

# **The Mouse Driver**

## **Version 2**

### **Programmer's Development Package**

For the Geneve 9640 Computer

© 1990 – Bruce Hellstrom

All Rights Reserved

THE MOUSE DRIVER VERSION 2  
PROGRAMMER'S DEVELOPMENT KIT  
© 1989, 1990 BRUCE HELLSTROM

## Table of Contents

OVERVIEW.....	4
COMPONENTS.....	4
HOW IT WORKS.....	4
SOME TERMS TO KNOW .....	5
1) Disabled.....	5
2) Inactive.....	5
3) Active .....	5
LOADING THE MOUSE DRIVER.....	5
REMOVING THE MOUSE DRIVER .....	5
ACTIVATING THE MOUSE DRIVER.....	6
DEACTIVATING THE MOUSE DRIVER.....	6
PROGRAMMING EQUATES.....	6
THE MOUSE DRIVER FLAG BYTES.....	7
LOADED flag .....	7
ENABLE flag .....	7
DSABLD flag.....	7
THE MOUSE DRIVER COMMAND SET .....	8
Command:   >0001 Set Mouse .....	8
Command:   >0002 Hide Mouse .....	9
Command:   >0003 Show Mouse.....	9
Command:   >04xx Change Mouse Speed .....	9
Command:   >0005 Disable Mouse Driver .....	10
Command:   >FFFF Disable Mouse Driver.....	10
USING THE READ ONLY REGISTERS.....	10
MSX .....	11
MSY .....	11
BUT1 to BUT3.....	11

MSPD.....	11
NOTES ON USING THE MOUSE DRIVER.....	11
IN CLOSING.....	12
ACKNOWLEDGEMENTS.....	13
ADDENDUM .....	13

THE MOUSE DRIVER VERSION 2  
© 1989, 1990 BRUCE HELLSTROM  
ALL RIGHTS RESERVED

## OVERVIEW

The Mouse Driver is a MDOS utility for programmers to effectively offer mouse support to new program development for the Geneve 9640 computer. It is a very easy to use and powerful utility designed to relieve the programmer of the task of tracking and updating mouse position and sprite relocation. The ultimate goal of this driver is to set a standard for other mice that may come along so that programs will not have to be modified to be compatible with another mouse. This in turn should generate more mouse driven software for the Geneve. This package with its examples should allow a programmer to see just how easy it can be to add mouse support to their software without a lot of time consuming code writing.

## COMPONENTS

The Mouse Driver programmer's developer kit comes with the following:

- 1) This documentation manual explaining in detail how to access the Mouse Driver with your program.
- 2) A diskette which contains a registered development copy of the Driver for development of mouse driven software. Also on the diskette is a Mouse Driver test program and commented assembly source code to use as an example for using and programming with the Driver. There is also a Readme file that may contain late breaking information that did not make it into this manual.

## HOW IT WORKS

The Mouse Driver is loaded into memory from the command line and then becomes a part of the MDOS interrupt routine. Once loaded it integrates itself into the interrupt structure and remains present until a cold boot is done or the computer is shut off. The Driver has three states of activity which are outlined in the next section. It can range from being totally transparent to MDOS to being constantly active even at the MDOS prompt. By use of several registers and flags, your program can constantly be informed of the current state and/or status of the Mouse Driver.

## SOME TERMS TO KNOW

There are three states of the Mouse Driver that you should be familiar with when reading this documentation.

- 1) **Disabled** – When in this state, the Mouse Driver will not return any information to the user program. It also does no tracking of the mouse or checks for any button presses. Basically, when disabled, the interrupt routine passes by the Mouse Driver. This state will keep the Mouse Driver from conflicting with programs that use the register space in high memory normally used by the Mouse Driver. The user may put the driver in a disabled state at any time by pressing all three of the mouse buttons at the same time. The programmer has a flag that can be checked to determine if the Driver has been put into this state. This is the normal state of the Driver when first loaded from the command line.
- 2) **Inactive** – When in this state, the Mouse Driver does not return any information to the user program but it is checking button status and it also checks the command register for an activation or disabling command. This state will overwrite some of the register area in high area and may be inappropriate for some problem programs that use memory in that area.
- 3) **Active** - In this state, the Mouse Driver is fully active, returning both location and button status to the user program and updating pointer location regardless of video mode if not hidden by the programmer.

## LOADING THE MOUSE DRIVER

The Mouse Driver is loaded into memory from the command line by typing MOUSE20 at the MDOS prompt at the MDOS prompt. This loads both the MOUSE20 and MOUSE21 files from your disk. If the load is successful, you will see a message: “MOUSE DRIVER INSTALLED” along with the current version number of the Driver and your registration number and name. If you try to load it after it has been previously installed, you will receive an error message. Upon successful installation, the Mouse Driver will be in a ‘disabled’ state.

## REMOVING THE MOUSE DRIVER

Once installed, the Mouse Driver becomes a part of the MDOS interrupt routine and cannot be removed without a cold-boot or turning the 9640 off completely.

## ACTIVATING THE MOUSE DRIVER

If the Mouse Driver is disabled as when first loaded, or after pressing all three buttons, it will be necessary to first write any non-zero value to the byte at >0062 of your program. This will move the Mouse Driver to the inactive state. You will see this address equated later with the other flags and registers your program can use to access the mouse.

If the Mouse Driver is inactive, then writing the >0001 command to the command register will activate the Driver. Commands are explained in detail in a later section.

## DEACTIVATING THE MOUSE DRIVER

If the Mouse Driver is active, it can be put in the deactive state by writing command >0005 to the command register.

To put the Mouse Driver in the disabled state can be accomplished in two ways.

- 1) Pressing all three buttons at the same time will put the Driver in the disabled state.
- 2) Writing command >FFFF to the command register will also put the Driver in the disabled state as above.

Unless in use by a program, the Mouse Driver should be left in a disabled state to avoid any conflicts with programs that do not expect a driver to be loaded in memory. Some programs use the area in memory where the Mouse Driver's registers are located and having the Driver in a de-activated or activated state may cause unpredictable results.

## PROGRAMMING EQUATES

The Mouse Driver can be accessed by programs in any language that allows direct access to memory of the Geneve. The Driver is accessed through a series of registers beginning at address >F200. The Driver also has three flag registers which allow the programmer to know the current status of the Driver at all times. Current versions of Myarc Advanced Basic conflict with the memory used by the Mouse Driver and the Driver should be placed in a disabled state before attempting to run Advanced Basic. A description of each of the registers and flag registers used by the Driver are below.

COMREG	EQU	>F200	The Mouse command register (2 bytes) (wr only)
MSX	EQU	>F202	The current mouse X position (2 bytes) (rd only)
MSY	EQU	>F204	The current mouse Y position (2 bytes) (rd only)

BUT1	EQU	>F206	Current status of mouse button 1 (1 byte) (rd only)
BUT2	EQU	>F208	Current status of mouse button 2 (1 byte) (rd only)
BUT3	EQU	>F20A	Current status of mouse button 3 (1 byte) (rd only)
MSSETX	EQU	>F20C	Mouse set X (for set mouse) (2 bytes) (wr only)
MSSETY	EQU	>F20E	Mouse set Y (for set mouse) (2 bytes) (wr only)
MSPD	EQU	>F210	Mouse speed setting (2 bytes) (rd only)
ENABLE	EQU	>0062	Mouse Driver enable register (1 byte)
LOADED	EQU	>0063	Mouse Driver Loaded flag (1 byte)
DSABLD	EQU	>1901	Mouse Driver disabled flag (1 byte) *

\*This flag is located in the System Header page of memory (Page 0)

## THE MOUSE DRIVER FLAG BYTES

**LOADED flag** – This flag allows the programmer to determine if the Mouse Driver has been previously loaded by the user. This byte is located at >0063 in your task's header page (execution page 0). This byte contains the hex value >FF if the driver is loaded.

**ENABLE flag** - This flag allows the programmer to activate the Driver when in a disabled state by writing any non-zero byte to this flag register. It is located at address >0062 of your task's header page (execution page 0). The programmer can determine if this is necessary by checking the DSABLD flag located in the system header page. (See section on the DSABLD flag.)

**DSABLD flag** - This flag allows the programmer to determine if the Mouse Driver has been disabled either by the user or by software. This byte is located at offset >1901 in the system header (page 0). This is so that any program may access this byte to determine status of the Driver. Your program must place page >00 in one of the mapper registers located between >F110 and >F117 before your program can check this flag. If the driver is disabled, this byte will contain a hex value of >00 if the Driver is disabled and a value of >FF if the Driver is in an active or inactive state.

## THE MOUSE DRIVER COMMAND SET

The programmer communicates with the mouse through a set of commands written to the command register at >F200. Various parameters are set to the necessary registers prior to writing the command. The commands and the necessary parameters are outlined in the next section.

NOTE: All writing to the mouse registers should be done with interrupts disabled (LIMI 0). Interrupts should be enabled after writing your command to the command register. Also, if your task re-maps a new page of memory into execution page 7, you should disable interrupts prior to mapping in the new page 7. You should disable interrupts prior to mapping in the new page and clear the word at >F200 before enabling interrupts again as a precaution against unwanted data accidentally writing a command to the Mouse Driver. An example of this is illustrated in the sample code distributed with this package.

### Command: >0001 Set Mouse

This command sets the current mouse position to the coordinates placed in the MSSETX and MSSETY registers. After placement of the X and Y coordinates, you write >0001 to the command register at >F200. The Mouse Driver will relocate the mouse to the coordinates you specify and activate either the sprite or a text mode flashing cursor pointer for either text mode 1 or 2. The Mouse Driver will automatically adjust itself for the current screen width dependent upon video mode. You should always execute this command after changing video modes to be sure the pointer sprite has the right character designation and color.

#### Example – Setting mouse at X 32 Y 32

LIMI	0	Interrupts OFF!
LI	R0,>0020	
MOV	R0,@MSSETX	Set X coordinate (pixel col)
MOV	R0,@MSSETY	Set Y coordinate (pixel row)
LI	R0,>0001	
MOV	R0,@COMREG	Execute command
LIMI	2	Enable interrupts

### Command: >0002 Hide Mouse

This command hides the mouse pointer sprite by changing its color to transparent so it is invisible. The sprite remains active as do the location registers. In text modes, this command will turn off the flashing text pointer.

#### Example

```
LIMI  0                      Interrupts OFF!
LI     R0,>0002
MOV    R0,@COMREG
LIMI   2                      Enable interrupts
```

### Command: >0003 Show Mouse

This command shows the mouse pointer sprite again after hiding it with the hide mouse command above. If for some reason the sprite was disabled while hidden such as a change to a text mode, this command will reactivate the sprite pointer. In text modes, this command will reactivate the flashing text cursor.

#### Example

```
LIMI  0                      Interrupts OFF!
LI     R0,>0003
MOV    R0,@COMREG
LIMI   2                      Enable interrupts
```

### Command: >04xx Change Mouse Speed

This command is used to change the current mouse speed only. The new speed setting is set with the command in the least significant byte of the command word. Valid speeds are >00 to >07 with >00 being the fastest possible setting.

#### Example

```
LIMI  0                      Interrupts OFF!
```

LI	R0,>0403	Command and speed setting
MOV	R0,@COMREG	
LIMI	2	Enable interrupts

Command: >0005 Disable Mouse Driver

This command will effectively deactivate the Mouse Driver without actually disabling the Driver itself. The Mouse driver will turn off the pointer sprite or flashing text cursor and will not return mouse location values or button status when deactivated. It does, however, still check the command register and checks for all three buttons being pressed to disable the Driver. Using the >0001 Set Mouse command will re-activate the driver once again. The main purpose of this command is for multi-tasking with a program with its own mouse support routines so as to not cause a conflict. While de-activated, the MSPD register will hold a value of >FFFF.

Command: >FFFF Disable Mouse Driver

This command will disable the Mouse Driver. This function can also be accomplished by pressing all three mouse buttons at the same time. Once disabled, the Driver will no longer return any information to the program or update any registers. In order to re-activate the Driver in this state, you must first write any non-zero byte to the byte at >0062 in your program and then use the >0001 command explained above.

Example

LIMI	0	Interrupts OFF!
LI	R0,>FFFF	
MOV	R0,@COMREG	
LIMI	2	Interrupts enabled

## USING THE READ ONLY REGISTERS

There are 6 read only registers which return information about the mouse. These registers are updated approximately 30 times per second and can be read with interrupts enabled.

The positions returned are pixel positions of the mouse and will not exceed the maximum pixel count for the current video mode. The maximum pixel row is always 212 pixels.

**MSX** - This register returns the current X (pixel column) position of the mouse between 0 and the maximum for the current video mode. This register is one word (2 bytes)

Example:      `MOV    @MSX,R0`

**MSY** - This register returns the current Y (pixel row) position of the mouse between 0 and 211 (all video modes). This register is also one word (2 bytes).

**BUT1 to BUT3** – These registers return the current state of the mouse buttons 1 to 3 (left to right). Upon the press of a button, the Driver returns values from >01 to >0A as the button is held down. If the user continues to hold the button, the Driver returns a value of >FF (-1). The counting sequence from >01 to >0A allows a user program to adjust the amount of time the programmer wants to allow the user to hold a button to compensate for programs that check for button presses at different times and also for changes in video modes which can vary results. This is illustrated in the sample source code included with this package. Proper use of this feature can create programs that vary rarely miss the press of a mouse button.

Example:	<code>CB      @BUT1,@H06</code>	* Check for button 1 press
	<code>JH      IGNORE</code>	* They are holding the button

**MSPD** – This register holds the current speed of the mouse 0-7 with 0 being the fastest speed. This register uses a full word with the speed right justified in that word.

## NOTES ON USING THE MOUSE DRIVER

This is some additional information for programmers on using the Mouse Driver.

The Mouse Driver register locations are from >F200 to >F211 of execution page 7 of MDOS. These areas of memory will be overwritten by the Mouse Driver. Your program is free to use any other space above >F212 normally. You may also freely swap any page into the page 7 execution space without disrupting the Mouse Driver. It is recommended that you turn interrupts off when

placing a new page 7 number into the mapper and use the statement CLR @>F200 after paging in your new page before turning on interrupts. This is case the new page might inadvertently hold a value such as >FFFF at that address which will disable the Mouse Driver.

The Mouse Driver should be disabled before entering Advanced Basic to avoid any conflicts with the mouse support of Abasic. Also, it appears that Abasic uses memory where the mouse registers are located.

You should always execute the set mouse command before trying to read any values from the mouse registers the first time. Until the mouse is set and activated, the values returned may be inaccurate.

Your program should be able to run normally with interrupts enabled as the Mouse Driver is very transparent in memory, however interrupts must be disabled when writing to the mouse registers and then re-enabled afterward. Since the Mouse Driver is a part of the interrupt routine, it is necessary for your program to allow interrupts to happen to be able to use the Driver effectively.

My main hope with this program is to develop a standard mouse interface that will lead to programs being written without worrying about mouse support. By standardizing a system of registers, programs can be written generically to allow access to the mouse without a lot of effort by programmers. Also should some other type of mouse come along, if a standard has been set, then a driver can be written for the new hardware that will eliminate the need for programming for several different mice.

## IN CLOSING

I sincerely hope that this Mouse Driver will help in the development of more mouse driven software for the 9640. The mouse support of the 9640 is just one of the greatly untapped resources available to programmers and users. If you have any comments on the Mouse Driver, please feel free to forward them to me at the address listed below.

The Mouse Driver is a registered commercial software and is not freeware or public domain. The copy included in this package is for your development use only and purchasing this package does not authorize duplication or distribution off the Mouse Driver. If you develop software which utilizes this Driver, it will be necessary for you to make arrangements either with myself or an authorized distributor if you wish to distribute the driver with your software. A fee is required for each copy of the Driver distributed and each copy must be encoded with the user's name and a unique registration number.

## ACKNOWLEDGEMENTS

I would like to thank Beery Miller and Al Beard for their help and support in the development of this product. Their suggestions and testing have made this project possible. I would also like to thank the tireless efforts of all programmers who are helping to keep the TI-99/4A and Geneve computers alive for the thousands of users out there.

Bruce Hellstrom

7055 N. Sepulveda Blvd, Apt. 5

Van Nuys, California 91405

(818) 782-2307

Delphi Address:       BLHELLSTROM

## ADDENDUM

Notes from Beery Miller (11/19/2022)

In the 1990's, Bruce Hellstrom added an additional serial based mouse driver that could be connected to the RS232. In late 2022, Beery Miller modified the existing Bus Mouse source code resulting in a mouse driver working with the TIPI and Raspberry PI connected to the PI's USB port.

Source code for all three versions are supplied in the distribution package. Filenames for the mouse drivers in this updated package have been renamed as such:

BUSMOUSE – Bruce Hellstrom's original Geneve 9640 Bus mouse driver plugging into the Geneve.

SERMOUSE – Bruce Hellstrom's original RS232 based mouse driver.

TIPIMOUSE – The TIPI/PI Mouse driver from modified source code. The TIPI Mouse Driver requires version 7.40 or higher of MDOS.