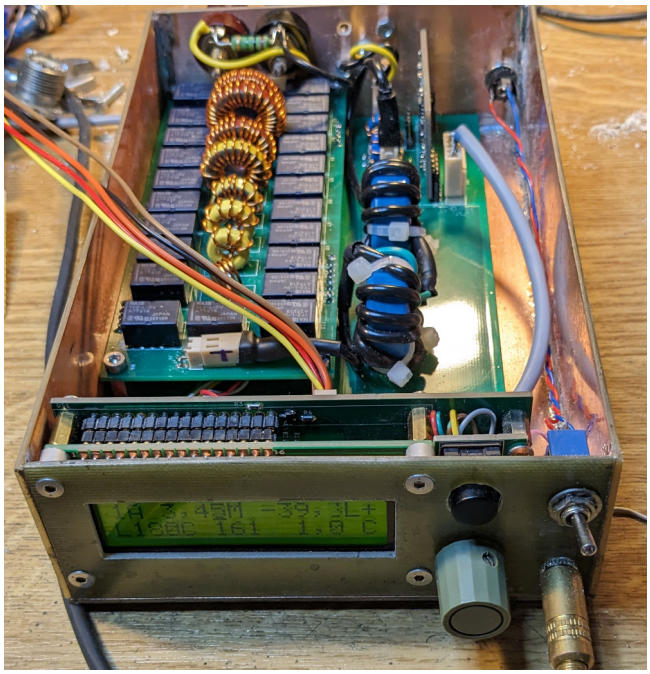
PicATUB QRP-Antennentuner
mit bistabilen Relais
SW Version 1.00

Andreas Lindenau DL4JAL

21. Juli 2023

Zusammenfassung

Der „PicATU20B“ ist ein Bastelprojekt von mir. Die FW im PIC habe ich neu geschrieben. Durch den Einsatz von bistabilen Relais, ist der Stromverbrauch sehr gering. Als Programmiersprache verwende ich Assembler. Assembler ist mir sehr geläufig. Ich habe eine große Sammlung an Sourcecode, den ich immer wieder verwenden kann.



Hier ist der PicATU20B zu sehen. Nach der Anpassung an die Antenne kann der PicATU20B aus geschaltet werden. Durch den Einsatz von bistabilen Relais bleibt die gefunden Einstellung auch im ausgeschalteten Zustand erhalten.

Inhaltsverzeichnis

Kapitel 1

Software

Die gesamte Software wurde in Assembler geschrieben. Der Hauptquelltext hat etwa 8600 Zeilen. Hinzu kommen noch die Quelltexte für die Pakete Gleitkommaberechnung, HF-Formeln und komplexe Berechnungen. Die Software ist über mehrere Jahre gewachsen und wird bei jedem ATU etwas abgewandelt. Die Grundfunktionen zur Suche der richtigen Antennenanpassung ist aber bei all meinen entwickelten Tuner gleich.

1.1 Display Inhalt

- Display Zeile 1
  - **1A** Antenne 1 ist ausgewählt (Speicherbereich 1 im ext.Eeprom)
  - **7,20c** Grequenz 7,20 MHz ist eingestellt. „c“ bedeutet das die Frequenz von der Fernbedienung gesendet wurde.
  - **−42,4** Sendepegel −42,4 dBm
- Display Zeile 2
  - **L 0** L hat den binären Wert von 0.
  - **C 96** C hat den binären Wert 96
  - **0,2** Return Loss beträgt 0,2 dB
- Display Zeile 1 und 2 ganz rechts
  - **LC-Variante** In beiden Zeilen sehen wir die eingestellte LC-Variante.

1.2 PowerON

Nach *PowerON* werden alle Baugruppen initialisiert. Die bistabilen Relais werden in die richtige Stellung gebracht. Alle Relais werden der Reihe nach eingeschaltet und anschließend wieder ausgeschaltet, da der geschaltet Zustand der Relais unbekannt ist. Dabei ist zu beachten das die 8 Relais des L-Gliedes in Ruhe alle eingeschaltet sind. Es müssen ja alle Induktivitäten kurz geschlossen sein wenn L den Wert=0 haben soll.

Anschließend wird die letzte gespeicherte Frequenz geladen und wenn vorhanden eine LC-Einstellung und die LC-Variante dazu. Der PicATU20B fängt nicht automatisch an zu tunen. Die Tune-Funktionen können nur über das *Menü* gestartet werden.

Wurde im SETUP in der Funktion *Timer I-Redukt.* der Timer aktiviert, reduziert sich die Stromaufnahme nach der angegebenen Zeit auf etwa 10mA. Die Hintergrundbeleuchtung des Displays wird abgeschaltet und die Versorgungsspannung der Messplatine. Ansonsten verbraucht der Tuner etwa 50mA.

Ist der Stromreduzierung noch nicht aktiv wird bei Senden im Display das SWR und die Sendeleistung angezeigt (die Messplatine hat noch ihre Versorgungsspannung).

Links: normaler Betriebszustand. Rechts: Ich habe den TX mit nur 15mW eingeschaltet. Das SWR und die Sendeleistung ist im Display zu sehen.

Power OFF

Wurde ein Anpassung an die Antenne erfolgreich beendet, können wird den PicATU20B auch ausschalten. **Alle Relais bleiben im geschalteten Zustand!** Das ist der große Vorteil der bistabilen Relais.

Frequenz-Erkennung

Der PicATU20B hat keine Frequenzmessung. Die momentan benutzte Frequenz wird im 10kHz mit dem Drehgeber eingestellt. Ist passend zu dieser Frequenz eine Abstimmeinstellung im externen Eeprom wird diese geladen und die Relais geschaltet. Mit der Taste im Drehgeber wird die Position des Kursors geändert, so das auch 1 MHz Schritte beim Verstellen der Frequenz möglich sind. So kann ganz schnell der gesamte Kurzwellen-Frequenzbereich eingestellt werden.

Externer Speicher für LC-Werte

Ich habe auf der Steuerplatine einen externen Eeprom-Speicher vorgesehen, der sehr groß ist (64kByte). Dadurch ist es möglich im gesamten Kurzwellenbereich aller 10kHz eine Abstimmeinstellung zu speichern. Da bei einem Frequenzbereich von 1,5 MHz bis 30 MHz immer noch viel Speicher frei ist, habe ich noch 4 weitere Frequenz-Bereiche 1,5 MHz bis 30 MHz vorgesehen. Es gibt also 5 Speicherbereiche von 1,5 MHz bis 30 MHz in denen aller 10 kHz eine Tuner-Einstellung gespeichert werden kann. Diese 5 Speicherbereiche bezeichne ich als

Antenne 1 bis 5. Pro Datensatz benötige ich 4 Byte. Für den Frequenzbereich 1,5 MHz bis 30 MHz benötige ich bei einem 10kHz-Raster 2850 Speicherplätze mit je 4 Byte. Das ergibt 11400 Byte. Folgende Speicherbereich ergeben sich:

- **Antenne 1** Speicherplatz 0 bis 11399
- **Antenne 2** Speicherplatz 11400 bis 22799
- **Antenne 3** Speicherplatz 22800 bis 34199
- **Antenne 4** Speicherplatz 34200 bis 45599
- **Antenne 5** Speicherplatz 45600 bis 56999

Der Speicherbereich 57000 bis 65535 bleibt frei. Da könnte man noch eine Sicherung des Pic-Eeproms einrichten. Vielleicht mache ich das später noch einmal.

1.3 Menü-Funktionen

Mit der *Einzeltaste* kommen wir in die *Menü-Funktionen*. Es folgen der Reihe nach die Funktionen.

Match, Match depp

In dieser Beschreibung habe ich beide Funktionen *(Match, Match deep)* zusammen gefasst. Die „Match-Funktion“ wurde von mir noch einmal grundlegend überarbeitet. Ich habe auf eine intelligente Flächensuche umgestellt. Die Fläche ergibt sich aus den Seitenlängen X(C:Kapazität) und Y(L:Induktivität). In unserem Fall ist X=C-Werte 0 bis 2047 und Y=L-Werte 0 bis 255. Das ist eine Fläche von 2047 x 255 Feldern, das ergibt multipliziert 512985 Felder. Das ist eine enorme Menge. Deshalb bin ich auf die Idee gekommen die erste Flächensuche mit exponentieller Schrittweite zu beginnen.

Grundmatch

Nachdem das Sendesignal erkannt ist beginnt die erste Such-Funktion *Grundmatch*. Das ist die Wichtigste. Die Schrittweite steigt quadratisch und es wird immer nur ein Relais der binären L-Kette und C-Kette eingeschaltet.

Das ist die erste Suche des SWR-Minimums. Die Suche beginnt mit C Wert= 0 und L Wert=0. Es folgt der nächste Schritt mit C-Wert=8 und L-Wert=1. Bei jedem weiteren Schritt verdoppelt sich der Wert. Es ergeben sich 9x9 also 81 L/C Kombinationen. Die L- und C-Werte kann man als eine quadratische Fläche betrachten mit 81 Feldern.

L-Werte 0, 1, 2, 4, 8, 16, 32, 64, 128

C-Werte 0, 8, 16, 32, 64, 128, 256, 512, 1024

	C	0,0pF	12,5pF	25pF	50pF	100pF	200pF	400pF	800pF	1600pF
	C-Glied	0	8	16	32	64	128	256	512	1024
L	L-Glied									
0,00uH	0									
0,125uH	1									
0,25uH	2									
0,5uH	4									
1uH	8									
2uH	16									
4uH	32									
8uH	64									
16uH	128									

Die Wertigkeit von L geht von 0 (0,00uH) bis 128 (16uH) und die Wertigkeit von C von 0 (0,0pF) bis 1024 (800pF). Die Suche auf der Fläche (gelb) beinhaltet 81 Felder. Bei jeder Kombination von L und C wird das Return-Loss gemessen. Das Feld mit dem höchsten Return-Loss merkt sich die Funktion und eine neue Suchfläche (hier grün) wird festgelegt. Die Schrittweite in der grünen Fläche wird neu gewählt, so das wieder etwa 64 Suchfelder entstehen.

Submatch

Submatch setzt die Suche fort mit reduzierter Suchweite in der quadratischen Fläche von L und C. Die Suche geht in unserem Beispiel C-Werte von 64 bis 256 und L-Werte von 8 bis 32 (oben grünes Feld). Die Schrittweite wird passend eingestellt. In unserem Beispiel C-Step=32 und L-Step=4. Mit der neue Schrittweite wird noch einmal die Suchfläche aktualisiert.

C-Wert	64	96	128	160	192	224	256
L-Wert							
8							
12							
16							
20							
24							
28							
32							

Jetzt ist die Schrittweite konstant linear. C Step 32 und L Step 4. Im Beispiel wurde neues RL maximum ist gefunden (Rot, L=16, C=96).

Die Funktion *Submatch* wird immer wieder wiederholt, mit reduzierter LC-Fläche und reduzierten Step.

C-Wert	64	80	96	112	128	144	160
L-Wert							
8							
10							
12							
14							
16							
18							
20							
22							
24							

Jetzt ist die Schrittweite halbiert.C Step 16 und L Step 2. Ein neues RL maximum ist gefunden (Rot, L=16, C=112).

Wiederholung der Funktion *Submatch* mit reduzierter LC-Fläche und reduzierten Step.

C-Wert	88	96	104	112	120	128	136
L-Wert							
12							
13							
14							
15							
16							
17							
18							
19							
20							

Die Schrittweite wird wieder halbiert. Bei C Step 8 und bei L Step 1. Ein neues RL maximum ist gefunden (Rot, L=15, C=104).

Die Schrittweite wird immer wieder halbiert und mit der Funktion *Submatch* neu gesucht. **Die Funktion *Submatch* wird beendet, wenn:**

- Ist das SWR kleiner 1,2 wird *Match* beendet. Der Match-Vorgang war erfolgreich. Die L/C Kombination wird gemerkt und verwendet und im externen Eeprom im 10kHz-Raster gespeichert.
- Ist das SWR 1,2 noch nicht erreicht aber beide Schrittweiten für L und C sind 1, wird *Match* ebenfalls beendet. Die beste L/C Kombination wird gemerkt und verwendet. Ist das SWR kleiner 1,5 wird diese Einstellung auch im externen Eeprom im 10kHz-Raster gespeichert.

So finde ich ganz schnell eine Impedanzanpassung. Es dauert im Idealfall etwa 10 bis 15 Sekunden.

Ist am Beginn der *Match-Funktion* im *Grundmatch* in der ersten L/C Variante das SWR $< 2{,}5$, wird in dieser L/C Variante weiter gesucht bis das SWR Minimum erreicht wurde. Ist im *Grundmatch* das SWR $> 2{,}5$, wird sofort in die nächste L/C Variante gewechselt, wieder beginnend mit *Grundmatch* usw...

Zuerst wird angenommen, dass die Antennenimpedanz > 50 Ohm ist, wie das meistens der Fall ist. **Folgende Suchreihenfolge habe ich programmiert:**

- **Variante1:** L/C Glied Impedanz größer 50Ohm, Grundmatch, Submatch....
- **Variante3:** C/L Glied Impedanz größer 50Ohm, Grundmatch, Submatch....
- **Variante2:** L/C Glied Impedanz kleiner 50Ohm, Grundmatch, Submatch....
- **Variante4:** C/L Glied Impedanz kleiner 50Ohm, Grundmatch, Submatch....
- Ist bei einer Varianten das beste SWR > 2,5, wird sofort in der nächsten Variante gesucht. Bei *Match deep* wird nicht bei SWR > 2,5 abgebrochen, sondern bis zum Ende weiter gesucht. Bis das SWR < 1,2 ist oder beide Schrittweiten 1 sind.
- Wird mit *Match* keine Anpassung gefunden sollte mit *Match deep* erneut eine Anpassung gesucht werden.
- Ein erneutes „Match deep“ bricht nicht mehr bei „SWR > 2,5“ ab, sondern sucht immer bis zum Ende (bis beide Schrittweiten 1 sind). So ist die Suche nach dem besten SWR intensiver. Es wird fast immer ein vertretbares SWR gefunden.
- Ist das SWR < 1,5, werden die Werte von L, C und die L/C Variante im Speicher des „10kHz-Segmentes“ abgelegt.

  Zusätzlich kann man noch die gefunden Einstellung mit dem Befehl *(B)and* für das ganze KW-Band abspeichern (in alle 10kHz-Segmente des Bandes).

ReMatch

Grundsätzlich sollte man beim Wechsel in einem anderen 10khz-Bereich noch einmal *ReMatch* aufrufen und Nachstimmen. *ReMatch* sucht so lange bis die Schrittweite bei L und C gleich 1 ist.

Auch hier gilt wieder, ist das SWR < 1,2 wird die gefundene Einstellung im 10 KHz.Segment des externen Eeprom abgespeichert. Oft werden aber noch kleiner SWR-Werte erreicht.

ReMatch deep

Die Funktion *ReMatch deep* habe ich für schwierige Fälle noch programmiert. Vorher wird manuell die LC-Variante gesetzt und anschließend *ReMatch deep* gestartet. *ReMatch deep* beginnt mit der Suche von ganz vorn *Grundmatch, Submatch* usw..

Auch bei den LC-Varianten „nur L“ und „nur C“ ist *ReMatch deep* zu verwenden.

Band Seg.save

Wird das erste mal eine Anpassung im KW-Band gesucht ist es ratsam die gefundene Anpassung für alle 10kHz-Segment des KW-Bandes zu speichern. An Hand der eingestellten Frequenz weiß die SW welches KW-Band gemeint ist.

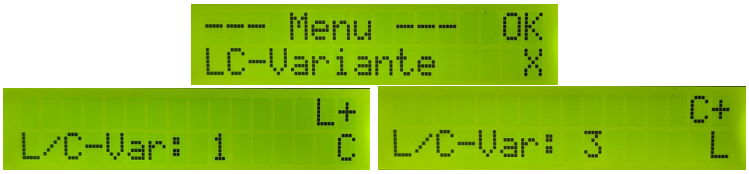
Steht die Frequenz im Tuner als Beispiel auf 3,76 MHz, weiß die SW das ist das 80m-Band und speichert die gefundene Einstellung in allen 10kHz-Segmenten von 3,4MHz bis 3,9MHz ab. Also noch 100kHz zusätzlich unterhalb und oberhalb des Bandes.

Das hat den Vorteil, wir haben im gesamten 80m-Band eine gültige Anpassung die mit *ReMatch*, *ReMatch4x4* oder *ReMatch8x8* nachgestimmt werden kann.

10kHz Seg.save

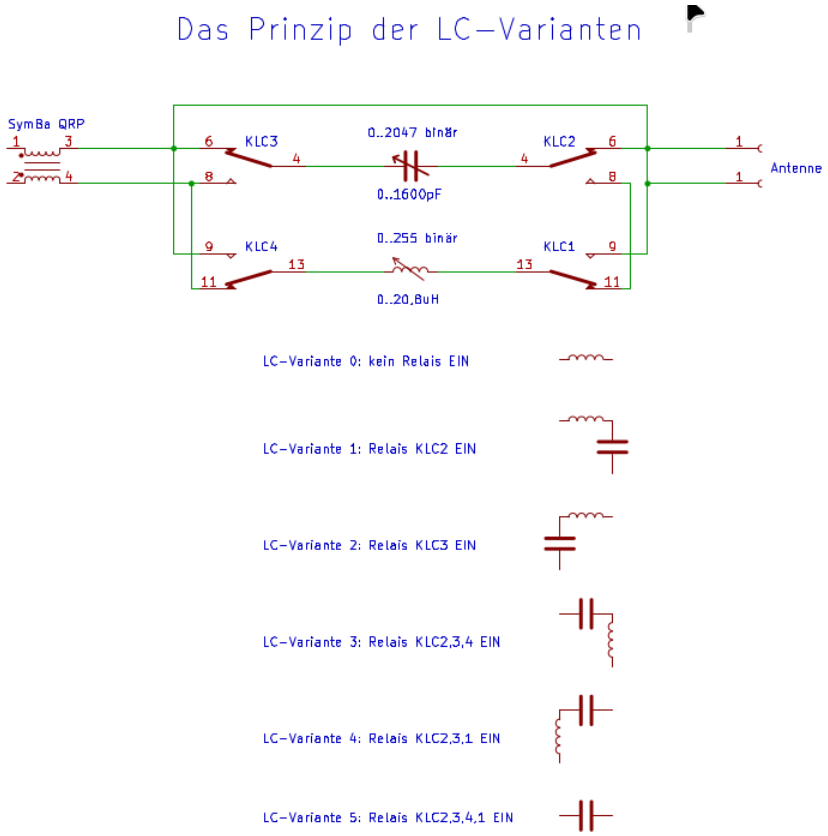
Diese Funktion speichert die LCV-Einstellung im 10kHz-Segment ab, auch wenn das SWR sehr schlecht sein sollte. Es gibt Situationen da ist das nötig.

LC-Variante



Hier habe ich L und C untereinander vertauscht. Das ergibt manchmal eine bessere Anpassung.

In dieser Funktion kann die LC-Variante manuell geändert werden.



Ant.Impedanz

Das ist nur ein Beispiel.

Wurde eine gute Anpassung gefunden, berechnet diese Funktion mit der eingestellten Frequenz, der LC-Variante und den Werten von L und C die Antennenimpedanz. Der Zahlenwert der Impedanz stimmt natürlich nicht ganz genau, aber eine gute Orientierung ist das auf alle Fälle.

ReMatch 4x4

Ist die Abweichung der Anpassung sehr gering kann man ganz schnell mit dieser *ReMatch-Funktion* das SWR verbessern.

ReMatch 8x8

Auch diese *ReMatch-Funktion* verbessert eventuell das SWR. Es wird ein etwas größerer Bereich abgesucht.

1.4 SETUP-Funktionen

Ein *langer Tastendruck* der Einzeltaste startet das SETUP-Menü.

Abbruch!

Das bedarf keiner Erklärung. Abbruch!

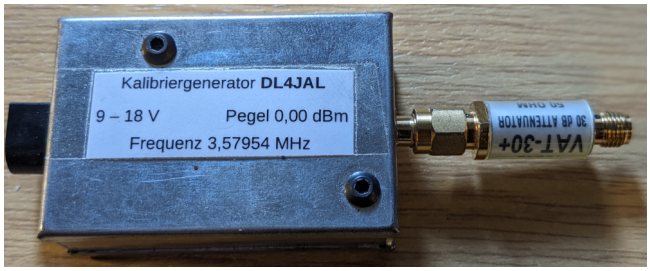
MK.Return Loss

In der PDF der Hardware und Aufbau des PicATU20B wurde diese Funktion schon behandelt. Ist der Richtkoppler aufgebaut und die Messplatine gesteckt, wissen wir nicht ob der Der Richtkoppler mit der Messplatine korrekt funktioniert. Im Kapitel Inbetriebnahme könnt ihr das nachlesen.

Kalib. Vorlauf

Der PicATU20B funktioniert auch ohne ein spezielle Kalibrierung des Vorlaufs und Rücklauf. Möchte man aber genauere Angaben der Sendeleistung kommt man um eine Kalibrierung der beiden AD8307 nicht herum.

Für die Kalibrierung benötigen wir eine möglichst genauen 0,0dBm-Generator und ein Dämpfungsglied 30dB.

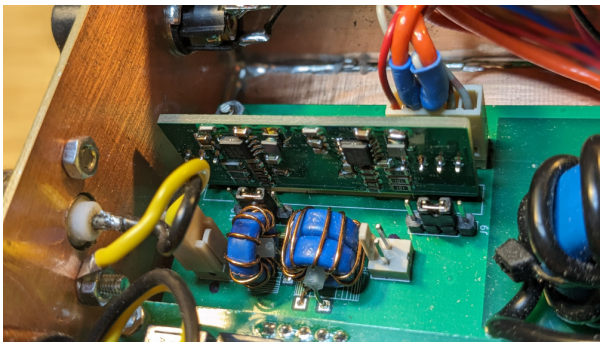


Mein 0,0dBm Generator mit aufgeschraubten Dämpfungsglied 30dB von *Mini-Circuits*.

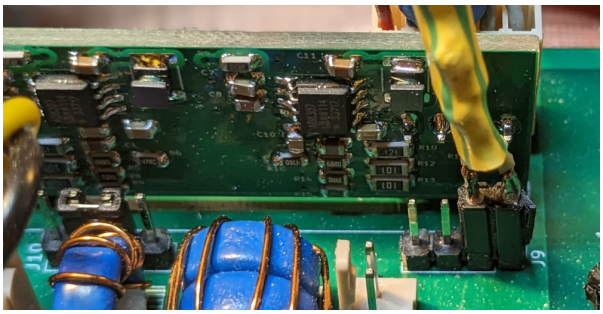
Ich habe mehrere Möglichkeiten einen genauen Pegel zu erzeugen. Mein Homemade-Doppel-DDS-VFO geht auch sehr genau im Pegel und Frequenz.

Doppel-VFO -140dBm bis etwa +3dBm in 0,1dB Schritten, IM3 -90dBc.
Frequenz 0,1Hz bis 150MHz.
Leistungsausgang Doppel-Pegel bis 15 Watt PEP IM3 -60dBc.
Frequenz 1,6 MHz bis 55 MHz.

- Zuerst wird die Brücke zwischen Richtkoppler und Messplatine geöffnet und das Kabel zum Generator angesteckt.



Wir sehen die beiden Brückenstecker. Links ist für den Rücklauf und rechts für den Vorlauf.



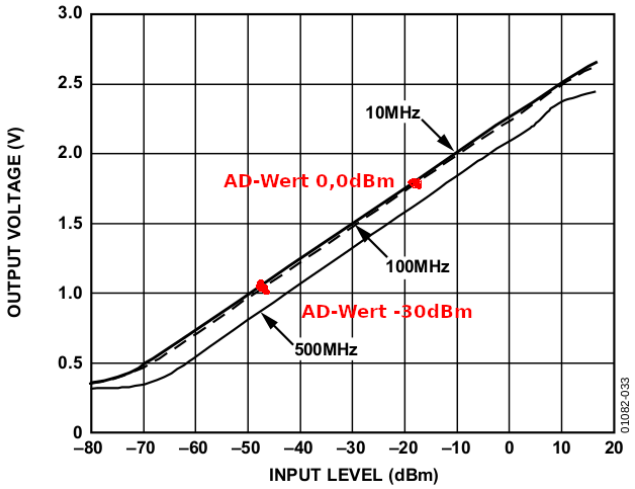
Das Kabel zum Generator wird angesteckt.

- Wir starten die Kalibrierfunktion und sehen im Display den Messwert des A/D Wandlers bei 0,0dBm und anschließende bei −30dBm.

Drücken wir *OK* werden die beiden Kalibrierergebnisse ausgerechnet und abgespeichert.

Die beiden Messergebnisse von „11004“ bei $0,0dBm$ und 6055 bei $-30,0dBm$ erscheinen sehr hoch. Der A/D-Wandler im PIC18F26K22 hat nur 10Bit Wandlerbreite. 10Bit ergeben ein maximale Zahl von 1023. Ich habe aber in der Messfunktion der Software eine Messschleife mit 32 Messungen hintereinander. Die Werte werden addiert. 32 mal 1023 ergibt eine maximale Zahl von 32736.

Wer sich für die Berechnungen interessiert kann hier weiter lesen. Zuerst betrachten wir die Messfunktion im Datenblatt des AD8307.



Ich habe das Diagramm aus dem Datenblatt heraus kopiert und die 2 Messpunkte für 0,0dBm und -30,0dBm eingezeichnet. Bei beiden Messpunkten müssen wir 18dB abziehen. Die Eingangsbeschaltung auf der Messplatine hat 18dB Dämpfung, die wir berücksichtigen müssen.

Im Vorfeld eine Erklärung der verwendeten Bezeichnungen in den Formeln:

Bezeichner	Erklärung
dBmpegel1	Messpegel1 in dBm
dBmpegel2	Messpegel2 in dBm
adcwertpunkt1	Werte des AD Wandlers im PIC beim Pegel 1
adcwertpunkt2	Werte des AD Wandlers im PIC beim Pegel 2
mkx	Wert X der linearen Funktion
mky	Wert Y der linearen Funktion

Formel 1: $mkx = \frac{dbmpegel1 - dbmpegel2}{adcwertpunkt1 - adcwertpunkt2}$

Formel 2: $mky = (adcwertpunkt1 * mkx * -1) + dbmpegel1$

Da der erste Pegel 0dBm beträgt, vereinfacht sich die Formel etwas.

Formel 1: $mkx = \frac{-30dBm}{adcwertpunkt1 - adcwertpunkt2}$

Formel 2: $mky = adcwertpunkt1 * mkx * -1$

Mit diesen beiden Werten aus der Berechnung der Kalibrierung können wir aus der Integer-Zahl des ADC-Wertes (0..32736) den Pegel in dBm berechnen.

Wir setzen die Zahlen der Kalibrierung in die Formeln ein:

$$mkx = \frac{-30dBm}{11004-6055} = 0,006062$$

$$mky = 11004 * 0,006062 * -1 = -66,706$$

Mit dem beiden Konstanden *mkx* und *mky* können wir nach jeder A/D-Wandlung jetzt mit einem Schritt den dBm-Wert berechnen. Die Formel für die Berechnung des HF-Pegels in *dBm* ist einfach und lautet:

$$Pegel(dBm) = ADCWert * mkx + mky$$

Zur Kontrolle setzen wir die beiden A/D-Werte für $0,0dBm$ und $-30,0dBm$ in die Formel ein. Zuerst 11004 für $0,0dBm$

$$Pegel(dBm) = 11004*0,006062+-66,706 = 0,000248dBm$$

und 6055 für $-30dBm$

$$Pegel(dBm) = 6055*0,006062+-66,706 = -30,00059dBm$$

Die Ergebnisse weichen nach der 4 Stelle hinter dem Komma ab, da ich Mkx und Mky etwas gerundet habe.

So einfach ist es mit „Mkx“ und „Mky“ aus dem Wert des ADC-Wandlers den Pegel in dBm zu errechnen.

Ich hoffe ihr habt meine Berechnungen verstanden.

Kalib. Ruecklauf

Die Kalibrierung des Rücklaufes ist wie oben beschrieben der gleiche Ablauf.

Timer I-Redukt.

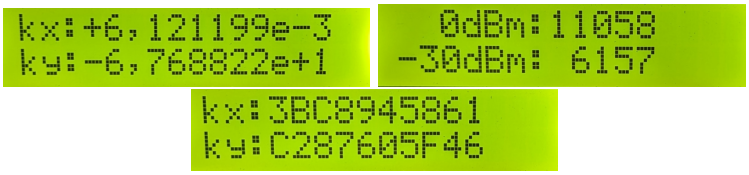
Der PicATU20B verbraucht etwa 50mA Strom bei 9Volt. Der Tuner kann also mit einer kleinen 9 Volt Blockbatterie betrieben werden. Damit nicht ständig 50mA fließen, kann nach einer vorgegebenen Zeit die Hintergrundbeleuchtung und die Spannung der Messplatine abgeschaltet werden. Der Stromverbrauch reduziert sich dadurch auf etwa 10mA. Der PIC bleibt noch aktiv, aber die online Anzeige im Display beim Senden geht nicht mehr, da an der Messplatine die Spannung fehlt.

Verschiedene Einstellungen.

View MKxy Vor

Anzeige der Kalibrierwerte vom Vorlauf. Es werden der Reihe nach angezeigt:

- mkx und mky als Float-Zahl in exponentieller Darstellung
- Die beiden A/D-Werte von $0,0dBm$ und $-30dBm$. Die beiden Zahlen sollte man sich aufschreiben für das *manuelle Kalibrieren*. Da werden nur die beiden Zahlen eingetragen und man ist fertig ohne einen HF-Pegel Einspeisung.
- Als Abschluss die beiden Konstanden mkx und mky als HEX-Zahl. Das ist nur für mich interessant als Programmierer.



View MKxy Rueck

Anzeige der Kalibrierwerte vom Rücklauf. Siehe vorherige Beschreibung.

Relais-Test

Der Relaistest wurde schon bei der Inbetriebnahme beschrieben. Es können alle Relais auf ihre Funktion getestet werden.

KL1 bis KL8 Relais.

KC1 bis KC11 Relais.

KLC1 bis KLC4 Relais.

Kal.manuell Vor

Das ist die Funktion, die ich schon kurz bei *View MKxy Vor* erwähnt habe. Es werden nur die beiden Integer-Zahlen eingetragen für $0,0dBm$ und $-30dBm$ eingetragen. Daraus berechnet die Software die Konstanden *mkx* und *mky*.

Kal.man. Rueck

Diese Funktion ist wie *Kal.manuell Vor*.

View dBm+A/D,V+R

Anzeige der dBm und A/D-Werte für Vorlauf und Rücklauf. Diese Funktion dient der Kontrolle der Messungen.

3 verschieden Anzeigen beim Experimentieren.

SET dBm Minimum

Hier wird der minimale dBm-Wert eingestellt. Oberhalb dieses Pegels beginnt der Tuner zu arbeiten *Match* usw...

SET dBm Maximum

Das ist der maximale Pegel beim Tunen. Oberhalb dieses Pegels werden keine Relais mehr geschaltet. Das ist ein Schutz der Relais bei schalten.

Antenne Clear

Der Speicherbereich im Eeprom für die entsprechende Antennennummer (Speicherbereich) wird gelöscht. Alles LCV-Werte gehen verloren.

Antenne Copy

Kopieren von einer Antennen-Nummer zu einer beliebigen anderen Antennen-Nummer. Alle LCV-Werte werden kopiert.

Antenne Select

Die Auswahl der Antenne (Speicherbereich 1..5) erfolgt hier.

Kapitel 2

Schlusswort

Dieses Projekt darf nicht kommerziell vermarktet oder genutzt werden. Alle Rechte liegen bei DL4JAL (Andreas Lindenau). Ich wünsche viel Spaß beim Basteln.

vy 73 Andreas DL4JAL

✉ DL4JAL@t-online.de
☞ www.dl4jal.eu