THE SCORPION

PART III: COMMANDS WITH RESPONSES

Harprit S. Sandhu
PO Box 4010
3402 North Mattis
Champaign, Illinois 61820

The previous two articles in this series have discussed how the Scorpion was designed, and described the first half of the Scorpion's instruction set. Part three, the last in the series, discusses the second half of the instruction set and describes how to extend the language.

Resetting the System. To avoid having to leave an externally connected terminal every time you need to reset the Scorpion, a reset instruction is provided. The "/1R" instruction is used to reset the Scorpion.

The reset instruction initializes all programmable memory and returns the system to power-up condition.

There will be situations during which the system cannot be reset through software. The reset instruction will not work if the operating system branches out of the loop that scans the UART for incoming characters or if the operating system has been adversely modified.

Set/Clear the Master I/O Byte. To conserve battery power, we have provided the ability to shut off all extraneous devices using power. This feature is controlled through the Master I/O byte.

Setting the Master I/O byte to hexadecimal "00" permits all I/O operations. Setting this byte to hexadecimal "FF" inhibits all I/O operations. The most significant bit overrides all other bits. Setting the most significant bit to "1" inhibits all other bits. The eight bits are assigned as follows:

- Bit 0—Speaker.
- Bit 1—Eyes.
- Bit 2—Lamps for ground tracker.
- Bit 3—Horizontal scanner.
- Bit 4—Vertical scanner.
- Bit 5—Left drive motor.
- Bit 6—Right drive motor.
- Bit 7—Master bit. Overrides all others.

Changing the value of any bit from "1" to "0" allows that device to operate. Setting the byte to "00" allows everything to operate. If bit seven is "1," all devices are inhibited. Thus, all activity can be inhibited independently of setting each bit.

The instruction for modifying the I/O byte is defined as "/3Yhh" where:

- "/" indicates a new instruction.
- "3" indicates three bytes will follow.
- "Y" indicates the I/O byte.
- "hh" indicates the hexadecimal value to be stored in the Master I/O byte.

Obey/Ignore Master I/O Byte. It is sometimes necessary to ignore the Master I/O byte status. This can be done with the "/3Zhh" command which is interpreted as follows:

- "/" indicates a new instruction.
- "3" indicates three data bytes follow.
- "Z" indicates the Obey/Ignore command.
- "hh" indicates the hexadecimal value of the bits used to determine whether the Master I/O byte is obeyed or ignored. Each bit of the "/3Zhh" command corresponds to a bit in the Master I/O byte. Setting a bit in the "/3Zhh" command to "0" means to ignore the corresponding bit in the Master I/O byte, setting the bit to "1" indicates the bit should not be ignored.

Moving Motors a Specified Amount. In part two, I described the speed control command which is used to turn on a motor and keep it running. Another command exists for moving motors a specific number of steps.

The instruction for moving the left motor is similar to "/8Cdff####," where:

- "/" indicates a new instruction.
- "8" indicates eight data bytes follow.
- "C" indicates the right motor is to be moved.
- "d" indicates the direction of travel ("+" or "−").
- "ff" indicates the speed at which the motor is moved ("00" to "99").
- "####" indicates the number of steps that the motor is to move ("0000" to "9999").

The instruction for moving the right motor is similar, except that the "C" is replaced with a "D."

If the specific move instruction is received while a motor is running in continuous mode, the following actions occur:

- A flag is set which indicates that the specific move mode is in effect.
- The specific move is initiated and completed.
- The flag is cleared indicating that the system is back in continuous move mode.
- The motor continues turning as before. The count indicating how far the motor has moved is not disturbed since the specific move is not added to it.

This software design means you can execute a 90-degree turn in the middle of a continuous move. The turn can be executed either forward or backward. This is a very useful maneuver within collision recovery or obstacle avoidance routines.

The Questions. In addition to motion control commands, we can ask the Scorpion questions about its system. When we give the Scorpion an instruction, the response comes from the electromechanical part of the system. When we ask a question, the response comes in the form of information that is received at the computer's RS-232 port. When we ask a question, we must be ready to receive an answer.

Reading Microswitches. The bumpers which surround the Scorpion are attached to a set of microswitches. When a microswitch is closed, the computer shuts off all energy to all output devices. This is done to conserve battery power and to protect the Scorpion from damage. The shutoff is accomplished by setting bit seven of the Master I/O byte to "1." Whenever the value of this bit is "1," all output to the speaker, eyes, tracker, and motors is inhibited. However, you can still communicate with the host computer over the RS-232 line. When bit seven of the Master I/O byte is set to "0," all output returns to its original state.

To recover from a collision, we must first decide on a recovery procedure. The Master I/O byte must be ignored since the microswitch will keep turning the system off. We can now back the Scorpion away from the obstruction and maneuver it to a position where it will not hit the obstruction again. Once the recovery is complete, we reset the Master I/O byte and start obeying it again.

The instruction to interrogate the microswitches is "/1B." The Scorpion responds with the phrase "Bhh." "B" indicates an answer related to the bumpers and "hh" represents bumper data as one hexadecimal byte. Each bit in the byte represents one of the eight microswitches.

Reading the Scanner Buffer. The infrared scanner is read by reading the scanner buffer. The question returns the contents of the scanner buffer. Part two of this article described the Scan instruction used to move the scanner motors and enter the data into the scanner buffer. The Scan instruction also determines the number of valid bytes that are in the buffer. For example, if a scan of 15 was requested from the horizontal scanner, the scanner buffer will contain 31 valid bytes of information.

The instruction for requesting the scanner buffer contents is "/1S." The Scorpion responds with "Sdhhhh...." where:

- "S" indicates a scan response.
- "d" indicates the scan direction, "H" for horizontal or "V" for vertical.
- "hh," each byte pair represents a single hexadecimal value. From one to 199 bytes of data may be returned, depending on the length of the previous scan.

Reading the Eyes. The instruction for reading the eye values is "/1E." The Scorpion responds with "E#," where:

- "E" indicates an eye response.
- "#" is an ASCII character which indicates the status for both eyes. "0" means both eyes are off, "1" means only the right eye is on, "2" means only the left eye is on, "3" means both the left and right eyes are on.

Reading the Tracker Lamps. The instruction for reading the tracker lamps is "/1G." The Scorpion answers "G#," where:

- "G" indicates a tracker lamp response.
- "#" is an ASCII character which indicates the status of both lamps. "0" means both lamps are off, "1" means both lamps are on.

Reading the Ground Tracker Phototransistors. The instruction for reading the phototransistors is "/1T." The Scorpion responds with "T#," where:

- "T" indicates a ground tracker response.
- "#" is an ASCII character which indicates the status of the phototransistors. The returned values are similar to that returned when the eye status is requested.

Reading the Motor Move Status. The instruction for reading how far the motors have moved since the last motor interrogation is "/1M." The Scorpion responds with an answer which looks like "Ms#### s####."

- "M" indicates a motor response.
- "s####" each group of five characters indicates the direction "s" and number of steps "####" moved since the last interrogation. The first group contains information about the left motor and the

second group of five contains information about the right motor.

The motor count registers are cleared after each interrogation. However, the Scorpion does remember the last move, just in case a transmission error occurs. This value can be read with a RAM Read instruction.

Reading the Busy Byte. It is often necessary to determine what processes are still in progress so that the next instruction can be delayed until the ongoing process is completed. Whenever the Scorpion starts an activity, it sets a flag which can be read by the host computer. The byte containing these flags is called the Busy Byte.

The instruction for reading the Busy Byte is "/1X." The Scorpion responds with "Xhh," where:

- "X" indicates an execution response.
- "hh" is the hexadecimal representation for one data byte.

Each bit in the returned data byte represents a particular function. If the bit is set to one, then the function is in progress. If the bit is set to zero, the particular function is not in progress. The eight bits are defined as follows.

- Bit 0—speaker tone in progress.
- Bit 1—at least one eye is on.
- Bit 2—the ground tracker lamps are on.
- Bit 3—the horizontal scanner is scanning, moving, or resetting.
- Bit 4—the vertical scanner is scanning, moving, or resetting.
- Bit 5—the left drive motor is on.
- Bit 6—the right drive motor is on.
- Bit 7—something is going on.

Check other bits to see what it is.

This information is useful to determine whan a scan or some other activity is completed so the next scan or activity will not overrun the one in progress.

Reading Memory. It is often necessary to know what various locations in memory contains. The instruction to read memory locations is "/7Raaaa##," where:

- "/" indicates an instruction.
- "7" indicates seven bytes of data follow.
- "R" indicates the memory (RAM) interrogation question.
- "aaaa" indicates the hexadecimal starting address in memory.
- "##" indicates the number of bytes to be read.

The Scorpion responds with:

"Rhhhh....,"

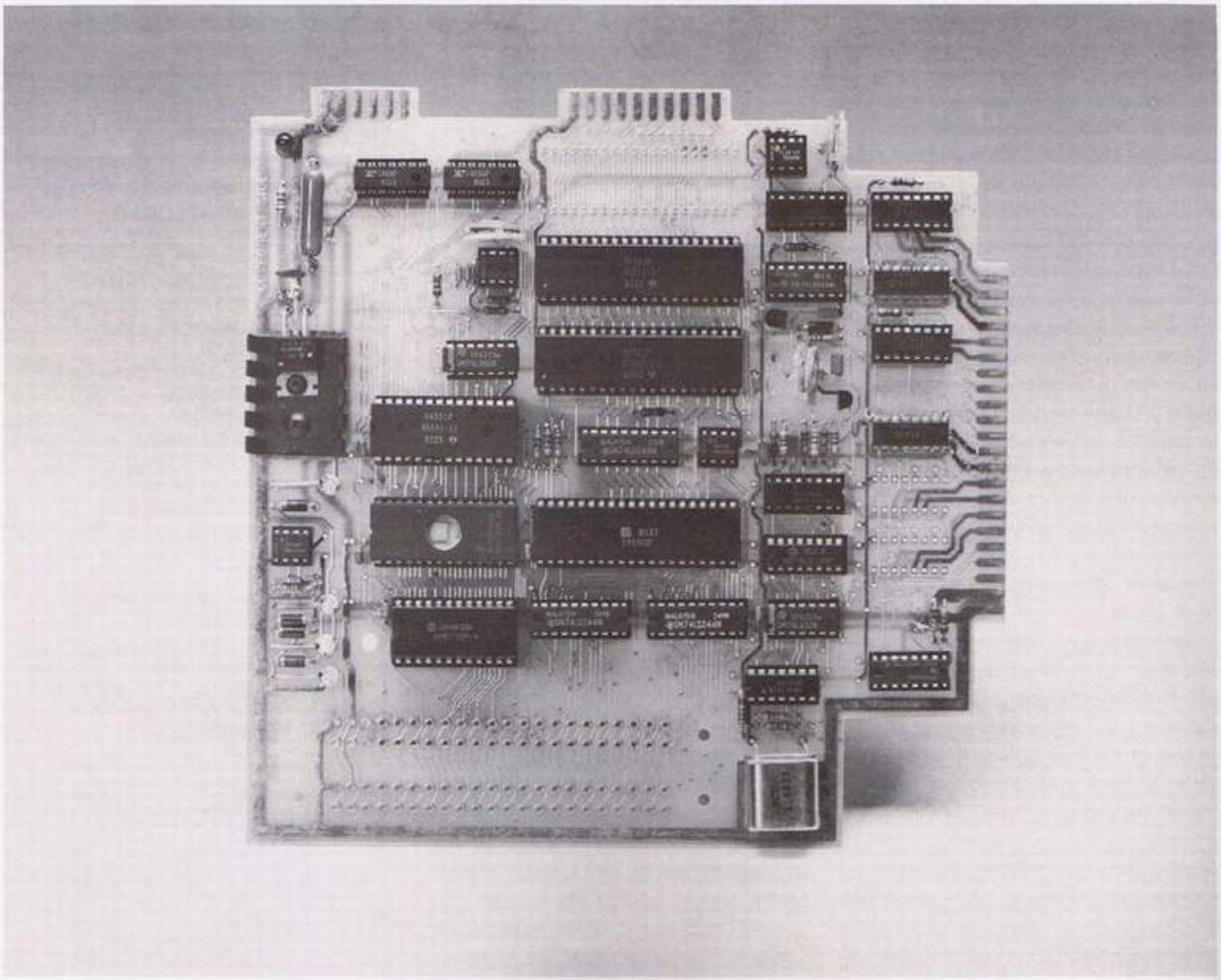

Photo 1. The 44-pin KIM-bus extension on the Scorpion controller card allows this system to be extended to a full 64 Kbyte microprocessor system. The spare 4202 stepper motor controllers can be used to control additional motors. A circuit pad is also provided for an additional 6522 which can be used to expand the Scorpion's input/output capacity.

where:

- "R" indicates information coming from memory locations.
- "hh," each byte pair represents a single hexadecimal value. The value represents the data contained in the memory locations.

Reading the Remaining Motor Move. When we are using a specific move command to move a motor a specific number of steps, we must have a way to determine when the move is complete so that we will not send the motor another move command until it has completed the first. The "/1C" instruction is used to determine how many steps are left until the left motor is finished. The "/1D" instruction returns similar information for the right motor.

The Scorpion responds with "C####" for the left motor and "D####" for the right motor. The four digits, represented by "####", indicate the number of steps still remaining for this motor. This number can range from "0000" to "9999".

Reading the Entire Machine. Since the Scorpion maintains the status of all parameters, an instruction has been provided which reads *almost* all parameters at once. The optical scanner data and memory data are not returned. This instruction, "/1P", allows almost all processing to be done at the host computer in one step.

The Scorpion responds to the "/1P" command with the answer

"PBhhE#G#T###Ms####s####
YhhXhhC####D####."

The "P" indicates all parameters are being returned. Each individual parameter is returned as described previously.

Language Extensions. Though it's fine to play with a machine that is all designed and laid out in front of you, the real fun comes from modifying the basic elements to your own liking. In order to do this, you must be able to modify the system hardware and software. We have made every effort to allow for easy modifications to the Scorpion. Of these modifications, the most interesting is probably the ability to modify the operating system.

The Scorpion can be sent commands which are appended to the built-in instruction set. When the Scorpion computer receives an unknown command, the command processing loop eventually ends up at a routine which assumes that the current instruction is an error. The erroneous instruction is then cleared and the routine jumps back to the top of the control loop. Since all unknown commands eventually reach this point, we can add new commands at this point.

The part of the control loop code which is contained in programmable memory can be altered with the "/7R..." command. You have to design the necessary 6502 code and provide the appropriate command interpretation and response. After you are done, you must provide the appropriate jump commands to re-enter the main control loop.

This ability to extend the language is necessary because the mother board provides places for connectors that allow the entire system bus to be extended to other peripheral cards. These cards may have a host of sophisticated capabilities ranging from distance sensing equipment to large memory banks. The simple method of altering the instruction set provides a way to control these external functions.

Your Own Operating System. In addition to extending the language instruction set, you can also create your own operating system for the Scorpion. This is an excellent exercise for anyone interested in robotics. The Scorpion provides tools for developing your own operating system in programmable memory without altering the original operating system stored in read-only memory.

The Scorpion is a versatile educational tool which can be used for simple experimentation with a roving robot or in-depth exploration of a robot's control structure.

The software for the Scorpion was designed by Foad Rekabi, BSEE, University of Illinois. The computer hardware was designed by Bob Syeszycki, BSEE, University of Illinois. The Scorpion's overall design was the responsibility of H. S. Sandhu, BS, MS, University of Illinois.

Reader Feedback

To rate this article, circle the appropriate number on the Reader Service card.

53	63	73
Excellent	Good	Fair