

CRIME STOPPER TIPS by MIKE SKALA

NOTE: THIS INFORMATION IS NOT BEING DISTRIBUTED TO ANYONE WHO IS NOT MARKETING SOFTWARE!
PLEASE KEEP CONFIDENTIAL!!!

The following information will enable the programmer to protect, as best possible, any software written in ASTRO-BASIC from being copied, disassembled, etc. There seems to be no completely fool-proof means of protection, but there are a few tricks that will protect us from the vast majority of "crooks".

First, let's look at the key to this whole concept; the HOOK VECTORS (as listed on pg 103 of the BASIC manual). BASIC is just a program, written in machine code, and stored in ROM where we can't go and change anything. Jay Fenton did leave us a little opening though with these HOOK VECTORS. These are a few bytes in RAM that are initialized by the BASIC cartridge upon reset. What happens, for example, your basic program is merrily running along when suddenly a screen interrupt is encountered. The software stops all other processing and calls out SCREEN INTERRUPT HOOK VECTOR to take care of that interrupt. What it finds there is a 3 byte instruction; JUMP TO %(8701). There it will take care of NT, BC, FC, etc, and then return. Let's say we change that to JUMP TO %(20258), by making $%(20218) = 20258$. Now at $%(20258)$ we put a little machine code to disable part of the keyboard, call $%(8701)$ and then return. This will occur 30 times every second! This is exactly what Brett did in his CRITTER program, and called it BACKGROUND/FOREGROUND processing. Here, our BACKGROUND program is disabling the keypad.

There is also the INPUT CHARACTER HOOK VECTOR, which branches the mainline program whenever there is an input from the keypad. I haven't found anything to put here to cause anything other than a crash. Make $%(20120) = 199$ and depressing any key is like pushing the reset button. Or set $%(20121)$ to any good "CRASH CALL" (a location you call to get a program bomb with good sound effects or colors etc. Try 6683 or 6490). This procedure is fine if you don't want to let the user touch the keypad at all.

Some games may require a bit of keypad input from the player, so let's learn how to selectively disable particular keys. First look a FIG 1 to see how our machine code will read our keypad as input ports, and what values are returned. Then look at my two examples.

(23) (22) (21) (20)

| | | | |
|----|-------|------|------|
| GO | PAUSE | HALT | LIST |
| 1 | 1 | 1 | 1 |
| 2 | 2 | 2 | 2 |
| 4 | 4 | 4 | 4 |
| 8 | 8 | 8 | 8 |
| 16 | 16 | 16 | 16 |
| 32 | 32 | 32 | 32 |

FIG. 1

* EXAMPLE # 1 *

```

% (2025B) = 245  PUSH AF      SAVE REGISTER
                219  IN A, (N)  READ PORT B(21)
                21    N
                254  CP n      SEE IF = 1
                1     n
                40   JRZ, c    IF SO JUMP TO % (20274)
                9     c
                254  CP n      SEE IF = 32
                32   n
                40   JRZ, c    IF SO JUMP TO % (20274)
                5     c
                241  POP AF     RESTORE REGISTER
                205  CALL nn    CALL NORMAL INT.
                253  n          ROUTINE
                33   n
                201  RET        GO BACK TO BASIC
% (20274) = 199  RST 0        RESET
    
```

SET % (2011B) = 2025B

This new INT. routine

will cause a RESET if HALT/RUN or BLUE SHIFT KEY is depressed. All other keys work as normal. The problem here is user could use the EDIT feature to look at program. They still couldn't change HOOK VECTOR as they would need to use BLUE SHIFT KEY.

```

% (2025B) = 245  PUSH AF
                219  IN A, (N)
                23    N
                205  CALL nn
                64    n  % (20288)
                79    n
                219  IN A, (N)
                22    N
                205  CALL nn
                64    n
                79    n
                219  IN A, (N)
                21    N
                205  CALL nn
                64    n
                79    n
                219  IN A, (N)
                20    N
    
```

SET % (2011B) = 2025B

* EXAMPLE # 2 *

```

% (20276) = 254  CP n
                32   n
                40   JRZ, c
                13   c
                205  CALL nn
                64   n
                79   n
                241  POP AF
                205  CALL nn
                253  n
                33   n
                201  RET
% (20288) = 254  CP n
                1    n
                40   JRZ, c
                1    c
                201  RET
% (20293) = 100  RST 0
    
```

This example will check and crash if GO, PAUSE HALT/RUN, LIST or the WORDS SHIFT KEY is depressed. All other keys will function normally.

NOTICE THAT % (20288) IS USED LIKE A SUB-ROUTINE, TO SEE IF A VALUE OF "1" IS FOUND IN ANY PORT.

% (20293) is where we go to crash. PUT any crash call you like here

% (20275) = 20 N

% (20293) = 100

DUMPING AND LOADING INSTRUCTIONS

Did you know `:PRINT` is the same command as `:PRINT %(16384), 1864` and of course `:INPUT` is the same as `:INPUT %(16384)`.

So, what we must do is dump (to tape) all the normal stuff, i.e. the screen, text & variables and now we must extend the `:PRINT` to dump the HOOK VECTORS, and if we have machine code in the buffer or stack, dump that too.

If you are just altering the HOOK VECTOR and have no additional machine code, dump to tape with:

```
%(20120)=199;:PRINT %(16384), 1885
```

Tape is then loaded with normal `:INPUT;RUN`

If you are dumping machine code as well, first determine how far you need go with this formula: $((\text{LAST BYTE OF MACHINE CODE}) - 16384) \div 2 + RM$. For example, I have 20 bytes of machine code at location 20258 to 20278. Therefore, we have $((20278 - 16384) \div 2) + RM = 1947$. So our dump is made with `:PRINT %(16384), 1947`. But now we have a problem upon input, because we are going to wipe out the user's line input buffer. When you use the direct command `:INPUT;RUN`, our machine goes to loc. 20154 (line input buffer) and begins executing. After executing the `:INPUT`, the machine has remembered where it was (loc 20156), but that `;RUN` is no longer there. What is now there is whatever was in our line input buffer when we dumped to tape. To ensure an auto-run upon load, use the following command:

```
PRINT ";RUN"; %(A)=B;:PRINT %(16384), C
```

A = address of HOOK VECTOR B = new instruction or location C = # of bytes

tape is now loaded & automatically run with simply `:INPUT`

The user can override the auto-run by putting a couple spaces before the `:INPUT` and therefore re-enter the new buffer past the `;RUN`. The keyboard will still be disabled. If you are selectively disabling certain keys, be sure the user can't type in `%(20118)=X` and correct the HOOK VECTOR. Just taking away the PRINT/= key and the LIST key is usually sufficient.

I HOPE THIS HELPS KEEP THE PRICE OF SOFTWARE DOWN!