Die visuelle Sprache C-Control plus

Marian Grahl
Matr.Nr. 194226
murki@cs.tu-berlin.de

Jörg Schneider
Matr.Nr. 194026
komm@cs.tu-berlin.de

22. Januar 2003

Inhaltsverzeichnis

Abbildung 1: Der C-Control I Mikrocontroller

1. Einleitung

Wir wollen die visuelle Programmiersprache C-Control Plus vorstellen. Diese Sprache wurde von der Elektronikversandfirma Conrad[3] speziell für die Mikrocontroller „C-Control"[2] entwickelt. Bei der betrachteten Sprachumgebung handelt es sich um die Version 1.07 aus dem Jahr 1997. Leider wurde die Umgebung seit dem nicht mehr weiterentwickelt und es sind lediglich für die „ Schwestersprachen" C-Control BASIC und C-Control Fuzzy neue Versionen erschienen.

Wir werden nachdem wir kurz auf den Einsatzzweck und die Zielsetzungen der Sprache eingegangen sind, die Konzepte der Sprache vorstellen. Darauf stellen wir die Entwicklungs- und Simulationsumgebung vor und darin als Beispielprogramm die Implementierung eines Taschenrechners. Abschließend wollen wir eine Einordnung im Bereich der visuellen Sprachen vornehmen, um dann zu prüfen, inwieweit die gesetzten Ziele erreicht wurden.

2. Zielsetzung der Sprache

Um die Zielplattform der Sprache besser zu verstehen, wollen wir an dieser Stelle einen kurzen Überblick über Mikrocontroller geben. Mikrocontroller sind prinzipiell vergleichbar mit herkömmlichen Personal Computern. Sie verfügen ebenso wie diese über flüchtigen und nicht flüchtigen Speicher und sind programmierbar. Allerdings haben sie häufig eine kompakte Bauform, die hier verwendete C-Control-Einheit ist auf einer 76×52 mm Platine untergebracht (siehe Abbildung 1), und sind eher auf Stromverbrauch und Robustheit als auf Leistung optimiert.

Eingesetzt werden Mikrocontroller vornehmlich für Steuer- und Regelanlagen. Zum Beispiel finden sie sich in Heizungsanlagen und Haushaltsgeräten. Dort ersetzen sie die immer komplexer werdenden Logikschaltungen, wobei sie meist einfacher zu handhaben und preiswerter sind. Vor allem ist es mit einem Mikrocontroller möglich die Logik nachträglich anzupassen. Allerdings müssen die Mikrocontroller programmiert und nicht als Logik definiert werden, hier soll die visuelle Sprache eine Brücke bieten.

Die Firma Conrad Electronic sieht die Zielgruppe ihrer Sprache vornehmlich in kleinen Firmen und Hobbyanwendern, die kleinere Steuer- und Regelautomatik als Einzelstücke oder in Kleinserie entwerfen. Vor allem für Anwender, die sich mit der Steuerungstechnik auskennen, aber kaum oder keine Programmierkenntnisse haben, ist diese Sprache geeignet. Sie sieht dafür den geringen Lernaufwand der graphischen Sprache als Vorteil.

Wie in der Einleitung schon erwähnt existieren noch zwei andere Entwicklungsumgebungen für diesen Mikrocontroller. Diese treten mit nahezu denselben Zielen an und wir werden daher am Schluß noch einmal die visuelle Umgebung mit ihnen vergleichen.

3. Das Konzept von CCplus

Das C-Control Plus zugrunde liegende Konzept ist eine Kombination aus einem Kontrollfluss und mehreren Datenflüssen, die diesen verfeinern.

Der Kontrollfluss spiegelt sich im imperativen Teil der Programmstruktur wieder. Dabei werden Programmzellen in einer Sequenz angeordnet. Diese Zellen werden dann auch in dieser Reihenfolge bei der ersten beginnend abgearbeitet. Zusätzlich können in den Zellen Sprungblöcke zur Steuerung des Programmflusses angegeben werden, so das auch von der Abarbeitungssequenz abgewichen werden kann. Damit sind die Programmzellen vergleichbar mit den Zeilennummern älterer, nicht objektorientierter BASIC-Interpreter wie z.B. GW-BASIC.

Die Programmzellen selbst bestehen dann aus einem Datenfluss. Jede Zelle kann auch mehrere nicht zusammenhängende Kontrollflüsse enthalten.

Abbildung 2: getypte und geordnete Verbindungen im Datenfluss

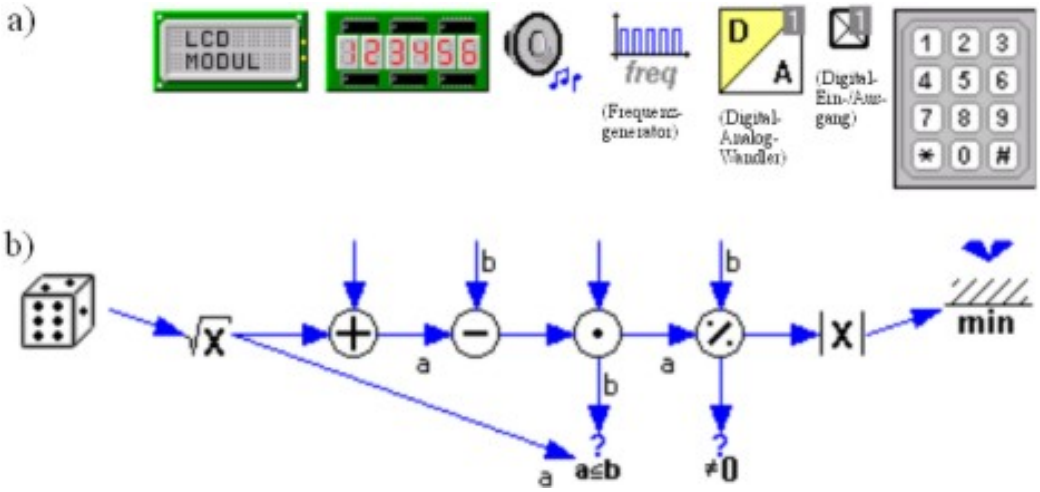

Abbildung 3: a) die Blöcke für die vordefinierten Ein- und Ausgabegeräte b) die Blöcke für die Berechnungen

Es gibt eine ganze Reihe von unterschiedlich gruppierten Blocktypen, die im Programm die Knoten des Kontrollflussgraphen darstellen. Als Datenquellen existieren Konstanten, die vordefinierten Eingabegeräte und Speicherzellen. Die Daten können dann in einer Speicherzelle abgelegt werden, damit darauf in einer anderen Zelle wieder zuggegriffen werden kann , auf einem der vordefinierten Ausgabegeräte ausgegeben werden oder verarbeitet werden kännen um dann wieder zur Verfügung zu stehen. Einige der Blöcke haben geordnete Eingänge, sie verfügen dann über eine begrenzte Anzahl an Eingängen, die mit einem Kleinbuchstaben markiert werden (siehe Abbildung 3). An jedem Eingang darf dann nur ein Wert anliegen. Andere Blöcke haben nur ein Art von Eingängen, die dann in der Anzahl nicht beschränkt sind. Ein Block hat gegebenenfalls genau einen Wert, der am Ausgang beliebig oft verwendet werden kann. Außerdem haben die einzelnen Blöcke je nach Intention getypte Ein- und Ausgänge. Wobei es dann auch generische Blöcke gibt (z.B. für den Vergleich), bei denen an allen Eingängen derselbe Typ anliegen muss. Der Datenfluss darf keine Zyklen enthalten. Allerdings ist es möglich dieselbe Speicherzelle mehrfach zu verwenden und in einem Datenflussdiagramm daraus zu lesen und dann darin zu Schreiben.

Wir wollen jetzt entlang der in der Entwicklungsumgebung vorgenommenen Gruppierung die wichtigsten Blocktypen kurz vorstellen:

- Die Gruppe Ein-/Ausgabe beinhaltet Blocktypen für die direkte Manipulation der Digitalein- und Ausgänge des Mikrocontrollers, A/D und D/A-Wandler und einen Frequenzgenerator. Ausserdem enthält sie noch eine Reihe von externen Zusatzkomponenten für die C-Control Produktreihe, wie eine Zehnertastatur, eine sechsstellige 7-Segment- Anzeige und ein 16×2 LC-Display, was sich insbesondere für die Simulation von Vorteil erweist.

- Die zweite Gruppe, Berechnungen, beinhaltet die vier Grundrechenarten, Minimum- und Maximumfunktion, einen Zufallsgenerator und die Bitschiebeoperationen. Subtraktion und Division haben dabei genau zwei geordnete Eingänge, während bei Addition und Multiplikation Anzahl und Ordnung nicht fest vorgegeben sind. Alle diese Blöcke arbeiten immer mit 16-Bit Ganzzahlen.

- Unter Vergleich sind alle möglichen Vergleichsoperationen zu finden, die mit ganzzahligen Werten sinnvoll sind. Als Typen sind aber hier nicht nur die 16-Bit Integer zugelassen, es lassen sich auch Datums und Zeitwerte jeweils paarweise vergleichen. Das Ergebnis ist wiederum eine Zahl, die sich als Wahrheitswert interpretieren lässt.

- Die Gruppe Logik fasst alle gängigen Logikoperationen wie NOT, AND, NAND usw. zusammen. Die Operationen arbeiten hierbei wieder 16Bit-getypt und werten die entsprechende logische Operation aus. Bis auf NOT, welches sinnvoller Weise nur einen Eingang hat, haben alle Logikblöcke eine unbegrenzte Anzahl von Eingängen. Das Erscheinungsbild der Logikblöcke entspricht dabei den Symbolen in der klassischen Schaltungslehre, so dass es insbesondere für Leute aus diesem Metier auch ohne besondere Vorkenntnisse in C- Control intuitiv verständlich ist.

- Die Gruppe Zeit vereint Blocktypen zum Auslesen von systeminterner Uhrzeit und Datum und deren jeweiligen Einzelkomponenten und einen Zeitgeber, dessen Ausgangswert sich alle 20 Millisekunden um eins erhöht.

- Variablen vereinigt die Blocktypen die notwendig sind, um auf die Speicherzellen des Benutzerspeichers zuzugreifen, die anders als bei gängigen Programmiersprachen aber nicht benannt werden können, sondern immer über ihre Speicheradresse referenziert bzw. identifiziert werden. Leider gibt es dabei keine Felder aus mehreren Werten oder eine Möglichkeit zur indirekten Adressierung.

- Die Gruppe Konstanten beinhaltet alle Konstantenblocktypen für alle in der Sprache vorkommenden Typen.

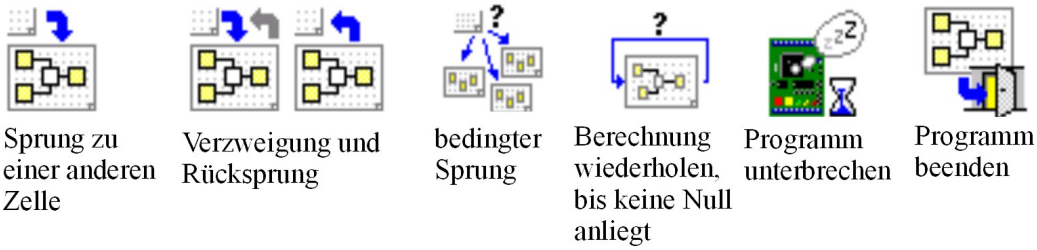

Abbildung 4: Die Blöcke für die Sprünge im Kontrollfluss

- Mit den Datenaufzeichnungsblöcken lässt sich eine Datenspeicherung im nicht flüchtigen EEPROM des Mikrocontrollers vornhemen.
- Die Gruppe Serielle Schnitstelle umfasst die notwendigen Kommuikationsblöcke.

Auf die letzte Gruppe mit den Blöcken zur Programmsteuerung wollen wir jetzt noch einmal genauer eingehen. Die wichtigsten Elemente sind in Abbildung 4 zu sehen. Die Blöcke zur Steuerung und Manipulation des Kontrollflusses werden wie reine Datenflussblöcke in den Programmzellen notiert und können dabei allein stehen oder auch mit einem Wert an ihrem Eingang versehen werden. Dabei gibt es unter anderen Äquivalente zu den BASIC Befehlen GOTO, GOSUB/RETURN und ON x GOTO. Wobei bei den ersten Dreien der Sprung nur erfolgt, wenn kein Element am Eingang angebunden ist oder dort ein Wert ungleich Null anliegt. Die Ziele der Sprünge werden in den Attributen der Blöcke vermerkt. Ausserdem existiert noch eine Pause, die den Controller für die angegebene Zeit einfriert. Sehr interessant ist auch der Schleifenblock, der die aktuelle Programmzelle so lang wiederholt, bis an seinem Eingang ein Wert ungleich null anliegt. Leider werden diese Sprünge im Kontrollfluss nicht explizit vermerkt.

4. Die Entwicklungsumgebung C-Control plus

Da wie schon erwähnt die C-Control Plus Entwicklungsumgebung die einzige ist, die verfügbar ist, wollen wir nur kurz auf diese eingehen. Die Bedienoberfläche gliedert sich in drei Teile. Im oberen Teil gibt es Menüs und Symbolleisten, mit denen zum Beispiel neue Blöcke eingefügt werden können, das Programm compiliert und simuliert werden kann, Haltepunkte eingefügt werden können und ähnliches. Auf der linken Seite ist der Kontrollfluss als Liste der Programmzellen des Programms dargestellt und im zentralen Teil kann die aktuell gewählte Programmzelle bearbeitet werden.

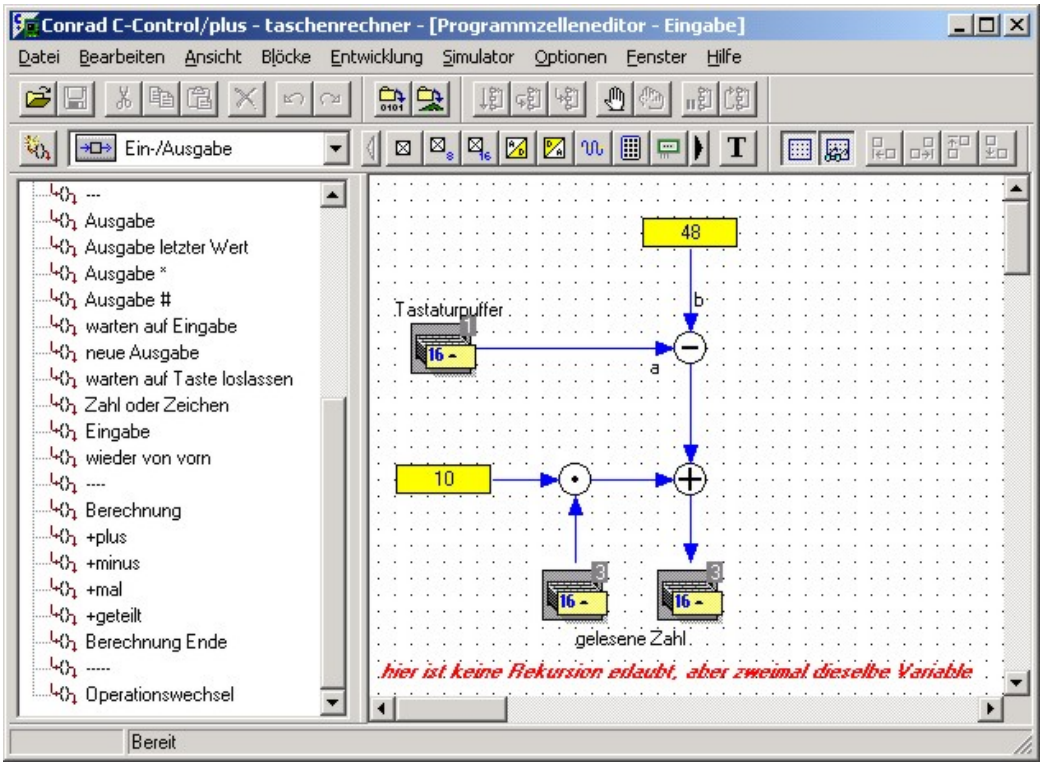

Abbildung 5: Die Entwicklungsumgebung C-Control plus

Die Benutzung der Entwicklungsumgebung ist etwas umständlich, da vor allem die Attribute der meisten Blöcke nicht direkt angezeigt werden und erst umständlich über das Kontextmenü abgerufen werden müssen. Die Entwicklungsumgebung ist zwar syntaxgesteuert und prüft auch schon die Verbindungstypen beim Erstellen, allerdings prüft sie zum Beispiel bei den Sprungblöcken die Sprungziele nicht bei der Eingabe, so dass diese Fehler erst beim Kompilieren auffallen.

Die Entwicklungsumgebung enthält außerdem noch einen Simulator für die entwickelten Programme. Dieser kann alle Ein- und Ausgänge sowie die Erweiterungsmodule darstellen. Daher kann man sich direkt die Zifferntastatur oder das LCD Display anzeigen lassen. Diese Simulation ist sehr gut in die Umgebung integriert. So werden im Einzelschrittmodus immer die aktuell bearbeitete Zelle dargestellt und es können die Werte der Speicherzellen beobachtet werden, leider geht das nicht für Zwischenergebnisse. Ein weiterer Nachteil ist, dass die Simulation nicht die Berechnungsgeschwindigkeit des Mikrocontrollers berücksichtigt, sondern die Berechnungen so schnell wie möglich ausführt, auf modernen PCs also um ein vielfaches schneller als auf dem Mikrocontroller.

5. Die Bespielanwendung

Bevor wir zu der Einordnung und Bewertung der Sprache kommen, wollen wir ein mit der Sprache programmiertes Beispiel genauer vorstellen. Als Beispielanwendung[4] haben wir einen simplen Taschenrechner implementiert, dessen Funktionsumfang und damit praktische Anwendbarkeit aber aufgrund der hardwaremäßigen Beschränkungen der C-Control Mikrocontroller sehr gering ist. Zum einen ist das Problem, dass die Tastatur für Simulationszwecke am Computer die einzige brauchbare Eingabequelle ist und sie außer den zehn Ziffern nur die Stern- (*) und Rautetaste (#) enthält, wodurch das Wählen der Rechenoperation durch die Sterntaste und das Ausführen derselben durch die Rautetaste gemacht wird. Doch während sich dieses Problem bei einem realen Aufbau durch das Zurverfügungstehen von digtalen Eingabeports noch beheben ließe, lässt sich die Tatsache, dass es sich bei C-Control um 16-Bit Mikrocontroller handelt, die maximal 16-Bit Ganzzahloperationen durchführen können, nicht auf praktische Weise lösen, wodurch der Rechen- und Eingabebereich auf den Wertebereich von -32768 bis 32767 beschränkt ist.

Im Folgenden werden wir nun auf die einzelnen Programmzellen unseres Taschenrechners näher eingehen und deren Funktion erläutern. Das Programm ist zunächst einmal in vier Teile gegliedert, dem Vorspann, der Eingabeverarbeitung (grob betrachtet der Hauptroutine), der Berechnung und dem Operationswechsel.

5.1. Der Vorspann

Der erste Teil, der Vorspann ist für die Funktionsweise des Taschenrechners unerheblich und soll deshalb an dieser Stelle keiner genaueren Betrachtung unterzogen werden, es sei nur soviel gesagt, dass dieser Bereich einen regelmäßig wechselnden Schriftzug auf das LC-Display ausgibt.

5.2. Das Hauptprogramm

In der ersten Zelle ,,Initialisierung" des zweiten Teils, werden alle Speicherzellen, die wir im folgenden verwenden werden mit null initialisiert. Das sind der Tastaturpuffer an Speicherstelle 1, die letzte gelesene Zahl an Speicherstelle 3, dass letzte Ergebnis an 7, sowie der Timer an 8 und die Ausgabevariante an 9, die wir zum Wechseln der oberen Anzeigezeile verwenden. In der Zelle ,,Ausgabe" springen wir in Abhängingkeit vom Wert der Variable Ausgabevariante zu den Zellen ,,Ausgabe letzter Wert", ,,Ausgabe #" oder ,, Ausgabe *". In diesen Zellen wird die aktuell gewählte Operation und die eingegebene Zahl in der zweiten Displayzeile angezeigt und entweder der letzte berechnete Wert, ,,* = Op. wechseln" oder ,,# = Berechnen" in der oberen Zeile angezeigt. Nach erfolgter Ausgabe wird in jedem der drei Fälle zur Zelle ,,warten auf Eingabe" gesprungen die ihrerseits die Zehnertastatur ausliest und den Wert an Speicherstelle 0 zwischenspeichert und ausserdem den Zähler in Speicherstelle 8 um eins erhöht. Diese Zelle wird solange wiederholt, bis entweder der Zähler den Wert 32767 erreicht oder eine Taste gedrückt wird. Sollte eine Taste gedrückt worden sein, wird zur Zelle ,,warten auf Taste loslassen" gesprungen, andernfalls geht es sequentiell weiter im Programmablauf mit der Zelle ,,neue Ausgabe", die die Variable Ausgabevariante an Speicherzelle 9 um eins erhöht und modulo fünf nimmt (damit sie immer im Bereich von einschließlich 0 bis einschließlich 4 liegt) und springt dann zurück zur Zelle ,,Ausgabe". Die Zelle ,,warten auf Taste loslassen", zu der nur dann gesprungen wurde, wenn eine Taste gedrückt wurde, wird so lange wiederholt bis keine Taste mehr gedrückt wird um sicherzustellen, dass es keine automatische Tastenwiederholung gibt, falls man zu lange auf eine Taste drücken sollte. Außerdem setzten wir hier noch die Variable Ausgabevariante wieder auf 0. In die Zelle ,,Zahl oder Zeichen" springen wir, wenn ein Stern eingegeben wurde zum Operationswechselblock, bei einer Raute zum Berechnungsblock. Sollte keines der beiden Zeichen gedrückt worden sein, so wissen wir, dass es sich bei der Eingabe nur um eine Zahl handeln kann, wir können also 48 vom Wert des Eingabepuffers abziehen, um vom ASCII- Code zum Wert zu kommen, und diesen Wert dann auf die verzehnfachte gelesene Zahl addieren. Nachfolgend können wir mit einem Sprung zur Zelle ,,Ausgabe" wieder von vorne beginnen.

5.3. Die Berechnung

Der dritte Teil führt die eigentliche Berechnung aus, das heißt, er führt die gewählte Operation mit dem letzten Ergebnis und der gelesenen Zahl als Operanden durch. Bei Addition, Subtraktion und Multiplikation erfolgt das wenig spektakulär jeweils in einer eigenen Zelle, wobei das Ergebnis berechnet und in der Variable letztes Ergebnis gespeichert wird und die gelesene Zahl wieder auf null gesetzt wird. Abschließend erfolgt ein Sprung zurück zur Ausgabe. Bei der Division erfolgt als erstes ein Test der gelesenen Zahl, sprich des Divisors, auf null, um eine Division durch null präventiv zu verhindern und stattdessen eine Fehlermeldung auf dem Display auszugeben.

5.4. Der Operationswechsel

Der vierte und letzte Teil führt einen Operationswechsel als Reaktion auf das Drücken der Sterntaste aus, dabei wird die Variable Operation an Speicherzelle 7 um eins erhöht und anschließend modulo 4 genommen, um so sicherzustellen, dass für die Variable immer $0 \leq x \leq 3$ gilt. Danach wird wieder zurück zum Hauptprogramm, sprich zur Zelle „ Ausgabe”, gesprungen.

6. C-Control als Visuelle Programmiersprache

In diesem Teil wollen wir uns der Einordnung von C-Control im Umfeld der visuellen Programmiersprache nach unterschiedlichen Merkmalen[1] widmen. Da die Software der Firma Conrad Electronic wie schon erwähnt die einzige Entwicklungsumgebung für C-Control Plus ist und keine explizite Definition der Sprache existiert, erscheint eine getrennte, unabhängige Bewertung kaum sinnvoll.

Ein wichtiger Punkt, nicht nur visueller Programmiersprachen ist das verwendete Programmierparadigma. C-Control vereint dabei zwei der bekanntesten, nämlich einen Kontrollfluss, der sich im Ablauf der Programmzellen widerspiegelt und einen Datenfluss innerhalb der einzelnen Programmzellen. Da es sich bei beiden Paradigmen um signifikante Merkmale der Sprache handelt und mit den Sprungblöcken auch eine starke Verknüpfung zwischen den verschiedenen Schichten besteht, muss man sie als Multi- Paradigmen-Sprache einordnen.

Grafische Objekte werden im Datenflussteil von C-Control über Anschlusspunkte und gerichtete Verbindungslinien miteinander verbunden und sie sind im Verhalten unabhängig von ihrer Position oder dem grafischen Kontext. Wir haben es also mit einer diagrammatischen Sprache zu tun.

Der Kontrollfluss wiederum ist nur durch eine Sequenz dargestellt und damit sehr ähnlich zu der Abarbeitung im Mikrocontroller. Auch in anderen Aspekten

ist die Nähe zu der Hardware zu erkennen. Während das direkte Kombinieren von Logikbausteinen für die Zielgruppe noch von Vorteil ist, so ist das niedrige Abstraktionslevel ansonsten eher hinderlich. So gibt es auch keine Konstrukte zur Ereignisbehandlung, wobei wir denken, das gerade dieses wichtig für die Regelungstechnik und zur Ausnahmebehandlung ist.

Trotz dieser fehlenden Konstrukte würden wir von einer ,,Universalsprache für die Regelungstechnik" sprechen. Vor allem da die Sprache in Anlehnung an den C-Control Basicdialekt entworfen worden ist, gehen wir davon aus, das sie auch entsprechend mächtig ist. Allerdings ist sie mit ihren Sprachelementen zu stark auf den einen Mikrocontroller und dessen Erweiteterungen zugeschnitten, um auch für andere Einsatzgebiete als die Regelungstechnik nützlich zu sein.

7. Fazit

Mit C-Control liefert die Firma Conrad eine Sprache mit Entwicklungsumgebung, die in ihrer aus rein pragmatischen Erwägungen gewählten Form als Visuelle Programmiersprache ein Produkt ist, das insbesondere für die avisierte Zielgruppe der programmierunerfahrenen Entwickler von Steuer- und Regeltechnik mit ihrer Nähe zu aus diesem Bereich bekannten Diagrammtypen sinnvolle praktische Einsatzmöglichkeiten für die Klasse der Visuellen Programmiersprachen zeigt. Allerdings fällt bei genauerem Hinsehen auf, dass die Sprache auf Basis der entsprechenden BASIC Version entstanden ist und einige der typischen Merkmale textueller Sprachen, wie z.B. den linearen Programmablauf, implantiert bekommen hat, dabei in ihrer Ausdrucksmächtigkeit aber hinter dieser zurück bleibt.

Weiterhin stellt die Entwicklungsumgebung, durch mangelnde technische Abstraktion ein großes Hindernis bei der Erlernung der Sprache, insbesondere für Neulinge dar. Sehr viele Probleme werden diese mit der Speicherverwaltung und der teilweise nicht intuitiven Bedienung der Entwicklungsumgebung haben. Für Experten sind dann wiederum die fehlende indirekte Adressierung und die fehlende Ereignissbehandlung hinderlich. Das die Sprache durch die Zielhardware eingeschränkt ist und ihre Elemente sich stark an den möglichen Erweiterungen orientieren, hilft wiederum beim Erstellen von Programmen für diesen Mikrocontroller.

Wenn wir die visuelle Entwicklungsumgebung jetzt der basicbasierten Entwicklungsumgebung gegenüberstellen, dann sehen wir vor allem wie bereits erwähnt den größeren Sprachumfang des Basicdialektes. Da die imperativen Konzepte bei der C-Control plus Sprache analog zu Basic definiert wurden, ist der Lernaufwand für beide sicher gleich.

Insgesamt muss man also zu dem Schluss kommen, dass die von uns getestete Version den Ansprüchen ihrer Entwickler nur sehr unzureichend gerecht wird. Au-

ßerdem fällt auf, das die visuelle Sprache in diesem Fall keine Vorteile gegenüber der textuellen Variante bringt.

Literatur

[1] Margaret M. Burnett and Marla J. Baker. *VPL: Visual Programming Languages*. WWW, http://www.cs.orst.edu/∼burnett/vpl.html.

[2] C-Control. *Projektseite mit den verschiedenen Entwicklungsumgebung*. WWW, http://www.c-control.de/.

[3] Conrad Electronic. *Webseite des Entwicklers und Vertreibers*. WWW, http://www.conrad.de/.

[4] Marian Grahl and Jörg Schneider. *Beispielanwendung Taschenrechner*. WWW, http://tfs.cs.tu-berlin.de/∼vila/VLfolien.html.