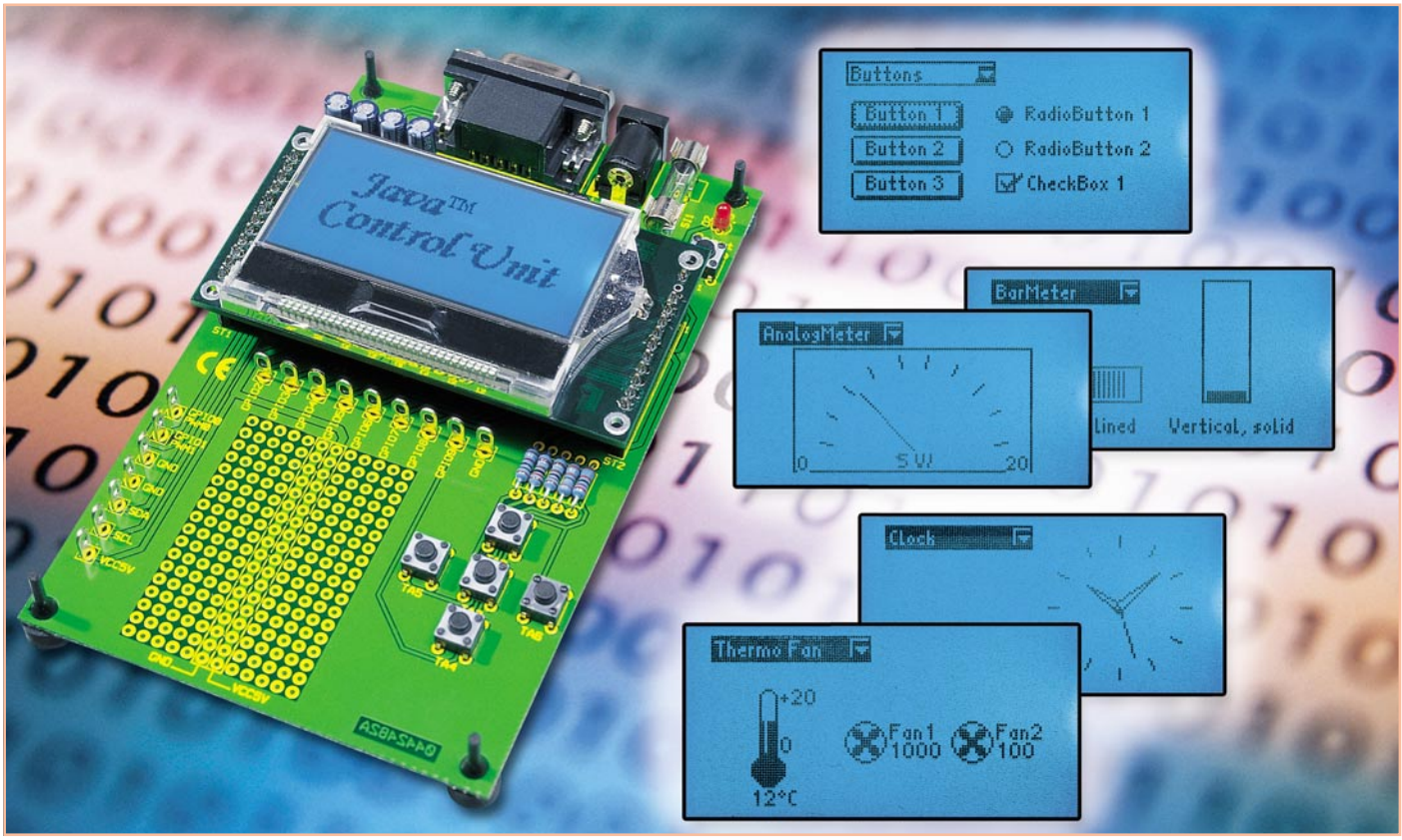
Steuern und visualisieren –



Java™-Control-Unit JCU 10 mit JControl-Technologie Teil 1

In der hier beginnenden Artikelserie stellen wir die Java™-Control-Unit vor. Hierbei handelt es sich um eine kompakte Platine mit einem Grafikdisplay und verschiedenen Schnittstellen, die, entsprechend programmiert, äußerst universell als autark arbeitende Steuer- und Anzeigeeinheit in eigenen Applikationen, etwa zur Haussteuerung, einsetzbar ist. Im ersten Teil vermitteln wir zunächst Grundlegendes zur hier verwendeten JControl-Hardware und zum Einsatz der Programmiersprache Java™ in Embedded Systems. In den folgenden Artikeln stellen wir die Hardware, die Java™-Control-Unit JCU 10 sowie das zur Programmentwicklung einsetzbare „Evaluation Board" vor, gehen auf das Erstellen von eigenen Applikationen ein und erläutern weitere Schaltungen, in denen die JCU 10 als Steuer- und Anzeigeeinheit eingesetzt und nach eigenen Vorstellungen programmiert werden kann.

Messen, Steuern und Regeln mit Java™

In eigene Applikationen eingebundene Steuer-, Bedien- und Anzeigesysteme, so genannte „Embedded Systems", gewinnen immer mehr an Bedeutung, etwa in der Gebäude- oder Messtechnik. Sie sind oft das Herz von Mess-, Stromversorgungs- und Ladegeräten, steuern Klimaanlagen, den Brötchen-Backofen in der Bäckerei oder gar (Teil-) Produktionsprozesse.

Das optische Interface zum Bediener stellen meist grafische Displays dar, die je nach Anwendung Grafiken, Bilder und verschiedene Schriften darstellen können. Für den Hobby-Elektroniker verschließt sich diese Technik bisher noch weitgehend – einerseits sind die Preise für ein komplettes System einschließlich Entwicklungsumgebung oft sehr hoch, zu hoch für den, der z. B. nur seine Heizung und Belüftung im Eigenheim steuern und dabei Prozesse familienfreundlich visualisieren will. Und andererseits erfordert die Programmierung fundierte Programmiersprachenkenntnisse, die für nur eine oder wenige Anwendungen zu erwerben, man oft nicht bereit ist.

Dennoch, der Reiz bleibt – die Verwendung von grafischen Displays in Mikrocontrollerschaltungen zum Anzeigen von Messwerten oder zur Anzeige von einzu-

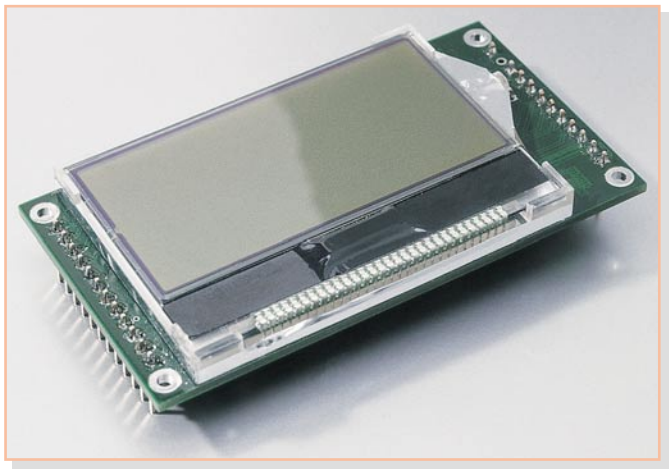


Bild 1: Die kompakte Java™-Control-Unit lässt sich leicht in eigene Applikationen einfügen.

gebenden Parametern oder Ablaufsteuerungen erleichtert die Bedienungen solcher Schaltungen ungemein (im Privatbereich kann man dies wohl „familienkompatibel“ nennen). Will man allerdings auf dem Display mehr darstellen als nur Text, so ist dies in „altbekannten“ Programmiersprachen wie C oder Assembler meist nur mit einem erheblichen Software-Aufwand möglich. In dieser Artikelserie wollen wir zeigen, dass dies einfacher und auch zu erschwinglichen Preisen möglich ist.

Der Mittelpunkt, um den sich alles dreht, ist eine kompakte, universell programmierbare Steuer- und Anzeigeeinheit – die „Java™-Control-Unit“. Der Mikrocontroller des Moduls wird in der Programmiersprache Java™ programmiert, er kann damit in Applikationsschaltungen Mess-, Steuer- und Regelaufgaben übernehmen.

Da der Einsatz von Java™ in Mikrocontrollersystemen noch relativ wenig verbreitet ist, werden wir uns in diesem ersten Teil, neben der grundsätzlichen Vorstellung der Java™-Control-Unit, im Überblick mit diesem Thema beschäftigen.

Puristen sei bereits an dieser Stelle gesagt, dass es sich bei dieser Thematik aufgrund der Herkunft von Hardware und Programmiersprache kaum vermeiden lassen wird, englische Begriffe einzusetzen. Diese werden jedoch, so weit sinnvoll, erläutert.

Die Hardware

Mit der Java™-Control-Unit (Abbildung 1) erhält der Anwender ein kompaktes Modul, dessen Abmessungen (ca. 42 x 80 mm) nicht wesentlich größer sind als das LC-Display selbst. Das Modul kann in Schaltungen also statt eines herkömmlichen Mikrocontrollers eingesetzt werden und bringt die Möglichkeit mit, Programmabläufe über grafische Menüs zu steuern und Mess-Ergebnisse, abgefragte Zustände oder anderes ansprechend und übersichtlich darzustellen.

Dem Hobby-Programmierer kommt bei der kompakten Bauweise und der Zusammenfassung von Controller und Anzeige auch zugute, dass diese Komponenten, deren Kosten in den meisten Anwendungen doch den größten Anteil ausmachen dürften, auf einem flexibel einsetzbaren Modul untergebracht sind. So kann z. B. das gleiche Modul zunächst mit dem „Evaluation Board“ programmiert und getestet oder in selbst gebauten Experimentierschaltungen eingesetzt werden. Wenn dann der endgültige Entwurf einer Schaltung fertig ist, kann man das Modul einfach in die Schaltung übernehmen und hat so keine überflüssige und teuer bezahlte Hardware mehr „in der Schublade liegen“.

Die Java™-Control-Unit basiert mit der JControl-Technologie auf der Programmiersprache Java™, die ein projektorientiertes und Hardware-unabhängiges Programmieren ermöglicht. Der Einsatz von Java™ als Programmiersprache für eingebettete Anwendungen ist noch nicht sehr verbreitet. Hier spielen sicherlich die aus der PC-Welt bekannten Vorurteile bezüglich der Ausführungsgeschwindigkeit und des Speicherplatzbedarfs wie auch vielleicht die Überwindung, sich mit einer neuen Programmiersprache zu beschäftigen, eine Rolle. Deshalb werden Mikrocontroller bislang fast ausschließlich in C oder Assembler programmiert.

Gehen wir jedoch zunächst einmal kurz auf die Hardware der Java™-Control-Unit ein. Deren Herzstück ist ein 8-Bit-Mikrocontroller, auf dem die JControl Virtual Machine JCVM8 implementiert ist. Auf dieser wird der Java™-Bytecode ausgeführt. Dem Controller stehen dabei 64 k Flash-Speicher zur Verfügung.

Als Anzeige kommt ein monochromes Grafik-LC-Display mit blauer Hintergrundbeleuchtung und 128 x 64 Pixel zum Einsatz. Die Beleuchtung und der Kontrast des Displays lassen sich per Software steuern.

Zur Ausgabe von akustischen Signalen steht ein Buzzer-Ausgang (PWM-Signal) zur Verfügung.

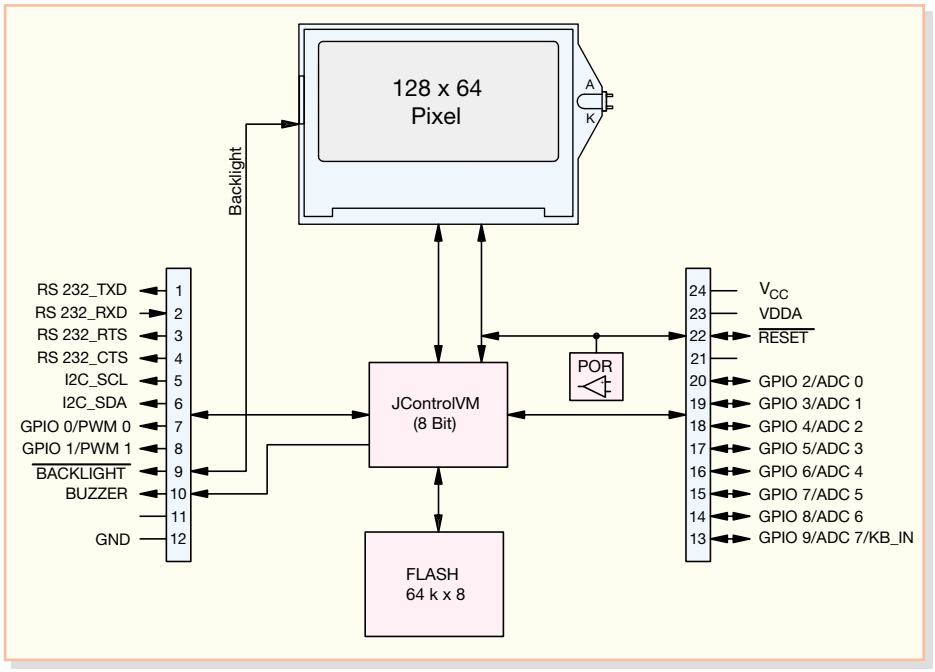


Bild 2: Blockschaltbild der Java™-Control-Unit

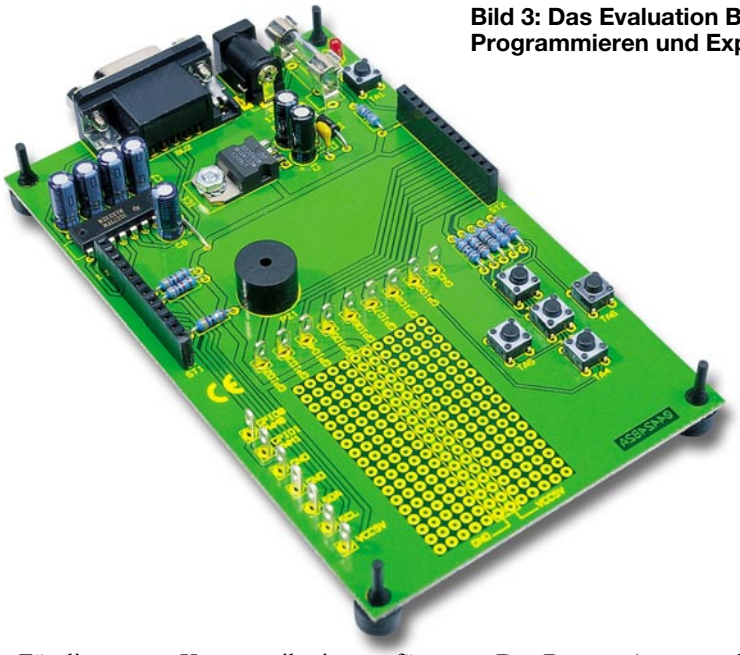


Bild 3: Das Evaluation Board für das Programmieren und Experimentieren

Für die externe Kommunikation verfügt die Java™-Control-Unit über eine I²C-Bus- und eine RS-232-Schnittstelle. Die Programmierung des Moduls erfolgt ebenfalls über die RS-232-Schnittstelle.

Weiterhin besitzt die Java™-Control-Unit zehn universell einsetzbare Ein-/Ausgänge (GPIO, engl. General Purpose Input Output), von denen zwei als PWM-Ausgänge, acht als analoge Eingänge und einer als analoger Tastatur-Decoder für bis zu zehn Tasten genutzt werden können.

Alle Signalleitungen sowie die Anschlüsse für die Spannungsversorgung (5 V) stehen an zwei 12-poligen Stiftleisten am rechten und linken Rand der Platine bereit.

Abbildung 2 zeigt ein Blockschaltbild der Java™-Control-Unit mit der Belegung der Anschlussleisten.

Das Evaluation Board

Im Verlauf der Artikelreihe werden wir ebenfalls ein Evaluation Board (Abbildung 3) vorstellen, das zum Programmieren des Moduls dient und auch zum Experimentieren genutzt werden kann. Die Java™-Control-Unit wird dazu einfach in die zwei Buchsenleisten auf dem Board eingesetzt. Das Board verfügt über eine Anschlussbuchse für ein Netzteil und stellt dem Modul eine geregelte Spannung zur Verfügung.

Zur Programmierung der Java™-Control-Unit befindet sich auf dem Board eine 9-polige Sub-D-Buchse, die mit der seriellen Schnittstelle des PCs verbunden wird.

Um eine Bedienung des Displaymoduls zu ermöglichen, sind auf dem Board fünf Taster vorhanden, die an den Anschluss des Tastatur-Decoders geführt werden. Die Taster haben die Funktionen „Up“, „Down“, „Left“, „Right“ und „Select“, mit denen sich die Navigation durch Bildschirmmenüs am einfachsten realisieren lässt.

Der Buzzer-Ausgang der Java™-Control-Unit wird auf einen Piezo-Signalgeber geführt, mit dem man einzelne Töne, z. B. als Bestätigung für Eingaben, oder aber auch ganze Melodien ausgeben kann.

Die frei konfigurierbaren Ein- und Ausgänge des Moduls sind mit Lötfahnen um ein Lochrasterfeld platziert, auf dem eigene Beschaltungen der Ein- und Ausgänge vorgenommen werden können.

Eine ausführliche Darstellung der Schaltungen mit Schaltplänen und Platinenbildern folgt im nächsten Teil der Artikelserie.

Die Programmiersprache

Da die Anforderungen im Bereich der Steuer- und Regelungstechnik mit Mikrocontrollern immer höher werden, gestalten sich sowohl die Controller als auch die Software immer komplexer.

Hier kommen die Vorteile der Programmiersprache Java™ zum Tragen. So kann nicht nur die Hardware der Java™-Control- Unit über vorgefertigte Klassen (engl. „classes“, bereits kompilierte Programmbausteine) angesprochen werden, sondern es sind durch das Einbinden der entsprechenden Klassen auch grafische Menüs und animierte Anzeigen auf relativ einfache Weise realisierbar.

Die Unabhängigkeit von der verwendeten Hardware wird bei Java™-Anwendungen durch die so genannte Virtual Machine (VM) erreicht. Bekannt ist dies sicherlich aus der PC-Welt, wo man Programme in Java™ schreibt und dann vom Compiler in den Java™-Bytecode, die so genannten Class-Dateien, übersetzen lässt.

Der Bytecode wird von einem Interpreter zur Laufzeit der Anwendung vom PC, auf dem man die Applikation startet, analysiert, und die Java™-Bytecode-Befehle werden, der verwendeten Plattform entsprechend, umgesetzt.

Jeder Java™-Interpreter, egal ob Entwicklungstool oder der allgemein als typischer Java™-Interpreter bekannte Web-Browser, ist somit eine Implementierung der Java™-Virtual-Machine, die es ermöglicht, einmal kompilierte Applikationen sowohl auf Windows-, Solaris- oder Mac-Plattformen auszuführen.

Bei der Java™-Control-Unit wird die JControl JavaVM eingesetzt, die speziell für den Einsatz im Mikrocontroller entwickelt wurde und sogar auf 8-Bit-Controllern implementiert werden kann.

Zur Java™-Plattform gehört neben der VM auch das Java™ Application Interface (API), das eine Zusammenstellung von fertigen Software-Komponenten enthält. Hierzu steht dem Programmierer die API-Spezifikation zur Verfügung, in der alle Informationen zu den einzelnen Klassen sowie deren Verwendung aufgelistet sind.

Vorteil Java™

Die Verwendung von grafischen Elementen macht, wie bereits angedeutet, bei vielen Anwendungen Sinn, da auf diese Weise Programme erstellbar sind, die sich – ähnlich einer PC-Bedienoberfläche – intuitiv bedienen lassen. Außerdem fällt es dem menschlichen Auge im Allgemeinen leichter, bekannte Bilder auszuwerten als Zahlen oder Texte. Und auch Personen, die eine Applikation nicht laufend bedienen, finden sich so besser darin zurecht.

Auch dem Hobby-Programmierer wird somit eine vergleichsweise einfache Möglichkeit geboten, die Fähigkeiten eines grafischen Displays in seinen Programmen auszuschöpfen, die, wie bereits ausgeführt, bei anderen Programmiersprachen wegen des hohen Programmieraufwands sicher nicht in Betracht gezogen würden.

Die Programmierung ist außerdem unabhängig von der verwendeten Hardware, da die Anwendungssoftware nur über die JCVM auf die Hardware zugreifen kann. Dies wiederum verringert das Fehlerrisiko, denn die in Java™ geschriebenen Programme können durch das Einbinden der entsprechenden Klassen für die verwendete Hardware auf andere Systeme übertragen werden, ohne dass der Programmierer der Anwendungssoftware die Hardware des Systems und deren Ansteuerung kennen muss.

Ein weiterer Vorteil der objektorientierten Programmierung in Java™ liegt in der Wiederverwendbarkeit von Programmteilen. So kann man vorgefertigte Klassen und auch selbst erstellte und bereits getestete Klassen einfach wieder in neue Projekte einbeziehen und verringert so den Programmieraufwand und das Auftreten von Programmierfehlern.

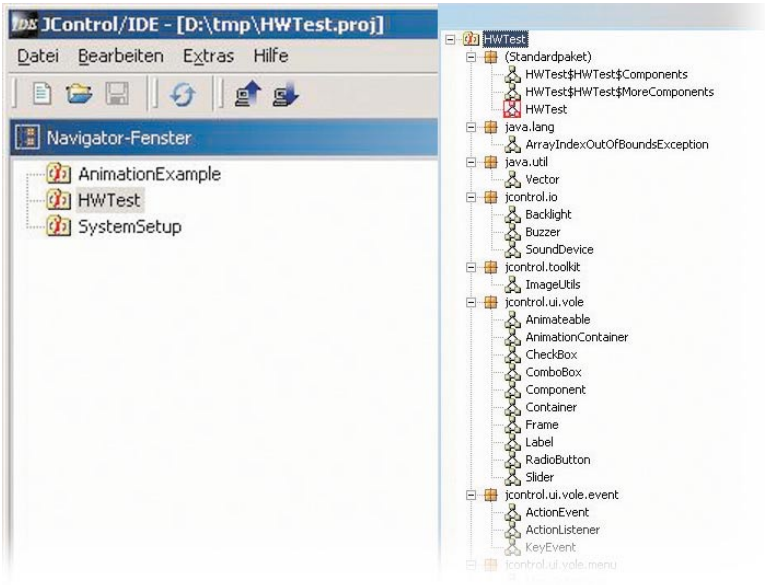


Bild 4: Screenshot der JControl/IDE

JControl/IDE – die grafische Bedienoberfläche zur Projektverwaltung

Bei der JControl-Technologie wurde auch besonderer Wert auf einfache Bedienung gelegt. Anwendungen sollen für jeden mit der vertrauten Entwicklungsumgebung erstellbar sein und dann (beinahe) auf Knopfdruck auf die Zielhardware heruntergeladen werden können.

Hierzu stellt die JControl-Webseite [1] eine integrierte Entwicklungsumgebung (Integrated Development Environment, IDE) zur Verfügung, mit der sich Projekte komfortabel erstellen und auch verwalten lassen.

Nachdem man ein Programm in Java™ geschrieben und dann kompiliert hat, müssen die dabei entstandenen Class-Dateien und andere verwendete Elemente wie etwa Bilder, die angezeigt, oder Melodien, die abgespielt werden sollen, zu einem Archiv zusammengestellt und in den Speicher der JControl-Geräte übertragen werden.

Die Entwicklungsumgebung übernimmt das Zusammenstellen der Class-Dateien für den Controller. Bilder, Melodien oder Schriftarten können vom Anwender in die Projekte eingebunden werden, diese werden dann zusammen mit den Class-Dateien vom JControl/IDE umgewandelt und in das Modul geladen. Dabei erkennt das Programm alle an den PC angeschlossenen JControl-Module automatisch, stellt deren Version fest und ordnet entsprechende Profile mit Meta-Informationen zu, die für die Umwandlung der Class-Dateien erforderlich sind. Abbildung 4 zeigt die JControl/IDE. In der linken Hälfte des Fensters werden die Projekte und in der rechten die in den Projekten verwendeten Klassen dargestellt.

Wenn in einem Projekt eigene Elemente, also Bilder, Melodien oder Schriftarten zum Einsatz kommen sollen, müssen diese in bestimmten Dateiformaten vorliegen. Um dies zu realisieren, stehen in der Entwicklungsumgebung Funktionen zur Verfügung, mit denen sich diese Dateiformate erstellen lassen.

Bilder können mit dem integrierten Zeichenprogramm „PictureEdit“ selbst erstellt werden, oder bestehende Bitmap-Dateien (.BMP) können importiert und dann im neuen Format gespeichert werden.

Ähnlich ist es mit den abzuspielenden Melodien. Sie können entweder als Midi-Dateien importiert werden, oder man „komponiert“ sich eine eigene Melodie mit Hilfe des integrierten Editors „MelodyEdit“, wie in Abbildung 5 dargestellt ist. Die eingefügten Töne werden dabei über die Soundkarte abgespielt. Die so erstellten Melodien werden später über den Buzzer-Ausgang der Java™-Control-Unit ausgegeben.

Bei den Schriftarten sind True-Type- und Bitmap-Definition-Fonts importier- und modifizierbar. So kann man ganz individuelle Schriften auf das Display bringen. Hierfür steht das Programm „FondEdit“ zur Verfügung.

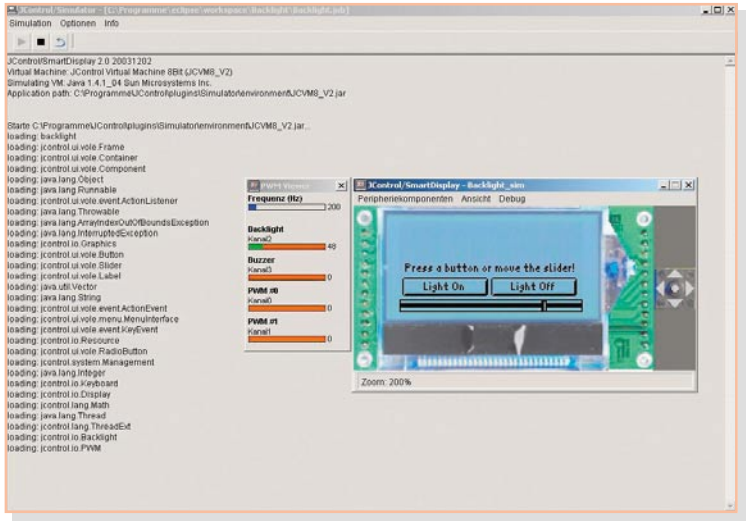


Bild 6: Screenshot des JControl-Simulators mit Beispielanwendung

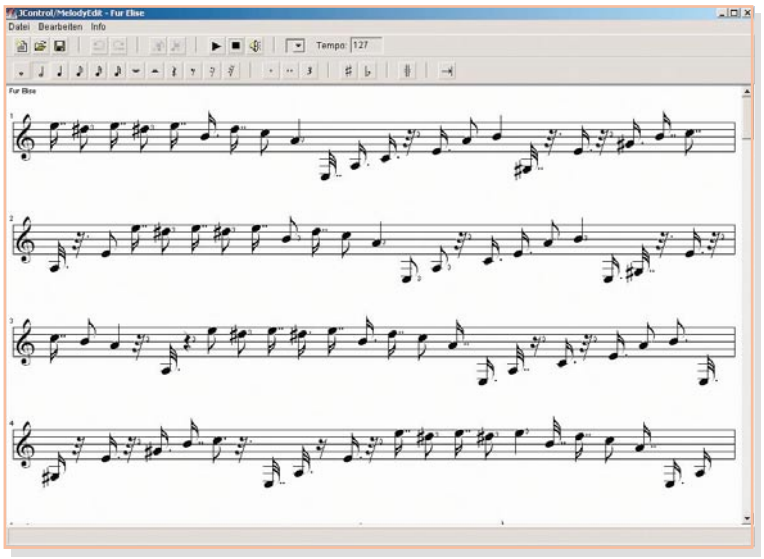
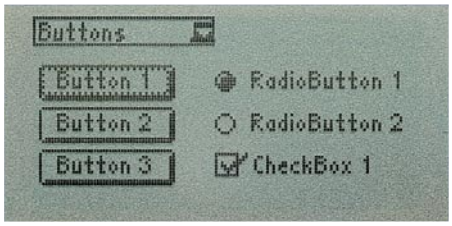


Bild 5: Screenshot des Programms „MelodyEdit“

Einfach simulieren

Ein weiteres wichtiges Feature der Entwicklungsumgebung ist der Simulator.

Hier hat man die Möglichkeit, Java™-Programme auf andere Plattformen zu portieren, genutzt. Der Simulator arbeitet unter Java™ 1.4.2 von Sun Microsystems,



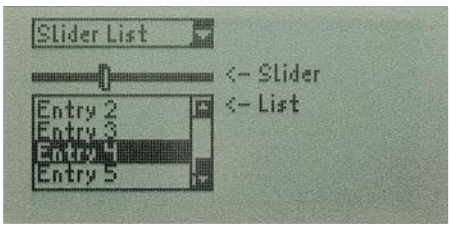


Bild 7: Übersicht über einige Grundelemente der Klasse JControl/Vole

und für ihn wurden die JControl-Klassen entsprechend angepasst, so dass sich ein Großteil der Funktionen der JControl-Geräte ohne Aufwand und zeitsparend am PC simulieren und testen lassen.

Im Simulator wird ein Displaymodul auf dem Computerbildschirm dargestellt, das den Displayinhalt so anzeigt, wie er auch später auf dem wirklichen Modul aussieht.

Neben dem Displaymodul werden noch vier Pfeil- und eine Select-Taste simuliert, die so angeordnet sind wie die Taster auf dem Entwicklungsboard. Hiermit lassen sich alle grafischen Darstellungen und die Navigation durch Menüs simulieren. Außerdem lassen sich Fenster aktivieren, in denen die Zustände der IO-Pins und der PWM-Ausgänge, Aktivitäten der RS-232-Schnittstelle und Inhalte des Flash-Speichers angezeigt werden. In Abbildung 6 ist ein Screenshot des Simulators dargestellt. Simuliert wird hier ein einfaches Programm, bei dem die Display-Hintergrundbeleuchtung mit den beiden Buttons (Schaltfelder) ein- und ausgeschaltet oder mit dem Schieberegler gedimmt werden kann.

Der PWM-Viewer zeigt für jeden PWM-Kanal eine Balkenanzeige mit Werten von 0 bis 255. Bei 0 befindet sich der PWM-Ausgang auf Low-Pegel. Je höher der angezeigte Wert ist, um so größer wird das Verhältnis von Puls- zu Periodendauer. Bei 255 liegt am PWM-Ausgang High-Pegel an. Die Displaybeleuchtung leuchtet bei einer PWM-Einstellung von 0 am hellsten, da sich zwischen Controller und Display noch ein invertierender Transistor befindet.

Die aufgeführten Editoren für Bilder, Melodien und Schriften sowie der Simulator lassen sich aus der JControl/IDE heraus starten, so dass dem Programmierer eine kompakte Entwicklungsumgebung zum Erstellen und Verwalten der Projekte zur Verfügung steht.

Die JControl/IDE wird laufend weiterentwickelt, so ist z. B. auch ein integrierter Java™-Compiler geplant. Mit der Bedienung der JControl/IDEs werden wir uns in einem späteren Artikel noch genauer befassen.

Grafische Benutzeroberflächen mit JControl/Vole

Die Klasse JControl/Vole ist eine Zusammenstellung von Elementen, die es dem Programmierer erlauben, auch auf kleinen Displays ansprechende und einfach zu benutzende Bedienoberflächen zu erstellen.

Zur Verfügung stehen hierbei einfache Elemente wie Knöpfe (Buttons), Schieberegler (Sliders), Ankreuzfelder (Checkboxes) oder Listen (Lists), wie sie beispielhaft in Abbildung 7 dargestellt sind, aber auch komplexere Elemente wie Zeiger-Messinstrumente, 2D-Diagramme und eine Reihe von grafischen Anzeigeelementen, wie man sie in Abbildung 8 sehen kann.

Die vollständige Implementierung der JControl/Vole würde etwa 60 kB Flash-Speicher in Anspruch nehmen. Da aber bei der Erstellung der Projekte nur die wirklich benötigten Elemente zum Projekt gelinkt werden, verringert sich der Bedarf an Speicherplatz entsprechend.

Um den Speicherbedarf der Klasse so gering wie möglich zu halten, wurde auch auf einige Funktionen, die bei der Erstellung von grafischen Bedienoberflächen für PC-Plattformen zur Verfügung stehen, verzichtet. So muss man z. B. die Elemente mit Hilfe von absoluten Koordinaten auf dem Display platzieren, da keine relativen Koordinaten unterstützt werden. Berücksichtigt man allerdings die Größe des Displays mit 128 x 64 Pixel, so bleibt die Anordnung der Elemente in der Praxis doch recht überschaubar und eine Positionierung mit absoluten Koordinaten stellt kein Problem dar.

Ausblick

Falls Sie durch diese kurze Einführung neugierig geworden sind und nicht warten wollen, bis die Java™-Control-Unit und das zugehörige Evaluation Board vorgestellt werden, sondern sich schon vorher einen Eindruck davon verschaffen wollen, wie die Programme auf dem Display dargestellt werden, können Sie sich bis zur nächsten Ausgabe schon einmal auf der JControl-Webseite [1] umsehen.

Dort finden sich neben der JControl/IDE auch fertige Demo- und Beispielprogramme, die den Java™-Quelltext, die

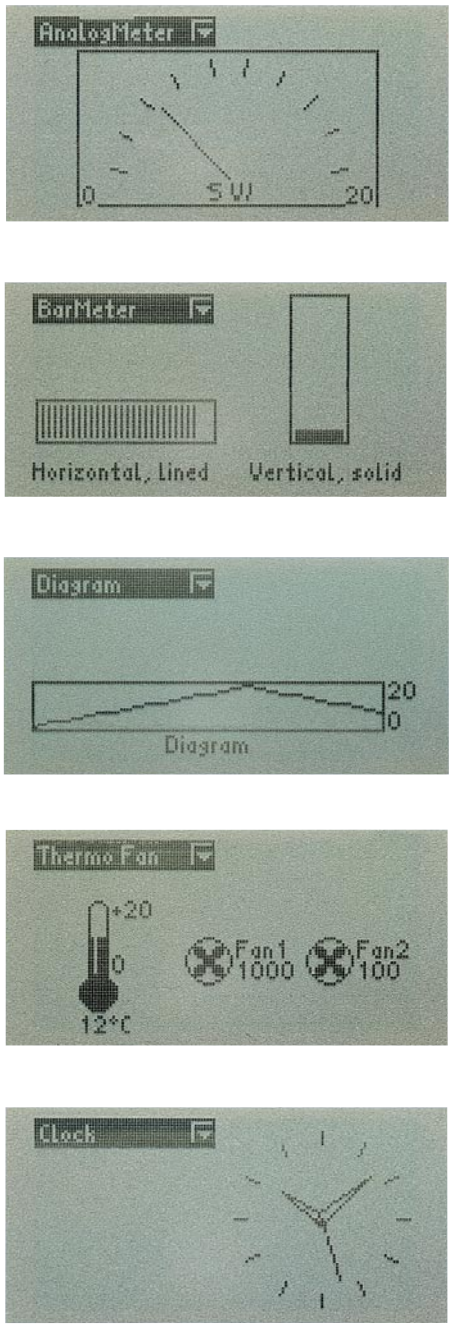


Bild 8: Auch komplexe Elemente findet man in der Klasse JControl/Vole

kompilierten Class-Dateien und auch die Projektdateien für die IDE enthalten. Diese Projekte können also direkt geöffnet und simuliert werden. Für die Simulationen ist hier die Einstellung „JControl/SmartDisplay“ zu verwenden. Unser Projekt wird später jedoch als „Java™-Control-Unit“ in den Simulator implementiert werden.

Im zweiten Teil der Artikelserie wenden wir uns zunächst ausführlich der Hardware, der Java™-Control-Unit und dem Evaluation Board zu.

Internet:

[1] http://www.jcontrol.org