C-Control/*BASIC* Steuercomputer Typ M und Programmieradapter

Bedienungsanleitung

Wichtig! Unbedingt lesen!

Bevor Sie den C-Control/*BASIC* Steuercomputer, den Programmieradapter oder angeschlossene Geräte in Betrieb nehmen, lesen Sie bitte diese Anleitung vollständig durch. Sie erläutert Ihnen die korrekte Verwendung und weist auf mögliche Gefahren hin.

Für Schäden, die aus der Nichtbeachtung der Bedienungsanleitung resultieren, besteht keinerlei Garantieanspruch und übernehmen wir keine Haftung!

Der Einsatz des C-Control/BASIC Steuercomputers Typ M erfordert einige schaltungstechnische und Computer-Grundkenntnisse. Sollten Sie über keine Erfahrung im Umgang mit PCs, Lötkolben und elektronischen Bauelementen verfügen, berät Sie unsere Technische Kundenbetreuung gern bezüglich geeigneter Einstiegsmöglichkeiten.

Hinweise zur beschränkten Garantie und Haftung

Der C-Control/*BASIC* Steuercomputer, nachfolgend auch mit „das Gerät“ bezeichnet, mit der im Mikroprozessor MC68HC05B6 als ROM-Maske integrierten und der zugehörigen PC-Software wird in vorliegender Form geliefert.

Conrad Electronic übernimmt keine Garantie dafür, daß die Leistungsmerkmale individuellen Ansprüchen entsprechen oder daß die Software im Mikroprozessor und die PC-Software in jedem Fall unterbrechungs- und fehlerfrei arbeiten. Der Anwender trägt das gesamte Risiko bezüglich der Qualität und der Leistungsfähigkeit des Gerätes inklusive aller Software.

Conrad Electronic garantiert die Funktion der mitgelieferten Applikationsbeispiele unter Einhaltung der in den Technischen Daten spezifizierten Bedingungen. Sollte sich das Gerät oder die PC-Software darüberhinaus als fehlerhaft oder unzureichend erweisen, so übernimmt der Kunde alle entstehenden Kosten für Service, Reparatur oder Korrektur.

Die Gewährleistung von Conrad Electronic beschränkt sich ausschließlich auf den Austausch des Gerätes innerhalb der Garantiezeit bei offensichtlichen Defekten an der Hardware, wie mechanischer Beschädigung, fehlender oder falscher Bestückung elektronischer Bauteile, ausgenommen gesockelter integrierter Schaltkreise und Steckbrücken.

Es besteht keine Haftung für Schäden die unmittelbar durch oder in Folge der Anwendung des C-Control/*BASIC* Steuercomputers entstehen. Unberührt davon bleiben Ansprüche, die auf unabdingbaren gesetzlichen Vorschriften zur Produkthaftung beruhen.

Bestimmungsgemäße Verwendung

Der C-Control/*BASIC* Steuercomputer dient zur programmierbaren Ansteuerung elektrischer und elektronischer Geräte, die mit Schutzkleinspannung betrieben werden. Diese Geräte können in beliebige technische Systeme integriert werden, die nicht direkt oder indirekt medizinischen, gesundheits- oder lebenssichernden Zwecken dienen oder durch deren Betrieb Gefahren für Personen oder Sachwerte entstehen können.

Zur Programmierung des Gerätes ist ausschließlich der originale Programmieradapter in Verbindung mit der mitgelieferten PC-Software zu verwenden.

Inhaltsverzeichnis

Einleitung

Was ist und was kann der C-Control/*BASIC* Steuercomputer Typ M?

Der C-Control/*BASIC* Steuercomputer ist ein kompakter Baustein für den universellen Einsatz in Meß-, Steuer- und Regelungsaufgaben und verfügt außerdem über die Fähigkeiten der seriellen Datenübertragung und der Datenspeicherung.

Die Mikroprozessortechnik ist aus dem heutigen Leben nicht mehr wegzudenken. In nahezu allen modernen elektronischen Geräten führen Mikroprozessoren Regie. Ihre „Intelligenz“ erhalten diese Chips durch Programmierung. Die Programmierung eines Mikroprozessors ist teilweise sehr kompliziert und erfordert ein umfangreiches Spezialwissen und teure Entwicklungswerkzeuge. Für Hobbyanwender und kleine Unternehmen bleibt der Zugang zur Mikroprozessortechnik somit meist versperrt. Mit dem C-Control- System eröffnen sich jedoch die Möglichkeiten dieser Technik für jeden interessierten Anwender.

Der C-Control/*BASIC* Steuercomputer als Typ M besteht aus den Kernkomponenten der bewährten Hardware der C-Control/*BASIC*-Unit (Conrad Electronic Best.-Nr. 95 05 72). Durch Weglassen nicht in jedem Anwendungsfall benötigter Bauteile und Anwendung der SMD-Technologie entstand eine miniaturisierte Version, die sich durch ihre geringen Abmessungen besonders für den Einsatz in elektronischen Handgeräten oder im Modellbau empfiehlt.

Die Programmierung des C-Control/*BASIC* Steuercomputers erfolgt in der weitbekannten und leicht zu erlernenden Programmiersprache BASIC. So wird der C-Control/*BASIC* Steuercomputer durch wenige Zeilen BASIC-Quelltext zur intelligenten Alarmanlage, zum komplexen Datenerfassungssystem, zur Steuerzentrale einer Heizungsanlage oder zum „Hirn“ eines kleinen Robotermodells. Das Feld der Anwendungsmöglichkeiten ist nahezu unbegrenzt.

Für den Kontakt zur Außenwelt stehen acht analoge Eingänge, zwei analoge Ausgänge sowie 16 frei als Ein- oder Ausgänge programmierbare Digitalports zum Anschluß von Sensoren, Schaltern, LED's, Transistoren oder Relais zur Verfügung.

Der Steuercomputer besitzt einen Eingang für einen DCF77-Funkuhrempfänger. Damit ist ein sekundengenaues Ausführen von

Schaltfunktionen realisierbar. Alternativ dazu ist über den DCF77-Eingang eine Frequenzmessung möglich.

Im Vergleich zur klassischen Version des Steuercomputers verfügt der Typ M zusätzlich über einen zweiten Frequenzmeßeingang sowie über einen Interrupt-Eingang. Das System wurde außerdem durch einen RESET-Baustein ergänzt.

Handhabungs- und Sicherheitshinweise

Allgemeines

Der C-Control/*BASIC* Steuercomputer und der Programmieradapter wurden gemäß den geltenden gesetzlichen Vorschriften einer Sicherheitsprüfung unterzogen und entsprechend zertifiziert (CE). Bei sachgemäßen Gebrauch gehen normalerweise keine Gesundheitsgefährdungen von den Geräten aus.

Der C-Control/*BASIC* Steuercomputer und der Programmieradapter sind als elektronische Geräte mit der dafür üblichen Vorsicht und Sorgfalt zu behandeln. Die Mißachtung der aufgeführten Hinweise oder eine andere als die bestimmungsgemäße Verwendung kann zur Beschädigung oder Zerstörung des Steuercomputers, des Programmieradapters oder angeschlossener Geräte führen.

Umgebungsbedingungen

Der C-Control/*BASIC* Steuercomputer und der Programmieradapter sind nicht gegen Lichtbogenüberschläge geschützt und dürfen nicht in Starkstromindustrieanlagen verwendet werden. Die maximalen Eingangsgrößen gemäß den Spezifikationen in den Technischen Daten dürfen nicht überschritten werden. Die Geräte sind nicht in Räumen oder Umgebungen einzusetzen, in denen brennbare oder ätzende Gase, Dämpfe oder Stäube vorhanden sind oder vorhanden sein können.

Nachdem der C-Control/*BASIC* Steuercomputer oder der Programmieradapter von einem kalten in einen wärmeren Raum gebracht wurden, dürfen sie nicht sofort in Betrieb genommen werden. Das möglicherweise entstandene Kondenswasser kann zu Funktionsstörungen oder zum Ausfall elektronischer Bauelemente führen.

Vermeiden Sie starke Magnetfelder, wie sie in der Nähe von Maschinen oder Lautsprechern vorkommen.

Versorgungsspannung

Auf keinen Fall darf die 230 Volt Netzspannung mit dem Programmieradapter oder mit dem C-Control/*BASIC* Steuercomputer verbunden werden!

Programmieradapter

Der Programmieradapter benötigt zu seiner Versorgung eine **stabilisierte Gleichspannung von 5V**. Verwenden Sie dafür nur ein geprüftes Labor- oder Steckernetzteil. Beachten Sie unbedingt den Anschlußplan! Eine Verpolung der Versorgungsspannung kann zur Beschädigung des Adapters führen.

C-Control/*BASIC *Steuercomputer

Zur Versorgung des Gerätes ist eine **stabilisierte Gleichspannung von 5V** zu verwenden. Diese ist von der Anwenderschaltung, in der das Gerät zum Einsatz kommen soll, zur Verfügung zu stellen und über die im Anschlußplan gekennzeichneten Kontakte zuzuführen. Beachten Sie unbedingt den Anschlußplan! Bei Verpolung der Versorgungsspannung kann der Steuercomputer zerstört werden.

Das Gerät darf nur im spannungsfreien Zustand auf die Anwenderschaltung gesteckt oder von dieser gezogen werden. Anderenfalls kann es zur Zerstörung des Steuercomputers oder angeschlossener Geräte führen. Die Forderung der Spannungsfreiheit gilt dabei für jeden Verbindungskontakt des Steuercomputers, nicht nur für die Pins zur Spannungsversorgung!

Ist der Steuercomputer zur Programmierung auf den Programmieradapter gesteckt, so wird der Steuercomputer vom Adapter spannungsversorgt. Vor dem Aufstecken/Abziehen des C-Control/*BASIC* Steuercomputers auf/vom Programmieradapter ist die Spannungsversorgung des Programmieradapters abzuschalten.

Sind Programmieradapter und Steuercomputer per Kabel (über fünfpolige Stiftleisten, siehe unten) verbunden, darf nur eines der Geräte mit einer Spannungsversorgung verbunden sein! Das jeweils andere Gerät wird über das Kabel mitversorgt.

Elektrostatische Entladungen

Besonders in trockener Luft kann sich der menschliche Körper elektrostatisch aufladen. Beim Kontakt mit leitenden Gegenständen baut sich diese Ladung mit einem kleinen Funken ab. Solche Entladungen beim Berühren elektronischer Bauelemente können diese zerstören. Vor dem Hantieren mit dem C-Control/*BASIC* Steuercomputer oder dem Programmieradapter sollten Sie einen großen, geerdeten Gegenstand berühren (z.B.: ein PC-Metallgehäuse, eine Wasserleitung oder ein Heizungsrohr), um eventuelle Aufladungen abzubauen.

Elektromagnetische Verträglichkeit (EMV)

Die vom C-Control/*BASIC*-Steuercomputer Typ M abgegebene Störstrahlung liegt unterhalb der gesetzlichen Grenzwerte (CE).

Der C-Control/*BASIC*-Steuercomputer Typ M kann jedoch selbst durch Störstrahlung aussendende Geräte oder elektrostatische Entladungen in der Umgebung in seiner Funktion beeinflußt werden. In diesen Fällen wird automatisch ein RESET ausgelöst, um ein definiertes Verhalten des Systems zu gewährleisten. Beachten Sie bitte, daß dann das eventuell unterbrochene BASIC-Programm neu gestartet werden muß (siehe unten).

Zur Gewährleistung eines sicheren Betriebes sollte der C-Control/*BASIC*-Steuercomputer Typ M nur in Geräten mit geschirmtem Gehäuse zum Einsatz kommen. Die Versorgungsspannungszuführung muß durch geeignete Maßnahmen (Stützkondensatoren, Drosseln) vor Spannungsspitzen/einbrüchen geschützt werden. Nicht benutzte Eingangsports sollten auf einen definierten Pegel gelegt werden.

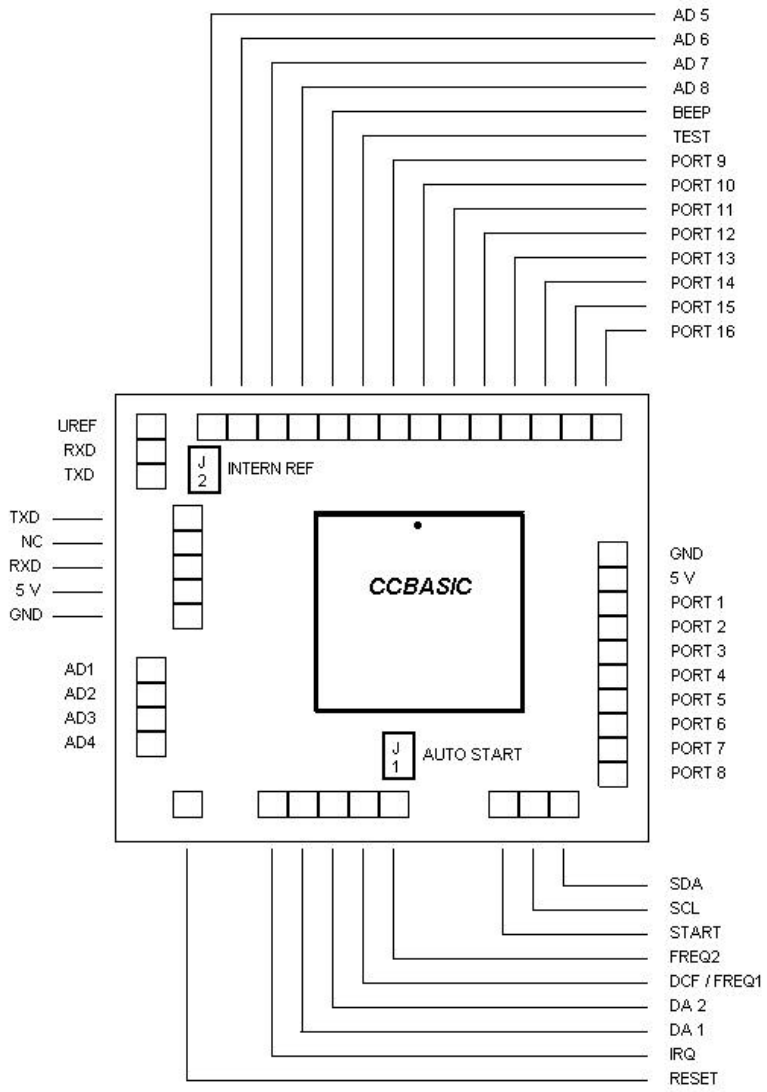
Beachten Sie bitte, daß von Ihnen aufgebaute Geräte vor ihrer Inbetriebnahme einer gesetzlichen Zulassung (CE-Zertifikat) bezüglich ihrer Gesamtfunktion bedürfen!

Aufbau und Funktionsweise

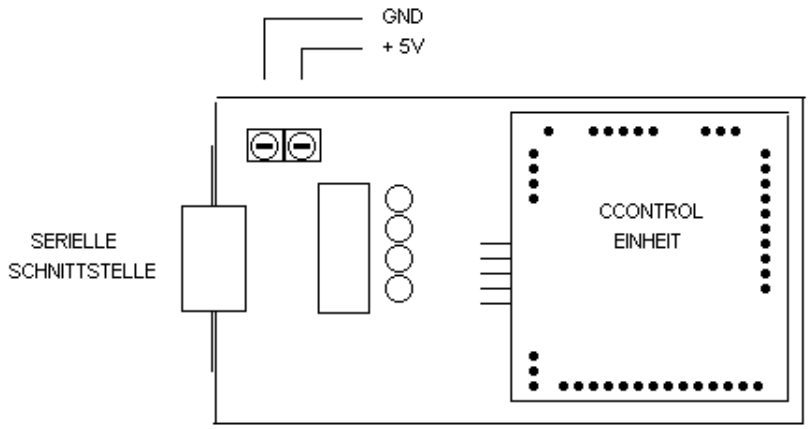
Überblick

Anschlußpläne

Anschlußplan des C-Control/*BASIC* Steuercomputers Typ M:



Anschlußplan des Programmieradapters:



Hardware

Herzstück des C-Control/*BASIC* Steuercomputers ist ein Mikrocontroller vom Typ MC68HC05B6 von MOTOROLA. Der Steuercomputer ist auf einer kleinen Platine aufgebaut. Darauf befinden sich neben dem Mikrocontroller ein Speicherchip, ein RESET-Baustein sowie einige wenige passive Bauelemente (Widerstände, Kondensatoren). Der Speicherchip hat eine Kapazität von acht Kilobyte. Er dient zur Aufnahme Ihres Anwenderprogramms und kann auch zur Aufzeichnung von Daten benutzt werden. Durch die EEPROM-Technologie bleiben alle Informationen auch nach Abschalten der Betriebsspannung erhalten. Der RESET-Baustein garantiert ein definiertes Abschalten des Systems bei Betriebsspannungsausfall.

Alle Ein- und Ausgänge sowie einige Systemsignale des Steuercomputers sind an Stiftleisten zugänglich. Über diese wird der Steuercomputer auf die Zielapplikation bzw. den Programmieradapter gesteckt. An einer Seite des Steuercomputers ragen fünf Pins hervor. Diese Seite ist die Seite 1, die Nummern der anderen Seiten ergeben sich im Uhrzeigersinn fortlaufend. Die fünf Pins an Seite 1 dienen zur Verbindung des Steuercomputers mit dem Programmieradapter per Kabel (nicht im Lieferumfang!), um den Programmieradapter auch als RS232-Pegeladapter für Ihre Applikation verwenden zu können. In diesem Fall wird der Programmieradapter über das Kabel auch mit 5V versorgt. Es wird keine zusätzliche Spannungsquelle benötigt.

Auf dem Programmieradapter befinden sich der RS232-Schnittstellenbaustein mit seinen externen Komponenten sowie der Sub-D-Steckverbinder zur Verbindung mit einem PC über ein Nullmodemkabel. Die Stiftleisten dienen zur Aufnahme des C-Control/*BASIC* Steuercomputers während der Programmierung.

Systemsoftware

Wie funktioniert C-Control/*BASIC*? Das von Ihnen erstellte BASIC-Programm wird von einem Compiler in eine Folge von Befehlsbytes umgesetzt. Die Befehle und die zugehörigen Parameterbytes werden über die serielle Schnittstelle und den Programmieradapter in den Steuercomputer übertragen, wo sie von dessen Betriebssystem im EEPROM-Speicherchip abgelegt werden. Durch das C-Control Konzept können Ihre Anwendungsprogramme in sehr kompakter Form gespeichert werden und belegen meist nur wenige hundert der über 8000 zur Verfügung stehenden Bytes. Damit bleibt ein großer Teil des Speicherchips frei und kann zur Aufzeichnung von Daten benutzt werden.

Sobald am START-Eingang Lowpegel anliegt, beginnt das Betriebssystem des Steuercomputers die Befehle nacheinander aus dem Speicher zu lesen und abzuarbeiten, bis zum Programmende-Befehl.

Kommunikation zwischen PC und dem C-Control/BASIC Steuercomputer über den Programmieradapter

Der Programmieradapter führt die Pegelwandlung der Schnittstellensignale durch, um die serielle Schnittstelle des Steuercomputers an den RS232-Standard anzupassen und eine Verbindung mit dem PC zu ermöglichen.

Die serielle Schnittstelle zwischen PC und dem Steuercomputer arbeitet bidirektional. Über sie werden neben den Befehlscodes zur Programmierung eines Anwenderprogramms auch einige Steuerbefehle übertragen. Während der Programmierung wird jedes übertragene Byte per Echo bestätigt.

Prinzipiell ist eine Verbindung von PC und Steuercomputer nur zur Programmierung erforderlich. Anschließend kann der Steuercomputer in die Zielapplikation eingesetzt werden und dort eigenständig arbeiten. Die Verbindung zum PC kann allerdings auch während der

Arbeit in der Zielapplikation bestehen bleiben und z.B. zur Übertragung von Meßdaten benutzt werden.

Jumper

Auf dem Steuercomputer können zwei Steckbrücken (Jumper) gesetzt werden.

- J1 - „Auto Start“ - verbindet den START-Eingang permanent mit GND. In diesem Fall beginnt das System sofort nach Zuschalten der Betriebsspannung bzw. nach einem RESET mit der Ausführung eines geladenen BASIC-Programms. Achtung: Um den Steuercomputer programmieren zu können, muß der Jumper abgezogen sein!
- J2 - „Interne Referenz“ - verbindet den Referenzspannungseingang mit dem 5V-Pegel. Alle Sensorsignale an den A/D-Eingängen werden dann auf die Betriebsspannung bezogen.

Software zur Programmierung des C-Control/BASIC Steuercomputers

Dem Programmieradapter liegt die Software zur Programmierung des C-Control/BASIC Steuercomputers bei. Sie besteht aus einer DOS-Version und einer Version für Windows 95 und Windows NT. Beide Versionen bieten eine integrierte Benutzeroberfläche zum Bearbeiten, Compilieren und Übertragen von BASIC-Programmen in den Steuercomputer.

Beide Versionen sind unabhängig voneinander einsetzbar. Arbeitet Ihr PC mit den Betriebssystemen Windows 95 oder Windows NT, benutzen Sie vorzugsweise die entsprechende Softwareversion. Es steht es Ihnen jedoch frei, auch die DOS-Version zu installieren und zu verwenden. Arbeitet Ihr PC unter DOS, OS/2 oder einem älteren Windows, so können Sie ausschließlich die DOS-Version verwenden.

Im Verzeichnis SAMPLES finden Sie einige Beispiele, die den Einstieg in die Programmierung des C-Control/*BASIC* Steuercomputers erleichtern.

Software für DOS

Installation

Die DOS-Software wird einfach durch Kopieren der zugehörigen Dateien auf eine Festplatte Ihres PCs installiert. Lesen Sie nach der Installation unbedingt das Readme-File im Programmverzeichnis! Dort finden Sie weitere aktuelle Informationen.

Programmaufruf

Um die DOS-Software zu starten, wechseln Sie in das Verzeichnis CCE, dann führen Sie CCE.EXE aus.

Programmeigenschaften

Die DOS-Software zeichnet sich besonders durch Ihre Eignung für ältere Computer (bishin zu PCs der XT-Generation!) aus. Bei

entsprechender PC-Hardware bietet das Programm eine farbige Oberfläche in Mehrfenstertechnik, mit Menü und Mausbedienung.

Weitere Informationen zu Umfang und Bedienung der DOS-Software finden Sie in der Online-Hilfe, die Sie über das Hilfemenü oder per Drücken der F1-Taste erreichen.

Software für Windows 95 und Windows NT

Installation

Zur Installation führen Sie SETUP.EXE von der Installationsdiskette aus. Lesen Sie nach der Installation unbedingt das Readme-File im Programmverzeichnis! Dort finden Sie weitere aktuelle Informationen.

Programmaufruf

Um die Windows-Software zu starten, rufen Sie nach erfolgreicher Installation das Programm CCEW32D.EXE vom Start-Menü oder vom Explorer aus auf.

Programmeigenschaften

Die Software für Windows 95 / Windows NT bietet eine Oberfläche im sogenannten MDI-Design (multiple document interface). Hinweise zu den einzelnen Programmelementen sowie zur Programmbedienung finden Sie wiederum in der Online-Hilfe.

Als herausragende Eigenschaft im Vergleich zur DOS-Software ist der integrierte Simulator/Debugger zu nennen, der ermöglicht, BASIC-Programme vor deren Einsatz in der Zielapplikation zu testen.

Erste Inbetriebnahme - Schritt für Schritt

Software-Installation

Installieren Sie zunächst die Software, wie im vorherigen Abschnitt beschrieben.

Bereitstellen einer Spannungsversorgung

Stellen Sie eine stabilisierte 5-Volt-Gleichspannung zur Verfügung, zum Beispiel aus einem Labornetzgerät. Schalten Sie die Spannungsversorgung aus und klemmen Sie den Programmieradapter über ein kurzes Kabel mit abisolierten Kabelenden polungsrichtig an.

Verbinden des Steuercomputers mit dem PC

Mit dem Programmieradapter wurde Ihnen ein Schnittstellenkabel ausgeliefert, ein sogenanntes Nullmodemkabel.

Stecken Sie das Nullmodemkabel an eine freie serielle Schnittstelle Ihres Computers. Viele Computer verfügen über eine 9-polige und eine 25-polige serielle Schnittstelle. Sollte bei Ihrem Computer nur noch eine 25-polige Schnittstelle frei sein, benötigen Sie einen zusätzlichen Adapter (nicht im Lieferumfang).

Das andere Ende des Nullmodemkabels stecken Sie am seriellen Schnittstellenstecker des Programmieradapters an.

Aufstecken des Steuercomputers

Falls der „Auto Start“-Jumper gesteckt ist, entfernen Sie diesen vom Steuercomputer.

Stecken Sie den Steuercomputer jetzt so auf den Programmieradapter, daß Seite 1 (die mit den fünf herausragenden Pins) zur Mitte des Adapters weist. Achten Sie darauf, daß alle Steckverbindungen sicher hergestellt werden und kein Pin verbogen ist.

Ein erstes Beispielprogramm erstellen

Starten Sie von der MS-DOS-Eingabezeile aus die integrierte Entwicklungsoberfläche CCE.EXE. Dieses Programm ist mit einer Standard-Oberfläche mit Menü und Mausbedienung ausgestattet und ermöglicht die Eingabe von BASIC-Quellcode (Editor), das Übersetzen (Compiler) und das Laden (Lader) des C-Control Codes in den Steuercomputer.

Öffnen Sie ein neues Editorfenster per Menü „Datei/Neu“. Speichern Sie das noch leere Programm unter dem Namen TEST1.BAS (Menü „Datei/Speichern unter“).

Im Editorfenster geben Sie dann bitte folgenden CCBASIC-Text ein:

```
#nochmal
PRINT "Hallo!"
PAUSE 50
GOTO nochmal
```

Speichern Sie das fertige Programm (Menü „Datei/Speichern“).

Wählen Sie dann im Menü „Entwicklung“ den Befehl „BASIC-Compiler“. Das Programm wird nun übersetzt (compiliert). Achten Sie dabei auf das Bildschirmfenster „Meldungen“. Dort erscheinen die Meldungen des Compilers über Verlauf und Erfolg der Übersetzung. Wenn Fehler im BASIC-Quelltext enthalten sind, werden diese im Meldungsfenster aufgelistet. In unserem einfachen Beispiel sollten keine Fehler auftreten, falls doch, dann überzeugen Sie sich bitte noch einmal von der Richtigkeit Ihrer Eingabe.

Das Beispielprogramm zum Programmieradaper übertragen

Stellen Sie zunächst über das „Optionen“-Menü ein, über welchen seriellen Port (COM1, COM2 ...) das C-Control System mit dem PC verbunden ist.

Falls das Meldungsfenster noch aktiv („oben“) ist, aktivieren Sie das Editorfenster erneut (durch Mausklick auf das Editorfenster oder Taste F6).

Schalten Sie jetzt die Spannungsversorgung am Programmieradapter ein.

Wählen Sie nun im Menü „Entwicklung“ den Befehl „Lader“. Die vom Compiler erzeugten Codes werden jetzt zum C-Control Steuercomputer übertragen. Meldungen über Erfolg oder Fehler bei der Übertragung erscheinen am Bildschirm des Laderprogramms.

Erscheint eine Fehlermeldung, so überprüfen Sie Ihre Schnittstelleneinstellung, stellen Sie nochmals sicher, daß der „Auto Start“-Jumper nicht gesteckt ist, daß kein anderes Programm die gewählte Schnittstelle gerade benutzt oder benutzt hat. Schalten Sie die Spannungsversorgung des Adapters gegebenenfalls mehrmals aus und ein. In seltenen Fällen kann es auch erforderlich sein, den PC neu zu starten.

Wenn die Übertragung fehlerfrei erfolgt ist, geht es nun daran, das Beispielprogramm auszuführen.

Das Beispielprogramm ausführen

Schalten Sie die Spannungsversorgung am Programmieradapter aus. Ziehen Sie den Steuercomputer vom Adapter und setzen Sie den „Auto Start“-Jumper.

Anschließend stecken Sie den Steuercomputer wieder auf den Programmieradapter. Der Programmieradapter dient jetzt als einfache Hardwarebasis für die Beispielapplikation.

Rufen Sie bitte im Menü „Entwicklung“ den Befehl „Miniterminal“ auf. Dadurch wird ein externes Programm gestartet, das die weitere Kommunikation mit dem C-Control System übernimmt.

Nach einer kurzen Copyright-Meldung erscheint ein schwarzer Bildschirm. Schalten Sie jetzt die Spannungsversorgung am Programmieradapter ein. Wegen des gesteckten „Auto Start“-Jumpers beginnt das System sofort mit der Programmausführung.

Was tut das Testprogramm? Der PRINT-Befehl sendet den Text „Hallo!“ über die serielle Schnittstelle an den PC. Dort wird der Text vom Miniterminal empfangen und angezeigt. Dann tritt das Testprogramm in eine kurze Pause und beginnt anschließend wieder von vorn. Das Testprogramm befindet sich in einer Endlosschleife und kann nur durch Ausschalten oder Reset angehalten werden. Die meisten C-Control Anwendungen arbeiten nach diesem Prinzip.

Wenn Sie genug „Hallo!“ gelesen haben, schalten Sie die Spannungsversorgung am Programmieradapter aus. Das Miniterminal beenden Sie mit der ESC-Taste.

Sie können jetzt das Testprogramm modifizieren und Ihre Experimentierschaltungen aufbauen, um zunächst weitere Erfahrungen mit C-Control zu sammeln. Vergessen Sie nicht, vor einer erneuten Programmierung den „Auto Start“-Jumper zu entfernen.

Anschluß externer Baugruppen

An den vier Seiten des Steuercomputers sind alle verwendbaren Ports sowie einige Systemsignale des Steuercomputers an Stiftleisten herausgeführt. Den Belegungsplan der Stiftleisten finden Sie im Abschnitt „Aufbau und Funktionsweise“, weiter oben in dieser Anleitung. Einige Beschaltungsbeispiele folgen am Ende dieser Anleitung.

Beschaltung der Digitalports (Pins PORT1 bis PORT16)

Verwendung eines Digitalports als Eingang

Digitaleingänge werden zur Abfrage von Schaltzuständen verwendet.

Wird ein Digitalport als Eingang benutzt, führt er im unbeschalteten Zustand undefinierten Pegel. Durch Beschaltung des Ports mit einem Pullup- oder Pulldown-Widerstand kann jedoch ein Pegel vorgegeben werden.

Verwendung eines Digitalports als Ausgang

Wird ein Digitalport als Ausgang verwendet, können daran nachfolgende IC’s, Transistoren oder Low-Current-Leuchtdioden direkt betrieben werden.

Der maximal zulässige Laststrom beträgt 10 mA pro Port. In jedem Fall ist eine ausreichende Strombegrenzung, zum Beispiel durch einen Widerstand, zu gewährleisten, da es sonst zur Zerstörung des Mikrocontrollers kommen kann!

Innerhalb des Mikrocontrollers erfolgt die interne Beschaltung eines Digitalports als Ausgang oder Eingang beim ersten Ausführen des Anwenderprogramms. Nach dem Zuschalten der Betriebsspannung oder nach einem Reset verhalten sich alle Digitalports zunächst elektrisch hochohmig und treiben keinen Strom.

Beschaltung der Analogports (Pins AD1 bis AD8)

An den A/D-Ports können positive analoge Spannungen im Bereich von 0...5 Volt - zum Beispiel Ausgangssignale von Sensoren - eingespeist werden. Die angelegten Spannungen werden vom internen 8-Bit-A/D-Wandler des Steuercomputers auf einen Wertebereich von 0 bis 255 abgebildet.

Eine angelegte Spannung darf den Wert von 5 Volt nie übersteigen, da das zur Zerstörung des internen A/D-Wandlers führen kann. Als zusätzlicher Schutz für den A/D-Wandler sind außerdem 10k-Widerstände in Reihe zu den A/D-Ports empfehlenswert.

Anlegen der Referenzspannung (Uref)

Bevor die A/D-Eingänge benutzt werden können, muß eine Referenzspannung mit dem Referenzspannungspin (Uref) des Steuercomputers verbunden werden. Der angelegte Spannungswert gilt als Obergrenze des Meßbereiches der A/D-Wandlung und entspricht dem Wandlungswert 255 ($FF hexadezimal).

Der günstigste Wert für die Referenz hängt vom Ausgangsspannungsbereich der eingesetzten Sensoren an den A/D-Eingängen ab. Meistens kann kann die Betriebsspannung direkt als Referenz benutzt werden (Jumper „Interne Referenz“). Der Referenz-spannungswert darf jedoch die Betriebsspannung von 5 Volt nie übersteigen!

Als Referenz für das untere Ende des Meßbereiches der A/D-Wandlung dient stets das Groundpotential (Masse, „Minus“) der Betriebsspannung.

Verwendung der D/A-Ausgänge (DA1 und DA2)

Die zwei 8-Bit-D/A-Wandler arbeiten nach dem Prinzip der Pulsweitenmodulation. In einem Zeitabschnitt (Modulationsintervall), der aus 256 Teilabschnitten besteht, wird ein D/A-Ausgang für die Dauer von sovielen Teilabschnitten high-gepulst, wie es dem 8-Bit-Wert entspricht, der zur Ausgabe bestimmt ist. Die Dauer eines Teilabschnittes beträgt 2µs, die des gesamten Modulationsintervalls 512µs (1953 Hz).

Zur Demodulation, also Wandlung in ein echtes Analogsignal genügt meist ein einfaches RC-Glied. Beachten Sie dabei jedoch die Restwelligkeit und den erzielbaren Maximalwert des Ausgangssignals. Beides ist abhängig von der Last, die nach dem RC-Glied folgt.

Anschluß einer DCF77-Aktivantenne (Pin DCF/FREQ1)

Das Ausgangssignal einer DCF77-Aktivantenne (low-aktiv) wird an Pin DCF/FREQ1 eingespeist. Der DCF-Eingang wertet die Signale aus und übernimmt bei Gültigkeit der empfangenen Daten Uhrzeit und Datum in die interne Echtzeituhr.

Alternativ zum Zeitempfang können mit dem Pin DCF/FREQ1 Frequenzen von CMOS/TTL-kompatiblen Rechtecksignalen bis ca. 5000 Hz (bei einer Auflösung von 1Hz und einem Fehler von maximal 1%) gemessen werden.

Frequenzmessung an Pin FREQ2

Über den Pin FREQ2 können Frequenzen von CMOS/TTL-kompatiblen Rechtecksignalen bis ca. 32000 Hz gemessen werden.

Tonausgabe am BEEP-Pin

Am BEEP-Pin können per BEEP-Befehl (siehe unten) Rechtecksignale (0/5V) ausgegeben werden, die z.B. durch Anschluß eines piezoelektrischen Schallwandlers (Schallwandler ohne interne Elektronik zwischen BEEP und GND) als Töne hörbar gemacht werden können.

Zur Erzielung höherer Lautstärken kann ein NF-Verstärkermodul an BEEP über einen Spannungsteiler (500k...1M-Poti) und einen Koppelkondensator (10µ-Elko) angeschlossen werden und an diesem ein passender Lautsprecher.

Auf keinen Fall darf ein niederohmiger Lautsprecher direkt mit dem BEEP-Pin verbunden werden, da das zur Zerstörung des Steuercomputers führen kann!

Der Interrupt-Eingang IRQ

Der IRQ-Pin ist auf dem Steuercomputer mit einem 10k-Pullupwiderstand high-gezogen. Wird beim Ausführen eines BASIC-Programms eine Low-Flanke am IRQ-Pin erkannt, so verzweigt die Abarbeitung des Programms vor dem nächsten BASIC-Befehl zu einer vom Anwender definierten Stelle (siehe INTERRUPT-Befehl).

Der RESET-Eingang

Der RESET-Pin ist auf dem Steuercomputer mit einem 10k-Pullupwiderstand high-gezogen. Wird am RESET-Pin Low-Pegel erkannt, werden alle Aktivitäten des Steuercomputers sofort unterbrochen und alle Ports und internen Speicher zurückgesetzt (=0).

Der START-Eingang

Der START-Eingang ist auf dem Steuercomputer mit einem 10k-Pullupwiderstand high-gezogen. Die Abarbeitung eines in den Steuercomputer geladenen Programms nach Einschalten der Betriebsspannung oder RESET beginnt, wenn der START-Eingang low-gezogen wird.

Bitte beachten Sie, daß ein gesetzter „Auto Start“-Jumper den START-Eingang permanent auf Low-Pegel hält.

Verwenden der seriellen Schnittstelle (Pins RXD und TXD)

Sender TXD

Am TXD-Pin wird das digitale Signal des Senders der seriellen Schnittstelle ausgegeben. Die Pegel sind CMOS/TTL-kompatibel. Vor dem Anschluß an einen PC muß eine Pegelwandlung erfolgen, z.B. mit dem Programmieradapter oder einer separaten Schaltung mit einem MAX232-IC.

Empfänger RXD

Am RXD-Pin können serielle Daten mit CMOS/TTL-Pegel eingespeist werden. Diese Signale werden vom Empfänger der seriellen Schnittstelle ausgewertet.

I²C-Bus-Schnittstelle (Pins SDA und SCL)

An den Pins SDA und SCL ist der I²C-Bus herausgeführt, der den Mikroprozessor des Steuercomputers mit dem seriellen EEPROM verbindet. Diese beiden Pins sind für spätere Erweiterungen reserviert und sollten nicht beschaltet werden!

Systemressourcen

Unter dem Begriff „Systemressourcen“ sind hier alle internen Funktionseinheiten zusammengefaßt, die sich nicht unmittelbar aus den Eigenschaften des Mikrocontrollers ableiten, sondern durch das auf dem Chip maskenprogrammierte Betriebssystem zur Verfügung gestellt werden. Wie diese Systemressourcen im BASIC-Programm angesprochen werden, wird weiter unten in der Befehlsübersicht beschrieben.

Timer

Im Hintergrund des Betriebssystems läuft ein mit 20 Millisekunden getakteter 16-Bit-Timer, dessen Wert jederzeit ausgelesen und zum Herstellen von Zeitbezügen im BASIC-Programm benutzt werden kann.

Echtzeituhr

Die per DCF77 empfangene Zeit- und Datumsinformation wird vom Betriebssystem in sieben interne Speicherzellen (Jahr, Monat, Tag, Wochentag, Stunde, Minute, Sekunde) übertragen und bis zur nächsten Synchronisation in Portionen von 20 Millisekunden erhöht. Die Ganggenauigkeit der Echtzeituhr zwischen den Synchronisations-zeitpunkten ist bestimmt durch die Abweichung des 4MHz-Quarzes von seiner Normalfrequenz von bis zu 0,1 Promille, abhängig von Streuungen in der Serienproduktion und von der Temperatur. Das entspricht einer Abweichung von bis zu 0,36 Sekunden pro Stunde.

Nach dem Zuschalten der Betriebsspannung und nach einem Reset startet die Uhr mit dem 01.01.97, 00:00:00 Uhr.

Die internen Speicherzellen für Datum und Uhrzeit können vom BASIC-Programm aus gelesen und beschrieben werden. Durch das Beschreiben der Zeitspeicherzellen kann die Uhr also auch ohne DCF77-Empfang gestellt werden. Für Programmtests oder bei geringem Anspruch an die Ganggenauigkeit kann so auf die DCF77-Antenne verzichtet werden.

Userbytes

Der Mikrocontroller MC68HC05B6 verfügt über insgesamt 240 Bytes RAM. Der C-Control Steuercomputer belegt davon größten Teil für Betriebssystemfunktionen (Stack, Timer, Uhr, DCF77-Rahmenpuffer, Schnittstellenpuffer, Zwischenspeicher für Berechnungen usw.). 24 Bytes stehen dem Anwender zur Verwendung in BASIC-Programmen zur Verfügung. Die Verwendung dieser Userbytes ist im Abschnitt zum DEFINE-Befehl weiter unten beschrieben.

Programmieren mit CCBASIC

CCBASIC ist der BASIC-Dialekt, der zur Programmierung des C-Control/*BASIC* Steuercomputers verwendet wird. Die Syntax entspricht in etwa der des Standard-BASIC. Bei einigen Befehlen gibt es Abweichungen oder Erweiterungen, die speziell auf die Hardware des Steuercomputers zugeschnitten sind.

Was ist ein Programm ?

Ein Programm ist die Beschreibung eines Informations-verarbeitungsprozesses. Im Laufe eines solchen Prozesses wird aus einer Menge von variablen oder konstanten Eingangswerten eine Menge von Ausgangswerten berechnet. Die Ausgangswerte sind entweder selbst Ziel der Informationsgewinnung oder dienen mittelbar zur Reaktion auf die Eingangswerte. Neben den eigentlichen Berechnungen kann ein Programm Anweisungen zum Zugriff auf die Hardware des Computers oder zur Steuerung des Programmflusses enthalten.

Ein BASIC-Programm besteht aus mehreren Zeilen sogenannten Quelltextes. Dabei enthält jede Zeile eine oder mehrere Rechen- oder Steueranweisung. Außer diesen Anweisungen selbst bestimmt ihre Reihenfolge ganz wesentlich die eingangs beschriebene Informationsverarbeitung. Die Ausführung der den Anweisungen entsprechenden Operationen durch den Steuercomputer erfolgt sequentiell, also nacheinander. Eine Folge von Programm-anweisungen mit einem bestimmten Ziel nennt man auch Algorithmus.

Datentypen

Daten sind die Objekte des Informationsverarbeitungsprozesses, sie repräsentieren die gespeicherten Informationen. Der C-Control/*BASIC* Steuercomputer verarbeitet und speichert ausschließlich ganzzahlige numerische Daten - sogenannte „Integerzahlen“ von 1, 8 oder 16 Bit.

Eine Variable von 8 Bit (Byte) kann nur nichtnegative Werte von 0 bis 255 aufnehmen. Der Wertebereich einer Integervariable von 16 Bit (Word) reicht von -32768 bis +32767. Achten Sie bei allen Berechnungen darauf, daß die Ergebnisse diese Grenzwerte nicht

über- oder unterschreiten, da es sonst zu sogenannten „Überläufen“ kommt.

```
a = 255 + 1
```

ergibt beispielsweise für a den Wert 0 und nicht 256, wenn a nur ein Byte repräsentiert!

```
a = -32768 - 1
```

ergibt 32767 und nicht -32769, wenn a ein Word repräsentiert!

Grundlegende Elemente der Programmiersprache CCBASIC

Allgemeines

Jede Programmzeile enthält eine oder mehrere Anweisung, die durch Doppelpunkte : getrennt sind.

Zeilennummern, wie in älteren BASIC-Dialekten üblich, sind nicht notwendig. Werden dennoch Zeilennummern angegeben, so können diese als Sprungziel verwendet werden.

```
10 ...
GOTO 10
```

Einen Einfluß auf die Reihenfolge der Programmoperationen haben die Nummern darüber hinaus nicht. Wenn beispielsweise im Quelltext auf eine mit 200 numerierte Zeile eine Zeile 100 folgt, wird trotzdem die Zeile 200 vor der 100 abgearbeitet.

Kommentare können zur Erläuterung des geschriebenen Programms mit in den Quelltext aufgenommen werden und steigern dessen Lesbarkeit und Wartungsfreundlichkeit. Ein Kommentar in CCBASIC beginnt stets mit einem Hochkomma und erklärt den Rest der Zeile zum nicht zum Programm gehörigen Text.

```
a = b + c    '... Kommentar ...
```

Bezeichner

Bezeichner sind Programmelemente aus alphanumerischen Zeichen (A bis Z, 0 bis 9) die in vom Programmierer festgelegter Weise Objekte, wie Variablen und Konstanten, bezeichnen. Label-Namen und die sogenannten „reservierten Worte“ sind ebenfalls Bezeichner.

Es erfolgt keine Unterscheidung von Groß- und Kleinbuchstaben. Ein Bezeichner beginnt stets mit einem Buchstaben oder mit einem Unterstrich. Leerzeichen innerhalb eines Bezeichners sind nicht erlaubt.

Variablen und Konstanten

Variablen und Konstanten sind Objekte des Informationsverarbeitungsprozesses. In CCBASIC speichern beide einen numerischen Wert. Während der Wert einer Konstante einmal angegeben wird und dann unverändert bleibt, kann sich der Wert einer Variablen im Lauf des Programms beliebig oft ändern.

Konstanten können in CCBASIC in dezimaler, hexadezimaler und binärer Form angegeben werden. Die Syntax für Hexadezimal- und Binärzahlen sei hier am Beispiel der Zahl 46 (dezimal) gezeigt:

```
&H2E
&B101110
```

Außerdem können per DEFINE-Zeilen (siehe unten) symbolische Konstanten vereinbart werden.

Auf Variablen wird stets über ihren Bezeichner zugegriffen. Dieser Bezeichner muß vor der ersten Verwendung der Variable im Programm in einer DEFINE-Zeile definiert werden.

Label

Label markieren bestimmte Punkte in der Folge der Programmoperationen. Label sind Ziele von Sprungoperation innerhalb eines Algorithmus. In CCBASIC stehen Label am Anfang einer Zeile und beginnen stets mit einem Doppelkreuz, dann folgt - ohne Leerzeichen - der Bezeichner des Labels.

Das Beispiel zeigt die Definition des Labels „Label1“ und die Verwendung in einem Sprungbefehl:

```
#label1 ...
GOTO label1
```

Terme

Ein Term ergibt sofort (als Variable oder Konstante) oder durch Berechnung einen bestimmten Wert. Terme sind Teile von Anweisungen und stehen beispielsweise bei der Zuweisung eines Wertes an eine Variable rechts des Zuweisungszeichens „=“. Terme werden durch Kombinationen von Operanden und Operatoren gebildet.

```
100
c
a + b
(ABS(x) - 13) * 10
```

Operanden und Operatoren

Ein Operand ist in der Grundform entweder eine Konstante, eine Variable oder ein Funktionsaufruf, kann aber auch selbst wieder ein aus Operanden und Operatoren zusammengesetzter Term sein.

Operatoren bezeichnen Rechenoperationen, die mit den umstehenden Operanden auszuführen sind. Dabei gibt es eine definierte Rangfolge der Operatoren (siehe Befehlsbeschreibung), die die Reihenfolge der Berechnungen bestimmt.

Funktionen

Eine Funktion führt eine definierte Operation - zum Beispiel eine Berechnung - durch und liefert durch ihren Aufruf einen Ergebniswert. Die meisten Funktionen erwarten ein oder mehrere Argumente, die in runden Klammern „( )“ nach dem Funktionsbezeichner übergeben werden und durch Kommas getrennt sind. Einige Funktionen werden ohne Argument aufgerufen. In diesem Fall werden keine runden Klammern geschrieben.

```
ABS(x)
MAX(a,b)
RAND
EOF
```

In CCBASIC sind alle unterstützten Funktionen vordefiniert. Deren Bezeichner gehören zu den reservierten Worten. Die Formulierung anwenderdefinierter Funktionen ist in CCBASIC nicht vorgesehen.

Zuweisungen

Die Zuweisung ist die einfachste Form einer Programmanweisung. Nach dem Bezeichner einer Variablen, der ein Wert zugewiesen werden soll, folgt das Zuweisungszeichen „=“ und dann ein Term, der den zuzuweisenden Wert bestimmt. Eine Zuweisung entspricht damit einer einfachen mathematischen Formel.

```
a = 10
b = x - y
c = SQR(a*a + b*b)
```

Befehle

Neben den einfachen Zuweisungen sind Befehle Anweisungen zur Ausführung von Programmoperationen durch den C-Control/*BASIC* Steuercomputer. Befehle beginnen stets mit einem reservierten Wort. Einige Befehle erwarten einen oder mehrere Parameter zur genauen Spezifikation der auszuführenden Programmoperation. Diese Parameter werden nach dem Befehlsbezeichner und einem Leerzeichen aufgeführt und dabei durch Kommas getrennt (Ausnahme PRINT, siehe Befehlsübersicht). Im Gegensatz zu den Argumenten beim Aufruf einer Funktion stehen die Befehlsparameter nicht innerhalb runder Klammern!

```
RANDOMIZE
PAUSE 100
BEEP 440,50,50
```

Anweisungen zur Steuerung des Programmflusses

Diese Anweisungen erlauben, die Reihenfolge der an sich streng sequentiell abgearbeiteten Programmoperationen zu steuern und an Eingangswerte des Informationsverarbeitungsprozesses anzupassen. Sie bieten eine hohe Flexibilität bei der Algorithmenformulierung und sind für die Lösung mancher anwendungstechnischer Probleme sogar Grundvoraussetzung.

Anweisungen zur Steuerung des Programmflusses bestehen aus einem oder mehreren reservierten Worten und erfordern in jeweils spezieller Weise eventuell weitere Angaben.

```
GOTO label1
IF a > b THEN GOSUB label2

FOR i = 0 TO 10 STEP 2
...
NEXT
```

Compileranweisungen

Zusätzlich zu den Programmanweisungen enthält ein CCBASIC-Quelltext Compileranweisungen, die zum Beispiel zum Anlegen von Datenblocks (Tabellen) oder zur Definition von Variablen- und Konstanten dienen.

Für Compileranweisungen gilt die Doppelpunktregel zum Trennen mehrerer Anweisungen in einer Zeile nicht. Es darf jeweils nur eine Compileranweisung in einer Zeile stehen.

Die DEFINE-Anweisung

Die **DEFINE**-Anweisung ist eine Compileranweisung.

Definition symbolischer Konstanten

Es ist guter Programmierstil, statt „magischer“ Zahlen im Programm

```
IF x > 1234 THEN GOTO alarm
```

besser symbolische Konstanten zu verwenden. Durch Vergabe signifikanter Bezeichner für Konstanten erhöht sich die Lesbarkeit des Quelltextes. Wenn alle Konstanten global definiert werden, ist ein Programm auch leichter zu warten. Das gilt besonders, wenn ein und dieselbe Konstante mehrmals im Programm benötigt wird.

Die Definition einer symbolischen Konstanten erfolgt wie folgt:

```
DEFINE bezeichner wert
```

Dabei ist wert entweder eine dezimale, hexadezimale oder binäre Zahl.

So sollte das Beispiel oben besser

```
DEFINE limit 1234
...
IF x > limit THEN GOTO alarm
```

lauten.

Definition von Variablen

Der C-Control/*BASIC* Steuercomputer stellt 24 Byte-Speicherzellen seines internen Speichers (RAM) dem Anwender zur Verwendung in seinen Programmen zur Verfügung. In diesem Speicherbereich werden alle Variablen eines BASIC-Programms gespeichert. Die 24 Bytes können je nach Bedarf auch bitweise oder als 16bit Integer (Word) verwendet werden.

Im Gegensatz zum Standard-BASIC müssen in CCBASIC alle vom Programm benutzten Variablen vor ihrer ersten Verwendung definiert werden. Dabei ist der Datentyp zu spezifizieren (Bit, Byte oder Word) und kann (für Bits muß!) eine Speicherzellennummer angegeben werden. Der Anwender muß selbst darauf achten, daß keine unerwünschten Überlappungen bei der Vergabe der Speicherplätze entstehen, da es sonst zum gegenseitigen Überschreiben der Variablen kommen kann. Beispielsweise belegen bit[18], byte[2] und word[1] jeweils einen Teil der Zelle 2 des Speicherbereiches.

- Definition einer Bitvariablen:

```
DEFINE bezeichner BIT[nr]
```

Dabei sind für nr Werte von 1 bis 192 (24 Bytes mit je 8 Bit) zulässig.

- Definition einer Bytevariablen mit Zellennummer:

```
DEFINE bezeichner BYTE[nr]
```

Dabei sind für nr Werte von 1 bis 24 (24 Bytes) zulässig.

- Definition einer Integervariablen mit Zellennummer:

```
DEFINE bezeichner WORD[nr]
```

Dabei sind für nr Werte von 1 bis 12 (ein Word belegt 2 Bytes) zulässig.

Wenn bei Byte- und Worddefinitionen die Zellenangabe [nr] weggelassen wird, übernimmt der Compiler die Aufteilung auf den Speicherbereich. Achten Sie dann darauf, daß nicht abwechselnd Bytes und Words definiert werden. Die folgenden Anweisungen

```
DEFINE a BYTE
DEFINE b WORD
DEFINE c BYTE
DEFINE d WORD
```

führen zu zwei ungenutzten (verschenkten kostbaren!) Bytes, zwischen a und b sowie zwischen c und d, da Words prinzipiell an den Bytes 1,3,5,7,... usw. der 24 Bytes ausgerichtet werden.

Besser wäre,

```
DEFINE b WORD
DEFINE d WORD
DEFINE a BYTE
DEFINE c BYTE
```

zu schreiben.

Die automatische Aufteilung der Variablen auf den Speicher durch den Compiler beginnt bei Zellennummer 1. Das obige (bessere) Beispiel belegt 6 Bytes. Bei Definition weiterer Bits, Bytes und Words mit Angabe der Zellennummer ist wieder auf unerwünschte Überlappung zu achten.

Ein bereits definierter Variablenbezeichner darf nicht ein zweites Mal definiert werden.

Definition von Digitalports

In CCBASIC wird auf Ports wie auf Variablen zugegriffen. Auch hier muß jeder verwendete Port zuvor definiert sein.

- Definition eines der 16 Digitalports:

```
DEFINE bezeichner PORT[nr]
```

Dabei sind für nr Werte von 1 bis 16 zulässig.

- Definition eines 8 Bit breiten Ports:

```
DEFINE bezeichner BYTEPORT[nr]
```

Dabei sind für nr nur die Werte 1 (Ports 1 bis 8 als Byteport) und 2 (Ports 9 bis 16) zulässig.

- Definition eines Bezeichners für den gemeinsamen Zugriff auf alle 16 Digitalports als ein 16 Bit Port:

```
DEFINE bezeichner WORDPORT[nr]
```

Für nr ist nur der Wert 1 zulässig.

Definition von Analogports

- Definition eines der 8 A/D-Ports:

```
DEFINE bezeichner AD[nr]
```

Dabei sind für nr Werte von 1 bis 8 zulässig.

- Definition eines der 2 D/A-Ports:

```
DEFINE bezeichner DA[nr]
```

Dabei sind für nr nur die Werte 1 und 2 zulässig.

Befehlsübersicht

Dieses Kapitel gibt einen kompletten Überblick über die CCBASIC Operatoren, Funktionen und Anweisungen.

Mathematische und logische Operatoren

- Grundrechenarten: **+ - * /**
- Der Modulooperator **MOD** liefert den Rest einer Integerdivision,

```
a = 10 MOD 3
```

ergibt beispielsweise für a den Wert 1.

- Vergleichsoperatoren: **>** (größer als), **<** (kleiner als), **>=** (größer oder gleich), **<=** (kleiner oder gleich), **=** (gleich), **<>** (ungleich)

Das Ergebnis einer Vergleichsoperation ist entweder -1 bzw. 255 (Vergleich wahr) oder 0 (Vergleich falsch).

```
a = 10 < 3
```

ergibt beispielsweise für a den Wert 0.

- logische Operatoren: **NOT** (Negation), **AND** (Und-Verknüpfung), **NAND** (Und-Verknüpfung mit anschließender Negation), **OR** (Oder-Verknüpfung), **NOR** (Oder-Verknüpfung mit anschließender Negation), **XOR** (Exklusiv-Oder-Verknüpfung)

Die logischen Operatoren können außer zur Formulierung von Bedingungen (meist in Verbindung mit Vergleichsoperationen) auch für binäre Byte- oder Wordmanipulationen benutzt werden.

- Schiebeoperatoren: **SHL** (nach links schieben), **SHR** (nach rechts schieben) werden zum bitweisen Verschieben von Bitmustern in Byte- oder Wordvariablen benutzt. Links des Operators steht der zu schiebende Wert, rechts die Zahl, um wieviel Bits verschoben werden soll. Beim Linksschieben entspricht jede einzelne Verschiebung einer Multiplikation mit 2, beim Rechtsschieben einer Division durch 2.

```
a = 10 SHL 3
```

entspricht also

```
a = 10 * 2 * 2 * 2
```

und ergibt beispielsweise für a den Wert 80.

Mathematische Funktionen und Befehle

Die Argumente x und y, je nach Funktion oder Befehl, sind stets Terme (Definition siehe oben).

- Die Wurzelfunktion **SQR(x)** liefert eine Näherung für die Quadratwurzel aus dem Argument x. Dabei werden die Nachkommastellen abgeschnitten.
- Die Signumfunktion **SGN(x)** ergibt 1, wenn der Wert des Argumentes x größer 0, und ergibt -1, wenn der Wert kleiner 0 ist. Für x = 0 ist auch das Ergebnis der SGN-Funktion gleich 0.
- Die Maximumfunktion **MAX(x,y)** ergibt x, wenn x > y ist, sonst y.
- Die Minimumfunktion **MIN(x,y)** ergibt x, wenn x < y ist, sonst y.

- Der Befehl **RANDOMIZE x** initialisiert den internen Pseudo-Zufallsgenerator des Steuercomputers mit dem Wert von x. Ein und derselbe Initialisierungswert führt stets zu einer identischen Folge von Zahlen. Die Spezialform **RANDOMIZE TIMER** lädt den Wert des freilaufenden Timers in den Generator.
- Die Zufallsfunktion **RAND** liefert den nächsten Integer-Zufallswert des Pseudo-Zufallsgenerators. Die Zufallszahlen werden nach dem multiplikativen Verfahren mit anschließender Modulodivision (siehe ein gutes Mathematikbuch) aus dem jeweils vorangehenden Wert erzeugt.

Rangfolge von Operatoren und Funktionsaufrufen

Bei der Berechnung von Termen mit Operatoren und Funktionen ist deren Rangfolge von entscheidender Bedeutung. Teilausdrücke mit Operatoren von hohem Rang werden vor denen mit einem niedrigerem Rang berechnet (vergleiche Rechenregel: „Punktrechnung vor Strichrechnung“). Bei gleichrangigen Operatoren erfolgt die Berechnung von links nach rechts.

Wie in der Mathematik kann jedoch durch Klammersetzung zusätzlich Einfluß auf die Berechnungsreihenfolge genommen werden. CCBASIC unterstützt maximal 3 Klammerebenen.

Im Sinne der Übersichtlichkeit eines Programmes sollten jedoch „wilde“ Klammerausdrücke vermieden und komplexe Berechnungen auf mehrere BASIC-Zeilen aufgeteilt werden.

Die folgende Liste zeigt die CCBASIC Operatorenrangfolge :

Rang	**Operatoren**
9	( )
8	Funktionsaufrufe
7	negatives Vorzeichen
6	* / MOD SHL SHR
5	+ -
4	> >= < <= = <>

3	NOT
2	AND NAND
1	OR NOR XOR

Anweisungen zur Steuerung des Programmflusses

- Schleife

```
FOR variable = anfang TO ende STEP schrittweite
...
NEXT
```

Die FOR-Schleife führt die Anweisungen bis zum NEXT solange aus, bis der Wert der variable gleich dem Wert des Terms ende ist. Vor dem ersten Durchlauf wird der Wert des Terms anfang berechnet und der Schleifenvariablen zugewiesen. In jedem Durchgang wird der Wert des schrittweite-Terms zur Schleifenvariablen addiert. In der Form

```
FOR variable = anfang TO ende
...
NEXT
```

beträgt die Schrittweite konstant 1.

Die Werte des ende-Terms und des schrittweite-Terms werden mit jedem Schleifendurchlauf neu berechnet. Das gestattet eine erweiterte Kontrolle des Programmverlaufes.

FOR-Schleifen können ineinander verschachtelt werden. Die Verschachtelungstiefe ist nur durch den für die Schleifenvariablen erforderlichen Speicherplatz beschränkt.

```
FOR v1 = anfang1 TO ende1
  FOR v2 = anfang2 TO ende2
    FOR v3 = anfang3 TO ende3
    ...
    NEXT
  NEXT
NEXT
```

Jede FOR-Schleife darf im Verlauf des Programms nur über ihre eigene NEXT-Anweisung laufen. Folgender Quelltext kann zwar

compiliert und in den Steuercomputer geladen werden, wird jedoch nicht wie vielleicht erwartet funktionieren:

```
FOR v1 = anfang1 TO ende1
...
  GOTO anothernext
...
NEXT

FOR v2 = anfang2 TO ende2
...
#anothernext
NEXT
```

Achten Sie außerdem auf den Wertebereich von Schleifenvariable und ende-Term!

```
DEFINE v BYTE

FOR v = 1 TO 1000
...
NEXT
```

wird zu einer Endlosschleife, da v als Bytevariable nie den Wert 1000 erreichen kann, sondern bereits nach 255 wieder auf 0 überrollt.

- Bedingte Ausführung

```
IF bedingung THEN anweisung1
```

oder

```
IF bedingung THEN anweisung1 ELSE anweisung2
```

Die IF...THEN...ELSE-Konstruktion ermöglicht die Anpassung des Programmflusses an Bedingungen zur Laufzeit des Programms. Als bedingung ist ein beliebiger Term einzusetzen. Ergibt dessen Berechnung einen Wert ungleich 0, dann gilt die Bedingung als erfüllt, und die anweisung1 wird ausgeführt. Werden zusätzlich ein ELSE und eine zweite Anweisung angegeben, so wird diese Anweisung alternativ ausgeführt, wenn der berechnete Term einen Wert gleich 0 ergibt.

Die gesamte IF...THEN...ELSE-Konstruktion muß in einer Quelltextzeile stehen. Anweisungsblöcke (mehrere Anweisungen) nach THEN und ELSE sind nicht zulässig.

- Sprunganweisung

```
GOTO label
```

Mit der GOTO-Anweisung kann der Steuercomputer veranlaßt werden, die Programmabarbeitung an einer bestimmten Stelle fortzusetzen. Als Ziel des Sprungs wird ein Label-Bezeichner angegeben. Das Sprungziel kann sich vor oder nach der GOTO-Anweisung im Quelltext befinden.

- Aufruf und Rückkehr aus einer Unterroutine

Der Aufruf einer Unterroutine erfolgt mit der Anweisung

```
GOSUB label
```

Dabei ist label der Anfangspunkt der Unterroutine.

In den sogenannten Unterroutinen sind Programmabschnitte zusammengefaßt, die mehrfach im Verlauf der Programmabarbeitung benötigt werden. Eine Unterroutine beginnt stets mit einem Label, enthält dann eine oder mehrere Anweisungen und abschließend ein

```
RETURN
```

Nach dem RETURN wird die Programmabarbeitung mit der Anweisung nach dem GOSUB fortgesetzt. Die Programmabarbeitung darf ohne ein vorheriges GOSUB niemals an eine RETURN-Anweisung gelangen.

Die maximal zulässige Verschachtelungstiefe bei Aufrufen von Unterroutinen aus Unterroutinen ist vier.

```
#hauptprogramm
  GOSUB sub1
  ...

#sub1
  GOSUB sub2
  ...
  RETURN

#sub2
  GOSUB sub3
  ...
```

```
  RETURN

#sub3
  GOSUB sub4
  ...
 RETURN

#sub4
 ...
 RETURN
```

- Wertgesteuerte Programmverzweigung

```
ON variable GOTO label0,label1,...labeln
```

oder

```
ON variable GOSUB label0,label1,...labeln
```

In Abhängigkeit des Wertes des Selektors variable erfolgt eine Programmverzweigung oder ein Unterroutinenaufruf zu den aufgelisteten Einsprungpunkten. Ist der Wert 0, dann wird zu label0 verzweigt, bei Wert gleich 1 zu label1 usw. Ist der Variablenwert negativ oder größer als die Anzahl der aufgeführten Sprungziele, dann wird die Programmabarbeitung ohne Verzweigung fortgesetzt.

- Programmverzweigung nach einem Interruptsignal am Pin IRQ

Wird während der Abarbeitung eines Programmes ein Interruptsignal (Low-Flanke) am IRQ-Pin detektiert, wird die momentan bearbeitete BASIC-Anweisung zum nächstmöglichen Zeitpunkt unterbrochen und die Programmabarbeitung an der zuvor mit dem Befehl

```
INTERRUPT label
```

festgelegten Stelle fortgesetzt. Der Rücksprung zur Ausgangsposition erfolgt durch den Befehl

```
RETURN INTERRUPT
```

Im folgenden Beispiel wird durch jeden Interrupt eine Leuchtdiode ein/ausgeschaltet:

```
DEFINE led PORT[8]
led = OFF

‘ Festlegen der Interruptroutine
```

```
INTERRUPT switch_it

‘ Endlosschleife
#loop
GOTO loop

‘ Interruptroutine
#switch_it
  tog led
RETURN INTERRUPT
```

Gehen weitere IRQ-Signale ein, während sich der Steuercomputer gerade in der Bearbeitung eines Interrupts befindet, wird maximal ein IRQ-Ereignis gespeichert und im Anschluß an RETURN INTERRUPT ausgeführt. Alle anderen Signale gehen verloren.

Die Interrupt-Routine kann durch mehrfachen Aufruf des INTERRUPT-Befehls beliebig oft umdefiniert werden, auch während der Ausführung einer Interruptroutine.

- Programmende

```
END
```

Gelangt der Steuercomputer im Verlauf der Programmabarbeitung zur END-Anweisung, wird die Programmabarbeitung beendet. Das System verharrt dann in einem inaktiven Zustand. Jetzt kann ein neues Anwenderprogramm übertragen oder die Ausführung per Start-Taster wieder gestartet werden.

- Verzögerung des Programmflusses

Die Anweisung

```
WAIT conditionterm
```

unterbricht die Programmausführung solange, bis die Berechnung des conditionterm einen Wert ungleich 0 ergibt.

```
define key port[9]
...
WAIT key
```

In diesem Beispiel wird solange gewartet, bis vom Digitalport 9 ein HIGH-Pegel (= logisch 1) gelesen wird.

Der PAUSE Befehl unterbricht die Programmausführung für eine gewisse Zeit. Der berechnete Wert des Parameterterms geht als Multiplikationsfaktor mit der Grundeinheit von 20 Millisekunden in die Festlegung der Pausenzeit ein.

```
PAUSE term
```

Beispielsweise wird durch den Befehl

```
PAUSE 50
```

die Programmausführung für ca. 50*20 Millisekunden = 1 Sekunde unterbrochen. Die maximale Zeitabweichung der tatsächlichen Pause vom angegebenen Wert beträgt dabei prinzipbedingt +/- 20 Millisekunden.

Kommunikation über die serielle Schnittstelle

- Datenausgabe

Die Datenausgabe erfolgt als Text über die serielle Schnittstelle des C-Control/*BASIC* Steuercomputers. Ist über den Programmieradapter und ein Schnittstellenkabel zum Beispiel ein PC mit einem Terminalprogramm angeschlossen, können die ausgegebenen Daten dort angezeigt werden.

```
PRINT term
```

gibt das Ergebnis der Berechnung von term aus.

```
PRINT "text"
```

überträgt den in Anführungszeichen stehenden Text.

In beiden Fällen wird an die Übertragung ein Zeilenvorschubzeichen angehängt, welches das Terminalprogramm veranlaßt, die nächste Ausgabe in der nächsten Bildschirmzeile vorzunehmen. Der Zeilenvorschub kann unterdrückt werden, wenn dem PRINT-Befehl nach dem Parameter (term oder "text") ein Semikolon hinzugefügt wird.

```
PRINT term;
```

oder

```
PRINT "text";
```

CCBASIC unterstützt außerdem mehrere Ausgaben mit einem PRINT-Befehl, wobei die einzelnen Parameter durch Komma oder Semikolon getrennt werden. Ein Komma fügt in die Ausgabe ein Tabulatorzeichen ein, das entsprechend den Einstellungen im Terminalprogramm als eine Anzahl von Leerzeichen am Bildschirm erscheint. Sollen zwei Ausgaben ohne Zwischenraum aufeinander folgen, so sind diese im PRINT-Befehl durch ein Semikolon zu trennen.

```
PRINT "a= ", a
PRINT "a= "; a
```

Ein einzelner PRINT-Befehl ohne Parameter gibt nur einen Zeilenvorschub aus.

```
PRINT
```

- Dateneingabe

Mit dem Befehl

```
INPUT variable
```

kann ein Integerwert von der seriellen Schnittstelle gelesen und für die anschließende Weiterbearbeitung in einer Variablen gespeichert werden.

Der Wert wird z.B. in einem Terminalprogramm an einem PC eingegeben und nach dem Drücken der ENTER-Taste per Schnittstellenkabel und Programmieradapter an den C-Control/*BASIC* Steuercomputer übertragen.

Der INPUT-Befehl wartet solange, bis eine komplette Datenübertragung vom Terminal empfangen wurde. Wird der INPUT-Befehl aufgerufen, ohne daß eine Datenübertragung vom Terminal erfolgt, wird das Programm endlos an dieser Stelle stehen bleiben! Hier hilft dann nur noch der Reset-Taster und der anschließende Neustart des C-Control/*BASIC* Gerätes.

- Byteweise Kommunikation über die serielle Schnittstelle

Während PRINT und INPUT kurze Zeichenketten zur Darstellung eines numerischen Wertes senden beziehungsweise erwarten, kann

es wünschenswert sein, einzelne Bytes seriell zu übertragen. Dafür bietet CCBASIC die Befehle PUT und GET.

```
PUT term
```

sendet den berechneten Wert eines Terms. Falls erforderlich, wird das Ergebnis zuvor auf den Byte-Wertebereich (0...255) reduziert.

```
GET variable
```

wartet auf ein seriell empfangenes Byte und speichert den Wert dann in der angegebenen Variablen.

- Weitere Schnittstellenbefehle und -funktionen

Wie beschrieben warten INPUT und GET unter Umständen endlos auf den Empfang serieller Daten. Soll ein „Aufhängen“ des Programms in dieser Art verhindert werden, kann vor jedem von INPUT oder GET durch Aufruf der Statusfunktion **RXD** ermittelt werden, ob empfangene Daten zur Verfügung stehen. Die Funktion liefert in diesem Fall den Wert -1. Ist der Schnittstellenpuffer leer, so ist das Funktionsergebnis gleich 0.

```
...
if RXD then GET thebyte
...
```

Die voreingestellte Übertragungsrate der seriellen Schnittstelle beträgt für Sender und Empfänger 9600 Bit pro Sekunde (baud). Mit dem BAUD Befehl können jedoch auch andere Raten eingestellt werden. CCBASIC enthält dafür einige vordefinierte Konstaten: **R1200**, **R2400**, **R4800**, **R9600** für die Raten 1200 bis 9600 Bit pro Sekunde.

```
BAUD R2400
```

schaltet beispielsweise Sender und Empfänger auf die Rate von 2400 Bit pro Sekunde um. Prinzipiell sind auch andere als die vordefinierten Raten, auch für Sender und Empfänger unterschiedliche, möglich. Die Übertragungsraten der seriellen Schnittstelle werden durch Teilung aus einem internen Takt des Mikroprozessors des C-Control/*BASIC* Steuercomputers abgeleitet. Der dem BAUD Befehl zu übergebende Bytewert enthält die erforderlichen Teilerwerte Nxx.

b7	b6	b5	b4	b3	b2	b1	b0
NP1	NP0	NT2	NT1	NT0	NR2	NR1	NR0

Bit 7 und 6 enthalten einen für Sender und Empfänger gemeinsamen Vorteiler NP. NP kann die Werte 1, 3, 4 und 13 annehmen. Die folgende Tabelle zeigt die dafür erforderlichen Einstellungen für NP1 und NP0:

Vorteiler	NP1	NP0
1	0	0
3	0	1
4	1	0
13	1	1

NT (Bit 3 bis 5) und NR (Bit 0 bis 2) bestimmen weitere Teilerwerte, getrennt für Sender (NT) und Empfänger (NR), entsprechend folgender Codierung:

Teiler	NT2 bzw. NR2	NT1 bzw. NR1	NT2 bzw. NR2
1	0	0	0
2	0	0	1
4	0	1	0
8	0	1	1
16	1	0	0
32	1	0	1
64	1	1	0
128	1	1	1

Die Übertragungsrate des Senders berechnet sich nach folgender Formel

Senderrate = 125000 / (NP * NT),

die des Empfängers entsprechend

Empfängerrate = 125000 / (NP * NR).

Die weiteren Schnittstellenparameter - 8 Datenbits, kein Paritätsbit, 1 Stopbit - sind fest und können nicht geändert werden.

Dateifunktionen

Die Dateifunktionen erlauben das Aufzeichnen von Meßwerten oder anderen Daten oder können zum Abspeichern von Information benutzt werden, die nach Ausfall der Betriebsspannung wieder in die Programmvariablen geladen werden sollen.

Der Speicherbereich im EEPROM-Chip nach dem Anwender-programm - meist der größte Teil - steht für diesen Zweck zur Verfügung. Der Speicherbereich wird als eine Datei verwaltet, auf die lesend oder schreibend zugegriffen werden kann, nachdem sie mit dem entsprechenden Attribut geöffnet wurde. Der Befehl zum Öffnen der Datei lautet wie folgt:

```
OPEN# FOR WRITE
```

oder

```
OPEN# FOR APPEND
```

oder

```
OPEN# FOR READ
```

Dabei bedeutet WRITE das Öffnen zum Schreiben mit Überschreiben eventueller alter Aufzeichnungen, APPEND das Öffnen zum Schreiben mit Anhängen der neuen an die alten Aufzeichnungen und READ das Öffnen zum Auslesen der Aufzeichnungen.

Es können nur Integerwerte gespeichert und gelesen werden. Jeder Wert belegt also 2 Bytes im EEPROM. Das Schreiben und Lesen erfolgt mit den Befehlen

```
PRINT# term
```

, wobei das berechnete Ergebnis des Terms gespeichert wird, beziehungsweise

```
INPUT# variable
```

, wobei variable eine definierte Integervariable des Programms bezeichnet.

Schreiben in und Lesen aus der Datei erfolgt streng sequentiell. Dafür wird intern ein Dateizeiger verwaltet, der nach jedem Zugriff um 1 erhöht wird.

Vor jedem Schreiben sollte geprüft werden, ob noch genügend Platz im EEPROM zur Aufnahme der Daten vorhanden ist. Dafür kann die Funktion **FILEFREE** abgefragt werden, die als Ergebnis die Größe des noch freien Speichers liefert (in Words). Folgendes Beispiel zeigt die Anwendung der Funktion

```
DEFINE a WORD
DEFINE b WORD
DEFINE c WORD
DEFINE blocksize 3
...
IF FILEFREE >= blocksize THEN GOSUB writeblock
...
#writeblock
   PRINT# a
   PRINT# b
   PRINT# c
RETURN
```

Vor jedem Lesen sollte geprüft werden, ob noch weitere Daten aufgezeichnet sind. Die Funktion dafür lautet **EOF** („end of file“). Ihr Ergebnis ist -1, wenn in der Datei keine weiteren Daten verfügbar sind, sonst 0. Die Abfrage der EOF Funktion sollte das Auslesen von Datenblocks in gleicher Weise rahmen, wie beim Schreiben der Daten. Zum obigen Beispiel würde also

```
IF NOT EOF THEN GOSUB readblock
...
#readblock
   INPUT# a
   INPUT# b
   INPUT# c
```

```
RETURN
```

gehören.

Nach Beenden eines Dateizugriffs sollte die Datei sofort wieder geschlossen werden. Erst dann sind die Daten vor einem Spannungsausfall oder Reset des Systems sicher. Der Befehl dafür lautet

```
CLOSE#
```

und hat keinen Parameter.

Portbefehle

- der Umschaltbefehl **TOG**

Prinzipiell erfolgt der Zugriff auf die Ports des Steuercomputers wie auf Variablen. Um einen Digitalport P einzuschalten, schreibt man

```
P = 1
```

und

```
P = 0
```

, um ihn auszuschalten.

Um den Port umzuschalten (EIN nach AUS; AUS nach EIN), kann man schreiben

```
P = NOT P
```

oder den Befehl

```
TOG P
```

benutzen. TOG steht für englisch „toggle“. Der TOG Befehl benötigt weniger Platz im EEPROM und wird schneller als die klassische NOT-P-Konstruktion ausgeführt.

Die Portvariable P darf beim TOG Befehl nur für einen einzelnen Digitalport stehen, nicht für einen Byte- oder Wordport.

- Deaktivieren eines Ports mit **DEACT**

Sobald einer Portvariablen erstmalig ein Wert zugewiesen wird, schaltet der Steuercomputer die zugehörigen Hardwarestrukturen im

Prozessorchip (Transitoren) auf Ausgangsbetrieb. Es fließt also entsprechend der angeschlossenen Schaltung Strom aus bzw. in den Prozessor (max. 10 mA zulässig!). Der Befehl

```
DEACT portvar
```

deaktiviert den angegebenen Port. Das heißt, der Port wird in einen hochohmigen Zustand geschaltet und arbeitet im Eingangsbetrieb.

Der DEACT Befehl darf auf einzelne Digitalports oder Byteports angewendet werden.

- der **PULSE** Befehl

Mit dem Befehl

```
PULSE portvar
```

wird ein Puls von einigen Millisekunden Breite am mit portvar bezeichneten Port ausgegeben. Das ist beispielsweise nützlich zum Schalten extern angeschlossener flankengetriggerter Logikschaltkreise. Steht der Port vor Ausführung des PULSE Befehls auf low (=0), wird ein High-Puls (0-1-0), ansonsten ein Low-Puls (1-0-1) ausgegeben.

Die Portvariable darf beim PULSE Befehl nur für einen einzelnen Digitalport stehen, nicht für einen Byte- oder Wordport.

Definition und Anwendung von Datentabellen

Im Standard-BASIC dienen DATA-Zeilen zum Ablegen von konstanten Datenblöcken, auf die dann sequentiell zugegriffen werden kann. CCBASIC unterstützt keine DATA-Zeilen, bietet jedoch ein weitaus flexibleres Werkzeug zur Definition und zum Zugriff auf Datenblocks. Konstante Daten können in Form von Tabellen abgelegt werden. Jede Tabelle bekommt einen Bezeichner (tablename) zugewiesen und kann beliebig viele Einträge enthalten, soweit der Programmspeicher Platz bietet. Jeder Dateneintrag (Cx) wird als Integerwert abgelegt und belegt somit zwei Bytes. Dabei können die Daten direkt im Quelltext aufgeführt werden

```
TABLE tablename C0 C1 C2 C3 ...
C4 C5 ...
... Cn
```

```
TABEND
```

oder vom CCBASIC-Compiler aus einer externen Textdatei importiert werden

```
TABLE tablename “tabfilename“
```

Die Tabellendefinitionen müssen stets am Ende eines Programms, hinter dem END Befehl stehen, da die Daten nahtlos hinter den vorangehenden Codebytes im EEPROM-Speicherchip abgelegt werden. Die Programmabarbeitung darf nie über Tabellendaten laufen, da die Daten sonst als BASIC Befehle interpretiert werden würden, was sicher zum Absturz des Systems führt.

Der Zugriff auf die Tabellendaten erfolgt mit dem Befehl

```
LOOKTAB tablename,index,variable
```

tablename bezeichnet eine gültige Tabelle, für index kann ein beliebiger Term stehen und die variable bezeichnet die Speicherzelle, in der das Ergebnis abgespeichert werden soll. Der berechnete Wert des index-Terms darf nicht negativ sein und maximal N-1 betragen, wenn die indizierte Tabelle N Einträge hat. Ergibt index den Wert 0, so wird C0 in der angegebenen Variablen gespeichert, für index gleich 1 C1 und so weiter. Folgendes Beispiel gibt den Inhalt einer Tabelle seriell aus

```
DEFINE value WORD
DEFINE i BYTE

FOR i = 0 to 3
  LOOKTAB mytab,i,value
  PRINT “mytab[“; i; “]=“; value
NEXT

END

TABLE mytab 12 -20 0 1000
TABEND
```

Am Bildschirm des Terminalprogramms sollte erscheinen

```
mytab[0]=12
mytab[1]=-20
mytab[2]=0
```

```
mytab[3]=1000
```

Besonders nützlich erweisen sich Tabellen beim Umsetzen von A/D-Werten in echte physikalische Größen. Eine Umsetzungstabelle hat dann in der Regel 256 Einträge. Der gemessene A/D-Wert geht dann als Tabellenindex in die Bestimmung der physikalischen Größe ein.

Zugriff auf die Echtzeituhr

Um den Stand der internen Echtzeituhr auszulesen und zu setzen, sind folgende globale Variablen definiert:

YEAR	Jahr (0...99)
MONTH	Monat (1...12)
DAY	Tag des Monats (1...31)
DOW	Wochentag (0=Sonntag...6=Samstag)
HOUR	Stunde (0...23)
MINUTE	Minute (0...59)
SECOND	Sekunde (0...59)

Beachten Sie bitte, daß während des Zugriffs die interne Uhr weiterläuft. Der Sekundenwert sollte daher stets zuerst ausgelesen werden. Steht er auf 59, so muß nach dem Lesen der letzten interessierenden Zeitinformation (z.B YEAR) der Sekundewert nochmals gelesen und auf =0 getestet werden. In diesem Fall ist das Auslesen der Echtzeituhr zu wiederholen, da eine neue Minute angebrochen ist (Extremfall Silvester mit Weiterschalten aller Stellen in Uhr und Datum).

Die Jahreszahl wird im C-Control System nur zweistellig abgespeichert. Sollte das Auswerten der Jahreszahl in Ihren Algorithmen eine Rolle spielen, beachten Sie bitte den bevorstehenden Wechsel von 1999 auf 2000, der sich in C-Control als Wechsel von 99 auf 0 darstellt.

Timer

Der interne 20-Millisekunden-Timer kann über den vordefinierten Bezeichner **TIMER** ausgelesen werden. Der Timer ist freilaufend und kann nicht gestellt oder rückgesetzt werden.

Weitere Befehle

- Ausgabe von Tönen mit **BEEP**

Der C-Control/*BASIC* Steuercomputer kann an einem seiner Pins (BEEP-Pin, entspricht Prozessorausgang TCMP1) Töne als Rechteckschwingungen ausgeben. Der Befehl dazu lautet

```
BEEP ton, tTon, tPause
```

Für die drei Parameter können Konstanten oder Terme eingesetzt werden. Dabei bestimmt ton die Tonhöhe nach der Formel

ton = 250000 / freq [Hz]

, tTon bestimmt die Dauer des Tons und tPause die Pause nach dem Ton. Die Einheit für die Zeitangaben beträgt 20 Millisekunden. Der Befehl

```
BEEP 568, 10, 3
```

gibt also für 10*20=200 Millisekunden einen Ton von etwa 440 Hz (Kammerton A) aus und macht danach eine Pause von 3*20=60 Millisekunden. Wenn nach einem BEEP kein weiterer BEEP folgt, kann die Pause auch auf 0 gesetzt werden. Ist für die Tonlänge 0 angegeben, wird ein Dauerton erzeugt. Der Tongenerator schaltet den Ton ein und fährt mit der Abarbeitung des BASIC-Programms fort. Mit dem Wert 0 für ton kann der Tongenerator wieder abgeschaltet werden.

- Frequenzmessung mit den Funktionen **FREQ, FREQ1 und FREQ2**

Ist am DCF77-Eingang keine Aktivantenne angeschlossen, so kann mit diesem Eingang alternativ eine Frequenzmessung erfolgen, deren Ergebnis mit der Funktion FREQ jederzeit abgefragt werden kann.

```
x = FREQ
```

Die Frequenzmessung basiert auf dem Pulszählprinzip bei einer Torzeit von 1 Sekunde. Die Messung erfolgt ständig im Hintergrund, parallel zur Abarbeitung des BASIC-Programms. Der Meßbereich reicht bis etwa 5 Kilohertz mit einem Meßfehler unter einem Prozent. Danach wird das Ergebnis zunehmend ungenauer.

Der Aufruf von FREQ1 ist identisch zu FREQ.

Der Aufruf von FREQ2 liefert den Meßwert vom zweiten Frequenzmeßeingang (bis 32000Hz).

- Stromsparmodus mit **SLOWMODE**

Anwendungen, die keine hohe Rechenleistung benötigen, können durch Aufruf des Befehls

```
SLOWMODE ON
```

den internen Takt des Mikroprozessors verlangsamen (1/16). In Kombination mit den Jumpern „LED“ und „232“ läßt sich so der Leistungsbedarf des Steuercomputers nochmals senken. Sollte im Verlauf des Programms wieder eine höhere Geschwindigkeit erforderlich sein, so läßt sich mit

```
SLOWMODE OFF
```

wieder der Ausgangszustand herstellen.

Achtung: Programme, die serielle Datenübertragungen verwenden, sollten den SLOWMODE nicht aktivieren, da die eingestellten Übertragungsraten mit dem Prozessortakt herabgesetzt werden.

Einbinden von Assemblerroutinen

Die folgenden Informationen richten sich an professionelle Anwender des C-Control/BASIC Steuercomputers und sind für die eigentliche BASIC-Programmierung nicht erforderlich!

Vorausgesetzt werden die Kenntnis des internen Aufbaus des Mikrocontrollers MC68HC05B6 und Kenntnisse in der Assemblerprogrammierung dieses Controllers. Außerdem wird ein Assembler für 68HC05-Prozessoren benötigt. An dieser Stelle sei das Buch „Motorola 68HC05“ von Zekeriya Zengin aus dem Heise-Verlag (ISBN 3-88229-034-X, Conrad Electronic Best.-Nr.: 91 91 79) empfohlen, das den Mikrocontroller vollständig beschreibt und auf

einer beiliegenden Diskette u.a. einen Assembler und zahlreiche Beispiele liefert.

Die meisten anwendungstechnischen Probleme lassen sich sicher ausschließlich durch ein BASIC-Programm lösen. Dennoch kann es vorkommen, daß für eine spezielle Aufgabe eine höhere Verarbeitungsgeschwindigkeit oder besondere Hardwarezugriffe erforderlich sind. Für diesen Fall stehen neben dem externen EEPROM-Speicherchip im Mikroprozessor selbst nocheinmal 255 EEPROM Bytes zur Aufnahme von in Assembler programmierten Routinen zur Verfügung. Diese Routinen können aus dem BASIC-Programm heraus aufgerufen werden. Der Befehl dazu lautet

```
SYS adr
```

wobei adr eine Konstante ist und die Adresse bestimmt, zu der gesprungen werden soll, beispielsweise &H101, da der interne EEPROM-Bereich an dieser Adresse beginnt. Der Assemblercode muß also per ORG-Befehl an die Adresse &H101 gelegt werden. Die Rückkehr aus einer Assemblerroutine zum BASIC erfolgt per RTS-Befehl. Der Datenaustausch zwischen BASIC und Assembler kann über die in der Datei SYSADR.INC aufgelisteten RAM-Adressen erfolgen.

Wie kommt der Assemblercode in den C-Control/*BASIC* Steuercomputer? Die in Assembler geschriebenen Zusatzroutinen werden in einer separaten Quelltextdatei (z.B. ADDONS.ASM) gespeichert. Anschließend wird der Assembler aufgerufen, daraus eine Objektcodedatei im S19-Format zu erstellen (z.B. ADDONS.S19). Lesen Sie dazu die Dokumentation zu dem Ihnen zur Verfügung stehenden Assembler. Ihr BASIC-Programm, das den SYS-Befehl enthält, muß mit dem Befehl

```
SYSCODE "ADDONS.S19"
```

den generierten Code einbinden. Der SYSCODE Befehl darf nur einmal in einem CCBASIC-Programm erscheinen und sollte am Ende, noch hinter eventuellen Tabellendefinitionen stehen.

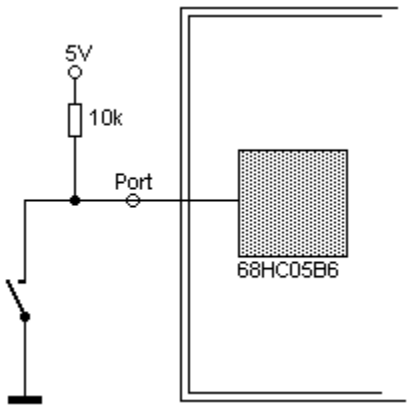
Tabellen und Abbildungen

Technische Daten (Steuercomputer)

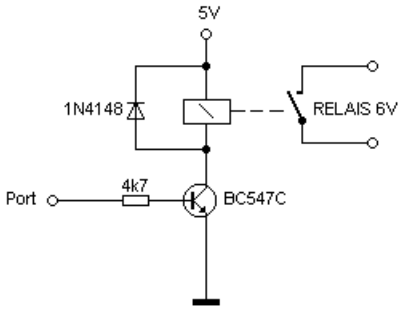
Betriebsspannung U_b	5V stabilisierte Gleichspannung, ±0,5V
Stromaufnahme	ca. 6 mA
Abmessungen	ca. 42,5 mm x 40 mm
Mikrocontroller	Motorola MC68HC05B6
	4 MHz Taktfrequenz
	6 Kilobyte maskenprogrammiertes Betriebssystem
Speicherchip für Anwenderprogramm und -daten	Microchip 24C65, serielles EEPROM mit I2C-Schnittstelle, 8k x 8 Bit
A/D-Ports	8 x 8 Bit A/D, 0...5 Volt gegen gemeinsame Masse
	Referenzspannung U_{ref} einstellbar (normal $U_b = U_{ref}$)
	Eingangsstrom ca. 10µA bei Wandlung
	absoluter Fehler ±1 Digit (= 1/256 vom Meßbereichsendwert) zuzüglich Fehler der Referenzspannung
Digitalports	16 Stück, frei als Ein- oder Ausgang programmierbar
	Pegel (0,2 mA Last an Ausgängen):
	$(U_b - 0,3V) < U_{out,high} < (U_b - 0,1V)$
	$0,1V < U_{out,low} < 0,3V$
	$(0,7*U_b) < U_{in,high} < U_b$
	$0V < U_{in,low} < (0,2*U_b)$
	maximal zulässiger Laststrom: ± 10mA

	Achtung: eine Überschreitung durch fehlende Strombegrenzung beim Anschluß von Spannungen an die Digitalports kann zur sofortigen Zerstörung des Mikrocontrollers führen!
D/A-Wandler	2 pulsweitenmodulierte Ausgänge, PWM-Rate 1953 Hz
DCF/ FREQ1-Eingang, FREQ2-Eingang	Digitalport mit 10k Pull-Up zum Anschluß einer DCF77-Aktivantenne mit Open-Collector-Ausgang
serielle Schnittstelle	RS232 mit Pegelwandler MAX232 oder Austauschtyp auf Programmieradapter
	Übertragung mit 9600 Baud, 8 Bit, 1 Startbit, 1 Stoppbit, kein Paritätsbit, kein Handshake
	Verbindung mit dem PC über Nullmodem-kabel

Anschluß eines Schaltkontaktes an einen Digitaleingang

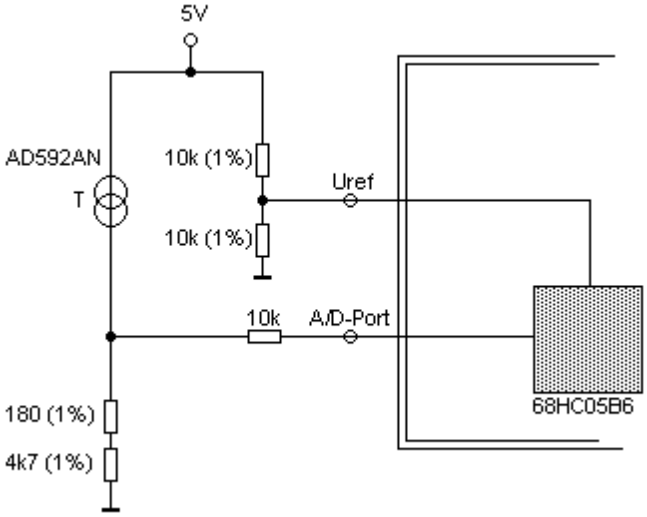


Anschluß eines Transitors zur Ansteuerung eines Relais



Anschluß eines Temperatursensors an einen A/D-Port

Das folgende einfache Beispiel zeigt den Anschluß eines Temperatursensors an die ControlUnit. Der Sensor AD592 arbeitet als temperaturabhängige Stromquelle und liefert 1µA pro Kelvin, bei 25°C (ca. 298 Kelvin) also 298 µA. Das Beispiel bietet eine Auflösung von ca. 2 Kelvin und einen theoretischen Meßbereich von 0 bis 512 Kelvin (ca. -273°C bis 239°C), was für einfache Temperaturregelungen ausreicht.



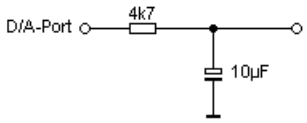
Durch die Wahl der Referenzspannung (= Vcc / 2 = 2,5V) und des Meßwiderstandes (2.5V / 0.000512A = 4882,8 Ω = ca. 4,7kΩ + 180Ω) ist die Umrechnung des A/D-Wandlerwertes in eine Grad-Celsius-Temperatur besonders einfach: T[°C] = AD * 2 - 273. Folgendes Beispiel zeigt eine sehr einfache Heizungssteuerung:

```
define sensor ad[1]
define heizung port[1]
define temperatur word

define einschwelle 20
define ausschwelle 30

#mess
  temperatur = sensor*2 - 273
  if temperatur < einschwelle then heizung = 1
  if temperatur > ausschwelle then heizung = 0
goto mess
```

Für höhere Ansprüche an Auflösung, Genauigkeit und andere Meßbereiche sind hardwareseitig ein größerer Aufwand an Bauelementen (Operationsverstärker, Referenzspannungsquelle) und softwareseitig eine Tabellendatei mit entsprechender Kennlinie erforderlich.

Anschluß eines RC-Gliedes am D/A-Port

Mit folgender einfacher Schaltung kann aus dem pulsweiten-modulierten Signal eines D/A-Ports eine analoge Gleichspannung gewonnen werden:



Für höhere Ansprüche an Restwelligkeit und Belastbakeit oder Umsetzung in andere Spannungsbereiche ist ein größerer Aufwand an elektronischen Bauelementen erforderlich.

Übersicht über die Befehle und reservierten Worte von CCBASIC

Abschließend sind an dieser Stelle nocheinmal alle reservierten Worte von CCBASIC in Tabellenform und in alphabetischer Reihenfolge zusammengefaßt, um bei syntaktischen Fragen einen schnellen Überblick zu bieten.

	Bedeutung	**Anwendungsbeispiel**
ABS(x)	Absolutbetrag von x, x: Term	y = ABS(-1)
AD[i]	spezifiziert einen A/D-Port bei DEFINE, i: 1...8	DEFINE poti AD[1]
AND	logische und bitweise UND-Verknüpfung	if (x<0) AND (y<0) ... y = x AND &H7F
APPEND	Datei öffnen zum Anhängen von Daten	OPEN# FOR APPEND
BAUD rate	Übertragungsrate der ser. Schnittstelle stellen, rate: Konstante	BAUD R2400
BEEP t,d,p	einen Ton ausgeben, t: Ton = 250000/Frequ. d: Dauer = d * 20ms p: Pause = p * 20ms	BEEP 400,25,10
BIT[i]	spezifiziert ein Userbit bei DEFINE, i: 1...192	DEFINE flag BIT[16]
BYTE[i]	spezifiziert ein Userbyte bei DEFINE,	DEFINE x BYTE[1]

	i: 1...24	
BYTEPORT[i]	spezifiziert einen 8-Bit-Digitalport bei DEFINE, i: 1...2	DEFINE leds BYTEPORT[1]
&B...	Angabe einer Binärzahl	&B1010000
CLOSE#	Datei schließen	CLOSE#
DAY	Tag-Wert der Echtzeituhr	DAY = 1 PRINT DAY
DA[i]	spezifiziert einen D/A-Port bei DEFINE, i: 1...2	DEFINE v DA[2]
DEACT p	deaktiviert einen Digitalport, p: Portvariable	DEACT led
DEFINE	Definition eines Bezeichners für einen Port, eine Variable oder eine Konstante	DEFINE sensor AD[1] DEFINE flag BIT[1] DEFINE limit 1000
DOW	Wochentag-Wert der Echtzeituhr	IF DOW=1 THEN PRINT “MONTAG“
ELSE	Alternativzweig bei einer bedingten Programmausführung	IF x=0 THEN y=0 ELSE y=5
END	Programmende	IF x THEN END
EOF	Test auf Dateiende beim Auslesen der Datei	IF EOF THEN GOTO xxx
FILEFREE	Test auf freien Speicherplatz beim Schreiben der Datei	IF FILEFREE THEN GOTO xxx

FOR	Programmschleifen	FOR i=1 TO 10
FREQ FREQ1 FREQ2	Abfrage der Frequenzmeßeingänge	x = FREQ
GET v	Byte von der seriellen Schnittstelle lesen, v: Variablenbezeichner	GET zeichen
GOSUB l	Sprung zu einer Unterroutine, l: Label-Bezeichner	GOSUB proc
GOTO l	Programmsprung, l: Label-Bezeichner	GOTO start
&H...	Angabe einer Hexzahl	&H1FFE
HOUR	Stunden-Wert der Echtzeituhr	PRINT HOUR
IF	Bedingte Programmausführung, nach IF folgt der Bedingungsterm	IF x < y THEN GOTO ready
INPUT v	Einlesen eines Zahlenwertes von der ser. Schnittstelle, v: Variablenbezeichner	INPUT x
INPUT# v	Einlesen eines Zahlenwertes aus der Datei, v: Variablenbezeichner	INPUT# x
INTERRUPT l	Definieren einer Interruptroutine l: gültiges Label	INTERRUPT event

LOOKTAB t,i,v	Laden eines Wertes aus einer Tabelle, t: Tabellenbezeichner i: Index v: Variablenbezeichner	LOOKTAB kty, mess, temp
MAX(x,y)	Ermittlung des größeren Wertes, x,y: Terme	z = MAX(x, 10)
MIN(x,y)	Ermittlung des kleineren Wertes, x,y: Terme	z = MIN(x, 10)
MINTUE	Minuten-Wert der Echtzeituhr	PRINT MINUTE
MOD	Operator für Modulodivision	x = 10 MOD 3
MONTH	Monats-Wert der Echtzeituhr	PRINT MONTH
NAND	logische und bitweise UND-Verknüpfung mit anschließender Negation	x = a NAND b
NOR	logische und bitweise ODER-Verknüpfung mit anschließender Negation	x = a NOR b
NOT	logische und bitweise Negation	y= NOT x
OFF	vordefinierte Konstante für 0	led=OFF
ON	vordefinierte Konstante für &HFFFF	led=ON

ON x GOSUB	Unterroutinenaufruf in Abhängigkeit von x	ON sel GOSUB p1, p2, p3
ON x GOTO	Programmsprung in Abhängigkeit von x	ON sel GOTO p1, p2, p3
OPEN# FOR m	Datei mit Modus m öffnen, m: APPEND, WRITE oder READ	OPEN# FOR WRITE
OR	logische und bitweise ODER-Verknüpfung	x = a OR b
PAUSE t	Programm-unterbrechung für t*20 Millisekunden	PAUSE 50
PORT[i]	spezifiziert einen Digitalport bei DEFINE, i: 1...16	
PRINT x	Ausgabe von Werten und Texten über die serielle Schnittstelle, x: Term oder Text in Anführungszeichen	PRINT “A=“; a, “B=“; b
PRINT# x	Aufzeichnen eines Wertes in der Datei, x: Term	PRINT# a
PULSE p	Ausgabe eines Pulses an einem Digitalport, p: Portvariable	PULSE clock
PUT x	Ausgabe eines Bytes über die serielle Schnittstelle, x: Term	PUT ch

RAND	Integer-Zufallszahl generieren	x = RAND
RANDOMIZE x	Zufallszahlengenerator neu initialisieren	RANDOMIZE TIMER
RETURN	Rückkehr aus einer Unterroutine	RETURN
RETURN INTERRUPT	Rückkehr aus einer Interruptroutine	RETURN INTERRUPT
RXD	Test, ob ein Byte seriell empfangen wurde	IF RXD THEN GET ch
READ	Datei öffnen zum Auslesen von Daten	OPEN# FOR READ
SECOND	Sekunden-Wert der Echtzeituhr	PRINT SECOND
SHL	Operator für bitweises Linksschieben	a = a shl 2
SHR	Operator für bitweises Rechtsschieben	a = a shr 2
SLOWMODE m	Prozessortakt/16, m: ON oder OFF	SLOWMODE ON
STEP	Angabe der Schrittweite bei der FOR-Schleife	FOR i=100 TO 0 STEP -5
SYS a	Aufruf eines Systemprogramms, a: Adresse	SYS &H101
SYSCODE	Angabe von Systemcodes oder einer S19-Datei	SYSCODE “TEST.S19“
SYSEND	Abschluß bei mit SYSCODE direkt eingegebenen Bytes	SYSCODE &H81 SYSEND

TABLE	Definition einer Datentabelle, direkt oder per Tabellendatei	TABLE kty “KTY10.TAB“
TABEND	Abschluß bei mit TABLE direkt eingegebenen Werten	TABLE tab 10 5 -3 TABEND
TIMER	Abfrage des 20-ms Timers	x = timer
TO	Angabe des Endwertes bei der FOR-Schleife	FOR i=1 TO 10
TOG p	Umschalten eines Digitalports, p: Portvariable	TOG led
WAIT condition	Warten auf Eintreten einer Bedingung (condition <>0) condition: Term	WAIT key
WORD[i]	spezifiziert einUserword bei DEFINE, i: immer 1	DEFINE x WORD[1]
WORDPORT[i]	spezifiziert einen 16-Bit-Digitalport bei DEFINE, i: immer 1	DEFINE all WORDPOR[1]
WRITE	Datei öffnen zum Schreiben neuer Daten	OPEN# FOR WRITE
YEAR	Jahr-Wert der Echtzeituhr	PRINT YEAR