HARDTALK

BY JEFFREY MAZUR

Six months ago we started this column with a detailed look at what goes on "under the hood" of an Apple II. For the most part, you had to accept this description on faith unless you are one of the few who thoroughly understands the Apple's schematic diagrams and firmware listings.

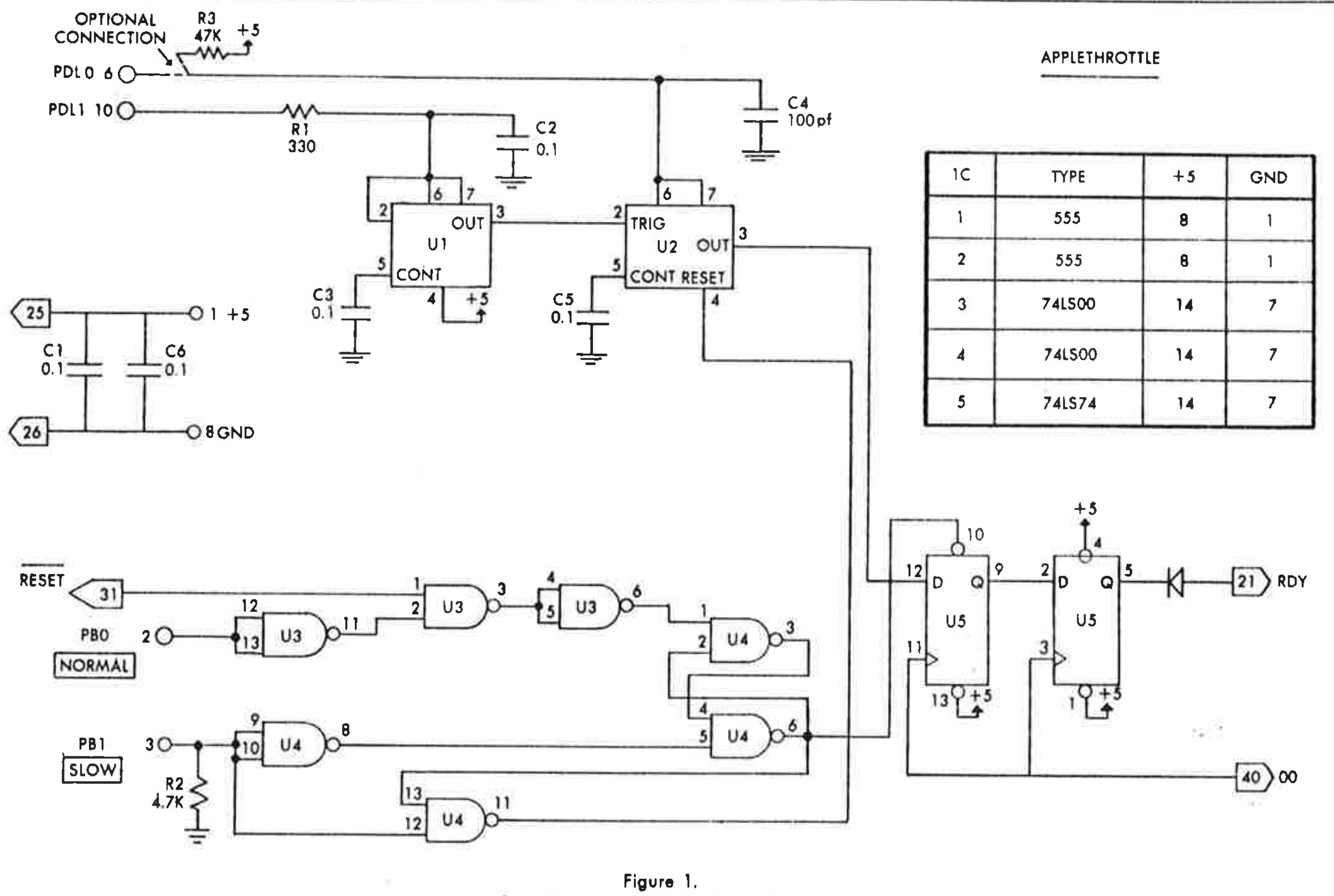
This month we will delve a little deeper into what makes the Apple tick. To aid in our investigation, we will present two hardware devices that can be used to strip away some of the mystery behind the computer. One of these devices is known as the Applethrottle. The complete schematic is included here for those who may wish to build their own.

The Applethrottle can be used to slow down or stop the operation of the cpu. This makes it possible to watch the computer work in slow motion. What normally takes place in the wink of an eye can now be examined in all of its detail.

To enable us to watch even closer, we can add a display board such as the one available from John Bell Engineering. This board contains a series of lamps that constantly monitor the address and data buses. By slowing the computer down to almost a standstill, we can actually watch the cpu executing a program—fetching instructions, loading and storing data, performing branches and jumps, and so on. Even experienced hardware and software experts are sure to find this quite interesting.

Whoa . . . Apple! Since the Apple is constantly performing between two hundred thousand and five hundred thousand operations per second, it is normally impossible to watch how the computer functions. It is possible, however, to build a device which will slow down the computer to a human pace. This is done by controlling the ReaDY line of the 6502. When this line is brought low, the cpu will halt after it finishes its current read or write operation. It will then just sit there and wait until the RDY line goes high again. This capability of a microprocessor is generally used when the cpu is connected to slow memory devices. On the Apple II, this signal is normally held high. Although the RDY line is available on all of the I/O slots, it is very rarely used by other peripheral boards. We can use it, however, to build a handy tool for understanding how the computer works.

Figure 1 shows the schematic diagram of the Applethrottle. At the top are two blocks labelled U1 and U2. These blocks represent two ICs which are known as 555 timer chips. Although U1 and U2 are both 555s, they are connected in slightly different ways to perform different functions. U1 is configured as an oscillator; that is, it puts out a continuous stream of pulses.



IC	TYPE	+5	GND
1	555	8	1
2	555	8	1
3	74LS00	14	7
4	74LS00	14	7
5	74LS74	14	7

Figure 1.
Schematic of the Applethrottle.

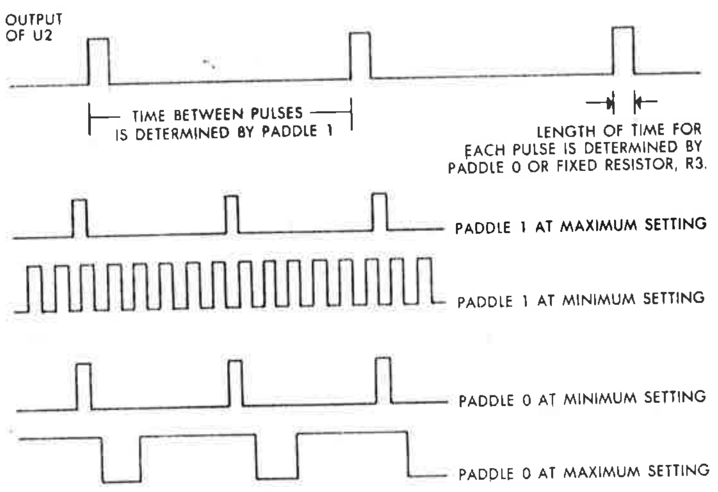

OUTPUT OF U2
TIME BETWEEN PULSES IS DETERMINED BY PADDLE 1
LENGTH OF TIME FOR EACH PULSE IS DETERMINED BY PADDLE 0 OR FIXED RESISTOR, R3.
PADDLE 1 AT MAXIMUM SETTING
PADDLE 1 AT MINIMUM SETTING
PADDLE 0 AT MINIMUM SETTING
PADDLE 0 AT MAXIMUM SETTING

Figure 2.

FIGURE 2 Possible waveforms at the output of U2. Overall observed speed of the computer depends upon the duty cycle of this signal which is the ratio of on-time to off-time.

The frequency of these pulses (how many per second) is determined by the external resistance of paddle 1. The output of U1 (at pin 3) is fed into U2 which is set up as a "one-shot" monostable circuit. This means that for every pulse coming in from U1, U2 will put out a pulse of its own.

The width of this pulse is variable and depends upon the setting of paddle 0. Furthermore, there is another signal coming into U2 at its *reset* pin. Whenever this signal is low it forces the output of U2 low. When the Applethrottle is engaged, the output of U2 will control the state of the Apple's cpu. When U2 is high, the cpu will run at full speed. When U2 is low, the cpu will stop.

The overall observed speed of the computer will depend upon how often the cpu is running versus how often it is stopped. Figure 2 shows how the various controls affect the waveform at U2's output and thus the computer's operation. The ratio of on to off time is known as the *duty cycle* of the waveform. Although both paddles can be used to vary the speed over a wide range, paddle 0 can be replaced by a fixed resistor for simpler operation.

On/off control of the Applethrottle is taken care of by U3 and U4 which provide a collection of NAND gates. To follow the logic of this circuit, remember that both inputs of a NAND gate must be high for its output to go low. Two of these gates (the ones on the right) are cross connected to form an RS (Reset/Set) latch. The output at U4 pin 6 will go low when the input at U4 pin 1 goes low. The output will then remain low until a low signal is sent into U4 pin 5. The inputs to this latch are basically derived from the two push-button inputs. Therefore, pressing the button on paddle 1 will engage the slow motion mode, while pressing the button on paddle 0 will restore normal speed operation.

Two other signals are worth mentioning. The *reset* signal from the Apple bus is also fed into the control circuit to make sure that the Applethrottle is turned off whenever a *reset* is performed. Another output from the *slow* switch (button 1) is fed into U2. This causes the cpu to halt whenever button 1 is held down. R2 is a pull-down resistor to ensure that the Applethrottle is turned off when no paddles are connected.

Unfortunately, the output of U2 cannot be fed directly onto the Apple's bus because of a timing requirement of the 6502. The specifications of this cpu require that the RDY signal change only during the phase one clock time. This means that the pulses coming out of U2 must be synchronized with the Apple's system clock. This is the job of U5, which consists of two D-type flip-flops. The first section is used to control the output to the cpu via the preset signal (pin 10). When this pin is high, it allows the variable pulse signal from U2 to pass through to the cpu. When the pin goes low (notice the small circle at pin 10 indicating that this is an active low input), the preset action causes the output to remain high regardless of the variable pulse signal. This allows the computer to operate in a normal fashion, that is, at full speed. The second half of U5 clocks the final output to the Apple bus at the trailing edge of the phase zero clock. This satisfies the timing requirements of the 6502. Finally, the ReaDY output is connected through a diode to present only an active low signal. This is necessary so that other boards connected to the bus can pull the RDY line low.

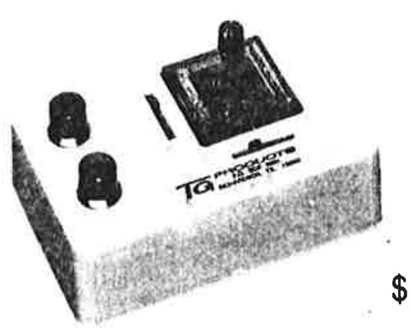
Building an Applethrottle. You can build your own Applethrottle using the schematic in figure 1. Start with a blank "prototyping board," available from Apple and other sources. Install the various components and connect them using point-to-point wiring. It is highly recommended that you use sockets for all of the ICs. For simpler operation, start with the fixed resistor in place of paddle 0. Of course, if you've never picked up a soldering iron before, you may wish to purchase an assembled unit from the manufacturer.

Initial Checkout. The Applethrottle requires a set of game paddles for its operation. These paddles plug directly into a socket on the Applethrottle. If you want to slow down games or other programs that also use paddles, you will need another set of paddles to plug into the Apple game I/O socket. After connecting the paddles to the Applethrottle, turn off the computer and remove the cover. The Applethrottle can now be inserted into any slot. Slot 7 is recommended since it is most likely to be empty and because it will allow the paddles to be easily changed from the same I/O connector to the Applethrottle. Dress the cable through one of the rear cutouts and then replace the cover.

Turn on the computer and type *call −151* if necessary to get into the Apple Monitor. If the computer seems dead or does not operate properly when it is turned on, remove the Apple-

apple compatible ® apple compatible

JOYSTICK



$59.95

- External trim adjustments to perfectly match the unit to your application and computer
- Self-centering on both axes (may be disabled)
- Two large pushbuttons
- Highest industrial quality components throughout
- Completely linear pure resistive circuit

Available at your dealer or order direct

Allow 2 weeks for shipment
All mail orders add $2.00 for postage and insurance
Dealer inquiries invited

Texas Residents add 5% tax

Apple is a Registered TM of Apple Computers, Inc.

Shipping Address
1104 Summit Ave
Suite No. 106
Plano TX 75074

TG PRODUCTS
P O BOX 2931
RICHARDSON TX 75080

214 424 8568

throttle and check for any construction mistakes, solder shorts, bad connections, and so on. After getting into the Monitor you can check normal operation of the computer by typing E000L and viewing the disassembled listing. If your paddles are not labeled, identify paddle 1 as follows:

While holding down the push button on one paddle, hit the reset key. If the computer beeps as usual, repeat using the other push button. The button which prevents a normal reset is button 1. This button can be labeled "slow/stop." Momentarily press the other button to restore normal operation (this button can be labeled "normal").

Next rotate paddle 1 to its midrange position and momentarily press button 1. Type L and hit return. Note that the listing is now slower and its speed can be varied by the paddle setting. The listing can also be stopped at any time by pressing, and holding, button 1. Press button 0 to restore normal speed operation.

Using the Applethrottle. The Applethrottle can be operated at any time to slow down or stop the computer. When engaged, all computer functions except memory refresh and video generation are affected. Thus the Applethrottle can be used in machine language, Basic, Pascal, and so on. It may be called upon during the list, trace, or run of any program. At slow speeds, the video display produces a rolling effect each time a new line of text is printed. This provides a perfect example of how the Applethrottle can be used to probe the workings of the computer.

When the Apple is listing out a long program, it appears as if it is scanning one long sheet of paper from top to bottom. New text continually appears at the bottom of the screen while the existing lines seem to glide up the screen until they disappear at the top. This function of the video display is referred to as *scrolling* and you've probably come to take it for granted. But if you stop to think about it, this is not such a simple task. Every time a new line is to be displayed at the bottom of the screen, the lowest twenty-three lines must each be moved up one line. Due to the Apple's memory-mapped video design, this amounts to moving 920 characters (23×40) and calculating new base addresses forty-six times. Fortunately, the speedy cpu can perform all this in a fraction of a second so we see a rather smooth moving display.

But when the Applethrottle is turned on, the cpu can be slowed down to the point where this ordinary task of scrolling now takes several seconds or more to complete. It then becomes obvious what actually happens every time a new line is printed. Watching in slow motion, you can see the cpu rewriting each line, one at a time, from the top of the screen down. Since the information on the top line is no longer needed, it is replaced by the characters on the second line. The text from the third line is then copied onto the second line and so on. It is also interesting to note that the copying is done from right to left. This process continues until the bottom line is reached. This line is then changed to all blanks to erase the previous information. Finally the new text is printed at the bottom of the screen.

After watching the cpu perform this operation, you should have a good grasp of what is involved in printing text to the screen. If you have a working knowledge of assembly language, or especially if you're trying to develop such skill, you can now look at the software routine that actually is responsible for what you saw. The *scroll* routine can be found at location $FC70 in the Apple's Monitor. It is actually part of the complete video out routine, VIDOUT, which begins at location $FBFD. Taking this back one more step, we find that VIDOUT is usually called by the Apple's character output routine, COUT at $FDED. Study the Apple Monitor listing to see if you can follow the assembly language routine that scrolls the screen. By understanding the software and being able to follow it through in slow motion, you can begin to gain a better understanding of how the Apple works.

One of the best features of the Applethrottle is that it requires no software and can be activated or deactivated at any time. Therefore its uses are almost limitless. You can use it to slow down or stop Applesoft program listings (no more typing *speed*= over and over again). Of course, it also does the same for Integer Basic and even gives you *stoplist* without the Autostart ROM. By adding a toggle switch in parallel with button 1, you can also have a "freeze" control. This would be great when you're playing a game and get interrupted. Just flip the switch and put the computer on hold while you take care of other things. Some games have such a feature built into their program, but this technique will work for any program.

Speaking of games, have you ever wondered how a particular program produces those spectacular graphics on the hi-res screen? Well, pop on the Applethrottle and watch the figures being drawn in slow motion. The ultimate gaming use for the Applethrottle is to slow down reaction-type games, making them much easier to play. This has accounted for at least one unbelievable score in *Snoggle*! No doubt you will come up with many more uses for the Applethrottle.

Because the Applethrottle slows down the overall operation of the computer, certain time-sensitive operations will not work properly with the throttle engaged. Specifically, cassette and disk I/O should not be performed when the Applethrottle is on. No harm will be done but an error message will (eventually) be printed. Other real-time software routines such as a serial printer driver (including the Apple High-Speed Serial Interface Card, for example) may not function at subnormal speeds. Thus you should return the computer to normal speed before any of these operations are performed.

Another problem can arise when the Applethrottle is set for very slow speed and you are typing at the keyboard. Since the keyboard input routine is slowed down along with everything else, it is very easy to type too fast, causing the computer to miss one or more keystrokes. To avoid this, you may wish to return to normal speed, enter from the keyboard, and then reactivate the Applethrottle before hitting return. Remember, the Applethrottle will always disengage whenever the reset key is pressed or if there are no paddles connected to it.

John Bell Engineering Apple II Display Board. Probing

SUPER RAM II
16K Language Card
With ROM SOCKET
2 Yr. Warranty
$ 125.

COMPUTER PRODUCTS
DESIGNING
MANUFACTURING
ELECTRONIC ENGINEERING

APPLE OWNERS

R. H. Electronics Inc.

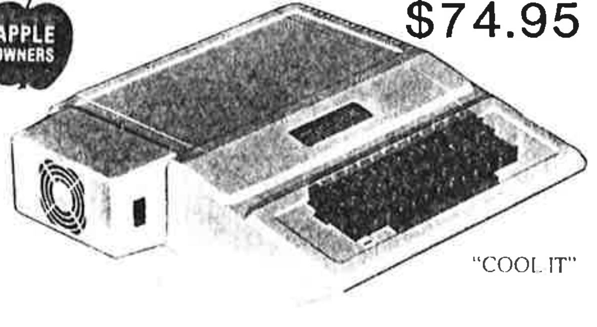
COPYRIGHT © 1981 - PATENTS PENDING

566 Irelan, Bin ST
Buellton, CA 93427
(805) 688-2047

SUPER FAN II ™

FOR YOUR APPLE II COMPUTER*

$74.95



"COOL IT"

- CLIPS ON – NO HOLES OR SCREWS • REPLACEABLE SWITCH
- AVAILABLE IN 120V OR 240V AND 50/60HZ • DURABLE MOTOR
- REDUCES HEAT CAUSED BY EXTRA PLUG-IN CARDS
- INCREASED RELIABILITY – SAVES DOWN TIME AND REPAIR CHARGES
- TAN OR BLACK COLOR
- LOW NOISE DUE TO DRAWING EFFECT OF AIR THROUGH YOUR COMPUTER AND SPECIAL FAN AND MOTOR DESIGN
- "QUIETEST FAN ON THE MARKET"
- EXTRA 120V OUTLET FOR MONITOR OR ACCESSORIES TURNS ON WHEN YOU TURN ON YOUR FAN (NOT AVAILABLE ON 240V MODEL)
- SOLD WORLD WIDE • UNIQUE 1 YEAR WARRANTY

COPYRIGHT © 1981 - PATENTS PENDING

HOW TO HOOK IT UP: 1) Clip it on your APPLE. 2) Unplug your 120V cable (you won't need it). 3) Plug short 120V cable from Super Fan II to the back of your computer. 4) Plug the supply cable from Super Fan II to your 120V power source. 5) Turn on the rocker switch and a built-in, red, ready light comes on. 6) You are all set to "COOL IT." This switch also turns your computer "off" and "on."

Dealer Inquires Invited

*Plus 6% California Sales Tax and $2.50 shipping charge per item

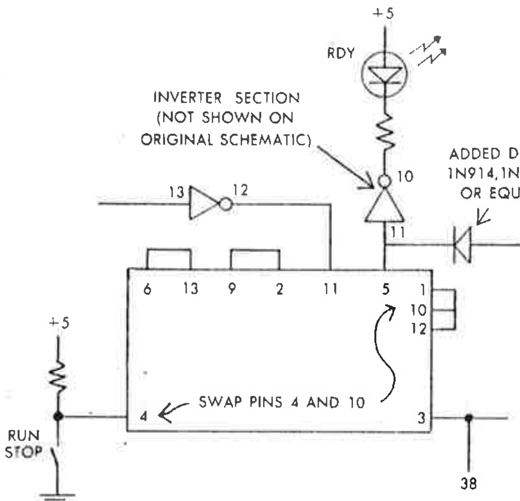

+5
RDY
INVERTER SECTION (NOT SHOWN ON ORIGINAL SCHEMATIC)
ADDED DIODE 1N914, 1N4148, OR EQUIV.)
10
13
12
11
21
6 13 9 2 11 5 1
10
12
+5
SWAP PINS 4 AND 10
4
3
RUN
STOP
38

Figure 3.
Modifications to the John Bell Engineering display board.

even deeper into the internal workings of the Apple can be accomplished with the display board from John Bell Engineering. This board contains sixteen LEDs to monitor the address bus and eight more for the data bus. These give a constant readout on what information is present on their respective buses. The board also has a three-position switch which can start, stop, or single-step the cpu. Under normal operation the LEDs flash on and off so rapidly that they appear to be on all the time. But when slowed down or stopped by the switch on the display board, they begin to reveal the true nature of how data is transferred on the buses.

Before going any further, we must point out that our evaluation board required two modifications to make it work. It seems that the original design of this board contained a couple of major flaws. Nevertheless, it turns out that both problems can be easily solved with a few simple wiring changes and the addition of one diode. The first correction involves the operation of the *run/stop* switch. Originally this switch was allowed to control the 6502 RDY line asynchronously, that is, without regard for the system clock phase as previously described. This, of course, had the disastrous effect that half the time the switch was operated the computer would hang up. Fortunately, by moving the switch signal to a different input of the same IC, we can make it act synchronously. The other problem was that the RDY output from the display board came directly from a TTL logic device. This does not comply with the Apple bus requirements that this signal be driven by an open collector or active low circuit. This means that any given peripheral board can pull the line low when needed, but should otherwise not interfere with the signal. The display board's TTL signal with its active high would prevent other boards from controlling the RDY line. Again the fix is simple—add a diode in series with the control line. These changes are shown in figure 3.

The manufacturer stated that the boards currently being sold have these corrections. For those of you that may have an older board (and probably couldn't figure out why it didn't work) we suggest you perform the following modifications. Besides the ability to solder, all you need is one small signal diode (1N914 variety), a sharp knife, and some small gauge wire such as is used for wire-wrapping.

1. Start by cutting two traces. The first is on the component side of the board and starts at the right side of R28, going downward toward and under IC6. Using an Exacto knife or similar tool, cut this trace cleanly so that there is no longer any connection. The other trace to be cut is on the back side of the board. It is the short one between pins 10 and 12 of IC6.

2. Using some small gauge wire, connect pins 4 and 12 of IC6. Then connect pin 10 to the right side (the left side when

```
FD1B:  E6  4E      KEYIN   INC  RNDL
FD1D:  D0  02              BNE  KEYIN2  INCR RND NUMBER
FD1F:  E6  4F              INC  RNDH
FD21:  2C  00  C0  KEYIN2  BIT  KBD     KEY DOWN?
FD24:  10  F5              BPL  KEYIN   LOOP
```

Figure 4.
Source code for KEYIN routine in Apple's Monitor.
© Apple Computer Inc.

looking from the back) of R28 that was isolated in the previous step.

3. To add the diode, locate the trace that goes to pin 21 of the edge connector. Follow it up to where it meets a hole in the board, flip it over, and note that this trace continues to pin 5 of IC6. Somewhere between the hole and pin 5, cut the trace. Then connect the diode to the same two points; the cathode goes to pin 5, the anode to the hole. That's it!

Using the Display Board. As an example of how to use the display board, we will step through one of the shortest, yet by far the most frequently executed, routines in the Apple Monitor—the keyboard input routine. This routine is located at $FD1B and is labeled KEYIN in the Monitor listings. Whenever the computer is waiting for you to type something in on the keyboard, this routine is being executed over and over again. Aside from getting data from the keyboard, this routine also performs one other task. It is constantly incrementing a two-byte counter in RAM which can be used as a random number generator.

Figure 4 shows the assembly language code for this routine. To follow this routine as it is being executed by the 6502, turn off the Apple and plug in the display board. Set the switch to RUN and then turn on the computer. You do not need to boot DOS, so if your drive starts spinning, just hit *reset* to stop it. Now move the switch on the display board to STOP (or use the Applethrottle). The computer will have stopped somewhere in the KEYIN routine. Activate the single-step switch as many times as necessary until the address bus shows $FD1B. This is the beginning of the routine and the data bus should read $E6, which is the contents of that byte in the Monitor ROM.

As you continue to single step through the program, you should get results similar to what's shown in figure 5. After pressing the switch twelve times (or possibly fourteen), you should be back to address $FD1B. This sequence will continue to loop around unless you press a key on the keyboard. Note that the data in location $4E can be anything but that each time through the loop it goes up by one. Also note that if you stop with address $C000 on the bus, the data LEDs will be arbi-

Address	Data
FD1B	E6
FD1C	4E
004E	Note 1
FD1D	D0
FD1E	02
FD1F	E6
FD21	2C
FD22	00
FD23	C0
C000	Note 2
FD24	10
FD25	F5
FD26	91
FD1B	E6
.	.
.	.
.	.
Repeats	

Figure 5
Sample results of single-stepping through KEYIN routine.

Note 1: The value in location 4E will be random but will increment by one each time through the loop.
Note 2: The data at C000 (keyboard memory-mapped address) will depend upon the last key pressed on the keyboard.

trary. If you now type on the keyboard, the ASCII representation of each key pressed will appear on the LEDs. The highest-order lamp, D7, will also be lit. This bit is used to flag the cpu that a key has been pressed.

If you take a close look at the results in figure 5, you should be able to follow the cpu as it performs its monotonous task. With each step, the 6502 puts out the contents of its program counter onto the address bus. This indicates where the cpu expects to find the next byte of its program. The data at this location is read into the cpu and, assuming it is the first byte of an instruction, is interpreted as the op-code for that instruction. In the case of location $FD1B, this data is $E6 which is the op-code for the INCrement zero page memory instruction.

The actual location to be incremented is found in the second byte of the instruction, which is fetched next. At $FD1C we read in a $4E, which is the desired location as shown in source listing (the $4E shows up in the object code listing in the third column; it is referred to in the source code by the label RNDL). The next address put out by the cpu will be $4E with one plus the old contents of this location appearing on the data bus. This new value is then written into location $4E and the cpu moves on to fetch the next instruction from $FD1D. Here it finds a $D0 followed by a $02, which stands for Branch if Not Equal (to zero) ahead two bytes. There is only a one in two hundred and fifty-six chance that this branch will *not* be taken. Therefore, our sample results show this branch as being taken.

But wait, you say! The next memory reference is to $FD1F. That is the beginning of the next instruction which the cpu has just decided to skip past. It turns out that there is a very good reason for this mysterious action which brings up one of the finer points about the 6502. What you are witnessing is called *pipelining* and it refers to the architecture of a cpu that allows it to sort of think ahead.

In the case of our branch instruction, the 6502 must check its internal flags to determine whether or not to perform the branch. While it is doing this, it is also possible to put out the next sequential address to start reading the next byte of code. Therefore, if the 6502 decides that the branch is not to be taken, it already has started to execute the next instruction—no wasted time here! If on the other hand, the branch is to be taken, then the 6502 must calculate the new address to place in the program counter. During this time, the next sequential byte is again being read. However, this time the byte is ignored since the next instruction must now be fetched from the new location.

Figures 6, 7, and 8 illustrate the possible branching actions on a cpu cycle-by-cycle basis. If you are interested in such detailed analysis of how the 6502 works, consult the 6500 hardware and software manuals such as put out by Synertek. They are highly technical but they reveal these subtle points about the 6502's operation. In case you think that this is all academic, there are many times when you may have to figure out exactly how many machine cycles a particular routine will take. Moving a program, even one that is completely relocatable, can have surprising results if relative branches now start crossing page boundaries for example. Even knowing that a branch instruction executes faster if the branch is not taken can be useful in writing speed intensive routines.

Teaming the Applethrottle and Display Board. The display board currently being sold still has some trouble in the single-stepping mode. One way to get around these problems is to use the Applethrottle to slowly step the cpu. This adds the convenience of variable speed and remote operation through the paddle controls. Unfortunately, even at its slowest setting, the Applethrottle allows the computer to execute many instructions per second. The LEDs on the display board still flicker too fast to watch. Adding a 22 MFD capacitor across C3 on the Applethrottle extends its range down to about one instruction per second. Note that this is roughly three hundred thousand times slower than normal. At this rate, the scrolling operation previously described would take nearly four hours to complete.

By combining the Applethrottle and the display board, there are many other secrets that can be discovered. For example, if you see something really interesting in a program that you have, it might be nice to examine the code that performs it. If the program is in machine language, you can use the Apple's built-in disassembler to view it. But how do you know where to look? You might have some clue as to what the code looks like (for example, if it involves the speaker, keyboard, screen, DOS, and so on, you could look for references to their corresponding addresses), but you would still have to run through the entire listing. This might take several passes and lots of time, and you still might not locate it. There is a better way however. Just plug in the Applethrottle and display board and run the program. When the routine of interest starts exe-

Cycle	Address Bus	Data Bus	External Operation	Internal Operation
1	0100	OP CODE	Fetch OP CODE	Finish previous operation, increment program counter to 101
2	0101	Offset	Fetch offset	Interpret instruction, increment program counter to 102
3	0102	Next OP CODE	Fetch next OP CODE	Check flags, increment program counter to 0103

Figure 6.
Illustration of relative addressing branch not taken

Taken from Synertek's *SY6500/MCS 6500 Microcomputer Programming Manual*, August, 1976. Reproduced by permission

Cycle	Address Bus	Data Bus	External Operation	Internal Operation
1	0100	OP CODE	Fetch OP CODE	Finish previous operation, increment program counter to 101
2	0101	+50	Fetch Offset	Interpret instruction, increment program counter to 102
3	0102	Next OP CODE	Fetch next OP CODE	Check flags, add relative to PCL, increment program counter to 103
4	0152	Next OP CODE	Fetch next OP CODE	Transfer results to PCL, increment program counter to 153

Figure 7.
Illustration of relative addressing branch positive taken, no crossing of page boundaries

Taken from Synertek's *SY6500/MCS 6500 Microcomputer Programming Manual*, August, 1976. Reproduced by permission.

Cycle	Address Bus	Data Bus	External Operations	Internal Operations
1	0100	OP CODE	Fetch OP CODE	Finish previous instruction
2	0101	−50	Fetch offset	Interpret instruction
3	0102	Next OP CODE	Fetch next OP CODE	Check flags add relative to PCL
4	01B2	Discarded Data	Fetch discarded data	Store adder in PCL and subtract 1 from PCH
5	00B2	Next OP CODE	Fetch next OP CODE	Put out new PCH and increment PC to 00B3

Figure 8.
Illustration of relative addressing—branch negative taken, crossing of page boundary

Taken from Synertek's *SY6500/MCS 6500 Microcomputer Programming Manual*, August, 1976. Reproduced by permission.

cuting, just stop the computer or slow-step it. One glance at the display board will tell you what address in memory is being executed. This should be close enough for you to find the beginning of the routine later with the disassembler.

The Apple II Display Board is available from John Bell Engineering for $49.95, assembled. As a kit, the cost is $42.95. The bare board alone can be purchased for $25.95.

Apple II Extender Board. Another handy tool for working with Apple hardware is an extender board. This is simply a small board that brings out, or extends, the peripheral connector inside the computer. This makes it possible to connect test equipment to various parts of a peripheral board without interference from other boards in adjacent slots. It can also be used to raise up the display board for easier viewing.

Anyone who works on complex, modular electronic systems knows the value of an extender board. The Apple II is no exception. Of course, an extender board is only used when experimenting with or repairing a peripheral card. The extender board pictured sells for $12.95 and is available from John Bell Engineering.

Mill Update. Last November we reported on the Pascal Speed-Up Kit from Stellation Two. This product, in conjunction with the Mill 6809 processor board, offered significant speed improvement for Pascal programs. However, we found that one of our benchmark test programs actually ran slower after installing the speed-up patches. This led to the discovery that the mathematical functions were still being performed by the Apple's 6502 instead of the 6809.

Since that time, we have received an additional package which is referred to as the Floating Point Update for the Speed-Up Kit. This transfers almost all of the computing power over to the 6809 leaving the 6502 to handle the input/output functions. Now the full power of the coprocessor can be utilized. Unfortunately, the two processors still alternate control—they do not operate simultaneously.

To test the speed improvement, we ran the same benchmark which calculated the squares of 10,000 numbers. The results from the previous test revealed that this program would take 47.8 seconds to complete with standard Apple Pascal. Adding the speed-up kit increased the time to 54.0 seconds. After installing the floating point patches, however, this program ran in 27.8 seconds. That's over 40 percent faster, which is quite impressive!

As an interesting aside, we wrote a program in Applesoft to accomplish the same task. In Applesoft this calculation takes almost forty-four seconds. While one of the Applesoft compilers could be used to speed this up, they would not offer much improvement to this program since it mostly involves mathematical calculations. All of the Applesoft compilers rely on the same 6502 math routines in ROM that are used by the interpreter. Thus the compiled program performs number-crunching at almost the same speed as the noncompiled program.

Installing the Floating Point Update is quite easy. All the necessary changes are made by executing one program on the REAL: diskette. Like the rest of the Speed-Up Kit, the improved capabilities are immediately available. There is no need to change any existing source or code files to make use of the Speed-Up. Another sample program is also included on the disk. This program makes extensive use of the floating point routines and shows a 51 percent improvement over the unmodified Pascal. Speed increases of up to 80 percent have been reported with this package. The Floating Point Upgrade sells for $35.

Also just announced for the Mill is the OS-9 operating system with Basic-09. This should greatly enhance the power of the Mill while providing a superior operating system for the Apple II. Watch this column for a complete review of this system in the near future. ⊐■

John Bell Engineering, Box 338, Redwood City, CA 94064; (415) 367-1137. Stellation Two, Box 2342, Santa Barbara, CA 93120; (805) 966-1140. Synertek, 3050 Coronado Drive, Santa Clara, CA 95051; (408) 988-5600. West Side Electronics Inc., Box 636, Chatsworth, CA 91311; (213) 884-4794.

Meat and Potatoes.

Hungry Apples cannot survive on chips alone. All the RAM chips in Silicon Gulch will not make your computer any more useful if it doesn't have an informed owner.

The best way to learn about your Apple is to read Softalk magazine. No other publication offers more timely tutorials and more interesting features about the Apple computer than Softalk.

Good cooking is hard to come by. So take advantage of our trial subscription offer to ensure Softalk agrees with your Apple's digestive system.

If you've never subscribed to Softalk and you own an Apple, you are eligible for a free trial subscription. Send in your name, address, and Apple serial number and we'll start sending you Softalk free of charge.

If you've been receiving Softalk regularly, you have already gotten or will get soon a notice for paid subscription.

Consider wisely before making your choice. Don't put your Apple on a diet of chips. Give it some meat and potatoes.

In addition, if you have a friend whose Apple is sick from misinformation send in the name, address, and serial number of your friend and we'll send you a free back issue of your choice. The issues available are: September 1980, April through August 1981, December 1981, and January through May 1982. All other issues are sold out.

If you're one of the lucky ones with a complete collection of Softalk, you can get a one month extension on your paid or trial subscription. A free month of Softalk or a free back issue for every new name you send. Such a deal!

Subscribe to Softalk!

Softalk Circulation
11021 Magnolia Boulevard
North Hollywood, California 91601