Neues zum EIR

Mehr als „nur“ ein Radio: Ein vielseitiges ARM7-Entwicklungs-Board

Von Antoine Authier & Harald Kipp

Dieser Beitrag zeigt, wofür man das Elektor-Internet-Radio auch verwenden kann: Für die Entwicklung von Erweiterungen und für eigene Projekte auf der Basis der faszinierenden ARM-Controller.

Zunächst geht es schwerpunktmäßig um die notwendigen Tools und wie man diese einsetzt. Anschließend behandeln wir die eigentliche Entwicklung und geben ein konkretes Beispiel rund um eine LED.

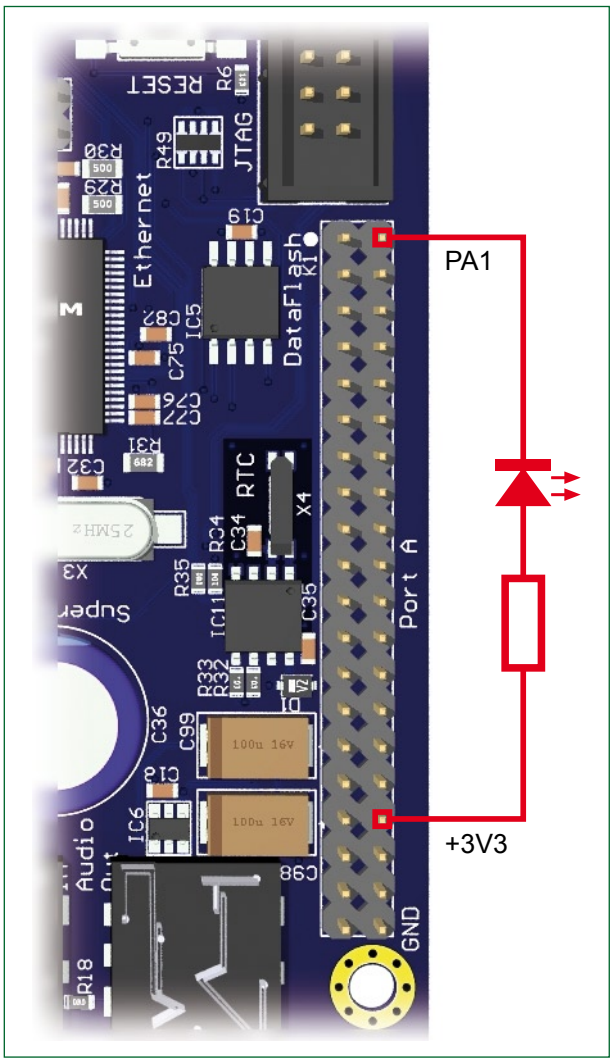

Bild 1.
Die Kathode der LED liegt an Pin 2 (PA1) und der Widerstand an Pin 34 von K1.

Entwicklungsumgebung

Als erstes installiert man natürlich die Entwicklungsumgebung. Für diesen Betrag wird Windows als Betriebssystem vorausgesetzt. Eine Installation unter Linux ist ebenfalls möglich. Die Infos hierzu finden sich auf der beim EIR-Kit mitgelieferten CD. Wir haben die Software auf PCs unter Windows XP mit SP2 mit einer P4- sowie unter Windows 2000 mit einer P3-CPU installiert.

Die auf der Kit-CD befindlichen Dateien finden sich unter der „Tools“-Seite. Wenn man die CD einlegt und startet, zeigt sich ein Menü. Man klickt dann auf den Link: Tools required to develop for the EIR.
Wenn sich dieses Menü nicht von selbst zeigt, dann genügt ein Doppelklick auf die Datei index.html im Wurzelverzeichnis der CD.

In der Abteilung für Windows kopiert man folgende Programme:

- Install AT91-ISP v1.10.exe
- yagarto-bu-2.18_gcc(...).exe

Die anderen Dateien werden für diesen Beitrag nicht benötigt.

Mit AVR91-ISP kann man den Mikrocontroller auf der EIR-Platine programmieren. Es handelt sich um den Typ ARM7TDMI.
Bei Yagarto handelt es sich um eine Entwicklungsumgebung mit gcc, dem C-Cross-Compiler für ARM, und Eclipse, der eigentlichen Programmierumgebung.
Zur Installation: Wir schlagen vor, diese Anwendungen in ein spezielles Projekt-Verzeichnis zu installieren. Der Pfad d:\EIR\Software wurde bei unseren Rechnern genutzt.

Nut/OS-Firmware

Nun geht es an das Auspacken der Sourcen zur Firmware.

Die notwendigen Dateien befinden sich ebenfalls auf der CD. Man klickt hierzu auf den Link firmware oben links im Menü.
Zuerst muss Nut/OS installiert werden. Man kopiert den Source-Code, indem man auf den Link On your Windows PC im Abschnitt development klickt und dann auf ethernut-4.5.2.exe auf der nächsten Seite. Dann wird das Programm gestartet. In unserem Fall landet es in d:\EIR\ethernut-4.5.2.
Zum Schluss der Installation wird noch die Option Start Nut/OS Configurator überprüft. Und wenn man schon mal hier ist, setzt man gleich die Parameter.

Der Konfigurator führt zur Auswahl einer Hardware-Descriptor-Datei. Hier wählt man die für das EIR passende Datei eir10b.conf aus.
Nun wird die Seite mit den Settings mit dem Edit-Menü via Edit>Settings oder den Tasten [Ctrl+T] ausgewählt. Unter dem Tab Build wird der Pfad für die Sourcen von Nut/OS (source directory) ausgewählt — d:\EIR\ethernut-4.5.2\nut in unserem Fall.
Nun werden bei arm-gcc die Pfade für Compilation und die Library-Installationsverzeichnisse angegeben: Build directory und Install directory. Das Installationsverzeichnis bekommt den Namen lib und muss innerhalb des Compilationsverzeichnisses angelegt werden, das selbst wieder ein Unterverzeichnis von Nut/OS sein muss. Also wählten wir d:\EIR\ethernut-4.5.2\nut\build und darin d:\EIR\ethernut-4.5.2\nut\build\lib (siehe auch den Screenshot in Bild 4).
Unter dem dritten Tab bei Tools müssen zwei durch ein Semikolon getrennte Verzeichnisse angegeben werden. Das erste ist der Ort für die Nut/OS-Tools d:\EIR\ethernut-4.5.2\nut\tools\win32 und das zweite der Ort für die Compilations-Kette für Yagarto - in unserem Fall also d:\EIR\Software\yagarto\bin.
Unter dem vierten und letzten Tab wird der Pfad für das Wurzel-Verzeichnis angegeben, das die Applikationen enthält: d:\EIR\ethernut-4.5.2\application (Achtung: Auf KEINEN Fall das Nut/OS-Source-Verzeichnis app verwenden!). Zum Schluss wird noch der Programmer arm-jom ausgewählt und auf OK geklickt.
Nach diesem Konfigurieren muss man nur noch die Nut/OS-Libraries kompilieren. Dieser Schritt kann einige Minuten in Anspruch nehmen. Man klicke auf Build Nut/OS aus dem Build-Menü. Falls nun Fehler gemeldet werden, sollte man seine Konfiguration genau überprüfen. Wenn aber alles geklappt hat, kann man das Programm verwenden.

Das erste Programm

Als Vorbereitung wird ein Unter-Verzeichnis in application erstellt, um darin das blink genannte Beispiel abzulegen. Nun wird das Archiv 080199-11.zip von der Webseite zum Artikel herunter geladen und in diesem Verzeichnis entpackt.
Sie werden sofort sehen, dass der Source-Code sehr simpel ausfällt und nahezu selbsterklärend ist, wenn man Kapitel 34 (Titel PIO: Parallel Input Output Controller) aus der Atmel-Dokumentation zum AT91SAM7SE512 gelesen hat. Diese Dokumentation kann von [2] herunter geladen werden.
Das PIOx_PER-Register erlaubt die Aktivierung der Pins von Port x eines PIO-Controllers und blockiert die Funktion interner Peripherie, die damit in Verbindung steht. In unserem Fall wird Pin PA1 als „generic input/output“ definiert.
Das PIOx_OER-Register erlaubt die Konfiguration der

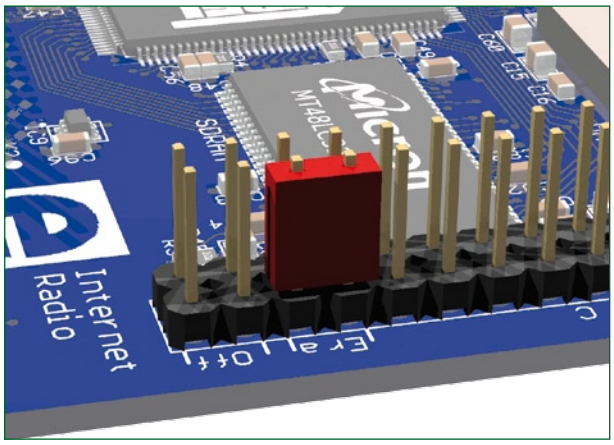

Bild 2.
Der Jumper zum Löschen des internen Speichers des Mikrocontrollers verbindet die Pins 34 und 36 von K3.

Pins von Port x als Ausgang. Pin PA1 wird als Ausgang geschaltet.
Nun kommt eine Endlos-Schleife, in der die LED abwechselnd ein- und wieder ausgeschaltet wird. Hierzu werden zwei weitere Register verwendet.
Mit PIOx_CODR kann man die Pins von Port x auf logisch „0“ schalten. Dann wird Strom durch die LED fließen und sie wird leuchten.
PIOx_SODR schließlich erlaubt umgekehrt die Umschaltung der Pins von Port x auf logisch „1“, was den Stromfluss durch die LED stoppt und das Leuchten beendet.
NutSleep ist ein von den Nut/OS-Libraries bereitgestellter Timer, der mit Millisekunden operiert. Er ist in der Datei sys/timer.h definiert.

Mit Hilfe des Kommandozeilen-Interpreters von Windows wird die Kompilierung des Source-Codes gestartet. Zuvor

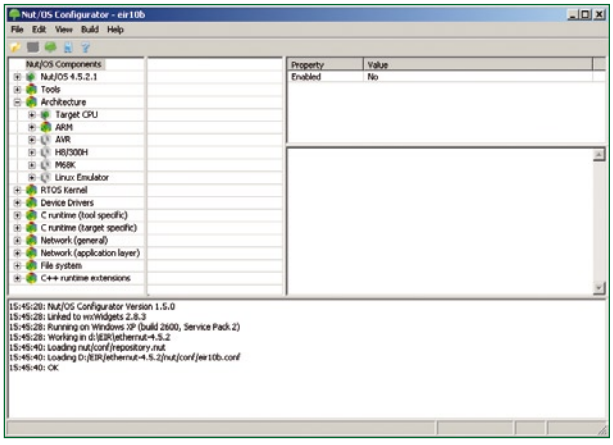

Bild 3.
Die fertige Konfiguration der EIR-Hardware. Hier ist Sorgfalt gefragt.

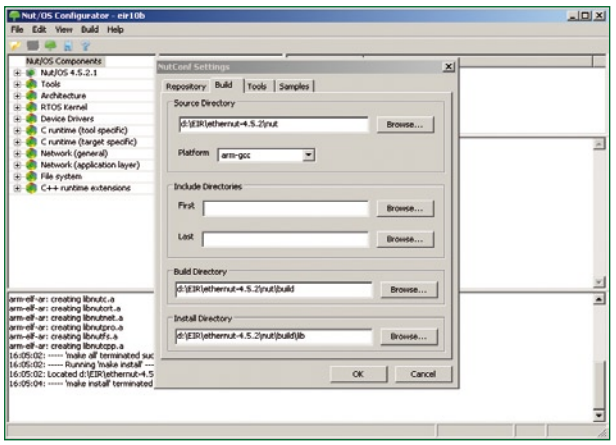

Bild 4.
Bei allen Konfigurationen müssen die absoluten Verzeichnis-Pfade angegeben werden.

sollte man allerdings die Box über die Konfiguration von PATH gelesen haben.
Für die Kompilierung führt man cmd.exe aus und wechselt in das Applikationsverzeichnis: cd d:\EIR\ethernut-4.5.2\application\blink.
Jetzt braucht man nur noch make clean einzugeben, um Reste früherer Compiler-Läufe zu beseitigen. Dann kommt der Befehl make, um die Binär-Datei zu erzeugen. Wenn alles gut ging, taucht am Ende die Datei blink.bin auf.

Ein bisschen Hardware

Bei der Hardware handelt es sich nur um einen Widerstand und eine LED. Mit Hilfe einer Buchsenleiste (einreihig und 20-polig genügt) schließt man diese Bauteile so wie in Bild 1 angegeben an das EIR-Board an.

Binäre Programmierung des Mikrocontrollers

Nun kommt das Tool SAM-BA zur Programmierung des Mikrocontrollers zum Einsatz. Hierzu wird das EIR-Board mit einem USB-Kabel an den PC angeschlossen. Als erste Schritte empfehlen sich das Löschen des Flash-Speichers und ein Neustart. Der Controller wechselt in den Programmier-Modus, indem er den passenden Code aus seinem ROM lädt. Hierzu werden die beiden Pins 34 und 36 von K3 (direkt neben der Beschriftung Era, siehe Bild 2) durch einen Jumper miteinander verbunden und dann der Reset-Taster betätigt. Anschließend wird der Jumper wieder entfernt.
Windows findet dann neue Hardware und will diese Peripherie auch installieren. Diese Installation müsste nun automatisch erfolgen, wenn zuvor AVR91-ISP korrekt installiert wurde.

Jetzt wird SAM-BA gestartet und \usb\ARM0 als Verbindung (connection) sowie der Chip AT91SAM7SE512-EK als Ziel ausgewählt. Dann auf Connect klicken.
Wenn nun der Tab Flash angezeigt wird, kann man die zu übertragende Datei via Open bei der Box Send File Name auswählen. Hier muss natürlich blink.bin stehen. Nun ein Klick auf Send File. Nach diesem Vorgang möchte die Software wissen, ob man bestimmte Speicher-Bereiche blockieren möchte. Die Antwort lautet No. Man kann nun überprüfen, ob alles wie gewünscht geklappt hat. Hierzu klickt man auf Compare send file with memory.

Wichtig: Es muss noch das Script Boot from Flash (GPNVM2) gestartet werden, indem man es aus der Liste auswählt und auf Execute klickt. Anschließend kann das Programm geschlossen werden. Nun ein Druck auf die Reset-Taste des EIR und die LED sollte blinken.

Ein zweites Beispiel

Es gibt auch noch das Archiv 080199-12.zip (auf www.elektor.com/EIR), das einen anderen Weg geht, um eine LED blinken zu lassen: Es nutzt die PWM-Fähigkeiten (Pulsbreitenmodulation) des Controllers.
Dieses Archiv enthält gleich zwei Verzeichnisse. Die auf der CD enthaltene Version von Nut/OS hat nämlich keine eingebaute Unterstützung für PWM beim AT91SAM7SE. Man muss also den Source-Code entsprechend updaten, indem man die beiden im Archiv enthaltenen Dateien at91_pwmc.h und at91sam7se.h nach d:\EIR\ethernut-4.5.2\nut\include\arch\arm kopiert. Die zweite Datei existiert bereits und muss überschrieben werden.
Nun wird das Unterverzeichnis pulse im Verzeichnis application erstellt und der Source-Code entpackt. Die Vorgehensweise bei der Kompilierung und Programmierung ist identisch mit dem ersten Beispiel und muss nicht nochmals beschrieben werden.
Wie schon erwähnt, wird dieses Mal das PWM-Modul des Controllers genutzt. Siehe hierzu die Kapitel 34 & 37 (Pulse Width Modulation Controller) der Dokumentation.

Zurück zur Radio-Firmware

Im Verzeichnis firmware der CD findet sich das Archiv webradio-1.2.1.zip. Man muss es nur im Applikationsverzeichnis entpacken, kompilieren, mit der resultierenden Binär-Datei webradio.bin das EIR-Board programmieren und schon kann man wieder Musik hören. Die Version 1.2.0 darf man hingegen nicht verwenden, denn diese wird nicht fehlerfrei kompiliert.

Viel Spaß beim Entwickeln! Und wenn Sie eine interessante Anwendung geschrieben haben: Zögern Sie nicht, uns zu informieren – mit uns sind sicher auch etliche Elektor-Leser sehr gespannt!

Source-Code und Anwendungen

Alle hier gemachten Angaben beziehen sich auf die Source-Code-Dateien und Programme, die auf der Kit-CD enthalten sind.

Unter Windows sollte man sicher gehen, dass die mit dem Kommando-Interpreter gestarteten Tools tatsächlich die von Yagarto sind. Um das Risiko der Konfusion durch die Installation multipler Compilations-Ketten zu minimieren, sollte man immer absolute Pfade für die aufgerufenen .exe-Dateien während der Konfiguration angeben und diese in Anführungszeichen einschließen. Beispiel: „c:\Programme\yagarto\bin“.

Setzen von PATH

Wir empfehlen die Erstellung einer Batch-Datei, die die Update-Befehle für die Umgebungsvariable PATH enthält, sodass Windows hierüber sowohl Nut/OS als auch die Yagarto-Tools finden kann.
Dieses Script sollte man in der Standard-Pfad-Liste von PATH speichern und beispielsweise seteirenv.bat nennen.

In unserem Beispiel enthält sie die Zeile: set PATH=d:\EIR\ethernut-4.5.2\nut\tools\win32;d:\EIR\Software\yagarto\bin;%PFAD%.
Sie sollte einmal gestartet werden, bevor man Nut/OS oder die Yagarto-Tools verwendet.

Ausstattung

- EIR-Kit
- USB-Kabel Typ A/B
- Stabilisiertes 12-V-Netzteil
- 1 rote LED
- 1 Widerstand 180 Ω
- 1 Buchsenleiste, 20-polig

(080199-I)

Links & Literatur

[1] www.ethernut.de/en/hardware/eir
[2] www.atmel.com/