Fuchsjagd ohne heraushängende Zunge:

Mini-Sender
und
Mini-Empfänger

für Fuchsjagden
im lizenz- und gebührenfreien
70cm-ISM-Band

Roland Walter, DL7UNO
www.rowalt.de, support@rowalt.de

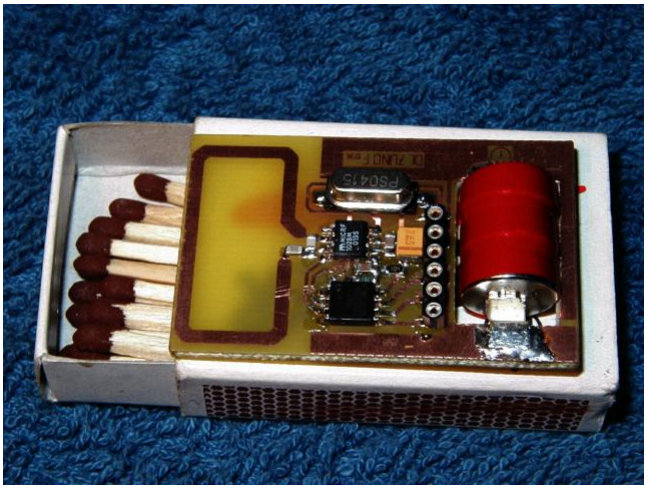
Das Mini-Fuchs-Projekt wurde in der Zeitschrift FUNKAMATEUR, Hefte 09+10+12/2002 veröffentlicht. Dies ist eine überarbeitete und erweiterte Fassung, in die viele Leser-Fragen und -Hinweise eingeflossen sind.
Ab 01. August 2003 gibt es den Fuchsjagd-Sender und den Peilempfänger auch fertig aufgebaut zu kaufen, da SMD und Mikrocontroller-Programmierung nun einmal nicht jedermanns Sache sind. Beide Geräte haben gegenüber den hier vorgestellten einige Veränderungen erfahren. Mehr dazu siehe www.rowalt.de/mc/ im Bereich "Shop".
Wichtig: Die früher erteilte Genehmigung zur kommerziellen Herstellung ziehe ich hiermit zurück. Das muß ich einfach tun, um mir nicht selbst ein Bein zu stellen. Der Selbstbau und die kostenlose Weitergabe bleibt selbstverständlich weiterhin erlaubt und ist gern gesehen.

Zum Projekt gehört die Datei MiniFox.zip, welche das AVR-Programm und die Leiterplattenlayouts im SprintLayout-Format und im EMF-(Enhanced Metafile)-Format enthält.

Abstract:
This PDF file contains the extended version of 3 articles published in the german journal FUNKAMATEUR 09+10+12/2002. The articles describe a transmitter and a receiver for amateur fox hunting contests where the competitors have to find one or more hidden radio transmitters.
The devices use the free ISM band 433,92 MHz so that no special licence is necessary for usage. Both transmitter and receiver is very cheap and simple to build and to use because they are based on the MICREL ICs MICRF002 and MICRF102 which need a minor number of external components. The transceiver gets the signals from an AVR microcontroller and sends the international Fox Morse codes MO, MOE, MOI, MOS and MOH. The AVR program source code is part of this PDF file.

Roland Walter, support@rowalt.de

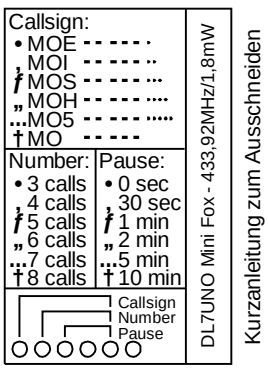
Mini-Fuchsjagdsender fürs 70cm-Band



In diesem Artikel wird ein kleiner, billiger und leicht aufzubauender Fuchsjagdsender vorgestellt, der samt Batterien und Antenne in eine Streichholzschachtel paßt. Er sendet im ISM-Bereich des 70cm-Bandes mit etwa 2,5dBm=1,8mW. Damit darf er auch an Nicht-Funkamateure weitergegeben werden (mehr dazu siehe weiter unten). Trotz der geringen Leistung und der verwendeten Mini-Antenne wurde auf freiem Feld eine Reichweite von bis zu 1,25km erzielt (Empfänger: FM-Handfunkgerät). Zur Komplettierung folgt ein weiterer Artikel, der einen passenden ebenso einfachen 70cm-Peilempfänger beschreibt, der allerdings nur eine Reichweite von etwa 350m zuläßt. Das genügt aber für unaufwendige Fuchsjagden in Parkanlagen, zumal der kleine Sender (oder mehrere davon) gut versteckt werden kann.

Der Fuchs erzeugt einen ASK-Ton von etwa 1 KHz, der sich sowohl mit AM- als auch mit FM-Empfängern gut demodulieren läßt. Der Fuchs beherrscht die in Deutschland zugelassenen Morsekennungen MO, MOE, MOI, MOS, MOH und MO5. Die Kennung kann durchgängig gesendet werden oder in in Gruppen von 3 bis 8 Rufen mit einstellbarer Pause zwischen 30 Sekunden und 10 Minuten.
Sender und Empfänger waren Nebenprodukte einer kommerziellen Entwicklung, es wurde aber darauf geachtet, daß alle Bauteile auch für den Amateur leicht zu beschaffen sind. Zum Aufbau der Schaltung sind keine Mikrocontroller-Kenntnisse erforderlich. Benötigt wird aber das AVR-Brennprogramm WinAVR und der AVR-Programmieradapter, den ich am Anfang meiner AVR-Einstiegsserie im FUNKAMATEUR 4/2002 und 5/2002 vorgestellt hatte (siehe auch www.rowalt.de/mc/).

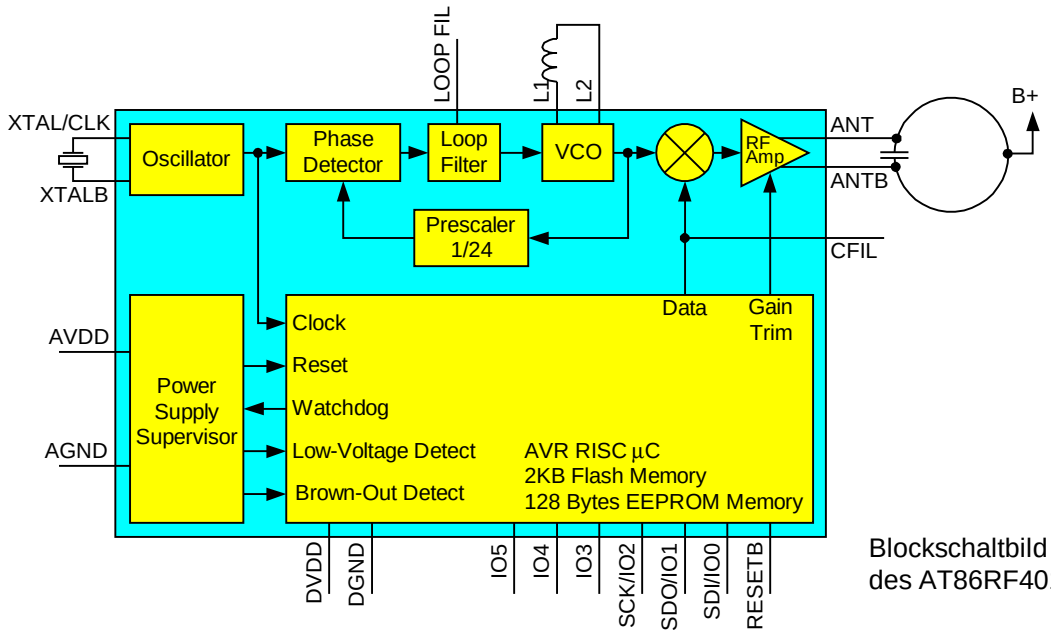
Der Fuchs aus Anwendersicht



Der Fuchs fängt an zu senden, sobald die Batterie eingelegt wird. Beispielsweise wird die Morsekennung MOS 5 mal ausgesendet und es folgt eine Pause von 30 Sekunden, bis wieder 5 mal die Morsekennung gesendet wird.
Für die Veränderung von Kennung, Anzahl der Kennungen und Pausenlänge gibt es drei Tastenanschlüsse. Beim Einstellen der Fuchs-Parameter muß zur akustischen Kontrolle ein Empfänger eingeschaltet werden. Nach jedem Tastendruck ist im Empfänger erst eine Dreitonfolge zu hören und dann 1...6 mal "Dit". Die Anzahl der "Dits" gibt die gewählte Einstellung an. Alle Einstellungen werden dauerhaft gespeichert und bleiben auch nach dem Entnehmen der Batterie bestehen.
Die Funktion der drei Tasten im einzelnen ist im Kasten links zu sehen.

Die Wahl der Bauteile

Zunächst war vorgesehen, den Fuchs mit dem brandneuen Atmel-IC AT86RF401 (anderer Name: AT29646) aufzubauen. Dieser IC enthält einen 70cm-CW-Sender und einen AVR-Mikrocontroller in einem Gehäuse, läßt sich mit einer Betriebsspannung von minimal 2 Volt betreiben und die Ausgangsleistung ist bis maximal 6dBm=4mW einstellbar. Das wäre an sich der IC der Wahl gewesen. Aber leider war der AT86RF401 noch nicht in Produktion und ich konnte nach langen Bemühungen lediglich ein (!) Entwicklungsmuster bekommen. Hinzu kam, daß es an Referenzschaltungen mangelte und vor allem genaue Informationen zur Antennenanpassung fehlten. Nachdem 3 Mails an den Support von Atmel-Grenoble nicht beantwortet wurden, legte ich den IC erst einmal auf Eis. Aber für den Amateur bleibt auch später noch ein gewichtiger Nachteil bestehen: Die Sendefrequenz ist beim AT86RF401 das 24fache der Quarzfrequenz. Damit wird fürs 70cm-ISM-Band ein 18,08MHz-Quarz benötigt. Quarze dieser Frequenz sind in geringen Stückzahlen nicht billig verfügbar. In Einzelfertigung würde der Fuchs mit 12,50 Euro pro Quarz also ziemlich teuer werden. Schade eigentlich.

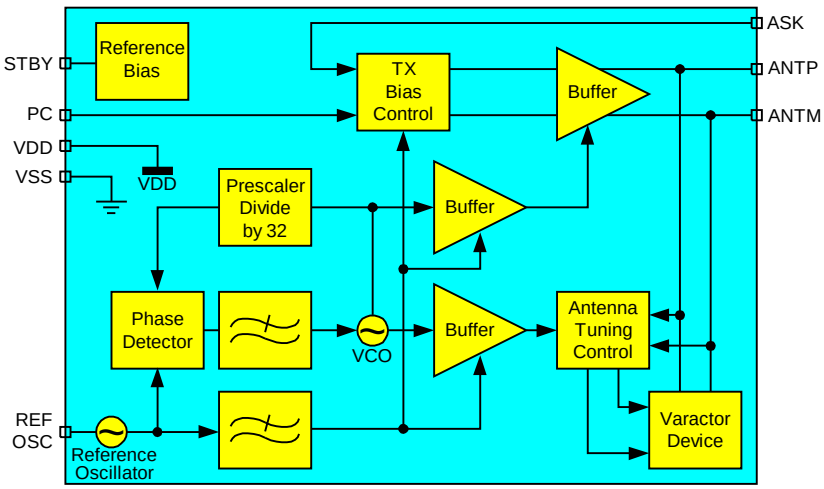


Blockschaltbild des AT86RF401

Die Wahl fiel dann auf den MICRF102BM von Micrel. Auch dieser IC ist ein UHF-Sender, allerdings ohne internen Mikrocontroller. Er liefert gegenüber dem AT86RF401 nur etwa die halbe HF-Sendeleistung, benötigt eine höhere Betriebsspannung und natürlich einen externen Mikrocontroller, der die Signale liefert. Das sind gewichtige Nachteile. Was aber ins Gewicht fiel, war die Frequenzaufbereitung: Beim MICRF102 ist die Sendefrequenz das 32fache der Quarzfrequenz, womit fürs 70cm-ISM-Band ein billiger Massenquarz von 13,56MHz verwendet werden kann. Der Micrel-Distributor Dacom Süd verkauft diesen Quarz für 1,90 Euro (SMD-Gehäuse) und Segor für 1 Euro (HC49-Gehäuse). Vernachlässigt man die zusätzlich benötigten 5 Kondensatoren und 2 Widerstände, dann kommt man beim Einzelhandel mit dem AVR zusammen auf einen Gesamtpreis von knapp 10 Euro.
Zum MICRF102 gibt es als kompatiblen Empfänger-IC den MICRF002. Da dieser IC grundsätzlich ein AM-Empfänger ist, kann er für einen einfachen Fuchsjagd-Empfänger Verwendung finden. Prinzipbedingt (Rauschen) erreicht man mit diesem Empfänger-IC aber nur Reichweiten bis zu 350 Metern auf freiem Feld (getestet). Der Peilempfänger weiter unten wird mit dem MICRF002 aufgebaut.

Die verwendeten HF-ICs waren nur als SMD-Variante beschaffbar. Ich ließ mich erstmalig auf diese Technik ein, war am Ende begeistert und habe alle Vorbehalte beiseite geworfen. Ein normaler 40W-Lötkolben mit guter Spitze und dünneres Lötzinn reichten völlig aus, solange die Lötkolbenspitze mit einem feuchten Schwamm von Zunder freigehalten wird. Als Spezialwerkzeug wird nur eine Kreuzklemm-Pinzette und eventuell eine Lupe mit Ständer benötigt, das ist alles. Dafür fällt das Bohren von Löchern weg.

Der MICRF102



Der MICRF102 ist als billiger Sender für ISM-Anwendungen wie drahtlose Türöffner und Fernsteuerungen vorgesehen. Die mögliche Sendefrequenz liegt im Bereich von 300 bis 470MHz. Die Antenne wird innerhalb eines bestimmten Bereiches automatisch abgestimmt. Die Ausgangsleistung kann mit einem externen Spannungsteiler eingestellt werden. Das Maximum liegt wie oben schon angegeben bei etwa 2,5dBm=1,8mW. Als Modulation wird ASK (Apli-tuden Shift Keying) verwendet, d.h. beim Wechsel zwischen einer logischen 1 und einer 0 wechselt der Träger zwischen voller und verminderter Sendeleistung. Darüber hinaus kann der MICRF102 über einen speziellen Eingang in den Standby-Modus geschaltet werden, bei dem er weniger als 40

Nanoampere Strom aufnimmt. Von dieser Möglichkeit habe ich im AVR-Programm wo immer möglich Gebrauch gemacht.

Der AVR

Es stehen zwei AVR-Typen zur Auswahl: Der AT90xx2343 und der ATtiny12. In unserem Fall können wir diese beiden AVR-Typen als pinkompatibel betrachten. Intern haben die beiden AVRs aber einige erhebliche Unterschiede.

Der ATtiny12 ist die deutlich bessere Wahl. Er ist billiger, hat eine etwas geringere Stromaufnahme und vor allem ist der von uns verwendete interne RC-Oszillator beim ATtiny12 viel stabiler und weniger spannungsabhängig als der vom AT90xx2343. Der günstigste Tiny-Subtyp ist der ATtiny12L-4SI (2,7...5,5V - 4MHz).

Wer erwägt, ein eigenes Programm im Bascom-AVR-Basic zu schreiben, *muß* den AT90xx2343 verwenden, denn Bascom benötigt bislang AVRs mit SRAM, und der ATtiny12 hat keinen. Es sollte möglichst der AT90LS2343 (2,7...6,0V) anstatt des AT90S2343 (4,0...6,0V) verwendet werden.

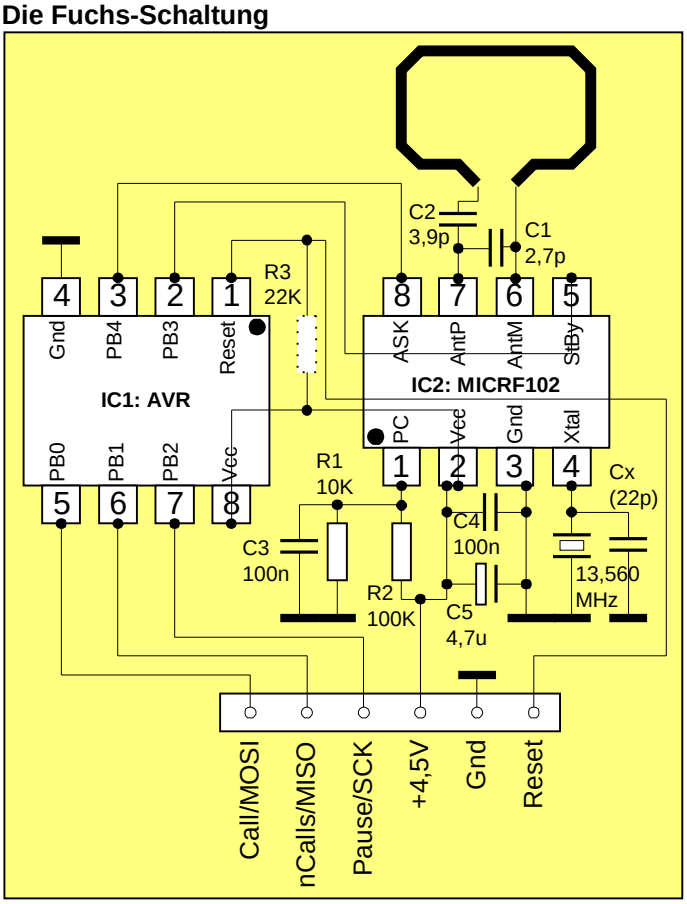
Das AVR-Programm habe ich in Atmel-Assembler geschrieben, weil es so mit nur zwei kleinen Änderungen für beide AVR-Typen compiliert werden kann. Den ausführlich kommentierte Quellcode und die fertig compilierten Dateien gibt es auf meiner Homepage (www.rowalt.de).

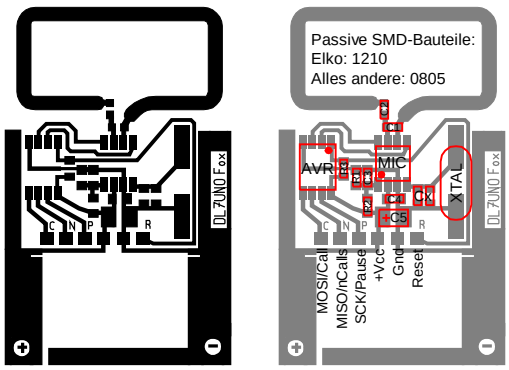
Die Stromversorgung

Ich habe mich für 3 Knopfzellen vom Typ LR44 (Alkaline) entschieden, bei denen ein Doppelpack für etwa 50 Cent zu haben ist. Die LR44-Zellen haben eine Kapazität von etwa 120mAh und geben zusammen 4,5Volt. Die Stromaufnahme des Fuchses bei 4,5V beträgt beim Senden des Tones 15,3 mA, bei den Pausen innerhalb der Morsezeichen 7,9 mA und in allen anderen Pausen 2,5 mA. Wird als Signal z.B. 5 mal MOS (dauert etwa 14 Sekunden) und anschließend 30 Sekunden Pause gewählt, dann beträgt die Stromaufnahme durchschnittlich 5,6mA. Falls ich mich nicht verrechnet habe, würde eine Batterieladung unter diesen Bedingungen für 21 Stunden Dauerbetrieb ausreichen.

Die Schaltung ist ab 3,6 Volt aufwärts betriebsfähig und ich habe auch einen 20minütigen "Crashtest" mit 7 Volt erfolgreich gewagt. Die kleine 6-Volt-Batterie 4LR44 (D=13mm/H=25,2mm/105mAh) wäre also ebenfalls verwendbar, beult die Streichholzschachtel aber etwas aus...

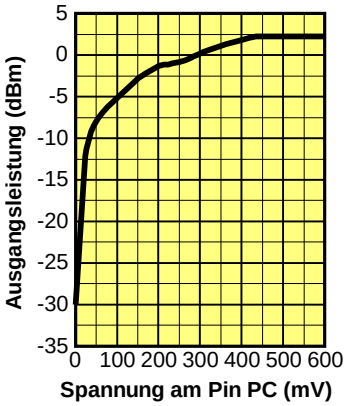
Die Fuchs-Schaltung



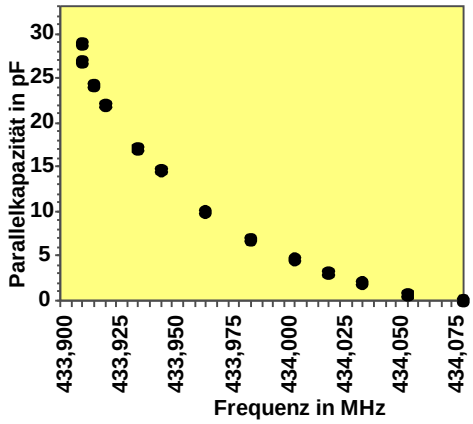


Die Platine ist 30mm x 48mm groß

Die Beschaltung des MICRF102 orientiert sich an der Hersteller-Applikation. Die Antenne ist eine geätzte Schleife mit etwa 3cm Umfang auf der Platine. Ich war überrascht, daß man den Fuch mit dieser Antenne noch in gut einem Kilometer Entfernung aufnehmen konnte (allerdings bei Sichtverbindung). Die Antenne sollte exakt die Abmessungen wie im Platinenlayout haben, denn anderenfalls muß C1 und C2 verändert werden, damit die IC-interne Antennen-Nachstimmung funktioniert. Das Datenblatt beschreibt, wie andere Antennen angepaßt werden können (siehe www.micrel.com).

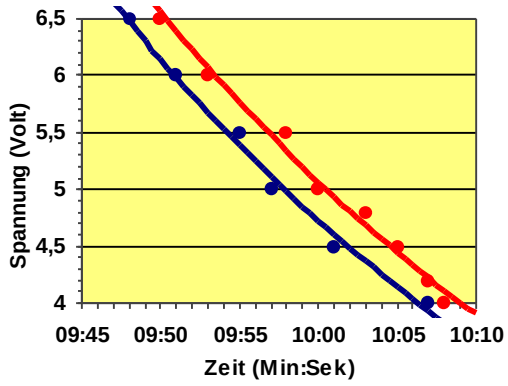


Am Pin PC bilden R1 und R2 einen Spannungsteiler, mit dem die Ausgangsleistung eingestellt wird (siehe Diagramm). Bei mehr als 0,35 Volt am Pin PC setzt eine interne Leistungsbegrenzung ein und die Ausgangsleistung wird nicht weiter erhöht. Ich habe den Spannungsteiler so gewählt, daß bei 4,5 Volt nach Möglichkeit die gesamte Batterielebensdauer über die maximal mögliche Leistung abgegeben wird. Eventuelle geringe Verluste bei der internen Leistungsbegrenzung habe ich in Kauf genommen. Wer eine andere Betriebsspannung verwendet, sollte unbedingt den Spannungsteiler verändern.
Hier soll noch auf einen kleinen Widerspruch im Micrel-Datenblatt hingewiesen werden: Der Text im Datenblatt gibt 0,35 Volt als Einsatzpunkt der Begrenzung an. Folgt man aber dem Diagramm (nicht dem Text), dann erhält man die volle Ausgangsleistung, wenn der Spannungsteiler auf etwa 0,45 Volt eingestellt wird. Wenn man davon ausgeht, daß die Batterie bei fast entladenem Zustand 3,5 Volt hat, dann wäre R1 auf dieser Grundlage mit 12K (statt 10K wie im Schaltplan) eigentlich besser bemessen.

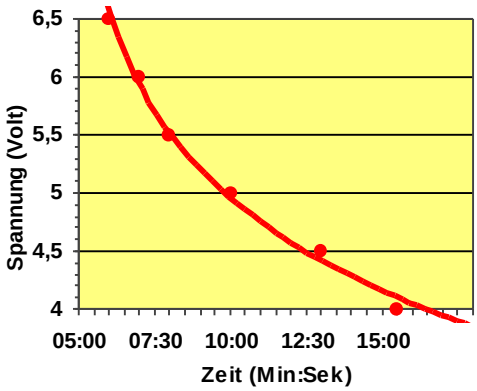


Zum Quarz war über die Schwingfrequenz hinaus leider keine Zusatzinformation verfügbar. Ich bin deshalb davon ausgegangen, daß der Quarz standardmäßig mit einer Parallelkapazität ("Bürde") betrieben wird. Das Diagramm links zeigt meine Meßergebnisse. Bei Kapazitäten ≥30pF schwang der Quarzoszillator nicht mehr an, exakt 433,92MHz (die Mittenfrequenz des ISM-Bereichs) wurde bei 22pF erziehlt. Ohne Parallelkapazität betrug die Sendefrequenz 434,080MHz (Achtung: Der ISM-Bereich ist 433,050 - 434,790MHz, das sind also 10KHz außerhalb des ISM-Bandes.).
Mein Leiterplattenentwurf sieht zur Quarzabstimmung Lötplätze für 2 "Fest"-Kondensatoren vor (Im Schaltbild als Cx bezeichnet), da SMD-Trimmkondensatoren recht teuer und schwer beschaffbar sind. Außerdem beginnt ein Trimmkondensator nie bei 0pF (Beispiele: 5...20pF und 7...30pF) und gerade im niedrigen pF-Bereich findet ja die größte Frequenzänderung statt. Ich habe zur Frequenzabstimmung ein Handfunkgerät als Empfänger verwendet und verschiedene Kondensatoren mit einem Plastikstab zur Probe gegen die Leiterplatte gedrückt. Aber so genau muß man eigentlich nur arbeiten, wenn lediglich Empfänger mit festen Kanälen zur Verfügung stehen.

Zum AVR wurde bereits einiges gesagt. Als einziges externes Bauelement wird beim ATtiny12 ein PullUp-Widerstand (R3) für den Reset-Eingang benötigt. Beim AT90xx2343 ist dieser Widerstand nicht erforderlich, da er dort bereits integriert ist.
Der AVR wird mit dem internen 1-MHz-RC-Oszillator betrieben, weshalb kein externer Quarz benötigt wird. Für die Morsesignale reicht ein RC-Oszillator ohne Frage aus, bei der Länge der Sendepausen merkt man aber doch die Ungenauigkeiten, die u.a. von der Betriebsspannung und -temperatur abhängen. Der RC-Oszillator des ATtiny12 erwies sich als erheblich besser, als der des AT90xx2343. Die folgenden beiden Diagramme zeigen das an einem Beispiel.



Theoretisch 10 Minuten beim ATtiny12 in Abhängigkeit von der Betriebsspannung. Roter Graph: 24°C, blauer Graph: 4°C



Theoretisch 10 Minuten beim AT90xx2343 in Abhängigkeit von der Betriebsspannung bei 24°C

Der AVR-Ausgang PB4 gibt die Morsesignale als hörbaren Ton von etwa 1KHz aus (wer will, kann das mit einem Piezo-Speaker oder sehr hochohmigen Kopfhörer kontrollieren).
Der Ausgang PB3 steuert den Standby-Modus des MICRF102. Während der Ausgabe eines Morsebuchstabens führt er durchgängig +5V und in den Pausen 0V.
Die Eingänge PB0 bis PB2 haben eine Doppelfunktion: Zum einen werden sie als Programmiereingänge verwendet (dazu weiter unten) und zum zweiten kann man hier drei Taster nach Masse für die Betriebseinstellungen anschließen. Die Verwendung von Tastern würde ich aber fast als Luxusvariante bezeichnen; in der Praxis waren sie eher unnötig und platzraubend. Die Halb-Luxusvariante ist eine zweite Platine mit einem Stecker und den drei Tastern, die man nur einmal für alle Füchse vorrätig hat. Die einfachste und in der Praxis tatsächlich ausreichende Variante ist ein Stück starrer Draht. Das eine Ende hält man an Masse und mit dem anderen Ende tippt man den gewünschten Anschluß an. Das ist sogar betriebssicherer, weil so in der Hektik der Fuchsjagd nichts versehentlich verändert werden kann.
Die Programmier-/Tasten-Anschlüsse wurden als einreihige SIL-Buchsenleiste ausgeführt, wie ich sie schon in meinen Bascom-Beiträgen (FA 4/2002) beschrieben hatte. Eine halbe 6polige IC-Fassung tut genauso ihren Zweck. Die Beine wurden abgekniffen und der verbliebene Rest kurz auf die Leiterplatte gelötet. Das ergibt eine mechanisch und elektrisch sehr sichere und praktikable Verbindung.
Für die Batteriehalterung wird ein rechteckiges Loch 29mm x 12mm in die Leiterplatte gesägt. Die LR44-Zellen haben einen Durchmesser von 11,6mm und eine Höhe von 5,4mm. Bitte daran denken, daß die Zellen seitlich nicht in Berührung mit Leiterbahnen kommen dürfen. Ein fester vertikal ins Loch gelöteter Bronze- oder Kupferstreifen bildet den Minus-Kontakt und ein eingebogener federnder Streifen den Plus-Kontakt. Die drei Zellen haben im Loch einen sehr guten Halt, insbesondere dann, wenn sie zusätzlich einlagig (das reicht!) mit Isolierband umklebt werden.

Programmieren des AVRs
Der AVR wird erst in der fertigen Schaltung programmiert. Dazu steckt man den Programmieradapter (siehe AVR-Einstiegsserie im FUNKAMATEUR 4+5/2002 und www.rowalt.de) auf die Platine und brennt das Programm mit WinAVR. Tiefere AVR-Kenntnisse sind dafür nicht erforderlich. Ansonsten verweise ich auf meine AVR-Serie, dort ist das genaue Vorgehen mit WinAVR genau beschrieben.

"Hidden Functions"
Für Meßzwecke oder "untypische" Anwendungen habe ich erweiterte Funktionen implementiert. Für diese Funktionen muß der EEPROM manipuliert werden. Auch dafür sind keine tieferen AVR-Kenntnisse erforderlich; man verändert einfach drei genau definierte Bytes in der EEPROM-Datei und brennt ***nur*** den EEPROM (also nicht den Flash-Speicher, der das Programm enthält).
Auf diese Weise kann man den Fuchs dazu bringen, lediglich einen 1KHz-Dauerton zu senden oder nur einen Träger. Außerdem kann die Anzahl der Rufe zwischen den Pausen auf bis zu 258 gesetzt werden und die Pausenlänge kann in 10-Sekunden-Schritten auf Werte zwischen 60sec...42min eingestellt werden. Das ist im Kommentar zum Quelltext genau beschrieben.

Gesetzliches
Abgesehen von den Vorgaben des Gesetzgebers sollte man als lizenzierter Funkamateur den Newcomern die Möglichkeit gönnen, endlich auch einmal Fuchsjagden durchführen zu können. Auch dann, wenn das ISM-Band im 70cm-Bereich liegt. Viele Lehrer und Sozialarbeiter werden Ihnen dankbar sein und außerdem sind Fuchsjagden eine ganz andere Sache als "ISM-Quatschfunk".
433MHz-ISM-Geräte gehören zu Klasse "1e" und dürfen in allen Mitgliedstaaten der EU grundsätzlich ohne Einschränkung (anmelde- und gebührenfrei) in Betrieb genommen und betrieben werden. Wer den Fuchs unter diesen Bedingungen weitergeben will, muß folgendes beachten:

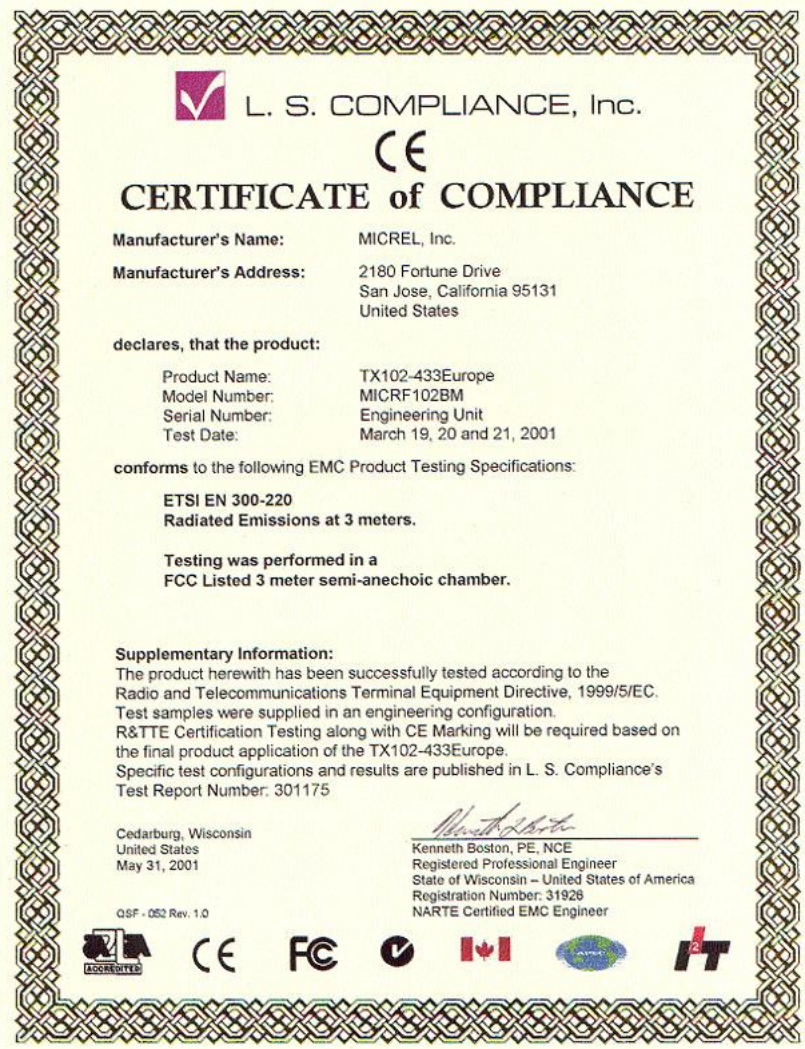
1. Der Fuchs wurde von einem Fachmann (lizenzierter Funkamateur) korrekt aufgebaut und getestet. Die Sendefrequenz muß im ISM-Bereich 433,050 - 434,790MHz liegen und die effektiv abgestrahlte Leistung muß <10mW betragen.
2. Direkt am Fuchs wird ein CE-Kennzeichen mit Jahreszahl angebracht, eine Typenbezeichnung und der Name des Herstellers.
3. Zusammen mit dem Fuchs wird eine EG-Konformitätserklärung übergeben (siehe unten) und eine Bedienungsanleitung mit "Angaben für den bestimmungsgemäßen Betrieb".
4. Der Nutzer führt keine technischen Veränderungen am Fuchs durch.

CE

Das sind die zentralen Bedinungen, wie ich sie auch mit anderen Fachleuten lange diskutiert habe. Das CE-Kennzeichen bringt man in eigener Verantwortung an, wenn man mit geeigneten Meßmitteln sicherstellen kann, daß die Fuchs-Parameter den Vorschriften entsprechen. Der Fuchs muß also nicht von einem Spezial-Labor oder einer Behörde geprüft werden.
Eine Leistungsmessung erübrigt sich im Prinzip, weil der MICRF102 konstruktionsbedingt noch nicht einmal in die Nähe von 10mW gelangen kann. Aufpassen muß man allerdings bei der Frequenz

abstimmung, weil der Quarz ohne Parallel-Kapazität eine Sendefrequenz außerhalb des ISM-Bereiches erzeugt (dazu siehe Grafik oben).
Statt des eigenen Namens sollte auch das Amateurfunkrufzeichen ausreichen. Die Angaben können auch auf die Leiterseite der Platine geätzt werden.
Unter "bestimmungsgemäßer Betrieb" ist manchmal ziemlicher Unfug zu finden. Beispielsweise könnte ein Wildschwein den Sender verschlucken, daran elend sterben und ein Tierschützer könnte davon einen schweren Schock bekommen u.s.w. Man gebe also angemessene und sinnvolle Hinweise. Das Verbot von Umbauten ("tuning") wäre z.B. ein solcher Hinweis.
In der EG-Konformitätserklärung nach Anhang II des EMVG bescheinigt der Hersteller die Übereinstimmung des Fuchs-Senders mit den Vorschriften des EMVG. Laut RegTP ist europaweit eine verkürzte Konformitätserklärung zulässig. Bei uns könnte sie wie folgt lauten:

EG-Konformitätserklärung
Hiermit erklärt [Name des Herstellers], daß sich der Fuchsjagd-Sender "DL7UNO-Mini-Fuchs" in Übereinstimmung mit den grundlegenden Anforderungen und den anderen relevanten Vorschriften der Richtlinie 1999/5/EG befindet. Die ausführliche Beschreibung des Fuchsjagd-Senders wurde in der Zeitschrift FUNKAMATEUR, Heft 08/2002 und 09/2002 veröffentlicht.



L. S. COMPLIANCE, Inc.

CE

CERTIFICATE of COMPLIANCE

Manufacturer's Name: MICREL, Inc.

Manufacturer's Address: 2180 Fortune Drive
San Jose, California 95131
United States

declares, that the product:

Product Name: TX102-433Europe
Model Number: MICRF102BM
Serial Number: Engineering Unit
Test Date: March 19, 20 and 21, 2001

conforms to the following EMC Product Testing Specifications:

ETSI EN 300-220
Radiated Emissions at 3 meters.

Testing was performed in a
FCC Listed 3 meter semi-anechoic chamber.

Supplementary Information:
The product herewith has been successfully tested according to the Radio and Telecommunications Terminal Equipment Directive, 1999/5/EC.
Test samples were supplied in an engineering configuration.
R&TTE Certification Testing along with CE Marking will be required based on the final product application of the TX102-433Europe.
Specific test configurations and results are published in L. S. Compliance's Test Report Number: 301175

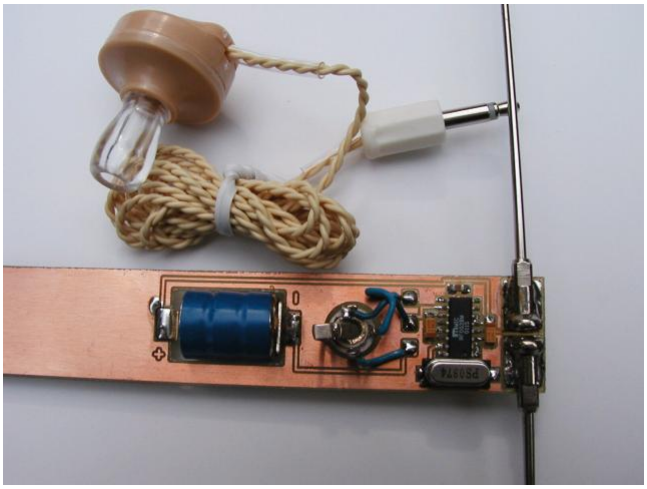
Cedarburg, Wisconsin
United States
May 31, 2001

Kenneth Boston, PE, NCE
Registered Professional Engineer
State of Wisconsin – United States of America
Registration Number: 31926
NARTE Certified EMC Engineer

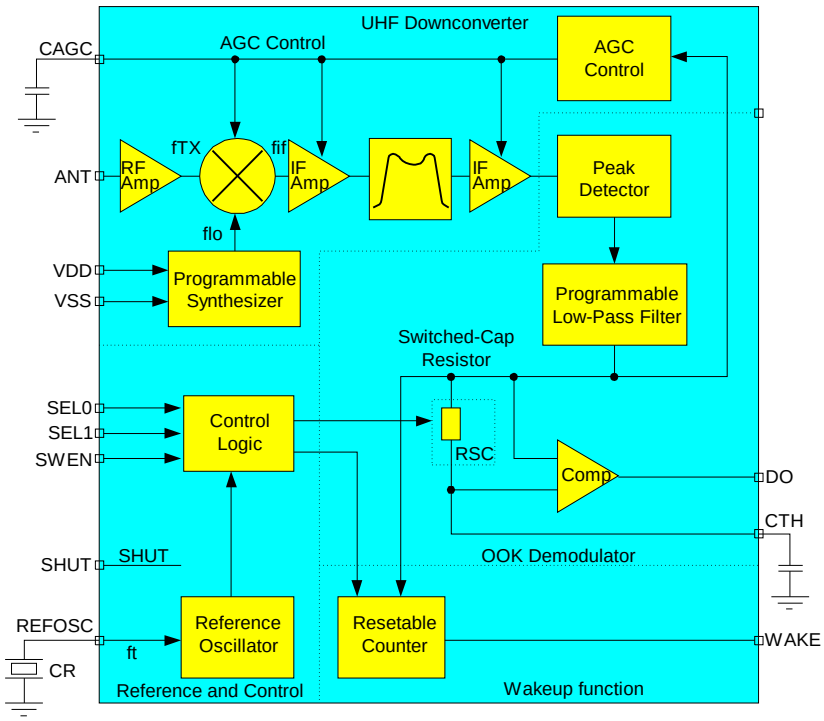
QSF - 052 Rev. 1.0

Die FCC-Erklärung zum MICRF102 (siehe links) wurde von der Micrel-Homepage heruntergeladen.
In diesem Dokument wird die Übereinstimmung mit der entscheidenden europäischen Norm ETSI 300-220 erklärt.
Alles andere müßte durch diesen Artikel bereits erklärt worden sein.

Einfacher Fuchsjagdempfänger fürs 70cm-Band



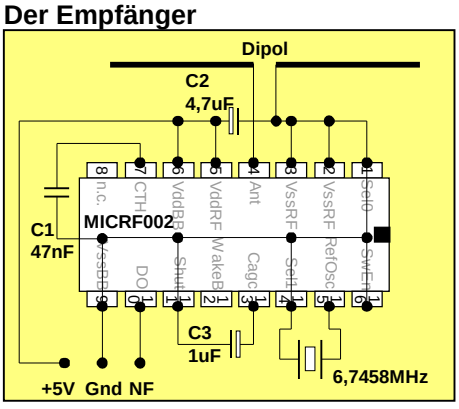
In diesem Beitrag wird ein sehr einfacher Peilempfänger für Fuchsjagden im 70cm-Band vorgestellt. Der Empfänger ist in erster Linie als Gegenstück für den kleinen Fuchs-Sender gedacht, den ich im Funkamateur 8/2002 vorgestellt hatte.
Vom eingesetzten Empfänger-IC sollte man keine Wunder erwarten. Seinen Zweck erfüllt er gut, aber mehr auch nicht. Den Schaltungsentwurf habe ich deshalb konsequent einfach und preiswert gehalten. Die Empfangsfrequenz ist fest auf den 70cm-ISM-Bereich eingestellt. Eine Frequenzabstimmung ist nicht vorgesehen, da der Empfänger breitbandig genug ist, um den gesamten ISM-Bereich gleich gut zu empfangen. Zusammen mit dem 1,8mW-Fuchssender, den ich im letzten Heft vorgestellt hatte, kommt man auf eine maximale Reichweite von etwa 350 Metern auf freiem Feld. Der ideale Fuchsjagd-Ort ist also die Parkanlage. Bei Laubwald mit dichtem Unterholz ist mit einer maximalen Reichweite von etwa 200 Metern zu rechnen.



Der Empfänger besteht im Wesentlichen nur aus dem Micrel-IC MICRF002BM (SMD-Gehäuse) bzw. MICRF002BN (DIL-Gehäuse). Hinzu kommen ein Quarz und drei Kondensatoren. Das ist alles.

Für den MICRF002 wird eine Betriebsspannung von 4,5 bis 5,5V angegeben. Maximal sind 7 Volt zulässig. Bei normaler Betriebsspannung und 433MHz nimmt der IC laut Datenblatt knapp 4,5 mA Strom auf (gemessen: 2,4 mA). Die gemessene untere Grenze für einen sicheren Betrieb war 3,9 Volt.
Zur Stromversorgung werden wie beim Mini-Fuchs im vorangegangenen Artikel 3 LR44-Knopfzellen eingesetzt.

Der Empfänger



Der MICRF002 ist ein Einfach-Superhet. ZF-Frequenz und ZF-Bandbreite sind abhängig vom eingesetzten Quarz. Bei unserer Quarzfrequenz von 6,7458MHz beträgt die ZF-Frequenz 1,184MHz und die 3dB-Bandbreite des ZF-Filters ist die Hälfte davon, also etwa 592KHz. Die Berechnung ist im Datenblatt (siehe www.micrel.com) zu finden. Das ZF-Filter ist im Chip integriert als Bandpaß 5. Ordnung. Da Selektionsmaßnahmen am Empfängereingang fehlen, wird die Spiegelfrequenz natürlich genauso gut wie die gewünschte Frequenz empfangen.
Die hohe Bandbreite und die Spiegelfrequenz sind der Preis, den man bei einem so einfachen Empfänger zahlen muß. Micrel's Firmenslogan "The Infinite Bandwidth Company™" (etwa: "Die Firma der unendlichen Bandbreite") ist also gar nicht so übel gewählt. Die Empfindlichkeit des Empfängers bestimmt sich in diesem Fall nicht aus dem thermischen, sondern dem atmos-

phärischen Rauschen, das man sich mit jedem KHz Bandbreite einfängt. Ein einfaches LC-Filter am Antenneneingang reicht aber nicht zur Steigerung der Empfindlichkeit. Damit könnte man lediglich Störungen durch elektrische Geräte o.ä. vermindern, was im Gelände aber gerade nicht das Problem ist. Ein getestetes SAW-Filter dämpfte zwar die (nicht sonderlich störenden) Nachbarsignale, verminderte die Empfindlichkeit aber doch erheblich. Ich entschied mich daher, alle zusätzlichen Selektionsmaßnahmen wegzulassen und beim Einfach-Konzept zu bleiben.
Der Demodulator hat ein Tiefpaßfilter 2. Ordnung, dessen Bandbreite mit den IC-Pins SEL0 und SEL1 auf 1,25KHz, 2,5KHz, 5KHz oder 10KHz eingestellt wird. Ich habe die niedrigste Bandbreite gewählt.
Mit C3 wird die Zeitkonstante für die automatische Verstärkungsregelung (AGC - Automatic Gain Control) festgelegt. Fürs Peilen wäre eine Handregelung an sich besser gewesen. Aber da dies nun einmal nicht ging, mußte ein Kompromiß gefunden werden. Eine sehr schnelle Regelung würde das Peilen sehr erschweren und eine sehr langsame Regelung stiehlt Zeit. Irgendwo dazwischen fand sich mit 1 uF ein recht ordentlicher Kompromiß. Der AGC-Kondensator sollte ein Tantal-, Keramik- oder Polyester-Typ sein.

Empfänger: VFO
Der VFO ist als programmierbarer Synthesizer ausgelegt, dessen Frequenz in unserem Fall das 64,5fache der Quarzfrequenz ist. Zum Wort "programmierbar" folgendes: Beim MICRF002 können über den Pin SWEN zwei grundsätzlich verschiedene Betriebsmodi gewählt werden. Zum einen kann der Empfänger als ganz normaler Superhet mit fester Empfangsfrequenz arbeiten, und zum anderen als Wobbel-Empfänger für den Betrieb zusammen mit primitiven LC-Sendern. Die Betriebsart wird also mit dem Lötkolben "programmiert". Wir verwenden den Festfrequenz-Modus und nur auf diesen werde ich mich beziehen.
Als mittlere Empfangsfrequenz verwenden wir 433,92MHz (ISM-Band). Zur Ermittlung der Quarzfrequenz geht man so vor:
Zuerst wird die VFO-Frequenz berechnet. Es gibt zwei mögliche Frequenzen, da der MICRF002 ein Superhet ist. Die Formel ist laut Datenblatt $f_{vfo} = f \pm 1{,}064 * (f / 390)$
Für 433,92MHz als Empfangsfrequenz erhalten wir die folgenden beiden VFO-Frequenzen:
VFO-Frequenz 1: 433,92MHz + 1,064 * (433,92MHz / 390) = 435,093MHz
VFO-Frequenz 2: 433,92MHz - 1,064 * (433,92MHz / 390) = 432,747MHz
Jetzt sucht man sich eine der beiden VFO-Frequenzen aus. Die höhere VFO-Frequenz ist hier die bessere, damit man möglichst weit weg von störenden Checker-Signalen ist. Die gewählte VFO-Frequenz geteilt durch 64,5 ergibt die Quarzfrequenz: 435,093MHz / 64,5 = 6,7456MHz
Den 6,7456MHz-Quarz gibt es für etwa 1,50 Euro bei Dacom Süd sowohl in SMD- als auch in bedrahteter Ausführung (HC18-flat). Der Quarz braucht aufgrund der hohen Empfängerbandbreite nicht abgestimmt zu werden. Mehr noch, der IC sieht das gar nicht vor, denn der interne Oszillator hat bereits eine 30pF-Kapazität integriert.

Der NF-Zweig
Der NF-Ausgang, eigentlich ein Datenausgang, wurde von Micrel nur für CMOS-Lasten ausgelegt. Man hätte also zumindest einen kleinen Transistor-Verstärker oder einen entsprechenden IC wie den LM386 anschließen müssen. Den IC verwarf ich sofort, weil ich einen unnötigen Stromverbrauch vermeiden wollte.

Beim Durchkramen der Bastelkisten fiel mir dann ein Kristall-Ohrhörer in die Hände. Solche Ohrhörer sind sehr hochohmig und so empfindlich, daß man ein Brummen hört, sobald man einen Eingang anfaßt und den anderen erdet.
So richtig glaubte ich trotzdem nicht an den Erfolg, denn der Ausgangsstom des Datenausgangs ist im Datenblatt mit typisch 10µA angegeben und 10µA * 5V macht 50µW. Aber da ich nicht dumm sterben wollte, machte ich den Versuch. Und siehe da: Es funktionierte und die Lautstärke war sogar mehr als ausreichend. Ich habe den Test vorsichtshalber bei mehreren ICs, bei höheren Betriebsspannungen und über längere Zeit wiederholt.
Kristall-Ohrhörer stammen eigentlich noch aus der Zeit der Miniröhren-Hörgeräte (weiland auch Schmalzbohrer genannt), haben auch das Design der Nierentisch-Ära, werden aber im Gegensatz zu Nierentischen nach wie vor hergestellt. Bei Segor (www.segor.de) und www.fau-lehrmittel.de gibt es Ausführungen mit 3,5mm-Klinkenstecker und einer Impedanz von 100KΩ, die ich auch getestet und für hervorragend befunden habe.
Zur Sicherheit noch einmal ganz deutlich der Hinweis, daß der IC-Ausgang *nicht* für niederohmige Lasten ausgelegt ist. Ohrhörer aus der Bastelkiste können auch dynamisch sein und haben dann eine

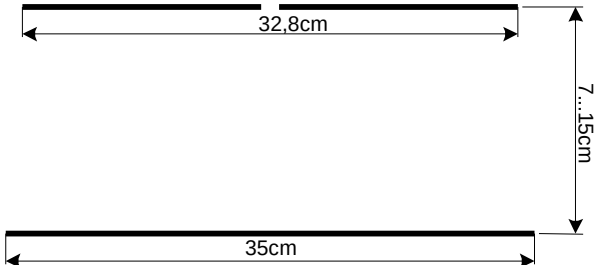
viel zu geringe Impedanz. Man nehme also wirklich nur Kristall-Ohrhörer und nichts anderes.

Es wird eine 3-polige Ohrhörer-Buchse verwendet, die gleichzeitig als Einschalter dient. Dazu muß der dritte Kontakt ein Schließer sein. Manche Öffner lassen sich übrigens durch leichtes Verbiegen der Kontakte zum Schließer umfunktionieren (siehe Bild). Da der Ohrhörer gegen Masse betrieben wird, können wir natürlich nicht die gewohnte Plus-Leitung schalten, sondern müssen es bei der Masse-Leitung tun. Zugegeben: Ein Stecker ziehen ist unbequemer als ein Schalterdruck. Aber der Empfänger muß während der Fuchsjagd im Prinzip nicht ausgeschaltet werden, denn 120mAh Batteriekapazität reichen für 50 Stunden Dauerbetrieb. Überlegt man sich die Anzahl der Fuchsjagden pro Jahr und die Selbstentladung, dann scheint das nicht unökonomisch zu sein.

Die Antenne

Ich habe lange überlegt, welche Peilantenne am sinnvollsten wäre. Mir erscheinen zwei Antennentypen als sinnvoll:

Als erstes kommt ein Lambda/4-Stab in Frage. Bei 433,92 MHz muß der Antennenstab 16,4cm lang sein. Auf diesen Antennentyp ist der Antenneneingang des MICRF002 optimiert. Die Stabantenne (Fahrradspeiche!) ist extrem robust und vereinfacht die mechanische Konstruktion extrem. Speziell bei Fuchsjagden mit Kindern ist diese Antenne zu empfehlen. Der Nachteil ist aber, daß die Stabantenne vor- und rückwärts gleich gut peilt (dazu mehr in der Einsteigeranleitung).



Eine ordentliche, aber auch aufwendigere und mechanisch empfindlichere Wahl ist der direkt gespeiste Dipol mit Reflektor. Der große Vorteil dieser Antenne ist, daß sie neben einem höheren Gewinn vor allem ein gut unterscheidbares Vor-Rück-Verhältnis hat. Die leichte Fehlanpassung gegenüber dem Lambda-Viertel-Stab fiel nicht merkbar ins Gewicht, sodaß auf zusätzliche Anpassungsmaßnahmen verzichtet werden konnte.

Als Antennenstäbe werden Fahrradspeichen mit den zugehörigen Messing-Muttern verwendet. Die Muttern für den Dipol werden auf einer Seite flachgefeilt und direkt auf die Platine gelötet. Als Gehäuse kann ein Rohr dienen oder man fertigt sich ein entsprechendes Gehäuse aus kupferkaschiertem Halbmaterial. Im einfachsten Fall ist es auch beim Dipol mit Reflektor möglich, die Platine als Antennenträger zu nehmen.

Ein Dipol mit unsymmetrischer Speisung "schielt" ein wenig, aber das spielt bei unseren 350 Metern Reichweite kaum eine Rolle. Die Länge des Dipols in Metern ist 142500/KHz, bei 433,92MHz also 32,8 cm. Sehr kritisch ist dieser Wert bei Empfangsantennen nicht.

Der Abstand zwischen Dipol und Reflektor hat Einfluß auf die Empfindlichkeit der Antenne, aber hier kann man in gewissen Grenzen Pragmatismus walten lassen: Bei kürzerem Abstand zwischen Dipol und Reflektor wird die mechanische Konstruktion viel einfacher und auch stabiler, aber dafür wird die Antenne ein ganz kleines bißchen unempfindlicher. Im Bereich von 0 bis etwa 6,5cm Abstand zwischen Dipol und Reflektor steigt die Antennengewinn (die Empfindlichkeit) sehr steil an. Es wäre also unklug, einen Abstand in diesem Bereich zu nehmen. Ab etwa 7,5cm oder 8cm aufwärts steigt der Antennengewinn nur noch sehr langsam an. Daher lohnt es sich ***aus praktischen Gründen*** (alte Ingenieurweisheit: Nicht so gut wie möglich, sondern so optimal wie möglich), als Abstand zwischen Dipol und Reflektor 7,5...8cm zu wählen. Der genaue Abstand ist abhängig von Umgebungseinflüssen, also z.B. der Platine samt Batterie und natürlich auch der Hand, welche die Antenne festhält. Wer Spaß daran hat, kann den Antennengewinn optimieren, indem man sich so weit vom Sender entfernt, daß dieser gerade noch so zu hören ist. Dann verschiebt man den Reflektorabstand so, daß der Sender etwas besser zu empfangen ist, vegrößert wieder den Abstand zum Sender u.s.w. Man wird feststellen, daß sich beim Verschieben (relativ flache) Maxima zu finden sind. Auf diese Weise kann man unter Umständen noch etwas herausholen, aber wie gesagt: Für unseren Zweck bringt das weniger, als man denken mag.

Die Bedienung

Wie oben bereits erwähnt ist eine Frequenzeinstellung unnötig und eine Absimmung des Quarzes bringt auch nichts. Ebenso ist eine Lautstärkeeinstellung nicht erforderlich. Die Betriebsspannung wird angelegt, indem man den Ohrhörer in die Klinkenbuchse steckt. Zusammengefaßt: Es gibt außer der Antennendrehung keine Bedienung im eigentlichen Sinne.

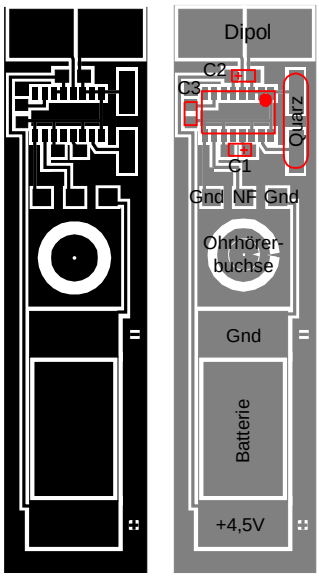
Hier noch einige Tips zur Peilung, die sich bei meinen eigenen Tests ergeben haben. Bei größerer Entfernung zum Fuchs (sagen wir mehr als 30 Meter) macht man eine Maximumpeilung. Dazu muß nichts weiter gesagt werden. Darunter ist aber eine Minimum-Peilung günstiger. Durch langsam-rhytmisches Drehen der Antenne um vielleicht 10-20° läßt sich so die Richtung zum Fuchs trotz AGC

gut feststellen. Auf diese Weise bekommt man den Fuchs bis auf etwa 1,5 Meter eingekreist, ohne daß der Empfänger abgeschirmt sein muß.

Fazit

Der Nachteil des einfachen Empfängers ist klar: Er ist recht unempfindlich und hat die Selektionsfähigkeit eines Scheunentors. Auf der Haben-Seite schlägt aber eindeutig der geringe Preis und die Einfachheit im Aufbau und in der Bedienung zu Buche. Das macht die Sache unaufwendig und unbürokratischer und man erreicht mehr Leute. Die geringe Reichweite wird außerdem etwas durch die geringe Baugröße der Füchse ausgeglichen - man sieht sie eben nicht schon in -zig Metern Entfernung. Die fehlende Frequenzselektion der Empfänger gleicht sich zumindest etwas aus, weil durch die geringe Reichweite die Peilwinkel zwischen den verschiedenen Füchsen größer werden.
Zusammengefaßt: Der Empfänger ist als einfaches und billiges Projekt vor allem für Einsteiger und die Arbeit mit Kindern und Jugendlichen eine feine Sache, die auch funktioniert. Aber er ist eben keine Konkurrenz zu hochwertigen Geräten. Für größere Reichweiten bei gleichem Sender wären z.B. LPD-Funkgeräte mit abgesetzter Peilantenne eine echte Alternative.

Den kleinen Fuchs-Sender würde ich dagegen ohne Abstriche als praktikabel bezeichnen. Vor allem ermöglicht der geringe Preis ganz andere Varianten der Fuchsjagd.
Beispielsweise macht folgende Variante Spaß: Man verstreut möglichst viele Füchse im Ge-lände und läßt sie von den Teil-nehmern wie Ostereier einsammeln. Gefun-dene Füchse müssen sofort ausgeschaltet werden. Erwischt ein Teilnehmer einen ander-en mit einem sendenden Fuchs in der Tasche, dann gehören ihm alle Füchse, die dieser be-reits eingesammelt hat. Wenn Ruhe im Äther ist, läßt man zur Zählung antreten und hängt bei Bedarf noch ein Stechen an. Die gemein-ste Variante ist die Suche in Gebäuden, weil dann noch Fehlpeilungen durch Reflektionen und 1001 Möglichkeit zum Verstecken hinzukommen.



Hier ist das Platinenlayout des Empfängers. Eprechend verlängert kann die Platine auch gleich als Träger für den Reflektor mitgenutzt werden. Statt des Dipols kann natürlich auch eine Stabantenne angeschlossen werden.

Bauteil-Beschaffung

Auf www.micrel.com ist neben den Datenblättern eine Liste von Distributoren für den MICRF002/MICRF102 zu finden. Ich persönlich habe die ICs bei Dacom Süd gekauft, weil man dort auch gleich die passenden Quarze beziehen kann.
Der Sender-IC MICRF102BM kostet bei Dacom Süd etwa 2,70 Euro und der zugehörige 13,56MHz-SMD-Quarz etwa 1,75 Euro. Der Empfänger-IC MICRF002BM ist für etwa 6,70 Euro zu haben und der dazu passende 6,7458MHz-SMD-Quarz für etwa 1,75 Euro. In den Preisen ist die MWSt. bereits enthalten. Hinzu kommt aber noch eine Versand-Pauschale von 7,50 Euro.
Daneben hat der Einzelhändler Segor die Micrel-ICs in seinen Katalog aufgenommen. Der 13,56MHz-Quarz ist ebenfalls bei Segor zu haben (auch in SMD), allerdings nicht der 6,7458MHz-Quarz. Hier habe ich nur einen 6,75MHz-Quarz mit HC49-Gehäuse finden können. Das sind 4,2KHz mehr, aber mit einer Parallelkapazität zum Quarz müßte man das gut ausgeichen können.

Für den ATtiny12L-4SI verweise ich auf den Einzelhandel. Segor hat ihn z.B. vorrätig. Übrigens: Zeitweise gab es bei diesem IC Lieferschwierigkeiten, weil ein Druckerpatronenhersteller den ATtiny12 jüngst in großen Stückzahlen eingesetzt hat, um das Nachfüllen von Tinte zu verhindern.

Kristall-Ohrhörer sind u.U. schwer zu bekommen. Segor hat sie aber definitiv.

Dacom Süd
Dacom Süd GmbH, Freisinger Straße 13, D-85737 Ismaning, Tel: 089-964880
Dacom Süd ist als Distributor eigentlich auf große Stückzahlen spezialisiert, verkauft aber auch einzelne Exemplare.
Internet: www.dacomsued.de

Segor
Segor Electronics, Kaiserin-Augusta-Allee 94, D-10589 Berlin, Tel: 030-4399843
Ladenverkauf und Versand
Internet: www.segor.de (Katalog online und downloadbar)

Ursula-Fau-Lehrmittel
Alternative Bezugsquelle für Kristall-Ohrhöhrer
Ursula Fau, Postfach 1323, 32772 Lage/Lippe, Tel: 05232-3115
Internet: www.fau-lehrmittel.de (Katalog online)

Fuchsjagdanleitung für Einsteiger

Was ist überhaupt eine Fuchsjagd?

Ganz einfach: Es werden eine oder mehrere Sender irgendwo versteckt und die Jäger müssen diese Sender (die Füchse) mit Hilfe von Peilempfängern finden. Echte Füchse aus Fleisch un Blut sind nicht beteiligt. Ich hoffe, das beruhigt die Tierschützer.

Fuchsjagden machen mehr Spaß, als man auf den ersten Blick denken mag. Ich (und natürlich andere) haben die Erfahrung gemacht, daß dabei irgend ein tief verborgener Jagdinstinkt wieder hochkommt und den eigentlichen Reiz ausmacht. Um Spaß an der Fuchsjagd zu haben, muß man also keineswegs ein Techniker sein. Die Grundlagen sind in ein paar Minuten erklärt und dann kann man schon loslegen. Tricks, Kniffe und Strategien sind natürlich eine andere Sache...

Fuchsjagden konnten früher im Prinzip nur von lizenzierten Funkamateuren ausgerichtet werden. Das hat sich aber mit der europaweiten Liberalisierung der Gesetze geändert. Der von mir vorgestellte Mini-Fuchs arbeitet im 70cm-ISM-Band, in dem eine lizenz- und gebührenfreie Nutzung möglich ist. Kurz zusammengefaßt: Den korrekt aufgebauten Fuchs darf jedermann benutzen - auch Kinder.

Wie läuft eine Fuchsjagd ab?

Es gibt viele Varianten.

Im einfachsten Fall wird ein Sender versteckt und die Teilnehmer starten im 10-Minuten-Abstand, um ihn zu suchen. Abgerechnet wird nach Zeit.

In einer erweiterten Variante werden mehrere Füchse versteckt, die alle gefunden werden müssen. Als Beweis beschriftet man jeden Fuchs mit einem Namen oder einem Code, denn die Teilnehmer abschreiben und dann vorweisen müssen.

Hat man viele Füchse zur Verfügung, kann man auch alle Teilnehmer gleichzeitig starten lassen. Hier kann man als Variante auch die Füchse einsammeln lassen bis keiner mehr übrig ist. Die Teilnehmer schalten einen gefundenen Fuchs sofort aus (Schalter muß vorhanden sein!). Wer mit einem sendenden Fuchs in der Tasche erwischt wird, muß **alle** bereits gefundenen Füchse an den herausrücken, der ihn erwischt hat.

Der Fuchs kann auch beweglich sein und in der Tasche eines "harmlosen Spaziergängers" stecken.

Am gemeinsten sind Fuchsjagden in Gebäuden, weil hier sehr viele Funkabschattungen und Reflektionen hinzukommen. Anfänger-Fuchsjagden sollten daher eher im Gelände (Parks o.ä.) stattfinden.

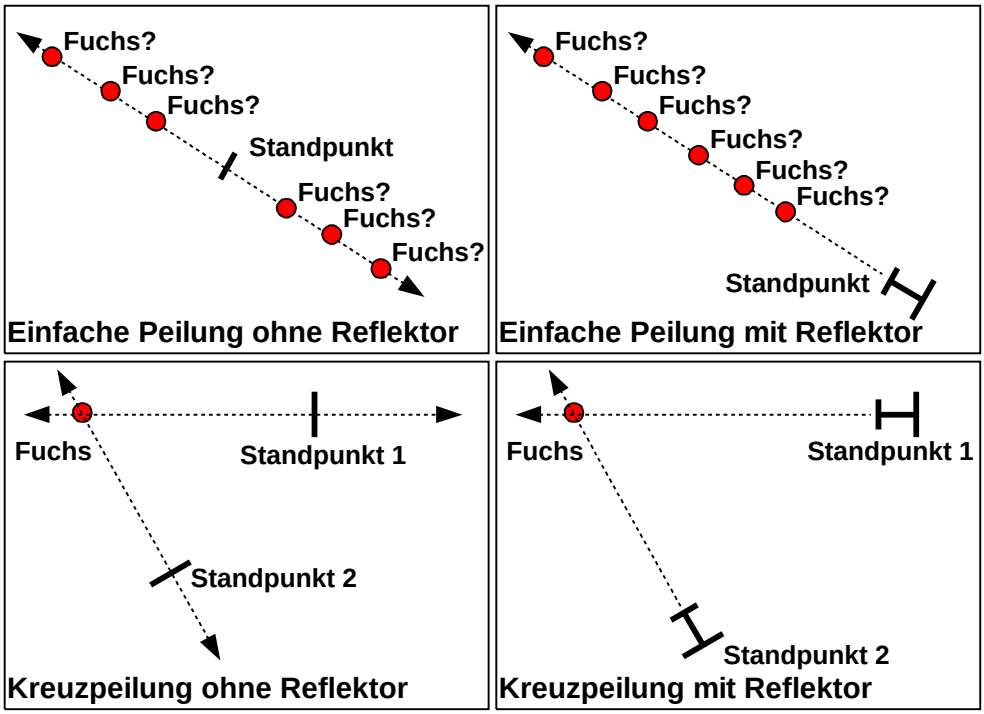
Wenn der von mir vorgestellte Simpel-Empfänger zum Einsatz kommt, sollte man übrigens keine hochwertigen Empfänger zulassen, weil die geringe Empfindlichkeit des Simpel-Empfängers den Benutzer sonst zu sehr benachteiligt. Ausnahme: Die Gelände-Grenzen sind genau festgelegt und die Füchse sind mit dem Simpel-Empfänger überall in diesem Gelände aufzunehmen.

Wie peilt man?

Fangen wir am besten bei der Antenne an. Für den direkten Anschluß an unseren Einfach-Empfänger kommen drei verschiedene Antennentypen als sinnvoll in Frage:

1. Einfacher Antennenstab (sogenannte Lambda/4-Antenne, bei 433,92MHz muß sie 16,4mm lang sein). Mechanisch extrem einfache und robuste Lösung (schnell aufzubauende Empfänger!), gute Reichweite, der Empfänger-IC des Simpel-Empfängers ist genau für diesen Antennentyp ausgelegt. Nachteil: Die Stabantenne empfängt vor- und rückwärts gleich gut. Dadurch hat man zwar die "Luftlinie" zum Fuchs ermittelt, weiß aber nicht die Richtung.
2. Einfache Dipol-Antenne (2 Lambda/4-Stäbe, einer davon an Masse). Auch der Dipol ist mechanisch einfach zu lösen. Er peilt etwas "schärfer" als die einfache Stab-Antenne und vergrößert deshalb theoretisch auch etwas die Reichweite. Die Anpassung an den Simpel-Empfänger ist aber nicht mehr so gut wie bei der Stab-Antenne. Durch die unsymmetrische Ankopplung an den Empfänger schielt der Dipol auch etwas. Der Empfang ist wie bei der Stab-Antenne vor- und rückwärts gleich gut und die Richtung zum Fuchs ist deshalb nicht feststellbar. Meine Erfahrung ist, daß der einfache Dipol kaum ***praktische*** Vorteile gegenüber der einfachen Stabantenne bringt. Die große Robustheit der Stabantenne und ihr einfacher Aufbau fällt hier schwerer ins Gewicht als alles andere.
3. Dipol mit Reflektor. Die Antenne ist im Verhältnis zum Empfänger recht groß und es ist schwieriger, sie mechanisch stabil aufzubauen. Aber die Antenne hat zwei gewichtige Vorteile: Durch den Reflektor erreicht man eine Reichweitenerhöhung und vor allem gibt es***eine*** eindeutige Peilrichtung.

Jetzt zur eigentlichen Peilung: Mit (nur) einer Peilung erhält man (nur) die Richtung, in der sich der Fuchs befinden muß. Peilt man aber von zwei oder mehreren Standpunkten aus (Kreuzpeilung), dann erhält man den Standort des Fuchses eindeutig. Bei der Kreuzpeilung hat eine Antenne ohne eindeutiges Vor-/Rück-Verhältnis an sich keinen Nachteil mehr, denn der Standort des Fuchses ist so oder so eindeutig bestimmbar.



Der Nachteil von Antennen ohne eindeutiges Vor-/Rück-Verhältnis liegt darin, daß man mit einer Wahrscheinlichkeit von 1:1 erst einmal in die falsche Richtung losläuft - und das kostet Zeit.

Trotzdem kann es sinnvoll sein, Empfänger nur mit der einfachen Stabantennen auszurüsten. Zum einen kann man so in kürzester Zeit mit geringem Aufwand viele Empfänger herstellen. Zum anderen sind die Empfänger aber auch mechanisch viel viel robuster. Und das spielt speziell bei Fuchsjagden mit Kindern eine erhebliche Rolle. Die zusätzlichen Fußwege aufgrund der nicht eindeutigen Peilrichtung fallen nur wenig ins Gewicht, da die Reichweite bei unserer Technik ohnehin nur maximal 350 Meter beträgt. Aus Gründen der Fairness sollten ***alle*** Teilnehmer nur mit Stabantenne arbeiten. Es bietet sich an, einfache Geländekarten auszugeben, auf denen die Teilnehmer ihre Peilungen eintragen können.

Wenn man eine Antenne mit eindeutigem Vor-/Rück-Verhältnis hat, läuft man nach der ersten Peilung einfach in Richtung Fuchs los. Auf dem Weg zum Fuchs peilt man immer wieder und kreist damit den Fuchs ein. Das Peilen sollte möglichst sogar im Gehen erfolgen. Es bringt in der Regel nichts, extra zu "idealen" Standpunkten für eine Kreuzpeilung zu laufen, weil man dadurch zu viel Zeit verliert.

Wellen, Schatten und andere UKW-Phänomene

Auf UKW merkt man recht deutlich, daß Funkwellen tatsächlich Wellen sind. Die verwendete Frequenz von 433,92MHz entspricht umgerechnet einer Wellenlänge von etwa 70cm. Und da eine Welle immer aus einer positiven und einer negativen Halbwelle mit je einem eigenem Berg besteht, kommt es zu dem Effekt, daß sagen wir 20cm Ortsänderung unter Umständen beim Empfang alles oder nichts bedeuten können.

Funkwellen haben wie Lichtwellen Schatten, wenn sich Objekte zwischen Sender und Empfänger befinden. Das gilt besonders für metallhaltige Objekte wie Gebäude, Zäune u.s.w., aber auch für wasserhaltige Objekte wie Hügel, nasses Unterholz o.ä..

Funkwellen können wie Lichtwellen reflektiert werden. Jeder elektrische Leiter kann als Reflektor wirken, z.B. ein Gartenzaun. Das Ergebnis kann eine Fehlpeilung sein, und zwar besonders dann, wenn sich in direkter Richtung zum Fuchs gerade ein schattenbildendes Objekt befindet (weil der Reflektor dann eventuell besser "strahlt" als der Fuchs selbst).

Und schließlich können Funkwellen auch noch durch Freileitungen fortgeleitet werden. Auch das kann natürlich zu Fehlpeilungen führen.

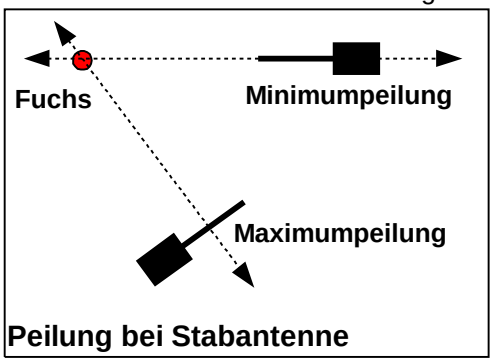
Aus den eben genannten Gründen macht es sich gut, den Empfänger ständig eingeschaltet zu lassen und im Gehen zu peilen. Auf diese Weise gewinnt man nicht nur Zeit, sondern findet nebenbei auch die besten Empfangspunkte. Außerdem fallen im Laufen Fehlpeilungen sehr schnell auf.

Anfängerfuchsjagden sollten in einem Gelände stattfinden, das möglichst frei von potentiellen Reflektoren und Wellenleitern ist. Bei Fortgeschrittenen kann aber gerade das als zusätzlicher Reiz eingesetzt werden (Fuchsjagden in Gebäuden!).

Die Peilung mit dem Simpel-Empfänger

Hochgezüchtete Peilempfänger von Funkamateuren haben Feldstärkemeßinstrumente und Dämpfungsregler am Antenneneingang. Das können und wollen wir uns beim Simpel-Empfänger nicht leisten, denn es geht auch so.

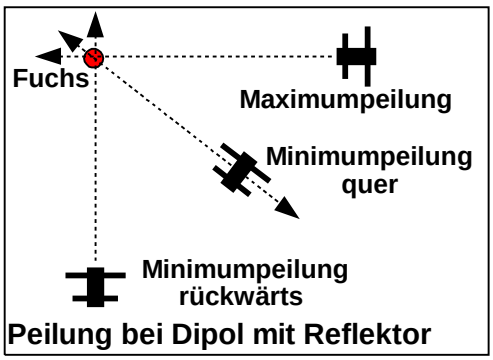
Bei unserer Peilung wird das Verhältnis des Fuchs-Signals zum UKW-Hintergrundrauschen einfach nach Gehör bestimmt. Mit der richtigen Peiltechnik geht das sehr gut.



Peilung bei Stabantenne

Zeigt die Stabantenne genau zum Fuchs, dann ist der Empfang am schwächsten und das Hintergrundrauschen wird stärker (Minimumpeilung). Zeigt die Antenne quer zum Fuchs, dann ist der Empfang am besten und das Rauschen am geringsten (Maximumpeilung).

In beiden Fällen steht die Richtung zum Fuchs erst nach der Kreuzpeilung fest - es seie denn, man kann den eigenen Körper als schattenbildendes Objekt einsetzen ;-)



Peilung bei Dipol mit Reflektor

Bei der Maximumpeilung zeigt der Reflektor (daran zu erkennen, daß er länger ist als der Dipol) vom Fuchs weg. Die Zeichnung zeigt, wie die Antenne ausgerichtet werden muß. Die Richtung stellt man fest, indem man in beide Richtungen peilt. Die Richtung mit dem besseren Empfang zeigt zum Fuchs.

Bei der Minimumpeilung, die quer zum Fuchs durchgeführt wird, ist die Richtung zum Fuchs nicht bestimmbar. Im Fuchs-Nahfeld kann sie trotzdem nützlich sein und kann durch eine Rückwärtspeilung ergänzt werden.

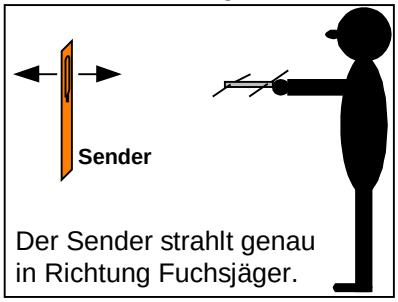
Maximumpeilungen macht man, solange man noch weit vom Fuchs entfernt ist, denn dort ist der Fuchs bei Minimumpeilung oft gar nicht mehr zu empfangen. Man schwenkt die Antenne am besten rhytmisch um die Richtung mit dem besten Empfang (geringstes Rauschen) und verkleinert allmählich dabei den Schwenkwinkel bis man die genaue Richtung hat.

So bald wie möglich setzt man die Minimumpeilung ein, weil sich die Richtung damit viel besser bestimmen läßt. In unmittelbarer Nähe zum Fuchs bringt die Maximumpeilung ohnehin keine eindeutigen Ergebnisse mehr (die Richtung läßt sich höchstens noch durch Rückwärtspeilung feststellen).

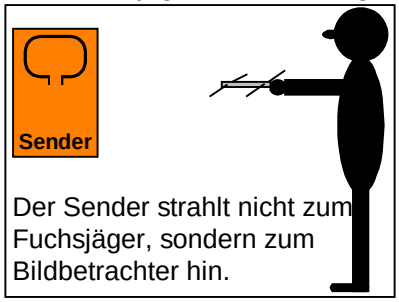
Sobald das Fuchs-Signal sehr sehr kräftig wird (weniger als 3 Meter Entfernung zum Fuchs), wird auch die Minimumpeilung schwierig. In diesem Fall hilft aber noch die AGC (Automatic Gain Control, automatische Verstärkungsregelung) des Empfängers. Diese vermindert automatisch die Empfänger-Empfindlichkeit, sobald der Sender gut empfangen wird. Es kann ausgenutzt werden, daß die Regelung mit einer festen Zeit arbeitet. Zwischen den Morsezeichen regelt der Empfänger seine Empfindlichkeit wieder hoch und es ist ein Rauschen zwischen den Morsezeichen zu hören. Die Dauer der Rauschabschnitte ist um so größer, je schwächer der Fuchs empfangen wird. Das kann auch in unmittelbarer Nähe zum Fuchs zur Beurteilung herangezogen werden.

Der Fuchs-Sender

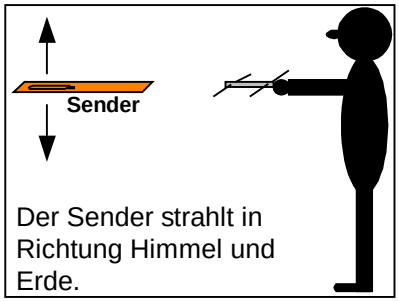
Beim Mini-Fuchs-Sender muß beachtet werden, daß er zwei Hauptabstrahlrichtungen hat. Die Strahlungsrichtung läßt sich am besten anhand einiger Zeichnungen erklären. Stellen wir uns dazu vor, wir hätten einige Fotos von einem Fuchsjäger beim Peilen gemacht:



Der Sender strahlt genau in Richtung Fuchsjäger.

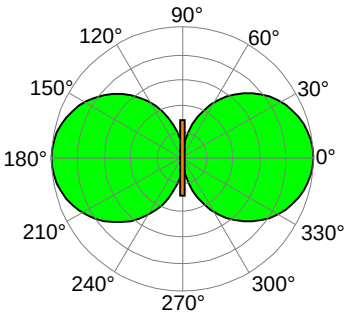


Der Sender strahlt nicht zum Fuchsjäger, sondern zum Bildbetrachter hin.



Der Sender strahlt in Richtung Himmel und Erde.

Es dürfte klar sein, daß man den Sender immer senkrecht (nicht liegend) betreiben sollte. Man muß aber damit leben, daß der Sender auch in senkrechter Lage in die falsche Richtung strahlen kann (wie im mittleren Bild).



Das Bild links zeigt die Abstrahlcharakteristik des Mini-Fuchs-Senders in der Draufsicht. Die Funkwellen werden in Form von großen Kugeln abgestrahlt. Im grünen Bereich der Grafik ist der Sender optimal zu empfangen, der Rest (quer zum Fuchs) ist "tote Zone", in denen der Sender nicht oder nur schwach zu empfangen ist. Das ist nicht gar so dramatisch, aber man sollte die Existenz der zwei toten Zonen im Hinterkopf behalten. Deshalb hier noch einmal der Tip: Im Laufen peilen! - Vielleicht ist der Fuchs schon nach wenigen Metern toter Zone wieder gut zu empfangen...

Beim Verstecken des Fuchses muß außerdem beachtet werden, daß Funkwellen von allem abgeschirmt werden können, was elektrischen Strom leitet - und dazu gehört zum Beispiel auch nasses Geträuch und Erde. Man meide also Stahlbeton und Gartenzäune, durchnäßte Mauern, Erdlöcher und Baumhöhlen (es seie denn der Baum ist trocken).

Zusammenfassung für den Fuchs:

- Ist die Batterie frisch? Funktioniert der Sender tadellos und mit voller Leistung?
- Der Fuchs darf nicht liegend betrieben werden, weil er dann in den Himmel strahlt.
- Gute Fuchs-Orte sind etwas erhöht und in ihrer Umgebung frei von schirmenden Gegenständen. Beispielsweise sind (erreichbare!) Baumzweige ein guter Ort für den Fuchs.
- Ein gutes Fuchsjagdgelände hat kein zu nasses Unterholz und keine Gebäude - es seie denn, man legt es genau darauf an.
- Nach dem Verstecken der Füchse prüft man stets den Empfang. Der Ort könnte schlecht sein, der Sender nicht in Ordnung oder die Kennung falsch.

Zusammenfassung für den Jäger:

- Ist die Batterie frisch? Funktioniert der Empfänger tadellos? Rauschen im Hörer reicht nicht, weil das Rauschen auch noch bei einer fast toten Batterie zu hören ist. Der Fuchs muß zu hören sein!
- Erst die Übung, danach der Wettkampf. Beim Üben kann man ruhig auch mal "Blinde Kuh" spielen, während der Fuchs für alle anderen gut zu sehen ist.
- Ist wirklich klar, in welcher Richtung die Antenne maximal empfängt und in welcher minimal?
- Gute Empfangsorte werden ermittelt, indem man im Gehen peilt. Erst dann stehenbleiben, wenn man einen guten Empfangsort hat.

Ergänzungen und FAQ

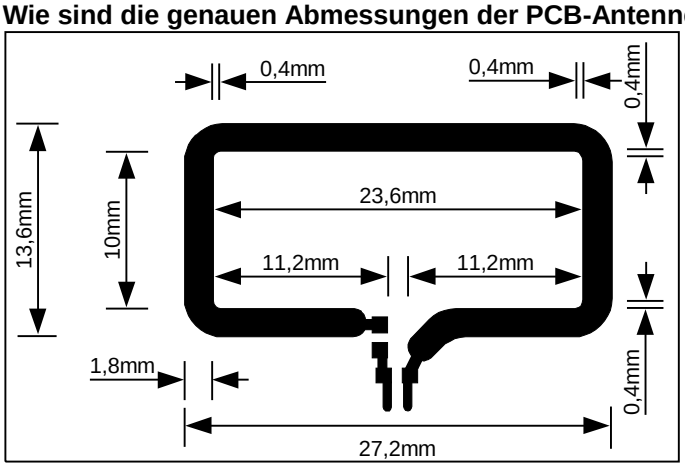
Lebensdauer der Batterie:
Die drei LR44-Knopfzellen sollen den Fuchs vor allem klein halten, ansonsten bin ich bei der Wahl dieser Zellen davon ausgegangen, daß man nicht jeden Tag eine Fuchsjagd macht. Falls doch (wirklich?), dann genau überlegen, was preiswerter ist: Ein Doppelpack (2 Zellen) LR44-Alkali kostet z.B. bei Conrad 50 Cent (3 Zellen also insgesamt 75 Cent). Alternativ kostet ein NiMH-Vierfach-Akku mit 4,8V zwischen 5 und 6 Euro. Das könnte sich lohnen, wenn man tatsächlich öfter eine Fuchsjagd vorhat - zumindest für die Fuchs-Sender. Außerdem hätte man den Vorteil, daß der Akku gelötet werden könnte. Beim Simpel-Empfänger halte ich Akkus aber für unrentabel.

Der Fuchs ist mit dem Simpel-Empfänger nur extrem schwach zu empfangen:
Batterie wechseln - und das ganze ist auch keine sogenannte dumme Frage:
Der Simpel-Empfänger arbeitet bei etwa 3,9 Volt noch korrekt, aber ab 3,8 Volt abwärts verliert er extrem an Empfindlichkeit. Das nach wie vor noch zu hörende UKW-Rauschen kann vortäuschen, daß der Empfänger noch korrekt arbeitet. Dieses Problem kann umgangen werden, indem man den Empfänger u.U. mit 4 LR44-Zellen betreibt (aber nicht mehr!).
Der Sender arbeitet bei Spannungen bis herunter zu etwa 3,6 Volt korrekt und verliert dann drastisch an Leistung. Bei etwa 3 bis 3,1 Volt findet man noch ein kleines "Hoch" und bei etwa 2,6 Volt ist auch dann nichts mehr zu hören, wenn man die Empfangsantenne direkt an den Fuchs hält.

Gibt es Sender und Empfänger auch fertig aufgebaut zu kaufen?
Neu: Da sich kein kommerzieller Hersteller finden wollte, habe ich die Fertigung in die eigene Hand genommen. Beide Geräte können in einer weiterentwickelten Form unter www.rowalt.de/mc/ im Bereich ”Shop” bestellt werden.

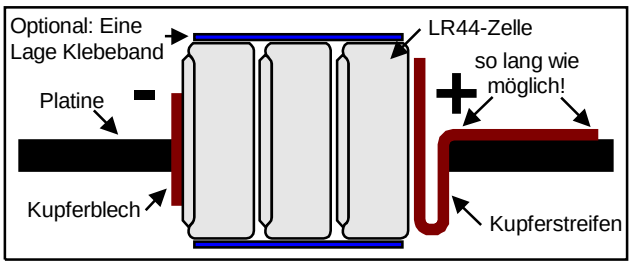
Muß man unbedingt WinAVR zum Brennen des AVRs nehmen?
Nein, wer einen anderen Programmer installiert hat, kann natürlich auch diesen nehmen. Die einzige Bedingung ist, daß man einen InSystem-Programmer verwendet (also einer, der den AVRs in der Zielschaltung brennen kann). Beispielsweise ist auch PonyProg (www.lancos.com) verwendbar oder das Atmel-STK500.

Wie sind die genauen Abmessungen der PCB-Antenne für den Fuchs-Sender?



Hier sind sie. Ich habe sie einer Micrel-ApplicationNote entnommen und dann von Inch in Millimeter umgerechnet.
(Inch * 25,4 = Millimeter)
Bei meinem Leiterplatten-Design in Sprint-Layout hatte ich die Originalzeichnung (!) von Micrel einfach als Bitmap hinterlegt und die Antenne darüber mit Leiterzügen nachgebildet. Da das bei meinen Füchsen gute Ergebnisse brachte, denke ich, daß die Abmessungen nicht ultrakritisch sind.

Was kann ich als Batteriehalterung nehmen?



Ich habe mir eine Batteriehalterung wie links im Bild selbst gebaut. Wenn man ein rechteckiges Loch entsprechender Größe in die Platine sägt, ergibt das eine sehr stabile Halterung. Die Kupferstreifen kann man aus Elektronik-Schrott gewinnen. Man sollte nicht mit Lötzinn sparen. Besonders beim gebogenen Streifen sollte man darauf achten, daß sich auch Zinn zwischen Platine und Kupferstreifen befindet. Das Lötzinn darf ruhig von der Platine auf die Kupferstreifen-Oberseite überlappen.
Da die LR44-Knopfzellen je nach Hersteller leider nicht exakt gleich hoch sind, sollte man von vornherein etwas Toleranz einplanen. Das Löten der Kontaktbleche macht man am bequemsten, indem man die Kontakte zwischen einem Stapel alter LR44-Zellen einklemmt.

Ich habe noch nie mit SMD-Bauteilen gearbeitet...
Bei unserem Sender und Empfänger ist das kein Problem, weil alle Bauteile noch groß genug sind. Folgendes muß man wissen:
Beim Kauf: Bei den Widerständen und Keramik-Kondensatoren die Baugröße 0805 verlangen (der Händler weiß dann schon...). Diese Baugröße läßt sich noch sehr gut verarbeiten. Die kleinere Baugröße 0603 ist schon recht "fummelig". Bei den Elkos einfach die kleinsten nehmen, die es gibt. Das wird hier im Normalfall die Baugröße 1206 sein.
Werkzeug: Es wird nur eine Kreuzklemmpinzette und feiner Lötdraht benötigt. Der normale Lötkolben reicht, wenn er stets mit einem feuchten Schwamm von Zunder freigehalten wird.
Passive Bauteile löten: Erst ***ein*** Lötpad verzinnen, dann Bauteil einseitig anlöten, dann ggf. Lage noch einmal korrigieren und zum Schluß die andere Seite anlöten.
ICs löten: Ein Eck-Lötpad verzinnen, dann den zugehörigen Eck-Pin des ICs anlöten und danach ggf. noch einmal die Lage korrigieren. Dann wird der diagonal gegenüberliegende Pin gelötet. Die restlichen Pins werden zum Schluß der Reihe nach gelötet.
Das ist alles und überhaupt nicht schwierig. Wirklich!

```
;*****************************************************************************
;*              Fox amateur radio transmitter with MICRF102                  *
;*             Roland Walter, DL7UNO, Mai 2002, www.rowalt.de                *
;*                        Last update: 29.11.2002                            *
;*                    Compiler: Atmel AVR Assembler                          *
;*****************************************************************************
;History:
;29.11.2002:
; - Comparator switched out. By default the comparator is switched on and
;   consumes about 0.5mA so that we can save about 20%
;
;This program works well with both AT90S2343 and ATtiny12.
;ATiny12 is better: It's more cheap and has a MUCH more stabile RC oscillator.
;The best sub-types are ATtiny12L-4 (2,7...5,5V - 4MHz) and
;ATtiny12V-1 (1,8...5.5V - 1,2MHz).
;If You want to use the AT90x2343 You should take the AT90LS2343 because it has
;a better voltage range (2,7...6,0V instead of 4,0...6,0V).
;
;The AVR runs with the internal 1MHz RC oscillator. The program is based on
;this frequency.
;The AVR outputs Morse calls with a frequency of about 1KHz. The user can set
;the following parameters using three buttons:
; 1.) Morse call sign: MOE, MOI, MOS, MOH, MO5 or MO
;     These are the call signs allowed in Gernmany (regulation AFuV, 23.12.1997)
; 2.) The number of calls between longer pauses (3,4,5,6,7 or 8 calls)
; 3.) The length of the pause (0/30sec/1min/2min/5min/10min)
;The selection is done by repeated button pushing. The current settings are
;stored in EEPROM so that the selection is hold also without power supply.
;Above this as hidden feature it's possible to set extended values by
;manipulating the EEPROM content:
; 1.) The call sign is stored in EEPROM address 1. A value of 0 generates
;     the call sign MOE, a 1 generates MOI, a 2 generates MOS and so on up to
;     value 5 which generates MO. Setting a value of 254 (0xFE) results a
;     permanent tone of about 1KHz, setting a value of 255 (0xFF) results
;     sending without any modulation. The last two settings are intended for
;     measurement only because they are not allowed for Fox hunting.
;     Values between 6...253 are not defined.
; 2.) At EEPROM address 2 the number of calls is stored. Setting this byte
;     to the value 0 generates 3 repeated calls, setting 1 results 4 calls,
;     setting 2 results 5 calls and so on. The maximum possible number of
;     calls is 255+3=258
; 3.) At EEPROM address 3 the pause length is stored. The first 6 addresses
;     (0...5) are IDs with the following meaning:
;     0=Off, 1=30sec, 2=1min, 3=2min, 4=5min, 5=10min
;     Values between 6...255 are interpreted as timing factor for the pause
;     length. The formula is: Value*10=PauseLength (in seconds). For example
;     a value of 10 results a pause of 10*10=100sec (1:40min) and a value of
;     51 results 42*10=420sec (4:00 minutes). Note that the pause length is
;     not very precise because it's based on a RC oscillator. The exact value
;     depends on the tempreature and power supply voltage. The values are
;     designated for 4,5 volts and 20°C. In a practical with 20°C and 4,5V
;     I got for the value 70 (700sec) a real pause of 694sec (20°C, 4,5V) using
;     the ATtiny12.
;
;The program is not optimized for elegance or what else but for simlicity so
;that also beginners get a chance to make changes. For example I haven't used
;any interrupt and the button states are determined by simple polling.
;By the way: Polling of button pins gives more flexibility for the I/O pin
;assignement.
;
;Further AVR hints:
;AT90S2343 and ATtiny12 you can see as to be pin compatible for our case.
;But an important difference between these devices is that the ATtiny12 has no
;SRAM but a 3-level-deep hardware stack only. Also the AT90S2343 has more flash
;and EEPROM memory. So if You want to rewrite the program in a high-level
;languge such as Bascom it's better not to use the ATtiny12.
;When burning be carefull with the Fuse bits. Normally the Fuse bits factory
;settings must not be changed. If You need some Fuse changes keep in mind that
;for example the SPIEN bit disables further programming and the RSTDISBL Fuse
;bit (ATtiny12) disables the reset pin which also disables further programming.
;
;MICRF102 hints:
;The MICRF102 is controlled by to wires:
; 1.) ASK data output:    1=Transmit,    0=NotTransmit
; 2.) Standby output:     1=ActiveMode,  0=StandbyMode (<0.04uA only)
;
;Power supply:
;I have used three LR44 alkaline cells which give 4,5 Volts together. The RC
;oscillator timing is based on this voltage. The LR44 cells have a capacity
;of about 120mAh (SR44 is more expensive but hase 165mAh).
;The absolute minimum voltage is 3,8 volts. Below there is no stabile operation
;possible. The maximum voltage (factory datasheet) is 6 volts. I have made a
;10 minutes "crashtest" using 8 volts without damages so it should also be
;possible to use a single 4LR44 battery as power supply which gives 6 volts
```

```
;(105mAh). Note that the external power resistors of the MICRF102 are optimized
;for use with 4,5V. For 6 volts operation You can save a bit current when You
;use other resistor values.
;***************************************************************************
;
;Choose the right AVR here:
.include "tn12def.inc"     ;Device: ATtiny12
;.include "2343def.inc"     ;Device: AT90S2343
; - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
;Constant settings:
;
;I/O pin assignment:
;Roland's old PCB layout:
;.equ MORSE_PIN    = 0    ;PB0: ASK data (Morse) output to MICRF102
;.equ CALLSIGN_PIN = 4    ;PB4: Call sign button (MO,MOE,MOI,MOS,MOH,MO5)
;.equ STANDBY_PIN  = 1    ;PB1: Standby output to MICRF102
;.equ NUMCALLS_PIN = 3    ;PB3: Number of calls between pauses (3,4,5,6,7,8)
;.equ PAUSE_PIN    = 2    ;PB2: Pause between a group of calls (0/30sec/1min/2min/5min/10min)
;
.equ MORSE_PIN    = 4    ;PB4: ASK data (Morse) output to MICRF102
.equ STANDBY_PIN  = 3    ;PB3: Standby output to MICRF102
.equ CALLSIGN_PIN = 0    ;PB0: Call sign button (MO,MOE,MOI,MOS,MOH,MO5)
.equ NUMCALLS_PIN = 1    ;PB1: Number of calls between pauses (3,4,5,6,7,8)
.equ PAUSE_PIN    = 2    ;PB2: Pause between a group of calls (0/30sec/1min/2min/5min/10min)
;
;
;Taste and habit depending settings:
.equ TONE_FREQU   = 160  ;Morse tone frequency (recommended: 160)
.equ LENGTH_DIT   = 80   ;Length of a Morse "Dit" (recommended: 100)
.equ LENGTH_DA    = 240  ;Length of a Morse "Da" (must be 3 times of "Dit")
;
;IDs for the variable FoxCallsign:
.equ CALL_MOE     = 0    ;Send Fox call sign "MOE"
.equ CALL_MOI     = 1    ;Send Fox call sign "MOI"
.equ CALL_MOS     = 2    ;Send Fox call sign "MOS"
.equ CALL_MOH     = 3    ;Send Fox call sign "MOH"
.equ CALL_MO5     = 4    ;Send Fox call sign "MO5"
.equ CALL_MO      = 5    ;Send Fox call sign "MO"
.equ CALL_TONE    = 254  ;Send 1KHz tone only    (hidden mode)
.equ CALL_CARRIER = 255  ;Send carrier only without modulation (hidden mode)
;
;IDs for the variable NumCalls:
.equ NCALLS_3     = 0    ;Send Fox call sign 3 times between longer pauses
.equ NCALLS_4     = 1    ;Send Fox call sign 4 times between longer pauses
.equ NCALLS_5     = 2    ;Send Fox call sign 5 times between longer pauses
.equ NCALLS_6     = 3    ;Send Fox call sign 6 times between longer pauses
.equ NCALLS_7     = 4    ;Send Fox call sign 7 times between longer pauses
.equ NCALLS_8     = 5    ;Send Fox call sign 8 times between longer pauses
;
;IDs for the variable LenSilence:
.equ SILENCE_NO     = 0    ;No silence period
.equ SILENCE_30SEC  = 1    ;Silence period of 30 seconds
.equ SILENCE_1MIN   = 2    ;Silence period of 1 minute
.equ SILENCE_2MIN   = 3    ;Silence period of 2 minutes
.equ SILENCE_5MIN   = 4    ;Silence period of 5 minutes
.equ SILENCE_10MIN  = 5    ;Silence period of 10 minutes
;
;EEPROM addresses (where the user settings are stored):
.equ EEADR_CALL       = 1  ;Address of the Fox call sign ID (a CALL_... value)
.equ EEADR_NCALLS     = 2  ;Number of calls between pauses (0...5==>3...8 calls)
.equ EEADR_LENSILENCE = 3  ;Length of the pause, see variable LenSilence
; - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
;Declare the variables:
.def Temp         = r16  ;For general use
.def TempCount1   = r17  ;For delay loops
.def TempCount2   = r18  ;For delay loops
.def TempCount3   = r19  ;For delay loops
.def TempCount4   = r20  ;For delay loops
.def FoxCallsign  = r21  ;Holds current used Morse code (0=MOE, 1=MOI and so on)
.def NumCalls     = r22  ;Holds the number of calls between the silence periods
.def TempNumCalls = r23  ;Holds the number of calls just done in a period
.def LenSilence   = r24  ;Holds the length of the silence periods
.def RespFrequ    = r25  ;Receives the tone frequency for the routine ResponseDit
.def CurrentID    = r26  ;Receives the current ID (Call sign, Pause, nCalls)
;***************************************************************************
.cseg     ;Begin of Code segment
.org 0    ;Address 0 is the Reset vector
rjmp Reset
.org 3    ;Skip the interrupt vectors (not really necessary but we do it)
;***************************************************************************
Reset:
    ;The next two instructions are for AT90S2343 only, comment out for ATtiny12:
    ;Initialize stack pointer:
    ;ldi Temp,RAMEND   ;RAMEND: Not with ATtiny12, because it has no SRAM
    ;out SPL,Temp      ;SP:     Not with ATtiny12, which has hardware stack only
```

```
    ;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    ;
    sbi ACSR,ACD          ;Switch out comparator; saves about 0.5mA (20% !!!)
    ;
    ;Load the user settings from EEPROM now:
    ;
    ;The second byte of the EEPROM (address 1) contains the FoxCallsign ID
    ldi Temp,EEADR_CALL   ;Load EEPROM address of FoxCallsign value
    out EEAR,Temp         ;Copy address into EEPROM address register
    sbi EECR,EERE         ;Set EEPROM Control Register to read access
    in FoxCallsign,EEDR         ;Read EEPROM byte into the FoxCallsign variable
    ;
    ;The third byte of the EEPROM (address 2) contains the number of calls
    ;which are to send immediately repeated. After a period of silence follows.
    ldi Temp,EEADR_NCALLS ;Load EEPROM address of NumCalls value
    out EEAR,Temp         ;Copy address into EEPROM address register
    sbi EECR,EERE         ;Set EEPROM Control Register to read access
    in NumCalls,EEDR        ;Read EEPROM byte into the NumCalls variable
    ;
    ;Now we set the length of the silence period between the set number of
    ;immediately sent Fox calls. The Silence routine contains a detailed
    ;description of the value.
    ldi Temp,EEADR_LENSILENCE ;Load EEPROM address of LenSilence value
    out EEAR,Temp             ;Copy address into EEPROM address register
    sbi EECR,EERE             ;Set EEPROM Control Register to read access
    in LenSilence,EEDR        ;Read EEPROM byte into the NumCalls variable
    ;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    ;Pin usage configuration:
    ;Set the button pins to input (0), all other output (1)
    ldi Temp,0b11111111    ;First we set all pins to output (1)
    out DDRB,Temp          ;Load the value into the DataDirectionRegisterB
    cbi DDRB,CALLSIGN_PIN  ;Set the Call sign selection pin to input (0)
    cbi DDRB,NUMCALLS_PIN  ;Set the NumberOfCalls selection pin to input (0)
    cbi DDRB,PAUSE_PIN     ;Set the Pause selection pin to input (0)
    ;
    ;Now activate PullUp for button input pins and set all other pins to 0:
    ldi Temp,0b00000000    ;Reset all pins to 0
    out PORTB,Temp         ;Load the value into the PORTB register
    sbi PORTB,CALLSIGN_PIN ;Set Call sign input pin to 1 (activate PullUp)
    sbi PORTB,NUMCALLS_PIN ;Set NumberOfCalls input pin to 1 (activate PullUp)
    sbi PORTB,PAUSE_PIN    ;Set Pause input pin to 1 (activate PullUp)
    ;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    ;Configure the the sleep mode:
    ;ldi Temp,0b00000000  ;No sleep mode
    ;out MCUCR,Temp       ;Set the value to the MCU Control Register MCUCR
;*****************************************************************************
;Before we go into Main loop we check whether the variable FoxCallsign is set
;for one of the hidden special modes. If so we don't enter the main loop and
;execute a special code:
    cpi FoxCallsign,254
    brlo MainLoop     ;<254: No hidden mode selected, execute main loop
    rjmp HiddenModes
;*****************************************************************************
MainLoop:
    ;
    ldi TempNumCalls,3         ;The number of calls is 3,4,5,6,7 or 8
    add TempNumCalls,NumCalls  ;(Re)set the number of calls to make in a period
    MainLoopPeriod:
    ;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    ;Generate Morse character "M": Da Da
    rcall Da
    rcall DitSilence
    rcall Da
    ;
    rcall DaSilence
    ;
    ;Generate Morse character "O": Da Da Da
    rcall Da
    rcall DitSilence
    rcall Da
    rcall DitSilence
    rcall Da
    ;
    rcall DaSilence
    ;
    ;The last Morse character depends on the user settings.
    ;
    cpi FoxCallsign,CALL_MO     ;FoxCallsign MO ==> no more tones, skip rest
    breq GenerateMO
    cpi FoxCallsign,CALL_MOE  ;FoxCallsign MOE ==> Rest: Dit
    breq GenerateMOE
    cpi FoxCallsign,CALL_MOI  ;FoxCallsign MOI ==> Rest: Dit Dit
    breq GenerateMOI
    cpi FoxCallsign,CALL_MOS  ;FoxCallsign MOS ==> Rest: Dit Dit Dit
```

```
    breq GenerateMOS
    cpi FoxCallsign,CALL_MOH  ;FoxCallsign MOH ==> Rest: Dit Dit Dit Dit
    breq GenerateMOH
    cpi FoxCallsign,CALL_MO5  ;FoxCallsign MO5 ==> Rest: Dit Dit Dit Dit Dit
    brsh GenerateMO5
    GenerateMO5:
    rcall Dit
    rcall DitSilence
    GenerateMOH:
    rcall Dit
    rcall DitSilence
    GenerateMOS:
    rcall Dit
    rcall DitSilence
    GenerateMOI:
    rcall Dit
    rcall DitSilence
    GenerateMOE:
    rcall Dit
    GenerateMO:
    ;
    rcall StandbyMode
    ;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    ;Now check wheter we have done the number of calls per period or not:
    dec TempNumCalls
    brne MainLoopPeriod    ;Number of calls per period is not done, next call
    ;The group of calls is finished if we reach this position
    rcall Silence          ;Generate a silence period
    ;
rjmp MainLoop
;*****************************************************************************
;Button handler routine; changes the EEPROM content and performs reset
CheckButtons:
    sbis PINB,CALLSIGN_PIN    ;Is the "Call sign" button pushed?
    rjmp CheckButtonCallsign  ;Jump to the CheckButtonCallsign routine
    sbis PINB,NUMCALLS_PIN    ;Is the "Number of calls" button pushed?
    rjmp CheckButtonNumCalls  ;Jump to the CheckButtonNumCalls routine
    sbis PINB,PAUSE_PIN       ;Is the "Pause length" button pushed?
    rjmp CheckButtonPause     ;Jump to the CheckButtonPause routine
    rjmp DoneCheckButtons     ;No button pushed, leave the routine
    ;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    CheckButtonCallsign:   ;Routine for the "CAll sign" button
    inc FoxCallsign
    cpi FoxCallsign,6       ;FoxCallsign must be 0...5
    brlo DoneCallsignCorr  ;Skip next line if FoxCallsign is <6
    ldi FoxCallsign,0      ;Reset callsign ID to 0
    DoneCallsignCorr:
    ;
    ;Write the new FoxCallsign ID into EEPROM:
    ldi Temp,EEADR_CALL    ;Load EEPROM address for the Fox call sign ID
    out EEAR,Temp          ;Copy address into EEPROM address register
    out EEDR,FoxCallsign   ;Load the FoxCallsign content into EEPROM data buffer
    sbi EECR,EEMWE         ;Enable EEPROM write access
    sbi EECR,EEWE          ;Write the loaded byte into EEPROM
    ;
    mov CurrentID,FoxCallsign ;The response tone below needs ID in variable CurrentID
    rjmp CheckButtonsEnd      ;Job done, give response tone and let restart the AVR
    ;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    CheckButtonNumCalls:   ;Routine for the "Number of calls" button
    inc NumCalls
    cpi NumCalls,6            ;NumCalls must be 0...5 (results 3...8 calls)
    brlo DoneNumCallsCorr  ;Skip next line if NumCalls is <6
    ldi NumCalls,0         ;Reset number of calls ID to 0
    DoneNumCallsCorr:
    ;
    ;Write the new number of calls ID value into EEPROM:
    ldi Temp,EEADR_NCALLS  ;Load EEPROM address for the number of calls
    out EEAR,Temp          ;Copy address into EEPROM address register
    out EEDR,NumCalls      ;Load the NumCalls content into EEPROM data buffer
    sbi EECR,EEMWE         ;Enable EEPROM write access
    sbi EECR,EEWE          ;Write the loaded byte into EEPROM
    ;
    mov CurrentID,NumCalls ;The response tone below needs ID in variable CurrentID
    rjmp CheckButtonsEnd   ;Job done, give response tone and let restart the AVR
    ;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
       CheckButtonPause:          ;Routine for the "Length of pause" button
    inc LenSilence
    cpi LenSilence,6          ;LenSilence must be 0...5
    brlo DoneLenSilenceCorr ;Skip next line if LenSilence is <6
    ldi LenSilence,0        ;Reset length of pause ID to 0
    DoneLenSilenceCorr:
    ;
    ;Write the new number of calls ID value into EEPROM:
    ldi Temp,EEADR_LENSILENCE ;Load EEPROM address for the pause length
```

```
    out EEAR,Temp              ;Copy address into EEPROM address register
    out EEDR,LenSilence     ;Load the LenSilence content into EEPROM data buffer
    sbi EECR,EEMWE          ;Enable EEPROM write access
    sbi EECR,EEWE           ;Write the loaded byte into EEPROM
    ;
    mov CurrentID,LenSilence  ;The response tone below needs ID in variable CurrentID
    rjmp CheckButtonsEnd      ;Job done, give response tone and let restart the AVR
    ;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    CheckButtonsEnd:
    ;No we respond the current selected option by a tone. The variable Temp
    ;must contain the currend set ID (Tone frequency, Pause or Callsign) because
    ;we give it out to the user.
    ldi RespFrequ,255   ;Set tone frequency of the response tone for ResponseDit
    rcall ResponseDit   ;Output the response tone
    ldi RespFrequ,128   ;Set tone frequency of the response tone for ResponseDit
    rcall ResponseDit   ;Output the response tone
    ldi RespFrequ,64    ;Set tone frequency of the response tone for ResponseDit
    rcall ResponseDit   ;Output the response tone
    ldi RespFrequ,0     ;Tone frequency 0: Pause only
    rcall ResponseDit   ;Output pause with "Dit" length
    rcall ResponseDit   ;Output pause with "Dit" length
    rcall ResponseDit   ;Output pause with "Dit" length
    ldi RespFrequ,128   ;Set tone frequency of the response tone for ResponseDit
    inc CurrentID       ;ID for Callsign/nCalls/Pause: not 0...5 but 1...6
    CheckButtonsEnd2:   ;Output the current ID as a number of Dits
    rcall ResponseDit   ;Output the response tone
    dec CurrentID       ;Current set ID for Callsign/nCalls/Pause
    brne CheckButtonsEnd2
    ;
    ;We must prevent fast repeated calls of the button routine which can occur
    ;because buttons are far from being perfect. This we do in a very simple
    ;way by starting the Watchdog which causes a reset after a defined time.
    ;
    ;Make the transmitter silent until the reset:
    cbi PORTB,STANDBY_PIN  ;Set MICRF102 StandBy output pin to 0 (Standby mode)
    ;
    wdr      ;Reset the WatchDog
    ldi Temp,0b00001011    ;Bit3: WatchdogEnable, Bit2,1,0: 011 = 120mSec (5V)
    out WDTCR,Temp         ;Write the byte into Watchdog Timer Control Register
    ;
    WaitResetLoop:          ;Make nothing until reset ("Stop" program)
    sbis PINB,CALLSIGN_PIN  ;Is the "Call sign" button still pushed?
       wdr                  ;Reset the watchdog (start watchdog again)
    sbis PINB,NUMCALLS_PIN  ;Is the "Number of calls" button still pushed?
       wdr                  ;Reset the watchdog (start watchdog again)
    sbis PINB,PAUSE_PIN     ;Is the "Pause length" button still pushed?
       wdr                  ;Reset the watchdog (start watchdog again)
    rjmp WaitResetLoop      ;
    ;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    DoneCheckButtons:
ret
;*****************************************************************************
;Generates a "Dit" for button response (called from CheckButtons routine)
;The variable RespFrequ must contain the frequency of the tone. The frequency
;has the same principle as used for the normal "Dit" and "Da" as set with
;the vconstant TONE_FREQU. If RespFrequ=0 then a pause only is generated.
;After the "Dit" also a silence period of one "Dit" length is generated.
ResponseDit:
    sbi PORTB,STANDBY_PIN   ;Set StandBy output pin to 1 (Standby mode disabled)
    ;
    cpi RespFrequ,0     ;Is the tone frequency set to 0?
    breq RespSilence    ;No tone, pause only
    ;
    ldi TempCount2,LENGTH_DIT ;Set the length of the Morse "Dit"
    ResponseDit2:
    ;Now generate one "sine" period of the wiched frequency given in RespFrequ
    sbi PORTB,MORSE_PIN       ;Set Morse output pin to high
    mov TempCount1,RespFrequ
    RespSinePeriod1:
    dec TempCount1
    brne RespSinePeriod1
    ;
    cbi PORTB,MORSE_PIN       ;Set Morse output pin to low
    ldi TempCount1,TONE_FREQU
    RespSinePeriod2:
    dec TempCount1
    brne RespSinePeriod2
    ;End of "sine"
    dec TempCount2
    brne ResponseDit2
    ;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    RespSilence:
    ;Now the silence period:
    cbi PORTB,MORSE_PIN       ;Set Morse output pin to low
```

```
    ;
    ldi TempCount3,2           ;Set the pause length
    RespSilence1:
    ldi TempCount2,LENGTH_DIT  ;Load the length of one Morse "Da"
    RespSilence2:
    ldi TempCount1,TONE_FREQU
    RespSilence3:
    dec TempCount1
    brne RespSilence3
    dec TempCount2
    brne RespSilence2
    dec TempCount3
    brne RespSilence1
ret
;***************************************************************************
;
Dit:
    sbi PORTB,STANDBY_PIN   ;Set StandBy output pin to 1 (Standby mode disabled)
    ;
    ldi TempCount2,LENGTH_DIT ;Set the length of the Morse "Dit"
Dit2:
    rcall CheckButtons        ;We must permanently check the buttons
    ;Now generate one "sine" period of about 1.9KHz. No nice sound but very simple.
    sbi PORTB,MORSE_PIN       ;Set Morse output pin to high
    ldi TempCount1,TONE_FREQU
SinePeriod1:
    dec TempCount1
    brne SinePeriod1
    ;
    cbi PORTB,MORSE_PIN       ;Set Morse output pin to low
    ldi TempCount1,TONE_FREQU
SinePeriod2:
    dec TempCount1
    brne SinePeriod2
    ;End of "sine"
    dec TempCount2
    brne Dit2
ret
;***************************************************************************
;Generates silence with the same length as "Dit"
DitSilence:
    cbi PORTB,STANDBY_PIN    ;Set StandBy output pin to 0 (Standby mode enabled)
    ;
    ldi TempCount2,LENGTH_DIT ;Set the length of the Morse "Dit" silence
DitSilence2:
    rcall CheckButtons        ;We must permanently check the buttons
    ;Now generate one "sine" period silence
    cbi PORTB,MORSE_PIN       ;Set Morse output pin to low
    ldi TempCount1,TONE_FREQU
SilencePeriod1:
    dec TempCount1
    brne SilencePeriod1
    ;
    nop                       ;Replacement for "Set Morse output pin to low"
    ldi TempCount1,TONE_FREQU
SilencePeriod2:
    dec TempCount1
    brne SilencePeriod2
    ;End of "sine" silence period
    dec TempCount2
    brne DitSilence2
ret
;***************************************************************************
Da:
    sbi PORTB,STANDBY_PIN   ;Set StandBy output pin to 1 (Standby mode disabled)
    ;
    ldi TempCount2,LENGTH_DA  ;Set the length of the Morse "Da"
Da2:
    rcall CheckButtons        ;We must permanently check the buttons
    ;Now generate one "sine" period of about 1.9KHz. No nice sound but very simple.
    sbi PORTB,MORSE_PIN       ;Set Morse output pin to high
    ldi TempCount1,TONE_FREQU
SinePeriodDa1:
    dec TempCount1
    brne SinePeriodDa1
    ;
    cbi PORTB,MORSE_PIN       ;Set Morse output pin to low
    ldi TempCount1,TONE_FREQU
SinePeriodDa2:
    dec TempCount1
    brne SinePeriodDa2
    ;End of "sine"
    dec TempCount2
    brne Da2
ret
```

```
;*****************************************************************************
;
;Generates silence with the same length as "Da"
DaSilence:
    cbi PORTB,STANDBY_PIN    ;Set StandBy output pin to 0 (Standby mode enabled)
    ;
    ldi TempCount2,LENGTH_DA  ;Set the length of the Morse "Da" silence
DaSilence2:
    rcall CheckButtons        ;We must permanently check the buttons
    ;Now generate one "sine" period silence
    cbi PORTB,MORSE_PIN       ;Set Morse output pin to low
    ldi TempCount1,TONE_FREQU
SilencePeriodDa1:
    dec TempCount1
    brne SilencePeriodDa1
    ;
    nop                       ;Replacement for "Set Morse output pin to low"
    ldi TempCount1,TONE_FREQU
SilencePeriodDa2:
    dec TempCount1
    brne SilencePeriodDa2
    ;End of "sine" silence period
    dec TempCount2
    brne DaSilence2
ret
;*****************************************************************************
;Sets MICRF102 into Standby mode for the time of 3x "Da" (Morse pause length)
;The length of the pause must not be accuratly. Might be You find a longer
;pause more usefull.
StandbyMode:
    cbi PORTB,STANDBY_PIN     ;Set StandBy output pin to 0 (Standby mode enabled)
    cbi PORTB,MORSE_PIN       ;Set Morse output pin to low
    ;
    ldi TempCount3,5          ;Set the pause (normally 3 times of "Da")
Standby1:
    rcall CheckButtons        ;We must permanently check the buttons
    ldi TempCount2,LENGTH_DA  ;Load the length of one Morse "Da"
Standby2:
    ldi TempCount1,TONE_FREQU
Standby3:
    dec TempCount1
    brne Standby3
    dec TempCount2
    brne Standby2
    dec TempCount3
    brne Standby1
ret
;*****************************************************************************
;Performs a silence period. This period is used between a defined number of
;sending the Fox code MOE,MOI and so on. In a contest all Foxes send at one and
;the same frequency normally. The silence period increases the probability that
;not too many Foxes send at the same time. Also it makes finding the foxes a bit
;more difficult.
;For the ID in LenSilence the following values are official defined:
;0=0sec (no silence), 1=30sec, 2=1min, 3=2min, 4=5min, 5=10min
;As a hidden feature the user can set values >5 into the EEPROM.
;The length of the silence period in seconds is LenSilence*10, for example
;if LenSilence has a value of 6 the silence period will be about 60 seconds.
;The maximum silence period is 255*10 = 2550 seconds = 42.5 minutes
;If LenSilence=0 no silence period is done and the call is permanently sent.
;The time is not exact of course because it's based on the RC oscillator!
;
Silence:
    cbi PORTB,STANDBY_PIN    ;Set StandBy output pin to 0 (Standby mode enabled)
    cbi PORTB,MORSE_PIN      ;Set Morse output pin to low
    ;
    cpi LenSilence,SILENCE_NO     ;LenSilence=0: No silence period wished
    breq SilenceDone              ;...Leave this routine
    cpi LenSilence,SILENCE_30SEC  ;LenSilence=1: 30sec
    breq SilenceSet1              ;...Set the internal counter value
    cpi LenSilence,SILENCE_1MIN   ;LenSilence=2: 1min
    breq SilenceSet2              ;...Set the internal counter value
    cpi LenSilence,SILENCE_2MIN   ;LenSilence=3: 2min
    breq SilenceSet3              ;...Set the internal counter value
    cpi LenSilence,SILENCE_5MIN   ;LenSilence=4: 5min
    breq SilenceSet4              ;...Set the internal counter value
    cpi LenSilence,SILENCE_10MIN  ;LenSilence=4: 10min
    breq SilenceSet5              ;...Set the internal counter value
    rjmp SilenceSetElse           ;For other values use formula time(sec)=ID*10
    ;
    SilenceSet1:
    ldi TempCount4,3     ;Set length of silence factor (3: 30sec) OK
    rjmp Silence1
    SilenceSet2:
    ldi TempCount4,6     ;Set length of silence factor (6: 60sec=1min)
```

```
    rjmp Silence1
    SilenceSet3:
    ldi TempCount4,12    ;Set length of silence factor (12: 120sec=2min)
    rjmp Silence1
    SilenceSet4:
    ldi TempCount4,30    ;Set length of silence factor (35: 350sec=5min)
    rjmp Silence1
    SilenceSet5:
    ldi TempCount4,60    ;Set length of silence factor (60: 600sec=10min)
    rjmp Silence1
    SilenceSetElse:      ;Hidden feature: Silence in sec = ID*10
    mov TempCount4,LenSilence  ;Copy LenSilence ID into counter
    ;
    Silence1:
    ldi TempCount3,52    ;Load counter factor, practical tested for 20°C/4.5V
    Silence2:
    rcall CheckButtons   ;We must permanently check the buttons
    ldi TempCount2,0     ;Set temp counter
    Silence3:
    ldi TempCount1,0     ;Set temp counter
    Silence4:
    dec TempCount1
    brne Silence4
    dec TempCount2
    brne Silence3
    dec TempCount3
    brne Silence2
    dec TempCount4
    brne Silence1
SilenceDone:
ret
;*****************************************************************************
;Routine for the hidden special modes. These modes the user can only select by
;manipulating the EEPROM (variable FoxCallsign). The hidden modes are not
;available by buttons.
HiddenModes:
    cpi FoxCallsign,CALL_CARRIER  ;Send carrier?
    breq HiddenModeCarrier        ;FoxCallsign=CALL_CARRIER: Send carrier only
    rjmp HiddenModeTone           ;FoxCallsign=CALL_TONE: Transmit a 1KHz tone
    ;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    HiddenModeCarrier:          ;Send carrier only without modulation
    sbi PORTB,STANDBY_PIN       ;StandBy output pin to 1 (Standby mode disabled)
    sbi PORTB,MORSE_PIN         ;Morse output pin to high
    rjmp HiddenModeCarrier
    ;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    HiddenModeTone:             ;Send 1KHz tone only
    sbi PORTB,STANDBY_PIN       ;StandBy output pin to 1 (Standby mode disabled)
    sbi PORTB,MORSE_PIN         ;Set Morse output pin to high
    ldi TempCount1,TONE_FREQU
    HiddenModeTone1:
       dec TempCount1
       brne HiddenModeTone1
       ;
       cbi PORTB,MORSE_PIN      ;Set Morse output pin to low
       ldi TempCount1,TONE_FREQU
    HiddenModeTone2:
       dec TempCount1
       brne HiddenModeTone2
    rjmp HiddenModeTone
    ;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
rjmp HiddenModes
;*****************************************************************************
;The user can change some values such as the Fox call sign, the morse speed and
;so on. These values are stored in EEPROM. It's important that we set some
;usable initial values:
;
;Addr  Description
;  0   Not used
;  1   Fox callsign: "MOE", "MOI", "MOS", "MOH", "MO5" or "MO"
;      With this byte also the "hidden modes" are selectable which are
;      designated for measurement purposes.
;  2   Number of Fox calls which are send repeated between silence periods.
;      Possible are 3...8 calls. An EEPROM value of 0 results 3 calls, a
;      value of 1 results 4 calls and so on.
;  3   Length of the silence period in seconds*10 after the set number of
;      Fox calls. The value 0 disables this feature and the Fox sends without
;      silence period.
;
.eseg    ;Begin the EEPROM content
.ORG 0
    .db 0              ;The first EEPROM address we leave empty
.ORG EEADR_CALL
    .db CALL_MOS       ;Preset sallsign "MOS"
.ORG EEADR_NCALLS
```

```
    .db NCALLS_5       ;Preset 5 calls
.ORG EEADR_LENSILENCE
    .db SILENCE_30SEC  ;Preset 30 seconds silence period
```