A look at Z80 peripheral chips

One of the important features of the Z80 was the family of peripheral chips that it was designed to work efficiently with. These chips permitted a computer to be built with a minimum of additional hardware.

Faggin and Shima discuss the importance of the peripheral chips to the success of the Z80. They also mention that the Z80's bus was designed to work well with them.

The configuration of these chips motivates much of the interrupt design of the Z80, so understanding these chips will help explain the interrupt system in the Z80.

Most of the information in this chapter comes from the Z80 “Family CPU Peripherals User Manual”.

The interrupt structure is discussed in detail in http://www.z80.info/zip/z80-interrupts_rewritten.pdf a Zilog Application note:“The Z80 Family Program Interrupt Structure”
Z80 peripherals are connected in a “daisy chain” configuration, where each peripheral is connected to the next peripheral in the chain in order of priority.

The idea of interrupt priority is if multiple peripherals can generate interrupts, the system needs to decide which peripheral gets handled if multiple interrupts happen at the same time. For instance, if a disk drive and a keyboard interrupt happen at the same time, it may be more important to handle the disk drive interrupt quickly before the next block of data is available. Since typing is much slower, it's probably safe to wait before handling a pressed key.

In addition, while handling a lower-priority interrupt, the CPU can be interrupted again to handle a higher-priority interrupt. (Although the CPU can block this by disabling interrupts.) Lower-priority interrupts are not permitted to interrupt the handling of higher-priority interrupts.

(An alternative, which the Z80 doesn't use is “round robin” priority, where the system goes through each peripheral in order, and no peripheral has a higher priority than any other.)

The Z80 peripherals implement priority by using the IEI input in a daisy chain. In turn, the peripheral feeds its IEO output into the IEI input of the next peripheral. Normally, a peripheral chip just passes the IEI input into the IEO output, so it passes down the chain. If a peripheral's IEI input is active, the peripheral has priority for the interrupt. In this case, it inactivates the IEO output, which blocks all the lower-priority peripherals down the chain. Thus, the IEI/IEO pins implement the priority system.

The peripheral chip that has priority for its interrupt puts an 8-bit interrupt vector on the bus. In interrupt mode 2, the CPU combines this vector address with the high-order address in the I register to determine the address of the interrupt vector routine.

The interrupt vector address in each peripheral chip is programmable. This allows a separate interrupt routine for each chip.

Not all bits of the interrupt vector are programmable. The low-order bit must be 0 since the destination address is 16-bytes (two bytes). The low-order byte of the address is in an even location in memory, and the high-order byte is in the next location.

For the counter chip, the next two bits indicate which of the four counters generated the interrupt. That leaves 5 bits programmable by the user.

One disadvantage of the daisy chain system is that the IEI/IEO signal must propagate through all the chips in sequence. If there are many chips, this signal will be delayed as it is processed by all the chips. To allow extra time for this processing, the CPU inserts two wait states (i.e. two extra clock cycles) when it handles an interrupt. A maximum of 4 peripheral chips are allowed without adding additional hardware to reduce the propagation delay.

One unusual thing about the interrupt system in the Z80 family is that

peripherals monitor the bus and watch for a RETI (return from interrupt) instruction. This lets the peripheral chip know when interrupt handling is completed. The M1 signal from the Z80 lets the chip know when an opcode is being fetched. This allows the peripheral to distinguish between a RETI instruction and data with the same values. (RETI is an ED prefix followed by 4D). If the chip's interrupt is currently being serviced, the chip uses RETI to determine that its interrupt routine is done. The chip then frees up IEO, so a lower-priority peripheral can resume interrupt processing.

There's a bit of complication with the RETI processing due to interrupt nesting. A higher-priority interrupt can take place during the processing of a lower-priority interrupt. When the higher-priority interrupt is handled, the lower-priority interrupt handler will continue. This ensures that a high-speed I/O device won't get blocked while a low-speed I/O device is getting handled. This nesting can happen multiple times, with several interrupt handlers in progress.

Note that while interrupts are disabled, the higher-priority interrupt will be ignored. This ensures even a low-priority interrupt handler won't get interrupted at a critical time. Of course, interrupts should be disabled for as short a time as possible, to avoid long delays in handling high-priority interrupts.

To distinguish between completion of the highest-level interrupt, and completion of a lower-priority interrupt (if a higher-priority interrupt didn't get acknowledged yet). The IEO line is pulsed during the prefix handling to distinguish these cases. (More explanation is needed.)

The first chip is the “Counter/Timer Channels” chip, called the “Z80 CTC”. This chip provides four programmable counter/timers.

Each counter/timer repeatedly counts down to zero

The Z80 CTC chip can be connected directly to the Z80 CPU without additional logic.

The interrupt handling of the CTC chip is important.
The counter timer chip either counts signals on input pins (in COUNTER mode), or counts clock signals (in TIMER mode). Each signal decrements the associated register. When the register reaches zero, the chip generates an interrupt and/or generates an external signal.

The counter timer chip can be used in many applications, from generating a baud rate signal to providing a timing signal, to counting external events.

DMA controller
A second important peripheral chip is the DMA controller, which provides Direct Memory Access. The purpose of DMA is to provide high-speed transfer of data between memory and an I/O device without going through the CPU.

The problem with moving data via the CPU is it is relatively slow. The CPU must read a byte from the I/O device, then write the byte to memory. This is repeated until the block is transferred. Even with block-transfer instructions, this is inefficient since each byte gets moved twice. In addition, during this time the CPU is occupied with memory transfer, and can't be used for other tasks. Finally, the latency to start copying a block is relatively high, since the CPU must receive an interrupt, enter the interrupt routine, and set up the appropriate registers before it can even start copying the data.

In a DMA, the data passes directly between the memory and the I/O device, bypassing the CPU. The full bandwidth of the memory is available for this transfer.

Using block transfer, a 4MHz Z80 can transfer about 200 Kbytes/sec. But the interrupt routine typically takes 5 to 10 μs.

By using DMA, memory-to-memory transfers or transfers between I/O and memory can be done much faster than if the CPU does the transfers.

The Z80 DMA chip not only provides DMA, but also allows the data to be

scanned for a particular byte or bit pattern while being accessed. This allows very fast memory search to be performed.

To use the DMA chip, the CPU writes to the chip how many bytes to copy, and the start address for the source and destination. The chip then performs the copy. (This is a conceptual description; the details are more complicated.)

The DMA chip has multiple modes for sharing the bus with the CPU: it can let the CPU use the bus after each byte transferred, it can grab the bus as long as there is data ready to transfer, or it can grab the bus until the whole data block is transferred.

The chip can be programmed to generate interrupts in various circumstances. It can interrupt before starting a DMA, when a DMA is done, or when a byte match is found in a search.

Parallel Input/Output

The Z80 PIO (Parallel Input/Output) chip provides two parallel I/O ports with all the logic to allow a peripheral device to be connected directly. Typical applications for this chip are keyboards or printers.

The chip has two ports (A and B), which can be configured for input or output. Input and output can be configured on a per-bit basis, so some bits are input and some are output. The A port can also be configured for bidirectional data transfer. The ports have “handshake” control lines, which allow the peripheral device and the chip to synchronize the data transfer.

One interesting interrupt feature is that a mask can be configured so when certain pins match a particular pattern, an interrupt will be triggered. This provides a lot of flexibility for when the CPU gets interrupted.

The interrupt handling uses the Z80 interrupt priority scheme, as with the other peripheral chips.

Serial Input/Output

The Z80 SIO (Serial Input/Output) chip provides two serial data ports for a

variety of applications. The chip handles asynchronous and synchronous byte-oriented protocols, as well as synchronous bit-oriented protocols. It also can be used for modems, floppy disk interfaces or cassette interfaces. (When the Z80 came out, cassette tapes were a popular storage medium for files. Data would be converted to an audio signal and recorded on tape using a standard cassette player. The audio signal could be played back to read in the data.)

The chip supported protocols such as HDLC (High-level data link control) and IBM's SDLC (Synchronous Data Link Control). These were popular communications protocols in the 1980s. To support SDLC, the chip performed special functions such as zero-insertion: to allow the sender and receiver to stay in sync, the data stream couldn't send more than five 1's in a row. If there were five contiguous 1's, the chip inserted a 0 after them.

The chip generates CRC codes and checks them. This provides error detection for various protocols.

One interesting thing about the chip is that 40-pin DIP packages were the largest common size at that time, which wasn't quite enough pins to provide all the desired signal pins. The chip was sold in various options with different “missing” pins. If you wanted all the signal pins, you needed to buy the 44-pin QFP (quad flat package) or PLCC (plastic leaded chip carrier) packaging.

The chip provided a complex set of interrupts for transmission, received character, or status changes, among other things. Over a dozen different conditions can generate interrupts.

Summary

To summarize, the peripheral chips are an important part of the Z80 family. Since they are designed to work with the Z80, they can be used without additional “glue” logic. They take advantage of the Z80's mode 2 interrupts to allow each peripheral to have a separate interrupt vector. This allows each device's interrupts to be handled quickly, without the CPU needed to poll each device to determine which device caused the interrupt.