Werner Nitsche
DL7MWN

Zwischenbericht LIF5000 vom 19.10.2007

1. Einleitung

Seit dem letzten Zwischenbericht ist diesmal viel Zeit vergangen, aber es ist auch viel geschehen und der LIF5000 ist nun fast fertig. Zuletzt wurde die Software geschrieben und der LO mit dem DDS-Baustein in Betrieb gesetzt. Jetzt fehlt nur noch ein Gesamtschaltplan und die Geräteverdrahtung. Die existiert zwar schon für Testzwecke, aber sie ist großzügig gehalten und die vielen langen Kabel passen nicht in das Gehäuse.

Die Inbetriebnahme des DDS-Bausteins war für mich ein sehr schwieriges Kapitel. Da ich bisher noch nie mit so einem Baustein gearbeitet habe, welcher über ein „exposed paddle“ verfügte, machte ich ungefähr alle Fehler, welche möglich waren. Manchmal wäre es hilfreich, wenn man das Datenblatt eines neuen Bausteins zunächst erst einmal gründlich durchliest und sich Gedanken über das macht, was da geschrieben steht. Aber ich war mir über die Bedeutung des „exposed paddle“ nicht im Klaren und ich habe es einfach ignoriert. Diesen Fehler musste ich sehr büßen und nun werde ich wohl künftig modernen Bausteinen mit einem „exposed paddle“ viel Respekt entgegen bringen.

In diesem Zwischenbericht möchte ich mich aber hauptsächlich mit der Programmierung des DDS-Bausteins in der Software beschäftigen.

2. Die Software

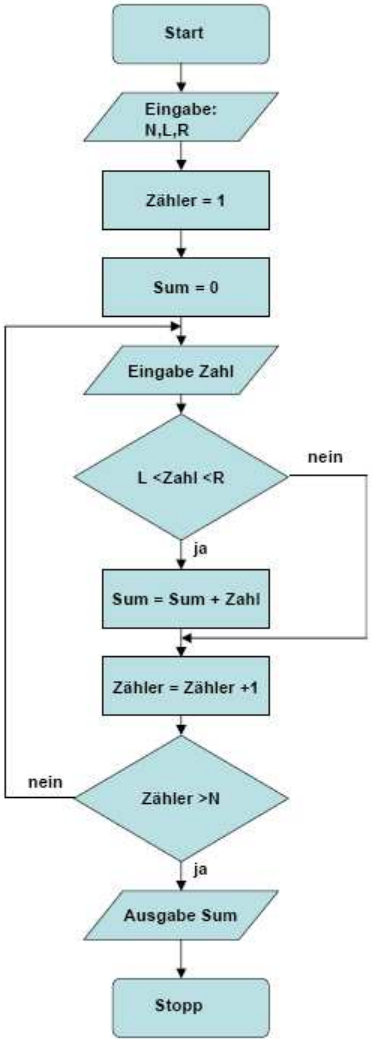
2.1 Progammier-Stil

Schreibt man eine Software, so macht man sich zuvor Gedanken über die zu erledigenden Aufgaben. Bei einer umfangreicheren Software stellt man sich den Programmablauf erst einmal schriftlich oder zeichnerisch dar und verfeinert ihn dann Stück für Stück, bis alle Details vollständig sind. Eine derartige Vorgehensweise mag zunächst etwas umständlich erscheinen, aber sie garantiert, dass die Software danach auch das macht, was man von ihr erwartet. Meistens geht es mit einer derartigen Vorgehensweise am schnellsten, weil nur wenige Fehler entstehen, welche man dann auch relativ einfach beheben kann.

2.1.1 Darstellungsform

Um Programmabläufe darzustellen, bieten sich verschiedene Möglichkeiten an. Ich habe die Software für den LIF5000 unter Zuhilfenahme des Pseudocodes designed. Der Pseudocode entspricht fast vollständig dem Programmcode und er kann durch diesen dargestellt werden. Zur Veranschaulichung möchte ich hier in einem kleinen Beispielen die verschiedenen Darstellungsformen erklären.

2.1.1.1 Ablaufdiagramm oder Flussdiagramm

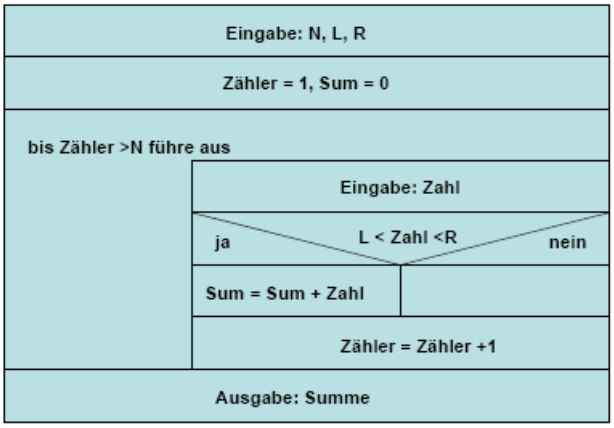


Das Flussdiagramm erlaubt jede beliebige Art der Programmierung. So wird auch die berühmte „Goto-Programmierung“ unterstützt, welche der Programmiersprache BASIC in der Vergangenheit das schlechte Image eingebracht hat.

Durch die Goto-Programmierung kann man z.B. in eine Subroutine springen und von dort ohne sie zu beenden wieder zurück ins Hauptprogramm. Wenn man das mehrmals macht, dann läuft der Stack über und der Rechner stürzt ab. Das Problem dabei ist, dass der Absturz erst kommt, wenn der

Stack überläuft. Und das kann in einem ganz anderen Programmteil passieren. In so einem Fall ist es sehr schwer, die wahre Fehlerursache zu finden. Goto-Befehle sollte man heute nicht mehr, oder nur unter ganz strengen Vorschriften verwenden.

2.1.1.2 Struktogramm oder Nassi Schneiderman Diagram



Das Struktogramm schreibt sehr strenge Regeln vor und es ist längst nicht alles erlaubt, was möglich wäre. Es dürfen auch keine Goto-Sprünge vorkommen. Aber das Programm ist überschaubarer und lässt sich dementsprechend leichter testen. Auch Fehler lassen sich leichter finden. Der große Nachteil des Struktogramms ist die umständliche Dokumentation. Dazu muss man viel Zeichnen und wenn man eine Korrektur macht, muss man wieder viel zeichnen. Man verwendet das Struktogramm nur noch selten in Verbindung mit einer softwareunterstützten Dokumentation, welche dem Anwender die viele Zeichenarbeit abnimmt. Ich kenne niemand, der das Struktogramm im Hobbybereich als Dokumentationsart verwendet.

2.1.1.1 Pseudocode

```
Pseudocode:
        read N, L, R
        Zähler = 1
        Sum = 0
        WHILE Zähler <= N
                read Zahl
                IF L< Zahl <R THEN
                        Sum = Sum + Zahl
                ÊND IF
                Zähler = Zähler + 1
        WIHLE END
        PRINT Sum
Pseudocode END
```

Der Pseudocode unterliegt absolut genau den gleichen Regeln wie das Struktogramm und ist daher auch genau so gut geeignet, um ein sauberes Programm zu schreiben. Der große Vorteil liegt darin, dass man

den Pseudocode mit einem normalen Texteditor schreiben kann, und dass er sich gleichermaßen für Assemblerprogramme wie für Hochsprachen eignet. Nur Hochsprachen bestehen schon fast aus denselben Elementen und wenn man den Pseudocode leicht abwandelt, dann hat man schon die Programmiersprache. So passt das alles ganz gut zusammen.

2.2 Der Programmablauf

Um alle nötigen Funktionen im LIF5000 anzubieten, musste ich eine Software mit folgenden Funktionen schreiben:

Initialisierung
Hauptprogramm
Display-Steuerung
Bedienerführung
Frequenzeingabe
Datenspeicher für 20 Stationstasten
Berechnung des Datenstroms zum DDS-Baustein
Übertragung des Datenstroms zum DDS-Baustein
Analoge und digitale S-Meter-Anzeige
Steuerung des Preselektors via I2C-Bus
Interrupt für Tasten
Timerinterrupt
RS232-Schnittstellen für Debug und Fernsteuerung

Wer sich dafür näher interessiert, der kann ja in die PDF-Datei „Steuerprogramm für LIF5000 V0.08“ auf meiner Homepage schauen. Im Gegensatz zu der vorläufigen Software habe ich diese anders formatiert. Sie hat nicht mehr die schöne Seiteneinteilung, dafür ist sie besser lesbar und braucht längst nicht soviel Speicher.

2.2.1 Das Programm

Eigentlich wollte ich ja alle Funktionen in der Software gründlich beschreiben. Nachdem ich damit angefangen hatte, bemerkte ich, dass es immer mehr wurde und ich doch noch immer am Anfang war. Nun habe ich eingesehen, dass das ein ganzes Buch würde und dass das bestimmt für den normalen Leser langweilig wäre. Und so habe ich mich entschieden, nur das Herzstück der Software, die Programmierung

des DDS-Bausteins AD9951, etwas näher zu beleuchten. Den Rest der Software kann man sich ja im Source-Listing selbst anschauen.

2.2.2 Parametrierung des DDS-Bausteins berechnen und übertragen

Für mich war das was ganz Neues und ich musste mich erst mit der Problematik vertraut machen. Ich habe im Internet gesucht, ob jemand so etwas schon mal gemacht hat und ob es ein veröffentlichtes Beispiel zur Programmierung gab. Sehr viele Stellen zu diesem Thema habe ich im Internet nicht gefunden. Meistens wurde so etwas für den PIC-Rechner und in Assembler-Sprache veröffentlicht. Nur „unser" Christian Hirt hat zufällig genau denselben Rechner und dieselbe Programmiersprache wie ich verwendet. Also lag es nahe, dass ich mal geschaut habe, wie er das Problem löste. Aber zunächst habe ich das nicht verstanden. So habe ich versucht, Informationen zum Algorithmus der Berechnung zu suchen. Da habe ich von OM Reinhold eine E-Mail bekommen, in welcher er mir die Ableitung genau beschrieb und in Form von Beispielen erklärte. Dafür möchte ich Reinhold an dieser Stelle noch einmal danken. Nachdem ich die Erklärung durchgearbeitet hatte, war ich in der Lage, mir jede beliebige Frequenz mit Mathcad zu berechnen. Da ist dann nur noch das Problem mit der Rechengenauigkeit von 64 Bit Auflösung übrig geblieben. Das Problem habe ich dann in einem zweiten Schritt mit Christians Veröffentlichung gelöst. Und weil die Beschreibung von Reinhold so wichtig und elementar ist, möchte ich die Ausführungen von Reinhold hier in leicht geänderter (angepasster) Form darstellen.

2.2.2.1 Wie wird der Datenstrom zum DDS-Baustein berechnet?

<u>a) Annahme Taktfrequenz ist exakt 400MHz !</u>

1 Hz in "Bits" entspricht genau (2^32 / 400MHz) = 10,73741824 Bits/Hz

<u>b) In HEX Format (32Bit Länge) umwandeln</u>
und zwar der Ganzzahlanteil mittels den ersten 8Bit (MSB's) sowie den Nachkommateil mittels den restlichen 24 Bits (der Windows Taschenrechner ist ideal dafür !)

Ganzzahl 10 entspricht: 0Ah

Nachkommateil: (0,73741824 * 2^24) = 12371825,09 entspricht: BC C7 71h

Womit 1Hz == 0A BC C7 71h

Dieser Wert muss jetzt als Konstante fest im µC einprogrammiert werden !

<u>c) Alles was der µC selber noch an Arithmetik durchführen muss</u>
ist die gewünschte Ausgangsfrequenz in Hz mit dem ermittelten, fest kodierten, 1Hz Hex-Wert multiplizieren und schon hat man das DDS Frequenzwort!

Bsp.: 14,120500MHz Empfangsfrequenz (Träger) in einem DC-RX

(14120500 * 0A BC C7 71h) = 09 09 80 B2 2D 98 F4h

Offensichtlich sind das jetzt deutlich mehr als 32 Bits (wie Du schon angemerkt hast), doch davon brauchen wir NUR die ersten 32 Bits, die man in einer passenden Variablen zur Weiterverarbeitung abspeichert und bekommen:

09 09 80 B2h ... was als Frequenzwort (nach dem Instruktionsbyte) dem DDS-Chip zu übergeben ist !

Testen wir mal ob's auch stimmt:

(09 09 80 B2h * 400MHz) / 2^32 = 14,12049998MHz

Freqenzfehler = 20mHz (min. Schrittweite eines 400MHz AD9951 ~ 93mHz) !!

d) Zum Korrekturwert

Die Taktfrequenz weicht mit sehr hoher Wahrscheinlichkeit von den exakten 400MHz ab. Deshalb muss der 1Hz HEX-Wert geringfügig nach oben (Taktfrequenz < 400MHz) oder unten (Taktfrequenz > 400MHz) korrigiert werden. Das Ganze baut man bspw. in einem Menüpunkt der Steuerung ein, damit der Nutzer die Korrektur später selbstständig durchführen kann.

Dazu gibt man z.B. 10MHz aus und schließt einen Frequenzzähler am DDS-Ausgang an, der jetzt natürlich nicht genau 10MHz misst. Nun vergrößern oder verkleinern wir den 1Hz HEX-Wert um 1 solange bis der Frequenzzähler genau die gewünschten 10MHz anzeigt. Der neu ermittelte HEX-Wert wird nun anstelle des Alten, fest im µC-EEPROM, gespeichert.

Ohne Frequenzzähler: RX auf 10.001000MHz einstellen und 10MHz WWV in LSB empfangen. Soundkarte anschließen, die 1kHz mittels FFT (hohe FFT-Punktzahl) darstellen. Nun den 1Hz HEX-Wert solange verändern bis die FFT genau 1kHz anzeigt um danach die neue 1Hz HEX-Zahl ins EEPROM zu übernehmen.

Nachdem ich nun grundsätzlich verstanden hatte, wie sich das Frequenz-Tuning-Word für den DDS-Baustein berechnet, habe ich mir eine Gleichung in Mathcad geschrieben, um schon einmal für jede Frequenz die den benötigten Datenstrom berechnen zu können. Das schaut dann so aus:

Eingabe DDS Ausgangsfrequenz: **Quarzfrequenz:**

Fout := 14120500 **Hz** Quarz := 400000000 **Hz**

$$Inp\_Val := \frac{Fout \cdot 2^{32}}{Quarz}$$

Inp_Val = 151617714.257920000000000

Inp_Val = 90980b2h

2.2.2.1 Und wie kann man das berechnen, wenn man nur einen 8-Bit Rechner hat?

Nun war da aber noch das Problem mit der hohen Rechengenauigkeit, welche von den meisten Programmiersprachen nicht unterstützt wird. Aber da ist mir wieder eingefallen, dass Christian das Problem ja schon gelöst hat. Also habe ich versucht, den Algorithmus von Christian mit Bleistift und Papier nachzurechnen, bis ich ihn endlich verstanden habe. Irgendwann hat das Ergebnis auch gestimmt und ich bin dazu übergegangen, mir meine eigene Software für diesen Algorithmus von Christians Software abzuleiten. Viele Möglichkeiten, diese anders zu machen, bestehen ja nicht und so kann man teilweise noch die Original-Software von Christian erkennen. Bei mir schaut es nun so aus:

```
Sub Calc_ad9951()
    'In dieser Subroutine wird das Frequenz Tunig Word für den DDS-Baustein AD9951
    'berechnet und die Daten an die Subroutine "Set_AD9951" übergeben. Die Tabelle
    'zur Berechnung seht am Ende der Software. Das verlangt BASIC so.
    'Input:      Frequenz     Frequenz
    'Output:    DDS-Baustein programmiert

    'Berechnen des FTW (Frequenz Tunig Word)
    Frequenz = Frequenz + Zf                    'ZF aufaddieren
    Frequenz_1 = Frequenz * 4                   'Frequenz_1 = (Empfangsfrequenz + ZF) * 4
    Lu1 = Lookup(0 , Mult)                      'Bits pro Hz aus der Tabelle laden (erster Wert)
    F_tw = Lu1 * Frequenz_1                     'Teilberechnung Frequenz Tunig Word

    For I = 1 To 8 Step 1
        Lu1 = Lookup(i , Mult)                  'Bits pro Hz aus der Tabelle laden
        Y1 = Lu1 * Frequenz_1                   'Arbeitsfrequenz * Bits pro Hz (Zwischenergebnis)
        Lud = Lookup(i , Divi)                  'Dekade laden
        Z1 = Y1 / Lud
        F_tw = F_tw + Z1                        'Frequnz Tunig Word zusammensetzen
        'Call Tst_prt                           'hier könnten Zwischenergebnisse ausgegeben werden!
    Next I

    Call Set_ad9951                             'Daten an den DDS-Baustein übertragen

End Sub

Mult:
     Data 10 , 7 , 3 , 7 , 4 , 1 , 8 , 2 , 4
Divi:
     Data 1& , 10& , 100& , 1000& , 10000& , 100000 , 1000000 , 10000000 , 100000000
```

Eigentlich ist das garnicht so schwierig. Entscheidend für diesen Algorithmus ist die Verwendung von Tabellen, welche umfangreiche Berechnungen ersetzen und eine schnelle Abarbeitung der Berechnung ermöglichen. Diese Berechnung wurde von Christians Ausführungen abgeleitet. Da es mir nicht gelungen ist, diese Routine besser zu beschreiben, habe ich mich dazu entschlossen, die Erklärung von Christian direkt zu übernehmen. Ich habe nur kleine Anpassungen vorgenommen. Das Original kann man auf Christians Homepage nachlesen.

Christian schreibt zu seinem Algorithmus:

a) Deklariere Werte als LONG. Es wird keine Fliesskomma-Arithmetik verwendet!
(Long hat einen positiven Wertebereich bis 2147483647)

b) Hole ersten Wert aus LookUp-Tabelle MULT (= 10)

c) Multipliziere damit die gewünschte Frequenz (ftwx = Ganzzahl, keine Kommastellen!)

d) Nun wird eine Schleife 8x durchlaufen

```
' FTW Berechnung im Long Format
F 'Frequenz in Hz
lu1 = LookUp(hole ersten Wert aus MULT-Tabelle)
ftwx = lu1*F
For i = 0 To 7
ftwx = ftwx + (LookUp(i+1,MULT)*F / LookUp(i, DIV))
Next i
```

e) Zerlege das berechnete ftwx in 4 Byte

f) Schreibe es in den AD9951

Was wurde hier getan?
Formel laut AD9951 DataSheet (wenn Taktfrequenz = 400 Mhz):
F = FTW X 400 MHz / 2^32

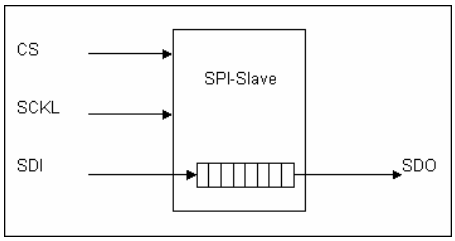
Umgeformt:
FTW = F x 2^32 / 400x10^6 = F x 10.73741824

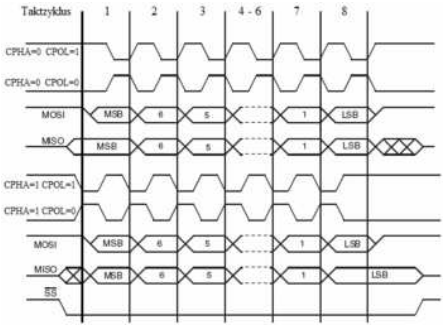
In der Berechnung nun zerlegt in Zehnerstellen:
Ftwx1 = 10 x F
Ftwx2 = ftwx1 + 7 x F / 10
Ftwx3 = ftwx2 + 3 x F / 100
…
Ftw = ftwx8 + 4 x F / 100 000 000

Vorteil:
Es wird nur mit dem LONG (einem langen Integerformat) gerechnet.

Nachteil:
Geringer Fehler (ich habe mir noch nicht die Mühe gemacht den maximal möglichen Fehler auszurechnen)

2.2.2.2 Das berechnete Frequenz Tuning Word an den DDS-Baustein übertragen





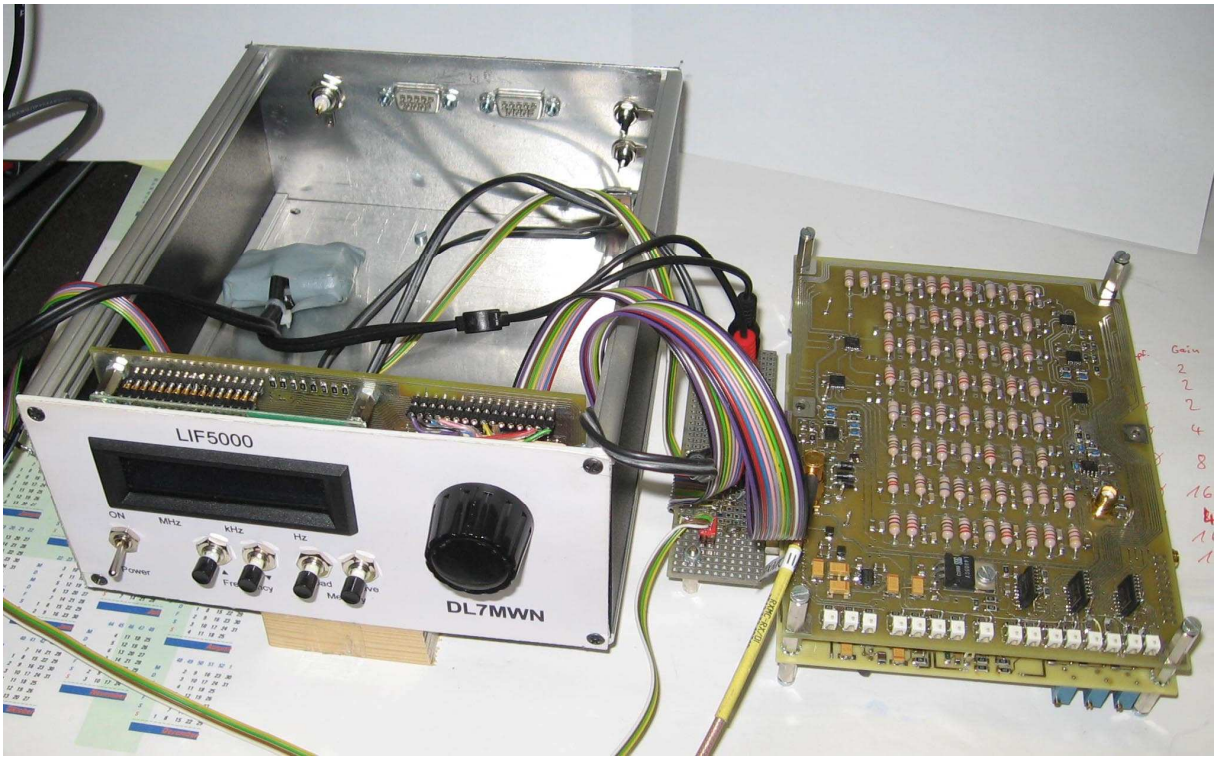
Nachdem das Frequenz Tunig Word nun berechnet ist, muss es noch über die SPI-Schnittstelle übertragen werden. Dazu hat das BASCOM-BASIC eigene Treiber, welche aber bestimmten Regeln unterliegen. Nachdem mir diese Regeln nicht gefallen haben, programmierte ich mir diese Treiber selbst. Vorteil meiner Arbeit ist es, dass man die von mir geschriebene Software leicht an andere Programmiersprachen (auch Assembler) und andere Rechner anpassen kann.

Und bei mir schaut es nun so aus:

```
Sub Set_ad9951
    'In dieser Subroutine werden dem DDS-Baustein der Inhalt des Control Function Register 2 und
    'das Frequenz Tuning (CFR4) übertragen.
    'Input:      F_tw    Frequenz Tuning Word
    'Output:     Daten im DDS-Baustein
```

```
    'Init Datenübertragung zum DDS-Baustein
    Reset_dds = 1 : Strobe_dds = 0 : Data_dds = 0 : Clk_dds = 0
    Tx_data = 0 : Tx_bit = 0

    'Daten an  das Control Function Register 2 (CFR2) übertragen
    Print : Print "CFR2"
    Tx_data = &B00000001                    'Adresse: Control Function Register 2 (CF2)
    Call Transmit_dds                       'Daten übertragen
    Tx_data = &B00000000
    Call Transmit_dds
    Tx_data = &B00000000
    Call Transmit_dds
    Tx_data = &B00101110                    'RefMult; VCO; Charge Pump
    Call Transmit_dds

    Strobe_dds = 1                          'Datenübernahme zum DDS-Baustein = 1
    Waitus 20
    Strobe_dds = 0                          'Datenübernahme zum DDS-Baustein = 0

    'Daten an  das Control Function Register 4 (CFR4) übertragen
    Print : Print "CFR4"
    Tx_data = &B00000100                    'Adresse: Control Function Register 4 (CFR4)
    Call Transmit_dds                       'Daten übertragen

    Hw = Highw(f_tw)                        'Frequenz Tuning Word übernehmen und in 2 Words zerlegen
    Lw = F_tw

    Tx_data = High(hw)                      'MSB High-Byte übertragen
    Call Transmit_dds

    Tx_data = Low(hw)                       'MSB Low-Byte übertragen
    Call Transmit_dds

    Tx_data = High(lw)                      'LSB High-Byte übertragen
    Call Transmit_dds

    Tx_data = Low(lw)                       'LSB Low-Byte übertragen
    Call Transmit_dds

    Strobe_dds = 1                          'Datenübernahme zum DDS-Baustein = 1
    Waitus 20
    Strobe_dds = 0                          'Datenübernahme zum DDS-Baustein = 0

End Sub


Sub Transmit_dds()
    'In dieser Subroutine wird ein Byte an den DDS-Baustein übertragen. Zum Test werden die selben
    'Daten auch noch über die serielle Schnittstelle ausgegeben.
    'Input:     Tx_data
    'Output:    Daten werden zu DDS-Baustein geschickt

    For I = 7 To 0 Step -1                  '8 Bit ausgeben, MSB zuerst
        Hz1 = 2 ^ I                         'richtiges Bit zur Übertragung auswählen
        Tx_bit = Tx_data And Hz1
        If Tx_bit <> 0 Then
            Print "1";                      'Datenbit an der seriellen Schnittstelle ausgeben
            Data_dds = 1                    'Datenbit setzen
       Else
            Print "0";                      'Datenbit an der seriellen Schnittstelle ausgeben
           Data_dds = 0                     'Datenbit rücksetzen
       End If
       Waitus 20                            'warten
       Clk_dds = 1                          'Clock generieren
       Waitus 20                            'warten
       Clk_dds = 0                          'Clock zurücksetzen
    Next I

End Sub
```

3. Was funktioniert nun schon am LIF5000



Eigentlich funktioniert nun schon alles am LIF5000, was ich mir so vorgestellt habe. Es gibt noch kleine Probleme mit dem LCD-Kontrast. Der Arbeitspunkt des LCD-Displays ist sehr steil und geringfügige Änderungen an der Spannung führen dazu, dass man das LCD-Display nicht mehr ablesen kann. Da muss ich einen Vorwiderstand einbauen, um den Strom besser zu steuern. Das ist aber kein grundsätzliches Problem für den LIF5000. Auch setzt der DDS-Baustein bei höheren Frequenzen oberhalb von 20MHz RX-Frequenz öfter mal aus. Vermutlich liegt das am Strombedarf und den Spannungen am DDS-Baustein. Das muss ich noch näher untersuchen.

Ich habe nun schon viele Stunden mit meinem LIF5000 auf Kurzwelle aber auch auf Mittelwelle und Langwelle gelauscht. Leider habe ich hier keine gute Antenne. Im Frequenzbereich <2Mhz habe ich einen Störpegel, welchen der LIF5000 mit S6 anzeigt. Diese Störungen ermöglichen mir einen einwandfreien und sauberen Empfang nur bei starken Sendern.

Auch einen Vergleich mit meinem WinRadio habe ich unter absolut gleichen Bedingungen schon gemacht. Die Ergebnisse fallen eher unterschiedlich aus. Grundsätzlich kann man sagen, dass mein LIF5000 die gleichen Sender wie der Winradio empfangen kann. Manche Sender lassen sich genau so gut empfangen, aber andere sind etwas mehr gestört wie beim WinRadio. Möglicherweise kommen da noch irgendwelche Mischprodukte bis in die ZF.

Natürlich habe ich auch schon stundenlang DRM-Sender in bester Tonqualität bei einem SNR >20dB empfangen. Das ist schon ein besonderer Genuss auf Kurzwelle.

Der LIF5000 ist ein Experimentier-Empfänger, welcher mir gezeigt hat, dass man die Spiegelfrequenz mit einem Polyphase-Netzwerk auch noch nachträglich ausfiltern kann. Über den gesamten Frequenzbereich von 7kHz bis 17kHz ZF ist die Spiegelfrequenzunterdrückung >40dB. Das hat bei mir in allen Fällen aus-

gereicht. Aber leider braucht man für das Polyphase-Netzwerk zusätzliche Verstärkung, was im LIF5000 zu etwas erhöhtem Rauschen führt.

4. Schlusswort

Dieser Arbeitsabschnitt hat sich diesmal weniger mit HF-Technik, sondern mehr mit Rechner und Software beschäftigt. In modernen Empfängerkonzepten geht nichts mehr ohne einen eigenen Rechner. Das Schreiben der benötigten Software ist mir eigentlich relativ leicht gefallen. Viele Probleme hat mir der DDS-Baustein bereitet, weil er einfach nicht funktionieren wollte. In so einem Fall ist es zunächst immer sehr schwer zu sagen, wo der Fehler wirklich liegt. Ich habe den Fehler in meiner Software gesucht und viele Testroutinen eingebaut, um die einwandfreie Funktion der Software nachzuweisen. Aber vielleicht war das ja auch ein Vorteil. So weis ich nun wirklich, wie die Programmierung des DDS-Bausteins funktionieren muss und wie man sie prüft. Nachdem ich mich dann endlich entschlossen habe, einen neuen DDS-Baustein einzubauen und das „exposed paddle“ auch richtig anzuschließen, ging es wieder schnell weiter.

Aber mein Ziel, einen funktionieren Empfänger zu bauen, der den gesamten Bereich von 50kHz bis 30MHz empfangen kann, ohne irgendwelche Bandschalter umzustellen, habe ich mit dem LIF5000 erreicht. Dabei stellte sich heraus, dass die Verwendung eines Preselektors unbedingt empfehlenswert ist. Die Kombination des digitalen Vorreglers auf dem Preselektor, welcher per Software gesteuert wird, und der linearen AGC in den ZF-Stufen, hat sich gut bewährt. Dieses Prinzip möchte ich bei meinen nächsten Arbeiten noch ausbauen.

Werner Nitsche
DL7MWN

An dieser Stelle möchte ich mich bei allen bedanken, die mir in diesem schwierigen Abschnitt bei meiner Arbeit mit Informationen und „Mut zusprechen“ ganz wesentlich geholfen haben. Besonders möchte ich aber OM Reinhold und Christian für ihre Hilfe beim Programmieren des DDS-Bausteins danken.

Natürlich freue ich mich auch wieder auf sachliche Kritik und Anregungen von Euch. Habt Ihr Erfahrung in der einen oder anderen Sache und würdet Ihr etwas grundsätzlich anders machen? Und warum? Das interessiert mich natürlich. Also schreibt mir entweder im QRP-Forum oder direkt an meine E-Mail-Adresse, wie bisher.

Meine E-Mail-Adresse lautet:
werner.nitsche@gmx.de

Euer Werner, DL7MWN

PS:
Und noch etwas zum Schluss. Ich habe die URL meiner Homepage gewechselt. Zunächst habe ich mit meiner Homepage beim DARC begonnen. Aber da stehen mir nur 10MB Speicher zur Verfügung. Diese Grenze habe ich schon längst überschritten. Da ich seit Ende letzten Jahres aber auch noch eine eigene Homepage mit 100MB Speicher besitze, bin ich jetzt einfach umgezogen. Bitte ändert die Einträge in Euren Lesezeichen und/oder Favoriten sowie die Links auf meine Homepage, soweit ihr welche auf Eurer Homepage angelegt habt.

Meine neue URL lautet:

http://dl7mwn.de/