A NEW PACKET-RADIO CONTROLLER BOARD

Terry Fox, WB4JFI
Vice-President, AMRAD
1819 Anderson Road
Falls Church, VA 22043

Abstract

This paper describes the AMRAD packet assembler/disassembler (PAD) to be released soon. It is Zilog Z80A based, uses a Zilog 8530 serial communications controller and is packaged on an S-100 pc board.

Introduction

One of the main problems with packet radio is that until recently there hasn't been a lot of hardware to support the various protocols being used. Except for a few pockets of activity, most of North America has adopted the idea of using a separate board (usually a single-board computer) to handle the actual generation and reception of frames. The first production board available to the amateur to do this was the Vancouver Amateur Digital Communication Group's Terminal Node Controller (TNC). This board was (and still is) available primarily as a bare board which had to be built up by the packet enthusiast. The board uses an Intel 8085 processor, 4k each of EPROM and static RAM, a serial or parallel device to communicate to your terminal/computer, and an Intel 8273 HDLC controller. The VADCG TNC moved the packet-radio software from the host computer to a separate board, and at the same time allowed many people to use a simple terminal with packet radio.

The next board that came out was designed by and is available from the Tuscon Amateur Packet Radio Corporation (TAPR). Its basic design philosophy is the same as the VADCG TNC in that it also handles all of the frame-level generation and reception of packets, requiring only a terminal or serial/parallel interface to a computer. Its actual hardware design is quite a bit different from the VADCG TNC, however. In addition to a different CPU (a Motorola 6809), it boasts quite a bit more memory (six byte-wide RAM/EPROM sockets normally fitted with 24k of EPROM and 6k static RAM), a different HDLC controller chip (Western Digital 1933), timed interrupts, a non-volatile memory, and a complete Bell 262 compatible modem (using the Exar modem chips). The TAPH group had time to study the VADCG TNC and made a lot of improvements when they designed their board. Another advantage to the TAPR TNC is that it is purchased as an assembled board, reducing greatly the chances of failure for the user. One of the many problems with the TAPR TNC actually has nothing to do with the board itself. TAPR has had a lot of problems getting the boards designed and into production. As of when this paper is being written, TAPR is getting the boards out to the last of their customers. This is one of the disadvantages of being out at the infamous leading edge. This also shows that the TAPR group doesn't want to send out TNCs that aren't as good as they can be, and the boards are definitly worth the wait.

The Amateur Radio Research and Development (AMRAD) group has been watching the progress of these boards with interest for quite a while now, and we figured it was time we got into the act. Most of us in AMRAD that are into the development stages of packet radio use S-100 based systems, usually with Z80 processors. So the TAPR TNC is rather difficult for us to write software for. Also, while having the modem on the TNC is nice for two-meter operation. However, when testing new ideas out for hf operation or btherwise when a different modem is needed, all the on-board modem does is take up board space.

We had some different problems with the VADCG TNC. The primary one is that the memory (both RAM and EPROM) are too small. I have modified the EPROM space to allow the use of either 2716's (up to 8k EPROM) or 2732's (up to 16k EPROM), but not many people want to take an X-ACTO (tm) knife to their board. Bill Danielson, N6FQR has modified the VADCG TNC to run 2k static RAM chips and also be able to download software from another computer to the TNC during software debugging, but here again, the board must be butchered.

We have come up with an S-100 board that contains an Intel 8273 protocol chip and some support logic so that we can write and debug software for the VADCG TNC in our S-100 systems before blowing any EPROMs for the VADCG TNC. This system works very nicely when coming up with changes in existing software, or working on some radical new idea without having to reburn EPROMs for every failure (no, I haven't written error-free code in a long time).

An additional problem with the VADCG TNC is that the baud rate on the packet channel is hardware selected and it goes down only to 600 bauds. Another modification was made to allow the baud rate to be software selectable and to allow the slower speeds needed for hf operation.

The AMRAD PAD

The further along we went with packet radio, the more kludges we needed to make on the VADCG TNC to allow us to do the experimentation we wanted. Since the TAPK TNC was in the initial design stages and using a processor we weren't accostomed to, it became apparent to us that it would be easier to design and build a whole new TNC board.

After making the decision to design our own board, we next had to decide what to put on the board, and what physical size it should be. I'm sure that it will come as a surprise to almost no one that the board will fit into an S-100 frame, and steal its power off the S-100 bus. This does not preclude the possibility of using the board stand alone with a single S-100 edge connector to supply power if the user isn't using an s-100 system.

Basic PAD Layout

Fig. 1 shows the basic layout of the AMRAD PAD (PAD stands for Packet Assembler/Disassembler). The PAD was designed to be very flexible. In addition to allowing the user to connect to it in either serial or parallel mode (the other boards do this also) it allows for fully adjustable baud rate on both serial ports, and if necessary, both serial ports can be programmed to be HDLC channels. It also has room for controlling the speed of a multi-speed programmable modem. The large amount of memory allows for downloading and debugging of programs in the PAD rather than needing another simulator board in a larger microcomputer. The large RAM space also allows a lot more space for buffers and other storage, meaning that the PAD should be able to run more than one connection per HDLC channel (something that could come in handy in the near future). The EPROM area can programmed to accept 2716, 2732, or 2764 devices, allowing plenty of room for expansion. A detailed description of the PAD board follows.

PAD Power Supply Circuitry

The power necessary to operate the PAD board is supplied thru the S-100 bus connector on the bottom of the board. The PAD uses three voltages, +8V at about an amp, +18V at about 50 milliamps, and -18V at approximately 100 milliamps. These

voltages are regulated on the board to supply the +5 volts and + 12 volts that the rest of the board requires. The five-volt bus has two 7805, TO-220 type voltage regulators, one on each side of the board. The + twelve-volt regulators use the smaller, TO-92 -style packages. These voltages are used primarily for the RS-232 driver chips and the real-time clock.

In addition to the power supply mentioned above, there is also a battery supply on board to run the real-time clock, the standby RAM chip, and some of these devices support logic. The PAD also has a sense circuit on board to tell the clock chip when-to go into the standby mode because the power is being shut down. This is accomplished by using a micro-power op amp that monitors both the battery voltage and the +5-volt bus, and sends an active low signal out when the main power is below the battery.

Z80 CPU and Support Circuitry

The PAD board is designed around the Ziloa Z80 processor. The master oscillator is an Intel 8080 support IC, the 8224, running at 18.432 MHz. This frequency was chosen because it is a multiple of the common baud-rate frequency of 1.8432 MHz, and also because it is high enough to be used in the refresh logic for the dynamic memory. The master oscillator is divided by five to produce a 3.6864-MHz clock for the Z80 CPU, along with the rest of the board. 'This frequency is a multiple of the 1.8432-MHz clock desired, so it can be used as the clock input to the serial-interface chip.

The Z80 reset logic consists of one half of a dual D flip-flop to make sure that all devices using reset get a properly timed signal. The Z80's NMI pin is optionally connected to a pushbutton to allow a way of interrupting CPU operations for debugging.

The Z80's RD and WR signals are buffered, since they are fed to many of the other devices on board. IORQ and Ml* are ORed together to produce INTA* that is used by the 8530 interrupt support logic.

EPROM Memory and Support Logic

There are four 28-pin sockets provided for EPROMs on the PAD board. They have been placed so that Textool Zero-Insertion-Force(ZIF) sockets can be used if the EPROMs are going to be changed frequently (they are recommended). A jumper area is provided to allow the EPROM spacee to be programmed for 2716, 2732, or 2764 type EPROMs. If 2716's are used, 6116 type RAM chips also can be plugged into the EPROM sockets, allowing up to 62k of RAM and one loader EPROM to be used. This mode of operation is advantageous when testing software written on a larger system and downloaded to the PAD. It also comes in handy when debugging the dynamic RAM, since the static RAM in the EPROM socket will allow monitors or memory testers to run properly.

A 74LS138 decoder is used to provide the chip enable to the EPROMs. There is also a signal generated called PROM*, which is used to automatically deselect the EPROM memory space from the dynamic memory.

Dynamic Memory and Support

The dynamic memory consists of eight 4164 type chips for a potential of 64k of memory. When this board was in its initial design stage, 4116 type devices were used since they provided a lot of memory for a very little amount of money and board space. As the PAD design progressed; the prices of the 64k chips dropped down to where it became cost effective to put them on instead. The refresh support logic is the same for both types of devices, and the power supply considerations are simpler for the 64k chips. The 64k devices also generate less noise on the power bus.

The sixteen address lines from the CPU are multiplexed onto the eight address lines of the 4164's with 74LS157 multiplexers driven by a select timing signal. This select signal, RAS*, and CAS* signals are generated by three D-type flip flops that are driven by the two clocks mentioned earlier, Ml*, MREQ*, and RFSH* signals from the Z80 CPU, and some additional logic. The PROM* signal mentioned earlier is added into the refresh logic to deselect the dynamic memory when an EPROM might be addressed instead of RAM.

The refresh logic is of standard design taken right out of the Zilog dynamic memory interface application note. The only differences are the addition of the two added address lines into the address multiplexers, and the use cf a times five clock instead of a times two clock. This higher clock rate is actually better because it causes the refresh signals to be within tolerences where a times two signal would cause some timing signals to be slightly out of specification.

Input/Output Decode Logic

Most of the port decode logic consists of one IC, a 74LS138 decoder. It sends chip-select signals to the various I/O devices, except for the real time clock and standby RAM chips. Because the real-time clock and stand-by RAM chips use many more ports than the other devices, they have separate chip-enable logic. The following is a brief chart of the port assignments on the PAD:

HEX ADDRESS	DEVICE
cm-03	8530 SCC
04-07	Terminal PI0
08-0B	Auxilliary PIO
0C-0F	Clock Interrupt Latch
10-13	Standby RAM Latch
40-5F	Real Time Clock Chip
80-FF	Standby RAM Chip

8530 and Support Logic

The two serial channels are generated by a relatively new device to the amateur packet enthusiast, the Zilog 8530 Serial Communications Controller, or SCC. When the PAD was in its initial design, it used the Intel 8273 protocol chip. This was the chip that most of us were familiar with, and software already existed to drive the 8273 properly. When we discovered the 8530, we realized that it could greatly simplify the PAD. Not only does it have two completely programmable channels, it also has two separatly programmable baud-rate generators, and two DPLLs. This one chip can support both the HDLC packet channel and the serial channel to the terminal or computer. In addition, if the terminal or computer is interfaced to the PAD board thru the parallel port, both serial channels can be used as HDLC packet channels. This might be useful when the PAD board is used with a larger computer as a gateway between two net-works. or if one channel is assigned permanently to one type of operation (say hf), and the other is used for a radically operation (say vhf or phone line interface to Telenet or ARPA).

Even though the SCC is designed by the same company as the Z80 and it is supposed to be compatible, there are some timing problems between the SCC and the CPU. The main problem has to do with interrupt-acknowldegement timing. The SCC is designed to work on the Zilog Z-BUS, and its daisy-chain interrupt structure is slightly different than the Z80 peripherals used on the rest of the PAD board. This timing problem is cured by adding a couple chips that are used to extend the time of an interrupt-acknowldegenent cycle, and also allow the SCC to be properly reset" by -hardware. The timing of this delay is accomplished by waiting the Z80 CPU with a delayed signal from a 74LS164 shift register. As with the dynamic refresh logic mentioned above, this modification is right out of an application note from Zilog.

Since both of the serial channels can be used as HDLC channels, it was decided to put timers on the RTS signal of both channels. These are 555 type timer designed to time out in about 30 seconds and stay off until reset by RTS changing back to an inactive state. The timer output of each of these time-out circuits are fed back into the CPU thru two bits on the auxilliary PIO, allowing the PIO to be programmed to generate an interrupt anytime the timer changes state. Not only does this give a fairly positive indication that the transmit command was successfully accomplished, but it also gives the CPU an indtcation if a time out has occured, and allows for a fairly graceful recovery (as opposed to hitting the reset button) after a time-out situation. These timers can be jumpered around in both channels if they aren't necessary.

The auxilliary PIO output side generates two

signals that can be jumpered into the transmit data signal from the SCC. These signals are useful for adding a cw i-d by changing the connected modem between mark and space independantly of the SCC. This means a cw i-d can be generated without having to change the SCC's operating mode and kludging the software as is done on the VADCG TNC. Hopefully this will simplify both the SCC support software and the cw i-d routine.

The rest of the SCC support logic consists of TTL/RS-232-C interface chips to allow the SCC to communicate with modems or other RS-232-C type devices.

The connectors used for the serial channels are the standard 26-pin insulation displacement connector (IDC) wired so that when crimped to a cable with a DB-25 on the other end, the RS-232-C signals will come out on the correct pins.

PIO Circuitry

The PAD has two 280 PIO chips on it. One is used for an optional terminal or computer interface if the user wishes to communicate with the PAD in parallel rather than serial mode. It can be left off if parallel interface is not required. A jumper is provided to continue the interrupt daisy-chain if this chip is not installed. If it is being used, its output signals are fed thru a 74LS244 octal buffer to provide enough drive to run the signals through a relatively long cable. The connector used for the parallel interfaces are the standard 26-pin IDC connectors and are wired the same as most parallel ports using this connector.

The auxilliary PIO is used to generate and receive several signals used by the PAD board. The parts of the aux PIO not needed for on-board functions are fed to a standard 26-pin connector, so they are available to the user for whatever he wants. In addition to the lines mentioned above that are used by the SCC interface, three lines from the out ut are sent to the channel A HDLC interface so they optionally can be used to control the speed of a multi-speed modem, such as might be used on hf.

Real-Time Clock and Support

The real-time-clock chip used is the National Semiconductor MM58167. In addition to keeping the time of day, it can generate timed interrupts to the CPU, and it also has an alarm function, allowing it to wake up the rest of a system at a preset time. Power for the real-time-clock chip is obtained from the plus twelve-volt bus thru a zener diode regulated power supply, some switching diodes and a battery supply. This allows for more time to cleanly shut down the system and prevent the clock chip from losing its memory.

The clock chip is not part of the Zilog Z80 family, and it does not generate any interrupt vector when it interrupts the CPU, so an octal tri-state latch was added to the board to provide an interrupt vector when the clock generates an interrupt. This vector is loaded by the CPU, so it can be programmed to be just about anything that the CPU will recognize.

The clock chip's power-down interrupt* is an open-drain output that is fed to the aux parallel 26-pin connector, allowing this signal to be used by external logic to power up a system.

Standby RAM Logic

The standby RAM is a 6514 low-power CMOS static RAM (organized as a 1024-by-4-bit device) that is addressed as a series of input/output ports. The upper three address bits to the RAM chip come from a four-bit latch. This allows the use of all 1024 nibbles in the RAM chip. The RAM is automatically deselected on power down, preventing it from being glitched by power transitions. The RAM can be used to store the amateur call, speed on both serial channels, whether the terminal/computer is using a serial or parallel channel, and many other PAD attributes that should be saved during power down. Using a battery-backed-up RAM instead of a NOVRAM allows the RAM to be updated much easier and more often, in addition to simplifying the hardware design.

Software Considerations for the PAD

The first software running on the PAD is a Z80 monitor modified to run on the PAD. The software required to run the PAD on AX.25 level 2 is being developed right now, and should be available just about the same time as the hardware. This software is written to take advantage of the PAD's features including the potential for downloading the protocol program and also allowing multiple connections. This software is taking a little longer to develop primarily because it is being written for a TDL Z80 assembler rather than using a higher-level language.

The Zilog 8530 SCC has sixteen write and nine read registers internally for each of the two serial channels. They are addressed by first writing a pointer to the control register of the channel of interest, then the requested register can be accessed. These registers allow each serial channels attributes to be programmed, from whether the channel is to be synchronous or asynchronous to the type of flag character used. The registers have to be programmed in a certain order to obtain proper operation, especially when using the SCC in HDLC mode. For more information on the SCC and its internals, see the AMRAD Newsletter for January of 1983.

The National MM58167A clock chip has 24 internal registers available to the programmer. Eight of these registers are the counters that contain the time, another eight contain the "alarm" time, and the rest are used to control or read the various internal functions of the chip, including such things as the interrupt handling, individual counter resets, and the standby interrupt (power down alarm) control.

The parallel input/output chips are standard Z80A PIO's. They have four modes of operation for each of the two parallel channels per chip (except for port B not being able to run mode 2. The four modes are:

Mode 0. Output mode.
Mode 1. Input mode.
Mode 2. Bidirectional mode.
Mode 3. Control mode.

The terminal PIO is capable of operating in any of the four modes with no problems. The auxilliary PIO is used to control and monitor several of the internal PAD operations, along with having a few bits left over for such things as modem-speed control. The aux PIO has the outputs of the transmit timers going to it for example, so if port A of the aux. PIO was put in mode 3, it could monitor the timers and generate an interrupt whenever the timers output changes state. The PIOs do generate Z80 style interrupt vectors, so this could be accomplished very easily, allowing an elegant recovery from a time-out condition.

PAD Availability

The PAD will be available only as a bare board for a while, due to capital limitations. I will be able to offer some of the harder-to-find components on a sporadic basis for a while. Hopefully we will be able to support the board better in the near future. Anyone interested in this board should write to me at the address listed above.

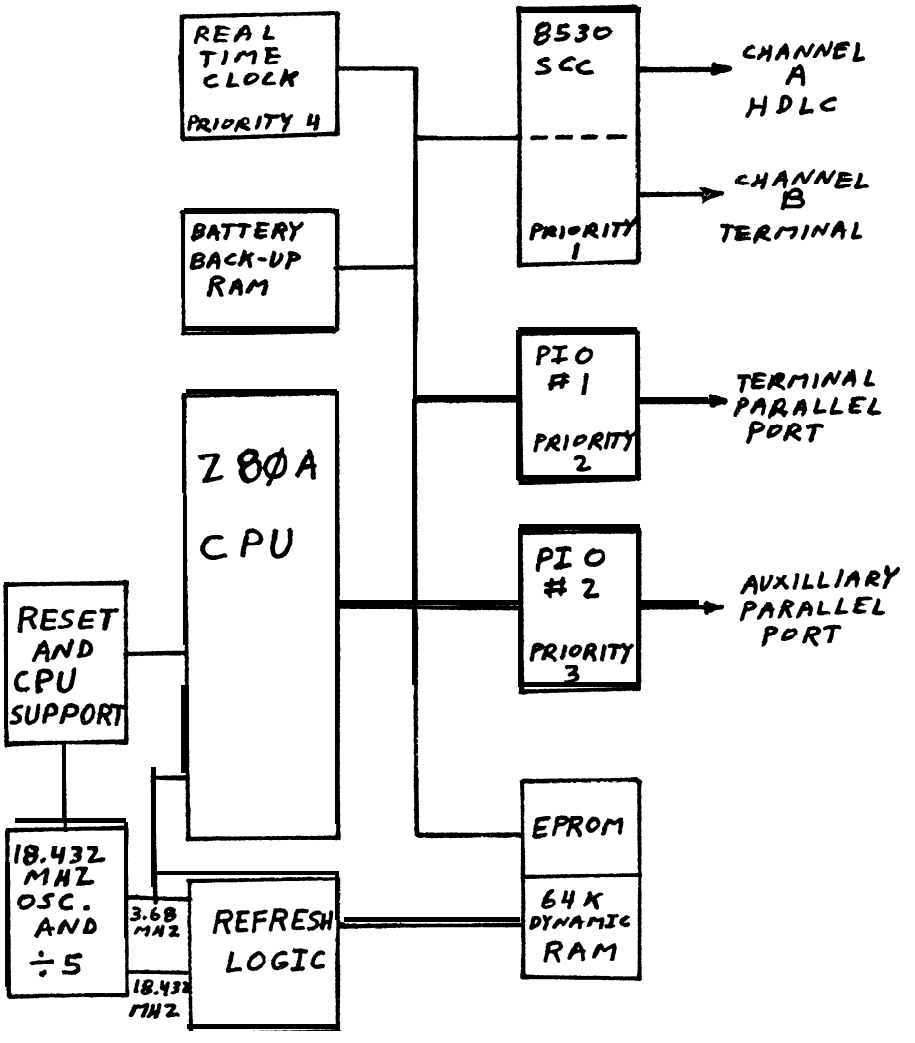

Fig. 1 - Block diagram of PAD.