Zeitschrift: Technische Mitteilungen / Schweizerische Post-, Telefon- und Telegrafenbetriebe = Bulletin technique / Entreprise des postes, téléphones et télégraphes suisses = Bollettino tecnico / Azienda delle poste, dei telefoni e dei telegrafi svizzeri

Herausgeber: Schweizerische Post-, Telefon- und Telegrafenbetriebe

Band: 51 (1973)

Heft: 3

Artikel: Der Kleinrechner als Hilfsmittel des Ingenieurs

Autor: Loretan, René Peter

DOI: https://doi.org/10.5169/seals-875282

Nutzungsbedingungen

Die ETH-Bibliothek ist die Anbieterin der digitalisierten Zeitschriften auf E-Periodica. Sie besitzt keine Urheberrechte an den Zeitschriften und ist nicht verantwortlich für deren Inhalte. Die Rechte liegen in der Regel bei den Herausgebern beziehungsweise den externen Rechteinhabern. Das Veröffentlichen von Bildern in Print- und Online-Publikationen sowie auf Social Media-Kanälen oder Webseiten ist nur mit vorheriger Genehmigung der Rechteinhaber erlaubt. Mehr erfahren

Conditions d'utilisation

L'ETH Library est le fournisseur des revues numérisées. Elle ne détient aucun droit d'auteur sur les revues et n'est pas responsable de leur contenu. En règle générale, les droits sont détenus par les éditeurs ou les détenteurs de droits externes. La reproduction d'images dans des publications imprimées ou en ligne ainsi que sur des canaux de médias sociaux ou des sites web n'est autorisée qu'avec l'accord préalable des détenteurs des droits. En savoir plus

Terms of use

The ETH Library is the provider of the digitised journals. It does not own any copyrights to the journals and is not responsible for their content. The rights usually lie with the publishers or the external rights holders. Publishing images in print and online publications, as well as on social media channels or websites, is only permitted with the prior consent of the rights holders. Find out more

Download PDF: 15.07.2025

ETH-Bibliothek Zürich, E-Periodica, https://www.e-periodica.ch

Der Kleinrechner als Hilfsmittel des Ingenieurs

René Peter LORETAN, Colchester (Grossbritannien)

681.325:621.39

Zusammenfassung. *Zwei Einsatzfälle eines Kleinrechners werden beschrieben, die Verwendung als Steuerelement in einem Messsystem für PCM und die Benutzung für numerische Berechnungen. Anhand dieser Beispiele werden die Vor- und Nachteile gegenüber verdrahteten Steuerungen auf der einen und Grossrechenanlagen auf der andern Seite dargelegt.*

Le calculateur miniature, auxiliaire de l'ingénieur

Résumé. *L'auteur décrit deux cas d'application d'un calculateur miniature, l'emploi comme élément de commande dans un système de mesure pour MIC et l'utilisation pour des calculs numériques. Se fondant sur ces exemples, il explique les avantages et inconvénients par rapport aux commandes câblées, d'une part, et aux grandes installations de calculateurs, d'autre part.*

Il piccolo calcolatore quale mezzo ausiliario dell'ingegnere

Riassunto. *Si descrivono due casi d'impiego di un piccolo calcolatore, cioè l'utilizzazione quale elemento di comando in un sistema di misurazione per PCM e per calcoli numerici. Sulla scorta di questi esempi si spiegano, da un lato, i vantaggi e gli svantaggi nei confronti di comandi con cablaggio e dall'altro, di impianti calcolatori grandi.*

1. Einleitung

Kleinrechner oder «Minicomputer» haben in der kurzen Zeit ihres Bestehens schon in zahlreichen Gebieten Anwendung gefunden [1], [2], [3]. Bei allen Unterschieden in der individuellen Ausführung weisen sie sämtliche Merkmale auf, die auch Grosscomputer charakterisieren. Am wichtigsten ist dabei die Tatsache, dass ein elektrisch veränderbarer Speicher vorhanden ist, in dem sowohl Programme als auch Daten abgelegt werden. Darum herum sind auch hier Steuerwerk, Rechenwerk und Ein/Ausgabeeinheit angeordnet, obwohl gewisse Teile oft sehr stark vereinfacht sind. Beispielsweise sind häufig nur wenige arithmetische Register vorhanden, die Wortlänge ist kürzer als bei Grossmaschinen, der Speicher in kleineren Blöcken organisiert. Die Ein/Ausgabeeinheit dagegen ist entsprechend der Tatsache, dass meist eine grosse Zahl von Aussengeräten betrieben wird, grosszügiger ausgestattet und oft in einer Weise ausgeführt, die den «Interface»-Aufwand in den Peripheriegeräten möglichst herabsetzt.

Der Fortschritt in der Halbleitertechnologie brachte es mit sich, dass die Preise der Kleinrechner in den letzten Jahren so stark gesunken sind, dass es möglich wurde, sie in Bereichen einzusetzen, die bisher der konventionellen Schaltungstechnik vorbehalten waren. Die Bezeichnung «Rechner» ist in solchen Anwendungen irreführend, da vor allem die Fähigkeit zur Steuerung komplizierter Abläufe mit zahlreichen Entscheidungsstellen massgebend ist und nicht die Durchführung arithmetischer Operationen. Beispielsweise können die einzelnen Bits einer Speicherzelle Stellungen von Relais in einem durch den Prozessor gesteuerten Gerät darstellen, das heisst, das durch sie gebildete Wort hat keineswegs die Bedeutung einer im Binärsystem dargestellten Zahl. Das Rechenwerk vieler Kleinrechner ist daher bis zu einem gewissen Grad auf diese Art von Verarbeitung spezialisiert. Meist ist in der Standardausführung nur die Möglichkeit zur Addition und Subtraktion vorhanden, die Zahlendarstellung mit gleitendem Komma sowie die entsprechenden Operationen sind nicht vorgesehen. Wenn sie gebraucht werden, müssen sie im einzelnen programmiert werden, indem zum Beispiel für die Darstellung einer Zahl drei aufeinanderfolgende Speicherzellen reserviert werden, wobei eine als Exponent und zwei als Mantisse interpretiert werden. Die Programme werden dadurch länger und laufen auch langsamer ab als in einer Maschine, in der diese Operationen im Rechenwerk enthalten sind.

Trotz dieser Einschränkungen lässt sich die Anschaffung eines Kleinrechnersystems für Rechenanwendungen in gewissen Fällen rechtfertigen, wobei sogar ein time-sharing-Betrieb mit mehreren Fernschreiberterminalen möglich ist [4], [5]. Häufiger ist der Fall, dass ein Prozessor, der für eine besondere Steueraufgabe beschafft wurde, in einem Teil der Zeit für Rechenzwecke verfügbar ist. Dieser Dienst bedingt keine zusätzlichen Investitionen und ist daher ausserordentlich vorteilhaft. Gerade für den Ingenieur kann oft eine grosse Lücke geschlossen werden, da häufig Berechnungen nötig sind, die auch mit programmierbaren Tischrechenmaschinen nicht mehr bewältigt werden können, für die aber anderseits die Erstellung eines Programmes in Zusammenarbeit mit einem im Stapelbetrieb arbeitenden Rechenzentrum, wegen der langen Umlaufzeiten, nicht in Frage kommt. Die Entwicklung eines Programmes im direkten Dialog mit dem Rechner ist die einzig vernünftige Methode zum Lösen derartiger Probleme und wird neuerdings auch bei Grosscomputern in Form des time-sharing verwirklicht. Der Kleinrechner bietet hier eine Alternative für Berechnungen vernünftigen Ausmasses, worauf später noch näher eingetreten wird.

In der Abteilung Forschung und Entwicklung PTT ergab sich diese Situation mit dem Kleinrechner PDP-8/I, der für die Steuerung des Modells einer PCM-Telephonzentrale beschafft wurde [6]. Diese Anlage, die nicht für einen Dauerbetrieb vorgesehen ist, ist zwar ständig an den Rechner angeschlossen, wird aber nur für Messungen, Demon-

strationen und zum Austesten von Änderungen an den Steuerprogrammen in Betrieb genommen. In der übrigen Zeit wird der Rechner für die Entwicklung des im folgenden beschriebenen Messsystems, zur raschen Auswertung von Telephonverkehrsmessungen und für numerische Berechnungen verwendet. Die universelle Verwendbarkeit eines Kleinrechners könnte wohl kaum besser gezeigt werden.

2. Einsatz eines Kleinrechners für Steuerungszwecke

Bevor auf die Beschreibung eines speziellen Anwendungsfalles eingegangen wird, sollen zuerst einige grundsätzliche Überlegungen angestellt werden, wie es zur Einführung der elektronischen Rechner in der Steuerungstechnik kam.

2.1 Fest verdrahtete und speicherprogrammierbare Steuerungen

Die beiden Ausführungen können am besten anhand zentralgesteuerter Telephonautomaten erläutert werden. Die zentrale Steuerung, die im allgemeinen als Markierer bezeichnet wird, wurde dabei ursprünglich in Relaistechnik ausgeführt, zum Beispiel in [7]. Ein solcher Relais-Markierer kann als klassisches Beispiel für eine Steuerung angesehen werden, deren Funktion durch die Verdrahtung zwischen den Bauelementen festgelegt ist. Neuere Systeme verwenden elektronische Elemente [8]. Dies brachte eine Erhöhung der Arbeitsgeschwindigkeit und führte gleichzeitig zu einer weitgehenden Normung der Grundschaltungen und Funktionsprinzipien, die heute in den TTL- und MOS-Schaltkreisfamilien einen hohen Stand erreicht hat. Die rein binäre Arbeitsweise ermöglicht die mathematische Behandlung beim Entwurf, zum Beispiel eine Verminderung der Anzahl Gatter zur Erfüllung einer bestimmten Funktion. Diese Normalisierung ging allerdings auf Kosten einer gewissen Raffinesse. In der Relaisschaltungstechnik hatte der Entwerfer die Möglichkeit, eine Schaltung durch Ausnutzung von Fehlstrombedingungen und durch Variation der Windungszahl und der Kontaktart weitgehend an einen bestimmten Fall anzupassen, wobei oft verblüffend einfache «Kunstschaltungen» zur Anwendung gelangen. Bei elektronischen Digitalsystemen verlagert sich das Problem vom eigentlichen Schaltkreisentwurf in die Verwirklichung einer geschickten Zusammenschaltung bestehender Grundelemente.

Elektronische Steuerungen werden oft «programmierbar» genannt. Dies bedeutet in den meisten Fällen, dass der Funktionsablauf durch Umstecken von Dioden in einem Umwerterfeld oder ähnliche Massnahmen geändert werden kann. Aus praktischen Gründen ist dabei die Zahl möglicher Kombinationen beschränkt. Von einer speicherprogrammierbaren Steuerung kann man erst dann sprechen, wenn das Gerät die erwähnten Eigenschaften einer elektronischen Rechenmaschine besitzt. Die Schematisierung wird damit noch eine Stufe weitergetrieben. Statt einer Reihe logischer Schaltkreistypen liegt nun ein vollständig homogenes Grundmaterial in Form eines in Worten konstanter Länge organisierten Speichers vor, das frei gestaltet werden kann, wobei die ganze Arbeit «auf dem Papier» gemacht wird, da das Endprodukt ein geschriebenes Programm ist.

Auf diese Weise erzielt man die in vielen Fällen gewünschte Anpassungsfähigkeit der Steuerung. Dies hat aber nicht nur Vorteile, wie die in allen grösseren Systemen dieser Art durch sporadisch auftretende Programmfehler verursachten Störungen beweisen. Der Grund ist darin zu sehen, dass, ähnlich wie beim Übergang zur Elektronik, raffinierte Speziallösungen zugunsten einer flexibleren Konzeption geopfert wurden. Beispielsweise wird kein vernünftiger Entwerfer einer verdrahteten Steuerung einen bestimmten Zähler, der maximal 8 verschiedene Stellungen einnehmen kann, mit 12 flip-flops aufbauen. Wird derselbe Zähler indessen programmässig verwirklicht, so ist dies durchaus üblich, ausser wenn eine grosse Zahl solcher Zähler gebraucht wird. Diese offensichtliche Verschwendung von Speicherkapazität lässt sich bei vielen Rechnern nicht beheben, da Teile von Worten programmässig angesteuert werden müssen und demzufolge für die Anwendung in einem Einzelfall mehr Speicher benötigt werden, als sich solche einsparen lassen. Der wirtschaftliche Nachteil wiegt nicht allzu schwer, da sich bei einem Grossspeicher in den Schaltungen für die Ansteuerung Einsparungen ergeben, so dass eine Speicherzelle hier verhältnismässig nicht so teuer ist. Die unausgenützte Speicherkapazität bildet aber ein Problem für die Sicherheit der Steuerung. Nimmt nämlich der als Beispiel gewählte Zähler einmal Werte an, die 8 übersteigen, so kann dies zur Auslösung völlig unüberblickbarer Vorgänge führen. Ein solch illegaler Wert kann durch eine elektrische Störung entstehen, doch ist es weitaus wahrscheinlicher, dass ein Programmfehler die Ursache ist. Besonders schwerwiegend ist der Fall, wenn auf diesem Weg eine falsche Adresse berechnet wird und zum Beispiel Teile des Programmes wie Daten angesteuert werden. Hier wirkt sich eine andere Art von Verschwendung negativ aus, nämlich die Tatsache, dass ein Medium, das zum Beispiel in jeder Mikrosekunde verändert werden könnte, zur Speicherung von Information verwendet wird, die unter allen Umständen konstant bleiben muss. Die vielen potentiell möglichen Stellungen eines Speicherwertes, zusammen mit der leichten Veränderbarkeit, stellen eine Gefahr dar. Das Problem ist so gross, dass häufig aus Sicherheitsgründen ein Festwertspeicher für gewisse Programme verwendet oder ein Speicherschutz eingebaut wird, der nur beim Laden vorübergehend wirkungslos ist.

Reine Rechnersteuerungen werden praktisch selten angewendet, und zwar aus Gründen der Wirtschaftlichkeit und

Geschwindigkeit. Meistens erfolgt eine Kombination mit relativ intelligenten Peripheriegeräten, die als unselbständige verdrahtete Steuerungen betrachtet werden können. Die Aufgabenteilung muss von Fall zu Fall gut abgewogen werden, was anhand des folgenden Beispiels illustriert wird.

2.2 Ein Mess- und Simulationssystem für PCM-Anwendungen

2.2.1 Problemstellung

Solange Pulscodemodulation (PCM) nur für Übertragungszwecke eingesetzt wird, können zur Prüfung der Ausrüstungen im wesentlichen die bei Trägersystemen üblichen Methoden eingesetzt werden, indem auf der Niederfrequenzseite des einen Terminals ein Signal eingespeist und mit dem resultierenden Signal am Ausgang des andern Terminals verglichen wird. Auch die ihrer Natur nach digitale Signalisierung, die beispielsweise beim Einsatz auf Bezirks- und Fernstrecken je einen Kanal in jeder Richtung umfasst, kann auf diese Weise mit Signalgeneratoren und Impulsschreibern geprüft werden, da die im Verhältnis zu den zu übermittelnden Signalen rasche Abtastung mit 500 oder 1000 Hz und die integrierende Wirkung der Relais auf der Empfangsseite das System unempfindlich gegen vereinzelte Fehler in der Übertragung macht. Messungen auf der Digitalseite umfassen im allgemeinen nur eine Kontrolle des Augendiagramms in kritischen Einsatzfällen und die globale Bestimmung der Fehlerrate.

Die Verhältnisse ändern sich aber sehr stark, wenn die Vermittlung in das System integriert wird. In diesem Fall werden einzelne Kanäle für Steuerungszwecke benützt. Da keine langsamen Aussengeräte direkt angeschlossen sind, könnte die volle Kanalkapazität zur Übertragung verwendet werden. Üblicherweise kommt jedoch ein fehlererkennender Code zur Anwendung. In jedem Fall ist jedoch der Ablauf dieser Vorgänge viel zu rasch, um wie bei den Streckensystemen auf konventionelle Weise geprüft zu werden. Bei der Inbetriebnahme der PCM-Laborzentrale [6] wurde es als grosser Mangel empfunden, dass die sich auf den Steuerkanälen abspielenden Vorgänge nicht direkt erfasst werden konnten. Trotz der sehr einfachen Konfiguration war es beim Auftreten eines Fehlers oft äusserst schwierig festzustellen, welche Einheit dafür verantwortlich war. Es war weder möglich, eine Fehlersituation genau zu erfassen, noch sie ein zweites Mal künstlich herbeizuführen, um das Verhalten der Programmsteuerung zu prüfen. Dies führte zur Schaffung eines Systems, das in der Lage sein soll, auch die in künftigen Vermittlungssystemen vorkommenden komplizierten Fernsteuerungsvorgänge zu erfassen. Im Vordergrund steht dabei das geplante integrierte Fernmelde-System IFS-1 [9] [10], bei dem Systemblöcke über einen normalen PCM-Kanal mit Hilfe von Telegrammen ferngesteuert werden.

Folgende Verwendungsmöglichkeiten stehen im Vordergrund:

Einsatz als Messsystem. In diesem Fall erfolgt ein Betrieb als Signalempfänger. Folgende Aufgaben können gelöst werden:

- Ermittlung der Amplitudenstatistik eines Sprachkanals,
- Überwachung eines (bekannten) Codemusters, Messung der Fehlerrate,
- Überwachung eines Steuerkanals, Protokollierung der ablaufenden Vorgänge, zum Beispiel in Form einer Telegrammstatistik, Melden von aussergewöhnlichen Zuständen.

Prüfung eines ferngesteuerten PCM-Systemblocks (zum Beispiel Terminal, PCM-Durchschalteeinheit) hinsichtlich seiner Reaktion auf Steuertelegramme.

Simulation eines ferngesteuerten Systemblockes hinsichtlich seiner Reaktion auf Steuertelegramme.

2.2.2 Geplanter Endausbau

Wie *Figur 1* zeigt, bildet der Prozessor das Kernstück der Anlage. Obwohl sich einzelne der beschriebenen Aufgaben sehr wohl mit einer speziellen verdrahteten Steuerung ausführen liessen, dürfte es kaum möglich sein, ein universelles Gerät zu bauen, da die Anforderungen zu stark variieren. Zudem kommen gerade in einer experimentellen Umgebung, wo noch manches geändert wird, die Vorteile der Programmsteuerung voll zum Zug. Es ist durchaus denkbar, dass die Anlage für gewisse Aufgabenstellungen eher zu viele Möglichkeiten bietet und daher für einen Einsatz in grösserem Rahmen zu aufwendig ist, so dass für diese Fälle ein Spezialgerät entwickelt wird. Das vorliegende System bietet aber dann die Möglichkeit, verschiedene fest verdrahtete Varianten zu simulieren.

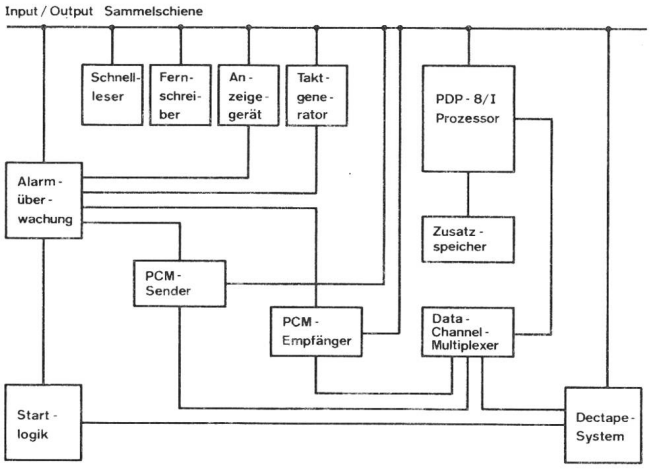

Fig. 1
Blockschema des PCM Messsystems im Endausbau

Die Konfiguration umfasst neben den normalen Computer-Peripheriegeräten (Fernschreiber, Schnelleser, decktape-Magnetband-System) die im folgenden beschriebenen Apparate:

Der *PCM-Empfänger* vermittelt den Übergang zwischen der ankommenden PCM-Vielfachleitung und dem Prozessor. Seine Hauptaufgabe besteht darin, aus dem ankommenden Impulsstrom von 2,048 Mbit/s einen bestimmten, vom Prozessor gewählten Kanal auszublenden und alle 125 μs direkt in den Kernspeicher zu transferieren. Auf die genaue Arbeitsweise soll hier nicht eingetreten werden, doch sei erwähnt, dass es bei vielen Anwendungen möglich sein wird, die hohe Informationsrate durch Weglassen bedeutungsloser Codekombinationen («Füllwörter») zu reduzieren. Übertragungsfehler werden ebenfalls im Empfänger erkannt. Da ein PCM-Wort 8 bit umfasst, der Prozessor PDP-8 hingegen mit 12 bit arbeitet, ist es möglich, die übrigen Bits zur Signalisierung solcher Fehler wie auch zur Erleichterung der Erkennung von häufig vorkommenden Spezialcodes zu benutzen. Falls die eigentliche Information mit Dipulsen übertragen wird, kann eine Codewandlung, wie in *Tabelle I* angegeben, wesentlich zur zeitsparenden Verarbeitung beitragen.

Der *PCM-Sender* besorgt die Ausgabe der vom Prozessor angelieferten Information in einem bestimmten Kanal und arbeitet analog dem Empfänger. Auch hier ist aus Gründen der Geschwindigkeit ein direkter Speicherzugriff vorgesehen.

Der *Taktgenerator* erzeugt alle 10 ms eine Programmunterbrechung, was zur Auslösung zeitabhängiger Programme gebraucht wird (real-time clock). Er hat dazu die wichtige Aufgabe, den einwandfreien Ablauf zu überwachen. Wird er nämlich nicht innerhalb 15 ms vom Prozessor bedient, wird Alarm ausgelöst, da in diesem Fall mit einem schwerwiegenden Programmfehler gerechnet werden muss. Über die *Alarmüberwachung* und die *Startlogik* kann dann eine Neubeladung des Programms vom Magnetband her erwirkt werden.

Das *Anzeigegerät* besitzt ein Feld von 9 × 12 Lampen, die programmässig angesteuert werden können. Es ist vorgesehen, einige von ihnen zur Signalisierung von Zuständen innerhalb des Rechners zu verwenden. Das Gerät bietet grosse Vorteile gegenüber dem Fernschreiber, insbesondere bei der Beurteilung von Fehlerzuständen, da die benötigte Information, die in den meisten Fällen ohnehin binär ist, rasch und geräuschlos angezeigt werden kann.

Der Entwurf dieses Systems zeigt die Anwendung einiger im Abschnitt 2.1 gestreifter Prinzipien. Verdrahtete Steuerungen wurden im Empfänger und Sender soweit eingesetzt, als es sinnvoll erschien, beispielsweise zum Auffinden des Synchronisationsmusters auf der ankommenden Vielfachleitung und zur Unterdrückung von Leerinformationen. Während die erste Aufgabe aus Zeitgründen nicht anders gelöst werden kann, steht die Alternative im letzteren Fall durchaus offen, wenn sie auch – wie noch gezeigt werden wird – nicht sehr effizient ist. Auf der anderen Seite würde das Erkennen ganzer Meldungen den Empfänger allzusehr komplizieren, so dass diese Aufgabe vollständig der Programmierung überlassen bleibt.

2.2.3 Konfiguration zur Entwicklung des Betriebssystems

Um frühzeitig mit der Programmierung beginnen zu können, wurde ein bestehender PCM-Empfänger, der die Ausblendung eines Kanals und Anzeige auf Lampen gestattet, mit einer Interface-Logik ausgerüstet. Dieses Gerät arbeitet im binär codierten Ternärcode (B-Code) und wurde im Zusammenhang mit dem Labormodell gebaut. Für den Anschluss an den Rechner wurde das Prinzip des programmierten Datentransfers mit Programmunterbruch gewählt [11]. Diese Methode ist zwar unwirtschaftlich im Blick auf Ausnützung der Prozessorzeit, lässt sich aber anderseits mit minimalem Aufwand verwirklichen. Die Einheit erzeugt zwei Signale:

- Programmunterbruch alle 125 μs zur Übertragung eines PCM-Wortes (Kanal)
- Programmunterbruch alle 2 ms (Überrahmen).

Die Bedingungen, die durch diesen Empfänger an die Programmierung gestellt werden, sind ausserordentlich hart. Kein Programmstück, das nicht unterbrechbar ist, darf länger als 125 μs dauern. Da der verwendete Rechner keine Prioritätsstruktur in seinem Unterbruchsystem aufweist, muss mit zahlreichen Möglichkeiten von ineinander verschachtelten Programmunterbrechungen gerechnet werden. Bei der Einführung des direkten Datentransfers werden dann diese Unterbrechungen wesentlich seltener vorkommen. Die inhärente Geschwindigkeit des Systems wird aber auch dann vorteilhaft sein, indem zum Beispiel in bestimmten Fällen auf eine Pufferung von Daten verzichtet werden kann. In jedem Fall gestattet der Versuchsaufbau einen sehr wirksamen Test der Programme.

Tabelle I. Codewandlung im PCM-Empfänger

Empfänger Code	Darstellung im Rechner											
Normales Dipulswort	O	O	O	O	O	O	O	O	X	X	X	X
Wort mit Dipulsverletzung	L	O	O	O	X	X	X	X	X	X	X	X
Freikanalcode	L	L	O	O	O	O	O	O	O	O	O	O
Freitelegrammcode	L	L	O	L	O	O	O	O	O	O	O	O
Startcode	L	L	L	O	O	O	O	O	O	O	O	O
Fehlercode	L	L	L	L	O	O	O	O	O	O	O	O

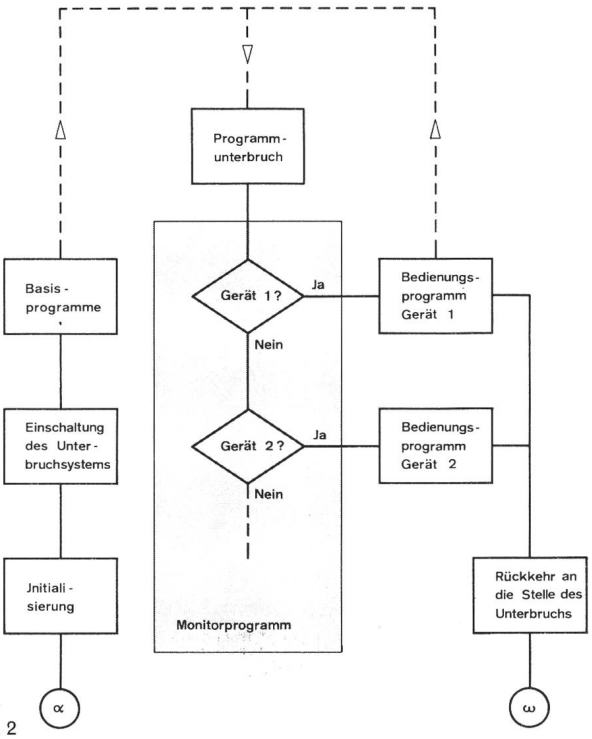

Fig. 2
Grundstruktur des Programmsystems

2.2.4 Organisation der Programme

Die zur Diskussion stehende Anlage stellt ein einfaches Echtzeitsystem dar, mit der Eigenschaft, dass gewisse Peripheriegeräte eine ausserordentlich schnelle Behandlung verlangen. Man kann daher eine Unterteilung in Überwachungs- oder Betriebsprogramme und Anwendungsprogramme vornehmen. Die Gesamtstruktur ist in *Figur 2* gegeben.

Das Betriebssystem

Die hauptsächlichsten Schwierigkeiten in der Echtzeitprogrammierung entstehen dadurch, dass der Prozessor laufend Daten mit Peripheriegeräten verschiedenster Geschwindigkeit austauschen muss. Während sich beim normalen Rechenbetrieb eine Überlappung der Ein- und Ausgabevorgänge zur Verbesserung der Ausnützung der Geräte aufdrängt, weil sonst Zentraleinheit und Aussengeräte abwechselnd aufeinander warten müssen [12], ist sie bei Anwendungen zu Prozesssteuerungs- und Überwachungsaufgaben Systembestandteil, da immer eine grössere Zahl von Vorgängen in der Aussenwelt gleichzeitig abläuft. Die Anpassung der Geschwindigkeiten erfolgt durch Pufferung der Information in einem Zwischenspeicher.

Eingabegeräte schreiben ihre Daten zunächst in einen solchen Speicher ein, dessen Inhalt vom Prozessor im geeigneten Augenblick übernommen wird. Die Daten für *Ausgabegeräte* werden vom Prozessor mit hoher Geschwindigkeit in einen Pufferspeicher eingeschrieben und vom Gerät mit der ihm eigenen Geschwindigkeit übernommen.

Obwohl auch spezialisierte Speicherblöcke möglich sind, wird meistens der Kernspeicher des Prozessors herangezogen. Die Zuordnung kann fix oder dynamisch aufgrund der gerade benötigten Kapazität sein. Im vorliegenden Beispiel wurden bestimmte Speicherbezirke einem Gerät fest zugeordnet, die im folgenden Tabellen genannt werden (zum Beispiel Druckertabelle, Empfängertabelle) und die zyklisch neu überschrieben werden. Bestimmte Speicherzellen dienen dabei als «Zeiger», wobei jede Tabelle je einen für den Prozessor und das Peripheriegerät besitzt. Besondere Beachtung wurde dem Problem des Überlaufs geschenkt, der auftritt, wenn eine Tabelle vollgeschrieben ist. Beispielsweise müssen weitere Meldungen für den Fernschreiber unterdrückt werden, wenn die Druckertabelle voll ist. Ein internes Signal («flag») informiert alle betroffenen Programme über diesen Zustand, damit solcherart blockierte Meldungen später doch noch ausgegeben werden können.

Schlimmer ist es, wenn in einer Eingabetabelle ein Überlauf auftritt, da in diesem Fall die betreffende Information verloren ist. In erster Linie muss verhindert werden, dass alte, noch unverarbeitete Daten dabei überschrieben werden. Ein Ausweg lässt sich nur finden, wenn die Geschwindigkeit der Verarbeitung gesteigert werden kann. Dies wird von Fall zu Fall verschieden sein. Im vorliegenden System kann es vorkommen, dass die Empfängertabelle voll ist. Läuft etwa ein Programm, das die ankommenden Telegramme analysiert, so muss auf die Ausführung von zeitintensiven Vorgängen verzichtet werden, indem eine Rückstufung auf ein einfacheres Analyseprogramm vorgenommen wird. Es ist daher unter Umständen vorteilhaft, einen Überlauf frühzeitig zu signalisieren, zum Beispiel wenn 80% der Tabelle gefüllt sind.

Die Aufgaben des Betriebssystems lassen sich in unserem Fall wie folgt formulieren:

- zeitliche Koordination aller Programme, Ermöglichung der korrekten Programmunterbrechungen,
- Kontrolle der Ein- und Ausgabe der Peripheriegeräte,
- Signalisierung zwischen verschiedenen Programmen,
- Kontrolle des korrekten Ablaufs, Alarmierung bei der Entdeckung von Überlaufzuständen,
- Bereitstellung von Unterprogrammen für alle wichtigen Vorgänge, die in den Anwendungsprogrammen vorkommen.

Zur Erläuterung soll die Behandlung des PCM-Empfängers beschrieben werden. Aus Geschwindigkeitsgründen ist die Verarbeitung einzelner Zeichen nicht sinnvoll. Das Überrahmensignal (später ein entsprechendes Signal nach

Empfang von 16 signifikanten PCM-Wörtern) startet das Empfängerprogramm. Dieses darf zwar im Mittel nicht länger als 2 ms dauern, wenn keine Leerinformation unterdrückt wird, doch werden gelegentliche Spitzen in der Empfangstabelle abgefangen. Ein neuer Durchlauf durch das Programm wird in diesem Fall verhindert, wobei ein Zähler die unerledigten Unterbrechungen registriert, die bei geringerer Belastung später abgebaut werden.

Die Anwendungsprogramme

Das Betriebssystem verwirklicht im wesentlichen die Struktur der Figur 2. Es wird für sich assembliert und bildet das Gerüst für den Einbau der Anwendungsprogramme. Diese bestehen aus verschiedenen Segmenten, die zu verschiedenen Zeiten aktiviert werden *(Fig. 3)*. Ein Tastaturbefehl bewirkt den Start des gerade geladenen Anwendungsprogramms, indem dessen Initialisierungsteil durchlaufen wird. Der weitere Ablauf ergibt sich aus dem Eintreffen von Programmunterbrechungen und Signalisation zwischen den verschiedenen Segmenten. Unter Umständen ist es günstig, die ganze Verarbeitung im Hintergrundprogramm zu konzentrieren und die Unterbrechungen von Sender, Empfänger und Taktgenerator nur zum Erhöhen von Zählern zu benutzen. Ist das Anwendungsprogramm beendet, gibt es die Kontrolle an das Betriebssystem zurück. Auf Wunsch kann es auch jederzeit vom Fernschreiber her beendet werden.

Gesamthaft legt man grossen Wert darauf, wenn immer möglich alle schwierigen und zeitlich kritischen Aufgaben im Betriebssystem zu konzentrieren. Durch Einbau von wirksamen Subroutinen, die mit sinnfälligen Bezeichnungen versehen wurden, ist versucht worden, die Anwendungsprogramme möglichst leicht lesbar zu machen. Bis jetzt wurde ausschliesslich in der Assemblersprache PAL-III programmiert. Ideal wäre es, an ihrer Stelle eine höhere Programmiersprache einzusetzen, doch konnte dies auf dieser Stufe noch nicht in Betracht gezogen werden. Die Verwendung von Makros in den Anwendungsprogrammen könnte vielleicht eine leicht verwirklichbare Alternative darstellen.

2.2.5 Beispiel für ein Anwendungsprogramm

Mit der zur Verfügung stehenden Konfiguration konnten zunächst nur Aufgaben des 1. Typs von Abschnitt 2.2.1 angegangen werden. Ein PCM-Generator gestattete dabei das periodische oder einmalige Aussenden von Meldungen mit einer Länge von 8 Wörtern zu je 5 dreiwertigen Positionen, die im folgenden mit +, — und ∅ bezeichnet werden.

Im vorliegenden Beispiel wurde versucht, mit den bestehenden Geräten die im IFS-1-System verwendete Telegrammsignalisierung nachzubilden. Dabei wurden ∅∅∅∅∅ als Freikanalcode und +++++ als Telegramm-Startcode definiert. Das Programm erkennt letzteren Code und druckt jedes «Telegramm» mit fortlaufender Numerierung auf einer Zeile aus. Es besteht aus zwei Segmenten, dem Initialisierungsteil und dem Empfangsprogramm.

Tabelle II zeigt ein Protokoll, das den Empfang von Telegrammen bestätigt, die durch Knopfdruck am PCM-Generator erzeugt wurden. Die Subroutinen des Betriebssystems ermöglichen dabei in einfacher Weise den Ausdruck von Texten und Daten in verschiedenen Formaten. So wird zum Beispiel die Telegrammnummer im dezimalen Format, der Inhalt dagegen im «A-Code»-Format ausgegeben.

3. Verwendung eines Kleinrechners zur Behandlung numerischer Probleme

3.1 Anforderungen an die Programmiersprache

Obwohl verschiedene Kleinrechner die Möglichkeit zur Programmierung in ALGOL oder FORTRAN bieten, wird davon im allgemeinen wenig Gebrauch gemacht. Folgende Gründe sind dafür massgebend:

- Die Ausrüstung mit Peripheriegeräten ist häufig ungenügend und verunmöglicht eine rationelle Herstellung von Programmen.
- Wegen beschränkter Speicherkapazität können oft nicht alle Möglichkeiten der betreffenden Programmiersprache ausgenützt werden.

Beispielsweise ist ein mit Fernschreiber und Schnelleser ausgerüsteter Rechner sehr gut geeignet, um vorhandene Programme auszuführen. Soll jedoch ein neues Programm erstellt werden, sind folgende Vorgänge notwendig:

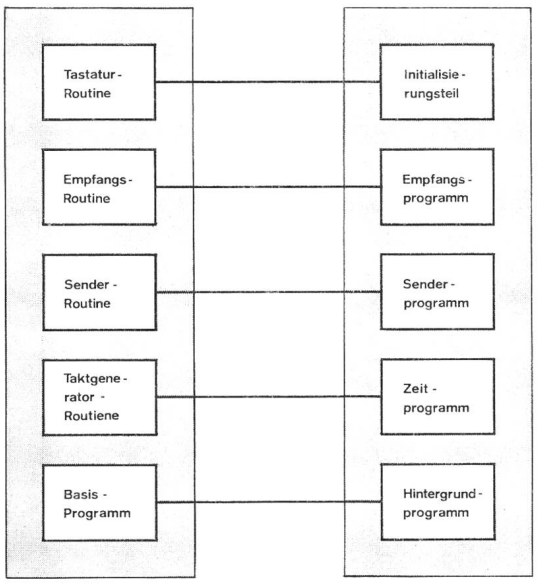

Fig. 3 Struktur der Anwendungsprogramme

Tabelle II. Anwendungsprogramm zum PCM-Messsystem für den Empfang von Befehlstelegrammen

```
START ANWENDUNGSPROGRAMM

EMPFANG VON TELEGRAMMEN
**********************
A-CODE-FORMAT:


+0001   00000   00000   00000   00000   00000   00000   00000
+0002   00000   00000   00000   00000   00000   00000   00000
+0003   00000   00000   00000   00000   00000   00000   00000
+0004   00000   00000   00000   00000   00000   00000   00000
+0005   -0000   0-000   00-00   00-00   000-0   00000-  00000+
+0006   -0000   0-000   00-00   00-00   000-0   00000-  00000+
+0007   -0000   0-000   00-00   00-00   000-0   00000-  00000+
+0008   --000   0--00   00--0   000--   00000-   -0000-  0++00
+0009   0+0-0   +0-0+   0-0+0   +0-0+   0-0+0   -0+0-   0+0-0
+0010   0+0-0   +0-0+   0-0+0   +0-0+   0-0+0   -0+0-   0+0-0
+0011   0+0++   +0-++   0-0--   +0-0-   +-0+0   -0+0-   0+++0
+0012   0+0++   +0-++   0-0--   +0-0-   +-0+0   -0+0-   0+++0
+0013   0+0++   +0-++   0-0--   +0-0-   +-0+0   -0+0-   0+++0
+0014   ++++0   ++++0   ++++0   ++++0   ++++0   ++++0   ++++0
+0015   ++++0   ++++0   ++++0   ++++0   ++++0   ++++0   ++++0
+0016   -+++0   -+++0   -+++0   -+++0   -+++0   -+++0   -+++0
+0017   -+++0   -+++0   -+++0   -+++0   -+++0   -+++0   -+++0
+0018   -+++0   -+++0   -+++0   -+++0   -+++0   -+++0   -+++0
+0019   -+++0   -+++0   -+++0   -+++0   -+++0   -+++0   -+++0
+0020   -+++0   -+++0   -+++0   -+++0   -+++0   -+++0   -+++0
+0021   -+++0   -+++0   -+++0   -+++0   -+++0   -+++0   -+++0
+0022   -+++0   -+++0   -+++0   -+++0   -+++0   -+++0   -+++0
+0023   -+++0   -+++0   -+++0   -+++0   -+++0   -+++0   -+++0
+0024   -+++0   -+++0   -+++0   -+++0   -+++0   -+++0   -+++0
+0025   -+++0   -+++0   -+++0   -+++0   -+++0   -+++0   -+++0
+0026   -+++0   -+++0   -+++0   -+++0   -+++0   -+++0   -+++0
+0027   -++-0   -++-0   -++-0   -++--   -++--   -++--   -++-0
```

- Erstellen eines Lochstreifens in der höheren Programmiersprache (source), entweder durch direktes Stanzen oder unter Benutzung eines Textverarbeitungsprogrammes, das die Korrektur von Schreibfehlern erlaubt.
- Laden des Compilers (Übersetzungsprogramm).
- Übersetzen des Programms. Falls syntaktische Fehler auftreten, ist eine Rückkehr zum ersten Vorgang notwendig. Bei erfolgreicher Compilierung stanzt die Maschine eine übersetzte Version.
- Laden, allenfalls mit vorgängigem Assemblieren des übersetzten Programms (gewisse Compiler erstellen ein Zwischenprodukt in Assemblersprache).
- Laden eines Bibliothekprogramms, das Systemsubroutinen enthält.
- Test des Programms. Falls logische Fehler auftreten, muss das Source-Programm korrigiert und der ganze Vorgang wiederholt werden.

Man sieht leicht ein, dass eine solche Arbeitsweise kaum in Kauf genommen wird, wenn gleichzeitig Einschränkungen bezüglich des Umfanges der Programme bestehen. Der Benutzer profitiert wenig davon, dass er den Rechner ganz zu seiner Verfügung hat, weil dieser während der meisten Zeit durch Ein- und Ausgabevorgänge blockiert ist. Die Rechenzeit des Computers, die auf diese Weise verschenkt wird, fällt weit weniger ins Gewicht als die Zeit, die der Benutzer durch Warten und Manipulieren von Lochstreifen verliert. Um den Rechner in dieser Richtung attraktiver zu machen, muss das Verfahren vereinfacht werden, etwa wie folgt:

- Eingabe des Programms in den Rechner durch einen einfachen Ladevorgang oder direkt über die Tastatur des Fernschreibers.
- Übersetzung und Ausführung ohne Ausgabe von Zwischendaten.
- Leichte Korrekturmöglichkeit am bestehenden Programm.

Ein solcher Ablauf lässt sich bei den üblichen Programmiersprachen nur durch Ausrüsten des Rechners mit einem Plattenspeicher und einem entsprechenden Betriebssystem erzielen, da es nicht möglich ist, das source-Programm, den Compiler und das übersetzte Programm gleichzeitig im Kernspeicher zu halten. Somit bleibt nur die Verwendung einer Programmiersprache übrig, bei der keine Compilierung erfolgt. Bei dieser Arbeitsweise, die interpretativ genannt wird, bleibt das originale Programm stets im Kernspeicher und kann daher jederzeit abgeändert werden. Bei der Ausführung wird eine Anweisung nach der andern durch einen sogenannten «Interpreter» analysiert und in den Maschinencode übersetzt. Der grosse Nachteil dieser Verarbeitungsweise besteht in der geringen Geschwindigkeit, was sich vor allem in Programmen mit Iterationen bemerkbar macht, da beispielsweise die Anweisungen in einer Schleife jedes Mal wieder auf Einhaltung der syntaktischen Regeln geprüft und übersetzt werden. Ebenfalls hier wird Rechenzeit verschwendet. Dies kommt aber dem Benutzer in vielen Fällen gar nicht zum Bewusstsein, weil bei der hohen internen Geschwindigkeit des Rechners der Gesamtablauf bei einfacheren Berechnungen letzten Endes doch durch die Ein- und Ausgabe von Parametern und Resultaten beschränkt bleibt.

3.2 Die FOCAL-Sprache

Eine eingehende Beschreibung dieser Sprache (*F*ormulating *O*n-line *C*alculations in *A*lgebraic *L*anguage) ist in [13], [14] gegeben. Es sollen hier nur die Punkte herausgehoben werden, die die Verwendbarkeit auf einem Kleinrechner gewährleisten, wobei das Programm in *Tabelle III* zur Illustration herangezogen wird:

- Die Sprache stützt sich auf einige wenige Befehle, deren Bedeutung zum Teil von FORTRAN entlehnt ist. Dabei wurden vor allem Ein- und Ausgabe so praktisch als möglich gestaltet und jeglicher Formalismus vermieden. Als Beispiel soll der «TYPE»-Befehl (Abkürzung T) genannt werden, der es ermöglicht, in einer Linie Berechnungen zu veranlassen und die Resultate in beliebigen Formaten zusammen mit Text zu drucken (Linie ⌀1.16).
- Befehle können entweder zeilenweise interpretiert oder aber, mit einer Nummer versehen, zur späteren Ausführung in einem Programm gespeichert werden. Die erste Betriebsweise bietet dem Benutzer einen äusserst komfortablen Tischrechner zur augenblicklichen Auswertung

von Formeln aller Art. Beispielsweise bewirkt die folgende Zeile die Tabellierung der Exponentialfunktion von 0...3 mit einem Schritt von 0.05:

FOR A = 0,0.05,3; TYPE % 4.02,A,%10.06, FEXP(A),!

Die Verfügbarkeit solcher Befehle bedeutet auch eine grosse Hilfe beim Prüfen von FOCAL-Programmen, da zum Beispiel bestimmte Werte schnell nachgerechnet werden können, was das Auffinden von Fehlern erleichtert.

- Programme können auf der Tastatur oder mit Hilfe von Lochstreifen eingegeben werden und sind stets im Kernspeicher greifbar. Änderungen sind in äusserst rationeller Weise mit Hilfe eines Befehls (MODIFY) möglich, womit einzelne Zeichen innerhalb einer Zeile verändert werden können. Das Programm wird erst dann zur späteren Verwendung gestanzt, wenn ein befriedigender Zustand erreicht ist, so dass lästige Wartezeiten weitgehend dahinfallen.

3.3 Anwendung für übertragungstechnische Probleme

Im Rahmen einer grösseren Arbeit betreffend PCM-Übertragung über Teilnehmerkabel wurden unter anderem die Betriebsdaten (Betriebsdämpfung und -phase, Eingangsimpedanz) von Tiefpassfiltern aus mehreren identischen Grundgliedern benötigt *(Fig. 4)*. Tabelle III zeigt das Programm, das aus einem Hauptteil (Zeilennummern 01.01... 01.98) und drei Subroutinen (02.02...02.27, 15.10...15.33, 15.40) besteht. Zunächst wird ein Titel ausgedruckt, anschliessend erfolgt die Abfrage von Parametern an der Tastatur (A-Befehl). Nach der Eingabe entsprechender Werte werden zunächst die Schaltelemente berechnet. In einer Schleifenanweisung folgen dann die gewünschten Berechnungen. Zunächst werden die Wellenparameter nach bekannten Formeln bestimmt. Die Subroutine mit den Zeilennummern 02.XX berechnet die Betriebsparameter nach den komplexen Formeln:

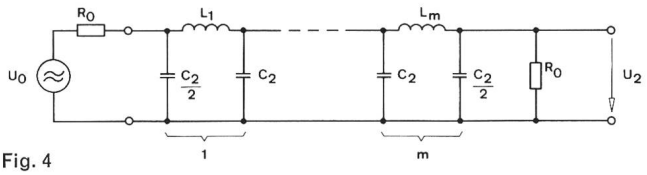

Fig. 4
Tiefpassfilter aus m Grundgliedern mit reeller Beschaltung

$$\Gamma_B = A_B + jB_B = \Gamma + \ln \frac{1 - r^2 \cdot e^{-2\Gamma}}{1 - r^2}$$

$$Z_1 = Z_W \cdot \frac{e^{\Gamma} + re^{-\Gamma}}{e^{\Gamma} - re^{-\Gamma}}$$

$\Gamma = A + jB$ Wellenübertragungsmass

$r = \frac{R_0 - Z_W}{R_0 + Z_W}$ Reflexionsfaktor

Dabei besorgen die Subroutinen in 15.XX die Umwandlung komplexer Zahlen von der kartesischen zur Polarform und umgekehrt. Nach der Rückkehr werden in Zeile 01.98 die Resultate ausgedruckt. *Tabelle IV* zeigt ein Berechnungsbeispiel.

Tabelle III. Beispiel für ein FOCAL-Programm

```
C-8K FOCAL #1969

01.01 T !"BETRIEBSVERHALTEN EINER TIEFPASSKETTE AUS GRUNDGLIEDERN"!
01.02 T "27.12.71 R.L."!!!
01.10 A !!"ABSCHLUSSWIDERSTAND"R0; A "K-FAKTOR"K
01.11 A "GRENZFREQUENZ"GG; A "ANZAHL GLIEDER"M
01.12 A "FREQUENZRASTER"DG; A "ANZAHL PUNKTE"N
01.16 T %, !"ELEMENTE: L1", K/(3.14159*GG)," C2",1/(3.14159*GG*K)
01.20 T !!!"FREQUENZ          AB          BB          ZB"
01.21 T "                 %5"!!
01.30 F I=1,N; D 1.9
01.31 Q
01.90 S OM=DG*I/GG; J (OM-1) 1.91,1.93,1.95
01.91 S ZI=0; S ZR=K/FSQT(1-OM↑2); S A=0; G 1.92
01.92 S B=2*M*FATN(OM/FSQT(1-OM↑2)); D 2.0; G 1.98
01.93 T "GRENZFREQUENZ"!; R
01.95 S ZI=-K/FSQT(OM↑2-1); S ZR=0; S B=M*3.14159; G 1.96
01.96 S A=2*M*FLOG(OM+FSQT(OM↑2-1)); D 2.0; G 1.98
01.98 T % R,DG*I," ",%,AB," ",BB," ",01," ",02,!

02.02 S PR=R0-ZR; S PI=-ZI; D 15.0
02.03 S 01=PB; S 02=PW
02.05 S PR=R0+ZR; S PI=ZI; D 15.0
02.07 S RB=01/PB; S RW=02-PW
02.09 S 01=RB↑2; S 02=2*RW
02.10 S PB=01*FEXP(-2*A); S PW=02-2*B
02.12 D 15.4; S PR=1-PR; S PI=-PI; D 15.0; D 2.03
02.14 S PB=RB↑2; S PW=2*RW; D 15.4
02.16 S PR=1-PR; S PI=-PI; D 15.0
02.17 S PB=01/PB; S PW=02-PW
02.20 S AB=A+FLOG(PB); S BB=B+PW
02.21 S PB=RB*FEXP(-A); S PW=RW-B; D 15.4
02.22 S PR=PR+FEXP(A)*FCOS(B); S PI=PI+FEXP(A)*FSIN(B)
02.23 D 15.0; D 2.03; D 2.21
02.24 S PR=FEXP(A)*FCOS(B)-PR; S PI=FEXP(A)*FSIN(B)-PI
02.25 D 15.0; S 01=01/PB; S 02=02-PW
02.26 S PR=ZR; S PI=ZI; D 15.0
02.27 S 01=01*PB; S 02=02+PW

15.10 S PB=FSQT(PR↑2+PI↑2); I (PR) 15.15,15.3,15.2
15.15 I (PI) 15.17,15.16,15.16
15.16 S PW= 3.14159+FATN(PI/PR); R
15.17 S PW=-3.14159+FATN(PI/PR); R
15.20 S PW=FATN(PI/PR); R
15.30 I (PI) 15.31,15.32,15.33
15.31 S PW=-1.570800; R
15.32 S PW=0; R
15.33 S PW=1.570800; R
15.40 S PR=PB*FCOS(PW); S PI=PB*FSIN(PW)
*
```

3.4 Praktische Grenzen

Aus dem im vorhergehenden Abschnitt gezeigten Beispiel ist ersichtlich, dass die FOCAL-Sprache, trotz ihrer bestechend einfachen Befehle, anspruchsvolle Berechnungen gestattet. Bei einem PDP-8-System mit einem Speicher von 8 k (8192 Worte) werden dabei 4 k für das FOCAL-Interpretationsprogramm und den Arbeitsspeicher verwendet, der Platz für Variablen und Rückkehradressen beim Aufruf von Subroutinen bietet. Praktisch ist die Speicherung von etwa 160 Variablen in Gleitkommadarstellung mit der in Tabelle IV gezeigten Genauigkeit möglich. Die zweiten 4 k gestatten die Ladung von Programmen mit einem Umfang von rund 7600 alphanumerischen Zeichen.

Das grösste von uns bisher erstellte Programm, das praktisch die ganze Speicherkapazität beansprucht, weist 197 Zeilen auf. Die Erfahrungen, die bei dessen Entwicklung gemacht wurden, können wie folgt beschrieben werden:

- Die Möglichkeit zum schrittweisen Aufbau und Test wurde voll ausgenutzt und erwies sich als äusserst nützlich. Beispielsweise kann ein Testprogramm für eine bestimmte Routine oft ohne Vorbereitung direkt eingetastet, ausgeführt und wieder gelöscht werden. Weil keine Compilierung geschieht, ist der damit verbundene Aufwand äusserst gering.
- Die schematische Numerierung der Zeilen wurde gelegentlich als störend empfunden, insbesondere wenn nachträglich Ergänzungen nötig werden. Zwar lassen sich verschiedene Zeilen durch einen GOTO-Befehl zusammenhängen (siehe Zeile 01.92), doch wirkt dies nicht sehr elegant.
- Die starre Speicherorganisation, die sich aus der Rechnerstruktur erklären lässt, ermöglicht verhältnismässig lange Programme mit einer bescheidenen Zahl von Variablen. Damit ist die Anwendung auf eine bestimmte Klasse von Problemen beschränkt.
- Die Durchführung von verschachtelten Schleifenanweisungen, die umfangreiche Berechnungen enthalten, ist nicht sinnvoll, da sehr bald Wartezeiten von einigen Sekunden bis zu Minuten auftreten können, welche die eingangs erwähnten Vorteile des direkten Rechenbetriebs illusorisch machen. Bei der Auslegung von Algorithmen ist daher diesem Punkt besondere Beachtung zu schenken.

Tabelle IV. Resultate des FOCAL-Programms

```
G

BETRIEBSVERHALTEN EINER TIEFPASSKETTE AUS GRUNDGLIEDERN
27.12.71 R.L.


ABSCHLUSSWIDERSTAND:300
K-FAKTOR:250
GRENZFREQUENZ:104000
ANZAHL GLIEDER:2
FREQUENZRASTER:10000
ANZAHL PUNKTE:25

ELEMENTE: L1= 0.765168E-03  C2= 0.122427E-07

FREQUENZ          AH              BH              ZB              ZW

=    10000 = 0.224694E-02 = 0.390709E+00 = 0.285134E+03 =-0.123757E+00
=    20000 = 0.454504E-02 = 0.780704E+00 = 0.255693E+03 =-0.162715E+00
=    30000 = 0.816434E-02 = 0.117392E+01 = 0.237109E+03 =-0.996790E-01
=    40000 = 0.520057E-02 = 0.157912E+01 = 0.244506E+03 = 0.171423E-02
=    50000 = 0.106421E-02 = 0.200563E+01 = 0.275918E+03 = 0.389223E-01
=    60000 = 0.796283E-04 = 0.245973E+01 = 0.304813E+03 =-0.196109E-01
=    70000 = 0.250069E-03 = 0.295213E+01 = 0.302537E+03 =-0.439382E-01
=    80000 = 0.467349E-02 = 0.352242E+01 = 0.322126E+03 = 0.178839E+00
=    90000 = 0.953426E-01 = 0.423602E+01 = 0.630719E+03 = 0.433119E+00
=   100000 = 0.452349E+00 = 0.499739E+01 = 0.155551E+04 =-0.820217E+00
=   110000 = 0.100906E+01 = 0.556330E+01 = 0.660101E+03 =-0.147617E+01
=   120000 = 0.157443E+01 = 0.592958E+01 = 0.430476E+03 =-0.154743E+01
=   130000 = 0.208879E+01 = 0.617883E+01 = 0.333060E+03 =-0.156303E+01
=   140000 = 0.255041E+01 = 0.636114E+01 = 0.277480E+03 =-0.156774E+01
=   150000 = 0.296740E+01 = 0.650203E+01 = 0.240607E+03 =-0.156944E+01
=   160000 = 0.334762E+01 = 0.661526E+01 = 0.213883E+03 =-0.157015E+01
=   170000 = 0.369734E+01 = 0.670888E+01 = 0.193372E+03 =-0.157046E+01
=   180000 = 0.402140E+01 = 0.678795E+01 = 0.176991E+03 =-0.157062E+01
=   190000 = 0.432356E+01 = 0.685586E+01 = 0.163523E+03 =-0.157070E+01
=   200000 = 0.460680E+01 = 0.691497E+01 = 0.152202E+03 =-0.157074E+01
=   210000 = 0.487350E+01 = 0.696693E+01 = 0.142518E+03 =-0.157076E+01
=   220000 = 0.512560E+01 = 0.701315E+01 = 0.134116E+03 =-0.157078E+01
=   230000 = 0.536471E+01 = 0.705452E+01 = 0.126742E+03 =-0.157079E+01
=   240000 = 0.559218E+01 = 0.709176E+01 = 0.120207E+03 =-0.157079E+01
=   250000 = 0.580914E+01 = 0.712555E+01 = 0.114366E+03 =-0.157079E+01
*
```

4. Schlussfolgerungen

Es war das Ziel dieser Arbeit, anhand von ausgewählten, verwirklichten Beispielen auf die Anwendung von Kleinrechnern aufmerksam zu machen. Als *Steuerelemente* gestatten sie die Lösung einer Vielzahl von Problemen unter Benutzung der gleichen Geräte, so dass sich der Aufbau von schlecht ausgenützten Spezialausrüstungen umgehen lässt. Als *Rechenmaschine* ist der Kleinrechner gegenüber den grossen Vorbildern beschränkt, kann aber doch Probleme von beachtlichem Umfang meistern. In jedem Fall ist er ein Hilfsmittel des Ingenieurs, dem auch in Zukunft ein breites Anwendungsfeld offensteht.

Adresse des Autors: R. P. Lorétan, Dep. of Electr. Eng. Science, Univ. of Essex, Wivenhoe Park, Colchester, Essex, Great Britain.

Literaturverzeichnis

[1] *Jurgen Ronald K.* Minicomputer applications in the seventies. IEEE Spectrum. 7 (1970) 8, p. 37...50.

[2] Div. authors. The Minicomputer and the engineer, series in Electronic Design. Vol. 19 (1971), 8, 9, 10, 11, 12, 13.

[3] *Ball Ch. J.* Communications and the Minicomputer. Computer. 4 (1971) 5, p. 13...21.

[4] *Van de Goor Ad, Bell Gordon, Witcraft Donald.* Design and Behavior of TSS/8: a PDP-8 Based Time-Sharing System. IEEE Transactions on Computers. C-18 (1969) 11, p. 1038... 1043.

[5] Digital Equipment Corporation, Introduction to programming, Maynard 1970, Chapter 10.

[6] *Lorétan R., Röthlisberger J.* Laboratoriumsmodell einer PCM-Vermittlungseinrichtung mit Programmsteuerung. Technische Mitteilungen PTT. 49 (1971), 6, S. 393...412.

[7] *Corner A. C.* The 5005-Crossbar telephone-exchange switching System. Post Office Electrical Engineers' Journal. 59 Part 1, p. 170...177, Part 2, p. 271...281.

[8] *Hain P., Nestel J., Sperlich J.* The New Telefunken Electronic Switching System EZM 3, IEEE Transactions on Communication Technology. COM-17, 1969, 6, p. 600...605.

[9] *Bachmann A. E., Lorétan R. P.* The integrated Digital Telecommunications System IFS-1. Eurocon 71 Digest (IEEE Region 8 Convention) B 8-3.

[10] *Fontolliet P. G.* Transmission of Control Information in IFS-1, 1972 International Zürich Seminar on Integrated Systems for Speech, Video and Data Communications, Proceedings, B5.

[11] *Kinter P. M.* Interfaces zwischen Regelvorrichtungen und Computern, Orbit. 6 (1971), 6, S. 21...27.

[12] *Barron D. W.* Computer Operating Systems, Chapman and Hall Ltd, London 1971, Chapter 2.

[13] *Merrill R.* Program your Minicomputer in Focal, Electronic Design. 18 (1970), 15, p. 86...89.

[14] Digital Equipment Corporation, Programming Languages Maynard 1970.