Werner Nitsche
DL7MWN

Zwischenbericht Steuerrechner für den LIF5000 vom 08.06.2007

So wird der LIF5000 ausschauen.

1. Vorwort

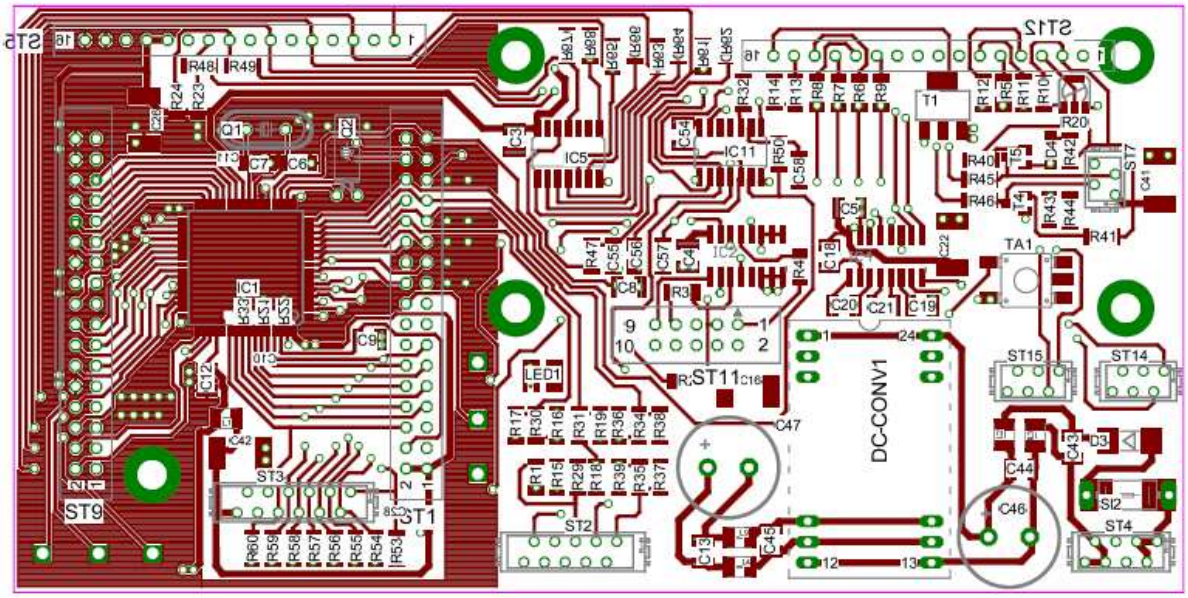
Es ist wieder eine weitere Etappe geschafft. Der Schaltplan für den Steuerrechner des LIF5000 ist abgeschlossen und nach bestem Wissen und Gewissen kontrolliert. Das war ein hartes Stück Arbeit, weil man nach meiner Meinung in Digitalschaltungen noch viel schwerere Fehler machen kann, als es bei Analogschaltungen möglich ist. Die Leiterplatte ist ebenfalls fertig und die Daten habe ich gerade zum Leiterplattenhersteller geschickt.

Um die Funktionsweise des Steuerrechners zu erklären, habe ich neben dem Schaltplan auch noch ein Manuskript zusammengestellt, aus dem die einzelnen Funktionen besser ersichtlich sind. Diese Unterlagen haben mir auch bei der Überprüfung des Schaltplans geholfen. Einige Fehler konnten so vermieden werden. Diese zusätzlichen Unterlagen haben die Bezeichnung:

Vom fertigen Mikrocontroller-Modul zum eigenen ATMEGA128-Steuerrechner für den LIF5000.

Dieses Manuskript ist ebenfalls auf meiner Homepage zu finden und wird zum Download angeboten.

1.1 Die Leiterplatte



So schaut die Leiterplatte aus. Rechts oben sieht man die Befestigungsbohrungen für das LCD-Display. Und links in der Mitte ist der Mikrocontroller ATMEGA128. Sämtliche Portein-/ausgänge wurden auf Teststecker (ST1 und ST9) gelegt und können für Erweiterungen oder Testzwecke genutzt werden. Da für die Massefläche bei diesen vielen Leitungen nur wenig Platz übrig blieb, habe ich die Leiterplatte im Bereich des Mikrocontrollers beidseitig (links) mit Masse geflutet. Im Bereich der restlichen Elektronik (rechts) wird nur die Unterseite mit Massefläche geflutet. Der Grund liegt darin, dass man Bauteile auf den Masseflächen nur schwer einlöten kann und die Gefahr eines Kurzschlusses sehr groß ist.

2. Die Funktionsgruppen

Das Herzstück des Steuerrechners ist natürlich der ATMEGA128 selbst. Aber mit einem Mikrocontroller alleine lässt sich auch nur wenig anfangen. So ist die Beschaltung um den Mikrocontroller von ganz wesentlicher Bedeutung für die Gesamtfunktion.

2.1 Die Stromversorgung

Der Mikrocontroller selbst arbeitet mit 5 Volt. Die Versorgung des LIF5000 erfolgt mit 12 Volt. Da der Mikrocontroller zusammen mit seiner Steuerelektronik kurzzeitig ganz schön Strom benötigt (z.B. während der Programmierung), soll kein analoges Netzteil verwendet werden. Die 12-V-Eingangsspannung würde in einem analogen Netzteil ja nur verheizt. So habe ich mich entschlossen, einen fertigen DC/DC-Converter von der Fa. Conrad einzusetzen. Natürlich ist mir bewusst, dass eventuelle Störungen dieses Wandlers direkt im Empfangsbereich des LIF5000 liegen. Aber ich denke, dass eine gute Entstörung das Schlimmste verhindern kann. Der Steuerrechner mit dem DC/DC-Converter sitzt ja auch auf einer eigenen Platine. Sollten sich wider Erwarten doch größerer Störbeeinflussungen einstellen, dann muss ich noch zusätzliche Entstörmaß-

nahmen vorsehen. Das entscheide ich aber erst, wenn ich solche Störungen messtechnisch erfassen kann.

2.2 Das ISP-Programmier-Interface

Den ATMEGA128 kann man auf verschiedene Weise programmieren. Ich habe mich im LIF5000 für ein ISP-Programmier-Interface (ISP = In-System-Programming) entschieden, wie ich es seit Jahren erfolgreich im Einsatz habe. Für dieses Interface gibt es die passenden Kabel mit eingebautem Programmer im Stecker fertig zu kaufen. Auch kann man es leicht selber bauen. Zum Programmieren selbst verwende ich die Programmiersoftware PonyProg. Diese Software kann im Internet kostenlos heruntergeladen werden und hat sich als Standard-Software zum Programmieren vieler CPUs und Mikrocontroller bewährt. Auch gibt es dafür im Internet eine sehr gute Dokumentation.

Wie schon der Name (ISP = In-System-Programming) sagt, kann der Mikrocontroller mit dieser Methode auf dem Board elektrisch gelöscht und wieder neu programmiert werden. Damit kann man deutlich angenehmer arbeiten, als mit manch anderem Mikrocontroller, den man jedes Mal aus der Platine entnehmen, ihn dann mit UV-Licht löschen und dann anschließend in einem externen Programmiergerät wieder neu programmieren muss. Nach dieser umständlichen Prozedur muss so ein Baustein ja auch noch wieder auf die Platine gesteckt werden, wobei mechanische oder sogar elektrische Beschädigungen passieren können. Bei mir haben sich die Prozessoren aus der AVR-Familie bestens bewährt.

In meinen Unterlagen „Vom fertigen Mikrocontroller ….“Habe ich auf Seite 2 das gesamte ISP-Programmierinterface dargestellt. Auf der rechten Seite ist sogar die Schaltung des Programmers abgebildet, sodass man sich die Gesamtfunktion besser vorstellen kann.

Auf meinem Steuerrechner erfolgt die Programmierung des ATMEGA128 ausschließlich über 2 Pins. Diese beiden Pins heißen PE0 und PE1. Leider werden gerade diese beiden Pins auch noch für die serielle Schnittstelle benötigt, sodass man diese Pins durch einen Analogschalter 74HC4053 zum Programmieren umschalten muss. Das erforderliche Umschaltsignal „PROG“ kommt aus der Programmiersoftware. Solange programmiert wird, leuchtet eine rote LED.

Downloading Adapter

Ribbon cable

2.3 Das I2C-Bus-Interface

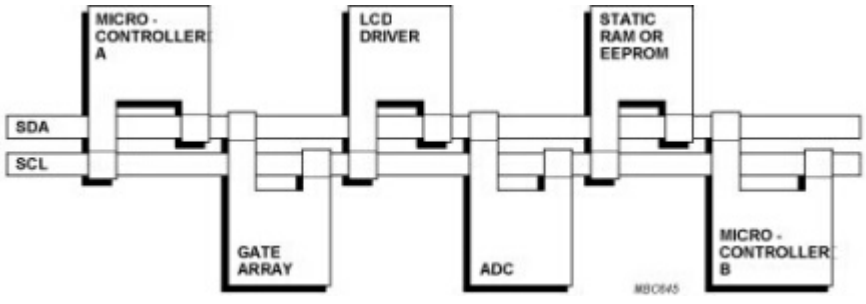
Um an den ATMEGA128 zusätzliche Ein- und Ausgänge anschließen zu können, muss dieser mit einem I2C-Bus-Interface erweitert werden. Speziell zum Preselektor werden alle Schaltsignale über den I2C-Bus übertragen. Ohne ihn könnte der Steuerrechner die Bandfilter nicht umschalten. Auch die Vorverstärkung und das S-Meter könnten ohne den I2C-Bus nicht bedient werden.

2.3.1 Aber was ist der I2C-Bus eigentlich?

In vielen modernen elektronischen Systemen wird häufig eine Kommunikation der einzelnen Bausteine untereinander benötigt. Man will aber auch nicht zig Leitungen durch das ganze System legen, also war ein Ziel, so wenig wie möglich Leitungen zu verbrauchen.

I^2C-Bus heißt ausgeschrieben Intercityexpress IC-Bus, zu Deutsch: Interner Integrierter-Schaltungs-Bus.

Philips ist die Mutter dieses Busses, der sich zum Quasi-Standard entwickelt hat. Manchmal wird er auch 2-Draht-Bus genannt, was auch nicht abwegig ist, da der Bus tatsächlich nur mit 2 bidirektionalen Leitungen auskommt (Masse und Versorgungsspannung nicht mitgerechnet).



Die erste Leitung wird mit SDA (= serial data) bezeichnet. Über diese Leitungen werden die eigentlichen Daten seriell übertragen.

Die zweite Leitung wird mit SCL (= serial clock) bezeichnet. Hier werden die Takt-Impulse gesendet.

Die ursprünglichen Spezifikationen sahen eine maximale Bus-Geschwindigkeit von 100kHz vor. Für heutige Anwendungen ist dies aber manchmal nicht mehr ausreichend, also wurden die Spezifikationen überarbeitet.

Jeder I2C-Baustein lässt sich über ein 7-bit breites Adressbus-"Byte" selektieren. Das 8. Bit des Adressbytes gibt an, ob auf den Baustein lesend oder schreibend zugegriffen werden soll. Die folgenden Bytes sind dann vom jeweiligen Baustein abhängig.

Eine Kommunikation findet immer zwischen einem sog. *Master* und einem sog. *Slave* statt. Es gibt auch Multi-Master-Modi, auf diese gehe ich aber hier nicht weiter ein.

Der Master sendet nun eine sog. *Start-Condition.* Dadurch werden nun alle Slaves am Bus hellhörig und vergleichen ihre Adresse mit der Adresse, die der Master anfordert. Nun können der Master und der Slaverechner Daten austauschen. Ist dies erledigt, sendet der Master eine *Stop-Condition*. Dadurch wird der Bus wieder freigegeben und das Spiel kann von vorne beginnen.

So sieht ein lesender Zugriff auf einen PCF 8574 aus:

	Adresse				Geräte-Sub-Adresse			R/W		Daten								
Start	0	1	0	0	x	x	x	1	ACK	x	x	x	x	x	x	x	x	Stop

Die ersten vier Bit sind vom jeweiligen Baustein abhängig und können nicht geändert werden. Die drei darauf folgenden Bit sind vom Baustein selbst abhängig, d.h. man kann diese Adresse nach belieben ändern (Pins am Baustein sind hierfür herausgeführt). Dadurch lassen sich insgesamt 8 gleiche Bausteine an einem Bus betreiben.

**Das achte Adressbit gibt nun noch an, ob gelesen oder geschrieben werden soll.
Nun muss der Slave ein Acknowledge an den Master senden, um zu bestätigen, dass er anwesend und bereit ist. Der Master kann nun die eigentlichen Daten auslesen. Möchte der Master weitere Daten lesen, muss er eine Acknowledge an den Slave senden. Benötigt er keine weiteren Daten, so bleibt dies aus und die Stop-Condition wird gesendet.**

Näheres dazu (Timing, etc.) steht in jedem Datenblatt, egal zu welchem I^2C-Basutein, drin.

Ein wichtiges Merkmal sei noch erwähnt: Es ist egal (nach oben hin, sprich langsamer) in welchem Zeitabstand diese Übertragungen stattfinden. D.h. ein Mikrocontroller muss keine genauen Timings einhalten, um mit seinem Slave zu sprechen!

Hinweis:

Dieser Artikel über den I2-Bus wurde weitgehendst aus dem Internet von „das ELKO“ übernommen. Man muss ja das Rad nicht immer wieder neu erfinden.

2.3.2 Wie ist der I2C-Bus hardwaremäßig realisiert

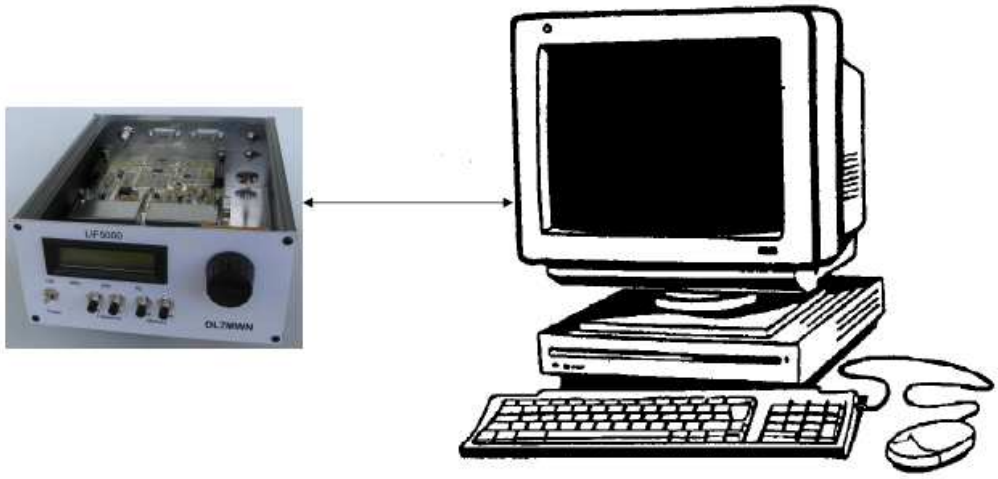
Wie wir nun wissen, werden für den I2C-Bus nur 2 aktive Signalleitungen benötigt. Diese heißen SDA und SCL und benötigen jeweils einen Pullup-Widerstand. Der Pullup-Widerstand bestimmt zusammen mit der Leitungslänge und den parasitären Kapazitäten die Zeitkonstante der Schaltflanken. Je nach Verwendung und Länge des I2C-Busses kann die Länge der Busleitungen kritisch sein. Um dieses Problem zu begrenzen, habe ich mich entschlossen, die Busleitungen mit einer Konstantstromquelle abzuschließen. Das bringt etwas mehr Funktionssicherheit in das Gesamtsystem. Zumindest behaupten das die Experten.

Auf der Seite 3 in meinem Manuskript habe ich die Schaltung des I2C-Buses, so wie er auf dem Steuerrechner-Board des LIF5000 realisiert ist, dargestellt. Das ist eine sehr einfache Schaltung, aber man kann die Konstantstromquellen gut erkennen.

2.4 RS232-Interface für Debug und Fernsteuerung des LIF5000

Der Steuerrechner ist mit zwei RS232-Schnittstellen ausgestattet. Gedacht ist das zur Fernsteuerung des LIF5000 von einem Programm auf dem PC. Schreibt man auf dem Steuerrechner eine Software, die einen bereits bekannten RX emuliert, so kann man den LIF5000 mit bestehender Software steuern. Ob es jeweils so einen Emulator auf diesem Steuerrechner geben wird, ist im Moment noch nicht entschieden. Aber zumindest besteht schon mal die Möglichkeit. Die zweite RS232-Schnittstelle ist für den Software-Debug gedacht. Da kann man ein Terminal (PC mit entsprechender Software) anschließen und z.B. den Ablauf der zu testenden Software im ATMega128 verfolgen oder gar beeinflussen. So etwas ist bisher immer sehr wichtig gewesen, um versteckte Fehler in einer Software zu finden.

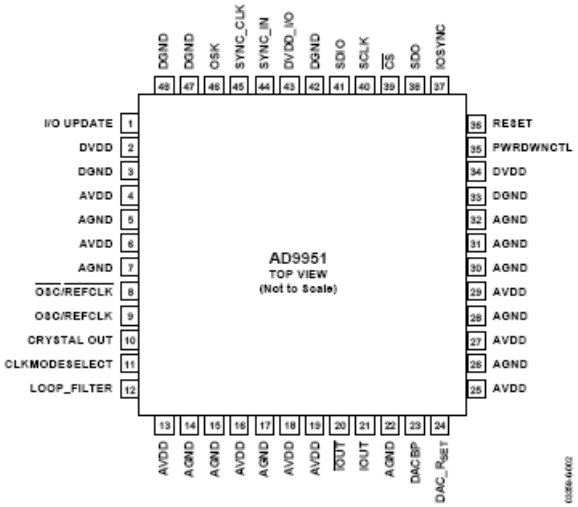
Die beiden RS232-Schnittstellen-Stecker im LIF5000 haben, wie beim PC, die Namen COM1 und COM2. COM2 ist über den RS232-Converter direkt mit dem ATMega128 verbunden. Die RS232-Schnittstelle COM1 ist wie bereits schon einmal erwähnt, über einen Analogschalter mit dem ATMega128 verbunden. Dieser Analogschalter schaltet den ATMega128 zum Programmieren automatisch an das ISP-Programmiergerät. Auf der Seite 4 in meinem Manuskript kann man die Schaltung sehen.



Kommunikation zwischen dem PC und dem LIF5000

2.5 Das Interface zum DDS-Baustein AD9951

Der DDS-Baustein, welcher als LO benutzt wird, ist über mehrere Signalleitungen mit dem Steuerrechner verbunden. Er ist in modernster Technik gebaut und arbeitet mit einer sehr kleinen Betriebsspannung. Die Verbindungssignale zu externen Bausteinen arbeiten mit 3,3 Volt. Unser relativ alter Steuerrechner arbeitet aber mit 5 Volt Versorgungsspannung und 5 Volt Signalen auf den Verbindungsleitungen. Um diese beiden Bausteine miteinander zu koppeln, sind 5 Spanungsteiler, bestehend aus jeweils 3 Widerständen, zwischen diese Bausteine geschaltet. Der dritte Widerstand ist jeweils nicht bestückt und dient nur als Platzhalter, falls doch ein dritter Widerstand notwendig werden sollte. Die verwendete Schaltung ist in meinem Manuskript auf der Seite 5 dargestellt.



2.6 Interruptsteuerung für die Bedienelemente

Wie schließt man mehrere Tasten und Drehschalter an einen Mikrocontroller an. Man weis ja, dass Tasten und Schalter nach der Betätigung prellen. Das bedeutet, dass z. B. so eine Taste bis zu 1ms lang mal ein und mal aus sein kann, bevor sich dann endlich ein stabiler Zustand einstellt. Elektronische Schaltungen arbeiten aber sehr schnell und sie schalten dann ebenfalls ca. 1ms lang ständig hin und her.

Damit kann man aber nichts vernünftiges anfangen. Aus diesem Grund muss man Tasten und Schalter vor der weiteren Verwendung erst einmal entprellen. Das kann man nun auf sehr verschiedene Weise tun. Der geringste Aufwand entsteht, wenn man das der Software überlässt. Aber dann muss die Software die Schalter auch wirklich >1ms lang beobachten. In dieser Zeit kann der Steuerrechner nichts anderes mehr erledigen.

Auch gibt es die Überlegung, ob man Schalter und Tasten pollen soll, oder ob man sie lieber mit einer Interrupt-Steuerung abtastet. Pollen bedeutet, dass man immer wieder in regelmäßigen zeitlichen Abständen Schalter und Tasten abfragt. Man merkt sich den Zustand der Schalter und vergleicht diesen bei der nächsten Abfrage. Hat sich was geändert, dann wertet man diese Änderung aus. Das dauert aber auch so seine Zeit, und wenn ein Schalter vor der nächsten Abtastung sich wieder geändert hat, dann bekommt man das evtl. nicht mit. Eine bessere Lösung ist es, mit der Betätigung einer Taste gleichzeitig einen Interrupt auszulösen. Dann unterbricht der Rechner seine aktuelle Arbeit und liest die Tasten sofort ein. Je nach Software kann der Rechner die Tasten dann sofort abarbeiten oder sich nur merken, dass eine bestimmte Taste betätigt ist.

Ich habe mich beim Steuerrechner für den LIF5000 für eine Interruptsteuerung entschieden, weil man damit in der Anwendung viel flexibler sein kann.

2.6.1 Die Funktionsweise der Interruptsteuerung

Auf dem Board des Steuerrechners sind an einem Stecker 8 Signalleitungen vorgesehen, über welche Tasten eingelesen werden können. Im LIF5000 werden davon aber nur 4 Signalleitungen benötigt. Wird nun eine Taste gedrückt, so wird das entsprechende

Steuersignal direkt an einen Porteingang des ATMega128 weitergeleitet. Gleichzeitig gelangt dieses Signal über ein 8-fach-NAND-Gatter an ein RC-Glied. Danach ist ein Schmitt-Trigger geschaltet, um wieder ein sauberes Digitalsignal zu generieren. Dieses Digitalsignal entsteht ca. 2ms, nachdem die Taste betätigt wurde. Nach dieser Zeit hat sich die Taste beruhigt und prellt nicht mehr. Der Steuerrechner bekommt nun das Interrupt-Signal (INT0) und liest die Tasten ein. Genau so funktioniert es auch mit dem Drehschalter, welcher zum Tunen der Empfangsfrequenz benötigt wird.

Die Schaltung kann man in meinem Manuskript auf der Seite 6 sehen.

2.7 Anschussbelegung ATMEGA128 im LIF5000

Auf der Seite 7 in meinen Unterlagen ist die Anschlussbelegung des ATMega128 zu finden. Sie wird benötigt, um einen schnellen Überblick der einzelnen Signalleitungen zu gewinnen. Dadurch kann eine Doppelbelegung vermieden werden. Im Originalschaltplan mit den BUS-Leitungen ist es schwierig, eine gute Gesamtübersicht zu bekommen. Auf der Seite 8 ist dann noch das Pinout des ATMEGA128 dargestellt.

5. Schlusswort

Der Steuerrechner hat sich bisher als schwierigste Leiterplatte des LIF5000 herausgestellt. Aber man kann ihn dafür später auch noch für andere Projekte verwenden.

Den Schaltplan konnte ich wegen der vielen Signalleitungen nur schwer überprüfen. Mein Manuskript mit den Einzelfunktionen hat mir dann geholfen, etwas mehr Überblick über die Gesamtschaltung zu gewinnen. Auch die Leiterplatte war sehr schwer zu layouten, weil vor lauter Signalleitungen kaum mehr Platz für die Massefläche und die Versorgungsspannung übrig blieb. Aber nun ist das ja erledigt und es geht wieder weiter.

Dann kommt noch die Software, die dem Ganzen dann die Funktionen gibt und danach beginnt die Inbetriebnahme des eigentlichen LIF5000. Zum Schluss wird es dann spannend, wenn alle Parameter des LIF5000 überprüft werden.

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