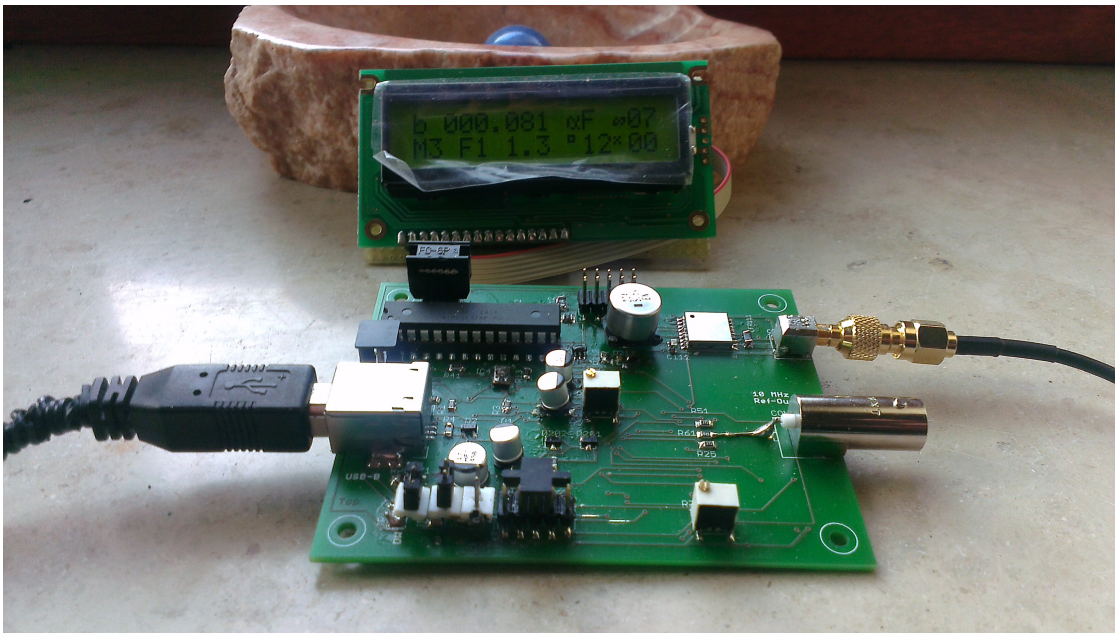
DF 4 IAH - 10 MHz Referenz Oszillator - V2

DF 4 IAH, Ulrich Habel

Ladenburg, im Jahre 2015

Von der Schaltungsidee zur funktionstüchtigen Leiterplatte mit bestückten SMD-Bauteilen

Mehr und mehr gehen die Hersteller von Halbleiterbauteilen dazu über, ihre integrierte Schaltkreise als SMD-Bauteile (Surface Mounted Device = Oberflächenmontage) vorzusehen. Als Amateur ist es leicht möglich, sich an die geänderte Anforderungen der Leiterplattenerstellung anzupassen. Dieses Skript zeigt den Werdegang von der Schaltungsidee zur funktionstüchtigen Platine auf. Die Schaltung, die hier beschrieben wird eignet sich dazu, ein normgerechtes, hochgenaues 10 MHz Referenz-Signal bereit zu stellen, wie sie für eine Vielzahl von Laborgeräten und vorhandene Oszillatorschaltungen genutzt werden kann. Mit Hilfe der GPS-Positions- und Zeitbestimmung kann eine hohe zeitliche Auflösung von besser als ±15 ns erreicht werden.



Inhaltsverzeichnis

1 Frequenznormal – 10 MHz, eine Referenz für viele Gerätetypen

Mit der Ausbreitung von Funkwellen geht einher, da sich diese bei einem mit *Wechselstrom* durchflossenem Leiter ablöst, sich zu fragen bei welcher Wechselgeschwindigkeit – auch als Frequenz bezeichnet – der Vorgang stattfindet. Sie hat im SI-Einheitensystem keine eigene Einheit, sondern ist als Kehrwert der Periodendauer $\frac{1}{s}$ aufzufassen, welcher der Wechselstrom benötigt, um die Nulllinie in steigender Richtung erneut zu durchlaufen. Die Frequenznormale kommen in Frage, wenn die Notwendigkeit vorliegt, eine große Frequenzgenauigkeit zu erreichen. Im Laufe der Zeit hat sich eine Frequenz von 10 MHz etabliert, die diese Funktion erfüllt.

Abbildung 1: Frequenznormal derzeit auf **0.055 ppb** (Parts Per Billion) genau

1.1 Im Bereich der Laborgeräte

Werden Geräte entworfen, so werden diese zuerst in einem Laboratorium entworfen und verbessert. Hier kommen Prüf- und Analyse-Geräte zum Einsatz, die bereits eine hochgenaue Zeitbasis benötigen. Nachfolgend eine kurze Zusammenstellung von typischen Vertretern im Labor, die dieses Frequenznormal mit einbeziehen können:

Pos	Gerätetypen	Funktion des Frequenznormals als
1	Frequenzgeneratoren	definierter Muttertakt für eine PLL oder DDS
2	Frequenzzähler	definierter Muttertakt für definierte Torzeiten
3	Spektrumanalysatoren (Spek)	Referenzfrequenz im Display zur Vermessung
4	Vektorielle Netzwerk Analysatoren (VNA)	Referenzfrequenz für interne Berechnungen

Tabelle 1: Auflistung Laborgeräte mit 10 MHz Eingang

Gerade der Punkt 2 ist für den Funkamateur und Bastler von besonderer Bedeutung, denn Funkgeräte werden häufig an solchen Frequenzzählern kalibriert.

Eine andere Nutzung des Frequenznormals wäre es, das Sendesignal in abgeschwächter Form auf den Empfängereingang zu legen, damit der Empfänger auf diese Sollfrequenz eingestellt werden kann. Dazu muss das Funkgerät jedoch in der Lage sein, diese 10 MHz empfangen zu können. Ist dies der Fall und weiterhin auch der Sender mit dem selben Oszillator verbunden, dann ist damit auch bereits die Sendefrequenz-Kalibrierung erledigt.

1.2 Im Feld – jenseits des Labortisches

Im weiteren Verlaufe einer Produktentwicklung kommt dann die Evaluierung, hier wird das entworfene System letztendlich in seiner zugedachten Umgebung eingesetzt. Da dies weitgestreut im „irgendwo“ sein kann, wird in der gemeinsamen Terminologie von dem Wort „im Feld“ gesprochen. Bekannt dürfte hier das Wort „Feldtest“ sein, was deutlich machen soll, dass hier das Produkt bereits den Labortisch verlassen hat.

Pos	Gerätetypen	Funktion des Frequenznormals als
1	Sendetechnik für Gleichwellenrundfunk	Ausrichtungsmarker für gemeinsame Frequenz- und Phasenlage
2	Funksysteme mit gemeinsamer Taktung	Muttertakt PLL-/DDS-Systemen, Zeitverteilungsscheiben

Tabelle 2: Auflistung der Geräte im Feld mit 10 MHz Eingang

2 Aktuelle Technologien einzusetzen lohnt sich

Von der Möglichkeit etwas selbst zu erschaffen und zu entwickeln geht eine Faszination aus. Dies trifft für den Funkamateur wie auch für die aktuell bekannt werdende „Maker-Szene“ gleichermaßen zu. Was das Arbeiten mit elektronischen Bauteilen in den letzten dreißig Jahren erschwert hat sind die schrumpfende Gehäuse der Bauteile den eigenen, werkenden Händen gegenüber. Auch sind motorischen Auslenkungen feindosierter zu unternehmen, die anzuwendenden Kräfte geringer, die Sehkraft stärker gefordert. Wenngleich ich nichts davon in Abrede stellen möchte, so will ich dennoch dazu motivieren, sich einmal mit diesen neuartigen SMD-Bausteinen (Surface Mounted Device) und SMT-Technologien (Surface Mounted Technology) näher zu beschäftigen. Denn viele Forderungen an den eigenen Körper und Geist können durch gezielte Maßnahmen erzielt werden:

- die Sehschärfe und Details lassen sich mit kostengünstigen Werkzeugen verbessern
- mit viel Licht lässt es sich erheblich besser arbeiten. Lupen mit Leuchtringen kombinieren diese beiden Punkte für rund 50,- €
- eine angepasste Löttechnik in Gerätschaften und Handhabung sind ein weiterer Punkt zu einem motivierenden Erfolg
- neue Lötvarianten wie Reflow-Löten und Löten mit Heißluft sind neue, interessante Gebiete, die es zu erforschen gibt
- eine ganze Reihe von Bauteilen, ICs und Modulen sind nur noch über ein SMD/SMT-Package erhältlich

Eine Türe wird aufgestoßen, wie bei einem neuen Hobby gibt es erst einmal eine angemessene Grundausrüstung aufzubauen und der Umgang mit dieser zu erlernen. Macht das nicht unsere technische Neugierde gerade aus, sich mit Neuem zu beschäftigen?

2.1 Motivation für kleine Bauteile - SMD und SMT

So winzig diese Bauteile zuerst aussehen, doch durch die Lupe werden sie groß – und in ihrem Inneren sind sie das allemal. Es gibt ganze Prozessoren in kleinen Gehäusen mit minimalem Energieumsatz – undenkbar die Fülle dessen, was am internationalen Bauteilemarkt alles angeboten wird! Da bleibt doch die Frage nicht aus: ist da auch ein Bauteil für mich dabei?

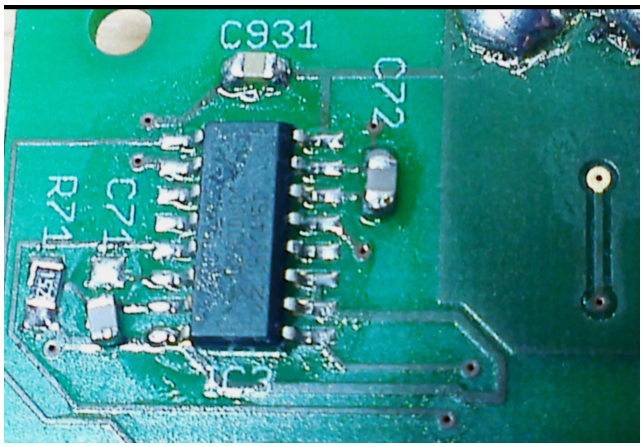


Abbildung 2: SMD Bausteine, hier ein 74HCT221 dual Monoflop

2.2 Rekonfigurierbare Logik: FPGAs und CPLDs auf Erfolgskurs

Mit Sicherheit sind Bauteile dabei, die das Interessensgebiet der Funktechnik und Datentechnik oder die Kommunikation abdecken. Häufig genügen heutzutage Prozessoren, die inzwischen eine komplexe Signalaufbereitungen per Software ermöglichen.

Ab einer bestimmten Frequenz wird es für die herkömmliche Prozessortechnologie jedoch schwierig, die zeitgenauen Impulse und hohen Frequenzen spezifikationsgerecht bereit zu stellen. Genau hier kommt eine andere Technologie zum Einsatz, die auf dem Gebiet der Logikbausteine fußt. Zuerst waren programmierbare Logikbausteine auf dem Markt wie PALs, die zwar einerseits programmierbar waren, aber andererseits direkte Logikgatter zur Verfügung stellten. Diese PAL-Strukturen wurden in den letzten zwanzig Jahren weiterentwickelt zu komplexen PAL-Bausteinen, auch als CPLDs bekannt. Weiterhin wurde der Ansatz noch etwas abgeändert, so werden mit FPGAs nun rekonfigurierbare Logikbausteine angeboten, die bereits häufig genutzte Logikstrukturen (de-)zentral zur Verfügung stellen (Datenregister, Schieberegister, Look-Up-Tables, Additions und Multiplikationseinheiten, Speicher, PLLs) und somit den Pool der reinen Kombinatorik entlasten. Diese FPGA-Technik ist nun soweit gediehen, dass sich in solchen Bausteinen ganze CPU-Kerne programmieren lassen und somit eine gesonderte CPU gar nicht mehr nötig ist.

Der Preis dieser FPGAs liegt zwar höher, als wenn zu einer Problemlösung direkt ein Chip entwickelt würde; das lässt aber manche Hersteller dennoch unerschrocken, sie verbauen FPGAs auch für Konsumer-Anwendungen und haben so die Möglichkeit, zu einem späteren Zeitpunkt eine Schaltungskorrektur durchzuführen. Für den Labor-Tisch dagegen sind sie erste Wahl, denn so können Fehler schnell bereinigt und Schnellstudien zu Solutions erstellt werden. Klassisch würde eine Digitaltechnik-Angelegenheit mit einem FPGA gelöst werden, jedoch beim Übergang in die (große) Serienproduktion auf ein maßgeschneidertes ASIC umgeschwenkt werden.

2.2.1 VHDL und Verilog als schaltungsbeschreibende Sprache

Wie wäre es, wenn nun der Übergang zwischen der FPGA- und ASIC-Technologie fließend gestaltet würde, es nur eine Logikbeschreibungssprache gäbe, die für beide Varianten nutzbar wäre? Es gibt derzeit zwei leistungsfähige Logikbeschreibungssprachen, die sich etabliert haben:

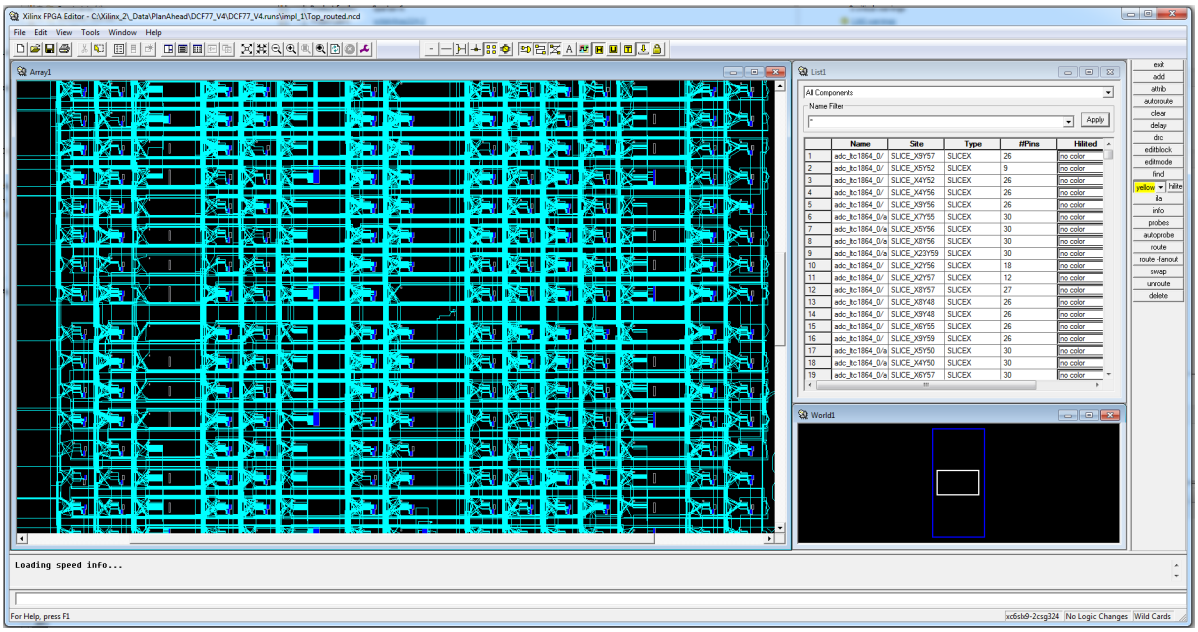


Abbildung 3: FPGA Aufbau: sichtbar die „Signal-Straßen“ und die „Funktions-Blöcke“

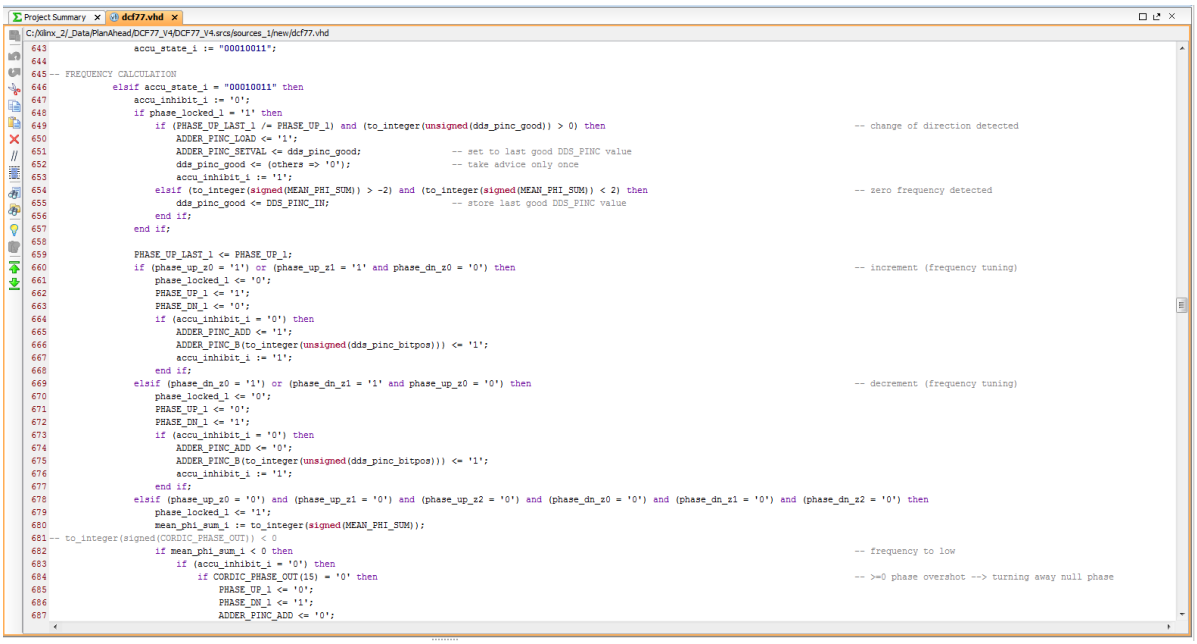
1. VHDL, welche auf der Programmiersprache Small-Talk aufsetzt. Durch die sehr strikt eingeforderte Einhaltung von Schnittstellen war das die erste Wahl für militärische Projekte. In der Elektronikentwicklung bedienen sich weiterhin die Europäer vermehrt dieser Sprache.
2. Verilog, die eine Ähnlichkeit zu der Programmiersprache C hat. Wer einmal die Sprache „C“, „C++“ und andere Dialekte davon erlernt hat, wird schnell von diesem Konzept seinen Nutzen ziehen können. Zum Zeitpunkt der Jahrtausendwende gaben die US-Amerikaner der Sprache Verilog den Vorzug.

Wichtig dabei ist zu wissen, dass die meisten Tools beide Logikbeschreibungssprachen unterstützen.

Der Autor favorisiert VHDL, da er diese Sprache bereits aus Berufsgründen nutzte. Was beim Einsatz einer solchen Entscheidung zu beachten ist, was diese „Investition“ sicherstellt und sie über die Zeit gesehen nicht wertlos wird:

- VHDL (IEEE, 1987) und Verilog (IEEE, 1995) sind als Logikbeschreibungssprache konzernübergreifend nutzbar, durch Normierung abgesichert
- Leistungsfähige Tools für Dateneingabe und zur Simulation sind von verschiedenen Herstellern verfügbar
- Verilog wäre für diese Elektronik-Entwicklung auf gleicher Augenhöhe ebenfalls möglich gewesen

VHDL Programm-Entwicklung und Downloading in das Lattice CPLD **ispMACH4032-V** erfolgt mit den beiden kostenlosen Tools **ispLEVER** und **Diamond Programmer**. Das ispLEVER verarbeitet die VHDL-Struktur und -Verhaltens-Beschreibungen des Projekts, welches daraus die Device-Konfiguration ermittelt. Diese Zwischendatei wird mit dem Programmier-Tool dann auf den nichtflüchtigen Halbleiter gebracht. Die Konfiguration solle über Jahrzehnte erhalten bleiben. Das CPLD kann jedoch bei Bedarf gelöscht und wiederbeschrieben werden. Durch diese Möglichkeit sind noch Schaltungsänderungen möglich, die keine Verdrahtungsänderung



```
643                 accu_state_i := "00010011";
644
645 -- FREQUENCY CALCULATION
646             elsif accu_state_i = "00010011" then
647                 accu_inhibit_i := '0';
648                 if phase_locked_1 = '1' then
649                     if (PHASE_UP_LAST_1 /= PHASE_UP_1) and (to_integer(unsigned(dds_pinc_good)) > 0) then                    -- change of direction detected
650                         ADDER_PINC_LOAD <= '1';
651                         ADDER_PINC_SETVAL <= dds_pinc_good;                 -- set to last good DDS_PINC value
652                         dds_pinc_good <= (others => '0');                   -- take advice only once
653                         accu_inhibit_i := '1';
654                     elsif (to_integer(signed(MEAN_PHI_SUM)) > -2) and (to_integer(signed(MEAN_PHI_SUM)) < 2) then           -- zero frequency detected
655                         dds_pinc_good <= DDS_PINC_IN;                       -- store last good DDS_PINC value
656                     end if;
657                 end if;
658
659                 PHASE_UP_LAST_1 <= PHASE_UP_1;
660                 if (phase_up_z0 = '1') or (phase_up_z1 = '1' and phase_dn_z0 = '0') then                                 -- increment (frequency tuning)
661                     phase_locked_1 <= '0';
662                     PHASE_UP_1 <= '1';
663                     PHASE_DN_1 <= '0';
664                     if (accu_inhibit_i = '0') then
665                         ADDER_PINC_ADD <= '1';
666                         ADDER_PINC_B(to_integer(unsigned(dds_pinc_bitpos))) <= '1';
667                         accu_inhibit_i := '1';
668                     end if;
669                 elsif (phase_dn_z0 = '1') or (phase_dn_z1 = '1' and phase_up_z0 = '0') then                              -- decrement (frequency tuning)
670                     phase_locked_1 <= '0';
671                     PHASE_UP_1 <= '0';
672                     PHASE_DN_1 <= '1';
673                     if (accu_inhibit_i = '0') then
674                         ADDER_PINC_ADD <= '0';
675                         ADDER_PINC_B(to_integer(unsigned(dds_pinc_bitpos))) <= '1';
676                         accu_inhibit_i := '1';
677                     end if;
678                 elsif (phase_up_z0 = '0') and (phase_up_z1 = '0') and (phase_up_z2 = '0') and (phase_dn_z0 = '0') and (phase_dn_z1 = '0') and (phase_dn_z2 = '0') then
679                     phase_locked_1 <= '1';
680                     mean_phi_sum_i := to_integer(signed(MEAN_PHI_SUM));
681 -- to_integer(signed(CORDIC_PHASE_OUT)) < 0
682                     if mean_phi_sum_i < 0 then                                                                           -- frequency to low
683                         if (accu_inhibit_i = '0') then
684                             if CORDIC_PHASE_OUT(15) = '0' then                                                           -- >=0 phase overshot --> turning away null phase
685                                 PHASE_UP_1 <= '0';
686                                 PHASE_DN_1 <= '1';
687                                 ADDER_PINC_ADD <= '0';
```

Abbildung 4: VHDL-Code Ausschnitt von einem komplexen Zustandsautomaten

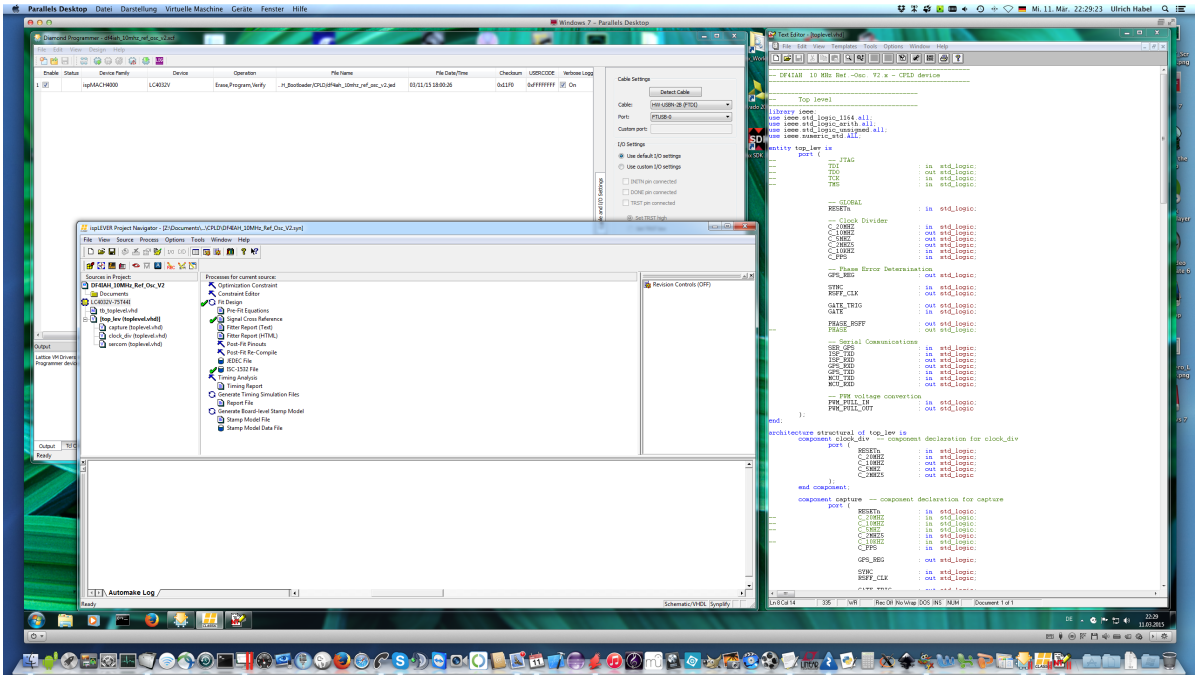


Abbildung 5: VHDL-Entwicklung bei Lattice: **ispLEVER** und **Diamond Programmer**

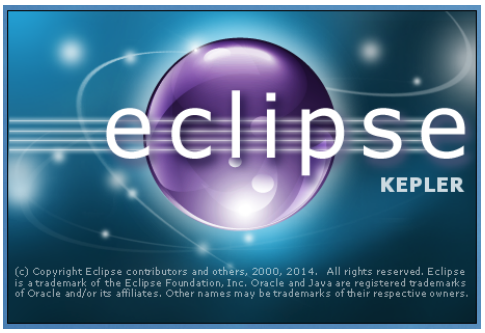


Abbildung 6: Eclipse begrüßt so eine neue Session

nötig machen. Nebenbei werden eine Vielzahl von Glue-Logik-Gatter in das CPLD integriert. Lediglich ein Monoflop musste als ein externes Device angekoppelt werden, welches durch dessen niedrige Grenzfrequenz die Phasenvergleichschaltung auf 2,5 MHz limitiert. Die dazu nötige Frequenzteilung von 10 MHz auf 2,5 MHz übernimmt das CPLD ebenfalls.

2.3 Programmcode mittels Eclipse und dem Plugin *AVR eclipse* erstellen

Als ein leistungsfähiger, aber dennoch günstigen Prozessor kommt der **Atmel ATmega 328P** zum Einsatz. Der Programmcode könnte zwar mit Hilfe von Atmel Entwicklungs-Tools erstellt werden, aber wer sich in Eclipse einmal eingearbeitet hat, wünscht sich diese IDE für viele Projekte einzusetzen. Speziell für die genannte MCU gibt es ein brauchbares Plugin, welches eine Programmierung in „C“ erlaubt und zugleich auch ermöglicht, den assemblierten Code auf den AVR-Controller zu befördern. Hierzu wird auf die bereits bestehenden Programm-Pakete **avr-gcc** und **avrdude** zurückgegriffen.

Ein paar wissenswerte Informationen über Eclipse:

- Eclipse ist eine Open-Source IDE zur Software-Erstellung
- basiert auf Java und war ursprünglich für Java-Entwicklung gebaut worden
- durch eine Vielzahl von Plugins sehr leistungsfähig geworden:
  - C und C++ Entwicklung möglich,
  - AVR-Eclipse Plugin für die ATmega-Baureihe der Atmel Controller
  - Versionierungssysteme wie Subversion (SVN) und GIT können aktiviert oder zugeladen werden
  - Plattformübergreifende Entwicklung auf MAC, Linux und Windows-PCs möglich

Eclipse kann auch Workflow- und UML-Grafiken, Modell-Übersichten und vieles Weitere erstellen – eine Vielzahl von vorhandenen Plugins erweitern Eclipse reichlich und sinnreich weiter. Das Tool ist kostenlos zu beziehen, die Paket-Sourcen jedoch sehr umfangreich, was allerdings generell beim Download von Entwicklungs-Software zu beachten ist. Für diesen Download sollte daher genügend Zeit und Datenvolumen eingeräumt werden.

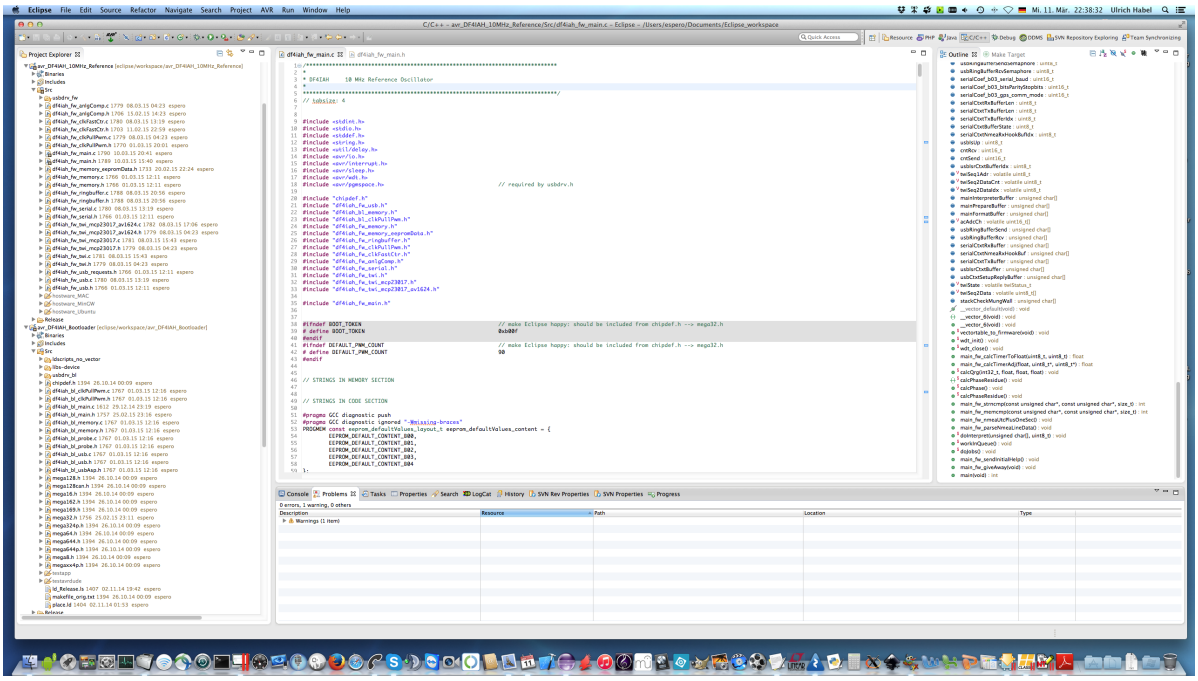


Abbildung 7: Der Arbeitsplatz bei Eclipse kann in weiten Bereichen konfiguriert werden

2.4 Mit Eagle zur funktionierenden, doppelseitigen Platine

- Eagle ist seit Jahren namhafter Hersteller für Leiterplattenentflechtungs-Software
- Autorouter ist inzwischen durchaus brauchbar, wenngleich „Experten“ auch heute noch von Hand entflechten
- Intuitive Bedienung und leicht einprägsamer Ablauf der Schaltungserstellung
- problemlose Akzeptanz bei den gängigsten Leiterplatten-Herstellern
- mit Hilfe von DRC-Rules, die von den Leiterplatten-Herstellern bezogen werden können, entstehen regelkonforme Layouts, die – ohne Überraschungen zu erhalten – gefertigt werden können

3 Von der Schaltungsidee zur funktionstüchtigen Leiterplatte

Den Traum, eine eigene Leiterplatte zu fertigen habe ich zwar schon mehrfach erfüllt. Aber bisher beschränkte ich mich auf kleine Leiterplatten, die ich auch selbst fabrizierte. Entsprechend wenig komplex waren die dabei auf der Platine genutzten Bauteile. Aber mit einem Auge auf die kommerzielle Fertigung schielend, fing ich an, eine private Entwicklungsumgebung für aktuelle und moderne Schaltungs-Designs aufzubauen.

3.1 Entwicklung am heimischen Mac / Linux / Windows-PC

Sollte es möglich sein, auf dem heimischen Rechner eine Schaltung zu entwerfen, die einerseits kommerziell gefertigt werden kann, aber auf der anderen Seite die Hobby-Kasse nicht übermäßig strapaziert?

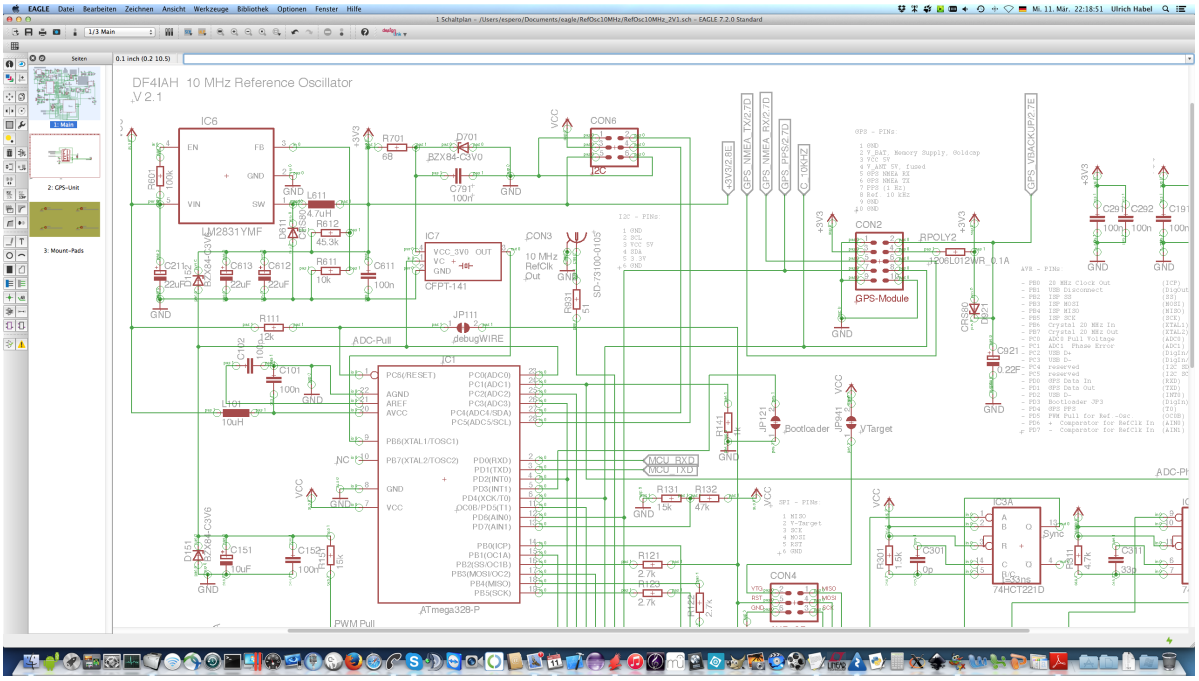


Abbildung 8: Beispiel für eine Schaltplaneingabe, sie ist leicht zu bewerkstelligen

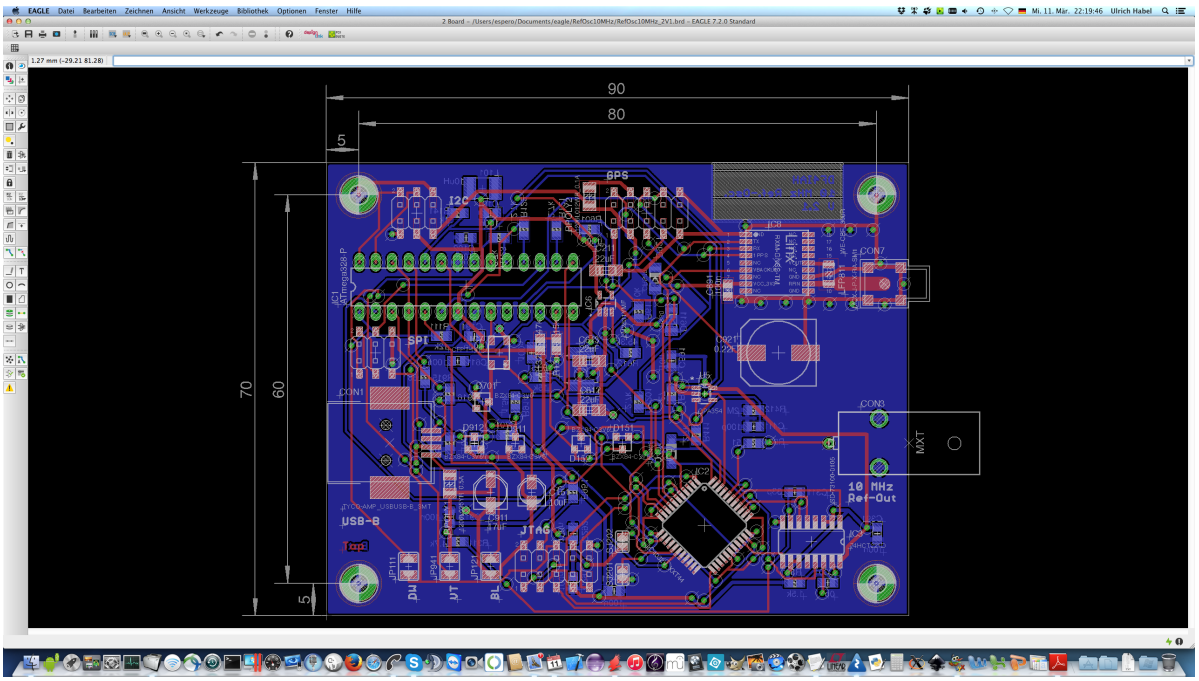


Abbildung 9: Vom Schaltplan zum „Board“ entwickelt, welches letztendlich fabriziert wird

3.1.1 Erhältliche Werkzeuge zu erschwinglichen Preisen

Mit dem Stift durchrechnen: was habe ich bereits vorhanden? Was brauche ich zusätzlich noch? Welchen Betrag lasse ich zu, in mein Hobby zu investieren? Der letzte Punkt ist nicht trivial, denn es handelt sich bei solchen Kosten nicht einfach um Gerätekosten, sondern um die Einrichtung einer Fortbildungszentrale im eigenen Shack. Darf Bildung etwas kosten? Ich denke schon. Rechnen wir mal durch ...

Pos	Entwicklungswerkzeug	Kosten
1	MAC / Linux / Windows-PC als Entwicklungs-Rechner	vorhanden (?)
2	Cad-Soft: EAGLE HOBBYIST	166,60 €
3	Lattice: ispLEVER (VHDL-Eingabe) & Lattice: Diamond Programmer	kostenlos
4	Lattice: JTAG USB Download-Cable	153,96 €
5	AVR-Software-Entwicklung: Eclipse und AVR-Plugin	kostenlos
6	AVR-Programmcode Download auf den ATmega: USBasp	unter 15 €
7	alternativ zu Pos 5: AVR Dragon, (Distributor: Avnet)	rund 62 US-$

Tabelle 3: Kostenaufstellung der Entwicklungswerkzeuge

3.1.2 Schaltungsidee: AVR-Controller regelt eigene Taktfrequenz

Weshalb einen solchen Prozessor einsetzen? Sicherlich gibt es auch eine Vielzahl weiterer Produkte, auch von anderen Herstellern, aber diese Serie fiel mir bereits häufiger auf: sei es ein **VNWA3**[1] oder ein **HF-Messzwerg**[2], die jeweils für sich ein erfolgreiches Produkt geworden sind.

- Atmel ATmega 328P (AVR) als MCU
  - preiswert: schon ab ca. 3,50 € als Einzelbauteil erhältlich
  - ist ein 20 MHz RISC-Prozessor mit reichhaltigen I/O-Komponenten
  - als 28 poliger DIL-Baustein leicht für Nachbauprojekte nutzbar
  - enthält eine I^2C-Bus kompatible Schnittstelle („TWI“)
- Eigenentwickelte Taktregelung soll weiteren Taktgenerator für die MCU einsparen
  - Vorteil: Kostenersparnis
  - Prozessor läuft somit ebenfalls GPS-synchronisiert
  - Das Handmuster V1 ergab, dass mit Software ein breiter Bereich von Regelungseinstellungen zur Verfügung stehen, die auch eine stark variable Taktquelle zu synchronisieren vermag
    - Beispiel: 20 MHz Quarz ohne Temperaturstabilisierung und einer Ziehspannung an seinen Varicap-Dioden konnte somit, mit einfachen und kostengünstigen Mitteln, in der Frequenz eingestellt werden
    - Wunsch: Störquelle „Temperaturschwankung“ soll entfernt werden. Zwei naheliegende Lösungsmöglichkeiten:

[1] Vector Network Analyzer von **DG8SAQ**, Link: http://sdr-kits.net/VNWA3_Description.html
[2] Nachbauprojekt in SMD-Technologie für ein HF-Messgerät von **DH8DAP** und **DB1BM** von funken-lernen.de

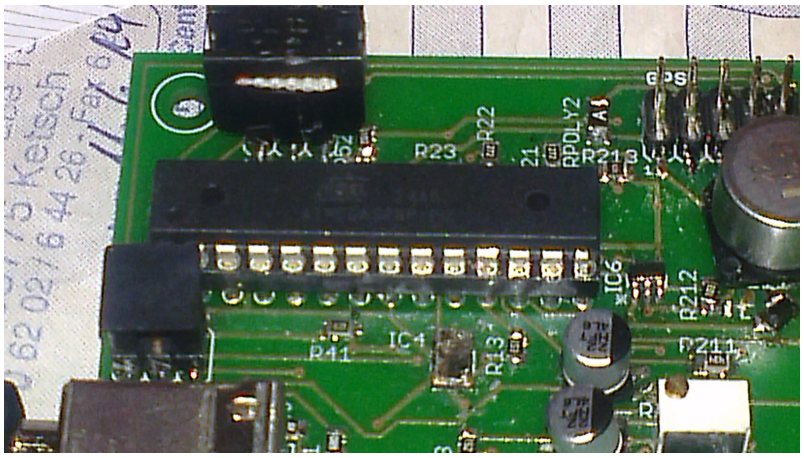


Abbildung 10: Atmel ATmega 328P im DIL-Gehäuse

  - Verwendung eines Quarzofens - benötigt reichlich elektrische Energie, hat eine lange Einschwingphase, setzt Wärmeenergie frei und nimmt mehr Platz in Anspruch als ein einzelner Oszillator
  - Verwendung eines (VC)TCXOs - bisher lagen dem Autor wenig Erfahrungen diesbezüglich vor, wie das Funktionsverhalten und Regelverhalten damit ausfällt

- Als Referenz-Signal soll ein **1 PPS (Pulse Per Second)** Signal eines GPS-Empfängers dienen, gegen welche der lokale 20 MHz Oszillator synchronisiert wird

Der eigentliche Anstoß für das Bauprojekt war ursprünglich der Erhalt zweier GPS-Module von Rockwell, welche den Zugang zu dem GPS-System herstellen. Aus der Fachpresse wurde schon häufiger berichtet, wie das GPS-System – quasi als Nebenprodukt – neben der 3D-Positionsinformation auch eine Zeitinformation zur Verfügung stellt. Die Genauigkeit des Synchronimpulses liegt in der Größenordnung von 10...30 ns. Im Gegensatz zu dem DCF-77 Funksender in Mainflingen, der eine Zeitauflösung von rund 1,5 µs zulässt, ist das demnach eine Verbesserung von etwa 2 Zehnerpotenzen. Ein solches Empfangs-Modul konnte ich über einen Elektronik-Shop besorgen und lag mit rund 30,- € etwa dort, was ich für ein Hobby-Objekt noch bereit war zu zahlen, um damit Experimente zu starten. Somit wurde der Grundstein für meinen Synchron-Takt geboren.

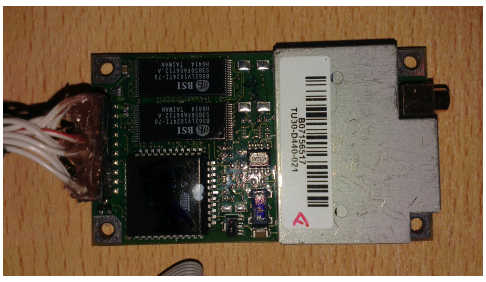


Abbildung 11: Das **Rockwell-Jupiter-GPS TU30-D440** Modul mit MCX-Buchse und doppelreihiger 2 mm-Stiftleiste

Die Ausmaße des GPS-Moduls sind recht üppig (71 mm x 41 mm) gegenüber meiner V1-Entwicklung im Europa-Kartenformat (160 mm x 100 mm). Von der Stromaufnahme aus gesehen einmal ganz zu schweigen.

Aus den obigen Ideen folgend wurde letztendlich eine konkrete Schaltung:

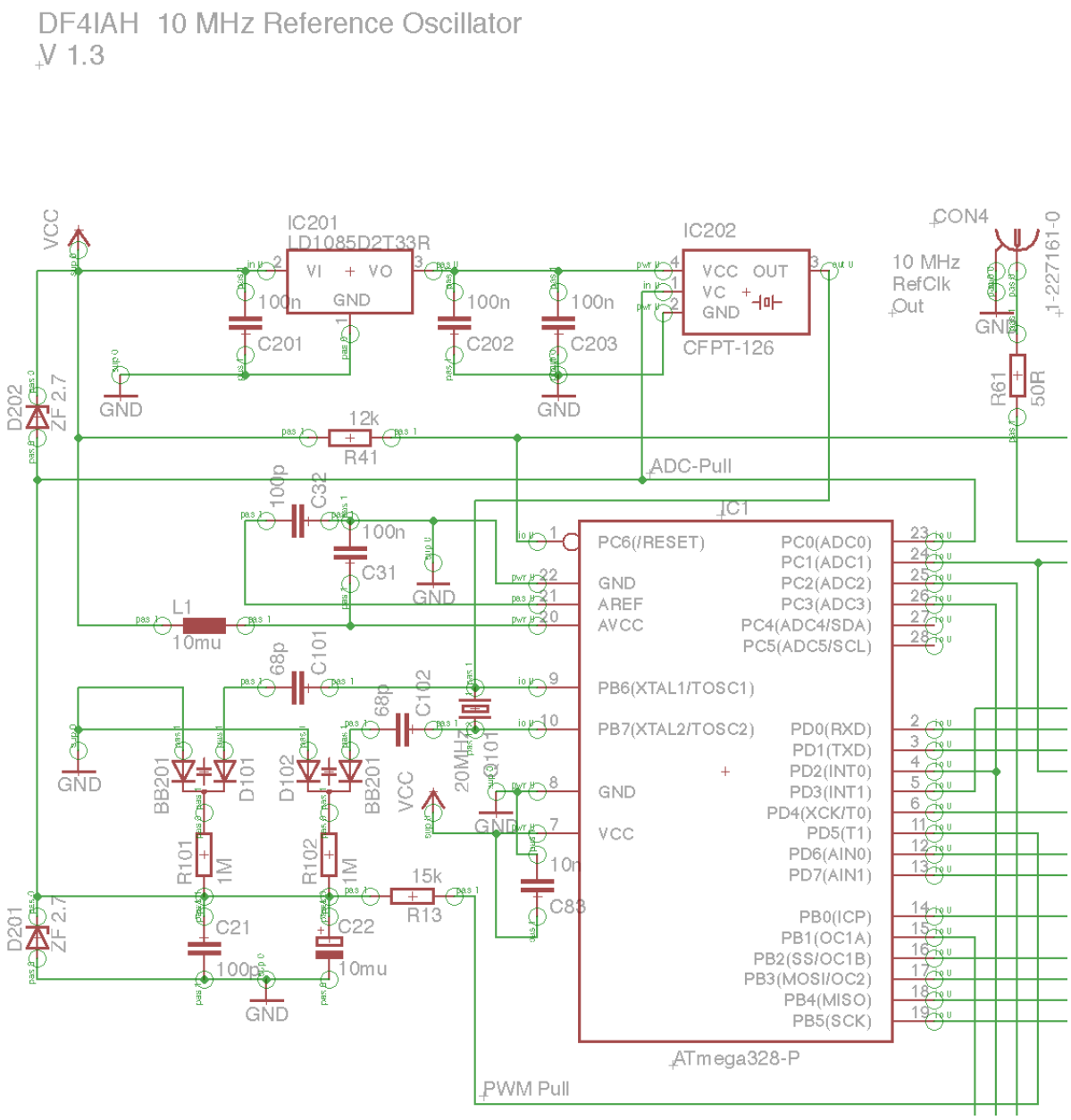


Abbildung 12: 20 MHz Quarz gezogen durch Varicap-Dioden – versorgt die MCU und die externe Schnittstelle

Wie bereits geschildert war keine Temperatur-Regelkomponente vorhanden, es genügte ein Anblasen des Quarzes um dessen Frequenz aus dem Phasenverbund zu lösen und kurzfristig einige Herz abweichen zu lassen. Das war mir im Voraus bei der Konzeption von **V1**[3] klar, aber über welche Fähigkeiten man als Entwickler durch eine Prozessoreinheit gewinnt hatte mich doch sehr positiv überrascht und motivierte dann, auf diesem Gebiet weiter zu entwickeln.

[3]Die obige Schaltung zeigt IC201 und IC202, die bereits als Bestückungsvariante anstatt des Quarzes angedacht waren.

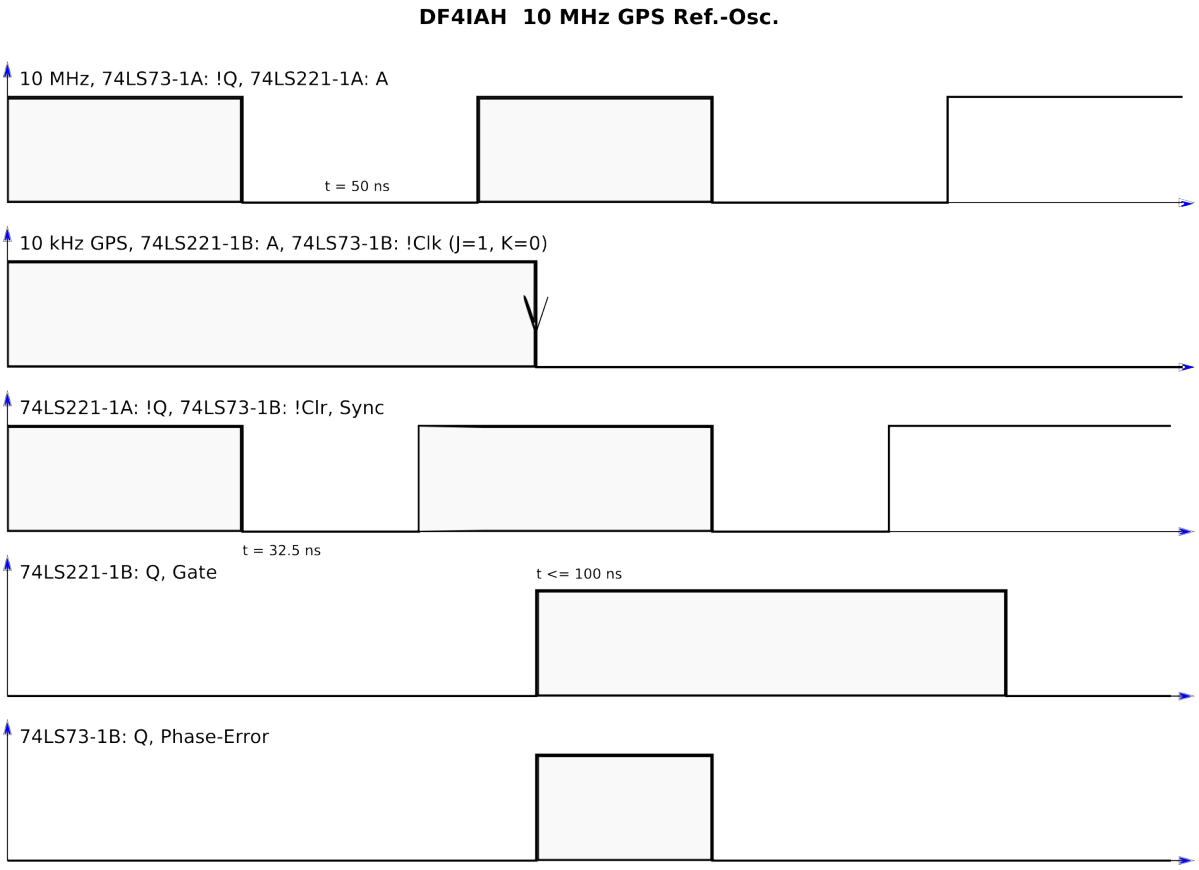


Abbildung 13: Phasen-Diskriminator - Überlegungen über das Timing-Verhalten

3.1.3 Phasen-Diskriminator mit einem JK-RS-FlipFlop realisieren

Eine weitere Idee bestand darin, die zuerst stattfindende Frequenzregelung durch den AVR-Frequenz-Counter zu bewerkstelligen. Eine Frequenzregelung um die 10 MHz Nominal-Frequenz wäre so zwar möglich, aber die Phasenlage zu dem GPS **1 PPS**-Signal bliebe dadurch noch undefiniert. Um diesen Freiheitsgrad zu kontrollieren wurde es nötig, die Phasenverschiebung zwischen dem einsekündigen und dem normierenden **1 PPS**-Signal zu ermitteln. Die Idee lag nun darin, ein Zeitfenster mittels eines Monoflops zu öffnen und per JK-RS-FlipFlop geschickt so zu verriegeln, dass als Resultat nur ein Differenzimpuls generiert würde, der per Tiefpassfilterung dem AVR-Controller an einem seiner ADC-Eingänge zur Verfügung stünde. Die dadurch messbare Spannung wäre proportional zu der Phasenlage des lokalen Oszillators. Somit wäre dann auch die Phasenregelschleife geschlossen. Eine Skizze untermauerte meine ersten Ansätze. Zu erwähnen sei auch, dass das genutzte GPS-Modul vom Typ **Rockwell-Jupiter-GPS TU30-D440** neben einem **1 PPS**-Signal auch ein darauf basierendes **10 kHz**-Signal anlieferte. Dieses betagte Modul hatte allerdings Eigenschaften, die für das Projekt sehr hinderlich waren.

Die obigen Konzepte kondensierten sich dann auf einer handgefertigten Europa-Lochpunkt-Platine, wie sie in Elektronik-Shops leicht bezogen werden können. Die ersten Aufbauten sind in der Regel Bastellösungen, wo hier und dort ein Konzept verworfen oder verbessert werden muss. Das war auch in diesem Fall nicht anders. Die nachfolgenden Bilder zeigen eindrücklich, wie Leitungen verändert werden mussten (aus der ursprünglichen, rechtwinkligen Anordnung kann als eine „ad hoc“ provisorische Lösung ein Draht quer anmontiert worden sein), oder auch dass falschkonzeptionierte Bauteile auf dem Board ihren Platz einnahmen (übergroße Netzdrossel anstelle einer Festinduktivität) und an vielen Stellen teure Potis eingesetzt werden, die später

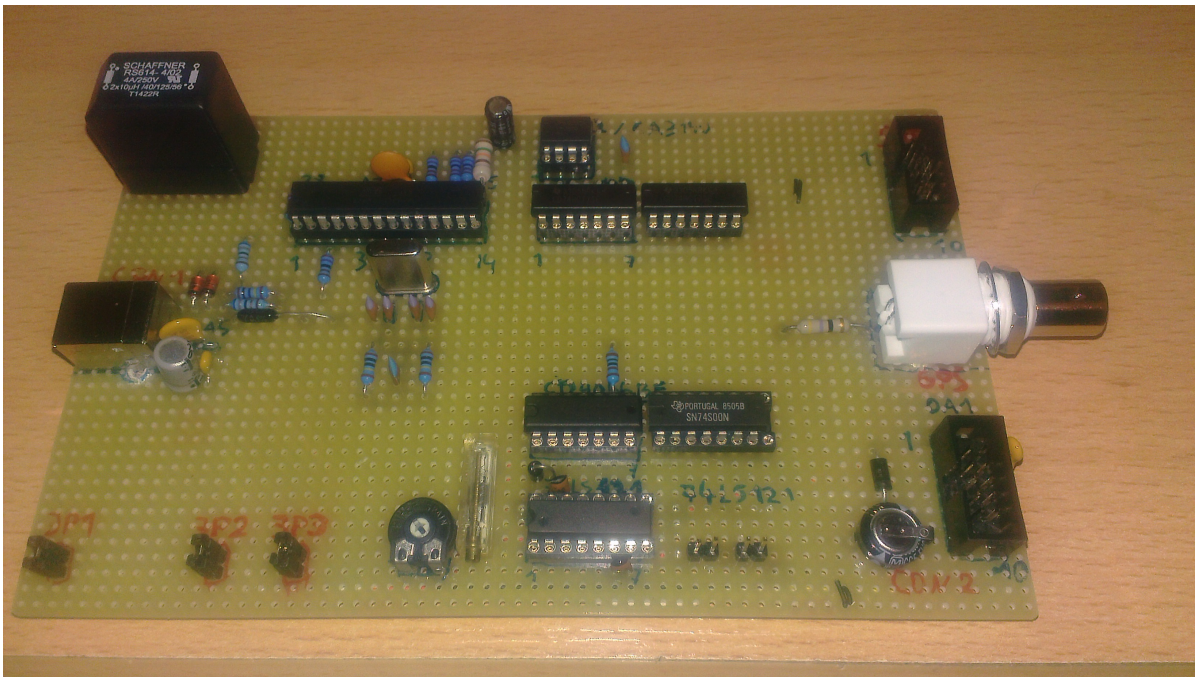


Abbildung 14: Die Vorderseite des Handmusters V1 – der Quarz ist hier noch diskret vorhanden

durch festeingestellte Spannungsteiler ersetzt werden.

3.1.4 USB Anbindung mittels VUSB-Programmcode

Weiterhin besitzt der Atmel **ATmega 328P** keinen dedizierten USB-Port Anschluss. Dennoch ist es mit einer zusätzlichen Programmerweiterung möglich, per Interrupt-Software eine Kommunikation darüber einzurichten. Das Produkt nennt sich **VUSB**[4] und ist frei bei nichtgewerblicher Nutzung.

> ***Danksagung***: hiermit möchte ich meinen Dank an die Entwickler von **VUSB** ausdrücken, die diesen Code entwickelt und ihn mit Auflagen für die Nutzung freigeben haben. Ein herzliches Dankeschön!

3.1.5 Was früher der Quarzofen... heute macht's der TCXO

Nach einer Weile der Software-Optimierung für das Ausregelverhalten des Quarz-Oszillators kristallisierte sich der Entschluss, eine explizite Temperaturregelung vorzusehen. Ich schätzte den Aufwand und das Erfolgsresultat grob ab und nahm den zweiten Lösungsweg an: es solle ein TCXO hierfür genutzt werden. Diese fertig aufgebaute Quarz-Oszillatoren sind als ICs erhältlich. Eine Untervariante dieser temperaturkompensierten Oszillatoren haben einen Voltage-Control (VC) Eingang, der mit einer „Ziehspannung“ belegt wird, um den Oszillator auf dessen Nominalfrequenz zu ziehen – das Übersetzungsverhältnis $\frac{PPM}{Volt}$ liegt bei dem nachfolgend genutzten Exemplar bei etwa 2,5 und ermöglicht so ein sehr feines Einjustieren.

[4]Link: http://www.obdev.at/products/vusb/index-de.html

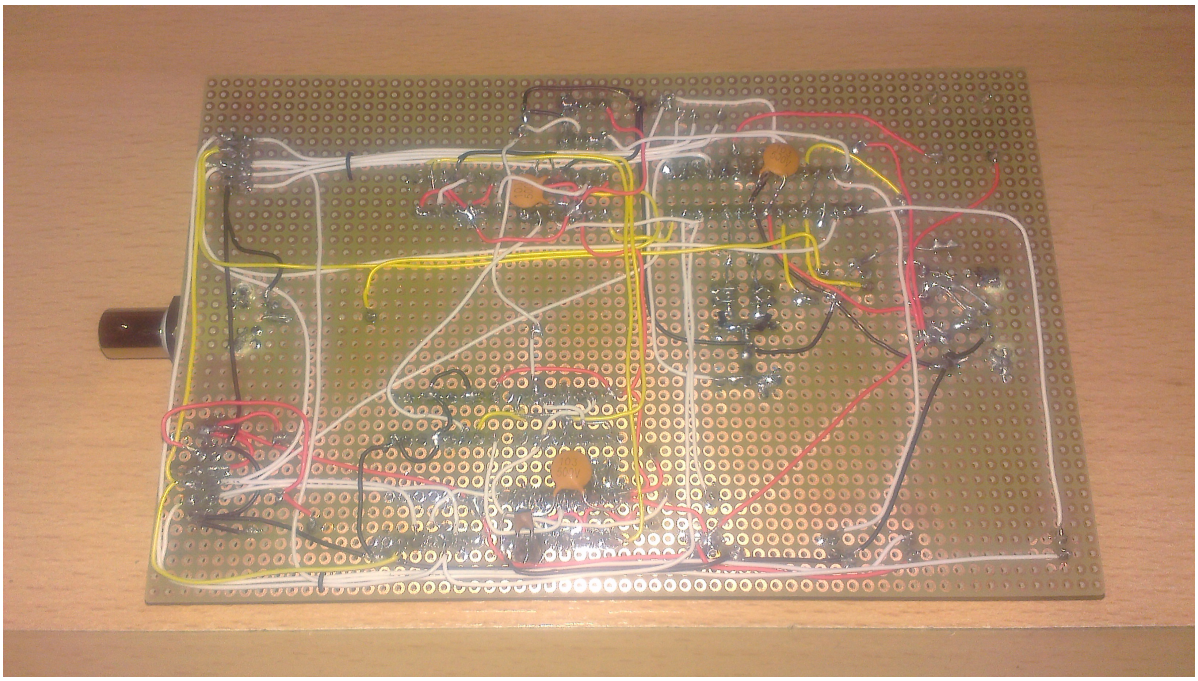


Abbildung 15: Rückseitig das Handmuster nach zahlreichen Optimierungen, erfolgsversprechend und motivierend zugleich

Abbildung 16: VCTCXO für 20 MHz: dieses Bauteil im Container, Größenvergleich anhand eines Radiergummis

Der Kostenpunkt: dieser liegt bei rund 8,- € und ist damit wesentlich teurer als ein einzelner Quarz, dafür hat man mit so einem Device aber eine weitreichende Expertise in der Temperaturkompensation mit eingekauft, was nicht dabei vergessen werden darf. So ist überhaupt erst ein Schaltungsaufbau ohne Quarzofen möglich, denn die Frequenz-Genauigkeit muss mindestens den Anspruch genügen, dass der jeweils nächste Sekundenimpuls mit weniger als ± 10° Phasenverschiebung bezüglich des 20 MHz-Taktes erreicht wird. Das entspräche einem Δ-t von ± 1,4 ns oder ± 1,4 ppb (Parts Per Billion) oder ± 3,5 mV Abweichung der Ziehspannung.

3.1.6 GPS on board - erhältliche GPS-Empfänger auf einem Chip

Der Prototyp V1 entstand mit sehr viel versprechenden Merkmalen. Der sensationelle Fang- und Nachführbereich wurde schon genannt, doch das externe Jupiter GPS-Modul hatte auf Grund seiner frühen technologischen Entwicklung auf dem GPS-Receiver Markt, noch recht begrenzte und sonderbare Eigenheiten. So wurde der Wunsch geboren, für die Weiterentwicklung einen

marktaktuellen GPS-Empfänger Baustein zu verwenden und diesbezüglich Erfahrungen zu sammeln. Doch das sollte noch warten, bis die V2 geboren wurde.

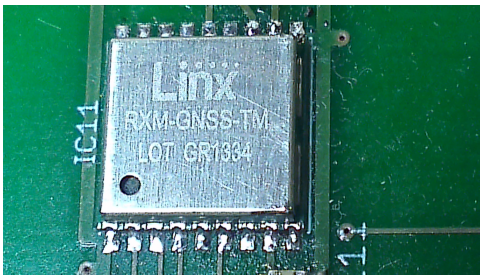


Abbildung 17: GPS-Receiver von Linx – navigiert mit GPS-, Galileo-, QZSS- wie auch mit Glonass-Satelliten

3.2 Leiterplattenfertigung bei BETA-Layout

Die nun mehrfach genannte V2 hatte demnach im Lastenheft stehen, dass einerseits ein VCTCXO verwendet werden müsse, auf der anderen Seite ein GPS-Modul aus aktueller Fertigung. Nach einer einwöchigen Recherche hatte ich mich auf folgende Bauteile festgelegt:

Pos	Bauteile	Einzelpreis
1	**GPS-Modul**: Linx RXM-GNSS-TM-B	53,35 €
2	**TCXO**: IQD CFPT-141 20 MHz	7,99 €
3	**CPLD**: Lattice ispMACH LC4032-V	2,57 €

Tabelle 4: V2 Bauteilauswahl - Auflistung der Großkomponenten

Hinzu kamen noch Kleinteile wie ein Schaltregler für 3,3 Volt oder ein OPA354 Operationsverstärker mit FET-Eingängen für die hochohmige Tiefpass Phasenregelschleife.

Bei der Auswahl der Platinen-Fertiger legte ich einen Schwerpunkt darauf, dass der Hersteller über entsprechendes Know-how verfügt und die bestellbaren Platinen-Exemplare bezahlbar blieben. Auch bot dieser Anbieter an, die gefertigten zwei Leiterplatten elektrisch zu prüfen, was finanziell noch in einem vertretbaren Rahmen lag. Neben vielen weiteren Leiterplatten-Fertigern schloss ich mich dann diesem Anbieter an; die Wahl hätte aber auch anders fallen können, sicherlich ohne ein anderes Resultat zu erhalten.

Auf Grund der in Eagle genutzten DRC-Dateien von BETA-Layout PCB-Pool waren alle in Eagle konstruierten Merkmale ohne Makel einsetzbar. Einzig wurden mir zwei Fehler präsentiert, die ich selbst fabriziert hatte. Bei einem Rip-Up von bereits verlegten Leiterbahnen übersah ich zwei Stellen, die noch als Stummel an einem Netz angeschlossen waren - das hätte zwar nicht zur Fehlfunktion geführt, aber eine aufmerksame BETA-Layout Mitarbeiterin wies mich darauf hin. Ich nahm dankend diese Information zum Anlass, eine korrigierte Fassung nachzureichen.

Bei BETA-Layout gibt es die pfiffige Variante, die Hauptschritte der Fertigung optisch zu Überwachen, auch als „Watch-Ur-PCB“ bezeichnet. Nachteilig ist einzig, dass nur jeweils der letzte

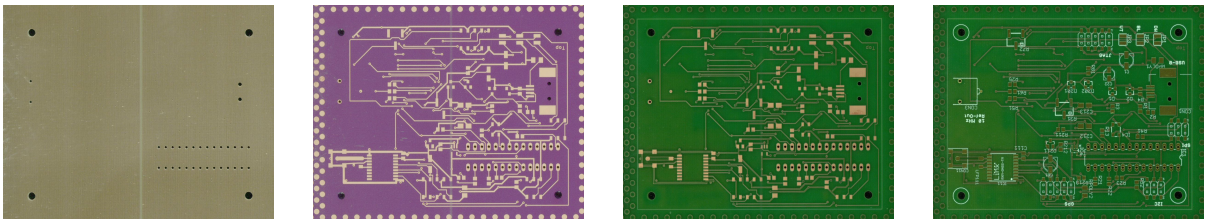


Abbildung 18: Der Fertigungsprozess der Platinenoberseite der V2.0

Prozessschritt auf deren Bilder-Server liegt, die zuvor angelegten Bilddateien werden überschrieben. In meinem Fall führte das dazu, dass ich bei der nächtlichen Fertigung einen Prozessschritt nicht mehr herunterladen konnte. Hier einmal ein paar Eindrücke der Fertigungsschritte.

...

Und dann der Tag, an dem die zwei Platinen sich an meinem Arbeitsplatz materialisierten. Keine zehn Tage zuvor noch die **Eagle-Board-Datei** als Software hochgeladen und nun sind die Hardware-Platinen hier. Großartig, weil es meine erste Platinen sind, die ich jemals habe extern fertigen lassen.

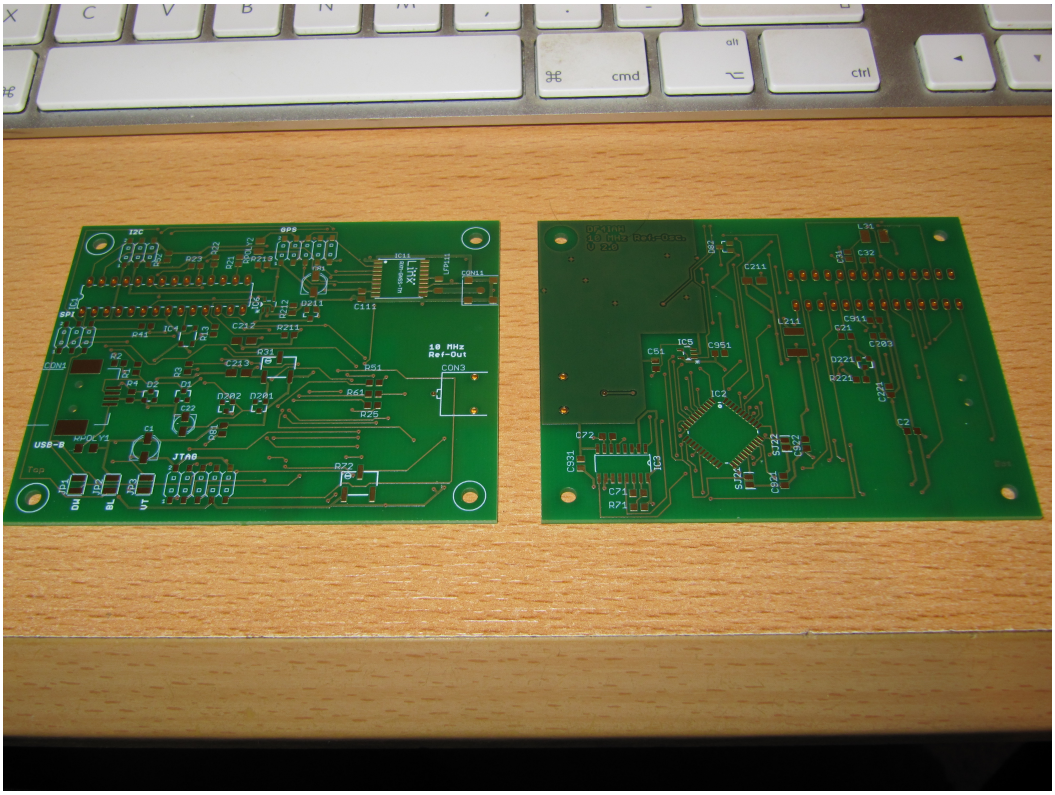


Abbildung 19: Die fertig produzierten Leiterplatten der V2.0

3.3 Inbetriebnahme einzelner Bauabschnitte bis zur Vollfunktion

Die Inbetriebnahme einer Platine erfolgt dahingehend, mit möglichst geringem Materialverlust eine Leiterplatte stückweise zu aktivieren und mögliche Fehlerquellen zu lokalisieren, zu beheben

und zu dokumentieren. So wird für eine zukünftige V2.1 jeder Fehler in Eagle festgehalten und braucht so nicht mehr von den Nachbauenden entfernt zu werden. Nachfolgend die Arbeitsschritte des Einschaltens:

1. Zuerst werden die Bauteile der Stromversorgung aufgebaut und dann schrittweise eingeschaltet:
   a) Fremdeinspeisung durch strombegrenztes Labornetzteil
   b) Schaltregler Impulse analysieren und „nachtunen“
2. Oszillator(en) werden aufgebaut und aktiviert:
   a) Zunahme der Stromaufnahme dabei überprüfen und
   b) das „Ziehverhalten“ des VC-TCXOs überprüfen und
   c) abschließend die Spannung für die Nominalfrequenz bestimmen
3. Danach folgt der AVR-Controller mit folgendem Prozedere:
   a) zuerst den Bootlader mit Hilfe eines AVR-Programmieradapters in den Chip flashen,
   b) dann das Einsetzen des Controllers in seinen Sockel und dem Aufsetzten des Jumpers „BL“,
   c) folgt nun das Verbinden mit dem USB-Anschlusskabel. Ein neues USB-Device wird erkannt.
   d) Mit Hilfe des Bootloaders die Firmware flashen und
   e) nach dem Programmieren dann den USB-Port trennen, Jumper „BL“ entfernen und wieder den USB-Port verbinden,
   f) ein unbekanntes USB-Device soll zu sehen sein.
   g) Das Terminalprogramm starten und die Einschaltmeldungen abwarten - das dauert etwa eine Sekunde.
   h) Ein Hilfe Text erscheint mit den aktuell verfügbaren Kommandos.
4. Nach der CPLD-Programmierung müssen Phasenvergleicher-Signale für den AVR-Controller anliegen.
5. Als Vorletztes kommt das GPS-Modul mit dem **1 PPS**-Signal und der NMEA 0183-Datenübertragung an die Reihe,
6. zuletzt die I²C-Schnittstelle – sie war bereits vorgesehen und dazu noch etwas Platz im ATmega 328P: der Anschluss eines LCD-Moduls. Alternativ zum PC-Anschluss kann an den I²C-Anschluss ein „**MCP23017**“ Port-Expander[5] angeschlossen werden, woran das LCD-Modul „**Conrad 184594 16x2 Zeichen**“[6] mit 8 Bit-parallelem Interface angeschlossen ist - so sind wichtige Statusinformationen auch ohne PC sichtbar. Dann erfolgt die Stromversorgung aus einem USB-Power-Pack heraus.

Jeder Schritt erforderte etwas Zeit. Aber auch lästige Arbeiten wie das „Aufbohren“ eines in Eagle von mir falsch angelegten Bauteils musste ich noch erledigen. Hier hatte ich einen Radius in ein Eingabefeld eingetragen, wo ein Durchmesser verlangt gewesen wäre. Das führte dazu, dass die BNC-Buchse auch bei sanftem Nachdruck keineswegs in ihre vorgesehene Löcher passen wollte.

[5] Bezug beispielsweise hier: **https://hbe-shop.de/Art-1332088-MICROCHIP-MCP23017...**
[6] Bezug beispielsweise hier: **http://www.conrad.de/ce/de/product/184594/...**

Das CPLD war erstaunlich schnell einsatzbereit und mit seiner Programmierung versehen, welche ich noch für gewisse Sonderfälle erweitern und korrigieren musste. Das wäre alles ohne Lötkolben gegangen, hätte ich ein Spezifikations-Detail von Lattice nicht falsch interpretiert: die Schaltzeiten des Lattice sind für mich ausreichend schnell. Auch der Ausgangstreiber war für den nachfolgenden Phasenvergleicher-Tiefpass eigentlich kräftig genug gewesen. Meine Falschannahme beruhte allerdings in einem nicht weiter dokumentiertem Detail - zwischen Flankensteilheit und Laufzeit einerseits und der Treiberstärke auf der anderen darf nicht geschlossen werden, dass der Treiber nach der dokumentierten Laufzeitverzögerung sofort zu 100 % seine definierte Ausgangsleistung erreicht. Dadurch war es mir mit dem Lattice CPLD nicht möglich, ein Zeitfenster für den Ausgangstreiber zu installieren. Durch einen kleinen Umbau gelang es mir jedoch auf dieses „Enable“-Signal gänzlich zu verzichten.

Jede Woche konnte ich so in meinem Ortsverband A20 von den aktuellen Entwicklungen berichten, bis dann am 08. März 2015 auch noch ein optionales LCD-Modul angeschlossen war, welches die V2 nun auch von einem PC-Terminal unabhängig macht – nun genügt ein USB-Power-Pack für die 5V-Versorgung der Schaltung. Die Stromaufnahme bei angedocktem LCD-Modul mit eingeschalteter Beleuchtung liegt nun bei ca. 95 mA. Wird das Modul ohne ein solches Display betrieben, pendelt sich die Stromaufnahme um 55 mA bei 5 V ein, was einer Energieaufnahme von **475 mW mit Display** oder **275 mW ohne ein solches Display** bedeuten würde.

3.3.1 Zusammenfassung & nennenswerte, gemachte Erfahrungen

Hier die gemachten Erfahrungen kurz angeschrieben:

- Viele neue integrierte Schaltungen sind nur noch als SMD-Varianten erhältlich – nicht auf den Spaß mit den neuen Bauteilen verzichten, sondern SMD-Technik dazu lernen!
- SMD-Installation ist leichter, als man es zunächst annimmt. SMD-Löttechnik und -Handhabung ist erlernbar und mit dem richtigen Werkzeug macht es Spaß damit zu arbeiten. **YouTube** ist hierbei eine wertvolle Schatztruhe, diese Löttechnik zu erlernen.
- PCB-Design-Rules von dem Leiterplatten-Hersteller herunterladen, installieren und verstehen. Möglichst die Begrenzung der „Design-Rules“ nicht vorzeitig ausschöpfen sondern Reserven vorsehen.
- Kontur der Platine mit erhältlichen Gehäusen abgleichen, möglicherweise lässt sich so noch etwas Geld sparen.
- Ebenso die Bohrungen für die Leiterplatte vorsehen. Metrisch oder zöllisch gerundete Maße sind leichter handhabbar.
- Mindestens ein „Angstloch“ in der Platine einplanen – nachträgliche Fädeldrahtinstallationen zwischen Vorder- und Rückseite somit ermöglichen.
- Ein kostenloses Tool eines Herstellers kann leicht die teureren Bauteile kompensieren. Erst bei Massenfertigung kehrt sich das Verhältnis um. (Beispiel: kostenlose XILINX – Vivado WebPack Lizenz für neuere FPGA Bausteine).
- Einarbeitungszeit in die Tools vorsehen. Nicht alle Systeme funktionieren geradlinig gedacht. Manchmal arbeiten kleinere Tools im Verbund. Recherche im Internet hilft, diesen Workflow zu finden und zu etablieren.
- Erhältlichkeit und Preise von Bauteilen können heftig schwanken, gerade wenn sie von einem Land mit anderer Währung bezogen werden müssen.
- Bei Preisvergleichen beachten, dass manche Anbieter Netto-Preise anschreiben, gerade wenn sie ihr Stammunternehmen außerhalb der EU haben.

Oder auch folgendermaßen zusammengefasst:

- Der Amateurfunk lebt: neue Technik begeistert viel eher eine junge Generation / Maker-Szene als eine „angestaubte“ Technologie.
- Abwägung: was mache ich selbst, was lasse ich fertigen?
- Kostengünstige Software ermöglicht dem Amateur, Leiterplattenentwurf sowie FPGA / CPLD-Entwicklung auch von zu Hause aus.
- Mit den neu gewonnenen Erfahrungen motiviert neue Ziele ansteuern.

4 Ausblick auf DF4IAH – V3

Derzeit ist die DF4IAH - V3 in Entwicklung, möglicherweise mit dem Zusatz „***Plug Ur SDR***“:

- die**V3** wird wieder ein **10 MHz-Frequenznormal** an Bord haben
- und diesmal mit einem **FPGA** ausgerüstet sein:
  - die MCU wird als eine Soft-CPU realisiert sein (beispielsweise von **opencores.org**),
  - Typ: **XILINX Artix** XC7A35T-2FTG256, max. nutzbare Taktfrequenz noch nicht evaluiert
  - Aufgabe: **GPS-synchronen Takt** zu generieren,
  - viele weitere Funktionen sind denkbar und möglich, Ideen sind gerne willkommen!
- optional: ein **Highspeed ADC** mit mindestens einem **16 Bit** Eingang:
  - Anwendungen: Breitband-Empfänger, Web-SDR Empfänger für das Internet bereitstellen
  - Ausmessen von Funkaussendungen: Laufzeitmessung, Hüllkurve „Fingerprints“ von Sendegeräten erstellen und wiedererkennen
- mehrere **I-/Q-Ausgänge**:
  - Bitbreite der DAC und Grundtyp noch nicht fixiert — eventuell 8 Bit mittels R-2R – Kette
  - Aufbau eines SDR-Senders/-Empfängers leicht möglich – mittels externer Mischer sofort „steckbar“ – fertig ist der »**DF4IAH - Plug Ur SDR - V3**«
  - als Frequenzbasis für Funkstationen, Repeater, digitale Betriebsarten: **PLL-Grundtakt** bereitstellen
  - eigene / neue Modulationsarten realisier- und ausmessbar
  - per **DDS** sehr feine Frequenzschritte möglich
  - per **Soft-CPU** auch komplexe Signale möglich bzw. auswertbar
  - Zeitauflösung bei ca. 5 ns: Laufzeitmess-System, Funkortung.

Viel Freude bei der Umsetzung Ihrer Ziele wünscht DF4IAH, Uli !

Weiterführende Literatur

Literatur

[1] Ulrich Habel, DF4IAH. *Vortragsfolien zu DF4IAH 10 MHz Ref-Osc. V2.* HTTP-Server: http://bg8net.dyndns.org/p/AFu/2015/DF4IAH-10MHzRefOsc-V2/Doc_LyX-TeX/DF4IAH-10MHzRefOsc_V2_Beamer.pdf

[2] Ulrich Habel, DF4IAH. *Von der Idee zur fertigen Platine - ein Werdegang des DF4IAH 10 MHz Ref-Osc. V2 Normfrequenz-Oszillators.* HTTP-Server: http://bg8net.dyndns.org/p/AFu/2015/DF4IAH-10MHzRefOsc-V2/Doc_LyX-TeX/DF4IAH-10MHzRefOsc_V2.pdf

[3] Ulrich Habel, DF4IAH. *Technischer Aufbau des DF4IAH 10 MHz Ref-Osc. V2 Normfrequenz-Oszillators.* HTTP-Server: http://bg8net.dyndns.org/p/AFu/2015/DF4IAH-10MHzRefOsc-V2/Doc_LyX-TeX/DF4IAH-10MHzRefOsc_V2_UKW-Berichte.pdf

[4] Ulrich Habel, DF4IAH. *Aufbauanleitung für den DF4IAH 10 MHz Ref-Osc. V2 Normfrequenz-Oszillator.* HTTP-Server: http://bg8net.dyndns.org/p/AFu/2015/DF4IAH-10MHzRefOsc-V2/Doc_LyX-TeX/DF4IAH_10MHzRefOsc_V2_Aufbauanleitung.pdf

[5] Ulrich Habel, DF4IAH. *Aufbauanleitung für die DF4IAH LCD-Module. V1 Anzeige-Einheit.* HTTP-Server: http://bg8net.dyndns.org/p/AFu/2015/DF4IAH-10MHzRefOsc-V2/Doc_LyX-TeX/DF4IAH_LCD-Module-1V0_Aufbauanleitung.pdf

[6] Ulrich Habel, DF4IAH. *Eclipse-AVR Projekt: DF4IAH 10 MHz Ref-Osc. V2 Sourcen für eine Atmel ATmega 328P MCU.* SVN-Server Leserechte: auf Anfrage.

[7] Ulrich Habel, DF4IAH. *VHDL-Sourcen für das CPLD Lattice LC4032V - DF4IAH 10 MHz Ref-Osc. V2.* SVN-Server Leserechte: auf Anfrage.

[8] Ulrich Habel, DF4IAH. *Eagle-Layout-Dateien für eine doppelseitige Leiterplatte - DF4IAH 10 MHz Ref-Osc. V2.* SVN-Server Leserechte: auf Anfrage.

Bill of Materials

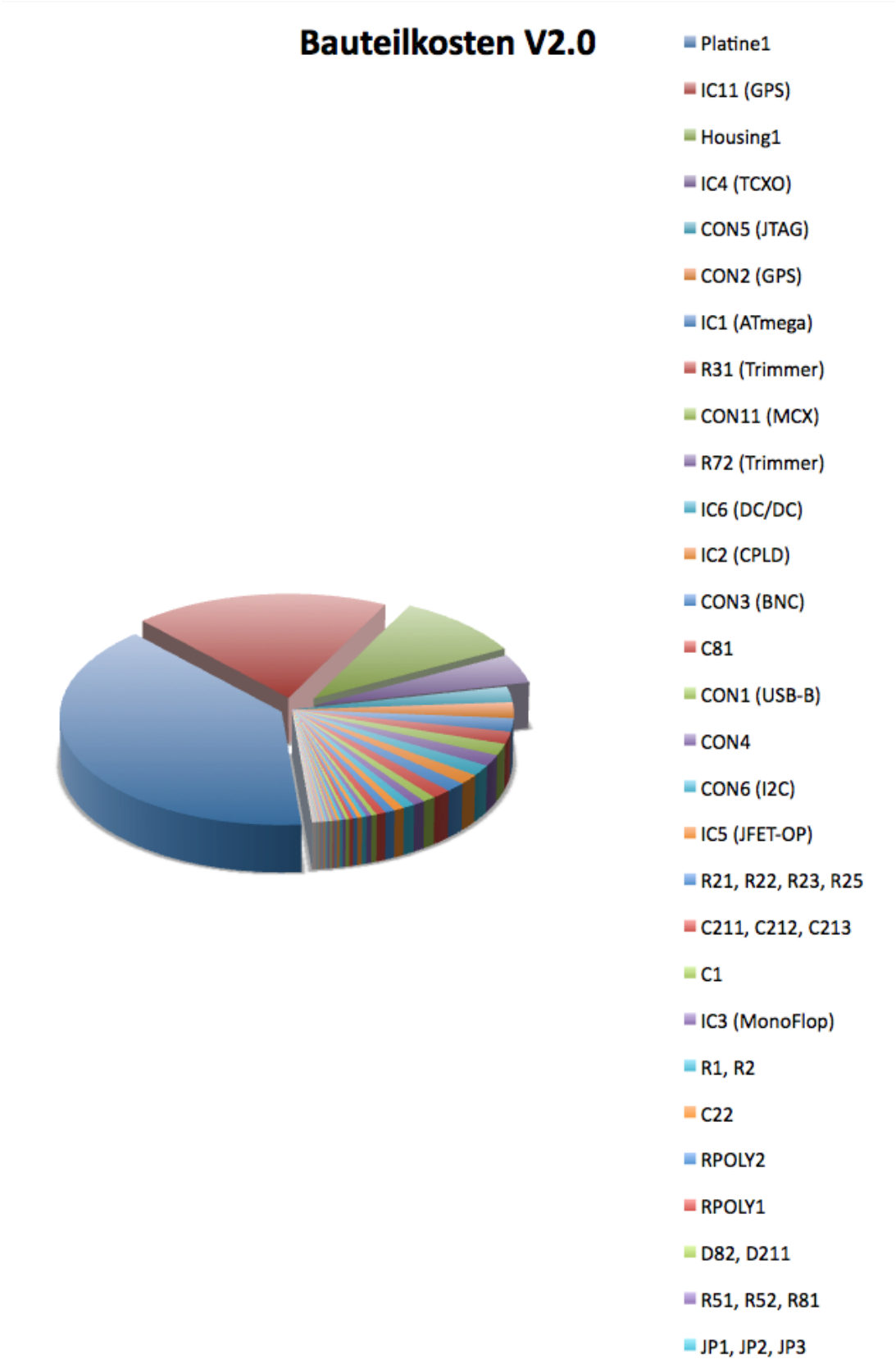
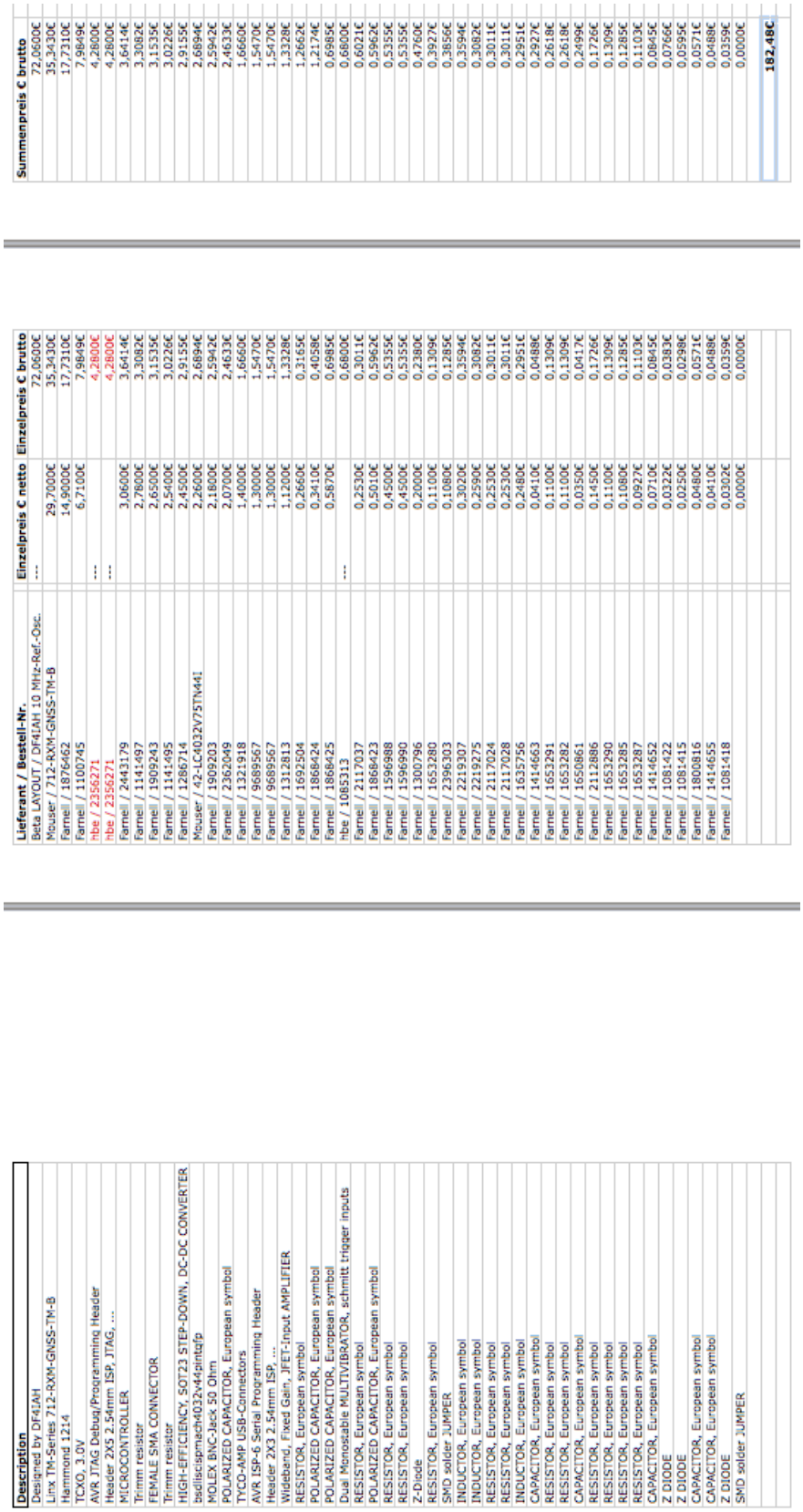


Abbildung 20: Bill of Materials - ohne LCD Ausgabe - Kostenschwerpunkte

Qty	Value	Device	Package	Parts
1	DF4IAH 10 MHz Ref.-Osc. V2	Doppelseitige Platine	90mm x 70mm	Platine1
1	712-RXM-GNSS-TM-B	712-RXM-GNSS-TM-B	712-RXM-GNSS-TM-B	IC11 (GPS)
1	DF4IAH 10 MHz Ref.-Osc. V2	Gehäuse	90mm x 70mm	Housing1
1	CFPT-141	CFPT-141	CFPT-141	IC4 (TCXO)
1	AVR-JTAG-10SMD	AVR-JTAG-10SMD	2X5SMD	CON5 (JTAG)
1	HEADER-2X5SMD	HEADER-2X5SMD	2X5SMD	CON2 (GPS)
1	ATmega328-P	MEGA8-P	DIL28-3	IC1 (ATmega)
1	50k	R-TRIMMTS63Y	RTRIMTS63Y	R31 (Trimmer)
1	MCX-J-P-X-RA-SM1	MCX-J-P-X-RA-SM1	MCX-J-P-X-RA-SM1	CON11 (MCX)
1	10k	R-TRIMMTS63Y	RTRIMTS63Y	R72 (Trimmer)
1	LM2831YMF	LM2831YMF	SOT95P2800X145-5N	IC6 (DC/DC)
1	LC4032V_XXT44	LC4032V_XXT44	TQFP44	IC2 (CPLD)
1	SD-73100-0105	MOLEX_BNC-JACK-50R_SD-73100-0105	MOLEX_BNC-JACK-50R_SD-73100-0105	CON3 (BNC)
1	0.22F	CPOL-EUC	PANASONIC_C	CB1
1	TYCO-AMP_USBUSB-B_SMT	TYCO-AMP_USBUSB-B_SMT	TYCO-AMP_USB-B_SMT	CON1 (USB-B)
1	AVR-ISP-6SMD	AVR-ISP-6SMD	2X3SMD	CON4
1	HEADER-2X3SMD	HEADER-2X3SMD	2X03SMD	CON6 (I2C)
1	LMH6645MF	LMH6645MF	SOT95P2800X145-5N	IC5 (JFET-OP)
4	2.7k	R-EU_R0805	R0805	R21, R22, R23, R25
3	22uF	CPOL-EUA/3216-18W	A/3216-18W	C211, C212, C213
1	47uF	CPOL-EUC	PANASONIC_C	C1
1	74HCT221D	74HCT221D	SO16	IC3 (MonoFlop)
2	68	R-EU_R0805	R0805	R1, R2
1	10uF	CPOL-EUB	PANASONIC_B	C22
1	1206L012WR_0.1A	R-EU_M1206	M1206	RPOLY2
1	1206L025YR_0.5A	R-EU_M1206	M1206	RPOLY1
2	CR508	ZENER-DIODESOT23	SOT23	D82, D211
3	1k	R-EU_R0805	R0805	R51, R52, R81
3		SJW	SJW	JP1, JP2, JP3
1	4.7uH	L-EUL5650M	L5650M	L211
1	10uH	L-EUL1812	L1812	L31
1	51	R-EU_R0805	R0805	R61
1	56	R-EU_R0805	R0805	R221
1	WE-CBF_300R	L-EUL3216C	L3216C	LFP111
6	100n	C-EUC0805	C0805	C2, C21, C31, C111, C203, C221
2	1.5k	R-EU_R0805	R0805	R3, R71
2	10k	R-EU_R0805	R0805	R211, R212
6	10n	C-EUC0805	C0805	C51, C911, C921, C922, C931, C951
1	15k	R-EU_R0805	R0805	R13
1	12k	R-EU_R0805	R0805	R41
1	100k	R-EU_R0805	R0805	R213
1	1M	R-EU_R0805	R0805	R4
1	10p	C-EUC0805	C0805	C71
2	BZX84-C3V6	BZX84CSMD	TO236	D1, D2
2	BZX84-C2V7	BZX84CSMD	TO236	D201, D202
1	33p	C-EUC0805	C0805	C72
1	100p	C-EUC0805	C0805	C32
1	BZX84-C3V0	BZX84CSMD	TO236	D221
2		SJ	SJ	SJ21, SJ22
	Bauteilkosten			

Abbildung 21: Bill of Materials - ohne LCD Ausgabe - Auflistung (1)



Description	Lieferant / Bestell-Nr.	Einzelpreis € netto	Einzelpreis € brutto	Summenpreis € brutto
Designed by DF4IAH	Beta LAYOUT / DF4IAH 10 MHz-Ref.-Osc.	---	72,0600€	72,0600€
Linx TM-Series 712-RXM-GNSS-TM-B	Mouser / 712-RXM-GNSS-TM-B	29,7000€	35,3430€	35,3430€
Hammond 1214	Farnel / 1876462	14,9000€	17,7310€	17,7310€
TCXO, 3.0V	Farnel / 1100745	6,7100€	7,9849€	7,9849€
AVR JTAG Debug/Programming Header	rbe / 2356271	---	4,2800€	4,2800€
Header 2X5 2.54mm ISP, JTAG, ...	rbe / 2356271	---	4,2800€	4,2800€
MICROCONTROLLER	Farnel / 2443179	3,0600€	3,6414€	3,6414€
Trimm resistor	Farnel / 1141497	2,7800€	3,3082€	3,3082€
FEMALE SMA CONNECTOR	Farnel / 1909243	2,6500€	3,1535€	3,1535€
Trimm resistor	Farnel / 1141495	2,5400€	3,0226€	3,0226€
HIGH-EFFICIENCY, SOT23 STEP-DOWN, DC-DC CONVERTER	Farnel / 1286714	2,4500€	2,9155€	2,9155€
bsdlascpmach4032v44pintqfp	Mouser / 42-LC4032V75TN44I	2,2600€	2,6894€	2,6894€
MOLEX BNC-Jack 50 Ohm	Farnel / 1909203	2,1800€	2,5942€	2,5942€
POLARIZED CAPACITOR, European symbol	Farnel / 2362049	2,0700€	2,4633€	2,4633€
TYCO-AMP USB-Connectors	Farnel / 1321918	1,4000€	1,6660€	1,6660€
AVR ISP-6 Serial Programming Header	Farnel / 9689567	1,3000€	1,5470€	1,5470€
Header 2X3 2.54mm ISP, ...	Farnel / 9689567	1,3000€	1,5470€	1,5470€
Wideband, Fixed Gain, JFET-Input AMPLIFIER	Farnel / 1312813	1,1200€	1,3328€	1,3328€
RESISTOR, European symbol	Farnel / 1692504	0,2660€	0,3165€	1,2662€
POLARIZED CAPACITOR, European symbol	Farnel / 1868424	0,3410€	0,4058€	1,2174€
POLARIZED CAPACITOR, European symbol	Farnel / 1868425	0,5870€	0,6985€	0,6985€
Dual Monostable MULTIVIBRATOR, schmitt trigger inputs	rbe / 1085313	---	0,6800€	0,6800€
RESISTOR, European symbol	Farnel / 2117037	0,2530€	0,3011€	0,6021€
POLARIZED CAPACITOR, European symbol	Farnel / 1868423	0,5010€	0,5962€	0,5962€
RESISTOR, European symbol	Farnel / 1596988	0,4500€	0,5355€	0,5355€
RESISTOR, European symbol	Farnel / 1596990	0,4500€	0,5355€	0,5355€
Z-Diode	Farnel / 1300796	0,2000€	0,2380€	0,4760€
RESISTOR, European symbol	Farnel / 1653280	0,1100€	0,1309€	0,3927€
SMD solder JUMPER	Farnel / 2396303	0,1080€	0,1285€	0,3856€
INDUCTOR, European symbol	Farnel / 2219307	0,3020€	0,3594€	0,3594€
INDUCTOR, European symbol	Farnel / 2219275	0,2590€	0,3082€	0,3082€
RESISTOR, European symbol	Farnel / 2117024	0,2530€	0,3011€	0,3011€
RESISTOR, European symbol	Farnel / 2117028	0,2530€	0,3011€	0,3011€
INDUCTOR, European symbol	Farnel / 1635756	0,2480€	0,2951€	0,2951€
CAPACITOR, European symbol	Farnel / 1414663	0,0410€	0,0488€	0,2927€
RESISTOR, European symbol	Farnel / 1653291	0,1100€	0,1309€	0,2618€
RESISTOR, European symbol	Farnel / 1653282	0,1100€	0,1309€	0,2618€
CAPACITOR, European symbol	Farnel / 1650861	0,0350€	0,0417€	0,2499€
RESISTOR, European symbol	Farnel / 2112886	0,1450€	0,1726€	0,1726€
RESISTOR, European symbol	Farnel / 1653290	0,1100€	0,1309€	0,1309€
RESISTOR, European symbol	Farnel / 1653285	0,1080€	0,1285€	0,1285€
RESISTOR, European symbol	Farnel / 1653287	0,0927€	0,1103€	0,1103€
CAPACITOR, European symbol	Farnel / 1414652	0,0710€	0,0845€	0,0845€
Z DIODE	Farnel / 1081422	0,0322€	0,0383€	0,0766€
Z DIODE	Farnel / 1081415	0,0250€	0,0298€	0,0596€
CAPACITOR, European symbol	Farnel / 1800816	0,0480€	0,0571€	0,0571€
CAPACITOR, European symbol	Farnel / 1414655	0,0410€	0,0488€	0,0488€
Z DIODE	Farnel / 1081418	0,0302€	0,0359€	0,0359€
SMD solder JUMPER		0,0000€	0,0000€	0,0000€
				182,48€

Abbildung 22: Bill of Materials - ohne LCD Ausgabe - Auflistung (2)