Dr. Marvin L. De Jong
Dept. of Math-Physics
School of the Ozarks
Pt. Lookout MO 65726

Catching Bugs with Lights

This hardware approach to software debugging could save you many headaches.

In debugging a program, how often have you wished you could *see* the contents of the accumulator or the status register at each step *without* pushing all those buttons? If you are interested in a simple hardware solution to this problem, read on.

Although my circuit was designed for the KIM-1, the idea certainly is applicable to other systems. Even if you're not interested in my Bug-Light circuit for programming purposes, it gives you one or more output ports in page zero of memory, and it makes a useful tool for teaching programming.

Introduction

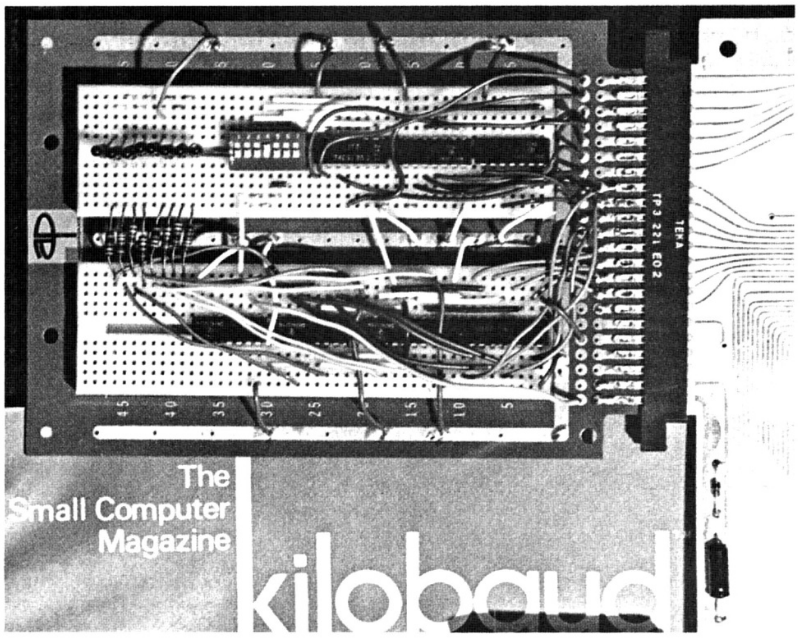
The KIM-1 monitor and a little hardware provide you with a single-step mode in which the program may be executed one instruction at a time. After each instruction is executed, the resident monitor program stores the contents of the accumulator, the status register, X-register, Y-register and other registers (see Table 1 for the locations of each register). The important registers are also saved in zero page when a break (BRK) command is placed in a program and the IRQ vector is 1C00. Both the single-step (SST) mode and the break-to-KIM monitor are used extensively in debugging programs.

Use of the SST mode is explained in the KIM-1 User Manual, while the break-to-KIM-monitor technique is explained in *The First Book of KIM*. With either technique the contents of the various registers may be read by using the keyboard to look up the locations in zero page where their contents are stored. For example, to see what the contents of the accumulator are after an instruction, simply address location 00F3 with the keyboard to display it on the seven-segment display.

It's a great feature, but it's slow. At least six consecutive key depressions must take place to examine a register, restore the program counter and execute the next instruction in the program. If you're following your program around some crazy loop to see why it never comes out, this procedure can take a lot of time. Perhaps my arthritic fingers and bouncy keys are the problem. There has to be a faster approach to the register display problem. A reasonable objective, I decided, was an LED display of each bit in a particular register, with no extra key depressions.

Enter Bug-Lights

To accomplish this objective I designed a circuit to decode the addresses of the locations where the various register contents were stored and allow the microprocessor to WRITE the same data to output ports with LEDs to represent each bit. Thus, when the monitor stores the contents of the status register at location 00F1, it also writes the same data to an output port whose address is 00F1. In this case the LEDs indicate the state of the various flags. If



What's in front of the Kilobaud? *The Bug-Light circuit.* *(Photo by Paul Campbell, S of O student)*

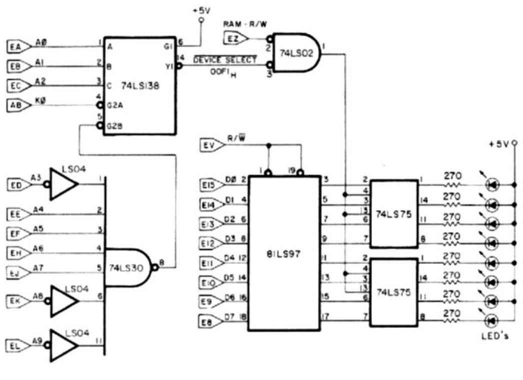


Fig. 1. The basic Bug-Light circuit.

Address	Label	Contents
00EF	PCL	Program Counter Low
00F0	PCH	Program Counter High
00F1	P	Status Register (Flags)
00F2	SP	Stack Pointer
00F3	A	Accumulator
00F4	Y	Y-Register
00F5	X	X-Register
00F6	CHKHI	Cassette Checksum High
00F7	CHKSUM	Cassette Checksum Low

Table 1. Zero Page Memory locations of the various registers.

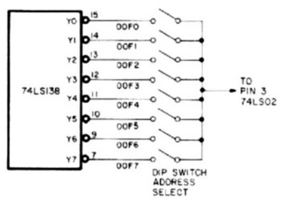


Fig. 2. Use of a DIP switch to select the register to be displayed.

the output port has address 00F3, then the LEDs will show the contents of the accumulator, in binary, of course.

Bug-Lights comes in three versions. The basic circuit is shown in Fig. 1. It will display one register only. A modification that increases the utility of the basic circuit is shown in Fig. 2. The DIP switch allows you to select which register you want to follow as you step through your program. If you really like blinking lights and/or do a lot of programming, see the chrome-plated modification to display up to eight registers simultaneously as outlined in Fig. 3.

Of course, the most important registers to display are the accumulator, the status register, the X and Y registers and perhaps the stack pointer. These displays would make an impressive yet functional front panel. My personal version has the DIP switch modification shown in Fig. 2. (The program counter low, PCL, is stored at address 00EF and cannot be observed with the Bug-Light circuit. I cannot recall ever using this register to debug a program.)

How Bug-Lights Works

We will begin with the address decoding circuitry. The 74LS138 decoder/demultiplexer will decode the lowest three address lines (A0, A1, A2) when G1 is at logic 1 and G2A and G2B are at logic 0. G1 is tied high, eliminating any further consideration of it.

In order to have both G2A and B at logic 0, the $\overline{\text{K0}}$ select from the KIM-1 and the output of the 74LS30 must be at logic 0. K0 will be low when address lines A10-15 are low. This is handled by the KIM-1 circuitry. You can see from Fig. 1 that the output of the 74LS30 is low when A4-A7 are at logic 1 and A3, A8 and A9 are at logic 0. The compilation of this information as the requirements to select the 74LS138 are shown in Example 1.

The 74LS138 decodes the lowest three address lines to produce active low *device select pulses* whenever addresses 00F0-00F7 are on the address lines. Each of the eight outputs of the 74LS138 corresponds to one of the eight addresses 00F0-00F7, which in turn include the address of the locations where the various registers are stored.

The device select pulse from the 74LS138 is inverted and ANDed with the inverted RAM-R/$\overline{\text{W}}$ signal from the KIM-1. This produces a positive pulse from the 74LS02, which occurs only on a WRITE cycle and when the correct address is placed on the address bus. For example, an STA 00F1 instruction will produce such a pulse in the circuit of Fig. 1. This pulse is applied to the gate inputs of the 74LS75 Bistable Latches.

As long as the positive pulse is applied to the 75LS75 gates, the Q outputs follow the D inputs, and the $\overline{\text{Q}}$ outputs are the D inputs inverted. At the trailing edge of the positive pulse, which occurs when the Ø2 clock signal on the KIM-1 changes from logic 1 to logic 0, the data at the D inputs is latched into the Q outputs. So when a WRITE occurs to 00F1, the data will appear at the Q outputs and it will be stored there, at least until another WRITE to 00F1 occurs.

The 81LS97 is a data bus buffer. It is activated only on a WRITE command when the R/$\overline{\text{W}}$ is low. If only one output port is desired and the data bus lines are kept short, then the 81LS97 may be omitted since the 6502 microprocessor can drive the 74LS75s directly. However, if you want to locate your lights on a front panel, or if you want to add sets of eight lights for several registers, then the bus driver becomes essential.

The LEDs are connected through current-limiting resistors to the $\overline{\text{Q}}$ outputs of the 74LS75s. They will glow when $\overline{\text{Q}}$ is low and Q is high. Thus a glowing LED corresponds to a logic 1 for the bit it represents while an LED in the off state corresponds to a logic 0.

Bug-Lights, Output Ports or Both

An added feature of the Bug-Light circuit is its ability to be used as an output port as well as a debugging tool. The Q outputs of the 74LS75s are not used for display purposes, and they contain the data that was written to them. Thus, they can be used as zero-page, memory-

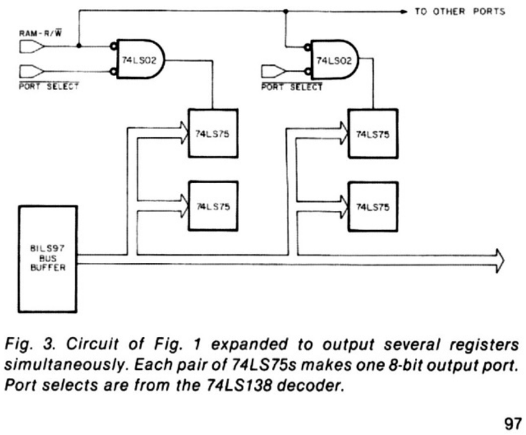


Fig. 3. Circuit of Fig. 1 expanded to output several registers simultaneously. Each pair of 74LS75s makes one 8-bit output port. Port selects are from the 74LS138 decoder.

```
A15 A14 A13 A12    A11 A10  A9  A8    A7  A6  A5  A4    A3  A2  A1  AØ—Address Line
 Ø   Ø   Ø   Ø      Ø   Ø   Ø   Ø     1   1   1   1     Ø   X   X   X—Logic Value
         Ø                  Ø                 F                 Ø-7    —Hex Number
```

Example 1.

```
BEGIN    LDA $Ø1
THERE    ASL A
         JMP THERE.
```

Example 2.

Integrated Circuit	+5 V	Ground
74LS138	16	8
74LS30	14	7
74LS02	14	7
74LS04	14	7
74LS75	5	12
81LS97	20	10

Table 2. Power connections for the Bug-Light integrated circuits.

mapped output ports.

An application program can make use of these ports to write a 7-bit ASCII word to some external device such as a video card, an IBM Selectric or some other device. A/D or D/A converters can be driven from these ports as easily as the PAD and PBD ports on the KIM-1 application connector. The only time the memory locations 00F0-F8 are used by the computer is in an NMI or IRQ jump to the monitor, that is, in debugging. So you can have your Bug-Lights and output ports as well.

Construction

Table 2 shows the power connections for each of the chips in the logic diagram. All the other connections are given in the figures. My version was built on a UNICARD I, containing two breadboard strips and an edge connector pad that matches the KIM-1 expansion pad. I soldered an edge connector to the UNICARD so I could plug the KIM-1 expansion pad into it. All the connections of the Bug-Light circuit except one are to the expansion pad on the KIM-1. All the connections are found on the pad symbols in Fig. 1. The $\overline{K0}$ select comes from the application pad on the KIM-1. Its pin number, AB, is also given.

Layout is not critical, and other approaches than the one I used will work. A wire-wrap approach might be more permanent and less expensive, although I have found that the circuits on the breadboards last indefinitely. The accompanying photograph shows my version. Power was stolen from the KIM-1 power supply, since both +5 V and ground are available at the expansion pad.

When you get your circuit built, say a one-port version, select the location you want to view with the DIP switch or by the appropriate connection. With the KIM-1 running in the monitor, address the location and store FF in it using the keypad on the KIM-1. All the LEDs should light. Change the contents of the port until you are sure that each LED is responding to the correct bit value. Stepping through the sequence 00, 01, 02, 04, 08, 10, 20, 40, 80 of data values will test each light in turn.

Next, load any program, set the KIM-1 up for the SST mode and step through the program. The lights should reflect the current contents of the register you have selected to view. I had no trouble. For once my design worked the very first time I tried it. I hope you have the same kind of success. If you don't, recheck all your wiring, check the polarity on your LEDs, make sure they all work and finally make sure you haven't made a mistake on numbering the pins on the ICs.

If some bits work and some don't, then exchange signal paths for the two bits. For example, if one bit is working, then the 74LS75 latch for this bit will also be working. Use the same latch for a nonworking bit to see if the problem is in the latch. The circuit is simple enough so that it should not take too long to figure out any problems.

Beginning programmers have a lot more trouble visualizing what is happening as a result of a certain instruction than veteran programmers imagine. One application of Bug-Lights is to illustrate the results of various instructions. For example, set up Bug-Lights to show the contents of the accumulator (00F3). Then write a short program (shown in Example 2) in which the accumulator is loaded with Ø1 followed by an ASL A in an infinite loop.

Now single-step through the program and watch the 1 move from right to left on the LEDs. Replace the ASL A with a ROL A and note the difference. Other instructions can be illustrated in the same way, giving students who have difficulty visualizing zeros and ones among bits and bytes an excellent visual aid. ■