An Inexpensive PC-Modem for 76.8kBit/s User Access

Thomas Sailer, HB9JNX/AE4WA,
and
Johannes Kneip, DG3RBU

August 14, 1998

Abstract

This article describes a simple and inexpensive modem intended to link end users at 76.8kBit/s to the high speed backbone network. The modem can be connected to standard PC's using the Enhanced Parallel Port (EPP) interface.

1 Introduction

The beginning of this project dates back about two years ago. The emergence of a high speed backbone led to the question of how to inexpensively link end users at appropriate speeds to the backbone.

One of the primary design goals was simplicity. We therefore decided to stick with FM and the well known G3RUH modulation format and just scale the technology up by a decade. This led to the allocation of a wideband duplex channel (200kHz per direction) in the 70cm band and the development of an appropriate transceiver [9].

Now the question was how to connect the transceiver to a computer. Most TNC's currently in use are hard pressed to operate at 9.6kBit/s, and therefore are inappropriate for data rates around 100kBit/s. Also, most TNC's connect to the host computer via the serial interface. With radio bit rates approaching the maximum bit rate of the serial interface of todays PC's, this interface becomes the bottleneck. While there exist TNC designs capable of doing 100kBit/s, they were considered too expensive (>$500).

Today, Packet Radio operators use graphical operating systems on their computers and want to use web browser technology also on amateur radio. Most popular amateur radio BBS software already has increasingly popular HTML interfaces to their message base, and HTTP servers are mushrooming also in Amateur Radio installations. These applications made TCP/IP popular, in fact this converted several well known TCP/IP adversaries on the amateur bands to TCP/IP users!

With TCP/IP a requirement, TNC's add little value. Most TNC's need to be switched into a dumb packet IO mode using protocols like 6PACK or KISS, requiring the host CPU to implement AX.25.

2 Design Considerations

As seen in the previous paragraph, we can save cost by implementing AX.25 in the host CPU. Todays PC's can do this with negligible overhead. Now the question is whether HDLC should be done in software or hardware. Measurements on several popular processors as well as older ones (table 1[1]) show that contemporary PC processors can do HDLC encoding and decoding up to a few hundred kBit/s with little overhead. It was therefore decided to do the HDLC encoding and decoding in software, making the hardware adapter even less complex.

CPU	CPU clock	Bogomips	Encoder	Decoder
	MHz		MBit/s	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

Table 1: Software HDLC encoder and decoder throughput

The next decision was what interface to use. The interface had to fast enough including headroom for expansion (excludes RS232 serial ports), widely available (excludes USB), and simple to use (excludes USB, ethernet). The remaining interface was the Enhanced Parallel Port, which is available on virtually any computer for several years already and which was standardized by IEEE [2].

We decided to use an already existing modem design [7]. The adapter to be designed thus had the following requirements:

- EPP interface
- Synchronous serial interface according to [7] to connect to existing modems [7, 6]
- Provide elastic buffering to allow block-wise processing of the data by the host CPU

3 The Initial Design

A search for suitable components ended at IDT's FIFO circuits 72131 and 72132 [4], which ideally suited the above requirements, since they additionally also contain a parallel to serial or serial to parallel converter

[1]Benchmarking was done using "optimized" C language, similar to what is in the Linux driver for this adapter, see for example linux-2.1.108:drivers/net/hamradio/baycom_epp.c

shift register respectively. Adding some glue logic and drivers to provide high drive capabilities on the EPP signals completed the initial design, which was published in [5].

Unfortunately, these IDT IC’s are quite expensive, and are getting more expensive, contrary to the general trend in microelectronics. The external modem adds to the cost, too. We therefore considered a redesign.

4 The Current Design

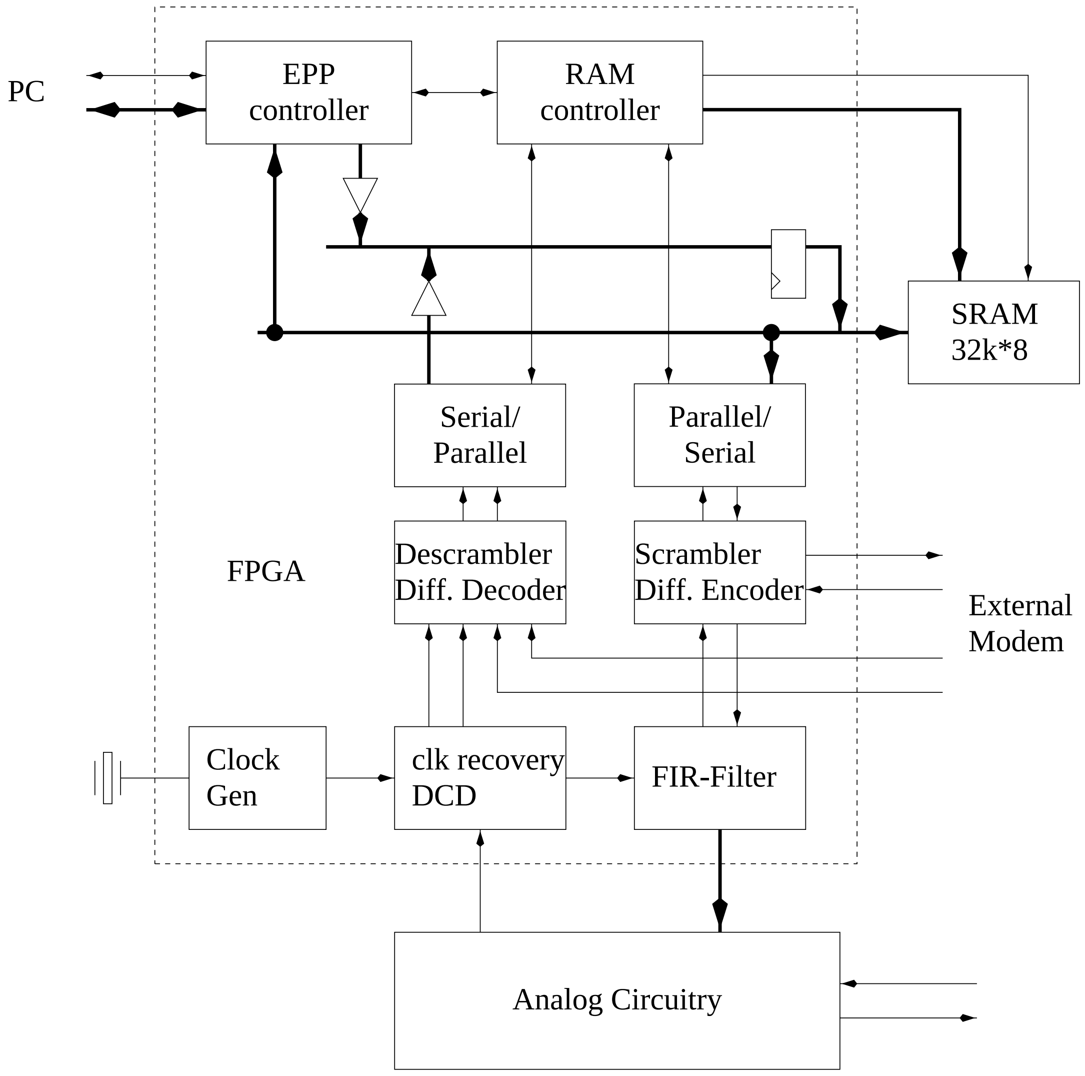

Figure 1: Block diagram of the circuit

An alternative to using dedicated FIFO IC's is to use standard SRAM and add appropriate control logic. Figure 1 shows a block diagram of the circuit of the redesigned adapter. The thick lines represent data paths, while the thin lines show control signals.

A system clock frequency in the range of 10–20 MHz seems adequate for bit rates in the 100kBit/s range and for the maximum transfer rate of the EPP port in the range 1–2 MByte/s. The relatively moderate clock frequency and the size of the logic required to implement the modem adapter suggest the use of a FPGA (Field Programmable Gate Array) to implement it.

Unlike other programmable logic families, FPGA's store their configuration in static RAM cells, which loose their information at power down. At power up, they need to reload their configuration data, which may originate from a special serial PROM, a standard EPROM, or directly from the PC. The latter was chosen since it is more flexible, cheaper and easier to develop. Also, modem options can be realised by patching the configuration data prior to download to the FPGA.

The configuration data is downloaded to the FPGA using the IEEE 1147 JTAG ("Joint Test Access Group") protocol. The protocol also allows testing of various parts of the adapter.

The building blocks of the modem will be described in more detail in the following subsections. Figure 6 shows the circuit diagram of the modem.

4.1 EPP Controller

The EPP controller handles the EPP protocol [2, 3] together with the PC. Data read and write cycles directly access the data FIFOs, while address read and write cycles access the status and control register respectively. Standard PC EPP controllers emit EPP address and data cycles on IO accesses to their base address +3 (0x37b for LPT1) and +4 (0x37c for LPT1). EPP controller programming details can be found in their data sheets [10, 11, 12]. Tables 2 and 3 list the meanings of the register bits.

4.2 RAM Controller

Figure 2 shows a block diagram of the RAM controller. The RAM controller coordinates RAM accesses, keeps address and byte counters, and generates the control signals for the static RAM access.

Figure 3 shows a read cycle, while Figure 4 shows a write cycle. The relatively slow cycles allow the use of inexpensive standard RAMs.

4.3 SRAM

The SRAM implements the data storage for the FIFOs. 32kBytes are huge for the application, but smaller RAM's are either more expensive or no longer available at all.

Bit	Purpose
7	0: DCD an
5:4	transmitter FIFO
	00 ≥ 0 bytes free
	01 ≥ 255 bytes free
	10 ≥ 1792 bytes free
	11 ≥ 1023 bytes free
3	PTT (transmitter FIFO not empty)
2:1	receiver FIFO
	00 ≥ 1793 bytes stored
	01 ≥ 1025 bytes stored
	10 ≥ 0 bytes stored
	11 ≥ 256 bytes stored
0	receiver FIFO not empty

Table 2: status register (EPP address read cycles)

Bit	Purpose
7	LED: STA
6	LED: CON
5	external modem connector: RESET
4	transmitter FIFO enable
3	receiver FIFO Enable
2:0	interrupt rate or FIFO status
	000 interrupts off
	001 read receiver FIFO count
	010 read transmitter FIFO count

Table 3: control register (EPP address write cycles)

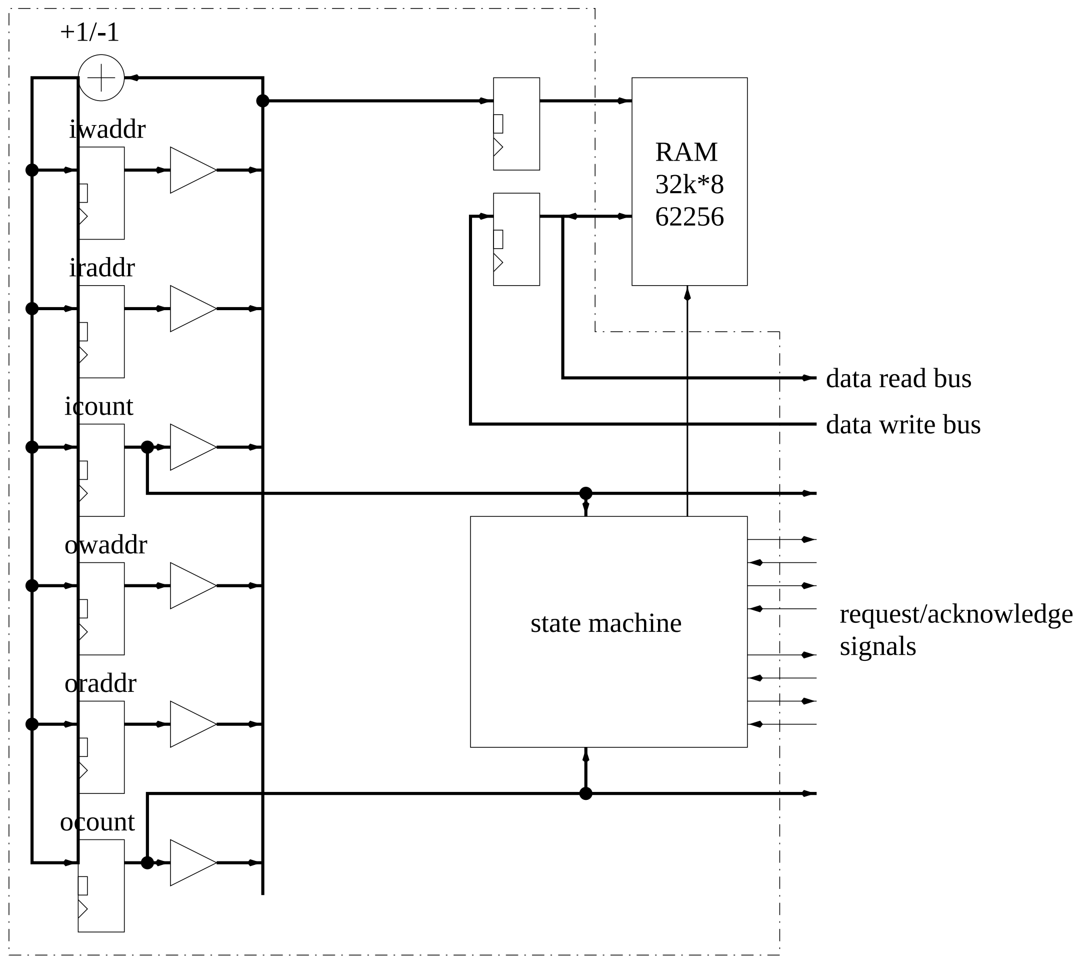

Figure 2: RAM controller

4.4 Serial to parallel and parallel to serial converter

The serial/parallel converters are doubly buffered, i.e. they contain an additional storage register besides the shift register. These blocks also contain the control logic to request emptying/filling of the storage register by the RAM controller and to transfer data between storage and shift register.

4.5 Scrambler/Descrambler

These blocks contain a G3RUH compatible scrambler/descrambler and a differential encoder/decoder. Both blocks can be bypassed independently to accomodate external modems which usually already contain one or the other functionality. These blocks also contain the switch between the internal and the external modem, as well as synchronisation circuitry for the external modem signals.

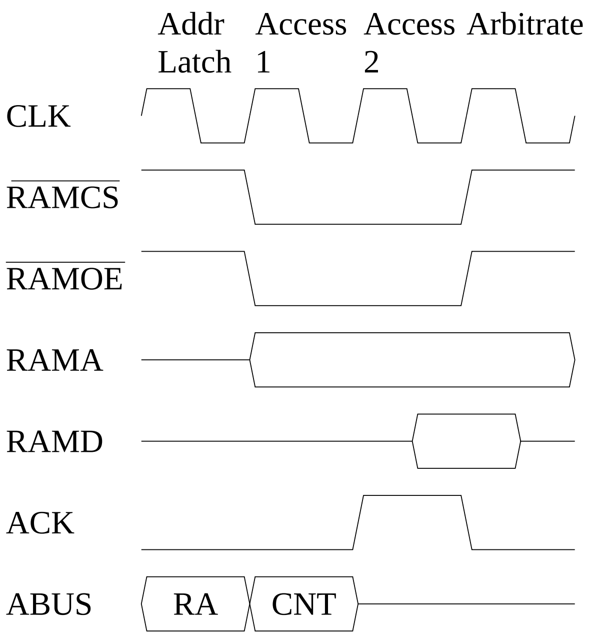

Figure 3: RAM read cycle

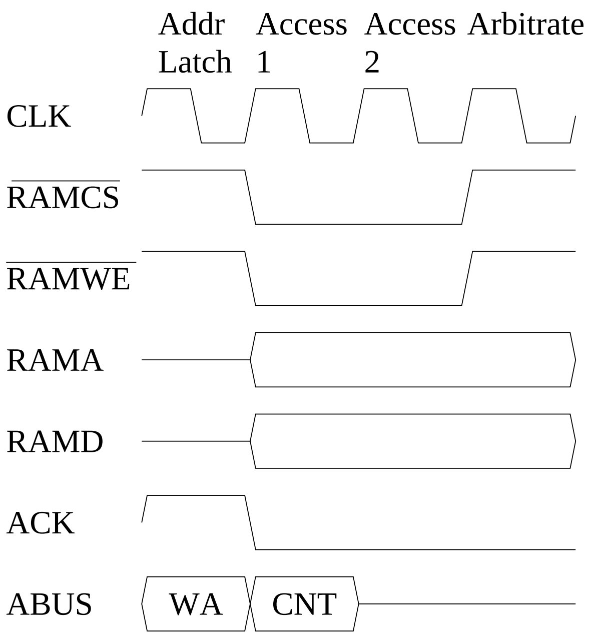

Figure 4: RAM write cycle

4.6 Receiver clock regeneration and DCD logic

The clock recovery block extracts the receive clock from the input data signal using a digitally implemented PLL. The input signal is oversampled 16 times. The DCD signal is generated from the phase values of input signal edges.

4.7 FIR Filter

Just like DF9IC [7] and the G3RUH designs, this modem too uses a four fold oversampling FIR filter of length 32. The oversampling simplifies the analog transmitter filter. Unlike the aforementioned modems, this modem does not use a table with precomputed filter values, instead it calculates the filter output for each cycle from the input data and the filter coefficients. This architecture suits the FPGA better and an external EPROM with the filter table can be omitted.

Since the modem uses an internal clock which is 16 times faster than the transmit clock, but the filter is supposed to be four fold oversampled, only 4 clocks remain to calculate the filter value. Fortunately, out of the 32 input values, 24 are zero thanks to the oversampling, and the other 8 are either $+1$ or -1. Therefore, one needs to perform only 8 adds to implement the filter. Since there are only 4 cycles, the filter is split into two halves with a dedicated final adder adding the outputs of both halves. Figure 5 shows the circuit.

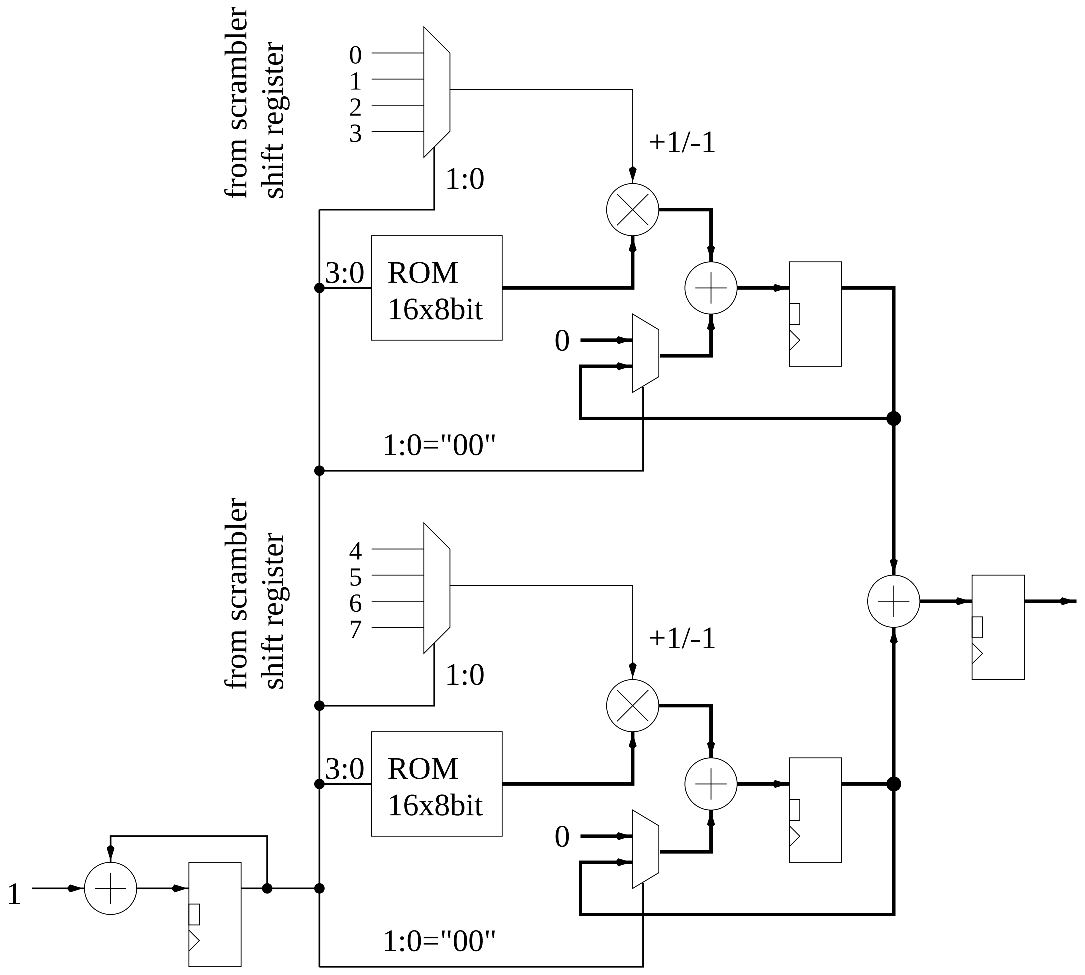

Figure 5: Transmitter FIR filter

4.8 Clock generator

The clock generator divides the frequency of the external oscillator by a variable divisor in the range 1–1024 and feeds the internal modem. The maximum clock frequency of the FPGA is 20 MHz.

4.9 Analog Circuitry

The analog circuitry was mostly taken from an existing modem [6]. The frequency determining components have been scaled to the intended bit rate (76.8kBit/s) and the DAC has been replaced by a resistor array for cost reasons.

5 Conclusion

Together with the transceiver in [9], this adapter provides cost effective high speed access to the fast backbone currently being built in central europe. While there are more bandwidth efficient solutions, they are usually considerably more expensive. The design fits onto a two layer PCB in half eurocard format (10cm $\times$ 8cm) using standard through hole components. The kit can therefore be built by less experienced amateurs. Kits and ready-made devices should be available by the time of publishing from Baycom [8].

Since the redesigned adapter is mostly compatible to the earlier design, there are already drivers available for major operating systems, such as a FlexNet driver for DOS/Windows95/Windows98 and a Linux driver.

The design is quite flexible. Although it has an internal modem, it can still bypass the internal one to accomodate an external one should the need arise. It also provides enough headroom to increase its bit rate if necessary.

6 Outlook

The prototype has been successfully tested running at 1MBit/s. The CPU overhead however was considerable, 25% measured using a 100 MHz Pentium computer. The bigger part of the overhead comes from the IO.

This could likely be changed by using ECP instead of EPP. Enhanced Capabilities Port (ECP) is an alternative parallel port protocol initially designed by Microsoft, but now also standardized by IEEE [2]. All parallel port implementations these days also support ECP. In ECP mode the controllers implement a FIFO to decouple CPU and parallel port, and they may use DMA to transfer the data between FIFO and main memory. Data transfer could therefore be offloaded from the CPU to the DMA controller.

The downside is that the ECP protocol is considerably more complex than the EPP protocol, especially data transfer direction changes are expensive. Also, the Microsoft reference implementation, which just about everyone copied, has additional restrictions not inherent to the protocol.

References

[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] Wolf-Henning Rech, DF9IC, *Modernes FSK-Modem – kompatibel zum Standard nach G3RUH*, Adacom Magazin 2, 1991

[8] Baycom $\underline{\diamond}$ Hard- und Software GmbH, Bert-Brecht-Weg 28, D–30890 Barsinghausen, phone ++49 (5105) 585 050, fax ++49 (5105) 585 060, `http://www.baycom.de/`, Email: `baycom@baycom.de`

[9] Martin Liebeck, DL2ZBN, und Alexander Kurpiers, DL8AAU, *Hochgeschwindigkeits-Packet-Radio – Baugruppen für das 70cm Band* Adacom Magazin 10, 1997

[10] *Standard Microsystems Corporation (SMSC) Personal Computer Input/Output Products* `http://www.smsc.com/main/catalog/pcio.html`

[11] *National Semiconductor Product Catalog: SuperI/O* `http://www.national.com/catalog/PersonalComputing_SuperIO_5Volt.html`

[12] *Winbond Serial and Parallel Port Controller* `http://www.winbond.com.tw/produ/perso5.htm`

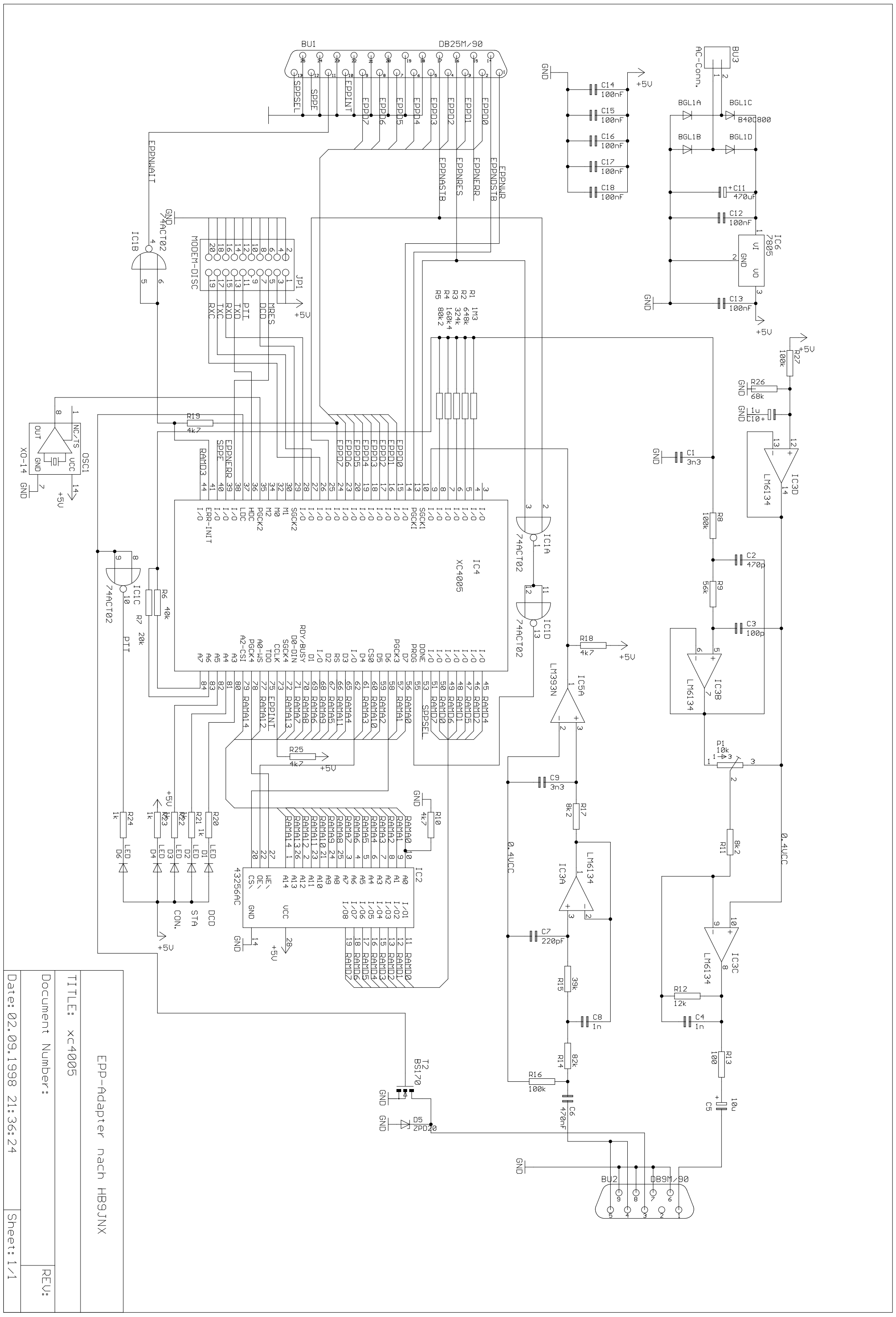

Figure 6: Circuit Diagram