MOUSE ROUTINE IN AN INTERRUPT

By The Missing Link

October 16, 2007

Abstract

OK, here is the text that accompanied the mouse routine I posted on alt.binaries.atari. Unfortunately the original file was self extracting. So I had to print it out and type it back in. I hope somebody will read the stuff below and use it in his or her own programm. The text and code where written for the Atari 8 bit, but the idea should work on other computers too.

number means that there is a small explanation for those not living in the Atari 8 bit world at the bottom of the text.

Mathy van Nisselroy*1*

In the last years the mouse became very popular, even on Atari 8-bit machines. I've seen a lot of routines, but none of them were working very good. They took almost all processor time in the main loop or they were very slow, etc. I talked to your editor *2* very often about this subject and we were both working on a better routine. I decided to put my mouse routine into a Pokey *3* Timer Interrupt....

A few month ago while I was shopping, I got the great idea. Finally I knew how to do it. I went home, started MAC65 *4*, typed some lines in 10 minutes, and the routine worked great!

The first routine worked in a DLI *5* and VBI *6*. So I knew that it was possible to put a mouse routine in such an interrupt. I could even load from disk or play some samples while I was playing with the mouse. A few days later Freddy *2* visited me and he was amazed. I showed him my routine and he was surprised because of the length and the frequency. The routine was short, very fast and worked very good. You should have seen his face when I was playing a mouse, while the computer loaded the directory into the machine....

Later I told to some other programmers about this routine and now we can see a lot of mouse routines which work in an interrupt. Most of these routines are still slow or you get a 'flying' cursor. If you move the mouse too fast, the cursor will move in reversed direction or will even NOT move. After I had seen these 'f*king' routines I decided to publish my routine on Mega Magazine *7* with some documentation, so everybody can use a better mouse routine. I've called it the 'TML Mouse Routine'. The routine is public domain, but it would be nice if you let me know (or MegaZine *7*) if you use the routine in one of your products. It would be nicer if you mention the creator in your programs....

How the routine works Depending on the mouse (ST or AMIGA) you will get 4 numbers if you move the mouse horizontal and 4 other numbers for the vertical direction. For example: Move the mouse from left to right and you will get the numbers (STICK(0 or 1) *8*) 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4 etc. (these aren't the right numbers, but it is only an example). If you read stick(0) twice, you have two numbers and will know in what direction the mouse moved... If you get a 1 and a 2, the mouse was moved to the right and if you get a 2 and a 1, the mouse was moved to the left. You can do the same thing for the vertical direction.

Now you can create a very big routine with compares, etc., but that will take a lot of time and you have to check the stick very often (Freddy did such a routine in the main loop and there was no time left for any other action), I have made a interrupt routine which checked the mouse 800 times in a second. This routine worked like all routines, get a new value from the mouse, compare it to the old value and do some action. This routine was also large and didn't work very good. The trick is the use of an 'action table'. Get the old value shift it to the left two times and add the new value. Now you will have a unique value between 0 and 15. Use this value as index for the table.

An example:

Old value = 2 (shifted 2x makes 8)

New value = 1 (add to 8)

Index value = 9

Earlier I wrote that a 2 and a 1 was a movement to the left, so we have to decrease the X-pointer. Position 9 will be a 255 in the index tabel.

Old value = 1 (shifted 2x makes 4)

New value = 2 (add to 4)

Index value = 6

1 and 2 is a movement to the right, we have to increase the X-pointer. We have to put on the sixth position in the index table a 1.

Conclusion: Now we have only to shift the old value 2 times, add the new value, use it as index, and add the value we've got to the X-position. We can do the same for the Y-position. This is a very fast routine which takes just some time. But still, this is not the final one....

Sometimes you will get the same value. (old = 1 and new = 1) This means that there was no mouse action or a very very fast action in unknown direction. Do nothing, so put 0 in your index table on position (1+4=5).

The same for 1 and 3. Of course, there must have been some action, but you can't know what kind of action. It is two steps if the mouse is moved to the right, but also two steps if the mouse was moved to the left. So no action on (3+4=7) in your index table.

The last part was the most important part of this textfile. If you don't know what kind of action there was, even if you are sure that there was some action, do nothing! If you do anything you will get a reversed mouse or a 'flying' one.

On the backside of this disk you will find a pre-screen of my new texteditor. You can play with it, open some windows, etc., but that's all. Just an example. You can also find the mouse-source on this disk. It's in MAC65 *4* format.

This one is public domain. Use it have a lot of fun, as long as you remember that you took it from this great magazine *7* and remember who wrote the routine. See you next time!

1 John Maris, founder of A.N.G.

2 Freddy Offenga, editor of *7*

3 Sound chip of the Atari 8bit computer

4 MAC65: Assembler software

5 DLI: small piece of software that tells the graphics co-processor (ANTIC) in the Atari 8 bit what to do

6 VBI: Vertical Blank Interrupt.

7 Mega Magazine or just MegaZine: A disk magazine for the Atari 8 bit computer. Seven issues appeared, then they stopped.

8 STICK (0) or STICK (1): an Atari BASIC command used to read joystick ports pins 1 through 4. For those who wanna look at the code, the PIA, which controls the joysticks in the Atari, is placed at $D300-$D303. Should the source use a different address between $D300 and $D3FF, just substract 4 off this number untill you get in the above mentioned range. BTW apart from the occasional SPACE and capital, I changed nothing about this text.