A Totally Awesome High-Speed Packet Radio I/O Interface for the IBM PC XT/AT/386 and Macintosh II Computers

Mike Chepponis, K3MC

Bernie Mans, AA4CG

ABSTRACT

This paper describes a plug-in card for IBM PC XT/AT/386 compatible computers or the Apple Macintosh II computer. It is designed to handle two 56 kilobit/sec full-duplex channels via DMA and 10 slow speed (19,200 bits/sec or less) channels via interrupts without main processor intervention. The board uses an NEC V40 processor and up to six Zilog 85C30 Serial Communication Controllers, together with either 256k bytes of DRAM or 1 megabyte of DRAM to offload the main processor from low-level interrupt fielding. It communicates with the main processor with an 8k byte memory window in the IBM version, and directly with Macintosh II main memory using the Bus Master concept of the NuBus. It is targeted at replacing existing TNCs in the high-speed networks of the future; special consideration has been given to the support of TCP/IP. Even the minimum (easily upgradable) implementation with only one 85C30 outperforms all existing TNCs, relying on the host PC only for bootstrapping, power, and of course, an effective user interface when called for.

History

The history of packet radio is replete with ideas, some great, some not so great. One thing we did with this project was to survey the existing solutions for serial communications with a host computer, and to generate a solution that did not have any of the handicaps of previous methods.

In the beginning, folks used TNCs, usually running code that expected the user to attach a "dumb" terminal to them on one end, and to attach an AFSK 1200 baud Bell 202-compatible modem to an ordinary FM radio via the external speaker and microphone jacks.

When Phil Karn, KA9Q, wrote TCP/IP software for the IBM PC XT/AT (and it was subsequently ported to many different machines, including the Macintosh, Atari, UNIX', and Amiga), he initially used "nailed-up" AX.25 connections, that is, ordfnary TNCs in the Transparent mode. The disadvantages of such an approach was obvious to all of us, and a simplified TNC interface designed specifically for communicating with computers (the so-called "KISS" TNC[2]) was implemented. The KISS interface allowed the host computer to completely control the TNC, without the bothersome command interface optimized for human use, and it quickly became the "way to go" to use TCP/IP on the air. Still, it used 1200 baud Bell 202-compatible modems on the radio side. Indeed, the KISS TNC was never intended to be anything but a stopgap measure.

Last year, Dale Heatherington, WA4DSY, designed a reproducible 56 kilobit/sec modem[3]. With the help of Doug Drye, KD4NC and a

1 UNIX is a trademark of AT&T Bell Laboratories.

2 Chepponis, M., and Karn, P., "The KISS TNC: A simple Host-to-TNC communications protocol," *AR R L Amateur Radio Sixth Computer Networking Conference,* pp. 38-43, Redondo Beach, 29 August 1987.

3 Heatherington, D. A., "A 56 Kilobaud RF Modem," *AR R L Amateur Radio Sixth Computer Networking Conference,* pp. 68-75, Redondo Beach, 29 August 1987.

host of other Georgia Radio Amateur Packet Enthusiasts Society (GRAPES) members, modem board sets, complete schematics, board layouts and parts lists became available at very reasonable cost. Since then, GRAPES has also made a complete 56k kit available. With the introduction of this modem, Dale needed to figure out a way to connect it with some packet radio hardware and software. Since the TNC-1 could not handle 56k bits/sec, and because the existing TNC code for the TNC-2 was unavailable, Dale chose to perform substantial modifications to the KISS TNC-2 code to permit it to handle 56k data. Also, since no other networking software could handle the requirements of this data rate, Dale chose Karn's TCP/IP software, because it already worked with the KISS TNC, and needed no modifications to run at this higher speed. Because the TNC was essentially a synchronous to asynchronous converter, the 56k radio side was converted into a 19.2 kbaud serial data stream and the effective throughput was limited to only 19.2 kbaud. This is how all 56k modems (to our knowledge) still operate.

What we wanted was true 56k baud system throughput, and this mandated an auxiliary I/O processor if we wanted to use the popular IBM PC XT/AT/386 and Macintosh II hardware. To be sure, four other cards plug into the IBM XT bus, eliminating the need for a TNC: the HAPN card, the 8530-based Eagle card, the Pat-Comm PC-100 and the DRSI card. All of these cards are not appropriate for these higher data rates because none of them have DMA nor on-board CPUs.

There are two other products that should be mentioned: the TAPR NNC and the PS-186. The TAPR NNC did not take off because it was underpowered for the jobs we had intended it to do. The PS-186, on the other hand, is an excellent choice for mountain-top, solar-powered IP switches, and other uses that require low power consumption and several medium-speed channels.

Our solution is particularly cost-effective. Utilizing the very inexpensive IBM PC/XT compatible systems available today with this I/O card costing approximately an additional $200, we can provide true 56 kilobit performance at a very reasonable cost. If one compares the cost of a comparable 1200 baud station, in terms of dollars per bit per second, the 56k solution is indeed *very* cheap!

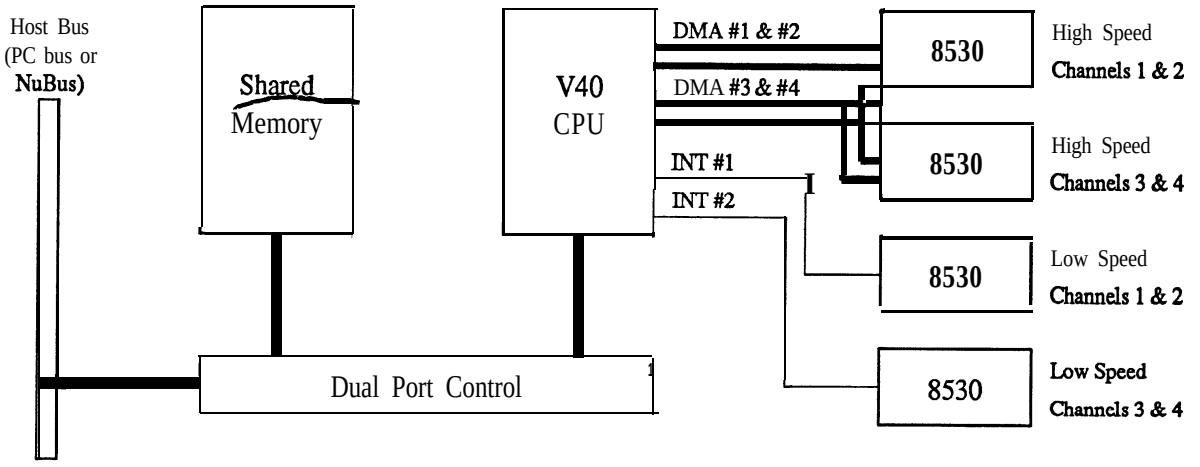
Justification

The board consists of an NEC V40 microprocessor, at least one 85C30, and at least 256k DRAM, all running at 8 MHz. The V40 provides, most importantly, instruction set compatibility with the existing IBM PC XT/AT environment (indeed, the V40 instruction set is a superset of the 8088 instruction set, and contains many instructions of the 80286 processor). It also provides four DMA channels (these four channels are used to provide DMA for the 85C30, allowing for very low processor overhead when all four channels, 2 input and 2 output, are running 56 kilobits, full duplex). There are three 16-bit counter/timers, 8 prioritized interrupts, and an NMI input. In addition, it provides DRAM refresh support. Running at 8 MHz, this chip provides ample horsepower to handle five more 85C30s, for ten more full-duplex channels, using an interrupt-driven scheme, for baud rates of 19,200 or less. An additional pair of 56k channels, full duplex, can be handled, for a total of' four full-duplex 56k channels, if we forsake the low-speed channels; in this case, two of the full-duplex channels would be the standard DMA-driven ones, and the other two full-duplex channels would be interrupt-driven.

The 85C30 is a CMOS version of the ultra-popular Zilog 8530 Serial Communications Controller chip, which is well known in the Amateur community, is highly flexible and very popular. It is used in the FAD PAD, the Eagle card, the DRSI card, the Pat-Comm TNC-220 and AEA's PK-232. With on-board digital PLL clock recovery, on-board baud rate generation, and multiple serial communications formats, it is indeed the "Swiss Army Knife" of serial communications controllers. In addition, the 85C30 can provide a single half-duplex T1 channel (1.544 megabits/se& when we have modems that run at these rates.

256k bytes of DRAM permit about 30 seconds of data buffer for a single 56k channel. When all four 56k channels are active (two input, two output), this memory provides more than 15 seconds of buffer per channel. This permits the host to be relatively lax about servicing this I/O card, and still not miss packets. It permits full utilization of the bandwidth available with the WA4DSY 56k modems.

If the (up to) ten extra low-speed channels are desired, provisions are made on the board



Simplified block diagram for the "Awesome" I/O Interface.

for installation of AMD 7910 modems, or TI 3 105 modem chips.

How it works

Basically, the card is a separate I/O processor, a computer on a board. In the case of the IBM PC XT/AT/386, an 8k byte memory window is used to communicate with the host. Several I/O ports are used for configuring the device. For example, the 8k byte memory window can be made to appear, under software control, into any available 8k byte window in the IBM PC's address space. This is done by using an I/O port with the high-order bit being a "memory enable" bit, and the remaining 7 bits used as the upper 7 bits of the address of where one wants the memory to appear.

The interface is extremely flexible. Another I/O port has a single bit which is tied directly to the V40 reset pin. Upon power-up, the 8k byte memory window is disabled, and the V40 is kept reset. The host enables the memory and places it into the address space where it is desired. Then it loads code into the 8k byte window, and then releases the reset pin by writing into the other host I/O port. At this point, the V40 is running.

With 256k bytes memory, the upper two address bits out of the V40 are ignored. This has the effect of mapping the 256k bytes into the V40's address space four times. The advantage of this is that when the V40 reset wire is held high, it jumps to location FFFFO. Since the 8k byte memory window into the host is always mapped into V40 addresses FE000 to FFFFF, it is trivial to initialize the I/O processor. Also, since the interrupt vectors for the V40 begin at location 00000, all of the interesting parts of the address space are made available without special tricks.

The 8k byte window is dual-ported with the V40 and host processor. In addition, the host always has priority when accessing this memory. It is occasionally necessary to insert wait states into the host processor's access requests, but since the V40 is mostly DMA driven by the 85C30, the host waits on the average only two T cycles, or 250ns at 8 MHz. This means, for example, that a lk byte packet can be transferred from the I/O card to the host in only 770 microseconds, on average.

In the Macintosh II the situation is even better. The NuBus, the bus used on that machine, is capable of having Bus Masters. A Bus Master can seize control of the bus, and perform data transfer operations between itself and either main memory and/or other I/O devices. One way to think of it is as a superset of DMA capabilities; perhaps one could call it "smart DMA". The Macintosh II writes the addresses into the board where to retrieve data (for transmitting) and where to deposit data (for receiving) and this board takes care of the rest! This means that the Macintosh II computer can be doing many more things, because it does not have to move memory blocks around like it needs to do in the IBM PC case. It also means that the 256k byte on-board memory buffer is not required to be quite so big, as the Macintosh II main memory can be used as the buffer.

One may ask why we didn't use DMA on the IBM PC XT/AT machines. In addition to

the limited number of DMA channels, the variable latency time of DMA servicing on the IBM machines complicates things, such that we would need a very sophisticated buffering scheme if we were to be certain that bytes were not dropped due to other I/O (especially hard disk or floppy activity) on the IBM PC's bus.

How to use it

The I/O card is particularly easy to program. We have identified two classes of programming, the lowest-level driver code and the application code, which makes use of the low-level drivers.

For the low-level code, we have kept things as general as possible (practicing the Computer Science principle of "delayed bindings") - that is, we have fixed very little about how one should program the card. We have fixed two host I/O addresses, as mentioned above, one for enabling the 8k byte memory and placing where desired in the host address space, and one for controlling the V40 reset wire. Other than that, the low-level programmer is free to define how to use the 8k byte memory window. In particular, which memory locations are used to specify which 85C30 channels, which memory locations are used to hold pointers and length-of-packet values, etc., are all flexible. Indeed, the entire structure is flexible from the low-level programmer's point of view. We do interrupt the main processor when the I/O card needs attention (and this interrupt is strappable so you can use a free interrupt of your host), such as when we've received an incoming packet or when we've finished transmitting a packet. But, given the amount of buffering we've provided, the host need not respond instantaneously to this interrupt. In addition, how the interrupt's reason is communicated to the host via the memory window is completely left up to the low-level programmer, enhancing flexibility.

For the applications programmer, three packages are available for the I/O Toolkit: one for initializing the V40, one for receiving a packet and one for transmitting a packet. These packages interface with both Aztec C and Turbo C for the IBM PC, and with Lightspeed C and MPW C for the Macintosh II. Of course, complete source code is provided with the Toolkit routines".

Applications

We have concentrated on the TCP/IP use in "standard" packet radio, but here are some other things we are planning to do with these cards.

The first thing is to build a complete 56k network node, based on Phil Karn's proposal[5]. Due to the stinging loss of 220 to 222 MHz, our options are more limited, but nevertheless, we intend to build a three-port IP switch, with channels on .43, .902 and 1.2 GHz. Two of these frequencies would be receive-only frequencies, and the other would be a transmit-only frequency, for one of the switches. The other switches in the set would have complimentary transmit / receive frequencies. Such a scheme guarantees that no collisions would take place, and with sufficient link margins, would assure perfect transmission/reception at all nodes.

Other uses are also apparent. For instance, full color digital SSTV with 256 x 256 resolution and 6 bits each of red, green, blue takes less than 30 seconds to transmit. Error free reception is possible, given that the picture is digital, transmitted with error detection and retries. In TCP/IP, the File Transfer Protocol could be used for pictures that must arrive, error-free, or UDP, with less-overhead, could be used if some loss or errors could be tolerated.

Digital voice is yet another interesting possibility. With the Delanco-Spry DSP board or the new 320C25-based DSP board that TAPR/AMSAT is working on plugged into a PC with the I/O processor card, audio from a microphone and preamp could be digitized by the DSP board, compressed, shipped via UDP on the RF link, then uncompressed and converted back to audio. "Voice mail" would be a real possibility with such systems!

---

4 Indeed, if there is one distinguishing feature common with all TCP/IP experimenters, it is the perceived duty to provide complete source code for *all* of our programs, free, as we believe that others can learn from our code and can further enhance it and re-release it back to the Amateur Community, in the best spirit of Amateur Radio.

5 Karn, P., "A High Performance, Collision-Free Packet Radio Network," *ARRL Amateur Radio Sixth Computer Networking Conference*, pp. M-89, Redondo Beach, 29 August 1987.

The DSP board could also act as modem for the I/O card. It would be easy to experiment with different modems, all digitally realized within the DSP card, choosing the best one for the transmission medium.

And we've only begun to explore the many applications for this high-speed interface.. . What we hope what we have communicated is that *real* 56k bits/sec allows much, much, more than simply faster bulletin board access or quicker "hunt-and-peck" ham-to-ham packet communication. A brave new world of distributed networks and file systems is open to the amateur with this interface and high-speed RF data links!

Status Updates

Those interested in the an up-to-date status report on this project are welcome send electronic mail via usenet or the Internet to mac@leg.ai.mit.edu or !pitt!aa4cg!bernie. You may also dial-up host AA4CG directly at 904/795-3211 at 2400, 1200, or 300 baud, eight-bits no parity. Login as user "packet" and password "radio".

Acknowledgements

We would like to thank Phil Karn, KA9Q, for suggesting the major features of this I/O processor, especially his comment "Design it like an Ethernet card[6]'' We also thank Bob Hoffman, N3CVL, for his expert typesetting, once again.

6 Indeed, we used the Western Digital WD8003E as our model.