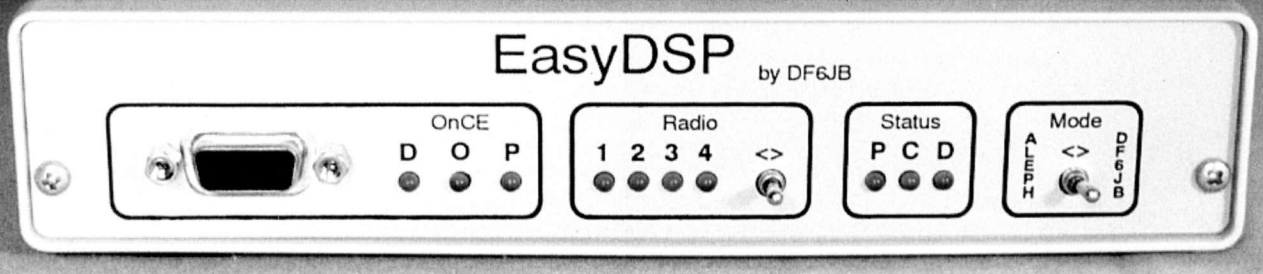
Ein universelles DSP-Zusatzgerät (1):

EasyDSP für Funkamateure



Ulrich Bangert, DF6JB

In zwei Teilen, etwas ausführlicher als sonst, geht es um den Werdegang eines DSP-Gerätes. Von der Idee über alle Stolpersteine hinweg bis zur fertigen „Höllenmaschine".

Elektronische Schaltungen entstehen nicht aus einem Vakuum heraus. Besonders, wenn die Schaltung ein gewisses Maß an Komplexität überschreitet, ihre Entwicklung sich also über Wochen, Monate oder gar, wie in diesem Fall, Jahre hinzieht, gibt es zahlreiche Einflüsse auf den Entwickler, die dazu führen, daß er bestimmte Dinge nun gerade so und nicht anders macht. Fachkollegen melden sich mit einem Kommentar, der wieder Anlaß zu Änderungen gibt, Diskussionen entspannen sich, der eine meint dies, der andere das genaue Gegenteil davon, der eine sagt: „Das wird zu teuer", der andere will unbedingt noch zusätzliches eingebaut haben. Der dritte wiederum erklärt, daß es hirnrissig ist, was man vorhat, und man sich daher der Gattung der Primaten zu Unrecht zugehörig fühle.
Meistens erfährt man über diesen Prozeß wenig und bekommt nur das fertige Produkt präsentiert. Eigentlich schade! EasyDSP hat eine solche Historie, und da ich es interessant finde, auch von anderen über die Geschichte ihrer Entwicklungen zu lesen, mache ich den Anfang. Wer sich nicht dafür interessiert und unmittelbar „Hardcore-Technik" lesen will, der überschlägt einfach diesen Teil!

Wie alles begann

Also: Gegen Ende 1991 hatte ich einen Faxkonverter konzipiert, dessen Merkmal ein eingebauter Mikrocontroller war. Er war optimal für die Zusammenarbeit mit dem JVFAX-Programm meines Freundes Eberhard Backeshoff, DK8JV, ausgelegt. Dieser Konverter hat unter dem Namen „EasyFax" einiges an Furore gemacht und ist weltweit in einigen tausend Exemplaren aufgebaut worden. Seitdem war es allerdings still geworden, und meine berufliche Entwicklung vollzog sich immer mehr zum reinen „Softie".
So kam es, daß ich irgendwann 1994 wieder etwas „basteln" wollte. Allein: Ich hatte keine gute Idee, was es sein sollte. Der Erfolg des EasyFax-Konverters war sicher darauf zurückzuführen, daß es zu einem hervorragendem PC-Programm (dem JVFAX) nun eine Hardware – eben EasyFax – gab. Das überzeugte mich, daß ich es erneut so anfangen müsse. Damit war ich auf einmal schlagartig bei Hamcomm!

Hamcomm

Dieses äußerst verbreitete RTTY-Programm von Django Schroeder, DL5YEC, erhält die Daten, die es verarbeitet, durch Einfügen eines Hamcomm- oder „Simpel"-Konverters in die NF-Leitungen des Transceivers. Dieser verstärkt das NF-Signal lediglich und begrenzt es hart, so daß es pegelangepaßt der RS232-Schnittstelle eines gewöhnlichen PCs zugeführt werden kann. Das eigentliche Decodieren geschieht im PC. Dieses Konzept hat neben dem Vorteil, daß es preisgünstig nachzubauen ist, auch einen Nachteil.
Wer in den siebziger Jahren aufmerksam verfolgt hat, was Hajo Pietsch, DJ6HP, zum Thema RTTY verfaßt hat [1], dem sind die Grenzen einer solchen Schaltungstechnik klar: Am Ausgang eines Begrenzers erscheint immer das stärkste Eingangssignal. Sprich: Damit man damit RTTY dekodieren kann, braucht man gute bis sehr gute Signal/Störabstände. An die Leistungsfähigkeit der von DJ6HP beschriebenen Filterkonverter kommt man mit Simpelmethoden nicht heran!
Andererseits ist ein Filterkonverter nach DJ6HP ein Gebilde aus vielen Operationsverstärkern sowie diskreten Bauelementen und bedarf somit zum Abgleich eines mittleren Meßparks. Hat man den erfolgreich absolviert, dann liegt ein optimaler RTTY-Konverter vor, allerdings festgelegt auf eine ganz bestimmte Mark/und Space-Frequenz und eine bestimmte Baudrate.
Das ist alles andere als universell, als es etwa der Simpel-Konverter darstellt. Er funktioniert mit jeder Baudrate und mit jeder Mark/Space-Kombination – d. h. wenn er wegen des vorgegebenen Signal/Störabstandes überhaupt funktioniert.
So verdichtete sich immer mehr der Plan, einen RTTY-Konverter zu konzipieren, welcher die positiven Eigenschaften der Simpel-Schaltung mit denjenigen eines Filterkonverters vereinigen sollte.
Da uns die Technik nun digital einstellbare und abgleichlose Filter als Switched-Capacitor-Filtern beschert hat, lag auf einmal alles auf der Hand. Mein Konverter würde die Simpel-Schaltung vollständig enthalten.
Hamcomm wäre also in der Lage, die Signaleigenschaften einer RTTY-Übertragung (Mark/ und Space-Frequenz, Baudrate) automatisch zu ermitteln. Ist dies geschehen, gibt Hamcomm an den Konverter den Befehl, sich nun in einen

optimalen Filterkonverter für exakt diese Signalparameter zu verwandeln.
Das Signal wird nicht mehr innerhalb Hamcomms, sondern im Konverter in ein demoduliertes digitales Signal gewandelt. Der kann dies als echter Filterkonverter auch besser.
Man erhält eine Decodierung in Filterkonverter-Qualität, gepaart mit der einfachen Bedienung des Hamcomm-Programms. Heureka!
In der Ansicht, etwas absolut Revolutionäres erdacht zu haben, verabredete ich einen Treff mit Django auf der Interradio '94, um ihm mein Konzept zu erläutern.
Es verlief etwas anders, als ich es mir vorgestellt hatte. Django ist ein umgänglicher Mensch, der sich zunächst einmal in Ruhe anhörte, was ich zu sagen hatte („....ist ja ganz schön und gut...."), mir aber unmißverständlich klarmachte: „Was Du vorhast, ist eine gute Idee, die man weiterverfolgen sollte. Aber: Das macht man doch heute nicht mehr mit dazu speziell hergestellter Hardware, sondern mit Software auf einem digitalen Signalprozessor!"

DSP: Das Maß der Dinge

So auf den Boden der Tatsachen zurückgeholt, versprach ich ihm, das Ganze auf seine Machbarkeit mit einem DSP-Prozessor hin zu prüfen.
Das war zu diesem Zeitpunkt ganz schön mutig, weil meine DSP-Erfahrung bislang darin bestand, daß ich mir von Hartmut, DL1YDD, ein Texas-Instruments-DSP-Starter-Kit ausgeliehen, mehrere Monate im Regal liegengelassen und dann zurückgegeben hatte. Unter Erfolgsdruck beschaffte ich mir schnell ein eigenes DSP-Starter-Kit. Es hatte einen leistungsfähigeren Prozessor – einen TMS320C50 – als das, was ich ausgeliehen hatte.
„So! Nun woll'n wer mal schnell die Software dafür schreiben!" dachte ich. Wer mit DSP-Programmierung anfängt, steht vor zwei Problemen, deren zweites man zumeist unterschätzt:
Man muß
a) sich in die Assemblersprache des gewählten Prozessors einarbeiten und
b) die der DSP-Programmierung zugrunde liegende „Signaltheorie" lernen.
Während a) in die Kategorie „machbar" fällt, ist b) ein echter Hemmschuh. Signaltheorie ist ein Wissenszweig, mit dessen Studium man sich etliche Universitätssemester beschäftigen kann. Programmiererfahrung, egal in welcher Sprache und auf welchem Prozessor, ist eher nutzlos. Wer nicht Signaltheorie selbst studiert hat, sollte sehr gute Mathematikkenntnisse mitbringen.
Komplexe Zahlen, Fourierintegrale, Z-Transformationen, Konvolutionen und Korrelationen, noch nie gehört? Dann tut man sich schwer darin, einen DSP-Prozessor erfolgreich zu programmieren.
Ich möchte keine Angst vor der DSP-Programmierung schüren. Im Gegenteil: Es liegt in meinem Interesse, wenn sich möglichst viele finden, die selber Software für mein EasyDSP entwickeln. Einleitendes Literaturstudium ist aber nötig.

Signaltheorie? Wie bitte?

Ich kannte zwar Begriffe der Signaltheorie aus meinem Studium, aber das liegt nun auch fast zwanzig Jahre zurück. So war das Wiederholen aufwendig.
Hinzu kam auch ein Eindruck, den ich schwer beschreiben kann: Ich wurde mit dem TI-Prozessor nicht richtig „warm" und hatte ständig das Gefühl, das falsche Stück „Silicon" unter den Fingern zu haben. TI ist mit großem Abstand Marktführer bei DSP-Prozessoren, also weiß man dort mit Sicherheit, wie man sie zu entwickeln hat. Es lag also eindeutig an mir selbst, aber je mehr ich mich mit dem TI-Produkt beschäftigte, desto weniger konnte ich das sicher gute Stück leiden.
Beides, die Signaltheorie und das sich nicht entspannende Verhältnis zum TI-DSP-Kit, führte dazu, daß mein Enthusiasmus immer mehr verflog, und ich werkelte eher lustlos herum. Froh war ich, daß ich Django gegenüber nichts absolut verbindlich gesagt habe, denn ich war drauf und dran, mich zu blamieren.

Einsteiger-Kit Retter in der Not

Dann hörte ich, daß Motorola ein DSP-Einsteigerkit für ein eigenes Halbleiterprodukt anbiete, den DSP56002EVM. Dieser DSP-Baustein beherbergt einen 56002-Prozessor mit 40 MHz Taktfrequenz. „Einen Versuch hast du noch", dachte ich und beschaffte mir das Teil.
Erneut war ein fremder Assembler zu erlernen, aber dann geschah so etwas wie ein Wunder: Ich verstand auf einmal, warum der Prozessor so und nicht anders gebaut war. Es wurde klar, was die Designer im Auge gehabt hatten, als sie ihm einen Befehl wie „macr x0,y0,a (r0)+,x0 (r1)–" in die Wiege legten.
Die Vorzüge eines echten, eingebauten Hardware-UART wurden schätzenswert, und der OnCE-Port – ein spezieller Anschluß für das Fehlersuchen in Programmen (Debuggen) – entpuppte sich gegenüber einem echten In-Circuit-Debugger als fast ebenbürtig.
Wenn sich dann die Motorola-Entwickler im Handbuch dafür entschuldigen, daß es ihnen – anders als gewohnt – bei diesem Prozessor leider nicht ganz gelungen sei, einen orthogonalen Befehlssatz zu verwirklichen, dann möchte man sie als mit Intel-Spezialbefehlen gebeutelter Programmierer dafür fast schon küssen.
In den 56002 habe ich mich spontan „verliebt", wenn Sie so wollen. Das bedeutet nicht, daß der „TI" schlechter ist.

Popelige Platine

Derartig neu motiviert, machte ich mich daran, das aus Kostengründen etwas „popelig" bestückte Experimentierbrett in einen benutzbaren Zustand zu bringen. So erhielt die DSP-Platine eine solide Grundplatte aus Kunststoff, und ich montierte „vernünftige" BNC- und Cinch-Buchsen sowie einige Taster für Interrupts auf Metallwinkel.
Dieses beinahe professionell anmutende Stück präsentierte ich Django auf der Interradio '95 – zusammen mit der Aussage, daß eine Weiterarbeit am DSP-Projekt, wenn überhaupt, auf diesem Prozessor basieren würde.
Wieder war Django nicht sehr begeistert, weil er zwischenzeitlich selber Erfahrungen beim Programmieren von TI-DSPs gemacht hatte. Der Motorola-Prozessor war für ihn Neuland, und er könne ihn nicht ad hoc programmieren.
Auch ein DL5YEC konnte mir nicht meine neue „Geliebte" ausreden, und er wünschte mir alles Gute und entließ mich zu einem weiteren Jahr der Planung und Experimente für „meinen" DSP. Mittlerweile war ich so sehr von dem Projekt gefangengenommen, daß ich jedem auf Befragen die Eigenschaften „meines" Gerätes schilderte, das es dies zu diesem Zeitpunkt noch gar nicht gab.
Nachdem der 56002 als Prozessor für mein Projekt feststand, war es nun nötig zu erfahren, was andere Menschen mit diesem Ding angefangen hatten. Ich stieß auf zwei Namen: Johan Forrer, KC7WW [2], sowie Jarkko Vuori, OH2LNS [3].
Johan hatte in der Augustausgabe 1995 der QEX den Artikel „Using the Motorola DSP56002EVM for Amateur Radio DSP Projects" veröffentlicht [4]. Auch die umfangreiche Liste früherer DSP-Publikationen weist ihn als brillanten Techniker aus. Das, obwohl er als Doktor der Forstwissenschaften beruflich absolut fachfremd ist.

Kontakte gesucht

Kurz vor Weihnachten 95 schrieb ich einen Brief mit einer Skizze dessen, was ich vorhatte. Sicherheitshalber legte ich einige „Beweise" dafür bei, daß gewisse Erfahrungen in Elektronikdesign vorhanden

waren, damit er nicht glaubte, daß ein komplett Verrückter sich meldet.
Denn, wie hatte er doch in seinem Artikel geschrieben: „...mastering a complex processor like the Motorola 56002 DSP is a formidable task. This is not something for the faint of heart." Na ja, hoffentlich glaubte er mir, daß ich nicht zu den besagten Feiglingen gehöre. Ich schrieb ihm sinngemäß: „Hallo Johan, ich will einen Kasten für Funkamateure konstruieren, der auf einem 56002 beruht. Wie sollte Deiner Meinung nach solch ein Kasten aussehen? Beste Grüße aus Germany."
Die Wartezeit auf seine Antwort kam mir viel länger vor, als sie wirklich war. Er war erfreut über mein Vorhaben und bot mir Hilfe an, fragte aber nach meiner E-Mail-Adresse. Die hätte ich wohl vergessen zu erwähnen. Denn: Per „Snail Mail" – zu deutsch Schneckenpost" – könne man doch kaum eine Diskussion führen.

E-Mail ist Pflicht

Ich hatte geglaubt, die Entwicklung an sich sei bereits aufwendig genug, und da kommt der Kerl mit „E-Mail". Das konnte ich höchstens buchstabieren. Wieder war Hartmut, DL1YDD, Helfer in der Not, und nur wenige Stunden später kannte mich das Internet.
Mit Johan entspann sich dank Internet und der Zeitverschiebung zwischen der amerikanischen Westküste und good old Germany ein für die CQ DL zu seitenschindender Dialog: Wenn ich ihm abends eine E-Mail schickte, konnte ich im Laufe des Tages mit einer Antwort rechnen. Die Ideen Johans wurden in EasyDSP realisiert. Er hatte auch welche, die unrealisierbar waren, aber in einem Brainstorming nahmen wir darauf keine Rücksicht.

Minimalismus als Basis

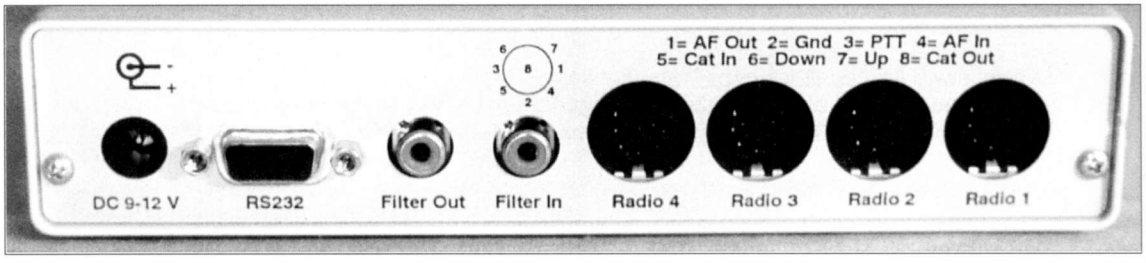
Die Geschichte von EasyDSP wäre unvollständig erzählt, wenn man nicht von Jarkko Vuori, OH2LNS, und Kaj Wiik, OH6EH [5], reden würde. Diese beiden haben im Umfeld der technischen Universität Helsinki bereits 1986 die „Alef Null"-Gruppe gegründet und mehrere DSP-Designs veröffentlicht.
Ihr jüngstes Produkt ist die DSP Card 4, welche auf einem 56001 beruht – einem Vorgänger des 56002 [6]. Die beiden sind Anhänger eines gelebten „Minimalismus", der sich darin äußert, daß die DSP Card 4 nur unbedingt notwendige Bauteile enthält, sonst nichts. Insofern hat das Ähnlichkeit mit den Evaluation Boards. Ich wollte mehr.
Kaj und Jarkko hatten viele Programme für ihre Platine geschrieben, u. a. ein Monitorprogramm namens Leonid, das in einem Flash-EEPROM residiert. Auch mit Jarkko ergab sich eine Diskussion.
Der Knackpunkt für mich: Würde ich mein Projekt kompatibel zur DSP Card 4 gestalten, so könnte ich einen Software-Fundus nutzen, bräuchte also nicht bei Null anzufangen. Dies hätte allerdings einen teilweisen 1:1-Nachbau der DSP Card 4 bedeutet.
Es waren aber auch einige mangelhafte Details darauf, die man auf das frühe Erscheinungsdatum der Schaltung – nicht etwa auf mangelnde Kompetenz der Entwickler! – zurückführen muß. Also: kompatibel oder doch lieber ganz von vorn? Wieder kam ich nicht richtig weiter.

Die Lösung: traumhaft

Manchmal träumt man die Lösung eines schwierigen Problems – so traumhaft erschien mir träumend ein Gerät. Zwar waren keine Details zu erkennen, aber ganz klar war ein Kippschalter auf der Frontplatte, der mit „Aleph" und „DF6JB" beschriftet war.
Das könnte es sein! Ein Gerät, das per Kippschalter eine DSP Card 4 in ein DSP nach DF6JB verwandelte und umgekehrt. Die Zutaten waren fast identisch, nur daß sie unterschiedlich verknüpft waren.
Das Ergebnis langen Nachdenkens darüber war, daß man einen ganzen Berg an TTL-MSI-Funktionen benötigen würde. Wörtlich zu nehmen.
Darum käme man nur durch Verwendung von GALs herum. GAL steht nicht für „Grüne Alternative Liste," sondern für „Generic Array Logic". GALs kann man an jeder Straßenecke kaufen, vereinzelt sollen sie sogar schon in Schaltungen gesichtet worden sein, die in der CQ DL veröffentlicht wurden.
Ich würde aber ein GAL mit mehreren hundert bis tausend Gatterfunktionen benötigen. Ich besorgte mir Unterlagen aller möglichen GAL-Hersteller und war danach dümmer als vorher.

Gal-Spezialist gesucht und gefunden

Ich rief Arno, DH4KAH, an. Ihm vertraue ich in Sachen Digitalelektronik blind, und er setzt GALs ein, wußte also etwas darüber. Er konnte mir immerhin sagen, daß die Software, welche er selber zum GAL-Design benutzt (Palasm von AMD), einfach zu erlernen ist und kein allzu großes Loch in die Hobbykasse reißt. Man kann auch einige 10 Kilo-DM dafür ausgeben.
Mit der Software spielte ich. Dabei stellte sich heraus, daß ein riesiges und sündhaft teures GAL nötig sein würde, wollte ich wirklich alles in einen Baustein packen.
Gott sei Dank gelang es, die benötigten Funktionen so zu splitten, daß sie sich in zwei mittelgroßen und deutlich preisgünstigeren GALs realisieren ließen, nämlich den Mach210 von AMD.
Jarkko erlaubte mir, alles, was er jemals für die DSP CARD 4 programmiert hatte, frei zu nutzen, wenn ich nur die Urheberschaft kenntlich machen würde. Das war selbstverständlich.
Alles gewann an Form. Ein „winziges" Problem war da noch: KC7WW hatte darauf bestanden, daß mein Gerät über einen OnCE-Port verfügen müsse. Der würde es als Plattform zur Entwicklung den EVM-Boards ebenbürtig machen.



OnCe again?

Vom 56002 aus gesehen, ist der OnCE-Port eine Anordnung aus sieben, teilweise bidirektionalen TTL-Leitungen. Sie müssen nach einem komplizierten Verknüpfungsmuster bedient werden [7]. Der Benutzer des EVM-Boards sieht hingegen nur eine serielle Schnittstelle, mit „OnCE port" beschriftet. Da muß also noch Elektronik dazwischen sein, und richtig: So ist es auch.
Als Vermittler zwischen diesen Welten werkelt auf dem EVM eine kleine Unterart des Motorola-68705-Mikrocontrollers. Die Assemblerquelle seiner Programmierung liegt dem EVM gleich bei. Mein ansonsten sehr universelles Programmiergerät konnte aber just diesen Baustein nicht programmieren!
Das Programm mußte also für einen anderen Prozessor umgeschrieben werden.

Bereits der Autor bezeichnete es als Alptraum, was die Weiterentwicklung angeht, und so war es auch. Es blieb nichts, als die be...scheidene Assemblerquelle Stück für Stück in Pascal (!) zu übersetzen und zu einem Compilat für 8031-kompatible Mikrocontroller zu verarbeiten.

Seitdem werkelt ein in Pascal programmierter ATMEL89C2051 bei mir als OnCE-Controller. Damit sind die Entwicklungsschwierigkeiten im wesentlichen geschildert.

Natürlich gab es kleine Schwierigkeiten beim Beschaffen der Bauteile. So benötigt der AD/DA-Wandler einen Quarz mit der Frequenz 24,576 MHz. Das ist ein Standardwert, und in der Tat: Bekommt man an jeder Ecke, also flugs 100 Stück davon gekauft. Einen setze ich in meinem EVM ein: Nichts geht mehr! „Ok, defekter Quarz, kommt vor bei Massenprodukten!" Ich nehme den nächsten: Gleiches Resultat, auch bei Änderung der Parallel-Cs. Der Quarz schwingt bei ungefähr 8 MHz. Nach den Wandler-Unterlagen will dieser unbedingt einen Grundwellenquarz.

Quarze und Geheimbünde

Nun haben sich aber alle deutschen Quarzhersteller in einem schwarzen Geheimbund zusammengetan, dessen oberstes Statut besagt, daß alle Serienquarze ab 20 MHz als Oberwellenquarze – nicht Grundwellenquarze –für die dritte, fünfte oder noch höhere Oberwelle gefertigt werden müssen.

Heftiges Herumtelefonieren fördert die Erkenntnis zutage, daß „natürlich" und „selbstverständlich" jeder Hersteller in der Lage ist, Grundwellenquarze mit dieser Frequenz zu produzieren; gegen ein „lächerlich" geringes Entgelt pro Quarz bereit sei, von obigem Statut abzusehen.

Ich werde bei einer Firma in Kalifornien (!) fündig. 1,50 US-Dollar pro Quarz. Insgesamt mit Zoll und Verpackung etwa 7 DM, aber immer noch billiger als die deutschen Geheimbündler.

Als Django den fertigen Kasten auf dem „Interradio-Nachfolger '96" dann sah, schien er doch beeindruckt. Es wird sicher bald eine Hamcomm-Version geben, die maximalen Nutzen aus der EasyDSP-Hardware zieht, denn wir vereinbarten eine Zusammenarbeit.

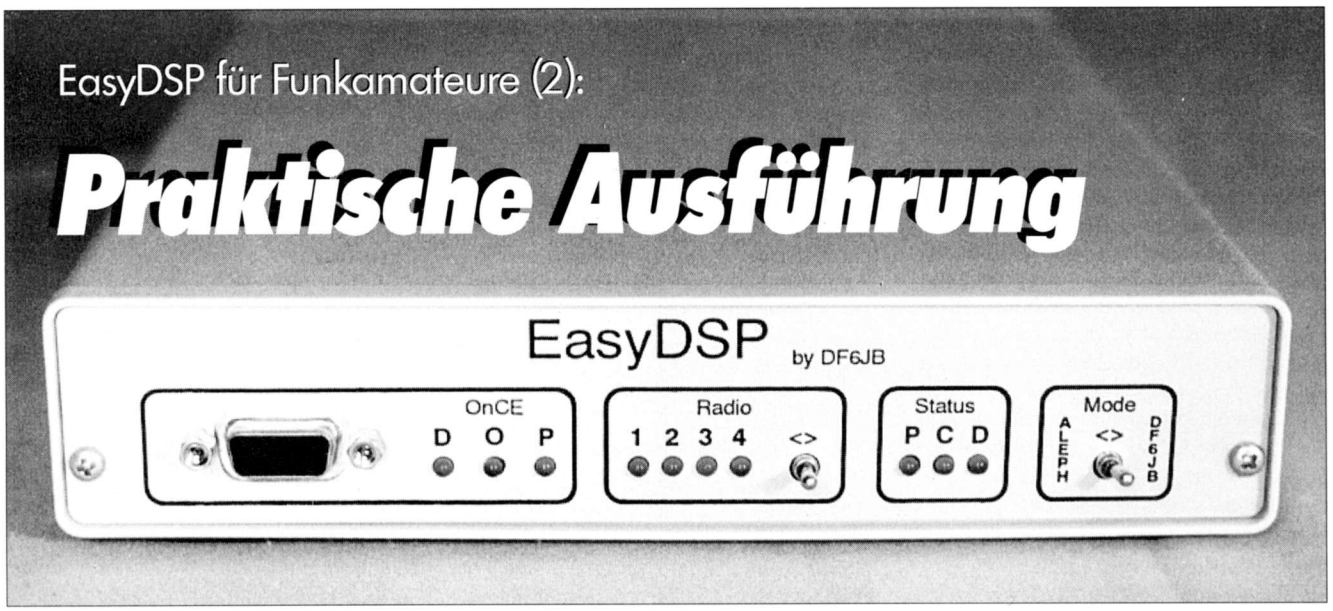
(wird fortgesetzt)

Literatur

[1] H.-J. Pietsch: Amateur-Funkfernschreibtechnik RTTY, Franzis-Verlag, RPB-Band 25
[2] forrerj@peak.org
[3] Jarkko.Vuori@hut.fi
[4] Johan Forrer: Using the Motorola DSP56002EVM for Amateur Radio DSP Projects, QEX August 1995
[5] kwi@cc.hut.fi
[6] DSP-Card4: FTP-Server nic.funet.fi
[7] Motorola: DSP56000 Digital Signal Processor Family Manual, Kapitel 10

EasyDSP für Funkamateure (2):

Praktische Ausführung



Ulrich Bangert, DF6JB

Nach soviel einleitenden Worten kommen wir nun zur Technik. Daß tatsächlich ein Gerät aus all diesen Vorüberlegungen entstanden ist, mag man den Bildern entnehmen. Klaus, DG2XK, der Fotoautor, hat EasyDSP bereits für sein Buch über SSTV-Technik ausführlich getestet [8].

Das Schaltbild von EasyDSP erstreckt sich über acht DIN-A4-Seiten. Interessenten schicke ich die Schaltungsunterlagen gegen einen mit 3 DM freigemachten C5-Umschlag mit eigener Anschrift gern zu. Der Schaltplan zeigt zwar alle Details, diese aber weit verzweigt. Deshalb ist er nicht sehr gut zum Verständnis des Gesamtkonzeptes der Schaltung geeignet. Hierzu betrachten wir lieber das Blockschaltbild.

CPU: Der Kern der Kirsche

Die CPU DSP56002 bildet den Kern der Schaltung. Rechts davon ist der RAM- und EPROM-Speicher zu erkennen. Da es sich um einen Prozessor mit 24 bit Wortbreite (= Datenbusbreite) handelt, bilden jeweils drei zu 32k×8 organisierte schnelle Cache-RAMs einen 32 kWorte großen Speicherbereich. Diese drei RAMs sind jeweils übereinander gezeichnet.
Während herkömmliche Prozessoren meist in der „Von-Neumann-Architektur" aufgebaut sind, bei der Programmspeicher und Datenspeicher physikalisch über den gleichen Adreß- und Datenbus angesprochen werden, sind Signalprozessoren fast alle in der „Harvard-Architektur" ausgeführt.
Dies bedeutet, daß für Programm- und Datenspeicher jeweils eigene Busse vorhanden sind, so daß der Prozessor parallel, also echt gleichzeitig, ein neues Datenwort und den dazu gehörigen Befehl holen kann. Diese Aufteilung kann soweit gehen, daß der Datenspeicher in zwei unabhängige Bereiche gesplittet wird, weil viele DSP-Algorithmen besonders schnell ablaufen, wenn sie parallelen Zugriff auf zwei Datenspeicher haben
Beim 56002, der in Harvard-Architektur aufgebaut ist, unterscheidet man zwischen P-Speicher (Programmspeicher), X-Speicher (Datenspeicher, über X-Adressierung angesprochen) und Y-Speicher (Datenspeicher, über die Y-Adressierung). Die „echte" Harvard-Architektur mit getrennten Bussen ist nur innerhalb des Chips realisiert. Der externe Speicher muß sich einen gemeinsamen Adreß- und Datenbus teilen. Trotzdem ist es wegen der dann möglichen Programmiertechniken sinnvoll, auch beim externen Speicher auf eine strenge Trennung der drei Kategorien zu achten. Evaluation Boards ermöglichen sie nicht, weil aus Kostengründen diese drei Speicherkategorien nicht streng getrennt sind.

In der Schaltung von EasyDSP stehen für X, Y und P-Speicher je drei 32 kWorte große Bereiche zur Verfügung, also insgesamt 288 kByte. Das reicht, um auch große DSP-Programme zu entwickeln. Damit die CPU ohne Waitstates auf das externe RAM zugreifen kann, muß dies ausreichend „schnell" sein.
Dies gilt nicht nur für die Zugriffszeit der RAMs an sich, sondern auch für die Adreßdecodierung, weil sich ihre „Propagation Delay" genannte Durchlaufverzögerung für Signale zu den Zugriffszeiten der RAMs addiert. Es ist leicht einzusehen, daß mit zunehmender Taktfrequenz der CPU immer wenige Zeit für den RAM-Zugriff zur Verfügung steht. In dieser Schaltung werden RAMs mit einer Zugriffszeit von 15 ns benutzt.

66 MHz kein Problem

Zur Adressdecodierung dient ein Gal 16V8 mit einem definierten Propagation Delay von 7 ns. Mit dieser Bestückung und einer für 40 MHz (!) spezifizierten CPU wurde festgestellt, daß das Board sich bis 70 MHz (!) Taktfrequenz sicher betreiben läßt. Es steht daher fest, daß mit 12-ns-RAMs, einem 5-ns-Decoder-Gal und einer echten 66-MHz-CPU ein absolut stabiler Betrieb bei 66 MHz möglich ist. Die 80-MHz-Version der CPU lag zum Testen noch nicht vor, aber bei 66 MHz ist wohl das Ende der Fahnenstange noch nicht erreicht. Das könnte auch an den „sauberen" Betriebsspannungen liegen, aber auch an der Multilayerbauweise der Platine sowie den fast siebzig (!) an wichtigen Punkten plazierten Stütz- und Abblockkondensatoren.

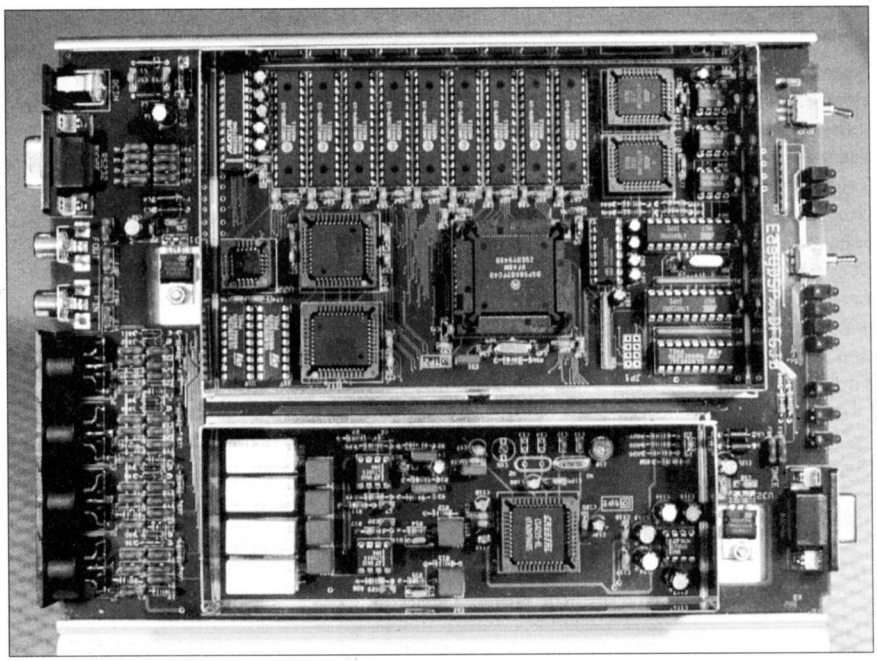
EEPROMS als Gedächtnisstütze

Rechts neben den drei RAM-Bänken findet man zwei einzelne 32 k × 8 Bausteine, die nicht in einer 24-bit-Anordnung gruppiert sind. Hierbei handelt es sich um EPROMs, genauer gesagt Flash-EEPROMs vom Typ AT29C256, aus denen die CPU nach einem Reset „bootet".

Welches der beiden EEPROMs zum Booten benutzt wird, entscheidet das Adressdecoder-Gal anhand des Mode-Signals, welches von der Frontplatte über den DF6JB/ALEPH-Mode-Umschalter erzeugt wird. Die CPU kann quasi aus den beiden EEPROMs zwei unterschiedliche „Betriebssysteme" laden.
Der clevere Leser mag sich nun wundern, wieso die Boot-Software in einem bytebreiten EPROM vorliegt und nicht etwa in 24-bit-Wortbreite. Die Motorola-Entwickler sorgten dafür, daß man mit üblichen Bausteinen und Mitteln arbeiten kann, und die Bootroutine setzt selbständig aus drei aufeinanderfolgenden Bytes des EEPROMs ein 24-bit-CPU-Wort zusammen. Die EEPROMs sind in der Schaltung programmierbar.

Fehlersuche at OnCe

Unterhalb der CPU erkennt man den bereits erwähnten OnCe-Controller, der mit einem programmierten Mikrocontroller des Typs AT89C2051 realisiert ist. Wie auf dem Experimentierboard führen serielle Datenleitungen über einen Pufferbaustein zu einer RS232-Schnittstelle. Diese serielle OnCe-Port-Schnittstelle, kompatibel zu allen Motorola-Entwicklungswerkzeugen, befindet sich auf der Frontplatte des Gerätes.
Der Gedanke war: Soll ein neues Programm entwickelt oder im aktuellen ein Fehler beseitigt werden, so geschieht dies am besten „in situ" – sprich: Mit allen Kabelverbindungen, so daß man das Gerät nicht aus dem Regal nehmen und rückseitig stöpseln muß. Einfach per Debug-Kabel unmittelbar eine Buchse auf der Frontseite kontaktieren, und los geht die Suche, so sollte es sein.
Wie auf dem EVM-Board ist auch hier eine LED vorhanden. Sie zeigt an, daß die CPU sich im Debug-Zustand befindet. Zusätzlich gibt es eine „Once -Port-Operating-LED", die serielle Aktivität auf dem OnCe-Port sichtbar macht. Etwa, wenn ein Programm unter Kontrolle des Debuggers – aber ansonsten in Echtzeit – abläuft.
Nebenbei: Auch eine Kontroll-LED für die CPU-interne PLL ist vorhanden, nur ist die Anzeige gegenüber dem EVM invertiert: Eine leuchtende LED deutet auf eine ausgerastete PLL hin. Dies schafft eine einheitliche Logik der LEDs. LED aus: Alles normal. LED an: Zustand außergewöhnlich.
Der OnCe-Controller übernimmt noch andere Aufgaben: Er überwacht die Mode-Leitung und erzeugt bei Wechseln das Reset-Signal für die CPU, damit sie im neuen Modus bootet. Und weil man heute wirklich viel Gehirnschmalz in so einen



Nach all den Schwierigkeiten nun doch realisiert: Easy DSP

Controller stecken kann, berücksichtigt er, ob man sich in einer Debug-Sitzung befindet oder nicht. Bei Debug-Sitzungen wird kein Reset erzeugt, weil das dann nämlich nicht gerade erwünscht wäre.
Darüber hinaus hat der OnCe-Controller eine wichtige Aufgabe im Alef-Modus: Das Alef-Null-Konzept beruht wesentlich darauf, daß man einen Resetgenerator mit integriertem „Wachhund", als Watchdog, benutzt. Es würde zu weit führen, die genaue Begründung hier zu geben, wichtig ist jedoch: Für den Alef-Null-Betrieb ist der Watchdog zwingend notwendig, für den DF6JB-Betrieb nicht, und für den Debug-Modus wäre ein solcher Watchdog geradezu tödlich. Denn wer soll den Watchdog bedienen, wenn man gerade ein Programm im Einzelschrittmodus abarbeitet?
Der OnCe-Controller kennt alle diese Abhängigkeiten und verhält sich ausschließlich im Alef-Null-Modus wie ein Watchdog.

Seriell – aber richtig

Rechts oben im Blockschaltbild befindet sich die „eigentliche" RS232-Schnittstelle, über die EasyDSP mit den PC-Programmen kommuniziert. Die seriellen Datenleitungen sind über einen Pufferbaustein mit dem UART, der in der CPU integriert ist, verbunden.
Die Statusleitungen führen hingegen über den Pufferbaustein zu einem der beiden großen GALs. Dies deshalb: Im Alef-Null-Modus wird die serielle Schnittstelle ausschließlich „seriell" benutzt, d. h. alle Informationen werden als serieller Datenstrom auf den RS232-Datenleitungen transportiert. Es gibt nur eine über Port PC3 realisierte Handshake-Leitung.
Der DF6JB-Modus sollte aber so universell wie nur irgend möglich sein. Es gibt viele PC-Programme, welche mit externer Hardware über eine RS232-Schnittstelle „reden", aber dies nicht seriell: Sie benutzen die vier Statusleitungen CTS, DSR, RI und RLSD, um darüber 4 bit parallel einzulesen.
Weil diese Programme byteorientiert arbeiten, benutzen sie die RTS-Leitung als Multiplex-Signal, um damit über einen im Peripheriegerät untergebrachten Multiplexer zwischen den oberen und den unteren 4 bit eines Bytes hin- und herzuschalten.
Manche Programme treiben es soweit, auch die DTR-Leitung als zweite Multiplexleitung zu mißbrauchen und damit die RS232-Schnittstelle als 4 × 4-bit-Parallelport zu nutzen.
Als ob dies nicht kompliziert genug wäre, kann es sein, daß ein PC-Programm durch Anlegen einer bestimmten RTS/DTR-Kombination dem Peripheriegerät etwas mitteilen möchte, das zu einer Reaktion im angesprochenen Gerät führen soll.

Reset via serielle Schnittstelle

Es ergibt sich noch folgendes Problem: Wenn jede mögliche DTR/RTS-Kombination „erlaubt" sein soll, wie schafft man es, über die serielle Schnittstelle einen Reset der CPU zu bewerkstelligen? Er ist zwingend, wenn ein neues Programm vom PC in das EasyDSP geladen werden soll.

Diese Funktionalität wird durch das I/O-Gal Mach210 oberhalb der CPU bereitgestellt. Aus Sicht dieser stellt es sich wie ein 18 bit breiter „Memory-Mapped" I/O-Baustein dar, welcher auf der Speicherzelle $FFC0 decodiert wird. Die unteren 16 dieser 18 bit sind genau die, welche über den Multiplexmechanismus vom PC gelesen werden können. Die oberen beiden Bit werden im Gal als eine Art Zusatzadressierung ausgewertet, nämlich dafür, ob die unteren 16 bit für den Multiplexermechanismus gedacht sind oder für etwas anderes.

Über das I/O-Gal kann die CPU ein serielles EEPROM vom Typ 93C46 lesen und beschreiben. Sie kann auch feststellen, welcher der vier Radio-Ports gerade der aktuelle ist und einen anderen einschalten.

Für den Reset der CPU muß ein Zustand der Schnittstelle gefunden werden, der beim normalen Betrieb nicht auftreten kann. Die Alef-Null-Gruppe hat es sich einfach gemacht: Da alle seriellen Kommandos zunächst von ihrem Monitorprogramm interpretiert werden, gibt es schlicht einen Befehl, der die CPU in den Schlafmodus versetzt. Der Watchdog wird nicht mehr bedient, und deshalb sorgt der Resetgenerator dann für ein Rücksetzen.

Dazu ist aber das „Mitspielen" der CPU notwendig, und der Reset soll ja ausgerechnet eine „aus den Fugen geratene" CPU wieder in geordnete Bahnen lenken. Also muß eine Elektronik her, welche auch dann einen Reset erzeugen kann, wenn die CPU „unpäßlich" ist. Um einen Reset der CPU zu erzeugen, muß der PC zunächst DTR und RTS in einen Grundzustand versetzen, danach die serielle Datenleitung in den „Break"-Zustand. Während dieser anhält, soll mindestens achtmal der Zustand der DTR-Leitung wechseln, überwacht durch den I/O-Gal.

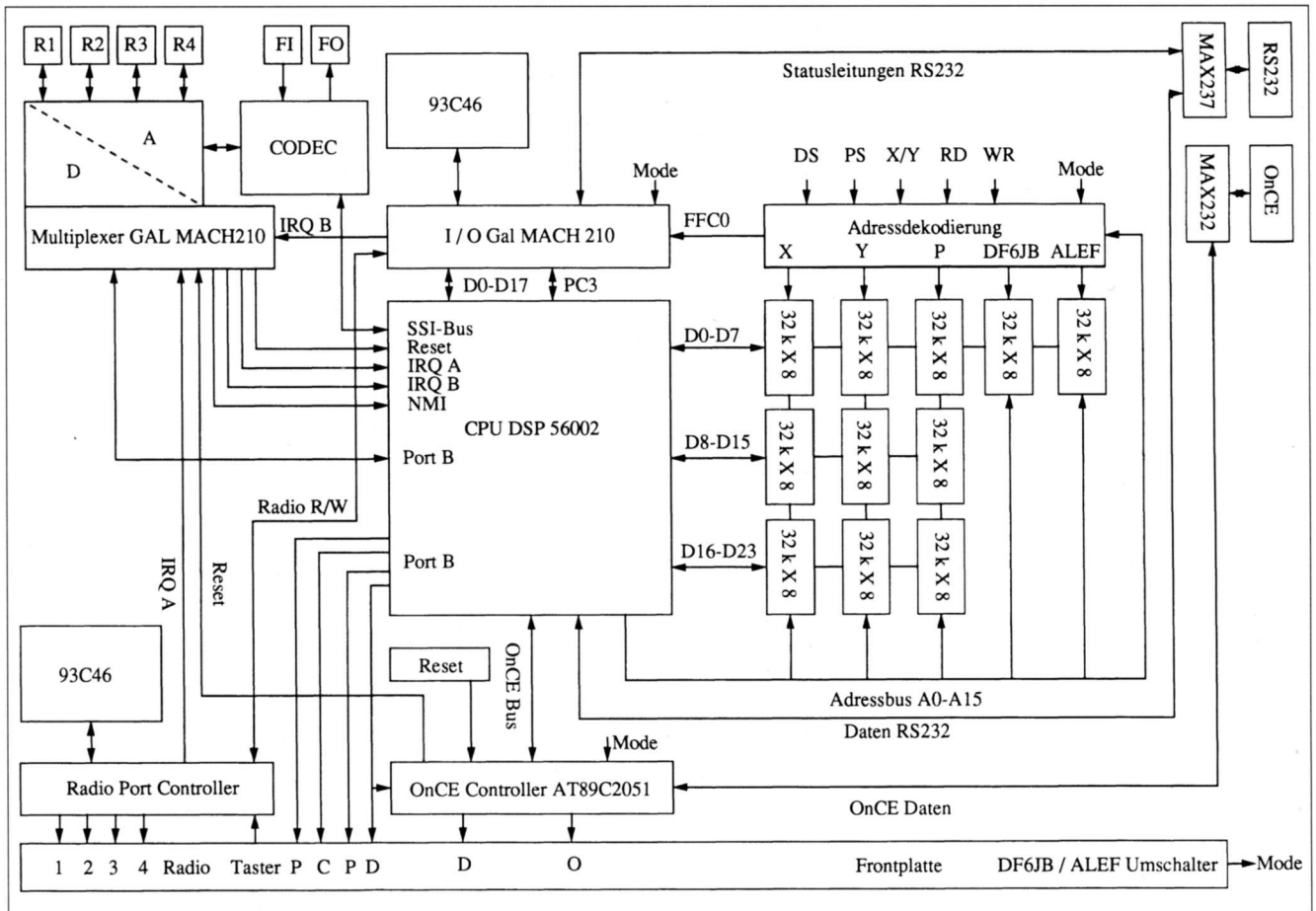
Schlauer Radio-Port

Da nun das Wort „Radio-Port" gefallen ist, muß als nächstes etwas dazu gesagt werden. EasyDSP hat vier Radio-Ports, also Buchsen zum Anschluß von Transceivern. Jeder dieser Radio-Ports ist eine achtpolige DIN-Buchse auf der Rückseite des Gerätes und stellt die Signale NF-Eingang, NF-Ausgang, Masse, PTT, Up, Down, CAT-In und CAT-Out zur Verfügung.

Die DIN-Buchsen sind so ausgelegt, daß ein fünfpoliger Stecker eingesteckt werden kann und die gewohnte Belegung vorfindet. Wer TNC-Kabel hat, kann sie unverändert benutzen. Die Pins 5–8 sind mit Signalen belegt, die nicht selbstverständlich auf TNCs zu finden sind. Es sind Up- und Down-Signale, die zur Frequenzeinstellung von Transceivern benutzt werden können.

Mit den Anschlüssen CAT-In und Out ist schließlich das Steuern von Transceivern mit echter serieller Schnittstelle (mit TTL-Pegeln) möglich. Alle ausgehenden digitalen Signale sind über ULN2803-Bausteine gepuffert. Alle eingehenden digitalen Signale sind bis zu Spannungen von einigen hundert Volt vor Zerstörung gesichert.

Der Radio-Port-Controller legt fest, welcher der Radio-Ports gewählt wird. Er ist links unten im Blockschaltbild zu sehen. Es ist ebenfalls ein kleiner Prozessor vom Typ AT89C2051. Er beobachtet, ob der Benutzer den Taster auf der Frontplatte nach links oder rechts bewegt und schaltet entsprechend – die passende LED dokumentiert dies. Die Information über den aktuellen Port signalisiert er über zwei TTL-Leitungen dem I/O-Gal und damit der CPU.



Das Blockschaltbild zeigt eher als das achtseitige Schaltbild, wie bei EasyDSP die Funktionseinheiten zusammenhängen

Den zuletzt aktiven Radio-Port „merkt" sich der Controller in einem seriellen EEPROM (93C46), so daß nach einem Aus- und Einschalten der zuletzt gewählte Transceiver wieder der aktive ist.
Auch – und das ist eine ganz wichtige Eigenschaft – erzeugt der Radio-Port-Controller ein Interruptsignal für die CPU, wenn der Radio-Port bzw. der Transceiver wechselt.
Daraus ergibt sich folgende interessante Eigenschaft des EasyDSP: Angenommen, der Benutzer arbeitet gerade in Packet Radio mit Transceiver A auf Radio-Port 1. Nun beschließt er, mit Transceiver B Packet Radio zu machen und schaltet auf dessen Radio-Port. Aufgrund des vom Radio-Port-Controller erzeugten Interrupts liest die CPU aus, welcher der gewählte Radio-Port ist, holt sich über das I/O-Gal aus dem 93C46 die zu diesem Port passenden Einstellungen der NF-Amplituden und stellt diese Werte im AD/DA-Wandler ein: Wir sind mit den optimalen Einstellungen für den neuen Transceiver QRV.
Die optimalen Einstellungen für die vier möglichen Transceiver wird mit einem unter Windows laufenden Setup-Programm in einem nichtflüchtigen Speicher (93C46) abgelegt.
Das eigentliche Multiplexen – gemischt analog/digital – zwischen den Ports übernimmt Gal Mach210, der sich um die digitalen Ein- und Ausgänge unmittelbar kümmert und für die analogen Leitungen die vier zweipolig umschaltenden Relais ansteuert.

Puffern bringt Sicherheit

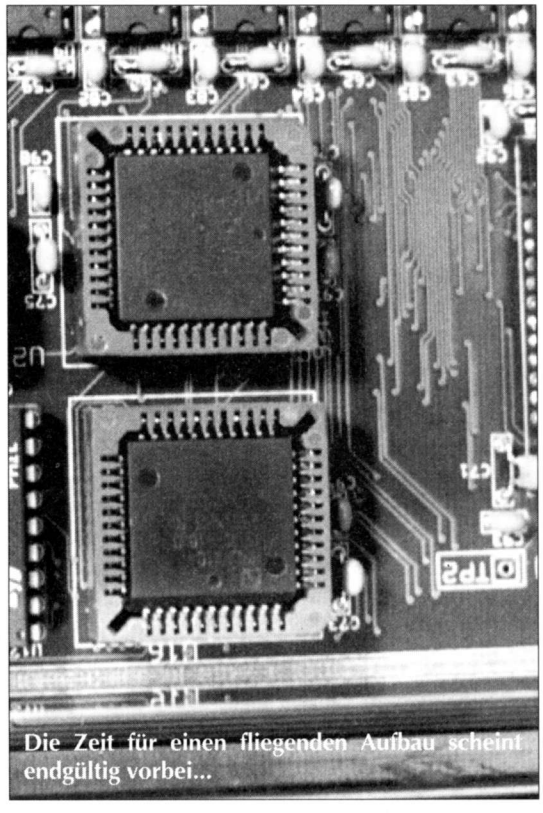
Im Blockschaltbild nicht gezeichnet, aber trotzdem wichtig: Die analogen Ein- und Ausgänge vom und zum AD/DA-Wandler sind durch extrem rauscharme Doppeloperationsverstärker LT1013 gepuffert. Die Eingangsimpedanz beträgt 220 kΩ, die Ausgangsimpedanz wenige Ohm. Etwas mehr als 6 V_{ss} Ausgangsspannung sind möglich, sie kann im AD/DA-Wandler in 1,5-dB-Schritten um 93 dB abgeschwächt werden.
Die nominale Eingangsspannung des AD/DA-Wandlers beträgt 1 V_{eff}. Ein programmierbarer Verstärker im AD/DA-Wandler kann die Empfindlichkeit, ebenfalls in 1,5-dB-Schritten, um mehr als 20 dB erhöhen. Eine Ladungspumpe mit LT1026 versorgt die Operationsverstärker symmetrisch.
Der Radio-Port-Multiplexer ist so ausgelegt, daß die über ihn geschalteten analogen Signale stets mit dem linken Kanal (des in Stereoausführung vorliegenden AD/DA-Wandlers) verbunden werden. Ein- und Ausgang des rechten Kanals an zwei Cinch-Buchsen auf der Rückseite dienen als universelle Ein- und Ausgänge, im Blockschaltbild als FI und FO bezeichnet.

Abschirmung vermeidet Probleme

Alle nach außen führenden elektrischen Verbindungen sind sorgfältig über LCL-Filter von Murata abgeblockt. Ebenfalls sind die „schnellen" Teile der Schaltung innerhalb des Gehäuses mit Weißblech gegen die Außenwelt abgeschirmt. So hält EasyDSP nicht nur die CE-Normen ein, und man hört keine „digitale Musik" mehr, wenn der Empfänger eingeschaltet wird. Dies ist eine Anwenderforderung, die noch schärfer ist als die CE-Norm. Für die Diskussionen um das dafür notwendige Know-how möchte ich bei TNC-Entwickler Ulf Kumm, DK9SJ, danken.
Die ganze Schaltung paßt bis auf zwei Spannungsregler, welche mit dem Gehäuseboden verschraubt sind, auf eine 178 × 233 mm große Platine. Da diese ein Vierfach-Multilayer in Feinstleitertechnik ist – mit Leiterbahnen und Durchkontaktierungen teilweise im 0,2-mm-Bereich! – entzieht sie sich der Fertigung selbst in einer noch so fortschrittlich ausgestatteten Amateurfunk-Waschküche. Deshalb ist der Abdruck des Layouts sinnlos. Bei Bedarf können Sie wegen der Platinen beim Autor anfragen.
Auch wenn man es anders machen wollte: Bei Herstellerhinweisen wie „...and connect this pin via zero impedance to a solid groundplane..." kommt man um ein Multilayer gar nicht herum.
Was wäre eine solche Schaltung ohne Metallgehäuse? Sie mögen mich steinigen, aber Amateurfunkzubehör gehört einfach nicht in Plastikschachteln, erst recht nicht, wenn darin digitale Elektronik mit 40 MHz „abbrummt".
Ein passendes – und schönes! – Gehäuse konnte ich trotz unzähliger Hersteller in Deutschland nicht finden, und so ist die Platine einem US-amerikanischen Lansing-Gehäuse auf den Leib geschrieben.

Auf dem Prüfstand

Das wichtigste Kriterium einer auf DSP-Technologie beruhenden Schaltung ist die Qualität der AD/DA-Wandlung. Hier werden Codecs (ein anderer Name für AD/DA-Wandler) der Firma Crystal Semiconductor benutzt. Es handelt sich um 16-bit-Wandler. 1 bit wird für die Darstellung des Vorzeichens der gemessenen Spannung benutzt, so daß effektiv 15 bit für die gemessene Spannung zur Verfügung stehen.
Da jedes Bit einer Dynamikauflösung von 6 dB entspricht, ergibt dies einen theoretisch nutzbaren Dynamikbereich von 15 mal 6 dB = 90 dB. Dies gilt allerdings nur für einen „idealen" AD-Wandler. In der Praxis zeigt wegen einiger Fehlerquellen, die hier nicht besprochen werden sollen,


Die Zeit für einen fliegenden Aufbau scheint endgültig vorbei...

der AD-Wandler bei kurzgeschlossenem Eingang nicht einfach „0" an. Einige Bits „flattern" immer ein wenig.
Der Hersteller schlägt als Test vor, die analogen Eingänge des Codecs unmittelbar am Chip gegen analoge Masse kurzzuschließen und in diesem Zustand einige Tausend Samples mit der höchsten erreichbaren Samplingrate zu sammeln. Dann wird aus diesen Samples der Effektivwert dieses „Pseudo-Rauschens" ermittelt und dieser Wert als Untergrenze für die Dynamikberechnung zugrundegelegt. Wenn man auf einen effektiven Dynamikbereich von 84 dB oder besser kommt, so ist das Layout optimal gelungen, zumindest, was die Energieversorgung des Codecs und die Kommunikationsleitungen zur CPU angeht.
Die Tests für EasyDSP waren wesentlich strenger. Statt die Eingänge des Codecs unmittelbar am Chip kurzzuschließen, ge-

schah dies erst durch einen Kurzschlußstecker auf der DIN-Buchse. So wird nicht nur der Codec alleine getestet, sondern die nun im Signalweg befindlichen Operationsverstärker, letztlich auch die Leiterbahnen der analogen Signale. Die hätten sich durch Übersprechen aus dem Digitalteil natürlich auch leicht Störungen „einfangen" können.

DSP-Software: Entwicklungsstand Ende Mai

Name	Kurzbeschreibung	Autor
AMFAX95	AM-Demodulator für JVCOM32 (Meteosat / Noaa)	DF6JB
AM14BIT	Wie oben, aber 14 bit Auflösung, aut. Weißabgleich in JVCOM32	DF6JB
FMFAX95	FM-Fax-Demodulator für JVCOM32	DF6JB
SSTV95	SSTV-Demodulator für JVCOM32	DF6JB
CW100-500	Fünf CW-Filter mit Bandbreiten von 100 Hz...500 Hz	DF6JB
CORELFLT	Automatisches CW-Filter nach dem Selbst-Korrelations-Prinzip	SP9VRC
NOTCH	Automatisches Notchfilter	OH2LNS
QRN	Automatisches Noisefilter	OH2LNS
FSK	1200 bit/s AFSK TNC	OH2LNS/DF6JB
FSK300	300 bit/s AFSK TNC	OH2LNS/DF6JB
G3RUH	9600 bit/s FSK TNC	OH2LNS/DF6JB
SIMPLE	Emulation eines Simpel-/Hamcomm-Konverters	DF6JB
ZORNSL	Wie oben, aber mit Vorfilter um 800 Hz zentriert	DF6JB
DIGISAT	Emulation des „FAXELLITE" Konverters f. d. Programm DIGISAT	DF6JB
MSCAN	Emulation des „MULTISCAN" Konverters für MSCAN v. PA3GPY	DF6JB
CODE3	Emulation des „LF3" Konverters für Code III	DF6JB
SENDPSK	AMSAT Phase-3D Telemetrie Simulator	DF6JB
BPSK400	Phase-3D Telemetrie Demodulator m. 1200 bit/s RS-232 Ausgang	DF6JB

Der nutzbare Dynamikbereich betrug fast 86 dB. Das deutet nicht nur darauf hin, daß der Codec etwas besser als vom Hersteller garantiert ist. Auch das Layout von EasyDSP hat womöglich das technisch machbare Limit erreicht. Ein so gutes Ergebnis hatte ich eigentlich nicht erwartet. Wer je Schaltungen entworfen hat, in denen Pegel über einen Bereich von mehr als 80 dB gemessen werden sollen, wird wissen, was ich meine.

Anwendungen

Nach dem Einschalten ist EasyDSP in beiden Betriebsmodi ein schrecklich dummes Gerät, das eigentlich gar nichts kann. Erst die Software im RAM von EasyDSP bestimmt den Charakter der Blackbox.

Schöne neue Welt: Zum einen ist es universell, weil man für jeden nur denkbaren Zweck nichts mehr an der Hardware ändern muß, sondern nur die geeignete Software lädt.

Ferner kommt der experimentelle Charakter des Konzepts durch diese Variationsfreundlichkeit zum Ausdruck. EasyDSP ist für denjenigen geeignet, der in seinem Shack mit einer universellen Hardware in kurzem zeitlichen Wechsel ganz unterschiedliche Betriebsarten auf wechselnden Transceivern „fahren" will.

Wer einen Knoten-TNC sucht, der über Monate und Jahre ein und dasselbe macht, ist damit weniger gut bedient.

Und schließlich: Etwas, das durch neue Software zu einem neuen Gerät wird, kann nie in dem gleichen Sinne „fertig" sein, wie ein Gerät, welches von vornherein für eine bestimmte Anwendung – und nur diese – gebaut wurde.

Gegenwärtig kann man mit EasyDSP folgendes machen: Alle DSP-Card4-Software läuft ohne Änderung sofort. Dies sind Modems für Packet Radio in 1200 bit/s und 9600 bit/s. Außerdem sind einige Filter vorhanden, darunter sog. automatische DeNoiser, wie man sie von DSP-Filtern kennt.

Die TNC-Emulationen verhalten sich wie KISS-TNCs, was nur auf den ersten Blick ein Nachteil ist. Zahlreiche Programme für digitale Kommunikation über Satelliten erfordern geradezu einen KISS-TNC, und wer terrestrisch Packet mit einem Hostmode-Programm macht, benutzt den ausgezeichneten KISS-Hostmode-Treiber TFKISS von Harald Huber, OE1HHC, und Gottfried Motowidlo, OE3GMW. Er unterstützt das DAMA-Protokoll. Ich setze die Kombination aus EasyDSP (als G3RUH-TNC) mit TFKISS und SP 7.5 seit einigen Monaten auf dem 9k6-Einstieg DBØCL ein.

Aus eigener Küche stammt ein Modem, bei dem EasyDSP den Konverter „Multiscan" für das SSTV-Programm „Mscan" vollständig emuliert.

Darüber hinaus existiert eine Emulation des „Simpel"-Konverters, so daß alle Software, die damit geht, auch mit dem EasyDSP funktioniert. Wer nun technisch doch keinen Sinn darin sieht, mit DSP-Technologie den Simpel-Konverter nachzuempfinden, der schaue sich die Betriebsergebnisse an!

EasyDSP tut einiges mehr als der Simpel-Konverter. Es ist ein messerscharfes Vorfilter integriert, und die eigentliche Demodulation kann man in DSP-Technik um Klassen besser realisieren. Der PC bekommt ein schon bereinigtes Signal angeboten, und die Ergebnisse sprechen für sich.

Eine Variante des Simpel-Konverters mit einem auf 850 Hz Mittenfrequenz zentrierten Vorfilter habe ich für das interessante „Zorns Lemma" von OM Ulrich Neuber, DL3ZAS, programmiert.

Eberhard, DK8JV, hat schon ein EasyDSP zum Testen und schneidert sein neues Programm DVCom32 gerade darauf zu. Es wird als echte 32-bit-Applikation unter Windows95 und Windows NT 4.0 laufen. Bitte nerven Sie Eberhard noch nicht mit Fragen nach dem Erscheinungsdatum! Er steht noch mitten in der Arbeit, aber es gibt schon imponierende Demos. Der Stand der Arbeiten ist derzeit so: SSTV und Fax (AM/FM) sind vollständig implemetiert. Zum Erscheinen des Heftes wahrscheinlich fertig: ein Spektrumanalysator 0...16 kHz mit 120 dB Dynamikbereich und Frequenzlupe.

Angesichts des baldigen Starts des Phase-3D-Satelliten programmiere ich gerade einen Demodulator für dessen Telemetrie. Einen Simulator für dieses 400-Bd-BPSK-Signal habe ich für andere Entwickler bereits im Internet veröffentlicht.

Software-Pool wäre prima

Mit der Software geht es also voran. Doch ist allein nur ein begrenztes Pensum zu erledigen, und deshalb möchte ich jeden, der selber Amateurfunkanwendungen für den DSP56002 programmieren möchte, bitten, mit mir Kontakt aufzunehmen. EasyDSP ist sicherlich eine bessere Entwicklungsplattform als jedes Evaluation Board. Die erzeugten Programme sind letztlich PC-Datenfiles (Textfiles in einem bestimmten Format mit der Extension „.LOD"). Sie lassen sich daher problemlos über Mailboxen und über das Internet transportieren. Es könnte einen Softwarepool geben, auf den alle EasyDSP-Anwender einen einfachen und kostenlosen Zugriff haben.

Ulrich Bangert, DF6JB
Ortholzer Weg 1
27243 Groß Ippener
Tel. (0 42 24) 9 50 71
Fax (0 42 24) 9 50 72
E-Mail: DF6JB@compuserve.com

Literatur

[1] H.-J. Pietsch: Amateur-Funkfernschreibtechnik RTTY, Franzis-Verlag, RPB-Band 25
[2] forrerj@peak.org
[3] Jarkko.Vuori@hut.fi
[4] Johan Forrer: Using the Motorola DSP56002EVM for Amateur Radio DSP Projects, QEX August 1995
[5] kwi@cc.hut.fi
[6] DSP-Card4: FTP-Server nic.funet.fi
[7] Motorola: DSP56000 Digital Signal Processor Family Manual, Kapitel 10
[8] Klaus Raban: Slow Scan Television von Simpel bis Hightech, Buch und CD-Rom, erscheint im Juni 1997