Elektor Internet Rad

Listening to radio programmes with the

Harald Kipp and Dr Thomas Scherer

In the good old days, you had to modulate audio signals onto an RF carrier so they could be received and demodulated to produce something more or less audible. Nowadays things are different: audio signals are compressed and put into IP packets that are 'streamed', and you can access every Internet radio programme in the world by receiving, buffering and decompressing these packages. This is all very easy with the state-of-the-art hardware described in this article.

Internet radio is something quite special: even the most sensitive shortwave receiver cannot come close to providing such a broad range of programmes, and the sound quality is simply not comparable. As the 'Internet broadcasters' that provide these programmes do not have to pump several hundred kilowatts of RF energy in the air (with the resulting 'electrosmog'), this type of broadcasting operation is also quite economical for relatively small target audiences.
We could say a lot more about the advantages of this new sort of radio (see inset), but what's more important is to answer the question posed in the next section.

Why not use a pure software solution?

First of all, we have to say that there are several programs (WinAmp, iTunes, VLC, etc.) that are available entirely free of charge for all possible operating systems and can be used to listen to Internet radio. Every true 21st-century person has a PC, Mac or Linux machine available somewhere, so why should you spend money on a non-virtual, physical device, and on top of that build it yourself?
Well, for one thing the hardware platform for a software radio consumes electricity, and quite a lot for this purpose. Anyone who spends a good deal of time listening to radio programmes with a PC is engaged in a very environmentally unfriendly activity. The solution proposed here manages to do the job with a power consumption of only 1 watt. If you use it 10 hours a day, the savings in electricity costs alone (relative to using a gamer PC as a radio) are enough to repay you investment within six months.

For another thing, there are applications for which a PC is not such a clever solution, such as connection to a stereo system. A DIY Internet radio based on Open Source technology is easy to extend and adapt to special requirements – and last but not least, the EIR keeps on working when your PC hangs or crashes .

Operating principle

As the EIR is a complex project that uses state-of-the-art hardware, it is impossible to deal adequately with all relevant topics in a single article. For this reason, the main objective of this article is to describe the hardware and tell you how to assemble it and put it into service. You can find additional information in documents on the Elektor website (www.elektor.com) and the project website [1], and there will be additional articles on this subject in future issues.

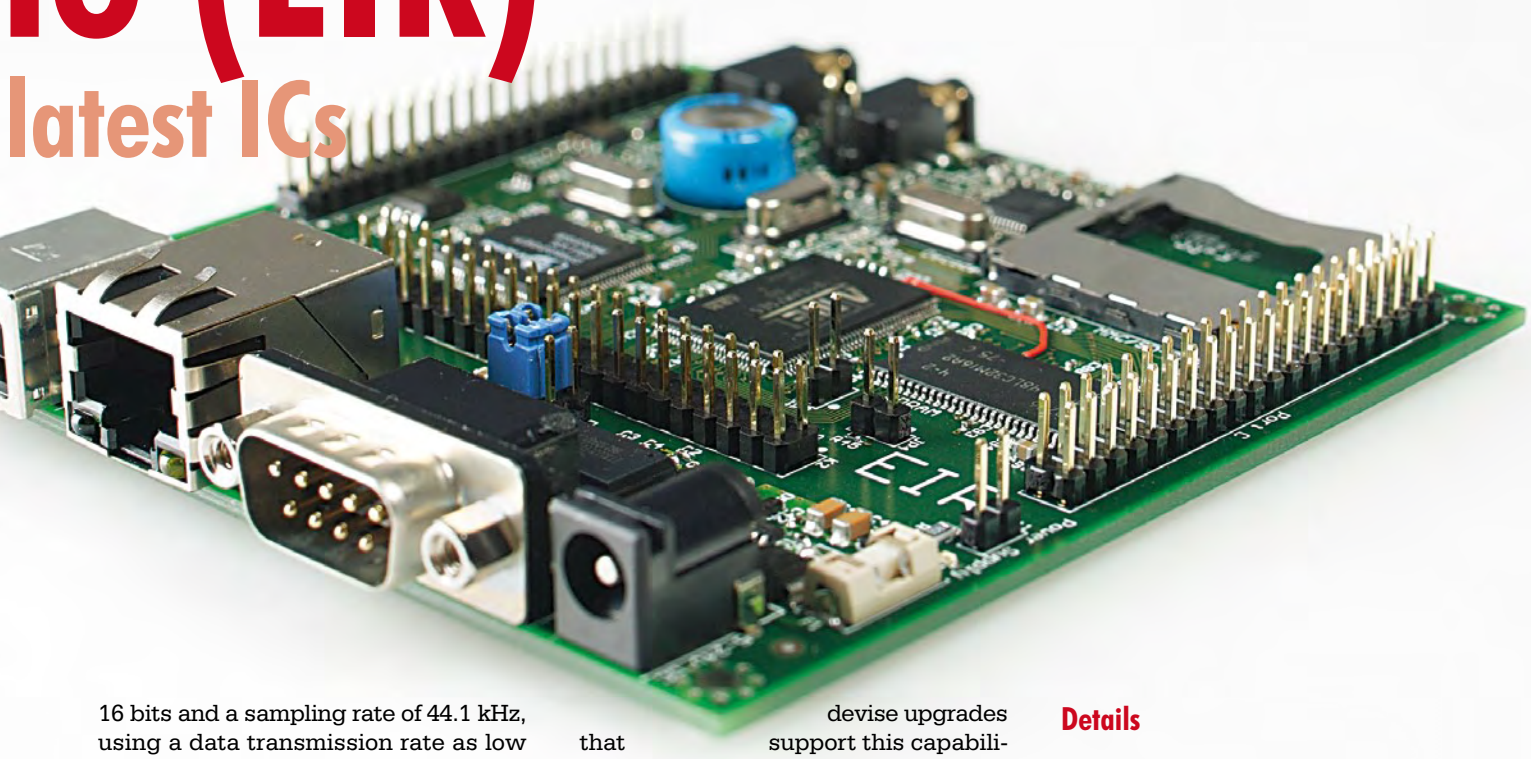
As you probably already realise, an Internet radio must receive, buffer and decode data streams. This means that one of its basic ingredients must be a reasonable microcontroller. As already mentioned in the last issue of Elektor [3], an ARM7 MCU [4] can provide the necessary processing power.
The basic architecture is shown in Figure 1. The MCU is shown in the upper middle of the diagram, and it has access to a healthy 64 MB of SDRAM – which is sufficient for the buffer and quite a few 'extras'. The MCU has room for the firmware, and there is also 4 MB of flash memory available for non-volatile data storage. A real-time clock backed up by a Supercap allows the circuit to be used to implement an alarm radio or other applications that depend on knowing the current time. To avoid having to exploit the full capacity of the ARM7 MCU, audio decoding is handled by a VS1053 IC [5], which is specifically designed for this purpose.
The EIR also provides a comfortable selection of interfaces: beside the mandatory Ethernet port (since the EIR has to access the Internet somehow), there is a USB programming interface for downloading new firmware, a serial port and a JTAG port (useful for debugging), and three useful expansion connectors at the port level.
To allow you to record broadcasts if you so desire, there is also a slot for an MMC/SD memory card.

General aspects

The incoming data streams are normally compressed to such an extent that they can handle sampled stereo data, which typically has a resolution of

io (EIR)

latest ICs



16 bits and a sampling rate of 44.1 kHz, using a data transmission rate as low as 192 kbit/s (or even less) instead of the normal rate of around 1.4 Mbit/s. This means that a buffer with a capacity of approximately 10 seconds can be implemented with around 256 KB of RAM. This may not sound like much nowadays, but it is still quite a hefty chunk of memory for a microcontroller. If you want to be on the safe side and also want to have room for Internet niceties and other 'extras', you can quickly end up with 512 kB or more. The ARM7 MCU selected for this design supports SDRAM, so the EIR with its 64 MB of RAM does not suffer from any shortage of memory.

We chose Nut/OS as the operating system. It is quite accommodating in comparison with Linux and can manage with less than 40 kB of memory. All in all, the software needs around 200 kB of memory. A capacity of 1 MB is ample for data storage. As the MCU already has 512 kB of flash memory on board for program data as well as an abundance of RAM, there are no bottlenecks. All of the software is Open Source, except for the flash programming software from Atmel.

Incidentally, the microcontroller has sufficient processing power to allow a second audio stream to be recorded on the SD card while another stream is playing. It will certainly not take very long for the Open Source community to devise upgrades that support this capability and other conceivable features.

In order to avoid constraining the form of any possible extensions, no user interface components (such as buttons or a display) are incorporated in the board. However, they can easily be connected via the expansion connectors. The EIR is intended to form the basis for user-designed expansions, and the firmware that comes with the board is thus designed to be used via an integrated website. However, the firmware is completely open, so other options are always possible.

Details

As you can see immediately from looking at the schematic diagram in **Figure 2**, this is a complex design. For this reason, the following description is based on the functional blocks.

- **Ethernet**

Access to the Internet is via an Ethernet connector with an integrated transformer and two LEDs. The green LED lights up when data is being transferred, while the yellow LED indicates that a connection is present. The Ethernet traffic is handled by a specialised IC (IC10, a DM9000E). Buffer IC9

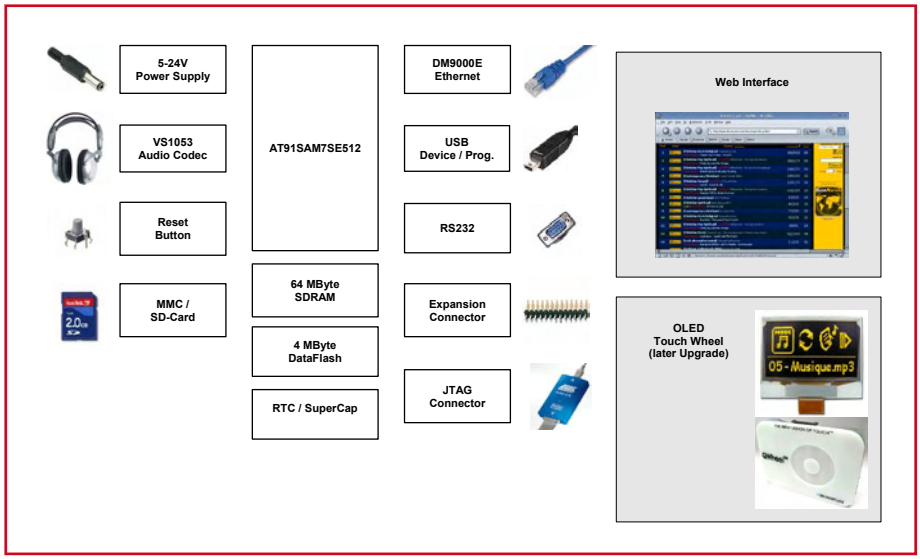


Figure 1. Block diagram of the Elektor Internet Radio.

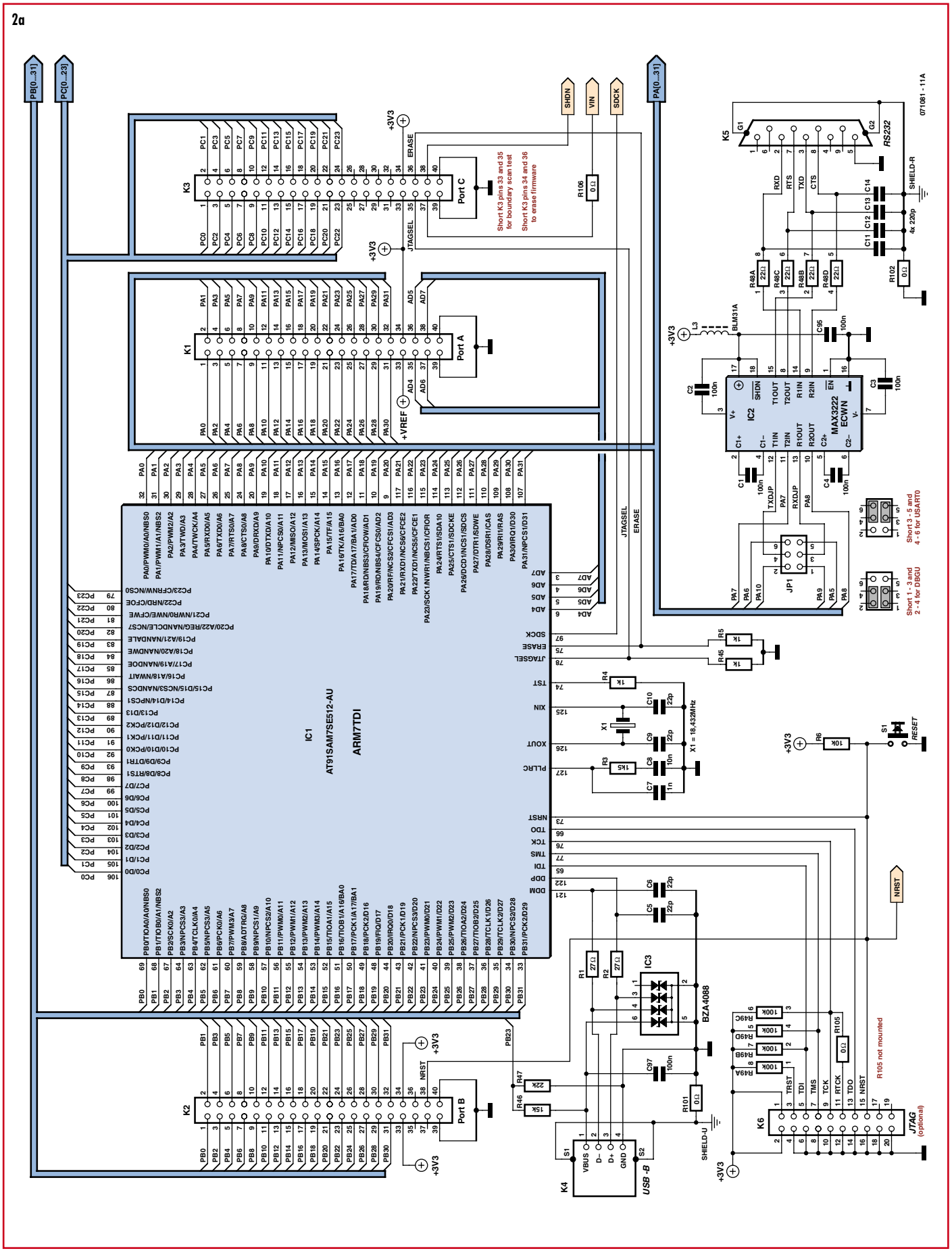
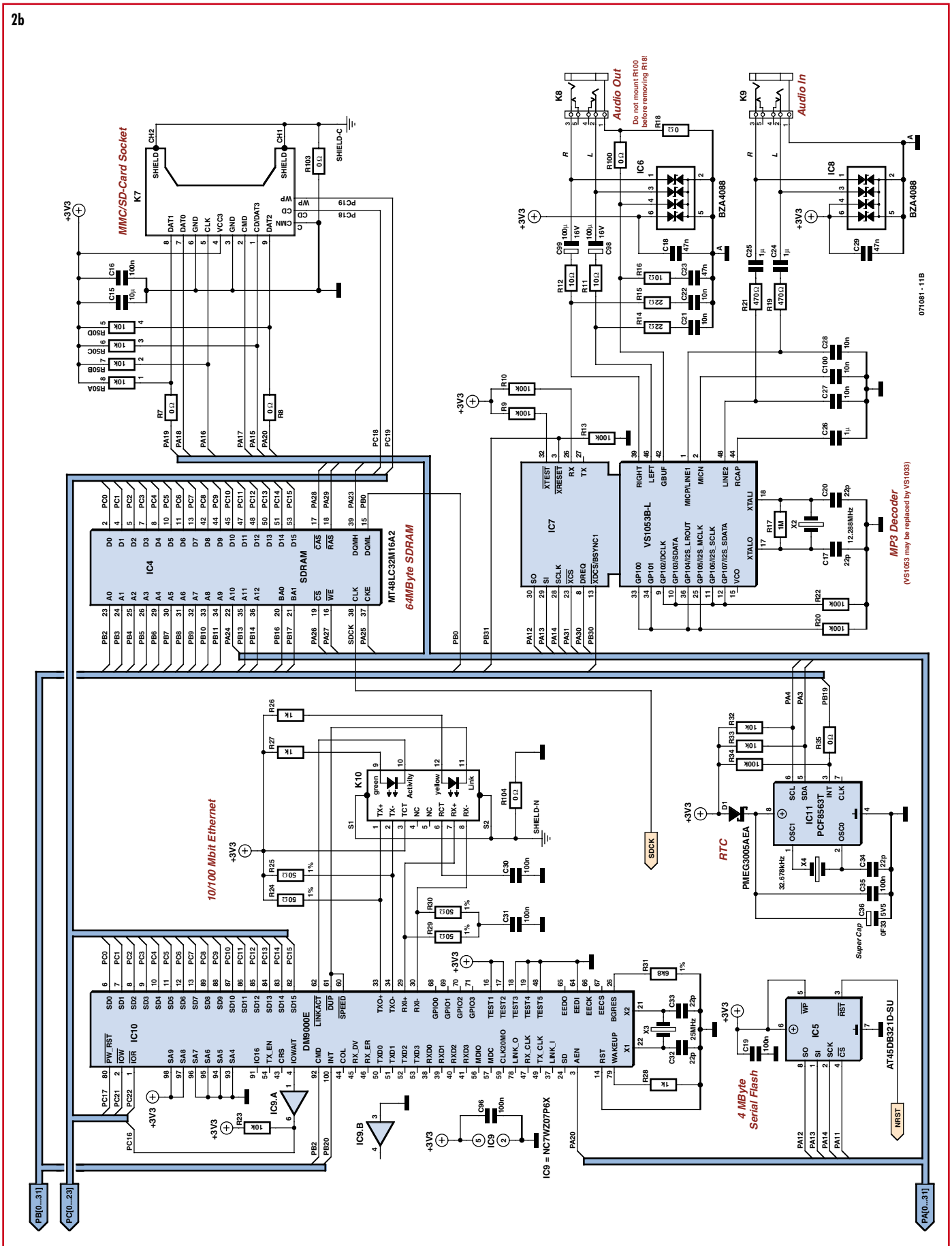
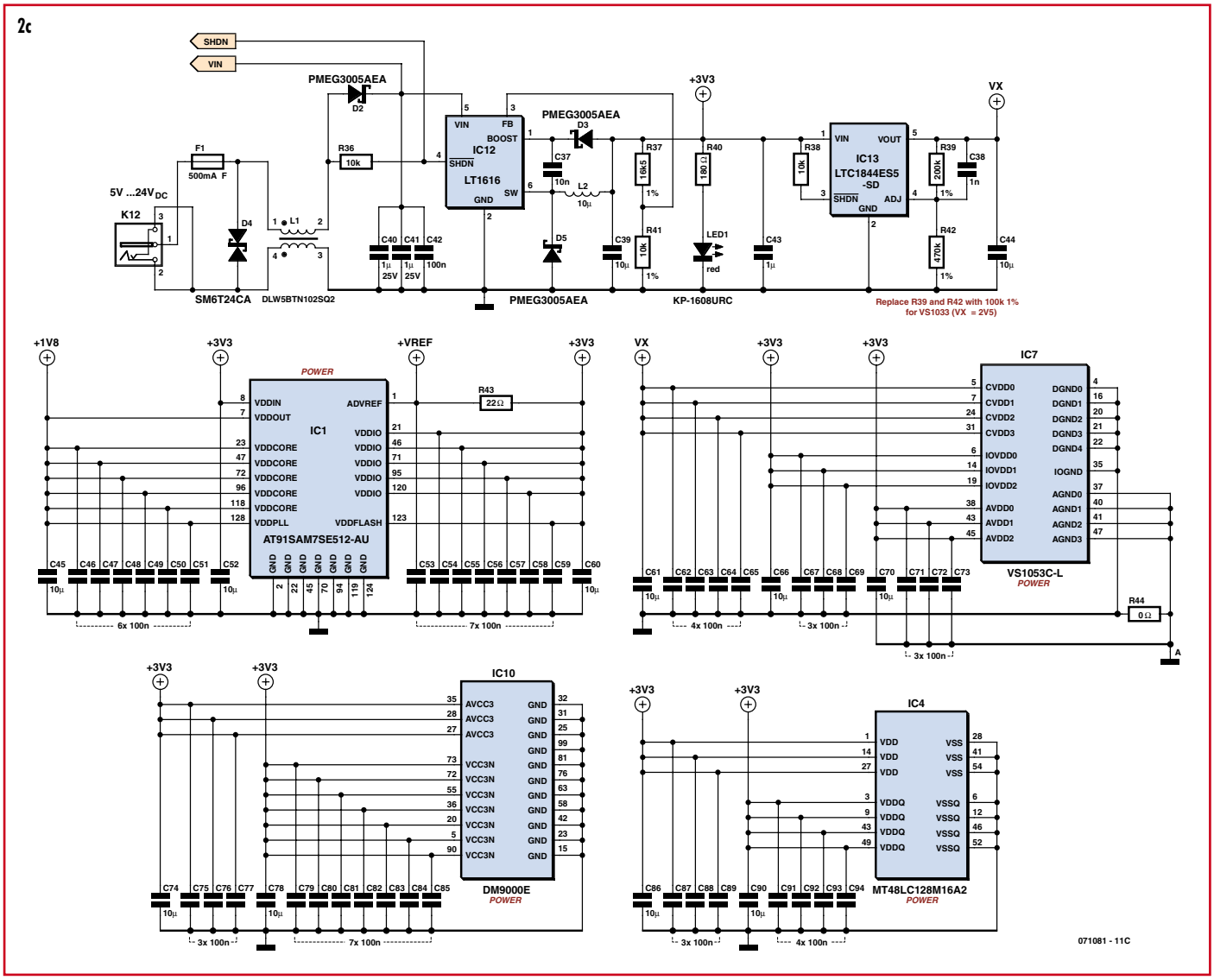


Figure 2. As you can see immediately from the schematic diagram of the EIR, this is a sophisticated project.

2b


MMC/SD-Card Socket
K7
Audio Out
Do not mount R100 before removing R18!
Audio In
K8
K9
IC6
IC8
BZA4088
IC4
SDRAM
MT48LC32M16A2
64MByte SDRAM
IC7
VS1053B-L
MP3 Decoder
(VS1053 may be replaced by VS1033)
10/100 Mbit Ethernet
K10
DM9000E
IC10
RTC
PMEG3005AEA
IC11
PCF8563T
4 MByte
Serial Flash
AT45DB321D-SU
IC5
IC9.A
IC9.B
IC9 = NC7WZ07P6X
Super Cap
12.288MHz
32.678kHz
25MHz
+3V3
PB[0...31]
PC[0...23]
PA[0...31]
071081 - 11B



allows the Wait input of the MCU to be used by expansion circuitry.

• Audio decoder

Although the ARM7 would just be able to handle decoding of MP3 or AAC data in software, a dedicated IC such as IC7 considerably reduces the load on the MCU, and it is specifically designed to handle HE-AAC and even Ogg-Vorbis data in addition to the usual MP3 versions. Logically enough, this also simplifies the application software. We used a pre-production sample from VLSI in the prototypes. If you have trouble obtaining this IC, you can use the VS1033 version (without Ogg Vorbis capability) instead. Although the MCU has a 1.8-V output for powering peripheral devices, a separate 1.8-V voltage regulator is provided for IC7 for reasons of stability. If you use the VS1033, R39 and R42 must be changed to 100 kΩ because it needs a 2.5-V supply voltage.

• Supplementary flash memory

It is necessary to store a large number of operational settings for the radio, and they must remain available even after a power outage (especially the station list). The internal flash memory of the MCU could be used for this purpose, but writing data to it is cumbersome. To simplify the process, a serial flash memory (IC5) is included in the design. With a capacity of 4 MB, it has room for extensive station lists and even more.

• Power supply

To keep the power consumption of the EIR as low as possible, it is powered by a switch-mode supply built around IC12. It provides around 5 watts of power at 3.3 V with an input voltage in the range of 5–24 V. As the EIR takes only 1 W for its own use, there is enough left over for DIY hardware extensions.

• Soldering

The EIR is built on a densely populated multilayer PCB with lots of tiny SMD components (see **Figures 3** and **4**), with some of the IC pins spaced only 0.5 mm apart. To avoid problems with DIY assembly, Elektor can supply a pre-assembled board with all SMDs already fitted (with the VS1053). All you have to do is to fit the relatively large components, which helps you avoid pernicious assembly errors. For the diehards, it is of course possible to build the board yourself using the layout artwork.

Functional test

For initial testing of the power supply, the load on the 3.3-V side should be at least several millampères (do not use it with no load). The voltage regulator should start working with at input voltage of at least 4 V and draw 50 to

150 mA, depending on the load. This will drop to 30–50 mA at 24 V. If everything is OK, LED1 will light up. After the ICs are fitted, you can use an oscilloscope to check that the crystal is working. If X1 is oscillating, the MCU should be ready for action.

The MCU comes with an on-board boot loader that supports communication from and to the RAM and with the flash memory as well as downloading new firmware. The AT91-ISP.exe file available from Atmel (see reference [6]) installs the program SAM-BA, which runs under Windows. After installing this program, connect the EIR to your PC via the USB port. After the power is switched on, Windows should select the appropriate driver automatically. Now you can start SAM-BA. Select 'USB' as the connection type and 'AT-91SAM7SE512-EK' as the device (this is largely compatible with the EIR).

You can download a simple testing firmware program from the Elektor website. With this firmware and an operational MCU and serial interface, you can check the other components, such as the Ethernet port and the audio decoder. After downloading the firmware to the MCU, you have to tell the EIR

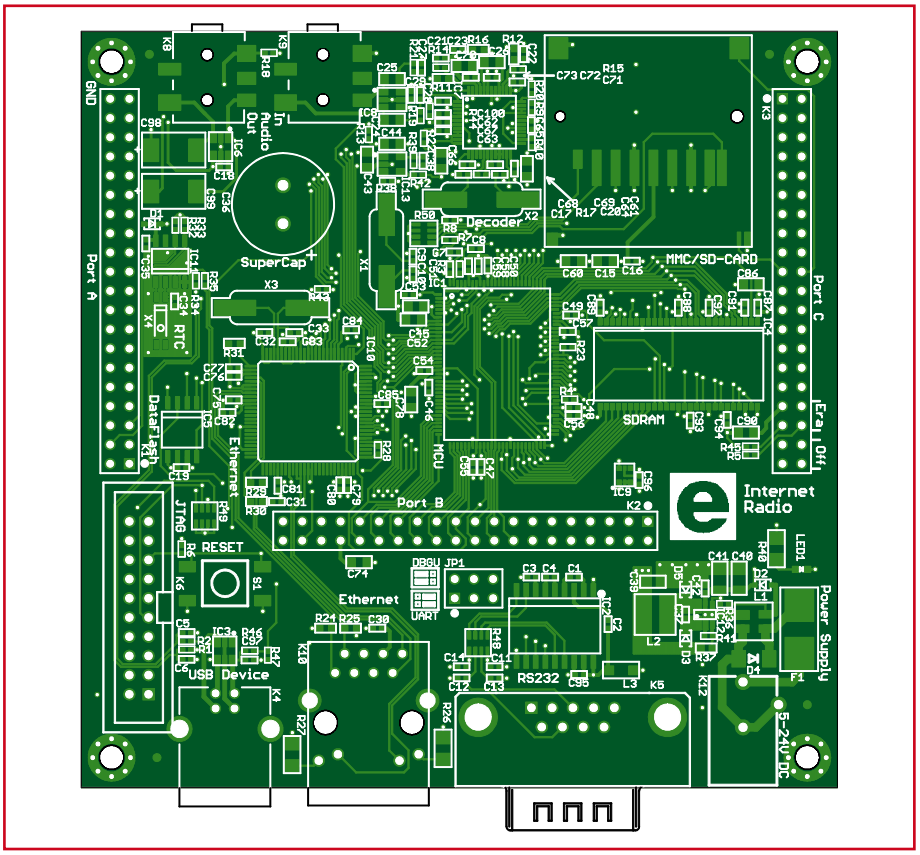


Figure 3. The component layout of the EIR. To avoid assembly problems, a board with prefitted SMD components is available.

COMPONENT LIST

Resistors

R1,R2 = 27Ω, SMD 0402
R3 = 1kΩ5, SMD 0402
R4,R5,R28,R45 = 1kΩ, SMD 0402
R6,R23,R32,R33,R36,R38 = 10kΩ, SMD 0402
R7,R8,R18,R35,R44 = 0Ω, SMD 0402
R9,R10,R13,R20,R22,R34 = 100kΩ, SMD 0402
R11,R12,R16 = 10Ω, SMD 0603
R14,R15,R43 = 22Ω, SMD 0402
R17 = 1MΩ, SMD 0402
R19,R21 = 470Ω, SMD 0402
R24,R25,R29,R30 = 50Ω 1%, SMD 0402
R26,R27 = 1kΩ, SMD 1206
R31 = 6kΩ8 1%, SMD 0603
R37 = 16kΩ5 1%, SMD 0603
R39 = 200kΩ* 1%, SMD 0402
R40 = 180Ω, SMD 1206
R41 = 10kΩ 1%, SMD 0402
R42 = 470kΩ* 1%, SMD 0402
R46 = 15kΩ, SMD 0402
R47 = 22kΩ, SMD 0402
R48 = 22Ω, array, CAY16
R49 = 100kΩ, array, CAY16
R50 = 10kΩ, array, CAY16
R100-R106 = 0Ω*, SMD 1206 (not required)
* see text

Capacitors

(SMD ceramic 6.3V unless otherwise indicated
C1-C4,C16,C19,C30,C31,C35,C42, C46-C51,C53-C59,C62-C65,C67,C68,C69,C71,C72,C73, C75,C76,C77,C79-C85,C87,C88,C89,C91-C97 = 100nF, SMD 0402
C5,C6,C9,C10,C17,C20,C32,C33,C34 = 22pF, SMD 0402
C7,C38 = 1nF, SMD 0402
C8,C21,C22,C27,C28,C37,C100 = 10nF, SMD 0402
C11-C14 = 220pF, SMD 0402
C15,C39,C44,C45,C52,C60,C61,C66,C70,C74,C78,C86,C90 = 10µF, SMD 0805
C18,C23,C29 = 47nF, SMD 0402
C24,C25,C26,C43 = 1µF, SMD 0805
C36 = 0.1F, Double Layer Cap FG0H104Z135
C40,C41 = 1µF 25V, SMD 1206
C98,C99 = 100µF 16V tantalum, SMD

Inductors

L1 = DLW5BTN102SQ2 (Murata)
L2 = 10µH, MSS5131 (Coilcraft)
L3 = BLM31A (Murata)

Semiconductors

D1,D2,D3,D5 = PMEG3005AEA (Philips)
D4 = SM6T24CA (STM)
IC1 = AT91SAM7SE512-AU (Atmel)
IC2 = MAX3222ECWN (Maxim)
IC3, IC6, IC8 = BZA408B diode array
IC4 = MT48LC32M16A2
IC5 = AT45DB321D-SU (Atmel)
IC7 = VS1053C-L (VLSI)*
IC9 = NC7WZ07P6X (Fairchild)
IC10 = DM9000E (Davicom)
IC11 = PCF8563T (Philips)
IC12 = LT1616 (Linear Technology)
IC13 = LTC1844ES5-SD (Linear Technology)
LED1 = KP-1608URC, red, SMD 0603 (Kingbright)

Miscellaneous

X1 = 18.432 MHz quartz crystal, SMD HC49SM
X2 = 12.288 MHz quartz crystal, SMD HC49SM
X3 = 25.000 MHz quartz crystal, SMD HC49SM
X4 = 32.678 kHz quartz crystal, SMD MC-146
F1 = fuse, 0.5A, fast, with holder, SMD OMNI-BLOK (Littelfuse)
K1,K2,K3 = 40-way SIL pinheader, lead pitch 2.54mm
K4 = USB-B socket, AMP-787780
K5 = 9-way sub-D plug, angled pins, US standard
K6 = 20-way boxheader, 2.54mm lead pitch
K7 = SD-card socket, SMD, FPS009-2700 (Yamaichi)
K8,K9 = 3.5-mm stereo jack socket, SMD, SJ1-3515 (CUI)
K10 = RJ-45 socket with Ethernet transformer and LEDs, SMD, RJLD-043TC (Taimag)
K12 = DC adaptor socket with 2-mm pin, TDC-002-3
JP1 = 6-way 2-row pinheader with 2 jumpers, 2.54mm lead pitch
S1 = pushbutton, SMD, LSH (Schurter)
PCB with pre-mounted SMD parts, Elektor Shop # 071081-1
Project software, archive 071081-11.zip; free downloads from www.elektor.com

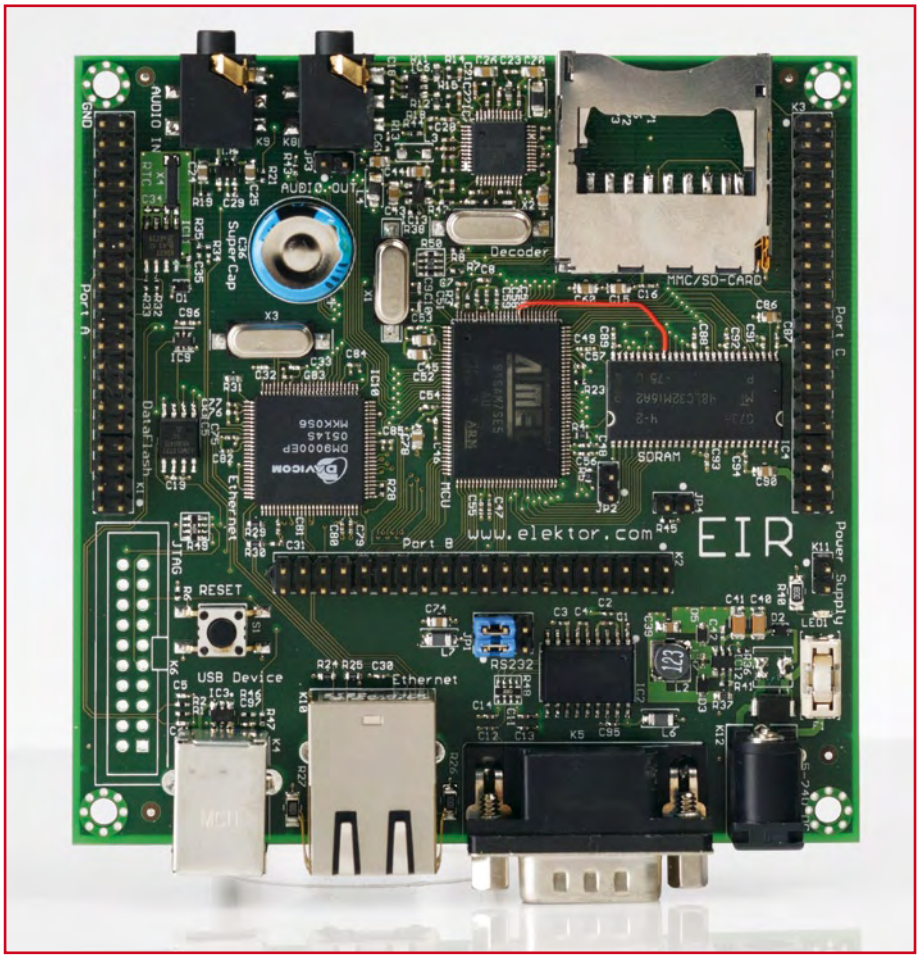


Figure 4. As you can see from the fully assembled prototype, DIY soldering isn't so easy here.

that it should boot from this firmware when it restarts. To do so, select the routine 'Boot from Flash (GPNVM2)' under 'Scripts' and click 'Execute'.

Then close SAM-BA and press the Reset button. Now the EIR can communicate with the PC via the serial interface and a null-modem cable (pin 2 & 3

Table 1. Expansion connector K1

Pin	Signal	Use	Pin	Signal	Use
1	PA0	Free	2	PA1	Free
3	PA2	Free	4	PA3	TWI SDA
5	PA4	TWI SCL	6	PA5	UART0 RxD via JP1
7	PA6	UART0 TxD via JP1	8	PA7	UART0 RTS
9	PA8	UART0 CTS	10	PA9	DBUG RxD via JP1
11	PA10	DBUG TxD via JP1	12	PA11	Data Flash Chip Select
13	PA12	SPI MISO	14	PA13	SPI MOSI
15	PA14	SPI SPCK	16	PA15	MMC Chip Select
17	PA16	MMC Clock	18	PA17	MMC Command
19	PA18	MMC DAT0	20	PA19	MMC DAT1 via R7
21	PA20	MMC DAT2 via R8	22	PA21	Free
23	PA22	Free	24	PA23	SDRAM DQMH
25	PA24	SDRAM A10	26	PA25	SDRAM CKE
27	PA26	SDRAM Chip Select	28	PA27	SDRAM WE
29	PA28	SDRAM CAS	30	PA29	SDRAM RAS
31	PA30	IRQ1, MP3 Interrupt	32	PA31	MP3 Command Select
33	Vref	ADC Reference	34	3.3V	Power
35	AD4	Analogue input (free)	36	AD5	Analogue input (free)
37	AD6	Analogue input (free)	38	AD7	Analogue input (free)
39	GND	Ground	40	GND	Ground

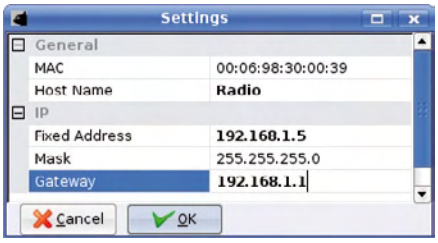


Figure 5. Screen shot of SAM-BA running under Windows 2000.

leads swapped), and you can use a terminal emulator to send commands to the EIR (we recommend TeraTerm [7] for Windows or Miniterm for Linux, and Macs have a built-in terminal utility).

Listening to the radio

Before you can start listening to the radio, you have to delete the test firmware from the EIR and install the radio firmware. To allow new firmware to be loaded, first connect pins 34 and 36 of connector K3 with a jumper, then press Reset, and finally remove the jumper. After this, the EIR will start up again with the boot loader, and you can use SAM-BA to download the radio firmware.

Now connect the EIR to your local network via the Ethernet port (using a hub, a switch, or an Internet router with several ports) and connect the audio output to a headphone or an amplifier.

If the LAN or the router you are using has an active DHCP server, the EIR will fetch a valid address and start playing the programme from the default station. If you prefer to use fixed IP addresses, proceed as follows. When you install Nut/OS, a small utility called 'Discover' is installed on the PC, and you can always use it to find the EIR (see **Figure 6**) and then configure the desired IP address. Enter the router address under 'Gateway', as illustrated in **Figure 7**. After this, you should be able to listen to the radio (**Figure 8**) with fixed IP addresses.

Prospects

As already mentioned several times, the EIR is a completely open-ended concept. The software and hardware (via the ex-

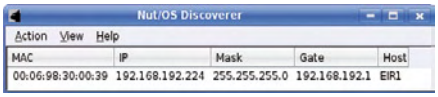


Figure 6. You can use this program (here running under Linux KDF) to find the EIR even if its IP address is unknown.

Table 2. Expansion connector K2					
Pin	**Signal**	**Use**	**Pin**	**Signal**	**Use**
1	PB0	SDRAM DQML	2	PB1	Free
3	PB2	Address Bus A2	4	PB3	Address Bus A3
5	PB4	Address Bus A4	6	PB5	Address Bus A5
7	PB6	Address Bus A6	8	PB7	Address Bus A7
9	PB8	Address Bus A8	10	PB9	Address Bus A9
11	PB10	Address Bus A10	12	PB11	Address Bus A11
13	PB12	Free	14	PB13	Address Bus A13
15	PB14	Address Bus A14	16	PB15	Free
17	PB16	SDRAM BA0	18	PB17	SDRAM BA1
19	PB18	Free	20	PB19	FIQ, RTC Interrupt
21	PB20	IRQ0, Ethernet Interrupt	22	PB21	Free
23	PB22	DataFlash Chip Select	24	PB23	USB Monitor
25	PB24	Free	26	PB25	Free
27	PB26	Free	28	PB27	Free
29	PB28	Free	30	PB29	Free
31	PB30	MP3 Data Select	32	PB31	MP3 Hardware Reset
33	3.3 V	Power	34	3.3 V	Power
35		Not used	36		Not used
37		Not used	38	NRST	Hardware Reset
39	GND	Ground	40	GND	Ground

pansion connectors) can be extended as desired, and you can certainly look forward to seeing several articles on this subject in future issues of *Elektor*.

With regard to software tools, source code and further developments, you should occasionally check the egnite project website [1] to see what's new. There you will find source code and installation files for Windows, Linux and OS X. You will also find links to development environments and other Open Source projects.

There are lots of possibilities. An obvious enhancement would be to add a few buttons and an LCD display so the EIR can be used as a convenient standalone device instead of only via a Web browser. And the memory card slot almost cries to be used for a supplementary MP3 player application.

(071081-1)

Internet radio

If you take a look on the Web, you'll be astounded to see that Google presently finds more than 21 million hits for 'Internet radio', so it's obviously a hot topic. The first experiments with packet-based 'broadcasts' were carried out as early as 1993, at around the same time as the first usable browser (NCSA Mosaic) and thus the dawn of the commercial Internet era. Quite early on, 'real' radio stations started broadcasting their programmes via Internet streaming in addition to conventional radio waves. Today you can receive tens of thousands of radio programmes with an ordinary Internet connection. In addition to a plethora of highly diverse niche programmes, you can now access nearly all public and commercial broadcasters.

The term 'streaming', which covers near-real-time transmission of time-referenced data such as audio or video content, refers to data streams that are as continuous as possible and require the originator to transmit a separate stream for each client, which can generate an enormous traffic volume and thus be a costly proposition if there are a lot of listeners. To keep the data rates within acceptable bounds, a lossy compression is usually used to compress the data before transmission, and the data is subsequently decompressed by the receiver. This means that an Internet radio receiver must incorporate a commonly used streaming decoder, such as MP3, Ogg Vorbis or Real Audio, regardless of whether it is purely software-based or uses dedicated hardware.

As it is not possible to guarantee a constant propagation delay for individual data packets with the HTTP and FTP protocols normally used on the Internet, the receiver must also have a data buffer with sufficient capacity, which delays reception by a few seconds and means that it is only 'quasi live'. This makes fast zapping between stations impossible. However, thanks to digitalisation this drawback is offset by stable audio quality, extreme (worldwide!) range, and a truly fathomless diversity of programmes. It is also possible to receive previously recorded programmes (missed broadcasts) by means of 'audio on demand', which no conventional radio station can offer.

Table 3. Expansion connector K3					
Pin	**Signal**	**Use**	**Pin**	**Signal**	**Use**
1	PC0	Data Bus D0	2	PC1	Data Bus D1
3	PC2	Data Bus D2	4	PC3	Data Bus D3
5	PC4	Data Bus D4	6	PC5	Data Bus D5
7	PC6	Data Bus D6	8	PC7	Data Bus D7
9	PC8	Data Bus D8	10	PC9	Data Bus D9
11	PC10	Data Bus D10	12	PC11	Data Bus D11
13	PC12	Data Bus D12	14	PC13	Data Bus D13
15	PC14	Data Bus D14	16	PC15	Data Bus D15
17	PC16	Bus NWAIT, Open Collector	18	PC17	Ethernet Hardware Reset
19	PC18	MMC Card Detect	20	PC19	MMC Write Protect
21	PC20	Free	22	PC21	Address/Data Bus NWE
23	PC22	Address/Data Bus NRD	24	PC23	Ethernet Chip Select
25		Not used	26		Not used
27		Not used	28		Not used
29		Not used	30		Not used
31		Not used	32		Not used
33	3.3 V	Power	34	3.3 V	Power
35	JTAGSEL	Boundary Scan Enable	36	ERASE	Firmware Erase
37	VIN	Unregulated 5–24 V via R106	38	SHDN	Power Shutdown
39	GND	Ground	40	GND	Ground

References and web links

[1] egnite project website:
www.ethernut.de/de/hardware/eir/

[2] Wikipedia article:
en.wikipedia.org/wiki/Internet_radio

[3] Ethernut and the Kipp Family:
Elektor, March 2008, pp 18–21.

[4] ARM7 MCU data:
www.atmel.com/products/at91/

[5] VS1053 data:
www.vlsi.fi/products/vs1053.html

[6] Link for AT91-ISP.exe:
www.atmel.com/dyn/resources/prod_documents/Install%20AT91-ISP%20v1.10.exe

[7] Windows terminal emulator:
ttssh2.sourceforge.jp/