

Tiger and Mouse

Gunther Zielosko

1. The computer mouse – a modern pointer

Cats (tigers) and mice *can* get along very well under certain conditions, as we will learn from the following application. When Microsoft invented the mouse, nobody could anticipate that it would become such an integral part of the computer just a few years later. Well, do you know every “hot key” for operating Windows programs? Now that we have got used to this little pet it stands to reason that we will use it with a BASIC-Tiger®. We won’t be able to copy Windows with a BASIC-Tiger®; however, we will come close using Tiger, mouse and a graphic display if necessary. Some mouse functions can also be interesting for standard BASIC-Tiger® applications – we will try all this.

Important note!

This application note deals with controlling a serial Microsoft mouse version 2.0a. Other mice (possibly also so-called “compatible” mice) are not tested in this application note. It is possible that such mice do not work because of data format, communication parameters, Tiger-Basic driver or else.

So how does a mouse work? Let’s start with the PC mouse. There are two or more buttons which are commonly used for adapting certain values or functions. These functions are usually determined by the position of the mouse pointer on the desktop and by the program. The position of the mouse pointer is controlled by the mouse’s movement. The mouse has a gummed (for better rolling) ball, which transfers the directions of movement (x and y) onto two rollers (figures 1 and 2).

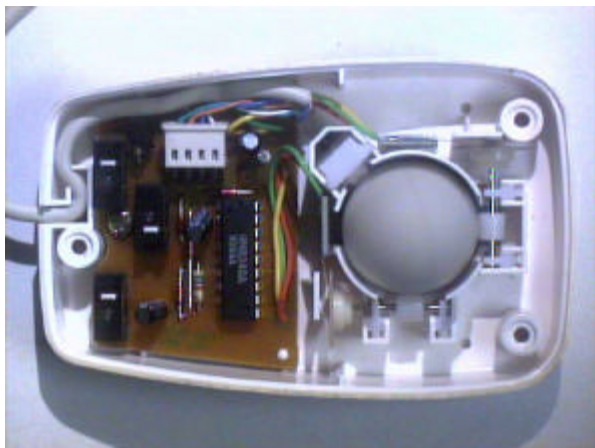


Fig. 1 Interior of a mouse



Fig. 2 Scanning with segmented disc

Depending on the construction there are perforated disks (for optical scanning) or discs with metallic segments (sliding contact scanning, see figure 2). When the mouse is moving the number of passing holes (or contact segments) is counted. The mouse is able to detect whether it moves forwards or backwards by using an ingenious trick. The whole thing is about measuring distances. The number of “steps”, their directions and the state of the buttons are serially outputted in certain intervals. This used to take place almost exclusively via the serial interface of the PC (since the BASIC-Tiger® has a serial interface this kind of mouse suits our purposes). Now there are also mice for PS/2 and USB ports, but these versions are not suitable for our experiments. If you purchased such a new mouse, you can surely do without the old serial mouse and therefore use it for BASIC-Tiger® applications (after thorough cleaning if necessary).

In order to understand the interaction of mouse and PC, we have to take a closer look at the data transmission protocol. But before that two hardware problems have to be solved in order to ensure that a bit is able to get from the mouse to the BASIC-Tiger® in the first place.

2. A question of levels

Up to now we have acted on the assumption that there are two interfaces at the BASIC-Tiger®. But this is not always the case, since Wilke Technology GmbH provides Tigers with real RS232 interface level and Tigers with TTL level. Timing and logic are the same; however, the levels are basically different. This is irrelevant in the case of the Plug-and-Play-Lab, the hardware of which ensures that both versions of the BASIC-Tiger® behave exactly the same. So the following considerations are not important for those of you who plan to experiment with the Plug-and-Play-Lab, but they are important for those of you who plan to set up their own hardware. Figures 3 and 4 show the levels of both BASIC-Tiger® versions:

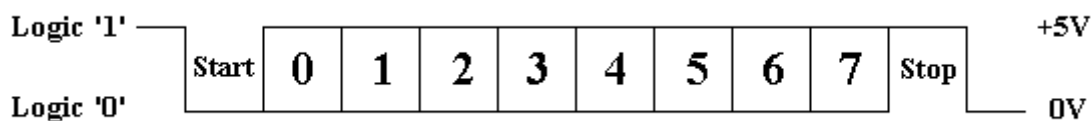


Fig. 3 TTL levels at modules without real RS232 interfaces

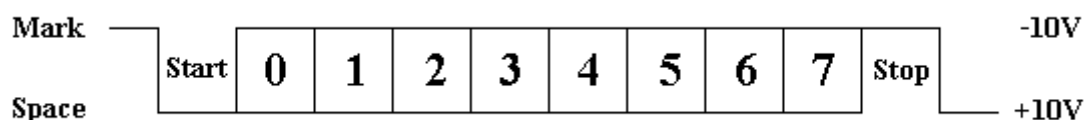


Fig. 4 RS232 levels at modules with real RS232 interfaces

Which one have you got? If you “only” have the TTL module, you do not have to give up – there is an alternative: MAXIM offers an IC, which converts one level to another in a simple way, even the higher +10 V and, above all, the negative -10 V are generated by the IC. You only need some electrolytic capacitors and the RS232 interface is finished (besides, this is

exactly how Wilke Technology does it in their modules and in the Plug-and-Play-Lab!). By the way, this IC, the MAX232, can be used for all cases, which require a real RS232 interface, so the whole effort has not to be taken only for the mouse. There is, however, a downer, because the MAX232 also negates the level logically. So you have to provide an inverter on every TTL side (where applicable this can be done by the software...). It should work with this circuitry, documentation of the circuitry was possibly already delivered with the Plug-and-Play-Lab. Just leave out everything there which is used for operating the LEDs. The MAX232 circuit is shown in figure 5. As you can see, the IC is even able to handle two simple serial interfaces.

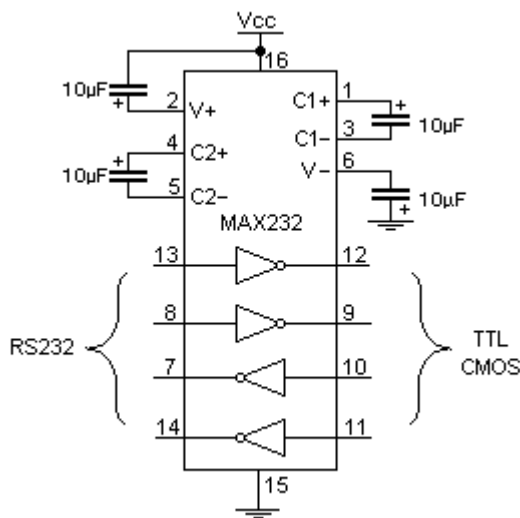


Fig. 5 MAX 232 circuit

3. Mouse power supply

If you take a look at the pin assignment of interface Ser0 at the Plug-and-Play-Lab you will notice that the following lines are connected: RxD0, TxD0, CTS0, RTS0 and GND. These connections are mere signal lines; there is no battery in the mouse either. So where does the mouse take its “energy” from? At the PC (this is where the mouse actually belongs), there are some more lines at the serial interface, but VCC or any other power line can’t be found here either. The solution lies in the fact that the single signal lines’ levels are “abused” as the mouse’s voltage supply via rectifiers. This is not so easy with the Plug-and-Play-Lab, since the voltages provided by the MAX232 are also tapped by other components (LEDs etc.). So we possibly need a small adapter for the serial interface at the Plug-and-Play-Lab, in order to additionally feed the mouse with +5 V from a Plug-and-Play-Lab pin contact. Besides this is only minor effort, because you will always need an adapter, since both mouse and Plug-and-Play-Lab have 9-pin SUB-D sockets. So we connect two 9-pin SUB-D plugs with short wires, as shown in figure 6. Please consider that pins 2 and 3 have to be “crossed” in this case (each pin 2 to pin 3!).

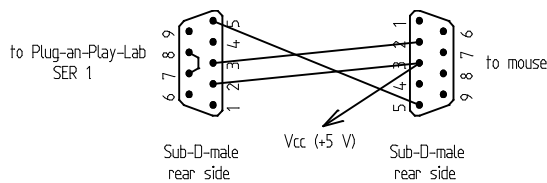


Fig. 6 Simple mouse adapter for the SER 0 interface

4. The Microsoft standard mouse format

Serial data transmission from the mouse to the PC takes place with 1,200 baud, no parity, and 2 stop bits. If the mouse is not in use, there is no serial transmission; every change (button pressed or released, movement in x or y direction) results in transmission of 3 bytes. The transmission sequence is byte 1 bit 0 to byte 3 bit 7; the following chart shows the meaning of every single bit of these 3 bytes:

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Byte 1	0 (1)*	1	L	R	Y7	Y6	X7	X6
Byte 2	0 (1)*	0	X5	X4	X3	X2	X1	X0
Byte 3	0 (1)*	0	Y5	Y4	Y3	Y2	Y1	Y0

Tab. 1 Data format of Microsoft standard mouse

Key:

L	left button's state	1 for pressed, 0 for not pressed
R	right button's state	1 for pressed, 0 for not pressed
X0-X7	X axis movement data	signed binary values
Y0-Y7	Y axis movement data	signed binary values

* There some mice generate a "1"

The assignment of the single bits could not have been more cryptic; however, the BASIC-Tiger[®] will be able to process the different single information to usable data.

5. Some data processing

As we have seen, the mouse always transmits 3 bytes, if at all. This can take place in a very fast sequence. Therefore it is first of all required to reliably detect the first byte of such a "packet". This byte differs from the others in logic "1" at the second highest position, which can be tested relatively easy with Tiger-Basic.

Two pieces of information can be extracted directly from table 1, i.e. both buttons' state. Byte 1 contains them at the bit position 4 (right button) and 5 (left button). Then the program has to test the respective bit and derivate, whether a button is pressed, and if so, which one. Here you

have to pay attention, because when releasing the button again 3 bytes are transmitted, this time with a 0 at the respective bit position.

Things become a bit more complex with the mouse position data (x and y direction), since these values are distributed to different bytes. Let's begin with the y information. The high-order bits Y7 and Y6 are in the first byte. First of all they are separated by a BIT_AND 00001100b command and then, using a bit shift command, shifted 4 times left to the position where they belong. Then byte 3 is processed with a BIT_AND 00111111b command so that only bits 5 to 0 of Y remain. They are already at the right position. Finally the separated bits of byte 1 and 3 are summed up to one number, which contains the y position in one byte. The same is done with the x information. From the first byte the high-order part is extracted by separation and shifting bits six times to the left and from the second byte the low-order part is extracted by separation. The complete x position is determined by also summing up both parts. But what about these data? Is this the actual mouse position? No! This is how to interpret these data:

The x and y values are bytes, i.e. values from 00000000b to 11111111b. In the case of the mouse values with a 0 at the position of bit 7 are treated as positive values and those with a 1 at the position of bit 7 as negative values. This is revealed by the following depiction:

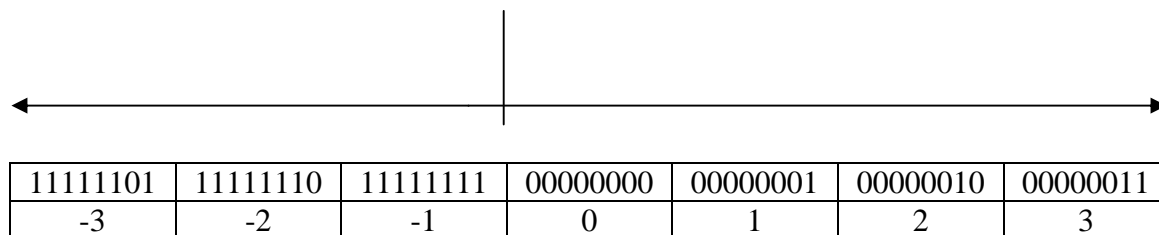


Fig. 7 Interpretation of binary signed numbers

As you can see, bit 7 decides on the number's sign. Those values with a positive sign are converted as usual for binary numbers. But we notice that the positive numbers only range to 127. Values greater than or equal 128 are detected as negative values. With these negative numbers conversion is carried out differently. One step backwards starting from 0 (i.e. 11111111, in common conversion 255) is interpreted as -1. Besides, such a signed presentation of numbers is often used. We, however, apparently will have to use a more complex way of calculating. But the effort remains within limits, as we will soon notice. That is to say exactly the desired behaviour can be achieved with a so-called two's complement.

In the case of bit 7 = 1 the BASIC-Tiger[®] can interpret the steps to be counted backwards as negative numbers with three simple commands (negate bit by bit, add 1, multiply with -1). In the case of bit 7 = 0 nothing is changed and the number is simply adopted.

Table 2 shows some examples for converting signed binary numbers:

Initial number	completely negated	1 added	sign	Decimal value
00000000	→	→	+	0
00000001	→	→	+	1
01111111	→	→	+	127
11111111	00000000	00000001	-	1
11111110	00000001	00000010	-	2
10000000	01111111	10000000	-	128
10000011	01111100	01111101	-	125

Tab. 2 Examples for signed binary numbers

But this is not the whole truth yet. Let's assume, the mouse would be in its initial position (e.g. after RESET or after pressing a key both positions would have the value 0, e.g. stored in each variable XPOS and YPOS). Then the first movement takes place, which can immediately be read as 3 bytes via the serial interface. The x and y position of this first data packet are added to XPOS respectively YPOS and stored. With the second data packet steps in positive respectively negative direction add again (relatively to the first step!). Those steps can be also simply added and so on. Finally you get the position of the mouse in both variables XPOS and YPOS via a series of single movements. If you return to the starting point, x and y should have the value 0 again (theoretically!). In case of fast movements the mouse reacts as follows: The last data packets only arrive when the mouse already stands still again, due to the relatively slow transmission. For this there are buffers in the system which can also detect such difficult motion sequences correctly.

In contrast to the mouse pointer on the desktop we get an absolute position in the BASIC-Tiger[®], which is restricted by the screen border in the case of the PC. A movement beyond the screen border is not registered any more. If the mouse remains on the surface, we can even register major position changes with virtually unlimited numerical values.

Wilke Technology GmbH developed a driver, which takes over the entire control and conversion of mouse values. This is supposed to prevent us from getting fed up with experimenting with the mouse because of too much maths. This application note will deal with the driver in chapter 7.

6. Why the effort?

In this section we will deal with possibilities for applying the Tiger mouse. Especially when setting up independent devices without the Plug-and-Play-Lab we will come to appreciate this simple device.

- First of all you can use the mouse as known from PC, as presented by the program "MAUS02.TIG". This, of course, requires a display.

- With little effort we gain 2 buttons, which can be used without having to integrate the usual keyboard driver.
- The serial connection SER 0 is often not used, so you can implement comfortable control functions even when being short of port lines.
- It is the actual strength of the mouse to be able to determine distances and positions. This can be handy for drive models but also for measuring.
- A menu control via a LC display, no matter if alphanumeric or graphic, could also be interesting. Depending on the motion direction you scroll through the menu, one button is pressed for adopting the current menu option (“Enter”) and the other one is pressed for quitting the menu (“ESC”).
- Last but not least values can be set very precisely. Just imagine you require a default value in a program. Just one movement of the mouse can lead you to the desired number very fast and exact.
- Finally another tip. You can also disassemble the mouse and use the scanning rollers for completely different purposes. Like this you can e.g. set up a “calliper” or a device for measuring the distance of curvilinear paths. Also think of the determination of position for a microscope stage or similar applications.

7. Software

Core piece of the mouse software is the driver "SER1C_K1.TDD", which deals with all important tasks of querying the mouse. Especially the adjustment of parameters for serial communication and processing the “nested” information from the mouse (see chapter 5) are automated.

Another note! You will find a BASIC instruction in the software which is not part of the manual (yet), the LIMIT instruction. It is defined as follows:

var_a = LIMIT (var_b, min_value, max_value)

Variable a, which is calculated from variable b, can only range between the values minimum and maximum. If it becomes greater or smaller, the respective limit value is set.

The functionality of the following software depends on all files, which are attached to this application note under „MAUS.ZIP“, being integrated into your Tiger software:

- Copy the new .INC files into the INC directory of TIGERBAS and replace all old .INC files by the also delivered files with the same name (hint: you should safely store the old files somewhere!).
- Copy the file "SER1C_K1.TDD" as well as all .TAC files into the BIN directory of TIGERBAS (save the old .TAC files somewhere and replace them with the new ones, as done with the .INC files).
- All .BMP files are only necessary for the graphic display and, where applicable, have to be copied to the directory, in which you also store the program “MAUS02.TIG”. Likewise

you can copy them to the directory FLASH, but keep in mind to set the path for the flash files accordingly in Tiger-Basic under Options -> Directories.

Two programs are offered with this article:

The program "MAUS01.TIG" uses the Plug-and-Play-Lab with its alphanumerical standard display and its interface SER0. An adapter is applied to this interface as presented above, so that a serial mouse can be connected directly. Now you can see the state of both buttons in the first row of your display, the last transmitted x and y coordinates as well as the cumulated numerical values of the x and y position.

"MAUS02.TIG" is the modification of a Wilke Technology demo program (MOUSE_01.TIG) which presents the possibilities of mouse control together with the graphic display. This requires a graphic toolkit or an according system. Also here a mouse adapter is required for the serial interface 0 as stated above. A map of Germany is filed in the picture buffer, which is much larger than the size of the display (240x128 pixels). The cursor moves on the visible part of the display, almost as under Windows™. Keeping the left mouse button pressed you can move the complete map "under" the display. Pressing the right mouse button inverts the image.

Have fun experimenting!