An on-chip high-level language sounds too good to be true. But a Tiny Basic in-situ interpreter is now at work to banish both prototype boards and development systems. Programming becomes easier, too.

On-chip Tiny Basic dumps development systems

Programs for a single-chip microcomputer can be written and debugged using little more than an external CRT with National Semiconductor's INS8073 Microinterpreter, which contains an on-chip 2.5-k Tiny Basic interpreter. A high-level language right on the silicon eliminates two expensive needs: a development system and a prototype board that is significantly different from the production board. And thanks to the interpreter, writing programs in NSC Tiny Basic also eliminates the need for such additional software as editor, assembler, or debugging programs.

Program debugging is simplified because NSC Tiny Basic programs, by being interpreted, do not need to be reassembled or recompiled each time a change is made. Program execution can be suspended, variables and other parameters examined and altered, and execution resumed at the point at which it was interrupted.

Program writing speeds up because of Tiny Basic's English-like simplicity. Editing and debugging are likewise aided since programs are stored as a sequence of ASCII characters that can be quickly checked for validity. Unlike most assembly-language programs in ROM, NSC Tiny Basic programs can be relocated, loaded, and executed anywhere in memory. This keeps Tiny Basic ROMs from becoming obsolete if their address locations must be changed in the final system.

Although Tiny Basic on the INS8073 executes more slowly than optimized machine languages, most applications for single-chip microcomputers are not very time-critical. For those applications that do require higher speed, NSC Tiny Basic can link to assembly-language routines via a LINK command, LINK EXPR>.

Ron Pasqualini, Microprocessor Design Engineer
Bruce Edmundson, Software Engineer
Jim Handy, Product Marketing Engineer
National Semiconductor
2900 Semiconductor Dr., Santa Clara, CA 95051

The INS8073 is the latest member of National Semiconductor's Series 70 family of single-chip microcomputers, whose architectures and instruction sets are specifically designed for applications requiring high-level language capability.

The 40-pin device incorporates both on-chip ROM (2.5 kbytes, which store the NSC Tiny Basic interpreter), and 64 bytes of RAM (Fig. 1). In addition, it provides an 8-bit arithmetic logic unit, an 8-bit accumulator, an 8-bit extension register, and 15 internal registers.

A system is easily expanded using standard peripherals and the INS8073's 16 address lines and eight data lines. Separate read and write strobe outputs indicate valid input/output data on the 8-bit data bus. The remaining I/O lines are dedicated to initialization, bus management, interrupt request, I/O cycle extension, and software-controlled I/O.

On the machine-language level, the INS8073 provides a set of 192, 8 and 16-bit instructions, which, in addition to register manipulation and Boolean operations, allow multiple addressing modes and 8 and 16-bit stack operations. DMA and multiprocessing systems can be implemented with almost no external hardware.

Minimum RAM-based system

A minimum RAM-based system consists of the INS8073 and enough external RAM to store the user's program plus 256 RAM bytes required by NSC Tiny Basic (Fig. 2). Only simple buffers are needed to translate the serial-I/O signals to RS-232 levels or to the current-loop levels for a TTY terminal. One kilobyte of external RAM can store a typical 30 to 60-line program.

External RAM can be located anywhere in the address range 1000H to FFBFH (approximately 60,000 bytes), but it must be contiguous. After power-up or after a NEW command has been entered, Tiny

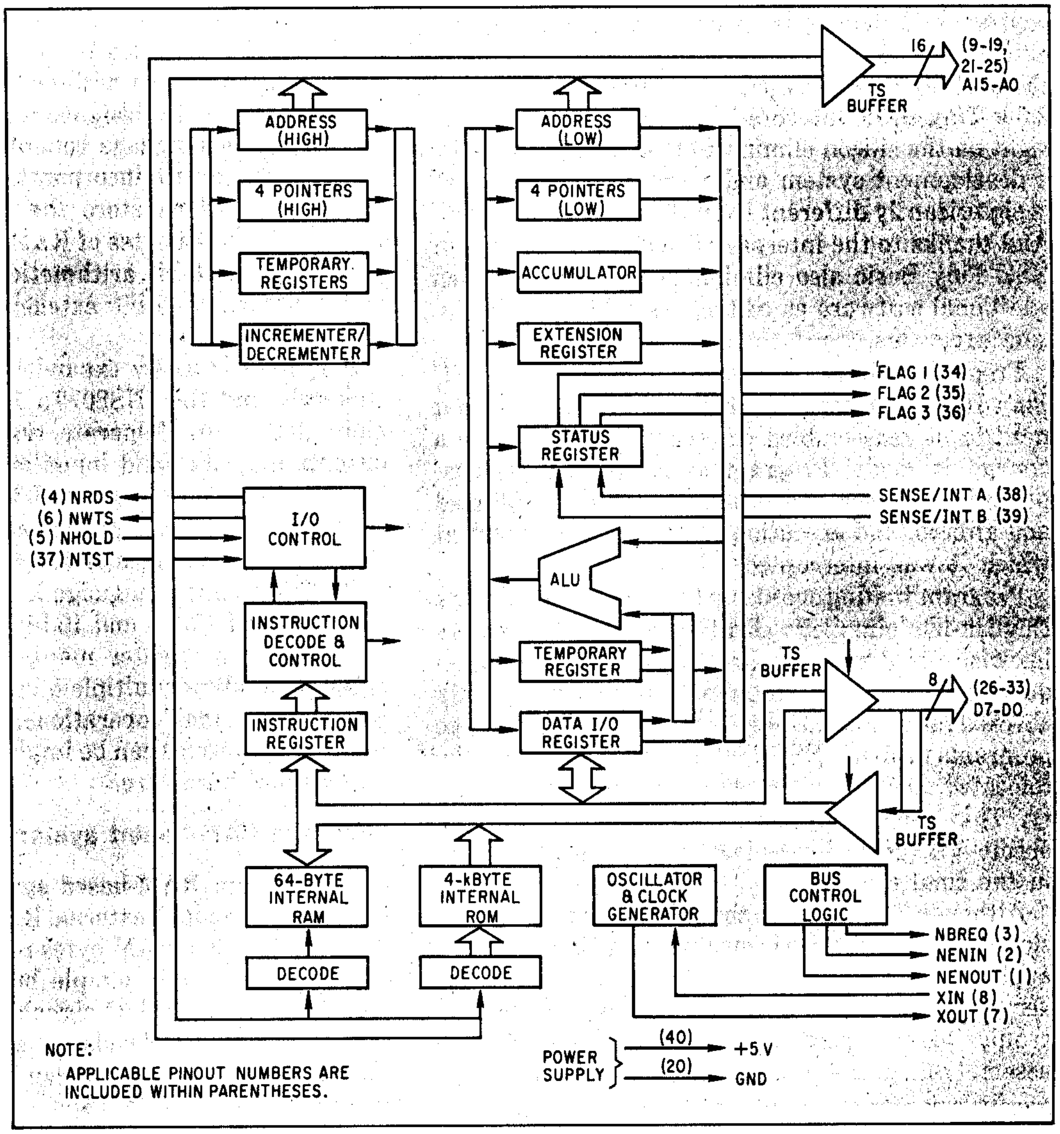
Basic will automatically determine the location and amount of external program RAM. After the limits of external RAM have been determined, the interpreter warns the user if he attempts to enter program lines beyond these limits.

When a TTY is employed for program I/O, a RAM-resident program may be saved on paper tape; what's more, because NSC Tiny Basic treats paper tape like a keyboard input, programs can be entered into RAM directly from paper tape as well. To ensure that all characters are properly accepted, Tiny Basic controls the paper-tape-reader relay. RAM-resident programs can also be saved on audio cassette tape or EPROM.

For real-time control applications, in which a terminal is not normally present, programs can be stored in ROM/EPROM. Execution will be automatically initiated following power-up if external ROM is located at memory location 8000H, because Tiny Basic examines this location following power-up to determine if it contains ROM. If so, execution automatically commences.

A minimum ROM-based system consists of three devices: the INS8073, a program ROM or EPROM, and at least 256 bytes of RAM (Fig. 3).

I/O capabilities differ for each system. Programs in the RAM-based system are transferred to or from the terminal via the F1 output and SA/INTA (sense/interrupt) input. Output F2 is used to enable or disable the paper-tape-reader relay. Consequently, the nonterminal related I/O resources of the minimum-RAM-based system consist of one sense



1. The INS8073 architecture includes clock and bus control logic in addition to the Tiny Basic interpreter in ROM and 64 bytes of on-chip RAM.

Behind the scenes

The NSC Tiny Basic interpreter resides within the INS8073 on-chip ROM at memory locations 0000H-09FFH. It uses all 64 bytes of on-chip RAM (FFCOH-FFFFH) and requires at least 256 bytes of additional external RAM.

At power-on or reset, Tiny Basic performs a non-destructive memory search to determine the address range of the external RAM, which may be located anywhere in memory above the first 4 kbytes. The first 256 bytes of external RAM are used to store variables, stacks, and buffers; the remainder may be used for program storage. Memory must be contiguous, since the nondestructive memory search will identify only the first contiguous RAM block located at or above address 1000H.

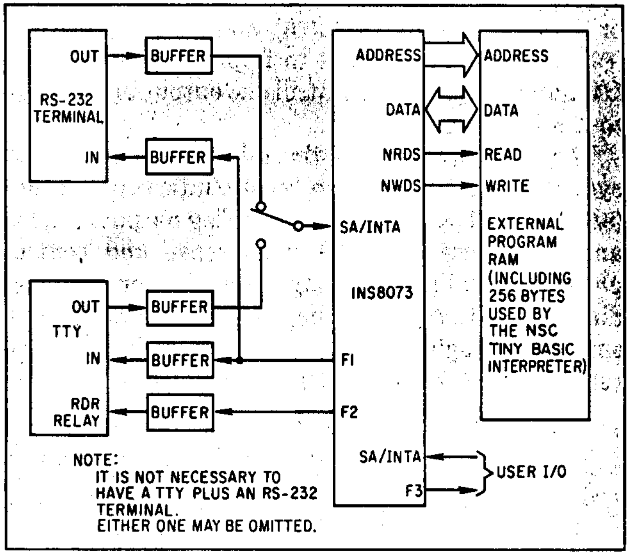
Because the INS8073 will execute a ROM-based program at power-on, it's ideal as a stand-alone real-time controller. After Tiny Basic locates external RAM, it tests address 8000H to determine if this location is RAM. If it is, Tiny Basic enters the command mode. If the address is not RAM, Tiny Basic assumes that it must be ROM, and attempts to execute the program stored there. If there is no ROM at 8000H, the interpreter detects that the first valid character is not the start of a line number, and enters the command mode.

Before a new program can be entered from the console, NSC Tiny Basic must be initialized by entering two commands. The first command, NEW ADDR, establishes the program starting address. The second command, NEW (CR), establishes the initial value of the program ending address. The NEW (CR) command writes the byte 7FH (the end-of-program indicator) at the program ending address, which is initially set equal to the program starting address, and is automatically incremented or decremented as new statements are added to or deleted from the program.

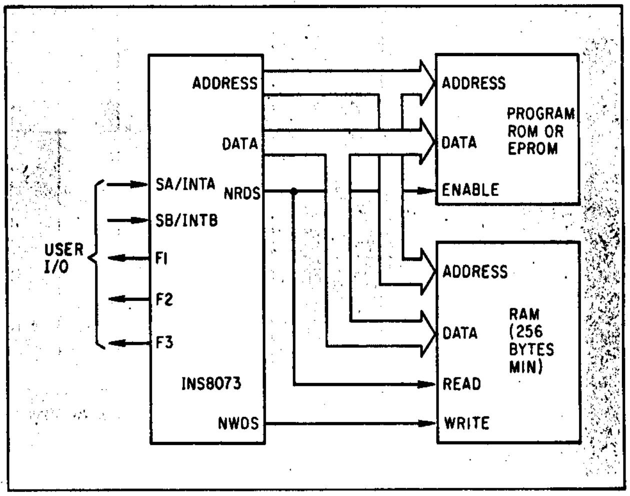
If a number of programs already exist in RAM, any one of them can be nondestructively executed by entering the NEW address command, followed by the RUN command. The parameter address must equal the starting address of the program to be executed. It is unnecessary to enter the RUN command to execute a ROM-resident program, since ROM programs will be executed immediately.

Tiny Basic has built-in I/O routines to serially interface with an RS-232 terminal or TTY at 110, 300, 1200, or 4800 baud. To interface with the wide variety of other peripherals, optional user-supplied I/O routines can be specified as assembly-language routines, which must be stored in ROM to service the print and input statements.

A "minimum" system—which calls for fewer components—still provides for a TTY interface plus an audio-cassette interface, an EPROM programmer, and decoding. Selection of the baud rate is automatic. Of course, the minimum configuration imposes certain restraints on the system design.



2. At least 256 bytes of external RAM are needed for the variables and stack used by Tiny Basic. The rest is available for user programs.



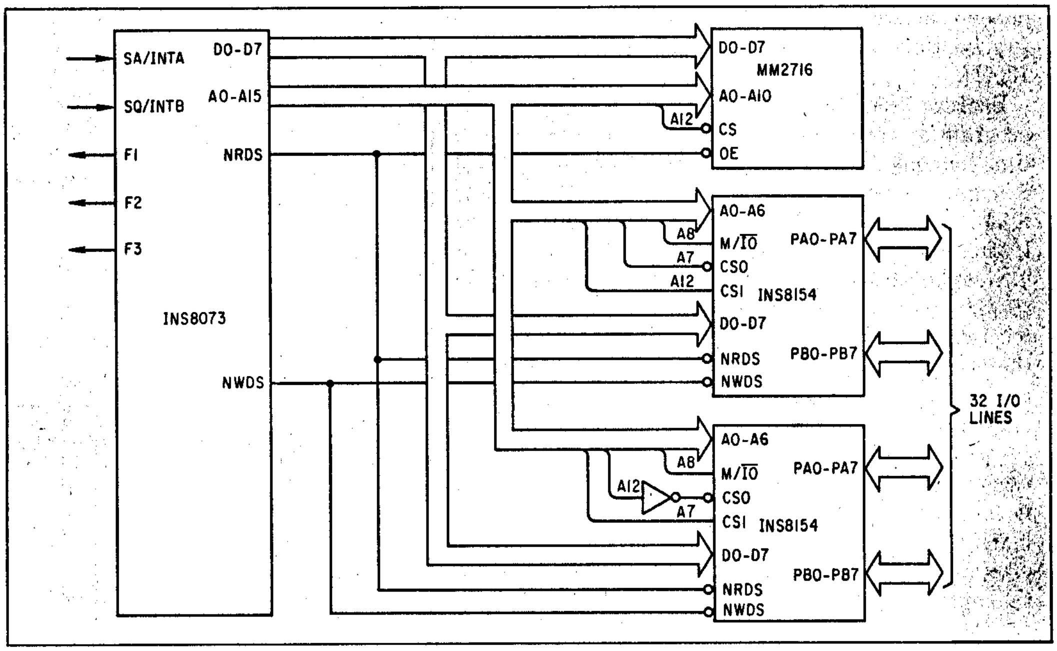
3. If no terminal is needed for the ROM-based system, additional I/O lines are available for control.

or interrupt input (SB/INTB) and one flag output (F3), which are sufficient to implement one serial-I/O data channel or to control one output in response to a single input.

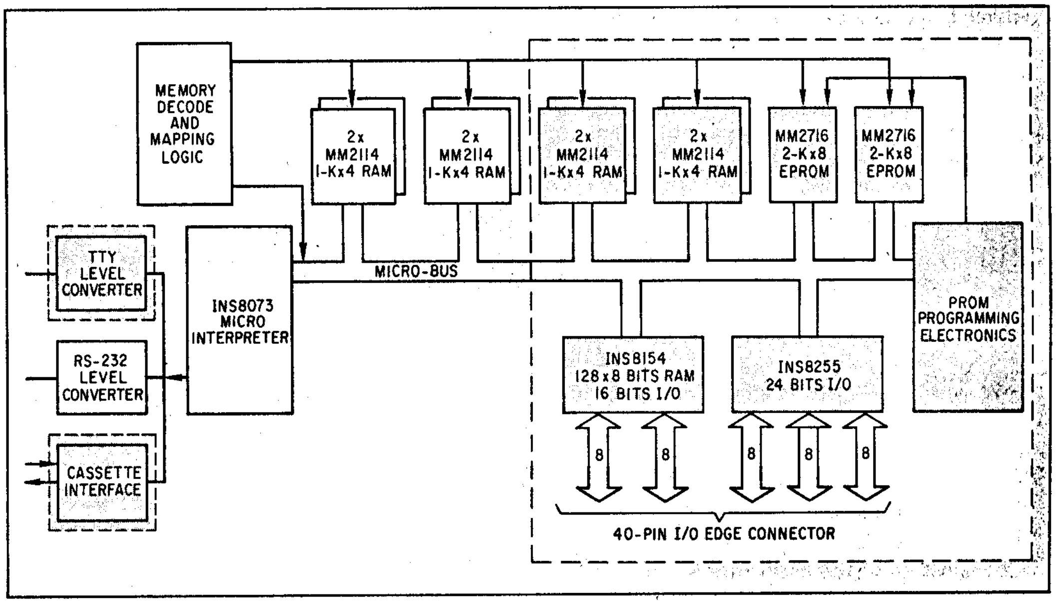
The I/O capability of the minimum ROM-based system consists of two sense/interrupt inputs (SA/INTA and SB/INTB) and three flag outputs (F1, F2, and F3). These five I/O lines sense and control physical variables including pressure, temperature, displacement, rpm, torque, etc. Each flag output supplies enough drive current for Darlington-type relay drivers, hammer drivers, and lamp drivers.

If no terminal is to be used in a ROM/EPROM-based system, the SA/INTA and F2 pins can be inhibited and used for general-purpose I/O by simply including the NSC Tiny Basic statement ON 1,0 in the program. In this case, the program should contain no INPUT or PRINT statements, because there will be no terminal input or output. When employed as a sense input, the SA/INTA pin is examined and tested by the STAT variable. If the ROM-based program is not required to output to a terminal, the F1 flag can also serve as a general-purpose output.

Expanding the I/O capability of the INS8073 for



4. Memory-mapped I/O greatly aids simple I/O expansion for increased controller versatility.



5. Devices enclosed in dotted lines are not necessary for operation, but are included to add features to the board.

any given system depends on the number and type of inputs or outputs to be sensed or controlled. I/O devices are usually "memory mapped"—treated as memory locations that are read-into to receive input, and written-into to supply output. Methods of decoding I/O addresses depend upon the type of I/O hardware to be decoded.

Figure 4 illustrates how to expand the I/O capability of the minimum ROM-based system shown in Fig. 3. Although it consists of just four LSI devices, it contains a complete high-level-language computer and control program, plus 37 I/O lines. Thirty-two of these lines come from the I/O ports on the INS8154 128 × 8-bit RAM-I/O chips. Each port line may be programmed as an input or output. Any line or port may be read or written via the NSC Tiny Basic @ operator, where the "factor" is the memory/peripheral address. In addition to providing I/O-expansion capability, the two INS8154s provide the 256 RAM bytes required by the NSC Tiny Basic interpreter.

Where serial program I/O is not desirable, the TTY or terminal can be eliminated by substituting user-written external subroutines for the internal I/O subroutines normally called by Tiny Basic. By supplying custom I/O subroutines, the user can directly interface the INS8073 to keyboards, printers, scanned displays, CRTs, and other devices.

Handling interrupts

NSC Tiny Basic supports interrupts via the ON statement in conjunction with the STAT variable. The statement STAT=STAT OR 1 will enable both the SA/INTA and SA/INTB interrupts at the "hardware" level, but not at the "software" level; the statement STAT=STAT AND #FE will disable both interrupts at the hardware level, automatically following V_{cc} power-on or after reset.

The ON statement enables interrupts at the software level. Thus, if the SA/INTA service subroutine begins on line 500, execution of the statement ON 1,500 will cause control to be transferred to this subroutine in response to an interrupt received on the SA/INTA line. Similarly, if the SB/INTB service subroutine begins on line 600, execution of the statement ON 2,600 will cause control to be transferred to this subroutine in response to an interrupt received on the SB/INTB line.

The SB/INTA and SB/INTB interrupts can be independently disabled at the software level by executing the statements ON 1,0 and ON 2,0, respectively. If interrupts are disabled, the SA/INTA SB/INTB pins can be employed as general-purpose sense inputs, which can be examined via the Tiny Basic STAT variable.

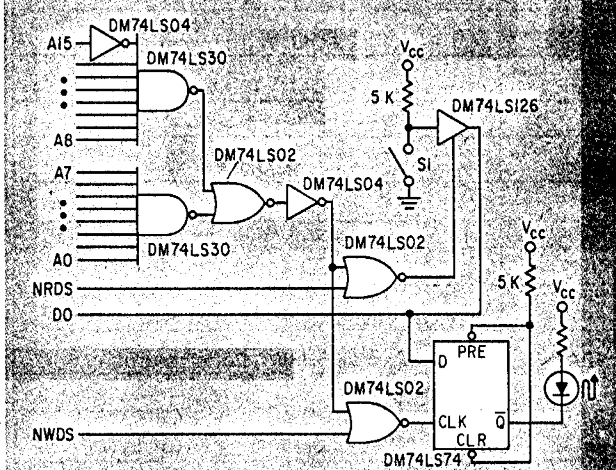
Interrupts generated by the INS8154 or the INS8255 can be connected to the SB/INTB pin of the INS8073. Then, the INS8073 SB/INTB pin may be used

Control with Tiny Basic

NSC Tiny Basic can easily sense and control external devices. When controlling an LED in response to the state of a switch, for example, the interrupt lines are not used, but the program examines a memory-mapped I/O device—a circuit that lights an LED by writing a ONE onto the data bus. The circuit is bidirectional so that the computer reads data bit 0 at that same address to determine the state of the switch. Whether data are being written to or read from that address depends on whether the read data (NRDS) or write data (NWDS) strobes are active (low).

In this example, the I/O device is at address 7FFH (only address bit 15 is 0). NSC Tiny Basic uses the @ operater to read and write to and from memory or a peripheral. Thus, typing @#7FFF would address the device in the example. Typing the command PRINT @#7FFF would result in a ONE on the terminal if the switch is open and a ZERO if it is closed. Likewise typing @#7FFF=1 will turn the LED on and @#7FFF=0 will turn it off.

Armed with this knowledge, the user can write an NSC Tiny Basic program to do any combination of things with the LED and switch. The sample program given here will blink the LED when the switch is closed. The duty cycle for the LED is determined by the NSC Tiny Basic DELAY statement DELAY<expr>, where <expr> indicates from 1 to 1040 milliseconds.




```
100 M=#7FFF:REM PUT THE DEVICE ADDRESS IN M
110 S=@M:REM SAVE THE VALUE OF THE SWITCH IN S
120 REM IF SWITCH OPEN BLINK LIGHT ON AND OFF
130 @M=S:GOSUB 210:REM LIGHT FOLLOWS SWITCH ON AND DO TIME DELAY
140 @M=Ø:GOSUB 210:TURN LIGHT OFF AND DO A TIME DELAY
150 GOTO 110
200 REM TIME DELAY SUBROUTINE
210 T=100:REM MAKE T BIGGER TO INCREASE DELAY
220 DELAY T
230 RETURN
```

to detect interrupts under control of the user's Tiny Basic program.

The upshot of all this? An extremely powerful and flexible "Basic-speaking" single-board computer can be built with no more than 19 components fitted on a single PC board no larger than 5 × 7 in. Such a system, shown in Fig. 5, does the following things:

- Allows the user to enter, debug, and execute RAM-based Tiny Basic programs up to 150 lines long.
- Directly interfaces to an RS-232 terminal and to a TTY.
- Supports multiple data rates—110, 300, 1200, and 4800 baud.

Hex Address	Memory Contents
0000 0BFF	INS8073 NSC Tiny Basic micro interpreter
	⋮
1000 13FF	RAM 0 (1 kbyte)
1400 17FF	RAM 1 (1 kbyte)
1800 1BFF	RAM 2 (1 kbyte)
1C00 1FFF	RAM 3 (1 kbyte)
2000 27FF	MM2716 EPROM programmer
	⋮
8000 87FF	ROM 0 (2 kbytes)
8800 8FFF	ROM 1 (2 kbytes)
	⋮
F700 F703	INS8255A
	⋮
FD00	Baud Rate Select
	⋮
FF00 FF7F	INS8154 RAM (128 Bytes)
FF80 FFA4	INS8154 I/O Ports/control
	⋮
FFC0 FFFF	INS8073 on-chip RAM (64 bytes)

Partitioning of the INS8073 64k addressing space

6. The INS8073 memory map places ROM starting at 8000H, where Tiny Basic will automatically begin execution.

- Directly interfaces to an inexpensive audio cassette recorder for storage and retrieval of NSC. Tiny Basic source programs and assembly level subroutines.
- Transfers RAM-resident programs into EPROM and vice versa.
- Runs an EPROM program in a real-time control application where a terminal is not present.
- Has I/O capability that makes it flexible enough to interface to most user systems.
- Provides a "scratchpad" RAM for use when assembly-language subroutines are invoked via the LINK statement.

An even smaller—but almost as powerful—system can be built using even fewer ICs plus a few discrete components. Even this "minimum" system will provide 1 kbyte of RAM and an RS-232/TTY interface, and optionally include an audio-cassette interface, an MM2716 EPROM programmer, automatic baud-rate selection, and complete decoding.

The components described can be arranged in memory in a variety of ways, but the economy gained imposes certain constraints on the system. For one thing, since decoding hardware uses a minimum number of low-cost ICs, the system components will be only partially decoded, resulting in multiple images of each component in memory. And although these multiple memory images may be present, the mapping hardware is arranged to make it impossible to enable more than one system component at a time. This also eliminates the possibility of a data-bus conflict as the result of a programming error.

In addition, NSC Tiny Basic program RAM must form a contiguous block so that the INS8073 can identify the beginning and end of the RAM that is actually present. The RAM and I/O ports of the INS8154 are located in the address range FF00H-FFA4H to allow assembly-language subroutines to directly address the INS8154 RAM I/O. Direct addressing eliminates the need to dedicate or multiplex a pointer to address the INS8154.

When on-card EPROM is present, it shall be located starting at address H8000H, where automatic execution of ROM programs begins. The system can then be used in real-time control applications without a terminal.

All these constraints are satisfied by the memory assignment shown in Fig. 6, which shows how the 64-k addressing space is to be partitioned.□