50. UKW-Tagung Weinheim

Ein 195 kHz breites, Ethernet-basiertes, Software-Defined Radio

Pieter-Tjerk de Boer, PA3FWM pa3fwm@amsat.org

Es wird ein digitaler Kurzwellenempfänger vorgestellt. Die Hardware kann über den Bereich 0 – 30 MHz abgestimmt werden, und digitalisiert dort ein 195-kHz-breites Teil des Spektrums. Das Signal wird über Ethernet einem Linux-Rechner übertragen. Software zeigt ein breites Spektrum- und "Wasserfall"-Diagramm, und demoduliert AM- und Einseitenband-Signale.

1 Einleitung

In der Funktechnik gibt es, wie auch in vielen anderen Gebieten der Technik, seit Jahren eine Entwicklung in Richtung von zunehmender Digitalisierung. Vor 10 oder 20 Jahren war die Signalverarbeitung in Amateurfunkgeräten noch ausschließlich analog (digitale Technik wurde nur für Steuer- und Regelzwecke verwendet), aber in neueren Geräten (sowohl käuflich erhältlichen als von Amateuren selbst entwickelten) werden Teile der Signalverarbeitung digital gemacht. Das entgültige Ziel dieser Entwicklung wäre ein Empfänger der nur noch aus einem Analog/Digital-Wandler (direkt an der Antenne angeschlossen) und einem Rechner besteht. Soweit ist die Technik zwar noch nicht, aber nach einer noch analog realisierten Bandbreitebegrenzung des Antennensignals (also einem Filter), kann man die restliche Signalverarbeitung vollständich digital machen. Ein so aufgebauter Empfänger wird auf englisch öfters als "Software-Defined Radio" bezeichnet: ein Funkgerät dessen Eigenschaften von der Software statt der Hardware definiert werden.

Die digitale Signalverarbeitung bietet einige Vorteile im Vergleich zur analogen Signalverarbeitung:

- präzisere Implementation: ein Computerprogramm macht eine genau mathematisch definierte Verarbeitung, unabhängig von Temperatur u.ä.;
- bessere Eigenschaften, z.B. steilere Filter;
- neue Techniken werden möglich die analog kaum realisierbar sind: wenn man weiß wie man es mathematisch machen kann, kann man es im Computer programmieren.

Ein deutliches Beispiel des letztgenannten Vorteils sind neue digitale Modi wie PSK-31, wo die digitale Signalverarbeitung normalerweise von einem PC mit Soundkarte gemacht wird.

In diesem Artikel möchte ich meine Entwicklung im Bereich Software-Defined Radios vorstellen: einen Kurzwellenempfänger in dem ein 195 kHz breites Stück des Spektrums digital verarbeitet wird.

2 Hardware

Aufgabe der Hardware ist es, ein Teil des Kurzwellenspektrums zu selektieren, in ein digitales Signal zu wandeln, und diese Daten zum Rechner zu schicken. Die Selektion wird in meinem Gerät, wie auch in vielen anderen Amateur-SDRs, nach dem Quadratur-Prinzip gemacht. Für die A/D-Wandlung und weiterleitung der Daten zum Rechner verwenden viele andere Amateur-SDRs die Soundkarte; mein Gerät enthält einen eigenen A/D-Wandler und schickt die Daten über Ethernet zum Rechner. Ein Blockdiagram ist in Abbildung 1 gezeigt.

2.1 Mischen und das Quadratur-Prinzip

Der Empfänger ist eigentlich ein Direktmischer: die Hochfrequenz-Eingangssignale (z.B., den Bereich 6900–7100 kHz) werden in einem Schritt mit 7000 kHz heruntergemischt, nach 0–100 kHz.

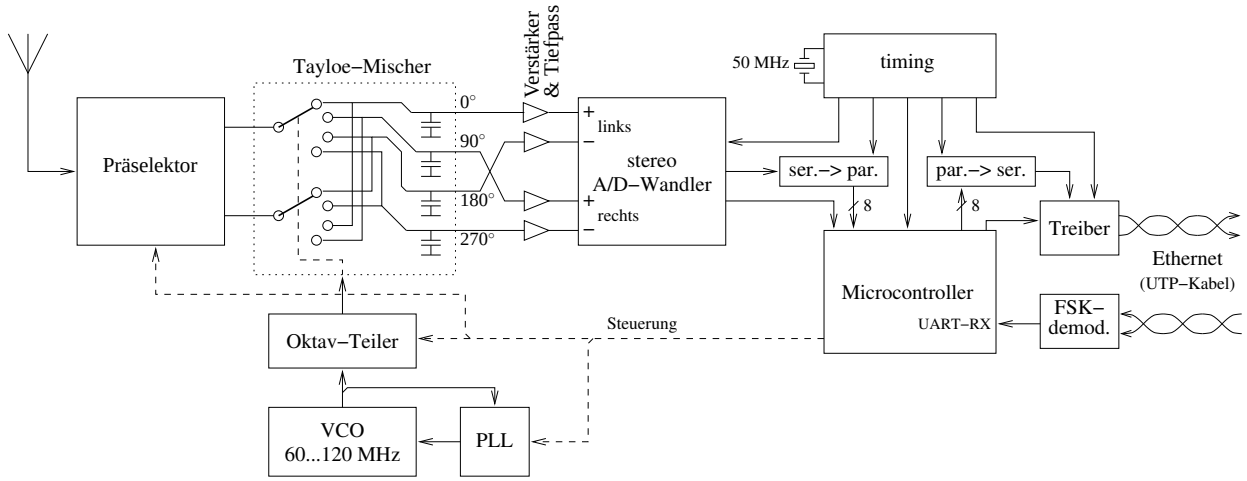

Abbildung 1: Blockdiagram der Hardware

Damit entsteht allerdings ein Spiegelproblem: wenn am Ausgang des Mischers ein Signal von z.B. 12 kHz erscheint, ist prinzipiell nicht klar ob das von einem HF-Signal bei 6988 oder 7012 kHz stammt.

Um dieses Problem zu lösen, wird das Quadratur-Prinzip verwendet. Man nimmt zwei Mischer, die beide das Antennensignal mit (in diesem Beispiel) 7000 kHz mischen, aber die beiden 7000 kHz Signale sind gegenüber einander um 90 Grad in Phase verschoben. Mit einem Eingangssignal zu den Mischern von entweder 6988 oder 7012 kHz, kommt selbstverständlich aus beiden Mischern wieder 12 kHz. Es stellt sich aber heraus daß die beiden 12 kHz Signale aus den beiden Mischern auch um 90 Grad gegenüber einander verschoben sind; und zwar so, daß die *Richtung* der Phasenverschiebung für das niedrigere Eingangssignal (6988 kHz) umgekehrt ist als für das höhere Eingangssignal (7012 kHz). Damit hat man also prinzipiell die Möglichkeit, das gewünschte Signal und das Spiegelsignal zu unterscheiden, ohne zusätzliche Filter vor den Mischern. Man braucht allerdings zwei statt einen A/D-Wandler, denn man hat nun ja zwei Mischerausgangssignale die zum Rechner geführt werden müssen.

Einen ganz einfachen Aufbau der zwei um 90 Grad verschobenen Mischer erlaubt der s.g. Tayloe-Mischer. Der Tayloe-Mischer ist eigentlich ein (integrierter) 4-stelliger Analogschalter (Typ FST3253) die mit der Oszillatorfrequenz “runddreht”, und so nach und nach vier Kondensatoren auflädt; auf den Kondensatoren entstehen so Signale die korrespondieren zu Phasenverschiebungen des Oszillatorsignals von 0, 90, 180 und 270 Grad. Die Ausgänge mit 0 und 180 Grad bilden zusammen einen symmetrischen Ausgang für 0 Grad, und die andere zwei für 90 Grad, und diese symmetrische Ausgänge passen gut zu den symmetrischen Eingänge des verwendeten A/D-Wandlers. Die Symmetrierung macht die Schaltung weniger empfindlich für Störsignale, z.B. aus dem digitalen Teil der Schaltung; aus gleichem Grund ist auch der Eingang des Tayloe-Mischers symmetrisch gemacht.

(Sehe z.B. [1] oder [2] für mehr Informationen über das Quadratur-Prinzip und den Tayloe-Mischer.)

2.2 Analog/Digital-Wandler

Die Leistung eines Software-Defined Radios wird zu einem grossen Teil von den Eigenschaften des A/D-Wandlers bestimmt. Insbesondere sind dabei die Abtastrate und die Auflösung wichtig.

Die *Abtastrate* besagt, wieviele Wandlungen pro Sekunde gemacht werden, und damit wie hohe Frequenzen im Eingangssignal noch im digitalen Ausgangssignal repräsentiert werden: laut des Nyquist-Theorems muß die Abtastrate wenigstens das Doppelte der höchsten Eingangsfrequenz sein. Konkret heißt das, daß wenn der A/D-Wander mit 200 kHz abtastet (also, 200 000 Messungen pro Sekunde macht), Eingangssignale mit Frequenzen zwischen 0 und 100 kHz verarbeitet werden können;

im obigen Beispiel mit Oszillator bei 7000 kHz ergibt das einen Empfangsbereich von 6900 bis 7100 kHz.

Die *Auflösung* besagt, wie genau jede "Messung" repräsentiert wird. Wenn einen A/D-Wandler nur 4 Bits hat, kann sein Ausgang nur $2^4 = 16$ verschiedene Werte annehmen. Das Eingangssignal ist allerdings analog, und kann also unendlich viel verschiedene Werte annehmen; demzufolge muß der Wandler abrunden. Der Unterschied zwischen der "echten", analogen Wert, und der digitalen (zwangsläufig abgerundeten) Wert, ist eigentlich ein Rauschen: eine zusätzliche, und im Wesentlichen zufällige, Störung. Um den Signal-Rausch-Abstand möglichst groß zu machen, wird also einen A/D-Wandler mit hoher Auflösung ("vielen Bits") gebraucht.

Also, der A/D-Wandler soll eine möglichst hohe Abtastrate haben damit der Empfänger breitbandig wird, und eine möglichst hohe Auflösung damit er einen großen Dynamikbereich hat. Leider ist es schwer, A/D-Wandler mit sowohl hoher Abtastrate als hoher Auflösung herzustellen; da muß man beim Bau eines Software-Defined-Radio also einen Kompromiß machen.

Übliche PC-Soundkarten haben A/D-Wandler mit zwei Kanälen (Stereo), einer höchsten Abtastrate von 48 kHz, und einer Auflösung von 16 Bit. Damit lässt sich nach dem Quadraturprinzip ein Empfänger aufbauen mit 48 kHz Bandbreite und einem Dynamikbereich von etwa 98 dB, was für Kurzwellen noch knapp ist (die Software des SDR-1000 benutzt die Pegelregelung der Soundkarte für AGC, siehe [1]). Es gibt einige spezielle Soundkarten mit mehr Auflösung und höheren Abtastraten; diese sind allerdings teuer, und/oder problematisch unter Linux. Ausserdem wäre es vorteilhaft den A/D-Wandler nicht im Innern des Rechners zu haben (wo es ja jede menge digitale Störsignale gibt), sondern näher an der Antenne.

Die hier vorgestellte Hardware enthält seinen eigenen A/D-Wandler: den PCM1804 von Burr-Brown / Texas Instruments. Dieses IC enthält zwei A/D-Wandler mit 24 Bit und 192 kHz Abtastrate. 24 Bit versprechen 146 dB Dynamikbereich, aber das schaft der PCM1804 nicht; der Dynamikbereich wird mit 112 dB spezifiziert, also effektiv etwa 19 Bit. Die Abtastrate von 192 kHz ist auch nicht ganz so gut wie sie klingt: wenn man diese verwendet, steigt das Rauschen bei Signalfrequenzen über 48 kHz. Trotz dieser Einschränkungen ist der PCM1804 deutlich besser als eine normale Soundkarte: 14 dB mehr Dynamikbereich, und doppelter (mit vollem Dynamikbereich) bis vierfacher (mit reduzierter Empfindlichkeit wegen des angestiegenen Rauschen) Bandbreite.

2.3 Ethernet

Irgendwie müssen die Daten aus dem A/D-Wandler zum Rechner geführt werden. Der PCM1804 macht 192 000 Messungen pro Sekunde, zu je 24 Bit für 2 Kanäle; insgesammt sind das 9216000 Bit/Sekunde. Eine solche Datenmenge schaufelt man nicht mehr einfach über die serielle RS232-Schnittstelle in den Rechner; auch für die parallele (Drucker-) Schnittstelle ist das ziemlich viel. Viele andere PC-Schnittstellen, wie ISA-Bus, PCI-Bus, USB, und Firewire, sind entweder auch zu langsam, und/oder ziemlich kompliziert (wenigstens für den Amateur).

Am Einfachsten erschien die Anbindung über 10 Mbit/s Ethernet. Von der Geschwindigkeit her reicht das ja (zwar gerade), und die Komplexität kann ziemlich niedrig sein. Es brauchen keine komplizierende Sachen wie automatische Detektierung/Identifizierung/Konfiguration gemacht zu werden (wie z.B. bei PCI oder USB). Und wenn man das Selbstbaugerät auf eine separate Ethernetkarte im Rechner anschließt (die Karten sind ja ganz billig), können keine Kollisionen mit Signalen von anderen Geräten auftreten; so entfallen die Teilsysteme um Kollisionen zu vermeiden und evtl. kollidierte Daten zu wiederholen die normalerweise bei Ethernet nötig sind.

Das Generieren der Ethernetframes wird von einem Atmel-Mikrokontroller (ATMega32) gemacht, zusammen mit einigem zusätzlichem Logik. Im Mikrokontroller läuft ein Programm das in einer Endlosschleife 762,9 mal pro Sekunde ein Ethernetframe generiert: zunächst einen korrekten Frameheader, dann 1536 Datenbytes (das sind 256 Samples von beiden Kanälen des A/D-Wandlers), und dann

ein Frame-Ende. Weil der Mikrocontroller mit nur 12,5 MHz getaktet wird, kann er nicht 10 million individuelle Bits pro Sekunde erzeugen; dafür wird ein Schieberegister gebraucht, das 1,25 million Mal pro Sekunde 8 bits aus dem Mikrokontroller übernimmt; jeden 10. Takt muß der Mikrocontroller also ein Byte bereitstellen. Der Datenfluß vom A/D-Wandler zum Mikrokontroller erfolgt auch seriell, und zwar mit 12,5 Mbit/s (wobei jedes 4. Byte bedeutungslos ist); dieser Datenstrom wird in ein zweites Schieberegister geschoben, das vom Mikrokontroller genau zu jedem 8. Takt ausgelesen wird. Die Endlosschleife in der Software muß die Daten also auch puffern: die Datenbytes kommen in sehr regelmäßigen Abständen an, werden aber in Gruppen von 1536 wieder ausgegeben. Leider hat der Mikrocontroller nicht ausreichend Rechenleistung um auch noch die 32-Bit-CRC (die "Prüfsumme") für das Ethernetframe zu berechnen: die gesendeten Frames haben deshalb eine falsche CRC, und der Rechner muß diese beim Empfang der Frames ignorieren.

Um das ganze möglichst einfach zu halten, sind die Takt des Ethernet-Senders und des A/D-Wandlers eng verknüpft. Deshalb wird der A/D-Wandler nicht mit 24,576 MHz betrieben (wie vom Hersteller spezifiziert), sondern mit 25 MHz, und deshalb ist die Abtastrate (und demzufolge die empfangene Bandbreite) 195,3125 kHz statt 192 kHz.

2.4 Synthesizer

Der Quadratur-Mischer braucht noch ein Oszillatorsignal, und zwar auf dem Vierfache der gewünschten Empfangsfrequenz, denn es wird ein Vierteiler verwendet um den Mischer mit genau 90 Grad fasenverschobenen Signalen zu versehen. Für einen Kurzwellenempfänger von 0 bis 30 MHz muß der Oszillator also von fast 0 bis 120 MHz liefern. Der Oszillator braucht aber nur in ziemlich groben Schritten, etwa 100 kHz, abgestimmt zu werden, weil der Empfänger so breit ist daß die Feinabstimmung in der Software gemacht werden kann.

In meinem Gerät ist hierfür ein VCO von 60 bis 120 MHz vorgesehen, der mit einem einfachen PLL (MC145170) in 100-kHz-Schritten abgestimmt wird. Um auch tiefere Frequenzen als 60/4=15 MHz zu erreichen, ist ein programmierbarer Oktavteiler aufgebaut, der das VCO-Signal noch mal durch 2, 4, 8, bis 256, teilen kann.

2.5 Präselektor

Wie die meisten Mischer, mischt auch der Tayloe-Mischer nicht nur mit dem eigentlichen Oszillatorsignal, sondern auch mit dessen ungeraden Harmonischen. Um Antennensignale in der Nähe dieser Harmonischen zu entfernen, ist ein Präselektor notwendig, der über dem ganzen Frequenzbereich abgestimmt werden kann. Die Abstimmbarkeit ist hier realisiert indem Kondensatoren und Spulen mit modernen integrierten Analogschaltern umgeschaltet werden.

2.6 Steuerung

Es ist wünschenswert daß der Synthesizer und der Präselektor elektronisch aus dem Rechner abgestimmt werden können. Dafür wird ein Datenstrom in umgekehrter Richtung gebraucht, also vom Rechner zur externen Hardware. Es liegt nahe, hierfür auch das Ethernet zu verwenden. Leider hat der verwendete Mikrokontroller nicht ausreichend Rechenleistung um noch einen 10 Mb/s Datenstrom zu verarbeiten, aber soviel wird auch gar nicht gebraucht: einige kbit/s reichen. Als Trick wird das Ethernetsignal in dieser Richtung als FSK-Signal betrachtet, mit "Tonen" von 5 und 10 MHz. Ein solches Signal lässt sich mit wenig Hardwareaufwand empfangen (digitalem FSK-Decoder und eingebautem UART des Mikrokontrollers), und kann auch einfach vom PC generiert werden (jedes Datenbyte wird in ein Ethernetframe gewandelt, das mit solchen Bitmustern gefüllt wird daß das Signal auf dem Kabel wie FSK aussieht).

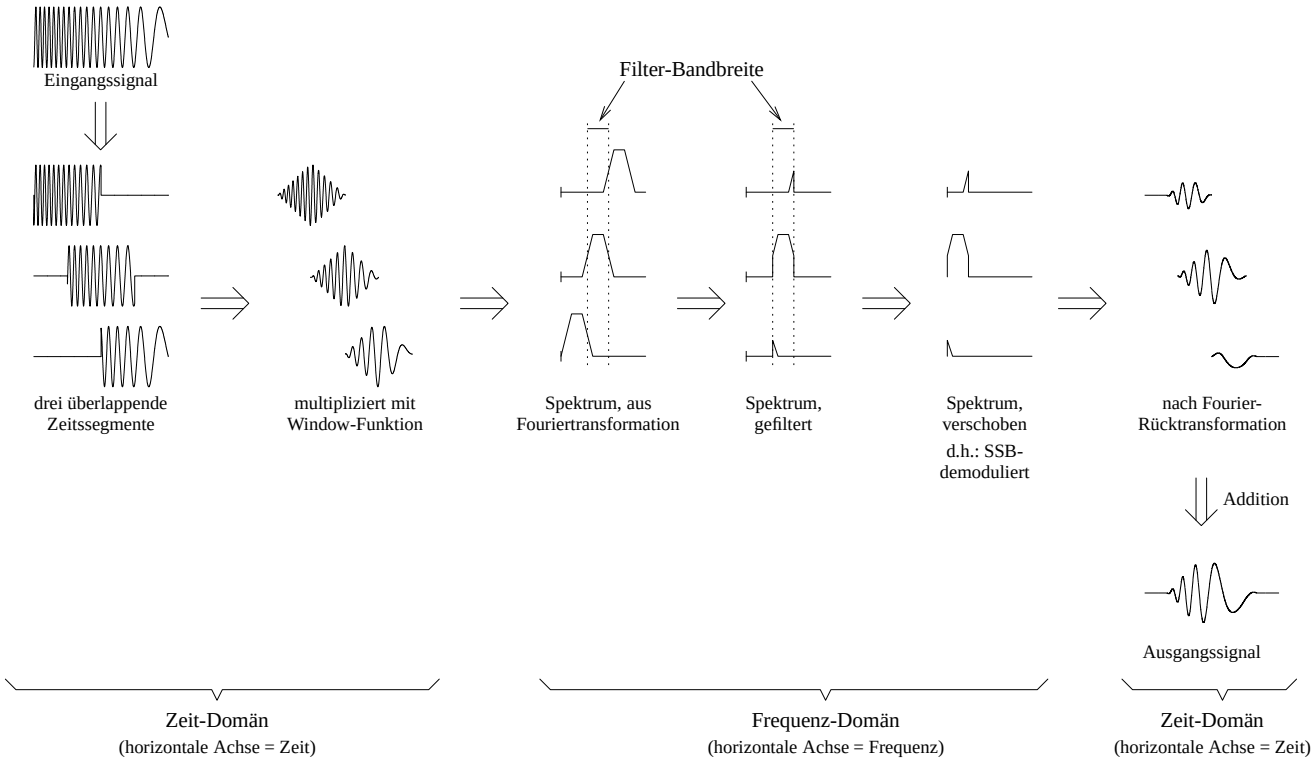

Abbildung 2: Ein SSB-Empfänger auf Basis von Fourier-Transformationen

Anm.: im Frequenzdomän muß nicht nur die Amplitudeninformationen betracht werden wie hier gezeigt, sondern auch die hier nicht gezeigten Phaseninformationen.

3 Software

Die Software im Rechner hat als Aufgabe, die Daten von 195 kHz Kurzwellenspektrum sinnvoll weiter zu verarbeiten. Erstens werden die Daten grafisch dargestellt, im Form eines Spektrum- und Wasserfall-Diagramm. Zweitens wird ein Teil des Spektrums selektiert und drittens wird das selektierte Teil des Spektrums demoduliert und hörbar gemacht.

3.1 Zeit- und Frequenzdomän

Die Daten die von der Empfängerhardware kommen, sind Daten im *Zeit-Domän*. Das heißt daß die einzelnen von der Hardware gelieferten Zahlen beschreiben wie groß das Signal auf aufeinanderfolgenden Zeitpunkten ist. Weil der Empfänger aber breitbandig ist, ist dieses Signal ein Gemisch von Signalen von vielen Funkstationen die alle innerhalb der Empfängerbandbreite senden. Um die Stationen zu trennen, ist es viel sinnvoller die Daten im *Frequenz-Domän* darzustellen, weil in Prinzip ja jeder Station ein bestimmter Frequenzbereich (innerhalb der Bandbreite unseres breiten Empfängers) zugewiesen ist. Bei der Darstellung der Daten im Frequenzdomän repräsentiert jeder Datenpunkt wie stark eine bestimmte Frequenz im Signal (über längere Zeit) enthalten ist. Wenn man die Daten im Frequenzdomän hat, wird es auch ziemlich einfach das Signal einer einzelnen Funkstation zu selektieren und nur dieses weiter zu verarbeiten (das heißt, zu demodulieren und hörbar zu machen).

Wie kommt man vom Zeitdomän zum Frequenzdomän (und zurück)? Dafür gibt es eine mathematische Rechnerei, die sogenannte Fourier-Transformation. Eine Variante davon, die speziell effizient von Rechnern ausgeführt werden kann, ist die sogenannte “Fast Fourier Transform” (FFT), die heute das Herzstück vieler Anwendungen digitaler Signalverarbeitung ist.

3.2 Prinzip und Beispiel der Signalverarbeitung mit Fourier-Transformation

Ein Beispiel davon, wie diese Transformation in Software eingesetzt werden kann um einen SSB-Empfänger zu realisieren, ist in Abbildung 2 gezeigt.

Linksoben ist das Eingangssignal gezeigt; in diesem Beispiel ist das ein einzelnes Sinussignal mit im Laufe der Zeit abnehmender Frequenz[1]. In Prinzip könnten wir dieses ganze Signal der Fouriertransformation zuführen; das würde allerdings heißen daß wir die Berechnung erst anfangen können nachdem das zu empfangene Signal beendet ist: eine nicht erwünschte Verzögerung. Deshalb möchten wir schon ein Teil (z.B. die erste 0,1 Sekunde) bearbeiten sobald das empfangen ist. Das ist gerade unter dem Eingangssignal gezeigt: wir teilen die Zeit in Segmente auf, die getrennt bearbeitet werden.

Als nächster Schritt werden die zeitliche Abschnitte mit einer s.g. Window-Funktion multipliziert; in diesem Beispiel ist diese Funktion dreieckig. Dies sorgt dafür daß das Signal im Mitten des Zeitssegments stark bleibt, aber an den Rändern schwach wird. Die Fast-Fourier-Transform "nimmt" nämlich "an" daß das Signal sich periodisch wiederholt, wie wenn die Enden des Zeitssegments aneinander gehängt wären; die Windowfunktion sorgt dafür daß dies nicht zuviel verzerrung bringt, indem das Signal an den Enden null gemacht wird. Jetzt ist auch klar, weshalb die Zeitsegmente überlappen müssen: sonst würden Teile des Signals (nämlich an den Rändern der Segmente) kaum in Betracht genommen.

Dann kann die Fourier-Transformation berechnet werden, für jedes Segment unabhängig. Das Ergebnis ist in der Abbildung grob skizziert; man sieht z.B. das im ersten Segment (wo die Frequenz des Eingangssignal ziemlich hoch ist), die Spitze im Spektrum mehr nach rechts (also, bei höherer Frequenz) liegt als beim letzten Segment (wo das Eingangssignal ja vor allem niedrigen Frequenzen enthält). Die so erhaltene Spektrums (ein Spektrum für jedes Zeitsegment) können als Spektrum- und Wasserfalldiagramme gezeigt werden.

Weil die Daten jetzt im Frequenzdomän vorliegen, ist es jetzt fast trivial ein bestimmtes Frequenzbereich heraus zu filtern: man setzt ganz einfach die Komponente die zu ungewünschten (also, weg zu filternden) Frequenzen gehören, auf 0, wie gezeigt in der nächsten Spalte der Abbildung.

In der nächsten Spalte wird das Signal als ein Einseitenbandsignal demoduliert: man schiebt einfach die Daten im Frequenzbereich so, daß der unterdruckte Träger bei 0 Hz kommt.

Nach Fourier-Rücktransformation sind dann wieder Signale im Zeitdomän verfügbar, und addierung (unter Inbetrachtnahme der Windowfunktion) der überlappenden Zeitsegmentteile liefert das entgültige Ausgangssignal. Da sieht man daß wir tatsächlich einen SSB-Empfänger haben: von dem ganzen "Sweep" im Eingangssignal, ist nur das mittlere Teil hörbar (weil nur das frequenzmäßig im Filterdurchlass liegt), und ausserdem ist das jetzt ein Sweep bis 0 Hz geworden.

Bemerkungen

Ein Vorteil des oben beschriebenen Verfahrens, ist daß das Filter sehr einsichtig ist: man deutet direkt im Frequenzdomän an, welche Frequenze man durchlassen möchte und welche nicht. Das so aufgebaute Filter ist allerdings nicht so gut wie es sein könnte. Dies zeigt sich darin, daß für bestimmte Kombinationen von Eingangssignalen und Filterbandbreiten die Ergebnisse der Fourier-Rücktransformationen für die aufeinanderfolgenden Zeitsegmente sehr verschieden sind und somit die Übergang vom einen zum nächsten Zeitsegment nicht ausreichend "glatt" verläuft. Diese Probleme können reduziert werden, indem man mehr überlappende Zeitsegmente nimmt: je mehr, desto besser, aber damit steigt auch der Rechenaufwand.

Prinzipiell besser ist es, ein s.g. FIR-Filter (Finite Impulse Response) zu verwenden (was evtl. wieder effizient mit Fouriertransformationen implementiert werden kann); weil diese Filter ohne Zeitseg-

[1] Das sieht vielleicht etwas artifiziell aus, aber solche Signale treten zum Beispiel auf wenn ein Funkamateur mit eingeschaltetem Sender über den Band dreht, um das SWR seiner Antenne zu messen.

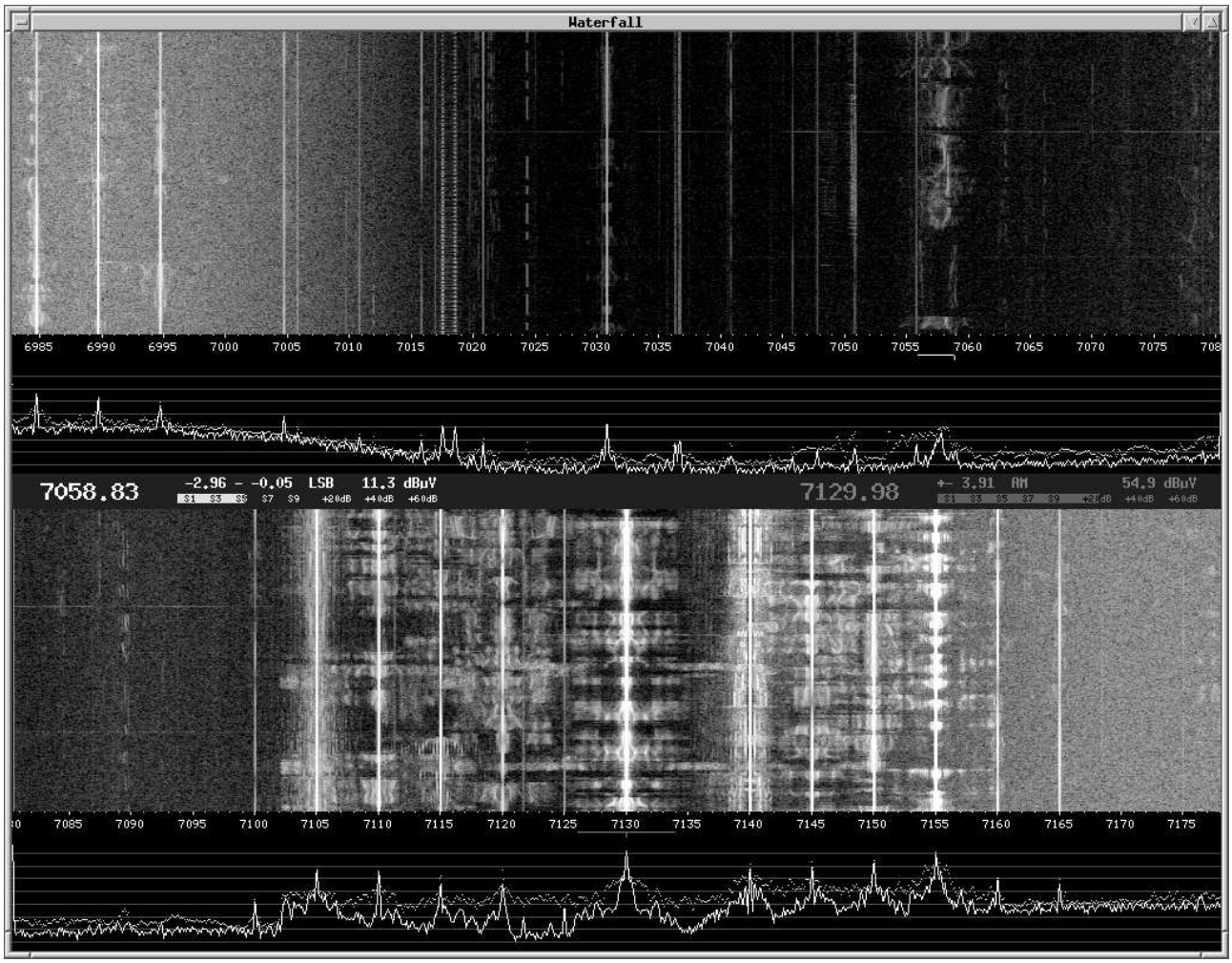

Abbildung 3: Bei 40 m: Funkamateure und Rundfunk

mentierung auskommen, treten die o.g. Problemen nicht auf; diese Filter sind allerdings weniger einsichtig.

In der Praxis hat sich gezeigt daß das beschriebene Verfahren eigentlich ganz gut funktioniert; und obwohl ich es anfangs nur als erstes Experiment implementiert hatte (weil die Zeitsegmentierung, Windowing und Fouriertranformation ohnehin schon für das Spektrum- und Wasserfalldiagram implementiert waren), habe ich noch keinen Grund gehabt auf FIR-Filter umzusteigen. (In meiner jetztigen Software werden FFTs mit 4096 Punkte gemacht, also mit Zeitsegmenten von etwa 21 ms, die zu 3/4 überlappen; es wird die Blackman-Windowfunktion verwendet.)

3.3 Das Wasserfalldiagramm

Die Benutzeroberfläche der Software wird von einem großen Spektrum- und Wasserfalldisplay dominiert; zwei Beispiele sind in Abbildungen 3 und 4 gegeben. Weil das empfangene Spektrum so breit ist, ist die Frequenzachse hier in zwei Teile geteilt: die niedrige 98 kHz sind oben, die obere 98 kHz unten angeordnet, beide mit einer Frequenzskala von links nach rechts.

In Abbildung 3 ist der Bereich von 6985 bis 7175 kHz gezeigt: also das gesamte 40-m-Amateurfunkband, und ein Teil des benachbarten Rundfunkbands. Ganz klar zu erkennen sind die starken Rundfunksender im 5 kHz Raster, mit symmetrischen Seitenbändern infolge der Amplitudenmodulation. Man sieht hier auch daß die Unterdrückung der Spiegelfrequenz noch nicht optimal ist: die Rundfunkträger sind auch unten im Amateurfunkband schwach sichtbar, was durch kleine Amplituden-

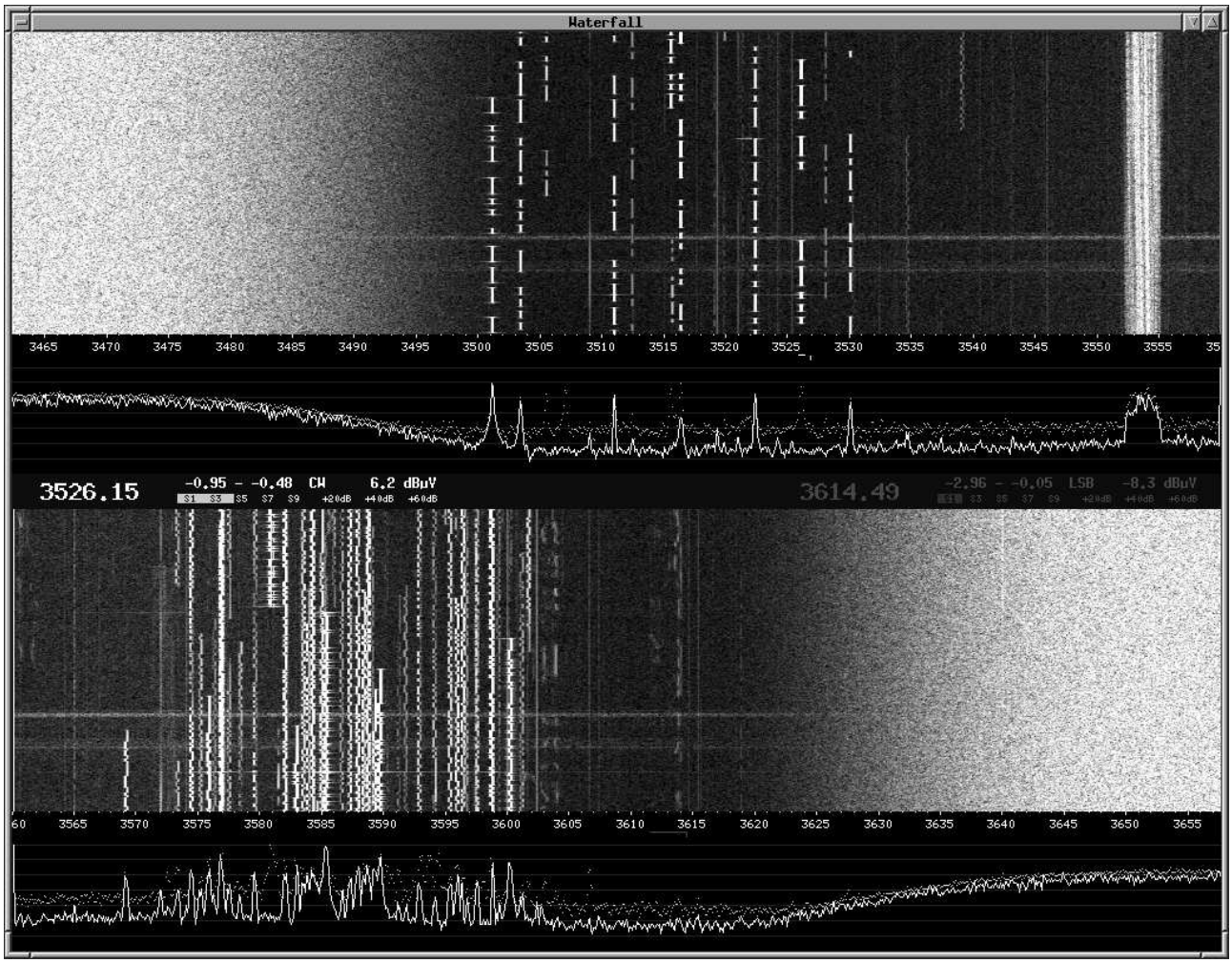

Abbildung 4: Morse und ein RTTY-Contest im 80m-Band

oder Phasendifferenzen zwischen den zwei Quadratursignalwegen verursacht werden kann.

In Abbildung 4 ist der Bereich von 3465 bis 3655 kHz gezeigt. Man sieht klar einige CW-Signale zwischen 3500 und 3530 kHz; das Signal bei 3500 kHz wird anscheinend zu hart geschaltet und ist demzufolge unnötig breit (“Keyclicks”). Weiter sieht man eine ganze Menge FSK-Signale (wahrscheinlich RTTY) zwischen 3570 und 3600 kHz, und bei 3614 kHz gibt es ein SSB-Signal. Das Signal mit fast rechteckigem Spektrum bei 3555 kHz ist vermutlich irgendeine moderne digitale Modulation, z.B. OFDM mit vielen Trägern wie u.a. von DRM benutzt wird; die langsame Änderung dessen Spektrums in der Zeit wird vermutlich von Propagation verursacht: selektives Fading.

Schließlich ist in beiden Abbildungen sichtbar daß das Rauschen an den Bandgrenzen steigt (und der Empfänger dort demzufolge weniger empfindlich ist): diese Unzulänglichkeit des verwendeten A/D-Wandlers wurde schon in Abschnitt 2.2 erwähnt.

3.4 Demodulatoren und Bedienung

Man will natürlich nicht nur sehen, was auf dem Band passiert, sondern auch noch mithören. Wie ein SSB-Signal in Software demoduliert werden kann, ist schon an Hand von Abbildung 2 gezeigt. Demodulation von AM-Signalen ist fast das gleiche, man muß nur beim Schieben im Frequenzdomän auch die beiden Seitenbänder addieren, unter Inbetrachtnahme der Phase des Trägers.

In z.B. Abbildung 3 ist zu sehen, wie der Demodulator dem Benutzer zugänglich gemacht wird. Der Demodulator ist hier abgestimmt auf ein Einseitenbandsignal bei 7059 kHz. Diese Frequenz wird

mitten links angezeigt (zusammen mit einer Signalstärkeanzeige), und eine horizontale Linie auf der Frequenzachse von 7056 bis 7059 kHz zeigt den Durchlaß des verwendeten Filters.

Die Abstimmung erfolgt entweder per Mouseclick (aber das ist ziemlich unbequem um genau auf einem Einseitenbandsignal abzustimmen), oder mit einem Drehknopf. Letzterer ist einfach realisiert durch ein altes Computermouse zu öffnen und den Kugel zu entfernen, so daß man die Encoderräder direkt von Hand betätigen kann: eins für die Frequenz und eins für die Filterbandbreite.

In Software kann man ohne viel Aufwand auch gleich zwei (oder mehr) Demodulatoren laufen lassen. So ist es möglich, zwei Frequenzen zur gleichen Zeit abzuhören, über die linken und rechten Lautsprecher der Soundkarte.

4 Ausblick

Das Gerät und die Software funktionieren jetzt, aber es gibt noch jede Menge Verbesserungsmöglichkeiten, entweder in Software oder für eine künftige neue Version der Hardware.

- Zunächst sollte die Abschirmung im Gerät verbessert werden. Im Augenblick gibt es noch kaum Abschirmung zwischen dem digitalen und dem analogen Teil der Hardware (weil ich zunächst das Ganze überhaupt mal zum Laufen bringen wollte), und so gerät noch zuviel Störung aus dem digitalen ins analoge Teil. Dies begrenzt natürlich den Dynamikbereich.
- Mehr Demodulatoren in der Software, z.B. für PSK-31, RTTY, DRM, u.s.w., oder (besser) eine Schnittstelle um existierende Software für solche Modulationen anzubinden.
- Besserer A/D-Wandler: vielleicht gibt es mittlerweile A/D-Wandler mit noch höherer Geschwindigkeit bei ausreichendem Dynamikbereich.
- Ethernet-Sender mit korrekter Prüfsumme: die jetztige Hardware kann, wie oben erwähnt, die Prüfsumme der Ethernetframes nicht ausreichend schnell berechnen; deshalb muß der Rechner die Prüfsumme ignorieren. Dafür muß man den Treiber für die Ethernetkarte im Linuxkernel modifizieren, wenn die Karte das schon überhaupt kann. Es wäre besser, den Ethernet-Sender in einer FPGA (programmierbarem Logik-Baustein) zu implementieren, statt in einem Mikrokontroller: in einer FPGA kann man mit wenig Aufwand die CRC errechnen, und auch alle andere Timing-Logik die jetzt um den Mikrokontroller benötigt wird, könnte in der FPGA programmiert werden. Dann wäre 100 Mbit/s Ethernet auch machbar, um einen schnelleren A/D-Wandler zu unterstützen.
- Schießlich könnte man auch überlegen, den VCO + PLL + Oktavteiler von etwas modernerem zu ersetzen, z.B. einer DDS, vielleicht auch in FPGA implementiert.

Literatur

[1] Gerald Youngblood, AC5OG, *A Software Defined Radio for the Masses*, QEX July/August 2002 – March/April 2003.

[2] Martin Klaper, HB9ARK, *Next Generation Transceiver: SDR-1000 – Ein Software Defined Radio für alle*, 49. UKW-Tagung Weinheim, 2004.

[3] Einige Amateur-SDR-Projekte im WWW:
`http://antennspecialisten.se/~sm5bsz/linuxdsp/linrad.htm`
`http://www.gnu.org/software/gnuradio/`
Website des Autors: `http://wwwhome.cs.utwente.nl/~ptdeboer/ham/sdr/`