Werner Nitsche
DL7MWN

Zwischenbericht Steuerrechner HSDR-4512 vom 28.02.2008

Bild 1 HSDR-4512

1. Vorwort

Nun ist neben dem mechanischen Aufbau die erste Etappe in der Elektronik geschafft. Der Steuerrechner ist fertig designed, das Layout erstellt und bereits beim Leiterplattenhersteller. Im Gegensatz zum LIF5000 werde ich die Leiterplatten für den HSDR-4512 mit Lötstopplack versehen lassen, weil ich die freien Flächen zwischen den Bauteilen mit Masse geflutet habe. Dadurch erreicht man eine stabilere, durchgehende Massefläche. Der Lötstopplack kostet zwar zusätzlich Geld, aber man kann dann die kleinen SMD-Bauteile, trotz Massefläche zwischen den Leiterbahnen, besser löten. Verwendet man keinen Lötstopplack, so besteht bei Verwendung gefluteter Massen die Gefahr, dass unter jedem Bauteil ein Kurzschluss zur Massefläche entstehen kann. Es ist schwierig, die Position eines solchen Kurzschlusses messtechnisch zu lokalisieren. So muss man Bauteil für Bauteil wieder auslöten, bis man ihn gefunden hat.

Grundsätzlich kann man sagen, dass die Steuerrechner vom LIF5000 und HSDR-4512 ganz ähnlich sind. Aber es gibt ein paar Änderungen für den HSDR-4512.

Werner Nitsche
DL7MWN

Dieser Zwischenbericht beschäftigt sich hauptsächlich mit den Details, welche im LIF5000 nicht behandelt wurden. Wer sich für alles interessiert, kann sich ja den „Zwischenbericht Steuerrechner für den LIF5000 vom 08.06.2007“ durchzulesen. Der ist ja ebenfalls auf meier Homepage zu finden.

2. Leiterplatte des Steuerrechners

Der Schaltplan zum Steuerrechner ist seit ein paar Tagen auf meiner Homepage zu finden. Das Layout ist ebenfalls fertig und in 2 Wochen kommt die fertige Leiterplatte. Die Bestückungsseite der Leiterplatte sieht man in Bild 2.

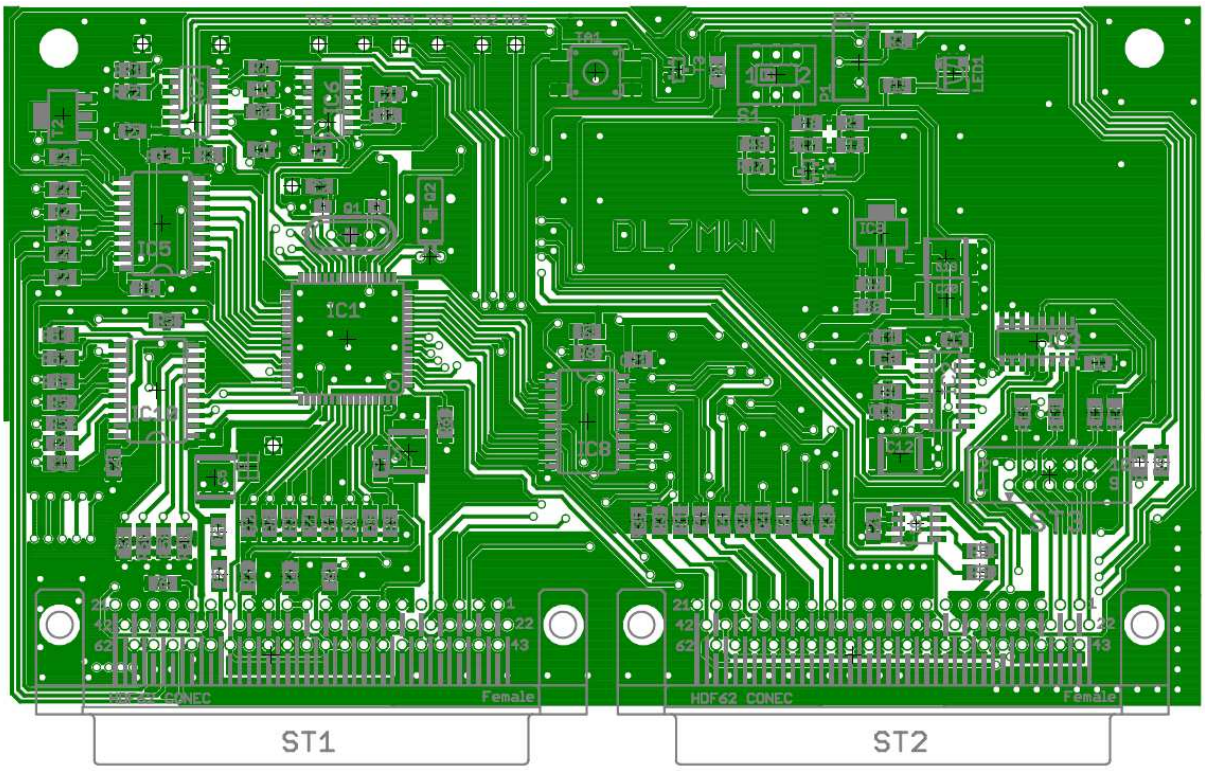


Bild 2 Bestückungsseite des Steuerrechners HSDR-4512

Weil man wegen der gefluteten Masse einige Details nicht so gut erkennen kann, habe ich die Leiterplatte in Bild 3 noch einmal ohne Masseflutung dargestellt. Schaut man sich nun die beiden Steuerrechner vom LIF5000 und vom HSDR-4512 an, dann sieht man, dass die Grundfläche des neuen Steuerrechners etwas größer ist. Auch ist der neue Steuerrechner mit Steckern versehen, welche das Einschieben der Platine in ein Einschubsystem ermöglichen. Man kann den Steuerrechner also ganz leicht rausziehen und in ein anderes Gerät einbauen. Weil auf dem neuen Steuerrechner noch genügend Platz frei ist, kann man die Schaltung auf dieser Platine noch erweitern. Das war auf dem alten Steuerrechner nicht mehr möglich. Durch die beidseitige, massive Masseflutung wird dieses Board HF-mäßig stabil. Es soll möglichst keine Störungen erzeugen und abstrahlen. Mit dem alten Steuerrechner konnte ich damit schon sehr gute Ergebnisse erzielen. Das liegt auch daran, dass der ATMEGA128 alle nötigen Baugruppen wie

EPROM und RAM-Speicher schon integriert hat und dafür kein Bus-System mehr auf der Leiterplatte benötigt. Bus-Systeme stören meistens sehr stark.

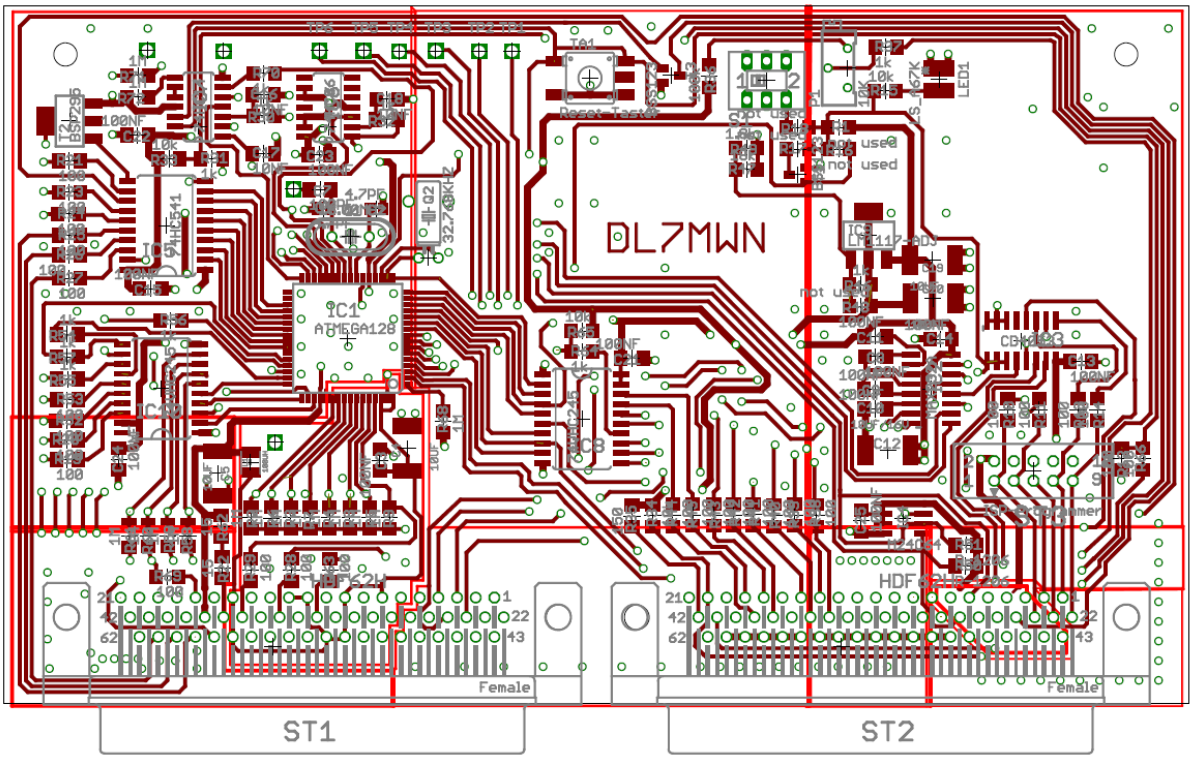


Bild 3 Bestückungsseite des Steuerrechners HSDR-4512 ohne Masseflutung

3. Baugruppen des Steuerrechners

Der Baustein „IC1“ auf der linken Seite mit den vielen Anschlüssen, welche in quadratischer Form angeordnet sind, ist der ATMEGA128. Er ist natürlich das Herzstück dieser Platine, aber mit ihm alleine kann man nichts anfangen, und so befindet sich um ihn herum noch einiges an zusätzlicher Elektronik.

3.1 Stromversorgung

Im Gegensatz zum alten Steuerrechner ist auf diesem Board kein DC/DC-Converter. Dieses Board benötigt von außen eine +5V-Stromversorgung. Es ist geplant, die Stromversorgung im HSDR-4512 zentral zu lösen. So lassen sich Störungen durch die DC/DC-Converter auch zentral unterdrücken. Das führt zu besseren Ergebnissen, als wenn in jedem Eck des Gerätes ein eigener DC/DC-Converter verwendet wird.

3.2 ISP-Programmier-Interface

Der Steuerrechner lässt sich über ein ISP-Programmier-Interface (ISP = In-System-Programming) programmieren. So funktioniert es auch im LIF5000. Für dieses Interface gibt es die passenden Kabel mit eingebautem Programmer im Stecker fertig zu kaufen. Wer sich darüber näher informieren will, der kann sich die Beschreibung des LIF5000 Steuerrechners durchlesen. Da bin ich auf diesen Punkt etwas ausführlicher eingegangen.

Downloading Adapter

Bild 4

Ribbon cable

Bild 5

3.3 Das I2C-Bus-Interface

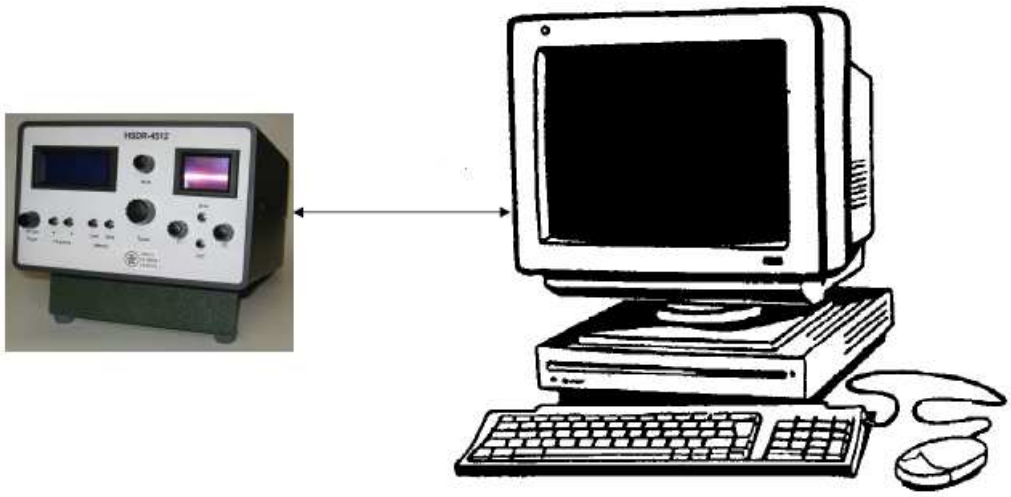
Der Steuerrechner verfügt über ein I2C-Bus-Interface. Über ihn kann er andere Boards steuern. So z.B. werden über den I2C-Bus die verschiedenen Filter auf dem Preselektor geschaltet. Auch das 6 dB-Dämpfungsglied auf dem Preselektor wird über den I2C-Bus vom Steuerrechner aus gesteuert.

Den I2C-Bus habe ich im LIF5000 sehr gründlich beschrieben. Deshalb werde ich auf ihn hier nicht näher eingehen. Wer sich für den I2C-Bus interessiert, der kann das in meinem Zwischenbericht vom 08.06.2007 nachlesen.

3.4 RS232-Interface für Debugzwecke

Der Steuerrechner ist mit zwei RS232-Schnittstellen ausgestattet. Über eine dieser Schnittstellen kann man den Programmablauf des Steuerrechners testen. Dazu benötigt man nur noch einen PC mit Hyperterminal und einer COM-Schnittstelle. Das kann aber auch über einen USB-Konverter erfolgen, wenn man über einen modernen PC verfügt, welcher keine COM-Schnittstelle mehr hat. Aber die Software teste ich und ich habe noch einen alten PC mit COM-Schnittstelle.

Die zweite RS232-Schnittstelle kann z.B. zur Kommunikation zwischen dem DSP und dem Steuerrechner dienen. Aber dazu gibt es mehr Möglichkeiten und so ist das im Moment noch nicht entschieden.



Kommunikation zwischen PC und dem HSDR-4512

Bild 6 Der HSDR-4512 wird im Debug-Mode betrieben

3.5 SPI-Interface

Der Steuerrechner vom HSDR-4512 verfügt über ein SPI-Interface mit 3 CS-Leitungen. Je nachdem, wie man die CS-Leitungen codiert, kann man bis zu 7 SPI-Slaves adressieren. Da der ATMEGA128 noch in der alten 5-Volt-Technik arbeitet, aber alle neuen digitalen Boards mit 3,3-Volt oder niedriger versorgt werden, müssen die Eingangs- und Ausgangspegel des Steuerrechners, welche zu diesen modernen Boards führen von 5 Volt auf 3,3 Volt umgesetzt werden. Das erledigt ein IC, welcher mit 3,3 Volt arbeitet und 5 Volt am Eingang toleriert. Das funktioniert in beide Richtungen ganz gut.

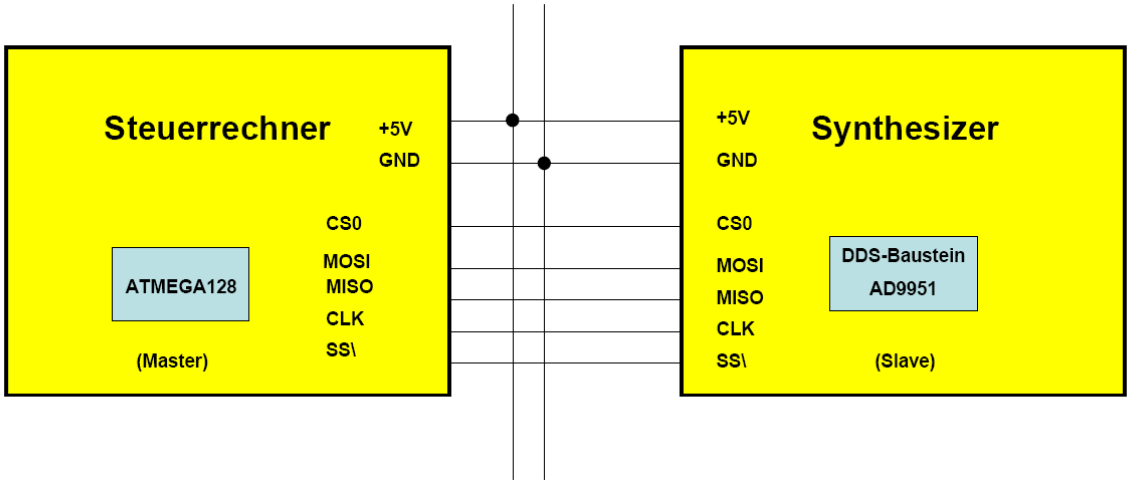


Bild 7 Struktur des SPI-Interfaces mit 1 CS-Leitung für einen Slave

3.5.1 Was ist ein SPI-Interface

SPI bedeutet Serial Peripheral Interface (serielle Peripherie Schnittstelle) und beschreibt einen synchronen seriellen Bus. Dieser Bus wurde aber niemals in einen Industrie-Standard überführt. Auch wird kein Softwareprotokoll vorgegeben, sodass sich jeder Benutzer sein eigenes „Handshaking“ programmieren muss. Eigentlich funktioniert ein SPI-Interface wie zwei Schieberegister. Zunächst nimmt ein Schieberegister mit 8 parallelen Eingängen ein Byte auf und schiebt es seriell zu einem zweiten Schieberegister, welches die Daten seriell aufnimmt und dann wieder parallel abgibt. Das SPI-Interface wurde nie mit Patenten belegt, wodurch es sich als lizenzfreies Interface schnell in unzähligen Implementierungen verbreitet hat.

Das SPI-Interface lässt sich leicht realisieren, weil der Hardwareaufwand sehr gering und billig ist. Es wird eine „Slave Select Leitung SS\“, eine „Clockleitung SCK“ und 2 Signalleitungen „MOSI und MISO“ benötigt. Optional kann man mehrere SPI-Slaves durch Chip-Select-Leitungen „CS0 bis CSxx“ anwählen. Dann kann ein SPI-Master fast beliebig viele SPI-Slaves steuern.

Daten werden gleichzeitig in beide Richtungen übertragen. Beim SPI-Interface handelt es sich also auch noch um eine Voll-Duplex-Schnittstelle. Die Übertragungsgeschwindigkeit ist hoch. Es lassen sich bis zu 1 Million Bits in der Sekunde übertragen. Die Polarität der einzelnen Leitungen ist nicht vorgegeben und kann je nach Anwendung beliebig festgelegt werden.

Das SPI-Interface ist ein einfaches, schnelles und preiswertes Dateninterface für einen Master und mehrere Slaves, das keinen Lizenzrechten unterliegt.

3.5.2 Wie funktioniert ein SPI-Interface

Ein SPI-Interface besteht immer aus einem Master und einem oder mehreren Slaves. Es handelt sich also um ein Master-Slave Interface. Der Master stellt das Taktsignal bereit und bestimmt dadurch die Übertragungsgeschwindigkeit. Der Slave schiebt die Daten mit dem Taktsignal in ein Register. Dabei ist ihm die Übertragungsgeschwindigkeit egal. Theoretisch könnte die Übertragungsgeschwindigkeit auch während der Übertragung schwanken oder gar kurz aussetzen.

Zwei Datenleitungen übertragen gleichzeitig Daten vom Master zum Slave und vom Slave zum Master. Diese Datenleitungen sind MISO (Master-In Slave-Out, Dateneingang Master) und MOSI (Master-Out Slave-In, Datenausgang Master).

Jeder Slave verfügt über einen „Slave-Select-Eingang“. Über diesen Eingang bestimmt der Master, wann Daten zwischen dem Master und dem Slave ausgetauscht werden sollen. Über CS (Chip Select-Leitung) wird ein bestimmter Slave selektiert. Betreibt ein SPI-Master nur einen Slave, so wird die CS-Leitung nicht benötigt. Sind mehrere Slaves

angeschlossen, dann gehen alle nicht selektierten Slaves in den TriState. Das heißt, ihre Ausgänge werden hochohmig und stören somit die Datenübertragung zwischen dem Master und dem angewählten Slave nicht. Es werden immer 8 Datenbits übertragen. Ist die Datenübertragung fertig, so werden im Statusregister automatisch Flags gesetzt. Auch kann ein Interrupt ausgelöst werden, um das Übertragungsende zu signalisieren. Das ist bei den verschiedenen Bausteinen und je nach Initialisierung unterschiedlich.

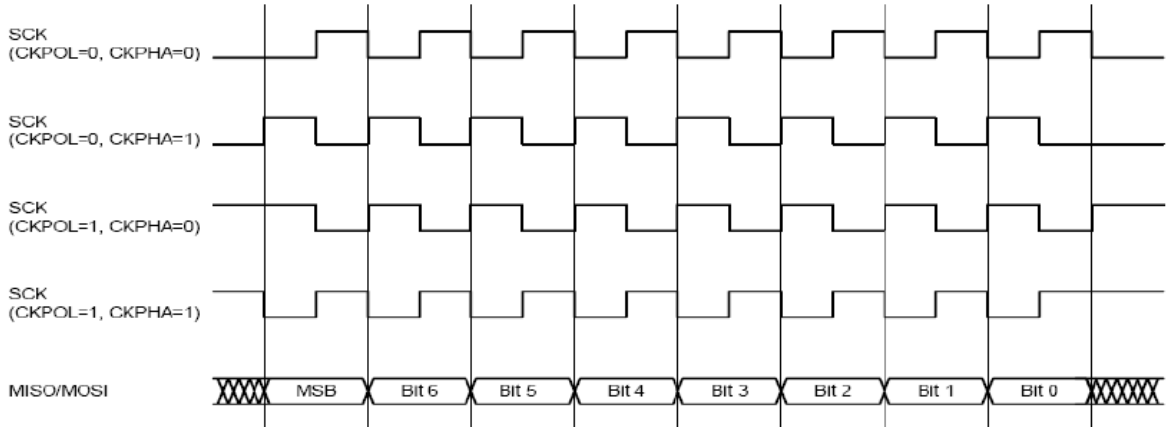


Bild 8 Das Timing eines SPI-Interface (ist nicht genormt und kann auch anders ausschauen)

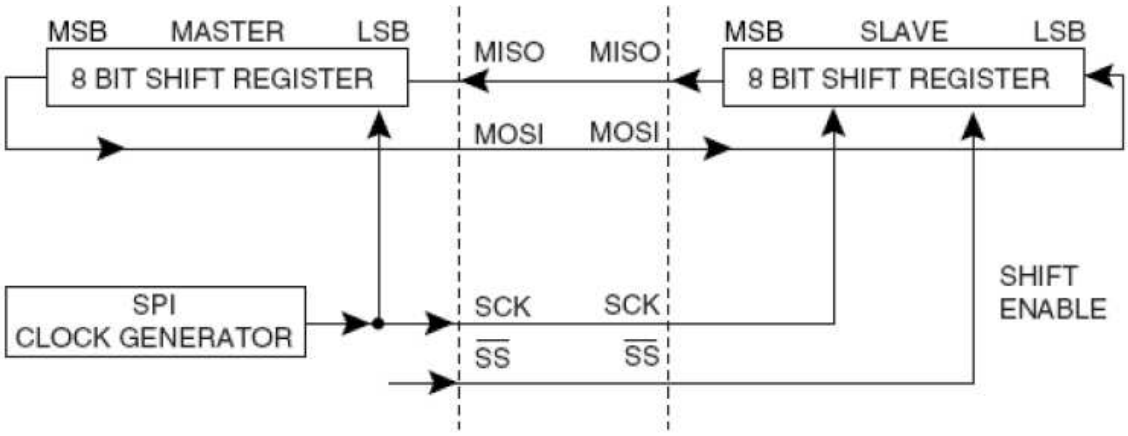


Bild 9 Hardware-Aufbau eines SPI-Interfaces

3.6 Bedienelemente auf der Frontplatte

Beim LIF5000 habe ich alle Bedienelemente mit einer Interruptsteuerung versehen. Wurde eine Taste gedrückt, so hat diese durch einen zusätzlichen Baustein einen Interrupt ausgelöst und der Steuerrechner konnte die gedrückte Taste sofort einlesen. Auch den Drehschalter zum Einstellen der Empfangsfrequenz hatte ich mit einer eigenen Interruptsteuerung versehen. Wer sich dafür interessiert, wie das funktioniert hat, der kann das beim Steuerrechner des LIF5000 nachlesen.

Eigentlich funktionierte das im LIF5000 sehr gut. Aber trotzdem war ich damit unzufrieden und habe mich beim HSDR-4512 für ein anderes Prinzip entschieden. Das Problem beim alten Konzept waren die vielen Signalleitungen zwischen Steuerrechner und der Frontplatte. Der HSDR-4512 verfügt aber zusätzlich noch über einen Mode-Schalter, welcher 11 Stellungen hat. Das wären alleine schon 11 zusätzliche Signalleitungen und dann noch die Signalleitungen zu den Tasten. Und dann müsste jede Schalterstellung durch zusätzliche ICs so verknüpft werden, dass bei jeder Änderung ein Interrupt ausgelöst wird. Das alles wäre zu aufwendig geworden und der ATMEGA128 hat gar nicht so viele, freie Porteingänge. Eine mögliche Lösung wäre gewesen, auch dafür einen I2C-Bus zur Porterweiterung einzusetzen. Aber dann hätte ich für die Frontplatte auch noch eine eigene Platine benötigt. Mit so einem Gedanken konnte ich mich nicht anfreunden. Dann ist mir eingefallen, dass ich in der Vergangenheit schon ganze Keyboards mit nur 3 Leitungen (+5V, GND und einer Signalleitung) zuverlässig eingelesen habe. Und das war dann auch die Lösung dieses Problems.

Bild 10 Frontplatte des HSDR-4512

3.7 Einlesen vieler Schalter mit einer Signalleitung

Als Überschrift klingt das schon mal ganz gut. Das ist es auch und es funktioniert wirklich ohne Hokus Pokus. Wir werden uns nun einmal anschauen, wie das aufgebaut ist.

3.7.1 Analoge Übertragung vieler Zustände auf einer Leitung

Die Überschrift verrät es schon. Wir brauchen dazu die gute, alte Analogtechnik. Wenn man einen Spannungsteiler, bestehend aus 12 gleichen Widerständen hat, dann kann man mit einem Drehschalter zwischen den Widerständen 11 verschiedene Spannungen abgreifen. Gibt man nun die Ausgangsspannung des Drehschalters auf einen A/D-Wandler und digitalisiert die Spannung, so entsteht für jede Schalterstellung ein anderer, digitaler Messwert. Ordnet man diesen Messwert nun im Rechner einer Schalterstellung zu, so kann man in unserem Beispiel 11 verschiedene Schalterstellungen eindeutig unterscheiden. Weil der gesamte Analogteil gewissen Störungen und Typenstreuungen unterliegt (Rauschen), ordnet man jeder Schalterstellung nicht nur einen einzigen digitalen Wert zu, sondern einen ganzen Wertebereich. Damit wird man unempfindlich

gegenüber Bauteiletoleranzen und Bauteilealterung. Je weniger Stellungen man unterscheiden muss, desto größer kann dieser Wertebereich sein und umso größer wird die Auslesesicherheit. In der Praxis kann man mit diesem Verfahren ganz gut 20 Schalterstellungen unterscheiden. Damit setzt man eine Genauigkeit von 5% voraus. Mit den modernen 1%igen Metallschichtwiderständen lässt sich das gut realisieren. Will man wirklich noch mehr Schalterstellungen unterscheiden, dann gibt es diese Widerstände auch noch mit einer Genauigkeit von 0,1%.

3.7.2 Die Hardware-Realisierung

Wenn man ohnehin einen ATMEG128 zur Verfügung hat, stehen einem 8 Analogeingänge zur Verfügung. Damit könnte man also 8 x 20 Schalterstellungen zuverlässig einlesen. Dazu benötigt man dann nur 8 Signalleitungen, eine Referenzspannung und einen GND-Anschluss. Die Referenzspannung gewinnt man am besten direkt aus der Versorgungsspannung des ATMEGA128. Ist diese Spannung nicht ganz genau, so macht das nichts aus, weil ja der Spannungsteiler am Drehschalter an derselben Spannung angeschlossen ist, wie der A/D-Wandler selbst. Somit wird der absolute Fehler dieser Spannung im A/D-Wandler kompensiert.

Im HSDR-4512 versorgen wir also den Widerstands-Spannungsteiler mit den +5 Volt des Steuerrechners. Ein Stufenschalter wird an den Spannungsteiler angeschlossen. Der Stufenschalter im HSDR-4512 hat 11 Stellungen. Also brauchen wir dazu einen Spannungsteiler mit 12 Widerständen. Der gemeinsame Anschluss des Stufenschalters wird dann über eine Signalleitung an einen Analogeingang des ATMEGA128 geführt. Mehr braucht man zur Realisierung dieser Methode nicht.

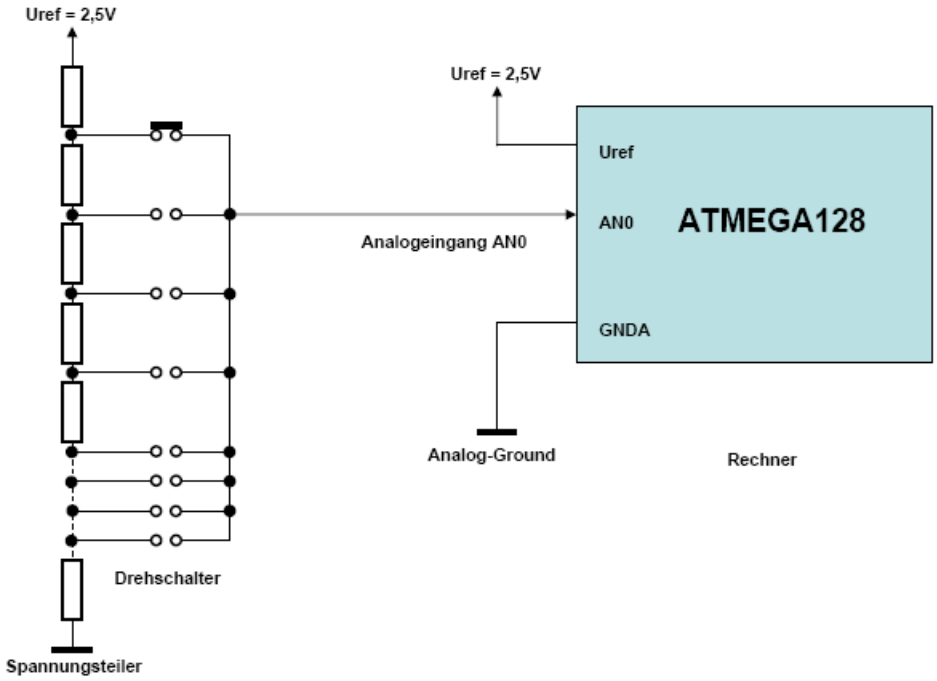


Bild 11 Einlesen eines mehrstufigen Schalters

3.7.3 Die Software-Realisierung

In jeder Software befindet sich ein Hauptprogramm. Dieses Hauptprogramm besteht eigentlich immer aus einer „Forever-Loop“. Das ist eine Programmschleife, die immer wieder ausgeführt wird. In dieser Programmschleife werden u.a. alle Schalter und Tasten abgefragt. Erst wenn z.B. eine Taste gedrückt ist, verzweigt das Programm und arbeitet die Funktion der Taste ab. In unserem Fall wird der A/D-Wandler zu Beginn einer jeden Programmschleife abgefragt. Sobald sich die Schalterstellung des Stufenschalters ändert, liegen neue A/D-Wandlerwerte vor. Diese Werte werden in einem Unterprogramm ausgewertet und als Ergebnis bekommt man dann die eingestellte Schalterstellung.

3.7.4 Programm-Beispiel

Zunächst sehen wir hier einen Ausschnitt aus dem Hauptprogramm. Es ist die „For Ever Loop“, welche der Rechner immer wieder neu durchläuft. Zu Beginn dieser „For Ever Loop“ wird auch der A/D-Wandler bestimmt und ausgelesen. Der gelesene Wert befindet sich dann in der Variablen „Adc_val“. Diese Variable dient zur Übergabe des A/D-Wandlerwertes an das Unterprogramm „Keycode“. Dieses Unterprogramm berechnet die Schalterstellung und speichert diese in der Variablen „T_val_tmp1“ ab. Durch diese Variable erkennt der Rechner, ob sich die Schalterstellung geändert hat oder nicht.

Danach folgt der allgemeine Programmablauf des Hauptprogramms. Ist der Rechner damit fertig, wiederholt sich die "For Ever Loop" von Neuem.

```
Init Steuerrechner                  hier wird der Steuerrechner initialisiert
Do
        'AD-Wandler bestimmen
        Channel = 7                 A/D-Wandler 7 wird selektiert
        Adc_val = Getadc(channel)   A/D-Wandler auslesen
        'Keycode decodieren
        Taste = Keydecode1(adc_val) Unterprogramm aufrufen
        If T_val_tmp1 <> Taste Then
                T_val_tmp1 = Taste  alten Tastenwert zwischenspeichern
        End If
        ´Allgemeiner Programmablauf
        ...
        …
        …
Loop
```

Hier ist nun das Unterprogramm, welches den A/D-Wandlerwert in die Schalterstellung umrechnet.

```
Function Keydecode0(byval Adc_val As Word) As Word
'  Diese Function bekommt den AD-Wandler-Wert und setzt ihn in den Keycode um.

  Select Case Adc_val
        Case 0 To 90
              Keydecode0 = 0              'Schalterstellung 1   --- AM-Demodulation
        Case 91 To 180
              Keydecode0 = 1              'Schalterstellung 2  --- USB-Demodulation
        Case 181To 270
              Keydecode0 = 2              'Schalterstelung 3  --- LSB Demodulation
        Case 271 To 360
              Keydecode0 = 3               'Schalterstellung 4  --- FM Demodulation
        Case 361 To 450
              Keydecode0 = 4               'Schalterstellung 5  --- CW
        Case 451 To 540
              Keydecode0 = 5               'Schalterstellung 6  --- PSK31
        Case Else
              Keydecode0 = 0               `ungültige Schalterstellung
  End Select

End Function
```

Das schaut ganz einfach aus, und das ist es auch. Der A/D-Wandler im ATMEGA128 hat 10 Bit Auflösung und kann damit 1024 verschiedene Werte erfassen. Weil diese Werte nicht so stabil sind und auch ein wenig rauschen, verwendet man einen ganzen Bereich für eine einzige Schalterstellung. Misst der A/D-Wandler z. B. einen Wert von 305, dann wird das in der Zeile erkannt, welche mit „Case 271 To 360“ beginnt. Daraus wird der Keycode „3“ generiert und als Ergebnis abgegeben. In unserem Beispiel entspricht das der Schalterstellung 4, welche zur FM-Demodulation genutzt wird.

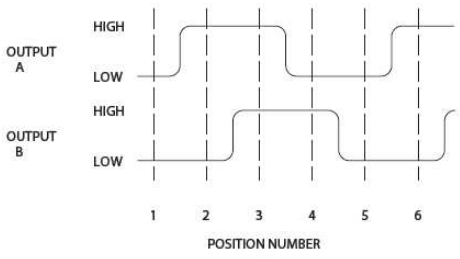
3.7.5 Drucktasten einlesen

Die Drucktasten werden in einem Widerstandsnetzwerk zusammengefasst und liefern ebenfalls je nach gedrückter Taste eine bestimmte Spannung. Diese Spannung wird von einem anderen A/D-Wandler-Kanal in den ATMEGA128 eingelesen und genau so wie der Drehschalter ausgewertet. Der Steuerrechner weis dann auch genau, welche Taste gedrückt ist. Normalerweise geht man davon aus, dass immer nur eine Taste gedrückt wird. Sollte es aber einen Sinn ergeben, dass mehrere Tasten gleichzeitig gedrückt werden, so kann man das Widerstandsnetzwerk so berechnen, dass bei jeder Tasten-kombination eine andere Spannung entsteht. Dann kann man auch erkennen, welche Tasten gleichzeitig betätigt sind.

3.7.6 Tunig-Knopf einlesen

Auch mit dem Einlesen des Tunig-Drehschalters beim LIF5000 war ich nicht ganz zu-frieden, weil da nur jeder zweite Impuls einen Interrupt auslösen kann. Das kommt daher, weil der Interrupt immer nur bei einer positiven Flanke vom Drehschalter ent-steht. Nun verknüpfe ich die Signale vom Drehschalter mit einem XOR-Gatter, welches das Signal vom Drehschalter einmal direkt auf einen Eingang und dasselbe Signal auf den zweiten Eingang durch ein R/C-Glied verzögert bekommt. Dabei entstehen bei jeder positive und negativen Flanke jeweils ein eigener Impuls, welcher wieder über beide Flanken verfügt. Dadurch führt im HSDR-4512 jede Flanke des Drehschalters zu einem Interrupt und man kann die Frequenz viel angenehmer einstellen.

Bild 12 Tunig-Drehschalter



CLOCKWISE ROTATION		
POSITION	OUTPUT A	OUTPUT B
1		
2	•	
3	•	•
4		•

• INDICATES LOGIC HIGH; BLANK INDICATES LOGIC LOW. CODE REPEATED EVERY 4 POSITIONS

Bild 13 Timing Drehschalter „Tuning“

3.8 Display-Treiber

Normalerweise kann man das LCD-Display direkt an die Port-Ausgänge des Steuerrechners anschließen. Da die Flanken der Display-Steuerung aber sehr schnell sind, reicht der Schaltstrom des Rechners nur für eine ganz kurze Verbindung zwischen Rechner und Display aus. So wurde das im LIF5000 realisiert. Da wird das LCD-Display direkt auf den Steuerrechner aufgesteckt und die Verbindungen zwischen dem Display und dem ATMEGA128 sind sehr kurz. Aber der Steuerrechner des HSDR-4512 sollte auf einer eigenen Steckkarte aufgebaut werden, um ihn auch in anderen Geräten zu verwenden. Das hatte zur Folge, dass der Abstand zwischen dem ATMEGA128 und dem Display größer wurde. Um eine saubere Verdrahtung zu ermöglichen, wurden die Signalleitungen zum Display auch noch über die Steckverbindung des Steuerrechners und von dort aus zum Display geführt. Wer sich das Datenblatt des Displays angeschaut hat, weis, dass die Signalleitungen dadurch viel zu lang werden und eine einwandfreie Funktion des Displays nicht mehr gewährleistet ist.

Bild 14 Alphanumerisches Display mit 4 Zeilen

Die Funktion des Displays ist deshalb nicht mehr gewährleistet, weil die Portausgänge des ATMEGA128 zu schwach und unsymmetrisch sind. Das führt zu ungleichmäßigen Flanken und die Signale auf den verschiedenen Leitungen kommen je nach Polarität zu unterschiedlichen Zeiten am Display an. Die Flankensteilheit ist ungleichmäßig je nach Polarität und insgesamt zu klein. Um das zu vermeiden, habe ich nun einen starken Bustreiber als Verstärker nach dem ATMEGA128 eingesetzt. Auch ist dieser Bustreiber in seiner Ausgangsimpedanz symmetrisch und sehr niederohmig. Mit jeweils einem 100 Ohm-Widerstand wird nun die Signalleitung angepasst. Das ist zwar keine so gute Anpassung, wie wir sie von einer Sendeendstufe zur Antenne erwarten würden, aber man kann damit eine relativ gute Signalintegrität erreichen. Die Steuersignale kommen zwar nicht perfekt am LCD-Display an, aber die Signalform und Steilheit reicht aus, um das Display zuverlässig anzusteuern. Sollte das trotzdem nicht ausreichen, kann man am anderen Ende der Signalleitung vor dem LCD-Display noch einen eigenen Empfangs-IC mit einer entsprechenden Pegelanpassung vorschalten. Üblicherweise ist so was aber für so eine kurze Verbindung zum Display, wie wir sie hier haben, nicht notwendig.

3.9 Reserve Ein.- und Ausgänge

Da ich mir mit den Analogeingängen für den Drehschalter und die Tasten einige Porteingänge einsparen konnte, sind mir nun noch 4 Eingänge und 4 Ausgänge als Reserve übrig geblieben. Diese Ein.- und Ausgänge habe ich durch einen IC auf 3,3 Volt Logikpegel umgewandelt und an den Stecker der Platine geführt. Oft ist man später sehr froh, wenn man noch einen Eingang oder Ausgang zur freien Verfügung hat.

4. Schlusswort

Nun ist der Anfang gemacht. Der Steuerrechner wird demnächst fertig und dient dann zur Abarbeitung der Bedienelemente und zur Steuerung des Preselektors. Später wird damit dann auch der Synthesizer gesteuert. Im Moment kann man damit den Preselektor in Betrieb nehmen.

Natürlich kommt damit auch wieder die Phase der Software-Entwicklung. Dieses Mal wird etwas mehr Software nötig, als beim LIF5000. Aber der größte Teil der Software kann vom LIF5000 übernommen und angepasst werden, sodass sich die Software-Entwicklung in Grenzen halten wird.

Und dann steigt die Spannung. Nach dem Preselektor kommt der eigentliche RX mit dem Mischer, dem Synthesizer, dem ADC und dem DDC sowie dem DSP. Für mich wird das alles sehr interessant und ich freue mich schon darauf. Ich werde versuchen, die Funktionsweise der neuen, digitalen Bausteine wie ADC, DDC und DSP und dem Synthesizer mit einfachen Worten zu erklären. Auch werden wir bestimmt einiges über die Software im DSP erfahren.

**Natürlich freue ich mich wieder auf sachliche Kritik und Anregungen von Euch.
Habt Ihr Erfahrungen 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