CyberMan™ *3D Controller*

LOGITECH

The most advanced way to master 3D games

For IBM Compatibles

CyberMan is a revolutionary input device that's built specifically for hard-core gamers. It allows you to move quickly and freely in 3D space, instead of enduring the limitations of 2 dimensional motion.

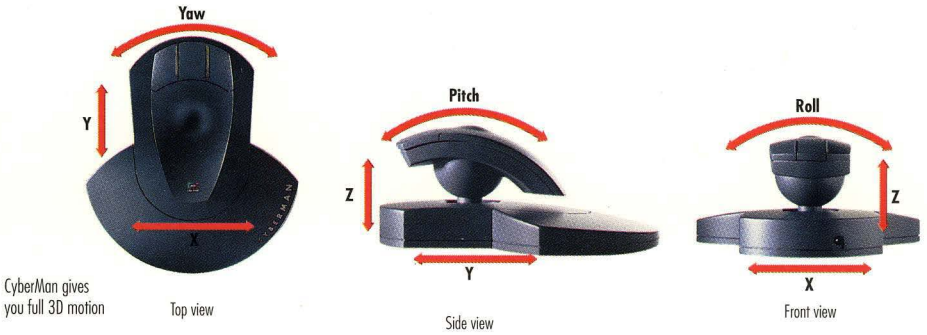
CyberMan makes you the master of space with a comfortable shape and high precision that quickly puts you where you want to be.

The new pulsating tactile feedback feature even lets you *feel* 3D. Now you'll feel the impact of your enemy's weapons as they bounce off your armor. Nothing else makes your 3D games so real.

Extra directional control enhances the game by giving you more freedom to move through the environment. It provides true proportional control in X and Y, and discrete control for Z, pitch, yaw and roll. Because CyberMan provides full 3D motion and includes support for virtual reality games and applications, it's an investment in high-quality entertainment that you won't outgrow.

A whole new way to play games

With all-in-one control, CyberMan significantly reduces the number of times you'll stop in the middle of 3D adventure to give a keyboard command. Absolute mapping to your screen and high resolution give you more motion and more precision with less movement. CyberMan is easy to use, and it's designed to be comfortable, no matter how many passages you navigate, galaxies you conquer, or villains you destroy.



CyberMan gives you full 3D motion

Top view

Side view

Front view

Feel the action with pulsating tactile feedback CyberMan is the first entertainment device that lets you feel the action as well as see and hear it. When you turn on the pulsating tactile feedback, you'll get a reaction from as simple a thing as bumping into a wall. This exclusive CyberMan feature makes the latest games more realistic because you'll feel everything that happens to your character. Depending on the game, you'll know when you've run into a locked door in a blind tunnel, feel bruised as the Amazon you're wrestling throws you to the ground, and feel the full impact of taking a shot in the back.

Works with the games you play

CyberMan works with all the games you play with a mouse – now and in the future. To get the full benefit of CyberMan's 3D motion and pulsating feedback, use it with games that fully support it. CyberMan is compatible with ordinary mice, ensuring compatibility with existing games that rely on conventional 2D mouse control. And, it also provides all the 3D control you'll need for the virtual reality games and applications that lie in your future.

Take advantage of full 3D motion The three-button head of CyberMan sits on a post set into the base. To change X and Y motion, slide the post from side to side or forward and backward. For up and down motion, simply pull up or push down on the CyberMan head. Pivot the head forward or backward to change pitch. Twist the head left or right to manipulate yaw. Tilt the head from side to side for roll. CyberMan makes it easy to get the power and freedom of true 3D control.

Compatibility Guaranteed CyberMan is guaranteed to be compatible with all Logitech and Microsoft mice.

Includes extra value 3D games Try out the latest in 3D games. The CyberMan package includes fast action interactive games so you can immediately experience the excitement of 3D motion and tactile feedback.

Built with Logitech quality

CyberMan is built to last. It is rugged and durable to endure the rigors of high speed, hard driving games. CyberMan comes with a limited lifetime warranty and is backed by Logitech's product support hotline 7 days a week. Logitech is the world's pointing device leader, combining value and quality with advanced technology.

Specifications

Height: 3.5in (89mm)
Length: 7.5in (190mm)
Width: 6.75in (171mm)
Cable length: 6 ft (1.8m)
Connector: 9-pin serial (9- to 25-pin adaptor included)

System Requirements

IBM PC or compatible system with 386 processor or higher PC or MS-DOS 3.3 or higher

CyberMan works without batteries, but the pulsating tactile feedback feature requires 2 AA batteries or an AC adaptor (not included).

Product Support

Call Logitech 7 days a week for technical support, or use our 24-hour electronic bulletin board. Take advantage of the FaxBack line to receive product information by return fax. There is also a Logitech Forum on CompuServe.

Product Support Hotline:	(510) 795-8100
Electronic Bulletin Board:	(510) 795-0408
FaxBack Line:	(800) 245-0000

Satisfaction Guaranteed

Logitech wants you to be perfectly happy with your CyberMan. If you are not completely satisfied with your investment, return CyberMan to your reseller within 60 days with the complete contents of the package and proof of purchase for a full refund. (Full details inside the package.)

Sales Information and Support

Call us for the dealer nearest you or product information during normal business hours.

In the U.S. and Canada: (800) 231-7717

LOGITECH
The Senseware™ Company

Logitech Inc.
6505 Kaiser Dr.
Fremont CA 94555
(510) 795-8500

CyberMan 3D
SWIFT Supplement

Version 1.0

(Final - 12/07/93)

LOGITECH

Logitech Inc.
6505 Kaiser Drive, Fremont, CA 94555

Published 1993
Printed in the United States of America

Logitech, Inc. ("Logitech") has made every effort to ensure the accuracy of this manual. However, Logitech makes no warranties with respect to this documentation and disclaims any implied warranties of merchantability and fitness for a particular purpose The information in this document is subject to change without notice Logitech assumes no responsibility for any errors that may appear in this document.

This edition applies to Logitech mouse driver version 6.31 and later.

Document· ---
Released December 1993

Copy Protection

The CyberMan Toolkit and accompanying disk, available only from Logitech, is not copy protected. This doesn't mean that you can make unlimited copies of it The CyberMan Toolkit and accompanying disk are protected by the copyright laws that pertain to computer software It is illegal to make copies of the contents of the disk, except for your own backup, without written permission from Logitech In particular, it is illegal to give a copy to another person

Trademarks

Logitech, the Logitech logo, and CyberMan are trademarks and CatchWord, MouseMan, ScanMan, and TrackMan are registered trademarks of Logitech, Inc.

IBM PC, XT, AT, and PS/2 are registered trademarks of International Business Machine Corp

Intel is a registered trademark and 80286, 80386, 286, 386, and 486 are trademarks of Intel Corp

Microsoft, MS, and MS-DOS are registered trademarks and Windows and Windows/386 are trademarks of Microsoft Corp

WATCOM is a trademark of WATCOM Systems, Inc

DOS/4G is a trademark of Rational Systems, Inc

High C is a trademark of MetaWare, Inc

Phar Lap and 386|DOS-Extender are trademarks of Phar Lap Software, Inc.

Other products mentioned are the sole property of their respective manufacturers

U.S Government Restricted Rights

Use, duplication, or disclosure by the Government is subject to restrictions as set forth in subarticle (c) (1) (ii) of The Rights in Technical Data and Computer Software clause at DFARS 252 227-7013 or subparagraphs (c) (1) and (2) of the Commercial Computer Software-Restricted Rights at 48 CFR 52.227-19, as applicable. Contractor/manufacturer is Logitech, Inc., 6505 Kaiser Drive, Fremont, CA 94555

Logitech Software License Agreement

This is a legal agreement between you (either an individual or entity), the end user licensee ("Licensee"), and Logitech Inc. ("Logitech"). If you do not agree to the terms of this Agreement, promptly return the disk package and other items that are a part of this product in their original package, with your payment receipt to your point of purchase for a full refund

No part of this software and all accompanying documentation, including manuals, binders and containers (the "Software") may be copied or reproduced in any form or by any means without the prior written consent of Logitech Inc , with the one exception that the Licensee may copy the software solely for backup purposes

License Grant. Logitech grants to the Licensee a nonexclusive right, without right to sublicense, to use this copy of this software on a single computer at a time. You may not rent or lease the Software, but you may transfer the Software on a permanent basis provided you retain no copies and the recipient agrees to the terms of this Agreement You may not reverse engineer, decompile or disassemble the Software. Further, you may not network the Software or otherwise use it on more than one computer or computer terminal at the same time. The Software is owned by Logitech or its suppliers and is protected by United States copyright laws and international treaty provisions

Limited Warranty. Logitech warrants that (a) the software will perform substantially in accordance with the accompanying written materials for a period of ninety (90) days from the date of receipt, and (b) any hardware accompanying the Software will be free from defects in materials and workmanship under normal use and service for a period of one year from the date of receipt. Any implied warranties on the Software and hardware are limited to 90 days and one (1) year, respectively. Some states do not allow limitations on duration of an implied warranty, so the above limitation may not apply to you.

End User Remedies. Logitech's entire liability and your exclusive remedy shall be, for any breach of warranty, at Logitech's option, either (a) return of the price paid or (b) repair or replacement of the Software or hardware that does not meet Logitech's Limited Warranty; provided that the Software and hardware must be returned either to Logitech or to the point of purchase with a copy of your receipt. This Limited Warranty is void if failure of the Software or hardware has resulted from accident, abuse or misapplication. Any replacement Software or hardware will be warranted for the remainder of the original warranty period or 30 days, whichever is longer.

No Other Warranties. LOGITECH DOES NOT WARRANT THAT THE SOFTWARE IS ERROR-FREE LOGITECH DISCLAIMS ALL OTHER WARRANTIES, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NONINFRINGEMENT OF THIRD-PARTY RIGHTS WITH RESPECT TO THE SOFTWARE OR HARDWARE LOGITECH'S LIMITED WARRANTY GIVES YOU SPECIFIC LEGAL RIGHTS YOU MAY HAVE OTHERS, WHICH VARY FROM STATE TO STATE

Limitation of Liabilities. IN NO EVENT SHALL LOGITECH OR ITS SUPPLIERS BE LIABLE FOR ANY CONSEQUENTIAL, INCIDENTAL OR INDIRECT DAMAGES OF ANY KIND WHATSOEVER (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF BUSINESS PROFITS,

BUSINESS INTERRUPTION, LOSS OF BUSINESS INFORMATION, OR OTHER PECUNIARY LOSS) ARISING OUT OF THE USE OF OR INABILITY TO USE THIS LOGITECH PRODUCT, EVEN IF LOGITECH HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. IN NO EVENT SHALL LOGITECH'S LIABILITY FOR ANY CLAIMS WHETHER IN CONTRACT, TORT OR OTHER THEORY OF LIABILITY EXCEED THE LICENSE FEE PAID. BECAUSE SOME STATES DO NOT ALLOW THE EXCLUSION OR LIMITATION OF LIABILITY FOR CONSEQUENTIAL OR INCIDENTAL DAMAGES, THE ABOVE LIMITATIONS MAY NOT APPLY TO YOU.

Term. This license is effective until terminated You may terminate it at any time by destroying the Software. It will also terminate upon conditions set forth elsewhere in this Agreement or if you fail to comply with any term or condition of this Agreement You agree upon such termination to destroy the Software together with all copies, modifications and merged portions in any form.

General. This is the entire agreement between you and Logitech which supersedes any prior agreement whether written or oral relating to the subject matter of this Agreement In the event of invalidity of any provision of this Agreement, the parties agree that such invalidity shall not effect the validity of the remaining portions of the Agreement This Agreement will be governed by the laws of the State of California The United Nations Convention on Contracts for the International Sale of Goods is specifically disclaimed Should you have any questions concerning this Agreement, you may contact Logitech by writing to Logitech Customer Service, 6505 Kaiser Drive, Fremont, CA 94555

Table of Contents

1

Introduction

Purpose

This document is a supplement to the *Logitech Pointing Device Toolkit* or earlier *Logitech Mouse Technical Reference & Programming Guide*. Used in conjunction with that toolkit, this document should provide developers with all of the information they need to fully support the Logitech CyberMan 3D™ Controller in their products.

Scope

This document describes the SWIFT 3D extensions to the conventional IBM PC mouse driver API, the features and SWIFT programming specifics of the CyberMan 3D Controller, and the complete RS-232 interface to the CyberMan controller. This document does not describe in detail conventional mouse API functions, and protocol specifications of Logitech serial pointing devices, even though they apply to the CyberMan controller. Those descriptions can be found in the *Logitech Pointing Device Toolkit.*

Document Organization

This document is composed of the following sections:

Section 1: Introduction

Section 2: Theory of Operation

Section 3: Function Reference

Section 4: CyberMan RS-232 Interface

Section 5: Technical Notes

Appendix A: Logitech Developer Support

Section 1 is this Introduction. Section 2 describes the general operation and programming model of the CyberMan controller, while section 3 is a complete reference to the relevant API functions. Section 4 describes the RS-232 level interface to the CyberMan controller, and Section 5 covers the odd technical detail and some common questions. Appendix A tells you how to get technical assistance.

Contents of Toolkit

The SWIFT Programmer's Toolkit consists of this document, and a diskette containing the following files:

README.TXT	specific information about the files on the diskette
MOUSE.COM	device driver version 6.31, with SWIFT/CyberMan support
SWIFT.H	'C' include file, SWIFT functions
SWIFT.C	'C' code for SWIFT functions
SWIFTCFG.H	include file to configure SWIFT module
DEMO.C	demo program using SWIFT module
WATCOM.MAK	wmake file for WATCOM C/386
BCC.MAK	make file for Borland C++ 3.10
MSC.MAK	nmake file for Microsoft C 6.00A or 7.00

2

Theory of Operation

This section describes the CyberMan peripheral as it appears to a programmer.

CyberMan is an interactive 3D controller for IBM PC compatible computers. It provides:

- Position information in three dimensions
- Orientation information on three axes
- Three buttons
- Tactile feedback to the user's hand, controlled by the application.

As seen through the driver API, CyberMan is a Logitech 3-button mouse, with additional features and functions. The additional functions are called SWIFT functions, and CyberMan is Logitech's first SWIFT device.

What is SWIFT?

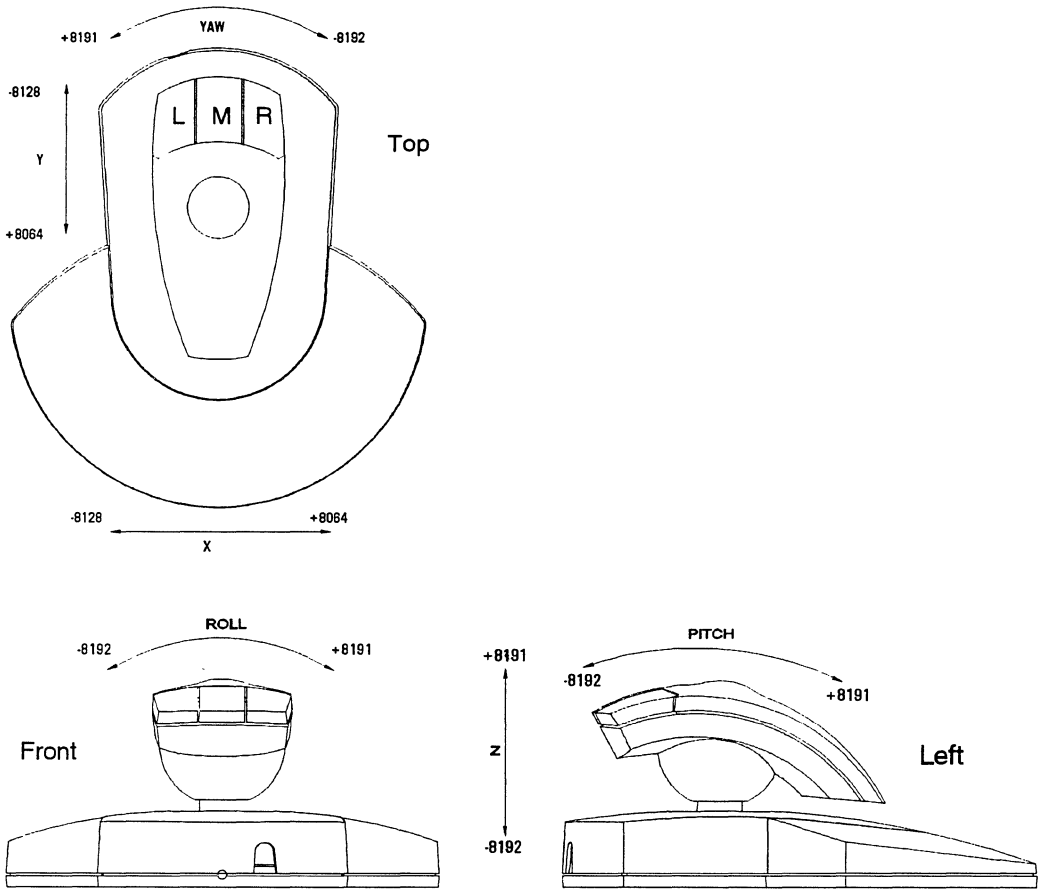
SWIFT: Senseware Interface Technology.

SWIFT devices are devices that give senses to the computer or output sensory feedback to the user via computer control. Currently the only SWIFT device is the Logitech CyberMan. In the future, other SWIFT devices may include different input devices (joysticks, 2D mice, 3D mice), tactile feedback units, visual display systems, temperature sensing devices, external robots, etc.

SWIFT Functions are functions provided by Logitech's mouse driver, starting with version 6.31. These are additional functions beyond the standard mouse functions - they are supported only if the driver recognizes an attached SWIFT device, such as CyberMan, during initial loading of the driver. All standard mouse driver functions are also supported.

Note: SWIFT is a growing specification. If any individual or company would like to add additional devices or commands to the SWIFT specification, please contact Logitech Developer Relations - contact information is in Appendix A.

Device Model



CyberMan

The data ranges for SWIFT position and orientation values are -8192 to +8191. *This does not imply that CyberMan will generate all possible values in this range*. Any version of CyberMan, and any other SWIFT device, will map the full range of physical device motion into approximately this range, with varying resolution. The following notes give specific mappings for CyberMan. The SWIFT Get Static Device Data function returns information about each of the six coordinates: Whether a coordinate is relative or absolute, whether the controller returns to center on that coordinate, and how many bits of resolution the controller has for that coordinate.

For CyberMan, the coordinates have the following properties:

All CyberMan coordinates are absolute.

Z, Pitch, Roll, and Yaw return to center, X and Y do not.

X and Y have 8 bits of resolution - to be precise, there are 254 distinct X and Y values, from -8128 to +8064, in increments of 64.

Z position, and the Pitch, Roll, and Yaw orientation coordinates have 2 bits of resolution, but only 3 values: -8192, 0, and +8191

SWIFT Functions

The following table shows the SWIFT functions that apply to CyberMan. All functions are typically invoked by placing the function number in register AX, other arguments as needed in the other processor registers, and executing an INT 33h.

If an application attempts to call a SWIFT Function in a Logitech mouse driver prior to version 6.31, or a Microsoft or other mouse driver that does not support the SWIFT extensions, the driver will ignore the call and return to the application without altering the contents of the AX, BX, CX, or DX registers.

Alternatively, the diskette that accompanies this manual contains a small module, SWIFT.C, that defines C-callable routines for most of the SWIFT functions. By compiling and linking this module into your program, and including the SWIFT.H header file into your calling modules, you can employ this more convenient interface to the CyberMan controller. See the comments in SWIFT.H for the latest information about the services provided by SWIFT.C.

CyberMan SWIFT Functions

AX Function Number	Description
5301h	get SWIFT 3D position, orientation, and button status
5330h	generate SWIFT tactile feedback
53C0h	exchange SWIFT event handlers
53C1h	get SWIFT Static Device Data and Driver Support Status
53C2h	get SWIFT Dynamic Device Data

Function Reference

Function 0Ch - Install Event Handler

Note: This is a standard mouse driver function - but the standard behavior of this function is extended for SWIFT devices. The following applies *only if* a SWIFT-extended driver is loaded, has detected a CyberMan or equivalent SWIFT device, *and the application has invoked SWIFT Function 53C1h.* These restrictions ensure that existing applications that use Function 0Ch and Function 14h are not affected - until all these conditions are met, this function acts in the conventional way. See also Function 14h, below.

This function sets the address of a user subroutine, called an 'event handler', that the mouse driver calls when specified events occur. When one of the specified events occurs, the driver temporarily stops execution of the main program (as with any interrupt) and calls the event handler. The SWIFT.C module included in this toolkit contains an example of an event handler.

Input:

AX = 000Ch
CX = event mask (see below)
ES:DX = address of event handler subroutine

Output

none.

Setting the event mask to 0 disables the event handler. Your program should disable the event handler before terminating.

Event Mask

Mask Bit	Condition
0	mouse cursor position changed
1	left button pressed
2	left button released
3	right button pressed
4	right button released
5	middle button pressed
6	middle button released
7	other button pressed
8	other button released
9	X coordinate changed
10	Y coordinate changed
11	Z coordinate changed
12	Pitch value changed
13	Roll value changed
14	Yaw value changed
15	'other' condition - including device status changed.

When the event handler is called (via a far call), the following information is available in the processor registers:

AX = event bits (same format as event mask above)
BX = button status (bit 0: left, bit 1: right, bit 2: middle)
CX = horizontal (X) cursor position (not device X coordinate!)
DX = vertical (Y) cursor position (not device Y coordinate)
SI **if** any of AX bits 7 through 15 are set:
= paragraph address of Event Handler Extended Information (below)
else
undefined.
DI *undefined.*
SS:SP = the stack of the mouse driver, or of another application. If your event handler needs much stack, it should switch to its own larger stack.
DS = data segment of the mouse driver - your event handler should save and load it appropriately, and restore it before returning.

Event Handler Extended Information

Word 0	X coordinate value
Word 1	Y coordinate value
Word 2	Z coordinate value
Word 3	Pitch value
Word 4	Roll value
Word 5	Yaw value
Word 6	Button Status Word
Word 7	device-specific Dynamic Device Data Word

See Function 5301h for a description of the coordinates and button status word, and see Function 53C2h for a description of the dynamic device data word. Note that the interpretation of the dynamic device data word is device-specific: It has the meaning described in this document only if a CyberMan or compatible device is active.

When Programming an Event Handler:

Don't call DOS from the event handler.

Don't take long in the event handler: Queue events for later processing.

Don't allow stack checking in the event handler (see your compiler documentation.)

Don't assume the stack is your stack, or that it has much room on it.

Don't assume your DS is loaded: See the example in SWIFT.C.

Do make the event handler a *far* procedure, but not an *interrupt* procedure.

Function 14h - Exchange Event Handlers

Note: This is a conventional mouse driver function - but the standard behavior of this function is extended for SWIFT devices. The following applies *only if* a SWIFT-extended driver is loaded, has detected a CyberMan or equivalent SWIFT device, *and the application has invoked SWIFT Function 53C1h.* These restrictions ensure that existing applications that use Function 0Ch and Function 14h are not affected - until all these conditions are met, this function acts in the conventional way.

This function temporarily substitutes a new event handler for the currently active one, if any. See Function 0Ch above for a description of event handlers.

Recommended practice is to use Function 14h during initialization of your program, saving the previous event mask and handler. At termination of your program, use Function 14h or Function 0Ch to restore the previous event mask and handler.

Input:

AX = 0014h
CX = new event mask (same as Function 0Ch)
ES:DX = address of new event handler

Output

CX = previous event mask (set by previous Function 0Ch or Function 14h)
ES:DX = address of previous event handler

Setting the event mask to 0 disables the event handler.

Function 5301h - Get Position, Orientation, and Button Data

This function returns the SWIFT device's current 3D position data, orientation data, and button status.

Input:

AX = 5301h
ES:DX = Address of status buffer

Output

ES:DX is unchanged.

Position, Orientation, and Button Data Structure Format

Word 0	X position
Word 1	Y position
Word 2	Z position
Word 3	Pitch value
Word 4	Roll value
Word 5	Yaw value
Word 6	Button Status Word

Button Status Word

Bit 0	Right button (1 = Pressed)
Bit 1	Middle button (1 = Pressed)
Bit 2	Left button (1 = Pressed)
Bit 3	Button 4
Bit 4	Button 5
Bit 5	Button 6
Bit 6	Button 7
Bit 7-15	Reserved

Function 5330h - Program Tactile Feedback

This function activates the tactile feedback feature of the SWIFT device.

Input

AX	= 5330h
BH	= motor on time, in units of 5 ms
BL	= motor off time, in units of 5 ms
CL	= tactile burst duration, in units of 40 ms

Output

None

This function begins or ends a 'burst' of tactile feedback. Any tactile burst in progress from a previous call to this function is cancelled. If duration (CL) > 0, the new burst begins immediately and lasts for CL*40 milliseconds. If CL is 0, any prior tactile burst is terminated, and no new burst is started.

A tactile burst consists of repeated On/Off cycles - the tactile feedback motor is turned on for BH*5 ms, then turned off for BL*5 ms, repeating for the duration of the burst.

A value of 0 in BL or BH is interpreted to mean '5 ms'.

Maximum tactile burst duration:	10.2 s (255 * 40 ms)
Maximum 'on' time per cycle:	1.275 s (255 * 5 ms)
Maximum 'off' time per cycle:	1.275 s (255 * 5 ms)
Minimum on or off time per cycle:	5 ms

NOTES:

Use tactile sparingly - if it is used too much, users may become desensitized (or annoyed). We suggest that you allow tactile feedback to be disabled, for those users who find it objectionable.

The tactile motor is the only reason the CyberMan controller needs external power or batteries. The unit is functional in every other respect without them. This means that tactile feedback is the sole cause of battery drain, when batteries are the active power source.

Function 53C0h - Install SWIFT Event Handler

This function sets the address of a user subroutine to be called by the SWIFT driver whenever any of a set of events occurs. The subroutine is called an *Event Handler*. The events of interest are represented by a '1' in the Call Mask. When one or more of the events defined by the call mask occur, the driver temporarily stops execution of your main program (as with any interrupt) and calls the event handler.

This function is a superset of the standard mouse driver functions Install Event Handler (Function 0Ch) and Exchange Event Handlers (Function 14h). Calling any of these functions will supercede any previous call to any of them. In other words, there is only one current Event Handler known to the driver.

Note: The standard mouse Function 0Ch and Function 14h services can be used with SWIFT devices - developers with existing code that uses these functions, and developers working in protected mode, may find it preferable to use Mouse Function 0Ch or Function 14h for event handling - see the preceding sections on these functions.

Please see the comments under Function 0Ch on guidelines for programming event handlers.

Input

AX = 53C0h
ES:DX = Address of Event Handler Setup Data

Output

ES:DX = Address of Previous Event Handler's Setup Data

Setting both call mask words to 0 disables the SWIFT event handler.

Mask words of previous event handler being both 0's indicate that no previous event handler routine is installed.

Event Handler Setup Data

Word 0	Call Mask Word 1
Word 1	Call Mask Word 2
Word 2	Address Offset of Event Handler subroutine
Word 3	Address Segment of Event Handler subroutine
Word 4	Address Offset of Event Data Structure
Word 5	Address Segment of Event Data Structure

Event Handler Call Mask

Word Identifier	Word Mask Bit	Event
Call Mask Word 1	Bit 0	cursor position changed
	Bit 1	left button pressed
	Bit 2	left button released
	Bit 3	right button pressed
	Bit 4	right button released
	Bit 5	middle button pressed
	Bit 6	middle button released
	Bit 7	button #4 pressed
	Bit 8	button #4 released
	Bit 9	button #5 pressed
	Bit 10	button #5 released
	Bit 11	button #6 pressed
	Bit 12	button #6 released
	Bit 13	button #7 pressed
	Bit 14	button #7 released
	Bit 15	Reserved
Call Mask Word 2	Bit 0	X Position changed
	Bit 1	Y Position changed
	Bit 2	Z Position changed
	Bit 3	Pitch changed
	Bit 4	Roll changed
	Bit 5	Yaw changed
	Bit 6	Dynamic Data changed
	Bits 7 - 15	Reserved

When the event handler is called (via a far call), the Event Data Structure is filled in as follows:

SWIFT Event Data Structure

Word 0	X position
Word 1	Y position
Word 2	Z position
Word 3	Pitch value
Word 4	Roll value
Word 5	Yaw value
Word 6	Button Status Word (see Function 1)
Word 7	Dynamic Device Data (see Function C2h)
Word 8	Event bits (same format as Call Mask Word 1)
Word 9	Event bits (same format as Call Mask Word 2)
Word 10	Horizontal (X) cursor position
Word 11	Vertical (Y) cursor position

See Function 5301h (above) for position, orientation, and button status data explanation.

See Function 53C2h (below) for interpretation of the Dynamic Device Data word.

Because the event handler is called from the driver interrupt code, special care must be taken with the stack and data segments. The current stack segment may not be that of your application upon entry to the event handler, so it is important to keep stack usage (e.g. procedure calls) to a minimum inside the handler. With most compilers, stack checking should be disabled for the event handler, as it can produce spurious failures.

Upon entry to the event handler, the data segment is that of the mouse driver. This may be a problem in some high level languages, if the compiler makes assumptions about the contents of DS - consult your compiler's documentation. Some compilers have a 'loadds' or similar modifier that takes care of this. Assembly language programmers should either reference variables using CS (if CS=DS) or set DS to the correct value and restore DS on leaving the handler.

Function 53C1h - Get Static Device Data and Driver Support Status

This function returns the device type and version number of the connected SWIFT device and also informs the application if the installed mouse driver supports SWIFT Function calls.

Input

AX = 53C1h
ES:DX = Address of Static Device Data buffer

Output

AX = Return Status,
1 = Driver supports SWIFT functions
Any other value = SWIFT functions not supported
ES:DX = Buffer address, same as Input

Static Device Data

Byte 0	Device Type
Byte 1	Major Version Number
Byte 2	Minor Version Number
Byte 3	X Coordinate Descriptor
Byte 4	Y Coordinate Descriptor
Byte 5	Z Coordinate Descriptor
Byte 6	Pitch Coordinate Descriptor
Byte 7	Roll Coordinate Descriptor
Byte 8	Yaw Coordinate Descriptor
Byte 9	Reserved

Device Type Encoding:

0 = No SWIFT device connected
1 = Logitech CyberMan connected
2 - 255 Reserved for Logitech SWIFT devices

Device major version number (e.g. 1 for version 1.50)
Device minor version number (e.g. 50 for version 1.50 or 3 for version 1.03)

Coordinate Descriptor

Bit 0-3	Bits of resolution (0..15)
Bit 4	Reserved
Bit 5	Return to Center (1 = return to center)
Bit 6	Absolute/Relative (1 = absolute, 0 = relative)
Bit 7	Reserved

Bits of resolution tells you the number of bits used by the device to encode the coordinate. The actual number of values or positions that the SWIFT device generates may be somewhat less than the bit resolution would suggest. For example, the CyberMan controller uses 2 bits of resolution for Z, Pitch, Roll, and Yaw - but only generates 3 distinct values for these coordinates.

Return to Center means that the SWIFT device returns to the center or neutral (absolute 0) position in this coordinate when the user releases the device or stops applying force in this coordinate.

Function 53C2h - Get SWIFT Dynamic Device Data

This function returns the dynamic device data of the connected SWIFT device.

Input

AX = 53C2h

Output

AX = Dynamic Device Data Word

Format of Dynamic Device Data Word

Bit 0	External Power Connected (1 = yes)
Bit 1	External Power Level (1 = Too High, 0 = OK)
Bits 2 - 15	Reserved (0)

For CyberMan, if the External Power Level is too high the tactile motor will not be allowed to function in order to prevent possible damage to the motor. Nominal external power level is +3V - we don't choose to specify the exact voltage that is considered 'Too High'.

The External Power Level is only checked when a tactile burst is initiated, if external power is connected. Therefore, recommended practice is to install an event handler of some kind, with the mask set to include 'dynamic data changed' - see the specific event handling functions for details. When an event occurs, check for this condition using the information passed to the event handling function. You might want to 'hit' the tactile feedback once during your program's initialization phase, just to force the over-voltage test.

What should you do if External Power Level Too High is reported? Issue a message to the user saying that the power supply they have connected to their CyberMan (assuming it is a CyberMan) is providing too high a voltage. While they may have connected 'the wrong' power supply to the unit, it's also possible they have a switchable power supply that is simply set inappropriately.

The External Power Level Too High condition is reset by disconnecting the external power, or by issuing a tactile burst after the input voltage has been corrected.

3

CyberMan RS-232 Interface

This section describes the serial RS-232 interface to the CyberMan 3D Controller, including the mechanical and electrical connection, as well as the character-level protocol between the host system and the CyberMan controller.

Mechanical/Electrical

CyberMan is shipped with a standard RS-232 subminiature DB-9S female connector, compatible with the serial ports on an IBM AT. The package also includes a DB-9 to DB-25 adapter.

Pin Assignments

9 Pin	25 Pin	Wire Name	Comments
shell	1	protective ground	
3	2	RXD (Receive Data)	data from host to device
2	3	TXD (Transmit Data)	data from device to host
7	4	RTS (Request To Send)	used for power, reset
8	5	CTS (Clear To Send)	
6	6	DSR (Data Set Ready)	
5	7	signal ground	
4	20	DTR (Data Terminal Ready)	used for power

CyberMan conforms to the hardware interface specifications of a Logitech Type M, V, or W serial mouse. Note that RTS+DTR together should be able to source about 10 mA.

Initialization

As with other Logitech serial mice, the CyberMan controller is reset when DTR and RTS go high after having both been low, or when DTR is high and RTS is dropped for at least 100 ms. The first case is typically a host system power-on or hard reset. The second case is generated for example by mouse driver function 0.

Depending on the specific serial device, sometime between 10 ms and 110 ms after RTS goes high, the device will send an identification sequence to the host. For CyberMan, this identification sequence is "M3" - the same as any Logitech Type M, V, or W 3-button mouse. The '3' will follow the 'M' by not more than 100 ms.

After a reset, the CyberMan controller emulates a 3-button, M+ Logitech mouse. This means that it selects the M+ protocol, at 1200 baud, 7 data bits, 1 stop bit, no parity. Full details of the M+ protocol are given in the *Logitech Pointing Device Toolkit.*

Commands and Reports

The CyberMan controller has two protocol modes: M+ mode, and SWIFT mode. In M+ mode the CyberMan acts like a normal 'A-device', sending 2-D status reports and so on. In SWIFT mode the CyberMan responds to additional special commands, and sends special 3D status reports and dynamic device status reports instead of normal X-Y mouse reports.

In M+ mode the CyberMan controller will respond to these commands:

*#	Send device identification
*c	Send copyright and version number in ASCII
*S	Enter SWIFT mode.

In SWIFT mode, communication parameters are set to 9600 baud, 8 data bits, 1 stop bit, no parity. In this mode, the CyberMan controller responds to two special commands:

!T	Initiate Tactile Feedback
!S	Send Static Device Status

In SWIFT mode, the controller does not generate M+ mouse reports. Instead it generates 3D input reports, and dynamic status reports, both described below. The only way to change the controller from SWIFT mode back to M+ mode is by doing a reset.

SWIFT Dynamic Status Report

This report is transmitted in two cases:
1) If the user plugs or unplugs the external power, typically an AC adapter, changing the External Power Connected status.
2) At the start of a tactile feedback burst, if External Power Level is found to be too high. *This is the only time the external power level is tested.*

Dynamic Status Report

	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
byte 1	1	1	0	res.	res.	res.	EPL	EPC
byte 2	0	res.	res.	res.	res.	res.	res.	res.
byte 3	0	res.	res.	res.	res.	res.	res.	res.
byte 4	0	res.	res.	res.	res.	res.	res.	res.

res.	reserved, set to 0.
EPL	External Power Level (0 = OK or not connected, 1 = Too High).
EPC	External Power Connected (1 = Yes, 0 = No).

CYBERMAN 3D SWIFT SUPPLEMENT 1.0

ERRATA

Section 3, "CyberMan RS-232 Interface", subsection "Commands and Reports", paragraph 3, page 20:

"In SWIFT mode, communication parameters are set to 9600 baud..."

This is incorrect; it should be "4800" baud.

Caution! When your program issues the "***S**" command, it must wait until the sequence has actually been transmitted before reprogramming the baud rate. In many environments, characters are queued for output in memory or in the UART itself. If the baud rate is changed before the ***S** has been transmitted, it will be sent at the new baud rate, and will not be correctly received by the CyberMan controller.

2/28/94

SWIFT Static Device Status Command

In SWIFT mode, upon receipt of the Static Device Status command "!S", the CyberMan controller sends the following Static Device Status Report.

Static Device Status Report

byte	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
1	1	0	1	res.	res.	res.	res.	res.
2	0	VMJ6	VMJ5	VMJ4	VMJ3	VMJ2	VMJ1	VMJ0
3	0	VMN6	VMN5	VMN4	VMN3	VMN2	VMN1	VMN0
4	0	X A/R	X RTC	res.	X B3	X B2	X B1	X B0
5	0	Y A/R	Y RTC	res.	Y B3	Y B2	Y B1	Y B0
6	0	Z A/R	Z RTC	res.	Z B3	Z B2	Z B1	Z B0
7	0	XR A/R	XR RTC	res.	XR B3	XR B2	XR B1	XR B0
8	0	YR A/R	YR RTC	res.	YR B3	YR B2	YR B1	YR B0
9	0	ZR A/R	ZR RTC	res.	ZR B3	ZR B2	ZR B1	ZR B0
10	0	res.	res.	res.	res.	res.	res.	res.
11	0	res.	res.	res.	res.	res.	res.	res.
12	0	res.	res.	res.	res.	res.	res.	res.

res.	reserved, set to 0.
VMJ	Version number, Major - 7 bits.
VMN	Version number, Minor - 7 bits.
X, Y, Z A/R	X, Y, and Z position data is absolute (1) or relative (0).
XR, YR, ZR A/R	X, Y, and Z rotation data is absolute (1) or relative (0).
X, Y, Z RTC	X, Y, and Z position return to center (1 = yes, 0 = no).
XR, YR, ZR RTC	X, Y, and Z rotation axes return to center (1 = yes, 0 = no).
X, Y, Z B3-B0	X, Y, and Z position data, bits of precision (0..15).
XR, YR, ZR B3-B0	X, Y, and Z rotation data, bits of precision (0..15).

An *Absolute* coordinate represents the position or orientation of the device - The value of an absolute coordinate in any given report is more or less proportional to displacement of the device in some dimension from a '0' position. It does not change value when the device is not moving in that dimension.

A *Relative* coordinate represents the latest change in position or orientation of the device in some dimension. The value of a relative coordinate in any given report is more or less proportional to the amount of change in some dimension, since the last report. It goes to 0 when the device is not moving in that dimension.

Return to Center means that when the user releases the device, or exerts no pressure on a given coordinate, the device automatically returns to the 'center' position.

SWIFT 3D Position, Rotation, and Button Data Report

In SWIFT mode, the CyberMan transmits the following 3D Position, Rotation, and Button Data Report whenever there is a change in button status or a change in any coordinate ince the last report was generated.

3D Position, Rotation, and Button Data Report

	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
byte 1	1	0	0	res.	res.	L	M	R
byte 2	0	x7	x6	x5	x4	x3	x2	x1
byte 3	0	x0	y7	y6	y5	y4	y3	y2
byte 4	0	y1	y0	z1	z0	xr1	xr0	yr1
byte 5	0	yr0	zr1	zr0	res.	res.	res.	res.

General notes: All coordinates are represented in 2's complement format.

res.	reserved (currently set to 0)
x7-x0	absolute X position: -128 to +127
y7-y0	absolute Y position: -128 to +127
z1-z0	absolute Z position: -2, 0, or +1
xr1-xr0	X-axis rotation (pitch): -2, 0, +1
yr1-yr0	Y-axis (roll): -2, 0, +1
zr1-zr0	Z-axis (yaw): -2, 0, +1
L, M, R	Middle, left, and right button status: 0, 1 (1=pressed)

SWIFT Tactile Feedback Command

On receiving the !T command sequence, the CyberMan controller begins producing *tactile feedback* - a mechanical pulse or vibration the user can feel, under the palm of his or her hand. The vibration is produced by a small DC motor with an eccentric weight on the shaft.

Tactile Feedback Command Sequence

byte 1	'!' ASCII code (21h)
byte 2	'T' ASCII code (54h)
byte 3	on-time, in units of 5 ms
byte 4	off-time, in units of 5 ms
byte 5	total duration, in units of 40 ms

Kickstart: There is a ramp-up or 'kickstart' time of 25 ms at the beginning of every tactile feedback burst.

Example: A tactile burst with on-time = 5 ms, off-time = 5 ms, total duration = 40 ms. This corresponds to the command sequence (in hex): 21 54 01 01 01. This will generate the following pattern:

ON	30 ms	25 ms kickstart + 5 ms on-time, 1st cycle
OFF	5 ms	5 ms off-time, 1st cycle
ON	5 ms	5 ms on-time, 2nd cycle
OFF		shut off after 40 ms total

4

Technical Notes

This section contains miscellaneous technical notes that don't fit into the preceding sections, as well as answers to common questions about programming and supporting the CyberMan controller.

User Interface Notes

• There are several different ways to employ CyberMan as an input device - whichever way you choose, please test it carefully for usability and user acceptance - this is in your interest as well as Logitech's. Also, because there is no standard way of using CyberMan, document the specific use of CyberMan with your software.

• If you use X-Y position to control motion or some other continuous program action, please leave a 'dead zone' around the center position - it is almost impossible for a user to position the controller at exactly (0, 0). Try having a dead zone that is 10%-20% of full range, but you will have to experiment. Outside the dead zone, make response (e.g. speed) approximately proportional to distance beyond the dead zone.

• If a software function, particularly in a game, is controlled by both X-Y and Pitch-Roll-Yaw, have the Pitch-Roll-Yaw input take precedence, or at least have the control inputs be additive.

• If you use the digital inputs (Pitch, Roll, Yaw, and Z-axis) to control motion, try giving them more 'analog' feel by using acceleration. For example, initial Pitch might cause slow forward motion, but as it is held, the rate of motion increases up to some maximum. This allows short bursts to be used for fine movement.

• Logitech would be happy to review and give constructive suggestions and feedback on the use of the CyberMan controller in your software product - contact Logitech Developer Relations as described in Appendix A.

Loading the driver HIGH

Logitech mouse drivers can in general be loaded HIGH under DOS 5.0 and later, but there is one complication.

The version 6.31 driver needs a contiguous block of at least 44KB to load, even though it only occupies about 21KB of memory after loading. This is a fairly large UMB, so it is not uncommon to see the mouse driver failing to load high because there is not a large enough free block. Usually, the order of loading can be changed so that the mouse driver is loaded earlier, and this resolves the problem.

Protected Mode

The mouse driver is a *real mode* driver. If your program is executing in protected mode, you must ensure that any buffers passed into the driver are allocated entirely in DOS memory. This restriction applies to the following pointers in the SWIFT API:

- The address of the status buffer passed to Function 5301h
- The address of the Event Handler Setup Data block passed to Function 53C0h
- The address of the Event Handler subroutine, passed in the Event Handler Setup Data block
- The address of the Event Data Structure, passed in the Event Handler Setup Data block
- The address of the Static Device Data structure passed to Function 53C1h

The sample C code included in the CyberMan Programming Toolkit, includes code that works in protected mode with the Watcom C/386 compiler, and the Rational Systems DOS/4GW DOS Extender. See the file SWIFT.C.

Note: Using the Rational Systems DOS/4GW DOS Extender provided with Watcom C/386, it is not possible to use the function 53C0h Event Handler service - use standard mouse driver functions 0Ch or 14h, as shown in the sample code.

Artwork

Logitech can provide you and your company with CyberMan-related artwork, in either mechanical or electronic form, for use in your packaging, advertisements, manuals, and other literature - contact Logitech Developer Relations for further information.

Appendix A

Logitech Developer Support

Logitech aids its developers with an automated fax service, electronic bulletin board services, and developer support.

If you need support for your Logitech toolkit, we recommend that you read this appendix first, so you'll know how and where to get it.

The following sections describe the available Logitech developer support services.

Customer Service, Technical Support or Developer Relations?

This section tells you who to contact for appropriate support.

Customer Service. Logitech Customer Service provides *non-technical* product support, such as product pricing, product replacement, upgrade and update information, product warranty, and order status.

Technical Support. Logitech Technical Support provides *technical* product support, such as software or hardware questions.

Developer Support. Logitech Developer Relations provides developer support, such as toolkit questions or how to register as a developer.

Internet E-mail to Logitech Support Services

Both Developer Relations and Technical Support can be reached via Internet E-mail: This is a very effective way to communicate with us.

Subscribers to CompuServe, America On-Line, and most other on-line services can send and receive Internet e-mail, as can users of many academic and corporate computer systems. Consult the printed or on-line documentation, or local system administrator for your on-line service or host computer system.

When contacting Logitech via e-mail, please state the nature of your request, and what product or products you are refering to. Also please include your real name, address, phone number, and fax number if you have one.

Logitech Developer Relations: **developer_support@logitech.com**

Logitech Technical Support: **tech_support@logitech.com**

Sending a Fax to Logitech Developer Relations

The fax number for Logitech Developer Relations is

(510) 713-5038

Please address all communication to: Attn. Developer Relations.

Logitech FaxBack Service

FaxBack™ is a toll-free, automated fax response service. Using your touch-tone telephone and fax machine, you can request many types of documents: most commonly-asked questions, available toolkits, technical notes, and developer services. FaxBack sends the documents to your fax machine in minutes.

First, call FaxBack and order the Logitech Developer FaxBack catalog that lists the latest available developer support documents. For the catalog, request document number 4700. To reach FaxBack, call:

(800) 245-0000 (in the US)

Logitech On-Line

If you have a modem, you can communicate with Logitech on the following electronic bulletin boards.

LBBS (Logitech Bulletin Board Service)

With a 300, 1200 or 2400 baud modem, call LBBS 24 hours a day, 7 days a week. Set the communication parameters on your modem to either: 7 bits, 1 stop bit, and even parity; or 8 bits, 1 stop bit, and no parity.

In the United States, call: **(510) 795-0408**
In Europe, call: **++41 (0) 21-869-98-17**

CompuServe

If you are a member of CompuServe Information Service, you can get the latest Logitech Product Support information.

From the CompuServe system prompt, type:

GO LOGITECH

Support Phone Numbers and Addresses Worldwide

This section includes support addresses and telephone numbers. The Logitech Developer Support phone line connects you to an automated attendant, which is monitored throughout the day in order to provide timely response from Logitech.

You can also write for support. Address your letter to the appropriate Logitech address and to the attention of Developer Relations and your toolkit (i.e. Attn: Developer Relations, CyberMan Toolkit). Please include your daytime phone number and the best time to reach you.

U.S.A. and Canada

Product Support: **(510) 795-8100**
(510) 505-0978 (fax)
Developer Support: **(510) 713-5338**
(510) 713-5038 (fax)
Logitech Inc.
6505 Kaiser Drive
Fremont, CA 94555

Switzerland, Europe, Africa, & Middle East

Product Support and Developer Support
(Switzerland): **++41 (0) 21-869-98-51**
For the rest of Europe: ++41 (0) 21-869-98-55
Logitech SA
CH-1122 ROMANEL/MORGES

Logitech Far East Ltd.

Product and Developer Support: **++886 (0) 2 746-6601**
No. 2 Creation Road 4, Science - Based Industrial Park
Hsinchi Taiwan R.O.C.