EIR – das Elektor-In

Radio hören mit modernsten Chips

Von Harald Kipp und Dr. Thomas Scherer

Das waren Zeiten, als man Töne noch analog einer Hochfrequenz aufmodulieren musste, um diese dann zu empfangen und daraus mehr oder minder gut Hörbares zu machen! Das geht heute anders: Audiosignal komprimieren, in IP-Päckchen verpacken, streamen, empfangen, puffern, dekomprimieren und schon hat man auf alle Internet-Radios dieser Welt Zugriff. Alles ganz einfach dank State-of-the-Art-Hardware, um die es hier geht...

Internet-Radio ist etwas ganz Eigenes, denn mit diesem Angebot an Musik und Informationen kommt kein noch so empfindlicher Kurzwellenempfänger auch nur annähernd mit und in Sachen Klangqualität sowieso nicht. Da die zugehörigen „Internet-Sendestationen" ja auch nicht hunderte von kW an elektrosmogener Hochfrequenz in die Luft pusten müssen, kommt der Betrieb eines solchen Senders für kleinere Benutzergruppen also ganz schön preiswert.
Es gäbe noch viel über die Vorzüge dieser neuen Art von Radio zu erzählen (siehe Kasten), doch wichtiger ist die Klärung der Frage:

Warum nicht reine Software?

Zunächst ist zu sagen, dass es diverseste Programme (WinAmp, iTunes, VLC etc.) vollkommen kostenlos für alle möglichen Betriebssysteme gibt, mit denen man Internet-Radio hören kann. Einen PC, einen Mac oder eine Linux-Maschine hat der Mensch des 21sten Jahrhunderts ja sowieso herumstehen. Weshalb also Geld für ein nichtvirtuelles, physisches Gerät ausgeben und gar noch selbst bauen?
Nun, zum einen braucht die Hardware-Grundlage eines Software-Radios Strom und das für diesen Zweck nicht zu knapp. Wer viel Radio mit dem PC via Internet hört, der handelt grob unökologisch. Die hier vorgestellte Lösung kommt mit gerade mal 1 W an Energieverbrauch aus. Bei 10 h Betrieb pro Tag hat sich das EIR gegenüber einem Gamer-PC als Radio schon innerhalb eines Jahres allein durch die eingesparten Stromkosten amortisiert.
Zum anderen gibt es Anwendungen, bei denen ein PC nicht clever ist: Beim Anschluss an eine Stereo-Anlage beispielsweise. Ein selbst gebautes Internet-Radio auf Open-Source-Basis kann man prima erweitern und an spezielle Anforderungen anpassen und last not least spielt das EIR auch weiter, wenn der PC hängt oder abgestürzt ist ;-)

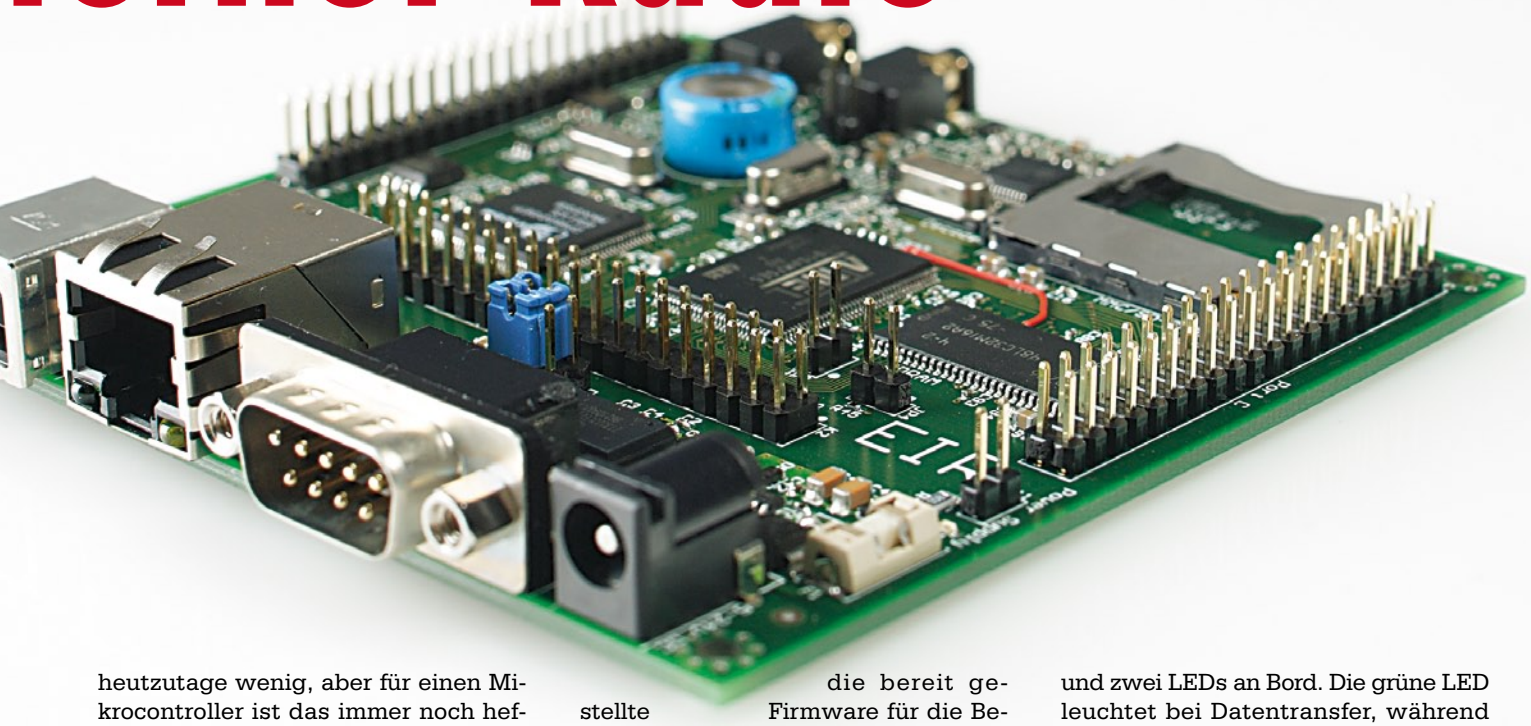
Das Prinzip

Da es sich beim EIR um ein komplexes Projekt mit modernster Technik handelt, ist es unmöglich, alle relevanten Aspekte in einem einzigen Artikel abzuhandeln. In diesem Beitrag geht es daher um die Beschreibung der Hardware und um deren Aufbau und Inbetriebnahme. Weitere Informationen finden sich in Dokumenten zum Artikel auf der Elektor-Website www.elektor.de, der Projekt-Website [1] und in zukünftigen Beiträgen.
Dass ein Internet-Radio Datenströme empfangen, puffern und dekodieren muss, dürfte klar sein. Ohne vernünftigen Mikrocontroller geht also gar nichts. Wie schon in der letzen Ausgabe [3] angedeutet, liefert eine ARM7-CPU [4] die dafür nötige Power.
Bild 1 zeigt den grundlegenden Aufbau: Die CPU (in der Mitte oben) hat Zugriff auf immerhin 64 MB an SD-RAM – genug für Puffer und ganz viel „Sonstiges". Für die Firmware ist Platz in der CPU und zusätzlich gibt es auch noch 4 MB Flash-Speicher für stabil abzulegende Daten. Eine durch einen Supercap gepufferte Echtzeituhr ermöglicht die Realisierung eines Radio-Weckers oder sonstige zeitabhängige Applikationen. Damit die ARM7-CPU nicht völlig ausgelastet werden muss, wird sie bei der Audio-Dekodierung von dem darauf spezialisierten Chip VS1053 [5] unterstützt.
An Schnittstellen ist das EIR wahrlich nicht arm: neben dem obligatorischen Ethernetanschluss – irgendwie muss das EIR ja ins Internet kommen – finden sich eine USB-Programmierschnittstelle zur Übertragung neuer Firmware, eine serielle und eine JTAG-Schnittstelle (gut fürs Debugging) und alleine drei nutzbare Erweiterungsstecker auf Port-Ebene.
Damit Sendungen gegebenenfalls aufgezeichnet werden können, ist auch noch ein Slot für eine MMC-SD-Speicher-Karte vorgesehen.

Generelles

Herein rauschende Datenströme sind in aller Regel so komprimiert, dass sich die typischen mit 16 bit aufgelösten und mit 44,1 kHz abgetasteten Stereodaten statt mit rund 1,4 Mbit/s mit weniger als einem Zehntel, nämlich mit 192 Kbit/s und weniger zufrieden geben. Um einen Puffer mit rund 10 s Kapazität zu realisieren, sind daher rund 256 KB RAM ausreichend. Das scheint

ternet-Radio



heutzutage wenig, aber für einen Mikrocontroller ist das immer noch heftig. Und wenn man sicher gehen will und auch noch Platz für Internet-Feinheiten und „Sonstiges“ haben möchte, dann ist man schnell bei 512 KB und mehr. Die ausgewählte ARM7-CPU unterstützt SD-RAM und so gibt es beim EIR mit den verfügbaren 64 MB keinerlei Speicherplatzprobleme.

Als Betriebssystem wurde das gegenüber Linux recht genügsame Nut/OS ausgewählt, das mit weniger als 40 KB auskommt. Alles in allem sind für die Software rund 200 KB nötig. Für Daten reichen 1 MB locker aus. Da die CPU alleine schon über 512 KB Flash für die Software verfügt und RAM in Massen zur Verfügung steht, gibt es keine Engpässe. Alle Software ist Open-Source - bis auf die das Flash-Programm von Atmel.

Der Controller ist übrigens leistungsfähig genug, um die SD-Karte zur parallelen Aufzeichnung eines zweiten Audio-Streams zu nutzen. Und es wird bestimmt nicht lange dauern, bis jemand aus der Open-Source-Gemeinde dieses und andere denkbare Features nachrüstet.

Um den möglichen Erweiterungen keine Richtung aufzuzwingen, wurden auf dem Board keine expliziten Bedienungselemente wie Taster oder Display vorgesehen. Über die Erweiterungssteckplätze kann Solches aber problemlos angeschlossen werden. Das EIR ist als Grundlage für eigene Erweiterungen konzipiert, und so ist die bereit gestellte Firmware für die Bedienung über eine integrierte Webseite gedacht. Da die Firmware völlig offen ist, muss das aber nicht so bleiben...

Details

Wenn man einen Blick auf den Schaltplan in **Bild 2** wirft, wird einem die Komplexität des Projekts sofort klar. Die folgende Beschreibung orientiert sich daher an den Funktionsblöcken:

- **Ethernet**

Das Internet kommt über eine Ethernetbuchse mit integriertem Übertrager und zwei LEDs an Bord. Die grüne LED leuchtet bei Datentransfer, während die gelbe das Vorhandensein einer Verbindung signalisiert. Der Ethernet-Verkehr wird von einem spezialisierten Chip (IC10 = DM9000E) abgewickelt. Der Buffer IC9 ermöglicht die Nutzung des WAIT-Eingangs der CPU durch Erweiterungen.

- **Audio-Dekoder**

Zwar würde ein ARM7 für die softwaremäßige Dekodierung vom MP3- oder AAC-Daten gerade so ausreichen, aber ein spezialisierter Chip wie IC7 entlastet die CPU beträchtlich und schluckt von Hause aus neben den

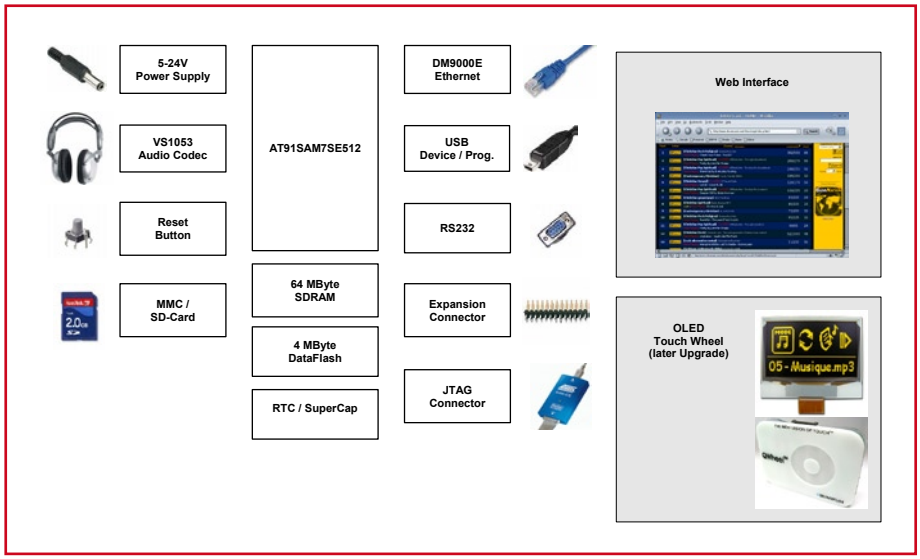


Bild 1. Prinzipschaltung des Elektor-Internet-Radios.

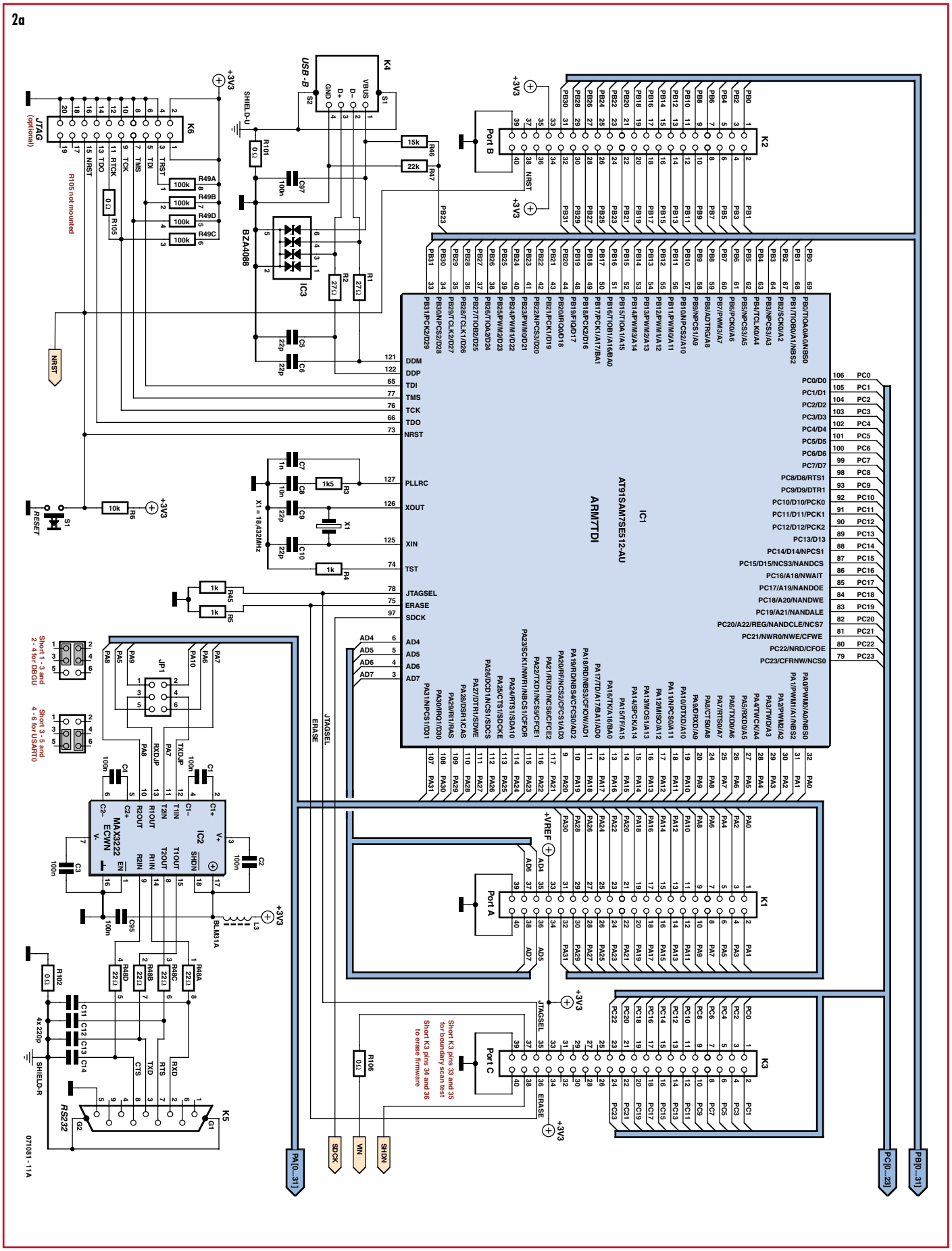
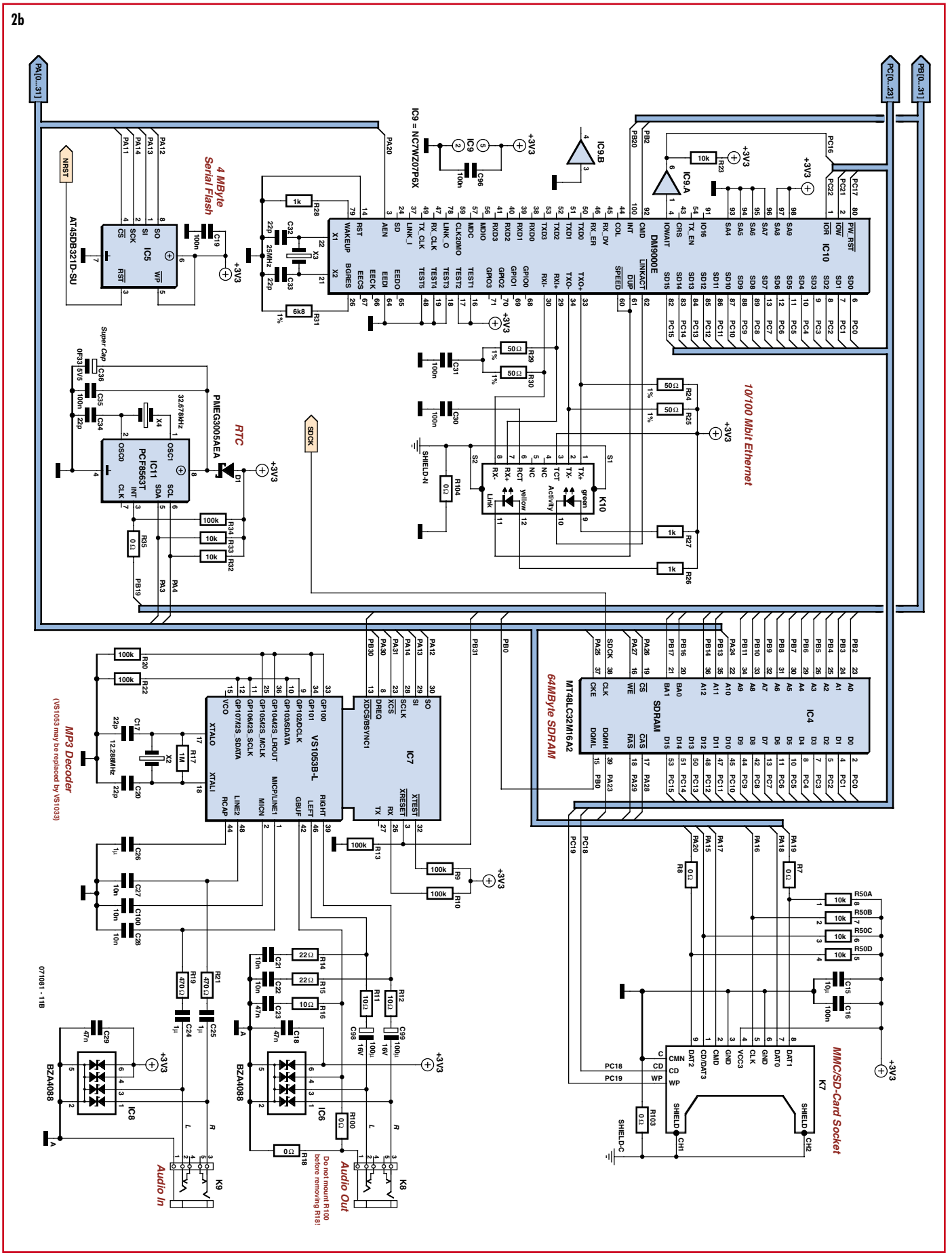
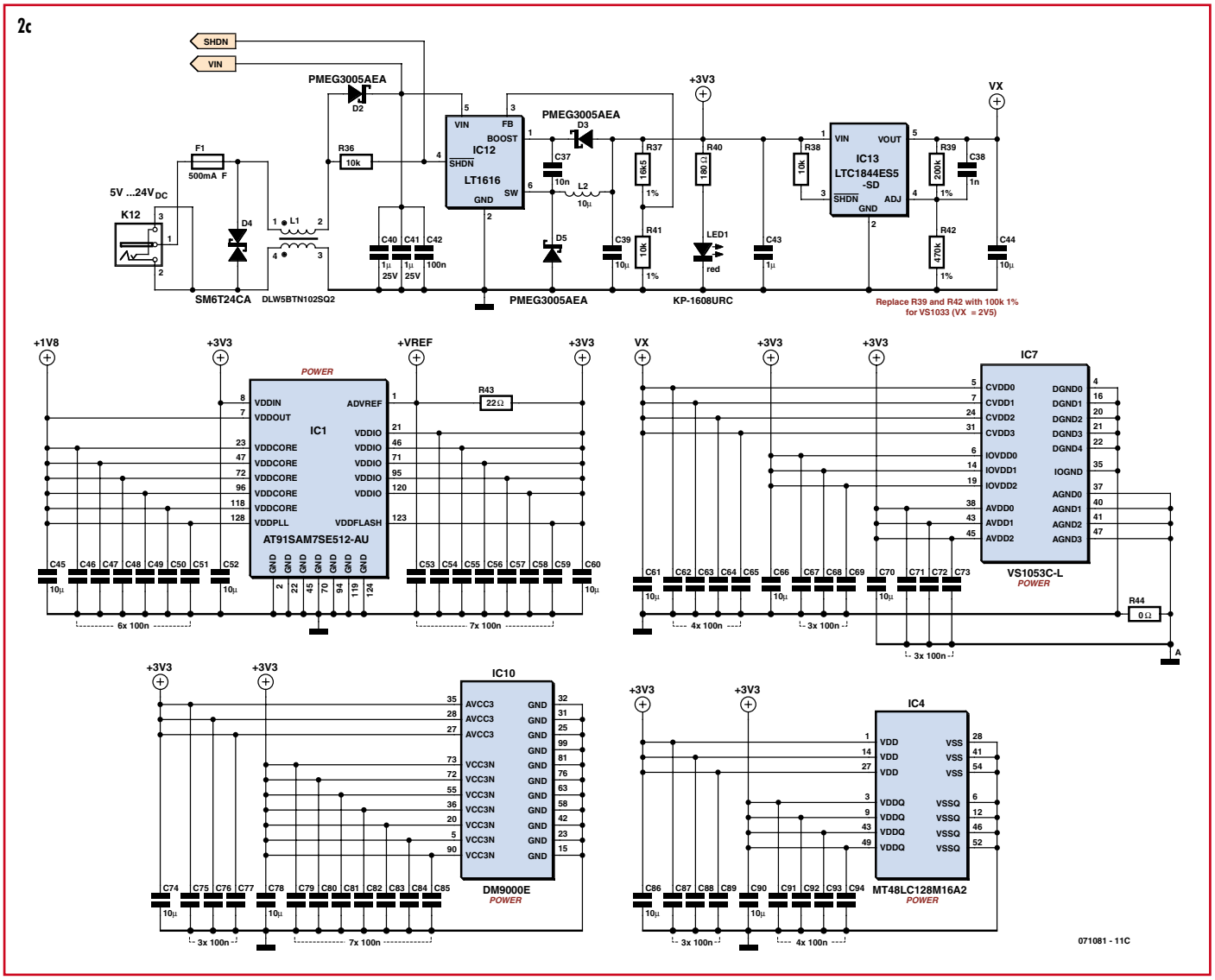


Bild 2. Die konkrete Schaltung des EIR macht augenblicklich klar, dass es sich hier um ein anspruchsvolles Projekt handelt.

2b


PA[0...31]
PB[0...31]
PC[0...23]
IC10
DM9000E
10/100 Mbit Ethernet
IC9 = NC7WZ07P6X
IC9.A
IC9.B
+3V3
AT45DB321D-SU
4 MByte Serial Flash
NRST
IC5
IC11
PCF8563T
RTC
PME3005AEA
Super Cap
32.678kHz
IC4
MT48LC32M16A2
64MByte SDRAM
SDRAM
SDCK
IC7
VS1053B-L
MP3 Decoder
(VS1033 may be replaced by VS1033)
12.288MHz
IC6
IC8
BZA4088
Audio In
Audio Out
Do not mount R100 before removing R18!
K7
MMC/SD-Card Socket
K8
K9
K10
S1
S2
green
yellow
Activity
Link
SHIELD-N
SHIELD-C
071081 - 11B



gebräuchlichen MP3-Varianten auch HE-AAC und sogar Ogg-Vorbis-Daten. Gleichzeitig vereinfacht sich damit logischerweise die nötige Software. Bei unserem Prototypen hatten wir ein Vorab-Muster von VLSI im Einsatz. Sollte es Beschaffungsprobleme geben, kann man auch die Variante VS1033 (ohne Ogg Vorbis) einsetzen. Obwohl die CPU einen 1,8-V-Ausgang zur Versorgung von Peripherie hat, wurde für IC7 aus Stabilitätsgründen ein eigener 1,8-V-Spannungsregler vorgesehen. Da die Variante VS1033 aber 2,5 V benötigt, müssen R39 und R42 dann auf je 100 kΩ geändert werden.

• Zusatz-Flash

Zum Betrieb des Radios sind umfangreiche Einstellungen zu speichern, die auch nach einem Stromausfall verfügbar bleiben sollen - allem voran die Senderliste. Das ginge auch mit dem internen Flash-Speicher der CPU, aber dieser ist umständlich zu beschreiben. Um dies zu vereinfachen, wurde ein serieller Flashspeicher (IC5) hinzugefügt, in dessen 4 MB sich auch umfangreiche Senderlisten und mehr ablegen lassen.

• Stromversorgung

Damit das EIR auch wirklich wenig Energie verbraucht, wurde es mit einem Schaltregler (IC12) versehen. Bei Eingangsspannungen im Bereich von 5...24 V stehen rund 5 W bei 3,3 V zur Verfügung. Da das EIR selbst nur 1 W benötigt, bleibt noch genug Energie für eigene Hardware-Erweiterungen übrig.

• Löten

Da es sich hier um eine mit winzigen SMDs dicht bepackte mehrlagige Platine handelt (siehe **Bild 3** und **Bild 4**), wobei einzelne IC-Pins nur 0,5 mm voneinander entfernt sind, bietet Elektor eine mit allen SMD-Bauteilen vorbestückte Platine (mit VS1053) an. Man muss also nur noch die normal bedrahteten Bauteile bestücken und vermeidet so tückische Fehler. Unentwegten steht der komplette Eigenbau anhand der Layouts natürlich frei.

Funktionstest

Für einen ersten Test der Stromversorgung sollte die 3,3-V-Seite mit einigen mA belastet werden – also nicht leer laufen. Ab 4 V am Eingang wird der Schaltregler arbeiten und je nach Belastung 50...150 mA aufnehmen. Bei 24 V reduziert sich das auf 30...50 mA. Ist alles in Ordnung, leuchtet LED1. Sind die ICs bestückt, kann man mit einem Oszilloskop die Quarze auf Funktion testen. Wenn X1 schwingt, müsste sich die CPU ansprechen lassen.
Die CPU ist schon von Hause aus mit einem Bootloader versehen, der außer

der reinen Übertragung neuer Firmware auch die Kommunikation mit dem RAM und mit dem Flash-Speicher ermöglicht. Von Atmel gibt es unter [6] die Datei „AT91-ISP.exe“, die nach Entpacken das Windows-Programm „SAM-BA“ enthält. Nach dessen Installation verbindet man das EIR via USB mit einem PC. Nach Einschalten der Stromversorgung sollte Windows automatisch den passenden Treiber aktivieren. Jetzt kann SAM-BA gestartet werden. Man muss dann als Verbindungsart USB und als Gerät das „AT91SAM7SE512-EK“ auswählen, welches zum EIR weitgehend kompatibel ist.

Von der Elektor-Webseite zum Artikel kann eine einfache Firmware für Testzwecke herunter geladen werden. Mit ihr und einer funktionierenden CPU plus der seriellen Schnittstelle können die anderen Bausteine wie Ethernet und Audio-Dekoder überprüft werden. Nach der Übertragung der Firmware muss dem EIR noch gesagt werden, dass es beim nächsten Start von dieser Firmware booten soll. Hierfür unter „Scripts“ die Routine „Boot from Flash (GPNVM2)“ auswählen und auf „Execute“ klicken - dann SAM-BA beenden und auf den Reset-Taster drücken. Nun wird das EIR via serieller Schnittstelle über ein Null-Modem-Kabel (Pins 2 und

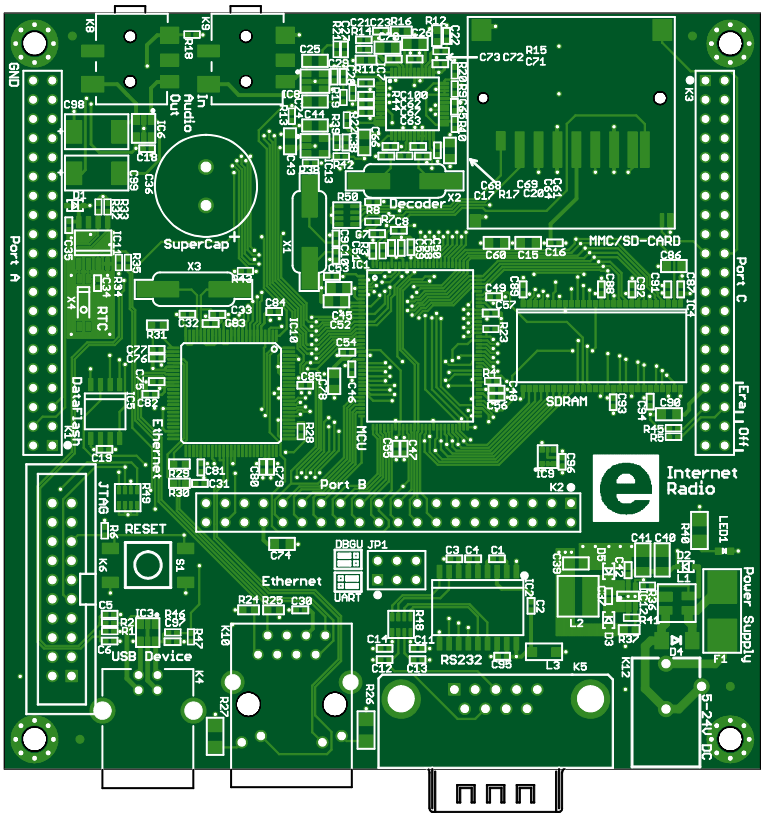


Bild 3. Der Bestückungsplan des EIR. Um Schwierigkeiten zu vermeiden, gibt es eine mit SMDs vorbestückte Platine.

Stückliste

Widerstände:
R1,R2 = 27 Ω, SMD 0402
R3 = 1k5, SMD 0402
R4,R5,R28,R45 = 1 k, SMD 0402
R6,R23,R32,R33,R36,R38 = 10 k, SMD 0402
R7,R8,R18,R35,R44 = 0 Ω, SMD 0402
R9,R10,R13,R20,R22,R34 = 100 k, SMD 0402
R11,R12,R16 = 10 Ω, SMD 0603
R14,R15,R43 = 22 Ω, SMD 0402
R17 = 1 M, SMD 0402
R19,R21 = 470 Ω, SMD 0402
R24,R25,R29,R30 = 50 Ω 1%, SMD 0402
R26,R27 = 1 k, SMD 1206
R31 = 6k8 1%, SMD 0603
R37 = 16k5 1%, SMD 0603
R39 = 200 k* 1%, SMD 0402
R40 = 180 Ω, SMD 1206
R41 = 10 k 1%, SMD 0402
R42 = 470 k* 1%, SMD 0402
R46 = 15 k, SMD 0402
R47 = 22 k, SMD 0402
R48 = 22 Ω, Array CAY16
R49 = 100 k, Array CAY16
R50 = 10 k, Array CAY16
R100...R106 = 0 Ω*, SMD 1206 (nicht erforderlich)
* siehe Text

Kondensatoren:
(SMD Keramik 6,3 V, falls nicht anders angegeben)
C1...C4,C16,C19,C30,C31,C35,C42,C46...C51,C53...C59,C62...C65,C67...C69,C71..C73,C75...C77,C79...C85,C87...C89,C91...C97 = 100 n, SMD 0402
C5,C6,C9,C10,C17,C20,C32,C33,C34 = 22 p, SMD 0402
C7,C38 = 1 n, SMD 0402
C8,C21,C22,C27,C28,C37,C100 = 10 n, SMD 0402
C11...C14 = 220 p, SMD 0402
C15,C39,C44,C45,C52,C60,C61,C66,C70,C74,C78,C86,C90 = 10 µ, SMD 0805
C18,C23,C29 = 47 n, SMD 0402
C24...C26,C43 = 1 µ, SMD 0805
C36 = 0,1 F, Double Layer Cap FG0H104Z135
C40,C41 = 1 µ/25V, SMD 1206
C98,C99 = 100 µ/16V, Tantal, SMD

Induktivitäten:
L1 = DLW5BTN102SQ2 (Murata)
L2 = 10 µ, MSS5131 (Coilcraft)
L3 = BLM31A (Murata)

Halbleiter:
D1...D3, D5 = PMEG3005AEA (Philips)
D4 = SM6T24CA (STM)
IC1 = AT91SAM7SE512-AU (Atmel)
IC2 = MAX3222ECWN (Maxim)
IC3, IC6, IC8 = Diodenarray BZA408B
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, rot, SMD 0603 (Kingbright)

Außerdem:
X1 =18,432-MHz-Quarz, SMD HC49SM
X2 = 12,288-MHz-Quarz, SMD HC49SM
X3 = 25,000-MHz-Quarz, SMD HC49SM
X4 = 32,678-kHz-Quarz, SMD MC-146
F1 = Sicherung, 0,5 A flink mit Halter, SMD OMNI-BLOK (Littelfuse)
K1, K2, K3 = 40-polige Stiftleiste, 2,54-mm-Raster, zweireihig
K4 = USB-B-Buchse, AMP-787780
K5 = 9-poliger Sub-D-Stecker, gewinkelt, US-Norm
K6 = 20-poliger Wannenstecker, 2,54-mm-Raster, zweireihig
K7 = SD-Karten-Fassung, SMD FPS009-2700 (Yamaichi)
K8, K9 = 3,5-mm-Stereo-Klinkenbuchse, SMD SJ1-3515 (CUI)
K10 = RJ-45-Buchse mit Ethernet-Übertrager und LEDs, SMD, RJLD-043TC (Taimag)
K12 = DC-Buchse mit 2-mm-Stift, TDC-002-3
JP1 = 6-polige Stiftleiste mit 2 Kurzschlussbrücken, 2,54-mm-Raster, zweireihig
S1 = Taster, SMD, LSH (Schurter)
Platine 071081-1 oder SMD-bestückte Platine 071081-71
Software-Download von der Elektor-Webseite

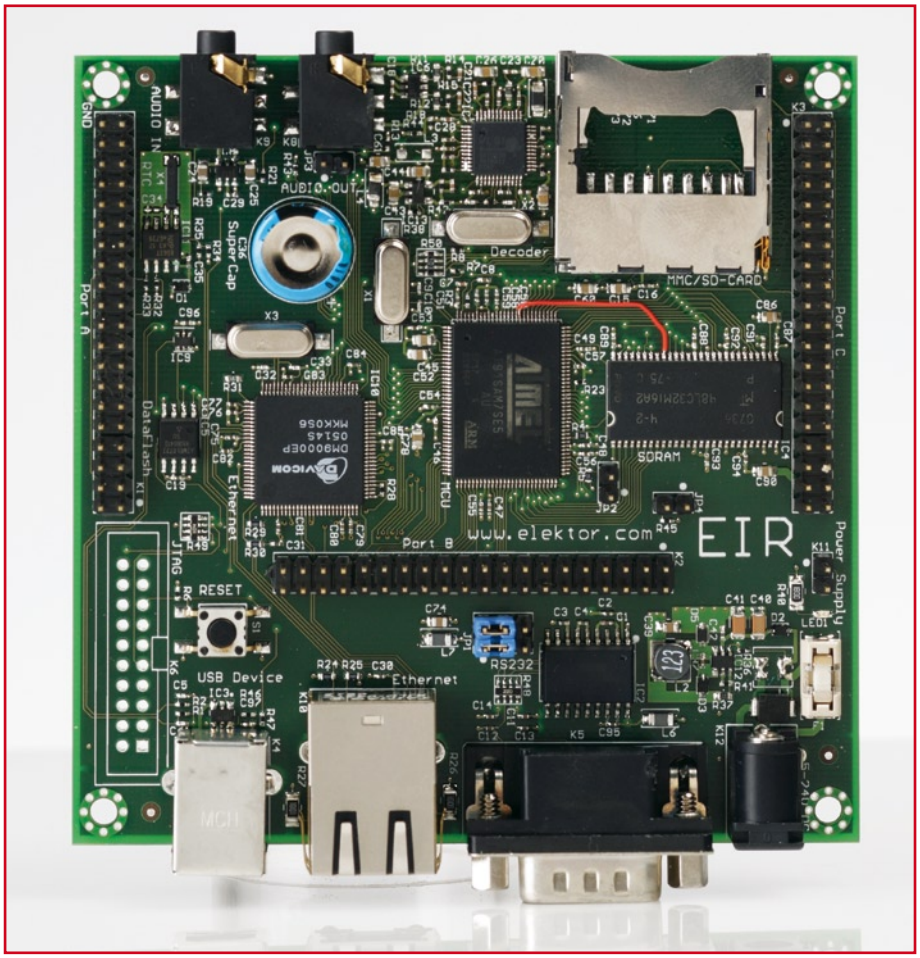


Bild 4. Der fertig bestückte Prototyp macht klar, dass manuelles Bestücken hier nicht so einfach ist.

3 gekreuzt) an den PC angeschlossen und über eine Terminal-Emulation kann man bei 115,2 kBaud bei 8/0/1 (Daten-, Parität- und Stopp-Bits) die Ausgaben des EIR verfolgen. Als Terminal-Emulation empfohlen wird für Windows TeraTerm [7] und für Linux Miniterm. Ein Mac hat ein Terminal immer dabei.

Tabelle 1. Erweiterungsstecker K1

Pin	Signal	Verwendung	Pin	Signal	Verwendung
1	PA0	frei	2	PA1	frei
3	PA2	frei	4	PA3	TWI SDA
5	PA4	TWI SCL	6	PA5	UART0 RxD über JP1
7	PA6	UART0 TxD über JP1	8	PA7	UART0 RTS
9	PA8	UART0 CTS	10	PA9	DBUG RxD über JP1
11	PA10	DBUG TxD über JP1	12	PA11	DataFlash 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 über R7
21	PA20	MMC DAT2 über R8	22	PA21	frei
23	PA22	Frei	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	AD Wandler Referenz	34	3.3V	Versorgung
35	AD4	Analogeingang, frei	36	AD5	Analogeingang, frei
37	AD6	Analogeingang, frei	38	AD7	Analogeingang, frei
39	GND	Masse	40	GND	Masse

Bild 5. Mit der Software (hier unter Linux KDE) findet man das EIR auch bei unbekannter IP-Adresse.

Radio hören

Bevor dies möglich ist, muss die Test-Firmware vom EIR entfernt und die Radio-Firmware geladen werden. Um ein neues Laden der Firmware zu ermöglichen, überbrückt man zunächst die Pins 34 und 36 bei K3 mit einem Jumper, betätigt Reset und entfernt dann den Jumper wieder. Anschließend bootet das EIR wieder mit dem Bootloader und mit SAM-BA kommt die Radio-Firmware drauf.

Jetzt wird das EIR über Ethernet mit dem lokalen Netz (via Hub/Switch oder den Internet-Router mit mehreren Ports) verbunden, und an den Audio-Ausgang schließt man einen Kopfhörer oder ein Verstärker an.

Sollte das LAN beziehungsweise der vorhandene Router über einen aktivierten DHCP-Server verfügen, so holt sich das EIR eine gültige Adresse und beginnt mit dem Abspielen des voreingestellten Radio-Senders. Falls feste IP-Adressen bevorzugt werden: Bei der Installation von Nut/OS wurde schon das kleine Tool „Discover“ auf dem PC installiert, mit dem sich das EIR auf jeden Fall finden (**Bild 5**) und dann auch die gewünschte IP-Adresse einstellen lässt. Wie in **Bild 6** gezeigt, trägt man unter Gateway die Adresse des Routers ein. Ab da müsste das Radio-Hören auch bei fixen IP-Adressen funktionieren.

Ausblick

Beim EIR handelt es sich, wie schon mehrfach erwähnt, um ein völlig offenes Konzept. Man kann sowohl die Software als auch die Hardware (über die Erweiterungsstecker) beliebig erweitern, und in Elektor wird man zu diesem Thema sicherlich noch einiges

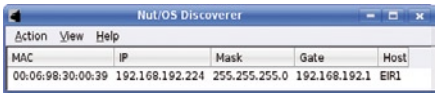


Bild 6. Mit Discover kann man eventuelle Netzwerk-Einstellungen des EIR vornehmen.

Tabelle 2. Erweiterungsstecker K2					
Pin	**Signal**	**Verwendung**	**Pin**	**Signal**	**Verwendung**
1	PB0	SDRAM DQML	2	PB1	frei
3	PB2	Adressbus A2	4	PB3	Adressbus A3
5	PB4	Adressbus A4	6	PB5	Adressbus A5
7	PB6	Adressbus A6	8	PB7	Adressbus A7
9	PB8	Adressbus A8	10	PB9	Adressbus A9
11	PB10	Adressbus A10	12	PB11	Adressbus A11
13	PB12	frei	14	PB13	Adressbus A13
15	PB14	Adressbus A14	16	PB15	frei
17	PB16	SDRAM BA0	18	PB17	SDRAM BA1
19	PB18	frei	20	PB19	FIQ, RTC Interrupt
21	PB20	IRQ0, Ethernet Interrupt	22	PB21	frei
23	PB22	DataFlash Chip Select	24	PB23	USB Monitor
25	PB24	frei	26	PB25	frei
27	PB26	frei	28	PB27	frei
29	PB28	frei	30	PB29	frei
31	PB30	MP3 Data Select	32	PB31	MP3 Hardware Reset
33	3,3 V	Versorgung	34	3,3 V	Versorgung
35		Nicht belegt	36		Nicht belegt
37		Nicht belegt	38	NRST	Hardware Reset
39	GND	Masse	40	GND	Masse

lesen können.
Bezüglich der Software-Tools und Sourcen sowie der Weiterentwicklungen wirft man am Besten gelegentlich einen Blick auf die Projekt-Webseite [1] von egnite. Hier finden sich Quelltexte und Installationen für Windows, Linux und OS X sowie Links zu Entwicklungsumgebungen und weiterführenden Open-Source-Projekten.
Vieles ist denkbar. Naheliegend wären ein paar Tasten und ein LCD-Display, damit sich das EIR nicht nur via Web-Browser, sondern auch als luxuriöses Stand-alone-Gerät nutzen lässt. Der Kartenslot schreit förmlich nach der zusätzlichen Nutzung als MP3-Player…

(071081e)

Internet-Radio

Ein Blick ins Internet und man wird geradezu erschlagen: Google findet zurzeit unter dem Stickwort „Internetradio“ über 2,3 Mio. Treffer – ein heißes Thema also. Dabei waren erste Experimente zu paketvermittelten „Sendungen“ schon 1993 zu beobachten, quasi gleichzeitig mit dem ersten brauchbaren Browser NCSA Mosaic und sozusagen in der Stunde Null des kommerziellen Internet. Schon recht früh begannen „richtige“ Radiosender damit, ihre bislang terrestrisch abgestrahlten Sendungen zusätzlich auch via Internet-Streaming zu verbreiten. Heute sind mit einem gewöhnlichen Internet-Anschluss zehntausende Radioprogramme zu empfangen. Neben einer Unmenge an thematisch unterschiedlichsten Spartenprogrammen hat man mittlerweile Zugriff auf nahezu alle öffentlich-rechtlichen und kommerziellen Sender.

Unter dem für die zeitnahe Übertragung zeitbasierter Daten wie Audio- oder Videoinhalte subsumierten Streaming versteht man möglichst kontinuierliche Datenströme, von denen die Senderseite pro Klient einen extra Stream bieten muss. Das kann ganz schön viel Traffic generieren und für den Sender kostspielig werden, wenn er sehr viele Zuhörer hat. Um die Datenraten erträglich zu halten, werden die Daten vor der Versendung in aller Regel verlustbehaftet komprimiert und beim Empfänger wieder dekomprimiert. Ein Internet-Radio muss also – ob rein sotwarebasiert oder als spezielle Hardware – über einige gängige Streaming-Decoder wie MP3, Ogg Vorbis oder Real Audio verfügen.

Da im Internet mit den üblichen Protokollen HTTP und FTP keine stabilen Laufzeiten für die einzelnen Datenpakete garantiert werden können, benötigt der Empfänger einen ausreichenden Datenpuffer, was den Empfang um einige Sekunden verzögert und also nur „quasi-live“ ermöglicht. Schnelles Zappen ist daher nicht möglich. Dafür wird man dank der Digitalisierung mit einer stabilen Tonqualität, einer extremen (weltweiten) Reichweite und einer gegenüber dem klassischen Rundfunk geradezu unüberschaubaren Programmvielfalt entschädigt. Zudem ist prinzipiell der „Empfang“ von Konserven (= verpassten Sendungen) als „Audio on Demand“ möglich, was kein konventionelles Radio bieten kann.

Tabelle 3. Erweiterungsstecker K3					
Pin	**Signal**	**Verwendung**	**Pin**	**Signal**	**Verwendung**
1	PC0	Datenbus D0	2	PC1	Datenbus D1
3	PC2	Datenbus D2	4	PC3	Datenbus D3
5	PC4	Datenbus D4	6	PC5	Datenbus D5
7	PC6	Datenbus D6	8	PC7	Datenbus D7
9	PC8	Datenbus D8	10	PC9	Datenbus D9
11	PC10	Datenbus D10	12	PC11	Datenbus D11
13	PC12	Datenbus D12	14	PC13	Datenbus D13
15	PC14	Datenbus D14	16	PC15	Datenbus D15
17	PC16	Bus NWAIT, Open Collector	18	PC17	Ethernet Hardware Reset
19	PC18	MMC Card Detect	20	PC19	MMC Write Protect
21	PC20	frei	22	PC21	Adress-/Datenbus NWE
23	PC22	Adress-/Datenbus NRD	24	PC23	Ethernet Chip Select
25		Nicht belegt	26		Nicht belegt
27		Nicht belegt	28		Nicht belegt
29		Nicht belegt	30		Nicht belegt
31		Nicht belegt	32		Nicht belegt
33	3,3 V	Versorgung	34	3,3 V	Versorgung
35	JTAGSEL	Boundary Scan Enable	36	ERASE	Firmware Erase
37	VIN	5-24 V ungeregelt über R106	38	SHDN	Power Shutdown
39	GND	Masse	40	GND	Masse

Literatur & Links

[1] Projekt-Webseite von egnite:
www.ethernut.de/de/hardware/eir/

[2] Wikipedia-Artikel:
de.wikipedia.org/wiki/Internet-Radio

[3] Ethernut und die Kipp-Familie:
Elektor März 2008, Seite 46...49.

[4] Informationen zur ARM7-CPU:
www.atmel.com/products/at91/

[5] Informationen zum VS1053:
www.vlsi.fi/en/products/vs1053.html

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

[7] Windows-Terminal:
ttssh2.sourceforge.jp/