1/1994

VIERTE DIMENSION

10. Jahrgang 1994 1. Quartal DM 10,--

:MESSEN;

Messung an Hüftgelenken

Messen via Centronics

:STEUERN;

DOS unter Kontrolle

Der I2C-Bus in Forth realisiert

:REGELN;

PID-Regler im Überblick

F P Fast Fourier Transformation

FUDGE, MS und Meßgenauigkeit

FORTH MAGAZIN
Organ der Forth-Gesellschaft e.V.

IMPRESSUM

Name der Zeitschrift
VIERTE DIMENSION
FORTH MAGAZIN
Organ der Forth-Gesellschaft e.V.

Herausgeber
Forth-Gesellschaft e.V.
Postfach 1110
85701 Unterschleißheim
Tel.: 089-3173784 oder
Forth-Mailbox Tel. 089-8714548 8N1

Redaktionsleitung
Rolf Kretzschmar (rk), (verantwortlich)
Rote Gasse 7, 52499 Baesweiler
(Redaktionsadresse)
Tel/Fax: 02401-88891

Redaktion
Arndt Klingelnberg (akg), Alsdorf
Tel.: 02404-61648 Fax: 02404-63039
Klaus-Peter Schleisiek (kps), Aachen
Tel/Fax: 0241-873462

Layout, Satz, Herstellung
ORGA Sport, Rilkestr. 8, 52477 Alsdorf
Tel/Fax: 02404-61425

Grafik, Illustration, Layout
Rolf Kretzschmar (rolf)

Anzeigenverwaltung
Arndt Klingelnberg
Straßburger Str. 12
52477 Alsdorf
Tel.: 02404-61648 Fax: 02404-63039

Redaktionsschluß
Feb., Mai, Aug., Nov.

Erscheinungsweise
vierteljährlich

Auflage
1000

Preis
Einzelheft DM 10,-, Abonnementpreis DM 50,-, bei Auslandsadresse DM 55,- inklusive Versandkosten

Das Thema

von Rolf Kretzschmar

Ich liebe Themenhefte!...solange sie mein Interessengebiet behandeln. Ich hasse Themenhefte!...wenn mich die Themen nicht interessieren. Im ersten Fall bekommt das Heft ein gelbes Post-it-Blättchen zur Krönung und zum schnellen Wiederfinden; im anderen Fall könnte ich es eigentlich ungelesen fortwerfen. Sind Sie an Messen, Steuern, Regeln interessiert? Schade; werfen Sie aber das Heft nicht weg! Vielleicht kennen Sie ja jemanden, dem Sie damit eine Freude machen können. Und Sie? Na bitte, ich hab's doch gewußt! Sie sind dann sicher mit mir der Meinung, daß Forth auf diesem Gebiet saugut zu gebrauchen ist. Und es wird gebraucht, wie *Claus Vogt* an einem Hüftgelenk zeigt, dem Belastungsinformationen mit Forth entlockt werden. Ich will jetzt nicht das Editorial zur Themenheftkurzübersicht verkommen lassen und sage daher nur noch, daß die Redaktion sich über die Beiträge zur ersten Themen-VD gefreut hat. Lange sah es so aus, als kämen nicht genug Artikel zusammen. Die Artikel, die nun keinen unmittelbaren Bezug zum Thema haben, dienen listigerweise dazu, den Themen-Ignoranten ein Alibi für's Nicht-Wegwerfen zu liefern.

Das andere Thema wird in diesem Heft leider nur in Form einer gelben Zahl auf dem Titelblatt angedeutet: Die Forth-Gesellschaft e.V. begeht das 10. Jahr ihres Bestehens. Keine Lobeshymnen sind eingegangen, kein Grußwort der C-, Pascal- oder Sonstwie Gesellschaft eingetroffen. Kein Programmierer hat sich bedankt, kein Profi seiner Freude Ausdruck verliehen.

Zehn Jahre. Forth wird weiter verwendet werden und Freunde haben.

Kein Thema!

Tschüß, Euer

R. Kretzschmar

FSSS

Forth simple, short and safe

von Arndt Klingelnberg
Straßburger Straße 12, 52477 Alsdorf, Tel.: 02404 - 61648, Fax: 02404 - 63039

Creating Words eignen sich in idealer Weise, um eine Mehrzahl von Worten, die fast Gleichartiges tun, zu erschaffen. Sie können optimiert werden, um ihren Gebrauch zu vereinfachen und gegen Fehler abzusichern, was die Creating Worte selbst allerdings verkompliziert. Das nachfolgende Beispiel zeigt gleichzeitig, wie der InputStream mehrfach genutzt werden kann.

Die 'Laufwerk-Wähl-Worte' mußten für meinen Bedarf modifiziert werden, weil erstens F-PC im Netzwerk mit vielen Laufwerken genutzt werden sollte, zweitens das Laufwerk aus einem Forth-Shell heraus nicht gewechselt werden darf.

Jedes Laufwerk-Wähl-Wort benötigt nun soviel Platz wie eine Variable (VALUE) mit einem zwei-buchstabigen Header, also 5 Byte Code-Space und 9 Byte Head-Space . Vorher waren es (SELECT) jeweils zusätzlich 16 Byte List-Space. Gespart: 16 * 24 = 384 Bytes.

Ja, es wäre sogar möglich, den 'body' (pfa) mit Hilfe von ! > zu modifizieren und damit ein Laufwerk zu 'remappen'.

Will man diese Value-Kompatibilität nicht, so könnte jeweils noch ein Byte gespart werden durch c, und c@.

Der Editor mag nicht, daß -während er aktiv ist, aber außerhalb von ihm- in einem aufgesetzten ForthShell am Drive bzw. dem FileHandle manipuliert wird, daher erfolgt ein EXIT für Fshell? == TRUE [1].

Anmerkung (1): Zum Verständnis: Im hier verwendeten F-PC-ak kann in ein 'getrenntes' Forth gesprungen werden, auch aus dem Editor heraus (OHNE den Editor zu verlassen), was häufig die Programmierung bequemer und schneller macht.

Listung zu FSSS (Arndt Klingelnberg)

```
\ DRIVE.seq     CHANGE drive, short simple safe,                    \ akg92oct24

: DRIVE:        \ create words to change DOS drive ( space efficient! ) \ akg92oct24
        >in @                           \ where are we in the input stream
    CREATE      ( <name> - )    \ Select drive n1 as the current disk drive.
        save!> >in                      \ save and set inputstream back
        bl word  dup>R
        c@ 2 <>                         \ test if length = 2
        R@ 2+ c@  ':' <> or             \ test if delimited by ':'
  ABORT" NO proper drive name"
        R> 1+ c@  upc 'A' -  ,          \ compile 0=A, 1=B etc.
        restore> >in                    \set inputstream to proper position
    DOES>       ( - )
        fSHELL?                         \ are we in a FORTH-shell
        IF      drop TRUE ?syserror     \ NO dos-shell from forth-shell \ akg92oct03
      EXIT
        THEN
        @ $0E ( 14 ) bdos ( DX fun - AL ) drop          \ change drive
        seqhandle >hndle @ -2 =
        if      -1 seqhandle >hndle !
        then ;


drive: a:       \ change dos-drive to  A:
drive: b:
drive: c:
\ ...
drive: z:

comment:                ===== für nicht-F-PC-ler: ===

upc     UpperCaseConvert nach Tabelle, kann auch 'ö' durch 'O' ersetzen.
        vereinfacht ersetzbar durch:
        ASCII _ ( $5F ) AND
save!> >in      eine kurze und ganz wesentlich schnellere Variante von
        >in @ >R >in !
restore> >in
        R> >in !
fSHELL?         sind wir vom Editor in eine ForthShell gesprungen , True == Ja
?syserror       <>0 ==> gibt eine Error Meldung aus
```

Bericht des Direktoriums zum 1. April:

Das Forth-Büro meldet zum 1. April:

April-News aus der Forth-Mailbox

Inhalt

Verschiedenes

Hüftkontakt

Eine Forth-Anwendung zur in-vivo Messung der Kräfte in künstlichen Hüftgelenken

von Claus Vogt
Bülowstr. 67, 10783 Berlin, Tel.: 030-2168938

Bei der hardwarenahen Programmierung kleiner Prozessor-Systeme hat die Programmiersprache Forth - neben C und anderen - ihren angestammten Platz behauptet. Aber auch größere Echtzeit-Anwendungen, die mit ansprechender Benutzungsoberläche, aufwendigen Rechenoperationen und hohem Datendurchsatz das PC/DOS-Gespann bis an seine Grenzen treiben, lassen sich in Forth schreiben. Am Biomechanik-Labor der FU Berlin erfaßt das Forth-Programm ATMA die Kräfte in künstlichen Hüftgelenken und anderen Implantaten.

Künstliche Implantate wie die Hüftgelenks-Endoprothesen haben Eingang in die tägliche Praxis einer orthopädischen Klinik gefunden. Die Untersuchung der mechanischen Belastung dieser Implantate liefert dem Hersteller Hinweise für die Verbesserung seiner Produkte. Der Arzt erfährt, welche Bewegungen kritisch für das Implantat sind. Der Patient kann sein Verhalten darauf einstellen und bewußter mit der Prothese leben.

Am Biomechanik-Labor des FU-Klinikum Steglitz (Berlin) werden seit 1988 Hüftgelenksprothesen mit inneren Miniatur-Kraftmeßsystemen ausgerüstet. Während der Patient bestimmte Übungen ausführt, werden die räumlichen Kräfte in der Prothese gemessen, telemetrisch übertragen, im PC ausgewertet und grafisch dargestellt. Ähnliche Kraftmeßsysteme werden derzeit auch für Wirbelsäulenimplantate und künstliche Schultergelenke entwickelt.

Das hier vorgestellte Programm dient der Erfassung, Vorverarbeitung und grafischen Darstellung der Meßdaten.

Messung und Übertragung der Kräfte

Für die gestellte Aufgabe sind handelsübliche Kraftaufnehmer viel zu groß. Aufnehmer, Elektronik und Sender sind in einem Hohlraum von etwa 8 mm Durchmesser unterzubringen. Durch Laser- und Elektronenstrahlschweißen wird die gesamte Elektronik absolut dicht gekapselt. Übliche Aufnehmer für räumliche Kräfte verwenden je Kraftkomponente zwei (oder vier) exakt ausgerichtete Dehnungsmeßstreifen (DMS), die als Halbbrücke (oder Vollbrücke) verschaltet werden. Bei Verwendung der Matrixmethode nach Bergmann [3] ist nur ein DMS je Kraftkomponente erforderlich und das exakte Ausrichten entfällt. Die Kraftkomponenten werden durch Multiplikation der DMS-Signale mit einer Übertragungs-Matrix ermittelt. Meßfehler, die sich durch Temperaturabhängigkeit und Nichtlinearität der DMS ergeben, werden auf numerischem Weg korrigiert. Hierzu ist ein zusätzlicher Temperatursensor erforderlich. Die erforderlichen Kennwerte werden vor der Implantation gemessen.

Die Energieversorgung von Meßaufnehmer und Sender erfolgt induktiv durch eine Spule, die dem Patienten um die Hüfte gelegt wird. Die aus dem Körper herausgesendeten Signale der einzelnen DMS werden seriell übertragen, die Information ist dabei im Abstand zwischen zwei übertragenen Impulsen kodiert (PIM=Puls-Intervall-Modulation). Die Ergebnisse können unmittelbar während der Messung am PC grafisch dargestellt werden. Außerdem werden sie für eine spätere Auswertung zusammen mit der Kamera-Aufzeichnung der Bewegung auf der HiFi-Spur einer Video-Kassette aufgezeichnet.

Rückblick

Nach aufwendiger Prüfung von Funktion und Sicherheit des Verfahrens erhielt der erste Patient zwei Meßprothesen im Mai und September 1988.

Die Entwicklung des Programmes ATMA (Algorithmus für Telemetrie und Meßwertaufnahme) begann 1987. Zwei Jahre später erforderte der historische Wildwuchs der Software eine Neukonzeption (ATMA II), deren Fertigstellung ich 1991 übernahm. Die Umstellung auf den Wirbelfixateur interne mit 8-Kanal-Telemetrie umd größeren Datenmengen erforderte eine Neuerstellung von Meßwertaufnahme und Berechnung, das Programm wurde mit einer römischen III verziert.

Die aus ATMA II übernommenen Programmteile waren in PC-Forth-Plus 3.2 geschrieben, einem weitverbreiteten kommerziellen Forth-System von Laboratories Microsystems Inc.. Sie umfaßten im wesentlichen Pulldown-Menüs, Mausbedienung und Verwaltungsaufgaben. Für die Erstellung neuer Programmteile kam das inzwischen verbreitete F-PC von Tom Zimmer in der Version 3.54 (später 3.56) zur Anwendung. Es bietet mehr Komfort. Erwähnt sei nur die VIEW-Funktion, die das Betrachten der Quelltexte aller vorhandenen und selbst entwickelnden Routinen auf Knopfdruck ermöglicht und die beim Vorläufer F83 schon seit fast 10 Jahren zum guten Ton zählt.

In F-PC wurden entwickelt:

- die Datenaufnahme in 8-Kanal-Technik
- die Berechnungsroutinen, die nun auf Fließkomma-Prozessor 80x87 basieren, da der DOS-Speicher für Kalkulationstabellen nicht mehr ausreichte
- die grafische Darstellung, die nach Beschleunigung verlangte.

Externe Meßapparatur.

Eine Spule um das Hüftgelenk versorgt den Meßsender in der Prothese mit Energie. Der Patient wird während der Untersuchung mit einer Videokamera gefilmt und die Meßsignale werden mit aufgezeichnet. Die drei Komponenten der räumlichen Gelenkkraft können unmittelbar während der Untersuchung auf dem Rechnermonitor kontrolliert oder nachträglich analysiert werden.

Die Portierung der Quellen vom 32-bit PC-Forth ins F-PC mit seiner PC-typisch verkorksten Segmentadressierung schien wenig erfolgversprechend. Daher werden beide Teile als separate EXE-Dateien erstellt und über ein dafür entwickeltes Mixed-Language-Interface gekoppelt. Es erlaubt Aufrufe der F-PC-Routinen von PC-Forth-plus, Assembler und Microsofts Quickbasic aus.

Datenaufnahme

Die von der Prothese ausgesendeten (oder vom Videorekorder kommenden) Signale sind pulsintervallmoduliert. Von einer im Biomechanik-Labor entwickelten PC-Einsteckkarte namens DACTRA [1] bzw. ihrem Nachfolgemodell PIM wird die Zeitdauer zwischen zwei Impulsen in einen 2-byte-Integerwert umgesetzt. Jeder Wert korrespondiert zu dem Signal eines Dehnungsmeßstreifens. Bei Eintreffen eines neuen Impulses - etwa alle 400 Mikrosekunden - wird ein Interrupt ausgelöst.

Die Interruptroutine liest das anliegende Signal ein. Anhand der Signalgröße erkennt sie mögliche Übertragungsfehler und stellt die seriell eingehenden Signale der verschiedenen DMS zu einem zusammenhängenden Datensatz zusammen. Eine zweite Interrupt-Routine, die 18,2 mal in der Sekunde von der Uhr des PC angestoßen wird, dient der Erkennung längerer Pausen im Datenfluß. Die nachgeordnete Software kann also problemlos zwischen korrekt eingegangenen Datensätzen, fehlerhaften Sätzen und einem Ausbleiben des Datenflusses unterscheiden. Eine Korrektur von Übertragungsfehlern wäre denkbar, ebenso die zeitsynchrone Abspeicherung weiterer Ereignisse im gleichen Puffer.

Die Speicherung erfolgt wegen der großen Datenmenge im Extended Memory. Aus Zeitgründen wird das Extended Memory im Real-Mode des 80386 angesprochen. Der dazu nötige Trick wurde in [2] beschrieben. Bisher zeigten sich keine Inkompatibilitäten mit anderen Treibern mit Ausnahme des Expanded-Memory-Managers EMM386.

Einige wenige Forth-Worte oberhalb der Interrupt-Ebene erlauben den einfachen Zugriff auf die eingegangenen Datenzyklen.

Die Interrupt-Routine ist im Assembler des F-PC geschrieben. Der mitgelieferte Debugger für Assemblercode (CODEBUG) vereinfachte die Fehlersuche sehr.

Berechnungen

Aus den aufgezeichneten Datensätzen mit den Signalen aller DMS werden die an der Prothese wirksamen Kräfte ermittelt und grafisch dargestellt. Hierzu sind folgende Berechnungsschritte nötig:

- Mittelwertbildung aus mehreren Datensätzen (wenn gewünscht)
- Temperaturkompensation mittels Polynom (s. Kasten "Kraftmessung..")
- Linearisierung mittels Polynom (s. Kasten)
- Matrixmultiplikation (s. Kasten)
- Transformation der Kräfte in Bildschirmkoordinaten.

Bis auf die Mittelwertbildungbildung werden alle Schritte (etwa 200 Fließkommaoperationen pro Datensatz) in 80-bit-Fließkommadarstellung auf dem 80x87-Koprozessor durchgeführt. Es liegen gemäßigte Echtzeitanforderungen vor. Von den im Abstand von etwa 3 bis 5 Millisekunden eingehenden Datensätzen brauchen nämlich nicht unbedingt alle am Bildschirm dargestellt werden. Bei einem PC mit 80386/33 MHz wird etwa jeder dritte Datensatz zur Bildschirmdarstellung herangezogen, was bei der Betrachtung im Echtzeitmodus bereits sehr hoch aufgelöste Kurvenzüge ergibt. Zum genaueren Betrachten kann das Bild im Auswertemodus (OFF-Line) festgehalten werden, hierbei werden alle Datensätze zur Darstellung herangezogen. Bei Wunsch kann im Auswertemodus eine Durchschnittsbildung über mehrere Datensätze erfolgen. Sie gleicht Fehler der Meßkette aus und beschleunigt die Ausgabe beim Zoomen. Eine weitere Beschleunigung der Ausgabe erreicht man mit der Option, nur ausgewählte Stützstellen (z.B. jede dritte Pixelspalte) für die Darstellung zu berücksichtigen.

Die Transformation in Bildschirmkoordinaten erfordert eine Addition und eine Multiplikation je darzustellender Kraftkomponente. Auf die üblichen Overflow-Prüfungen konnte nach der Umstellung auf Floating-Point-Arithmetik erfreulicherweise verzichtet werden, solange von Benutzerseite keine völlig unsinnigen Darstellungsbereiche gewählt werden. Das Abschneiden der Kurvenzüge am Bildrand erfolgt aus Geschwindigkeitsgründen punktweise. Dadurch werden Linien, die den Rand schneiden, etwas zu flach dargestellt, was bei hinreichend dichten Stützstellen nicht sichtbar ist. Linien, die komplett außerhalb des Bildbereichs liegen, werden - ähnlich wie beim Oszilloskop - auf den Bildrand projiziert, was die Wahl des richtigen Maßstabs erleichtert.

Auf die einfache Konfigurierbarkeit wurde unter ATMA III viel Wert gelegt. Die Konfigurierung erfolgt über Pulldownmenüs und Eingabemasken. Alle gewählten Einstellungen können auf Platte gesichert werden, sodaß schnell zwischen verschiedenen Meßaufgaben umgeschaltet

Meß-Endoprothese.

Eine Prothese (Titan mit Keramik-Kugel wurde abgewandelt. Im Prothesenhals wurden drei Dehnungssensoren angebracht (Halbleiter-DMS). Eine induktiv gespeiste Telemetrieschaltung erfaßt die Dehnungssignale und die Prothesentemperatur und sendet sie nach außen. Die Elektronik ist hermetisch eingeschweißt. Durch hohe Prüfbelastungen, Berechnungen und biologische Tests wurde die Sicherheit des Implantats nachgewiesen.

werden kann.

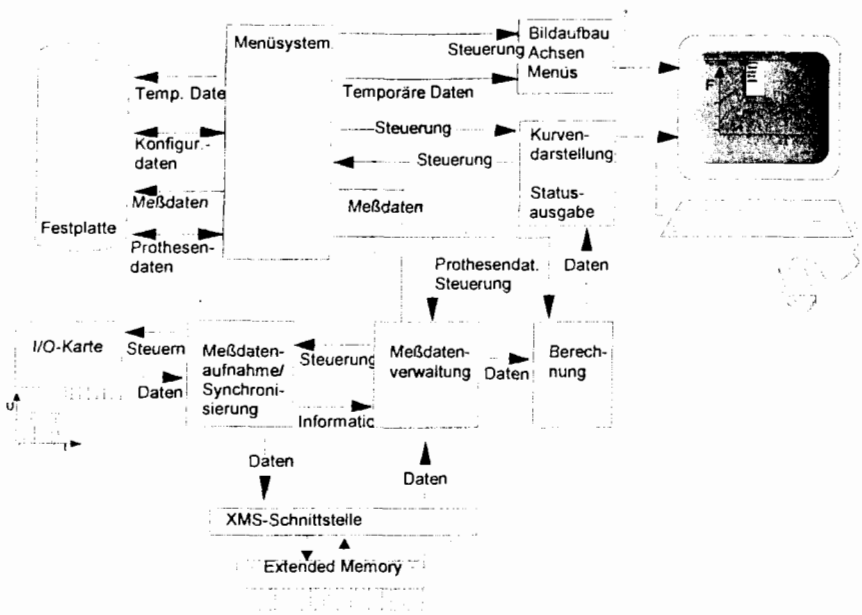
Zeitverhalten und Optimierung

Insgesamt sind bei 8 Kanälen ca. 200 Floatingpointoperationen je Meßzyklus durchzuführen, wobei alle 3 bis 5 Millisekunden ein neuer Zyklus eintrifft. Um den Zeitbedarf zu reduzieren, wurden Polynomberechnung und Vektormultiplikation in 80x87-Assembler codiert. Es stellte sich heraus, daß bei Assemblerprogrammierung gegenüber Forth-Routinen, die auf dem schnellen Floating-Point-Paket des F-PC (FFLOAT.SEQ) aufsetzen, der Zeitgewinn nur etwa einen Faktor 2 bis 4 erreicht. Der 80x87 arbeitet ebenso wie die Sprache Forth stackorientiert. Das schnelle Fließkomma-Paket hält nun die unteren acht Elemente des Floating-Point-Stacks in den 80x87-Registern und reduziert dadurch die Anzahl der nötigen Speicherzugriffe. Dies erzeugt sehr schnellen Code, da der Zeitbedarf eines 80x87-Speicherzugriffs in der gleichen Größenordnung wie der einer Addition oder Multiplikation liegt.

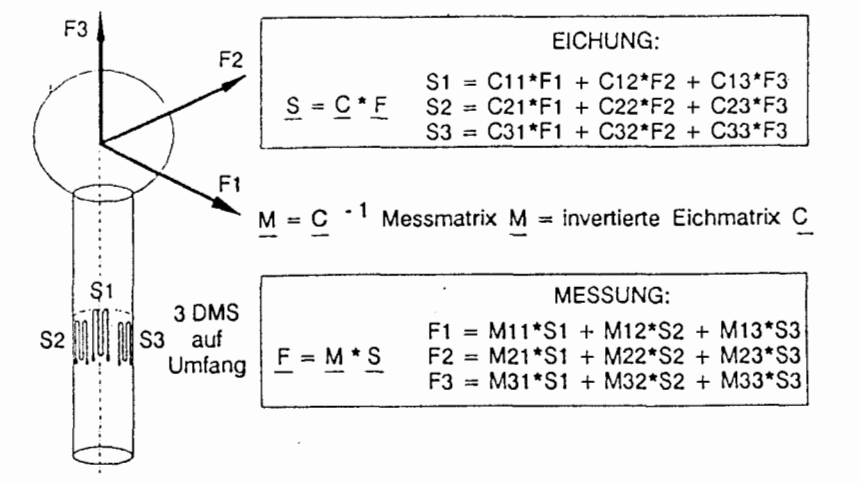
Messungen ergaben, daß der zeitliche Flaschenhals weniger in der Berechnung als vielmehr in der grafischen Darstellung der Ergebnisse liegt. Das Verhältnis beträgt etwa 1 : 10. Interessant erscheint auch der Geschwindigkeitsvergleich verschiedener Grafikbefehle untereinander. So verbraucht unter PC-Forth-Plus 3.2 das einmalige Verändern der Hintergrundfarbe mehr Zeit als das Plotten von 8 Punkten mit 8-maligem Setzen der Vordergrundfarbe. Ein Vergleich zwischen PC-Forth-Plus 3.2 und F-PC 3.54 ergab einen deutlichen Vorsprung der schnelleren F-PC-Plotroutinen. Das Setzen eines Punktes in einer jeweils anderen Farbe ist hier mehr als 10mal so schnell.

Schluß

Das Vorurteil, Forth sei nur für kleine Programme geeignet, läßt sich nicht aufrechterhalten. Die vielen kleinen Routinen behindern manch-



Programmaufbau: Die Meßdatenerfassung ATMA besteht aus zwei lauffähigen Programmen, die im Bild unterschiedlich unterlegt sind. Der untere (hardwarenahe) Teil ist in Forth-PC 3.56 geschrieben. Er legt die von der Hardware eingehenden Signale im Extended Memory ein, stellt die notwendigen Verwaltungs- und Berechnungsfunktionen zur Verfügung und zeichnet die Kurvenzüge. Der höhere Teil ist in PC-Forth plus 3.2 erstellt und enthält neben der Benutzungsschnittstelle mit Pulldown-Menüs und Mausbenutzung diverse Verwaltungsroutinen, die Schnittstelle zum Massenspeicher und die Routinen zum Bildschirmaufbau. Die beiden Teile kommunizieren über eine spezielle Schnittstelle miteinander, die im Bild nicht dargestellt ist.



Matrixaufnehmer für drei Kraftkomponenten.
Zur Messung von Kräften werden fast ausschließlich Dehnungsmeßstreifen (DMS) verwendet. Sie werden an entsprechender Stelle aufgeklebt. Wird nun das Trägermaterial infolge Krafteinwirkung gedehnt, reagieren die DMS mit einer meßbaren Widerstandsänderung, die meist proportional zur eingeleiteten Kraft ist. Kraftmeßeinrichtungen enthalten i.a. je zu messender Kraftrichtung zwei oder vier exakt ausgerichtete DMS, die in einer Halb- oder Vollbrücke verschaltet sind. Hierdurch kann die Genauigkeit und Empfindlichkeit erhöht und der Temperatureinfluß vermindert werden.

Bei Verwendung der Matrixmethode nach Bergmann ist nur ein DMS je zu messender Kraftkomponente erforderlich. Die DMS brauchen nicht exakt ausgerichtet werden, es reicht, wenn ihre Richtungsvektoren linear unabhängig sind. Um beispielsweise die Größe der in horizontaler Richtung angreifenden Kraft zu ermitteln, werden die Signale der drei DMS mit unterschiedlichen Gewichten multipliziert und addiert. Mathematisch entspricht dies einer Matrixmultiplikation des Signalvektors mit der Gewichtungsmatrix.

Die Matrixmethode eignet sich für die Kraftmessung in Prothesen besonders gut, da sie mit wenigen DMS auskommt und ein exaktes Ausrichten der DMS in der etwa 8 mm durchmessenden Bohrung nur schwer zu bewerkstelligen ist. Allerdings wird eine aufwendigere Berechnung erforderlich. Vor der Matrixmultiplikation wird die Nichtlinearität der DMS und ihre Temperaturabhängigkeit rechnerisch kompensiert. Die Temperaturkompensation bedingt zusätzlich einen zusätzlichen Temperatursensor.

mal den Überblick, aber die komfortablen Möglichkeiten des Entwicklungssystems gleichen dieses Manko aus. Der Vorteil des schnellen interaktiven Testens bleibt auch bei großen Programmen bestehen, allerdings erfordert eine 'schnell mal' durchgeführte Änderung einer tiefliegenden Routine eines 10000-Zeilenprogramms eine Gedenkminute für den Compiler. Hier können Borland und Microsoft-Sprachen inzwischen mithalten.

Das zumindest unter Forth-Programmierern verbreitete Urteil, Echtzeitberechnungen müßten aus Zeitgründen in Ganzzahlarithmetik erfolgen, kann bei gemäßigten Echtzeitanforderungen im unteren Millisekundenbereich in Frage gestellt werden. Die Hardware hat sich so weit entwickelt, daß Fließkomma-Operationen durchaus attraktiv geworden sind.

Die Verwendung einer Programmierumgebung aus dem Public-Domain-Bereich (Tom Zimmers F-PC, s. [4]) hat sich trotz aller Unkenrufe als sinnvoll erwiesen. Ungereimtheiten und Fehler, wie es sie in jedem Compiler gibt, brauchen in Forth nicht durch 'Drum-Rum-Programmieren' umgangen werden. Einfacher ist oft eine Änderung am Forth-System, die sich beim F-PC - dank Lieferung mit komplettem Quelltext - recht schnell bewerkstelligen läßt. Es waren nur geringe Änderungen erforderlich, ähnlich viele wie beim kommerziellen PC-Forth-Plus 3.2, das allerdings ohne Quelltexte geliefert wird und dazu erst dekompiliert werden mußte.

Das Mixed-Language-Interface zwischen PC-Forth-Plus und Tom Zimmers F-PC läuft stabil. Eine Schnittstelle zum Aufruf von F-PC aus Microsoft-Sprachen heraus wurde erstellt, aber mangels Anwendungsfall bisher kaum getestet.

Einige allgemein verwendbare Teile des hier beschriebenen Programms sind frei erhältlich. Dies betrifft insbesondere das Mixed-Language-Interface, die Schnittstelle zum Extended-Memory und die Nutzung von Sonderzeichen im F-PC. Sie sind zusammen mit dem neuesten F-PC für etwa 100.-DM beim Forth-Vertrieb Jörg Staben, Hagelkreuzstr.23, 40721 Hilden zu beziehen, der auch Kontakte zur Forth-Gesellschaft e.V. vermittelt.

❑

Literatur

[1] Friedmar Graichen: Implantierbares telemetrisches Übertragungssystem zur In-vivo-Messung der Belastung künstlicher Hüftgelenke. Berlin: Schiele und Schön, 1990. (Biomedizinische Technik; Bd.3)

[2] Harald Albrecht: Grenzenlos - vier Gigabyte im Real-Mode des 80386 adressieren. In: c't 1/90, Seite 212 ff

[3] Georg Bergmann: Ein Dreikomponenten-Kraftaufnehmer und seine Anwendung für die Messung von Gelenkkräften am Tier. Dissertation, TU Berlin 1981.

Kraftmessung mit der Matrixmethode

Die instrumentierte Prothese enthält drei Dehnungsmeßstreifen und einen Temperatursensor. Aus diesen vier Signalen gilt es, die drei Kraftkomponenten in den drei Raumrichtungen zu berechnen. Die Berechnungsschritte sind Kompensation des Temperatureinflusses der DMS, Kompensation der Nichtlinearität der DMS, und eine Matrixmultiplikation.

Die Temperaturkompensation erfolgt mittels eines Polynoms zweiten Grades. Aus den DMS-Signalen Z_i und dem Signal R des Temperatursensors ergeben sich die korrigierten Signale Z'_i. Die Kompensation erfolgt für alle Kanäle gleich, die Eichkonstanten u_0 bis u_3 und R_0 sind kanalunabhängig.

$$Z'_i = (u_0 + u_1 Z_i + u_2 Z_i^2 + u_3 Z_i^3) \cdot (R - R_0) + Z_i$$

Die Kompensation der Nichtlinearitäten erfolgt ebenfalls mittels Polynom. Aus den kompensierten Signalen Z'_i ergeben sich die linearisierten Signale S_i. Allerdings sind die Linearisierungskoeffizienten $k_{i,0}$ bis $k_{i,-3}$ für alle DMS verschieden,. Diese Kanalabhängigkeit wird durch den Index i angedeutet.

$$S_i = k_{i,0} + k_{i,-1} Z'^{-1}_i + k_{i,-2} Z'^{-2}_i + k_{i,-3} Z'^{-3}_i$$

Polynome bis zum sechsten Grade sind programmseitig möglich, werden allerdings selten benutzt, da sie zum "Überschwingen" neigen.

Die Multiplikation mit der Matrix $M_{i,j}$ erzeugt aus den linearisierten Signalen Si die Kräfte F_x, F_y, F_z, in den drei Raumrichtungen.

$$\begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} = \begin{bmatrix} M_{11} & M_{12} & M_{13} \\ M_{21} & M_{22} & M_{23} \\ M_{31} & M_{32} & M_{33} \end{bmatrix} \cdot \begin{bmatrix} S_1 \\ S_2 \\ S_3 \end{bmatrix}$$

Die Koeffizienten der Matrix M werden bei der Kalibrierung zweckmäßigerweise in Bezug zum Prothesenhals ermittelt. Für die Interpretation der Ergebnisse ist ein anderes, auf den Körper bezogenes Koordinatensystem günstiger, das in allen drei Raumrichtungen gedreht ist. Die entsprechenden Winkel, um die dieses System gegenüber dem Prothesensystem gedreht ist heißen beim Hüftgelenk: Halswinkel h, Antetorsionswinkel a und Femurwinkel f. Die Drehung erfordert eine körperbezogene Meßmatrix M', die durch Multiplikation der Kalibrierungsmatrix mit den drei Tranformationsmatritzen entsteht und die Matrix M in o.g. Gleichung ersetzt:

$$M' = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(f) & -sin(f) \\ 0 & sin(f) & cos(f) \end{bmatrix} \begin{bmatrix} cos(a) & -sin(a) & 0 \\ sin(a) & cos(a) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} cos(h) & 0 & sin(h) \\ 0 & 1 & 0 \\ -sin(h) & 0 & cos(h) \end{bmatrix} \cdot M$$

Die Matrixmethode erfordert gegenüber der üblichen Halb- oder Vollbrückenschaltung nur einen Dehnungsmeßstreifen je Kraftkomponente. Eine exakte Ausrichtung ist nicht erforderlich, solange die Richtungsvektoren der DMS linear unabhängig bleiben. Bei kleinen und schlecht zugänglichen Meßobjekten rechtfertigt dies den zusätzlichen Aufwand für die Berechnung und die Ermittlung der nötigen Parameter durch Kalibrierung.

PreisRätsel aus VDI-nachrichten-magazin 1994 no.01 p.30 **Arndt Klingelnberg**

Die Original-Aufgabenstellung:

Hobby-Programmierer, aufgepaßt!

Sollen in einem Programm die Werte von zwei Variablen vertauscht werden, so wird der Tausch standardmäßig mit einer Hilfsvariablen h als Dreieckstausch vorgenommen:

```
h := a;
a := b;
b := h;
```

Aufgabenstellung/Frage:

Ist es möglich, den Austausch der Werte ohne eine Hilfsvariable durchzuführen? Wie könnte das bewerkstelligt werden?

Eine Antwort wäre:

Statt der Hilfvariablen wird/werden ein oder auch zwei Processor-interne(s) Register verwendet. Auch gibt es für 8086 den ASM-Befehl XCHG der einen Austausch vornimmt, allerdings muß in diesem Fall auch mindestens einer der Werte vorher in den Processor (in ein Register) geladen werden. Eine Hilfsvariable ist dabei NICHT notwendig.

Aber diese recht trivialen Lösungen sind wohl nicht gemeint, obwohl sie eigentlich der unpräzisen (!) Aufgabenstellung entsprechen.

Daher ein Versuch der exakteren Formulierung der Aufgabe und für Forther:

Statt auf Variable, Speicherstellen bzw. den Stack wird die Aufgabe auf Processor-interne Register bezogen: Es sind die Werte in zwei Registern auszutauschen ohne ein drittes Register (oder etwa auch ein ein-BIT-breites Register, wie z.B. im Processor-Flag) zu nutzen, oder den möglicherweise vorhandenen XCHG-Befehl oder ähnliches zu nutzen.

Gehaltvolles — zusammengestellt von Klaus-Peter Schleisiek

Inhaltsübersicht: Forth Dimensions der Forth Interest Group, USA

4, Vol.15 Nov/Dez 1993

5, Vol.15 Jan/Feb 1994

6, Vol.15 März/April 1994

Der I²C-Bus

in Forth realisiert

von Wolfgang Schemmert
Strahlenberger Str. 123, 63067 Offenbach, Tel.: 069-8001208

Wer umfangreichere Meß- und Steuereinrichtungen entwickelt, kennt diese Situation: eine zusätzliche Anzeige ist erforderlich oder es fehlen Eingabeeinheiten. Hier bietet sich der I²C Bus an, denn mit nur vier Verbindungsleitungen (2 Bus, 2 Stromversorgung) läßt sich immer noch eine Zusatzhardware irgendwo in einer Ecke oder an der Frontplatte anbringen. Dieser Artikel beschreibt die Entwicklung einer Treiber-Software für den I²C-Bus und leistet damit eine kleine Entwicklungshilfe.

Was ist ein I²C-Bus, wie funktioniert er?

Der I²C-Bus ("Inter-IC-Bus") wurde ursprünglich entwickelt, um die digitalisierten Steuer- und Kontrollfunktionen in Fernsehern und anderen Konsumer-Geräten einfach zu vernetzen. Inzwischen hat er sich auch zu einem beliebten Inter-Modul-Bus in der professionellen Meßtechnik gemausert. Es gibt eine Palette nützlicher Chips für Meß- und Steueranwendungen, die bereits ein I²C-Interface eingebaut haben. Für größere Entfernungen, etwa weiter als 3 bis 4 m, ist der I²C-Bus wegen seiner relativ hochohmigen Open-Collector-Technik allerdings nicht so geeignet. Außerdem braucht er - im Gegensatz zu "richtigen" Industriebussen - zwei Signalleitungen: eine serielle Datenleitung SDA und eine Clockleitung SCL. (Eine Zusammenfassung der I²C-Slang-Begriffe finden Sie im Kasten "Der I²C-Slang".)

Den Philips-Entwicklern ist bei der Definition des I²C-Protokolls eine sehr raffinierte Lösung gelungen, um komplexe bidirektionale Datenübertragungen mit sehr geringem Hardware- und Verwaltungsaufwand auf die beiden Datenleitungen zu packen. Der grundsätzliche Hardware-Aufbau erfolgt wie in Bild 1.

Von allen theoretisch möglichen Konfigurationen des I²C-Bus ist nur eine praktisch interessant und wird auch im folgenden ausschließlich betrachtet: nämlich die Ankoppelung von Slave-Peripherie-Modulen an einen als Single-Master wirkenden Mikrocontroller. Der Mikrocontroller alleine diktiert dann den SCL-Takt und sagt den einzelnen Slave-Bausteinen, wann sie Daten empfangen oder auf den Bus senden sollen.

Zwar ist z.B. auch der CAN-Bus oder Bitbus sehr einfach zu programmieren, aber nur wenn man einen entsprechend hochintegrierten Controller einsetzt. Der I²C-Bus hingegen läßt sich im Single-Master-Betrieb ohne weiteres "per Hand" mit normalen Mikroprozessor-Ports ansteuern.

Es ist vorteilhaft, wenn diese nicht als Gegentaktausgänge ausgeführt sind, sondern in Open-Collector Technik oder mit n-Kanal Transistoren und hochohmigen Lastwiderständen, wie z.B. beim 8051.

Bei Prozessoren, deren Ports über Datenrichtungs-Register verfügen, lassen sich Open Collector Ausgänge folgendermaßen emulieren: Zuerst schreibt man in das Ausgangs-Datenregister eine 0 für das betreffende Bit. Die Umschaltung zwischen HIGH und LOW wird nun allein mit dem Datenrichtungs-Register vorgenommen. Für HIGH am Ausgang wird das Register auf INPUT geschaltet, für LOW am Ausgang wird es auf OUTPUT geschaltet. Da beim INPUT üblicherweise nicht das Daten-Register, sondern der physikalische Eingangspegel gelesen wird, haben wir zugleich Sender und Empfänger auf einer einzigen Portleitung realisiert.

Forth-Programmierung des I²C-Bus

In jedem I²C-Slave-Chip ist herstellerseitig ein 4 Bit langer Baustein-Typ-Identifier eingebrannt, der manchmal eine ganze Bausteinfamilie beschreibt. (Schließlich kann der I²C-Bus ja nur 16 unterschiedliche Typen unterscheiden.) So haben alle seriellen Speicher den Identifier $A, obwohl es x verschiedene Einzelausführungen gibt. Je nach Bausteintyp sind 0 bis 3 zusätzliche Adressleitungen aus dem Gehäuse herausgeführt, die hardwaremäßig auf 0 oder 1 gelegt werden und somit die Unterscheidung

Stichworte

I²C-Bus
Papid Prototyping
SPS (speicherprogrammierbare Steuerung)
serielles EEPROM
8051
H8/325
6511 (Mini-Bee, RSC-Forth)

Hardware-Aufbau des I²C-Bus

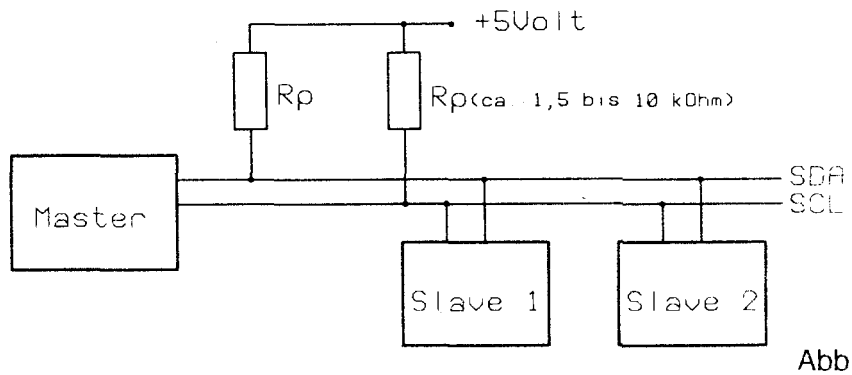


Abb. 1

einer gewissen Anzahl Bausteine mit gleichem Identifier an einem Bus gestatten.

Die komplette Slaveadresse setzt sich also folgendermaßen aus 8 Bit zusammen:

- Bit7 ... Bit 4:
  Typ-Identifier
- Bit3 ... Bit 1:
  =0, oder falls programmierbar vorgesehen: Unterscheidung zwischen mehreren gleichen Chips oder Selektion interner Bereiche des Chips.
- Bit 0:
  = 0 wenn der Master Daten schreiben will
  = 1 wenn der Master Daten lesen will. Mit dem untersten Bit der Slave-Adresse definiert der Master also die Datenrichtung.

Bei den Slave-Chips wird bauartbedingt unterschieden zwischen Modellen mit Subadressen und einfacheren ohne Subadressen. Mit der Subadresse werden die verschiedenen Unterfunktions-Register angesprochen (bei Speicherchips die einzelnen Speicherzellen).

Wenn die Adressierung erfolgreich beendet ist, folgen eines oder mehrere Datenbytes.

Aber da kommen wir nun Schritt für Schritt hin:

Leider ist der I²C-Bus für die Forth-Programmierung ein wenig zeitkritisch: Der I²C-Bus verträgt maximal Taktfrequenzen bis 100 kHz, also Periodendauern bis 10 Mikrosekunden herunter. In reiner Forth-Programmierung brauchen 8-Bit Mikrocontroller aber etwa 50 bis zu 500 Mikrosekunden für einen solchen Zyklus. Daher sollten die elementaren Worte eines I²C-Pakets in Maschinencode formuliert werden: sie werden pro Übertragung eines Bytes mehrere Dutzend mal aufgerufen.

Zuerst wird im Programmbeispiel ein Codemodell für die 8051-Familie vorgestellt, wobei SCL auf P1.6 und SDA auf P1.7 gelegt wird. Danach wird der maschinenabhängige Teil für den H8/325 und den 6511 (Mini-Bee, RSC-Forth) angepaßt. Leider stand für den 8051 und H8/325 gerade kein Forth Assembler zur Verfügung, deshalb wurden diese Worte direkt in binärem Maschinencode kompiliert.

Um den Programmcode übersichtlicher darzustellen, wurde er komplett in einem extra Kasten abgedruckt. Der maschinenabhängige Teil wird am Ende des Kastens auf H8/325 und 6511-Prozessoren portiert. Grundsätzlich wird HEX als Zahlenbasis angenommen.

Die Programmierung

Mit zwei Leitungen kann das Bussystem vier Zustände annehmen:

- dataHI_clockHI
- dataLO_clockHI
- clockLO_dataHI
- clockLO_dataLO

Zum Datenempfang muß nur der Zustand der SDA-Leitung kontrolliert werden, der Master muß sich selber natürlich vorher auf HIGH = Empfang schalten. Dies alles wird mit dem Wort `dataRECEIVE` ausgeführt.

Nach dem Einschalten sind SDA und SCL bei allen I²C-Bausteinen zunächst hochohmig (HIGH), mit dem gleichen Portzustand muß auch der Master-Controller in den Datenverkehr einsteigen. Eröffnet wird die Partie vom Master mit der "STARTBEDINGUNG", zuerst geht SDA auf LOW danach SCL (Bild 2).

Obwohl wir sie erst zum Schluß brauchen, formulieren wir spiegelbildlich die "STOPBEDINGUNG", mit der jede Datenübertragung abgeschlossen wird und das I²C-Interface wieder für die nächste Startbedingung "scharf" gemacht wird: zuerst SCL auf HIGH, danach SDA. Es ist schon beinahe verwunderlich, daß diese Bedingungen nicht während des seriellen Datenverkehrs mal "zufällig" auftreten, aber das I²C-Protokoll ist konfliktfrei durchgestylt.

Wie bereits angedeutet, beginnt der Master den eigentlichen Akt der Datenübertragung mit der Aussendung einer Slave-Adresse. Meistens wird innerhalb eines Teilprogramms nur mit einem einzigen Slave kommuniziert. Deshalb vereinfacht es die Stackverwaltung, wenn SLAVE-ADRESSE als Variable angelegt ist.

Die Definition von Start und Stop

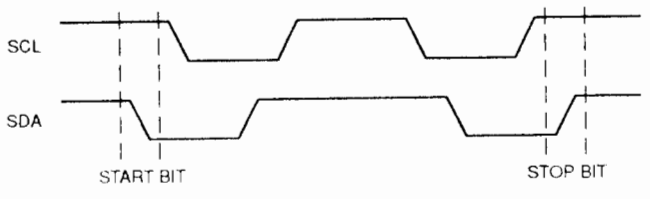


Abb. 2

Ein Byte wird verschickt

Zuerst wollen wir ein Datenyte zum Slave-Chip senden. Bit 0 der Slave-Adresse ist dann immer 0. Sowohl das Adress- als auch das Datenbyte muß nun unter Clock-Steuerung seriell auf SDA übertragen werden, und zwar mit dem MSB zuerst. Für jedes Bit wird erst der Datenpegel auf SDA eingestellt, dann macht SCL einen positiven Impuls. In der positiven Flanke von SCL übernimmt der Slave-Baustein das Bit. Erst wenn SCL wieder LOW ist, wird der nächste Datenpegel auf SDA eingestellt usw. Der Pegel des jeweils zu sendenden Bits wird festgestellt durch Links-Rotieren bzw.-Schieben des zu sendenden Bytes und Ausmaskieren.

Das wird mit dem bereits in Forth High-Level programmierten Wort `sendBIT` ausgeführt. `CROL` ist darin mein spezielles Forth-Wort zum effektiven Einsatz des 8051-Befehlssatzes für 8-Bit Rotation. Rotiert wird nur das Low-Byte. Das durchrotierte Bit wird jeweils in Bit 8 eingetragen. Genausogut tuts ohne sonstige Änderungen auch jedes 16-Bit LSHIFT Wort oder notfalls eine Multiplikation mal 2, die natürlich sehr viel Rechenzeit frißt..

Das komplette Byte wird seriell mit `SENDI2C` übertragen. Nach dem Empfang des achten Bit quittiert der Empfänger unter Aussendung des ACKNOWLEDGE-Bit: Das SDA des sendenden Bausteins geht auf HIGH (=hochohmig, = passiv) und SCL geht ebenfalls auf HIGH. Wenn der empfangende Baustein die Meldung ver-

standen hat, zieht er seinerseits die SDA-Leitung nach LOW. Damit wird der Sender aufgefordert, weitere Daten zu übermitteln. Der Sender (=Master in unserem Beispiel) wird anschließend das Datenbyte ebenfalls mit SENDI2C übertragen und am Ende wieder auf das Achnowledge-Bit achten. Wenn das ok ist, folgt die Stopbedingung.

Im Wort SLAVE-ACK wird der Acknowledge Status des Slave überprüft und im Fehlerfalle mit ABORT die Übertragung komplett abgebrochen.

Eine Variante davon ist das Wort SLAVE-ACK>FLAG, das nicht abbricht sondern das Acknowledge-Ergebnis als Flag auf dem TOS weitergibt. Eine Anwendung hierfür folgt an Ende des Artikels.

WRITE schließlich ist das komplette Forth-Wort zum Byte-Schreiben.

Ein Byte wird gelesen

Im Gegenzug soll nun ein Byte aus einem Slave-Baustein gelesen werden.

Genau wie bei der Datensendung wird zuerst der Slave adressiert. Das Bit 0 der Slave-Adresse muß aber für eine Lese-Anfrage = 1 sein. Für den Empfang eines Bit ist das Wort recvBIT konstruiert, das in recvI2C 8 mal angewendet wird.

Das Wort READ endlich holt ein komplettes Byte aus dem Slave-Chip.

Als Beispiel: Minimal-SPS

Als konkretes Beispiel für die Anwendung der I²C-Bus-Worte verweilen wir bei der Ansteuerung für einen Peripherie-Erweiterungsbaustein.

Der PCF8574 hat 8 I/O-Portleitungen, die in der Schaltungstechnik ungefähr den Ports der 8051-Mikrocontroller-Familie entsprechen. Wie jene braucht der PCF8574 kein Datenrichtungs-Register; wenn ein Pin als Eingang arbeiten soll, muß er zuvor "HIGH" beschrieben sein. Ausgelesen wird immer sein physikalischer Pegel.

Daher hat der PCF8574 auch nur

Listing zu: Der I²C-Bus (W. Schemmert) **i2c.seq**

```
\ Programmpaket I2C Bus fuer 8051 Family
\ Angenommen, es besteht die Hardware Konfiguration :  SDA - P1.7
\                                                      SCL - P1.6

HEX

VARIABLE SLAVE-ADRESSE

CODE dataHI_clockHI
        D2 c, 97 c,       \ SDA bei P1.7, Bit 897
        D2 c, 96 c,       \ SCL bei P1.6, Bit 896
END-CODE

CODE dataLO_clockHI
        C2 c, 97 c,
        D2 c, 96 c,       \ clockHI muß nach Änderung von data kommen !
END-CODE

CODE clockLO_dataHI
        C2 c, 96 c,       \ clockLO muß vor Änderung von data kommen !
        D2 c, 97 c,
END-CODE

CODE clockLO_dataLO
        C2 c, 96 c,
        C2 c, 97 c,
END-CODE

: dataRECEIVE   ( -- b )
\ Testet den extern eingepraegten Pegel der SDA-Leitung und gibt Flag aus
        clockLO_dataHI
        dataHI_clockHI  \ SDA und SCL HIGH
        BIT@
        8. AND
        clockLO_dataHI  \ SDA LOW, SCL HIGH
;


\  ENDE DER MASCHINENABHAENGIGEN TEILE ********************************

: TXHI  ( -- )
\ SCL positiver Impuls mit SDA = HIGH
          clockLO_dataHI    \ SCL LOW, SDA HIGH
          dataHI_clockHI    \ SDA und SCL HIGH
          clockLO_dataHI    \ SCL LOW, SDA HIGH
;

: TXLO  ( -- )
\ SCL positiver Impuls mit SDA   LOW  !
          clockLO_dataLO    \ SDA und SCL LOW
          dataLO_clockHI    \ SDA LOW, SCL HIGH
          clockLO_dataLO    \ SCL LOW, SDA HIGH
;

: STARTBEDINGUNG   ( -- )
\ Startbedingung definiert als neg SDA Flanke waehrend SCL = H
          dataHI clockHI    \ SDA und SCL HIGH
          dataLO clockHI    \ SDA LOW, SCL HIGH
          clockLO_dataLO    \ SDA und SCL LOW
;

: STOPBEDINGUNG    ( -- )
\ Stopbedingung definiert als pos SDA Flanke waehrend SCL = H
          dataLO clockHI    \ SDA LOW, SCL HIGH
          dataHI clockHI    \ SDA und SCL HIGH
;

: sendBIT ( b -- b rot )
\ Master sendet 1 Bit auf SDA und produziert die SCL-Impulse dazu.
\ links rotiert : MSB zuerst gesendet
        CROL  DUP 100 AND  \ statt CROL funktioniert auch Forth-83   100 AND
        IF      TXHI
        ELSE    TXLO
        THEN
;
```

ein einziges Datenregister, er ist also ohne Slave-Adresse sehr einfach am I²C-Bus zu programmieren.

Der Baustein ist sehr praktisch, um z.B. mit geringem Verdrahtungsaufwand eine abgesetzte Bedieneinheit an der Frontplatte eines Gerätes zu realisieren oder in einem 19"-Rahmen Relais auf anderen Einschüben anzusteuern.

Eine Beispiel-Hardware, die auch als Minimal-SPS in Forth programmiert werden kann, zeigt Bild 3. Die Aufteilung in 4 Eingänge und 4 Ausgänge ist willkürlich, soll aber in diesem Beispiel eingehalten werden. (Dort ist auch das Sockel-Layout des PCF8574 abgebildet)

Relais-Ansteuerung

Bei der Ansteuerung von Relais mit dem PCF8574 sollten zwei Besonderheiten beachtet werden: Alle Ports initialisieren sich beim Einschalten auf HIGH. Relaistreiber wie ULN2003 können wegen der nach Plus hin recht hochohmigen Ports nicht direkt angeschlossen werden. Es muß daher ein HCMOS-Inverter zwischen PCF8574 und Relaistreiber eingefügt werden. Damit ist zugleich das AUS-Schalten der Relais beim Anlegen der Betriebsspannung gewährleistet.

Eine weniger "professionelle", aber für einfache Projekte sich ebenfalls anbietende Methode ist die Ansteuerung der Relais mit diskreten pnp-Transistoren. Damit ist man allerdings auf stromfressende 5V-Relais fixiert. In Bild 4 sind beide Lösungen jeweils für einen Ausgang skizziert.

Bei als Dateneingang verwendeten Ports ist weiter zu bedenken, daß sie konstruktiv bedingt im Leerlauf immer auf HIGH liegen. Bild 4 zeigt alternativ zwei mögliche Eingangsverdrahtungen. Die Variante mit dem Optokoppler ist "professioneller", braucht aber leider eine externe Steuerspannung.

Tasten und Schalter als Eingabe

Für Taster, Schalter oder gar den

Fortsetzung des Listing zu: Der I²C-Bus

```
: SENDI2C ( b -- )       \ Sendet 1 Byte seriell via I2C-Bus
                         \ Bussteuerung ist in sendBIT
        sendBIT
        sendBIT
        sendBIT
        sendBIT
        sendBIT
        sendBIT
        sendBIT
        sendBIT
        DROP
;


: SLAVE-ACK  ( -- )
\ Prueft Acknowledge-Impuls des Slave. Abbruch bei Fehler
        dataRECEIVE
        IF      STOPBEDINGUNG 1 ." I2C:ACK error" ABORT
        THEN
;


: SLAVE-ACK>FLAG  ( -- TIF )
\ wie SLAVE-ACK, aber mit Resultat-Flag auf dem TOS statt Abbruch
        dataRECEIVE
        IF      0       \ FALSE
        ELSE    1       \ TRUE
        THEN
;


: MASTER-ACK  ( -- )
\ Master gibt ACK an Slave
        clockLO_dataLO   \ SDA und SCL LOW
        dataLO_clockHI   \ SDA LOW, SCL HIGH
        clockLO_dataHI   \ SCL LOW, SDA HIGH
;

: WRITE  ( Data -- )
\ schreiben in I2C-Device ohne Subadressen
        STARTBEDINGUNG
```

Anwendungsbeispiel mit I²C-Bus

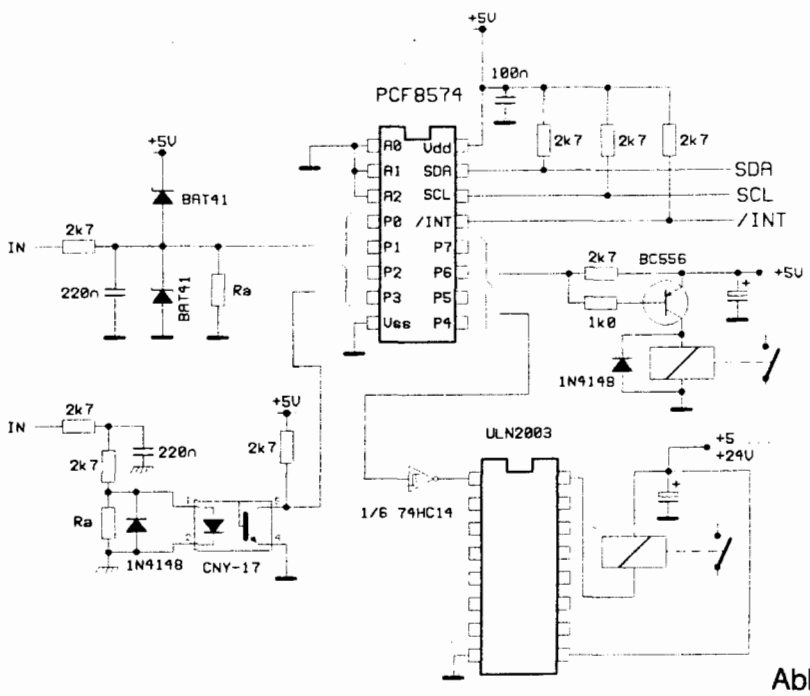


Abb. 3

Anschluß von Lichtschrankenmodulen ist die galvanisch gekoppelte Ausführung angemessen. Bei dieser einfachen Schaltungsvariante ist der Längswiderstand immer ein Kompromiß: er soll einerseits den Porteingang vor Überlast schützen, andererseits muß er klein genug sein, um den Eingang auch wirklich auf LOW zu ziehen.

In beiden Schaltbeispielen dient der Widerstand Ra zur Anpassung des Schaltpunktes an die konkreten Anforderungen. Beide Schaltungen überleben Eingangsspannungen bis etwa +/- 24 Volt.

Der PCF8574 hat als besonders angenehme Hardware-Eigenschaft (die aber nichts mit dem I²C-Bus zu tun hat) einen separaten Interrupt-Ausgang: Sobald sich der Eingangspegel an einem beliebigen Port ändert, wird die Interrupt-Leitung so lange nach LOW gezogen, bis der betreffende Baustein ausgelesen wurde.

Damit erübrigt sich das recht aufwendige "Pollen" über den seriellen I²C-Bus. Weil die Interrupts in allen Forth-Systemen recht unterschiedlich implementiert sind, sei darauf jetzt aber nicht weiter eingegangen.

Programmierung

Nun endlich zur Programmierung der Minimal-SPS:

Um den typischen Wert des PCF8574 in SLAVE-ADDRESSE einzutragen, definieren wir ein aussagekräftiges Aufrufwort: ($40 ist der Typ-Identifier des PCF8574)

```
: PCF8574 ( b – )
  2 * 40 + SLAVADRESSE !
;
```

(b ist die selektierte Sub-Adresse dieses Bausteins, eingestellt mit den Pins A0 bis A2)

Um beispielsweise alle Relais in der Schaltung nach Bild 1 durchzuschalten, wird das Byte $0F an den Slave gesendet. Das Nibble für die Eingänge ist grundsätzlich auf HIGH zu halten, weil sonst die Port-Treiber das Eingangssignal kurzschließen!

```
: RELAIS_SCHALTEN ( b –)
  \ Schaltet d. Relais wie das obere Nibble von b
   0 PCF8574
   NOT 0F OR WRITE ;
```

Fortsetzung des Listing zu: Der I²C-Bus

```
        SLAVE-ADRESSE @ SENDI2C  \ Slave-Adresse
        SLAVE-ACK
        SENDI2C                  \ Byte
        SLAVE-ACK
        STOPBEDINGUNG
;


: revcBIT ( b - b_rot )
\ Byte wird nach links hereingeschoben und Bit 0 ggf gesetzt
\ Bezogen auf das Endresultat wird MSB zuerst bearbeitet.
        CROL             \ statt CROL funktioniert auch Forth-83  1 LSHIFT
        dataRECEIVE
        1F      1 OR
        THEN
;


: RECVI2C ( - b )
\ empfangt 1 Byte seriell via IIC-Bus; Busssteuerung dazu in revcBIT
        0                \ wird successive zum Ergebnis aufgebaut
        revcBIT
        revcBIT
        revcBIT
        revcBIT
        revcBIT
        revcBIT
        revcBIT
        revcBIT
;

: READ ( - data )
\ Byte lesen aus I2C-Device ohne Subadresse bzw. "current read"
        STARTBEDINGUNG
        SLAVE-ADRESSE @ 1+ SENDI2C
        SLAVE-ACK
        RECVI2C
        STOPBEDINGUNG
;

: RANDOM_WRITE  ( Data Subadresse  – )
\ schreiben in I2C-Device mit Subadressen
```

Zeitverhalten beim Lesen und Schreiben

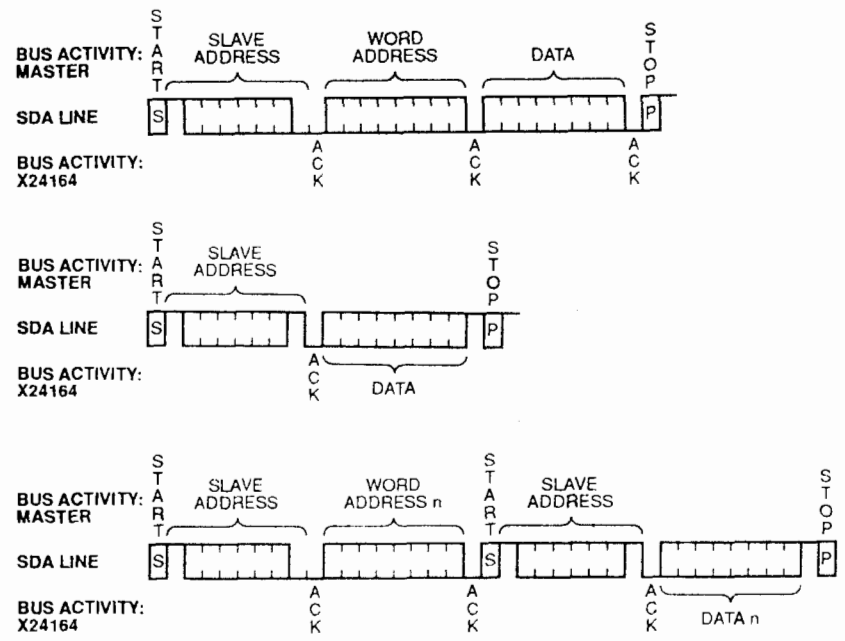


Abb. 4

Zum Lesen der Eingänge definieren wir vorher für jeden Eingang eine eigene Status-Variable. Das ist von Vorteil, wenn die Eingangsabfrage später in komplexere Steueraufgaben eingebunden werden soll.

```
VARIABLE EINGANG1
VARIABLE EINGANG2
VARIABLE EINGANG3
VARIABLE EINGANG4
0 PCF8574
: EINGANG_LESEN ( — b )
     FALSE EINGANG1 !
     FALSE EINGANG2 !
     FALSE EINGANG3 !
     FALSE EINGANG4 !
     READ
     DUP 1 AND IF TRUE
       EINGANG1 ! THEN
     DUP 2 AND IF TRUE
       EINGANG2 ! THEN
     DUP 4 AND IF TRUE
       EINGANG3 ! THEN
     8 AND IF TRUE
       EINGANG4 ! THEN
     ;
```

Zustände up-to-date halten

Um SPS-Steuerungs-Programme in Forth zu schreiben, ist die Abfrage der Eingangszustände im richtigen Zeitpunkt durchaus ein nichttriviales Problem. Im Idealfall sollten sie, in einem Hintergrundprozeß abgefragt, immer "griffbereit" vorliegen.

Allerdings muß die Abfrage mit dem Forth-Highlevel-Programm so synchronisiert werden, daß während eines Steuer-Zyklus die Eingangsdaten nicht unkontrolliert verändert werden.

Andernfalls entsteht das Problem der "Nichtkonsistenz", das zu Steuerfehlern führen kann. Diese Problematik muß von Anwendung zu Anwendung entschieden werden.

Für unsere Minimal-Hardware können wir uns ohne Interrupt oder Multitasking mit folgendem Ansatz zufriedengeben:

Wenn die Steuerung keine "Eintagsfliege" sein soll, muß sie zyklisch in einer Endlosschleife arbeiten. Es läßt sich ein allgemeines Rahmenprogramm vorgeben, in das an-

Fortsetzung des Listing zu: Der I²C-Bus

```
        STARTBEDINGUNG
        SLAVE-ADRESSE @ SEND12C  \ Slave-Adresse
        SLAVE-ACK
        SEND12C             \ Subadresse
        SLAVE-ACK
        SEND12C             \ Byte
        SLAVE-ACK
        STOPBEDINGUNG
;


: OPEN_READ ( Subadr — )
\ Eroeffnungssequenz fuer div. Leseprozesse
        STARTBEDINGUNG
        SLAVE-ADRESSE @ SEND12C  \  Slave-Adresse
        SLAVE-ACK
        SEND12C
        SLAVE-ACK
        STARTBEDINGUNG
        SLAVE-ADRESSE @ 1+ SEND12C
        SLAVE-ACK
;

: RANDOM_READ ( Subadr — data )
\ Byte lesen aus I2C-Device mit Subadresse bzw. "random read"
        OPEN_READ
        RECV12C
        STOPBEDINGUNG
;

: PCF8574 ( b — )
          2 * 40 + SLAVE-ADRESSE !      \ Hardware-Offset A0 .. A2 addieren
;


: TEA6300  ( — )
          80 SLAVE-ADRESSE !            \ kein Hardware Offset
;


: X24C02  ( b — )
          2 * A0 + SLAVE-ADRESSE !      \ Hardware Offset A0 .. A2 addieren
;

: SHOWMEM
\ Inhalt eines X24C02 EEPROM ausgeben
        CR
        0 X24C02
        0 OPEN_READ
        RECV12C
        0 3 .R 3 .R 2 SPACES
        100 1 DO
                R@  3 .R MASTER-ACK RECV12C 3 .R 2 SPACES
\ ACHTUNG, beim RSC FORTH muss R statt R@ stehen !
        LOOP
        STOPBEDINGUNG
;

: POLLWRITE ( b — )
\ Den Speicher mit "b" fuellen
        0 X24C02
        DUP 0 RANDOM_WRITE
        100 1 DO
                BEGIN
                        STOPBEDINGUNG
                        STARTBEDINGUNG
                        SLAVE-ADRESSE @ SEND12C  \ Slave-Adresse
                        SLAVE-ACK>FLAG   \ antwortet, wenn Schreiben fertig
                UNTIL
                R@ DUP . SEND12C SLAVE-ACK      \ Subadresse implizit erhoehen
\ ACHTUNG, beim RSC FORTH muss R statt R@ stehen !
                DUP SEND12C SLAVE-ACK           \ naechstes Datum schreiben
        LOOP
        STOPBEDINGUNG
;
```

wendungsspezifisch die Worte IST_SONSTNOCHWAS? und AKTOREN_SETZEN individuell nachgetragen werden müssen. Wenn diese Worte vektorisiert werden und/oder ZYKLUS insgesamt als Hintergrund-Task läuft, kann auch interaktiv in die Steuerung eingegriffen werden. Dann darf aber aus oben beschriebenen Gründen PAUSE nur einmal am Ende von ZYKLUS vorkommen. In AKTOREN_SETZEN werden die in den beiden Worten davor zusammengetragenen Zustands-Daten kombinatorisch ausgewertet und entsprechende Stellaufträge losgeschickt.

```
: ZYKLUS ( - )
     BEGIN
        EINGANG_LESEN
        IST_SONSTNOCHWAS?
        AKTOREN_SETZEN
        ( PAUSE )
        ABBRUCH?
     UNTIL
;
```

Hier wird auch klar, warum oben der Aufwand mit den vier Eingangs-Variablen getrieben wurde.

Bausteine mit Subadressen

Nun werden wir den Schwierigkeitsgrad steigern und zu Bausteinen mit Subadressen übergehen.

Die Subadresse wird auf die Slave-Adresse folgend ebenso wie diese übertragen. Zweckmäßigerweise übergibt man sie auf dem Stack. Daraus folgt unmittelbar aus WRITE das Forth-Wort RANDOM_WRITE. Zum schnellen Schreiben von Datenblöcken kann häufig WRITE statt RANDOM_WRITE verwendet werden. Details siehe unten bei "Current Write!"

Das Lesen mit Subadresse gestaltet sich etwas trickreicher. Zunächst gibt es verschiedene Adressierungsmodi:

- "Random Read": liest eine explizit adressierte Subadresse. Dazu wurde RANDOM_READ aus READ entwickelt. Auffälliger Unterschied: SLAVADR muß zweimal übertragen werden: erst zum Eintragen der Subadresse, dann um den Lesevorgang einzuleiten.
- "Current Read": In den meisten

Fortsetzung des Listing zu: Der I²C-Bus

```
: SPEED?  ( -- )
\ Dauer von RANDOM_READ kann mit dem Oszi gemessen werden.
\ bei RSC-FORTH: ?TERMINAL statt KEY?
       BEGIN 0 RANDOM_READ DROP 100 0 DO LOOP KEY? UNTIL ;

\ ### ##########################################################################

\ Hier kommt nun der maschinenabhängige Teil für H8/325

\ Programmpaket I2C Bus fuer H8/325
\ Angenommen, es besteht die Hardware Konfiguration :  SDA = P5.2
\                                                      SCL = P5.5
\ Bei Konflikt mit Nachbarports Datenreg. u. Daten-Richt-Reg. in Shadow-Reg
\ zwischenspeichern! Hier gibt es zufällig keinen Konflikt, weil die Nachbar-Ports
\ als serielle Schnittstellen vom Controller überschrieben werden.
\ P5DR  ist Hardware Adresse $FFBA
\ P5DDR ist Hardware Adresse $FFB8
HEX

VARIABLE SLAVE-ADRESSE

CODE dataHI_clockHI
       F000 , 30B8 ,     \ mov.b #0,R0H
                         \ mov.b R0H,@P5DDR: SDA und SCL Eingang
END-CODE

CODE dataLO_clockHI
       F000 , 30BA ,     \ Reinitialisierung: P5DR = LOW
       F004 , 30B8 ,     \ mov.b #4,R0H:
                         \ mov.b SDA Ausgang, SCL Eingang
END-CODE

CODE clockLO_dataHI
       F020 , 30B8 ,     \ mov.b #$20,R0H
                         \ mov.b R0H,@P5DDR: SDA Eingang, SCL Ausgang
END-CODE

CODE clockLO_dataLO
       F024 , 30B8 ,     \ mov.b #$24,R0H
                         \ mov.b R0H,@P5DDR: SDA und SCL Ausgang
END-CODE


: dataRECEIVE   ( -- b )
\ Testet den extern eingepraegten Pegel der SDA-Leitung und gibt Flag aus
       clockLO_dataHI  \ SCL LOW, SDA HIGH
       dataHI_clockHI  \ SDA und SCL HIGH
       FFBA C@
       4 AND
       clockLO_dataHI  \ SCL LOW, SDA HIGH
;

\ ### ##########################################################################
\ hier kommt der maschinenabhängige Teil für R6511 (Mini-Bee)
\ Siehe auch Anmerkungen oben im Hauptteil
\ wegen Besonderheiten des RSC-Forth

\ Programmpaket I2C Bus fuer R6511 (Mini-Bee)
\ Angenommen, es besteht die Hardware Konfiguration :  SDA = PC1
\                                                      SCL = PC0
\                              Port C hat die Hardware-Adresse 2

HEX

0 VARIABLE SLAVE-ADRESSE

CODE dataHI_clockHI
       3 # LDA,
       2 STA,
       NEXT JMP,
END-CODE

CODE dataLO_clockHI
       1 # LDA,
```

➪

I²C-Bausteinen gibt es einen Subadressen-Zeiger, der nach jedem Schreib- und Lesevorgang auf die nächsthöhere Subadresse zeigt. "Current Read" nutzt diesen Umstand aus, indem auf die Sendung der Subadresse verzichtet wird. Mit READ wird also immer von der Subadresse gelesen, auf die der Zeiger gerade gerichtet ist (der wird dabei zugleich inkrementiert). Beim Lesen von Datenblöcken ergibt das eine Zeitersparnis.

- "Sequential Read": Noch mehr Zeit kann eingespart werden, wenn der Mikrocontroller den Datenempfang mit MASTER-ACK quittiert. Der Slave-Baustein gibt dann unmittelbar den Inhalt der inkrementierten Subadresse aus, Adressierung entfällt. Der Code für diesen Fall muß anwendungsspezifisch zusammengestellt werden, als Elementar-Beispiel siehe SHOWMEM.

In Bild 4 sind die mit RANDOM_WRITE und READ und RANDOM_READ ausgeführten Impulssequenzen noch einmal grafisch dargestellt. (Quelle: XICOR Datenbuch)

Fortsetzung des Listing zu: Der I²C-Bus

```
        2 STA,
        NEXT JMP,
END-CODE

CODE clockLO_dataHI
        2 # LDA,
        2 STA,
        NEXT JMP,
END-CODE

CODE clockLO_dataLO
        0 # LDA,
        2 STA,
        NEXT JMP,
END-CODE

: dataRECEIVE   ( - b )
\ Testet den extern eingepraegten Pegel der SDA-Leitung und gibt Flag aus
        clockLO_dataHI  \ SCL LOW, SDA HIGH
        dataHI_clockHI  \ SDA und SCL HIGH
        2 C@
        2 AND
        clockLO_dataHI   \ SDA LOW SCL HIGH
;

CODE CROL  ( w - w' )
\ so schiebt der RSC-Forther ( nach K.P. Schleisiek )
        TOP ASL,     \ schiebt niederwertiges Byte 1 links, MSB ins Carry
        TOP 1+ ROL,  \ Carry ins Bit 8
        NEXT JMP,
END-CODE
```

serielle Speicherbausteine

Alle bisher vorgestellten Komandos sind notwendig, um serielle Speicherbausteine mit I²C-Interface, wie den X24C02 gezielt auslesen und beschreiben zu können.

Die Subadresse beschreibt hier die anzusprechende Speicherzelle, der X24C02 hat genau 256 Bytes Speicherkapazität.

Bei den "dickeren" EEPROMs, etwa dem X24C16, kann die Chip-Nummer nicht mehr hardwareseitig eingestellt werden, sondern die Bits 1 bis 3 der Slave-Adresse selektieren eine "Bank". Die Select-Pins A0 bis A2 sind trotzdem herausgeführt und müssen an Masse gelegt werden! Bild 5 zeigt das Pin-Diagramm der X24C02 und X24C16 Bausteine. TEST muß an Masse gelegt werden.

Pinbelegung des X24C16

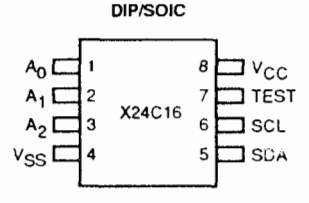


Abb. 5

Die seriellen EEPROMs sind eigentlich ideale Bausteine für Forther, um kleine bis mittlere Anwendungen direkt auf dem Board dauerhaft abzuspeichern.

Allerdings ist die Übertragungszeit für 2 kByte serieller Information schon beträchtlich, vor allem beim Schreiben, wo zwischen zwei Bytes jeweils ca. 5 MS Pause zum "Brennen" eingehalten werden muß. Dazu gibt es bei den meisten Bausteinen einen "Page Mode" der allerdings teils herstellerspezifisch ist und überdies etwas kniffelig zu programmieren, weshalb hier auf die entsprechenden Datenbücher verwiesen werden muß.

Eine andere Methode zum Abkürzen der Brennzeit bietet das "Pollen" des Acknowledge, wozu bereits oben vorsorglich SLAVE-ACK>FLAG definiert wurde: Solange ein Byte "am Brennen" ist, gibt der Chip kein Acknowledge auf die Slave-Adresse. Ein Beispielcode ist in POLLWRITE ausgeführt. Er beschränkt sich allerdings auf das zur Demonstration notwendige Minimum und muß für eine praktische Anwendung vervollständigt werden. Es gibt keine Garantie, daß die Methode bei allen EEPROM-Fabrikaten funktioniert!

Als interessantes, einfach zu programmierendes Beispiel mit hohem Freizeitwert sei zum Schluß der Chip TEA6300 erwähnt: das ist ein Audio-HIFI-Stellglied. Eine Hardware-Bauanleitung dazu gab es wohl mal in einem ELEKTOR,Heft.

Bei den Code-Beispielen für die unterschiedlichen Prozessoren ergeben sich, je nach Maschinencode-Anteil doch recht unterschiedliche Ausführungszeiten: Für einmal RANDOM_READ braucht der 8051 etwa 15 Millisekunden, der H8/325 braucht 2 Millisekunden, die Mini-Bee mit 2-MHz 6511 etwa 19 Millisekunden. Das I²C-protokollbedingte theoretische Minimum liegt bei etwa 400 Mikrosekunden.

I²C Slang

I²C-BUS: Abkürzung für "Inter-IC-Bus" Von Philips entwickeltes serielles Bussystem, speziell zur Verbindung von intelligenten Bausteinen innerhalb von Modulen und Geräten. Das System ist patentgeschützt, im Kaufpreis eines I²C-Chips ist zugleich die Lizenz für seine Anwendung enthalten.

SDA = Serial Data: Bidirektionale Leitung für Adressbytes, Datenbytes und Steuerbits.

SCL = Serial Clock Leitung: Taktet die Datenübertragung auf SDA. Der Empfänger übernimmt den Pegel von SDA während der positiven Flanke von SCL. Neue Datenpegel auf SDA werden immer eingestellt, während SCL = LOW ist. Außerhalb der Clock-Impulse bleibt SCL LOW. (Ausnahmen von dieser Regel:"Startbedingung", "Stopbedingung",und Test der Bereitschaft von Slave-Modulen).

Startbedingung: Jede Datenübertragung muß eingeleitet werden mit der Startbedingung: SDA geht von HIGH nach LOW, während SCL = HIGH ist.

Stopbedingung: Die Stopbedingung beendet eine Datenübertragung und gibt die Bustleitung wieder frei: SDA geht von LOW nach HiGH, während SCL = HIGH ist.

Open-Collector-Busleitung: Jeder Sender besteht aus einem Transisor, deren Kollektoren (bzw. Drains bei MOS-Bauteilen) untereinander und mit einem gemeinsamen Lastwiderstand verbunden sind. HIGH Zustand eines Senders ist synonym mit "Transistor sperrt", LOW ist synonym mit "Transistor leitet". Die Empfänger tasten den Pegel an der Kollektorverbindung möglichst hochohmig ab. Vorteile der Open-Collector Technik: einfach. Geringe Gefahr von Bauteilschäden bei Fehlern in der Verkabelung oder beim Stecken von Baugruppen während des Betriebs. Leichte Verbindung von Komponenten mit unterschiedlicher Betriebsspannung und Technologie. Nachteil: Relativ hochohmig (minimaler Kollektorwiderstand des I2C Bus bei 5V Betriebsspannung 1,5 kOhm). Die Leitungskapazität (Größenordnung 100 pF/Meter) begrenzt die Übertragungsgeschwindigkeit. Laut Philips Datenbuch sind maximal 400 pF insgesamt zugelassen, der Bus darf demnach maximal 4m lang sein.

Wired OR / Wired AND: Die Open-Collector Schaltungstechnik verbindet implizit das gesamte Netzwerk zu einer negativen ODER-Schaltung (bzw. was nach den Gesetzen der Boole'schen Logik dasselbe ist, zu einer positiven UND-Schaltung) . Sobald ein Baustein durchschaltet, ist sein Pegel dominant LOW über alle anderen HIGH-Pegel.

Acknowledge: Der Empfänger quittiert das letzte seriell empfangene Byte mit einem LOW-Pegel auf SDA und fordert auf zum Weitersenden.

Slave-Adresse: Individuelles Adressierungsbyte jedes vom Bus aus ansprechbaren I2C-Bausteins. Das obere Nibble ist im Baustein eingebrannt als Typ-Identifikation. Das unterste Bit ist =0, wenn der Bus-Master Daten nachfolgend Daten einschreiben will, und =1, wenn der Bus- Master Daten lesen will. Die Bits 1 bis 3 sind entweder herstellerseitig fest vorgeschrieben (dann normalerweise =0) oder bieten Möglichkeiten zur Unterscheidung mehrerer identischer Bausteine an einem Bus bzw. Selektion von Datenbereichen innerhalb des Chips.

Subadresse: Nach der Slave-Adresse zusätzlich übertragenes Adressierungsbyte zum Ansprechen bestimmter Register oder Speicherzellen im adressierten Chip.

Adressierungsarten: (Nicht jeder I²C-Baustein untertützt alle Adressierungsarten. Es kommen auch bausteintypische Varianten vor. Datenblatt studieren!) —Random Read/Write: Selektion eines bestimmten Registers in einem bestimmten Chip durch Übertragung von Slave-Adresse und Subadresse. Chips mit mehreren Registern haben einen internen Subadressen- Zeiger, der nach jedem Schreib/Lese-Zugriff auf das danach folgende Register inkrementiert wird. —Current Read/Write: Selektion eines bestimmten Chips nur mit der Slave-Adresse. Es wird die Datenzelle bearbeitet, auf die der Subadressen-Zeiger gerade gerichtet ist, dieser danach inkrementiert. Bei Chips ohne Subadressen ist Current Read/Write die einzig mögliche Adressierungsart. —Sequential Read/Write: Die Adressierung erfolgt nur einmal zu Beginn. Unmittelbar nach dem ACK kann das nächste Byte eines Datenblockes übertragen werden.

Master, Slave, Arbitration: Ein Master ist ein IC das einen Datentransfer einleitet, Clockimpulse erzeugt und den Datentransfer abschließt. Ein Slave ist ein von einem Master adressiertes IC Konzeptionell ist der I²C-Bus ein Multi-Master-Bus, d.h. mehrere Master können sich auf einem Bus die Kontrolle über die Slaves teilen, ohne daß es zur Zerstörung von Busnachrichten kommt. Die Buszuteilung wird geregelt durch die Prozedur der Arbitration: Während der von ihm ausgegebenen HIGH-Impulse hört jeder Master die Leitung ab. Sobald jetzt ein LOW-Impuls eintrifft, kann der (außer Acknowledge) nur von einem Master höherer Priorität kommen. Der Schwächere gibt nun auf. Implizit stellt dieses Verfahren sicher, daß durch konkurrierende Master keine Nachrichten zerstört werden. Der vorliegende Artikel beschränkt sich auf Single-Master Konfigurationen.

Was ist Forth?

Forth ist ein immerwährender Grund für die jährlichen Treffen all der netten Leute,

die bisher sich nicht einigen konnten (mit sich und untereinander), was Forth ist.,

obwohl sie ihren Lebensunterhalt damit bestreiten,

und das mit Spaß und Erfolg.

Sergei N. Baranoff
St. Petersburg

(frei übersetzt von akg)

Was ist Forth?

Forth ist das, was dich schneller programmieren läßt, und zwar für mehr mit weniger.

Rob Spruit
Siegen

(übersetzt von akg)

Diese zwei Gewinner wurden in Marienbad auf der euroFORTH93 vom DamenBegleit-Team ausgewählt.

Was ist wichtig, um Forth attraktiver zu machen und für etwas mehr Verbreitung zu sorgen?

Ein Vorschlag in Marienbad auf der euroFORTH93 (der auch sicherlich von Erfolg gekrönt wäre):

Nach ProgrammiererINNEN Ausschau halten.

Pudding mit Sahne

FUDGE, die MS (MilliSecond) und andere Betrachtungen zur Zeit

von Arndt Klingelnberg
Straßburger Str. 12, 52477 Alsdorf

Basis aller Steuerungs- und Meßaufgaben ist eine solide Zeitbasis. Auch die Kommunikation mit Meßgeräten usw. erfordert ein exaktes Timing. Überall dort, wo kein freier HardwareTimer/Zähler vorhanden ist, was für (sehr) einfache Controller wie auch bei ach so modernen PCs gilt, wird gerne eine unkompliziert handzuhabende DO .. LOOP genutzt, um Kurzzeit-Verzögerungen zu realisieren. Verbesserungen an dieser Schleife werden vorgestellt, wobei die große InteressenGruppen der ZF und F-PC Nutzer konkrete Lösungen vorfindet. Dazu für 'alle' einige weitere nützliche Hinweise zu Timern/Delays und zu diskreten (digitalen) Meßwerten allgemein.

Der Tick mit dem Ticker

Der so 'komfortable' PC bietet erst ab 55 ms (dem 'Ticker') einen Zugang zur RealTimeClock (RTC), und das dann aber eben auch in einer Stufung von 55 ms. DOS selbst bietet zwar anscheinend einen Zugang in 10 ms Schritten, aber die Werte beziehen sich auch auf den Ticker und springen demgemäß mit 50 bzw. 60 ms, sind also nochmal ungenauer als der Tiker. ZF und F-PC nutzen die DOStime für den Timer, F-PC-ak den Ticker.

Nun kann der Fehler eines nichtkontinuierlichen, also diskret-springenden Systems auf verschiedene Weise minimiert werden. Zum einen kann der Start-/Bezugspunkt innerhalb einer Stufung sinnvoll festgelegt werden und zwar exakt auf die Hälfte. Bei einem Timer ergibt dann ein gleichlang dauerndes Ereignis auch (fast) immer ein gleiches Ergebnis. Das bedeutet eine geringe Streuung und damit eben eine maximale Genauigkeit. Diese Methode eignet sich also hervorrangend für EinzelMessungen. Ist nun aber genug Zeit vorhanden für eine MehrfachMessung mit anschließender Mittelung, so ist gerade dieser spezielle Startzeitpunkt falsch. Dann muß der Start-/Bezugspunkt jedesmal anders, und zwar möglichst zufällig in der Stufung liegen. Genau: er muß mit exakt gleicher Wahrscheinlichkeit alle Werte der Stufung annehmen. Der Meßwert wird also mit einem exakt definierten Rauschen überlagert. In der digitalen AudioTechnik wird dieses Verfahren mit Dither bezeichnet und ist die Voraussetzung, daß die digitale AudioTechnik dank 'oversampling' und 'noise shaping' überhaupt die Qualtität der analogen HiFi-Technik erreichen bzw. übertreffen kann.

Stichworte

ms
delay
timer
discret
distribution
dither
HiFi
VolksForth
TurboForth
F83 .. ZF
F-PC (-ak)

Heisenberg oder Dither ?

Je nach gefordertem Ergebnis ist also eine anderere Voraussetzung zu wählen. Wärend der Timer in F-PC-ak von Einzelmessung umschaltbar auf Mehrfachmessung zur Mittelung (siehe unten) ist, starten die Timer in ZF und F-PC unkontrolliert. Die EinzelMessung ist also ungenau, aber mehrere Messungen können mit Erfolg gemittelt werden. Die Startpunkte sind allerdings manchmal doch korreliert zum Ticker, also doch nicht wirklich streng zufällig, und damit sind die Ergebnisse oft trügerisch, zumindest bei kurzen Zeiten mit hohen Anforderungen an die Genauigkeit. Auffällig wird das, wenn direkt hintereinander mehrere gleichartige Ereignisse gemessen werden. Außer dem ersten Ergebnis können die folgenden systematisch immer ähnliche Abweichungen zeigen. Trügerisch wird das beim Vergleich kurzer, leicht VERSCHIEDENER Ereignisse. Dann kann z.B. das 2., 5. und 9. immer kürzer ausfallen und einer Fehlinterpretation des Ergebnisses sind Tür und Tor geöffnet.

Einen Test können Sie einfach durchführen:

```
28 timer ms
```

Bitte beachten Sie, daß Sie hier sowohl die `MS` als auch den `TIMER` testen! Ein relativ ungenauer `TIMER` wird Ihnen also NIE eine exakte Bewertung der `MS` erlauben. Sie haben es also mit einer Abart des Henne-Ei-Problems bzw. des Boot-Problems zu tun, Sie können (zuverlässig) nur wechselweise die Genauigkeit hochkitzeln.

Experimentieren Sie einfach mit Ihrem System: Lösen Sie die Messung mehrfach aus, versuchen Sie die Streuung zu erfassen, variieren Sie die 28-ms (halber Ticker) nach oben oder unten bis sich deutlich andere Ergebnisse zeigen.

In ZF muß zunächst die `MS` grob kalibriert werden (siehe unten), die ist nur bei URalt-PCs und alt-ATs o.k. . Heben Sie den Wert für fudge von 65 auf 500 (386DX40).

```
28 timer ms
```

Nutzen Sie die KommandoZeilen History in F-PC um die Zeile mehrfach 'abzuschicken'. Die ak-Version erlaubt Ihnen zusätzlich auch den Wert zu edieren und dann wechselweise auf verschiedene (!) Werte aus der History rückzugreifen. In ZF kann mit der ESC-Taste gearbeitet werden, einfacher ist es aber, 20 Zeilen in eine Datei zu schreiben und dann per FIND_REPLACE zu verändern.

Die '28' wurde bewußt vor `TIMER`

gesetzt, um die Interpretationszeit für die Zahl nicht mit zu messen. Um die sehr hohe InterpreterZeit (mehrfache unbestimmt-langsame Division) gänzlich zu vermeiden, gibt es das Konzept des `TIMER'` , der das Execution-Token eines (!) folgenden Wortes ermittelt und es dann innerhalb der ZeitMeßroutine ausführt.

```
28 timer' ms
```

In diesem Fall MUSS zwangsläufig die Zahl vor `TIMER'` stehen.

Als Ergebnis Ihres Tests werden Sie in F-PC-ak einen recht harten Wechsel erleben von immer nur 0 auf immer nur 55 ms , also einen Fehler von nur max. +/-28 ms. Die Abweichungen bei F-PC und zumal ZF werden wesentlich höher ausfallen.

Versuchen Sie, in F-PC-ak `ON> AVERAGING?` davor zu setzten, und zwar jedesmal (!), das ist selbstrücksetzend. Damit wird ein exakter 'Dither' eingeschaltet und Sie können die Ergebnisse mit Erfolg mitteln. Wenn Sie jetzt 20-mal eine 12 ms Verzögerung ausmessen und die Ergebnisse mitteln, so sollten Sie diese 12 ms auch als Ergebnis erhalten, und das obwohl das System nur eine 55 ms -Stufung kennt.

Bong ! Das sind die Wunder der Mathematik rund um diskrete Systeme. Mit höheren, und zwar teilweise auch unsystematischen Fehlern geht letzteres natürlich auch in ZF oder F-PC. Dieser 'oversampling' Trick kann eben aus 10-bit DACs (Digital Analog Wandler) 12 bit herausholen, ja sogar aus 1-bit Wandlern 18 bit. Bedingung ist, daß die Stufung zwar grob sein darf, aber sie muß exakt sein. In unserem Fall bedeutet das, daß die 55 ms halt nicht 52 oder mal 57 ms betragen dürfen. Aber auch das ist so nicht ganz richtig, sie müssen halt wärend der Meßzeit im MITTEL stimmen, Schwankungen vermindern die Qualität des Dithers. Beim PC stimmen sie mit Quarz- Genauigkeit. Allerdings kann irgendeiner der vielen Interrupts — und einige haben auch mal eine höhere Priorität als die Timer-Interrupts — den Ticker verzögern. Der folgende Ticker wird dann wieder an der (absolut) exakt richtigen Position kommen, wenn auch relativ zum letzten früher.

Code1:

```
VARIABLE fudge          300 fudge !
: ms    ( u -- )        \ ms delay
        0
        ?DO   pause   fudge @ 0
              DO
              LOOP
        LOOP ;
```

Code2:

```
\ time-elapsed ( -- d ) \ in 10 ms
: fudge-cal     ( -- )    \ MS-Kalibrierung
    fudge @
    time-reset 990 dup ms time-elapsed
    drop 10 * */ fudge ! ;
```

Ürigens stammt der Trick, einen Timer ganz definiert starten zu lassen, nicht von mir, sondern von *Tom Zimmer*. Er führt für die Kalibrierung von `FUDGE` in `BUFSIZE-INIT` ersteinmal ein RTC-timer-delay 'leer' aus, nur um eine gleichbleibende Ausgangposition zu erhalten. Die `MS` in F-PC wird also jedesmal beim Starten von F-PC kalibriert. Daß `BUFSIZE-INIT` selbst nach Optimierung der Koeffizienten die `MS` aber nur ungenügend genau kalibrieren kann, ist irgendwo in den neuen Processoren ≥386DX begründet. Ab dort (Cache?) zeigen sich große Unterschiede im Zeitverhalten von `BEGIN ... UNTIL` gegenüber `DO ... LOOP` . Eine DO-LOOP-Schleife über BEGIN-UNTIL zu kalibrieren muß dann also ungenau werden.

Um Mitternacht (das ist 12:00 a.m.) feiern viele Timer zumal in PCs die Geisterstunde. Nur wenige Timer arbeiten beim Datumswechsel zuverlässig, weil das kostbare Rechenzeit und GehirnSchmalz bedeutet und ... die zuverlässige Durchführung eines Tests ist nicht trivial. Katastrophale Abweichungen sollten aber zumindest vermieden werden, sollten (!) ... .

Solche TagesWechsel-Bugs können entweder nur irritieren (wie z.B. beim Norton Commander bis v.3.x und Ordnen der Dateien nach Zeitpunkt) oder Sie machen komplette teure Meßreihen unbrauchbar oder ein fehllaufender Timer zerstört ... . Ich erwähne das, um ein ausreichende Vorsicht gegenüber Timern zu empfehlen. Ähnliche Probleme entstehen übrigens auch beim Jahrhundertwechsel in 6 Jahren. Nach Schätzungen werden Programme im Milliarden-Wert unbrauchbar, da eine Modifikation unmöglich erscheint.

Vielleicht noch eine Anmerkung zur menschlichen Zeitauflösung: Verschiedene Forschungsergebnisse deuten darauf hin, daß der Mensch bis fast hinauf zu 20 ms zwei Ereignisse als gleichzeitig erkennt. Der PC ist also um mehr als eine binäre Größenordnung zu schlecht.

Problemus digitalis

Diese an den Timern auftretenden Probleme wurden deshalb hier so ausführlich dargestellt, da sie auch allgemein in der digitalen bzw. (genauer:) diskreten (Meß-)Technik in der einen oder anderen Weise auftreten. So klingen viele (frühere) digitalen Ton-Aufnahmen deshalb so scheußlich, weil so etwas wie Dither nicht beachtet wurde, weil eben statt intelligent gerundet, integermäßig gekappt wurde. Hüten Sie sich also vor 'DDD' CDs mit älteren Aufnahmen. Analoge Aufnahmen, also 'ADD' waren damals typisch haushoch überlegen.

Nun der Kuchen

FUDGE : englisch für eine Pudding-Kuchen Variante, für die SchokoVersion gilt:

```
fudge @ ich ! " hmmm" say
```

Die nachfolgende Kalibrierung läuft auf ZF (Moers) und F-PC v. ...2.25...3.50...3.561x..??? . Sie verbessert aber auch älteren F-PC ak-Versionen, ist in ak v. ≥4.0 aber in einer ausgefeilteren Version bereits enthalten. Auf dem Vorläufer F83 (*Laxen* und *Perry*) gibt es zwar einen FUDGE und die MS dazu, aber es ist zumindest in der Standard- Ausstattung noch kein TIMER definiert. Irgendein exaktes Zeitnormal muß aber zum Abgleich vorhanden sein.

FUDGE-CAL dauert eine Sekunde und sollte den Fehler auf unter 15% vermindern. Ein erneuter Aufruf kann den Fehler, zumal wenn die Korrektur groß war, noch etwas verkleinern. Desto höhere Werte `FUDGE` erreicht, desto genauer funktioniert die `MS` und es besteht die Chance, auch eine `200us` -Schleife oder etwas ähnliches zu bau-

en. Das ist z.B. für das ausreichend 'sanfte' Beschleunigen von Schrittmotoren sehr sinnvoll (Echtzeit'92: 2. Platz ProgrammierWettbewerb für Forth, VD 92Q2 p.20 .. 24).

F-PC erreicht gut doppelt so hohe FUDGE-Werte wie ZF , weil die DO..LOOP Schleifen mehr als doppelt so schnell durchlaufen werden. Dieser Geschwindigkeitsvorteil wird deutlich geringer, wenn debuggt wurde. Der Aufruf von DEBUG in F-PC installiert ÜBERALL ein zentrales >NEXT und verändert so auch die MS. Der oben angegebene Fehler gilt also ausschließlich NON-debug. Nach Debug muß entweder neu kalibriert werden oder F-PC voll neu gestartet (und kalibriert) werden. F-PC in der ak-Version erreicht noch höhere FUDGE -Werte, da PAUSE in die äußere LOOP verschoben wurde, weiterhin wird die MS dort nach DEBUG kompensiert. Bei Multitasking wird die MS deutlich und unbestimmt langsamer. Durch die Verlagerung von Pause konnte dieses Problem in F-PC-ak zwar extrem gemindert werden, bleibt aber immer noch überdeutlich.

Das KalibrierInterval von 990 ist bewußt gewählt. Es ist durch 55 und 10 teilbar und bietet weiterhin einen guten Kompromiss zwischen Genauigkeit und DäumchenDrehen.

FUDGE-CAL ist in CODE2 dargestellt. Vorsicht, dieses FUDGE-CAL kann FUDGE nur um maximal Faktor 31 erhöhen, sonst Überlauf ! Also für i586DX160-4 bitte mit AnfangsWerten von mindestens 1000 starten.

Unter F-PC-ak wird bei höheren MS Delay-Zeiten die RTC mitbenutzt. Das ist wichtig zumal bei Rechnern mit PowerManagement (wie Olivetti Quaderno) oder unter (RoundRobin) MultiTasking. Die damit verbundene ungewisse Rechner-Schnelligkeit ergibt eine recht undefinierte Dauer der MS -Schleife. Läuft ein Rechner unter PowerManagement, so bietet die KonfigurationsDatei von F-PC-ak die Möglichkeit, einen Value so zu verändern, daß recht 'brutal' bereits oberhalb 55 ms die RTC mitbenutzt wird. Trotz dieser aufwendigen Tricks können sich immer noch große Fehler ergeben (vergleiche auch unten: VolksForth), aber eben keine katastrophalen Fehler mehr, der Rechner bleibt benutzbar. So läßt sich bei einem Quaderno bei 1/4-Normal-Takt der Fehler unterhalb 55 ms auf -10% .. +340% und oberhalb 110 ms auf -55 ms bis +190 ms eingrenzen.

Im TurboForth (Jedi, Frankreich) basiert die MS wie in *Tom Zimmer*'s Lösungen auf dem F83 von *Laxen und Perry*. Code2 muß etwas umgestrickt werden, aber das sollte anhand von TIMER.fth nicht schwerfallen.

Im PC-VolksForth nutzt *Klaus Schleisiek (-Kern)* bei der MS die RTC und zwar den TICKER. Der Fehler beträgt -55 ms .. +2 ms , je nach Verzögerungswert und Startzeitpunkt. MS muß hier allerdings zusammen mit WAIT genutzt werden. MS alleine wandelt nur (kappend) in 55 ms-Ticker-Einheiten um. Auch die anderen TimerWorte fragen den Ticker ab, sind also genauer als in ZF oder F-PC. Etwas Vorsicht ist bei TICKS angebracht: wegen Overflow sollten die Zeitintervalle max. $7FFF Tickers lang sein (signed, also 15 bit). Auch in Volksforth könnte also eine feinstufige MS -Variante wie in CODE1 gebaut werden und ähnlich wie in CODE2 kalibriert werden.

Auf so einem Controller wie dem TDS2020 mit H8/532 ist dieses ganze Zeitproblem von 'gestern'. Exakte KurzzeitDelays in Stufungen von 814 ns, 2 µs, 60 µs und 1 ms stehen in HighLevel zur Verfügung, und der Timer liefert Werte in 1 µs Genauigkeit. Doch auch dort können einige oben angestellte Überlegungen weiterhelfen. Schalten Sie versuchsweise den 250-stufigen PulsWeitenModulationsAusgang (PWM) fortwärend, und zwar zeitlich genau kontrolliert zwischen 120 und 121 hin und her. Eine Mittelung zaubert Ihnen jeden Zwischenwert. Bei 2 Zeiteinheiten á 121 und 1 Zeiteinheit á 122 gibt das dann 121.33 . Das kostet allerdings viel (!) Rechenpower und macht den Ausgang langsam (Mittelung per Filter). Aber wenn es denn dringend benötigt wird.

Und ANS ?

MS gehört zu den 'facility extension words' in ANS.

```
MS ( u – )
    Wait at least
    u miliseconds.
```

Demnach darf die MilliSekunde beliebig langsam sein und in keinem Fall schneller. Ich würde fast sagen: "Zum Glück erfüllt keine (!) der obigen Implementationen die ANS Bedingungen". Tempus fugit !

❑

NEW-MS **Arndt Klingelnberg**

In der Vierten Dimension 1992 Q4 p.21 beschrieb Thomas Beierlein eine MS die den BIOS Service INT $15 Func $83 nutzt. In F-PC-ak wurde dies als US (als 32bit µs DOUBLE) integriert und zwar mit der unter 2. beschriebenen Zusatzfunktion. Obwohl hiermit eine sehr exakte MS zu Verfügung steht, ist es trotzdem lohnend, sich mit FUDGE zu befassen:

- Die BIOS Rountine ist nur auf ATs vorhanden, also nicht auf XTs, auch nicht modernen wie z.B. auf einem Quaderno//Laptalk. Gerade alte (statt Verschrottung) oder moderne (kleine, stromsparende) XTs eignen sich aber für abgesetzte Steuerungs-/Meßaufgaben.
- TSRs oder MultiTasking können die Nutzung vereiteln, die genutzte BIOS Routine steht nur einmal (!) zur Verfügung. Bei 'besetztem' EventWait beträgt die Verzögerung ohne jede Warnung Null, es sei denn, es wird eine 'Notfall'-Do..Loop eingeklinkt, damit ist aber alles wieder beim Ausgangpunkt.
- Es sind keine kleineren ZeitStufungen als ca. 1 ms möglich (ein Problem z.B. beim Schrittmotor).

Der Rechner wird übrigens wärend der Wartezeit langsamer. Eine Kritik an einem zu langsamen Warten wäre Meta-NonSense, solange nicht - wie in der erweiterten Version möglich - wärend der WarteZeit etwas Vernünftiges ausgeführt wird (per deferred function).

Das BIOS — zumindest eines meiner (akg) BIOSse — scheint die Werte zu skalieren und zu kompensiern ! Kurze Verzögerungen sind auf einem 386DX40 trotz des FunktionsOverheads ca. 300 µs zu kurz, lange Zeiten o.k. ! Die Welt steht also Kopf.

Messen via CENTRONICS

Interruptgesteuerte Meßwerterfassung eines A/D-Wandlers mit seriellem Interface am Druckerport des PC

von Bernd Beuster
Carl-Benz-Str. 1a, 55131 Mainz

Interruptprogrammierung im PC beschränkt sich in den meisten Fällen auf die Programmierung des Keyboard- oder des Timerinterrupts. In diesem Beitrag wird eine mögliche Anwendung einer Meßwerterfassung via Interrupt über die in jedem PC vorhandenen CENTRONICS-Parallelschnittstelle vorgestellt.

Ziel dieses Artikel ist es, zu zeigen, daß Interruptprogrammierung keine Schwarze Magie ist, sondern eine einfache und sichere Angelegenheit, sofern man nur die Spielregeln beachtet.

Das Datenformats des A/D-Wandlers

Der verwendete A/D-Wandler stellt seine Daten in einem festen Zeitraster zur Verfügung (100Hz, 150Hz bzw. 200Hz). Da er, ähnlich wie die Sigma-Delta-Wandler, mit einer Rauschformung arbeitet, ist es unbedingt notwendig, daß keine Daten aus dem kontinuierlichem Datenstrom verloren gehen. Andernfalls würde das Spektrum des Quantisierungsrauschen verändert, so daß keine hinreichende Rauschunterdrückung durch das nachfolgende Digitalfilter mehr möglich wäre.

Das Datenformat der seriellen Schnittstelle wird in Abb.1 gezeigt. Das Signal /DAV (data available) zeigt an, daß die Daten im seriellen Schieberegister gespeichert worden sind und ausgelesen werden können. Dazu wird an /SCLK (seriell clock) der Schiebetakt gelegt, dessen fallende Flanke den Ausgang SDO (seriell data out) von Tri-State zu Low- bzw. Highpegel zieht. Die steigende Flanke von /SCLK stellt den Tri-State-Zustand von SDO wieder her. Der verwendete Wandler liefert 15-Bit-Daten, LSB zuerst.

Die Datenschnittstelle von der A/D-Wandlerkarte zum PC erfolgt über Optokoppler an dessen CENTRONICS-Druckerport (Abb.2). Dadurch brauchen keine Zusatzkarten im PC installiert werden. Die galvanische Trennung verhindert unerwünschte Störungen und schützt die Hardware. Die Versorgungsspannung für die Optokoppler auf der PC-Seite erfolgt über die Datenausgänge der CENTRONICS-Schnitt stelle. Werden mehrere davon parallel geschaltet, so ist deren Stromergiebigkeit ausreichend für die Optokoppler.

Die erste Möglichkeit stellt die Meßwerterfassung durch Polling dar. Hierzu wird auf die fallende Flanke von /DAV gewartet, dann erfolgen die 15 Schiebetakte zum Auslesen des seriellen Datenpuffers. In der verbleibenden Zeit bis zur nächsten fallenden Flanke von /DAV muß jegliche Datenverarbeitung abgeschlossen sein, um obige Bedingung zu erfüllen. Schon ein Zeilenvorschub im Textmodus kann bei langsamen Rechnern zu Problemen führen.

Als zweite Möglichkeit bietet sich eine interruptgesteuerte Meßwerterfassung an. Die Meßdaten werden zeitsynchron mit dem A/D-Wandlertakt in einen Puffer geschrieben und können von der Auswertesoftware asynchron wieder ausgelesen werden. Einzige Voraussetzung ist, daß die Gesamtzeit der Datenauswertung durch die Auswertesoftware so kurz ist, daß ein Überlauf des Datenpuffers vermieden wird.

Stichworte
Meßwerterfassung
Centronics
Interruptprogrammierung

Datenformat der seriellen Schnittstelle

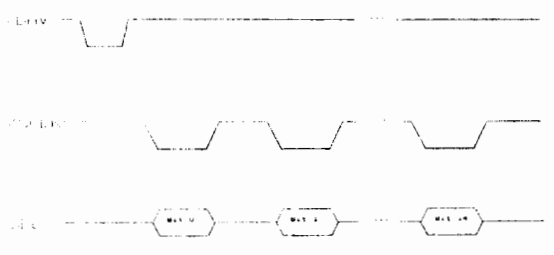


Abb. 1

Der Anschluß des AD-Wandlers

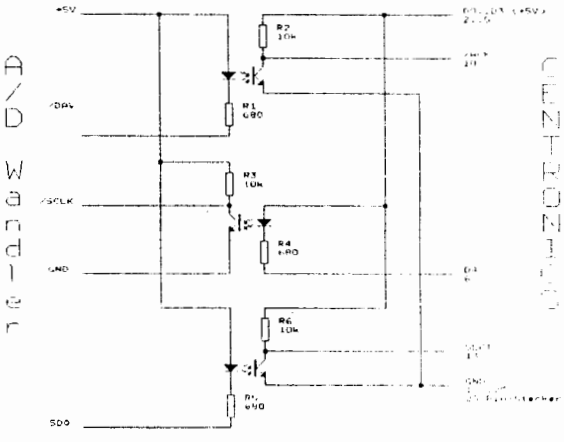


Abb. 2

Die Schnittstelle

Die CENTRONICS-Schnittstelle ist TTL-kompatibel. Hier sind nur die Datenleitungen D0..D4, und die Steuerleitungen SLCT und /ACK von Interesse (Abb.2). Über die Datenleitungen D0..D3 wird, wie schon oben erwähnt, die Spannungsversorgung der Optokoppler an D4 der Schiebetakt /SCLK bereitgestellt. Die Datenleitungen D5..D7 sind in diesem Programmbeispiel auf Null gesetzt, können aber auch Zusatzfunktionen, wie Meßstellenumschaltung o.ä. vornehmen. SLCT ist mit SDO verbunden und /ACK mit /DAV.

Zur Programmierung der CENTRONICS-Schnittstelle benötigt man die Portadressen ihrer einzelnen Register. Verwendet man LPT1, so findet man die Portadresse des Datenregisters an der Adresse $0040:0008. Die Portadresse des Statusregisters ist demzufolge Datenregisteradresse+1 und die des Controlregisters Datenregisteradresse+2. Die Bits des Datenregisters sind den Datenleitungen D0..D7 zugeordnet. Die einzelnen Bits des Status- und Controlregister haben folgende Bedeutung:

Statusregister
(nur lesender Zugriff möglich):

- 0..2:reserviert
- 3:/ERROR
- 4:SLCT
- 5:PE
- 6:/ACK
- 7:/BUSY (negiert)

Controlregister

- 0:STROBE (negiert)
- 1:AUTO FEED XT (negiert)
- 2:/INIT
- 3:SLCT IN (negiert)
- 4:IRQ enable (1=enable)
- 5..7:reserviert

Einige der Bits sind im Gegensatz zu den Signalen auf der Datenleitung negiert, was bei der Programmierung zu beachten ist. Bit_4 des Controlregisters erlaubt eine Auslösung des Hardwareinterrupts. Sobald die Leitung /ACK auf low geht, wird die Datenleitung IRQ_7 aktiv, und es wird ein Interrupt ausgelöst. Die Zuordnung von LPT1 und IRQ_7 ist mit Hilfe von Jumpern oder DIP-Schaltern auf der I/O-Karte vorzunehmen, bei ATs kann man meist noch IRQ_5

Deklaration der Konstanten — ADCCONST.SEQ

```
HEX
\ Portadresse CENTRONICS
  40 08 @L        CONSTANT DATAREGISTER   \ LPT1
\ 40 0A @L        CONSTANT DATAREGISTER   \ LPT2
  DATAREGISTER 1+ CONSTANT STATUSREGISTER
  DATAREGISTER 2+ CONSTANT CONTROLREGISTER

\ Portadresse des PIC
20              CONSTANT PIC-CONTROL
PIC-CONTROL 1+  CONSTANT PIC-MASK

\ Bitpositionen
40      CONSTANT /DAV   \ /ACK
10      CONSTANT SDATA  \ SLCT
10      CONSTANT /SCLK  \ D4
0F      CONSTANT VOPTO  \ D0..D3
DECIMAL
```

Messwerterfassung mit INT 15 (B. Beuster) — ADCINTRPT.SEQ

```
\ NEEDS DIS8086.SEQ \ für Testzwecke
NEEDS ADCCONST.SEQ

\ Verzoegerung um DELAY# Takte
VARIABLE DELAY#
 100 DELAY# !    \ i486/33Mhz
\ 16 DELAY# !    \ i486/8Mhz
\  5 DELAY# !    \ i286/8MHz
\  0 DELAY# !    \ XT/4.7MHz

LABEL WAIT%
        MOV CX, DELAY#
        INC CX          \ sonst bei cx=0 65536 loops
        HERE LOOP       \ Warteschleife
        RET   END-CODE

VARIABLE INTCOUNTER     \ Interruptzaehler
16 CONSTANT /PUFFER     \ Anzahl der gepufferten Messwerte
CREATE MESSWERTE /PUFFER CELLS ALLOT \ Messwertpuffer
VARIABLE INT            \ Pointer der Interruptroutine
VARIABLE ADC@           \ Pointer der ADC@-Routine

VARIABLE DATAOUT
\ D0..D3: Spannungsquelle für Optokoppler
\ D4    : /SCLK
\ D5..D7: frei
/SCLK VOPTO OR DATAOUT ! \ Init Variable

LABEL INT15
        PUSH AX   PUSH BX   PUSH CX   PUSH DX   PUSH DI   PUSH DS
        MOV AX, CS   MOV DS, AX
        XOR BX, BX      \ Ergebnis = 0
        MOV CX, # 15    \ 15 Bits empfangen
        HERE
           MOV DI, CX
\ fallende Flanke /SCLK
           MOV DX, # DATAREGISTER    MOV AL, DATAOUT
           XOR AL, # /SCLK   OUT DX, AL   CALL WAIT%
\ SDATA einlesen
           MOV DX, # STATUSREGISTER   IN AL, DX
           TEST AL, # SDATA
           0= IF  CLC  ELSE  STC  THEN   RCR BX, # 1 \ Bit in Stelle 15 schieben
\ steigende Flanke /SCLK
           MOV DX, # DATAREGISTER    MOV AL, DATAOUT
           OUT DX, AL   CALL WAIT%
        MOV CX, DI   LOOP
        SHR BX, # 1 \ Korrektur auf 16 Rechtsverschiebungen
\ Messwert in BX in den Messwertpuffer schreiben
        MOV AX, INT    XCHG AX, BX     MOV DX, BX \ ax=Messwert bx=dx=Pointer
        ADD BX, BX      MOV MESSWERTE [BX], AX
\ Pointer zirkular inkrementieren
        INC DX          CMP DX, #  /PUFFER
```

wählen (entspricht dann Interrupt_13).

Der Hardwareinterrupt INT 15

Dieser Interrupt ist der Parallelschnittstelle des Rechners zugeordnet. Das BIOS hat hier nur ein einfaches IRET als Interruptroutine, aber ein Druckerspooler kann hier im Hintergrund die Daten zum Drucker senden.

Ein Low-Impuls an der /ACK-Leitung (acknowledge) aktiviert die Leitung IRQ_7 des PIC (peripherie interface controller), der dem Prozessor dann über die Prozessorleitung INTR einen Interrupt anmeldet. Danach übergibt der PIC der CPU die 8-Bit-Nummer der zugehörigen Interruptroutine (in diesem Fall 15) über den Datenbus.

Automatisch wird jetzt das Flagregister auf dem Stack gerettet und das Interruptflag IF sowie das Traceflag TF des Prozessors rückgesetzt, so daß andere Interruptanforderungen über INTR erst einmal gesperrt sind. Dann rettet die CPU ihren aktuellen Programmcounter, schaut in der Interruptvektortabelle nach der Segment:Offset-Adresse der zugehörigen Interruptnummer und springt dann an diese Adresse.

Die Interruptroutine muß dem PIC eine Bestätigung des Empfangs der Interruptanmeldung geben und schließt mit einem IRET-Befehl ab, der die Flags vom Stack zurücklädt und das Programm an der Stelle fortsetzt, wo es durch den Interrupt unterbrochen wurde.

Die Interruptroutine

Obwohl eine Hochsprachenimplementierung prinzipiell möglich wäre, wurde die Interruptroutine in Assembler geschrieben. Dadurch haben auch XTs eine Chance, den Echtzeitanforderungen zu genügen. Außerdem ist die Routine nicht sonderlich kompliziert.

Zuerst - und das ist ganz wichtig - müssen die Register gerettet werden, die in dieser Routine Verwendung finden. Als nächstes ist das Datenseg-

Fortsetzung des Listings zur Meßwerterfassung mit INT 15

```
        0= IF  XOR DX, DX  THEN   MOV INT DX
\ Interruptzaehler inkrementieren
        INC INTCOUNTER WORD
\ jetzt dem PIC ein EOI melden
        MOV AL, # $20
        OUT PIC-CONTROL # AL
        POP DS   POP DI   POP DX   POP CX   POP BX   POP AX
        IRET
END-CODE


DEFER ADC@
comment:
\ Hi-Level-Version
: INT-ADC@  ( -- u )  BEGIN INTCOUNTER @ UNTIL  INTCOUNTER DECR
        MESSWERTE ADC@ @  DUPR  CELLS + @
        R 1+ DUP  /PUFFER = IF  DROP 0  THEN  ADC@ ! ;
comment;

\ Lo-Level-Version
CODE INT-ADC@  ( -- u )
        XOR AX, AX
        BEGIN  CMP AX, INTCOUNTER  0 UNTIL   DEC INTCOUNTER WORD
        MOV BX, ADC@   MOV DX, BX
        ADD BX, BX   MOV AX, MESSWERTE [BX]
        INC DX   CMP DX, # /PUFFER
        0= IF  XOR DX, DX  THEN   MOV ADC@ DX
        1PUSH   END-CODE


: POLLING-ADC@  ( -- u )
        BEGIN  STATUSREGISTER PC@  /DAV AND     UNTIL \ auf /DAV = 1 warten
        BEGIN  STATUSREGISTER PC@  /DAV AND 0=  UNTIL \ auf /DAV = 0 warten
        INLINE  INT 15  NEXT  END-INLINE \ INT 15 durchfuehren
        INT-ADC@ ;

\ Interruptvektor 15 sichern, setzen, restaurieren
2VARIABLE OLDINT15
CODE SAVE-INT15  ( -- )
        PUSH ES
        MOV AX, # $350F
        INT $21
        MOV OLDINT15 BX
        MOV OLDINT15 2+ ES
        POP ES
        NEXT    END-CODE

CODE RESTORE-INT15  ( -- )
        PUSH DS
        MOV AX, CS
        MOV DS, AX
        MOV DX, OLDINT15
        MOV DS, OLDINT15 2+
        MOV AX, # $250F
        INT $21
        POP DS
        NEXT    END-CODE

CODE SET-INT15  ( -- )
        MOV DX, # INT15
        MOV AX, # $250F
        INT $21
        NEXT    END-CODE

\ Interupt 15 verbiegen
SAVE-INT15  SET-INT15

\ Interrupt fuer PIC und CENTRONICS-Port ENABLE/DISABLE
CODE CLI   CLI   NEXT   END-CODE
CODE STI   STI   NEXT   END-CODE
: EI   CLI   PIC-MASK PC@  $7F AND  PIC-MASK PC!
       CONTROLREGISTER PC@ $10 OR  CONTROLREGISTER PC!   STI
       ['] INT-ADC@ IS ADC@ ;
: DI   CLI   PIC-MASK PC@  $80 OR   PIC-MASK PC!
       CONTROLREGISTER PC@ $0EF AND CONTROLREGISTER PC!  STI
```

⇨

ment mit den Codesegment zu laden (sind in F-PC gleich), damit die Speicherlese- und Schreiboperationen funktionieren. Da die Interruptroutine in F-PC mit dem dazugehörigen Assembler geschrieben wurde, ist daher das Code- bzw. Datensegment der Interruptroutine gleich dem der Variablen in F-PC. Man könnte sich das Setzen des Datensegments sparen, müßte dann aber immer einen CS:-Präfix vor sämtlichen Speicheroperationen schreiben. Wird das vergessen, schreibt und liest die CPU genau in dem Datensegment, in dem sie war, bevor der Interrupt aufgerufen wurde.

Nun kann das Auslesen der seriellen Daten beginnen. Es wird eine fallende Flanke an /SCLK gelegt, eine halbe Periode der Schiebetaktfrequenz gewartet und dann das entsprechende Bit eingelesen. Nun wird /SCLK wieder auf High gesetzt und wieder eine halbe Periode gewartet. Im Register BX erfolgt die Seriell-Parallel-Wandlung.

Nachdem der Meßwert vollständig eingelesen wurde, wird er in den Ringpuffer geschrieben, und der Pufferzeiger >INT der Interruptroutine wird modulo inkrementiert. Zur Synchronisation mit dem Wort ADC@, welches den Meßwert aus dem Ringpuffer liest, wird die Variable INCOUNTER inkrementiert.

Zum Schluß wird noch dem PIC eine Bestätigung gesendet und es werden alle geretteten Register wieder vom Stack zurückgeschrieben.

Die Frequenz der Schiebetaktroutine sollte so hoch wie möglich sein, damit möglichst viel Zeit für die Meßwertauswertung übrigbleibt. Sie wird neben der Taktfrequenz des Prozessors durch die Anstiegsgeschwindigkeiten der Optokoppler und der seriellen Schnittstelle des A/D-Wandlers begrenzt.

Initialisieren der Interruptroutine

Um einen Hardwareinterrupt zu bedienen sind folgende Voraussetzungen erforderlich:

- Die alten Interruptvektoren sind zu sichern und durch die der Interruptroutine zu ersetzen.
- In dem Controlregister der CENTRONICS-Schnittstelle und im Enableregister des PIC muß das dem Interrupt entsprechende Bit gesetzt werden.

Punkt 1 wird durch SAVE-INT15 und SET-INT15, Punkt 2 durch EI (enable interrupt) erledigt. Zusätzlich wird die CFA der interruptgesteuerten Meßwerterfassung INT-ADC@ nach ADC@ geschrieben. Dabei ist zu beachten, daß eine Null an der Bitposition des PIC den Interrupt freigibt und eine Eins diesen sperrt.

In diesem Beispiel wurden SAVE-INT15 und SET-INT15 schon während des Kompilierens aufgerufen, bei Stand-alone-Applikationen ist das natürlich von der Initialisierungsroutine zu übernehmen.

Wenn man die Forth-Umgebung wieder verläßt, muß man die alten Zustände wieder herstellen, d.h. das Bit im Controlregister der CENTRONICS und des Enableregister des PIC rücksetzen und die alten Interruptvektoren wieder zurückschreiben. Diesen Zweck erfüllt MYBYE-FUNCTION mit DI und RESTORE-INT15, welche von BYE aufgerufen wird.

Interruptgesteuerte Meßwerterfassung

Zu Beginn einer Messung werden die Pufferzeiger der Interruptroutine >INT und der Meßwerterfassung ADC@ sowie der Interruptzähler INTCOUNTER mit ADC-START zurückgesetzt.

INTCOUNTER dient der Synchronisation mit der Interruptroutine. "Überholt" der Pufferzeiger >INT den Zeiger >ADC@, so liegt ein Überlauf vor und wird von ADC-ERROR? entdeckt.

INT-ADC@ wurde aus Geschwindigkeitsgründen wieder in Assembler geschrieben. Es wartet, bis ein Meßwert im Ringpuffer vorhanden ist und dekrementiert INTCOUNTER. Dann liest es den Messwert an der Stelle des Pufferzeigers >ADC@ aus und inkrementiert den Zeiger modulo der Puffergröße.

(Fortsetzung auf Seite 38)

Fortsetzung des Listings zur Meßwerterfassung mit INT 15

```
      ['] POLLING-ADC@ IS ADC@ ;

: START-ADC   CLI  INT 0!  ADC@ 0!  INTCOUNTER 0!  STI ;
: ADC-ERROR?   INTCOUNTER @  /PUFFER U ABORT" Pufferuberlauf!" ;

\ Bei Verlassen von FORTH die alten Zustände wieder herstellen
: MYBYEFUNC   DEFERS BYEFUNC   DI RESTORE-INT15 ;
' MYBYEFUNC IS BYEFUNC

\ Interaktiv die Verzoegerung einstellen
: SET-DELAY   CLS
       ." Cursor oben   +10" CR
       ." Cursor unten  -10" CR
       ." Cursor rechts + 1" CR
       ." Cursor links  - 1" CR
       BEGIN KEY
          DUP 200 = IF DROP   10 DELAY# +!  ELSE
          DUP 208 = IF DROP   DELAY# @  10 -  0MAX DELAY# !  ELSE
          DUP 205 = IF DROP   DELAY# INCR   ELSE
              203 = IF        DELAY# 0DECR  ELSE
              EXIT  THEN THEN THEN THEN
       AGAIN
       CR ." DELAY# = "  DELAY# ? ;


: ADCTEST  ( # -- )  1 ?ENOUGH  ?DUP 0= ABORT" muß größer als 0 sein!"
       CLS CURSOR-OFF  START-ADC
       BEGIN   ADC-ERROR?
          DUP  0.
          ROT 0 ?DO  ADC@ 0  D+  LOOP  \ Messwerte aufaddieren
               10 10 AT INTCOUNTER @ 6 U.R
               10 12 AT  6 UD.R
       KEY? UNTIL DROP CURSOR-ON ;
EI                                    \ Interrupt an
```

FPFFT

Floating-Point-Fast-Fourier-Transformation

von Bernd Beuster
Carl-Benz-Str. 1a, 55131 Mainz

Die FFT spielt in der Meßtechnik bei der Analog-Digital-Wandlung eine wichtige Rolle. Daher freuen wir uns, daß wir eine Floating-point-Lösung vorstellen können.

Warum Floatingpoint?

Forth wird im allgemeinen mit dem Begriff Integerarithmetik verbunden. Die Operatoren mit gemischter Genauigkeit können in vielen Fällen den Gebrauch von Floatingpointarithmetik überflüssig machen. Wo allerdings der Wertebereich des zu verarbeitenden Signals große Bereiche überstreicht, wird einem der Einsatz von Integerarithmetik stark erschwert und man muß viel zusätzliche Zeit aufbringen, um die Algorithmen anzupassen.

Auch die Aussage, daß Floatingpointarithmetik um Größenordnungen langsamer ist, kann bei modernen Prozessoren mit internem oder externem Coprozessor nicht mehr gelten. Im Gegenteil, die Skalierungen mit */ und ähnlichen Operatoren können zeitraubender sein, als die direkte Floatingpointrealisierung in Hardware. In diesem Artikel wird allerdings das Software-Floatingpointpaket des F-PC von *Robert L. Smith* verwendet.

Verwendungszweck

Ich arbeite u.a. an hochauflösenden (bis zu 25 Bit) DC-Analog-Digitalwandlern, die, ähnlich wie die Sigma-Delta-Wandler, eine Rauschformung des Quantisierungsrauschens aufweisen.

Bei Wandlern ohne Rauschformung ist das Leistungsspektrum des Quantisierungsfehlers im gesamten Frequenzband $0..f_s/2$ gleichverteilt (weißes Rauschen). f_s ist die Abtastfrequenz des A/D-Wandlers.

Wird nun dieses Leistungsspektrum mit einen digitalen Tiefpaß um den Faktor K verringert, so ergibt sich eine Verringerung der Leistung des Quantisierungsfehlers um den Faktor K, bzw. die Verringerung des Fehlers des Meßwertes um den Faktor Wurzel(K). Mit anderen Worten: Um ein K-fach genaueres Ergebnis zu bekommen, muß man K^2 Meßwerte verarbeiten. Das kann bei großen K zu extrem langen Meßzeiten führen.

Bei einem A/D-Wandler mit Rauschformung sind die Leistungspektren des Quantisierungsfehlers zum größten Teil in den oberen Bereich des Frequenzbandes transformiert, so daß bei einer Filterung mit einem geeignetem digitalen Tiefpaß eine überproportionale Verringerung der Quantisierungsleistung um den Faktor K^{2n+1} erreicht wird, wobei n die Ordnung des Sigma-Delta-Wandlers darstellt. Üblich sind Ordnungen von n=1..4. Wer einen CD-Player oder DAT-Recorder mit 1-Bit-Technik besitzt, der kann mit Sicherheit einen Sigma-Delta-Wandler sein eigen nennen.

Beim Experimentalaufbau dieser

Stichworte

FFT
Floating-Point
F-PC

Listing zu: FPFFT (B. Beuster) FFT.SEQ

```
\ FFT und IFFT
comment:
Die FFT- und IFFT-Routinen sind wie in |1| definiert, mit dem
Zusatz, daß die IFFT die Skalierung um N durchführt.
Die Programmstruktur ähnelt der FFT/IFFT von |2|.

|1| William H. Press, "Numerical Recipes in C", Cambridge University,
    1988, pp. 398-447.
|2| Joe Barnhart, "Forth and the Fast Fourier Transform",
    Dr. Dobb's Toolbook of Forth, pp. 239-260.
comment;

NEEDS UCASE.SEQ      \ Wil Badens Ultimate CASE Statement
NEEDS SFLOAT.SEQ     \ Software-Floatingpoint by R.L. Smith

ONLY FORTH DEFINITIONS ALSO   DECIMAL

FLOATS

\ ********** Math-Tools ************************************
: FCELLS  ( n1 -- n2 )  F#BYTES * ;
: FCELL+  ( n1 -- n2 )  F#BYTES + ;
: FARRAY  ( n -- )  DUP CONSTANT   FCELLS ALLOT  DOES>  ( -- a ) ;
: FELEMENT  ( a1 i -- a2 )  FCELLS +  CELL+ ;

: F2*  ( FS: r1 -- r2 )  FDUP F+ ;
: F2/  ( FS: r1 -- r2 )  0.5 F* ;
: FSQR ( FS: r1 -- r2 )  FDUP F* ;

: ZCELLS  ( n1 -- n2 )  2* FCELLS ;
: ZCELL+  ( n1 -- n2 )  F#BYTES 2* + ;
: ZARRAY  ( n -- )  DUP CONSTANT   ZCELLS ALLOT  DOES>  ( -- a ) ;
: ZELEMENT  ( a1 i -- a2 )  ZCELLS +  CELL+ ;
: RE.ZELEMENT  ( a1 i -- a2 )  ZELEMENT ;
: IM.ZELEMENT  ( a1 i -- a2 )  ZELEMENT FCELL+ ;
: ZVARIABLE   CREATE  1 ZCELLS ALLOT  DOES> ( -- a ) ;
```

Wandler stellte es sich als günstig heraus, die Spektren des Quantisierungsfehlers grafisch darzustellen, da man damit sehr gut Störungen im Aufbau erkennen kann. So ist z.B. der Verlauf der zu erwartenden Rauschformung bekannt und kann mit dem gemessenen qualitativ verglichen werden. Fehler in der Stromversorgung, wie Netzbrumm, erscheinen als Spektrallinien im Leistungsspektrum.

Wahl der FFT/IFFT

Die Schnelle Fouriertransformationen (FFT) und die Inverse Schnelle Fouriertransformation (IFFT) sind schon in [2] und [3] in Forth programmiert worden. Sie laufen mit 16-Bit Genauigkeit.

Für den obigen Anwendungsfall wurde aber Floatingpoint Genauigkeit mit einer Dynamik von >60 dB verlangt. Eine wahre Fundgrube für Algorithmen ist [1], wobei man sich an "... in C" als Forther nicht stören sollte - dasselbe Buch gibt es auch für FORTRAN und PASCAL. Die Algorithmen in diesem Buch sind alle mathematisch exakt und numerisch optimiert, und es wird zu jedem die mathematische Herleitung angegeben.

Aus Gründen der Übersichtlichkeit des Programms wurden Operatoren für komplexe Zahlen wie in [2] erstellt und die Algorithmen in [1] als komplexe Algorithmen umgeschrieben. Das führt nicht notwendigerweise zum schnellsten Programmablauf, erhöht aber die Übersichtlichkeit. Wer Wert auf Geschwindigkeit legt, muß die innersten Schleifen in Assembler optimieren; am besten gleich in Verbindung mit dem Co-Prozessor. Für die Echtzeitanforderungen der obigen Anwendung reichte die Geschwindigkeit aus.

Die Datenstrukturen

Für die Programmentwicklung wurde F-PC in Verbindung mit der Software-Floatingpointerweiterung SFLOAT von *Robert L. Smith* verwendet. Diese benutzt einen separaten Floatingpointstack, was die Programmierung erheblich vereinfacht.

Fortsetzung des Listings zu FPFFT (B. Beuster)

```
: Z!  ( a -- ; FS: z -- )  FSWAP DUP F!  FCELL+  F! ;
: Z@  ( a -- ; FS: -- z )  DUP  F@  FCELL+  F@ ;
: ZDUP   ( FS: z1 -- z1 z1 )  F2DUP ;
: ZSWAP  ( FS: z1 z2 -- z2 z1 )  FSWAP 3 FNSWAP FSWAP 2 FNSWAP ;
: ZOVER  ( FS: z1 z2 -- z1 z2 z1 )  FPOP FPOP
   FOVER FOVER   FPUSH F-ROT   FPUSH F-ROT ;
: Z2DUP  ( FS: z1 z2 -- z1 z2 z1 z2 )  ZOVER  ZOVER ;

: ZJ*  ( z1 -- z2 )  FNEGATE FSWAP ; \ z2 = j*z1
: ZKONJ ( FS: z1 -- z2 )   FNEGATE ; \ komplex konjugiert
: Z+  ( FS: z1 z2 -- z3 )  FROT F+   F-ROT F+  FSWAP ;
: Z-  ( FS: z1 z2 -- z3 )  FROT F\-  F-ROT F-  FSWAP ;
: Z+! ( a -- ; FS: z -- )  DUP Z@  Z+  Z! ;
: Z*  ( FS: z1 z2 -- z3 )  FOVER 3 FPICK F* FOVER 5 FPICK F*
   F+ FPOP   FROT F*  F-ROT F*  F\-  FPUSH ;

DEFER ?Z2/
: Z2/  ( FS: z1 -- z2 )  FSWAP F2/  FSWAP F2/ ;
: -Z2/ ( FS: z1 -- z2 )  FSWAP -0.5 F*  FSWAP -0.5 F* ;

: ZPWR ( FS: z -- power )  FSQR   FSWAP FSQR  F+ ;
: |Z|  ( FS: z -- abs )    ZPWR  FSQRT ;



\ ************* FFT/IFFT - Routinen ***************************
\ Hilfsgrößen zur Stackvereinfachung
0 VALUE FFT-VECTOR
0 VALUE N
: !FFT-VECTOR  ( a -- )  1 2ENOUGH  DUP !> FFT-VECTOR  @ !> N ;

: FFT-ELEMENTS  ( i j -- ai aj )
   FFT-VECTOR ROT ZELEMENT   FFT-VECTOR ROT ZELEMENT ;

\ vertauscht Element[i1] mit Element[i2]
: SWAP-VECTOR  ( i1 i2 -- )  FFT-ELEMENTS
   DUP Z@   OVER Z@   Z!  Z! ;

\ ohne SWAP-VECTOR 17ms @ 1286/20; N=256
\ mit  SWAP-VECTOR 40ms (13ms @ i486/33)
: BIT-REVERSE  0  N 0 DO
       DUP I > IF  DUP I SWAP-VECTOR  THEN
       N 2/ SWAP
       BEGIN  2DUP <=  OVER AND
       WHILE  OVER -  SWAP 2/ SWAP  REPEAT +
   LOOP  DROP ;

VARIABLE -EXP? \ TRUE bei negativem Exponenten
ZVARIABLE U
ZVARIABLE W

comment:
Normalerweise ist
    W = exp(jx)
      = cos(x) + j*sin(x)
Aus Genauigkeitsgründen wird aber
    W = -2.0*sin(x/2) + j*sin(x)
      = cos(x) - 1 + j*sin(x)
Das muß dann bei INCR-ANGLE mit einer zusätzlichen Addition
von U berücksichtigt werden.
comment;
: IFFT-ANGLES  ( i -- )
   PI  IFLOAT F/  -EXP? @ IF  FNEGATE  THEN
   FDUP F2/  FSIN  FSQR  -2.0 F*  FSWAP FSIN  W Z!
   F1.0 F0.0 U Z! ;
: INCR-ANGLE   W Z@  U Z@  Z*  U Z+! ;

DEFER FFT-SKAL
: BUTTERFLY  ( i j -- )   FFT-ELEMENTS  ( ai aj )
   DUP  Z@  U Z@  Z*                   ( ai aj ; FS: z )
   OVER Z@  ZOVER Z-  FFT-SKAL Z!      ( ai aj ; FS: z )
   DUP  Z@        Z+  FFT-SKAL Z! ;
: INNER-LOOP  ( startpt incr -- )
   DUP ?Z/   ROT N  SWAP I-  DO   I 2DUP +  BUTTERFLY   OVER +LOOP  2DROP ;
: (FFT-KERNEL)  ( i -- )   DUP 2*  SWAP 1+ 1 DO
```

Bei den komplexen Zahlen liegt der Imaginärteil auf der obersten Stackposition. In den komplexen Arrays ist an der Speicherstelle n der Realteil und an Speicherstelle n+1 der Imaginärteil gespeichert.

Die Datenstruktur der Arrays ist folgendermaßen: In der ersten Speicherzelle liegt eine 16 Bit Zahl, welche die Anzahl der Elemente des Arrays darstellt. Dadurch wird es möglich mit:

```
arrayname FFT
arrayname IFFT
```

zu arbeiten, ohne immer die Information über die Anzahl der Datenelemente auf dem Stack mit angeben zu müssen. Danach folgen die einzelnen Datenelemente.

FFT und IFFT komplexer Zahlen

Es soll hier nicht auf den Algorithmus der FFT/IFFT eingegangen werden, sondern es werden nur die Datenstrukturen dargestellt.

Im Gegensatz zu [1] und [2] wurde mit Arrays gearbeitet, die mit den Index 0 beginnen und mit dem Index N-1 aufhören. Für die FFT wird ein Array mit N komplexen Datenelementen übergeben:

h(0)
h(1)
...
h(N-1)

Als Ergebnis erhält man ein komplexes Array mit je N/2 positiven und negativen Frequenzen:

H(0) (Gleichanteil)
$H(f_s/N)$
...
$H(f_s[N/2-1]/N)$
$H(+/-f_s/2)$ (Kombination der positiven und negativen Frequenzen)
$H(-f_s[N/2-1]/N)$
...
$H(-f_s/N)$ (fs ist die Abtastfrequenz.)

Bei der IFFT ist analog das komplexe Frequenzarray zu übergeben, und man erhält das komplexe Datenarray. Zusätzlich wurde im Gegensatz zu [1] bei der IFFT noch eine Skalie-

Fortsetzung des Listings zu FPFFT (B. Beuster)

```
    I OVER INNER-LOOP   INCR-ANGLE  LOOP  DROP ;

: FFT-KERNEL   N 1 DO
    1 !FFT-ANGLES   I (FFT-KERNEL)  I +LOOP ;

: [FFT]    BIT-REVERSE
   -EXP? OFF  ['] NOOP IS FFT-SKAL  FFT-KERNEL ;
: [IFFT]   BIT-REVERSE
   -EXP? ON   ['] Z2/  IS FFT-SKAL  FFT-KERNEL ;

\ 770 ms @ i286/20MHz; N=256 Daten reell
: FFT   ( a -- )  !FFT-VECTOR   [FFT] ;
\ 990 ms @ i286/20MHz; N=256 Daten reell
: IFFT  ( a -- )  !FFT-VECTOR   [IFFT] ;


\ ******* FFT und IFFT von reellen Daten ***************************
: !REAL-FFT-ANGLES
    PI  N IFLOAT F/  -EXP? @ IF  FNEGATE  THEN
    FDUP F2/  FSIN  FSQR  -2.0 F*  FSWAP FSIN  ZDUP W Z!
    FSWAP F1.0 F+  FSWAP U Z! ;


: HLP1  ( a1 a2 -- ; FS: -- z )  SWAP Z@  Z@
    ZKONJ  Z+  Z2/ ;
: HLP2  ( a1 a2 -- ; FS: -- z )  SWAP Z@  Z@
    ZKONJ  Z-  ZJ* Z2/ ;
: HLP3  ( a1 a2 -- )  0 Z@  Z*  Z2DUP Z-   ZKONJ Z!  Z+  Z! ;

: (REAL-FFT-KERNEL)  ( a1 a2 -- )
    2DUP  HLP1   2DUP HLP2   HLP3 ;


: <REAL-FFT>  ( -- a ; FS: -- z )  FFT-VECTOR 0 ZELEMENT
    DUP Z@   ZDUP F+   F-ROT F-  ;
: (REAL-FFT)  ( a -- ; FS: z -- )  <REAL-FFT>  Z! ;
: (REAL-IFFT) ( a -- ; FS: z -- )  <REAL-FFT>  Z2/ Z! ;

: !REAL-FFT-VECTOR  ( a -- )  1 ?ENOUGH DUP !> FFT-VECTOR  @ 2/ !> N ;

: REAL-FFT-KERNEL  ( a -- )  !REAL-FFT-VECTOR
    [FFT]   !REAL-FFT-ANGLES
    N 2/ 1 DO
        I  N OVER -  FFT-ELEMENTS  (REAL-FFT-KERNEL) INCR-ANGLE  LOOP
    (REAL-FFT) ;

: REAL-IFFT-KERNEL  ( a -- )  !REAL-FFT-VECTOR
    !REAL-FFT-ANGLES
    N 2/ 1 DO
        I  N OVER - FFT-ELEMENTS  (REAL-FFT-KERNEL) INCR-ANGLE  LOOP
    (REAL-IFFT)  [IFFT] ;


\  490 ms @ i286/20MHz; N=256
: REAL-FFT  ( a -- )
    ['] -Z2/ IS ?Z2/   REAL-FFT-KERNEL ;
\  550 ms @ i286/20MHz; N=256
: REAL-IFFT ( a -- )
    -EXP? ON    ['] Z2/ IS FFT-SKAL \ wird in REAL-FFT von [FFT] erledigt
    ['] Z2/  IS ?Z2/   REAL-IFFT-KERNEL ;


\ ********** Power Spectrum Density von reellen Daten *********
DEFER FFT-WINDOW-FUNCTION
FVARIABLE FLOAT(2PI/N)
PI F2* FCONSTANT 2PI

: RECTANGLE-WINDOW  ( FS: r1 i -- r2 )  FDROP ;
: HANNING-WINDOW  ( FS: r1 i -- r2 )
    F0.5 F+  FLOAT(2PI/N) F@ F*  FCOS  F2/  F0.5 F\-  F*
    F2* ;
: HAMMING-WINDOW  ( FS: r1 i -- r2 )
    F0.5 F+  FLOAT(2PI/N) F@ F*  FCOS  0.46 F*  0.54 F\-  F*
    1.85 F* ;
: BLACKMAN-WINDOW  ( FS: r1 i -- r2 )
```

rung um den Faktor 1/N durchgeführt. Dadurch wird diese Routine mathematisch exakt, dauert aber etwas länger, als die dazugehörige FFT. Wen das stört, der muß im Wort BUTTERFLY die Skalierungen entfernen.

FFT und IFFT reeller Zahlen

In der Realität hat man es in der Regel mit reellen Meßwerten zu tun, deshalb ist es angebracht, der Transformation reeller Zahlen etwas mehr Aufmerksamkeit zu widmen.

Für die FFT komplexer Zahlen könnte man die Imaginärteile im komplexen Datenarray auf Null setzen. Das ist aber nicht optimal hinsichtlich der Rechenzeit und des Speicherplatzes, da bei reellen Daten die Transformierten an den negativen Frequenzen die komplex konjugierten der positiven Frequenzen sind, d.h. H(-f)=H(f)*.

Deswegen wird für eine FFT/IFFT eine Transformation der Länge N-1 durchgeführt, und eine entsprechende Vor- bzw. Nachbehandlung der Ergebnisse durchgeführt.

Für die FFT reeller Zahlen wird ein reelles Datenarray der Länge N übergeben:

h(0)
h(1)
...
h(N-1)

und man erhält den positiven Teil der komplexen Frequenzen zurück. Da der Wert an H(0) und an H(f_s/2) reell und voneinander unabhängig sind, teilen sich diese beiden Werte den Speicherplatz des komplexen Arrayelementes 0.

Re: H(0)
Im: H(+/-f_s/2)
H(f_s/N)
H(f_s2/N)
...
H(f_s[N/2-1]/N)

Fortsetzung des Listings zu FPFFT (B. Beuster)

```
    F0.5 F*  FLOAT(2PI/N) F@ F*  FDUP FCOS  F2/  0.42 F\-
    FSWAP   F2* FCOS 0.08 F*  F+   F*
    2.38 F* ;

' HANNING-WINDOW IS FFT-WINDOW-FUNCTION

: FFT-WINDOW  ( a -- )   DUP @  ( a # )
    2PI  DUP IFLOAT F/  FLOAT(2PI/N) FI \ dient der Beschleunigung
    ?FOR
        DUP R@ FELEMENT  DUP
        F@  R@ IFLOAT  FFT-WINDOW-FUNCTION  F!
    NEXT  DROP ;

\ führt eine FFT mit reellen Daten durch
\ Windowing verringert Aliasing
: PSD  ( a -- )  ?ENOUGH  DUP FFT-WINDOW  REAL-FFT ;

\ erzeugt Leistung der Frequenz[i] aus dem FFT-Vector
\ Leistung wird verdoppelt (+3dB), da positive und negative
\ Frequenzen (außer DC-Linie).
: FFT>PWR  ( PS: i -- ; FS: -- r )
    CASE 0=  OF  FFT-VECTOR 0 RE.ZELEMENT F@  FSQR      ELSE
    CASE N = OF  FFT-VECTOR 0 IM.ZELEMENT F@  FSQR F2*  ELSE
    FFT-VECTOR  SWAP ZELEMENT Z@  ZPWR  F2*  THEN  THEN
    N IFLOAT  F/ ;


\S ********** Beispiele ****************************************

\ 1. FFT/IFFT komplexer Daten
16 ZARRAY ZDATA
: INIT-ZDATA   ZDATA @ ?FOR  R@ IFLOAT  FDUP -2.0 F*
    ZDATA R@ ZELEMENT Z!  NEXT ;
: .ZDATA   ZDATA @ 0 DO
    CR I .   TAB
    ." RE: "  ZDATA I RE.ZELEMENT F@   10 15 F.R   TAB
    ." IM: "  ZDATA I IM.ZELEMENT F@   10 15 F.R  LOOP ;

: TEST1   CR ." FFT/IFFT komplexer Daten"
    CR CR ." INIT" INIT-ZDATA  .ZDATA  KEY DROP
    CR CR ." FFT"  ZDATA FFT   .ZDATA KEY DROP
    CR CR ." IFFT" ZDATA IFFT  .ZDATA ;

\ 2. FFT/IFFT reeller Daten
16 FARRAY FDATA
: INIT-FDATA   FDATA @ ?FOR  R@ IFLOAT  FDATA R@ FELEMENT F!  NEXT ;
: .FDATA   FDATA @ 0 DO
    CR I .   TAB
    FDATA I FELEMENT F@   10 15 F.R   LOOP ;

: TEST2   CR ." FFT/IFFT reeller Daten"
    CR CR ." INIT"      INIT-FDATA       .FDATA  KEY DROP
    CR CR ." REAL-FFT"  FDATA REAL-FFT   .FDATA KEY DROP
    CR CR ." REAL-IFFT" FDATA REAL-IFFT  .FDATA ;


\ 3. PSD reeller Daten
: .FFT-WINDOW-FUNCTION
    ['] FFT-WINDOW-FUNCTION >BODY @  >NAME .ID ;

: INIT-PSD   FDATA @ ?FOR  R@ IFLOAT 2PI F*  0.23 F*  FSIN
    FDATA R@ FELEMENT F!  NEXT ;

\ wandelt Leistung in Dezibel (1.0 = 0 dB)
: PWR>DB   ( r1 -- r2 )  1E-10 FMAX FLOG  F10.0 F* ;
: .POWER   ( i -- )  FFT>PWR  PWR>DB  1 6 F.R  ."  dB";
: .POWER-SPECTRUM  N 1-  0 DO
    CR I . TAB  I .POWER  LOOP ;

: TEST3   CR ." PSD reeller Daten mit " .FFT-WINDOW-FUNCTION
    CR CR ." INIT" INIT-PSD    .FDATA  KEY DROP
    CR CR ." PSD"  FDATA PSD   .POWER-SPECTRUM ;
```

❑

PSD - Schätzung der Spektralen Verteilung

Oftmals ist man an der Verteilung der Leistungspektren interessiert. Hierzu definiert man in [1] die Power Spectral Density (PSD):

$$P_h(f) = |H(f)|^2 + |H(-f)|^2 \quad 0 \le f < \infty$$

Bei reellen Funktionen wird $P_h(f) = 2|H(f)|^2$.

Enthält das Datenarray Frequenzen, die nicht genau auf den Frequenzen der Fouriertransformierten liegen, so entstehen Aliasingeffekte. Um diese zu verringern multipliziert man die Eingangsdaten mit einer Fensterfunktion, was dann im Frequenzbereich einer Faltung entspricht. Eine optimale Fensterfunktion gibt es nicht, deswegen wird diese vektorisiert ausgeführt, so daß man sich die für jeden Anwendungsfall passende aussuchen kann.

Da die Fensterfunktionen (bis auf das Rechteckfenster) die Gesamtleistung des Daten verringern, werden zum Ausgleich die Daten mit einem Korrekturfaktor multipliziert.

Programmiertrick zur Vermeidung von Rundungsfehlern

Bei der FFT/IFFT in [1], wird eine komplexe Zahl U mit dem komplexen Winkel $W = e^{jx} \cos(x) + j \sin(x)$ multipliziert. Nun ist aber bei kleinen x der Kosinus nur wenig kleiner als 1 ($=1-x^2/2$), so daß die Genauigkeit durch die Länge der Mantisse der Floatingpoint-Darstellung des Rechner beeinträchtigt wird. Aus diesem Grund wird in dem Algorithmus nach [1] auf den Kosinus verzichtet, und durchgängig mit dem Sinus gerechnet, der bei kleinen x sehr genau (=x) ist.

$$\mathrm{Re}\{W\} = -2\sin^2(x/2) = \cos(x)-1$$
$$\mathrm{Im}\{W\} = \sin(x)$$

Deshalb muß bei der komplexen Multiplikation mit U noch eine zusätzliche komplexe Addition von U vorgenommen werden, um die Subtraktion von Re{W} wieder auszugleichen:

$$U := U + UW$$

Verbesserte FOR/NEXT-Version FORNEXT.SEQ

```
\ Verbesserte FOR/NEXT-Version
\ Bernd Beuster  11/28/93 12:16
comment:
Mit FOR..NEXT wird das orginale Verhalten von F-PC 3.56 erreicht,
d.h. es erfolgen u+1 Schleifendurchläufe. (FOR) macht dasselbe wie
>R, dient aber dem Decompiler als Label.
?FOR..NEXT liefert exakt u Schleifendurchläufe und ist daher konsistenter.
comment;

FILES DEFINITIONS
VARIABLE FORNEXT.SEQ
FORTH DEFINITIONS

DECIMAL

VARIABLE DUMMY-ADDRESS  \ für ?>RESOLVE in NEXT
CODE (FOR)  ( u -- )
    SUB RP, # 2          POP 0 [RP]  \ count zum Returnstack
    NEXT   END-CODE
: FOR  \ compile time: ( -- ?1 a1 ?2 a2 )
    COMPILE (FOR)  TRUE DUMMY-ADDRESS  ?<MARK ; IMMEDIATE

CODE (?FOR)  ( u -- )
    SUB RP, # 2          POP 0 [RP]  \ count zum Returnstack
    MOV IP, ES: 0 [IP]  NEXT         \ Sprung zum Schleifenende
END-CODE
: ?FOR  \ compile time: ( -- ?1 a1 ?2 a2 )
    COMPILE (?FOR)  ?>MARK  ?<MARK ; IMMEDIATE

: NEXT  \ compile time: ( ?1 a1 ?2 a2 -- )
    2SWAP ?>RESOLVE  COMPILE NEXT|  ?<RESOLVE ; IMMEDIATE

\ bricht Schleifendurchlauf beim nächsten NEXT ab
CODE UNNEXT   MOV 0 [RP], # 0   NEXT   END-CODE

CODE ?UNNEXT ( ? -- )  POP CX
    CX<>0 IF  MOV 0 [RP], # 0  THEN   NEXT   END-CODE



\S
: TEST  ( u -- )   FOR  R@ .  NEXT ;
: ?TEST ( u -- )  ?FOR  R@ .  NEXT ;
: TEST1 ( -- )    10 ?FOR  R@  DUP .  3 * ;
```

Programmbeispiele

`TEST1` und `TEST2` führen je eine FFT/IFFT mit komplexen bzw. reellen Daten durch. Als Testsignal wurde ein Sägezahn gewählt. Man kann damit die Datenstrukturen der Arrays und die Rechengenauigkeit untersuchen.

`TEST3` dient der Ermittlung der PSD. Als Testsignal dient ein Sinussignal, das absichtlich kein ganzzahliges Vielfaches der Abtastperiode ist, um den Einfluß der Fensterfunktionen auf die PSD zu untersuchen.

Literatur

- [1] William H. Press, "Numerical Recipes in C", Cambridge University Press, 1988, pp.8-447.
- [2] Joe Barnhart, "Forth and the Fast Fourier Transform", Dr. Dobb's Toolbook of Forth, pp.9-260.
- [3] Klaus Kohl, "Fourieranalyse in FORTH", Vierte Dimension 3/92, S.-15.

(Die "ultimative CASE-Anweisung", die im Quellcode zu FPFFT benötigt wird, ist auf Seite 39 abgedruckt)

DOS unter Kontrolle

von Friederich Prinz
Hombergerstr. 335, 47443 Moers, Tel: 02841-58398

Wen hat es nicht schon einmal geärgert, daß eine DOS-Meldung unnötig den Bildschirm verunstaltet? Abhilfe schafft die Installation eines temporären 'Critical Error Handlers'.

Bei der Arbeit mit ZF's Oberfläche, vor allem bei der Arbeit mit *Tom Zimmer's* "Dateiauswahlfenster" (dem Editor vorgeschaltet) wird immer wieder als zumindest störend empfunden, daß sogenannte 'kritische Fehler' zu den bekannten rigorosen Fehlermeldungen des Betriebssystems (MSDOS) führen. Ganz besonders das Anwählen eines Diskettenlaufwerkes, in dem sich keine Diskette befindet, führt, je nach Version des DOS, zu der meist sehr störenden Bildschirmausgabe

```
A)bbrechen W)iederholen
U)ebergehen.
```

Störend ist diese Ausgabe vor allem dann, wenn durch sie dauerhaft die Bildschirmausgabe einer eigenen Applikation zerstört wird, so geschehen bei Arbeiten an einer technisch und optisch verbesserten Version der Dateiauswahl in ZF.

Möglichkeiten, dieses 'Verhalten' des Betriebssystems abzustellen, zeigt der nachfolgende Kurzaufsatz für das ZF Forth auf.

Kritische Fehler in DOS

MSDOS/PCDOS betrachtet einige potentielle Fehler als so kritisch, daß es sich veranlaßt sieht, eine Meldung an den Menschen vor dem Bildschirm auszugeben, ohne Rücksicht auf bestehende Bildschirminhalte zu nehmen. Solche Fehler sind zum Beispiel offene Diskettenlaufwerke, auf die zugegriffen werden soll, oder auch 'offLine' wartende Drucker. Wann immer ein Fehler dieser Art auftritt, verzweigt die aufrufende DOS-Routine in den Interrupt 24h, dessen Abarbeitung die PC-Nutzer unter anderem die oben angegebene Meldung auf dem Monitor verdanken. Zunächst gibt es keine Möglichkeit, alle potentiellen Fehler im Zusammenhang mit Diskettenlaufwerken im Vorfeld abzufragen. Ein Laufwerk kann einfach 'leer' sein, oder es ist nicht verriegelt, oder eine Diskette auf die geschrieben werden soll, ist schreibgeschützt. Alle diese Informationen kann eine beliebige Applikation erst erfahren, wenn durch den Versuch eines Zugriffes DOS selbst diese Informationen 'abgesammelt' hat.

Stichworte
ZF
DOS-Fehlermeldung
INT 24h

DOS 'austricksen'

Nun mag die oben beschriebene Fehlermeldung auf der Kommandozeilenebene des DOS noch recht sinnvoll sein. Schließlich muß der Anwender darüber informiert sein, warum z.B. seine Leseversuche keinen Erfolg haben. In Applikationen, die nach einem solchen Fehler ihre Bildschirminhalte erneut aufbauen müssen, oder noch empfindlicher geschädigt werden (bis zum Abbruch der Applikation), wäre es sinnvoller, wenn sich DOS's Meldung durch eine applikationsspezifische Meldung ersetzen ließe. Das ist, wie noch gezeigt werden soll, durch eine Manipulation des Interruptvektors 24h recht einfach möglich.

Der Interrupt 24h ist technisch als FAR CALL Routine implementiert, deren Aufrufadresse in der Tabelle der Interruptvektoren zu finden ist. Hier können (fast) beliebige Manipulationen ansetzen. Die einfachste Alternative ist dabei, wie bereits im Aufsatz über die "UHR" (siehe VD Ausg. 4.'92 - 1.'93 - 2.'93) beschrieben, die Adresse der DOS-Routine temporär gegen die Adresse einer eigenen Routine auszutauschen. Dieser 'Trick' ist weder 'illegal' noch unüblich. Die Anwender von Werkzeugprogrammen wie zum Beispiel Norton's Commander, sehen täglich praktische Beispiele hierzu.

Forth und der INT 24 h

Was beliebige DOS-Applikationen können, kann FORTH selbstverständlich schon lange', zumindest wenn es sich um ein FORTH handelt, das auf DOS aufsetzt', wie z.B. das ZF. Eine der wenigen, aber wichtigen Voraussetzungen ist dabei die Verfügbarkeit eines 'integrierten' Assemblers, mit dessen Hilfe sich die Manipulationen im Betriebssystem problemlos realisieren lassen. Hier muß beachtet werden, daß der im nachfolgenden Beispiel benutzte Assembler die UPN befolgt. Unter Berücksichtigung der UPN dürfte sich das Beispiel relativ einfach auch nach INFIX notierenden Assemblern übersetzen lassen, mit einer Ausnahme. Die Übergabe von TRUE an die HighLevel Konstante 'Fehlerflag' ist abhängig von der jeweiligen Implementierung des Gesamtsystems (Forth). Hier muß im Bedarfsfall vor Einsatz der Routine (LABEL CEH.TEMP) eine Kontrolle erfolgen.

Das 'Umbiegen' des Interruptvektors auf eine eigene Routine geschieht einfach dadurch, daß zunächst die Adresse des temporären Handlers an das Register DX übergeben wird. DX soll den Offset der 32-Bit Adresse enthalten, unter welcher die neue Routine für DOS zu finden sein wird. Die Segmentadresse wird in DS übergeben. Da zur Laufzeit von ZF, wie bei den meisten DOS-basierenden F83-Systemen auch, DS und CS stets gleiche Inhalte haben (oder zumindest haben sollen), und weil der Code der neuen Routine im CodeSegment des laufenden ZF's liegt, erübrigt sich prinzipiell die explizite Angabe des DS. Im Beispiel wurde eben darauf verzichtet. Systeme die mit dem Bei-

spiel nicht zurecht kommen, sollten hier zunächst DS auf dem Stack sichern, CS nach DS laden und nach Beendigung der Operation DS restaurieren.

Der temporäre Handler selbst 'macht nicht viel'. Im Beispiel übergibt er, weil hier nicht mehr verlangt wird, lediglich ein TRUE an die HighLevel Konstante "Fehlerflag". Selbstverständlich könnte das "CEH.Temp" alle aus diesem TRUE resultierenden Aktionen selbst übernehmen und steuern. Statt dessen wird die Reaktion auf den fehlgeschlagenen Zugriffsversuch durch eine HighLevel-Definition bestimmt. Dieser Weg wurde einfach deshalb gewählt, weil COLON-Definitionen in 'echten' Applikationen größere Flexibilität ermöglichen. Neben der einfachen Aussage, daß der Zugriffsversuch fehlgeschlagen ist, ließen sich über das Register DI detaillierte Informationen über die Art des Fehlers erfragen. DOS übergibt beim Eintritt in den INT 24h über DI diese detaillierten Informationen, die in einer erweiterten Form eines temporärern Handlers auf dem im Beispiel beschriebenen Wege der HighLevel Ebene bekannt gemacht werden könnten.

Das Installieren des neuen Handlers (Überschreiben des Originals !) geschieht über einen Aufruf der Funktion 25h des ServiceInterrupts 21h. Selbstverständlich könnte dieses Überschreiben auch 'direkt' ausgeführt werden. Die Adresse des Vektors ist sehr einfach berechenbar. Allerdings sollte darauf verzichtet werden, weil direktes Überschreiben zu Inkonsistenzen führen kann, und zwar dann, wenn irgendein bereits laufender Task DOS veranlaßt hat, gerade den INT 24h auszulösen. In den meisten Arbeitsfällen ist dies nicht sehr wahrscheinlich, aber bereits die Arbeit mit DOS's PRINT Utility kann solche Fehlersituationen provozieren.

Deinstallieren des temporären Handlers

Grundsätzlich gilt bei allen Experimenten mit der Interruptvektoren-Tabelle, daß zu verändernde Vektoren zuvor zu sichern sind. Eine Umge-

Listing zu: DOS unter Kontrolle (Friederich Prinz)

```
\ ------------------------------------------------------------------------
\ Beispielprogramm für die Anwendung eines temporär installierten CEH
\ 'Critical Error Handler' in einer Applikation
\
\ Forthgruppe Moers, Friederich Prinz, Juni '93
\                    ZF Forth
\                    MSDOS / PCDOS 3.x bis 5.0
\ ------------------------------------------------------------------------

    CREATE   Joker                              \ 'WildCard' für DOS-DIR
             ASCII A C, ASCII : C, ASCII \ C,   \ A:\*.*0 ( ASCIIZ )
             ASCII * C, ASCII . C, ASCII * C,
                  0 C,

    128 CONSTANT ZFDTA          \ DTA im ProgrammSegmentPräfix von ZF
                                \ (siehe Kurs "Dateihandling" der
                                \ (Forthgruppe Moers)

      0 CONSTANT Fehlerflag     \ ...wird vom temporären INT 24h Handler
                                \ 'beschickt'.

\ ------------------------------------------------------------------------

HEX

LABEL CEH.Temp ( -- )           \ Der eigentliche, temporäre CEH
                DS PUSH         \ Datensegment des orig. INT 24h sichern
        CS PUSH DS POP          \ und auf Datensegment der Applikation
FFFF # ' FehlerFlag 2+ #) MOV   \ setzen, dort ist das Fehlerflag definiert.
                DS POP          \ Datensegment restaurieren,
             3 # AL MOV         \ und dem originalen Handler mitteilen, daß
                  IRET          \ er den aktuellen Funktionsaufruf abbrechen
                                \ soll ( Notiz: ...erst ab DOS 3.x ! )

CODE Set.CEH.New ( -- )         \ INT 24h Handler auf die eigene, temporäre
   CEH.Temp # DX MOV            \ Routine setzen. Da CodeSegment und Daten-
       2524 # AX MOV            \ segment im ZF gleich sind, wird hier auf
              21 INT            \ explizite Umsetzung verzichtet.
NEXT C;


CODE Set.CEH.Org ( -- )         \ INT 24 h Handler wieder auf die originale
              DS PUSH           \ Routine zurücksetzen.
         18 #) DX MOV           \ Eine Kopie des originalen Vektors
         20 #) DS MOV           \ 'kritischer Fehler' steht im PSP
       2524 # AX MOV            \ des Systems. Offs 18 = Offset
              21 INT            \              Offs 20 = Segment
              DS POP            \ ( siehe Beschreibung )
NEXT C;

DECIMAL

\ ------------------------------------------------------------------------

: .Fehler ( -- )
    SAVESCR
    22 10 AT ." +-----------------------------------"
    22 11 AT ." | Achtung - Laufwerk nicht bereit |"
    22 12 AT ." +-----------------------------------+"
    KEY DROP
    RESTSCR ;

: Char? ( b -- b f )            \ Test : ist Byte auf dem Stack ein
    DUP DUP                     \        Zeichen ? ( a..z | A..Z )
    97 122 BETWEEN SWAP
    65  90 BETWEEN OR ;

: .Eintrag ( -- )               \ Verzeichniseintrag ausgeben
    ZFDTA 30 + 12 TYPE CR ;

: MeinDir ( Lwk -- )            \ DIR auf angegebenes Laufwer (default = A:)
    ( Aufrufkonvention : MEINDIR 'char'    --- char = A .. Z | a .. z )
    ( ------------------------------------------------------------- )

    FALSE =: Fehlerflag                     \ ---
```

➪

bung wie ZF, die einen Vektor während ihrer Laufzeit auf eine Adresse im eigenen Arbeitsbereich 'umbiegt', **muß** vor ihrer Beendigung den veränderten Vektor restaurieren. Geschieht dies nicht, führt das in den allermeisten Fällen zu undefinierten Zuständen des Systems - sprich: zum Absturz. Der Grund hierfür ist einleuchtend und recht einfach zu erklären. Wenn eine Umgebung wie ZF, oder eine aus dem ZF heraus gestartete Applikation beendet wird, gibt diese Umgebung, die in sich nichts anderes ist, als jedes 'normale' DOS-Programm auch, ihren Arbeitsspeicher an das Betriebssystem zurück. Wenn der manipulierte Vektor nicht restauriert wurde, zeigt er ab diesem Zeitpunkt auf einen Speicherbereich, der mittlerweile ganz anderen Code oder beliebige Daten enthalten kann. Das Betriebssystem ist inkonsistent geworden ! Die Folge ist, mit hoher Wahrscheinlichkeit, ein Systemabsturz.

Im Beispiel wurde trotzdem darauf verzichtet, den originalen Vektor, sprich: Segment- und Offsetadresse des CEH, vor der Manipulation zu sichern. Dies hat seine Berechtigung darin, daß eben diese Adresse grundsätzlich jedem von DOS gestarteten Programm mitgegeben wird. Im PSP (Programm Segment Präfix) eines jeden von DOS gestarteten Programms wird an der Adresse 18h die Segmentadresse und an 20h der Offset des Originalhandlers vermerkt ! Allein die Tatsache, daß DOS's Entwickler dem CEH hier ganz offensichtlich eine besondere Bedeutung beimessen, zeigt, daß man bei MicroSoft durchaus damit gerechnet hat, daß von den oben beschriebenen Manipulationen reger Gebrauch gemacht wird. Tatsächlich reichen als 'Kindprozesse' gestartete Programme stets den Vektor weiter, den der jeweilige 'Vaterprozess' installiert hat. Deutlich wird dies bei dem Versuch, einen temprären Handler zu installieren und zum Beispiel via einer der SYS Funktionen von ZF (oder auch F-PC) einen kritischen Fehler zu verursachen. Trotz des installierten temporären Handlers werden solche Versuche mit den eingangs beschriebenen originalen Meldungen quittiert. Der originale Handler arbeitet wieder. SYS Aufrufe laden den Kommandointerpreter COMMAND.COM nach, der in seinem eigenen PSP die Adresse des originalen CEH mit sich herum trägt.

Dieses Faktum macht sich das Beispiel zu Nutze und verzichtet, wie bereits beschrieben, darauf, den originalen Vektor zu sichern. Statt dessen werden die zur Restaurierung notwendigen Adressinformationen aus dem PSP des ZF übernommen.

Letzlich sorgt für das 'Zurückschreiben' wieder der Aufruf INT 21h.

❑

Fortsetzung des Listings zu: DOS unter Kontrolle

```
Set.CEH.New                          \ temporären CEH setzen
ZFDTA SET-DTA                        \ DOS die DTA bekannt geben
CR
BL WORD 1+ C@                        \ LWK-Buchstaben ausfiltern
DEPTH 0<> IF Char? IF Joker C!       \ Laufwerksbuchstabe auf dem
                  ELSE DROP          \ Stack, dann an 'Joker'
                  THEN               \ übergeben
          THEN
Joker FINDFIRST  0= IF .Eintrag      \ 1. Suchaufruf in DOS
                    THEN
Fehlerflag IF .Fehler                \ Reaktion auf den tempor.
           THEN                      \ CEH !!!
BEGIN                                \ Alle weiteren Suchaufrufe
  FINDNEXT DUP 0= IF .Eintrag        \ im DOS-Verzeichnis, bis
                  ELSE DROP TRUE     \ 0 signalisiert, daß kein
                  THEN               \ gültiger Eintrag mehr im
UNTIL                                \ Verzeichnis ist.
Set.CEH.Org ;                        \ Originalen INT 24h Handler
                                     \ restaurieren
```

❑

40 MB für 10 DM — Arndt Klingelnberg

Neuheiten zum Forth Controller TDS2020

Nicht ganz, aber: 40 MB in 10 DM. 33 mm (1.3") mißt ein 10 DM-Silberling, und so groß ist hp's KittyHawk mit 20 oder 40 MB. Damit ist die Festplatte fast kleiner als der Processor H8/532 im Sockel. Neben bis zu 512 kB DIP SRAM kann auf den Controller TDS2020 (VD 93 Q4) ein oder zwei PCMCIA Steckkarten Aufsätze oder eben auch eine oder zwei Festplatten (80 MB) aufgesteckt werden. Der Leistungsbedarf beträgt 1.8 W (peak) beim Schreiben von Daten. Beim Datenloggen werden Daten erst ins SRAM gesammelt und dann in größeren Abständen auf die HD übertragen. Dank Schlafmodus beträgt so die durchschnittliche Stromaufnahme unter 4 mA !

Die Innereien der KittyHawk bilden magnetisch gesputterte Glasplatten. Die geringen Massen und ein BeschleunigungsSensor machen die MiniFestplattenEinheit besonders robust. Die kurze Aufwachzeit aus dem Schlafmodus sind die Voraussetzung für eine Batterie-Versorgung.

Forth-mäßig wird die Platte angesteuert wie ein 32bit-addressierter Speicher `E! E@ 2E! EC!` (random-access) oder wie eine andere AusgabeEinheit (statt serielle Schnittstelle oder LCD) über `EMIT` oder `TYPE` oder . (sequentiell). Auch ein klassisches BlockInterface samt Editor ist vorhanden. LowPower und eine übersichtliche Programmierung waren die Entwicklungsziele. In 100 ms steht ein beliebiger 1024 Byte Block im Zugriff, bei sequentiellem Zugriff sind es 47 ms. Bei 40 MB im direkten Zugriff mit 32 bit Adressen, und zwar flat, träumt ein DOS-PC-ler, der Segmentierung und damit Rechnerei gewohnt ist. Und welches PC-Forth-System erlaubt es schon, die Festplatte als (virtuellen) Speicher für Store und Fetch zu verwenden.

Der PCMCIA-Adapter zum TDS2020 verfügt nun über einen eigenen 12V Generator für FLASH-Eprom-Karten, was den Betrieb vereinfacht. Auch hier kann der SRAM -DIP auf der Grundplatine zum Stromsparen und schnellen Schreiben genutzt werden.

Trigangle, London weist ferner darauf hin, daß nun eine optimierte (Cooley-Tukey) FFT Funktion mit unterschiedlichen Fenstern verfügbar ist. Gegenüber der Demo-FFT läuft diese Version bis zu 100-mal schneller und erlaubt bis zu 2^{14} Punkte ! Die Grund-FFT Operation benötigt für 256 Punkte 200 ms. (zu FFT siehe VD 93 Q3 und VD 94 Q1). Besondere optimierte Versionen in LowLevel können entwickelt werden.

1993 October führte Triangle auf der NEC Messe (GB) den TDS2020 im Mulit-Tasking vor. Drei Programme teilten sich ein Grafik-LCD, der 4. Prozess war der interaktive Forth Task über die serielle Schnittstelle: Debugging oder 'nur' ParamterÜbergabe wärend des Betriebes.

PID-Regler im Überblick

von Rafael Deliano
Steinbergerstr. 37, 82110 Germering

Kurze Einführung in die Grundlagen und digitale Realisierung. Vorzüge und Einschränkungen der Lösung mit Mikroprozessoren.

Als Beispiel für einen einfachen Regelkreis sei hier ein Gleichstrommotor, dessen Drehzahl kontrolliert werden soll, als Regelstrecke dargestellt (Bild1). Normalerweise werden derartige Regler als Analogschaltung realisiert, weshalb dieser hier als Ausgangspunkt dient. Am Soll-Eingang (w) des Reglers wird über ein Poti die gewünschte Geschwindigkeit als Gleichspannung vorgegeben. Am Ist-Eingang (x) liegt eine Spannung aus einem Sensor an, die der Drehzahl proportional ist. Beide Signale werden über Subtraktion verknüpft. Ist das resultierende Signal 0 Volt, läuft der Motor mit der gewünschten Drehzahl. Bei falscher Drehzahl entsteht eine Spannung, deren Polarität anzeigt, ob zu schnell oder zu langsam. Der Betrag ist der der Abweichung proportional. Mit dieser Fehlerspannung wird nun der eigentliche PID-Kern angesteuert. An seinem Ausgang befindet sich ein Leistungsverstärker, der den hohen Strom für den Motor abgeben kann. Der PID-Regler wird mit 3 Parametern eingestellt (Bild2):

P: das Proportionalglied

Dies ist ein reiner Verstärker. Ein Operationsverstärker z.B. funktioniert wie ein reiner P-Regler, der durch eingebaute extrem hohe Verstärkung seiner Regelstrecke die Rückkopplung fast völlig ausregelt. Funktioniert beim OP (Operationsverstärker) wunderbar, weil man sich dort die passende Rückkopplung aussuchen kann.

Regelstrecken wie ein Motor verhalten sich jedoch wie in Serie geschaltete Tiefpässe und haben damit eine starke Phasendrehung. Kombiniert mit zu hoher Verstärkung würde das zu Schwingung führen. Die Verstärkung muß deshalb relativ niederig bleiben. Damit tritt der Pferdefuß des P-Reglers besonders deutlich hervor: er kann den Fehler nur "fast" ausregeln. Er braucht einen Restfehler, den er verstärken kann, damit überhaupt ein Korrektursignal am Ausgang entsteht. Dieser Restfehler ist beim OP winzig, weil die Verstärkung hoch ist. Bei niederiger Verstärkung jedoch nicht mehr vernachläßigbar.

I: der Integrator

Er summiert aus der Größe des Fehlersignals und der Dauer, die es anliegt, das Korrektursignal. Je länger das Fehlersignal anliegt, desto größer wird das Korrektursignal. Der I-Regler reagiert zwar langsam, aber er kann den Fehler voll eliminieren.

D: der Differenzierer

Er bildet die Differenz des Eingangssignals über die Zeit. Der D-Anteil beschleunigt die Regelung. Ein Fehlersprung am Eingang wird mit einem Korrekturimpuls am Ausgang beantwortet. Regelstrecken, die verzögert ansprechen, werden durch dieses kurzzeitige Übersteuern stärker angeregt, und der Ausgleich kann so schneller erreicht werden. Auf eine langsamen Änderung des Fehlersignals wird dieser Parameter nicht ansprechen.

Nicht jede Regelung wird mit allen 3 Parametern optimiert.

Die Regelstrecke bestimmt die Komplexität des Reglers. Meist genügt ein P-, I-, PI-, oder PD-Regler. PID ist weder überall notwendig, noch günstig. Praktisch wird jedoch der Regler universell als PID ausgelegt und im Einsatzfall werden die nichtbenötigten Parameter auf Null eingestellt.

Stichworte

Regeln,
PID-Regler,
FIR-Filter

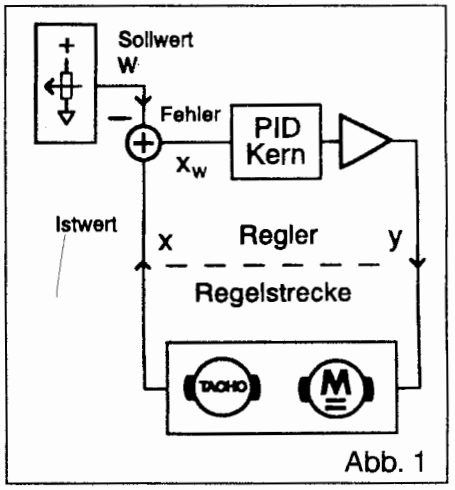


Abb. 1

Die wichtigsten Regler mit OPs realisiert und ihre Sprungantwort

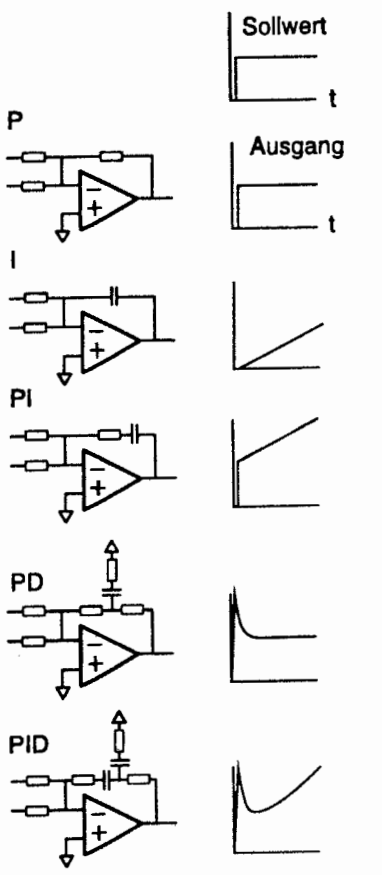


Abb. 2

Die digitale Lösung

Alternativ zur analogen Realisierung mit OPs ist auch eine digitale Lösung auf einem Mikroprozessor möglich. Dem Vorteil der Flexibiliät bei Einstellung und Variation der Parameter stehen dann Einschränkungen bei Dynamik und Geschwindigkeit entgegen. Soweit die Sensoren und Stellglieder nicht direkt digitale Siganle verarbeiten, werden für den Mikroprozessor A/D- und D/A-Wandler als analoge Schnittstellen benötigt. Wie das Blockschaltbild (Bild3) zeigt, ist die Realisierung recht simpel. Die Z-1-Blöcke bedeuten nur, daß hier eine Speicherstelle verbraucht wird, in der das augenblickliche Signal auf den nächsten Abtastzeitpunkt weitergereicht wird.

Das Listing zeigt ein äquivalentes FORTH-Programm. Die Abtastrate muß konstant, z.B. 1msec sein. Deshalb wird im Beispiel immer ein Timer gestartet. Die Bestimmung der Koeffizienten ist ein eigenes, komplexes Thema, auf das hier deshalb nicht eingegangen werden kann.

Neben der hier dargestellten groben Rechteckapproximation gibt es für I und D auch die Trapez-Approximation, die für niederige Abtastfrequenzen besser interpoliert. Sie hat jedoch wenig praktische Bedeutung, da eine isolierte Herabsetzung der Abtastrate im PID-Algorithmus nicht sinnvoll ist. Die A/D- und D/A-Wandler brauchen hohe Abtastraten, um ein kontinuierliches Signal zu gewährleisten.

Typisch sollte die Abtastfrequenz 5-10x höher als die höchste auftretende Frequenz im Regelkreis sein. Da 8-Bit-CPUs nur mit wenigen kHz abtasten können, eignen sie sich nur für langsame mechanische Prozesse, die Zeitkonstanten im Bereich von 100msec haben. Signalprozessoren sind schneller, aber deutlich teuerer. Analoge Regler hingegen arbeiten problemlos bis einige 10kHz.

Das nächste Problem ist die Dynamik, also das Verhältnis zwischen maximalen und minimalen Eingangssignalen. Forth kann zwar mit 16 Bit rechnen, die Wandler erreichen aber meist nur 8 - 12 Bit Auflösung. Insbesondere ein I-Regler kann nicht genauer als seine Wandler ausregeln.

Die Dynamik eines rauscharmen OPs mit 30V-Versorgungsspannung ist mit Wandlern ohnehin kaum erreichbar. Allerdings relativiert sich dieser Vorteil der analogen Schaltung, weil viele Sensoren weniger als 8-Bit Auflösung haben. Gleiches gilt für die Ausgangsstufe. Ein kleiner DC-Motor kann mit maximal einigen 100mA angesteuert werden. Wird mit 1mA aber ohnehin nichtmehr laufen, weil man damit die Reibung des Getriebes nicht mehr überwindet.

Eindeutiger im Vorteil ist die digitale Lösung, wenn flexible Einstellung der Koeffizienten nötig ist. Man kann für verschiedene Betriebszustände unterschiedliche Datensätze laden. Z.B. für Anlaufphase, Gleichlauf, Abbremsen. Man kann verhindern, daß unerlaubte Kombinationen der PID-Parameter eingestellt werden, die zu Schwingung führen. Man kann letztendlich selbstlernende Regler implementieren. Bei analogen Schaltungen werden P, I und D mit Potis eingestellt und man hat deshalb alle diese Möglichkeiten nicht.

Digitaler PID-Kern

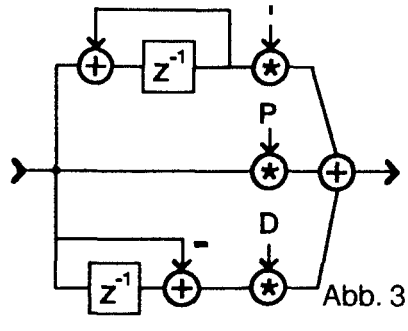


Abb. 3

Frequenzgang des Reglers

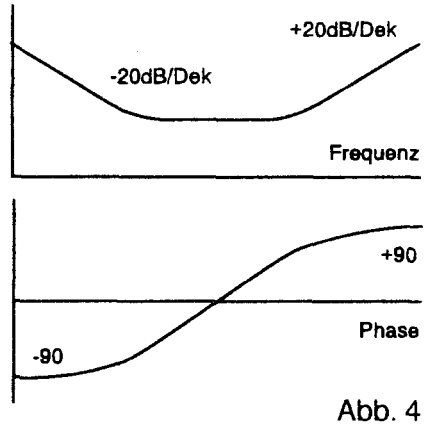


Abb. 4

Die Filtersichtweise

Der PID-Kern kann auch als Filter interpretiert werden (Bild4). Er ist nicht viel mehr als die Kombination von Grundverstärkung (P), 1poligem Tiefpaß (I) und 1poligem Hochpaß (D). Konsequenterweise bietet sich die Realisierung als digitaler Filter an. Diese verdrängen deshalb die direkte Implementierung des PID-Reglers immer mehr. Es ist heute aber vielfach noch erforderlich, daß die Koeffizienten erst beim Aufstellen des Geräts vor Ort eingestellt werden. Die Wirkung der einzelnen PID-Parameter ist anschaulich und das technische Personal, das die Einstellung durchführt, kennt sie. Zumindest für die Benutzeroberfläche wird das PID- Modell einstweilen erhalten bleiben. Danach wird allerdings in Zukunft oft die Umrechnung in FIR-Koeffizienten erfolgen und die eigentliche Berechnung als FIR-Filter durchgeführt werden.

❑

Der PID-Regler in Forth

```
x CONSTANT SOLL-WERT
x CONSTANT P-WERT
x CONSTANT I-WERT
x CONSTANT D-WERT

VARIABLE I-RAM
VARIABLE D-RAM

: P-KERN \ ( N1 -- N2 )
  P-WERT *
  ;
: I-KERN \ ( N1 -- N2 )
  I-RAM @ + DUP I-RAM ! I-WERT *
  ;
: D-KERN \ ( N1 -- N2 )
  D-RAM @ OVER D-RAM ! SWAP - D-WERT *
;
: PID-KERN \ ( Error -- D/A-Data )
  DUP P-KERN OVER I-KERN + SWAP D-KERN +
  ;
: PID-REGLER \ ( -- )
  A/D @ \ Istwert
  SOLL-WERT -
  PID-KERN D/A !
  ;
: RUN \ ( -- )
  0 D-RAM !
  0 P-RAM !
  BEGIN
    START-TIMER
    PID-REGLER
    WAIT-END-OF-TIMER
  AGAIN
  ;
```

❑

Aus Alt mach Neu

von Wolfgang Schemmert
Strahlenberger Str.123, 63067 Offenbach Tel 069-800 12 08, Fax 069-82 59 57

Redefiniton von Worten tief unten im Kernel

Inkrementelles Kompilieren ist eine schöne Sache. Die (all)gemeine Forth-Logik setzt dabei stillschweigend voraus, daß nach dem Bottom-Up Prinzip alle irgendwann getesteten und für OK befundenen Worte auch wirklich richtig sind. Oder: während des Programmierens stellt sich heraus, daß der ursprüngliche Faktorisierungsansatz doch nicht so optimal war und dieses oder jenes Wort besser ein anderes Stackbild erzeugen sollte. Um diese Dinge tatsächlich in einer experimentellen Arbeitsweise tief unten im fertigen Kompilat nachzubessern und ohne Risiko auch wieder verwerfen zu können, wurden die Worte RE: und UNDO-RE geschrieben (Das Colon am Namensende von RE: soll die Verwandtschaft zur normalen Forth-Kompilation andeuten).

RE: erzeugt keinen neuen Header, sondern sucht stattdessen nach dem aufgeführten Namens-String. In diesen bereits existierenden Header wird statt der alten "ca" ein Zeiger auf den neuen Code eingehängt. (Klappt natürlich nur, wenn der alte Code im RAM ist !) Die alte "ca" wird - leicht wieder zutage förderbar - vor dem neuen Code ins Dictionary eingetragen. Der neue Code wird ab dem bisherigen HERE + 2 Bytes konventionell ins Dictionary kompiliert. Es ist möglich, beliebig viele RE: hintereinander auszuführen, das neueste ist danach immer aktiv. (Verbrutzelt leider jedesmal Speicher.)

Mit UNDO-RE wird wieder der nächst ältere Code aktiviert. Der damit gelöschte Code ist noch im Speicher, auf ihn kann allerdings nur noch per Vektorverbiegung zugegriffen werden. Bei mehrmaliger Ausführung von UNDO-RE ist schließlich irgendwann wieder der Originalcode aktiv. Für Schnellzugriffe wird das letzte neu erzeugte RE: in der Variablen NEW gespeichert. RENEW aktiviert die neueste redefinierte Version des angesprochenen Wortes und mit UNDO-RE kann die Kette dann erneut aufgedröselt werden. RENEW ist ganz praktisch, um zwischen einer alten und einer neuen Version Hin und her zu schalten. Aber Vorsicht! RENEW ist nicht völlig narrensicher! Es produziert fehlerhafte Resultate, wenn an mehreren Worten gleichzeitig redefiniert wird oder wenn nicht redefinierte Worte versehentlich RENEWt werden !

Normalerweise können die RE: definitionen mit ; beendet werden. Das Wort ;RE ist nur für einen fast akademischen Sonderfall: nämlich wenn soeben das CURRENT Verzeichnis geändert wurde und als erstes eine Redefiniton gemacht werden soll. Was dann passiert, ist abhängig von der Implementation des verwendeten Forth-Kerns.

Der Code läuft auf einem von mir modifizierten, direkt gefädelten eForth. Für andere Forth Systeme ist möglicherweise ein anderer Offset zwischen "na" und dem Zeiger auf "ca" (hier 4 Bytes oberhalb von "na") anzusetzen und die Zeigerhandhabe ist der Fädeltechnik anzupassen. Bei meiner Modifikation sind Header und Code ineinander verschachtelt. Bei Kernen mit getrennten Headern ist das Wiederfinden des Orignal-Codes aufwendiger.

❑

Stichworte
Redefinition

Listing zu: Aus Alt mach Neu (W. Schemmert) REDEFI.SEQ

```
VARIABLE NEW

: RE:  ( -- ; <string> )
   token name? ?DUP          \ token ( -- a ; <string> )   name? ( a -- ca na | a F)
                             \ also: "token" parst den nachfolgenden Wortnamen nach
                             \ Adresse a, "name?" erzeugt daraus "na" und "ca",
                             \ falls nicht im Dictionary gefunden stattdessen
                             \ FALSE Flag on TOS.
   IF
      CELL- CELL- SWAP       \ ( -- CodeField alt_ca )
                             \ Achtung, Offset implementationsabhaengig !
      HERE ALIGNED           \ DP sicherheitshalber auf gerade Adresse setzen

      DUP CELL+ DP !         \ update DP und rueckspeichern
      !                      \ alte ca an vorherigen DP wegen UNDO speichern
      HERE DUP NEW !         \ neue ca merken fuer RENEW
      SWAP !                 \ und in alten Header eintragen
      doLIT doLIST call,
                             \ doLIST kompilieren. doLIST ist das "secondary
                             \ preamble runtime primitive", in den meisten Forth
                             \ Systemen irrefuehrend mit DOCOL bezeichnet.
      ]                      \ ... und nun den aeusseren Interpreter auf
                             \ normale Kompilation umschalten.
   ELSE ABORT" name" THEN
;

: ;RE  ( -- )
               \ Normalerweise kann man die Re-Definition mit Semikolon
               \ beenden. Es gibt einen Sonderfall, wo dies zum Fehler
               \ fuehrt: Wenn soeben CURRENT geaendert wurde und dann als
               \ erste Definition eine Re-Definition durchgefuehrt wird.
               \ Dann MUSS mit ;RE abgeschlossen werden !!!
   COMPILE exit
   [COMPILE] [         \ [ ist immediate, soll in ;RE kompiliert werden.
;  IMMEDIATE RESTRICT
```

Moderne Betriebssysteme

aus Sicht eines Forthers

von Arndt Klingelnberg
Straßburger Straße 12, 52477 Alsdorf, Tel.: 02404 - 61648, Fax: 02404 - 63039

Es hat mich als F-PC-Nutzer einige Zeit gekostet, zu erfahren, was denn nun ein MSdos v.6.0 oder OS/2 bringen kann. Ich möchte Ihnen hier einige Erfahrungen mit Betriebssystemen vorstellen.

Einmal erlaubt MSdos ab v.5.0 recht sinnvoll, memory im UpperMemoryBlock zu allokieren; z.B. ein TSR-Programm in sinnvolle Bereiche zu schieben. DRdos (nun: NOVELL) kann das auch, teilweise sogar besser, aber leider eben anders. Auf NovellDOS V.7.0 warte ich noch.

Da ich recht aufwendig F-PC metacompiliere mit der Erstellung eines SIZE-segmentes und einer Label-Liste, was beides später leichtere Analysen, einfachere Patches, eben mehr Info bringt, benötige ich 520kB oder sogar 540 kB freien Speicher. Gelang mir das früher fast nur durch Nutzung des Graphic-VideoBereiches, so beschert mir nun msDOS v.6.0 trotz Netzwerktreibern knapp 590kB, auf einer meiner Maschinen sogar 610kB. Letzteres schaffte MEMMAKER, das eben auch solche Teile des Netzwerkes nach oben (UMB) legte, wo ich bei eigenen Versuchen lediglich den Griff zum 'little red handle' und mich in Geduld übte.

Auch kam ein interessanter Quirk zutage. Laden Sie 'mal den Norton Commander high. Jetzt Vorsicht: *.COM funktioniert meist nicht mehr, es sei denn, sie haben oben 'rum 64kB frei. Aber *.exe Programme, zumal solche wie F-PC, die aus vielen einzelnen allocierten Teilen bestehen, laden sich teilweise dort oben hin und schaffen unten noch mehr Platz.

Mit OS/2 hat man natürlich problemlos 'beliebig' viel Platz; kein Problem mit der luxuriös-informativen Variante der MetaKompilierung. Dann können Sie auch gleichzeitig mehrere F-PC laufen lassen. Nun, nicht jeder will Original und 'Fälschung' gleichzeitig laufen lassen und das auch noch in älterer und letzter Version, aber z.B. zweimal die Fälschung, weil in einem ALLWORDS gerade eine Liste ALLER Worte zusammenstellt, da muß man/fra nicht mehr zwischendruch Tee trinken gehen, sondern arbeitet mit dem zweiten. Und stürzt eines ab, ... so macht man/fra halt auf dem nächsten weiter.

Aber eines würde mich interessieren, OS/2-ler und Windows-ler sind hier angesprochen: Sollte F-PC auch Rücksprünge rund um die Taste 'F4' erlauben und das eben auch per Maus oben links? Oder ist oben rechts tragbar und F10 wie üblich DOS-mäßig?

❑

Stichworte
MS DOS
OS/2
Betriebssysteme

Fortsetzung des Listing zu: Aus Alt mach Neu

```
: UNDO-RE  ( -- ; <string> )
   token name? ?DUP \ Kommentar siehe oben
   IF
       2DUP DUP DUP C@ + 5 +
       WITHIN          \ im Header eingetragene ca pruefen:
                       \ na < ca < na+count+Reserve ?
                       \ = Test auf Originalcode als UNDO-Limit
                       \ (funktioniert nur wenn Code und Header geschachtelt)
       IF
           ABORT"  can't UNDO original"
       ELSE
           SWAP CELL- @ SWAP
           CELL- CELL- !   \ alte ca in Header rueckspeichern
       THEN
   ELSE ABORT"  name" THEN
;



: RENEW  ( -- ; <STRING> )
   token name? ?DUP \ Kommentar siehe oben
   IF
       SWAP DROP    \ ca vergessen
       NEW @        \ juengste Redefinition
       SWAP CELL- CELL- !  \ eintragen
   ELSE ABORT"  name" THEN
;
```

❑

Unzufrieden

Leserbrief von Rafael Deliano,
Steinbergerstr. 37, 82110 Germering

Ich bin, wie so viele, bei der e.V. 1989 Mitglied geworden. Vor dem Hintergrund der damaligen Novix/RTX-Euphorie gab es die Hoffnung, daß der Verein der Hobbyismus-Ecke entwachsen würde und sich zu einem Forum für "ernsthafte", industrielle Anwendung entwickelt. Die damalige Zusammensetzung des Direktoriums sprach dafür. Auch das Engagement von Aumiller/Luda, die der Gestaltung der VD eine Qualität gaben, die sie vorher nicht hatte und jetzt erst langsam wieder erreicht. Leider muß man heute sagen, daß der Sprung nach vorn nicht gelungen ist. Es wurde nur, analog zur Entwicklung auf Computerseite, volksFORTH durch F-PC ersetzt. Was geblieben ist, ist eine höhere Zahl von Mitgliedern und eine aufwendig gemachte VD. Und dabei liegt auch ein Problem: bei konventionellen Zeitschriften ist das Verhältnis von redaktionellem Anteil zu Werbung ca. 40 zu 60%, wobei die Einnahmen aus Anzeigen gegenüber denen aus Abonnement meist dominieren. Der Kostendeckungsgrad aus Anzeigen ist bei der VD heute vernachläßigbar. Das Hochglanzformat der VD setzt aber Einahmen aus Werbung voraus, oder es treibt den Mitgliedsbeitrag in unrealistische Höhen und dominiert das Budget der e.V. so, daß andere Aktivitäten ausgeschlossen sind. Letzteres ist derzeit der Fall. Die zwingende Folge ist, daß für die e.V. "viele" Mitglieder nicht mehr nur erfreulich, sondern inzwischen lebensnotwendig sind, um die Fixkosten jeder VD durch Stückzahlen amortisieren zu können. Das spricht für den Wachstumsmarkt Hobbyismus, seit immer mehr PCs an Privathaushalte verkauft werden.

Dort sind viele neue Mitglieder, allerdings mit minimaler Kaufkraft, d.h. reine PD-Benutzer. Keine Zielgruppe für Anzeigen also, aber die e.V. kann sich hier durch Mitgliedsbeiträge finanzieren. Diese Mitglieder werden jedoch keine dreistelligen Beiträge akzeptieren wie sie bald nötig werden. Und im Verhältnis zu ihrer großen Zahl finden sich bei den "Uneigennützigen" deutlich weniger Aktive als bei den "Eigennützigen". Die VD (und e.V.) muß von den Mitgliedern nicht nur bezahlt, sondern auch gemacht werden. Der Mitgliedsbeitrag bezahlt keine Redaktion die Artikel recherchiert und schreibt. Es ist deshalb nicht möglich sich einfach zurückzulehnen und zu warten, bis die VD eines Tages im Briefkasten liegt. Leider ist das nicht allen Lesern klar. Für sie ist die VD eine Zeitschrift wie jede die sie am Kiosk bekommen (höchstens etwas teuerer). Und sind der sicheren Überzeugung, daß sie für ihr Geld etwas verlangen können. Augenblicklich klebt die e.V. jedenfalls am Hobbyismus. Derzeit vielleicht die einzig mögliche Linie, aber langfristig sicher ein Problem.

Bei den industriellen Anwendern sieht die Situation ungünstig aus: wenige Mitglieder, eventuell Einnahmen aus Werbung. Aber die auch erst, wenn sich ein glaubwürdiger Markt für FORTH in Deutschland entwickelt hat. Also lange nicht. Der Anteil einschlägiger Mitglieder in der e.V. beträgt ca. noch 5- 10%, sinkende Tendenz. Wer glaubt sie fühlen sich in diesem Verein wohl, irrt.

Die wesentlichen Probleme von "ernsthaftem" FORTH kann man sicherlich nicht dem Verein anlasten. Da wären das mangelnde Angebot und das oft unzureichende Preis/ Leistungs- verhältnis bei den kommerziellen Produkten. Das Mißtrauen der Anbieter untereinander, das wirksame Kooperation ausschließt. Technische Probleme im wichtigen Bereich Tischcomputer, wie weiter, linearer Adreßraum der von den traditionellen 16-Bit FORTHs nicht bewältigt wird. Sowie registerorientierte CPUs die auf andere Programmiersprachen optimiert sind. Ferner ein natürlicher Konzentrationsprozeß am Markt, der erst Pascal und dann C zugute kam. Es gibt jedoch auch ein unterschwelliges Imageproblem in der Industrie. Nämlich: "FORTH ist (lausige) PublicDomain-Software für (verschrobene) Hobbyisten." Spätestens wenn ein Kunde ein Exemplar der VD in die Hand bekommt, ist für ihn dieser Verdacht zur Gewissheit geworden. Insofern sehe ich (und wohl noch andere Betroffene) das halbherzige Schielen der e.V. Richtung Industrie mit mehr als gemischten Gefühlen. Wenn in der VD einschlägige Artikel erscheinen, ist das harmlos, weil das Blatt in der Industrie nicht gelesen wird. Das Agieren der e.V. auf der "Echtzeit" ist schon etwas gefährlicher. Zwar wissen die Besucher dann, daß es FORTH gibt, aber welchen Eindruck sie durch die e.V. von FORTH dabei bekommen ist eine andere Sache. Bisher sind die besten Kunden die, die nichts von FORTH wissen, also keine Vorurteile haben. Nicht nur die vielen FORTHs sind also inkompatibel, auch die Anwendergruppen sind es. Der Anspruch der e.V. alle vertreten zu wollen (und überall ein Paar Mark Mitgliedsbeitrag einzusammeln) ist zwar wohlmeinend, aber derzeit unrealistisch. Dazu müßte sie ihr Angebot und ihr Erscheinungsbild differenzieren. Also neben einer (PD-)VD andere Publikationen für andere Zielgruppen herausbringen.

NEWS NEWS NEWS NEWS NEWS NEWS NEWS

von Friederich Prinz, Moers

Mike Warot hat zum FORTH/2 inzwischen eine Version 0.39 Beta freigegeben, die den Stand vom 31.01.'94 enthält. Darin sind einige Dinge ein wenig anders, als das in der Version 0.29 noch der Fall war. Aber die Unterschiede sind nicht so furchtbar groß. Ich mußte halt ein paar 'Nacharbeiten' ansetzen, um die hier vorgenommenen 'Anbindungen' auch weiterhin aufnehmen zu können. Mike will mit dem FORTH/2 zu ANSI kompatibel werden und bleiben. Ich versuche nach Kräften, ihm das auszureden. Mit ANSI geht Forth m.E. einmal mehr den Bach der Popularität 'runter. Meines Wissens sieht ANSI nichts von dem für Forth vor, was Forth unter OS/2 erst 'richtig interessant' macht. Dann können wir gleich bei DOS und Mikrocontrollern bleiben :-(

Einen ähnlichen Weg geht wohl auch *Dirk Zoller*, wenn ich meine noch spärlichen Informationen richtig interpretiere. Dirk Zoller hat mit C ein zu ANSI kompatibles Forth entwickelt, das den Namen PFE trägt. PFE steht für 'Portable Forth Enviroment' und soll, wie der Name schon sagt, auf jedes System portierbar sein, dem ein C-Compiler zur Verfügung steht. Deshalb gehören zum 'Lieferumfang' von PFE vor allem die umfangreichen Sourcen des Systems. Ich habe von *Zbigniew Diaczyszyn* eine Version des PFE zur Verfügung gestellt bekommen. Diese Version enthält eine unter DOS (habe ich nicht ausprobiert) und OS/2 (funktioniert!) startbare PFE.EXE. Das ist das o.a. Enviroment. Es macht schon äußerlich einen sehr 'ordentlichen' Eindruck. Technisch kann ich noch gar nichts dazu sagen. Ich hab's gerade einmal aufgerufen und will mich in den nächsten Tagen ein wenig mehr damit befassen :-)

Aber wenn das PFE, weil kompatibel zu ANSI. auf 'echtes Multitasking' , auf GUI von OS/2 und auf Multithreading verzichten soll, dann wird das kaum zum System meiner Wahl. Ich kann noch nichts Näheres sagen, im Gegensatz zu *Zbigniew*, der mit dem PFE recht einverstanden ist.

(Fortsetzung: "Messen via CENTRONICS" von Seite 24)

Pollinggesteuerte Meßwerterfassung

Eine gute Möglichkeit, die Interruptroutine zu testen, stellt ihr Aufruf über den Softwareinterrupt INT_15 dar. Dadurch läuft man nicht gleich in Gefahr, daß der Rechner nach der Initialisierung des Hardwareinterrupts auf Grund eines Programmierfehlers ins Nirwana springt und man weiß nicht warum.

Das Wort `DI` (wie disable interrupt) dient diesem Zweck. Es sperrt das entsprechende Bit im CENTRONICS-Controlregister und im Enableregister des PIC und speichert die CFA der pollinggesteuerten Meßwerterfassung `POLLING-ADC@` nach `ADC@`.

Zur Ablaufsynchronisation muß der Softwareaufruf der Interruptroutine mit dem /DAV-Signal synchronisiert werden. Ist dieser erfolgt, ruft man einfach `INT-ADC@` auf. Ansonsten kann man alle Teile des Programms beibehalten. Allerdings wird ein Überlauffehler logischerweise nicht erkannt.

Beispielprogramm

Mit Hilfe von `SET-DELAY` kann man interaktiv mit Hilfe eines Oszilloskops die optimale Taktfrequenz des Schiebetaktes einstellen.

Im Beispielprogramm

```
ADCTEST ( # — )
```

werden # Meßwerte eingelesen und zur Mittelwertbildung aufsummiert. Anschließend wird das Ergebnis auf dem Bildschirm angezeigt.

❑

Inserentenverzeichnis

Forth, Forth und Forth (akg)

```
: num>ASCII ( n1—n2 )
   DUP [CHAR] 0 + SWAP
   9 > IF  7 + THEN ;

: num>ASCII ( n1—n2 )
   DUP>R '0' + R> 9 >
   IF  7 + THEN ;

: num>ASCII ( n1—n2 )
   DUP>R '0' + R> 9 >
   7 AND + ;
```

Verwandelt eine Stelle einer Zahl (0..9 , 0..15 , 0..35) in ein ASCII-Zeichen (0..9 , 0..F , 0..Z). Die oberste Definition verwendet straight-forward portables Forth. Die letzte Zeile nutzt sowohl F-PC Spezialitäten (dup>r 'a') wie auch Forth83/ANS-Interna (TRUE==-1 für zweierkomplement-Machinen), letzteres nach 'Thinking Forth' von *Leo Broodie*. Die mittlere Version ist für mich (akg) der Kompromiss zwischen Begreifbarkeit und Schnelligkeit.

[CHAR] (ANS) kann durch ASCII (Forth83) ersetzt werden.

❑

Die ultimative CASE-Anweisung UCASE.SEQ

```
\ Wil Baden, "The Ultimate CASE Statement"
\   in: Dr. Dobb's Toolbook of Forth  Vol. II,
\       editors of Dr. Dobb's Journal, pp. 100-105, M&T, 1987.

FORTH DEFINITIONS

' DUP ALIAS CASE
: OF  ( n ? — )  [COMPILE] IF  COMPILE DROP ; IMMEDIATE
: =OR  ( n1 ?1 n2 — n1 2 )  2 PICK =  OR ;

\S
\ in F-PC: BREAK performs EXIT THEN
: CRAPS  ( n — )
        CASE 7 = 11 =OR          OF  ." You win" BREAK
        CASE 2 3 BETWEEN  12 =OR  OF  ." You lose" BREAK
        . ." is your point" ;

: WHATEVER  ( n — )
    CASE 0=
        OF  ." Zero"                BREAK
    CASE 0<
        OF  ." Negative"            BREAK
    CASE  DUP 1- AND 0=
        OF  ." Power of 2"          BREAK
    CASE  '0' '9' BETWEEN
        OF  ." Digit"               BREAK
    CASE ',' '/' BETWEEN  :' = OR
        OF  ." Punctuation ,-./:"   BREAK
    DROP ." Whatever" ;
```

Diese CASE-Anweisung wird im Quellcode zu FPFFT von Bernd Beuster verwendet

❑

In dem gleichen Maße, in dem die Komplexität eines Systems steigt, vermindert sich unsere Fähigkeit, präzise und zugleich signifikante Aussagen über sein Verhalten zu machen. Ab einer gewissen Stufe sind Präzision und Signifikanz (oder Relevanz) fast sich gegenseitig ausschließende Charakteristika.

InkompatibilitätsPrinzip 1965,
Lotfi A. Zadeh (der Fuzzy-Papst)

Wer aber nicht eine Million Leser erwartet, sollte keine Zeile schreiben.

J.W. Goethe
(zu Eckermann, 1825may12)

Termine akg

INTERDECK'94 rund um DEC
1994apr12—14,
NancyHalle, Karlsruhe

Hannover (Industrie) Messe
1994apr20—27
Hannover

Forth Kongress94 / Mitgliedsversammlung
1994mar//apr, Malente

Echtzeit/iNET'94 mit ProgrammierWettbewerb
1994jun14—16,
CCH, Hamburg

Computer Graphics Leipzig
1994aug30—sep02,
Leipzig

BIK'94 Büro- und Kommunikationstechnik
1994aug30—sep03,
Leipzig

MessComp
1994sep13—15, Wiesbaden

Photokina mit HiFiCologne
1994sep21—27,
Köln

IPC & SPS
1994oct11—13,
Sindelfingen

Systec
1994oct25—28,
München

Computer & Communication
1994oct27—30,
Bangkok

Electronica
1994nov08—12,
München

Hobby und Electronik
1994nov24—27,
Stuttgart

(bitte geben Sie uns ihren Input zu für Forthler sinnvollen Veranstaltungen: Forth-Kurse, Forth-Anwendungen, Forth-Kunden, Forth-Konkurrenten, Hardware für Forthler)

Forth-Gruppen regional

Berlin
Claus Vogt
Tel. 0+30 - 2 16 89 38 p
Treffen: nach Absprache

Rhein-Ruhr
Jörg Plewe
Tel. 0+208 - 49 70 68 p
Treffen: jeden 1. Samstag im Monat im S-Bahnhof Derendorf
Münsterstr. 199, Düsseldorf

Moers
Friederich Prinz
Tel. 0+2841 - 5 83 98 p
Treffen: jeden Samstag 14:00
Arbeitslosenzentrum, Donaustr. 1
Moers

Darmstadt
Andreas Soeder
Tel. 0+6257 - 27 44

Mannheim
Thomas Prinz
Tel. 0+6271 - 28 30 p
Ewald Rieger
Tel. 0+6239 - 86 32 p
Treffen: jeden 1. Mittwoch im Monat, Vereinslokal Segelverein Mannheim e.V., Flugplatz Mannheim-Neuostheim

µP - Controller Verleih

Rafael Deliano
Steinbergstr. 37,
82110 Germering
Tel.:0+89 - 8 41 83 17

Gruppengründungen, Kontakte

Regional

Stuttgart
Wolf-Helge Neumann
Tel. 0+711- 8 87 26 38 p

Fachbezogen

8051 ... (Forth statt Basic,e-FORTH)
Thomas Prinz
Tel. 0+6271 - 28 30 p

Forth-Hilfe für Ratsuchende:

Forth allgemein

Jörg Plewe
Tel. 0+208 - 49 70 68 p
plewe@mpi-dortmund.mpg.de

Karl Schroer
Tel. 0+2845 - 2 89 51 p

Jörg Staben
Tel. 0+2103 - 24 06 09 p

Spezielle Fachgebiete

Anfänger und Wiedereinsteiger
Gerd Limbach
Tel. 0+2051 - 25 51 12 p
Mo. + Di. 20:00 - 22:00

32FORTH (Atari)
Rainer Aumiller
Tel. 0+89 - 6 70 83 55 gp

FORTHchips (FRP1600, RTX, Novix ...)
Klaus Schleisiek-Kern
Tel. 0+40 - 2 20 25 67 p

F-PC & tCOM, ASYST (Meßtechnik), embedded controller (H8/5xx//TDS2020, 8051 ... eFORTH...), FUZZY
Arndt Klingelnberg
Tel. 0+2404 - 6 16 48 agp

Gleitkomma-Arithmetik
Andreas Döring
Tel. 0+721- 59 39 35 p

HS/Forth (Harvard Softworks)
Wigand Gawenda
Tel. 0+30- 44 69 41 p

KI (Künstliche Intelligenz), OOF (Object Oriented Forth)
Ulrich Hoffmann
Tel. 0+431 - 80 12 14 p

Unterricht mit FORTH
Rolf Kretzschmar
Tel./Fax 0+2401 - 8 88 91 ap

UUCP (FORTH ... per eMAIL)
Andreas Jennen
Tel. 0+30 - 3 96 52 27 ap

volksFORTH/ultraFORTH/RTX-FG-Forth/Super8Forth
Klaus Kohl,
Tel. 0+8233 - 3 05 24 p
Fax. 0+8233 - 99 71 f

Forth-Vertrieb

volksFORTH (PC, ST, C64) / F68K (68000) / ...
Johannes Teich
Ettaler-Mandl-Weg 19
82418 Murnau
Tel. 0+8841 - 29 43
jgt@bbs.forth-ev.de

Forum: Forth-Mailbox

Forth-Mailbox (BBS)
Jens Wilke (SysOp)
Tel. 0+89 - 8 71 43 52 p
Mailbox 0+89 - 8 71 45 48,
300-2400 baud (8N1)

Hinweise

Zu den Telefonnummern

f == FAX
a == Anrufbeantworter, hier können Sie Ihren Ansprechpartner eventuell vorinformieren, erwarten Sie bitte keinen (kostspieligen) Rückruf
g == geschäftlich, zu erreichen innerhalb typischer Arbeitszeiten
p == privat, zu erreichen außerhalb typischer Arbeitszeiten

Die Adressen des Forth e.V. (Forth Büro) und der Redaktion / Anzeigenverwaltung finden Sie im Impressum.

komfortabler - schneller - vielseitiger - praxisnäher - komfortabler - schneller - vielseitiger

F-PC-ak
version 4.2
DAS Forth für den PC

```
: ENVIRONMENT?  ( c-addr u -- false )
      2drop  FALSE ;

: POSTPONE      ( "<spaces>name" -- )
      ?comp
      defined dup  0= ?missing  0<
      IF      COMPILE compile
      THEN COMPILE, ; IMMEDIATE
```

F-PC-ak mit INSTALL-Programm
INSTANT-Forth (der Sofort-Start in Forth mit F-PC)
F-PC Original v.>3.5614 und tCOM v.>2.21/2.29
andere Forth-Systeme zur Referenz
ausgesuchte DOS-Utilities (z.B. ModemProg. für Forth-MailBox)

F-PC-ak KomplettPaket (7 HDdisks) DM156 (UPdate-Preise erfragen),
F-PC- & Forth- Kursus DEUTSCH & Englisch (Haskell/mis) DM20,
F-PC Dokumentation, Dr. Tim Hendtlass, 'Real Time Forth' DM40/DM55,
F-PC User Contributions (Libraries: US1, US2, AK1) 3 HDdisks je DM15.
VersandPauschale: DM12 bei Vorkasse, 90mm-HD-Floppies (1m44).

Arndt Klingelnberg \ StrassburgerStr. 12 \ D-52477 Alsdorf
Tel. 0+2404 - 6 16 48 \ Fax. 0+2404 - 6 30 39

komfortabler - schneller - vielseitiger - praxisnäher - komfortabler - schneller - vielseitiger

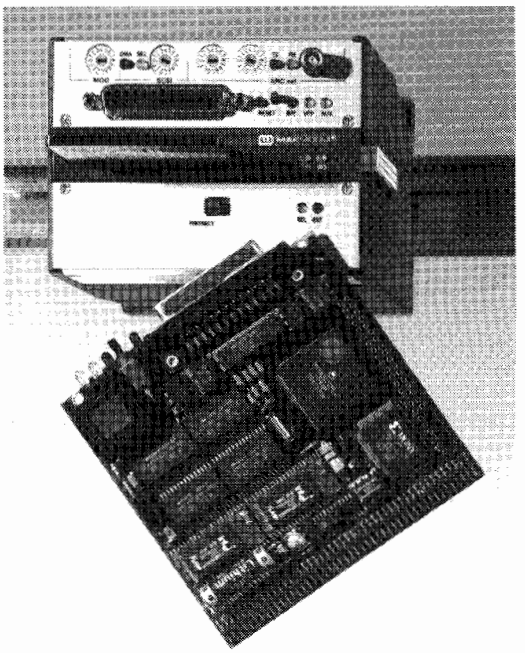
68HC11F1
Microcontrollerboard

- **universell**
  flexibel erweiter- und konfigurierbar
  RS232, 8 AD-Kanäle (8 bit), ADC-Referenz, 30 Portleitungen, 4 programmierbare Chipselects, flexibles Timersystem, bis zu 32k RAM und 64k ROM
  ideal fur Prototypen und Kleinserien
- **sicher**
  Watchdog, Clockmonitor, Power Fail Detektor,
  RAM-Standby Chipselect-Verriegelung
- **stromsparend**
  HCMOS Technologie
  Lowpowermodes optimal nutzbar
- **platzsparend**
  65mm·100mm
- **Entwicklungstools**
  Sourcecodedebugger und Entwicklungsumgebung
  MONI11F1 incl. Dokumentation und optimierendem Forth-Compiler TCOM6811

MONI11F1 für PC/XT/AT **99,- DM**
68HC11F1-Board, komplett, betriebsbereit **299,- DM**
alle Preise incl. Mwst.

Holger Dyja Naumannstr.13 10829 Berlin
Tel. 030 / 784 12 57

Großhirn an Kleinhirn!
BrainCells zur Prozeß-Automation



- **BrainCells**® sind Baugruppen mit definierter Funktionalität, die autonom oder im Verbund eines Feld-Bus-Systems (z.B. Interbus S, ArcNet) arbeiten können.
- Die Zentral-Baugruppe basiert auf dem Hochleistungs-Prozessor 68332. Viel Speicher (2MB-Flash bis 2MB stat. Ram) sowie das Multi-Tasking-Betriebssystem MAKforth sind intigriert.
- Die Module werden auf 35mm Normschienen montiert. An die Zentraleinheit lassen sich bis zu 256 weitere Module anstecken. Dabei können einzelne Module zur Leistungssteigerung selbst wieder CPU's beinhalten, oder passiv die Leistung der Haupt CPU nutzen.
- Eine Vielzahl unterschiedlicher Baugruppen ist erhältlich.
  - Messwertaufnehmer für alle passiven Geber DMS, PT 100, Induktivtaster usw.
  - Schrittmotor-Steuerung
  - Hydraulik-Steuerung mit hoher Geschwindigkeit und Dynamik. (Fährt z.B. eine 8to-Achse in 20msec 100mm punktgenau.)
  - SCSI-Festplatten Controller

  usw.

Dr. Weiss GmbH
Dossenheimer Weg, 69198 Schriesheim
Telefon 06203/6987-0, Telefax 06203/698712

FORTH-SYSTEME GMBH

Postfach 1103,
79200 Breisach

Telefon (0 76 67) 5 51
Telefax (0 76 67) 5 55

Telefon Schweiz:
(055) 53 65 55

UR/FORTH

- FORTH-83 Standard
- Für MS-DOS, OS/2, 80386
- Direkt gefädelte Code Implentationen mit dem obersten Stackwert im Register um größtmögliche Ausführungsgeschwindigkeit zu erreichen
- Segmentiertes Speichermodell mit Programm, Daten, Headers und Dictionary Hash Table jeweils in einem getrennten Segment
- Komplett gehashtes Dictionary führt zu extrem schneller Übersetzung
- Mächtige neue String Operatoren (Suche, Extraktion, Vergleich und Addition) sowie einen dynamischen String-, Speichermanager
- Kann mit Objektmodulen, die in Assembler oder anderen Hochsprachen erzeugt wurden, gelinkt werden
- Native Code Optimizer zur direkten Umsetzung in 80 x 86 Code im Lieferumfang

FORTH-83 Metacompiler

Der LMI FORTH Metacompiler wird mit komplettem Quellcode für ein ausführlich ausgetestetes, Hochgeschwindigkeits FORTH 83 Kern ausgeliefert, wobei Sie die Auswahl aus folgenden Zielprozessoren haben:

- 8086/8088
- Z80/HD64180
- 8080/8085
- 68000
- Z8
- 1802
- 6809
- 8096/97
- 78310
- 8031/32/535
- 6303
- 6502
- V25
- 68HC11
- RTX 2000
- 80C166

Sie erzeugen schnelle und kompakte Anwendungen, indem Sie Ihre Quellprogramme mit unserem Forth Nucleus zusammenstellen und ihn mit dem LMI FORTH Metacompiler übersetzen.

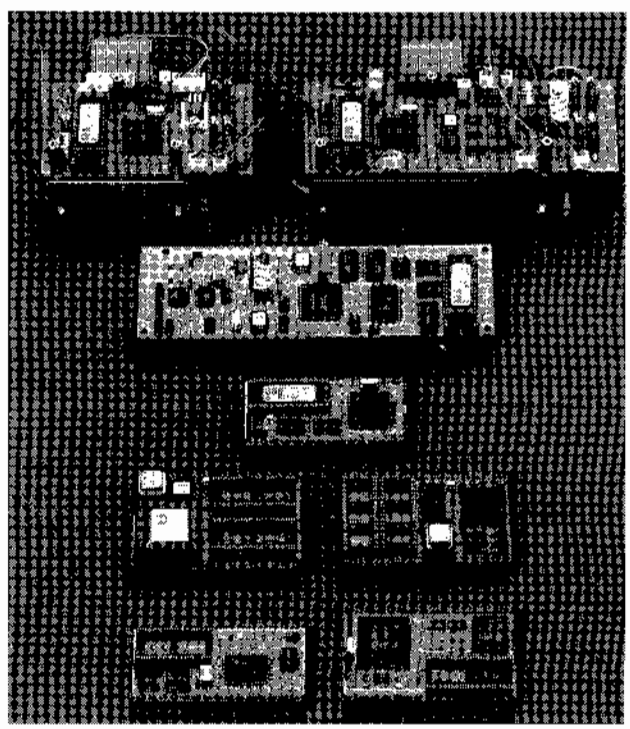
WinFORTH

- UR/FORTH kompatibel
- Windows Funktionen werden voll unterstützt
- Erweiterte Debugging-Hilfsmittel
- Online Windows Hilfe
- Coprozessor Unterstützung möglich
- Software-Gleitkomma-Paket
- Viele Beispielprogramme
- Upgrades von UR/FORTH Systemen auf WinFORTH sind preisgünstig zu erhalten

SRS II

- Serieller ROM Emulator
- Unterstützung folgender Bausteine:
  27256, 27512, 271000, 27010, 27020, 27040
- Minimale Zugriffszeit 100 ns
- Maximale Baudrate 115.200 bits/s
- Highspeed Interface als Option
- Gleichzeitiger Zugriff von Host und Zielprozessor
- Zusätzliche serielle Schnittstelle über den ROM-Sockel
- Intel-Hex, Motorola-S oder ASCII/binär Formate werden unterstützt
- Der SRS II ist nur 157 x 94 x 36 mm groß
- SRS63 kompatibel

ModuNORM



CPU-Steck-Module im Scheckkartenformat:

- 8 Bit z.B. 6303
- 16 Bit z.B. V25
- Highspeed RTX-2000/1
- 80C166
- Softwareunterstützung durch SwissFORTH
- Thermodrucker und Controller
- LCD Grafik-Controller

Bitte fordern Sie unseren Produktkatalog und die Preisliste an. FORTH-Gesellschaftsmitglieder erhalten bis zu 10% Rabatt (artikelabhängig).