12 C-Control-Plus mit Grafikoberfläche

Der C-Control Steuercomputer kann nicht nur in CC-Basic programmiert werden, sondern es gibt auch die Version C-Control-Plus mit einer vereinfachten graphischen Programmieroberfläche. Sie eignet sich auch für Anwender, die sich nicht in die Syntax einer Programmiersprache einarbeiten möchten. Ein Programm entsteht durch Zusammenfügen von Programmzellen mit graphischen Funktionsblöcken. Für jeden Funktionsblock existiert ein eigenes Symbol, das aus einem Menü der verfügbaren Blocktypen ausgewählt werden muß. Jedes Symbol kann auf der Grafikoberfläche seiner Programmzelle frei plaziert und mit anderen Symbolen verknüpft werden. Die meisten Funktionsblöcke erhalten Parameter, die durch Einstellung ihrer Eigenschaften eingestellt werden müssen.

Die Hardwareeigenschaften des C-Control-Plus unterscheiden sich nicht von denen des C-ControlBASIC Steuercomputers. Deshalb gelten die in den ersten Kapiteln beschriebenen technischen Eigenschaften unverändert auch für diese Version. Die im folgenden beschriebenen Programme sind zum Teil direkte Umsetzungen von weiter oben beschriebenen BASIC-Programmen mit den Mitteln der graphischen Benutzeroberfläche, so daß direkte Vergleiche der Programmiersysteme nicht schwerfallen. Fast alle Aufgaben lassen sich auf beide Arten lösen, wobei von Fall zu Fall mal die eine und mal die andere Programmietumgebung zu der einfacheren Lösung führt.

Die einzige wirkliche Einschränkung von C-Control-Plus gegenüber C-Control/BASIC betrifft die Ausfbhrung von Maschinenprogrammen, die in der graphischen Plus-Version nicht möglich ist. Auf der anderen Seite sind aber insbesondere die Ein- und Ausgabeelemente LCD-Display, Zifferntastatur und Beeper wesentlich komfortabler einsetzbar, da der Anwender sich nicht mehr um die hardwarenahe Ansteuerung kümmern muß.

12.1 Digitale Ein / Ausgaben

Digitale Ports können in CC-Plus als Bitports , Byteports mit acht Einzelbits oder als Wordports mit einer Breite von 16 Bit gewählt werden. Das erste Anwendungsbeispiel soll einen einfachen Blinkgeber realisieren, wie er bereits in Kap. 3.3 mit dem BASIC-Listing 3.2 vorgestellt wurde. Hier wird der

Byteport 1 als Ausgabeport eingesetzt. Die Portnummer muß als Eigenschaft des Parts eingestellt werden. Port 1 ist der voreingestellte Wert, den man bei Bedarf ändem kann.

Abb. 12.1 zeigt das Programm mit drei Programmzellen. Die erste Programmzelle ON enthält eine Konstantenzuweisung des Zustands 1 an den Ausgangspott. Port 1 wird damit eingeschaltet. In der selben Programmzelle kann man auch den Wartebefehl unterbringen. Hier wird die Konstante 25 dem Funktionsblock PAUSE zugewiesen. Dabei ergibt sich eine Wartezeit von 25 x 20 ms, also von 0,5 Sekunden.

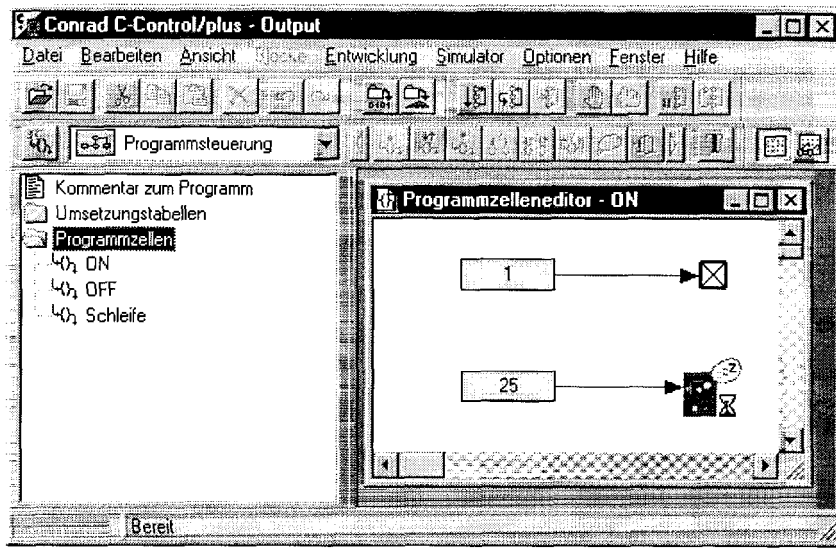


Abb. 12.1 Ein einfacher Blinkgeber

Das Ausschalten des Ausgabeports erfolgt über eine eigene Programmzelle. Prinzipiell kann man die Reihenfolge der Programmschritte innerhalb einer Zelle nicht eindeutig festlegen. Sie hängt u.a. von der Reihenfolge der Eingaben ab, die später am graphischen Aufbau nicht mehr erkennbar ist. Genaugenommen ist es sogar problematisch, den Wartebefehl in der selben Programmzelle unterzubringen, was hier nur aus Gründen der übersichtlichen Darstellung getan wurde. Wird dieser nämlich einmal vor und einmal nach der Portausgabe ausgeführt, dann kommt es zu einer Fehlfunktion, deren Ursache nur schwer aufzuspüren ist.

Wenn also ein Ablauf mit einer genau definierten Reihenfolge von Aktionen erwünscht ist, sollten mehrere Programmzellen verwendet werden. Hier wird deshalb eine zweite Programmzelle ON gebildet. Man kann den Inhalt der ersten Zelle markieren und in die zweite Zelle kopieren. Dann genügt es, mx den Wert der ersten Konstanten in 0 zu ändem (vgl. Abb. 12.2).

Der Aufbau einer Endlosschleife erfordert ebenfalls eine eigene Programmzelle, die nur eine Verzweigung enthält (vgl. Abb. 12.3). Das Verzweigungsziel, hier also die Zelle ON, muß als Eigenschaft der Verzweigung eingegeben werden.

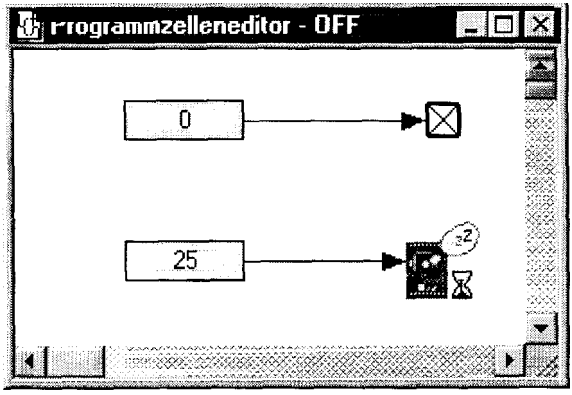


Abb. 12.2 Ausschalten des Ports

Listing 12.3 Aufbau der Endlosschleife

Das Programm läßt sich auch anders aufbauen, wenn man die logische Funktion Invertieren (NOT) einsetzt. Abb. 12.4 zeigt eine Lösung. Hier wird zusätzlich eine Bitvariable eingesetzt, deren Inhalt immer wieder invertiert wird. Durch Übergabe an den Ausgangsport entsteht das erwünschte Rechtecksignal.

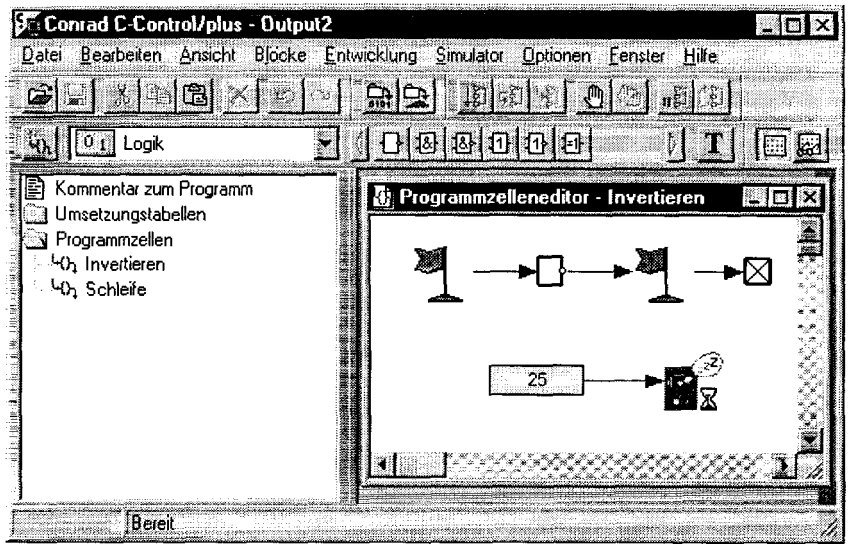


Abb. 12.4 Aufbau eines Blinkers mit Inverter

Prinzipiell lassen sich auch kompliziertere Bitmuster über Einzelhitausgaben erzeugen. Es ist jedoch oft einfacher, Byteports zu verwenden, wie es bereits in Kap. 3.3 demonstriert wurde. Ein mit dem dort vorgestellten Programm nach Listing 3.4 vergleichbares Programm wird in Abb. 12.4 gezeigt. Hier werden aufsteigende Bitmuster mit den Werten 1,2,4, 8, 16,32,64 und 128 ausgegeben, so daß em einfaches Lauflicht mit acht Lampen gesteuert werden kann.

Alle Ausgaben und die zugehörigen Wartebefehle wurden hier in eine Programmzelle gesetzt. Dieses Verfahren eignet sich für schnelle Versuche, ist jedoch problematisch, wenn Änderungen erforderlich werden. Insbesondere ist es nicht möglich, nachträglich ein zusätzliches Bitmuster an einer definierten Position einzufügen. Der bessere Programmierstil ist es also, für jede Ausgabe eine eigene Programmzelle vorzusehen. Dieses aufwendigere Verfahren wurde hier deshalb nicht gewählt, weil es nur schwer übersichtlich zu dokumentieren ist.

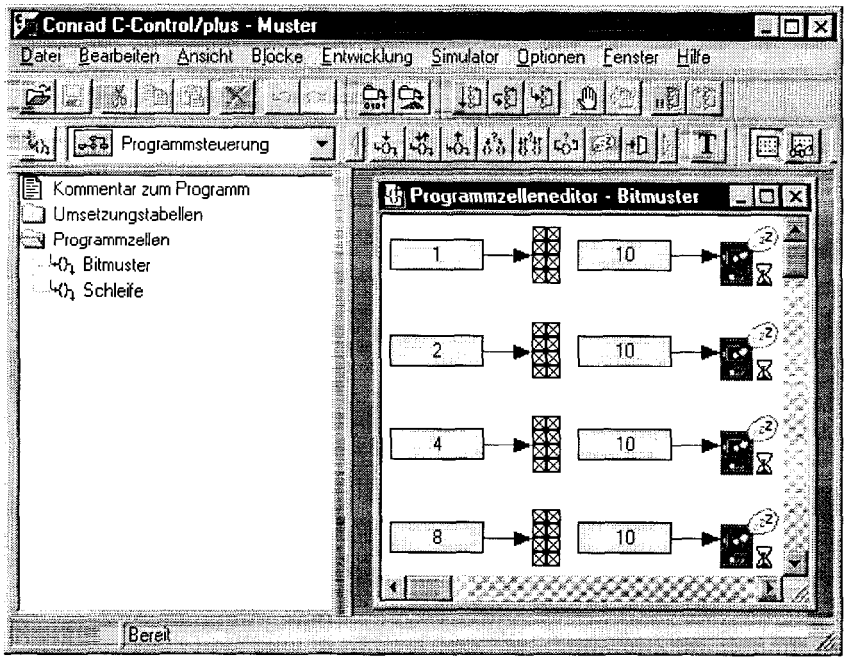


Abb. 12.5 Bitmusterausgaben an einen Byteport

Digitale Eingaben sind z.B. erforderlich, wenn man Schalter, Taster oder Reed-Kontakte abfragen will. Eingaben können logisch verknüpft werden und bestimmte Ausgangszustände beeinflussen. Ein einfaches Anwendungsbeispiel ist ein bistabiles Relais. Über einen Tastschalter kann das Relais in den AN- oder in den AUS-Zustand geschaltet werden, wobei jede folgende Betätigung den Zustand wieder umschaltet. Es gibt zwar bistabile Relais als fertige Bauteile, mit einer logischen Steuerung lassen sich jedoch auch normale Relais einsetzten, und man kann leicht erweiterte Eigenschaften definieren.

Mit dem Programm nach Abb. 12.6 soll ein Tastschalter am Eingang 9 angeschlossen werden und ein Relais am Ausgang 1 steuern. Digitale Eingänge der Control-Unit sind im Ruhezustand ON, so daß ein Taster gegen Masse angeschlossen werden muß. Jede Betätigung setzt den Port 9 in den OFF-Zustand. Das Programm soll auf diese negative Flanke reagieren und das Relais umschalten, Dazu wird der Funktionsblock WARTEN AUF WERT eingesetzt, der den Programmablauf solange verzögert, bis ihm ein Wert ungleich Null zugewiesen wird. Da ein OFF-Pegel ausgewertet werden soll, muß das Signal am Eingangsport invertiert werden.

Jede Betätigung der Taste führt zu einem kompletten Durchlauf der Programmzelle. Dabei wird jedesmal der Zustand eines Bitspeichers invertiert und an den Ausgangsport ausgegeben, so daß das Relais umschaltet. Ein zusätzlicher PAUSE-Befehl sorgt für die notwendige Tastenentprellung. In einer eigenen Programmzelle AUS werden der Inhalt dieses Bitspeichers und der Ausgangspott zur Initialisierung nach dem Programmstart in den OFF-Zustand versetzt. Eine dritte Programmzelle dient zum Aufbau einer Endlosschleife mit einer Verzweigung zur Programmzelle UMSCHALTER. Jede kurze Betätigung des Tasters führt zu einem Umschalten des Relais. Eine dauerhafte Betätigung erzeugt ein selbständiges Blinken.

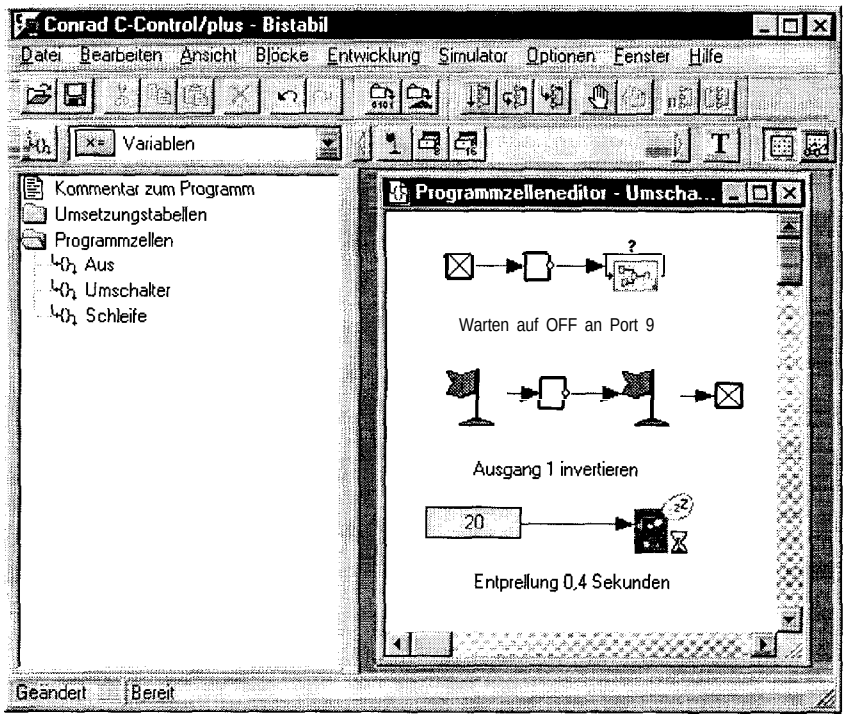


Abb. 12.6 Steuerprogramm für ein bistabiles Relais

Mit umfangreichen logischen Verknüpfungen digitaler Eingangssignale lassen sich z.B. Alarmanlagen programmieren. Das Programmbeispiel nach Abb. 12.7 stellt eine direkte Umsetzung der in Kap. 3.6 vorgestellten Alarmanlage dar. Die Verbindung mit den einzelnen Kontakten erfolgt daher nach dem Schaltplan in Abb. 3.16. Die Anlage verknüpft acht Meldeleitungen und kann über einen Freigabeschalter aktiviert werden.

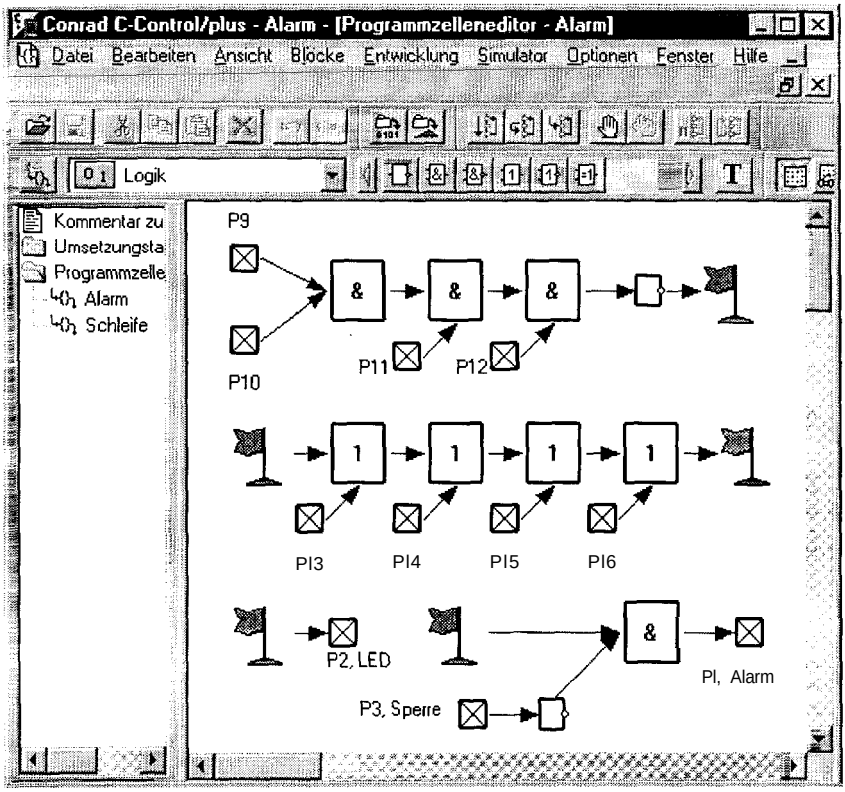


Abb. 12.7 Eine Alarmanlage mit acht Eingängen

12.2 Analoge Ein- und Ausgänge

Analoge Eingänge der Conrol-Unit liefern g-Bit-Werte (Bytes) im Bereich 0 bis 255. Entsprechend werden an die analogen Ausgänge Bytewerte übergeben. Meßwerte lassen sich auf verschiedene Weise bearbeiten und ausgeben.

Ein erstes Anwendungsbeispiel bildet einen analogen Inverter. Eine ansteigende Spannung am Analogeingang 1 wird als abfallende Spannung an den Analogausgang 1 ausgegeben, indem alle Meßwerte von 255 subtrahiert

werden. Im Ergebnis führt ein Eingang an 0 V zu einer Ausgangsspannung von 5 V, während umgekehrt die höchste Eingangsspannung von 5 V (bei einer Referensspannung von 5 V) zur Ausgangsspannung 0 V führt. Die Programmzelle nach Abb. 12.8 wird in einem Gesamtprogramm in einer Endlosschleife immer wieder ausgeführt.

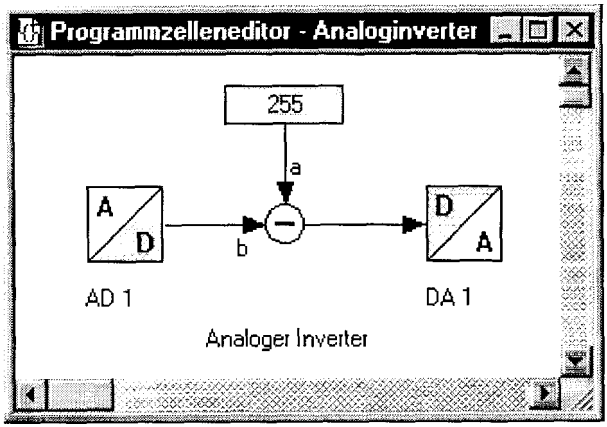


Abb. 12.8 Invertierung analoger Signale

Die analogen PWM-Ausgänge der Control-Unit lassen sich vorteilhaft für Leistungssteuerungen anwenden. In Kapitel 7.1 wurde ein zweikanaliger Dimmer für Gleichstomanwendungen beschrieben. Ein entsprechendes Schaltbild wird in Abb. 7.2 gezeigt. Das erforderliche Steuerprogramm wird hier nun graphisch umgesetzt. Abb. 12.9 zeigt das Programm mit seinen Fallunterscheidungen zur Auswertung der einzelnen Steuertasten. Die Tasten an den Eingangsports 1 bis 4 steuern über ihre aktiven OFF-Pegel entsprechende Unterprogrammauthufe.

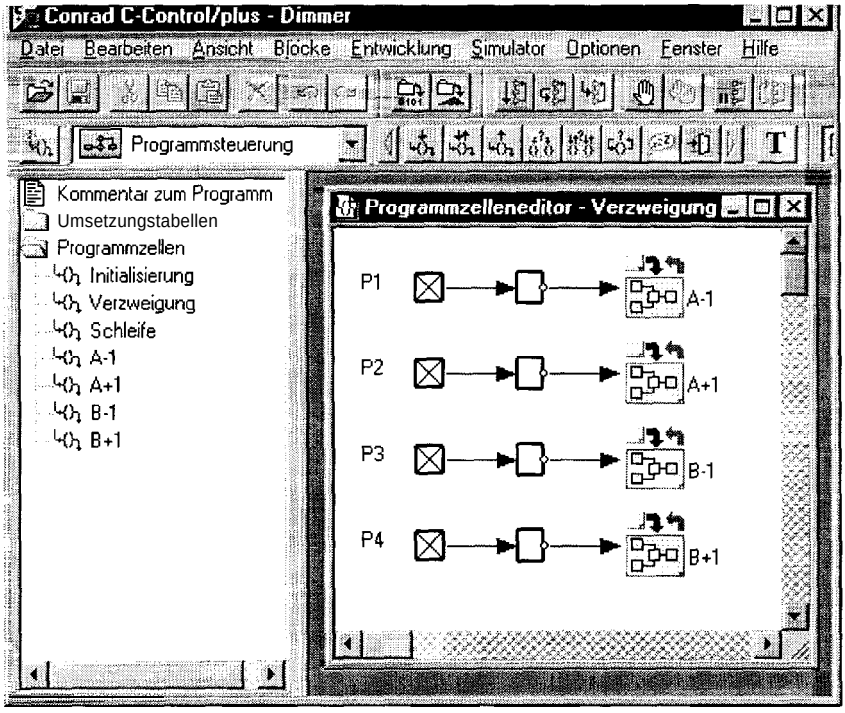


Abb. 12.9 Ein Zweikanal-Dimmer mit vier Steuertasten

Jedes der vier Unterprogramme erhöht oder erniedrigt entsprechende Variablen und steuert die Angabe an den zugehörigen PWM-Ausgang. Abb. 12.10 zeigt das Unterprogramm A+l zur Erhöhung der Ausgangsspannung an Kanal A. Über die MINIMUM-Funktion wird der Wertebereich bis 255 begrenzt, um Überläufe zu verhindern. Falls der alte Wert bereits 255 war, wird er durch 254 ersetzt. Nach der Erhöhung um Ems liegt wieder der Maximalwert 255 vor.

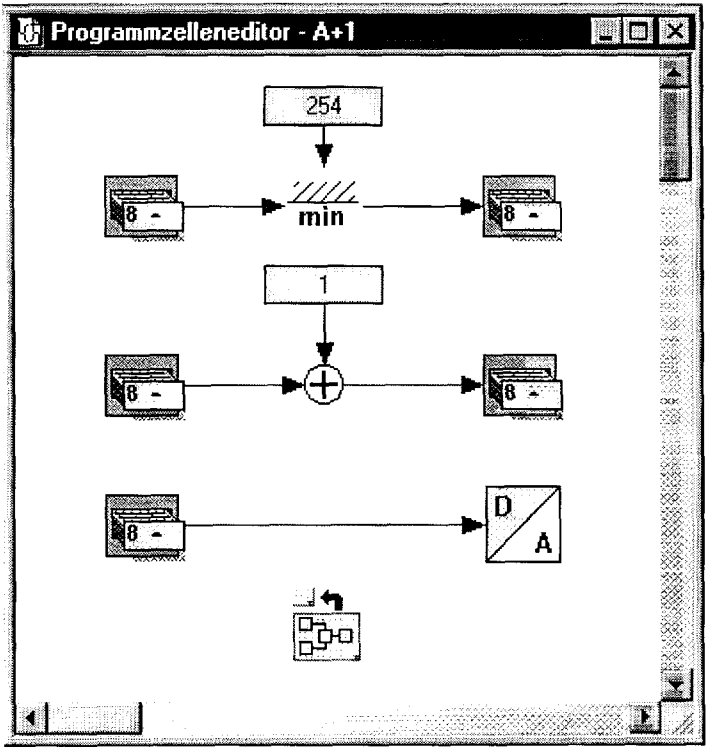


Abb. 12.10 Unterprogramm zur Spannungserhöhung

Analoge Eingänge können z.B. Potis abfragen, um bestimmte Parameter an ein Programm zu übergeben. Im folgenden Beispiel soll ein einfacher Zeitschalter über ein Poti eingestellt werden. Abb. 12.11 zeigt den Aufbau des Programms. Der digitale Ausgang 1 wird zunächst eingeschaltet. Dann wird die Spannung an AD 1 gemessen und dem PAUSE-Block zugewiesen. Nach Ablauf der eingestellten Wartezeit wird der digitale Ausgang schließlich wieder ausgeschaltet. Das Programm benötigt keine Schleife, da es direkt über die Starttaste der Control-Unit ausgelöst werden kann.

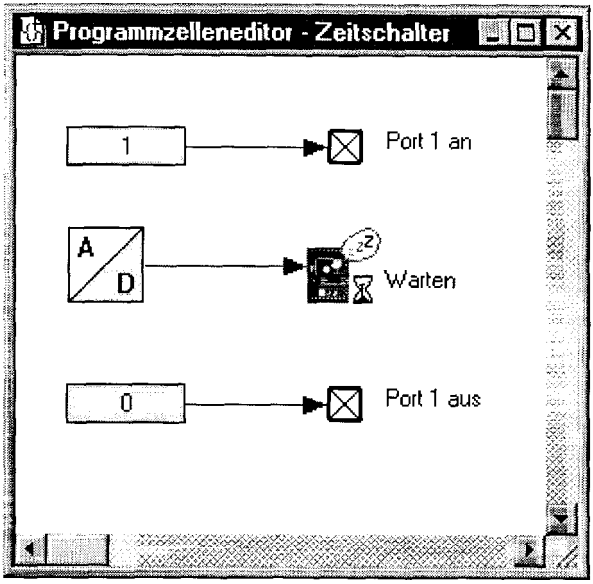


Abb. 12.11 Ein Zeitschalter mit Potisteuerung

Für Test- und Prüfzwecke kann es sinnvoll sein, Änderungen einer Spannung hörbar zu machen. Hierzu bietet sich der BEEP-Ausgang an, der direkt einen kleinen Piezo-Schallgeber ansteuern kann. Der Funktionsblock BEEP erhält wahlweise die drei Parameter Frequenzfaktor, Tonlänge und Pause. Die Frequenz wird über einen Teilerfaktor festgelegt und steigt daher mit kleinerem übergabewert an. Damit eine steigende Spannung auch durch eine höhere Frequenz angezeigt wird, muß eine Umrechnung durchgeführt werden. Der AD-Wandler liefert Ergebnisse im Bereich 0 bis 255. Subtrahiert man diese von 512, ergibt sich für die Tonhöhe ein Variationsbereich von ca. einer Oktave. Änderungen sind in jedem Spannungbereich gut hörbar. Abb. 12.12 zeigt das einfache Programm.

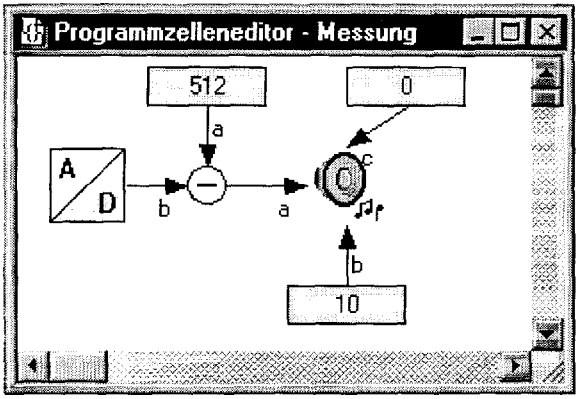


Abb. 12.12 Akustische Ausgabe von Spannungswerten

Die Tonausgabe über den BEEP-Ausgang kann auch für die Ausgabe einfacher Tonfolgen oder Melodien benutzt werden, die sich direkt als Eigenschaften des Ausgangs eintragen lassen. Der Anwender füllt dazu nur eine Liste mit Tonhöhen, Ton- und Pausenlangen aus.

f2.3 LCD-Dispiay und Zitferntastatur

C-Control-Plus enthält bereits Funktionsblöcke zur Ansteuerung eines ein- oder zweizeiligen LCD-Displays mit je 8 Zeichen und eines 1 Oer-Ziffernblocks. Beide Elemente befinden auch auf dem Application-Board zum System, können aber auch selbst angeschlossen werden. Eine besondere Starke der graphischen Programmieroberfläche ist es, daß der Anwender sich nicht mehr um die Artsteuerung dieser EimAusgabeelemente kümmern muß. Es genügt vielmehr, die entsprechenden Symbole zu wählen, wobei im Hintergrund der erforderliche Code generiert wird.

Das LCD-Display wird hier im 4-Bit-Modus (DO bis D3 bleiben frei) angesteuert und kommt daher mit einem Byteport aus. Das Display belegt deshalb nur den Byteport 2, so daß der komplette Byteport 1, also die digitalen Leitungen 1 bis 8 für andere Zwecke frei bleiben. Es gilt folgende Anschlußbelegung:

P9	D4
Pl0	D5

Pl1	D6
Pl2	D7
Pl3	R/W
Pl4	RS
Pl5	E
Pl6	N.C.

Ein erstes Anwendungsbeispiel soll Meßwerte von zwei analogen Eingängen anzeigen (vgl. Abb. 12.14). Die Software bindet automatisch den erforderlichen Ansteuerungscode mit ein. Ein auszugebender Text muß als Eigenschaft des Displays eingegeben werden. Für variable Ausgaben wie z.B. Zahlen werden Doppelkreuze # als Platzhalter eingefigt. Da mehrere Platzhalter möglich sind, erhält das Display als Mündungsblock mehrere logische Eingänge. Abb. 12.13 zeigt die Festlegung der Display-Eigenschaften fbr den Fall, daß zwei Analogausgänge abgefragt und angezeigt werden sollen. Hier wird nur eine Zeile verwendet.

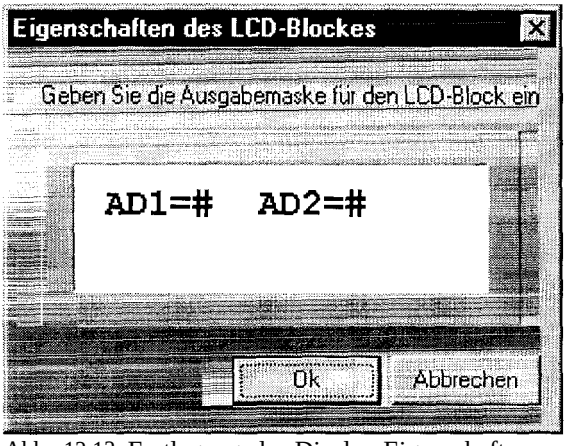


Abb. 12.13 Festlegung der Display-Eigenschaften

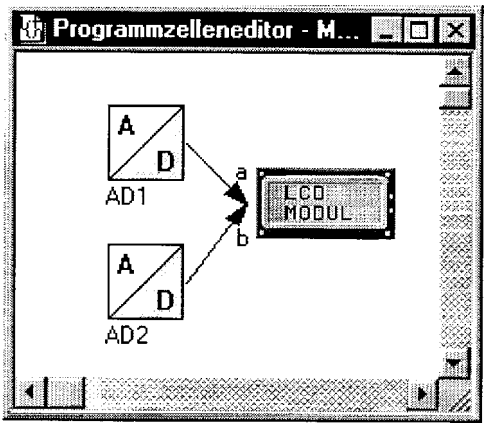


Abb. 12.14 Display-Ausgabe von zwei Meßwerten

Analogwerte werden in diesem Beispielprogramm unverarbeitet als Zahlen im Bereich 0 bis 255 angezeigt. In vielen Fällen ist noch eine Umrechnung oder Umformung erforderlich. Mit der graphischen Programmieroberfläche des C-Control-Plus ist es z.B. sehr einfach, kleine Anzeigegeräte mit Sensoren zu programmieren. Der Umgang mit Sensoren wurde bereits im Kap. 6 beschrieben. Abb. 12.15 zeigt das Beispiel eines Luftfeuchtemessers. Hier wird der Rechenblock TABELLE eingesetzt, um Meßwerte des AD-Wandlers zu interpretieren. Dazu wurde die Tabellendatei HYG.TAB geladen. Einfache Tabellen fl.ir eigene Sensoren lassen sich auch direkt editieren.

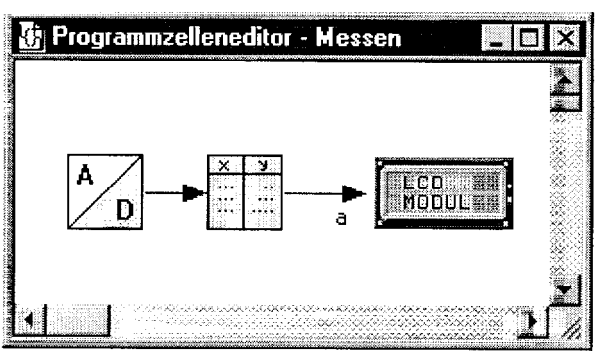


Abb. 12.15 Ein Luftfeuchte-Messer mit LCD-Ausgabe

Ebenso einfach ist der Aufbau einer Uhrenanzeige. Die einzelnen Zeit- und Datumsinformationen müssen einzeln an das Display übergeben werden. Abb.

12.16 zeigt das Programm. Verwendet man eine aktive DCF77-Antenne, dann erhält man eine Funkuhr. Der DCF-Eingang kann alternativ auch zur Messung von Frequenzen eingesetzt werden.

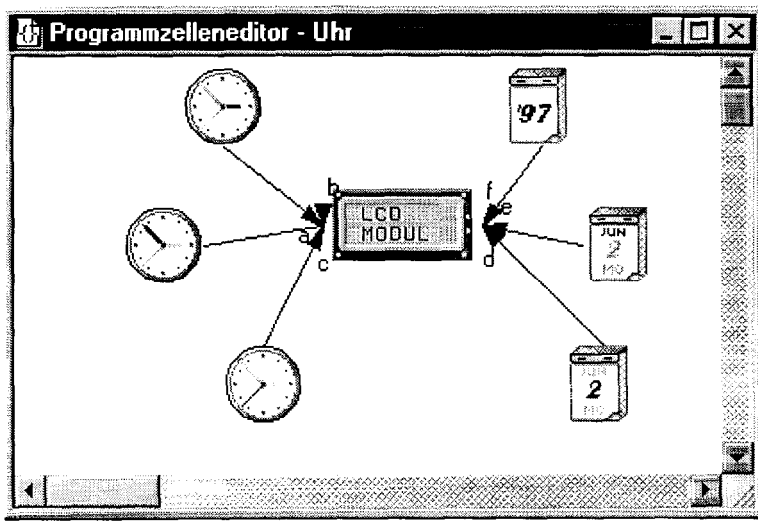


Abb. 12.16 Eine LCD-Uhr

Außer einer einfachen Ausgabeeinheit benötigt man fir viele Anwendungen auch Eingaben, z.B. über eine Zifferntastatur. Der von der graphischen Programmieroberfläche unterstützte Ziffernblock, wie er z.B. am Application-Board vorhanden ist, muß wie in Abb. 4.3 über einen Widerstands-Spannungsteiler mit dem Analogeingang 8 verbunden werden. Der vom zugehörigen Funktionsblock gelieferte Tastaturcode sollte zunächst in eine Variable geschrieben werden. Die Ergebnisse entsprechen den ASCII-Zeichen der Tasten. Solange keine Taste gedrückt wird, ist das Ergebnis Null. Man kann daher über entsprechende Vergleiche die notwendigen Programmverzweigungen vornehmen.

Abb. 12.17 zeigt ein Anwendungsbeispiel der Zifferntastatur. Hier soll die Kanalumschaltung eines Meßgeräts mit mehreren Eingängen über die Tasten gesteuert werden. Man muß beachten, daß der analoge Eingang 8 bereits durch die Tastatur belegt ist. Hier werden die Kanäle 1 bis 3 zur Auswahl gestellt. Die entsprechenden Tasten liefern die Codes 50-52. Durch Vergleiche wird die Verzweigung in die einzelnen Unterprogramme gesteuert. Das Programm

ist leicht erweiterbar und könnte z.B. auch die digitalen Eingänge und den Frequenzeingang berücksichtigen.

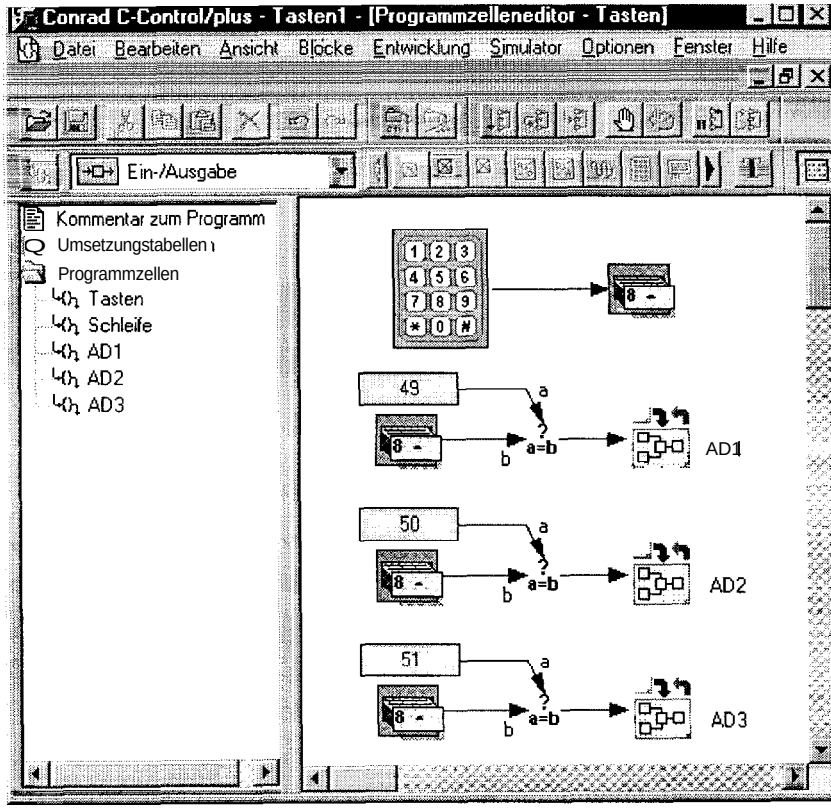


Abb. 12.17 Umschaltung von Meßbereichen über die Tastatur.

Will man die Tastatur nicht zur Einzeltastensteuerung, sondern zur Eingabe von Werten verwenden, dann ist ein etwas größerer Rechenaufwand erforderlich, um die einzelnen Ziffern zu einer Zahl zusammenzufügen. Das folgende Beispiel beschreibt die Eingabe eines Zahlenwertes im Bereich 0 bis 255 zur Steuerung eines analogen Ausgangs. Der Ausgabewert soll gleichzeitig zur Kontrolle auf dem LCD-Display sichtbar werden. Da die Anzahl der Eingabestellen nicht festliegt, soll die Doppelkreuz-Taste # als Entertaste, also zur Bestätigung der vollständigen Eingabe dienen.

Zur Initialisierung wird zunächst die WERT-Varibale auf Null gesetzt. Die eigentliche Tastenabfrage erfolgt in der Programmzelle ABFRAGE. Solange keine Taste gedrückt wurde, verzweigt das Programm nach WEITER, wo nur ein Wartebefehl zur Tastenentprellung steht. Ein Tastaturcode von 35 zeigt die Doppelkreuz-Taste an und führt in die Ausgabe. Jeder andere Tastaturcode muß als Ziffer interpretiert werden und führt in die Programmzelle AUSWERTUNG.

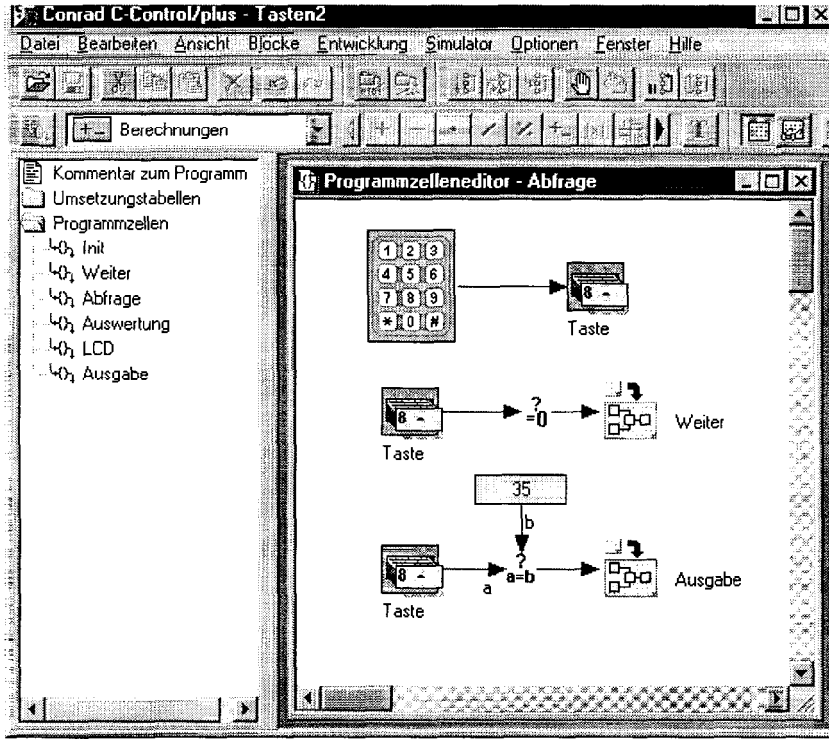


Abb. 12.18 Abfrage der Zifferntastatur

Zur Auswertung einer neuen Taste müssen die zuerst eingegebenen Ziffern zunächst um eine Stelle nach links geschoben werden. Dies geschieht durch eine Multiplikation mit 10. Aus dem Tastaturcode wird dann die Ziffer gewonnen, indem eine Umwandlung des ASCII-Wertes in eine Zahl erfolgt. Da die Ziffern 0 bis 9 als ASCII-Zeichen 48 bis 57 dargestellt werden, braucht

man hier nur den Wert 48 zu subtrahieren. Der so umgewandelte Ziffernwert wird dann zum alten Zahlenwert addiert. Jede neu eingegebene Ziffer wird in der Programmzelle LCD dargestellt, die wiederum zur Auswertung der nächsten Taste nach WEITER verzweigt. Erst die Bestätigungstaste # führt in die Programmzelle AUSGABE, in der der eingegebene Zahlenwert an den Analogausgang gegeben wird. Ein abschließende Verzweigung zu INIT erlaubt die nächste Eingabe.

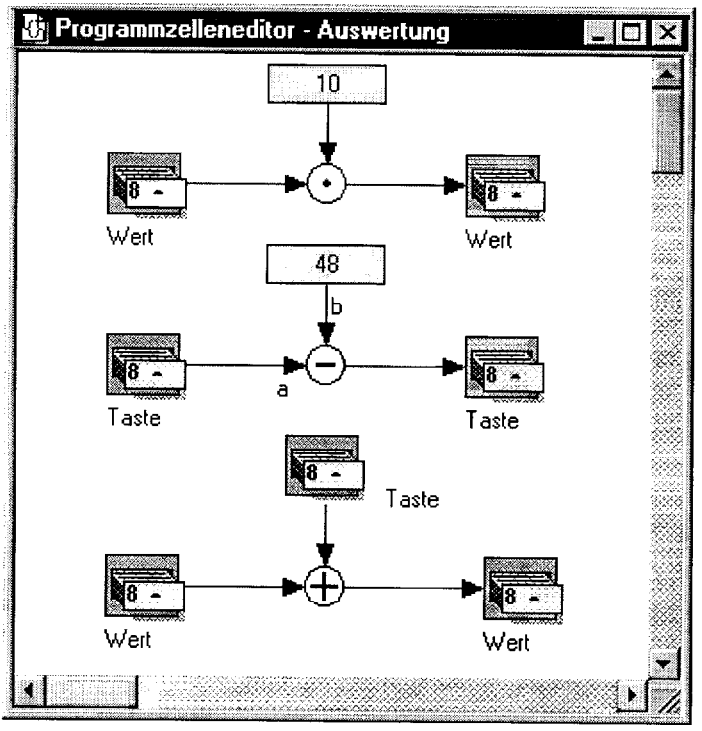


Abb. 12.19 Auswertung von Zifferntasten

12.4 Die serielle Schnittstelle

Die technischen Grundlagen der seriellen Schnittstelle wurden bereits in Kap. 9 vorgestellt. Hier sollen nun einige Anwendungsbeispiele gezeigt werden, die

die graphische Programmierung mit den Funktionsblöcken zur Schnittstelle demonstrieren. Grundsätzlich können einfache Textausgaben über die Schnittstelle geleitet werden, so daß es sehr einfach ist, z.B. Meßwerte an ein Terminal zu senden. Umgekehrt ist es auch möglich, Texte zu empfangen. Für viele technische Anwendungen werden dagegen oft die schnelleren Einzelbyte-Übertragungen verwendet.

Als erstes soll das einfache Interface nach Listing 9.1 mit C-Control-Plus umgesetzt werden. Die Control-Unit verwendet einen Ausgangspott mit den Pottleitungen 1 bis 8 und einen Eingangsport mit den Leitungen 9 bis 16. Jede Byte-Ausgabe wird vom Interface mit einem gelesenen Byte vom Eingabepott beantwortet. Entsprechend den BASIC-Befehlen PUT und GET existieren auch in der Plus-Programmietumgebung Funktionsblöcke zur Verarbeitung von Einzelbytes. Abb. 12.20 zeigt das einfache Programm.

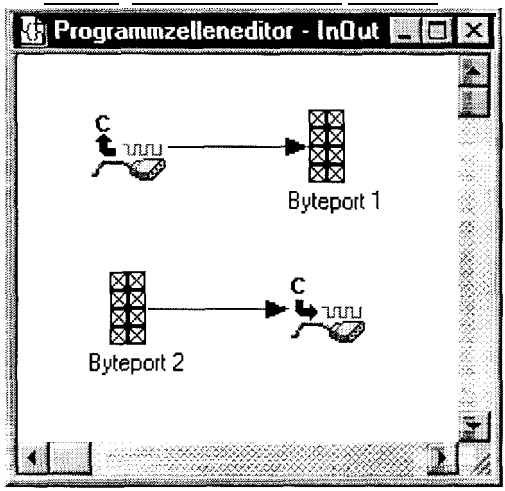


Abb. 12.20 Ein einfaches serielles Interface

Die Programmzelle INOUT wird in einer Schleife immer wieder angesprungen. Man kann daher mit einfachen PC-Programmen schnell veränderliche Ausgaben steuern. Eine mögliche Anwendung ist die Ansteuerung von Schrittmotoren, mit denen sich z.B. kleine Roboter-Modelle bewegen lassen.

Programme zur seriellen Schnittstelle lassen sich vielfach auch sinnvoll mit den Möglichkeiten der Datenspeicherung im EEPROM der Control-Unit kombinieren. Die bereits in Kap. 9.4 vorgestellte Offline-Datenerfassung verwendet ein serielles Übertragungsprogramm fLir die gemessenen Daten. Eine laufende Messung kann zu beliebigen Zeiten durch den PC unterbrochen werden, um alle bisher erfaßten Daten auszulesen. Hier soll nun das Programm nach Listing 9.9 mit den Mitteln von C-Control-Plus umgesetzt werden, wobei die Bezeichnungen der Unterprogramme zur besseren Vergleichbarkeit aus dem BASIC-Listing übernommen wurden. Das Programm nach Abb. 12.21 bis 12.25 demonstriert zugleich den Gebrauch der Funktionsblöcke zur Datenspeicherung im EEPROM. In einer Programmzelle ABLAUF werden laufend die Unterprogramme MESSEN und WARTEN aufgerufen.

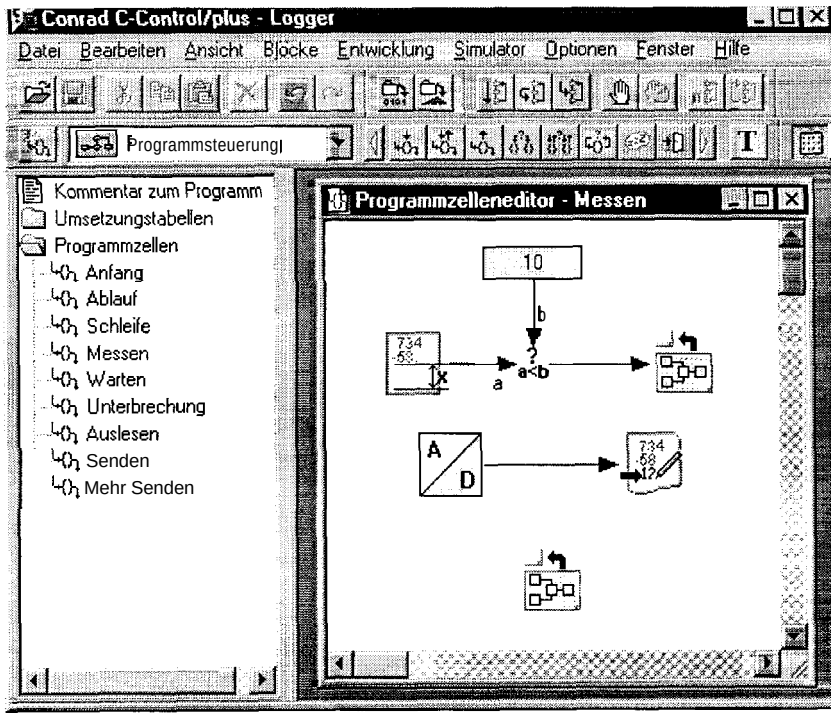


Abb. 12.21 Messen und Speichern von Daten

Die Programmzelle MESSEN nach Abb. 12.21 erfaßt und speichert einzelne Meßwerte. Vor jeder Messung wird der noch verfügbare Speicherplatz

überprüft. Wenn weniger als zehn freie Bytes bleiben, wird die Programmzelle durch einen vorzeitigen Rücksprung beendet. Abb. 12.22 zeigt die Prozedur WARTEN, in der zugleich Aktivitäten der Schnittstelle beobachtet werden, um eine eventuelle Unterbrechung durch das Terminal zu erkennen.

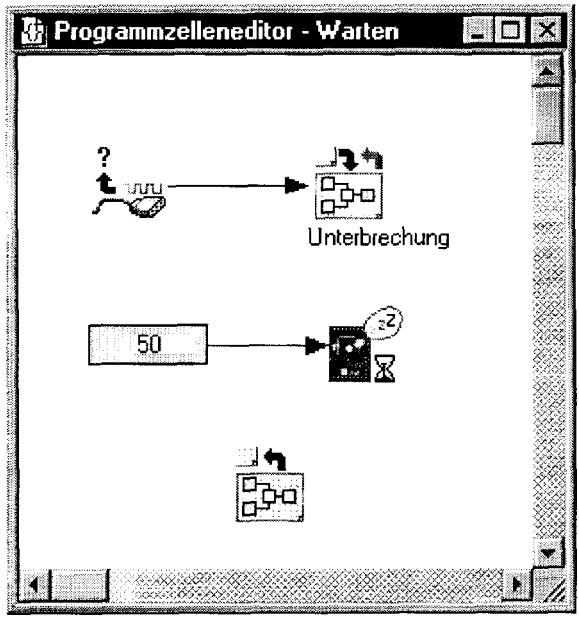


Abb. 12.22 Auswertung eventieller Unterbrechungszeichen

Die eigentliche Prozedur UNTERBRECHUNG (vgl. Abb. 12.23) überprüft jedes ankommende Zeichen zunächst daraufhin, ob es das vereinbarte Unterbrechungszeichen ESC (AKI1 27) ist. Bei einer Übereinstimmung wird die Zelle AUSLESEN aufgerufen.

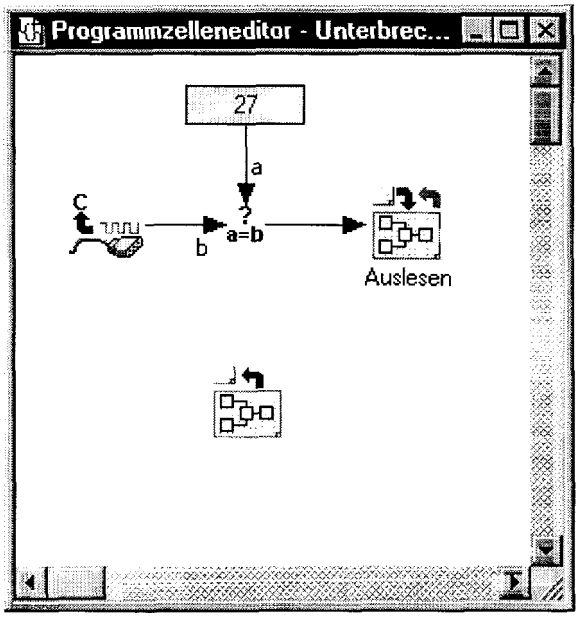


Abb. 12.23 Auswertung des Unterbrechungszeichens

In der Zelle AUSLESEN (vgl. Abb. 12.24) muß zunächst die EEPROM-Datei für Schreibzugriffe geschlossen werden (vgl. CLOSE#). Dann erst kann sie zum Auslesen geöffnet werden (vgl. OPEN# FOR READ). Für den eigentlichen Auslesevorgang wird die Programmzelle SENDEN als Unterprogramm aufgerufen. Sobald alle erfaßten und gespeicherten Zeichen über die serielle Schnittstelle an den PC gesandt wurden, wird die EEPROM-Datei erneut f?ir Schreibzugriffe geöfbet (vgl. OPEN# FOR APPEND). Mit dem Rücksprung aus der Zelle AUSLESEN wird dann in die Datenerfassung zurückverzweigt.

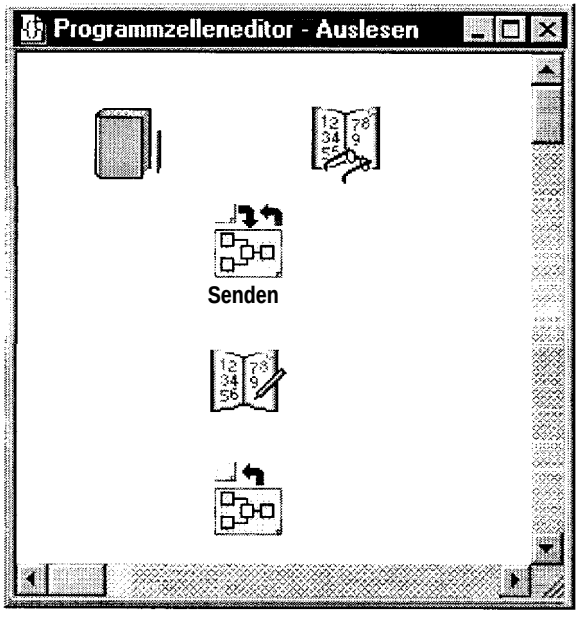


Abb. 12.24 Dateioperationen zum Auslesen

In der Programmzelle SENDEN (vgl. Abb. 12.25) wird jeweils ein Zeichen aus dem EEPROM gelesen und als Byte über die serielle Schnittstelle abgesandt. Danach erfolgt ein Test auf das Dateiende. Solange noch Daten vorliegen, verzweigt das Programm in die folgende Zelle MEHRSENDEN, die nur einen Sprungbefehl zu SENDEN enthält. Erst nach dem letzten Byte wird ein Rücksprung eingeleitet, womit die weitere Datenerfassung fortgesetzt wird.

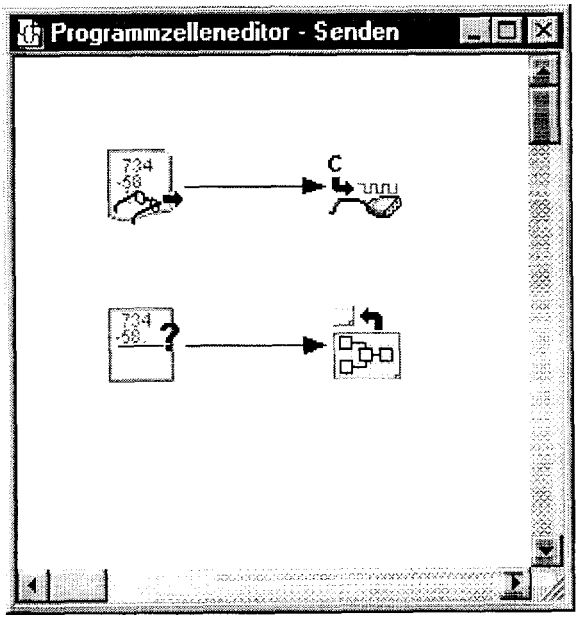
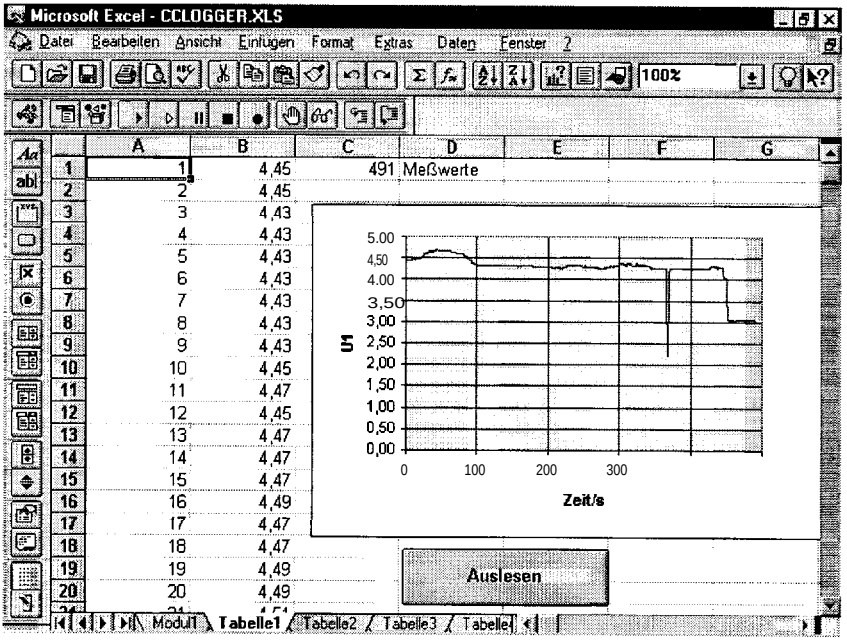


Abb. 12.25 Auslesen und Überprüfung auf das Dateiende

Das Programm läßt sich vorteilhaft für Langzeit-Datenerfassgen einsetzen. Schon vor dem Ende der Meßzeit kann jederzeit der bisher erfaßte Datenbestand ausgelesen und ausgewertet werden. Nach kurzer Unterbrechung wird dann jeweils die Messung fortgesetzt, wobei weitere Daten angehängt werden. Jedes erneute Auslesen liefert wieder alle Meßdaten seit Beginn der Messung. Das Ende der Messung ist erst erreicht, wenn der Speicherplatz im EEPROM mit insgesamt 8 Kilobyte erschöpft ist.

Die Meßdaten lassen sich z.B. direkt in Excel auslesen und auswerten. Dazu wird ein Excel-Makro geschrieben, das direkt auf die serielle Schnittstelle des PC zugreift (vgl. [SI). Man spart auf diese Weise den Umweg über ein spezielles Ausleseprogramm. Die Meßdaten erscheinen sofort in einer Excel-Tabelle und je nach Wunsch auch in einem passend formatierten Diagramm. Abb. 12.26 zeigt ein Meßergebnis. Das Verfahren kann nicht nur zur reinen Datenerfassung, sondern auch für Steuerungs- und Regelungsaufgaben eingesetzt werden, wobei ein geeignetes Programm in der Control-Unit jeweils die Kommunikation über die Schnittstelle festlegt.


Microsoft Excel - CCLOGGER.XLS
Datei Bearbeiten Ansicht Einfügen Format Extras Daten Fenster ?
100%
A B C D E F G
1 1 4,45 491 Meßwerte
2 2 4,45
3 3 4,43
4 4 4,43
5 5 4,43
6 6 4,43
7 7 4,43
8 8 4,43
9 9 4,43
10 10 4,45
11 11 4,47
12 12 4,45
13 13 4,47
14 14 4,47
15 15 4,47
16 16 4,49
17 17 4,47
18 18 4,47
19 19 4,49
20 20 4,49
5.00
4,50
4.00
3,50
3,00
2,50
2,00
1,50
1,00
0,50
0,00
U/V
0
100
200
300
Zeit/s
Auslesen
Modul1 Tabelle1 Tabelle2 Tabelle3 Tabelle4