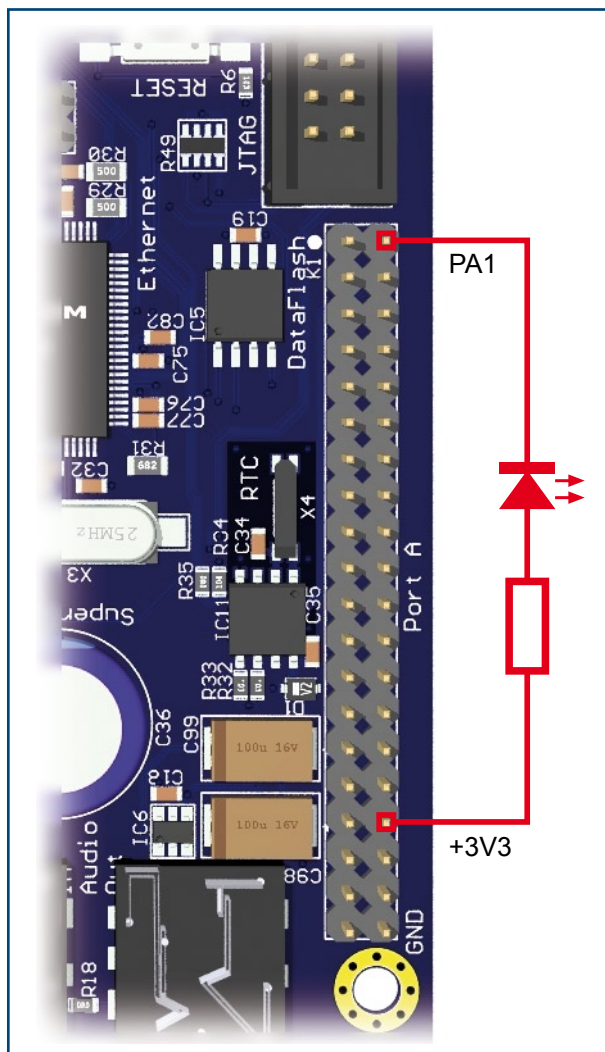


# Other uses for EIR

## Much more than just a radio: a powerful development card for ARM7

Antoine Authier & Harald Kipp

**This article explains how to use the Elektor Internet Radio board for developing your own extensions and realizing your own projects around this amazing piece of condensed technology.**



**Figure 1.**  
Connect the LED cathode to  
pin 2 (PA1) and the resistor  
to pin 34 of connector K1.

In the first part, we're going to describe the tools needed and how to use them; in the second, we'll be developing a concrete example around an LED.

### Development environment

Let's start by installing the working environment. In this article, we've chosen to present the development tools available for Microsoft's Windows operating system.

Installation under Linux is possible (see the procedure described on the CD-ROM). We used Windows XP SP2 on a Pentium 4 and Windows 2000 on a Pentium 3.

Let's start by recovering the files required from the CD-ROM supplied with your kit: they are available on the 'Tools' page. When you insert the CD-ROM into your computer, a menu is displayed in your browser; click on the link named: *Tools required to develop for the EIR*.

(If this menu fails to display automatically, double click on the [index.html](#) file in the CD-ROM root directory)

Then in the Windows section, copy the following programs:

- [Install AT91-ISP v1.10.exe](#)
- [yagarto-bu-2.18\\_gcc \(...\) .exe](#)

The others are not required for the purposes of this article.

AVR91-ISP lets you program the microprocessor at the heart of the EIR's ARM7TDMI.

Yagarto is a software suite that includes `gcc`, the C cross-compiler for ARM, and Eclipse, the programming environment.

Now let's install these applications. We've chosen to install them in a special directory for this project `d:\EIR\software` in order to illustrate our example clearly.

## Nut/O.S. firmware

Now we're going to deploy the sources of the firmware. The files needed are also on the CD-ROM. Click on the *firmware* link, from the top left menu.

Nut/O.S. has to be installed first. Copy the sources by clicking the link *On your Windows PC* in the *development* section; then *ethernut-4.5.2.exe* on the next page. Run this program. We've chosen to install it in `d:\EIR\ethernut-4.5.2`.

At the end of installation, check the option *Start Nut/OS Configurator* — while we're about it, we'll set these parameters.

The configurator asks us to choose a hardware descriptor file; select the one corresponding to the EIR, called `eir10b.conf`.

Then go into the Settings page from the Edit menu *Edit>Settings* or press [Ctrl+T]. Under the tab *Build*, enter the pathname for the Nut/O.S. sources (*Source Directory*) — `d:\EIR\ethernut-4.5.2\nut` in our example.

Select the *arm-gcc* platform then configure the paths for the compilation and library installation directories, *Build directory* and *Install directory*. The install directory must be called *lib* and must be within the compilation directory, which itself must be a sub-directory of Nut/O.S. So we've chosen `d:\EIR\ethernut-4.5.2\nut\build` and `d:\EIR\ethernut-4.5.2\nut\build\lib` (see screenshot in Figure 4).

Under the third *Tools* tab you'll need to enter two paths separated by a colon. The first points to the Nut/O.S. tools `d:\EIR\ethernut-4.5.2\nut\tools\win32` and the second to those of the compilation chain installed with Yagarto, i.e. in our case `d:\EIR\software\yagarto\bin`.

Under the fourth and last tab, enter the root path for the directory that is going to contain your applications; here we've chosen `d:\EIR\ethernut-4.5.2\application` (**attention**: whatever you do, don't choose the Nut/O.S. sources' sub-directory *app*). Lastly, choose the *arm-jom* programmer, and then click OK.

Once the configuration is finished, you need to compile the Nut/O.S. libraries. This step may take a few minutes; click on *Build Nut/OS* from the *Build* menu. In the event of an error, re-check your configuration. If everything has worked properly, you can now close this program.

## Our first application

Now create a sub-directory in *application* to develop our example — let's call this *blink*. Download the **080199-11.zip** archive from the article page, and unzip it to this location.

You'll note that the source code is simple (almost self-explanatory, if you've read Chapter 34, *PIO: Parallel Input Output Controller*, of the AT91SAM7SE512 documentation, downloadable from the ATMEL website [2]).

The `PIOx_PER` register allows us to enable each pin of port *x* of a PIO controller and disables the function of any internal peripheral that might be associated with it. As you can see here, we are configuring pin PA1 as a generic input/output.

The `PIOx_OER` register allows us to configure each pin of port *x* as an output. Here we're configuring pin PA1 as an output.

Then comes the infinite loop where the LED is alternately lit and extinguished. This action is provided by two further registers.

First off, `PIOx_CODR` which allows us to force each pin of

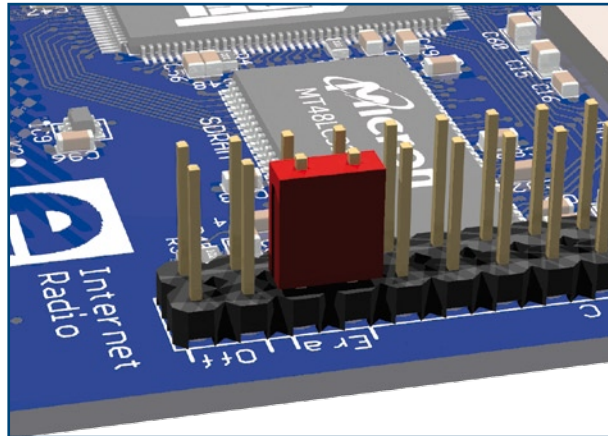


Figure 2. The jumper for erasing the microprocessor's internal memory is fitted between pins 34 and 36 of connector K3.

port *x* to logic 0. In this situation, there is a potential drop across the LED terminals, and it lights up.

And then `PIOx_SODR` which allows us to force each pin of port *x* to logic 1; in this situation there is no longer any potential drop across the LED terminals, so it goes out.

`NutSleep` is a timer provided by the Nut/O.S. libraries, which needs to be provided with a duration in milliseconds, defined in the file `sys/timer.h`.

Now we're going to work with the Windows command interpreter in order to compile this source code. Before starting, read the box about `PATH` configuration.

So open a command interpreter `cmd.exe` and go to your application directory — `cd d:\EIR\ethernut-4.5.2\ap-`

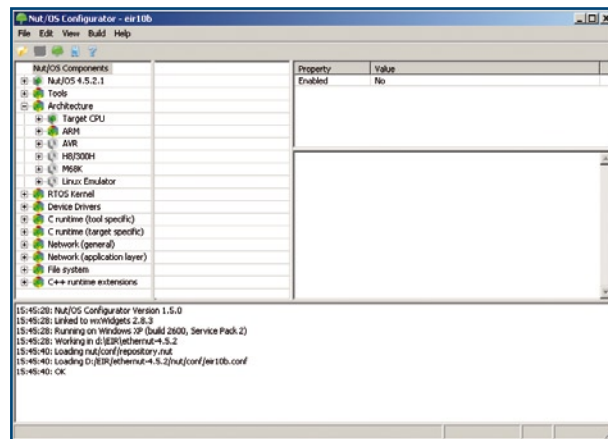


Figure 3. Have fun with navigating the EIR hardware parameters. Do not change anything!

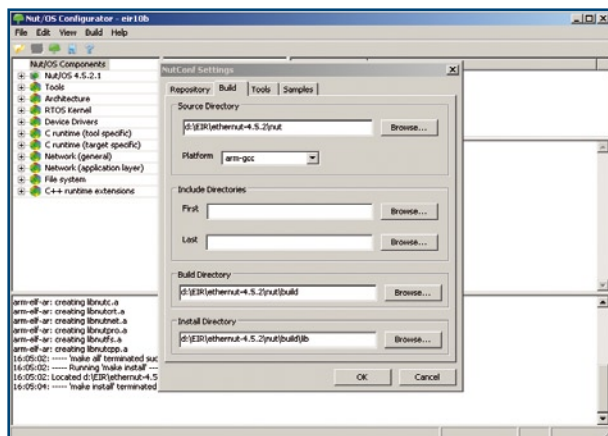


Figure 4. For all configuration operations it is essential to fetch the absolute paths of the subdirectories.

`plication\blink` here.

Then all you have to do is type `make clean` to purge previous compilations, then `make` to build the binary file. If everything has gone OK, you'll see the file `blink.bin` appear.

### A bit of hardware

With the aid of a female connector, connect the resistor and LED to Port A as per the circuit in **Figure 1**.

### Binary programming on the microprocessor

Now we're going to use the `SAM-BA` tool to program the microprocessor.

Connect the USB cable between the EIR and your computer. First of all, you need to erase the microprocessor's Flash memory and restart it so it will go into USB programming mode by loading the appropriate program from its ROM. To do this, connect the jumper provided between pins 34 and 36 of connector K3 (alongside the legend **Era**, see **Figure 2**); then press the `reset` button. Remove the jumper.

Windows ought then to find and install a new hardware peripheral. Installation ought to be automatic, if `AVR91-ISP`

the microprocessor's PWM peripheral.

You'll note that the archive contains two directories in its root. The version of Nut/O.S. supplied on the CD-ROM does not contain the description of the PWM peripheral for the AT91SAM7SE. So you need to update your source tree and extract the files `at91_pwmc.h` and `at91sam7se.h` to `d:\EIR\ethernut-4.5.2\nut\include\arch\arm` (the second file already exists, you'll need to overwrite it). Then create a sub-directory `pulse` in your `application` subdirectory and unzip the corresponding source code. The procedure for compiling and programming remains the same, see above.

The source code describes a simple application for the microprocessor's PWM module; refer to Chapters 34 & 37 (*Pulse Width Modulation Controller*) of the microprocessor documentation.

### Getting back to the basic firmware: the radio

In the directory `firmware` of the CD-ROM you'll find the archive `webradio-1.2.1.zip`. Then all you need do in order to listen to the radio is unzip this into your application folder, compile it, and program your EIR with the `webradio.bin` binary generated.

Don't use version 1.2.0, it doesn't compile.

### Source code and applications

The information published here is based on the use of the source files and applications contained on the kit CD-ROM.

Under Windows, take care that the compiler and tools executed by the command interpreter are indeed the Yagarto ones. To reduce the risk of confusion following the installation of multiple compilation chains, always make sure you specify absolute access paths for the tool executables, and during the configuration stages, enclose them in quotes — e.g., "c:\program files\yagarto\bin".

### Setting the PATH

We recommend you create a batch file to contain the update command for your (environment variable) `PATH` so that Windows will go there to find the Nut/O.S. and Yagarto tools.

You need to store this script in your `PATH` default paths list; call it for example `seteirenv.bat`.

In our example, it will contain the line `set PATH=d:\EIR\ethernut-4.5.2\nut\tools\win32;d:\EIR\software\yagarto\bin;%PATH%`.

Run this before invoking the Nut/O.S. or Yagarto tools.

has been installed correctly.

Run the application `SAM-BA` and select `\usb\ARM0` as the means of connection and AT91SAM7SE512-EK as target, then click on *Connect*.

Check that the tab *Flash* is displayed and select the file for programming by clicking on the *Open* icon on the *Send File Name* line. Find and select your `blink.bin` binary. Click on *Send File*. Once this operation is over, the software asks you if you want to block some regions of memory; answer *No*. You can also re-check that everything has gone OK by clicking on *Compare send file with memory*.

**Important:** Now run the script *Boot from Flash (GPNVM2)* by selecting it from the list and clicking *Execute*. You can close the application. Then press the `reset` button on the EIR: the LED should flash.

### A second example

We also provide you with the archive `080199-12.zip` (downloadable free from [www.elektor.com/EIR](http://www.elektor.com/EIR)) with another example that let's us make the LED pulse, thanks to

Enjoy your development. And if you have produced an interesting application, don't hesitate to tell us all about it!

### Equipment required

- EIR kit
- USB cable type A male – type B male
- stabilized 12 V power supply
- red LED
- 180 Ω resistor
- 2×20 pin female socket strip for headers.

(080199-1)

### Bibliography and web links

- [1] [www.ethernut.de/en/hardware/eir](http://www.ethernut.de/en/hardware/eir)
- [2] [www.atmel.com/](http://www.atmel.com/)