I/Q Modulator

Es soll ein I/Q Modulator entstehen mit dem man auf einfache Art und Weise HF erzeugen und modulieren kann. Einzig ein Lokaler Oszillator(LO) und eine Soundkarte wird benötigt.

Projektstatus

- Schaltplan erstellt: **Rev3 erledigt, vorgelegt** — *Sebastian Weiß 2014/02/03 06:52*
- Layout erstellt: **Rev3 erledigt, vorgelegt** — *Sebastian Weiß 2014/02/03 06:52*
- Review Schaltplan: **erledigt, Rev 3** — *Stefan Biereigel 2014/02/03 09:27*
- Review Layout: **erledigt, Rev 3** — *Stefan Biereigel 2014/02/03 09:27*
- Fertigung Leiterplatte: **erledigt** — *Sebastian Weiß 2014/02/03 14:17*
- Inbetriebnahme Leiterplatte: **erledigt** — *Sebastian Weiß 2014/02/06 14:47*

Schaltungsbeschreibung

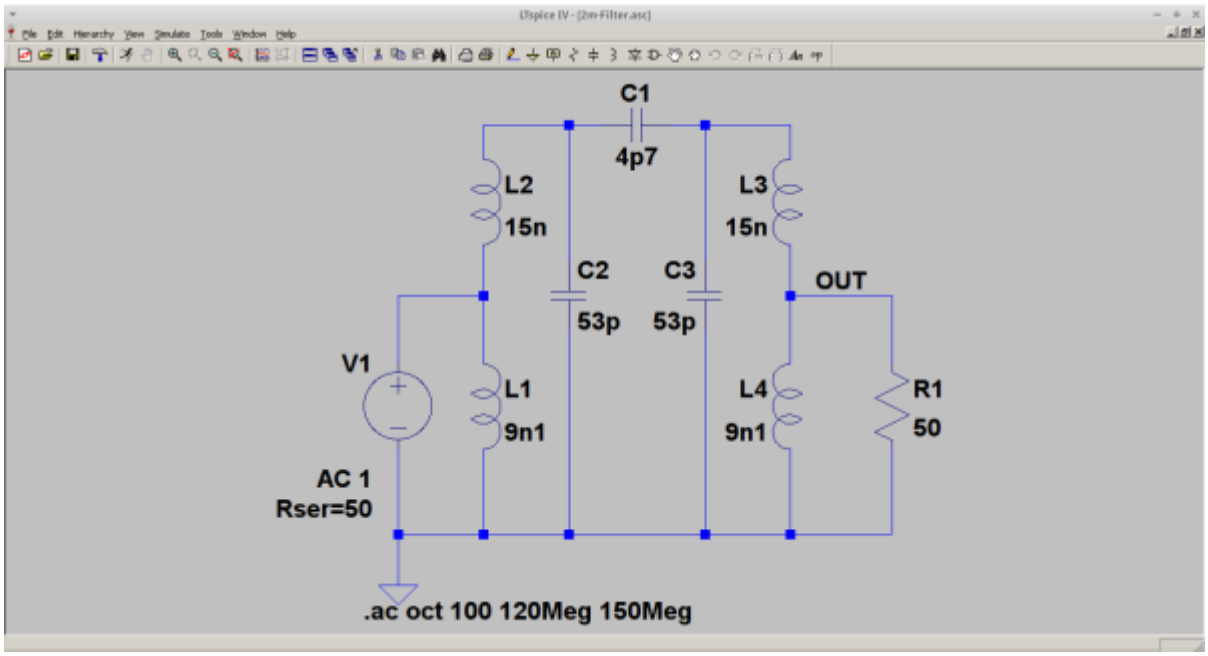
Die Signale einer handelsüblichen (preiswerten) Soundkarte sind massebezogen(single-ended). Die Basisbandsignale(I/Q) aus der Soundkarte werden in differentielle Signale umgewandelt. Für die Verwendung der Schaltung mit einem Rechtecksignal als LO dient das Filter am LO-Eingang. Danach wird der LO einem Puffer zugeführt, der differentielle Signale für den Modulator bereitstellt. Zusätzlich kann das LO-Signal per Dämpfungsglied im Pegel angepasst werden

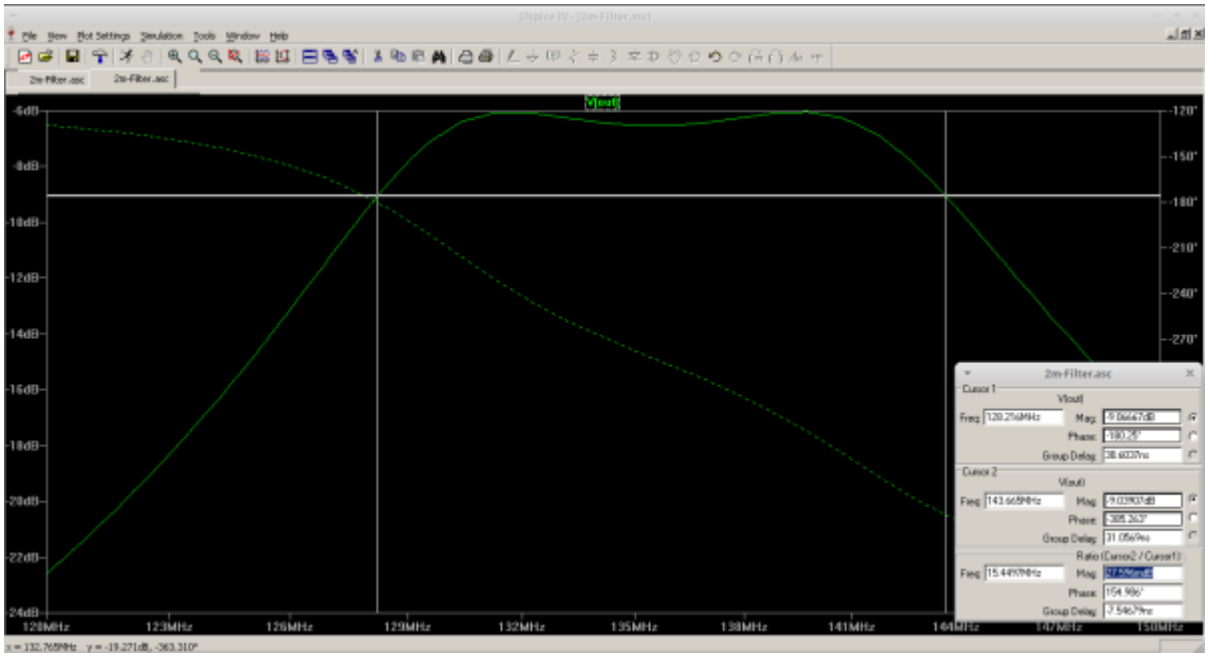
Zu erreichbare Daten:

- Frequenzbereich: 140..500MHz → 2m- und 70cm-Band nutzbar
  - LO-Buffer: 10..500MHz
  - Mischer: 140..1000MHz
- Frequenzbereich BB: 0..80MHz → auch für (D)ATV geeignet: DIGI lite
- Ausgangsleistung: 0dBm (1mW)

LO-Filter

Das Design des Filters ist schon erprobt, es macht sich gut im 2m-LNA vom SDR. Initial bekam ich damals ™ eine 1cmx2cm große Leiterplatte von Winni DL2AWT mit genau dieser Filtertopologie. Folglich simulierte ich das Ding mal und fand passende Werte, die es auch bei Reichelt gibt. C2 und C3 sind dabei Trimmkondensatoren(3-10pF) mit 47pF Keramikkondensator parrallel geschalten.





LTSpice-File

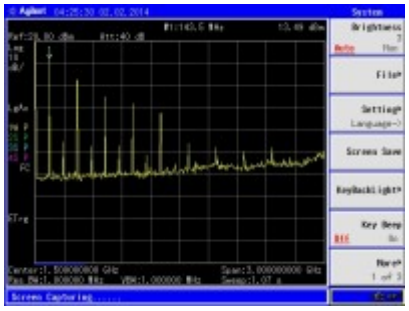
Erreichbare Daten:

- Einfügedämpfung: 0dB
- 3dB-Bandbreite: 15.5MHz

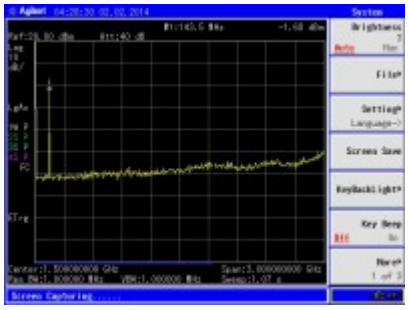
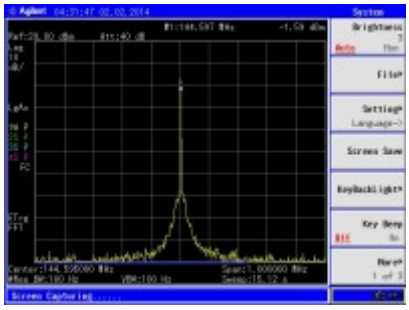
Untersuchung verschiedener LOs

Si570

<spoiler> Überblick Spektrum bei Sollfrequenz 144,6MHz:

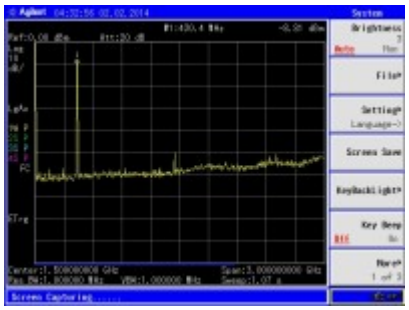
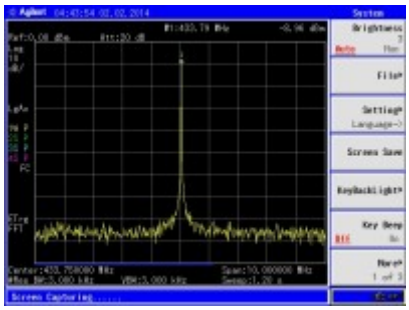


Spektrum mit 2m-Filter:

- Leistung: -1,6dBm

Spektrum mit 70cm-Filter:

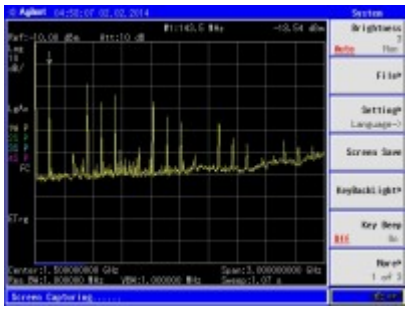



- Leistung: -9dBm

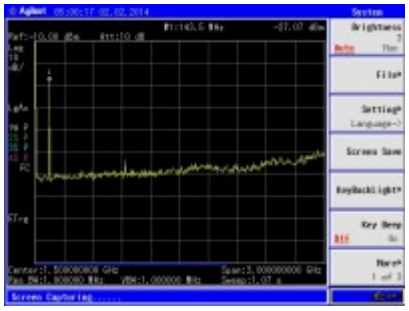
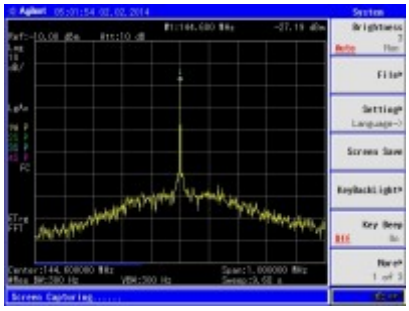
</spoiler>

VNWA

<spoiler> Überblick Spektrum bei Sollfrequenz 144,6MHz:

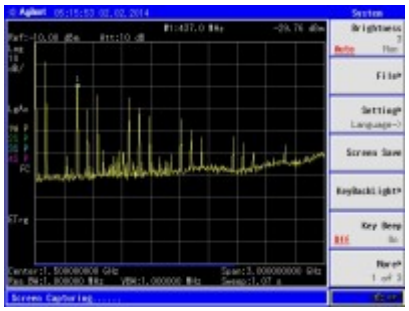


Spektrum mit 2m-Filter:

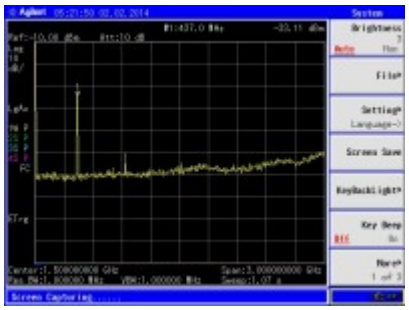
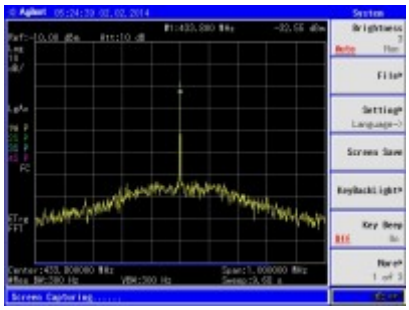



- Leistung: -27,2dBm

Überblick Spektrum bei Sollfrequenz 433,8MHz:



Spektrum mit 70cm-Filter:

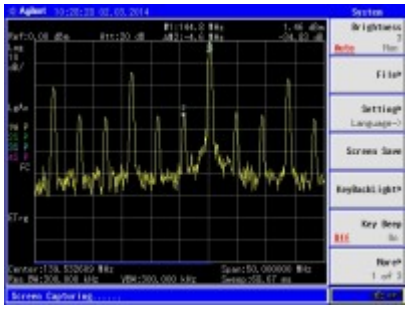
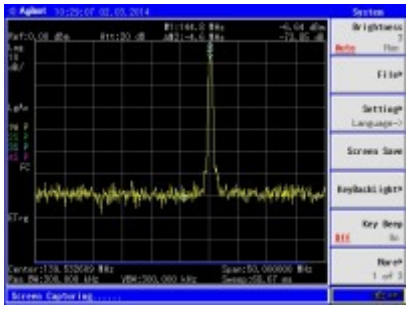



- Leistung: -32,6dBm

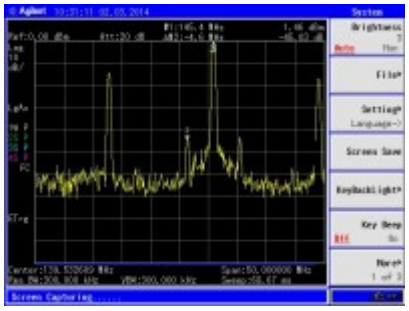
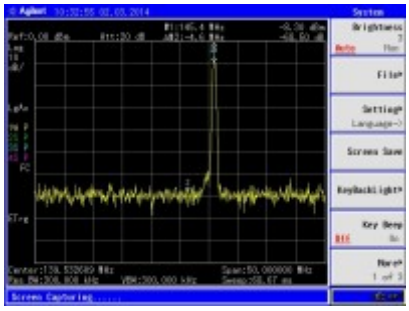
</spoiler>

Raspberry Pi per GPCLK0

<spoiler> Spektrum APRS(144,800MHz) links direkt, rechts mit 2m-Filter:

Spektrum 145,450MHz links direkt, rechts mit 2m-Filter:

C-Code zur Ansteuerung: <spoiler>

PiLO.c

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <ctype.h>
#include <dirent.h>
#include <math.h>
#include <fcntl.h>
#include <assert.h>
#include <sys/mman.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <signal.h>
#include <malloc.h>
#include <time.h>

#define BCM2708_PERI_BASE        0x20000000
#define GPIO_BASE                (BCM2708_PERI_BASE + 0x200000) /* GPIO
controller */
#define PAGE_SIZE (4*1024)
#define BLOCK_SIZE (4*1024)

int  mem_fd;
char *gpio_mem, *gpio_map;
char *spi0_mem, *spi0_map;

// I/O access
volatile unsigned *gpio = NULL;
volatile unsigned *allof7e = NULL;

// GPIO setup macros. Always use INP_GPIO(x) before using OUT_GPIO(x)
or SET_GPIO_ALT(x,y)
#define INP_GPIO(g) *(gpio+((g)/10)) &= ~(7<<(((g)%10)*3))
#define OUT_GPIO(g) *(gpio+((g)/10)) |=  (1<<(((g)%10)*3))
#define SET_GPIO_ALT(g,a) *(gpio+(((g)/10))) |=
(((a)<=3?(a)+4:(a)==4?3:2)<<(((g)%10)*3))

#define GPIO_SET *(gpio+7)  // sets   bits which are 1 ignores bits
```

```
which are 0
#define GPIO_CLR *(gpio+10) // clears bits which are 1 ignores bits
which are 0
#define GPIO_GET *(gpio+13) // sets   bits which are 1 ignores bits
which are 0

#define ACCESS(base) *(volatile int*)((int)allof7e+base-0x7e000000)
#define SETBIT(base, bit) ACCESS(base) |= 1<<bit
#define CLRBIT(base, bit) ACCESS(base) &= ~(1<<bit)
#define CM_GP0CTL (0x7e101070)
#define GPFSEL0 (0x7E200000)
#define PADS_GPIO_0_27  (0x7e10002c)
#define CM_GP0DIV (0x7e101074)
#define CLKBASE (0x7E101000)

struct GPCTL {
    char SRC         : 4;
    char ENAB        : 1;
    char KILL        : 1;
    char             : 1;
    char BUSY        : 1;
    char FLIP        : 1;
    char MASH        : 2;
    unsigned int     : 13;
    char PASSWD      : 8;
};

void txon()
{
    if(allof7e == NULL){
      allof7e = (unsigned *)mmap(
                  NULL,
                  0x01000000,  //len
                  PROT_READ|PROT_WRITE,
                  MAP_SHARED,
                  mem_fd,
                  0x20000000  //base
              );
      if ((int)allof7e==-1) exit(-1);
    }

    SETBIT(GPFSEL0 , 14);
    CLRBIT(GPFSEL0 , 13);
    CLRBIT(GPFSEL0 , 12);

    // Set GPIO drive strength, more info:
http://www.scribd.com/doc/101830961/GPIO-Pads-Control2
    //ACCESS(PADS_GPIO_0_27) = 0x5a000018 + 0;  //2mA -3.4dBm
    //ACCESS(PADS_GPIO_0_27) = 0x5a000018 + 1;  //4mA +2.1dBm
    //ACCESS(PADS_GPIO_0_27) = 0x5a000018 + 2;  //6mA +4.9dBm
    ACCESS(PADS_GPIO_0_27) = 0x5a000018 + 3;  //8mA +6.6dBm(default)
```

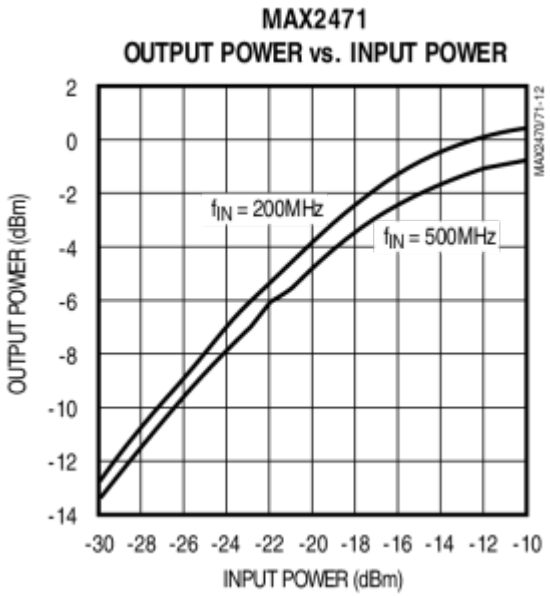
```
    //ACCESS(PADS_GPIO_0_27) = 0x5a000018 + 4;  //10mA +8.2dBm
    //ACCESS(PADS_GPIO_0_27) = 0x5a000018 + 5;  //12mA +9.2dBm
    //ACCESS(PADS_GPIO_0_27) = 0x5a000018 + 6;  //14mA +10.0dBm
    //ACCESS(PADS_GPIO_0_27) = 0x5a000018 + 7;  //16mA +10.6dBm

    struct GPCTL setupword = {5/*SRC*/, 1, 0, 0, 0, 1,0x5a};
    ACCESS(CM_GP0CTL) = *((int*)&setupword);
}

void txoff()
{
    struct GPCTL setupword = {5/*SRC*/, 0, 0, 0, 0, 1,0x5a};
    ACCESS(CM_GP0CTL) = *((int*)&setupword);
}

void handSig() {
  exit(0);
}

// Set up a memory regions to access GPIO
void setup_io()
{
    /* open /dev/mem */
    if ((mem_fd = open("/dev/mem", O_RDWR|O_SYNC) ) < 0) {
        printf("can't open /dev/mem \n");
        exit (-1);
    }

    /* mmap GPIO */

    // Allocate MAP block
    if ((gpio_mem = malloc(BLOCK_SIZE + (PAGE_SIZE-1))) == NULL) {
        printf("allocation error \n");
        exit (-1);
    }

    // Make sure pointer is on 4K boundary
    if ((unsigned long)gpio_mem % PAGE_SIZE)
        gpio_mem += PAGE_SIZE - ((unsigned long)gpio_mem % PAGE_SIZE);

    // Now map it
    gpio_map = (unsigned char *)mmap(
                    gpio_mem,
                    BLOCK_SIZE,
                    PROT_READ|PROT_WRITE,
                    MAP_SHARED|MAP_FIXED,
                    mem_fd,
                    GPIO_BASE
                );

    if ((long)gpio_map < 0) {
```

```
        printf("mmap error %d\n", (int)gpio_map);
        exit (-1);
    }

    // Always use volatile pointer!
    gpio = (volatile unsigned *)gpio_map;


}

int main(int argc, char *argv[])
{
  setup_io();
  txon();
  ACCESS(CM_GP0DIV) = (0x5a << 24) | (6 << 12) | (896<<2); // 6, 928 für aprs 144,796; 6, 912 für 145,125; 6, 896 für 145,450
  return 0;
}
```

</spoiler> </spoiler>

LO	Frequenz	Leistung
Si570	144,6MHz	6,3dBm
Si570	433,8MHz	-1dBm
VNWA	144,6MHz	-19,2dBm
VNWA	433,8MHz	-24,6dBm

Der I/Q-Modulator will -10dBm..0dBm am LO-Eingang. Der LO-Buffer verstärkt das LO-Signal wie folgt:



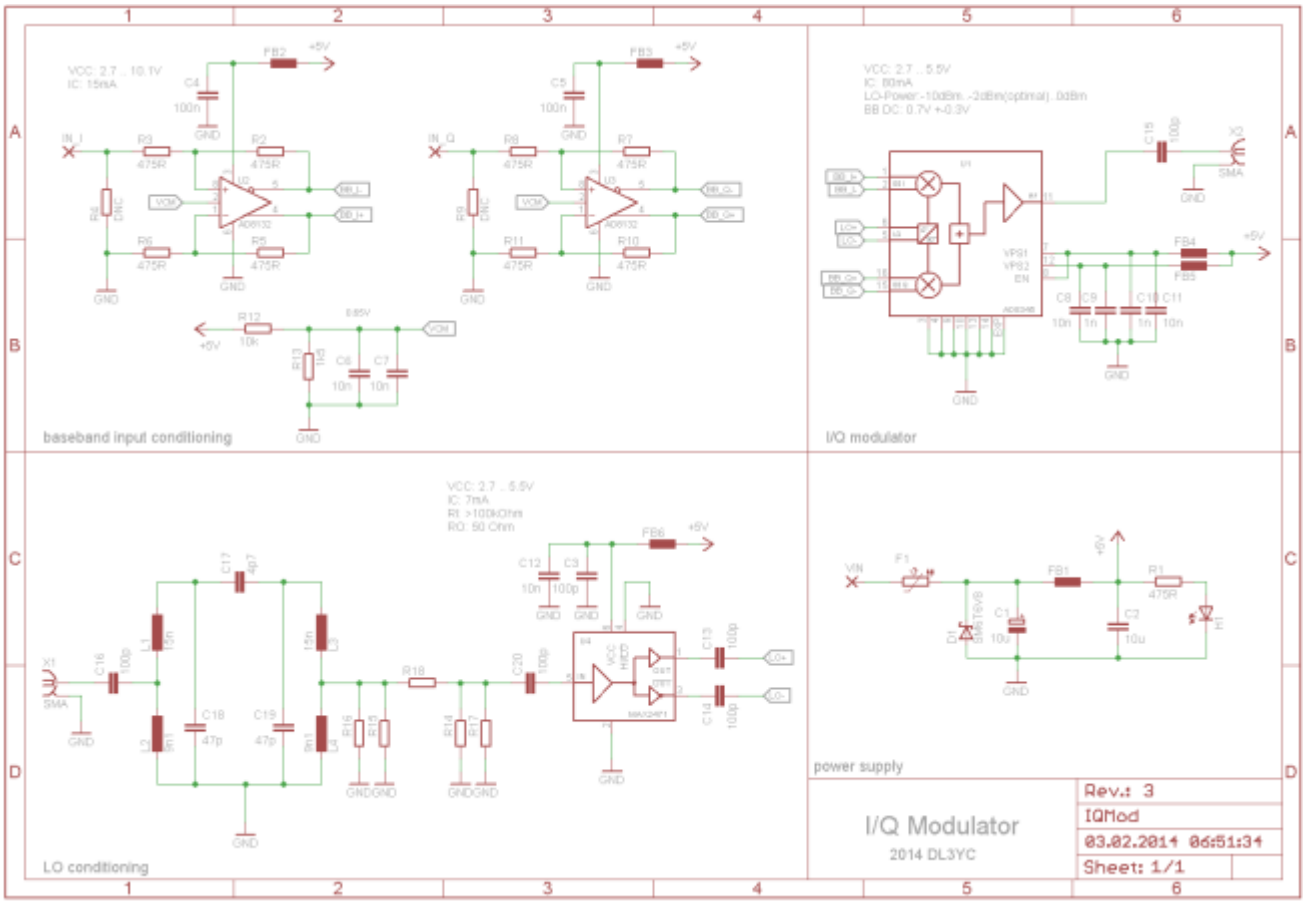
Somit muss der LO mindestens -26dBm liefern.

TODO:

- ATTiny45 PLL-Ausgang: Spektrum, Leistung - z.B. 48MHz → 2m(3. Harmonische) und 96MHz → 70cm(3.Harmonische)
    - es ergeben sich 144MHz(Bandanfang 2m) und 432MHz(Bandmitte 70cm), Untersuchung der möglichen Frequenzschritte
    - ATTiny auch als Modulator(TinyFox)?
- würde man dann quasi seine NF erst Frequenzmodulieren und dann hoch/runtermsichen um es spektral zu verschieben? klingt spannend :)
    - genau, man macht die FM als Basisbandverarbeitung und hat dort die bekannten pos. und neg. Frequenzen; danach gehts in den Mixer

Schaltplan

Revision: 3



- als PDF-Datei
- als Eagle-Datei

Vorberechnung des Dämpfungsglieds

Dämpfung	R15,R16	R14,R17	R18
0dB	-	100Ω	-
-20dB	120Ω	56Ω	237Ω
-25dB	120Ω	56Ω	475Ω

Links zu den verwendeten Schaltkreisen:

- differentielle Treiber AD8132
- I/Q Modulator AD8345
- VCO Buffer MAX2470

Inspiriert durch: Khaled Al Rifai

Änderungen

Revision 1

Sebastian Weiß 2014/01/31 22:53

- LO-Aufbereitung um Dämpfungsglied ergänzt für Verstärkung des Buffers zu kompensieren und Pegel-Anpassung an den verwendeten LO vornehmen zu können

Review:

- Am AD8132 ist I+ am negativen Ausgang des Opamps, und I- am positiven –> Symbolfehler, Pin 4 und 5 getauscht?
  - Wenn, dann ist das auch schon im Datenblatt von AD8345 Seite 14 falsch.
  - das machen alle ™ so: ADC Input Driver
  - es ergibt auch Sinn, wenn es eine Gegenkopplung statt einer Mitkopplung gibt
  - auf den invertierenden Ausgang per Kreis im Symbol hinweisen
  - Eingangspolarität am AD8345 tauschen

Revision 2

Sebastian Weiß 2014/02/01 15:54

- AD8132 Symbolanpassung: Markierung invertierender Ausgang
- AD8345
  - Symbolanpassung: optische Anpassung
  - Footprint: Exposed-Pad abgerundet
- Polarität der differentiellen Basisbandsignale korrigiert

Review:

- ist OK so — *Stefan Biereigel 2014/02/02 08:59*
- MAX2470 hat hochomigen Eingang → 50Ω Widerstand zur Anpassung des Eingangs hinzufügen
- Koppel-Kondensator vor dem Eingang des MAX2470 hinzufügen — *Sebastian Weiß 2014/02/02 13:07*

Revision 3

— *Sebastian Weiß 2014/02/03 06:52*

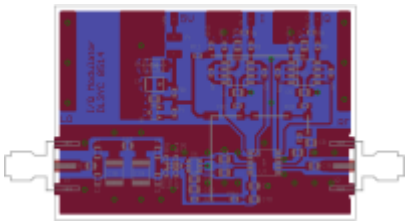
- Koppel-Kondensator vor dem Eingang des MAX2470 hinzugefügt.

Review

- was ist mit dem angesprochenen 50 Ohm Widerstand dort am Eingang? Weggelassen? — *Stefan Biereigel 2014/02/03 09:24*
    - Nein, der wird mit in die 2 Widerstände des Dämpfungsglieds eingerechnet

Layout

Revision: 3



- als PDF-Datei
- als Eagle-Datei

Änderungen

Revision 1:

Review:

- Positionierabstand D1 / C1 evtl vergrößern
- Leiterbahn zu X2 verbreitern (Anpassung) - Relevant?
    - 2mm sind nicht relevant → ganze Leiterplatte verkleinern
- Schrift vergrößern?
    - war durch ein group all/change size mit den Bauteilnamen verkleinert wurden
- I/Q-Anschlüsse evtl weiter zusammenrücken? Würde das Anlöten von Klinkenkabel o.ä. vereinfachen
- Befestigungsbohrungen
    - Gehäuse soll eine kleine Box aus Leiterplattenmaterial werden → Masse noch drumherum zum anlöten

Revision 2:

- PCB horizontal verkleinert
- Abstand C1 ↔ D1 vergrößert
- Anpassung an korrigierte Polarität der differentiellen Basisbandsignale

Review:

- Positionierung X1 und X2 - aus dem Gehäuse muss dann die komplette, (quadratische?) Grundfläche der SMA-Stecker ausgefräst werden, wenn das so gewollt ist - OK, ansonsten müssten sie etwas weiter nach innen gesetzt werden
- ansonsten so OK
  - Die Steckverbinder werden wohl doch SMB, SMA-Buchsen verteuern die Leiterplatte um 1000%

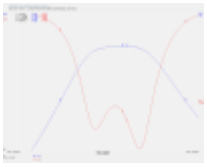
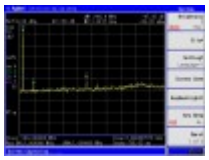
Revision 3:

- Koppelkondensator C20 hinzugefügt

Review:

- kann man C20, R17, R14 usw denn so noch vernünftig bestücken? Sieht sehr eng dort aus.
- ansonsnten keine Regressionen festzustellen

Anleitung zur Inbetriebnahme

1. Bestückung Spannungsaufbereitung
   1. 0..10V an VIN anlegen
   2. Auslösespannung TVS-Diode: *6,8 V*(Soll: 6,5V)
   3. Verpolungstest: 0..-30V, Spannung an Betriebsspannung der Schaltkreise: max. -0,8V
2. Bestückung X1(SMA, LO-Eingang) und Filter ohne Dämpfungsglied; Koax von Mittenanzapfung L3/L4 zu VNA
   1. VNWA-Messung LO-Filter: 
   2. Spektrum Si570 mit Filter: 
3. Bestückung Rest, 50Ohm-Abschluss an X1 + X2
   1. Stromaufnahme bei VIN=5V: *xxx mA* (Soll:90..120mA) **TODO**
   2. Prüfung der Spannung über C3, C4, C5, C8, C10 (Soll: VIN) **OK**
   3. Spannng und Ripple VCM: *0,8 V*(Soll: 0,7V)
4. Messungen per Oszilloskop, Signalgenerator an X1 f=144MHz; f=1kHz an IN_I/IN_Q:
   1. Phasenbeziehung zwischen LO+ und LO-: *xxx*°(Soll: 180°) **entfällt**
   2. Phasenbeziehung zwischen BB_I+ und BB_I-: *xxx*°(Soll: 180°) **entfällt**
   3. Phasenbeziehung zwischen BB_Q+ und BB_Q-: *xxx*°(Soll: 180°) **entfällt**
5. Spektrumanalyzer an X2, Signalgenerator an X1 f=144MHz, IN_I und IN_Q mit 1kHz SSB-Signal
   1. Spiegelfrequenzunterdrückung: *xxx dB*
   2. LO Leakage: *xxx dBc*

Inbetriebnahme-Schaltplan

Ansteuerung

Um normale Audio-Dateien FM-moduliert als I/Q-Signale mit einer Soundkarte ausgeben zu können, gibt es folgendes kleines Tool: **fmmod**

fmmod.c

```
/* complex frequency modulator for i/q signal processing
 * Sebastian Weiss <dl3yc@darc.de>
 * Di 4. Feb 16:10:01 CET 2014
 */

#include <stdio.h>
#include <stdlib.h>
#include <sndfile.h>
#include <math.h>

#define BUFFER_LEN  1024

void usage(void)
{
    printf("Usage: fmmod inputfile outputfile [carrier] [freqdev]\n \
            inputfile: path to input file(needs to be single channel)\n
\
            outputfile: path to output file\n \
            carrier: carrier frequency in Hz(standard: 0.0)\n \
            freqdev: frequency deviation in Hz(standard: 3000)\n");
}

void process_data(double *data, int count, int fs, double fc, double
freqdev)
{
    static double old_phase = 0.0;
    double cumsum[BUFFER_LEN];
    double phase;
    int i;

    if (count == 0)
        return;

    cumsum[0] = data[0]/fs;
    for (i = 1; i < count; i++) {
        cumsum[i] = cumsum[i-1] + data[i]/fs;
    }

    for (i=0; i < count; i++) {
        phase = 2*M_PI*fc/fs*(i+1) + 2*M_PI*freqdev*cumsum[i] +
old_phase;
        data[2*i] = cos(phase);
        data[2*i + 1] = -sin(phase);
```

```
    }

    old_phase = fmod(phase, 2*M_PI);
    return;
}

int main(int argc, char *argv[])
{
    static double data[2*BUFFER_LEN];
    SNDFILE *infile, *outfile;
    SF_INFO sfinfo;
    int readcount;
    double carrier;
    double freqdev;

    if ((argc > 5) || (argc < 3)) {
        usage();
        return 1;
    }

    freqdev = (argc == 5) ? atof(argv[4]) : 3000.0;
    carrier = (argc > 3) ? atof(argv[3]) : 0.0;

    infile = sf_open(argv[1], SFM_READ, &sfinfo);
    if (!infile) {
        printf("%s: Not able to open input file %s\n", argv[0],
argv[1]);
        sf_perror(NULL);
        return 1;
    }

    if (sfinfo.channels > 1) {
        printf("%s: Not able to process more than 1 channel\n",
argv[0]);
        return 1;
    }

    sfinfo.channels = 2;

    outfile = sf_open(argv[2], SFM_WRITE, &sfinfo);
    if (!outfile) {
        printf("%s: Not able to open output file %s\n", argv[0],
argv[2]);
        sf_perror(NULL);
        return 1;
    }

    while ((readcount = sf_read_double(infile, data, BUFFER_LEN))) {
        process_data(data, readcount, sfinfo.samplerate, carrier,
freqdev);
        sf_write_double(outfile, data, 2*readcount);
```

```
    }

    sf_close(infile);
    sf_write_sync(outfile);
    sf_close(outfile);

    return 0;
}
```