A simple TNC for megabit packet-radio links

Matjaz Vidmar, S53MV

1. Computer interfaces for packet-radio

Computers were essential parts of packet-radio equipment right from its beginning more than two decades ago. Since at that time computers were not easily available and were much less capable than today, most amateurs started their activity on packet-radio with an old ASCII terminal. The ASCII terminal required an interface called TNC (Terminal Node Controller). The TNC interface lead to a standardization of the protocol used and to a worldwide acceptance of the AX.25 standard.

Today there are many different interfaces called TNC. The most popular is the TNC2, originally developed by TAPR (Tucson Area Packet Radio) and afterwards cloned elsewhere. Lots of software was written for the TNC2 too, ranging from simple terminal interfaces to complex computer interfaces and even network nodes.

As more powerful computers became available, some functions of the TNC were no longer required. In fact, some early TNC software, designed to work with dumb ASCII terminals, represented a bottleneck for efficient computer file transfer or multi-connect operation. Most functions of the TNC were therefore transferred to the host computer using the simple KISS protocol, originally developed for TCPIP operation only. Unfortunately, the KISS protocol adds additional delays in any packet-radio connection.

Today most computers allow a direct steering of a radio modem up to about 10kbit/s, making the TNC completely unnecessary. For higher speeds, different interface cards were developed. These cards are plugged directly into the ISA bus of IBM PC clones to avoid the delays and other problems caused by external interfaces.

The development of new TNCs and related software almost stopped. Although there is lots of software available for the popular TNC2 or its clones, most of the software is of very poor quality. For medium-speed operation at 38400 or 76800bps, even the simple KISS software is unreliable. The popular TNC2 will not go much beyond 76800bps, while more recent TNCs lack software support.

The first megabit PSK transceivers [3], [5], [6] were intended for packet-radio node interconnects [1]. Their price and complexity was relatively unimportant compared to the reliability requirements and radio-range performance. For megabit operation, the packet-radio nodes had to be equipped with Direct-Memory Access (DMA) controllers and efficient software. Besides our Slovenian "SuperVozelj" nodes and the similar Italian "Itanet" project, most packet-radio node hardware and software was unable to operate at megabit speeds. In most parts of Europe, the FlexNet network is

effectively blocked by the inefficient 9600bps FSK modems.

The Zero-IF technology made simple and efficient megabit PSK radios very affordable even in the form of NO-TUNE projects [2], [4], [7], [8], [9]. The next step was to develop a simple user interface. Marko Kovacevic, S57MMK, modified and improved the popular Canadian "PI" packet interface card [10]. The S57MMK card includes a Z8530 SCC serial interface chip, plugs into an ISA slot in a PC-computer bus and uses the PC DMA for fast data transfers. Strictly technically speaking, a DMA interface card represents the best technical solution for megabit packet-radio.

Unfortunately, an ISA card can not be plugged into any computer bus. Some computers even do not have ISA slots, like laptops. The operation of some computers is unreliable on its own even without additional hardware. Poorly-written software and some other peripheral devices may not tolerate a packet-radio interface card on the same bus. An additional interface card, regardless of how well it is designed and built, may cause problems with incompatible hardware and software.

Since new computers are equipped with faster, incompatible and difficult-to-use buses, it is increasingly more difficult to design packet-radio interface cards that plug directly into the computer bus. TNCs in different forms are therefore coming back, using one of the standard ports available on any computer like RS232, printer port, Ethernet connector or even USB. Since any such interface adds additional delays, using a TNC is not the best technical solution.

Unfortunately, a TNC has to be used in all cases when a plug-in interface card on the computer bus can not be used. Therefore suitable TNCs have to be developed. In this article I am going to describe a very simple TNC that connects to the RS232 port on the computer. The TNC allows the operation with the described 1.2Mbit/s PSK radios [2], [4], [7], [8], [9], but on most computers the real bottleneck is the 115.2kbit/s RS232.

2. Megabit TNC design

Megabit serial interfaces usually require DMA hardware for fast and efficient data transfers. DMA controllers are therefore used in the "SuperVozelj" [1] and in other high-speed packet-radio node projects. Besides adding complexity, a DMA controller also requires a critical handshake with the CPU, since the CPU and the DMA must be synchronized to access the same bus.

The development of high-speed serial interfaces is slowly moving away from dedicated hardware like DMA controllers. For example, the MC68360 communications processor includes four fast serial ports with DMA data transfers that are internally built with a single RISC processor. The actual protocol as well as data transfer to or from the CPU bus is selected by simply changing the software running on the RISC processor.

In the case of amateur packet-radio, most AX.25 frames are relatively short. AX.25 therefore requires lots of processing for moderate data transfers. The required CPU

processing power usually exceeds that of a DMA controller. A detailed analysis shows that a DMA controller may be completely unnecessary even for megabit AX.25 terminals.

A simple solution was therefore sought for a megabit TNC, including the selection of easy-to-get components. A careful analysis of the instruction set and required clock cycles of the popular 68000 microprocessor family has shown that megabit transfers from or to the well-known Z8530 serial-communications controller can be handled by interrupts, provided that a few registers are dedicated permanently to the interrupt-service routines.

For the first experiments I modified an old "SuperVozelj" CPU card with a MC68010 processor. The experiments have shown that for simplex 1.2288Mbit/s operation with PSK radios [2], [4], [7], [8], [9], a minimum clock speed of about 15MHz is required, in good agreement with the estimates of the interrupt-routine clock cycles. A completely new TNC CPU board was then designed around up-to-date CMOS parts like the MC68HC000 CPU and Z85C30 SCC.

The wiring diagram of the CPU, memory and serial port is shown on Fig.1. Just like the "SuperVozelj" node, the 16-bit CPU in the TNC boots from a single, cheap and relatively slow 8-bit 27C256 EPROM. The 16-bit instructions are read from the 8-bit EPROM in two steps, while the upper 8-bits of the instruction are temporarily stored in the 74HC374 latch. The content of the EPROM is copied to the 16-bit-wide RAM immediately after reset to allow fast instruction fetching without any wait states.

The Z85C30 serial-communications controller is connected to the lower 8 bits of the data bus. Simple and fast interrupt routines are triggered by the two outputs: /REQA (radio channel to INT3) and /REQB (RS232 interface to INT1). The latter are usually used for DMA transfers. The /INT output of the Z83C30 is not used, since the latter includes many different interrupt sources, unnecessarily slowing down the interrupt-service routine or in the worst case delaying an important high-speed data transfer because of an unimportant status interrupt.

Since the /INT output of the Z85C30 is not used, the interrupt vector for the MC68HC000 has to be provided externally. The MC68HC000 is able to generate interrupt "autovectors" on its own by asserting /VPA during an interrupt acknowledge, however this may add up to 15 additional wait states for every interrupt! The interrupt vector is therefore provided by the 74HC244 in exactly the same way as the /VPA, but without additional wait states.

The MC68HC000 microprocessor, different memory chips and Z85C30 SCC require some bus interface logic as shown on Fig.2. To avoid metastable problems, all clocks are derived from a single crystal oscillator. The crystal oscillator supplies the MC68HC000 clock and wait-state generator directly. The same clock is divided by 2 by the first half of the 74HC74 to obtain the Z85C30 PCLK.

The 74HC138 is used to decode the upper address lines. The 2N2369 transistor is used to protect the CMOS-RAM content while the +5V power supply is turned off and during reset.

Of course, the CMOS RAM receives a continuous supply voltage +CMOS at all times from a backup battery. The /RESET signal is first buffered (1/4 74HC08) and is supplied to the microprocessor (pins /RESET and /HALT) and Z85C30 (both /SCCWR and /SCCRD low at the same time).

Unlike the more powerful members of the MC68k family, the MC68HC000 does not have a vector-base register (VBR). The exception-(interrupt)-vector table is therefore always located at the beginning of its address space. The interface logic is therefore used to switch the beginning of the address space from EPROM to RAM. At reset, the EPROM exception-vector table is used. During normal startup of the TNC, the EPROM table is replaced by the RAM table. The RAM table can be modified as well as accessed without wait states.

The changeover from EPROM to RAM is handled by the second half of the 74HC74. This flip-flop is reset by the /RESET signal to enable the EPROM access. The microprocessor then copies the EPROM content into the RAM. After the RAM contains valid software, the flip-flop is set by the microprocessor. The EPROM is removed from the microprocessor's address space and can no longer be accessed by the software. The EPROM can only be accessed again by applying an external /RESET signal.

The switching EPROM/RAM is controlled by the signals ROM and /ROM. The ROM signal also enables the wait-state generator with the 74HC164 shift register. Besides slowing down the CPU, the 74HC164 also steers the upper 8 bits of the instruction into the 74HC374 latch and increments the lowest (A0) address of the EPROM. After the access to the EPROM is disabled, the wait-state generator is disabled too and the /AS signal is fed directly to /DTACK through two gates of the 74HC02. The /DTACK signal is generated in all cases, while /BERR input to the MC68HC000 is not used.

The MC68HC000 bus control signals also need some decoding to steer popular memory chips and peripheral devices: the R//W signal has to be split into two independent /WR and /RD signals. 74HC10 and 74HC32 gates are used for this purpose. The signals /UWR, /LWR, /UWR and /LRD control the access to the RAM, the signals /SCCWR and /SCCRD steer the Z85C30 and the signal /ROMOE enables the EPROM.

3. Practical TNC construction

In the practical construction I decided to build the TNC as a series of modules that can be connected together as required. The power supply, radio modem and RS232 interfaces are therefore built as separate modules on their own printed-circuit boards. The TNC printed-circuit board therefore only includes the CPU, memory and serial port. In this way the same TNC board can also be used for other purposes, for example with different modems or as a simple two-channel packet-radio node.

The CPU, memory, serial port and bus interface logic are located on a double-sided printed circuit board with the dimensions of 120mmX120mm. Both sides of the printed-circuit board are shown on Fig.3. The corresponding component location is shown on Fig.4. The module only has two connectors.

A four-pin connector is used for the power supply: main +5V supply, battery-backed +CMOS, /RESET signal and ground.

Both channels of the Z85C30 serial port are available on a 20-pin connector together with +5V, +CMOS and ground. Besides serial data inputs and outputs, the control lines /DCD, /CTS, /RTS and /DTR as well as both clocks RTXC and TRXC are available for both channels. These allow to connect the serial port to different modems and/or other interfaces.

The MC68HC000 microprocessor is usually available in a 68-pin PLCC package and is generally used in GPS receivers, fast hard-disk controllers and other applications where the the processing power of a simple 8-bit microcontroller is not sufficient. Since the soldering of a PLCC package is not a simple operation, the printed-circuit board is designed for a standard PLCC socket that spreads out the PLCC pins to a comfortable 2.54mm (0.1") grid.

The pins of the 68-pin PLCC socket are arranged in two rows on each of the four sides of the square socket. The socket pin-out is therefore somewhat different from the bare PLCC package as shown on Fig.5. The MC68HC000 microprocessor has two unused pins labeled "NC" in the 68-pin PLCC version. The microprocessor should be carefully pushed in the PLCC socket. On the other hand, a special tool is required to extract the microprocessor from the socket. Special tools can be avoided if the PLCC socket is further inserted in a 68-pin PGA socket (large version 11X11) that is finally soldered into the printed-circuit board.

While building the megabit TNC, it makes sense to install all integrated circuits on good-quality sockets. The TNC will work with 128kbyte RAM chips (628128) or with 32kbyte RAM chips (62256). In the latter case, the 28-pin 32kbyte RAM chips should be inserted in the bottom part of the 32-pin sockets so that pin 1 of the RAM corresponds to pin 3 of the socket. The TNC will also work with 32kbyte, 64kbyte or 128kbyte "cache" RAM chips from old "486" motherboards. Since these RAMs are packaged in narrower packages, their pins should be spread out or a suitable adapter should be built to fit in the wider standard RAM socket.

The relatively short bus connections allow high clock speeds for the TNC CPU. Experiments have shown that the maximum clock speed does not depend much on the CPU used, but rather on the RAM access time. A 10MHz version of the MC68HC000 will work with 70ns RAMs up to 33MHz, while a 16MHz version of the MC68HC000 will reach 40MHz with 20ns "cache" RAMs. The CMOS version of the SCC chip Z85C30 also allows much higher clock frequencies than specified: a 8MHz version usually works perfectly at 15MHz PCLK (CPU at 30MHz).

Since all tested combinations of CPUs and memories always achieved a maximum clock frequency of at least 25MHz, a CPU clock of at least 20MHz is recommended for the megabit TNC. The exact clock frequency also depends on the usable division modulos inside the Z85C30 to obtain standard data rates. If the 1.2288MHz clock on the radio side comes from the scrambler module [2], [4], [8], [9], then the Z85C30 dividers only supply the RS232 baud-rate.

A RS232 baud-rate of 115.2kbit/s can be obtained from a

CPU clock frequency of 14.7456MHz, 22.1184MHz or 29.4912MHz. Although the TNC may work at 14.7MHz, this clock is too low for reliable operation at 1.2288Mbit/s on the radio side. Therefore 22.1MHz or 29.4MHz are recommended. Of course, any clock frequency can be used for the CPU if an external clock of 1.8432MHz is supplied to RTXCB. While using the TNC for lower kilobit data rates on the radio port, there is much more freedom in choosing the clock frequency.

4. Radio and RS232 interfaces

The megabit TNC is designed to work with different modems. The most common combination is a scrambler/ bit-synchronizer on channel A and a RS232 interface on channel B od the serial-communications controller Z85C30. The scrambler/bit-synchronizer [2], [4], [8], [9] includes many modem functions like clock recovery and descrambling for the received data and clock generation and data scrambling for the transmitted data.

The scrambler is connected to the Z85C30 channel A with five signal wires plus ground. The signals include: received data (RXD), clock (connected to RTXC), carrier detect (/DCD), PTT command (/RTS) and transmitted data (/TXD). In this case the input /CTS, output /DTR and clock TRXC are not used on channel A. An unused CMOS input should be connected to ground or to any other signal.

Some other modems for lower kilobit rates may require internal clock recovery inside the Z85C30. In this case, the modem is connected with just four wires besides ground: received data (RXD), carrier detect (/DCD), PTT command (/RTS) and transmitted data (TXD). The TNC software programs the Z85C30 to supply the regenerated clock from the internal DPLL on the TRXC output. An external jumper is required to bring this clock to RTXC. Of course, the correct divider modulo should be set by the software.

If more than one modem is installed in the TNC, then the channel A inputs RXD, /DCD and RTXC have to be switched among the different modems and/or other clock sources. All of the modems should receive their supply voltage at all times so that it is not necessary to switch the channel A outputs /RTS and TXD.

A TNC is usually connected to a host computer through a RS232 interface. Unfortunately, the latter represents a real bottleneck for a megabit TNC, since the highest RS232 baud-rate is 115.2kbit/s on most computers. The RS232 interface also includes a polarity inversion on all signals and a logic-level shift to higher positive and negative voltages.

A very simple RS232 interface can be built with the integrated circuit MAX232 as shown on Fig.6. The MAX232 includes two RS232 transmitters, two RS232 receivers and two DC/DC converters to obtain +10V and -10V from a single +5V supply. Unfortunately, the MAX232 also includes low-pass filters on both transmitters to limit radio interference. These low-pass filters also limit the available data rate to about 150kbit/s.

The RS232 interface is built on a small, single-sided

printed-circuit board with the dimensions of 54mmX44mm, as shown on Fig.7. The printed-circuit board is supported by an angled female D52 connector so that additional mounting screws are not required. The corresponding component location is shown on Fig.8. The female D25 connector is wired to fit the male RS232 D25 connector on PC COM ports.

While using the RS232 interface on channel B of the Z85C30, the /DCD input and /DTR output remain unused. An unused CMOS input should be connected to ground or to any other signal. If the internal clock source in the Z85C30 is used, then a jumper from TRXC to RTXC is required. An external clock source can also be connected to the RTXC input. In the latter case, the external clock frequency should equal 16-times the desired baudrate or 1.8432MHz for 115.2kbit/s.

5. Power supply and reset circuit

The described TNC is intended to be operated from a 12V power supply with the negative grounded like most radio-amateur equipment. Of course, the digital circuits of the TNC require a +5V supply. If any program parameters are stored in the CMOS RAM, then the latter requires a battery-backed supply +CMOS. Finally, the TNC requires a reset circuit that operates reliably regardless of the power-up or power-down sequences.

The circuit diagram of the power supply and reset circuit is shown on Fig.9. A simple switching regulator is used to obtain +5V from the available +12V external supply. The switching regulator achieves an efficiency of about 80%, thus halving the power consumption and heat generation when compared to a conventional linear regulator. The +CMOS supply is backed by a 3.6V 60mAh NiCd battery. An additional protection of the CMOS RAM content is provided by the /RESET signal by disabling the RAM chip select.

The reset circuit senses the input voltage to the power supply. The /RESET signal is only released after the input voltage exceeds 9.5V. An additional delay is generated by the 1.2kohmX470uF RC time constant. At power down, the 470uF capacitor is discharged immediately through the 1N4001 diode, so that the /RESET signal is applied before the switching regulator stops operating. Finally, the reset circuit includes a small hysteresis (47kohm resistor) to generate a clean /RESET signal on its output.

The TNC power supply is built on a single-sided printed-circuit board with the dimensions of 75mmX45mm, as shown on Fig.10. The corresponding component location is shown on Fig.11. Several mounting pads are provided for different-style NiCd batteries.

The total current drain of the described TNC equipped with a scrambler/bit-synchronizer and RS232 interface amounts to about 200mA at +5V or about 100mA at +12V with the described power supply. At these current levels, the BD138 switching transistor does not require a heat sink. On the other hand, the 100uH chokes must be able to handle these currents too: chokes of the size of 1/2W resistors or even larger should be used in the power supply.

6. Megabit TNC software

The TNC software should allow a fast and reliable data transfer between the TNC and the host computer or another terminal. The software of the first TNCs was intended to be used with dumb ASCII terminals. The RS232 communication was therefore designed for direct typing on the keyboard and readout on the CRT screen. Of course, such a simple interface is not the best solution for computer file transfer or multi-connect operation.

WA8DED attempted to improve the communication between the TNC and the host computer with his HOSTMODE protocol. In the HOSTMODE protocol, the host computer continuously polls the TNC to get new information. This requires the RS232 interface being much faster than the radio interface, making the HOSTMODE protocol useless for faster radio links.

The WA8DED software was cloned elsewhere. Maybe the best-known clones are those from NordLink. Unfortunately, no one removed some important bugs from the original WA8DED code that corrupt the data and crash the TNC software or the host computer. Even worse, the same bugs were also transferred to the software that emulates a HOSTMODE TNC on a PC computer (TFPCX, TFKISS etc). For this reason, the HOSTMODE protocol is now almost forgotten.

The only RS232 protocol used by most TNCs seems to be the KISS (Keep It Simple Stupid) protocol. The KISS protocol simply transfers the same, unprocessed AX.25 frames over the RS232 interface. The AX.25 protocol is therefore handled by the host computer except for a few timing functions tightly connected to the radio modem, like receive/transmit changeover.

The KISS protocol transmits the data as 8-bit bytes on an asynchronous serial interface. The beginning and the end of AX.25 frames are marked with FEND ($C0) characters. When a FEND character appears inside an AX.25 frame, it is replaced by the sequence FESC, TFEND ($DB, $DC). The special character FESC also requires a replacement if appearing inside an AX.25 frame. FESC is replaced by FESC, TFESC ($DB, $DD). By definition, the KISS protocol does not make any use of the RS232 control lines (CTS, RTS etc).

The KISS frames are equipped with an additional control byte in front of the AX.25 frame. The control byte allows the selection of up to 16 radio ports on a multi-port TNC (upper four bits) and setting a few parameters of each radio channel (lower four bits). An AX.25 frame is identified by the lower four bits set to zero, while a value different from zero sets some TNC parameters. An exception is represented by the control byte $FF that is usually used to switch the TNC (back) to another mode of operation.

Most KISS frames have the first (control) byte set to zero ($00), since we only have single-channel TNCs and most KISS frames carry AX.25 frames. Setting the TNC parameters TX delay, TX tail, ppersistance and slottime and selecting simplex or duplex operation as well as quitting the KISS mode of operation with special control bytes proved to be a poor solution in practice. Any errors on the RS232 interface may program the TNC in some useless operating mode and crash

the AX.25 link.

Although the RS232 link includes just a short length of cable between the TNC and the host computer, transmission data errors occur frequently on the RS232 interface. The main cause is the poor design of PC interrupts. A lower-priority interrupt may stop the interrupt-driven RS232 data transfer leading to loss of data. Serial ports with FIFO registers may save some data, but they are still unable to guarantee an error-free data transmission.

RS232 errors can be tolerated in TCPIP operation, since all TCPIP frames include their own, internal CRC checksum. Of course, RS232 errors corrupt the data in conventional AX.25 contacts. Special KISS protocols including a two-byte CRC at the end of the frame, just like in the HDLC AX.25 frames, have been developed to detect and automatically reject corrupted frames.

Of course, both the TNC software and the host-computer software should detect automatically what kind of KISS protocol is being used: the last two bytes in a frame may be a 16-bit CRC computed in different ways or simply two valid data bytes. SMACK (Stuttgarts Modifiziertes Amaturfunk-CRC-KISS) is a KISS protocol with two CRC bytes at the end of each frame. The SMACK frames are marked with a $80 control byte at the beginning of each frame, since there are no TNCs available with more than 8 radio ports.

SMACK is using the same CRC polynomial-division algorithm as conventional AX.25 HDLC frames, except that the polynomial generator is started from an all-zero condition. The latter is a poor choice and may not detect some types of errors on the RS232 interface. The FlexNet group corrected this problem by developing their own CRC algorithm. KISS frames with a FlexNet-CRC are marked with a $20 control byte at the beginning of each frame, since multi-port TNCs are no longer being used.

The requirements for the TNC software are exactly defined: the TNC should communicate in the conventional-KISS protocol without CRC as well as in both CRC versions, SMACK and FlexNet-CRC. All three KISS protocols are being used by almost all available packet-radio software running on host computers. The critical timing and operational parameters of the TNC are best burned once forever in the TNC EPROM rather than being modified by RS232 errors in an uncontrolled way.

The megabit TNC has some additional requirements. Since the radio interface is an order of magnitude faster than the maximum RS232 speed, the TNC should check the call signs of all frames and reject useless frames addressed to other stations. An even better solution is a small packet-network node to handle the retries on both the radio and RS232 sides.

The described TNC was first tested with simple KISS software including both CRC variants. Besides the currently-available KISS/SMACK/FlexNet software, many other software upgrades are planned for the described TNC, including a small packet-radio node similar to the SuperVozelj nodes. All software is available as documented 68k ASM source as well as compiled EPROM files on our packet-radio mailbox S50BOX.

The current KISS software supports a simple interface

to adjust the important program parameters like filter call sign, TX delay, TX tail, ppersistance and KISS protocol (no-CRC, SMACK or FlexNet). These parameters are simply typed in the "unproto" mode and transmitted as UI (beacon) frames to the callsign "TNC". The TNC will answer with UI frames too, showing the current parameter settings. This simple protocol resulted very reliable and at the same time accessible at any time from any packet-radio terminal program.

The current software version copies the default parameters from the EPROM after every reset and does not make any use of the CMOS battery. However, future versions will probably use the nonvolatile CMOS RAM at least to store the program parameters and call sign.

There are also a few restrictions imposed by the hardware simplicity of the described megabit TNC. Three address registers A4, A5, A6 and two data registers D6 and D7 are used at all times by the high-speed interrupt routines, so they can not be used by other TNC software. The radio-port interrupts are designed for simplex operation only.

7. Megabit TNC applications

The described megabit TNC was initially intended to prove that a simple circuit can perform much better than complex and expensive hardware. In particular, complicated DMA circuits and other interfaces are probably not required for high-speed packet-radio. The described megabit TNC probably shows the evolution of future packet-radio hardware: most problems will be solved with standard, inexpensive parts programmed for our purposes.

Practical experiments with the described TNC equipped with simple KISS software and FlexNet driver running on the host computer have shown an average data throughput of about 25kbit/s or 3kbyte/s in a real network with many other users active on the same channel at the same time. A small-node software in the TNC to handle separately the retries on the radio side as well as on the RS232 side could probably double this figure before reaching the 115.2kbit/s RS232 bottleneck. Some improvement could also be obtained with a better driver than FlexNet, since the timing constants of the latter are programmed for 9600bit/s modems and can not be changed.

For the packet-radio user, a megabit TNC on the RS232 interface is certainly a slower solution than a DMA card in the computer bus, mainly thanks to the 115.2kbit/s RS232 bottleneck. The megabit TNC is therefore an interesting solution only when a DMA card can not be used, like laptop computers, or due to conflicts with other hardware and/or software in the same computer. Of course, a megabit TNC equipped with an Ethernet, USB or even parallel printer-port interface could perform much better.

For the network developer or sysop, the megabit TNC may have advantages too. Most important, one single user can never get all of the capacity of a megabit channel thanks to his/her RS232 bottleneck, thus blocking the access to other users. A megabit TNC also allows testing new protocols in the network without changing the actual user terminal software running on

host computers.

A megabit TNC also represents an ideal interface to older node hardware, like the RMNC/FlexNet nodes used elsewhere in Europe. The efficient megabit PSK transceivers, although described in detail in several places: [2], [3], [4], [5], [6], [7], [8], are not widely used outside Slovenia and Italy, probably just because popular packet-radio node hardware can not operate at megabit speeds directly, without a megabit TNC interface.

Finally, the described megabit TNC equipped with CMOS parts can also work as a simple, very low-power digipeater already with the currently-available KISS software. The overall current consumption of the TNC is about 100mA at 12V. Adding about 200mA for the 23cm PSK transceiver, the whole digipeater can be powered with a 50W solar panel and a large "diesel" car battery.

References:
---

[1] Matjaz Vidmar: "1.2Mbit/s SuperVozelj packet-radio node system",
Scriptum der Vortraege, 40. Weinheimer UKW Tagung,
Weinheim, Germany, 16-17 September 1995, pages 240-252.

[2] Matjaz Vidmar: "23cm PSK Packet-radio TRX for 1.2Mbit/s user access",
Scriptum der Vortraege, 41. Weinheimer UKW Ttagung,
Weinheim, Germany, 21-22 September 1996, pages 25.1-25.15.

[3] Matjaz Vidmar: "13cm PSK Transceiver for 1.2Mbit/s Packet Radio",
15th ARRL and TAPR DIGITAL COMMUNICATIONS CONFERENCE,
Seattle, Washington, USA, September 20-22, 1996,
pages 145-175.

[4] Matjaz Vidmar: "23cm PSK Packet-Radio RTX for 1.2Mbit/s User Access",
15th ARRL and TAPR DIGITAL COMMUNICATIONS CONFERENCE,
Seattle, Washington, USA, September 20-22, 1996,
pages 176-202.

[5] Matjaz Vidmar: "13cm PSK Transceiver for 1.2Mbits/s Packet Radio, Part-1",
VHF-Communications 3/1996, pages 130-147.

[6] Matjaz Vidmar: "13cm PSK Transceiver for 1.2Mbit/s Packet Radio, Part-2 (conclusion)",
VHF-Communications 4/1996, pages 194-205.

[7] Matjaz Vidmar: "23-cm-Packet-Radio-Transceiver fuer 1.2-Mbit/s-Benutzerzugriffe, Teil 1",
AMSAT-DL Journal 3/1996, pages 42-44.

[8] Matjaz Vidmar: "Design des 23-cm-Null-ZF-PSK-Transceivers"
AMSAT-DL Journal 4/1996, pages 11-26.

[9] Matjaz Vidmar: "23cm PSK Packet Radio Transceiver for
    1.2Mbit/s User access",
    VHF-Communications 2/1997, pages 74-96.

[10] Marko Kovacevic: "PC komunikacijska kartica za hitri
    packet-radio",
    CQ ZRS 5/1997, pages 38-42.

****

List of figures:

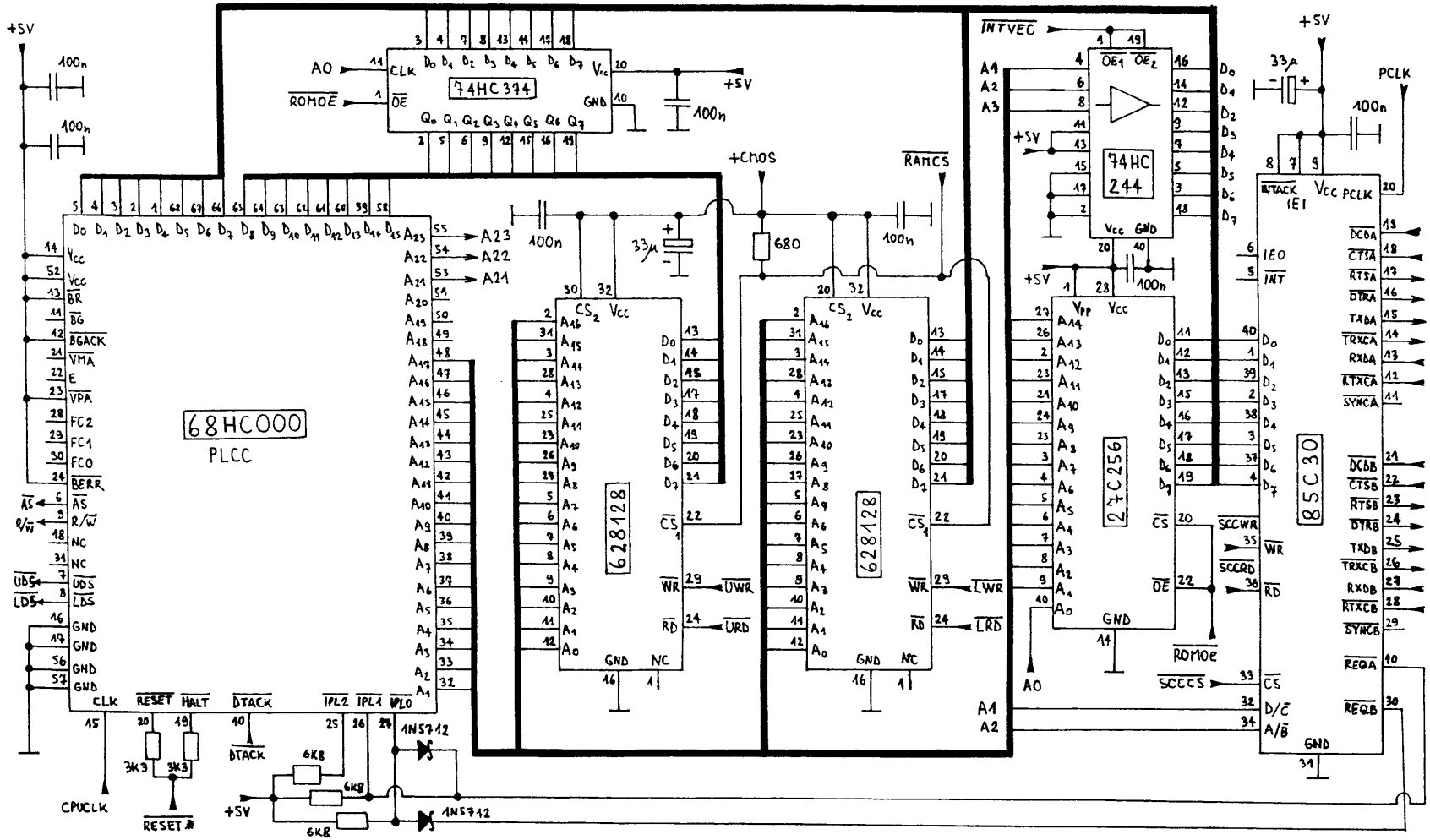

Fig. 1 – CPU, memory and serial port.

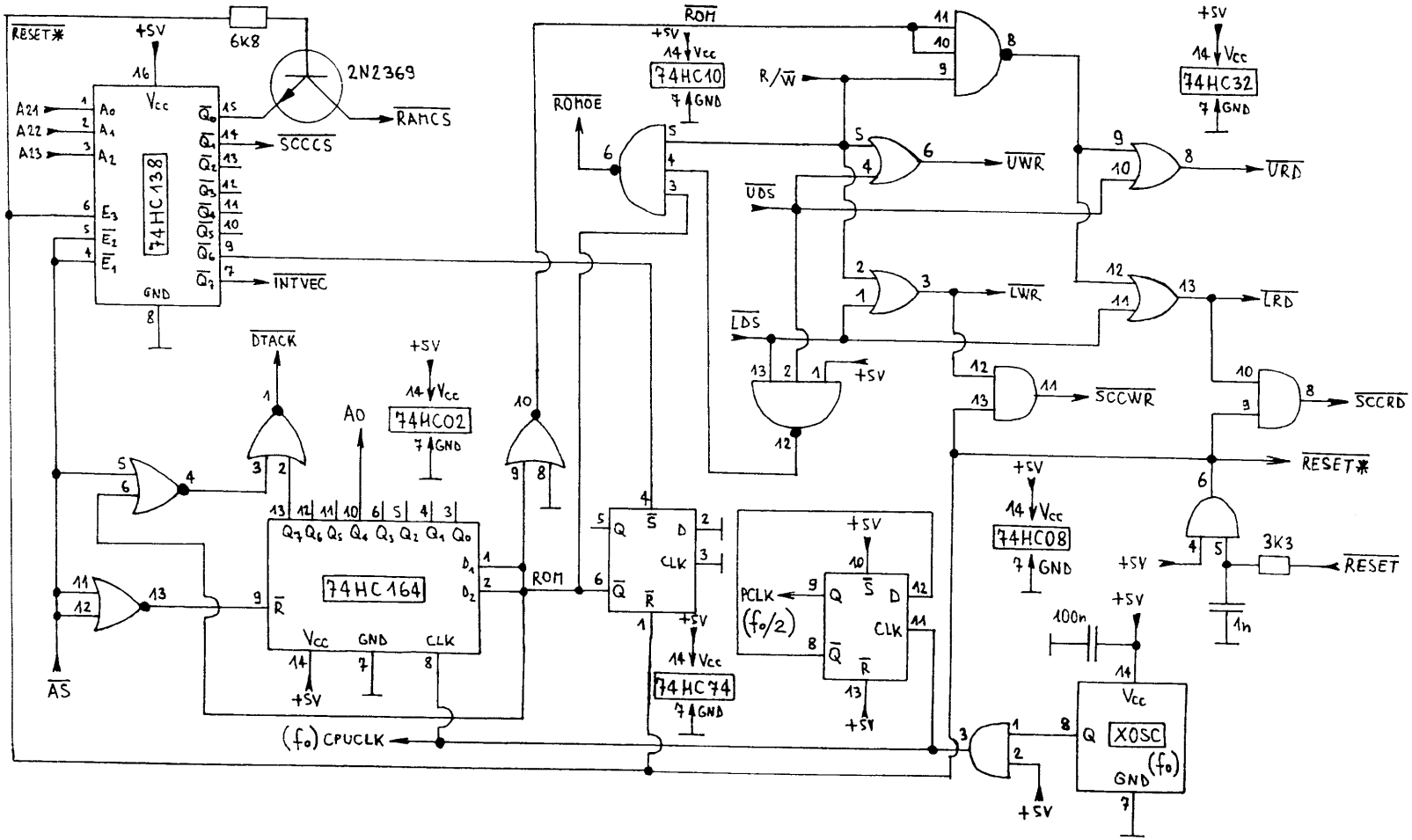

Fig. 2 - Bus interface logic.

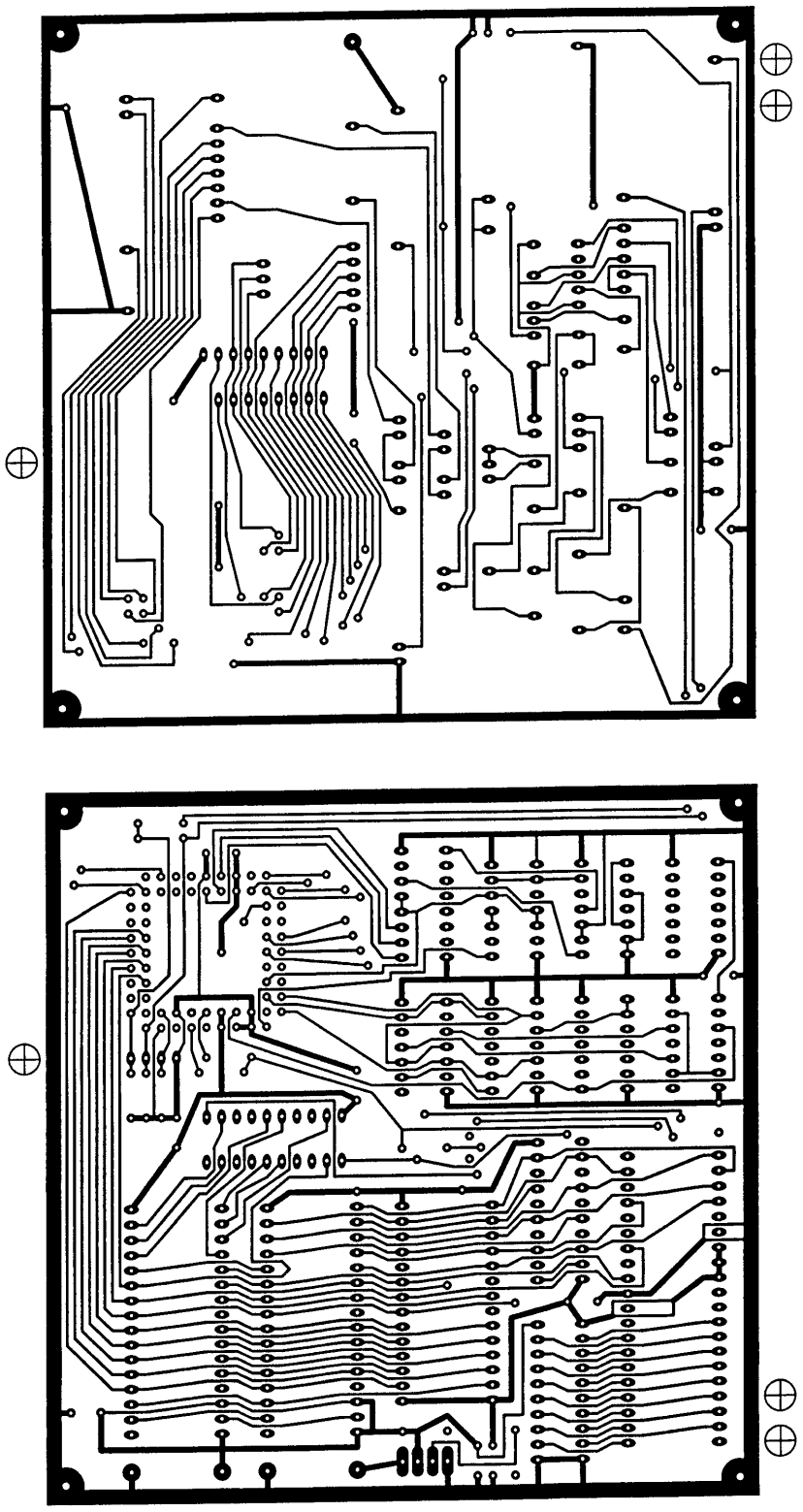

Fig. 3 — TNC double-sided printed-circuit board.

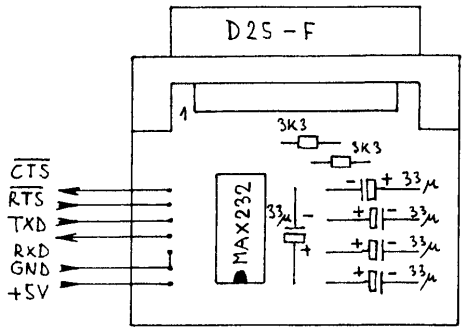

Fig. 8 - RS232 component location.

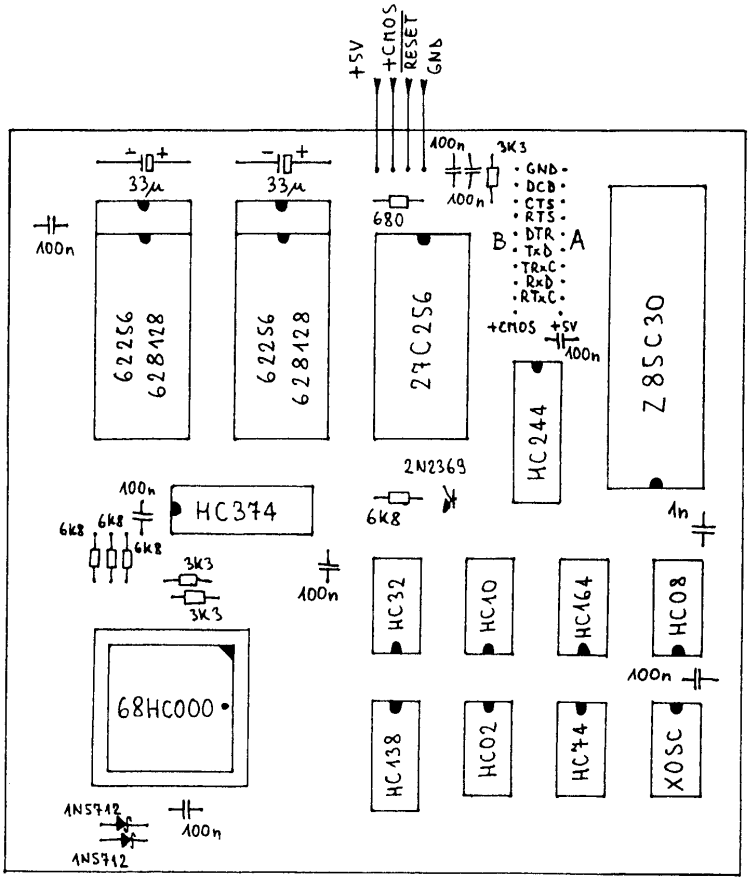

Fig. 4 - TNC component location.

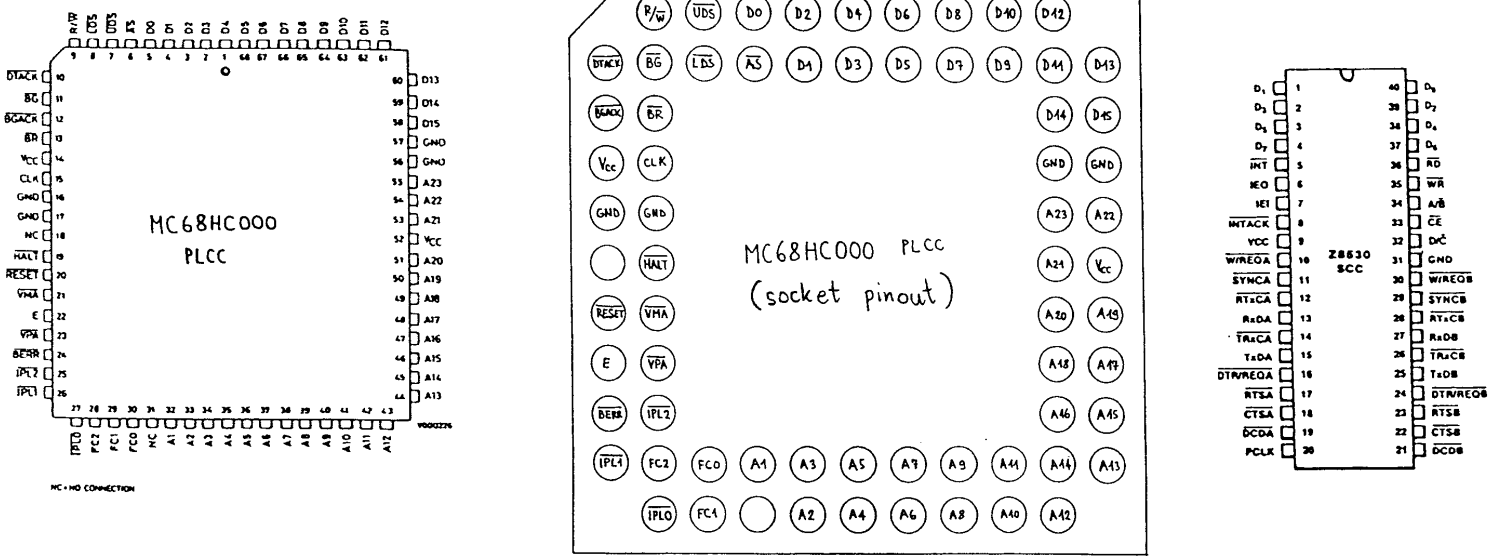

Fig. 5 - CPU and serial-port pinouts.

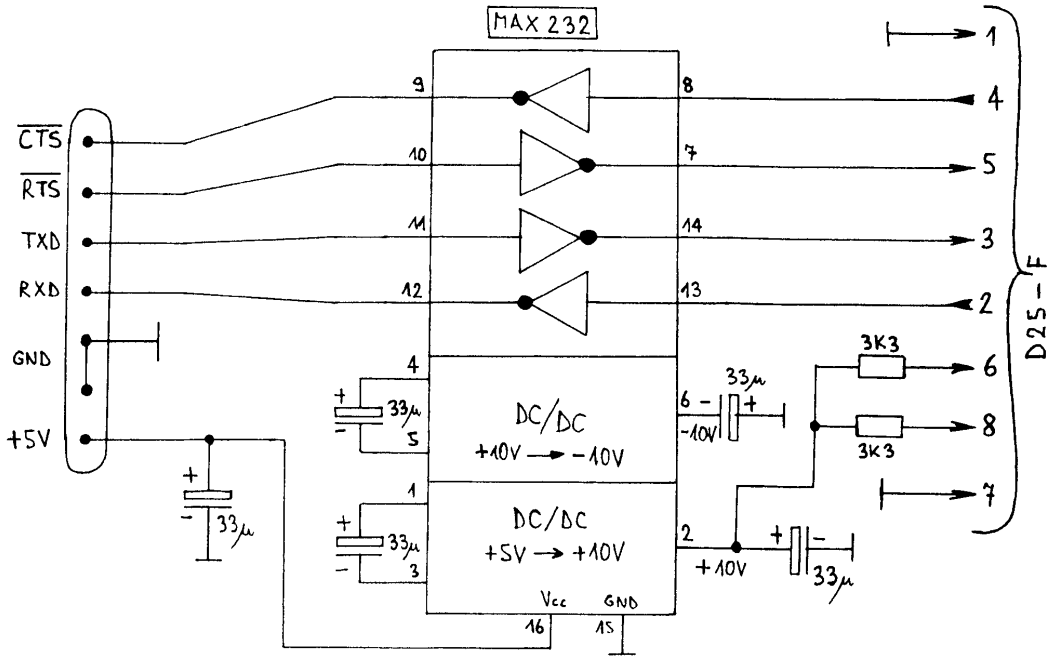

Fig. 6 - RS232 interface.

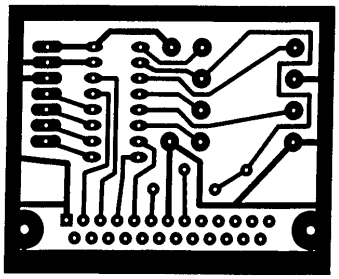

Fig. 7 - RS232 printed-circuit board.

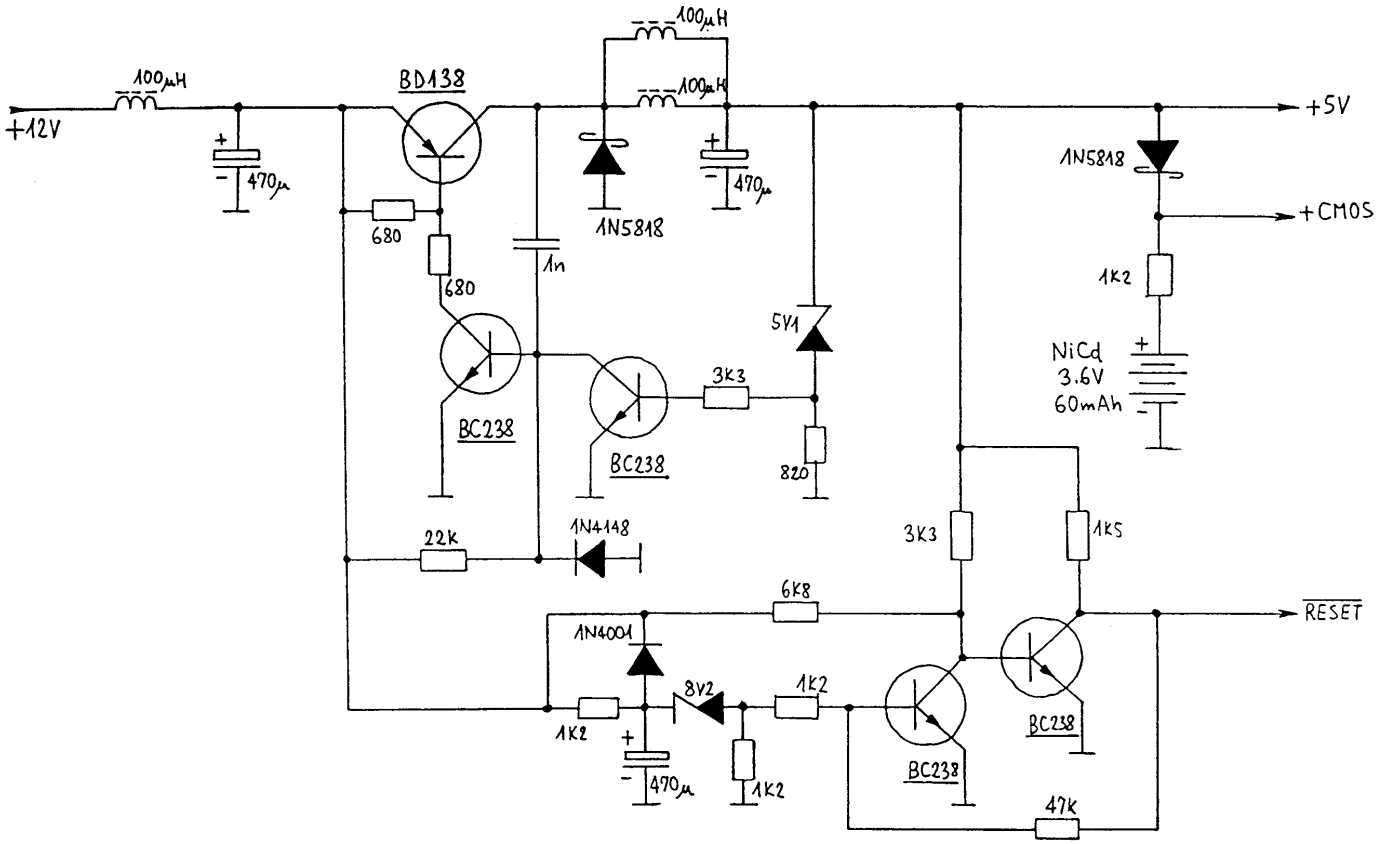

Fig. 9 - Power supply and reset circuit.

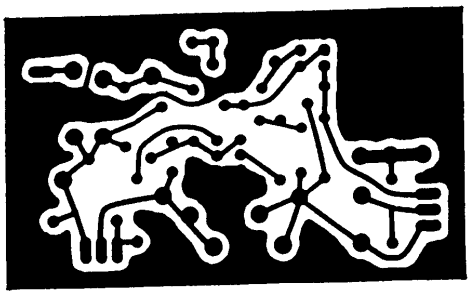

Fig. 10 - Power-supply printed-circuit board.

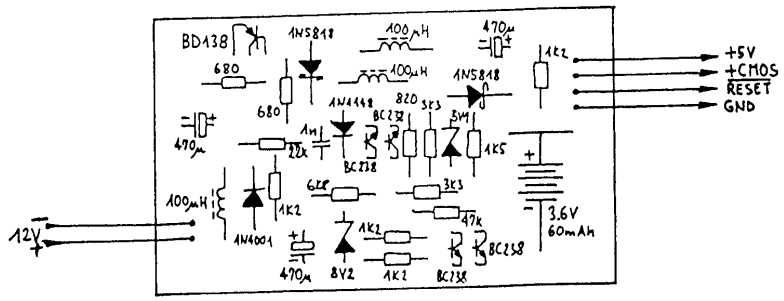

Fig. 11 - Power-supply component location.