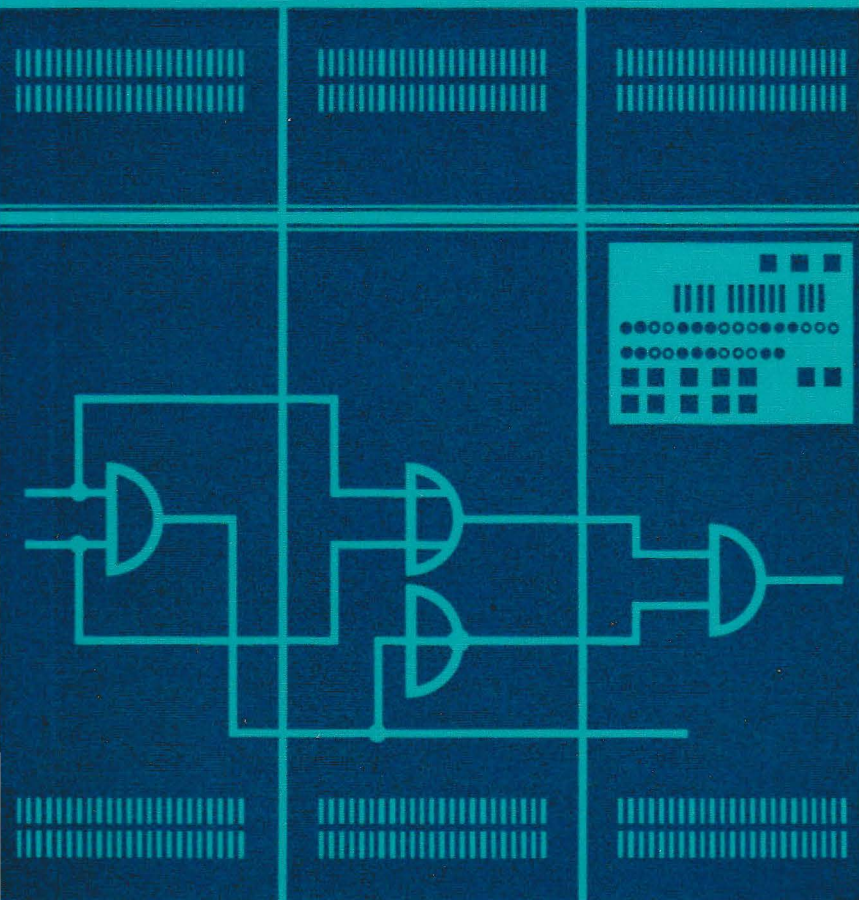


Siemens-System 300 für Prozeßautomatisierung

Band 1: Zentraleinheiten



Heller

Siemens-System 300
für Prozeßautomatisierung
Band 1: Zentraleinheiten

Siemens-System 300 für Prozeßautomatisierung

Band 1: Zentraleinheiten

von Ehrenfried Heller

2. Auflage

ISBN 3-8009-1094-2

Herausgeber und Verlag: Siemens Aktiengesellschaft, Berlin · München

© 1971 by Siemens Aktiengesellschaft, Berlin · München

Alle Rechte vorbehalten, auch die des auszugsweisen Nachdruckes, der fotomechanischen Wiedergabe und der Übersetzung sowie der Bearbeitung für Ton- und Bildträger, für Film, Funk und Fernsehen, für den Gebrauch in Lerngeräten jeder Art.

Printed in Germany

Vorwort

Die Automatisierung von technischen Vorgängen in Industrie, Öffentlichkeit und privatem Bereich ist längst Realität. Überall entbindet die Automatik die Menschen von Routine und z. T. von Verantwortung. Dabei werden jedoch die Aufgaben fortlaufend vielfältiger und verwickelter, so daß im gleichen Maß die Mittel zur Automatisierung an Kompliziertheit zunehmen.

Ingenieure und Techniker, die in den letzten drei Jahrzehnten automatisierte Maschinen und Anlagen einzusetzen hatten, folgten nicht ohne Mühe dieser stürmischen Entwicklung. Mit Relais als Bedienungselementen ähnelten sich die Schaltbilder für Haupt- und Hilfsstromkreise weitgehend. Als jedoch die Halbleiterbauelemente in die Steuerungstechnik eingeführt wurden, schienen fremde Bauteile die vertrauten Bilder zu verändern. Tatsächlich arbeiteten diese neuartigen Elemente nach eigenen, ungewohnten Gesetzen; jedoch nur die neue Terminologie verwirrte, während die Einrichtungen im ganzen nach den bisherigen Gepflogenheiten der Automatisierungstechnik verwendet werden konnten.

Der Schritt zur *Datenverarbeitungsanlage* als Automatisierungsmittel war wesentlich einschneidender. Diese „speicherprogrammierte Automatik“, die Vorgänge durchführt, deren Veranlassung vom Außenstehenden nicht mehr wahrgenommen wird, hat nichts mehr mit den klassischen Steuerungen gemeinsam — so scheint es —; deshalb wird dieser Bereich noch gerne den Spezialisten überlassen.

Das vorliegende Taschenbuch soll dazu beitragen, den Kreis der Interessenten am Prozeßrechner zu erweitern; es setzt die grundlegenden Kenntnisse der logischen Verknüpfungen (mit Halbleiterschaltungen) voraus und versucht, das Stoffgebiet speziell für solche Interessenten darzustellen, die auf Grund ihrer Vorbildung den Zugang dazu nicht selbstverständlich finden.

Diesem Ziel dient auch eine stark vereinfachende Erklärung der Vorgänge in einer DVA: das Rechner-Analogon. Es geht davon aus, daß der technisch Vorgebildete ja immer Parallelen zu ihm bekannten Vorgängen sucht, wenn er eine „Maschine“ verstehen lernen will. Dies ist vor allem bei den außergewöhnlichen Bedingungen wichtig, unter denen eine so perfekte Automatik, wie sie der Rechner darstellt, arbeitet.

München, im Februar 1971

Vorwort zur zweiten Auflage

Wegen der starken Nachfrage kann die zweite Auflage bereits nach Ablauf eines Jahres fast unverändert aufgelegt werden.

Ein zweiter, weiterführender Band „Peripheriegeräte“ erscheint gleichzeitig in erster Auflage.

Berlin und München, im Januar 1972

SIEMENS AKTIENGESELLSCHAFT

Inhalt

1. Einführung	8
1.1. Die Verarbeitung von Daten	12
1.2. Hauptbestandteile der Rechneranlage und des Rechners	15
1.3. Arbeitsweise der Zentraleinheit	17
2. Information für den Rechner	19
2.1. Binäre Signale	19
2.2. Darstellung von Informationen	21
3. Bestandteile der Zentraleinheiten	28
3.1. Arbeitsspeicher	28
3.2. Rechenwerk	33
3.3. Steuerwerk	42
3.4. Ein-Ausgabekanal	45
3.5. Ausnahmen	48
4. Wirkungsweise des Steuerwerks	57
4.1. Befehle für interne Operationen	57
4.2. Befehle für externe Operationen	74
4.3. Prioritätensteuerung	79
5. Grundsätzliches über die Bedienung der Zentraleinheit	82
5.1. Zusammenhang zwischen Hardware und Programmierung	
5.2. Erstellung eines Programms	82
6. Schaltkreistechnik	88
Literaturverzeichnis	94
Stichwortverzeichnis	95

1. Einführung

Die Prozeßrechneranlage gleicht einem zusammenhängenden Komplex von selbständigen Betrieben, welche in gegenseitiger Abstimmung bestimmte Güter erzeugen. Den entscheidenden Anteil an der Produktion — und damit an Bedeutung — hat die übergeordnete Fabrik des Verbandes. Bei ihr laufen alle Wege zusammen; ihr arbeiten die andern Werke zu, und wenn sie einige Waren nicht selbst unterbringen kann, so gibt sie diese an getrennt errichtete Silos zur Aufbewahrung weiter (Bild 1).

Alles untersteht einer leitenden Dienststelle, die die Vorschriften für die Auswahl des Materials und für dessen Verarbeitung abfaßt. Sämtliche Anweisungen gehen der Fabrik zu und werden von ihr im jeweils richtigen Moment an die kleineren Werke ausgegeben.

Aus Gründen der Rationalisierung nimmt die Fabrik nur Material in genormten Paketen entgegen. Intern dürfen ferner ausschließlich solche Pakete transportiert und aufbewahrt werden, die 24 gleichgroße „Normalwürfel“ aufnehmen, weil sie dann den Platz in Beförderungsmitteln und Regalen genau ausfüllen. Da aber zahlreiche Betriebe nur kleine Pakete mit sechs Würfeln Inhalt bereitstellen können, müssen jeweils vier davon in der Fabrik zu einer Einheit zusammengefaßt werden.

Die externen Lieferanten produzieren zwar nach eigenen Gesetzen, sie bringen das Material aber, bevor sie es an die Fabrik abliefern, in diese Einheitsformen. Dabei ist es gleichgültig, ob sie die Güter als einzelne Würfel zusammentragen oder ob sie größere Mengen auf einmal fertigen.

Wenn die Leitung der Fabrik Waren abrufen, befördern die Transportarbeiter der Außenwerke ein Paket nach dem andern zu dem „Pfortner“, den sie auf der an ihrem Betrieb vorbeiführenden Straße erreichen. Standardgüter gelangen langsamer zum einen Pfortner, Eilgüter schneller zu einem anderen. Der Pfortner meldet jedes ankommende Paket im Lagerhaus an. Dort beendet man den gerade laufenden Arbeitsgang, nimmt das auf dem Fließband anrollende Material entgegen und ordnet es im Lager ein.

Damit jedes Paket wiedergefunden wird, steht auf ihm eine bestimmte Regalfachnummer. Diese wurde dem Lieferanten bei der Erteilung des Fertigungsauftrags mitgeteilt.

Zu der Fabrik gehören Lagerhaus mit Lagerbüro, die Maschinenhalle mit einer Universalmaschine und die Pförtnerhäuschen. Das Personal setzt sich aus Robotern zusammen, die nur einige fest umrissene Aufgaben erledigen, aber nichts lernen können. Sie sind überaus fleißig und arbeiten sehr schnell, brechen jedoch jede Tätigkeit ab, wenn ihnen etwas befohlen wird, was ihnen nicht „erklärt“ wurde; sie erkennen aber nicht die Unsinnigkeit eines Befehls, sondern führen alles nach den Anweisungen aus.

Das Lagerhaus nimmt jedes Material, das entsprechend angeliefert oder intern bearbeitet wurde, auf. Nur in seinen Regalen ist es auffindbar und schnell zu greifen. Überall sonst in der Fabrik kann es verloren gehen. Daß zu dem Material auch die Bearbeitungsvorschriften in Paketen angeliefert und hier ordentlich gestapelt werden müssen, ist eine ungewöhnliche Eigenart des Betriebs.

Die Maschine in der Werkhalle läuft automatisch. Sie kann die Werkstoffe auf unterschiedliche Art mischen, nach mancherlei Gesichtspunkten zerlegen, kombinieren und anderes mehr. Für jeden Arbeitsgang läßt sich die Betriebsart an einem Schaltbrett gesondert einstellen. Zwei Regale mit je einem Fach, der „linke und der rechte Akkumulator“ (LA und RA), stehen am Halleneingang.

Über den Pförtner lassen sich Empfang und Versand der Güter dirigieren. Der Pförtner bestimmt die Reihenfolge, wenn gleichzeitig mehrere Lieferanten oder Empfänger auf die Abfertigung warten.

Innerhalb der Fabrik sind die Transportwege einspurig, lassen also keinen Gegenverkehr zu. Sie laufen strahlenförmig vom Lagerhaus weg, so daß dieses in (fast) alle Vorgänge mit einbezogen wird. Um Unordnung zu vermeiden, erledigt das Fabrikpersonal in jedem Zeitpunkt nur eine Sache; dies fällt aber darum nicht auf, weil alles unvorstellbar schnell abläuft.

Die interne Werksorganisation bestimmt die Reihenfolge der Arbeiten nach rationellen Gesichtspunkten. Der wichtigste ist, daß der Hauptspeicher immer „aktuelles“ Material für das Weiterproduzieren bereithält. Ferner gilt unbedingt, daß jede angefangene Tätigkeit

auch zu Ende geführt wird. Unterbrechungen oder Änderungen im Betriebsablauf sind nur dann zulässig, wenn der abgebrochene Vorgang logisch richtig fortgesetzt werden kann.

Eine typische Aufgabe für die Fabrik ist der einfache Mischvorgang. Hierzu ruft die Werkleitung bei zwei Lieferanten die vorgeschriebenen Rohstoffe ab. Die Anweisungen für den Handlungsablauf sind bereits früher von der „Konzernleitung“ erlassen worden und liegen mit dem Material zusammen im Lagerhaus bereit.

Im Lagerbüro liegt ein Stoß von Karten, die von oben nach unten fortlaufend die Fachnummern von 0 bis 16 383 tragen und immer in dieser Reihenfolge bleiben müssen. Wenn der Lagerverwalter nun z. B. die Aufforderung erhält, mit der Arbeit bei Fach 4711 zu beginnen, dann legt er den Kartenstoß so vor sich hin, daß die Nummernfolge oben mit 4711 beginnt.

Als erstes schickt er einen Mann ins Lagerhaus, der den Auftrag hat, das Paket in Fach 4711 zu öffnen und die darin enthaltene Anweisung zu lesen. Sie lautet: „Trage das Paket aus Fach 4715 in die Maschinenhalle und lege es auf das Regal LA“. Der so gelesene Befehl wird sofort ausgeführt.

Der Lagerverwalter hat unterdessen den Zettel mit der Nummer 4711 unter den Haufen geschoben und damit die Nummer 4712 aufgedeckt. Wenn die Vollzugsmeldung für den ersten Befehl eingetroffen ist, schickt er einen Mann zum Fach 4712 und läßt ihn das dort lagernde Paket öffnen. In diesem wird als weiterer Auftrag gefunden: „Bringe das Paket aus Fach 4716 in die Maschinenhalle, nimm dort auch das Paket vom Regal LA, schütte den Inhalt von beiden in die Maschine, drücke auf den Knopf mit der Aufschrift „Addition“ und trage nach Beendigung des Arbeitsvorgangs das fertig abgepackte Ergebnispaket von der Maschine zum Regal LA“.

Wenn dieser Vorgang erledigt ist, veranlaßt der Lagerverwalter die Ausführung des Befehls im Fach 4713, der besagt: „Schaffe das Paket vom Regal LA in der Maschinenhalle in das Lagerhaus und lege es im Fach 4717 ab“. Ist diese Anordnung befolgt, hat der Mischvorgang seinen Abschluß gefunden.

Alle weiteren Tätigkeiten laufen in ähnlicher Weise ab. Der Lagerverwalter sucht immer den nachfolgenden Auftrag im Regalfach mit

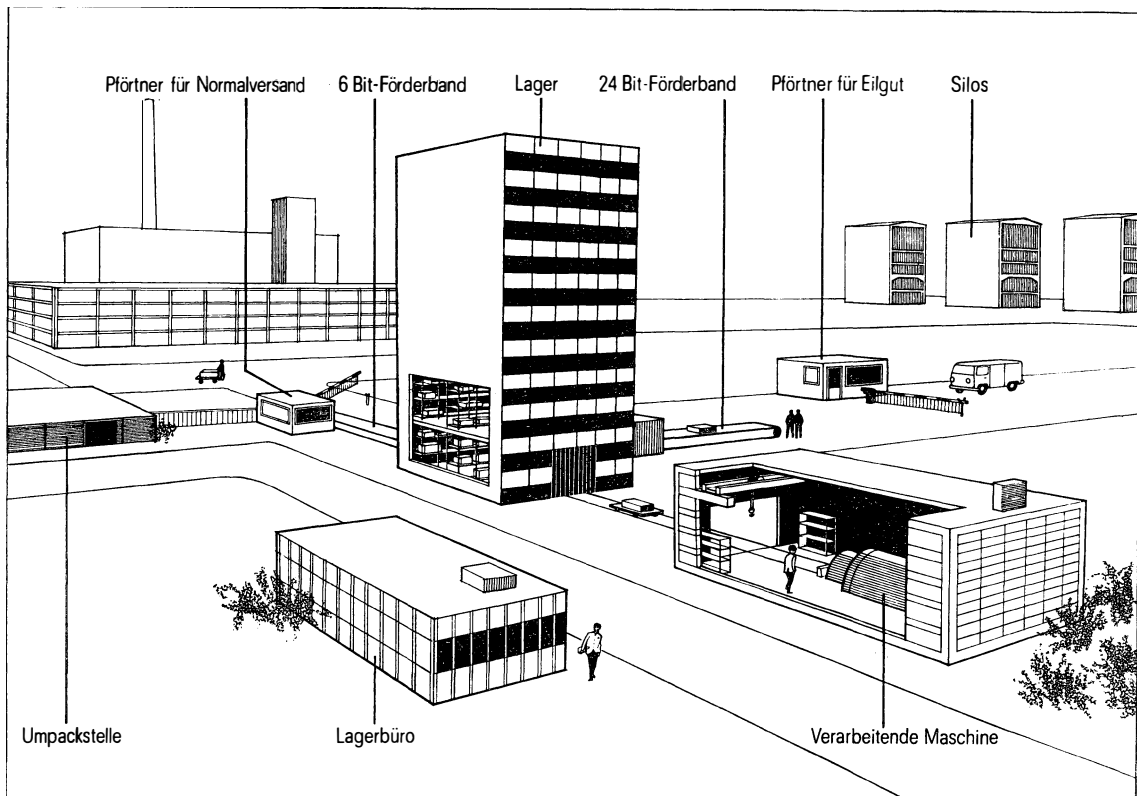


Bild 1: Analogon eines Prozeßrechners

der nächsthöheren Nummer; soll die Nummernreihe verlassen werden, steht dies mit genauer Zielangabe in einem „Befehlspaket“, oder es wird ihm von dem „externen Vorgesetzten“ eigens mitgeteilt.

In geschilderter Weise lassen sich die meisten Arbeitsabläufe in einem Prozeßrechner verständlich darstellen. Da niemandem die zwingende Logik in den Arbeitsbedingungen der Fabrik und in der Folge der Handlungen entgeht, wird auch klar, wie selbstverständlich jede Arbeit das gewünschte Ende findet, wenn sie nach einer richtig konzipierten Reihe von Befehlen durchgeführt wird.

1.1. Die Verarbeitung von Daten

Eine Datenverarbeitungsanlage (DVA) kann *Informationen aufnehmen, speichern*, nach logischen oder arithmetischen Gesichtspunkten *verknüpfen* und wieder *ausgeben*. Die Informationen sind aus Elementen zusammengesetzt, die sich als physikalische Größen darstellen lassen und einzeln oder in Gruppen einen bestimmten Sinn erhalten. Ihre Form braucht nicht in allen Teilen der Anlage dieselbe zu sein; auch kann sie während eines Arbeitsablaufs mehrmals wechseln. So findet man in einer DVA bei der Umsetzung der Daten neben elektrischen auch magnetische oder z. T. sogar mechanische Vorgänge.

Einer physikalischen Größe, durch die ein Informationselement verkörpert werden soll, ordnet man unterscheidbare Beträge zu. In der Gesetzmäßigkeit, nach der die jeweilige Größe diese Beträge annimmt, kommt der Code zum Ausdruck. Er ist für die Darstellung der Informationen in einer vollständigen Maschineneinheit verbindlich. Die Codierung geschieht in einer reversiblen Weise; so muß eine codierte Information ohne weiteres in ihre Ausgangsform zurückverwandelt werden können. Auch muß eine Verknüpfung von codierten Informationen zu dem gleichen Ergebnis führen, das sich bei dem gleichen Verknüpfungsvorgang mit den „nicht codierten“ Informationen eingestellt hätte.

Die übliche DVA ist eine elektronische „Maschine“. Die Logik, nach der sie Daten verarbeitet, wurde von ihrem Konstrukteur in sie hineingedacht und ist in ihren Stromkreisen „materialisiert“. Deshalb muß der Anwender diese „Logik“ der Lösung seiner Probleme zugrundelegen.

Der Benutzer einer Datenverarbeitungsanlage (DVA) verfaßt Bedienungsvorschriften, nach denen die Anlage die von ihm gestellten Aufgaben erledigt. Wie jede andere Maschine tut die DVA nur das, was ihr „befohlen“ wird. Also muß die Folge der Arbeitsschritte, die zur richtigen Lösung der Aufgabe führen, genau vorgegeben sein. In diesem Sinn repräsentiert die DVA eine Automatik, deren Dienst für den Menschen im sehr schnellen und zuverlässigen Wiederholen eines bestimmten Vorgangs liegt.

Da die DVA aber keinen Schritt unvorbereitet, d. h. selbständig tun kann, muß der Benutzer nicht nur den normalen Ablauf ihrer Tätigkeit vorschreiben, sondern auch die Reaktion auf Ausnahme- und Fehlerfälle festlegen. Unregelmäßigkeiten in der Funktion der DVA müssen hier ebenso berücksichtigt werden, wie unnormale Zustände und Einflüsse, die die Arbeit beeinträchtigen könnten.

Die Lieferanten von Datenverarbeitungsanlagen stellen meistens Gruppen von Maschinen (sogenannte *Familien*) mit gemeinsamen Eigenschaften und einheitlichen Konstruktionsmerkmalen her. Die einzelnen Maschinen einer Familie unterscheiden sich vor allem durch den Umfang der eingebauten Schaltungseinheiten; hierdurch wird festgelegt, welche Aufgaben mit welchem Schwierigkeitsgrad in einer bestimmten Zeit erledigt werden können.

Der Aufbau und die Leistungsfähigkeit einer DVA bestimmen ihr Anwendungsgebiet; denn natürlich arbeitet nicht jede Familie auf jedem Sektor gleich gut und wirtschaftlich. Eine eignet sich besonders für die Behandlung von kommerziellen, eine andere von wissenschaftlichen und eine dritte von technischen Problemen. In Grenzgebieten überlappt sich ihre Benützbarkeit.

Es ist am gebräuchlichsten, *frei programmierbare DVA* einzusetzen; diese sind so konstruiert, daß man mit ihnen jede *Aufgabe* nach beliebigen Arbeitsanweisungen lösen kann.

Die *fest verdrahteten Rechner* besitzen im Gegensatz dazu einen Aufbau, der nur *einen Weg* zum Erledigen eines Vorgangs anbietet. Sie gleichen den Automaten der klassischen Technik, weil sie die gleichen Schritte immer in der gleichen Reihenfolge zurücklegen. Will man diese Arbeitsweise ändern, muß man auch die elektrischen Einbauten abwandeln.

Die Entwicklung der *Prozeßrechner* fußte auf der Forderung der Industrie nach einer vielseitig anwendbaren Automatik; diese sollte durch eine preisgünstige DVA mit eigens dafür ausgewählten Eigenschaften befriedigt werden. Das Führen eines Prozesses setzt ein zeitgerechtes, unmittelbares Verfolgen des Arbeitsablaufs voraus (*Real-time-Betrieb*). Hierzu gehört auch die rechtzeitige Reaktion auf besondere, im Zeitpunkt des Auftretens nicht erwartete Vorkommnisse im Betrieb (*Alarmbearbeitung*), so daß das Geschehen während des Prozeßablaufs zweckmäßig und direkt beeinflußt werden kann.

An den zahlreichen Vorgängen, aus denen ein industrieller Prozeß besteht, sind Schaltgeräte, Antriebe und Überwachungseinrichtungen beteiligt. Das Anfahren einer Maschine, ein Sammelschienenwechsel in einer Schaltanlage, das Abfragen von Schalterstellungen und andere betriebliche Routinen fallen oft zeitlich so an, daß beim Betrachter der Eindruck gleichzeitig ablaufender Handlungen entsteht. Tatsächlich kann der Rechner, der für jede Tätigkeit ein eigenes Programm bereithält, nur eines nach dem anderen abarbeiten; dies geschieht allerdings in äußerst kurzer Zeit, obwohl jede mechanische Einrichtung relativ viel Zeit benötigt, bis sie ein Kommando ausführt. Deshalb werden die Programme so gestückelt, daß zwar die Mechaniken jeweils beschäftigt werden, daß aber die nächste Aktion während der Befehlsausführung bereits vorbereitet wird; so gibt es für den Rechner keine unnötigen Wartezeiten. Diese Teilabfertigung der Programme — während der immer mehrere Programme in Bearbeitung stehen und nach und nach abwechselnd fortgesetzt werden — nennt man *Simultanarbeit* der Programme.

Hierbei unterscheidet der Rechner die Dringlichkeit der einzelnen Abläufe und zieht jeweils die wichtigsten Operationen den weniger wichtigen vor (*Unterscheiden von Prioritäten*).

Diese besonderen Eigenschaften befähigen eine DVA, als Prozeßrechner einen Prozeß zu führen. Sie werden durch die Verwendung von geeigneten Peripheriegeräten ergänzt, die einen sehr engen Kontakt zwischen Prozeß und Rechner gewährleisten.

Soll ein Prozeßrechner möglichst erfolgreich sein, muß der Prozeß — der bisher vielleicht nach empirisch ermittelten „Erfahrungswerten“ abgewickelt („gefahren“) wurde, — auf seine Gesetzmäßigkeiten hin untersucht und in gut überschaubare Teilvorgänge zerlegt werden.

Als Automatik kann der Rechner dann *steuernd und regelnd*, aber auch nur zur *Überwachung* benutzt werden.

Die Automatisierung ist nicht in erster Linie auf eine Einsparung von Betriebspersonal ausgerichtet. Auch andere wirtschaftliche Gesichtspunkte wie Erzielung besserer Qualität, höherer Produktionen bei gleichbleibender Beschaffenheit des hergestellten Gutes, eines verlustärmeren Energiehaushalts und einer besseren Ausnützung der am Prozeß beteiligten Maschinen bei gleichzeitig bestmöglicher Schonung sind oft ausschlaggebend. Auch soll das Personal in der „gesteuerten Anlage“ von Routineaufgaben befreit und vor Überforderung durch zuviele einzelne Überwachungsaufgaben geschützt werden.

1.2. Hauptbestandteile der Rechneranlage und des Rechners

Eine Rechneranlage besteht aus der *Zentraleinheit* (ZE) und einer unterschiedlich hohen Anzahl von *Peripheriegeräten* (Bild 2). Jedes Gerät, das mit dem Rechner Daten austauschen soll, benötigt Einrichtungen, die dem Datentransport und der eigenen Funktion dienen. Diese Einrichtungen arbeiten teils elektrisch und teils mechanisch und sind nicht immer mit dem eigentlichen Gerät in einem Gehäuse vereinigt. Trotzdem faßt man alle diese Einrichtungen einschließlich der mechanischen Bestandteile in dem abstrakten Begriff „*Externes Element*“ (EXE) zusammen; diese lassen sich in vier Gruppen gliedern:

- Geräte für Ein- und Ausgabe von Daten, wie sie auch in der kommerziellen Datenverarbeitung Verwendung finden, z. B. Lochkarten- und Lochstreifengeräte, Blattschreiber und Schnelldrucker; man nennt sie auch *Standardperipherie*;
- Geräte für die Speicherung von Daten auf magnetischen Datenträgern außerhalb der ZE; sie heißen *Externe Speicher*;
- Einrichtungen zur Erfassung und Ausgabe von Prozeßdaten und Signalen, die zur Prozeßführung notwendig sind; das sind *Prozeßelemente* und
- Einrichtungen zum Anfahren, Bedienen, Überwachen und Abstellen der Rechneranlage mit der Bezeichnung *Bedienungselement*.

Das an der ZE angebrachte Bedienungsfeld wird nur zu direkten Eingriffen in die Stromkreise benützt und ist Bestandteil der ZE.

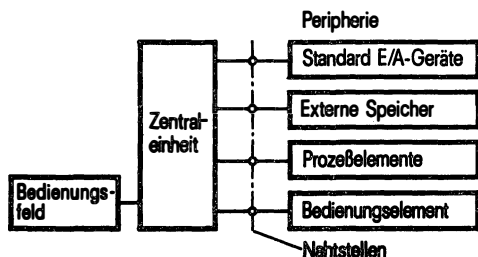


Bild 2:
Zentraleinheit
mit externen
Elementen (EXE)

Die Zentraleinheiten *aller* DVA — also auch der Prozeßrechner — setzen sich in der Regel aus Arbeitsspeicher, Steuerwerk, Rechenwerk und einem Verbindungsglied zur Peripherie, dem Ein-Ausgabekanal, zusammen. Diese Unterteilung gilt auch für viele Datenverarbeitungsanlagen unterschiedlichster Herstellung; einzig die Namen für die Teile der Maschinen wechseln von einem Hersteller zum anderen.

Der *Arbeitsspeicher* (ASP) nimmt alle Daten und ihre Bearbeitungsvorschriften auf und hält sie zur Abholung bereit. Das *Steuerwerk* sorgt für den Transport der Daten durch die Stromkreise der ZE und veranlaßt die Verarbeitung nach den gegebenen Anweisungen (dem Programm). Das *Rechenwerk* verknüpft die Daten in der vorgeschriebenen Weise und der *Ein-Ausgabekanal* (E/A-Kanal) gewährleistet einen geregelten Datenverkehr mit der Peripherie (Bild 3).

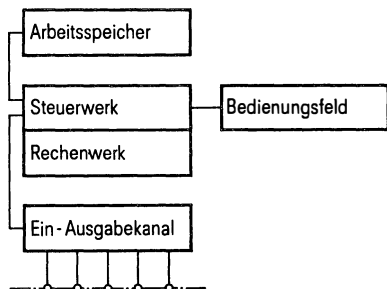


Bild 3: Die Zentraleinheit

1.3. Arbeitsweise der Zentraleinheit (ZE)

Der Prozeßrechner verarbeitet wie jede DVA binär verschlüsselte Informationen. Diese können über geeignete Eingabegeräte vom Prozeß oder von abgelochten Datenträgern her in die ZE gebracht werden. Dort sollen sie in sehr kurzer Zeit auf bestimmte Kriterien hin untersucht oder miteinander verknüpft werden. Steuerwerk und Rechenwerk führen diese Operationen in wenigen Mikrosekunden aus. Das Ergebnis wird abgespeichert und kann über Ausgabegeräte als Sollwert in eine Regeleinrichtung, als Steuerkommando für einen Schaltvorgang oder als Anzeige zur Bekanntgabe an das Betriebspersonal in die Prozeßanlage zurückkommen.

Die äußerst hohe Geschwindigkeit der internen Operationen setzt voraus, daß Daten und Bearbeitungsvorschriften greifbar im Rechner vorhanden sind. Sie werden im ASP so abgelegt, daß sie gezielt abgeholt werden können. Die Bearbeitungsvorschriften und die Vergleichsgrößen sind Bestandteile des *Programms*; man locht sie auf Lochstreifen oder Lochkarten ab und lädt sie vor dem Arbeitsbeginn in den ASP. Die Prozeßdaten gelangen während des Programmablaufs über das Prozeßelement in den ASP; auch ihr dortiger Platz wurde durch den Programmierer festgelegt.

Man muß sich vergegenwärtigen, daß jedes noch so schnelle Peripheriegerät um Größenordnungen langsamer ist als die ZE; eine komplette Addition von zwei Zahlen, einschließlich des Abholens der Operanden aus dem ASP und des Abspeicherns des Ergebnisses im ASP dauert bei den meisten der hier besprochenen Maschinen 9 μ s. Der Kontakt eines schnellen Relais dagegen schlägt in ca. 5 ms um und ein Blattschreiber (die Schreibmaschine der DVA) benötigt 100 ms für das Drucken eines Buchstabens. Schon die Ausgabe eines Zeichens vom ASP in die Steuerung eines EXE läuft möglicherweise 20 μ s. Es wird daher verständlich, daß das Zeitproblem bei der Handhabung eines Prozeßrechners wesentlich ist.

Im Rechner kann in jedem Augenblick nur ein Arbeitsgang ablaufen. Deshalb muß der Programmierer die verschiedenen Geschwindigkeiten von Peripheriegeräten und ZE so in Einklang bringen, daß jede Operation in der ZE und zwischen ZE und EXE zum richtigen Zeitpunkt erfolgt. Dieses Bemühen unterstützt der Geräteentwickler dadurch, daß er die Arbeitsabläufe in ZE und EXE weitgehend ent-

koppelt und der ZE unnötige Aufenthalte und Wartezeiten erspart. Das EXE arbeitet selbständig; es muß zwar von der ZE zur Arbeit aufgefordert werden, läuft dann aber mit eigener Geschwindigkeit weiter und sucht von sich aus zur rechten Zeit Kontakt mit dem Rechner.

In den Prozeßrechneranlagen gelten also Grundsätze, die eine größtmögliche Effektivität der einzelnen Geräteeinheiten garantieren; dieses gilt — mit wenigen Einschränkungen — auch für den Prozeß selbst. Manche Anlage muß erst durch die Umstellung der technischen Einrichtungen auf eine Zusammenarbeit mit dem Rechner vorbereitet werden. So z. B. sparen kurze Transportwege viel Zeit und vergrößern damit die Übersicht. Gute Organisation unterstützt also den Rechner bei der optimalen Ausnützung der Prozeßanlage.

2. Informationen für den Rechner

2.1. Binäre Signale

Im Stromkreis gibt es zwei scharf unterscheidbare Zustände, die so charakterisiert werden können:

Strom fließt — Strom fließt nicht, Spannung liegt an — Spannung fehlt, ein Kontakt ist geschlossen — ein Kontakt ist offen. Mit Hilfe einer Lampe oder eines Elektromagneten (z. B. zur Entklinkung eines Schlosses) kann man sie sichtbar machen. Man leitet daraus dann den abstrakten Begriff *Signalzustand* ab. Da es nur jeweils *zwei* Möglichkeiten gibt, spricht man von *binären Signalen* (Bild 4).

Zum Darstellen binärer Signale benötigt man streng genommen zwei scharf voneinander abweichende Potentiale bzw. Spannungen. In der Halbleitertechnik ist dies — da dort für das „Schalten“ keine Kontakte zur Verfügung stehen — nicht ohne weiteres nachzubilden; deshalb müssen hier binäre Signale aus einem gut darstellbaren *Bereich* herausgeschnitten und als „gegensätzliche“ Zustände definiert werden. Dabei ist der Absolutwert der beiden Signalpegel für die Definition unwichtig. Der Bereich zwischen den Pegelwerten wird jeweils übersprungen und hat keine Bedeutung. Dagegen kann jeder Pegel als

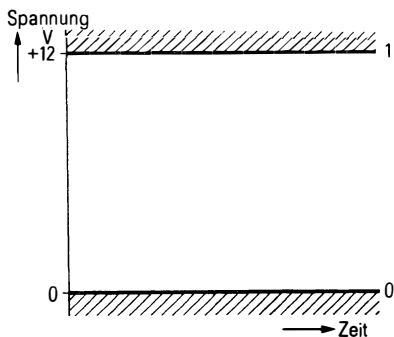


Bild 4: Signalzuordnung

Schwelle betrachtet werden, deren Über- bzw. Unterschreitung den betreffenden Signalzustand kennzeichnet (Bild 4).

Die voneinander unterschiedenen Bereiche erhalten die Bezeichnungen 0 und 1. Üblicherweise ordnet man 1 dem positiven Spannungswert zu.

Da die Signalzustände in den Stromkreisen mit Halbleiterbauelementen mit hoher Geschwindigkeit wechseln und ihr Auf- und Abbau einem „fließenden Übergang“ gleicht, benötigt man zur Identifizierung eines Signals ein zusätzliches Kriterium: den *Takt*. Jede Gerätesteuerung und vor allem die ZE besitzen deshalb einen Taktgenerator, der die Vorgänge in den zusammenarbeitenden Schaltungseinheiten „koordiniert“. Besonders in parallel betriebenen Stromkreisen muß für gleichzeitiges Aufnehmen und Festhalten der Signalzustände gesorgt sein. Hierzu dient ein weitverzweigtes Taktsystem, das an entsprechenden Torschaltungen (z. B. mit einem zusätzlichen 1-Signal an einer UND-Verknüpfungsstufe) die Weiterleitung des anstehenden Signals freigibt und damit das resultierende Signal als gültig festlegt.

Informationen setzen sich aus einem oder mehreren Signalen zusammen. Die Auswahl der Signalzustände, die eine Information verkörpern sollen, muß so getroffen werden, daß man aus den Signalen jederzeit wieder den Informationsinhalt entnehmen kann. Dann spricht man von Codierung. Der Code ist an sich frei wählbar; doch kann er nicht willkürlich geändert werden, wenn wie hier bei den Rechnern in den Stromkreisen bestimmte Codebestandteile bereits „materialisiert“, d. h. durch Stromwege festgelegt sind.

Dem Prozeßrechner werden aus der überwachten Anlage neben Informationen, die aus Binärsignalen zusammengesetzt sind, auch analoge Meßwerte angeboten. Der Rechner kann diese Analogwerte nicht direkt verarbeiten. Sie müssen vorher in Digitalwerte umgewandelt werden. Dies bewirkt eine Einrichtung, die den Meßbereich in Stufen gleicher Höhe „einteilt“ und jeden Analogwert in den nächstgelegenen digitalen Stufenwert umsetzt. Die Höhe der Stufe ist ein Maß für die Genauigkeit der Anordnung. Jeder Meßwert wird dann durch Binärsignale in parallelen Stromkreisen ausgedrückt.

Setzt man die Binärsignale in parallelen, zusammengehörenden Stromkreisen zu einer Information zusammen, so kann man die nebenein-

ander auftretenden 0- und 1-Signale als Ziffern einer Dualzahl auffassen. Diese Dualzahl stellt dann die Codierung der Information dar. Legt man keinen Zahlenwert zugrunde und benützt man nur die Symbolik der Dualarithmetik, so erhält man einen allgemein anwendbaren Code.

Jede 1 oder 0 in dieser Informationsdarstellung wird mit *Bit* bezeichnet. Das Bit ist die Sonder-Einheit für die Anzahl der Binär-entscheidungen.

Jedes Bit nimmt im Rahmen einer Bitkombination eine Stelle ein, der eine bestimmte Aussage zugeordnet worden ist.

Man geht aber auch soweit, den Signalzuständen 0 und 1 einen Sinn zu geben, der über die Schaltkreistechnik hinaus anwendbar ist. So kann man binäre Verschlüsselungen auch durch Spannungsschwankungen (oder Flußwechsel in einem Magnetsystem) ausdrücken, deren Anzahl innerhalb eines Taktes ein bestimmtes Signal verkörpert. Dann kennzeichnen z. B. *eine* Schwingung je Takteinheit eine 0 und *zwei* Schwingungen eine 1. Diese Art der Codedarstellung ermöglicht eine Aufzeichnung von Informationen auf magnetische Datenträger, die unabhängig von den Stromschwankungen beim „Schreibvorgang“ später ein eindeutiges „Lesen“ zuläßt. Für die Fernübertragung von Daten gibt es noch weitere Signaldarstellungen. Die Wahl der Verschlüsselung hängt dann vor allem von den Vorschriften für den Betrieb des Verbindungsweges ab (z. B. CCITT).

2.2. Darstellung von Informationen

Im vorausgehenden Abschnitt wurde erwähnt, daß jedem Binärsignal ein eigener Informationsinhalt zugeordnet werden kann; parallel auftretende Signale lassen sich aber auch zu einer „Summeninformation“ zusammenfassen. Die so entstehenden Dualzahlen seien hier kurz behandelt:

Wie die Dezimalzahl auf der Basis 10 entwickelt sich die Dualzahl auf der Basis 2. Die einzelnen Ziffern einer Zahl erhalten einen Stellenwert, der abhängig von der Position in der Zahl einer bestimmten Potenz entspricht.

Beispiel: $10^0 = 1$; $10^1 = 10$; $10^2 = 100$; $10^3 = 1000$ usw.
 $2^0 = 1$; $2^1 = 2$; $2^2 = 4$; $2^3 = 8$ usw.

Man kann die Dualzahlen nach einfachen Regeln aus den Dezimalzahlen errechnen. Die Dezimalzahl 101 (zum Unterschied zur Dualzahl so geschrieben: $(101)_{10}$) entspricht der Dualzahl $(1100101)_2$, dies zeigt folgende Rechnung:

$$(101)_{10} = 1 \times 64 + 1 \times 32 + 0 \times 16 + 0 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1.$$

Wenn man die Stellenentwicklung der Dualzahlen nicht als Tabelle greifbar hat, kann man auch rechnen:

$$\begin{array}{rcl} 101 : 2 & = & 50 \text{ Rest } 1 \\ 50 : 2 & = & 25 \text{ Rest } 0 \\ 25 : 2 & = & 12 \text{ Rest } 1 \\ 12 : 2 & = & 6 \text{ Rest } 0 \\ 6 : 2 & = & 3 \text{ Rest } 0 \\ 3 : 2 & = & 1 \text{ Rest } 1 \\ 1 : 2 & = & 0 \text{ Rest } 1 \end{array}$$

Die im Rest erscheinenden Ziffern ergeben von unten nach oben gelesen die gesuchte Dualzahl.

Die Umrechnung dual/dezimal nimmt man am leichtesten so vor, daß man für jede „1“ in der Dualzahl das dezimale Stellenäquivalent (1, 2, 4, 8, ...) sucht und diese Werte addiert.

Neben den binär codierten Dualzahlen werden oft auch *tetradisch verschlüsselte Dezimalzahlen* verwendet. In diesem Fall benötigt man für jede Dezimalziffer eine vierstellige Dualzahl (Tetrade). Die Codierung kann durch die einfache Zahlenfolge von 0000 bis 1001 ausgedrückt werden; man spricht dann von dem BCD-Code (binary coded decimal figures). In der SIMATIC¹⁾-Technik spielt der Aiken-Code²⁾ eine größere Rolle, weil er leichter in ein Überwachungssystem einbezogen werden kann; die Verschlüsselung sieht dann so aus:

¹⁾ In Bausteintechnik ausgeführte elektronische Schaltkreissysteme des Hauses Siemens für Prozeßautomatisierung

²⁾ Symmetrisch aufgebauter, stellenbewertbarer Dezimalcode

0 entspricht	0000	9 entspricht	1111
1 „	0001	8 „	1110
2 „	0010	7 „	1101
3 „	0011	6 „	1100
4 „	0100	5 „	1011

Hier ergibt die Summe jeder Zeile bei den Dezimal- wie bei den Dualzahlen immer 9.

Alle Bitkombinationen, die eine „willkürlich“ zusammengesetzte Reihe von binären Codeelementen bilden, führen den Namen: *Bitmuster*. Dies ist ein Sammelbegriff, der alle binär codierten Informationen umfaßt.

In einem Prozeßrechner des Siemens Systems 300 gibt die Anordnung der Stromkreise eine feste Länge für jede Information vor. 24 Bitstellen bilden ein *Wort*, das in der ganzen Maschine parallel verarbeitet wird. Man unterscheidet Befehls- und Datenwörter. Eine *Date* kann ein Meßwert, eine Sammelmeldung von Schaltzuständen in der Prozeßanlage oder eine Zahl sein, mit der eine arithmetische oder logische Operation ausgeführt werden soll. Ein *Befehlswort* ist die verschlüsselte Form einer Arbeitsanweisung an den Rechner, die den Anstoß für eine Tätigkeit des Rechen- oder Steuerwerkes gibt.

Die schon sehr frühzeitige Entwicklung der Standardperipherie führte dazu, daß die Lochstreifen-, Lochkarten- und die schreibenden und druckenden Geräte nur einzelne Zeichen entgegennehmen oder ausenden können. Die Summe aller in Texten üblichen Zeichen wie Dezimalziffern, Buchstaben und Satzzeichen nennt man *Alphanumerische Zeichen*. Zu diesen kommen noch „Befehlszeichen“ für Wagenrücklauf, Zeilenvorschub, Textende usw., die zur Bedienung der schreibenden Geräte gehören.

Die 64 alphanumerischen Zeichen und Umschaltbefehle lassen sich mit je 6 Bitstellen ausdrücken. Je vier von ihnen können somit in einem Maschinenwort untergebracht werden. Da ein Maschinenwort aber anstelle von alphanumerischen Zeichen auch andere Informationen enthalten kann, nennt man jede beliebige 6-Bit-Kombination ein *Teilwort*.

Für Lochkarten- und Lochstreifengeräte wird jedes 24-Bit-Wort automatisch in vier Teile zerlegt, auch wenn es keine selbständigen

Tabelle 1 Codierung der Zeichen im Siemens-System 300

Bedeutung		Arbeitsspeicher		Lochkarte	Lochstreifen	Blattschreiber	
Symbol oder Abkürzung	Bemerkung	äquiva- lente Dezimal- zahl	Binärmuster B A 8 4 2 1		1 2 ● 3 4 5	BU/ZI	Druck- zeichen
ZWR	Zwischenraum (Blank)	0		kein Loch	● ○	B,Z	ZWR ⁷⁾
1		1	1	1	○ ○ ● ○ ○	Z	1
2		2	1	2	○ ○ ● ○ ○	Z	2
3		3	1 1	3	○ ● ○ ○ ○	Z	3
4		4	1	4	○ ● ○ ○ ○	Z	4
5		5	1 1	5	● ○ ○ ○ ○	Z	5
6		6	1 1	6	○ ● ○ ○ ○	Z	6
7		7	1 1 1	7	○ ○ ● ○ ○	Z	7
8		8	1	8	○ ● ○ ○ ○	Z	8
9		9	1 1	9	● ○ ○ ○ ○	Z	9
0		10	1 1	0	○ ● ○ ○ ○	Z	0
-	gleich	11	1 1 1	3—8	○ ● ○ ○ ○	Z	—
'	Apostroph	12	1 1	4—8	○ ● ○ ○ ○	Z	'
:	Doppelpunkt	13	1 1 1	5—8	○ ● ○ ○ ○	Z	:
>	größer als	14	1 1 1	6—8	● ○ ○ ○ ○	Z	> ⁸⁾
SRU	Schwarz-Rot-Umschaltung ¹⁾	15	1 1 1 1	7—8			SRU ⁸⁾
RSU	Rot-Schwarz-Umschaltung ¹⁾	16	1	2—8			RSU ⁸⁾
/	Schrägstrich	17	1 1	0—1	○ ● ○ ○ ○	Z	/
S		18	1 1	0—2	○ ● ○ ○ ○	B	S
T		19	1 1 1	0—3	● ○ ○ ○ ○	B	T
U		20	1 1	0—4	○ ○ ● ○ ○	B	U
V		21	1 1 1	0—5	○ ● ○ ○ ○	B	V
W		22	1 1 1	0—6	○ ○ ● ○ ○	B	W
X		23	1 1 1 1	0—7	○ ● ○ ○ ○	B	X
Y		24	1 1	0—8	○ ● ○ ○ ○	B	Y
Z		25	1 1 1	0—9	○ ● ○ ○ ○	B	Z
	nicht belegt ²⁾	26	1 1 1	0—2—8			⁵⁾
,	Komma	27	1 1 1 1	0—3—8	● ○ ○ ○ ○	Z	,
(Klammer auf	28	1 1 1	0—4—8	○ ○ ● ○ ○	Z	(
	nicht belegt ²⁾	29	1 1 1 1	0—5—8			⁵⁾
NM	Namengeber (Werda) f. Blattschreiber	30	1 1 1 1	0—6—8	○ ● ○ ○ ○	Z	⌈ ⁶⁾

ZL	Zeilenvorschub für Blattschreiber	31	1 1 1 1 1	0—7—8	○ ●	B,Z	ZL ⁶⁾)
—	Minus (Bindestrich)	32	1	11	○ ○ ●	Z	—
J		33	1	11—1	○ ○ ● ○	B	J
K		34	1	11—2	○ ○ ● ○ ○	B	K
L		35	1	11—3	○ ● ○ ○ ○	B	L
M		36	1	11—4	● ○ ○ ○ ○	B	M
N		37	1	11—5	● ○ ○ ○ ○	B	N
O		38	1	11—6	● ○ ○ ○ ○	B	O
P		39	1	11—7	○ ● ○ ○ ○	B	P
Q		40	1	11—8	○ ○ ● ○ ○	B	Q
R		41	1	11—9	○ ● ○ ○ ○	B	R
ZI	Ziffernumschaltung ³⁾	42	1 1 1	11—0	○ ○ ● ○ ○ ○	B,Z	ZI ⁶⁾)
\$	Dollar	43	1	11—3—8	● ○ ○ ○ ○	Z	\$
*	Stern	44	1	11—4—8	○ ● ○ ○ ○	Z	*
%	Prozent	45	1	11—5—8	○ ○ ● ○ ○ ○	Z	%
;	Semikolon, Endezeichen	46	1	11—6—8	○ ○ ● ○ ○ ○	Z	:
WR	Wagenrücklauf ⁴⁾	47	1	11—7—8	● ○ ○ ○ ○	B,Z	WR ⁶⁾)
+	Plus	48	1	12	○ ● ○ ○ ○	Z	+
A		49	1	12—1	○ ○ ● ○ ○ ○	B	A
B		50	1	12—2	○ ● ○ ○ ○ ○	B	B
C		51	1	12—3	○ ● ○ ○ ○ ○	B	C
D		52	1	12—4	○ ● ○ ○ ○ ○	B	D
E		53	1	12—5	○ ● ○ ○ ○ ○	B	E
F		54	1	12—6	○ ● ○ ○ ○ ○	B	F
G		55	1	12—7	○ ● ○ ○ ○ ○	B	G
H		56	1	12—8	○ ● ○ ○ ○ ○	B	H
I		57	1	12—9	○ ● ○ ○ ○ ○	B	I
BU	Buchstab.-Umschaltg. f. Blattschreib.	58	1	12—0	○ ○ ● ○ ○ ○ ○	B,Z	BU ⁶⁾)
.	Punkt	59	1	12—3—8	● ○ ○ ○ ○ ○	Z	.
)	Klammer zu	60	1	12—4—8	○ ● ○ ○ ○ ○	Z)
#	Nummer, Itrungszeichen	61	1	12—5—8	○ ● ○ ○ ○ ○	Z	#
<	kleiner als	62	1	12—6—8	● ○ ○ ○ ○ ○	B	□ ⁵⁾
BEZ	Bereichsendezeichen	63	1	12—7—8	○ ○ ○ ○ ○ ○	B	BU ⁶⁾)

1) Farbumschaltung beim Blattschreiberelement, Bedienungselement, Bedienungsblasschreiber

2) frei verfügbar

3) Ziffernumschaltung beim Blattschreiber

4) Wagenrücklauf beim Blattschreiber

5) Bei Ausgabe des Zeichens wird Zwischenraum gegeben und die Anzeige "nicht zugelassenes Zeichen" gesetzt

6) Betriebsschaltzeichen, kein Abdruck

7) Wird bei Eingabe nicht in den Arbeitsspeicher übernommen

Zeichen enthält; die Teilwörter erscheinen dann auf dem Datenträger im geräteeigenen Code. Dies ist zulässig, weil die Rückführung der eigencodierten Zeichen in den Interncode des Rechners nach den gleichen Gesetzen wie die Verschlüsselung durchgeführt wird.

In anderen DVA, wie z. B. denen des Siemens Systems 4004, benutzt man eine Wortstruktur, die von der hier besprochenen abweicht. Die Grundeinheit ist dort das *Byte*, aus dem Wörter variabler Länge zusammengesetzt werden können. Ein Byte umfaßt 8 bit.

Mit externen Speichern oder Geräten, die nichtmagnetische Datenträger verwenden, tauscht die ZE nur größere Datenmengen aus. Sonst würde der Zeitbedarf für den Aufbau einer Verbindung zum EXE in keinem realen Verhältnis zur Übertragungszeit stehen. Man bezeichnet eine größere Anzahl von Wörtern, die mengenmäßig nicht festgelegt ist, als *Block*.

Im Gegensatz zum Datenwort, in dem jede beliebige Bitzusammenstellung vorkommen kann, erhält ein *Befehlswort* eine genau festgelegte Struktur. Das liegt daran, daß den Bitstellen im Befehlswort bestimmte Funktionen in den zugehörigen Stromkreisen des Steuerwerks zugeordnet sind. Einige Bitstellen kann man sogar als „Schalter“ verstehen, die — mit 1 besetzt — besondere Stromwege zuschalten und damit den Sinn des Befehls modifizieren. Der Rechner ist somit selbst in der Lage, die programmierten Befehle zu lesen und ihren Inhalt, z. B. für die Behandlung der Datenwörter, als Bearbeitungsvorschriften zu interpretieren. Die Befehlswörter ersetzen vergleichsweise den Schaltwörter, der eine handbediente Schaltanlage durch Drücken bestimmter Tasten nach einer vorliegenden Anweisung steuert.

Die Maschine kann von sich aus kein Befehlswort von einem Datenwort unterscheiden, da die Bitfolge eines Befehlswortes auch eine Dualzahl sein kann. Die vom Programmierer beeinflusste Behandlung des Wortes durch das Steuerwerk stellt jedoch den Gegensatz heraus. Der Maschinenbefehl für die Prozeßrechner des Systems 300 hat den Aufbau:

Bitstelle	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Inhalt	a														A	M	S	U	Op					

Er wird unterteilt in einen *Adreß- oder Parameterteil* a, einen *Modifikationsteil* A/M/S/U und einen *Operationsteil* Op.

Der Adreß- oder Parameterteil enthält eine Operandenadresse,

- oder eine Verschiebezahl für Rechenwerkoperationen,
- oder die Kanalnummer eines angesprochenen EXE,
- oder eine Angabe für eine externe Steuerung
(einen sogenannten Versorgungsparameter).

Im Modifikationsteil bedeutet

- A die Angabe, mit welchem Akkumulator (Hauptregister des Rechenwerks) der Befehl ausgeführt werden soll
(linker Akku 0, rechter Akku 1);
- M die Stelle, an der eine 1 in Verbindung mit dem eingeschalteten Schalter „Markierungssprung“ am Bedienungsfeld ein Stoppen des Programmablaufs nach Ausführen dieses Befehls veranlaßt;
- S eine Adreßsubstitution; darunter versteht man das Ersetzen der in den Bitstellen 1 bis 14 angegebenen Adresse durch eine andere, wenn diese Bitstelle mit 1 besetzt ist;
- U die Gelegenheit für eine Programmunterbrechung; liest der Rechner hier eine 1, so besteht nach der Ausführung des Befehls die Möglichkeit, auf ein anderes Programm überzuwechseln.

Der Operationsteil enthält in 6 Bitstellen die Verschlüsselung der Befehle an das Steuerwerk des Rechners. Von den 64 verschiedenen Bitkombinationen, die jeweils einen eigenen Schaltungszustand in den Schaltkreisen des Steuerwerks und des Rechenwerks herstellen können, sind bisher maximal 55 ausgenützt worden.

Für die Zentraleinheiten 301 und 306 wurde der Inhalt des Befehlswortes etwas verändert. Darauf wird an anderer Stelle noch eingegangen.

3. Bestandteile der Zentraleinheiten

3.1. Arbeitsspeicher

Der Arbeitsspeicher (ASP) dient als Aufbewahrungsort für alle Daten und die Vorschriften (Befehle), nach denen die Daten zu verarbeiten sind. Er gleicht einem Lagerhaus, in dem die Waren in vielen Regalen gestapelt werden. Jede Sendung, die hier ihren Platz finden soll, muß mit einer Fachnummer versehen sein, damit sie gezielt abgelegt und sicher wieder aufgefunden wird. Diese Nummer ist eine Adresse und stellt gleichzeitig eine Kennzahl dar, mit der die hier gelagerte Ware identifiziert werden kann.

Der ASP eines Prozeßrechners besteht heute aus Ferritringkernen [1], durch die matrizenartig Kupferleiter gefädelt sind; die Halterung besorgt ein Profileisenrahmen. Für jedes zu speichernde Bit muß ein Ringkern vorhanden sein. Jeweils 24 Kerne bilden die Worteinheit, sodaß die Kapazität des Speichers in Wörtern bzw. in Kilowörtern (gleich 1024 Wörter) ausgedrückt wird. Die normalen Ausbaustufen der Zentraleinheiten 302, 304 und 305 umfassen 8K oder 16K Wörter. Für die ZE 301 und 306 sind jeweils vier Ausbaustufen mit unterschiedlicher Kapazität vorgesehen.

Der Informationszustand eines Ringkerns wird durch die Magnetisierungsrichtung festgelegt und ist somit von der Verdrahtung des Speichers abhängig. Der Magnetisierungszustand nach einem Einschreibvorgang bedeutet 1-Signal. Zum Lesen dieser Information muß der Kern ummagnetisiert werden, wodurch sein Inhalt zerstört wird. Soll die Information erhalten bleiben, muß sie nach dem Lesen wieder eingeschrieben werden. Dieser Vorgang ist ein wichtiger Teil des Arbeitsspeicher-Zyklus.

Um einen Arbeitsspeicher mit 16K Wörtern Kapazität benützen zu können, müßte man 16 384 Drähte durch jeweils 24 Ringkerne führen, da die Kerne für die Speicherung eines Wortes durch einen gemeinsamen Draht gesetzt werden können. Diese hohe Anzahl an Magnetisierungsdrähten kann man reduzieren, wenn man die Ring-

kerne matrixenartig anordnet und den Ummagnetisierungsstrom auf zwei Drähte aufteilt (Bild 5).

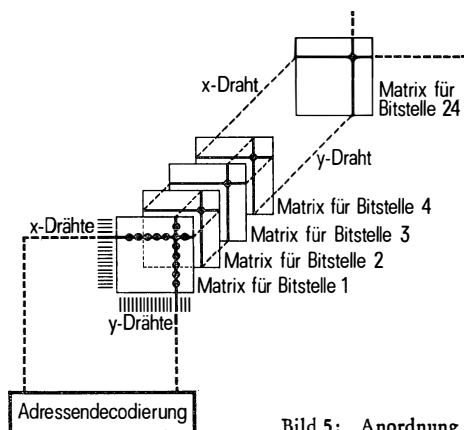


Bild 5: Anordnung der Ringkernmatrizen

Da jeder der beiden Drähte *allein* keine Änderung des Magnetisierungszustandes in einem Kern hervorrufen kann, wird nur der Kern angesprochen, in dem sich die beiden Drähte treffen. Man bezeichnet sie jeweils als x- und y-Draht [1], [2].

Bei Vollausbau des ASP mit 16 384 Wörtern Kapazität müssen 24 x 16 384 Kerne vorhanden sein. Jede Matrix wird von 128 x- und 128 y-Drähten durchzogen, die zusammen die notwendigen 16 384 Kreuzungspunkte bilden, an denen sich die Kerne befinden.

Bei einem Lese- oder Schreibvorgang wird durch ein Drahtpaar (einen x- und einen y-Draht) in jeder Matrix nur ein Kern beaufschlagt und dessen Reaktion erwartet. Deshalb kann man dieselben x- und y-Drähte durch alle 24 gleichartig aufgebauten Matrizen ziehen; sie treffen sich jeweils in Kernen gleicher Lage, die zusammen eine komplette Wortinformation enthalten (Bild 6).

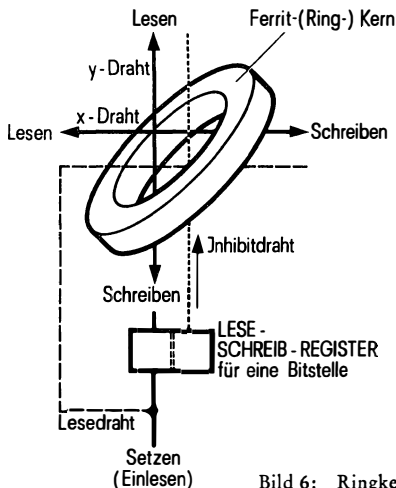


Bild 6: Ringkern mit vollständiger Beschaltung

Der Informationsinhalt eines Kerns läßt sich dadurch feststellen, daß man die magnetisierenden Drähte entgegen der „Setzrichtung“ mit dem Magnetisierungsstrom beschaltet. War vorher die Information 1 enthalten, so entsteht beim Ummagnetisieren in einem zusätzlich durch den Kern geführten Lesedraht ein Stromstoß, der über einen Verstärker zum Setzen eines Flipflop verwendet wird. Damit ist die Kerninformation für die Stromkreise zugänglich.

War vor dem Zuschalten des Lesestroms in den x- bzw. y-Draht die Information 0 im Kern enthalten, so wird im Lesedraht kein Strom induziert, der das Flipflop beeinflussen könnte.

Durch den x- und den y-Draht werden die 24 Kerne eines Wortes immer in den Remanenzzustand 1 gebracht. Wenn einige von ihnen die Information 0 aufnehmen sollen, ist dies nur dadurch möglich, daß man die Wirkung *eines* Drahtes aufhebt. Das Signal dazu stellt das Flipflop. Beeinflusst man die Wirkung des y-Drahtes durch das Signal, das an dem Ausgang der Rücksetzseite auftritt, so wird der y-Draht immer dann wirkungslos, wenn auf der Setzseite des Flipflop ein 0-Signal ist. Der Draht, der das Setzen des Kernes ver-

hindert, heißt Inhibitdraht. Durch ihn erhält der Kern beim Einschreiben die Information 0.

Da das Schreiben oder Lesen im Arbeitsspeicher jedesmal nur einen Kern in jeder Ebene betrifft, genügt es, nur einen Lese- und einen Inhibitdraht durch alle Kerne der Matrix zu führen und sie mit einem Flipflop zu verbinden. Dieses Prinzip bleibt auch erhalten, wenn die Funktion dieser Drähte aus konstruktiven Gründen auf mehrere parallelgeschaltete aufgeteilt wird.

Alle Flipflops für die einzelnen Speicherebenen werden zu dem *Lese-Schreib-Register* zusammengefaßt. Dieses Register hat sozusagen die Funktion eines „Pfortners“ für den ASP. Jede Information, die abgespeichert werden soll, muß vorher im Lese-Schreib-Register stehen, und jede Information aus dem ASP ist erst im Lese-Schreib-Register greifbar.

Diese Darstellung ist stark vereinfacht und zeigt nur das Wesentliche zum Verständnis der Vorgänge um den ASP. Sie muß noch durch den Hinweis auf die Funktion des Taktsystems ergänzt werden, das das Schreiben und Lesen in Taktschritte zerlegt, die nacheinander ablaufen. Ein Taktgenerator stößt diese Schritte an, d. h. er öffnet sozusagen ein Verknüpfungsglied nach dem anderen auf dem Weg der Information durch die Stromkreisabschnitte.

Die 24 Ringkerne, die ein Wort im Arbeitsspeicher aufnehmen, nennt man eine *Zelle*; sie wird mit Hilfe einer Adresse angesteuert. Enthält die Zelle ein Datenwort, so steht ihre Adresse in dem Befehlswort, das eine Operation mit dieser Date veranlaßt. Ist der Zelleninhalt dagegen ein Befehl, der gelesen werden soll, so gibt das Steuerwerk die Adresse vor.

Das Adressieren einer Zelle bedeutet, aus 128 x- und 128 y-Drähten je einen bestimmten auszuwählen. 128 Kombinationen lassen sich mit 7 Bitstellen bilden. Also braucht man 2 mal 7 Bitstellen, um das richtige Drahtpaar festzulegen. Diese 14 Bitstellen werden in einer Decodierschaltung entschlüsselt; der erforderliche hohe Magnetisierungsstrom für die Ringkerne wird durch Schaltermatrizen (mit Leistungstristoren) bereitgestellt (Bild 7).

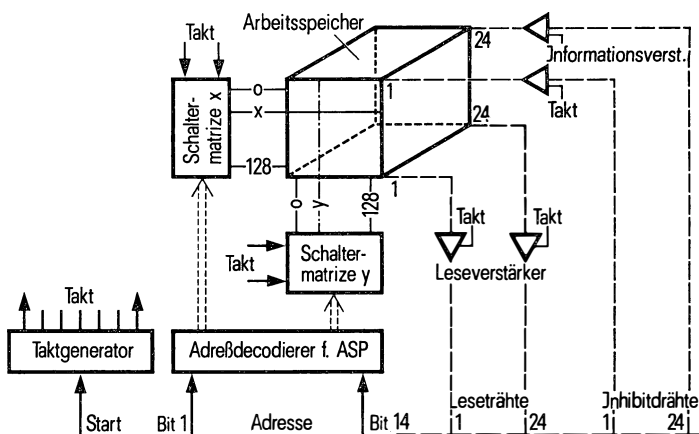


Bild 7: Arbeitsspeicher mit Peripherie

Anmerkung

Unter *Decodierung* versteht man folgendes: Das Anlegen eines Bitmusters an die Eingänge einer Schaltungseinheit hat an den Ausgängen eine „Signalreihe“ zur Folge, die für das gewählte Eingangs-Bitmuster typisch ist, d. h. bei keinem anderen auftreten kann. Die übliche Decodierschaltung hat so viele Ausgänge, wie Variationsmöglichkeiten der Eingangssignale denkbar sind. Mit ihr wird eine Signalkombination so entschlüsselt, daß an einem Ausgang ein Signal erscheint, das sich an keinem anderen wiederholt. Dieses charakteristische Signal benützt man zur Beeinflussung anderer, nachgeschalteter Stromkreise.

Im Prozeßrechner werden Adressen, Befehle und andere Informationen decodiert. Die Wirkungsweise einer Decodierschaltung läßt sich am einfachsten an der gezeigten Relaiskombination erklären (Bild 8).

Der Schaltzustand der Relaispulen und damit die Lage der Kontakte sind maßgebend, an welchen Ausgängen rechts die links angelegte Spannung durchgeschaltet wird. Bezeichnet man den erregten Zustand einer Spule mit 1 und den nicht erregten mit 0, so gibt es für die drei Spulen acht Möglichkeiten für stationäre Betriebszustände (von 000 bis 111). Zu jedem gehört *ein* bestimmter, Spannung führender Ausgang rechts. Das Auftreten dieser Spannung an diesem Ausgang ist dann die Decodierung der am Eingang — an den Relaispulen — angelegten Bitkonstellation.

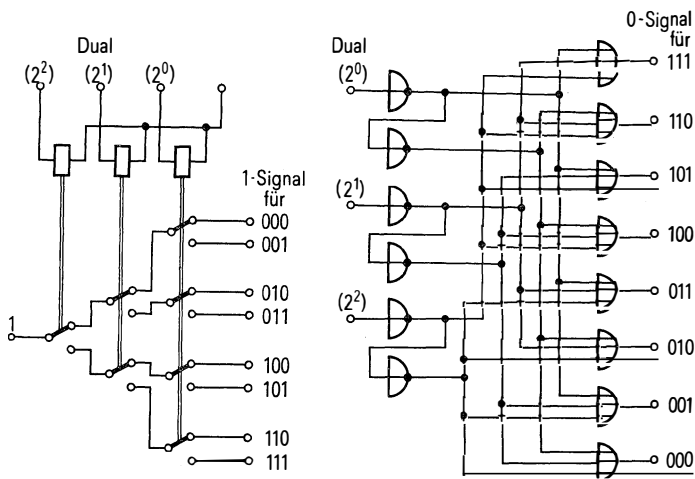


Bild 8: Decodierschaltungen

Zwischen der am Eingang anstehenden Signalkombination und dem Ausgang, der ein von allen anderen Ausgängen abweichendes Signal führt, besteht also ein fester Zusammenhang. Ist das charakteristische Signal bei der Relaischaltung als 1-Signal angegeben, so zeigt die Halbleiterschaltung daneben (Bild 8), daß es auch das 0-Signal sein kann. Beide Decodiereinrichtungen haben gemeinsame Grundzüge in ihrem Aufbau und wurden deshalb aus vielen anderen als besonders instruktiv ausgewählt [3].

3.2. Rechenwerk

Das Rechenwerk ist der Teil des Rechners, in dem die eigentliche Verarbeitung von Daten abläuft. Im Rechenwerk werden auch die Operanden eines Programms nach verschiedenen Gesichtspunkten untersucht, verändert oder zueinander in bestimmte Beziehung gesetzt. Der Prozeßrechner stellt eine DVA dar, die wenig zu rechnen hat; sie wird vornehmlich dazu benutzt, die Bitmuster der Prozeßgrößen zu vergleichen und nach logischen Gesichtspunkten zu verknüpfen.

Anmerkung

So kann man die Funktion des Rechenwerks mit der einer Universal-Werkzeugmaschine vergleichen, die ein Werkstück in ganz unterschiedlicher Weise bearbeiten kann. Eine solche Maschine besitzt in der Regel ein Steuerkonsol mit Anwahltasten, über die man einen Arbeitsgang anwählen kann, der dann automatisch abläuft.

Es gibt Tätigkeiten, die nur eine Date betreffen, die sich schon im Rechenwerk befindet. Das Ergebnis steht anschließend im Rechenwerk zur Verfügung. Bei Arbeitsgängen, an denen zwei Daten beteiligt sind, muß immer der erste Operand im Rechenwerk griffbereit sein, bevor der zweite aus dem ASP geholt und mit dem ersten verknüpft werden kann. Dies ist die wesentliche Eigenschaft von „Einadreßmaschinen“, die mit *einem* Befehl nur *einen* Operanden ansprechen können. Auch nach diesem Arbeitsgang ist das Ergebnis der Operation im Rechenwerk abzuholen.

Da diese Vorgänge mit einem Datentransport aus dem ASP zum Rechenwerk verbunden und nur dann beendet sind, wenn das Ergebnis wieder im ASP sichergestellt ist, zählt man die Befehle für den Datentransfer zum oder vom Rechenwerk auch zu den Verarbeitungsbefehlen. Im Zusammenhang mit dem Rechenwerk werden also folgende Aufgaben des Rechners erledigt:

- Das Rechnen mit Zahlen nach den vier Grundrechnungsarten,
- das logische Verknüpfen von Daten,
- das Verschieben des Bitstelleninhalts eines Wortes und
- der Transport von Daten zwischen ASP und Rechenwerk.

Das Rechenwerk besteht — abhängig vom Rechnertyp — aus einer unterschiedlich großen Anzahl von Registern und Torschaltungen, unter denen die *Akkumulatoren* die Hauptregister darstellen. Die ZE 301 und 302 besitzen je einen, die ZE 303 bis 306 je zwei.

Diese Akkumulatoren sind die Zwischenspeicher für die Daten, die verarbeitet oder als Ergebnis einer Operation in den ASP abgeholt werden sollen. Sind in einer Maschine zwei Akkumulatoren vorhanden, so kann man ein Zwischenresultat von zwei zusammenhängenden Rechengängen jeweils in einem Akku stehen lassen, während mit dem anderen gearbeitet wird.

Die Multiplikation von Zahlen, die jeweils in einem 24-Bit-Wort Platz finden, kann zu einem doppelt so großen Produkt führen. In Umkehrung dazu hat ein Dividend auch die Länge zweier Wörter,

wenn Divisor und Quotient je ein Wort ausfüllen. Auch aus diesen Gründen brauchen Rechner, mit denen durch Hardware möglichst genau multipliziert und dividiert werden soll, zwei Akkumulatoren, um Produkt bzw. Quotienten unterzubringen.

Die Akkumulatoren in den Maschinen vom Typ 303 an aufwärts besitzen zusätzlich eine Stelle \emptyset , mit deren Hilfe Bereichsüberschreitungen beim Rechnen feststellbar sind. Diese Fehler „melden“ sich nicht selbst, sondern müssen im Verdachtsfall nachgeprüft werden.

Also sind die Akkumulatoren Zwischenspeicher des Rechners, die im Verarbeitungsbereich der Daten liegen und zu denen das Steuerwerk Zugriff hat. Andere Register des Rechenwerks können zwar vom Steuerwerk beeinflußt werden; die Möglichkeit zur Datenentnahme aber fehlt. Den Akkumulatoren gleichen in der Funktion nur noch die Exponentenregister der ZE 305 und 306.

Die Daten werden ausschließlich im Addierwerk verknüpft. Wenn behauptet wird, daß ein Rechner nur addieren und Bitmuster verschieben kann, so ist das nicht abwegig; denn alle Hauptrechnungsarten lassen sich auf diese Grundoperationen zurückführen, wenn man bestimmte Voraussetzungen schafft. Dies soll kurz erläutert werden.

Eine Elektronik führt eine *Addition* in der gleichen Weise aus, wie sie der Mensch schriftlich vornehmen würde:

dezimal	130	dual	10 000 010
	+ 46		+ 00 101 110
	176		10 110 000

Nur muß man dabei beachten, daß nach den Regeln der Dualarithmetik $1 + 1 = 10$ ist, daß also schon bei dieser Summierung ein Übertrag für die nächste Stelle entsteht.

Eine *Multiplikation* baut auf der Addition auf:

dezimal	12 · 14	dual	1100 · 1110
	28		0
	14		0 .
	168		1110 . .
			1110 . . .
			10 101 000

Hier folgt auf die Multiplikation mit einer Bitstelle des linken Faktors die Verschiebung des Ergebnisses der nächsten Multiplikation. Da es nur darum geht, ob der Faktor rechts (mit der richtigen Stellenlage) in die Rechnung einbezogen werden muß oder nicht (1 heißt ja und 0 nein), und der Rechner jede Addition in einem getrennten Arbeitsgang erledigen muß, besteht das Multiplizieren für die Maschine nur aus Addieren und Verschieben. Und Verschieben ist in Halbleiterschaltungen eine leicht realisierbare Tätigkeit.

Eine *Subtraktion* setzt voraus, daß man im Rechner positive und negative Zahlen voneinander unterscheiden kann. Dies wurde bei den bisherigen Betrachtungen außer Acht gelassen. Da man unter einer Subtraktion die Addition einer positiven mit einer negativen Zahl versteht, muß man diese einfache Rechenregel auch dem Addierwerk „begreiflich“ machen.

$$\text{Beispiel} \quad + 142 + (- 87) = + 55,$$

Man muß also gewisse Vereinbarungen treffen, die beim Rechnen mit verschlüsselten Zahlen, die negativen Wert haben, zu beachten sind.

Die erste Vereinbarung ist, immer ein ganzes Maschinenwort für die Darstellung einer Zahl heranzuziehen, auch wenn nach links viele Stellen mit Null zu besetzen sind.

Die Zahl darf aber auch nur ein Register füllen (Ausnahmen bei Rechenergebnissen in zwei Akkumulatoren). Überzählige Stellen werden durch Auf- oder Abrundung aus dem betrachteten Bereich herausgenommen. Damit liegen die Grenzen für arithmetische Operationen fest.

Zum zweiten wird vereinbart, daß eine positive Zahl auf der Bitstelle 1 mit 0, eine negative mit 1 gekennzeichnet wird. Damit besteht jede Zahl im ASP grundsätzlich aus der Vorzeichenstelle und 23 Betragstellen. Die Vorzeichenstelle wird wie die Betragstellen mitverarbeitet.

Die dritte Vereinbarung besagt, daß eine negative Zahl als „Zweierkomplement“ der betragsmäßig gleichgroßen, positiven Zahl dargestellt wird. Unter einem Zweierkomplement versteht man hier die Differenz des Betrages zwischen der positiv gleichgroßen Zahl und der

kleinsten Zahl, die man mit 24 Bitstellen darstellen kann. Ein *Beispiel* mit einer überschaubaren Stellenzahl soll dies erläutern:

In ein Register mit zehn Bitstellen kann man das Vorzeichen und eine Zahl mit neun Betragstellen einschreiben. Die höchste darstellbare positive Zahl ist $(511)_{10}$, das entspricht dual 111 111 111. Das Zweierkomplement wird also mit $(512)_{10}$ entsprechend dual 1 000 000 000 gebildet.

Die oben angeführte Rechnung enthält 87 als negative Zahl. Das Zweierkomplement hat dann den Wert: $512 - 87 = 425$.

Die Operation lautet:

dezimal	+ 142	dual	0 010 001 110
	+ (— 87)		+ 1 110 101 001
	+ 55		10 000 110 111

Die 1 aus dem Überlauf der ersten Stelle kann im Register nicht mehr erscheinen und spielt deshalb keine Rolle mehr. Das Ergebnis der Addition zeigt die Zahl 55 mit dem richtigen positiven Vorzeichen.

Für die Rechnerstromkreise ist diese „Umcodierung“ der negativen Zahlen einfach durchzuführen. Vergleicht man die Zahl 87 mit der Zahl 425, — also dual 001 010 111 mit 110 101 001, — so erkennt man, daß sich die negative Zahl aus der positiven durch die Invertierung jedes Bits und die Addition von 1 auf die Stelle mit dem niedrigsten Wert bilden läßt.

Das Rechenwerk muß also für den Umgang mit negativen Zahlen mit einem Inverter und einem Einsaddierer versehen sein, die diese Zahlen vor dem Durchlauf durch das Addierwerk in die richtige Form bringen.

Eine *Division* kann auf die wesentlichen Ausführungsbestimmungen für die bisher behandelten Rechnungsarten zurückgeführt werden. Es gibt verschiedene Algorithmen, nach denen man das Konzept für den Schaltungsaufbau entwerfen kann. So muß wie gesagt beim Siemens System 300 der Dividend zu Beginn der Operation im linken und rechten Akkumulator stehen. Der Divisor wird durch ein ganzes Wort dargestellt, so daß der Dividend über 46 und der Divisor über 23 Be-

tragsstellen verfügt. Das Ergebnis steht immer im rechten Akkumulator. Folglich muß dafür gesorgt sein, daß kein Ergebnis mit mehr als 23 Betragstellen entsteht. Zu diesem Zweck muß der Divisor u. U. um Potenzen von 2 erhöht werden, was bei der Auswertung des Rechenergebnisses zu berücksichtigen ist.

Eine einfache Division kann bei reduzierter Stellenzahl so ablaufen:

<p>dezimal $+ 168 : + 14 = + 12$</p> $ \begin{array}{r} - 14 \\ \hline 28 \\ - 28 \\ \hline - - \end{array} $	<p>dual $0010101000 : 01110 = 001100$</p> $ \begin{array}{r} 10010 \\ \hline 101110 \\ \hline 01110 \\ \hline 111001 \\ \hline 01110 \\ \hline 001110 \\ \hline 10010 \\ \hline 000000 \\ \hline 10010 \\ \hline 100100 \\ \hline 01110 \\ \hline 10010 \end{array} $
---	--

Die beim Dividieren von Hand üblichen Subtraktionen werden im Rechenwerk zu Additionen. Dabei schreibt die Verarbeitungsregel vor, wann der Divisor direkt oder invertiert addiert werden muß. Das Beispiel zeigt die Abhängigkeit der Ergebnisziffer (rechts) von der ersten Stelle des Additionsergebnisses bei jeder Teiloperation. Der Rest des Dividenden, der bei der Rechnung mit Dezimalzahlen nicht auftritt, ist eine „Ungenauigkeit“ des Algorithmus. Er ist im linken Akkumulator greifbar und kann daraufhin überprüft werden, ob er zu Recht eingetragen ist.

Durch diese Beispiele für die vier Grundrechnungsarten soll nur in kurzer Form erklärt werden, daß das Rechenwerk eines Rechners relativ einfach aufzubauen ist, wenn man geeignete Algorithmen für die Rechenoperationen wählt. Unter diesem Blickwinkel ist auch das Rechnen mit Brüchen zu betrachten.

Die *Festkommarechnung* geht davon aus, daß das manuelle Rechnen mit Dezimalbrüchen das Komma zunächst nicht berücksichtigt. Das Komma wird vor der Rechnung aus den Zahlen entfernt und anschließend wieder stellengerecht eingetragen. Man arbeitet also mit ganzen Zahlen, wobei nur Ziffern mit gleichem Stellenwert zueinander in Beziehung gebracht werden.

Da der Rechner keine Funktionseinheiten für Kommaverarbeitung besitzt, muß der Programmierer das Komma im Programm in der notwendigen Weise berücksichtigen. Diese Methode des Bruchrechnens enthält für eine Automatik zahlreiche Fehlermöglichkeiten und ist deshalb schwierig zu handhaben. Zur Lösung von umfangreichen mathematischen Aufgaben ist sie nicht verwendbar.

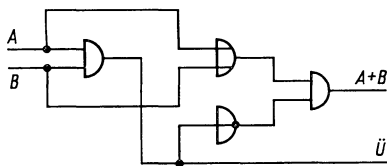
In einem solchen Fall bietet sich die *Gleitkommarechnung* an, d. h. das Rechnen mit Bruchzahlen, die als „Zahlenrest“ (Mantisse) und Exponent ausgedrückt werden. Die Kommastellung geht in die Größe des Exponenten ein. Beide an der Operation beteiligten Zahlen werden stellenrichtig verknüpft, auch wenn die Exponenten unterschiedlich sind. Jede Verschiebung der Mantisse verändert auch den Exponenten. Das Ergebnis erscheint normalisiert; dabei ist die duale Mantisse im Akkumulator soweit nach links geschoben, daß die Bitstellen 1 und 2 unterschiedlich besetzt sind (bei positiven Zahlen z. B. 01). Das Komma steht dann zwischen Vorzeichen- und erster Zahlenstelle (dezimales Beispiel: 0,0052 wird $0,52 \cdot 10^{-2}$).

In den ZE 305 und 306 sind die Gleitkommarechnungen Hardwareoperationen. Die Normalisierung der beteiligten Zahlen nimmt der Rechner selbst vor. Für die Exponentenrechnung besitzen diese Zentraleinheiten ein eigenes Verknüpfungssystem, in dem das Vorzeichen auch richtig mitbehandelt wird.

Anmerkung

Der mitunter zu findende Ausdruck „Gleitpunktrechnung“ stammt aus dem Angelsächsischen; dort wird das bei uns gebräuchliche Komma als Punkt dargestellt.

Der Aufbau des Rechenwerks ist zum größten Teil mit elementaren Verknüpfungsgliedern (Gattern) [4], [5] zu beschreiben. Dies soll an zwei überschaubaren Beispielen erklärt werden.

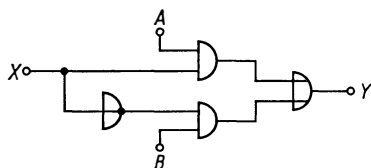


A	B	$A + B$	\ddot{U}
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Bild 9: Prinzipschaltung und Wertetabelle eines Halbaddierers

Der in Bild 9 gezeigte Halbaddierer kann zwar zum Addieren von zwei Ziffern benützt werden. Ihm fehlt jedoch die Auswertung eines Übertrags. Der Volladdierer bezieht in jeder Stufe (d. h. für jede Stelle) den Übertrag der vorhergehenden in die Rechnung mit ein.

Als Beispiel für das „Umschalten“ einer Verarbeitungseinheit durch das Anlegen verschiedener Signalzustände an bestimmte Eingänge sei eine Schaltung zur Komplementbildung angeführt (Bild 10):



X	A	B	Y
0	1	0	0
0	0	1	1
1	1	0	1
1	0	1	0

Bild 10: Prinzipschaltung und Wertetabelle eines Inverters

Setzt man voraus, daß der Weg der Information von X nach Y verläuft, so sieht man, daß das Anlegen von bestimmten (einander entgegengesetzten) Signalen an A und B den Durchlauf des Informationsbits beeinflusst. Über die Eingangssignale an A und B würde also vorgeschrieben, ob der Inverter positive Zahlen unverändert oder negative invertiert weitergeben soll.

Das entsprechende Umschaltkommando wird vom Befehl im Programm abgeleitet, das heißen kann: „Transferiere Ein Plus“ oder „Addiere“ bei positiven, bzw. „Transferiere Ein Minus“ oder „Subtrahiere“ bei negativen Zahlen. Das macht also deutlich, daß die Verarbeitung der Daten im Rechenwerk tatsächlich wie durch eine Tastenanwahl vor sich geht, da die Stromkreise dieser „Universalmaschine“ — umschaltbar für mehrere Anwendungsfälle — unterschiedliche Funktionen ausüben können.

Im Rechenwerk laufen alle Vorgänge taktgesteuert ab. Hierzu wird die Taktfolge des Taktgenerators am Arbeitsspeicher übernommen und zum Teil übersetzt.

Das Prinzip des Rechenwerks wird durch Bild 11 anschaulich.

Die Addition von zwei Operanden besteht aus drei Arbeitsschritten. Beim ersten Befehl des Programms wird der erste Operand aus seiner Arbeitsspeicher-Zelle über das Lese-Schreib-Register in den Akkumulator gebracht, beim zweiten Befehl läuft der zweite Operand von seiner ASP-Zelle über das Lese-Schreib-Register durch den auf „Durchlaß“ geschalteten Inverter zum Addierwerk und wird dort mit dem Inhalt des Akkumulators summiert. Das Ergebnis gelangt in den Akkumulator, in dem der erste Operand vorher war, zurück. Der dritte Befehl holt diese Summe aus dem Akkumulator über das Lese-Schreib-Register in die dritte Zelle des ASP.

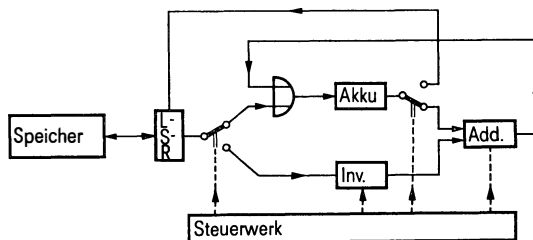


Bild 11: Prinzip des Rechenwerks

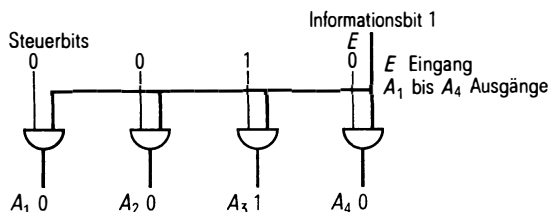


Bild 12: „Mehrfachweiche“ für ein Informationsbit

Bild 12 weist darauf hin, daß es zwischen den einzelnen Baugruppen „Weichen“ gibt, die den Daten den Weg weisen. Die Verstellung dieser Weichen ist Aufgabe des Steuerwerks.

Anmerkung

Unter einer Weiche kann man sich UND-Stufen vorstellen, denen die Information gemeinsam angeboten wird; sie läuft aber nur über *die* UND-Stufe weiter, an der mit dem Schaltsignal des Steuerwerks die UND-Bedingung erfüllt ist.

Solche Weichen für Informationen sind in der Halbleitertechnik üblich. Sie werden auch zur schematischen Darstellung von Vielfachschaltern benützt.

3.3. Steuerwerk

In dem Rechnerteil *Steuerwerk* (Bild 13) sind alle Register und Torschaltungen zusammengefaßt, die zum Lesen eines Befehls und zum Transportieren der Daten in der Zentraleinheit dienen. Es sind dafür bestimmte Vorschriften notwendig.

Wie schon erwähnt, läßt sich das Steuerwerk mit der Belegschaft einer Fabrik vergleichen, die das im Lager deponierte Material zur Verarbeitungsmaschine bringt, dort in bestimmter Weise behandeln läßt und das Endprodukt im Fabriklager wieder ablegt. Es ist auch Aufgabe dieses Personals, Material von externen Fertigungstätten (vgl. Lochkarten- und Lochstreifengeräte und Prozeßelemente) und von

fehlsfolgen wird nur durch Sprünge erreicht. Aus Gründen der Übersichtlichkeit folgen die Daten den Befehlen in geschlossenen Blöcken oder stehen in einem gemeinsamen Datenteil.

Der Ausgangspunkt für jeden Arbeitsschritt im Steuerwerk ist der Befehlszähler. Dieser stellt ein Register dar und enthält jeweils die Adresse der ASP-Zelle, in der der nächste auszuführende Befehl steht. Deshalb muß der Befehlszähler auch zu Beginn jeder Rechnerarbeit „geladen“ werden.

Das Adreßregister nimmt jede Adresse auf, die ihm vom Befehlszähler, vom Befehlsregister oder von der Steuerung eines EXE zur Ansteuerung einer ASP-Zelle übergeben wird. Es hat 16 Bitstellen, weil es — wie weiter unten noch erklärt wird — mit den beiden letzten Stellen dafür sorgen muß, daß Teilwörter bei einem Externverkehr in die richtige Position oder aus den richtigen Bitstellen des Lese-Schreib-Registers transportiert werden.

Damit der Befehlszähler immer die aktuelle Programmfortsetzadresse enthält, erhöht der Adreßaddierer den Zählerinhalt um eins, wenn ein Befehl aus dem ASP gelesen worden ist.

Zur Untersuchung des Inhalts eines Befehlswortes wird hinter das Lese-Schreib-Register das Befehlsregister geschaltet. Der Befehl hat — wie erwähnt — eine durch die Hardware vorgeschriebene Struktur. Seine letzten sechs Bitstellen enthalten den codierten Befehl, aus dem eine Decodierschaltung entnimmt, was mit den ersten 14 Bitstellen geschehen soll. Vom Befehlsregister aus kann der Adreßteil in das Adreßregister (zur Ansteuerung eines Operanden), in den Befehlszähler (als Befehlsadresse beim Übergang auf einen neuen Programmteil), in das Rechenwerk (als Verschiebezahl) oder an ein EXE weitergegeben werden. Die Stellen 2 bis 5 werden u. U. auch dem Elementauswahlregister übergeben. Hier stellt also die Decodierschaltung die entsprechenden „Weichen.“

Das Prioritätenetzwerk (Bild 14) entscheidet, für welche Tätigkeit des Steuerwerks der jeweils nächste Arbeitsspeicher-Zyklus freigegeben wird.

Stehen an mehreren Flipflops auf der Setzseite Anforderungen auf einen ASP-Zyklus an, so wird die UND-Bedingung am Ausgang nur für diejenige Anforderung erfüllt, die die jeweils höchste Priorität besitzt. Höchste Priorität wird durch die niedrigste Nummer ausge-

drückt. Wie zu sehen ist, läßt das 0-Signal der Rücksetzseite eines angesteuerten Flipflops bei keinem Schaltungsausgang mit niedrigerer Wertigkeit ein 1-Signal durchkommen. Erst wenn die Anforderung der höherpriorioren Stufe rückgesetzt ist, kommt die der nächstniedrigen zum Zug.

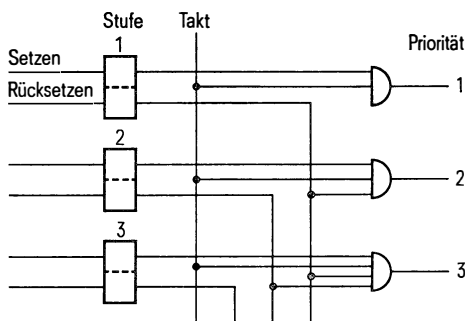


Bild 14: Prinzipschaltung eines Prioritätenetzwerks

Diese Schaltung kann allen Beschreibungen zugrundegelegt werden, in denen von einer geregelten Bevorzugung eines bestimmten Stromkreises bzw. dessen Tätigkeit gegenüber anderen die Rede ist.

3.4. Ein-Ausgabekanal

Das Bindeglied zwischen den bisher behandelten Rechnerbestandteilen und den Externen Elementen ist der Ein-Ausgabekanal (E/A-Kanal). Er erfüllt — wie erläutert — im übertragenen Sinn die Aufgabe eines Pfortners, der am Fabriktor die Ein- und Ausgabe des Materials überwacht, der ferner bei einem Warenempfang die Weiterleitung durch die Fabrik durch telefonische Mitteilungen vorbereitet und für einen geordneten Ablauf des Meldungs- und Warenaustauschs verantwortlich ist.

Im E/A-Kanal des Prozeßrechners müssen zwei Dinge besonders beachtet werden.

Erstens ist es notwendig, Informationen von außen so zeitsparend wie möglich in die Rechnerbearbeitung einzuschleusen und Informatio-

nen aus dem ASP bei möglichst geringer Beanspruchung der ZE durch die EXE abholen zu lassen.

Zweitens muß man den Signalpegel anpassen, da die ZE 302, 304, 305 und 306 für ihre (integrierten) Halbleiterschaltungen der ECL-Technik niedrigere Signalspannungen ($< 5\text{ V}$) benützen als die externen Steuerungen aus SIMATIC-H-Bauteilen (mit 12 V).

Wegen des kompakten Aufbaus der zentralen Steuerung einer Rechneranlage — nur die externen Geräte stehen im Rechnerraum verteilt — kann man in ihr nicht so deutlich, wie in anderen elektrotechnischen Anlagen, Klemmleisten als „Bereichsabschlüsse“ erkennen. So sind im Rechnerschrank auch die Steuerungen von externen Elementen eingebaut, deren Verbindungen zur ZE nur für Eingeweihte erkennbar sind. Diese Übergänge von einem Teil („Bereich“) der Rechneranlage zum anderen heißen generell *Nahtstellen*.

Die Nahtstellen zu gleichartigen Gerätesteuern hin haben einen einheitlichen Aufbau. Sie werden nach *Standardnahtstellen* und *Nahtstellen am Schnellkanalzusatz* unterschieden. Standardnahtstellen übergeben die Informationswörter teilwortweise, während Schnellkanal-Nahtstellen ganze Wörter weiterleiten. Wie in der ZE selbst werden die Informationen mit den EXE bitparallel ausgetauscht. Die ZE 302 und 303 besitzen keine Schnellkanal-Nahtstellen und nur 5 bzw. 6 Standardnahtstellen. Die größeren Rechnertypen sind mit 10 Standardnahtstellen und maximal 5 Schnellkanal-Nahtstellen ausgestattet. Die Geräteverbindungen über jede Nahtstelle werden *Kanal* genannt. Wie später beschrieben wird, unterscheidet man zwischen einem Programm- und einem Datenkanal an jeder Anschlußstelle des E/A-Kanals. Der Programmkanal enthält die Leitungen für die Befehlsübergabe an die Steuerung des EXE und der Datenkanal die Leitungen für den Datentransport und die ASP-Adressierung. Dazu kommen in beiden Kanälen noch mehrere Signalleitungen.

Da an eine ZE normalerweise mehrere EXE angeschlossen sind, die je einen Kanal belegen, ist für ihren Verkehr mit der ZE die Kanalpriorität wichtig; sie richtet sich nach sehr unterschiedlichen Gesichtspunkten. Man könnte dem mechanisch langsamsten Gerät, das zudem noch Teilwortverkehr abwickelt, eine hohe Priorität zugestehen. Dabei wäre berücksichtigt, daß es bei einer größeren Operation auf jeden Fall sofort einen neuen Arbeitsgang einleiten kann, wenn der vorher-

gehende abgeschlossen ist. Man muß aber auch dem EXE, das bei einem Fehler im Prozeßverlauf oder in den daran beteiligten Maschinen ein gerade laufendes Programm unterbrechen und z. B. die Stillsetzung veranlassen kann, hohe Priorität zubilligen. Dann könnte der Rechner bestimmt zeitgerecht reagieren.

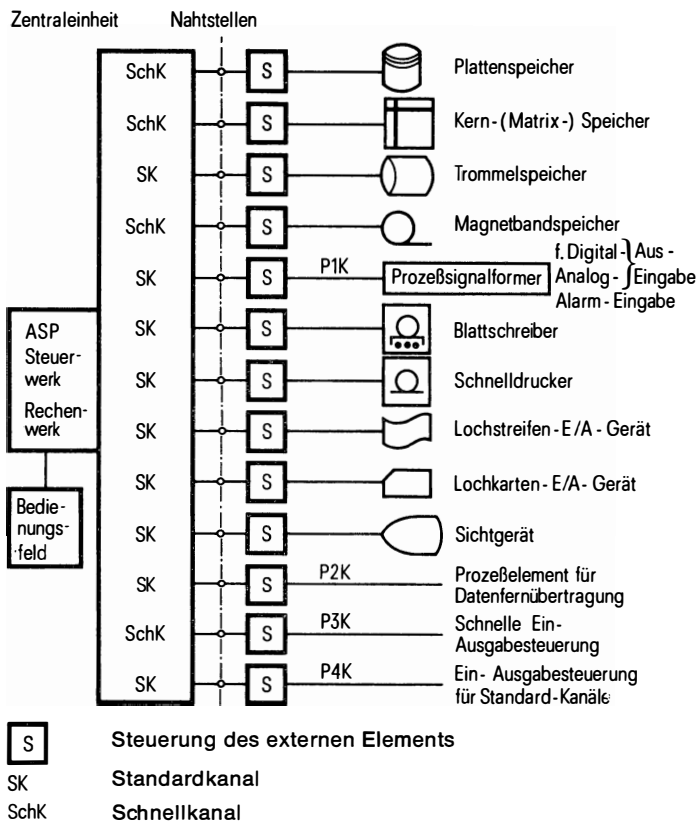


Bild 15: Zentraleinheit und anschließbare EXE

Aus dem Betriebsverhalten und der Bedeutung jedes externen Elements in der Rechneranlage hat man eine bestimmte „Rangordnung“ ermittelt und die Richtlinien für eine Einreihung der EXE in einem Programm zusammengefaßt. Mit diesem wird die Gerätekonzellation jeder in Auftrag genommenen Rechneranlage untersucht und als Ergebnis die Prioritätsstufe jedes EXE, die in der Kanalnummer zum Ausdruck kommt, festgelegt.

Da die externen Elemente wegen ihrer mechanischen Eigenschaften unterschiedlich schnell arbeiten und der Datentransport zwischen jeder Gerätesteuerung und der ZE innerhalb sehr kurzer Zeit stattfindet, kann die ZE ohne weiteres mehrere Geräte gleichzeitig arbeiten lassen. Diese *Simultanarbeit* wird durch die Einteilung nach Kanalprioritäten nur dann beeinflußt, wenn mehrere EXE im gleichen Augenblick eine Anforderung auf Datenverkehr mit der ZE stellen. Die Wartezeit des EXE mit der niedrigsten Priorität ist aber meistens so kurz, daß sie keinen spürbaren Einfluß auf seine Gesamtarbeitszeit hat.

Die externen Elemente, die mit den Zentraleinheiten 302, 303 und 304, 305, 306 zusammenarbeiten können, sind in Bild 15 dargestellt.

3.5. Ausnahmen

Aus den bisherigen Betrachtungen wurden wiederholt die Zentraleinheiten 301 und 306 ausgeklammert. Beide weisen in Aufbau und Wirkungsweise so grundlegende Unterschiede gegenüber den anderen ZE der Rechnerfamilie 300 auf, daß sie getrennt besprochen werden müssen.

Die ZE 301 (Bild 16) ist die kleinste aus der hier behandelten Rechnerfamilie. Sie wurde an verschiedenen Stellen einfacher ausgeführt, arbeitet aber nach gleichen organisatorischen Prinzipien wie z. B. die ZE 302. Ihre internen Befehle sind voll kompatibel¹⁾. Ein Programm, das für eine größere Maschine geschrieben wurde, läuft (unter Einschränkungen) auch auf dem Rechner 301.

¹⁾ „Kompatibilität ist hier die Fähigkeit unterschiedlicher Anlagentypen, ein Programm mit einer bestimmten Codierung direkt zu verstehen“; Voraussetzung dafür ist, daß eine Maschine, deren Hardware einzelne Befehle nicht decodieren kann, von selbst eine Umsetzung dieser Befehle in eine geeignete Software mit für sie lesbaren Befehlen einleitet.

Die Änderungen betreffen vor allem die Nahtstellen, den Arbeitsspeicher und die verwendete Schaltkreistechnik. Folgende Einzelheiten sind erwähnenswert:

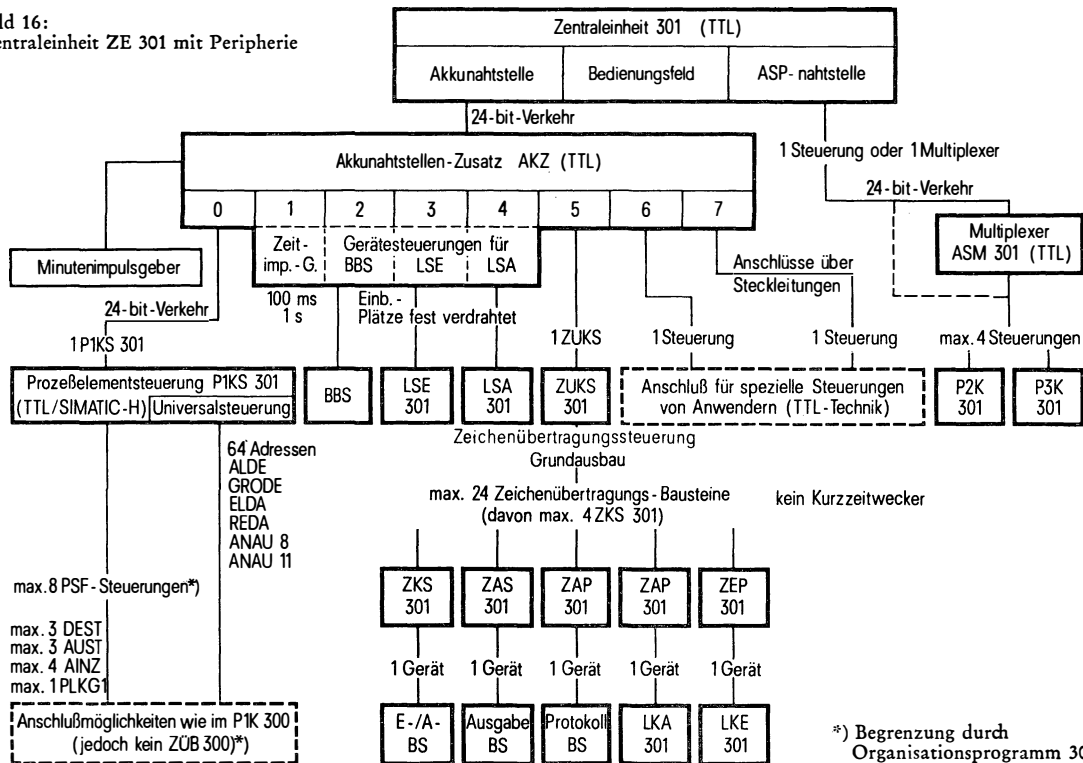
1. Der Arbeitsspeicher läßt sich aus Einheiten zu je 4K Wörtern — bis zu einem maximalen Speichervermögen von 16K Wörtern — zusammensetzen. Jede Erweiterung um 4K Wörter bedeutet die Montage eines zusätzlichen, fabrikfertig voll bestückten Rahmens mit den erforderlichen Steckverbindungen.
2. Die ZE 301 ist aus integrierten Halbleiterschaltungen in TTL-Technik aufgebaut. Ein ASP-Zyklus dauert 1,6 μ s.
3. Die ZE 301 besitzt keinen Ein-Ausgabekanal. Daher können nur Signale mit dem Pegel der TTL-Technik ausgegeben oder von externen Steuerungen übernommen werden.
4. An der ZE 301 gibt es nur zwei Nahtstellen. Beide übertragen Wörter mit 24 Bitstellen parallel.

Die Akkunahtstelle liegt direkt am Akkumulator.

Die Arbeitsspeichernahtstelle ähnelt der Schnellkanal-Nahtstelle einer ZE 305 mit dem unmittelbaren Zugriff zum ASP.

5. Für die üblichen Ein- und Ausgabegeräte sind am *Akkunahtstellen-Zusatz AKZ* feste Anschlußplätze vorgesehen. Die programmtechnische Organisation schreibt diese Plätze verbindlich vor. Die hier anschließbaren Geräte sind nicht gegen gleichartige der ZE 302 usw. auswechselbar (mit Ausnahme der Signalformer des Prozeßelements P1K), da ihre Steuerungen den Nahtstellen-Bedingungen genügen müssen.
6. Über die Arbeitsspeicher-Nahtstelle kann jeweils nur eine Gerätesteuerung verkehren. Durch Einsatz des *ASP-Multiplexers* ASM stehen maximal vier Anschlußmöglichkeiten für externe Gerätesteuerungen zur Verfügung. Dieser Multiplexer ist mit einem gesteuerten Vielfachschalter vergleichbar. Bei einer Ausgabe wird er durch einen Befehl der ZE so gestellt, daß ein Datenverkehr mit der gewünschten externen Gerätesteuerung möglich ist. Für eine Eingabe besitzt der Multiplexer ein Prioritätsnetzwerk in vier Stufen, das den Datenverkehr für die Steuerung zuläßt, deren Anforderung die höchste Priorität hat.

Bild 16:
Zentraleinheit ZE 301 mit Peripherie



*) Begrenzung durch
Organisationsprogramm 301

Bei dieser ZE stellt der Akkumulator einen Zwischenspeicher für externen Datenverkehr dar. Vor einer Ausgabe ist jedes Wort aus dem ASP mit einem Transferbefehl dorthin zu bringen; vom Akkumulator holt es die Steuerung des externen Elements ab. Eine Eingabe endet gleichfalls mit der Ablage des Wortes im Akkumulator, von wo es durch das Programm in den ASP befördert werden muß.

Zwischen dem Lese-Schreib-Register und der ASP-Nahtstelle befinden sich in der ZE 301 keine Zwischenregister. Die externen Steuerungen arbeiten mit dem ASP so rasch zusammen, daß das auszugebende Informationswort noch während des laufenden ASP-Zyklus vom ASP bis zum Zwischenregister der Steuerung gelangt.

Die ZE 306 (Bild 17) ist die größte Zentraleinheit des Siemens Systems 300. Hauptsächlich ist sie durch einen Speicherplatz von mehr als 16K Wörtern im ASP und durch gesteigerte Eingriffsmöglichkeiten in den Programmablauf gekennzeichnet; nebenbei gestattet sie höhere Rechengenauigkeit. Diese Eigenschaften setzen ein stark geändertes Steuerwerk und ein anders konzipiertes Rechenwerk voraus.

Die Nahtstellen zur Peripherie stimmen mit denen der ZE 305 überein. Somit gibt es also zehn Standardkanäle und maximal fünf Schnellkanalzusätze. Neu kommt eine Nahtstelle für „schnelle Alarmer“ hinzu, die durch den Anschluß einer „Steuerung zur Unterbrechung des Programms durch schnelle Alarmer“ (SUSA) auf 24 Eingabemöglichkeiten erweitert werden kann. Diese „schnellen Alarmer“ veranlassen eine unbedingte Programmunterbrechung. Sie sollten nur durch außergewöhnliche Vorkommnisse in der Rechnerperipherie oder gegebenenfalls im Prozeßverlauf ausgelöst werden, wenn die Funktion der Rechneranlage in Frage gestellt ist; dies gilt auch für Vorkommnisse, die so selten auftreten — aber dabei sehr wichtig sind — daß das normale Arbeiten der Rechneranlage nicht zu stark beeinträchtigt wird.

Der E/A-Kanal nimmt auch hier — wie in der ZE 305 — die Signalumsetzung von der ECL- zur SIMATIC-H-Technik und umgekehrt vor.

Der ASP der Zentraleinheit ZE 306 setzt sich aus 16K-Einheiten, auch Moduln genannt, zusammen; sie lassen die Speicherung von maximal 64K Wörtern zu. Da diese Maschine auch nur Wörter mit 24 Bitstellen parallel verarbeitet und die Struktur des Befehlswortes nicht grundsätzlich geändert werden sollte, lassen sich die Zellen des

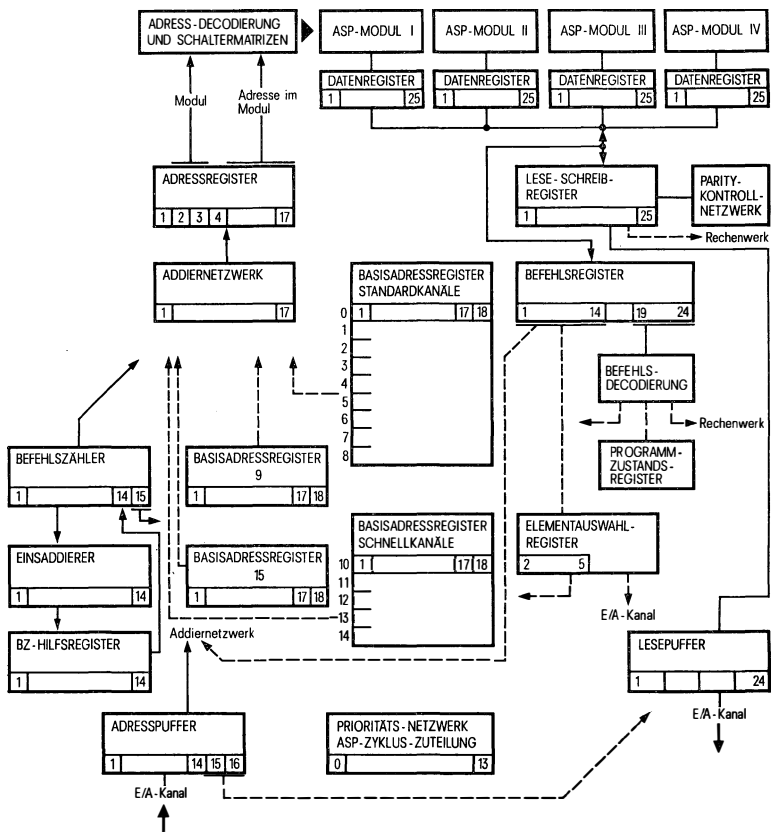


Bild 17: Steuerwerk der Zentraleinheit ZE 306

ASP bei einem Ausbau von mehr als 16K Wörtern nicht mehr direkt adressieren. Die notwendige 16-Bit-Adresse entsteht durch die Addition einer sogenannten „virtuellen“ Adresse mit 14 Bitstellen und dem Inhalt eines Basisadreibregisters, aus dem der Anfangspunkt der Adreßberechnung entnommen wird.

Jeder Modul besitzt ein Datenregister, das die Funktion des Lese-Schreib-Registers bei kleineren ZE übernimmt und durch ein — für alle vier Moduln gemeinsames — „Modus“-Register für die Schreib- und Löschvorgänge im Speicher beeinflusst wird.

Das Lese-Schreib-Register der ZE 306 hat gegenüber den Datenregistern übergeordnete Funktionen, behält aber seine bisherige Pförtner-Stellung vor dem ASP bei. Jedes Wort in einer ASP-Zelle besteht hier aus 25 Bitstellen. Das 25. Bit ist ein Paritybit (Kontrollbit); es wird in einem Parity-Kontroll-Netzwerk, das ungerade Parität herstellt, gebildet und in die Zelle mit eingeschrieben. Gelangt ein gelesenes Wort in das Lese-Schreib-Register, so wird es wieder auf ungerade Parität geprüft und das enthaltene Paritybit mit dem Prüfungsergebnis verglichen.

Jedes Ausgabewort auf dem Weg zur Peripherie durchläuft wie jedes Eingabewort auf dem Weg zum ASP einen „Lesebuffer“; dieser wurde vorgesehen, damit der Datenaustausch mit den EXE in einer für die externe Steuerung angemessenen Geschwindigkeit ablaufen kann. Die Umsetzung der Geschwindigkeit wurde auf ein Verhältnis 1 : 3 (bzw. 0,6 : 1,8 μ s) zwischen Steuerwerk und E/A-Kanal festgesetzt. An diesem Lesebuffer wird auch die Zerlegung des ASP-Wortes in Teilwörter für die Elemente an den Standardkanälen und das Zusammensetzen der Teilwörter bei einer Eingabe vorgenommen. Die Teilwortsteuerung erfolgt durch die Adresse, die ein EXE in den Adreßbuffer schreibt.

Das Befehlsregister erhält die Befehlsörter direkt von den Datenregistern. Das Elementauswahlregister und die Befehlsdecodierung werten den Adreß- und Befehlsteil eines Befehlswortes wie bei der ZE 305 aus. Wesentliche Unterschiede ergeben sich dadurch, daß die Adreßbits durch zusätzliche Markierungen auf den Bitstellen 16 und 18 mehrfach interpretierbar sind. Diese Zusätze beziehen sich vor allem auf Befehle, die zur Organisation des Rechnerbetriebs bei den geänderten Speicherverhältnissen usw. neu hinzukommen.

Die Einrichtungen zur Speicheradressierung haben gegenüber der Zentraleinheit ZE 305 ein stark verändertes Aussehen. Der Befehlszähler behält zwar seine Funktion, enthält aber mit seinen 14 Adreßbits nur dann eine gültige Adresse, wenn diese im Bereich des ersten Speichermoduls (zwischen 0 und 16 383) liegt. Da sein Inhalt in jedem anderen Fall vergrößert werden muß — was außerhalb dieses Registers zu geschehen hat — benötigt man für die Erzeugung der Folgeadresse im Programm außer dem Einsaddierer noch ein Befehlszähler-Hilfsregister. Die *Basisadreßregister (BAR)*, die durch das Organisationsprogramm geladen werden, enthalten jeweils eine Adresse, die gewissermaßen den Anfangspunkt für einen 16-K-Speicher mit beliebiger Lage im ASP darstellt. Für einen normalen Programmablauf werden hier zwei solche Basisadressen als ausreichend erachtet, weshalb man BAR 9 und BAR 15 zur engen Zusammenarbeit mit dem Befehlszähler heranzieht. Die Bitstelle 15 im Befehlszähler gibt bei jeder Adresse an, welche Basisadresse (aus BAR 9 oder aus BAR 15) zu dieser virtuellen Adresse dazuzuzählen ist, damit die richtige Adreßzahl entsteht. Der Inhalt des Befehlszählers und der Inhalt des angewählten BAR werden im Addiernetzwerk zusammengezählt und dann als Summe im Adreßregister abgespeichert.

Operandenadressen aus dem Befehlsregister werden in gleicher Weise wie der Befehlszählerinhalt behandelt; nur gibt hier die Bitstelle 16 des Befehlswortes an, ob mit Basisadreßregister BAR 9 oder BAR 15 eine reelle Adresse gebildet werden soll.

Für die Erzeugung von ASP-Adressen ist jedem EXE ein eigenes BAR zugeordnet (mit Ausnahme des Bedienungselements). Dieses BAR hat die Nummer des Kanals, an den das betreffende EXE angeschlossen ist. Somit wird automatisch zu jeder ASP-Adresse, die von einem EXE kommt, der Inhalt des dazugehörenden BAR addiert. Eine Ausnahme bilden die Adressen, die zu Fehlermeldungen und sonstigen Anzeigen der EXE gehören; diese Adressen liegen im Bereich niedriger als 1024 und zwischen 16 256 und 16 383.

Die Addition einer virtuellen Adresse, also einer „Hausnummer“ im Bereich zwischen 0 und 16 383 für ein in sich abgeschlossenes Programm, mit der Basisadresse für diesen Programmkomplex nennt man *Adreßmodifikation*. Die dadurch entstehende Adresse ist reell.

Unter einer *Adreßtransformation* wird verstanden, daß einer binären Adreßzahl einige Bitstellen von links weggenommen werden, so daß

man z. B. aus der Zahl 16 256 (dual: 111 111 100 000 00) durch Hardware-Einrichtungen die Zahl 384 (dual: 1 100 000 00) entstehen läßt. „Hardware-Adressen“ aus EXE-Steuerungen, die im Bereich über 16 255 liegen, können so auf einfache Weise in einen anderen Bereich gebracht werden.

Für die Adressierung gelten in der ZE 306 folgende Hauptregeln: Jedes Anwenderprogramm benützt virtuelle Adressen, die durch das Betriebssystem des Rechners — das Organisationsprogramm — in einen bestimmten Bereich gelegt werden. Es erfolgt also eine Adreßmodifikation.

Das Organisationsprogramm selbst enthält nur reelle Adressen, läuft demnach im Bereich des ersten ASP-Modul mit den Adressen von 0 bis 16 383. Wie später noch zu zeigen ist, werden die Adressen von Zellen für die Aufnahme von Fehlermeldungen usw. aus externen Geräten durch Schalter fest eingestellt. Das Steuerwerk der ZE 306 transformiert diese Adressen automatisch in den Anfangsteil des Organisationsprogramms, also von 16 256 bis 16 383 in 384 bis 511.

Natürlich sind es Hardware-Einrichtungen, die diese unterschiedliche Behandlung der Adressen hervorrufen. Das Programmzustandsregister wirkt wie ein Vielfachschalter mit den drei Stellungen: PZ-AUS, PZ, USZ. Dabei bedeutet PZ den *privilegierten Zustand* und USZ den Zustand „*Unterbrechung durch schnellen Alarm*“. Die Stellung PZ-AUS gehört zur Abarbeitung eines normalen Anwenderprogramms und veranlaßt eine Modifikation aller gelesenen Adressen. Der Programmierer „denkt“ nur in einem beliebigen 16K-Bereich, den das Organisationsprogramm durch Laden des BAR 9 reell festlegt. Soll während des Programmablaufs auch ein weiterer 16K-Bereich in Anspruch genommen werden, markiert das der Programmierer, wodurch der zugehörige Befehl neben der virtuellen Operandenadresse eine 1 auf Bitstelle 16 des Befehlsregisters erhält. Diese 1 veranlaßt die Addition der Operandenadresse mit dem Inhalt von BAR 15, in das bei der Vorbereitung dieses Programms die zuständige zweite Basisadresse geladen wurde.

Soll sich der Befehl selbst auf den zweiten ASP-Bereich beziehen, zeigt das die 1 auf Bitstelle 15 des Befehlszählers.

Da das Organisationsprogramm nur reelle Adressen enthält, muß vor

seiner Benützung auf den Zustand PZ umgeschaltet werden. Es bleiben dann die Adressen der folgenden Programmteile unverändert.

Meldet sich ein schneller Alarm, so veranlaßt er direkt eine Umschaltung auf den Zustand USZ; die anzusteuernenden Adressen bleiben dann auch hier unverändert.

Die Umschaltung PZ auf PZ-Aus und umgekehrt, sowie von USZ auf den Zustand vor Eintreffen des schnellen Alarms wird durch einen besonderen Befehl (UNT⁹⁾ vorgenommen. Nicht ohne weiteres ist es möglich, die Programmzustände zu ändern und von einem Programmlauf auf einen anderen überzugehen. Den Abspringpunkt in einem Programm muß man so festhalten, daß — nach Erledigung des zweiten Programms — man wieder zum ersten zurückkehren und dieses folgerichtig fortsetzen kann.¹⁾

Es gibt auch Ausnahmen in der Behandlung von Adressen. Z. B. mit der Bitstelle 18 kann die Interpretation einer Adresse dann variiert werden, wenn sie in einem Befehl nicht die Programmunterbrechung bedeuten *kann*.

Um eine größere Rechengenauigkeit für das Rechenwerk der ZE 306 gegenüber der der ZE 305 zu erzielen, besitzt die ZE 306 um zehn Bitstellen verlängerte Register. Die Festkommarechnung und die normale Gleitkommarechnung laufen wie bei der ZE 305 ab. Auf besondere Befehle hin können die Gleitkommaoperationen aber auch mit 33 Betragstellen abgewickelt werden. Man erreicht das dadurch, daß man die höherwertigen Stellen der Mantisse als 24-Bit-Wort in eine ASP-Zelle bringt und die 10 restlichen Stellen (als Bit 15 bis 24) mit dem 11stelligen Exponenten (als Bit 3 bis 14) zu einem zweiten ASP-Wort kombiniert. Durch die Befehle gelangen die Bitstellen in richtiger Reihenfolge in die nun 34stelligen Register und werden dort parallel verarbeitet.

¹⁾ Vielfältige Rücksichten, die zu nehmen sind, gehen über den Rahmen dieser Abhandlung hinaus; sie sollten in den Gerätebeschreibungen nachgelesen werden.

4. Wirkungsweise des Steuerwerks

4.1. Befehle für interne Operationen (Tabelle 2)

Die Erklärungen zum Steuerwerk der ZE 302 bis 305 enthalten den Hinweis, daß alle Operationen in einem Rechner durch Befehle veranlaßt werden. Die Befehlsworte haben eine bestimmte Struktur, die von den Stromkreisen des Steuer- und Rechenwerks vorgeschrieben wird. Die ersten 14 Bitstellen enthalten in den meisten Fällen die Operandenadresse, die letzten 6 Bitstellen die Verschlüsselung des auszuführenden Befehls und die restlichen Bitstellen Hinweise, wie die Adresse oder der Befehl verstanden werden sollen.

Befehle für interne Operationen sind Transferbefehle, arithmetische und logische Verknüpfungsbefehle, Verschiebepbefehle und Sprungbefehle. Zu den arithmetischen Verknüpfungen gehören auch Rechnungen mit Adressen. Im folgenden soll mit wenigen, typischen Befehlen erklärt werden, was für das Verstehen des Ablaufs der Operationen in der ZE wichtig ist. Dabei wird der Befehlsumfang der ZE 305 zugrundegelegt.

● *Die Transferbefehle* betreffen den Transport der Operanden vom ASP zu einem Akkumulator und zurück. Sie bereiten jede Bearbeitung der Daten im Rechenwerk vor. Will man zwei Daten verknüpfen, muß man die erste zunächst in einen Akkumulator bringen, bevor man die zweite nachholt und mit der ersten arithmetisch oder logisch verarbeitet.

Der Befehl „Transferiere Ein Plus“ (TEP) bringt einen Operanden aus der Zelle, deren Adresse im Befehlswort angegeben ist, in den mit Bitstelle 15 bezeichneten Akkumulator. Die gesamte Operation des Rechners besteht aus dem Lesen und dem Ausführen des Befehls. Für jede der beiden Tätigkeiten ist ein ASP-Zyklus notwendig, so daß eine Operationszeit von 3 μ s auftritt. Folgende Teilschritte werden dabei durchlaufen (vgl. Bild 13):

Die Adresse des Befehlswortes steht im Befehlszähler. Auf die Freigabe des ersten Zyklus „Befehl lesen“ hin wandert diese Adresse in das Adreßregister und wird in der Adreßdecodierung zur Zellenan-

Tabelle 2: Maschinenbefehle der Zentraleinheiten des Siemens-Systems 300

58

Befehlsart	Befehl	Bedeutung	Operationszeit in μ s					
			Modell 301	Modell 302	Modell 303	Modell 304	Modell 305	Modell 306*)
Arithmetische Befehle	ADD)x(Festpunktaddition	3,2	3,0	91,6	3,0	3,0	1,2
	ADT)x(Festpunktaddition und Transfer Aus	—	—	—	4,5	4,5	2,4
	SUB)x(Festpunktsubtraktion	3,2	3,0	91,6	3,0	3,0	1,2
	MLT)x(Festpunktmultiplikation	—	—	308—1410	16,5	16,5	8,4—12
	DIV)x(Festpunktdivision	—	—	—	16,5	16,5	9—12,6
	GAN)x(Gleitpunktaddition normalisiert	—	—	—	—	7,5—37,5	3—19,8
	GSN)x(Gleitpunktsubtraktion normalisiert	—	—	—	—	7,5—37,5	3—19,8
	GMN)x(Gleitpunktmultiplikation normalisiert	—	—	—	—	21,0—22,5	9,6—13,8
	GDN)x(Gleitpunktdivision normalisiert	—	—	—	—	19,5	10,2
	KPL	Komplementiere	—	—	91,6	3,0	3,0	1,2
Adreß- arithmetische Befehle	ADA a	Addiere Adresse	—	—	—	1,5	1,5	0,6
	SBA a	Subtrahiere Adresse	—	—	—	1,5	1,5	0,6
	LAP a	Lade Akku mit Adresse Plus	—	—	—	1,5	1,5	0,6
	EHA)a(Erhöhe Adresse in Zelle um Eins	—	—	—	4,5	4,5	1,8
	ENA)a(Erniedrige Adresse in Zelle um Eins	—	—	—	4,5	4,5	1,8
	TEA)a(Erhöhe Adresse um Eins	3,2	3,0	—	3,0	3,0	1,2
Verschiebe- befehle	VAL v	Verschiebe Arithmetisch Links	—	—	41,6—197	3,0—37,5	3,0—37,5	1,2—20,4
	VAR v	Verschiebe Arithmetisch Rechts	—	—	41,6—274	3,0—37,5	3,0—37,5	1,8—21
	VLL v	Verschiebe Logisch Links	1,6 (v=1)	1,5 (v=1)	41,6—197	3,0—37,5	3,0—37,5	1,2—20,4
	VLR v	Verschiebe Logisch Rechts	1,6 (v=1)	1,5 (v=1)	41,6—197	3,0—37,5	3,0—37,5	1,2—20,4
	VDL v	Verschiebe Doppelt Links	—	—	41,6—197	3,0—37,5	3,0—37,5	1,2—20,4
	VDR v	Verschiebe Doppelt Rechts	—	—	41,6—197	3,0—37,5	3,0—37,5	1,2—20,4
	VSE)x(Verschiebe und Suche erste Eins	—	—	80—114	4,5—18,0	4,5—18,0	1,8—9

Logische Befehle	UND)x(Und	3,2	3,0	91,6	3,0	3,0	1,2
	ODR)x(Oder	3,2	3,0	91,6	3,0	3,0	1,2
	UGL)x(Ungleich	3,2	3,0	91,6	3,0	3,0	1,2
	ODL)x(Oder und Lösche	—	—	—	3,0	3,0	1,2
Transferbefehle	TEP)x(Transfer Ein Plus	3,2	3,0	91,6	3,0	3,0	1,2
	TEL)x(Transfer Ein und Lösche	3,2	3,0	91,6	3,0	3,0	1,2
	TEM)x(Transfer Ein Minus	3,2	3,0	91,6	3,0	3,0	1,2
	TAS)x(Transfer Aus	3,2	3,0	116,6	3,0	3,0	1,2
	TEX)x(Transfer Ein Exponent	—	—	—	—	3,0	1,2
	TAX)x(Transfer Aus Exponent	—	—	—	—	3,0	1,2
Sprungbefehle	SPR a	Springe	1,6	1,5	41,6	1,5	1,5	0,6
	SGN a	Springe, falls Akku Gleich Null	1,6	1,5	41,6	1,5	1,5	0,6
	SUN a	Springe, falls Akku Ungleich Null	1,6	1,5	41,6	1,5	1,5	0,6
	SAP a	Springe, falls Akku Plus	1,6	1,5	41,6	1,5	1,5	0,6
	SAM a	Springe, falls Akku Minus	1,6	1,5	41,6	1,5	1,5	0,6
	SUL a	Springe, falls Überlauf	—	—	41,6	1,5	1,5	0,6
	SKU a	Springe, falls kein Überlauf	—	—	41,6	1,5	1,5	0,6
	UNT a	Unterprogrammsprung	3,2	3,0	66,6—108	3,0	3,0	1,2
Organisationsbefehle	STP	Stop	1,6	1,5	41,6	1,5	1,5	0,6
	NOP	Null-Operation	1,6	1,5	41,6	1,5	1,5	0,6
	EPR	Element prüfen	—	1,5	50	3,0	3,0	0,6
Ein-Ausgabebefehle	EAW	Elementauswahl	Operationszeit ist abhängig von der Art des Elementes					
	EVS	Elementversorgung						

)x(Inhalt der Zelle mit der angegebenen Adresse

a Adreßteil des Befehlswortes

)a(Adreßteil des Inhalts der Zelle mit der angegebenen Adresse

v Zahl der Verschiebschritte

*) Das Modell 306 hat die doppelte Anzahl Gleitpunktbefehle und 3 zusätzliche Organisationsbefehle

steuerung entschlüsselt. Der Zelleninhalt gelangt in das Lese-Schreib-Register und in das Befehlsregister. Unterdessen wird auch der Befehlszählerinhalt um eins erhöht, so daß die nächste Befehlsadresse bereitsteht.

Die Befehlsausführung beginnt mit der Decodierung des Operationsteils Bitstelle 19 bis 24 des Befehlswortes. Der beaufschlagte Ausgang des Befehlsdecodierers schaltet den Adreßteil Bitstelle 1 bis 14 zum Adreßregister durch. Die Operandenadresse wird decodiert, der Operand ins Lese-Schreib-Register übernommen und von dort — beeinflußt durch die Befehlsdecodierung und die Bitstelle 15 — in den angegebenen Akkumulator weitergeleitet. Die Bitstelle 0 des Akkumulators wird der Bitstelle 1 angeglichen. Damit steht der Operand zur weiteren Bearbeitung bereit und das Steuerwerk kann den nächsten Befehl lesen.

Bei dem Befehl „Transferiere Ein Minus“ (TEM) durchläuft der Operand den Inverter so, daß aus der positiven eine negative Zahlendarstellung wird. Der Befehl „Transferiere Ein und Lösche“ (TEL) sorgt dafür, daß der Zelleninhalt des Operanden nach dem Lesen nicht mehr wieder eingeschrieben wird. Auf den Befehl „Transferiere Aus“ (TAS) wandert der Operand aus dem Akkumulator über das Lese-Schreib-Register in die angesteuerte Zelle. Die Befehle für den Transfer von Exponenten TEX und TAX gleichen den Befehlen TEP und TAS. Exponenten stehen im ASP in den Bitstellen 3 bis 14 und werden durch den TEX-Befehl in das Exponentenregister des Rechenwerks gebracht.

● *Die arithmetischen Befehle* ähneln den Transferbefehlen. So wird der Befehl „Addiere“ (ADD) wie der TEP-Befehl gelesen. Er enthält ebenfalls eine Operandenadresse und benötigt für die Verarbeitung des Summanden einen zweiten ASP-Zyklus. Damit beträgt seine Operationszeit auch 3 μ s. Der über seine Adresse gefundene Operand wandert über das Lese-Schreib-Register ins Rechenwerk und wird dort nach den arithmetischen Regeln zum Inhalt des bezeichneten Akkumulators addiert. Das Ergebnis erscheint nach der Addition in diesem Akkumulator. Wird der zugelassene Zahlenbereich mit 23 Betragsstellen überschritten, zeigt sich das durch ungleiche Besetzung der Bitstellen 0 und 1 des Akkumulators. Dadurch wird das zugehörige Überlaufregister (ein Flipflop) gesetzt; dieses kann nach seinem Inhalt abgefragt werden.

Die „Subtraktion“ (SUB) entspricht im Ablauf der Addition einer negativen Zahl zum Akkumulatorinhalt, d. h. der Inhalt der angesteuerten Zelle wird vom Inhalt des Akkumulators abgezogen. Damit hat die Subtraktion dieselbe Laufzeit wie die Addition.

Der Befehl „Komplementiere“ (KPL) bezieht sich auf den Inhalt des bezeichneten Akkumulators. Er wird hauptsächlich zur Umwandlung einer positiven in die entsprechende negative Zahl verwendet. Da der Befehl gelesen und im Rechenwerk ausgeführt werden muß, benötigt er auch 3 μ s Gesamtoperationszeit.

Für die Ausführung der Befehle „Multipliziere“ (MLT) und „Dividiere“ (DIV) gelten die Erklärungen über den Aufbau des Rechenwerks. Nach Lesen des Befehls und Transfer des adressierten zweiten Operanden läuft die Verarbeitung der beiden Daten im Rechenwerk ab. Das Addieren, Komplementieren und Verschieben der Operandenteile dauert 15 μ s lang; dann ist das Ergebnis im Akkumulator greifbar. Während dieser Arbeit ist das Steuerwerk nur zum Teil „beschäftigt“. Das Adreßregister und das Lese-Schreib-Register können in dieser Zeit von einem EXE zum Datenverkehr mit dem ASP herangezogen werden.

Wesentliches, was bisher über die Festkommazahlen gesagt wurde, gilt auch für die Behandlung von Gleitkommazahlen. Neu kommt hier dazu, daß nicht nur zwei, sondern vier Operanden zu verknüpfen sind. Vorbereitungsbefehle bringen die zwei Mantissen in die Akkumulatoren und einen Exponenten in das Exponentenregister. Dann erst kommt der Verknüpfungsbefehl, durch den der zweite Exponent ins Rechenwerk wandert. Die Verarbeitung von Mantissen und Exponenten läuft nebeneinander her; beide Teiloperationen beeinflussen sich in rechnerisch richtiger Weise.

In den Beschreibungen der ZE 305 sind die Voraussetzungen für das Rechnen mit Gleitkommazahlen, die Algorithmen für die Operationen und wie die Ergebnisse zu finden sind, enthalten. Um sie zu verstehen, muß man sich vor Augen halten, daß die ZE eigentlich zwei „Rechenwerke“ benötigt, weil sie die Mantissen und die Exponenten gleichzeitig, jedoch von einander getrennt, bearbeiten muß. Beide „Rechenwerke“ tauschen aber Signale aus, weil unter Umständen von Zahlen mit unterschiedlichen Exponenten ausgegangen wurde, die stellenrichtig zu verarbeiten sind. Das einfachste Beispiel dafür ist die Addition. Werden zwei Zahlen mit unterschiedlichen Exponenten

zusammengezählt, so müssen die Mantissen vor der eigentlichen Rechnung so gegeneinander verschoben werden, bis beide Exponenten gleich sind, damit nur Ziffern mit gleichem Stellengewicht addiert werden.

Die Bearbeitungszeiten bei Gleitkommaechnungen sind unterschiedlich lang, weil sie von der Anzahl der Einzelschritte abhängen, die für Exponentenvergleich, Durchführung der Rechnung und Normalisierung der Ergebniszahl notwendig sind.

● *Verschiebefehle* werden nach zwei Kriterien eingeteilt. Ist der Operand eine Zahl, die nach arithmetischen Gesichtspunkten behandelt werden muß, so ist mit einer Verschiebung der „Einsen“ auf andere Bitstellen eine Potenzierung mit positivem oder negativem Exponenten verbunden. Das bedeutet für die Rechner-Hardware, daß sie die Regeln der Mathematik berücksichtigen muß, soweit es überhaupt mit wirtschaftlichen Mitteln zu machen ist. Von außen werden grundsätzlich nur Nullen in den Akkumulator nachgezogen. Aufgaben, die etwas anderes erforderlich machen, sind mit Software zu lösen. Bei der Verschiebung einer negativen Zahl hilft sich die Hardware so, daß sie komplementiert, verschiebt und rückkomplementiert, damit das Ergebnis richtig ist.

Die Befehle „Verschiebe Arithmetisch Rechts“ (VAR) und „Verschiebe Arithmetisch Links“ (VAL) werden wie der TEP-Befehl gelesen, aber nur im Rechenwerk ausgeführt. Im Befehlswort ist die Anzahl der Verschiebeschritte an der Stelle einer Adresse in die Bitstellen 9 bis 14 eingetragen und wird von dort in den Teil des Rechenwerks übernommen, der parallel zu der Bitverschiebung das Abzählen der Verschiebeschritte vornimmt. Theoretisch könnten mit diesen 6 Bitstellen 63 Schritte durchgeführt werden. Mehr Stellen des Befehlswortes übernimmt das Rechenwerk nicht. Es gibt aber auch keine Einrichtung, die verhindert, daß eine unsinnig hohe Zahl von Verschiebeschritten im Bereich unter 63 zurückgelegt wird.

Die Bearbeitungszeit für diese Befehle hängt davon ab, wie oft verschoben werden soll. Das Ergebnis der Verschiebung steht in dem im Befehl gekennzeichneten Akkumulator.

Die zweite Art der Verschiebung eines Akkumulatorinhalts richtet sich nach „formalen“ Gesichtspunkten. Man setzt hier voraus, daß das Wort nur ein „beliebiges“ Bitmuster ist, dessen Stelleninhalt z. B.

in eine für die Identifizierung geeignete Stellung verschoben wird. Auch hier werden natürlich nur Nullen nachgezogen (Bild 18).

Die Befehle lauten: „Verschiebe Logisch Links“ (VLL) und „Verschiebe Logisch Rechts“ (VLR) und enthalten wie die arithmetischen Verschiebefehle die Verschiebezahl und den Hinweis auf den beteiligten Akkumulator.

Die *doppelte Verschiebung* ist eine besondere Variante der logischen Verschiebung und benützt zusammengekoppelte Akkumulatoren. Die dazugehörigen Befehle bewirken, daß die Akkumulatoren in unterschiedlicher Weise verbunden werden, so daß man bei einer Verschiebung einen Teil eines Akkumulatorinhalts oder den ganzen in den anderen Akkumulator bringen kann. Man verwendet diese Befehle bevorzugt zum Teilen eines Maschinenwortes, damit man die beiden Teile getrennt weiterbehandeln kann. Der Unterschied zu den Befehlen VLL und VLR zeigt sich darin, daß beim Zurechtrücken eines Wortes durch die einfache Verschiebung einige Bitstellen über das Registerende hinausgeschoben werden und damit nicht mehr zur Verfügung stehen, während im anderen Fall der hinausgeschobene Rest im anderen Akkumulator aufgefangen wird (Bild 18).

Die Befehle unterscheiden sich wie folgt: Bei „Verschiebe Doppelt Links“ (VDL) und „Verschiebe Doppelt Rechts“ (VDR) ist die Bitstelle 1 des rechten Akkumulators (RA) mit der Bitstelle 24 des linken (LA) verbunden. Bei „Verschiebe Doppelt Links“ (VDL') und „Verschiebe Doppelt Rechts“ (VDR') unter gleichzeitigem Setzen der Bitstelle 15 mit 1 ist die Bitstelle 1 des LA mit der Bitstelle 24 des RA zusammengeschaltet. Hier kann eine sinnvolle Zahl von Verschiebeschritten auftreten, die 32 übersteigt; deshalb wurden für die Darstellung der Verschiebezahl 6 Bitstellen gewählt.

Die ZE 301 und 302 enthalten in ihrem Rechenwerk nur die Ausrüstung für die Verschiebung des Akkumulatorinhalts um eine Stelle. Man kann deshalb nur die Befehle VLL 1 und VLR 1 ausführen, so daß zu größeren Wortverschiebungen mehrere Befehle gegeben werden müssen.

Während die Ausführung der bisher behandelten Verschiebefehle von der Vorgabe einer Anzahl von Verschiebeschritten abhängt, verläuft ein weiterer Befehl in der ZE 304 und 305 gewissermaßen umgekehrt. Mit „Verschiebe und Suche die erste Eins“ (VSE) verschiebt

das Rechenwerk den Inhalt des angegebenen Akkumulators solange nach links, bis in der Stelle 0 die erste 1 erscheint. Der Verschiebezähler zählt hierbei nicht, wie bei den anderen Befehlen, abwärts auf 0, sondern von 0 an aufwärts. Die ermittelte Verschiebezahl wird am Schluß der Operation in die Zelle eingetragen, deren Adresse im VSE-Befehl enthalten war.

● *Logische Befehle* für die Verknüpfung von zwei Operanden unterscheiden sich nur gering vom Additionsbefehl. Jeder Befehl wird gelesen, der Operand dann über seine Adresse angesteuert und ins Rechenwerk gebracht. Dort findet praktisch ein Stellenvergleich zwischen dem Wortinhalt des Akkumulators und dem neu dazugekommenen Operanden statt. Das Ergebnis erscheint in *dem* Akkumulator, der den ersten Operanden enthielt. Für jede dieser logischen Verknüpfungen werden zwei ASP-Zyklen benötigt, so daß sich eine Operationszeit von 3 μ s ergibt.

Das Rechenwerk führt auf logische Befehle hin die Funktionen „UND“ (UND), „ODER“ (ODR), „UNGLEICH“ (UGL) bzw. Exklusivoder und „ODER und Lösche“ (ODL) mit Löschen der angesteuerten ASP-Zelle aus. Bevorzugte Anwendungsfälle sind: für die UND-Verknüpfung das Herausstellen des Inhalts von bestimmten Bitstellen eines Wortes, für die ODR-Funktion das Einfügen einer 1 an einer bestimmten Bitstelle und für die UGL-Funktion das Markieren der Bitstellen, in denen sich zwei Wörter unterscheiden. Mit dem Befehl „ODL“ kann man zwei Wörter, in denen unterschiedliche Bitstellen mit 1 besetzt sind, z. B. Meldungen von EXE, zur einfacheren Auswertung zu einem zusammenfassen.

● *Die Sprungbefehle* unterscheiden sich von allen bisher genannten Befehlen dadurch, daß sie *nicht* auf Operanden einwirken. Schon bei der Erklärung des Steuerwerks wurde erwähnt, daß der Befehlszähler die nächste Befehlsadresse durch Erhöhen der letzten um 1 erzeugt. Das setzt voraus, daß die Befehle eines Programmablaufs immer in aufeinanderfolgenden Zellen stehen, was aber in der Praxis nicht realisierbar ist; außerdem wird in jedem Programm der Punkt erreicht, an dem zwischen zwei Möglichkeiten der Programmfortsetzung zu wählen ist. Der Rechner muß also „Entscheidungen fällen“. Folglich ist es notwendig, an der Stelle, an der die bisher durchlaufene Zellenfolge verlassen wird, den Befehlszähler umzuladen. Dies bewirken die Sprungbefehle.

Man unterscheidet bedingte und unbedingte Sprungbefehle. Die bedingten untersuchen den im Befehlswort bezeichneten Akkumulator nach einem durch die Hardware erfaßbaren Kriterium. Ein solches ist der Inhalt der Bitstelle 1, die — wie bekannt — das Vorzeichen für eine Zahl angibt. „Springe falls Akkumulator Plus“ (SAP) bedeutet ein Verlassen der bisher verfolgten Befehlsreihe, wenn die Bitstelle 1 des Akkumulators mit 0 besetzt ist. Dagegen wird mit „Springe falls Akkumulator Minus“ (SAM) die normale Fortsetzung der Reihe verlassen, wenn die genannte Bitstelle 1 enthält. Es braucht wohl nicht besonders betont zu werden, daß diese Abfrage nicht nur zur Untersuchung von Zahlen nützlich ist.

Die Befehle „Springe falls Akkumulator Gleich Null“ und „Springe falls Akkumulator Ungleich Null“, (SGN) und (SUN), veranlassen eine Überprüfung des Inhalts des bezeichneten Akkumulators und davon abhängig eine Änderung des Befehlszählerinhalts. Weiterhin ist interessant, ob eine Rechenoperation über den Zahlenbereich eines Akkumulators hinausgegangen ist. Da das der Rechner nicht selbstständig anzeigt, muß der Inhalt des zugehörigen Überlaufregisters abgefragt werden. Dies bewirken die Befehle „Springe falls Überlauf“ (SUL) und „Springe falls Kein Überlauf“ (SKU).

Die genannten bedingten Sprungbefehle sind zwar wie für mathematische Operationen formuliert — z. B. „Springe falls Akkumulator Gleich Null“ —, doch ist das nur ein Hilfsmittel. Ausschlaggebend für die Verwendung dieser Befehle ist nur ihr Vollzug in den Stromkreisen von Steuer- und Rechenwerk, also die Abfrage bestimmter Registerinhalte.

Der wichtigste unbedingte Sprungbefehl ist „Springe“ (SPR). Er schreibt auf jeden Fall in den Befehlszähler eine neue Adresse ein, von der ab der Rechner das Programm fortsetzen soll.

Die genannten Sprungbefehle haben gemeinsam, daß sie in den Bitstellen 1 bis 14 die Adresse enthalten, bei der — evtl. unter besonderen Bedingungen — das Programm fortgesetzt werden soll. Sie werden wie der TEP-Befehl gelesen und in das Befehlsregister gebracht. Die Decodierung des Operationsteils hat zur Folge, daß die Adresse direkt zum Befehlszähler durchgeschaltet wird. Da nur ein ASP-Zyklus notwendig ist, dauert die Operation 1,5 μ s. Ist bei den bedingten Sprungbefehlen keine Umladung des Befehlszählers erforder-

lich, geht das Programm selbstverständlich mit dem nächsten Befehl der angefangenen Reihe weiter (Bild 19).

Ein unbedingter Sprungbefehl mit weiterreichenden Konsequenzen ist der „Unterprogrammsprung“ (UNT). Er dient dazu, oft zu wiederholende Befehlsroutinen in beliebige Programme einzuschleusen. Die Umwandlung der Dezimalzahlen (dargestellt als 6-Bit-Zeichen) in Dualzahlen und die Rückverwandlung sind solche typischen „Unterprogramme“, die an vielen Stellen der Prozeßbearbeitung Anwendung finden. Zur Platzersparnis stehen solche Unterprogramme nur einmal im ASP, aber so, daß sie von überall her durch einen Unterprogrammsprung erreicht werden können. Über den UNT-Befehl wird das laufende Programm verlassen, dieses muß allerdings nach Durchlauf des Unterprogramms an der Stelle fortgesetzt werden, wo es unterbrochen wurde. Deshalb wird die Fortsetzadresse des laufenden Programms in der ersten Zelle des Unterprogramms abgespeichert. Auf diese Weise ist das Unterprogramm universell verwendbar. Nach dem Durchgang durch das Unterprogramm steht die erarbeitete Größe an einer zugänglichen Stelle zur Verfügung. Der letzte Befehl ist ein SPR-Befehl zu der Zelle, deren Adresse in dem ersten Wort des Unterprogramms steht, also zu der Fortsetzadresse des Hauptprogramms. So mündet der Programmlauf wieder in das eigentliche Bearbeitungsprogramm.

Der UNT-Befehl hat folglich zwei wichtige Funktionen: Er muß die Sicherstellung der Rücksprungadresse im Hauptprogramm veranlassen und außerdem den Befehlszähler mit der „Anfangsadresse“ des Unterprogramms laden. Dies geschieht so, daß der Befehlszählerinhalt nach dem Befehlslesen über das Lese-Schreib-Register in den ASP geschafft wird. Als Ansteuerungsadresse dienen die Bitstellen 1 bis 14 des Wortes im Befehlsregister; sie gelangen über das Adreßregister zur Adreßdecodierung und sind die erste Wortadresse des Unterprogramms (Bild 20).

Diese Adresse wird auf dem üblichen Weg von dort zum Befehlszähler weitergereicht, durchläuft also den Einsaddierer und steht — um eins erhöht — zum Fortführen des Programms zur Verfügung. So beginnt die Bearbeitung des Unterprogramms praktisch an der zweiten Zelle. In die erste hat der Programmierer ursprünglich nichts (d. h. lauter Nullen) eingetragen, damit die Adresse aus dem Befehls-

zähler immer Platz findet. Zum Lesen und Ausführen des UNT-Befehls werden zwei ASP-Zyklen, also 3 μ s benötigt.

In diesem Zusammenhang wird ein *Organisationsbefehl* wichtig, der eigentlich kein Befehl ist, aber in eine Befehlsreihe eingebaut sein kann. Das ist die „Nulloperation“ (NOP). Das Charakteristische dieses Pseudobefehls ist, daß sein Operationsteil Bitstelle 19 bis 24 nur aus Nullen besteht und trotzdem vom Steuerwerk akzeptiert wird. Der Adreßteil des Befehls ist beliebig besetzbar. So z.B. beginnt man ein Unterprogramm mit NOP 0 — was einer leeren Zelle entspricht — und läßt dann durch das Steuerwerk die Fortsetzadresse des Hauptprogramms eintragen. Im anderen Fall benützt man den NOP-Befehl zur Bereitstellung von bestimmten Größen in einem Hauptprogramm (sogenannte Parameter), die von einem Unterprogramm abgeholt werden sollen. Bei den ZE 301 und 302, die keine „höheren“ Adreßrechnungsbefehle kennen, dient der NOP-Befehl außerdem zur Darstellung von wichtigen Adressen. Die Anfangsadresse einer Liste hat man z.B. gerne in einer bestimmten Zelle greifbar, damit man die Liste jederzeit durchsehen kann; wenn man weiß, in der wievielten Zelle der Liste eine gerade benötigte Größe steht, kann man diese Zelle über die Anfangsadresse der Liste durch Adreßrechnung ermitteln.

Der NOP-Befehl wird wie ein TEP-Befehl gelesen und das Befehlswort in das Befehlsregister eingeschrieben. Eine Auswertung entfällt, auch wenn die Substitutionsstelle Bit 17 besetzt ist. Da aber der Befehlszähler während dieses ASP-Zyklus erhöht wird, kann anschließend der im Programm nachfolgende Befehl bearbeitet werden. Die Operationszeit beträgt 1,5 μ s.

Weitere Organisationsbefehle sind „Stop“ (STP) und „Element prüfen“ (EPR). Beide werden normalerweise vom Programmierer nicht verwendet; sie dienen Prüf- und Wartungsaufgaben! Der STP-Befehl hält den Programmablauf ebenso auf wie ein Drücken der Stop-Taste auf dem Bedienungsfeld. Zur Fortsetzung des Programms muß dann die Start-Taste betätigt werden.

Der EPR-Befehl spricht ein bestimmtes EXE an und ist deshalb mit der entsprechenden Kanalnummer zu versehen. Er gleicht einem Wahlschalter — ähnlich einem bedingten Sprungbefehl —, der den weiteren Programmablauf vom Betriebszustand des angesprochenen EXE

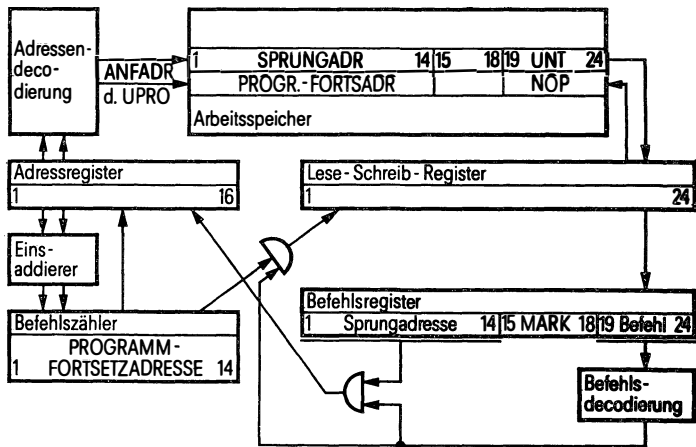


Bild 20: Lesen und Ausführen eines Unterprogrammsprungs

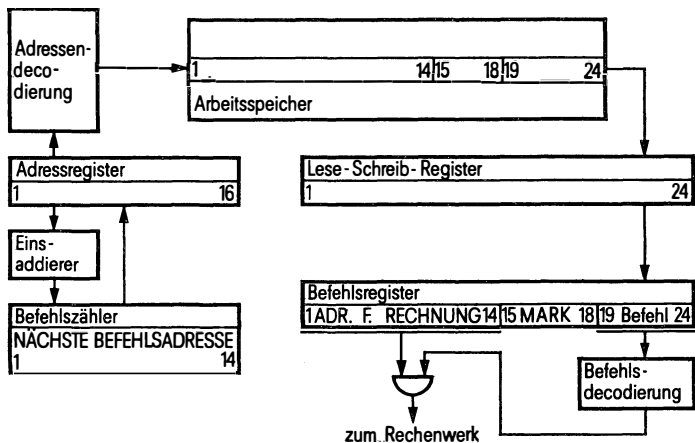


Bild 21: Lesen und Ausführen eines Adreßrechnungsbefehls (LAP, SBA, ADA)

abhängig macht. Ist das EXE bereit, einen Befehl zu übernehmen, was durch ein bestimmtes Quittungssignal angezeigt wird, so erhöht das Steuerwerk der ZE den Befehlszählerstand um zwei (statt um eins). Das bedeutet, daß das Programm mit dem übernächsten Befehl fortgesetzt wird. Kann das EXE keinen Befehl übernehmen, weil es noch „beschäftigt“ ist, so gibt es das Signal „Abweisend“ zurück. Dann wird der Befehlszählerstand um eins vermehrt. Man kann also dem EPR-Befehl einen Sprung zum Beginn der Prüfroutine oder zu einem anderen Programm folgen lassen (SPR-Befehl), das den Zustand des EXE berücksichtigt.

Im Gegensatz zu den bedingten Sprungbefehlen, bei denen in eine beliebige Zelle des ASP gesprungen werden kann, wirkt der EPR-Befehl wie ein Hardware-Schalter, der nur die Wahl zwischen den beiden Folgezellen läßt. An jeder Programmkanal-Nahtstelle gibt es nämlich einen besonderen Eingang für die Weiterstellung des Befehlszählers, der von verschiedenen EXE beaufschlagt werden kann. Gibt ein EXE an dieser Stelle ein Signal in die ZE ein, muß der Eins-addierer des Steuerwerks den Befehlszählerstand ein zweites Mal erhöhen, nachdem er ihn beim Befehlslesen schon einmal auf die normale Folgeadresse gestellt hat.

Der EPR-Befehl wird wie der TEP-Befehl gelesen und in das Befehlsregister gebracht. Die Decodierung bewirkt die Übernahme der Bitstellen 2 bis 5 in das Elementauswahlregister und damit die Ausgabe eines Signals an die Steuerung des angesprochenen EXE. Die Ausführungszeit hängt davon ab, wie lange das EXE für die Rückmeldung braucht. Bei der ZE 304 und 305 wird der Lesezyklus von einem Leerlaufzyklus gefolgt.

● *Die Adreßarithmetischen Befehle* beziehen sich auf die Handhabung oder Änderung des Adreßteils im Befehlswort selbst oder in einem anderen Wort. Da in diesen Bitstellen 1 bis 14 in der Regel eine Adresse enthalten ist, spricht man von Adreßrechnungsbefehlen; doch kann man mit ihnen natürlich auch eine beliebige, verschlüsselte Größe bearbeiten. Die folgenden Befehle sind nur in den ZE 304 und 305 verdrahtet. In den ZE 301 und 302 muß man bei adreßarithmetischen Handlungen die normalen Additions- und Subtraktionsbefehle benutzen; die einzige Ausnahme ist der Befehl zur Erhöhung der Adresse um eins.

Um eine Adresse isoliert in einen Programmlauf zu bringen, gibt der Programmierer den Befehl „Lade mit Adresse Plus“ (LAP). Der Befehl wird wie der TEP-Befehl gelesen und steht im Befehlsregister zur Ausführung bereit. Diese erfolgt im gleichen ASP-Zyklus und besteht aus der Übergabe der Bitstellen 1 bis 14 an den bezeichneten Akkumulator, in dem die Bitstellen 15 bis 24 gelöscht werden. Der Vorgang dauert 1,5 μ s.

Will man dasselbe mit der ZE 301 oder 302 erreichen, so muß die in Frage kommende Adresse mit einem NOP-Befehl in einer Zelle festgehalten und der Zelleninhalt durch einen TEP-Befehl in den Akkumulator geschafft werden.

Die Erhöhung einer Adresse durch den Befehl „Addiere Adresse“ (ADA) und die Verminderung durch „Subtrahiere Adresse“ (SBA) führen das Steuerwerk und das Rechenwerk gemeinsam aus. In beiden Fällen wird das Befehlswort in das Befehlsregister gebracht, dann der Adreßteil Bit 1 bis 14 — wie vorhanden oder invertiert — zum Adreßteil des Wortes addiert, das sich gerade in dem bezeichneten Akkumulator befindet. Das Ergebnis ist in jedem Fall, daß der Inhalt des betreffenden Akkumulators *nur* an den Stellen 1 bis 14 geändert wurde. Auch ein Überlauf wird nicht erfaßt. Beide Befehle benötigen einen ASP-Zyklus und damit 1,5 μ s Bearbeitungszeit (Bild 21).

Der Befehl „Transferiere Ein und Erhöhe Adresse“ (TEA) und die Befehle „Erhöhe Adresse und Transferiere Aus“ (EHA) und „Erniedrige Adresse und Transferiere Aus“ (ENA) dienen zur Vereinfachung der Programmierung beim Auswerten von Datenlisten. Im Unterschied zu den bisher behandelten Adreßrechnungsbefehlen beziehen sich diese Befehle auf ein ASP-Wort, dessen Adresse im Befehl genannt ist.

Der TEA-Befehl wird gelesen und steht nach dem ersten ASP-Zyklus im Befehlsregister. Nach der Befehlsdecodierung wird das Wort angesteuert, dessen Adresse im Befehl steht, und in das Lese-Schreib-Register gebracht. Von dort läuft es durch das Rechenwerk, wird um 1 auf Bitstelle 14 erhöht und gelangt wieder in den Akkumulator. Der Befehl ist somit eine Zusammenfassung von TEP und ADD. Fügt man noch einen TAS-Befehl dazu, der den Akkumulatorinhalt in die Herkunftszelle zurückbringt, so kann man die Einzelanweisungen durch einen EHA-Befehl ersetzen. Der ENA-Befehl läßt sich

durch TEP, SUB und TAS ausdrücken, wobei der SUB-Befehl nur den Adreßteil Bitstelle 1 bis 14 verkleinert. Es liegt auf der Hand, daß der TEA-Befehl zwei ASP-Zyklen (also 3 μ s) und die Befehle EHA und ENA jeweils drei ASP-Zyklen (also 4,5 μ s) benötigen.

Ein weiterer Befehl dieser Gattung ist nur formal ein Adreßrechnungsbefehl. Mit „Addiere und Transferiere Aus“ (ADT) wird ein Wort aus dem ASP gelesen, zu dem Inhalt des im Befehl gekennzeichneten Akkumulators addiert und in die Herkunftszelle rückgespeichert. Die Klassifizierung als Adreßrechnungsbefehl rührt daher, daß das Additionsergebnis nicht auf Überlauf geprüft wird. Wenn aber sicher ist, daß die Addition des ASP-Wortes mit dem Akkumulatorinhalt keine Bereichsüberschreitung zur Folge hat, kann der ADT-Befehl die Befehle ADD und TAS ersetzen.

Der ADT-Befehl wird gelesen und ins Befehlsregister gebracht. Die Bitstellen 1 bis 14 dienen nach der Befehlsdecodierung zur Ansteuerung einer ASP-Zelle und bleiben für den nächsten ASP-Zyklus greifbar. Das angesteuerte Wort gelangt über das Lese-Schreib-Register ins Rechenwerk und wird zum Akkumulatorinhalt addiert. Das Ergebnis der Rechnung wird aus dem Akkumulator zurück zum Lese-Schreib-Register gebracht und in die „noch angesteuerte“ Zelle geschrieben. Die Operationszeit beträgt 4,5 μ s, da 3 ASP-Zyklen notwendig sind.

Die Adreßrechnungsbefehle TEA, EHA, ENA und ADT sind also „Summenbefehle“, die schon vorhandene Stromkreise benutzen. Einige Signale, die über den folgenden ASP-Zyklus oder einen weiteren beibehalten werden, sorgen für die Aneinanderkettung der Befehlschritte und ersparen dem Programmierer die Einzelbefehle. Der Vorteil bei der Benützung dieser Befehle liegt in der Kürzung des Programms und damit des Speicherbedarfs, außerdem fallen die ASP-Zyklen für das Lesen der „angeketteten“ Befehle weg.

Abschließend ist noch darauf hinzuweisen, daß die genannten Operationszeiten auf die ZE 302, 304 und 305 bezogen sind. Die ZE 306 mit ihrer Zykluszeit von 0,6 μ s benötigt für die gleichen Befehle nur jeweils 40 % davon. Bei der ZE 301 vergrößern sich die Zahlen um wenigstens, weil eine Zykluszeit von 1,6 μ s zugrundegelegt werden muß.

In den internen Befehlsverarbeitungen spielt die *Adreßsubstitution* eine wesentliche Rolle. Sie hilft dem Programmierer bei Adreßmani-

pulationen. Wird z. B. zur Auswertung von mehreren Daten (wie Meßgrößen aus dem Prozeß) dieselbe Befehlsfolge benützt, so muß man sie nach jedem Durchgang auf den nächsten Operanden beziehen. Dazu wären in einigen Befehlswörtern die Adressen auszuwechseln. Da dies umständlich und unübersichtlich ist, schreibt man in diese Befehlswörter die Adresse einer Hilfszelle, die die (jeweils) aktuelle Operandenadresse enthält. Man braucht dann nur die Adresse in der Hilfszelle auszutauschen. Die Befehlsfolge bleibt unverändert.

Da die betreffenden Befehlswörter nur die Zelle nennen, in der die Adresse der zu behandelnden Date steht, wird diese Adresse als substituiert markiert. Daraufhin muß zuerst jeder so gekennzeichnete Befehl nach dem Lesen richtig adressiert werden, damit er ausgeführt werden kann.

Bereits bei der Struktur eines Befehlswortes wurde erklärt, daß das Steuerwerk an der Bitstelle 17 erkennt, ob die Bitstellen 1 bis 14 die tatsächliche Operandenadresse enthalten oder nicht. Ist die Substitutionsstelle mit 1 belegt, so bedeutet das, daß der Inhalt der adressierten ASP-Zelle als „Befehl“ zu interpretieren ist. Das angesteuerte Wort wird in das Lese-Schreib-Register gebracht. Von ihm übernimmt das Befehlsregister nur die Bitstellen 1 bis 14 und 17 und „korrigiert“ damit den vorausgegangenen Befehl. Befindet sich nun die richtige Operandenadresse im Befehlsregister, so stellt das Steuerwerk auf Bitstelle 17 eine 0 fest. Dann wird der Befehl in der normalen Weise ausgeführt. Ist die Bitstelle 17 dagegen wieder mit 1 besetzt, so ist ein weiterer Adressenaustausch im Befehlsregister notwendig. Dies geht solange, bis die Bitstelle 17 eine 0 aufweist.

In der ZE 306 ist die Anzahl der programmierbaren, aufeinanderfolgenden Substitutionen auf vier begrenzt.

Substitutionen werden bei allen Befehlen ausgeführt, nur nicht bei NOP; der NOP-Befehl hat ja keinen „Befehlsvollzug“ zur Folge. Im übrigen läßt eine Substitution die Bitstellen 15, 16 und 18 bis 24 des Befehlsregisters unverändert. Die entsprechenden Stellen des abgefragten ASP-Wortes haben also keine Bedeutung.

In der ZE 306 sind noch neun Befehle mehr als in der ZE 305 ausgebaut. In erster Linie gehören dazu sechs Befehle für die Gleitkommarechnung mit einer 34-Bit-Mantisse (einschließlich Vorzeichen-

stelle); sie werden von denen für die Gleitkommarechnung mit 24-Bit-Mantisse durch die Angabe „rechter Akkumulator“ unterschieden, so daß hier die 1 auf Bitstelle 15 die Zuschaltung der Verlängerungsregister im Rechenwerk und die vorgesehene Bearbeitung des Mantissenrestes im Exponentenwort in die Wege leitet. Die restlichen drei Befehle betreffen organisatorische Maßnahmen. Der Befehl „Lade Elementauswahlregister“ (EPR') veranlaßt, daß die Bitstellen 2 bis 5 des Befehlsregisters in das Elementauswahlregister übertragen werden. Folgt diesem Befehl der weitere: „Lade Basisadrefregister“ (SPR'), so wird mit Bitstelle 1 bis 14 des Befehlswortes eine Zelle angesteuert, in der in Bitstelle 1 bis 18 eine Basisadresse mit einer Angabe für den Speicherschutz steht. Diese 18 Bitstellen werden aus dem ASP über das Lese-Schreib-Register in das Basisadrefregister gebracht, dessen „Adresse“ bzw. Nummer im Elementauswahlregister enthalten ist.

Über den Befehl „Organisationsunterbrechung PZ / PZ-AUS / USZ-AUS“ (UNT') wurde bereits berichtet. Mit ihm ändert das Organisationsprogramm den Inhalt des Programmzustandsregisters.

Die Kurzbezeichnungen dieser drei Befehle sind nicht mehr vom Text her verständlich; sie wurden vom Bitmuster der Befehlsverschlüsselung abgeleitet. Dazu wurden die Stellen 15 mit 1 besetzt.

Das Elementauswahlregister wird innerhalb eines ASP-Zyklus geladen. Ein Basisadrefregister erhält die Adresse erst mit dem zweiten Zyklus. Ebenso hat die Abwicklung des UNT'-Befehls die Dauer von zwei Zyklen. Bei jeder Umschaltung auf „Zustand Aus“ wird nämlich die Zelle angesteuert — ihre Adresse steht im Befehlswort — mit der das fortsetzende Programm beginnt. Mit der Zuschaltung des Zustandes PZ werden die Informationen, die zur Fortsetzung des unterbrochenen Programms nach der Rückschaltung notwendig sind, im ASP abgespeichert. Dazu gehört vor allem die Fortsetzadresse. (Die Auswertung dieser Bitstellen nach Umschaltung auf „Zustand Aus“ geschieht mit Software).

4.2. Befehle für externe Operationen

Die Befehle, welche interne Vorgänge in der ZE einleiten, richten sich an Bauelemente, deren Abstände voneinander so kurz sind, daß

die Signallaufzeiten genau bestimmt werden können. Befehle für externe Operationen sprechen dagegen auch EXE-Steuerungen an, die mehrere Meter von der ZE abgesetzt an mechanischen Geräten angeordnet sind. Bei ihnen sind die Zeitbedingungen nicht so klar definierbar. Außerdem stellt ein EXE eine selbständige Einheit dar, die nach eigenen Gesetzen arbeitet und somit nicht zu jedem beliebigen Zeitpunkt mit der ZE korrespondieren kann.

Folglich ergibt sich nur dann eine gute Zusammenarbeit, wenn sich die ZE und das EXE bei der Befehlsgabe verständigen und beim Transport der Daten möglichst wenig behindern. Diese Bedingungen zielen darauf, daß das Steuerwerk die Befehle an ein EXE erst dann als gegeben betrachten kann, wenn ihr Empfang quittiert worden ist — auch wenn das mehr Zeit in Anspruch nimmt. Ferner muß es für die langsamen EXE immer möglich sein, in den Datenverkehr mit dem ASP zu treten, wenn sie dazu in der Lage sind; nur dann können sie überhaupt sinnvoll eingesetzt werden.

Die Regeln für den externen Verkehr lassen sich demnach so zusammenfassen:

- Jede externe Operation kann nur auf einen Befehl der ZE hin ausgeführt werden, auch in dem besonderen Fall, daß das EXE selbst den Anstoß zur Kontaktaufnahme mit der ZE gibt, weil z. B. dort ein für die Prozeßabwicklung wichtiges Ereignis im Prozeßablauf festgestellt wurde.
- Die externe Stelle, die etwas ausführen muß — also das EXE als solches bzw. ein bestimmtes Bauteil in seinem Verband — muß von der ZE her gezielt angesprochen werden. Die Adressierung eines Befehlsempfängers in der Peripherie geht also über die Angabe des Anschlußkanals oft hinaus.
- Die Befehle für die Ein- und Ausgabe von Informationen müssen sicherstellen, daß der Datenverkehr in gewünschter Weise abläuft. Deshalb enthalten sie genaue Angaben, von welcher Stelle etwas zu welcher anderen Stelle transferiert werden soll. Bei der Eingabe eines Meßwertes sind z. B. die Meßstelle und eine bestimmte ASP-Zelle „Datenweg-Endstationen“. Die EXE-Steuerung speichert diese Angaben (auch Parameter genannt), weil das externe Gerät unter Umständen geraume Zeit braucht den Befehl auszuführen. Für die ZE wäre es bei einer Vielzahl von simultan arbeitenden EXE sehr

schwierig, den Zusammenhang zwischen angebotener Information und zugehörigem Platz im ASP zu wahren. Das selbständig arbeitende EXE „verwaltet“ folglich den — ihm von der ZE zur Verfügung gestellten — Speicherplatz selbst.

● Die Bits eines Befehlswortes, das an ein EXE gerichtet wird, sind zum Teil „Miniaturschalter“, deren Stellung 0 oder 1 in der externen Steuerung echte Schaltfunktionen übernehmen. Die in Frage kommenden Bitstellen sind also, abhängig von dem Aufbau der externen Steuerung, richtig zu besetzen.

Es gibt zwei Arten von Befehlen für den externen Verkehr, die von den ZE 302 bis 306 im gleichen Sinn interpretiert werden. Befehle, die die ZE 301 ausgibt, werden am Schluß dieses Abschnitts beschrieben. Wie die externen Steuerungen auf diese Befehle reagieren, beschreibt Teil 2 des Buches.

Der Befehl „*Element Auswählen*“ (EAW) stellt die Verbindung zum EXE her; seine Kanalnummer ist in den Bitstellen 2 bis 5 des Befehlswortes enthalten. Der Befehl wird durch die duale 5 auf den Bitstellen 19 bis 21 gekennzeichnet.

Das Befehlswort wird wie der TEP-Befehl gelesen und gelangt dann ins Befehlsregister. Nach der Decodierung werden dort die Bitstellen 2 bis 5 entnommen und als Kanalnummer in das Elementauswahlregister eingeschrieben. Dieses Register bleibt solange gesetzt, bis es durch einen weiteren EAW-Befehl (oder einen EPR-Befehl) umgeladen wird. Ist das EXE fähig, den Befehl entgegenzunehmen, so werden ihm die Bitstellen 1 bis 14 und 22 bis 24 aus dem Befehlsregister zugeführt. Häufig stehen in den letzten drei Stellen des EAW-Befehls Kennungen oder Steuerbits, die die externe Steuerung beeinflussen sollen. Deshalb werden diese Befehle als EA 0-... EA 7-Befehle bezeichnet, wobei die Zahl auf den Inhalt der letzten drei Stellen hinweist.

Erst dann ist die Operationszeit für den EA-Befehl beendet, wenn das EXE ein Quittungssignal zur ZE zurückgibt, das im Steuerwerk das Lesen des folgenden Befehls erlaubt; die Operationszeit variiert relativ stark und läßt sich deshalb nicht genau festlegen.

Die Angaben aus einem EA-Befehl genügen in den meisten Fällen nicht, um ein EXE in Aktion zu bringen; es müssen weitere Anweisungen folgen.

Der Befehl „*Element Versorgen*“ (EVS) ist ein Wort mit Parametern, das wie ein Befehl gedeutet werden muß und deshalb in der Befehlsfolge steht; es ist durch die duale 6 auf den Bitstellen 19 bis 21 markiert und kann — analog zum EA-Befehl — durch einen Ausdruck EV 0 ... EV 7 erklärt werden. Der EV-Befehl wird wie der TEP-Befehl gelesen und steht anschließend im Befehlsregister bereit. Nach der Decodierung übernimmt die Steuerung des EXE die Bitstellen 1 bis 14 und 22 bis 24, deren Inhalt in unterschiedlichem Sinn ausgelegt wird; die Bitstellen 1 bis 14 können die Adresse der ASP-Zelle enthalten, mit der verkehrt werden soll; sie können aber auch die Adresse des Bauelements im EXE bedeuten, das den Inhalt der ASP-Zelle übernehmen soll bzw. von dem eine Information erwartet wird. Ferner können sie auch den Umfang des Datenaustauschs durch Angabe der Zeichenanzahl oder des Adressenbereichs im ASP festlegen — wie später noch erläutert werden wird. Bei bestimmten EXE stellen die Bitstellen 22 bis 24 der EV-Befehle zusätzliche Einzelsignale dar, welche die übrigen Angaben erläutern. Für die Operationsdauer gilt dasselbe wie bei den EA-Befehlen.

EA- und EV-Befehle haben gemeinsam, daß das Steuerwerk immer erst die Quittierung durch das EXE abwartet, auch wenn zunächst das Signal „Abweisend“ von dort kommt. Die Programmsteuerung läßt Leerzyklen ablaufen, die nur durch einen Datenverkehr mit einem anderen EXE unterbrochen werden können. Ein neuer Befehl wird dann erst gelesen, wenn der letzte tatsächlich erledigt worden ist. So benötigt man zur Einleitung einer externen Operation einen EA-Befehl und — abhängig vom Aufbau der EXE-Steuerung — zusätzlich einen oder mehrere EV-Befehle. Letztere können entfallen, wenn der externe Informationsgeber oder -abnehmer durch die Kanalnummer genau bezeichnet ist, und die ASP-Adresse durch Hardware-Einrichtungen, z. B. durch Schalter, fest eingestellt angesteuert werden kann. Man adressiert immer dann durch Hardware, wenn der Zielpunkt (die ASP-Adresse o. a.) im Betrieb nicht geändert zu werden braucht.

Der vereinfachte Aufbau der Nahtstellen einer ZE 301 führt unter anderem auch zu einer besonderen Struktur der Befehle für Externoperationen. Der Datenverkehr über die Akkunahtstelle benötigt keine Vorschriften für die Abspeicherung von Informationen, weil der Datenweg im Akkumulator endet. Der Transfer zum oder

vom ASP wird mit den üblichen Befehlen TEP bzw. TAS angeordnet. Der Datentransport über die ASP-Nahtstelle läuft zwar ähnlich dem an der Schnellkanalnahtstelle einer ZE 305 ab, doch wird die Befehlsgabe dafür zu einer Aufforderung an die EXE-Steuerung, sich die erforderlichen Angaben selbst aus dem ASP zu holen. Hierzu ist jedem EXE eine Zelle im ASP (die Elementversorgungszelle) fest zugeteilt, in die das Programm nacheinander die erforderlichen „Parameter“ bringt. Ist die Zelle geladen, und der Befehl zur Abholung erteilt, steuert das EXE die durch Hardware adressierte Zelle an und nimmt deren Inhalt auf. Dieser Vorgang wiederholt sich so oft, bis die EXE-Steuerung alle Anweisungen erhalten hat und mit der Arbeit beginnen kann.

Der Befehl für eine Ein- oder Ausgabe an der ASP-Nahtstelle heißt „Element Auswählen“ (EVk) und wird wie der TEP-Befehl gelesen und in das Befehlsregister gebracht. Von diesem Befehlswort sind nur die Stellen 19 bis 24 wichtig. Die duale 6 in den Bitstellen 19 bis 21 kennzeichnet den Befehl für die ASP-Nahtstelle. Bitstelle 23 und 24 enthalten die Nummer des EXE; sie werden dem ASP-Multipllexer zugeführt und wählen über diesen „Vielfachschalter“ den Datenweg zu der EXE-Steuerung an. Hierdurch kommt Signalspannung an die Adreßdurchschaltung im EXE, so daß die Elementversorgungszelle im ASP der ZE angesteuert wird. Ihr Inhalt gelangt über das Lese-Schreib-Register und den Datenkanal in die externe Steuerung. Dieses Wort enthält in den letzten Bitstellen einen Hinweis, in welches Register der Steuerung die Information eingetragen werden soll. Das Quittungssignal des EXE nach der Übernahme des Wortes meldet der ZE, daß das Steuerwerk den nächsten Befehl lesen kann. Auch hier ist die Programmsteuerung bis zum Empfang des Quittungssignals gesperrt. Die Operationsdauer umfaßt zwei ASP-Zyklen für Befehl- und Operandenlesen, zuzüglich der Laufzeit der Steuer- und Meldungssignale.

Für eine Datenausgabe über die Akkunahtstelle wird zunächst der Akkumulator mit dem auszugebenden Wort geladen. Dann folgt der Befehl an den Akkunahtstellenzusatz (und damit an ein bestimmtes EXE) „Element Auswählen“ (EAK), der die Weitergabe des Akkumulatorinhalts auslöst. Dieser Befehl enthält auf den Bitstellen 1 bis 3 die Kanalnummer des EXE am AKZ, auf den Bitstellen 5 bis 14 nähere Bezeichnungen wie die Adresse des ausgewählten Bauteils

eines EXE und von Bitstelle 19 bis 24 weitere Kennungen. Die Bitstellen 19 bis 21 zeigen die duale 5 als Angabe, daß der Befehl an den AKZ gerichtet ist. Bitstelle 22 markiert die Verkehrsrichtung: Aus- oder Eingabe. Bitstelle 23 ist für den Anstoß des Alarmgruppenregisters oder einer Analogeingabe bestimmt, etwas zu tun (nähere Angaben siehe Teil 2), und die Bitstelle 24 trifft die Auswahl zwischen Digital- und Analoggröße.

Die Dateneingabe ähnelt der -ausgabe; wenn der Befehl zur Eingabe gegeben wird, übernimmt die externe Steuerung das Wort und schaltet es zum Akkumulator durch. Von dort wird es durch einen TAS-Befehl in den ASP geholt.

Der EAK-Befehl wird wie ein TEP-Befehl gelesen und im Befehlsregister identifiziert. Die Bitstellen 1 bis 14 und 22 bis 24 laufen weiter an den AKZ. Dieser stellt fest, welche Gerätesteuerung angesprochen ist, und schaltet die Bitstellen 4 bis 14 und 22 bis 24 an das entsprechende EXE durch; dessen Steuerung übernimmt das im Akkumulator vorbereitete Ausgabewort oder sie fordert das Eingabewort vom Gerät oder vom Signalformer des Prozeßelements an. Bei der Eingabe ist der Akkumulator mit dem Wortregister der EXE-Steuerung durchverbunden und erhält gleichzeitig mit diesem die Wortinformation. Wenn das Wort in der EXE-Steuerung steht, gibt diese das Zeichen für das Operationsende ans Steuerwerk der ZE, so daß dort der nächste Befehl gelesen werden kann.

Die Operationsdauer beinhaltet demnach einen ASP-Zyklus und die Signallaufzeiten für Wortinformation und Meldesignale.

4.3. Prioritätensteuerung

Die Tätigkeiten des Steuerwerks einer ZE haben eine bestimmte Rangfolge, damit ein Programmablauf in einem festen Rahmen bleibt. Wenn ein Befehl, der eine weitere Aktion auslöst, gelesen wird, so muß diese beendet sein, bevor der nächste Befehl gelesen werden darf. Enthält das Befehlsword eine substituierte Adresse, so muß erst die „echte“ Operandenadresse gefunden sein, bevor ein Operandentransport in Frage kommt. Hat der gelesene Befehl eine Befehlsverschlüsselung, die der Rechner nicht identifizieren kann, muß das Steuerwerk stehenbleiben, da offenbar ein Fehler vorliegt.

Diese Beispiele zeigen logische Abhängigkeiten auf, die für sich sprechen. Sie müssen auch „dem Rechner klar sein“.

Also benötigt das Steuerwerk einer ZE eine Einrichtung, die auf solche Abhängigkeiten achtet, das *Prioritätennetzwerk*. Sein Aufbau wurde mit der Hardware des Steuerwerks erklärt. Da es bestimmt, welche Tätigkeit den nächsten ASP-Zyklus beanspruchen darf, spricht man auch von einem *Zuteilungsnetzwerk für ASP-Zyklen*.

Die Rangordnung für die normalen Steuerwerksabläufe ist:

- Priorität Ø Datenverkehr mit externem Element,
 1 Substituieren einer Adresse,
 2 Operandentransfer und -verarbeitung,
 3 Leerlaufzustand,
 4 Stopzustand,
 5 Ausführen einer Programmunterbrechung (z. B. einer BAP),
 6 Lesen eines Befehls.

Wie erwähnt können mehrere Anforderungen gleichzeitig gestellt werden; die Flipflops an den Eingängen der Prioritätensteuerung halten diese fest, bis sie entsprechend ihrer Einstufung erledigt worden sind. Die Priorität Ø hat die höchste Dringlichkeit. Folglich bekommt ein EXE, das sich hier meldet, den nächsten Zyklus für einen Datenaustausch mit dem ASP, auch wenn gerade ein Befehl gelesen, aber noch nicht ausgeführt ist. Der Befehl steht während der Zykluszeit im Befehlsregister bereit und behindert den Datenverkehr nicht. Dies gilt auch für längere Operationen im Rechenwerk und bei Externverkehr, wenn der ASP nicht (mehr) beansprucht wird und für das Steuerwerk Leerlaufzustand herrscht.

Man kann also sagen, wenn Daten und Operanden aneinander vorbeikommen, ohne sich zu treffen, ist unter bestimmten Voraussetzungen sogar ein Gleichzeitig im Rechner möglich. Die Verkehrsregelung geschieht durch Einzelsignale (auf die in dieser Beschreibung nicht eingegangen werden kann), die unzulässige Aktionen verhindern.

Der Stop-Zustand wird durch den Stop-Befehl, durch Drücken der Taste Stop auf dem Bedienungsfeld oder durch einen Fehler hervorgerufen. (Die Abwicklung einer Programmunterbrechung wird in Teil 2 behandelt.)

Die oben genannten Prioritäten stellen das Gerüst jeder Prioritätssteuerung dar. Einzelne Stufen können aber auch mehrmals unterteilt sein. Man unterscheidet dann mehrere Anlässe, die zur gleichen Reaktion im Rechner führen, um Fehler im automatischen Betriebsablauf vor Handeingriffen und diese vor bedingten Programmänderungen zu berücksichtigen. Dies gilt vor allem für eine Programmunterbrechung.

In der ZE 306 sind die Programmebenen — wie schon erwähnt — durch Hardware-Einrichtungen unterteilt. Man sieht das sehr deutlich an der Reihenfolge der Prioritäten:

- Priorität 10 Unbedingte Unterbrechung durch schnellen Alarm,
- 11 Unbedingte Unterbrechung durch Befehl UNT, um den Zustand PZ einzustellen,
- 12 Bedingte Unterbrechung durch BAP,
- 13 Befehl lesen.

Diese Prioritätsreihe gibt zwingend vor, daß im Zustand PZ, in dem das Organisationsprogramm tätig ist, keine bedingte Programmunterbrechung (BAP) zum Zuge kommt und daß der schnelle Alarm auf jeden Fall vor den Zuständen PZ und PZ-AUS rangiert. Andererseits liegt auch auf der Hand, daß die Entgegennahme eines schnellen Alarms sinnlos ist, wenn sich der Rechner gerade im Fehlerzustand befindet und gestoppt wurde.

5. Grundsätzliches über die Bedienung der Zentraleinheit

5.1. Zusammenhang zwischen Hardware und Programmierung

Die konventionelle Steuerungseinrichtung für eine umfangreiche Prozeßmaschinerie ist meistens in einem Schrank eingebaut, an dessen Frontseite für die Einflußnahme auf den Prozeßablauf einige Taster und Schalter montiert sind. Das Bedienungspersonal erhält — meistens schriftlich — Anweisungen, wie es die Schaltgeräte für jede Aufgabe einstellen und wie es auf Anzeigen, die auf den Meldegeräten erscheinen können, reagieren soll. Eine übergeordnete Dienststelle oder der Lieferant der Anlage stellt die Vorschriften zusammen, nach denen die Maschinenanlage und die Steuerung den Zweck erfüllen, den man der Projektierung zugrundegelegt hat.

Der Prozeßrechner wird — in Sonderfällen — als Automatik eingesetzt, die vollkommen selbständig — ohne unmittelbaren Einfluß des Menschen — ihre Aufgaben erfüllt. Sie ersetzt dann zum großen Teil die oben genannte Steuerungseinrichtung, das Bedienungspersonal und die schriftlich gegebenen Anweisungen in der Hand des Steuermanns. Die übergeordnete Dienststelle ist der Programmierer, der die Instruktionen für den Einsatz des Rechners verfaßt und als Programm in den ASP einschreibt. Der Rechner liest diese Anordnungen selbst und „schaltet“ seine Stromkreise entsprechend. Im Unterschied zum Betriebspersonal kann der Rechner dabei keinen Fehler machen, wenn die ihm übergebenen Arbeitsvorschriften stimmen. Nur die Zuverlässigkeit seiner Bauteile beeinflusst seine Betriebssicherheit.

Man spricht davon, daß der Rechner die Befehle des Programms „liest“ und „ausführt“. Seine Tätigkeiten bewegen sich somit in einem Rahmen, den der Sprachgebrauch dem Menschen zuordnet. Wenn ein Rechner aber einem Menschen entsprechen soll, so kann er Befehle nur „lesen“, wenn sie in der ihm eigenen Sprache abgefaßt sind; dies ist die Maschinensprache; sie besteht aus Bitkombinationen mit 24 Stellen, jede besetzt mit 0 oder 1, und nimmt so auf die internen Stromkreise des Rechners Einfluß. Wenn die einzelnen Wörter (bzw. Bit-

kombinationen) sinnvoll zusammengestellt wurden, erfüllen sie ebenso einen Zweck wie die zu einem bedeutungsvollen Satz zusammengestellten Wörter der üblichen menschlichen Sprache.

Zu Beginn der Anwendung von Datenverarbeitungsanlagen wurde das Programm in der Maschinensprache geschrieben. Bald stellte sich heraus, daß dies kein bequemer Weg war, eine so komplizierte Maschine zu bedienen; deshalb suchte man geeignetere Mittel für eine Programmierung, die dem menschlichen Verständnis mehr entgegenkommt.

Es wurden höhere Programmiersprachen entwickelt; sie sind flüssiger zu schreiben und verwenden eine sinnfälligere Symbolik. Damit ergab sich gleichzeitig die Notwendigkeit, die Symbole dieser höheren Sprachen — die ja wiederum eine Art Verschlüsselung darstellen — in die Maschinensprache zurück zu übertragen, weil die DVA nur damit arbeiten kann.

Den Zusammenhang zwischen dem Arbeiten der Maschine und den in den ASP einzuschreibenden Arbeitsanweisungen sieht man am besten daran, wie ein Rechner „angefahren“ wird. Geht man vom Zustand *Null* aus, also wenn die Versorgungsspannung abgeschaltet und der Magnetisierungszustand der Ringkerne [2] [3] [6] im ASP unbekannt ist, so ist der Rechner funktionsuntüchtig. Das gilt aber auch noch für den Folgezustand, wenn der Operator (der für den Rechnerbetrieb verantwortlich ist) die Versorgungsspannung zuschaltet. Würde man versuchen, in diesem Augenblick Informationen mit einem Lochkarten- oder Lochstreifengerät in den ASP einzugeben, so könnte sie der Rechner nicht aufnehmen. Ihm fehlen die Arbeitsvorschriften, die er aus dem ASP lesen müßte. Außerdem muß er selbst — wie schon erwähnt, — die Veranlassung zu einer Eingabe durch ein externes Gerät geben.

Im Zustand *Null* und *eingeschaltet* muß also „mit Gewalt“ ein „Mini-programm“ in den ASP geladen werden, das den Rechner in den Stand versetzt, selbst weitere Programme nachzuholen. Man nennt diesen Vorgang *Ureingabe*.

Programme werden normalerweise von Datenträgern — wie Lochkarten oder Lochstreifen — in eine DVA eingelesen. Wenn externe Speicher vorhanden sind, benützt man wegen ihrer hohen Eingabegeschwindigkeit auch diese.

Damit der Rechner selbst die Programme übernehmen kann, werden ihm die dafür notwendigen Befehle im Maschinencode als elektrische Signale übergeben; sie gelangen in die Anfangszellen des ASP ab Zelle 0. Unter diesen Maschinenwörtern sind in erster Linie EA- und EV-Befehle, die die Kanalnummer des vorgesehenen Eingabeelements, die Anfangsadresse im ASP, die Länge des Eingabeblocks und evtl. Schaltsignale für die externe Steuerung enthalten. Mit diesen ersten Befehlen fordert der Rechner ein Hilfsprogramm an, das stufenweise das Organisationsprogramm einliest.

Es gibt auch Ureingabeeinrichtungen, die den ASP durch Hardware-Schaltungen — also ohne Datenträger — mit den ersten Befehlen laden. Diese können in der ZE oder in einem EXE eingebaut sein. Die ZE 301 läßt sich nur anfahren, wenn das Ureingabeprogramm vom Bedienungsfeld aus von Hand eingetastet wird. So sind bei Datenverarbeitungsanlagen verschiedenen Fabrikats unterschiedliche Möglichkeiten in Gebrauch. Abhängig von der Art des Eingabemediums umfaßt das Ureingabeprogramm bis zu 20 Maschinenwörter.

5.2. Erstellung eines Programms

Es ist verständlich, daß man Programme für eine DVA schreiben kann, ohne viel über die Hardware der Maschine zu wissen. Wenn man die Symbolik der verwendeten Programmiersprache exakt berücksichtigt, läßt sich das geschriebene Programm einwandfrei und ohne Fehler in die Maschinsprache übertragen. Die Handhabung der Sprachelemente schreibt eine „Grammatik“ vor, wie es sie für jede Sprache gibt.

Es gibt Programmiersprachen, die den Maschinencode wortgetreu darstellen. Sie nehmen auf die Hardware der DVA Rücksicht und heißen deshalb *maschinenorientierte* oder *Assembler-Sprachen*. Für das Prozeßrechnersystem 300 heißt sie PROSA 300 (Programmiersprache mit symbolischen Adressen). Ihr Name kommt davon, daß sie anstelle von Zahlen einprägsame Abkürzungen für die Adreßangaben benützt, z. B. ANFADR für Anfangsadresse statt der Zellennummer. Außerdem enthält sie sinnfällige Buchstabengruppen für die Verschlüsselung der Befehle (ADD für Addiere, SPR für Springe usw.), so daß ein Programm bei einiger Übung flüssig gelesen werden kann.

Man wird unabhängiger von den speziellen Eigenschaften einer DVA, wenn man eine *problemorientierte oder Compiler-Sprache* verwendet. Diese ist auf ein bestimmtes Fachgebiet zugeschnitten und deshalb international verständlich. Bekannt sind die Sprachen *ALGOL* (*algorithmic language*) für technisch-wissenschaftliche Aufgaben, *FORTRAN* (*formula translation*) bevorzugt für mathematische Aufgaben und *COBOL* (*common business oriented language*) für kommerzielle Angelegenheiten. Sie bestehen aus englischen Sprachelementen in leicht erfaßbarer Form.

Es ist einleuchtend, daß ein Programm, das in einer höheren Sprache abgefaßt ist, am schnellsten durch eine DVA in den Maschinencode übertragen wird. Das notwendige *Übersetzerprogramm* (der Assembler oder Compiler) muß so aufgebaut sein, daß es die mit der Programmierung einer Aufgabe verfolgten Absichten verwirklichen läßt. Das gilt besonders für den Compiler. Dieser muß aus einem codierten Ausdruck eine Befehlsfolge ableiten, die nicht nur zu dem angestrebten Ziel führt, sondern nach Möglichkeit auch alle Randbedingungen berücksichtigt und das Auftreten von Fehlern erfaßt. Das führt dazu, daß das Maschinenprogramm und damit das Arbeiten des Rechners unmittelbar von der Qualität des Übersetzers abhängt.

Im Normalfall werden folgende Programme für den Betrieb eines Rechners eingesetzt:

Das schon öfterst er wählte *Organisationsprogramm* (ORG) regelt den Ablauf aller Externoperationen, koordiniert die Zusammenarbeit zwischen unterschiedlichen Programmen, speichert wichtige Daten beim Programmwechsel ab, führt die anlageninterne Fehleruntersuchung durch und ist „Mädchen für alles“ beim Aufrechterhalten eines geordneten Datenverkehrs. Es ist bausteinartig zusammenfügbar; sein Umfang richtet sich vor allem nach der Geräteausrüstung der jeweiligen Prozeßrechneranlage.

Ein *Überwacherprogramm* hilft dem Programmierer, Unstimmigkeiten in seiner Befehlsfolge zu finden. Es gibt zwar keinen Überwacher, der die Logik im Programm konsequent verfolgen könnte; doch erleichtert jedes Überwacherprogramm die Überprüfung eines Programmablaufs unter gewissen Gesichtspunkten.

Ein *Wartungsprogramm* ermöglicht dem Personal für die Fehlersuche, den Rechner unter außergewöhnlichen Bedingungen zu testen. Die

Reaktionen der ZE auf bestimmte Prüfungen geben Hinweise für das Auffinden von defekten Bauteilen.

Ein *Anwenderprogramm* sind Betriebsanweisungen, nach denen ein Rechner die Aufgaben am Einsatzort erfüllt, z. B. die Führung eines Prozesses der Industrie, die Erarbeitung von Forschungsergebnissen in einem wissenschaftlichen Institut u. a. Die Anwenderprogramme sind im Regelfall auf spezielle Betriebsbedingungen zugeschnitten, während die vorher genannten Systemprogramme von jedem Rechner des gleichen Typs benutzbar sind.

Zur Erleichterung der Programmierung von Anwenderaufgaben, die in den Bereichen der Prozeßautomatisierung immer wieder vorkommen, werden in wachsendem Umfang *Programmsysteme* entwickelt. Diese behandeln die häufig anstehenden Probleme in möglichst umfassender Weise, z. B. die Übernahme von analogen Meßwerten und von digitalen Anzeigen aus dem Prozeß, die Überwachung von Prozeßgrößen auf Über- und Unterschreiten von Grenz- und Sollwerten, die normierte Zusammenstellung von Drucktexten und anderes. Das Programm besteht aus einer größeren Anzahl von Teilprogrammen, die nach Bedarf durch gezielte Auswahl zu einem geschlossenen Komplex zusammengefügt werden können.

Die Programmierung eines Prozeßrechners ist nicht Gegenstand dieser Erörterungen. Hier soll nur der enge Zusammenhang zwischen Hardware und Software herausgestellt werden, der bei der Projektierung einer Prozeßrechneranlage im Auge behalten werden muß.

So muß die Hardwareseite in der Lage sein, alle Möglichkeiten, die in der Gerätetechnik der Anlage enthalten sind, nutzbringend einzusetzen. Dabei kommt es besonders auf folgende Punkte an:

- bestmögliche Ausnützung des Befehlsvorrates einer ZE mit Rücksicht auf die Programmlaufzeit.
- kürzestmögliche Signallaufzeit zu den EXE durch zweckmäßige Geräteaufstellung,
- Kenntnis aller Fehleranzeigen in jedem EXE zur Abschätzung der notwendigen Rechnerreaktionen und
- Beurteilung, ob bestimmte Signale, die eine EXE-Steuerung normalerweise abgeben kann, zu außergewöhnlichen Anzeigen verwendbar sind, die im speziellen Anwendungsfall benötigt werden.

Weiterhin ist es vorteilhaft, die Grenzen der Fähigkeiten der Anlagen-Hardware zu kennen:

Die Einschwingzeiten beim Zuschalten von Meldesignalen aus dem Prozeß,

die Folgen eines Stromausfalls in der ZE und bei jedem EXE, die durch den Stromverbrauch bedingten Ausbaugrenzen bei der Bestückung externer Elemente,

die Möglichkeit einer Simultanarbeit zwischen gleichartigen externen Geräten und anderes.

Der Programmierer muß darüber informiert sein, welche seiner Ansprüche an dem Nichtvermögen der Hardware scheitern und wo ein geschicktes Programmieren über diese Schwierigkeiten weghilft. So ist von besonderer Wichtigkeit:

Die möglichst frühe und ausreichend genaue Abschätzung des Bedarfs an Speicherplatz,

die richtige Einschätzung der Geschwindigkeit jedes EXE und die davon abhängige Einordnung aller externen Operationen in ein passendes Zeitschema für den Programmablauf.

6. Schaltkreistechnik

Die Zentraleinheiten arbeiten zum Teil mit unterschiedlicher Geschwindigkeit. Dies hängt von der Art der verwendeten Schaltkreistechnik ab. Tabelle 3 zeigt die Unterschiede in der Ausstattung.

Tabelle 3: Schaltkreistechniken

DTL-Technik Dioden-Transistor-Logic	ZE 303	EXE-Steuerungen, P1K, (in SIMATIC H)
TTL-Technik Transistor-Transistor-Logic	ZE 301	Prozeßelemente P2K, P3K, P4K
ECL-Technik Emitter-Coupled-Logic	ZE 302, 304, 305, 306	

Die Physik des Transistors und seine Anwendung in der Technik wird in der Spezialliteratur ausführlich behandelt. Sie ist die Grundlage für das Verständnis eines Rechneraufbaus. Darüber hinaus gibt es aber einige Zusammenhänge, die bei der Projektierung einer Rechneranlage beachtet werden müssen; deshalb wird ein kurzer Streifzug durch das Gebiet der Anwendung integrierter Halbleiterschaltungen nützlich sein [2], [4].

An der Grundschialtung eines NPN-Transistors kann man den Zusammenhang zwischen Kollektorstrom I_c und Basisstrom I_b aufzeigen. Legt man z. B. +12V als 1-Signal und 0V als 0-Signal fest, so erhält man die Abhängigkeit des Ausgangssignals (gleichbedeutend mit der Kollektor-Emitter-Spannung U_{CE}) vom Eingangssignal (dargestellt durch die Basis-Emitter-Spannung U_{BE}).

Das Kennlinienfeld des Transistors zeigt den Kollektorstrom I_c abhängig von der Kollektor-Emitter-Spannung U_{CE} bei unterschiedlicher Basis-Emitter-Spannung U_{BE} . Wird der Transistor als Schalter benützt — wie im dargestellten Beispiel — so interessieren nur die Punkte P_1 und P_2 , die die stationären Zustände darstellen; diese Punkte müssen nicht immer auf die extrem möglichen Spannungen bezogen werden

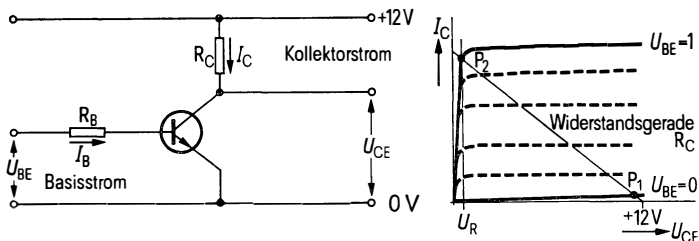


Bild 22: Beschaltung und Kennlinienfeld eines NPN-Transistors

(z. B. 12V oder 0V bei Vernachlässigung der Widerstände), sondern können bei der gleichen Versorgungsspannung auch näher zusammen-geschoben sein.

Damit nehmen die Signale 1 und 0 in unterschiedlichen Anwendungs-fällen beliebige Werte an. Der Signalhub (die Spannungsdifferenz) kann auch in den negativen Bereich gehen.

Trägt man den Kollektorstrom und die Kollektor-Emitter-Spannung abhängig vom Basisstrom auf, so zeigt sich, daß sie sich gegenläufig verändern. Der Kollektorstrom geht von einem bestimmten Basis-strom an auf einen Sättigungswert. Wenn man den Arbeitspunkt P_2 so weit nach rechts rückt, daß diese Sättigung erreicht ist, spricht man von *gesättigter* Technik. Bei kleinerem Signalhub handelt es sich um *ungesättigte* Technik.

Das Produkt I_C mal U_{CE} ist die Verlustleistung P_v des Transistors (Bild 19). Von ihr hängt die Erwärmung des Bauelements ab. Liegen die Arbeitspunkte des Transistors in den Gebieten niedriger Verlust-leistung, so ist die Erwärmung gering, da der Bereich hoher Leistung beim Umschalten schnell durchlaufen wird. Schiebt man die Arbeits-punkte aber näher zusammen, so nimmt der Transistor seine stati-schen Zustände im Bereich höherer Verlustleistung ein. Dann ist er einer weit größeren Erwärmung ausgesetzt und wird störanfälliger.

Diesen Gegebenheiten ist das Schaltverhalten gegenüberzustellen. Die Umschaltung dauert bei maximalem Signalhub wesentlich länger als bei einem reduzierten. Also kann man in der Schaltkreistechnik die Schnelligkeit nur vergrößern, wenn man gleichzeitig eine höhere Er-

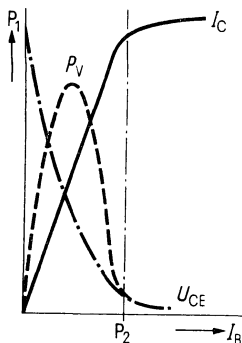


Bild 23: Verlustleistung eines Transistors abhängig von dem Basisstrom

wärmung in Kauf nimmt. Man wird also überall da, wo nicht unbedingt größte Geschwindigkeiten verlangt werden, „langsame“ Techniken einsetzen, um den Aufwand zur Wärmeableitung in Grenzen zu halten [7]. Die Umschaltzeiten liegen bei „schnellen“ Schaltkreisen um eine, bei „langsamen“ zwischen drei und zwanzig Nanosekunden. Nach der oben erwähnten Definition gehören die DTL- und TTL-Schaltkreise zur gesättigten und die ECL-Technik zur ungesättigten Technik.

Aus den Folgerungen, die im einzelnen aus den genannten Eigenschaften integrierter Halbleiterschaltungen zu ziehen sind, kann für diesen Überblick der Hinweis auf den Unterschied des ASP-Zyklus in der ZE 301 (TTL) und der ZE 306 (ECL) entnommen werden.

Es ist besonders hervorzuheben, daß für die SIMATIC-H-, die TTL- und die ECL-Technik unterschiedliche Betriebsspannungen gewählt wurden. So können die Signale aus einem Schaltkreis des einen Typs nicht ohne Hilfsmittel im Schaltkreis eines anderen Typs ausgewertet werden. Die Folge ist die Notwendigkeit von Anpassungsschaltungen an den „Nahtstellen“, z. B. im E/A-Kanal der ZE 302, 304, 305 und 306, wo ECL- und SIMATIC-H-Technik zusammentreffen. Ferner wird nun auch verständlich, warum an die ZE 301 (TTL) keine EXE der übrigen Zentraleinheiten angeschlossen werden können; sie besitzen in den meisten Fällen Steuerungen in SIMATIC-H.

Der Übergang von diskreten Bauelementen — wie sie die Steuerungen in SIMATIC-H zeigen — auf integrierte Halbleiterschaltungen beim Aufbau von Zentraleinheiten brachte eine große Raum- und Materialersparnis. Integrierte Halbleiterschaltungen sind aus vielen Bauelementen (z. B. Transistoren, Widerständen, Kondensatoren usw.) zusammengesetzt, die geschlossene Funktionsblöcke bilden und keine sichtbaren Verbindungen aufweisen. Damit entfällt viel Feinstverdrahtung.

Während einzeln zusammengeschaltete Bauteile vor der Montage nach passenden Toleranzbereichen ausgewählt werden, damit sie eine zuverlässig arbeitende Einheit bilden, gelten für eine Fertigungsserie von Funktionseinheiten sofort einheitliche Toleranzen. Auch das Temperaturverhalten dieser Serie ist gleich. So wird die Erzeugung elektronischer Schaltungen durch die Verwendung integrierter Halbleiterschaltungen oft billiger und einfacher, sofern man die benötigten Elemente überhaupt produzieren kann. Die kurzen Verbindungswege in jedem Funktionsblock tragen außerdem noch zu einer niedrigen Signallaufzeit bei.

Abschließend soll noch eine NAND-Verknüpfung gezeigt werden, die einmal in der TTL-Technik und weiterhin in DTL-Technik aufgebaut ist. Es ist zu erkennen, daß die NAND-Stufe in TTL-Technik wesentlich schneller ist, als die entsprechende Stufe in DTL-Technik (Bild 20).

Die beiden NAND-Stufen verknüpfen zwei Eingangssignale nach der UND-Bedingung und kehren es um. Sie unterscheiden sich dadurch, daß die Eingangsdiode der DTL-Schaltung in der TTL-Technik durch einen Transistor mit „Multiemitter“ ersetzt werden.

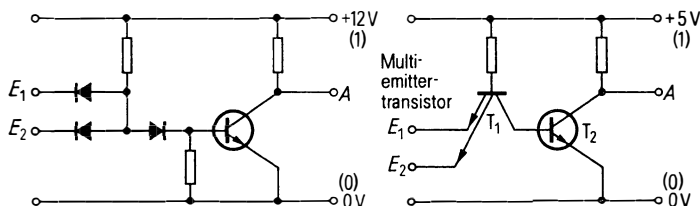


Bild 24: NAND-Stufen in DTL- und TTL-Technik

Bei Anliegen von 1-Signalen an den beiden Eingängen E_1 und E_2 führen beide Schaltungen am Ausgang A 0-Signal. In der TTL-Einheit wird der Transistor T_2 leitend. Nach dem Umschalten eines Eingangs auf 0 geht die Basis-Emitter-Spannung gegen Null. Dieser Übergang erfolgt umso schneller, je eher das Basispotential des Transistors T_2 abgeleitet ist. In der DTL-Schaltung bremst die Diode vor der Basis den Potentialabbau so stark, daß die TTL-Einheit um ein Vielfaches der Zeit schneller reagiert.

Die NAND-Stufe zeigt auch sehr instruktiv, welche Bedeutung die Vereinbarung hat, nach der einer bestimmten Spannung ein bestimmter Signalzustand zugeordnet wird. Erhält die positive Spannung die Bezeichnung 1, so ergeben sich die Beziehungen zwischen den Eingangs- und Ausgangssignalen entsprechend einer NAND-Stufe:

E_1	E_2	A
0	0	1
0	1	1
1	0	1
1	1	0

Legt man aber für die positive Spannung den Signalzustand 0 fest, so „entsteht“ eine NOR-Stufe:

E_1	E_2	A
0	0	1
0	1	0
1	0	0
1	1	0

Diese Beispiele zeigen, daß der Konstrukteur für größere Schaltungen viel Freiheit hat, die Signaldefinitionen festzulegen. Da er aber aus fertigungstechnischen Gründen versuchen wird, möglichst viele gleichartige Bausteine zu verwenden, muß er bei Reihenschaltungen u. U. Wechsel in den Definitionen hinnehmen, um am Ausgang der Schaltungsanordnung wieder die ursprüngliche Form der Information zu erhalten; dies ist einer der Gründe dafür, warum — wie eingangs bemerkt — die Information beim Durchlauf durch den Rechner nicht überall die gleiche Form beibehält. Nur mit einer guten Übersicht über die Stromkreise der Maschine kann man also den Informationsfluß in der Hardware verfolgen.

Tabelle 4: Potenzen von 2

2^n	n	2^{-n}
1	0	1,0
2	1	0,5
4	2	0,25
8	3	0,125
16	4	0,062 5
32	5	0,031 25
64	6	0,015 625
128	7	0,007 812 5
256	8	0,003 906 25
512	9	0,001 953 125
1 024	10	0,000 976 562 5
2 048	11	0,000 488 281 25
4 096	12	0,000 244 140 625
8 192	13	0,000 122 070 312 5
16 384	14	0,000 061 035 156 25
32 768	15	0,000 030 517 578 125
65 536	16	0,000 015 258 789 062 5
131 072	17	0,000 007 629 394 531 25
262 144	18	0,000 003 814 697 265 625
524 288	19	0,000 001 907 348 632 812 5
1 048 576	20	0,000 000 953 674 316 406 25
2 097 152	21	0,000 000 476 837 158 203 125
4 194 304	22	0,000 000 238 418 579 101 562 5
8 388 608	23	0,000 000 119 209 289 550 781 25
16 777 216	24	0,000 000 059 604 644 775 390 625

Literaturverzeichnis

- [1] Langer, E.; Schmitt, R.:
Rechteckferritkerne
Berlin · München: Siemens Aktiengesellschaft 1967
- [2] Höppl, H.: Prosa 300 für Prozeßautomatisierung
Bd. 1: Einführung
Berlin · München: Siemens Aktiengesellschaft 1970
- [3] Nordmeyer, B. D.:
Schaltungsaufgaben in der Fernschreib- und Signaltechnik
Berlin · München: Siemens Aktiengesellschaft 1969
- [4] Reiß, K.; Liedl, H.; Spichall, W.:
Integrierte Digitalbausteine
Kleines Praktikum
Berlin · München: Siemens Aktiengesellschaft 1970
- [5] Heim, K.:
Schaltungsalgebra
Berlin · München: Siemens Aktiengesellschaft 1969
- [6] Keller, G.; Pumpe, G.; Krämer, R.:
Elektronische Schaltungen in der Fernschreib- und Signaltechnik
Berlin · München: Siemens Aktiengesellschaft 1970
- [7] van Leyen, D.:
Wärmeübertragung
Grundlagen und Berechnungsbeispiele aus der Nachrichtentechnik
Berlin · München: Siemens Aktiengesellschaft 1971

Stichwortverzeichnis

- Addition 35
- adreßarithmetische Befehle 68
- Adreßmodifikation 54
- Adreßsubstitution 70
- Adreßtransformation 54
- Akkumulatoren 34
- Akkunahstellen-Zusatz 49
- alphanumerische Zeichen 23
- Alarmbearbeitung 14
- Anwenderprogramm 84
- Arbeitsspeicher (ASP) 16
- Arbeitsspeicher-Multiplexer (ASM) 49
- arithmetische Befehle 60
- Assembler-Sprache 82

- Basisadreßregister 54
- Bedienungselement 15
- Befehlswort 23, 26
- Bit (bit) 21
- Bitmuster 23
- Block 26
- Byte 26

- Compiler-Sprache 83

- Date 23
- Decodierung 32
- Division 37

- Einadreßmaschinen 34
- Ein-Ausgabekanal 16
- Externes Element (EXE)** 15
- Externe Speicher 15

- Familien 13
- fest verdrahteter Rechner 13
- Festkommarechnung 39
- frei programmierbarer Rechner 13

- Gleitkommarechnung 39

- Kanal 46
- Kompatibilität 48

- Lese-Schreib-Register 31
- logische Befehle 64

- Multiplikation 35

- Nahtstellen 46

- Organisationsbefehle 67
- Organisationsprogramm 83

- Prioritäten 14
- privilegierter Zustand (PZ) 55
- Programm 17
- Programmsysteme 84
- Prozeßelemente 15
- Prozeßrechner 14

- Real-time-Betrieb 14
- Rechenwerk 16

- Simultananarbeit 14
- Schnellkanalnahtstellen 46
- Sprungbefehle 64

Standardnahtstellen 46
Standardperipherie 15
Steuerwerk 16
Subtraktion 36

Takt 20
Teilwort 23
tetradisch verschlüsselte
 Dezimalzahlen 22
Transferbefehle 57

Übersetzerprogramm 83
Überwacherprogramm 83

Unterbrechung durch schnellen
 Alarm (USZ) 55
Ureingabe 81

Verschiebebefehle 62

Wartungsprogramm 83
Wort 23

ZE 301 48, 75, 82
ZE 306 51, 71, 79
Zelle 32
Zentraleinheit (ZE) 15

Dieses Taschenbuch stellt einen Leitfaden zum Verständnis der Prozeßrechner des SIEMENS-SYSTEM 300 dar. Mit leicht verständlichen Beispielen wird versucht, die komplizierte Arbeitsweise der DVA auch dem Interessenten näherzubringen, der nur einige Grundbegriffe der logischen Verknüpfung von Informationen und der Halbleitertechnik beherrscht. Es wird der Aufbau der Zentraleinheit in engem Zusammenhang mit der Vorbereitung und der Abwicklung der Datenbearbeitung dargestellt und dadurch die untrennbare Verknüpfung von Hardware und Software unterstrichen.

Ein zweiter Band „Peripheriegeräte“ informiert weiterführend über Aufbau und Zusammenarbeit von Zentraleinheit und Externen Elementen innerhalb der gesamten Prozeßrechneranlage.