#FXTN

This Tech Note is about transferring programs or data between a PC and an Atari Portfolio - DIP Pocket PC (PPC).

Parallel Port and File Transfer

1. Parallel Port

An IBM PC parallel port has three 8-bit registers; Data, Status & Control, in that order, typically based at IO address 378h. IBM used inverters on some of the control and status lines, as well as inverting some of the status bits in the BIOS. For example, when examining “IO Error” on a PC, you would read IO port 379h, and if the value of bit 3 is 0 that indicates that the printer has an error. A BIOS Int 17h Fn 2 status request inverted this bit, so that the status byte returned indicates an IO Error if bit 3 is 1.

The PPC Parallel Port peripheral was originally intended to provide a printer compatible interface. This was designed around an 8255 PIO device, rather than using 74-series logic. The 8255 has four 8-bit IO ports - A, B & C plus a config port D, at a base address of 8078h. The value written to port D on initialisation determines whether ports A, B & C are configured as inputs or outputs.

The PPC Int 17h BIOS configures ports A & B as outputs, and port C as an input. This maps to IBM’s Data, Control & Status lines, in that order. The reason why port C was used as an input for the 5 status lines is because it is the only port that supports interrupts. A future PPC peripheral with a parallel port might have wanted to use interrupts, although it would need to provide an interrupt vector register at 807Fh (just like SIVR for the serial port).

All of the lines from ports A, B & C go directly to the Parallel Port 25 pin D-connector. None are inverted. Plus some are not in the same bit positions as on a PC.

This is why if you patch a PC Parallel Port utility by replacing IO address 378h with 8078h and so on, it doesn’t work. As the program code might check the busy line, but would be reading the control port on a PPC as the control & status register are transposed. The different bit positions for some lines plus no inversions just make things worse.

The PPC was designed to be PC compatible from a BIOS & DOS standpoint - but not from an IO standpoint, as the effort to implement PC-compatible IO ports would have been too onerous.

2. Smart Cable File Transfer

The Parallel Port peripheral also supports File Transfer between a PC and the PPC. However, that was not the original intention.

There were plans for a Smart Cable peripheral that used only 4 signalling lines plus signal ground. One pair was to send and acknowledge data from the PPC to a PC, and the other pair was to do the same from a PC to the PPC. The File Transfer BIOS and PC application would deconstruct bytes into a series of single bits, and send them sequentially, and vice-versa. This very simple design meant that only a few 74-series devices would be needed, which along with a 5 line cable would have resulted in a simple low-cost solution to transfer files.

For various reasons, this Smart Cable peripheral wasn’t progressed. Instead, the Parallel Port peripheral was used to implement File Transfer.

3. Parallel Port File Transfer

In the late 1980s, most PCs had a “standard” parallel port with 8 output-only data lines, 4 output-only control lines, and 5 input-only status lines. This was before EPP & ECP. So a PC utility could have been written to send data via its data lines, and receive data via its status lines. Data would be sent as a byte, and received as 2 nibbles.

However, the PPC’s File Transfer BIOS had already been developed for the Smart Cable. A design decision was made to re-purpose it for the parallel port with minimal modification. This is the reason why it uses only 4 signalling lines plus signal ground, rather than using 8/4+4 bit transfers.

The File Transfer BIOS on the PPC configures the 8255’s port A, port B and one half of port C as inputs, and the other half of port C as an output, contrary to how the Int 17h BIOS configures these ports. This is so that data sent from a PC on data lines D0 and D1 is received by the PPC via a standard parallel cable - rather than having to crossover any lines. The PPC sends data to the PC using the “Paper error” and “Select” lines (which are configured as outputs), which the File Transfer application on a PC receives via Int 17h Fn 2.

This is why the PPC’s File Transfer only uses 4 signal lines (nominally D0, D1, Select & Paper) and signal ground. No other lines are needed. And why a standard parallel cable can be used, as the send & receive signal crossover is done at the port level rather than requiring a specially wired cable.

PC Card Drive

The PC Card Drive can be used to read and write PPC RAM cards on a PC. However, there are two constraints:

- There is a limited supply of PC Card Drive hardware surviving.

- The Card Drive device driver CD.SYSY makes direct I/O accesses which are not supported by WinXP onwards.

At first glance, the PC Card Drive ISA bus card looks quite complicated, with a large number of 74-series logic devices. Several 74LS244 octal buffers and 74LS245 transceivers, and four 74LS191 4-bit binary counters.

It is addressed and controlled via the PC’s ISA bus using just 4 IO registers. These register are at a base address which is user-selectable via the 4-gang DIL switch, with the default of “A” equating to IO base 350h. There is no memory addressing or an IRQ required from an ISA bus perspective.

The first IO register is a data register. The next two are address low and address high. The last register has a control function. This control register is organised as follows:

b7 Unused
b6 Count down (0= count up)
b5 Stop auto-counting
b4 Unused
b3 Unused
b2 Unused
b1 Unused
b0 A16 at the memory card connector

In normal operation, CD.SYS will use the absolute sector number specified (a 16-bit 0-based value) to determine the absolute memory card (CCM) address to be read or written. As the sector number is 16-bit, the Card Drive cannot manage a paged SRAM card in excess of 32Mb, assuming 200h bytes per sector.

A 32Mb paged SRAM CCM, with the PPC's software paging register at offset 0Ah within the first sector of every 128 Kb page, has a maximum sector number of FF00h. This is FFFFh minus FFh, the total number of sectors set aside at the start of each 128 Kb page for the software paging register.

CD.SYS's address algorithm will determine which 128 Kb page a specified sector is within, and set the software page register. It will check if the offset within that 128 Kb page is within the first or second 64 Kb, and set address line A16 in the control register accordingly. The sector offset within the first or second 64 Kb is written to the address high & low registers.

If data is then read from or written to the data register sequentially, the Card Drive converts this into a memory read or write to the CCM at the memory address selected.

The memory address is auto-incremented on every repeated data read or write. Once the address has been set, writing 200h bytes of data to IO port 350h will result in those values being written sequentially to the specified sector in the memory card.

Bit b5 in the control register disables the auto-counting provided by the 4x 74LS191s. Bit b6 selects auto-decrement rather than auto-increment; auto-decrement is not used by CD.SYS.

The Card Drive ISA slot PCB is only two layers, front and back, so the logic functionality could be reverse engineered visually and replaced by a more modern alternative, should someone be capable and motivated to do so.

A driver could be developed for WinXP and above, that allows a small number of IO addresses to be read and written, which the limited number of IO reads and writes in a patched version of CD.SYS could invoke.

Serial Port File Transfer

While files can be transferred via Zip drives, CF cards and other approaches, there is a simple alternative solution in addition to the Parallel Port File Transfer and the PC Card Drive.

Files can be transferred via the PPC's serial port peripheral, however this requires a serial File Transfer utility to be on the PPC which isn't there by default. There is an easy way to provide this.

A small PC utility can be used to send data via the PC's Serial Port, which is received by the PPC and saved to a file using "COPY AUX B.COM".

The baud rate has to be set accordingly at both ends for this initial transfer, the lower the better. There is a 128 byte size limit. Plus "AUX" is a character device, so some care is required to ensure that the binary data doesn't contain special control characters that alter the received data.

The received file b.com contains a small serial bootstrap utility, this can be invoked on the PPC to receive a file from the serial port, and a PC file transfer utility can then send a more capable serial transfer program from the PC to the PPC.