programm-BCs und deren Herstellung auf dem Thermodrucker ohne das Plottermodul, auf dem ThinkJet-Drucker und dem Plotter wird in allen Einzelheiten erklärt. Eine große Zahl von Tabellen der Funktions- und Alphazeichen 0-127, 0-255 (replace und append), für das synthetische Programmieren, für ein simuliertes, synthetisches Tastenfeld sowie der XROM-Befehl aller zur Zeit verfügbaren Systemmodule sind willkommene Arbeitserleichterungen auch für den Anwender, der keinen Drucker besitzt. Es werden eine Menge praktischer Anwendungen gezeigt für das synthetische Programmieren, zur Datenmasseneinlesung für Schriftfahnen oder Querdruck, zur Herstellung beliebiger Textzeilen und zum Umgang mit speziellen LBL, GTO bzw. XEQ-Befehlen.
Jedes Kapitel enthält komfortable Programme zur BC-Herstellung oder für Umrechnungen in Form von Listing und BCs, insgesamt über 90 Programme! Nach Möglichkeit sind alle Programme dreifach vorhanden: 1. zur Herstellung nur mit Rechner und Thermodrucker ohne Systemmodule; 2. zusätzlich mit dem XF-Modul und 3. außerdem mit dem Plottermodul. Durch 1. sind die Möglichkeiten der BC-Herstellung weitestgehend auch dem Benutzer erschlossen, der nicht über Zusatzmodule vefügt. In einem Anhang werden nicht HP-BCs wie der Europäische Artikelnummern-BC und einige anderen Typen vorgestellt. Weiterhin wird ein Ausblick auf die zukünftige Entwicklung gegeben.
Wer die Möglichkeiten seines Lesestiftes mit dem HP-41 System optimal nutzen und selbst BCs anfertigen will, braucht dieses leicht verständlich geschriebene Buch. Es wird ihm bald zur unentbehrlichen Arbeitshilfe werden.

Dearing, J. S.: Tricks, Tips und Routinen für Taschenrechner der Serie HP-41
1984, 220 Seiten, ISBN 3-88538-801-4, DM 36.-
Dieses Buch enthält über 350 Routinen und Tips für den HP-41. Von elementaren Tricks und pfiffigen Abkürzungen bis hin zu komplizierten synthetischen Programmen aus dem PPC-Modul und umfangreichen Druck-Routinen ist alles vertreten, was die große Gemeinde der HP-41-Benutzer im Laufe von Jahren herausgefunden und zusammengetragen hat. Dem Autor ist die herkulische Leistung zu verdanken, die Fülle dieser Ergebnisse gesammelt, gesichtet und geordnet zu haben. Der Übersetzer, Heinz Dalkowski, hat in Zusammenarbeit mit dem Autor das Original erweitert und in einem Nachwort die Funktionen des neuesten Rechners aus der Serie HP-41, des HP-41CX, beschrieben. Ein umfangreiches Stichwortverzeichnis von über 1000 Einträgen erschließt die Sammlung lückenlos und läßt zielsicher auffinden, worüber man sich zu informieren wünscht. Man spart so manche Programmierstunde und kommt jedem merkwürdigen Verhalten des HP-41 auf die Spur.

Horn, J.: HP-71 Basic - leicht gemacht
1986, ca. 200 Seiten, ISBN 3-88538-807-3, ca. DM 40.-
Der HP-71 bietet bei erstaunlich geringen Abmessungen Anwendungmöglichkeiten in einer Vielfalt, wie sie bisher bei Rechnern dieser Größe unbekannt war. Es ist klar, daß zu einem so komfortablen Rechner auch entsprechend umfangreiche Handbücher gehören. Deren Lektüre stellt aber, nicht nur für Computer-Neulinge, eine teils harte, teils zu gehaltvolle Kost dar, deren Genuß durchaus zu Schwierigkeiten führen kann. In diesem Vergleich spielt nun das Buch "HP-71 Basic - leicht gemacht" die Rolle des Kräuter-Likörs, der Völlegefühl und Verstimmungen beseitigt und damit weiteren Appetit auf das ganze Menue macht. Der betont lockere Stil des Originaltextes wurde auch in der deutschen Bearbeitung voll beibehalten, weil der Umgang mit einem Rechner vorwiegend Freude und möglichst wenig Streß erzeugen soll.

Jarett, K.: Erweiterte Funktionen des HP-41 - leicht gemacht
1986, 240 Seiten, ISBN 3-88538-803-0, DM 44.-
Das Buch beschreibt die Eigenschaften des erweiterten Speichers und der X-Funktionen, mit denen der HP-41C/CV aufgerüstet werden kann und die im HP-41CX fest eingebaut sind. Da das Bedienungshandbuch zum Umgang mit X-Modulen nur knappe Hinweise gibt, war ein Buch nötig, das die Fähigkeiten der X-Funktionen und des HP-41CX vollständig beschreibt. Der Autor, ein führender Experte des HP-41 Systems und Pionier der synthetischen Programmierung, hat dieses Buch in seinem unnachahmlichen Stil - einfach, klar und doch präzise - geschrieben. Die vorliegende deutsche Ausgabe von Heinz Dalkowski wurde darüber hinaus in vielerlei Hinsicht inhaltlich ergänzt. Nach der Lektüre kann der Leser die X-Funktionen wirkungsvoll einsetzen und, wenn er Kenntnisse in synthetischer Programmierung besitzt, die Kraft dieser Kunst mit der der X-Funktionen verbinden. Insbesondere bekommt er über 30 ausgereifte Programme, die von den führenden Experten auf dem Gebiet stammen, an die Hand, darunter einen umfangreichen Text-Editor für den HP-41C/CV, ein Adressenverzeichnis-Programm, eine Simulation des HP-16, mathematische Programme, Programme zum Übertragen von Textdateien auf Magnetkarten, sowie - in Verbindung mit synthetischer Programmierung - Programme zum Reparieren fehlerhaften Verhaltens einiger speziellen Vorgänge beim Einsatz von X-Funktionen (Betriebssystemfehler). Sämtliche Programme sind als BCs abgedruckt, insgesamt 4181 Bytes! Wer aus seinem Kraftpaket herausholen will, was drin steckt, benötigt dieses Buch.

Jarett, K.: Synthetisches Programmieren auf dem HP-41 - leicht gemacht
1985, 170 Seiten, eine Quick Reference Card beiliegend, ISBN 3-88538-802-2, DM 40.-
Der Autor wendet sich an HP-41 Benutzer, denen die gründliche aber anspruchsvolle Darstellung der synthetischen Programmierung durch W. C. Wickes ("Synthetische Programmierung auf dem HP-41C/CV", Heldermann Verlag) Schwierigkeiten bereitet. Es gelingt ihm, in ausführlicher Weise einen Zugang zu diesem Gebiet zu bereiten, der zugleich abwechslungsreich und spannend ist. Andererseits berücksichtigt er neueste Erweiterungen des HP-41 durch den Hersteller und bringt Programme, welche die Funktionen aus dem X-Modul und dem Time-Modul beinhalten, wodurch viele synthetische Programme - z. Bsp. das Tastenzuweisungsprogramm - wesentlich kürzer und schneller werden. Für Kundige ist dieses Buch somit eine spielend lesbare Ergänzung des Buches von Wickes, für Anfänger ist es die ideale Einführung.
Die Übersetzung besorgte in gewohnt fachmännischer Weise Heinz Dalkowski.

Meschede, W.: Plotten und Drucken auf dem HP-41 Thermodrucker
1985, 176 Seiten, ISBN 3-88538-805-7, DM 36.-
Dieses Buch enthält 18 Programme zum Plotten auf den HP-41 Thermodruckern und zusätzlich 224 Zeichen in drei Darstellungsformen und alle benötigten Zahlencodes für BLDSPEC und die synthetische Programmierung der Zeichen. Jedem Programm ist ein Programm-Ablaufplan und eine ausführliche Bedienungsanleitung beigegeben, damit sowohl reine Programm-Anwender, als auch Selbst-Programmierer voll auf ihre Kosten kommen. Durch diese Programme stellen selbst logarithmische Skalierung, die Darstellung mehrerer Funktionen in einem Schaubild, hochauflösendes Plotten, sowie Histogramme (Balkendiagramme) keine Probleme mehr dar und durch kleine Änderungen ist die Anpassung an ganz spezielle Anforderungen leicht möglich. Hat man ein Programm längere Zeit nicht mehr benutzt, ermöglichen die Kurzanleitungen im Anhang ein schnelles Rekapitulieren der sehr einfachen und komfortablen Bedienung. Zum Schluß werden dann in Kapitel 7 alle Wünsche nach ganz speziellen Zeichen für die selbst programmierte Ausgabe - einschließlich Querschrift - erfüllt.
Alle Programme sind als BCs abgedruckt, insgesamt 6755 Bytes! Wer graphische Ausgaben oder mehr als nur die einfachen 127 Zeichen benötigt, kann an diesem Buch nicht vorübergehen.

Stroinski, W.: Zusammenfassung der Bedienungs- und Programmier-Anleitungen für I/O-ROM, IB- und IL-Interface der HP-Rechner der Serie 80
1986, 296 Seiten, ISBN 3-88538-806-5, DM 58.-
Handbücher in deutscher Sprache sind meist nur für den Computer und die wichtigsten Peripherie-Geräte (Drucker, Monitor, Massenspeicher) erhältlich, für die "seltenere" Peripherie, durch deren Anschluß der Computer erst seine volle Wirksamkeit erlangt, sind die Beschreibungen häufig nur in englischer Sprache lieferbar. Unabhängig von der Qualität der vorhandenen Sprachkenntnisse ist das Verstehen dieser neuen Materie sicherlich einfacher, wenn die Beschreibungen in deutscher Sprache vorliegen.
Für die Hewlett-Packard-Rechner der 80er Serie (HP-83/85 bzw. HP-86/87) wird mit diesem Buch über das I/O-ROM und die Interfaces für HP-IB (IEEE 488 bzw. IEC 625) und HP-IL (Interface-Loop) dieser Mangel behoben. Es enthält die Übersetzungen der nachfolgend genannten HP-Druckschriften in korrigierter Form:

I/O-ROM, Owner's Manual, 00087-90121, Jan. 83
HP-IB Interface, Owner's Manual, 82937-90017, Jan. 82
HP-IL Interface, Owner's Manual, 82938-90001, Jan. 82.

Der Vorläufer dieser Handbücher (I/O Programming Guide, 00085-90142) wurde ebenfalls berücksichtigt, wenn die dort gegebenen Erläuterungen umfangreicher waren, als in den neueren Beschreibungen. Auch für die GPIO-, BCD- und Serial Interfaces sind im Syntax-Anhang vollständige Angaben über die Auswirkungen der einzelnen Anweisungen zu finden.

Wickes, W. C.: Synthetische Programmierung auf dem HP-41C/CV, 2. erw. Aufl.
1984, 165 Seiten, ISBN 3-88538-800-6, DM 36.-
Die englische Originalausgabe dieses Buches ist in den U.S.A. ein Bestseller unter der Literatur über Kleinrechner geworden und auch in Deutschland wurden über 8000 Exemplare verkauft. Die deutsche Ausgabe von Heinz Dalkowski enthält gegenüber dem Original zahlreiche Verbesserungen, Verfeinerungen und Ergänzungen und wird zu Recht die "Bibel der synthetischen Programmierung" genannt.
Der Autor führt den Leser in leicht verständlicher Weise "durch" den HP-41C/CV, entdeckt ihm alle seine verborgenen Fähigkeiten und geht so inhaltlich weit über das hinaus, was das "Handbuch" bietet. Der Leser lernt die Synthetische Programmierung kennen, durch die der Rechner zu unglaublichen Taten veranlaßt werden kann: Erzeugung neuer Zeichen in der Anzeige; Verwendung des Alpha-Registers als arithmetisches Daten-Register; vollständige Benutzerkontrolle über alle Flags (einschließlich der normalerweise unzugänglichen Systemflags); Zugriff auf sämtliche Informationen über den Zustand des Rechners; schnelle Alphabetisierung von Alpha-Daten; Erzeugung neuer Töne; Verwandlung von Programmzeilen in Daten und umgekehrt; programmierter Zugriff auf beliebige Zeilen in ROM's; Herstellung programmierender Programme.
Synthetische Programmierung ist nicht nur für Hobby-Anwender, sondern in gleicher Weise für professionelle Benutzer von Interesse, wenn es darum geht, Speicherplatz zu sparen oder die Bearbeitungsgeschwindigkeit zu erhöhen.
Die zweite Auflage des Buches ist um inzwischen bekannt gewordene Fortschritte der synthetischen Programmierung erweitert worden: Der Byte-Schnapper, Programme zur automatischen Erzeugung synthetischer Programmzeilen, die Funktion eGoBEEP, Programme zur Herstellung vollständiger hexadezimaler Speicherauszüge, synthetischer Vorstoß ins X-Memory.
Obwohl dieses Buch manche Schwierigkeit enthält, was für den einen oder anderen Leser die vorausgehende Lektüre des Buches von K. Jarett, "Synthetisches Programmieren auf dem HP-41 - leicht gemacht" (ebenfalls im Heldermann Verlag) empfehlenswert macht, ist dieses Buch unerläßlich für jeden HP-41C/CV/CX Benutzer, der die Fähigkeiten und Möglichkeiten des Rechners voll ausschöpfen will.

DIE HP-PALETTE DES HELDERMANN VERLAGES

Albers, K.: HP-41 Barcodes mit dem HP-IL-System. 320 Seiten, 44.00 DM, ISBN 3-88538-804-9 (1986)

Dalkowski, H., Fegert, S.: Eine Programmsammlung für den HP-41. ISBN 3-88538-809-X (1987)

Dearing, J. S.: Tricks, Tips und Routinen für Taschenrechner der Serie HP-41. Deutsche Ausgabe von H. Dalkowski. 220 Seiten, 36.00 DM, ISBN 3-88538-801-4 (1984).

Emery, K.: HP-41 Mikrocode-Programmierung für Anfänger. ISBN 3-88538-808-1 (1987)

Horn, J.: HP-71 Basic - leicht gemacht. Deutsche Ausgabe von W. Stroinski. Ca. 200 Seiten, ca. 40.00 DM, ISBN 3-88538-806-5 (1986)

Jarett, K.: Erweiterte Funktionen des HP-41 - leicht gemacht. Deutsche Ausgabe von H. Dalkowski. 240 Seiten, 44.00 DM, ISBN 3-88538-803-0 (1986).

Jarett, K.: Synthetisches Programmieren auf dem HP-41 - leicht gemacht. Deutsche Ausgabe von H. Dalkowski. 170 Seiten, 40.00 DM, ISBN 3-88538-802-2 (1985). Dem Buch liegt eine Quick Reference Card bei.

Meschede, W.: Plotten und Drucken auf dem HP-41 Thermodrucker. 176 Seiten, 36.00 DM, ISBN 3-88538-805-7 (1985).

Stroinski, W. (Herausgeber): Zusammenfassung der Bedienungs- und Programmier-Anleitungen für I/O-ROM, IB- und IL-Interface der HP-Rechner der Serie 80. 296 Seiten, 58.00 DM, ISBN 3-88538-806-5 (1986).

Wickes, W. C.: Synthetische Programmierung auf dem HP-41C/CV. Deutsche Ausgabe von H. Dalkowski. 165 Seiten, 36.00 DM, ISBN 3-88538-800-6 (1983).

HP-41 Kombinierte Hex/Dezimale Byte Tabelle, 7x11.5 cm Plastikkarte, 6.00 DM.

HP-41 Quick Reference Card, 7x15 cm Plastikkarte, 8.00 DM. (Diese Karte liegt dem Buch "Synthetisches Programmieren auf dem HP-41 - leicht gemacht" von K. Jarett kostenlos bei.)

Alle Produkte sind zu den oben angegebenen Preisen ohne Versandkosten direkt vom Verlag erhältlich, die Plastikkarten nur auf diese Weise. Bitte richten Sie Ihre Bestellung an

Heldermann Verlag Berlin
Nassauische Str. 26
D-1000 Berlin 31

Handbücher in deutscher Sprache sind meist nur für den Computer und die wichtigsten Peripherie-Geräte (Drucker, Monitor, Massenspeicher) erhältlich, für die "seltenere" Peripherie, durch deren Anschluß der Computer erst seine volle Wirksamkeit erlangt, sind die Beschreibungen jedoch häufig nur in englischer Sprache lieferbar. Unabhängig von der Qualität der vorhandenen Sprachkenntnisse ist das Verstehen dieser neuen Materie sicherlich einfacher, wenn die Beschreibungen in deutscher Sprache vorliegen.

Für die Hewlett-Packard-Rechner der 80er Serie (HP-83/85 und HP-86/87) wird mit diesem Buch über das I/O-ROM und die Interfaces für HP-IB (IEEE 488 bzw. IEC 625) und HP-IL (Interface-Loop) dieser Mangel behoben. Es enthält die Übersetzungen der nachfolgend genannten HP-Druckschriften in korrigierter Form:

I/O-ROM, Owner's Manual, 00087-90121, Jan. 83
HP-IB Interface, Owner's Manual, 82937-90017, Jan. 82
HP-IL Interface, Owner's Manual, 82938-90001, Jan. 82.

Der Vorläufer dieser Handbücher (I/O Programming Guide, 00085-90142) wurde ebenfalls berücksichtigt, wenn die dort gegebenen Erläuterungen umfangreicher waren, als in den neueren Beschreibungen. Auch für die GPIO-, BCD- und Serial Interfaces sind im Syntax-Anhang vollständige Angaben über die Auswirkungen der einzelnen Anweisungen zu finden.

ISBN 3-88538-806-5