The Use of the Z80 I/O Processor by the TRS-XENIX Operating System

Jerry Dunietz

Robert Powell

Microsoft Corporation

The Microsoft Building
10700 Northup Way
Bellevue, WA 98004
(206) 828-8080
decvax!microsoft!jerryd

The Radio Shack Model 16 computer system uses two processors — a Motorola Mc68000 16:32 bit processor and a Z80 8 bit processor. Only the 8 bit processor has access to I/O devices, such as the console/keyboard, built-in serial ports and parallel port, built-in floppy drive(s), and hard disk drive(s). The Z80 can access the Mc68000 memory and can generate three distinct interrupts to the Motorola processor. The TRS-XENIX system was designed to take advantage of this system architecture. The features dependent upon the dual processor design generally improve the system in one of two areas — I/O performance and configurability.

This talk describes the design of these features in some detail. Points of interest include:

- minimizing the Mc68000 interrupt burden in high-speed serial I/O without delaying XOFF processing;
- designing one interface between the Z80 and Mc68000 to handle all character-at-a-time devices, whether serial or parallel;
- designing one interface between the Z80 and Mc68000 to handle multiple sizes and formats of hard and floppy disks;
- making the TRS-XENIX system dynamically adjust to the format of an inserted floppy disk;
- making the TRS-XENIX system auto-configure for the root and swap devices and their sizes;
- detecting the removal of a currently open floppy disk;
- for Microsoft's internal development and debugging use, providing a mechanism by which the Z80 may detect the crash of the Mc68000.

The Use of the Z80® I/O Processor by the TRS-XENIX™ Operating System

Jerry J. Dunietz
and
Robert D. Powell

Microsoft Corporation
10700 Northup Way
Bellevue, WA 98004

decvax!microsoft!jerryd
decvax!microsoft!bobp

ABSTRACT

The Radio Shack Model 16 computer system uses two processors—a Motorola MC68000 16:32 bit processor and a Zilog(R) Z80 8 bit processor. Only the 8 bit processor has access to I/O devices, such as the console/keyboard, built-in serial ports and parallel port, built-in floppy drive(s), and hard disk drive(s). The Z80 processor's ability to access MC68000 memory, along with the existence of interrupt lines between processors, provides the only means for the MC68000 to communicate with the outside world. The TRS-XENIX I/O system takes advantage of this hardware architecture. Subject to constraints on implementation time, it was designed to minimize I/O overhead for the MC68000, while providing maximum I/O compatibility and configurability.

1. HARDWARE ARCHITECTURE.

The Radio Shack Model 16 computer includes a complete 64 Kbyte Z80-based computer system, with an additional MC68000 processor and associated RAM. The Z80 processor and its memory plug into the system bus, with access to all I/O devices and peripherals. The Z80 can map the screen controller's RAM into the high end of its address space, and can read the built-in console keyboard. It also deals with the 2 built-in serial lines and 1 parallel port, and the floppy and hard disk controllers.

The MC68000 processor uses its own private bus for its memory, and has no access to the system bus. The processor board includes its own memory management unit (MMU), providing relocation and protection for user mode accesses to MC68000 RAM. The MMU is capable of dealing with 1 Megabyte of RAM, although it may not be possible to physically install this much MC68000 RAM in a specific machine. The processor board also includes a programmable interrupt controller.

Although the MC68000 processor has absolutely no access to the Z80 RAM, the Z80 does have access to all of MC68000 RAM. It can map any 16 Kbyte bank of MC68000 memory into its address space. (However, the hardware does not support DMA to or from the MC68000

© Microsoft Corporation, 1983
Zilog and Z80 are registered trademarks of Zilog, Inc.
TRS is a trademark of Tandy Corporation.
XENIX is a trademark of Microsoft Corporation.

address space.) The Z80 controls the halt and reset lines of the MC68000. Additionally, the Z80 independently controls three different interrupt lines to the MC68000 board.

1.1. Multiple Hardware Configurations.

The Model 16 computer exists in a variety of forms. In particular, any Radio Shack Model II or Model 12 Z80-based computer can be upgraded to Model 16 capabilities. Although there have been various configurations of Z80 processor board, disk drives, and serial interfaces for the Model II, Model 12, and Model 16 over the years, one version of the TRS-XENIX system was required to run without modification on all of these systems.

1.1.1. Processor Board.

On Model 16 and upgraded Model 12 computers, the MC68000 is able to interrupt the Z80. However, upgrades of older Model II's have no such capability. Additionally, the Z80 procedures for setting the baud rates on the serial lines vary slightly on different Z80 processor boards.

1.1.2. Floppy Disk Drives.

Radio Shack has varied the type of 8-inch floppy drive used on Model II/12/16 systems during their production history. Some have different seek times than others. Some are capable of detecting that a disk drive door has been opened in the time since a previous access. Some are single-sided, others double-sided. Some drives stop the drive motor between accesses, others do not.

1.1.3. Hard Disk Drives.

Thus far, Radio Shack has sold two different hard disk drives for the Model 16 family. The currently sold hard disk has a 12 Megabyte capacity. Previously, Radio Shack sold an 8 Megabyte hard disk.

2. DESIGN GOALS.

Specific goals guided the design of the TRS-XENIX I/O system. Implementation time had to be minimal. The Z80 and MC68000 binaries had to be compatible across all machine configurations. We needed to maximize data interchange capabilities with other machines. In particular, we required the ability to read and write all existing TRSDOS$^{(R)}$ floppy disk formats, as well as standard IBM floppy disk formats. Interrupt burden on the main (MC68000) was to be minimized; on other systems with only one processor and a "dumb" UART, the interrupt burden of 9600 baud serial output can consume almost all of a CPU's time. Finally, the size of the TRS-XENIX kernel had to be minimized, requiring simple device drivers to control multiple devices.

2.1. OVERALL DESIGN

The TRS-XENIX I/O system uses the Z80 processor to simulate several intelligent device controllers. The parameter blocks and communications areas for these devices live in the low bank of MC68000 memory, although I/O transfers can take place to or from any arbitrary MC68000 address. When the MC68000 code is first started, it informs the Z80 processor of the location of the parameter blocks by putting a pointer to the parameter blocks into a specific vector table entry.

Each individual parameter block contains several byte-sized registers. One register in each parameter block is generally a control and status register, or CSR. The CSR contains, possibly among other bits, a GO bit, an INTERRUPT ENABLE bit, and an INTERRUPTED bit. The GO bit is used as a semaphore to control access to the CSR and associated registers.

TRSDOS is a registered trademark of Tandy Corporation.

When the MC68000 initiates a command, it sets up various command parameters, then modifies the CSR, setting the GO bit. Once the GO bit has been set, manipulation of the CSR and most of the device control registers for a given device are turned over to the Z80 processor; the MC68000 may not modify the CSR, transfer address, etc., until the I/O request is completed by the Z80. When the Z80 completes the I/O request, it modifies any registers appropriate to pass status information back to the MC68000, and then clears the GO bit of the CSR. Once the GO bit has been cleared, control of the CSR passes back to the MC68000.

The INTERRUPT ENABLE bit and INTERRUPTED bit are used to provide full support for both polled and interrupt-driven operation. If the MC68000 does not set the INTERRUPT ENABLE bit when initiating a command, the Z80 will not interrupt the MC68000 when the I/O operation completes. The INTERRUPTED bit is set by the Z80, and cleared by the MC68000. The Z80 sets the INTERRUPTED bit when it generates an interrupt from this "device" to the MC68000. The Z80 will not generate another interrupt for this "device," until this bit is cleared by the MC68000 device driver. Thus, if the MC68000 falls behind the Z80 in servicing interrupts, the Z80 will not attempt to generate superfluous interrupts.

Because the TRS-XENIX code had to run without modification on MC68000 processors which cannot interrupt the Z80, the MC68000 device driver code does not rely on this interrupting ability. The Z80 polls the device control registers in the MC68000 address space, waiting for the MC68000 to initiate a request. The Z80 polling loop is interruptible by the peripheral devices. At any given moment, the MC68000 may have multiple I/O requests in service—one transmit request per serial line plus console and parallel port, one hard disk request, one floppy disk request, and of course the implied request for incoming characters.

The three interrupt lines from the Z80 to the MC68000 are each used. One line provides a constant frequency clock interrupt (see below), one is used for all disk interrupts, and one for character device interrupts.

3. THE SERIAL INTERFACE.

For development speed and interrupt overhead reasons, the TRS-XENIX implementation bases its serial interface on a standard UNIX™ operating system device driver, that of the Digital Equipment Corporation DH11.

3.1. One Interface for All Character Devices.

Communication between the Z80 and MC68000 processors is identical for all character-oriented devices. Specifically, the console/keyboard, serial lines, and parallel port all look absolutely identical to the MC68000. This interface design minimizes the size of the MC68000 device drivers, reducing the amount of MC68000 RAM required for the TRS-XENIX kernel and maximizing the amount of RAM available for user programs. It also makes it easier for Radio Shack to add any additional character devices which they might choose to introduce and support.

Adding new character devices involves changing one constant in the MC68000 code; however, modification to the Z80 code is still required. On the other hand, because the TRS-XENIX system owner gets no tools to modify the Z80 code in any way, it becomes next to impossible for her to add her own device driver.

3.1.1. Input.

Character oriented input to the TRS-XENIX system is buffered through a receiver FIFO. The FIFO is implemented as an array of two-byte structures, with two byte-sized registers holding indexes into this array. (There is an additional control register.) One of these two index registers is modified only by the Z80, and keeps track of point at which the Z80 inserts characters into the FIFO. The other register is modified only by the the MC68000, and is used to indicate the location of the next entry which the MC68000 will remove from the FIFO. The buffer is

UNIX is a trademark of Bell Laboratories.

manipulated in a circular fashion, using the values of the two index registers to indicate FIFO full and FIFO empty conditions. Because each index register is under the control of a different processor, there is no need to use any additional special construct to provide for mutual exclusion between the processors in the manipulation of the FIFO data structures.

Each entry in the receiver FIFO contains three pieces of information: the received character (of course), the line number, and status information. Status information includes indications of receiver overrun, parity error, framing error, etc.

As with other devices, there are INTERRUPTED and INTERRUPT ENABLE bits associated with the receiver FIFO. To further reduce interrupt burden on the MC68000, the Z80 dynamically and transparently adjusts the FIFO alarm level, which is similar to the silo alarm level of the DH11. Under high input rates, the Z80 does not necessarily contemplate interrupting the MC68000 every character. Instead, it waits until some number of characters are placed into the FIFO, or until some unit of time expires. In order to minimize the speed with which the system responds to an XOFF (control-S), the insertion of an XOFF character into the FIFO precipitates an interrupt regardless of the FIFO alarm level.

3.1.2. Output.

Like the input function, the serial output interface presented by the Z80 is patterned after the DH11. An output request includes a pointer to MC68000 physical memory, along with a count of the number of bytes to be transferred. The MC68000 can stop an output request in progress using a control register separate from the line's CSR; when the Z80 signals the end of a transmission with an interrupt, it passes back to the MC68000 the number of characters actually transferred. If the MC68000 requests that the Z80 stop an output request in progress, then the Z80 will interrupt with the number of characters actually transferred before the stop request was processed.

In order to take advantage of the DH-like serial interface, the size of the standard character buffer structure was doubled; approximately 30 characters can be transmitted per Z80 request.

3.1.3. Modem Control.

The initial release of the TRS-XENIX system does not fully support modem control signals. However, the registers required for support were designed in from the start.

There are two byte-sized modem control registers for each serial line. One is writeable only by the MC68000, and is used to set and clear the various modem control outputs. The other is writeable only by the Z80, and is used to pass the status of the modem control inputs back to the MC68000. A full implementation of modem control support could either poll the modem control inputs at fixed time intervals, or else pass modem control transition information to the MC68000 through the status bits in the receiver FIFO entries.

4. THE DISK INTERFACE.

The Radio Shack hardware has separate controllers for the hard disk subsystem and floppy disk subsystem. Therefore, the TRS-XENIX system supports the overlapping of a single hard disk request with a single floppy disk request. The disk system was designed so that the same code could handle hard disks of various sizes and floppy disks of various formats, without requiring the novice user to know anything about the issues involved.

The parameter blocks and interface methods used by the MC68000 for the hard and floppy disk systems are absolutely identical. The only distinction made between the hard and floppy disks is in the handling of errors. There are two disk parameter blocks, one for the floppy disks and one for the hard disks.

4.1. Dynamic Configuration.

The TRS-XENIX system user needs to learn only one set of device names for disks. For example, the file /dev/fd0 refers to the floppy disk in drive 0, regardless of its format or capacity. The device drivers automatically adjust to the format, geometry, and capacity of a disk drive at open time.

When the TRS-XENIX system opens a disk drive which is not already open, it issues a special "INIT" call to the Z80. The Z80 returns disk geometry information to the MC68000. This information includes write protection status, a mixed-format indicator, cylinder count, sector size, track size, and number of tracks per cylinder. The kernel uses this data to build internal disk size and partition tables, as well as check for a write protection of a drive opened for writing.

Of the various pieces of disk geometry information, one needs explanation. The mixed-format indicator applies only to floppy disks. If set, it indicates that the given disk geometry information does not apply to track 0 of side 0; that track is recorded in single density format with 26 sectors of 128 bytes each. IBM standard double density floppies are formatted in this manner, as are standard TRSDOS and TRS-XENIX floppy disks.

The disk partition tables built for a disk drive have four defined partitions, with four more reserved for future use. One partition is for a reserved boot area. This partition refers to side 0, track 0 of mixed format floppy disks, and to the first two tracks of a hard disk.

Another refers to the entire disk, minus the reserved boot area. On hard disks, the sum of the size of these two partitions will not equal the size of the entire physical disk. This apparent discrepancy arises because several tracks are reserved for bad-track forwarding.

The remaining two partitions are used to refer to a file system, if any, on the non-boot portion of the disk, and to the remaining disk blocks which follow that file system.

4.1.1. Hard Disk.

For the hard disk, the Z80 determines disk geometry information by reading it from a special area of track 0 of the hard disk. This area is written at format time, and includes a cylinder count and head count for the drive. The standard hard disk formatting program requests this information from the user, referring her to her hard disk manual for this data. The formatter also records a bad track map on track 0, based on user input and its own discovery of flawed areas.

4.1.2. Floppy Disk.

The Z80 determines the format of floppy disks through a trial and error technique, rather than by reading pre-recorded information. It determines density by trying to read the first sector header on both track 0 and track 1 in single and double density. It reading the header information, it also learns the sector and track sizes. The floppy disk drive itself indicates if a disk is double-sided.

4.1.3. Root and Swap Devices.

The sizes of the file system-related partitions are determined by looking for a valid superblock at the start of the non-boot area of a disk. These partitions are principally used so that the kernel can dynamically adjust to the start and size of the swap area. In a structure separate from the disk parameter block, the Z80 passes to the MC68000 an indication of which disk drive was used to boot the system. The distributed TRS-XENIX kernel binary places the root and swap areas on the boot disk, although a user can easily configure a system which ignores the boot disk information.

4.2. Normal Operation

The MC68000 requests the Z80 to initiate a transfer to or from any arbitrary MC68000 address. The transfer may include any number of sectors, and the Z80 will transparently take care of transfers which cross track boundaries on any disk. The Z80 also handles bad track forwarding on the hard disk.

4.2.1. Write Verification.

On any disk write request to the Z80, the MC68000 can include a request for read verification of all data written. The TRS-XENIX system includes a utility which turns on and off write-verification for the floppy disk subsystem and/or the hard disk subsystem. Write-verification can be useful on floppy disks in particular, although it does very seriously impact disk performance. (Without write-verification, the TRS-XENIX system can use the "raw" disk interface to read or write an entire track in one revolution of the floppy disk.)

4.3. Floppy Disk Drive Configuration.

With each disk request, the MC68000 specifies disk seek rate, whether or not to wait for drive motor spin up, and whether or not the drive can detect the occurrence of a door-open condition between accesses. This data is fixed for hard disks, but varies for the assorted floppy disk drives which Radio Shack has sold. The distributed TRS-XENIX kernel assumes "safe" values for these constants, which will work—although not necessarily optimally— with any Radio Shack floppy drive. A utility is provided which both configures the running TRS-XENIX kernel for the actual floppy disk parameters of the user's system, and patches the file system copy of the kernel. The user runs this utility once upon system installation, and possibly additional times when new floppy drives are added to her system.

4.4. Disk Errors—Removal of Active Disk.

Almost any UNIX system equipped with floppy disks presents an accident waiting to happen. What do you do when the user removes a floppy disk from an open, active drive? Can you detect a floppy disk change, when the original floppy was holding a mounted file system?

Current Radio Shack floppy drives provide a disk changed indicator, alerting the system that a drive door was opened since the last access to that drive. Other drives provide no such warning; unless you happen to try to access the drive at a moment when there is no inserted floppy disk, you will not notice any special condition.

If the Z80 detects that a drive door has been opened since the last access, it rejects all MC68000 requests except for the already-described INIT call. If the MC68000 receives a disk-changed or drive-not-ready error on an open drive, it prints a console error message and flags that drive. All further I/O to that floppy drive is aborted until the drive is closed and reopened. (Messages are printed when accesses are disabled, and later when the drive status is reset.) This behavior protects the user from scribbling unexpectedly on a newly inserted floppy disk, or from crashing the system because of inconsistent file systems on the removed and inserted floppy disk. However, it does not provide a graceful way for the user to recover from her possible mistake. The floppy disk error behavior may change in future releases.

5. THE CLOCK.

The Z80 provides a standard constant frequency clock interrupt to the MC68000 for time-keeping purposes. However, this interrupt has associated with it a simple data structure giving it somewhat more power than a simple interrupt line.

5.1. No Loss of Clock Ticks.

On each interrupt, the Z80 provides the 68000 with an integer number of clock ticks which have occurred since the last acknowledged tick. That is to say, at each timer interval, the Z80 checks the clock INTERRUPTED bit. If this bit has not been cleared by the MC68000, the TRS-XENIX kernel would normally miss this clock tick, for it has not serviced the last one. However, the Z80, upon detecting this condition, arranges to indicate two clock ticks on the next interrupt. The MC68000 will increment the time by two clock ticks on that next interrupt, thereby losing no time.

5.2. Crash Detection.

The dual processor design of the Model 16 gave Microsoft the ability to build sophisticated debugging tools for our internal development use. In particular, it gave us the ability to detect a kernel crash and dump MC68000 physical memory to floppy for later debugging. The Z80 processor assumes that the MC68000 has crashed if after enabling clock interrupts, the MC68000 ignores some number of consecutive clock ticks.

6. THE FUTURE.

Like all software systems, the initial release of the TRS-XENIX I/O system can be improved in a number of ways. Tandy may add additional features, such as full support for modem control, as described above. Of perhaps greater interest would be any schemes used by Tandy for improving I/O throughput and/or decreasing various forms of system overhead.

6.1. Use of MC68000 Interrupt to Z80.

As already stated, the initial release of the TRS-XENIX system does not take advantage of the Z80 interrupt line controlled by the MC68000 on Model 16's and upgraded Model 12's. The reason stated for ignoring this hardware feature is that it is not supported on upgraded Model II's.

The initial release of the system already has code which determines whether or not it is running on an upgraded Model II. Slight differences in dealing with the serial lines require the Z80 code to determine what sort of processor board it is running on. Having determined this piece of information, the I/O system could take advantage of it, by requiring the MC68000 to generate a Z80 interrupt after setting up any command. This change would eliminate the need for the Z80 to always run a polling loop examining MC68000 memory. The elimination of these accesses to the MC68000's address space would eliminate some cycle stealing and would increase effective speed of the MC68000 processor.

6.2. Command FIFO.

Whether or not the MC68000 can interrupt the Z80 to indicate the presence of a new I/O request, the Z80 must currently poll all device parameter blocks to find the new request. One could increase the response speed to a new request by building a command FIFO between the two processors. Thus when the MC68000 issued a request, it would first set up the parameter block, then put a pointer to that parameter block into the command FIFO, and only then (possibly) interrupt the Z80. The Z80 would no longer have to scan over the set of all devices to find the actual request made.

6.3. Track Buffering.

Room has been carefully left in the Z80 code to allow for the future addition of one or two floppy disk track buffers. If the Z80 did track buffering, then file system access to floppy disks would speed up. The floppy file system interleave factor of two to one would effectively be eliminated.