MULTIPROCESSING FOR THE IMPOVERISHED

Part 4: the IBM PC bus

by Brad Rodriguez

This article first appeared in The Computer Journal #68 (Jul/Aug 1994).

MEMORY

I'll start this installment off with something easy: the on-board memory, shown in Figure 1 [page **4** of the PDF file]. U20 is a 27256 (32K byte) EPROM. Recall from the previous installment that the EPROM is the only device which does *not* use the address generated by the memory mapping logic; instead, it uses the CPU address lines A0-A14 directly. OEPROM\ comes from the decoding logic, and will be asserted (low) only for processor Read cycles when A15 is high -- i.e., any read to CPU addresses 8000-FFFF hex.

You might wonder, then, why A15\ (the logical inverse of A15) is connected to the EPROM's chip enable input. It's true that this signal is logically redundant, since the EPROM's output enable can only be asserted when A15 is high (A15\ low). You could tie the EPROM's CE\ input low and the circuit would still work. But many EPROMs will consume less power when CE\ is high, so it's worthwhile disabling the EPROM whenever you can. Connecting A15\ to CE\ costs no extra logic -- I needed A15\ anyway -- and reduces the average power consumption of the board.

U21 is a byte-wide static RAM. Since the RAM appears in the 1 MB "mapped" address space, this chip uses address lines MA12-MA16. (A0-A11 are the same for all devices.) RAMRD\ and RAMWR\ are asserted (low) for read and write cycles, respectively, to addresses in the 127 KB "on-board RAM" range, E0000-FFBFF. Again, tying ONBOARD\ to the RAM chip enable input is logically redundant, but costs nothing and saves some power.

Thanks to the consistent "JEDEC" pinout of byte-wide memory devices, you can install a 128Kx8, 32Kx8, or 8Kx8 RAM chip in the U21 socket. (If you install a 28-pin device in this 32-pin socket, be sure to install it in the "bottom" of the socket, i.e., with pin 1 of the IC inserted into pin 3 of the socket.) The low address, data, and control lines are all the same. Only the high four address lines (MA13-MA16) are different, with the following results:

> 128Kx8 RAM (e.g. 628128): all four address lines are significant, so the chip occupies the full 127 Kbyte space. The last 1K of the RAM is inaccessible because this space is reserved for I/O.
>
> 32Kx8 RAM (e.g. 62256): the high two address lines are don't-cares (not even connected). This means that the mapped addresses E0000, E8000, F0000, and F8000 will all access the same location in the RAM chip. The net effect is that the 32K RAM appears *four times* in the 127K address space. The first three "appearances" allow access to the full 32K of RAM, but the fourth "appearance" (F8000-FFBFF) is shortened by 1K for I/O.
>
> 8Kx8 RAM (e.g. 6264): the high three address lines are don't-cares; but MA13 is tied to the *active-high* chip enable input CE2 of the 8K RAM. This means that MA13 must be high to access this chip, and so the 8K RAM will appear *eight times* in the on-board RAM space, starting at addresses E2000, E6000, EA000, EE000, F2000, F6000, FA000, and FE000. Again, the last "appearance" of the RAM is shortened by 1K for I/O.

You can do the same trick with U20, and install a 16Kx8 or 8Kx8 EPROM (making sure that the unused address lines are *high*). But there's little advantage to this: 27256s are as cheap as 2764s these days, and -- unlike the RAM -- you can't re-map the unused EPROM address space to other devices.

With a 2 MHz 68B09 (8 MHz oscillator), you should use 200 nsec or faster memory chips. The 1 MHz 6809 can use slower parts...but it's getting hard to find parts slower than 200 nsec!

PARALLEL I/O AND TIMERS

Figure 1 also shows U22, a Zilog Z8536 Counter/Timer/Parallel I/O chip. This chip provides two 8-bit parallel ports, a 4-bit control port, plus three 16-bit counter/timers. Several similar "multifunction" devices exist, including the Rockwell 6522 and Intel 8256. After much deliberation, I selected the Zilog part based on ease of interfacing, low cost, availability, and flexibility (in roughly that order).

Recall that eight I/O selects, IO0\ to IO7\, are generated for the eight 128-byte regions in the 1K "on-board I/O" address space FFC00-FFFFF. Select line IO2\ is used here. The Z8536 uses only two address lines, A0 and A1, and thus needs only four bytes of the 128-byte region FFD00-FFD7F. (Since address lines A2-A6 are "don't-cares", the chip appears 32 times in the 128-byte region.) LCLIORD\ and LCLIOWR\ are the read and write signals for devices in the "on-board I/O" space.

The interrupt output from this chip, INT\, can be jumpered to either the NMI\ or the IRQ\ input of the 6809. The remaining interrupt signals on the Z8536 pertain to Zilog's interrupt daisy-chaining scheme, and can't be used with the 6809 (at least, not without a *lot* of extra work). IEI is Interrupt Enable In, and must be high for this chip to generate an interrupt. IEO is Interrupt Enable Out and can be ignored. INTACK\, when held low, causes the Z8536 to place an "interrupt vector" on the data bus. Since the 6809 can't use this interrupt vector, INTACK\ is simply tied high.

The Z8536 requires a clock signal for its operation. This clock need not be synchronized with the CPU, but the BUFOSC signal is convenient and just the right frequency. If you are using faster 6809s, with a 6 MHz or 8 MHz oscillator, make sure you buy a Z8536 of the right speed (6 or 8 MHz).

The parallel I/O pins are brought out to J7. The strange wiring of this connector is for ease of PCB layout.

EXPANSION BUS

It might be desirable to expand the I/O of a single processor, without going through the rigamarole of the multiprocessor bus. Perhaps you want one of the 6809s on the bus to be a SCSI server, or perhaps you're using this as a single-board computer (SBC). The expansion bus allows you to connect additional I/O devices to the "private bus" of the CPU.

Since this is intended exclusively for I/O chips, only the low 7 address lines are brought to the connector. Five of the I/O selects are brought out, so you can add five I/O chips with no extra decoding logic. You can use either the "Intel-style" RD\ and WR\, or the "Motorola-style" R/W\ and E, to control the I/O read and write operation. (Warning: many 65xx family peripheral chips will *not* work with this board, due to address timing problems.) The "add-on" I/O also has access to the CPU interrupt lines NMI\ IRQ\ and FIRQ\, RESET\, and the oscillator signal BUFOSC.

The best way to use this bus is to build a "daughterboard" that plugs directly onto the CPU card. Don't use a long piece of ribbon cable here; the added capacitance on the bus lines will foul everything up, even the memory and I/O on the CPU board. Also, you are limited to *one* LSTTL load on the E signal. In general, if the daughterboard is going to be at all complex, you should also buffer the address bus, data bus, and R/W\.

THE IBM PC BUS INTERFACE

Most multiprocessor busses -- such as VME bus, Multibus, or Q-bus -- are much too complicated for a simple educational project like this one. Originally I had intended to invent my own multiprocessor bus. However, after reading about a 6809 board using the IBM PC bus (TCJ #64 p.47), and discussions with various Interested Parties, I realized the advantages of being able to use IBM PC peripheral cards.

A bit of history: the original PCs all had 8-bit CPUs, and so allowed 8-bit peripherals to plug into 8-bit expansion slots through a 62-pin edge connector. The PC/AT, with its 16-bit CPU, needed a 16-bit bus; but IBM wanted to keep using the cornucopia of 8-bit plug-in cards. So they kept the 8-bit bus, and put the additional 8 data bits (plus some more address bits and extra control lines) on a second, 36-pin edge connector. This is why you see AT (286), 386, and 486 motherboards having a certain number of "8-bit" and "16-bit" expansion slots. The "8-bit" slots have one edge connector, and the "16-bit" slots have two. All of the old 8-bit peripherals work fine in a 16-bit system. (To confuse the issue, a *third* edge connector has now been invented for 32-bit peripherals.)

Since the 6809 is an 8-bit processor, I only desire to support the basic, 8-bit, PC bus. (There's no shortage of 8-bit perhipheral cards!) But this bus does not include control signals for multiprocessing, and leaves no pins unused. A separate processor arbitration bus is required. Rather than mess around with ribbon cables or custom edge connectors, I decided to use the second edge connector of the PC/AT-style cards for the arbitration bus. This allows me to use commonly available 16-bit "passive backplanes" for the multiprocessor system. (A "passive backplane" is a board which has only the edge connectors for the PC bus. The CPU and all of the rest of the motherboard electronics sit on a plug-in card. In theory, this lets you upgrade CPUs more easily.)

Important: the second edge connector is NOT used in accordance with the PC specification. You can NOT use 16-bit IBM PC cards with the ScroungeMaster II, and you must NOT plug the SM II into a 16-bit slot of a PC motherboard. Use ONLY 8-bit peripheral cards, and ONLY a passive backplane! Passive backplanes are available at swapmeets -- I found several at the

Trenton Computer Festival -- and Alltech Electronics Co. (602 Garrison Street, Oceanside, CA 92054, phone 619-721-7733, fax 619-721-2823) has advertised 16-bit backplane boards for $10 in Nuts & Volts.

Figure 2 [page **3** of the PDF file] shows the bus logic. (For a complete explanation of the IBM PC bus signals, see the sidebar.) U16 through U19 are the tri-state bus drivers. These drivers can only be enabled when the signal DRIVENBL\ is low, which you will recall occurs only when a) this CPU is attempting to access an "external" address and b) the bus has been granted to this CPU.

U19 is a bidirectional driver for the data bus. During CPU Write cycles this outputs data from D0-D7 to the PC bus lines XD0-XD7. During CPU Read cycles, data from the peripheral card on XD0-XD7 is output to the 6809 bus D0-D7. The direction of transfer is controlled by the 6809's R/W\ signal, via the DIR input of the 74LS245.

U16, U17, and U18 buffer the address bus from the 6809. This is always output by the CPU, so these are unidirectional buffers. U18 also buffers the control lines IORD\, IOWR\, MEMRD\, and MEMWR\. Like the address lines, these are always output by the CPU and input by the peripheral cards. They must pass through a 74LS244 buffer so that they can be disabled when another CPU "owns" the bus.

The *active-high* RESET line is input by all CPUs and peripherals. It is normally generated by the motherboard. Since we are using a passive backplane, it must be generated by a plug-in card. To simplify matters, any 6809 board can output this signal by installing JP1. Only *one* 6809 board (the "bus master") should have this jumper installed. U9A is a simple RC reset circuit; U9B inverts the active-high RESET to the active-low signal used on the 6809 board.

The CLK and OSC lines are also normally generated by the motherboard. CLK must be synchronous with the processor, but is otherwise rather vaguely defined (see sidebar); to meet this requirement, all of the 6809s use CLK as their (4, 6, or 8 MHz) oscillator signal. OSC is a 14.31818 MHz signal -- a hangover from the days of 4.77 MHz XTs, useful for video display circuits but not much else. CLK and OSC are produced by one designated "master" 6809 board, by the simple expedient of installing Y1 and Y2 *only* on that board. U7F buffers CLK for each 6809 board, since there are many loads placed on this line.

The PC bus has six *active-high* interrupt lines, IRQ2-IRQ7. For simplicity, I have decreed that each 6809 board can handle *at most one PC bus interrupt line.* Thus, to handle all PC interrupts, you would need six 6809 CPUs. Jumper JP16 selects which of the interrupt lines this CPU will respond to as the "external interrupt" XIRQ. This is then inverted (see the CPU schematic) and may be routed to the CPU's IRQ input, or a "pseudo interrupt" on an I/O chip.

DMA is not supported by the ScroungeMaster II. The complications of DMA and multiprocessor arbitration are more than I wanted to attempt for this project, and the CPU board was big enough already! So, the DMA control lines DRQ2, DACK0\, DACK1\, DACK2\, and DACK3\ are all ignored. For experimentation I allow the DMA request lines DRQ1 and DRQ3 to be jumpered as interrupt inputs to the 6809. This will not cause an electrical conflict with PC peripheral cards, but you *cannot* use the DMA function of those cards, since the required DACKn\ acknowledge signal is not generated. (Maybe some TCJ reader will design a plug-in DMA controller board.)

You also cannot use plug-in dynamic RAM boards. Most of these depend upon the DACK0\ signal for refresh timing.

The IORDY (a.k.a. WAIT\) signal is routed directly to the wait-control logic, as described in the previous installment.

Strictly speaking, pullup resistors should be installed on the motherboard for the signals IORD\, IOWR\, MEMRD\, and MEMWR\. This is so they won't be inadvertently asserted (pulled low) when all of the CPUs are "off the bus." (I should have put an optional pullup on the 6809 board; sorry, I forgot.) Likewise, the interrupt lines should have weak pulldowns, so that "missing" peripheral cards don't produce an interrupt. This can be handled by removing JP4 when the selected external interrupt line is not in use.

BUS ARBITER

U31 (on Figure 1), and U28 (on Figure 2), constitute a "round-robin" bus arbiter. U31 is a free-running three-bit counter which is present *on the "bus master" CPU board only.* It runs at the BUFOSC rate (synchronous with the 6809s) and counts 0,1,2,3,4,5,6,7,0,1,2,3... in binary.

Each of the eight CPUs which can be installed in a system is assigned a unique number from 0 to 7, by installing the corresponding jumper in JP17. When the counter reaches that number, *and* that CPU is requesting the bus (REQ is high), the corresponding output of U28 will go low, and thus GRANT\ will go low.

This low signal is also "diode-OR'ed" onto the GRAB\ line on the arbitration bus. (The diode, and pullup resistor R10, are necessary because all the other CPUs are trying to pull this line high.) Any CPU can pull GRAB\ low, to indicate that it has "grabbed" the bus. This halts the counter U31 (via inverter U27D), which "freezes" the count at the CPU number which now

"owns" the bus. The counter will remain frozen until that CPU de-asserts (pulls low) its REQ line, signalling that it is done with the bus. Then all the outputs of U28 go high, and the count continues.

No matter what order the CPUs make their requests, they will always be granted the bus in strict rotation. This satisfies the requirement (mentioned in the previous installment) that no one CPU can wait more than 15 usec for the bus. *Remember that U31 is installed in only one CPU board; U28 is installed in every CPU board.*

SIDEBAR: THE IBM PC BUS

The IBM PC bus is undoubtedly the most widely-used and poorly-documented computer bus in history. Much of my information comes from reading PC peripheral card schematics, the technical reference manual for my old XT clone, and the excellent book by Lewis Eggebrecht [EGG90] which I only discovered this year. This is a brief summary, starting on the component side of the connector:

IOCHK\ (I/O Check) can be pulled low by a peripheral card to signal an error condition. The ScroungeMaster II ignores it.

D0 through D7 are the data bus.

IORDY, when pulled low, stretches the memory or I/O access cycle. This is called WAIT\ on Z80s and MRDY on the 6809.

AEN (Address Enable) is output high by the PC motherboard to indicate that a DMA cycle is in progress. Since the SM II has no DMA, it ties this line low.

A0 through A19 are the 20-bit address bus.

RESET is the active-high reset line.

GND, +5, -5, +12, and -12 are, I hope, obvious.

IRQ2,3,4,5,6, and 7 are six active-high interrupt lines. Each of these can be pulled high by a peripheral card to generate a CPU interrupt. Normally each line can be used by only one IBM PC peripheral card. The PC has an eight-input interrupt controller chip; the SM II limits each CPU to handling one bus interrupt.

DRQ1,2, and 3 are three active-high DMA Request lines. Each of these can be pulled high by a peripheral card to request a Direct-Memory-Access data transfer. Normally each line can be used by only one PC peripheral card. The PC has a four-input DMA controller chip; the SM II does not support DMA.

DACK1\, DACK2\, and DACK3\ are the active-low acknowledge lines for DRQ1, 2, and 3. These are not supported by the SM II.

DACK0\ on IBM PCs and PC/XTs indicates that a dynamic RAM refresh is taking place. (The 8-bit PCs used one DMA channel as a cheap & simple refresh controller.) This is not supported by the SM II.

SMEMR\ and SMEMW\ are the memory read and write strobes. When active (low), they indicate that a valid 20-bit address is on the bus, and a data transfer can take place.

IOR\ and IOW\ are the I/O read and write strobes. When active (low), they indicate that a valid *10-bit* address (A0-A9) is on the bus, and a data transfer can take place. Intel and Zilog CPUs distinguish between I/O and memory references; and in the IBM PC, only 10 address bits are used for I/O. For the 6809 a 1K address space is designated "I/O" and the IOR\ and IOW\ signals are generated accordingly. Note that a bus cycle can be a memory *or* an I/O cycle, but not both -- thus, if IOR\ is low (active), SMEMR\ had better be high.

CLK is a clock signal derived from the 8088 support chips, and really not defined any better than that. It is synchronous with the CPU; its exact timing specs are prohibitively complex to emulate with the 6809. Fortunately, it's rarely important.

T/C (Terminal Count) when high, indicates that a DMA operation is complete. This is not supported by the SM II.

ALE (Address Latch Enable) when high, indicates when the address is being output by the 8088 CPU. Like CLK, it is difficult to emulate on the 6809, and rarely used.

OSC is a 14.31818 MHz signal used for old (CGA) video cards and not much else. It is not synchronized to the CPU clock.

For more details, I highly recommend Eggebrecht's book.

REFERENCES

[EGG90] Eggebrecht, Lewis C., *Interfacing to the IBM Personal Computer*, Second Edition, SAMS, 17111 North College, Carmel, Indiana, 46032 (1990), ISBN 0-672-22722-3. The best book I've found for descriptions of the IBM PC and PC/AT busses. Available from JDR Microdevices, 2233 Samaritan Drive, San Jose, CA 95124, phone 408-559-1200, fax 408-559-0250, if you can't find it locally.