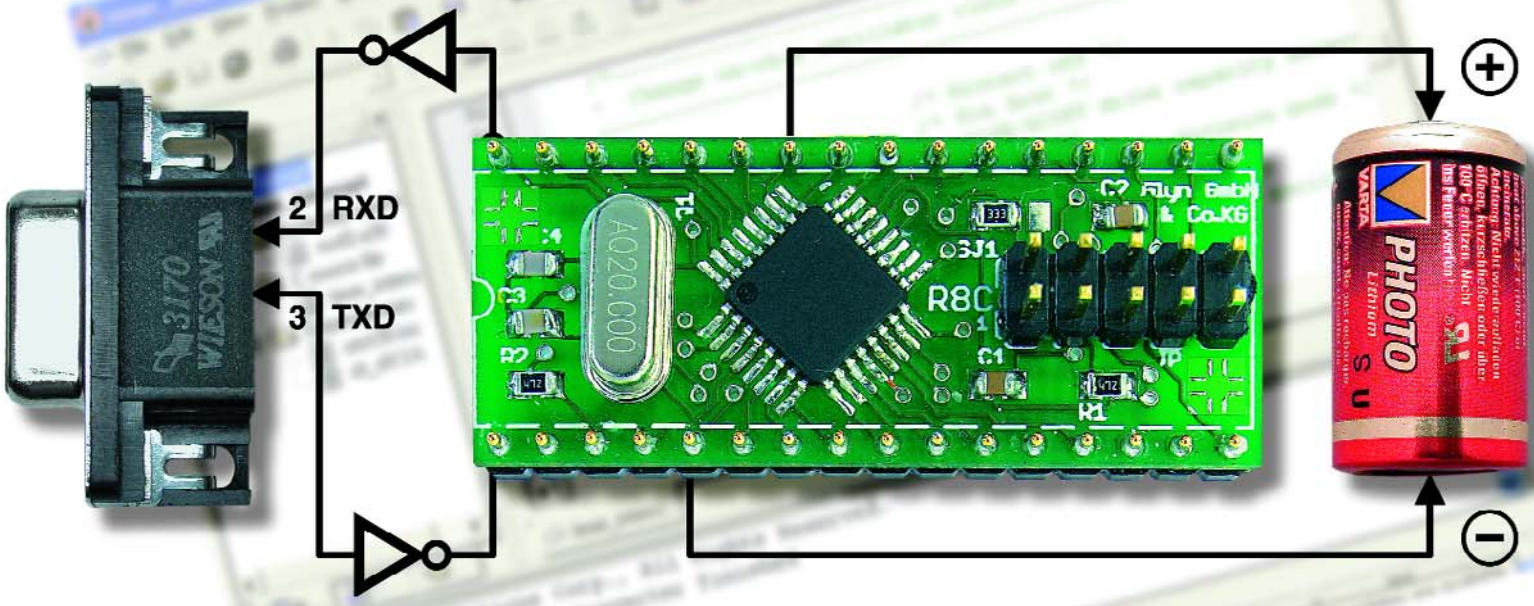
Die kleine 16-bit-M



Von Gunther Ewald und Burkhard Kainka

Elektor und Glyn machen es möglich: Zum ersten Mal enthält eine europäische Elektronik-Zeitschrift ein komplettes Mikrocontroller-Board inklusive Software-CD. Im letzten Heft haben wir den R8C von Renesas schon einmal vorgestellt. Dieses Mal wollen wir sofort loslegen.

Nun ist es endlich so weit! Für alle ELEKTOR-Leser gibt es eine Platine mit dem Mikrocontroller R8C/13 und die nötige Software zum Nulltarif – exklusiv zur Verfügung gestellt von der Firma Glyn. Für die R8C/Tiny-Familie des Herstellers Renesas sprechen drei gewichtige Argumente. Erstens erhält man 16-bit-Rechenpower für wenig Geld, zweitens einen kostenlosen und dennoch sehr leistungsfähigen C-Compiler, und drittens ist kein Programmiergerät erforderlich, weil einfach über die RS232-Schnittstelle geflasht wird. In der letzten Ausgabe haben wir das Board und die Software schon einmal vorgestellt (wer den Artikel verpasst hat, kann ihn kostenlos von der ELEKTOR-Website unter www.elektor.de herunterladen; auf der Homepage findet man in der rechten Spalte einen Link auf das R8C-Projekt). In diesem Artikel wollen wir nun mit dem Platinchen loslegen.

Die Hardware

In der Klarsichtverpackung auf der ersten Seite dieser ELEKTOR-Ausgabe finden Sie die SMD-bestückte flache Platine und zwei Stiftleisten, die Sie noch selbst einlöten müssen (**Bild 1**). Sie erhalten so ein komplettes Prozessor-Modul in Form eines 32-poligen DIL-ICs (**Bild 2**). Auf der Platine ist noch ein Platz für einen 14-poligen Pfostenstecker vorgesehen, der zunächst nicht bestückt zu werden braucht, weil er nur für den Debugger E8 nötig ist.

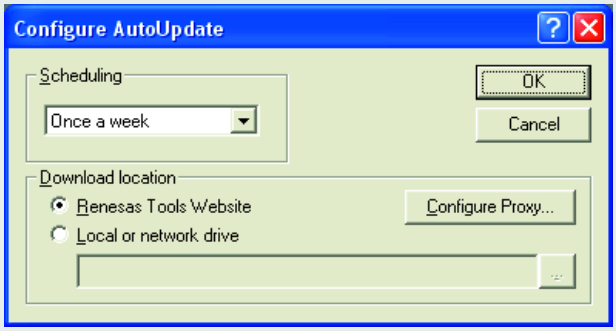
Der eigentliche Mikrocontroller R8C/13 befindet sich im 32-poligen SMD-Gehäuse in der Bauform LQFP mit einer Größe von 7 x 7 mm und einem Pinabstand von 0,8 mm. Der Aufdruck R5F21134FP#U0 verrät, dass es sich um einen R8C/13 mit 16 KByte Flash-ROM handelt. Den R8C/13 haben wir deshalb ausgewählt, weil er die Eigenschaften seiner „Brüder“ R8C/10, R8C/11 und R8C/12 mit einschließt. Zusätzlich sind bereits ein 20-MHz-Quarz mit den erforderlichen Kondensatoren und einige weitere Kondensatoren und Widerstände auf der Platine vorhanden. Insgesamt handelt es sich um ein komplettes Mikrocontroller-System! Wäre schon ein Programm gela-

aschine

Unser R8C-Starterkit ist da – und los geht´s!

Die Software

Bei der Installation der benötigten Software sollten Sie sich genau an die Reihenfolge halten, damit später auf Ihrem PC alles so aussieht wie hier beschrieben. Zuerst wird der Monitor/Debugger KD30 installiert, danach der C-Compiler NC30 mit der Entwicklungsumgebung HEW und ein Update. Diese Reihenfolge ist wichtig, weil die HEW dann schon den Debugger vorfindet und richtig einbindet. Als Nächstes wird das Debugger-Package installiert, um den Debugger mit in die Entwicklungsumgebung zu integrieren. Später braucht man nur noch die HEW zu starten und hat alles zusammen auf dem Bildschirm. Schließlich muss noch das Flash Development Toolkit FDT von Renesas installiert werden, mit dem sich fertige Programme in den Controller laden lassen.



Nach dem Einlegen der CD erscheint zuerst eine Produktübersicht im PDF-Format. Wenn Sie diese beenden, sollten Sie das Grundverzeichnis der CD sehen. Alternativ können Sie das Laufwerk mit der rechten Maustaste anklicken und dann „Öffnen" wählen, um das PDF-Intro zu überschlagen. Die wichtigsten Verzeichnisse auf der CD sind \Software und \Sample_NC30. Sie enthalten alle erforderlichen Programme (die nun installiert werden sollen) beziehungsweise erste Beispielprojekte.

1. KD30

Auf der CD finden Sie im Verzeichnis \Software\kd30400r1\ die ausführbare Installationsdatei KD30V410R1_E_20041203.EXE. Starten Sie die Installation und bestätigen Sie den Pfad C:\MTOOL\.

2. NC30

Starten Sie die Setup-Datei nc30wav530r02_2_ev.exe aus dem Verzeichnis \Software\ nc30v530r0_hew\ der CD. Zuerst müssen Sie wählen, ob Sie die japanische oder die englischsprachige Variante installieren wollen – den meisten unserer Leser wird die englische Sprache vertrauter sein. Das Programm schlägt für die Installation den Pfad c:\programme\Renesas vor, den Sie bestätigen sollten.

Ein zweiter Pfad wird für die Toolchain vorgeschlagen: C:\Renesas\NC30WA\V530R02. Den Pfadnamen sollten Sie ebenfalls – wie auch alle folgenden Vorschläge - akzeptieren.

Am Ende der Installation wird ein individueller Site-Code angezeigt, den Sie aber nicht weiter beachten müssen, da er nur für die Registrierung der Software nötig ist (wenn Sie die Vollversion des Compilers kaufen, bekommen Sie ohnehin eine CD mit einem eigenen Freischalt-Code).

Im Anschluss an die Installation der HEW startet die Installation des AutoUpdate. Bestätigen Sie (wie im Bild gezeigt) ein wöchentliches Update von der Renesas Tools Website. Auf diese Weise halten Sie Ihre Software jederzeit „frisch". Falls der PC, auf dem ihre Software installiert wird, keinen Internetzugang besitzt, können Sie die Installation des AutoUpdate getrost abbrechen. Auf der CD befindet sich die letzte aktuelle Update-Datei, die Sie als Nächstes installieren müssen.

Nach der Installation schaut der Updater gleich nach, ob es etwas Neues gibt und lädt automatisch die neuesten Änderungen. Das erste Update wird dann automatisch installiert. Danach ist zuerst ein Neustart des PCs fällig.

3. Update der HEW

Dieser Schritt ist nur erforderlich, falls Sie nicht ohnehin schon das neueste Update über das Internet geladen und installiert haben. Starten Sie das erste Update der HEW von der CD, indem Sie die Datei hewv40003u.exe im Verzeichnis \Software\HEW_V.4.00.03.001_Update auf der CD aufrufen. Damit wird ihr Compiler auf den letzten Stand (Database Version 7.0) gebracht. Dies ist wichtig, da die Beispielprojekte für diese Version erstellt wurden. Sie können zwar ältere Projekte laden, die dann automatisch geupdated werden, ein Rückschritt auf eine ältere Version des Compilers ist aber nicht so einfach möglich.

4. Das Debugger Package

Starten Sie die Installationsdatei m16cdebuggerv100r01.exe aus dem Verzeichnis \Software\Debugger Package\ auf der CD. Folgen Sie den Anweisungen des Installationsprogramms und bestätigen Sie die Lizenzbedingungen. Alles andere passiert automatisch. Dann ist ein erneuter Neustart des Rechners nötig.

5. Das Flash Development Toolkit

Installieren Sie das FDT mit der Installationsdatei fdtv304r00.exe aus dem Verzeichnis \Software\Flasher_FDT der CD. Bestätigen Sie alle vorgeschlagenen Einstellungen. Der Rest erfolgt automatisch.

Wenn alles installiert ist, finden Sie unter Start/Programme eine Programmgruppe Renesas. Die beiden entscheidenden Programme darin sind der „High-performance Embedded Workshop" und das „Flash Development Toolkit".

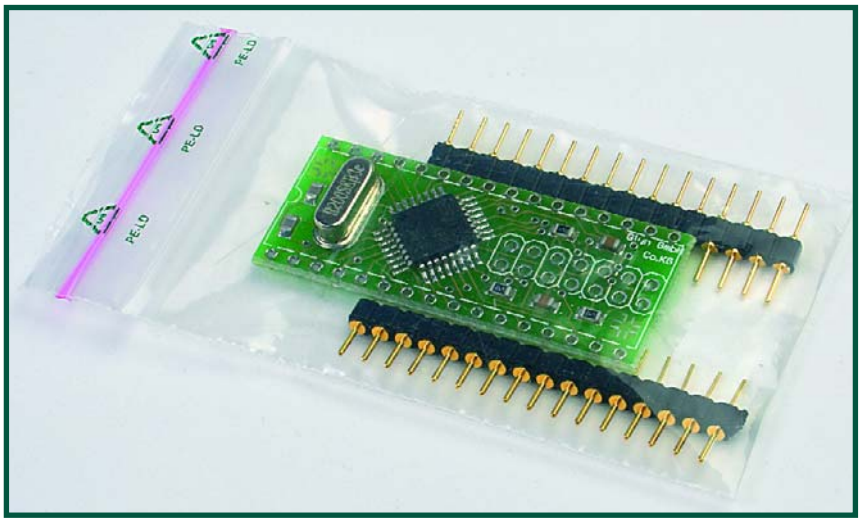


Bild 1. Die mitgelieferte Hardware ist ein komplettes Mikrocontrollersystem!

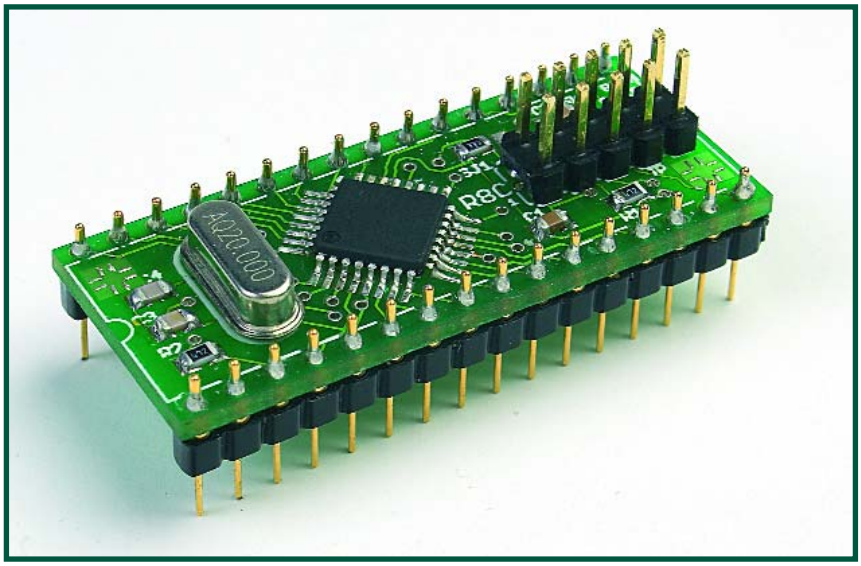


Bild 2. Hier ist die Platine mit Stiftleisten bestückt.

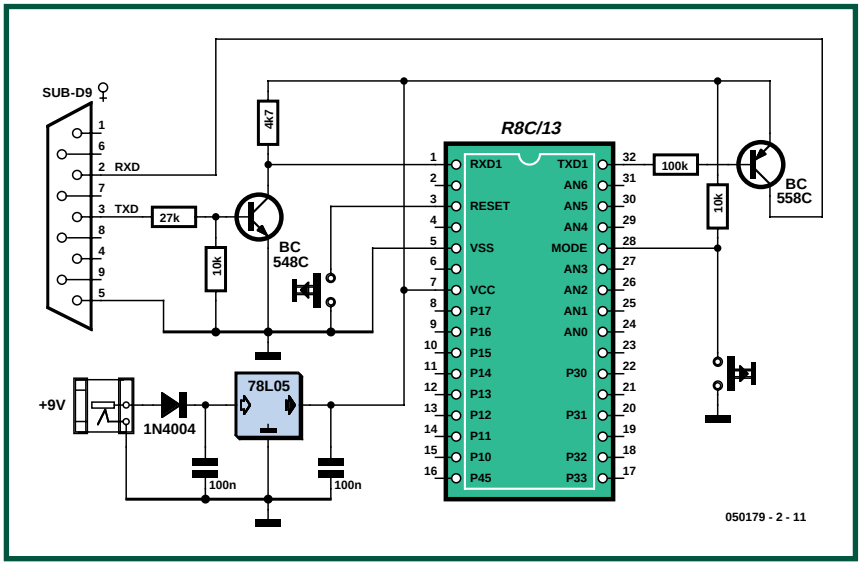


Bild 3. Der Schaltplan zeigt ein Minimalsystem für den ersten Test.

den, müsste man nur noch eine Betriebsspannung von 3,3 V oder 5 V anlegen und man hätte eine vollständige Anwendung.
Zum Laden eines Programms braucht man kein spezielles Programmiergerät, sondern nur eine serielle Schnittstelle. Der Controller besitzt nämlich eine serielle Debug-Schnittstelle und ein entsprechendes Bootprogramm, über das sich die Software ins Flash-ROM übertragen lässt.

First Contact

In der folgenden Ausgabe von ELEKTOR stellen wir ein vollständiges Entwicklungssystem mit RS232- und USB-Schnittstelle vor. Aber so lange wollen Sie natürlich nicht mit Ihren ersten Versuchen warten. Deshalb soll hier eine Lösung beschrieben werden, welche lediglich etwas Material aus der Bastelkiste benötigt:

- eine Spannungsversorgung mit 5 V, am besten über einen Spannungsregler,
- invertierende Pegelwandler zum Anschluss an die RS232-Schnittstelle,
- ein Reset-Taster,
- ein Mode-Schalter zum Einschalten des Programmier-Modus.

Die Controllerplatine verbindet alle Pins des Mikrocontrollers eins zu eins. Wie schon gesagt, sind auch ein Quarz, einige Kondensatoren und ein paar Widerstände mit auf dem Board. Der Schaltplan (**Bild 3**) zeigt nur die für den „Schnellstart“ wichtigsten Anschlüsse, damit man sich möglichst schnell orientieren kann. Die nicht bezeichneten Anschlüsse müssen unbedingt frei bleiben! **Bild 4** zeigt einen Probe-Aufbau auf einer Steckplatine.

Die serielle Schnittstelle des PCs wird hier über Transistor-Inverter angeschlossen. Zwar könnte man ebenso gut einen MAX232 einsetzen, aber die Transistoren sind wahrscheinlich schneller in der Bastelkiste zu finden. Ein NPN-Transistor BC548C invertiert das TXD-Signal vom PC und führt es an den RXD1-Eingang des Controllers. Dieser Eingang hat keinen internen Pullup, deshalb ist der Kollektorwiderstand wichtig. In Gegenrichtung steuert TXD1 den PNP-Transistor BC558C an. Der RXD-Eingang am PC hat seinen eigenen Pulldown, sodass man hier auf den Kollektorwiderstand verzichten kann.

Auf die Plätze, fertig, Flash!

Der MODE-Eingang des Controllers entscheidet darüber, ob nach einem Reset das interne Boot-Programm oder ein geladenes Userprogramm ausgeführt wird. Bei geöffnetem Schalter ist MODE über einen Widerstand von 10 kΩ hochgezogen, sodass das Userprogramm startet. Will man ein Programm in den Flash-Speicher laden, muss der Mode-Schalter geschlossen werden, MODE wird also low. Danach muss einmal kurz der Reset-Taster betätigt werden. Damit startet der Controller den Debug-Mode, in dem neue Software ins Flash-ROM geladen werden kann. Nach der Übertragung öffnet man den MODE-Schalter und betätigt noch einmal den Reset-Taster. Auf diese Weise startet man das so geladene Programm. Aber vor dem Vergnügen kommt erst einmal die Pflicht, nämlich die Installation der Software. Eine Schritt-für-Schritt-Anleitung finden Sie im Kasten. Bitte halten Sie sich genau an die Reihenfolge, damit später auf Ihrem PC alles so aussieht wie beschrieben.

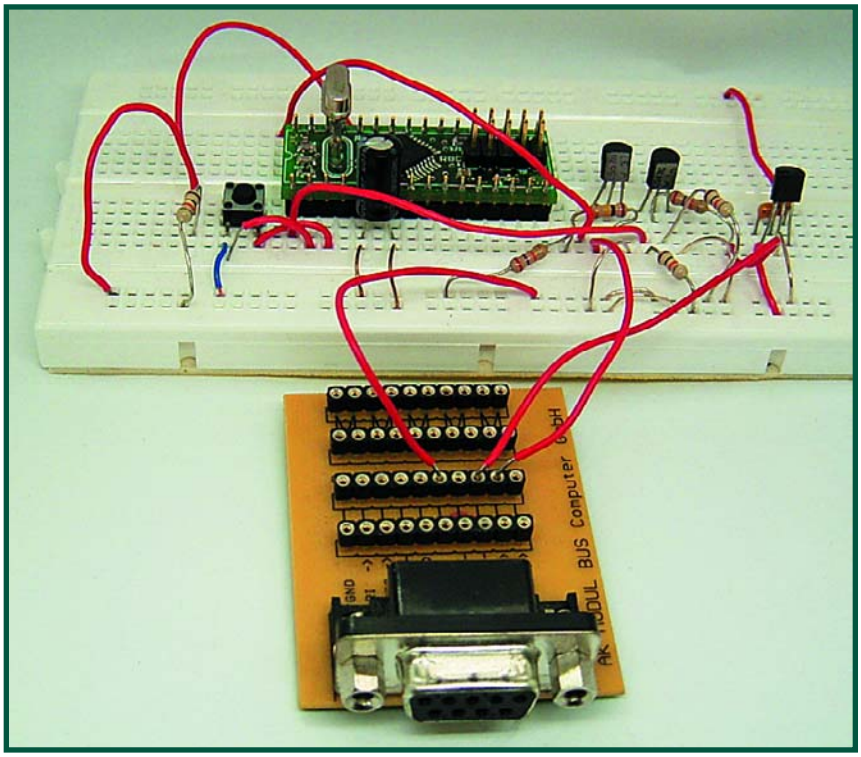


Bild 4. Auf einer Labor-Steckplatine haben wir das Test-System probeweise aufgebaut.

Zuerst soll das „Flash Development Toolkit“ ausprobiert werden, indem ein fertiges Programm in den Controller geladen wird. Die Schritte zum Entwickeln eines eigenen Programms wollen wir nämlich einfach erst einmal überschlagen, damit ein Erfolg nicht zu lange auf sich warten lässt.
Das FDT finden Sie nach erfolgreicher Installation in der Programmgruppe Renesas im Windows-Startmenü. Das Programm gibt es dort in einer umfassenden Version und in der kompakten Basic-Version. Starten Sie das „Flash Development Toolkit Basic“ (**Bild 5**).
Beim ersten Start nehmen Sie zunächst die nötigen Einstellungen vor (später können Sie dieses Menü über „Options/New Settings“ erreichen). Wählen Sie den Controller-Typ R5F21134 und das obere der beiden angebotenen Kernel-Protokolle (**Bild 6**). Im nächsten Fenster wird die verwendete Schnittstelle COM1 bis COM4 ausgewählt. Das dritte Fenster verlangt die Angabe einer Baudrate für die Verbindung zum Controller. Geben Sie hier 9600 Baud ein (**Bild 7**).

Und schließlich muss noch angegeben werden, ob der Controller gegen Auslesen geschützt werden soll. Da die Gefahr krimineller Industriespionage bei den ersten Übungsprogrammen äußerst gering ist, kann man auf jeden Schutz verzichten. Übernehmen Sie die Einstellungen wie in **Bild 8**. Damit ist alles vorbereitet.

Hurra, es blinkt!

Verbinden Sie nun Ihre Hardware mit der angegebenen COM-Schnittstelle. Schließen Sie den Mode-Schalter gegen Masse, und betätigen Sie einmal kurz den Reset-Taster. Der Mikrocontroller befindet sich nun im Boot-Modus und wartet darauf, dass man ihm Daten schickt.
Laden Sie nun ein fertig kompiliertes Programm in den Controller. Auf der CD finden Sie das Verzeichnis Sample_NC30 mit einigen Beispielprojekten. Unter anderem ist das Projektverzeichnis Sample_NC30\ port_toggle und darin wiederum das Unterverzeichnis port_toggle\ Release enthalten. Im letztgenannten Unterordner findet man die Datei port_toggle.mot. Dies ist ein Programm im Motorola-Hex-Format, das direkt in den Controller geladen werden kann.
Geben Sie den Pfad auf diese Datei an und starten Sie den Ladevorgang mit „Program Flash“. Der Vorgang dauert etwa zwei Sekunden. Zuerst wird das Flash gelöscht, dann das neue Programm übertragen. Wenn alles korrekt funktioniert hat, erscheint die Erfolgsmeldung „Image successfully written to device“. Öffnen Sie den Mode-Schalter und betätigen Sie kurz die Reset-Taste. Damit startet das gerade geladene Programm.
Es toggelt die ersten vier Portanschlüsse des Ports 1 (also P1_0 bis P1_4) mit einem langsamen Takt, sodass man die Zustände über LEDs mit Vorwiderständen beobachten kann. Ports des R8C und der anderen M16-Mikrocontroller sind in Ausgaberichtung sowohl im High-Zustand als auch im Low-Zustand niederohmig. Man darf LEDs also nur mit Vorwiderständen (z.B. 1k) anschließen (**Bild 9**).
In der gleichen Weise wie beschrieben können Sie auch alle folgenden Beispiele in den Controller übertragen. Sollte sich der R8C einmal nicht flashen lassen (communication error), muss man ihn eine Minute spannungslos machen, damit der interne RAMloader gelöscht wird. Der Fehler kann auftreten, wenn Sie vorher mit dem Debugger gearbeitet haben.

R8C als Musikus

Noch ein Hardwaretest gefällig? Dann laden Sie einmal die Mot-Datei des Projekts R8C_Jingle_Bells in den Controller. Schließen Sie einen kleinen 8-Ω-Lautsprecher oder einen Kopfhörer mit einem Vorwiderstand von 1 kΩ an (**Bild 10**) und starten Sie den Control-

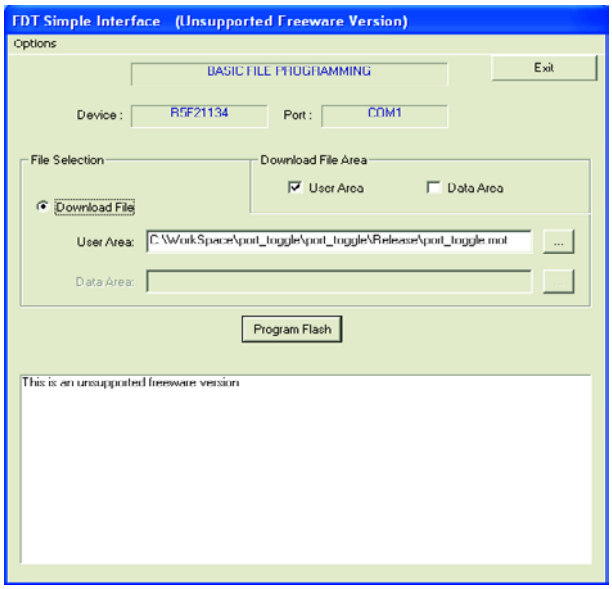


Bild 5. Das Flash Development Toolkit Basic.

ler. Nun hören Sie eine einfache Melodie. Es musste natürlich etwas Weihnachtliches sein!
Das Programm arbeitet übrigens ohne den Quarzoszillator, sondern allein mit dem internen High Speed Ringoszillator mit 8 MHz. Hält man die Tastspitze eines Oszilloskops an die Quarzanschlüsse Xin (Pin6) oder Xout (Pin 4), ist kein Taktsignal zu sehen (ganz im Gegensatz zum Projekt Port_Toggle). Der interne RC-Oszillator reicht für die Aufgabe völlig aus, denn offensichtlich hört man keine schrägen Klänge aus dem Lautsprecher. Man könnte jetzt einen R8C/13 flashen, mit einem Piezolautsprecher und einer 3-V-Knopfzelle verbinden und an den Weihnachtsbaum hängen bzw. in eine Weihnachtskarte legen.
Jetzt wissen wir, dass die Hardware funktioniert. Und schon kribbelt es in den Fingern, selbst etwas zu programmieren, oder etwa nicht? Also gut, starten Sie die integrierte Entwicklungsoberfläche HEW. Damit es am Anfang nicht zu schwierig wird, soll erst einmal ein vorhandenes Projekt verändert werden. Außerdem werden wir am Anfang ohne den Debugger arbeiten.

Das hohe C

Mit der HEW können Sie Assembler-Projekte erstellen. Allerdings ist das

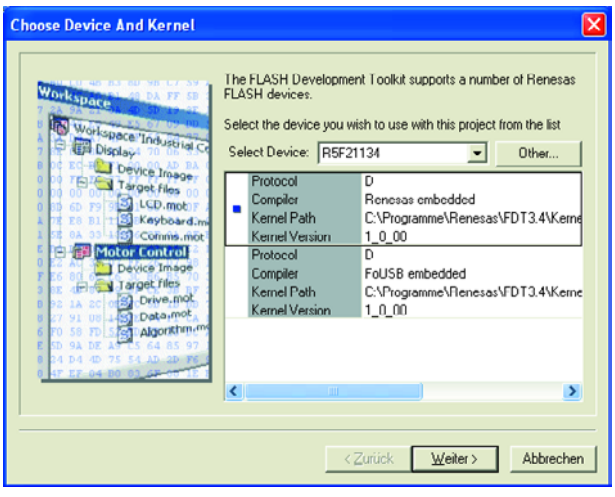


Bild 6. Auswahl des Controllers und des Übertragungsprotokolls.

Programmieren in Assembler beim R8C wesentlich schwieriger als in C. Es gibt nämlich so viele unterschiedliche Datenformate und Adressierungsarten! Der C-Compiler macht dagegen alles richtig. Sie müssen gar nicht wissen, ob der Controller gerade ein Word, ein Byte oder ein Bit bearbeitet. Daher ist C in diesem Fall auch für diejenigen einfacher, die bisher nur mit Assembler gearbeitet haben. Die ungewohnten Schreibweisen erklären sich mit einigen Beispielen wie von selbst. Deshalb: Keine Angst vor C!
Kopieren Sie zunächst das komplette Projekt „port_toggle“ von der CD auf Ihren PC. Verwenden Sie z.B. das Verzeichnis C:\WorkSpace, das auch von der HEW vorgeschlagen wird, wenn ein neues Projekt begonnen wird. Starten Sie dann in der Programmgruppe Renesas den „High-performance Embedded Workshop“. Beim Start erscheint ein Auswahlfenster, in dem Sie angeben können, ob Sie ein neues Projekt erstellen oder ein bestehendes laden wollen.
Mit Datei/Open Workspace öffnen Sie die Datei port_toggle.hws. Links sehen Sie danach alle Dateien, die zum Projekt gehören. Klicken Sie auf port_toggle.c, um den eigentlichen Quelltext zu öffnen.Dann sollte alles so aussehen wie in **Bild 11**.
Dieses Projekt soll nun einmal probeweise kompiliert werden. Zuerst muss klar sein, ob Sie eine Debug-Version oder eine Release-Version erzeugen wollen. Für den Anfang wollen wir noch ohne den Debugger arbeiten. Wählen Sie daher unter Build/Build Configurations die Einstellung Release. Danach starten Sie die Übersetzung mit Build/Build All. Damit wird der C-Quelltext übersetzt, gelinkt und als Mot-File in das Ausgabeverzeichnis \Release geschrieben. Der ganze Vorgang wird unten im Build-Fenster aufgelistet. Am Ende erhalten Sie die ersehnte Erfolgsmeldung:

Build Finished
0 Errors, 1 Warning

Null Fehler, sehr gut! Eine Warnung kommt öfter mal vor und ist nicht weiter tragisch. In diesem Fall lautet sie: „Warning (ln30): License has expired, code limited to 64K (10000H) Byte(s)“. Das muss Sie aber nicht beeindrucken, denn Sie arbeiten mit der freien Version des Compilers, und 64 KByte sind ohnehin viel mehr als der R8C/13 fassen kann. Wenn Sie möchten, können Sie das Ausgabefile noch einmal in den

Controller laden und starten. Es wird genauso gut funktionieren wie das Mot-File auf der CD.

Nun zum Quelltext:

```
while (1)           /* Loop */
      {
      p1_0 = 0;
      p1_1 = 0;
      p1_2 = 0;
      p1_3 = 0;
      for (t=0; t<50000; t++);
      p1_0 = 1;
      p1_1 = 1;
      p1_2 = 1;
      p1_3 = 1;
      for (t=0; t<50000; t++);
      }
```

Im Kern des Programms finden Sie eine einfache Schleife, die sich selbst erklärt. Zuerst werden vier Portbits eingeschaltet, dann kommt eine Warteschleife, dann werden die Bits wieder ausgeschaltet, und schließlich wird wieder gewartet. Selbst wer keine Erfahrungen mit C hat, sieht gleich, wo er hier etwas verändern kann. Sie könnten z.B. die Warteschleifen verkürzen und die ganze Sache etwas schneller machen. Statt 50.000 soll dann nur bis 25.000 gezählt werden. Oder Sie versuchen einmal, wie schnell es wird, wenn Sie nur bis 2 zählen lassen. Man kann die Warteschleife sogar ganz entfernen, indem man sie mit dem Kommentarzeichen „//“ unwirksam macht. Mit einer LED ist dann jedenfalls nichts mehr zu erkennen, aber ein Oszilloskop zeigt ein Rechtecksignal hoher Frequenz. Nach jeder Änderung müssen Sie das Programm mit Build All übersetzen und dann mit FDT erneut in den Controller laden.

Taktgefühl

Kommen wir zum Oszillator des Controllers. Das Beispiel Port_Toggle verwendet den Quarzoszillator mit 20 MHz. Wie bereits weiter oben erwähnt, gibt es aber auch noch zwei interne RC-Oszillatoren mit 125 kHz und 8 MHz. Tatsächlich startet der Controller immer zuerst mit dem Low-Speed-Ringoszillator. Ein Blick in das Datenblatt des R8C/13 zeigt die komplexe Takterzeugung des R8C mit insgesamt drei Oszillatoren und mehreren optionalen Teilerstufen.
Das Beispielprojekt Port_Toggle demonstriert, wie man vom 125-kHz-Oszillator auf den Quarzoszillator umschaltet:

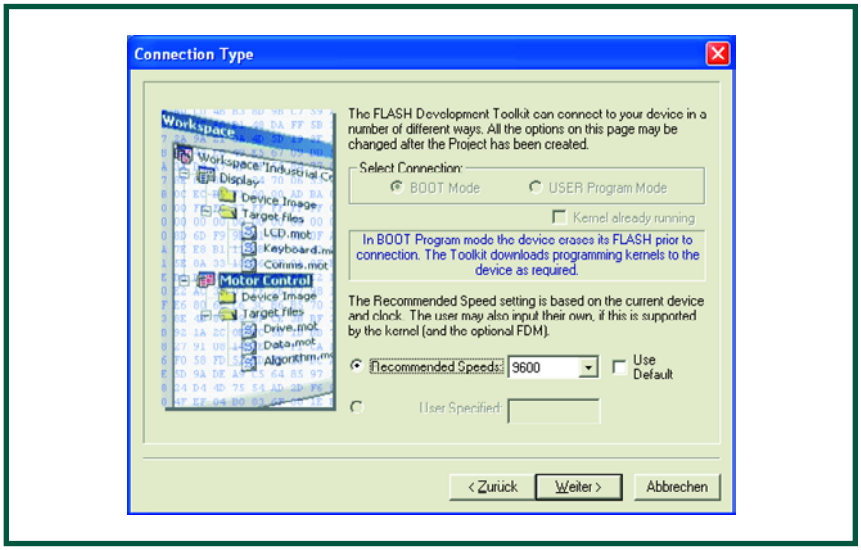


Bild 7. Hier wird die Baudrate gewählt.

```
    prc0 = 1;     /* Protect off */
    cm13 = 1;     /* Xin Xout */
    cm15 = 1;     /* XCIN-XCOUT
drive capacity select bit : HIGH
*/
    cm05 = 0;      /* Xin on */
    cm16 = 0;      /* Main clock =
No division mode */
    cm17 = 0;
    cm06 = 0;      /* CM16 and CM17
enable */
    asm("nop");    /* Waiting for
stable of oscillation */
    asm("nop");
    asm("nop");
    asm("nop");
    ocd2 = 0;      /* Main clock
change */
    prc0 = 0;      /* Protect on */
```

Das Listing zeigt die entscheidenden Programmzeilen am Anfang des Quelltexts. Es müssen einige Steuerbits im System Clock Control Register 0 und 1 umgeschaltet werden, die sich allerdings zuerst in einem geschützten Modus befinden. Der Schutz wird in der ersten Zeile aufgehoben, damit die entscheidenden Bits geändert werden können. Wenn alles umgeschaltet ist, muss man noch etwas warten, bis der gerade eingeschaltete Oszillator stabil schwingt. Dann wird der Takt um- und der Schreibschutz wieder eingeschaltet. Von jetzt an arbeitet das weitere Programm mit 20 MHz.
Entfernen Sie nun einmal den gesamten Block einschließlich prc0 = 1 und prc0 = 0 aus dem Listing. Damit entfällt die Umschaltung des Oszillators, sodass der

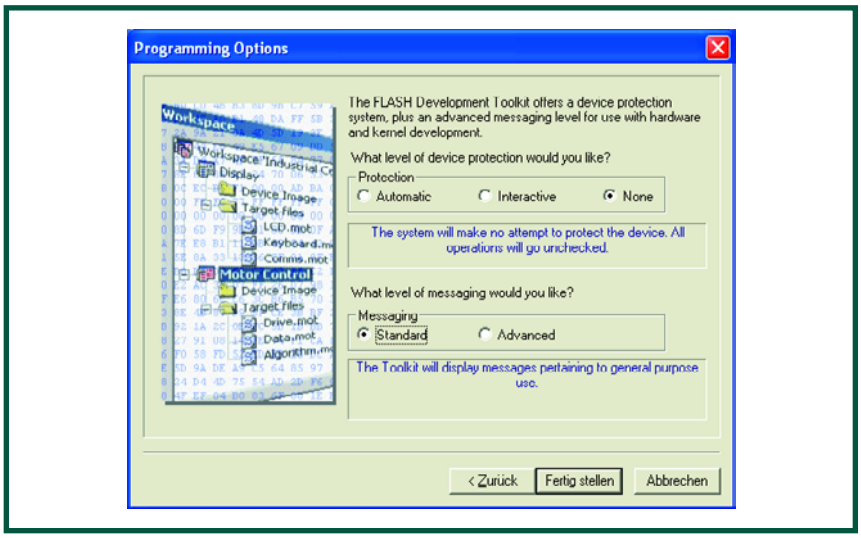


Bild 8. Ein Ausleseschutz ist nicht nötig.

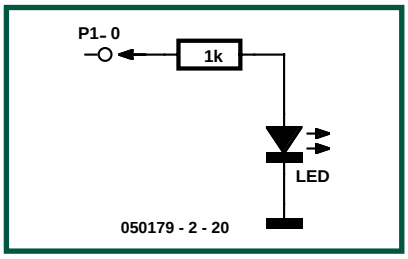


Bild 9. So ist eine LED an P1_0 anzuschließen.

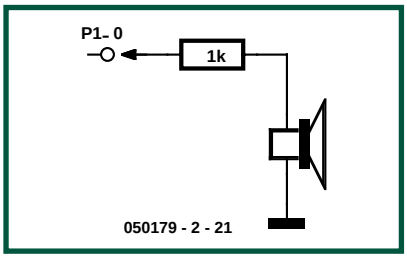


Bild 10. Anschluss eines Lautsprechers an den Mikrocontroller.

Controller weiter mit 125 kHz arbeitet. Damit man danach aber in endlicher Zeit auch etwas zu sehen bekommt, müssen außerdem die Warteschleifen um den Faktor 100 verkürzt werden:

```
void main(void)
{
    pd1 = 0x0F;  /* Set Port 1.0 -
1.3 be used for output*/

   while (1)      /* Loop */
        {
        p1_0 = 0;
        p1_1 = 0;
        p1_2 = 0;
        p1_3 = 0;
        for (t=0; t<500; t++);
        p1_0 = 1;
        p1_1 = 1;
        p1_2 = 1;
        p1_3 = 1;
        for (t=0; t<500; t++);
        }
}
```

Das Blinken ist nun etwas langsamer, aber der Controller arbeitet extrem sparsam.

Ein Kompromiss zwischen ganz schnell und sehr langsam liegt in der Verwendung des internen High-Speed-Ringoszillators mit 8 MHz. Wie man ihn verwendet, verrät das Beispielprojekt R8C_Jingle_Bells. Kopieren Sie die folgenden, entscheidenden Zeilen in Ihr Programm:

```
   prc0 = 1;  // Enable High Speed
Oscillator ( 8MHz )
   hr00 = 1;
   asm("NOP");
   asm("NOP");
   hr01 = 1;
   prc0 = 0;
```

Schauen Sie sich auch einmal das Projekt Timer_Interrupt an. Wer zum ersten Mal mit einem der Timer arbeiten möchte, kann sich natürlich mühsam durch das Datenblatt des R8C/13 arbeiten. Besser ist es aber, wenn man von einem funktionierenden Beispiel ausgeht und dann die entscheidenden Passagen im Datenblatt liest, um selbst kleine Änderungen auszuprobieren. Außer der Initialisierung des Timers verrät das Beispiel Timer_Interrupt auch, wie man in C eine Interrupt-Funktion realisiert. Noch mehr Beispiele findet man in den Application Notes auf der CD.

Wer jetzt schon ein ganz eigenes Projekt erstellen möchte, findet die nötigen **Infos auf der ELEKTOR-Website** unter www.elektor.de (siehe den R8C-Link auf der rechten Spalte der Homepage). Anhand eines einfachen Beispiels erklären wir dort, wie man Schritt für Schritt zur eigenen Anwendung kommt, ohne auf vorhandene Projekte zurückgreifen zu müssen. Und wenn Sie noch Fragen haben: Auf der ELEKTOR-Website wurde auch ein **moderiertes Forum** zum R8C-Starterkit eingerichtet.

(050179-2)

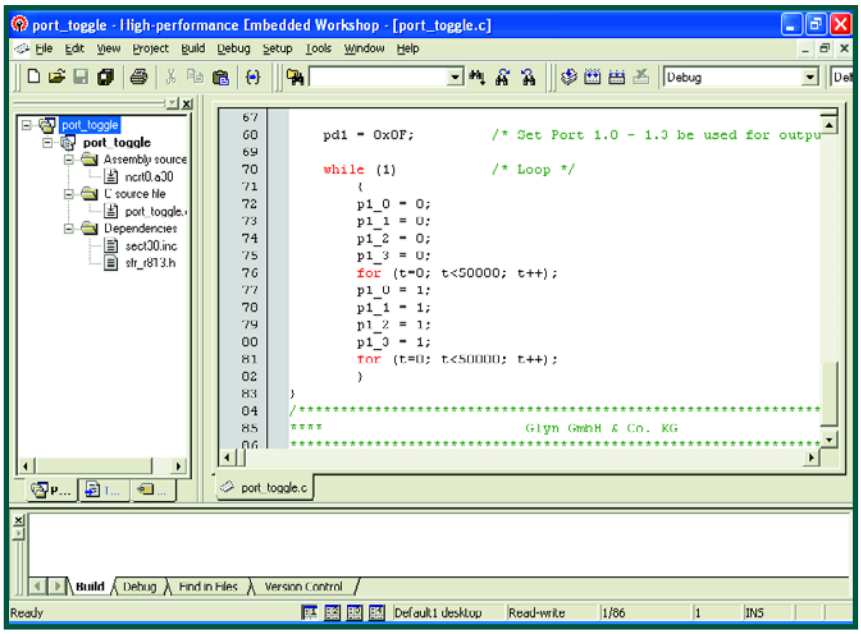


Bild 11. Das Projekt port_toggle in der Entwicklungsoberfläche.

Bezugsquellenhinweis:

Glyn GmbH & Co. KG
www.glyn.de
E-Mail: elektor-r8c@glyn.de