

Alternative Implementierung des EPP Adapters

Thomas Sailer, HB9JNX/AE4WA, und
Johannes Kneipp, DG3RBU

14. August 1998

1 Einleitung

Im Zuge der Einführung breitbandiger Userzugänge bestand die Notwendigkeit, eine Lösung für den User zu schaffen, womit er einfach und preiswert ein Modem an seinen PC anschließen kann. Daher wurde der EPP-Adapter entwickelt und auf der letztjährigen Darmstädter Tagung präsentiert [5]. Messungen auch an älteren PC's (486er) hatten ergeben, daß HDLC mit der geforderten Bandbreite (einige hundert kbit/s) durchaus mit minimalem Aufwand in Software codiert und decodiert werden kann. Tabelle 1 zeigt der gemessene HDLC-Durchsatz einiger gängiger Prozessoren.

CPU	CPU clock MHz	Bogomips	Encoder MBit/s	Decoder MBit/s
Intel 486DX2	66	33	4.31	3.20
Intel Pentium	75	30	6.24	5.64
Intel Pentium	100	40	9.37	7.48
AMD K5	100	200	15.00	12.69
Cyrix 6x86MX	166	166	19.51	15.53
UltraSparc 1	166		25.75	19.16
AMD K6	200(?)	465	24.73	19.05

Tabelle 1: Erreichbare Software-HDLC-Encoder- und Decodergeschwindigkeiten

Daher wurde im Hinblick auf eine preiswerte Lösung darauf verzichtet, einen HDLC-Controller in Hardware zu realisieren. Auf der Suche nach einer möglichst verbreiteten und einfachen Schnittstelle, die die benötigte Bandbreite bewältigen kann, drängte sich der durch die IEEE normierte erweiterte Parallelport [2] geradezu auf. Damit musste der zu entwickelnde Adapter die 8bit-parallele EPP-Schnittstelle an das bit-serielle Modem anpassen und Daten kurzzeitig zwischenspeichern können, damit sie der PC blockweise verarbeiten kann, das Modem jedoch trotzdem einen kontinuierlichen Datenstrom sieht. Eine Suche nach geeigne-

ten Bausteinen endete beinahe zwangsweise bei den Bausteinen 72131 und 72132 von IDT [4]. Diese IC's beinhalten ein FIFO mit seriellem Eingang und parallelem Ausgang bzw. parallelem Eingang und seriellem Ausgang. Je einer dieser Bausteine und ein wenig Beigemüse vervollständigten den Adapter.

Doch leider scheinen diese Bausteine nicht dem allgemeinen Trend der Elektronik zu folgen und werden statt billiger immer teurer. Heute sind diese FIFO's mit über 35 DM pro Stück zu veranschlagen (zwei werden pro Adapter benötigt), sodaß sich die Suche nach Alternativen aufdrängt. Im Folgenden wird ein neues Design des EPP-Adapters vorgestellt.

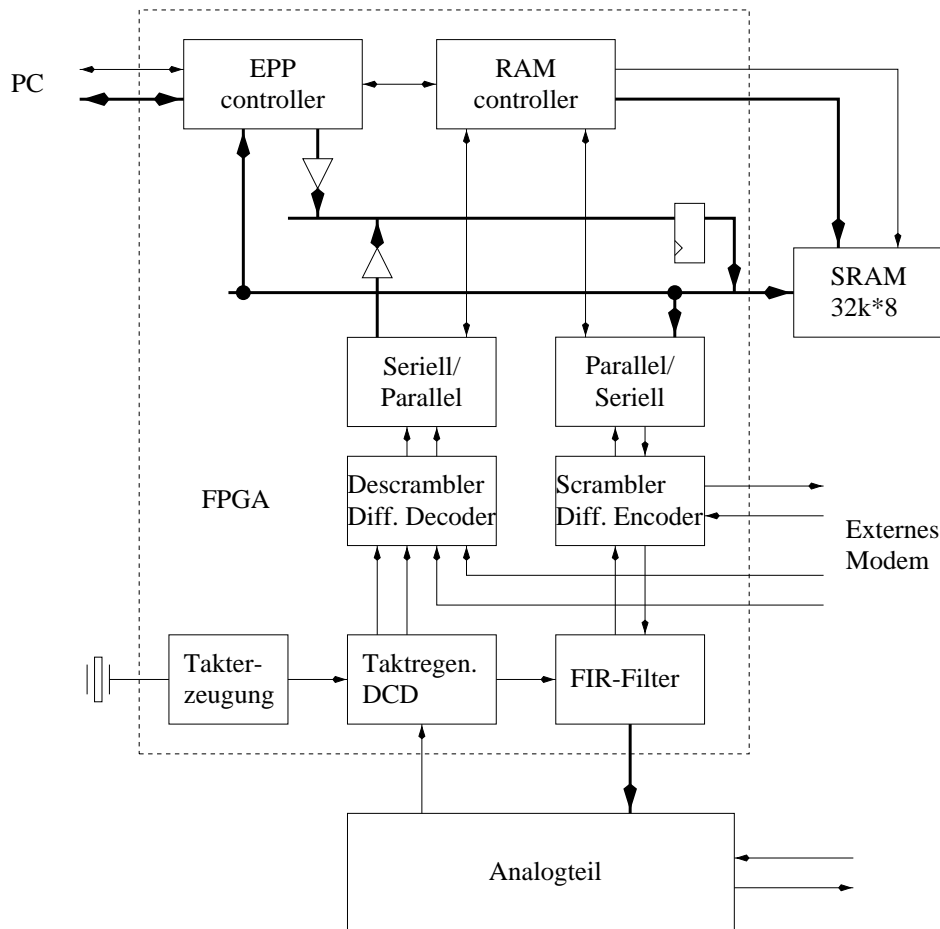


Abbildung 1: Die Schaltung

2 Die Schaltung

Eine Alternative besteht nun darin, FIFO's mittels eines normalen statischen RAM-Bausteins und geeigneter Ansteuerlogik zu realisieren. Da der Adapter nur wenige kbit Speicher

benötigt, ist dies die günstigste Speicherart. DRAM's sind in dieser Größenordnung nicht billiger, verursachen aber einige Schwierigkeiten durch die schmale Schnittstelle (1 oder 4 Bit) und die kompliziertere Ansteuerung (Refresh). Der ganze Rest der Schaltung, also die seriell/parallel-Wandler, die Bedienung der EPP-Schnittstelle und das Einspeichern und Auslesen des RAMs kann mit einem programmierbaren Logikbaustein realisiert werden. Viele programmierbare Logikbausteine enthalten heute zwar auch RAM, die Größen sind jedoch sehr klein und gehen zu Lasten der Logikkapazität, womit das FPGA-interne RAM entsprechend teuer ist.

Die geforderte Datenrate von einigen hundert kbit/s und die maximal übertragbaren 1–2 MByte/s über die EPP-Schnittstelle lassen einen Systemtakt von 10–20 MHz als ausreichend erscheinen. Die zu erwartende Grösse und die moderate Taktfrequenz sind eine Domäne der FPGA (Field Programmable Gate Array) genannten IC's.

Der Logikbaustein muss irgendwo seine Konfigurationsdaten speichern. Drei Klassen können unterschieden werden. Antifuse-Bausteine können nur einmal programmiert werden und kommen daher hier nicht in Frage. EEPROM/FLASH basierte Bausteine können oft reprogrammiert werden, sind aber trotzdem nach dem Einschalten sofort betriebsbereit. SRAM basierte Bausteine können beliebig oft reprogrammiert werden, dies muss aber jedesmal nach dem Einschalten passieren. Die Konfigurationsdaten können üblicherweise aus einem EPROM, einem speziellen seriellen PROM, oder wie hier direkt aus einem Rechner stammen.

Das neue EPP-Modem verwendet einen SRAM-Basierten FPGA des Typs XCS10-3PC84 (alternativ XC4005E-4PC84) vom Marktführer Xilinx [1].

Abbildung 1 zeigt das Blockschaltbild der Schaltung. Die dicken Linien sind Datenpfade, die dünnen Linien Kontrollsignale. Abbildung 6 zeigt das Schaltbild des Modems.

Die "Baublöcke" des Modems sind:

- EPP Controller
- RAM Controller
- SRAM
- Seriell-Parallel und Parallel-Seriellwandler
- Scrambler/Descrambler
- Taktregeneration und DCD
- FIR-Filter
- Takterzeugung
- Analogteil

2.1 EPP Controller

Der EPP-Controller wickelt mit dem PC das EPP-Protokoll [2, 3] ab. Das EPP-Protokoll kennt vier Zugriffstypen. Datenlese- und Schreibzyklen greifen direkt auf die FIFO's zu, währenddem Addresslese- und Schreibzyklen zum Austausch von Statusinformationen verwendet werden. Standard-PC-EPP-Controller generieren EPP-Adresszyklen bei IO-Zugriffen auf deren Basisadresse+3 (0x37b bei LPT1) und EPP-Datenzyklen bei IO-Zugriffen auf die Basisadresse+4 (0x37c bei LPT1). Details zur Ansteuerung der PC-EPP-Controller können den Datenblättern der Hersteller entnommen werden [8, 9, 10]. Die Tabellen 2 und 3 listen die Bedeutung der Bits bei Adresszyklen auf.

Bit	Bedeutung
7	0: DCD an
5:4	Sende-FIFO Füllstand
00	≥ 0 Bytes frei
01	≥ 255 Bytes frei
10	≥ 1792 Bytes frei
11	≥ 1023 Bytes frei
3	PTT (Sende-FIFO nicht leer)
2:1	Empfangs-FIFO Füllstand
00	≥ 1793 Bytes gespeichert
01	≥ 1025 Bytes gespeichert
10	≥ 0 Bytes gespeichert
11	≥ 256 Bytes gespeichert
0	Empfangs-FIFO nicht leer

Tabelle 2: Statusregister (EPP address read Zyklen)

2.2 RAM Controller

Abbildung 2 zeigt das Blockschaltbild des RAM-Controllers. Der RAM-Controller koordiniert die Zugriffe auf das RAM, führt die Adress- und Füllstandszähler und generiert die für den Zugriff auf das statische RAM notwendigen Adress- und Steuersignale.

Abbildung 3 zeigt einen Lesezyklus und Abbildung 4 einen Schreibzyklus. Der relativ langsame Zugriff auf das RAM erlaubt die Verwendung von Standard-RAM's.

2.3 SRAM

Das SRAM implementiert den Datenspeicher der FIFO's. Mit 32kByte ist es zwar reichlich gross, wurde aber gewählt, weil es sehr gut erhältlich ist und kleinere RAM's kaum mehr

2.4 Seriell-Parallel und Parallel-Seriellwandler

Bit	Bedeutung
7	LED: STA
6	LED: CON
5	Modemdisconnect: RESET
4	Sende-FIFO Enable
3	Empfangs-FIFO Enable
2:0	Interruptrate bzw. FIFO Füllstandserweiterung
000	Interrupts aus
001	Lese Empfangs-FIFO-Füllstand
010	Lese SendefIFO-Füllstand

Tabelle 3: Controlregister (EPP address write Zyklen)

erhältlich und wenn dann teurer sind.

2.4 Seriell-Parallel und Parallel-Seriellwandler

Die Seriell-Parallel-Wandler sind doppelt gepuffert, d.h. neben dem eigentlichen Schieberegister enthalten sie noch ein zusätzliches Register zum Zwischenspeichern der Daten. Ebenfalls enthalten ist die Steuerlogik zum Speichern/laden des Schieberegisters ins/vom Speicherregister und zur Kommunikation mit dem RAM-Controller.

2.5 Scrambler/Descrambler

In diesen Blöcken befinden sich die für G3RUH-kompatiblen Modems erforderlichen Scrambler/Descrambler und Differenzencoder und -decoder. Beide Subblöcke lassen sich einzeln überbrücken, um externe Modems anschließen zu können, die meist den einen oder anderen Block schon enthalten.

Ausserdem befindet sich in diesen Blöcken die Umschaltung zwischen internem und externem Modem sowie die Synchronisation der Signale des externen Modems auf den internen Takt.

2.6 Taktregeneration und DCD

Die Taktregeneration extrahiert aus den Eingangsdaten das Empfangstaktsignal mittels einer digital realisierten PLL. Das Eingangssignal wird mit der 16-fachen Bitrate abgetastet. Aus dem Phasenwert der Flanken des Eingangssignales wird das DCD-Signal generiert.

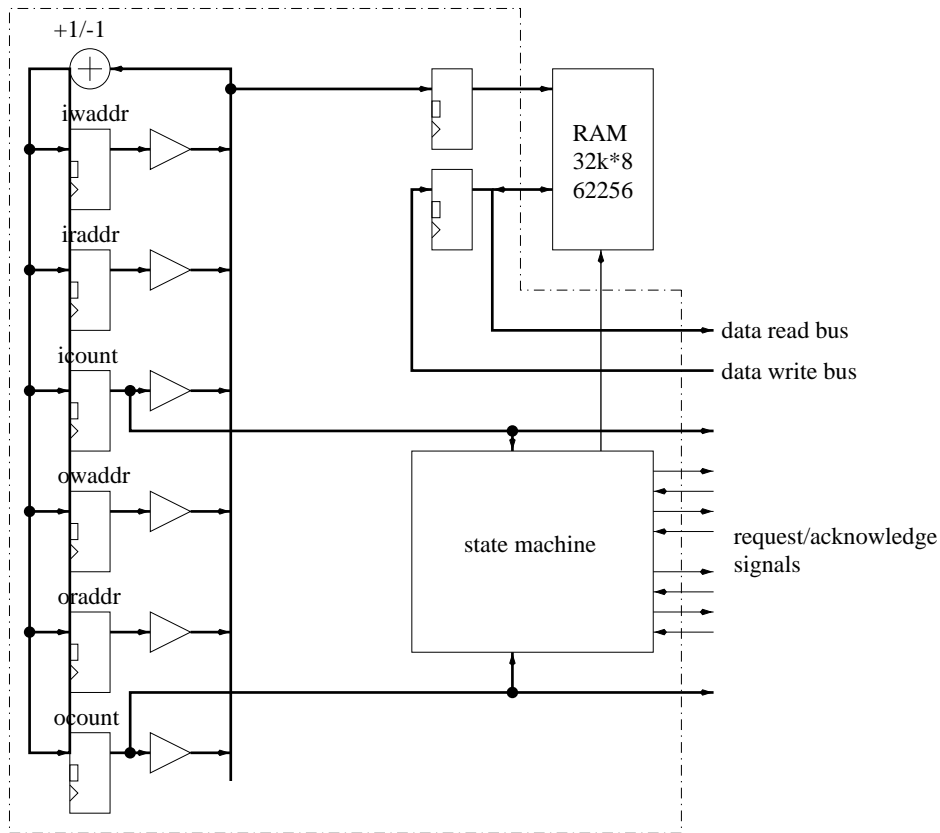


Abbildung 2: Der RAM-Controller

2.7 FIR-Filter

Wie die DF9IC und G3RUH-Modems verwendet auch dieses Modem ein vierfach überabgetastetes FIR-Filter der Länge 32. Die Überabtastung vereinfacht das nachfolgende analoge Sendefilter. Anders als die erwähnten Modems hingegen verwendet dieses nicht eine Tabelle mit vorberechneten Filterausgangswerten, sondern errechnet den Filterwert selber aus den Koeffizienten und den Sendebits.

Da das Modem ein 16-faches Taktsignal verwendet und das Filter vierfach überabgetastet betrieben wird, bleiben 4 Taktzyklen zur Berechnung des Filterwertes. Glücklicherweise sind jeweils 24 der 32 Eingangswerte 0 (dank der Überabtastung) und die restlichen 8 entweder +1 oder -1, sodass die Berechnung des Filterwertes nur acht Additionen erfordert. Da jedoch nur 4 Taktzyklen zur Verfügung stehen, wurde das Filter in zwei Hälften aufgeteilt. Beide Werte werden am Schluss mit einem separaten Addierer zusammengezählt und in den Ausgangsspeicher geschrieben. Abbildung 5 zeigt die Schaltung.

2.8 Takterzeugung

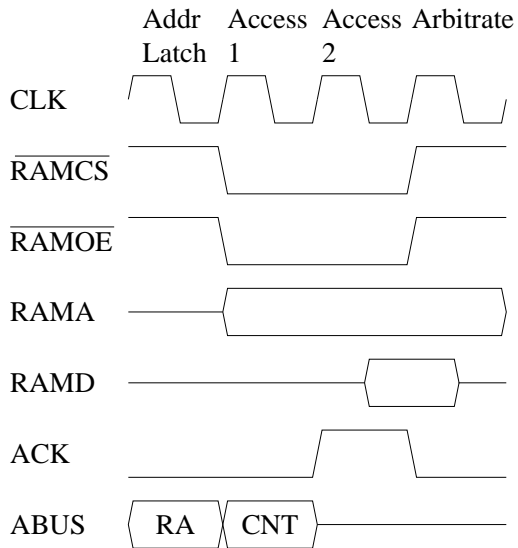


Abbildung 3: RAM-Lesezyklus

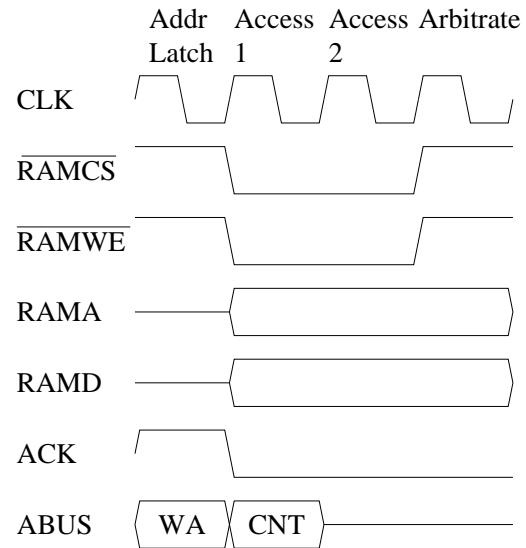


Abbildung 4: RAM-Schreibzyklus

2.8 Takterzeugung

Die Takterzeugung teilt die Frequenz des externen Quarzoszillators durch einen variablen Divisor im Bereich von 1–1024 und speist damit das interne Modem. Die maximale Taktfrequenz des FPGA beträgt 20 MHz.

2.9 Analogteil

Der Analogteil des Modems entspricht weitgehend demjenigen des FSK+-Modems [6]. Einerseits wurden die frequenzbestimmenden Bauteile der Filter für die vorgesehene Bitrate (76.8kBit/s) skaliert, andererseits wurde aus Kostengründen der DAC durch ein Widerstandsnetzwerk ersetzt.

3 Differenzen zum Original-EPP-Adapter

Der FPGA-EPP-Adapter ist weitgehend kompatibel zum konventionellen EPP-Adapter [5]. Trotzdem gilt es einige Differenzen zu beachten. Vorhandene Treiber sollten aber einfach anpassbar sein.

- die Konfigurationsdaten müssen zuerst heruntergeladen werden
- die FIFOs sind 16kByte anstatt 2kByte
- die Interruptrate beträgt konstant ≈ 120 Int/s und ist unabhängig von der Modembitrate

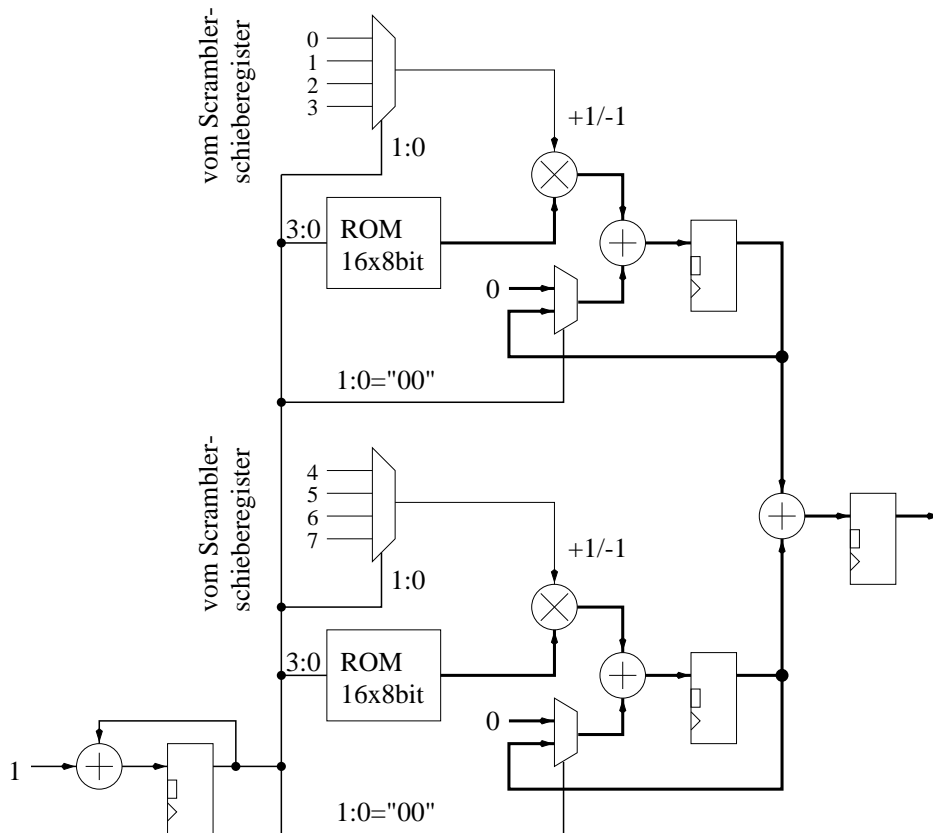


Abbildung 5: Das Sende-FIR-Filter

3.1 FPGA-Konfiguration

Bevor der Adapter benutzt werden kann, müssen die Konfigurationsdaten ins FPGA heruntergeladen werden. Dazu wird das IEEE 1147 JTAG ("Joint Test Access Group") Protokoll verwendet. Dieses Protokoll kann auch für Adaptertests verwendet werden, da damit fast alle Pins des FPGAs "ferngesteuert" werden können.

Die Initialisierung des Adapters sieht nun wie folgt aus:

1. Reset des FPGA
2. Modempräsenz testen (per JTAG die Länge des Instruktionsregisters und des Boundaryregisters testen)
3. Identifikation des Modems (per JTAG boundary scan)
4. Download der FPGA-Konfigurationsdaten
 - Parametrierung des Modems durch Patchen der Konfigurationsdaten

4 Ausblick

4.1 Fazit

Durch die Integration des Modems in den FPGA konnte eine signifikante Kostenersparnis realisiert werden. Durch die weitgehende Kompatibilität zum Originaldesign hielten sich die Anpassungen an der bestehenden Software in engen Grenzen. Die grösste Schwäche des Originaladapters, die Füllstandsbits, konnte beseitigt werden. Das Resultat ist eine Lowcost-Lösung für Useranstiege im Bereich von einigen kBit/s bis zu einigen hundert kBit/s.

Bausätze und Fertigeräte können bei Baycom [7] bezogen werden.

4.2 EPP oder ECP?

Durch die grosse Flexibilität des FPGA's braucht man sich nicht auf eine kompatible Nachbildung des Originaladapters zu beschränken, es können auch Änderungen betrachtet werden.

Der Prototyp wurde bereits erfolgreich mit 1MBit/s getestet. Der CPU-Overhead war dabei aber schon beträchtlich, 25% gemessen auf einem Pentium 100-PC. Der kleinste Teil davon ist aber zum HDLC-Decodieren nötig, weitaus der grösste Teil wird mit der Datenübertragung über den Parallelportadapter verbraucht.

Dies könnte durch Verwendung des ECP-Modus geändert werden. Im ECP-Modus besitzt der Parallelportadapter FIFO's zur Entkoppelung des Datentransfers, ausserdem kann der Parallelportadapter die Daten per DMA direkt in den PC-Hauptspeicher schaufeln. Der Nachteil des ECP Protokolls ist aber, dass vor allem Richtungswechsel des Datentransfers deutlich komplizierter sind als mit dem EPP-Protokoll. Ausserdem haben die meisten Parallelportchiphersteller wohl die Referenzimplementation von Microsoft kopiert, die einige zusätzliche, unnötige Probleme verursacht. Auch IOMEGA scheint es nicht zu wagen, in ihrem Parallelport-ZIP-Treiber DMA zu verwenden. Ob diese Befürchtungen begründet sind, lässt sich aber wohl mit einem Feldversuch schnell herausfinden.

Literatur

- [1] Xilinx Corporation *The Programmable Logic Data Book*
<http://www.xilinx.com>
- [2] *IEEE Standards BBS Information Technology*
<http://standards.ieee.org/catalog/it.html>
- [3] *Introduction to the IEEE 1284-1994 Standard*
<http://www.fapo.com/1284int.htm>

- [4] IDT Corporation *Specialized Memories & Modules*
<http://www.idt.com>
- [5] Wolf-Henning Rech, DF9IC, Johannes Kneip, DG3RBU, Gunter Jost, DK7WJ, und Thomas Sailer, HB9JNX, *Ein Modemadapter für den EPP*, 41. Weinheimer UKW-Tagung, Skriptum, 1996
- [6] Johannes Kneip, DG3RBU, *Das FSK+-Modem mit Echoduplex*, Adacom Magazin 10, 1997
- [7] Baycom \diamond *Bavarian Packet Radio Group*
<http://www.baycom.de/>
- [8] *Standard Microsystems Corporation (SMSC) Personal Computer Input/Output Products*
<http://www.smsc.com/main/catalog/pcio.html>
- [9] *National Semiconductor Product Catalog: SuperI/O*
http://www.national.com/catalog/PersonalComputing_SuperIO_5Volt.html
- [10] *Winbond Serial and Parallel Port Controller*
<http://www.winbond.com.tw/produ/perso5.htm>

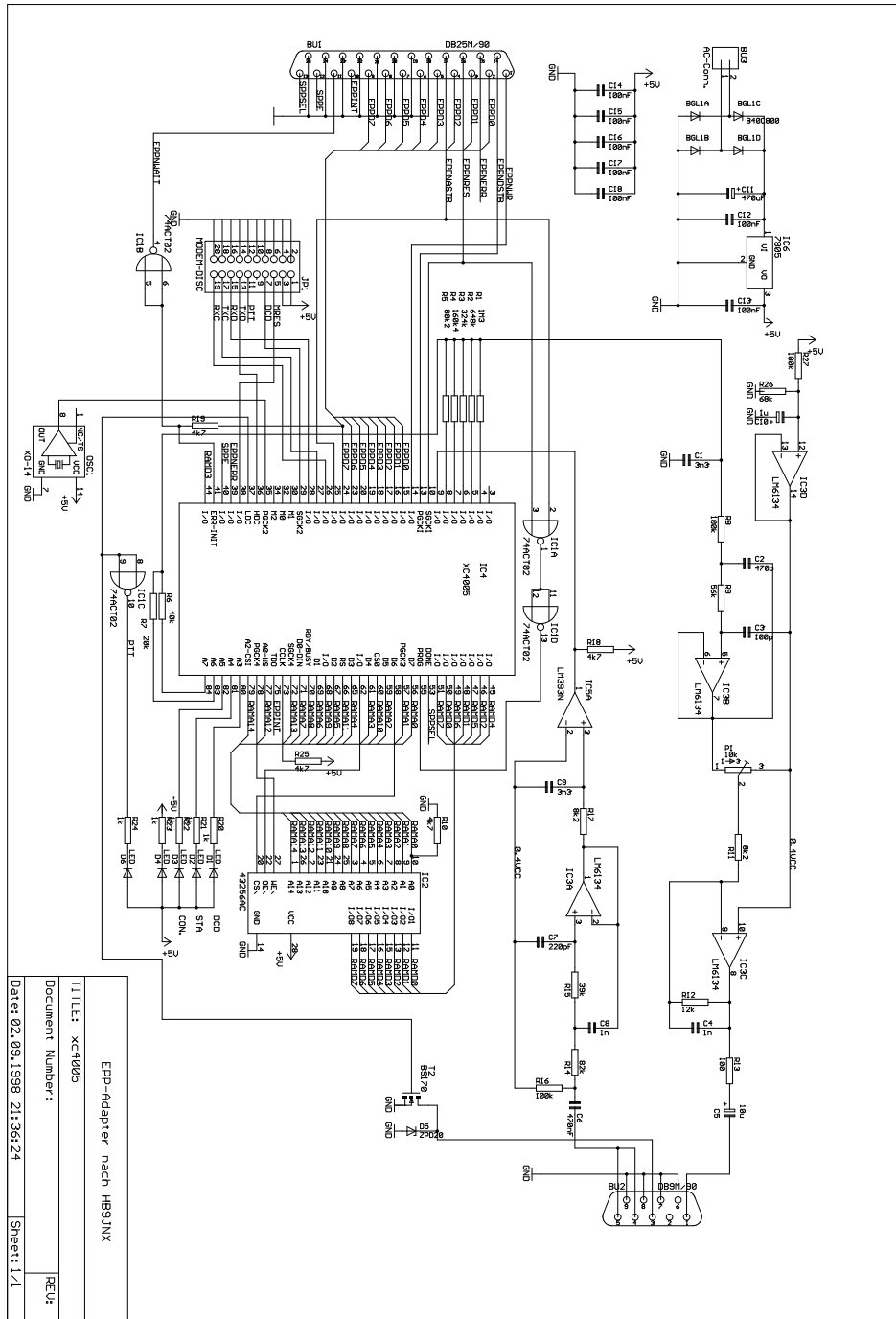


Abbildung 6: Schaltplan