This is the text-only version of the MicroScribe-3D Software Development Kit Manual. Figures have been omitted. If you would like a copy of the manual, please contact Immersion Corp at (800) 893-1160. Cost is $10 plus shipping.

CHAPTER 1 Product Overview

Congratulations on your purchase of an Immersion MicroScribe-3D. This manual provides you with the information you need to quickly integrate the Immersion MicroScribe-3D into your computer system.

FUNCTIONAL OVERVIEW

MICROSCRIBE-3D

The MicroScribe-3D is a high-resolution three-dimensional digitizing system that allows you quickly generate detailed computer models of physical objects, models, or mechanical parts by simply tracing their contours. The models can be saved in standard file formats and exported to your favorite 3-D graphics and animation or 3-D modeling software packages. The MicroScribe-3D can also be used as a powerful 3-D user interface device. By specifying x, y, and z positions and roll, pitch, and yaw orientations, the MicroScribe-3D can be used to control cursor movement, point-of-view, light sources, or any other 3-D positioning task in a graphical virtual environment. The MicroScribe-3D is constructed from strong, light-weight graphite components, high precision optics for extreme accuracy and resolution, and instrumentation bearings for smooth movement and operation. The system has a large 50" spherical workspace with a mean accuracy of 0.015".

The MicroScribe-3DX offers superior accuracy. The 3DX has the same 50" workspace as the regular 3D model, but the accuracy is down is 0.009". This model is recommended for engineering professionals and scientists.

The MicroScribe-3DL provides accuracy of 0.025" and offers an expanded workspace of a 66" sphere. The added workspace size makes digitizing more flexible.

PARTS OVERVIEW

The MicroScribe is shown in Figure 1 with the various parts of the system highlighted.

FIGURE 1 MicroScribe 3-D Parts and Features: the stylus, the stylus holder, the digitizing arm, the counterweight, the power indicator, and the base.

TECHNICAL OVERVIEW

IMMERSION 6 DEGREE OF FREEDOM ARM

The MicroScribe-3D is a system of mechanical linkages ending in a stylus that allows the user to specify x, y, and z positions and roll, pitch, and yaw orientations. The system consists of a free-standing, self-contained digitizing arm with high-resolution optical encoders and an embedded microprocessor and electronics to communicate with a host computer via a standard RS-232 serial port. The position and orientation of the stylus is uniquely determined by the configuration of the five linkage arm. The on-board microprocessor determines this configuration by reading the optical encoders and reports this information to the host computer. The device also takes input from optional pedals and buttons which can be plugged into its back panel. Future models will include an optional rotary table, which offers added versatility and enbles the digitizing of even larger objects.

When the MicroScribe-3D is turned on in the home position, it is automatically calibrat-

ed. Next, it will automatically determine the serial baud rate of the host computer and begin communications. In normal operation, the device waits for and responds to commands from the host computer. The host computer can request information from the device such as the joint angles, timing information, communications and other system parameters, or the state of remote input switches.

Writing software for the device requires simple serial communications functions and an understanding of Immersion Corp.'s communications protocol. Both are provided by the Software Development Kit (SDK) described in Chapter 3. The SDK is a convenient library of C functions that makes programming the MicroScribe-3D quick and straightforward. Chapter 4 sections "Command Specification" and "Response Specification" provide exact software protocol specifications for the MicroScribe-3D. Although programmers will be able to write software for these devices without reading these sections, it is highly recommended that they do so.

Release Notes

The MicroScribe-3D was introduced in April of 1995. This manual contains a complete programmer's reference for the MicroScribe-3D. The Software Development Kit (SDK) is functionally similar to the Developer's Programming Library (DPL) of the older Immersion Probe and Immersion Personal Digitizer, but they are not compatible. The most notable differences are:

¥ The MicroScribe-3D will work only with firmware version MSCR1-0 or later. The Probe and Personal Digitizer will work only with firmware version HCI2-0 or later.

¥ The MicroScribe-3D SDK uses the general Denavit-Hartenberg parameter format to describe the digitizer arm.

¥ The MicroScribe-3D SDK no longer allows the user to change factory settings for home position and physical parameters.

CHAPTER 2 Background Information

This chapter covers the following topics:

¥ Modes of Operation

¥ Setting Up and Caring for Your Microscribe-3D

¥ MicroScribe-3D Reference Coordinate System and Suggested Usage

Modes of Operation

Selecting Modes

The MicroScribe was designed to be a versatile tool for general purpose manual interaction with three dimensional spaces. To allow the device to meet the needs of a wide range of applications, we have provided a variety of operating modes from which to choose. These different modes of operation are completely selectable from software, even in mid operation. As a result no buttons or awkward DIP switches are needed to configure the device for your particular application. For specific technical details on selecting modes of operation from software, refer to Chapter 3. The following section briefly describes each mode of operation and its beneficial features and limitations. Please review the modes of operation and choose the one that most closely matches your needs.

Description of each Mode of Operation

High Speed Position Tracking (3 DOF)

In this mode the MicroScribe only reports the X,Y,Z position information of the stylus. This mode is faster than the full position and orientation mode, which requires addition-

al calculations. Thus for high speed applications where only position information is required, this mode is optimal. Functions for this mode (e.g., arm_stylus_3DOF_update(), arm_stylus_3DOF_motion()) are explained in Chapter 3.

6 DOF Position and Orientation Tracking

In this mode the MicroScribe reports the X,Y,Z position and roll, pitch, yaw orientation information of the stylus. Functions for this mode (e.g., arm_stylus_3DOF_update(), arm_stylus_3DOF_motion()) are explained in Chapter 3.

Foreground Operation

In foreground mode, the host computer issues a command and waits idly for the Immersion MicroScribe-3D to respond. This takes just under a millisecond, plus communications time if the baud rate is not very high. If the host application is also doing complex sound or graphics processing, then the application may be too sluggish. To reduce the application's lag, the host can use a higher baud rate or make use of the time between requesting information and receiving it from the communications hardware with background or motion-sensing mode.

Background Operation

In background mode, the host issues a command and is then immediately free to perform other processing while waiting for the response packet. The host must periodically check on the status of the data that it is waiting for, but it is otherwise free to update graphics displays, deal with other peripherals, or plan its next task. When a response packet arrives, it will be parsed as usual, and its data will be stored in the appropriate arm_rec . The host can then issue another command to the MicroScribe-3D when it needs to. For programming details, see the function arm_check_bckg() in the next section. All functions related to background commands contain the string _bckg in their names.

Motion-Sensing Operation

In motion-sensing mode, the host simply issues one command at the beginning of the sequence, and the communications hardware responds with a whole series of data. Response packets can be generated by any change of state of the MicroScribe-3D. Such state changes include button presses, analog (controller) signal changes, passage of time, and physical motion of the arm itself. When issuing the command, the host specifies which of these should generate a response packet. Motion-sensing mode terminates when any new command is received by the Immersion MicroScribe-3D. It is very important to check for data more frequently than the specified repeat rate. Otherwise, data will pile up in the host's input buffer, eventually filling it up and causing some data to be lost.

Setting Up and Caring for Your Microscribe-3D

Moving the MicroScribe-3D Around the Home or Office

In addition to using the proper packaging materials for transporting your MicroScribe-3D, the user must also take precautions when carrying it around the home or office.The MicroScribe's counterweight can induce large jolts to the Microscribe-3D if it is moved around with the stylus inserted in the stylus holder (i.e. in the home position). The large handle-like base of the MicroScribe-3D can be used to carry the system around only after the stylus has been withdrawn from the stylus holder. Carry the Microscribe-3D with both hands to prevent the links from hitting against each other. Although the stylus holder has been designed to release the stylus when the system is transported, please use caution and follow the above recommendations to eliminate the potential for damage to the MicroScribe-3D.

WARNING: Never transport the MicroScribe-3D with the stylus inserted into the stylus holder. This applies even to moving the system around the home or office.

MicroScribe System Connections

The following must be connected to the back of your MicroScribe-3D before you begin operation:

1. Serial cable to the host computer

2. Power supply cable.

3. Digital Foot Pedal/Accessory cable. (optional, supplied separately)

WARNING: Because pinouts on similar cables can differ from those provided by Immersion Corporation, all plug-in accessories must either be provided by or approved by Immersion Corporation to avoid voiding your MicroScribe-3D manufacturer's warranty.

The third socket allows you connect your MicroScribe-3D to a number of optional accessories, including a digital foot pedal, a twin digital foot pedal, and/or a rotary table. A digital foot pedal allows the user to digitize without typing on the keyboard to select or specify points. Up to two digital foot pedals can be used, and a twin pedal is available from Immersion Corporation. The third socket of the MicroScribe-3D also supports a rotary table (release date still unavailable). The rotary table allows a user to fix a workpiece on the table and then digitize all sides simply by rotating the table. Firmware, hardware, and software support for the rotary table already exists in your MicroScribe-3D. A rotary table (or any accessory similarly designed) and up to 2 digital foot pedals can be connected to a single MicroScribe-3D. The two foot pedals plug into two sockets on the rotary table.

CAUTION: While handling all cables, hold them by the connector and not by the cord.

FIGURE 2 MicroScribe-3D Rear Panel Connections. From left to right: power switch (ON when pressed in the left position), power input (connect to 2.1 mm connector from power transformer), serial I/O (connect to computer using mini-DIN 8 cable), and digital pedal/accessory input (connect to digital pedal, rotary table, and other accessory).

First connect the serial cable. The serial cable is used to communicate with the host computer and also provides protection against static electricity build-up in the arm. Because of these critical functions, please be sure that the serial cable is inserted firmly into the large round socket second from the right on the back of the MicroScribe-3D. See Figure 3 for details.

IMPORTANT: If you ever receive an electrical shock from your MicroScribe-3D, power the unit down immediately and make sure that the serial cable is connected firmly to both the MicroScribe and to your computer.

Plug the digital foot pedal (if provided) into the large round socket all the way to the right on the back of the MicroScribe-3D. The pedal is designed to be foot operated and should therefore be placed on the floor under the desk at which the MicroScribe will be used. Other accessories may plug into the pedal port as well. Please consult documentation on such products for instructions on proper use.

Be sure the MicroScribe-3D is turned off (the switch on the back should be pressed to the right) before connecting the power supply cable. If off, plug the round plug from the power supply into the opening on the back panel of the MicroScribe-3D as shown in Figure 3.

Make sure that all cables are out of the way to prevent tripping on them, accidently pulling them out, etc.

Bolting the MicroScribe3-D down (optional)

The MicroScribe-3D is designed to sit safely and securely on a flat, rigid surface without the aid of any fasteners. However, you may opt to mount it to your work surface by

using a single bolt (size 3/8"-18) inserted through your work surface up through the center of the MicroScribe-3D base. Be sure that the bolt you extends NO MORE than 1/2" (1.25 cm) into the MicroScribe-3D. Longer bolts might interfere with the internal electronics and damage the system. Do NOT bolt the MicroScribe-3D down to an electrically conducting surface unless that surface is grounded and not subject to electrical interference or noise.

The threading of the mounting hole corresponds to standard tripod mounts, so you can also mount your MicroScribe-3D to a large tripod. Immersion Corporation sells a tripod unit for use with the MicroScribe-3D. Call for details.

Changing the Counterweights (optional)

Some MicroScribe units allow for the user to change the size of the counterweight. To do this, you should purchase a counterweight set from Immersion Corp. After adding or removing weights from the counterweight, be sure that the counterweight section still rests up against the stylus holder when the stylus is in the holder. This ensures proper calibration when the system is powered up. If the counterweight section does not press up against the stylus holder, you can manually push it against the stylus holder while you turn the power on to the device to ensure proper calibration.

Changing Stylus Tips (optional)

Should you need to replace the silver digitizing tip at the end of the stylus, simply remove it by unscrewing it counterclockwise. Only the silver portion of the tip can be removed. Be sure that when you replace the tip that it is tightened firmly clockwise.

Replacement tips and alternate tips are available directly from Immersion Corp. Please call for details.

IMPORTANT: Changing the stylus tip on your MicroScribe-3D to a non-standard tip may reduce its accuracy, especially if you do not carefully measure the dimensions of your replacement tip.

IMPORTANT: The MicroScribe expects the standard stylus tip to be in the stylus when it is powered up. If you want to change to a non-standard stylus tip, you must power up the MicroScribe with the original stylus tip in the stylus holder for at least 1 second, then change tips. If you power up the MicroScribe with a different tip than the standard tip, your MicroScribe will be out of calibration.

I f you change to a longer or shorter stylus tip, you can easily get the new X,Y,Z location and orientation of the new tip by doing the following:

1) Determine the difference in inches between your tip and the standard tip (if your tip is longer, the difference is positive and vice-versa).

2) Add or subtract that difference to the variable arm.D[5], which is nominally -5.285 inches (note the negative sign), but differs slightly from MicroScribe to MicroScribe

arm.D[5] = arm.D[5] + difference, if your stylus is shorter

arm.D[5] = arm.D[5] - difference, if your stylus is longer

Make this change to arm.D[5] somewhere in your program after calling the arm_connect() function, but before calling any of the reporting functions (e.g., arm_styluls_6DOF_update(), arm_stylus_6DOF_motion()). Now subsequent calls to reporting functions will calculate the X,Y,Z location and orientation of the new tip.

If your stylus has an offset from the centerline of the stylus (see Figure 3), you need to make similar changes to the variables arm.D[4] and arm.A[5]. The nominal values of arm.D[4] and arm.A[5] are 0.32 inches and 0.4 inches, respectively, but both variables differ slightly from MicroScribe to MicroScribe. Figure 3 shows the standard stylus tip, which is aligned with the stylus centerline. If your new stylus tip is offset to the right of the stylus centerline in Figure 3A, add that offset to arm.A[5]. If your new stylus tip is offset to the left of the stylus centerline in Figure 3A, subtract that offset from arm.A[5]. If your new stylus tip is offset to the right of the stylus centerline in Figure 3B, subtract

that offset to arm.D[4]. If your new stylus tip is offset to the left of the stylus centerline in Figure 3B, add that offset to arm.D[4].

FIGURE 3 MicroScribe-3D Stylus Offsets.

MicroScribe-3D Reference Coordinate System and Suggested Usage

Recommended Configuration for Digitizing with the MicroScribe-3D

Figure 4 shows the recommended setup for start-up and digitizing (see Figure 7 for a more detailed example). Note that the rear panel of the MicroScribe-3D faces away from the user and that the MicroScribe-3D is set up along the right side of the digitizing workspace. Placement can be completely arbitrary, since the MicroScribe can digitize anywhere within its 50" spherical reach. However, the suggested setup is ergonomically better for right-handed users. For left-handed users, place the MicroScribe-3D along the left side of the digitizing workspace. It is recommended that you first try a configuration like the one in Figure 4, and then customize the positioning of the MicroScribe-3D based on your experience and special needs. When possible, avoid working near full extension to ensure the maximum maneuverability and performance. Also avoid working too close to the MicroScribe-3D base to prevent mechanical obstruction when digitizing.

FIGURE 4 MicroScribe3-D Desktop Configuration and Suggested Workspace. We suggest that you work on one side of the unit, with the center of your workspace about halfway from full extension of the arm.

If you are using the MicroScribe-3D for an application which tracks motions of the arm to control a 3D position or cursor and you find that you are limited by the workspace size provided, you may wish to try one of the following two suggestions.

1. Scale the workspace by adjusting the scale factor between the physical motion of the MicroScribe-3D and its representation in the computer environment. Be careful not to scale too much, however, as small motions resulting from a less-than-rock-solid grip or vibrations can produce large motions in a highly scaled environment.

2. Index the workspace. This is an excellent application for a digital input device, such as a pedal. When you reach the limits of the workspace, you can press the pedal, return the Stylus Arm to a workable space, and then release the pedal, resuming the tracking action of the MicroScribe-3D from the point at which you left off earlier. This is similar to lifting a normal 2-dimensional mouse off the mouse pad and relocating it.

Euler and Fixed Reference Frames

The MicroScribe software reports the angles in three ways: xyz fixed, zyx fixed, and yxz fixed. The angle reporting scheme is set through a MicroScribe set-up call in the software library.

The Fixed reference frame indicated that all rotations are given with respect to fixed coordinate axes. The angles are reported in the order shown (x, then y, then z for a xyz fixed scheme). These angles are also referred to as roll, pitch, and yaw.

For the fixed reference frame, imagine the coordinate axes fixed to the MicroScribe base, with the origin placed at the initial home position. The fixed angles are reported in

the order requested. For a Fixed xyz angle request, the rotation around the x axis is given first, followed by the rotation around the original y axis and then around the original z axis.

Euler reference frames are also available. Euler reference frames are with respect to the object in question, in this case the MicroScribe stylus tip.

For the Euler reference frame, imagine the set of axes fixed to the MicroScribe stylus. For a Euler xyz reporting scheme, first the rotation around the Stylus' x axis is given, followed by the rotation around the new y axis, followed by the rotation around the new z axis. The new axes are found from the previous rotations, since as the rotations are performed, the axes, fixed to the stylus, rotate as well. Thus the key difference between Euler and Fixed reference frames is that in Euler, the axes shift with each angle given, and in Fixed, the axes stay fixed in space.

Euler and Fixed reference frames directly correspond if you reverse the order of angle reporting. For instance, xyz Euler is the same as zyx Fixed. Another example: yzx Fixed is the same as xzy Euler.

FIGURE 5 Inserting the stylus into the home position and twisting the final joint element. Twisting
the arm as shown may be necessary for the stylus to move easily into the rest position.

The Home Position

The MicroScribe-3D needs to be in the "home position" when it is powered up. The home position simply refers to the digitizer at rest with the stylus inserted into the resting hole on the MicroScribe-3D base. Figure 7 shows the proper way to place the arm in the home position. Note that both the stylus is fully inserted and the counterweight section of the arm is up against the stylus holder. If the stylus seems to hit a stop that prevents you from reaching the home position, the stylus must be untwisted as shown in Figure 5. If the arm still seems to resist going into the home position, either your unit is damaged or you are doing something incorrectly. Call Immersion Corporation for help.

If the bottom portion of the arm with the counterweight does not rest up against the stylus holder, you may hold it up against the stylus holder to keep it in the proper home position. Your MicroScribe-3D should naturally rest in this position, but adjustments to the counterweight or damage to the system might cause it to do otherwise. If this is the case, you may wish to contact Immersion Corp. for assistance.

FIGURE 6 MicroScribe-3D Home Position. Notice that the stylus is fully inserted and the counterweight section is held up against the stylus holder.

The Base Angle

As long as the stylus is resting in the holder when the system power is turned on, the MicroScribe-3D will work properly regardless of the orientation of the base joint (the base joint is free to rotate even with the stylus in the holder). However, it is recommended that the Microscribe-3D be powered up in the orientation shown in Figure 7. Notice that the base is facing the back of the desk as suggested earlier, but also that the arm itself is rotated to be in the plane of the screen. The stylus is to the user's right. When powered up in this orientation, a rightward motion of the digitizer produces a rightward

motion on the screen, and a leftward motion of the digitizer produces a leftward motion on the screen. This makes it more natural to use the MicroScribe-3D.

Powering Up and Digitizing

With the system unpacked, connected, and oriented properly, you are ready to activate the system and begin digitizing. For proper initial calibration, the MicroScribe-3D needs to be in the "home position," when the system power is turned on and the indicator light comes on. If the MicroScribe-3D is not in the home position when the power comes on, it will not know its configuration and will therefore provide wildly inaccurate data. If this happens, simply turn the power off, place the unit in the proper home position, and turn the power on again. The unit should then properly calibrate itself.

FIGURE 7 MicroScribe-3D Start-up Position. For right handed users, we recommend using the system as shown since the digitizer coordinate frame will match that of the screen.

CHAPTER 3 Programming the MicroScribe-3D in C

The Immersion MicroScribe-3D communicates with the host computer system over the serial port via a binary code. To insulate programmers from the drudgery of serial communication and binary decoding, Immersion Corp. provides the Immersion MicroScribe-3D Software Development Kit, or Immersion MicroScribe-3D SDK. The SDK allows C programmers to access the Immersion MicroScribe-3D using intuitive high-level function calls. Of course, for those who still want to program at the direct hardware level, Chapter 4 explains the necessary binary code in full detail with header files and technical documentation. This chapter explains all the high-level commands and variables built into the Immersion MicroScribe-3D SDK.

This chapter includes the following sections:

3.1, The Immersion HCI Controller, describes how the low-level communications hardware and software relates to applications that use a MicroScribe-3D.

3.2, Library Files, lists the C files that make up the SDK and briefly describes the types of functions included in each one.

3.3, Porting to Different Computing Platforms, briefly notes that all machine-specific C code is isolated in a relatively small driver file, for portability.

3.4, Strings vs. Enumerated Types, describes the SDK's implementation of error codes and other sets of arbitrary symbols.

3.5, Units and Computation, describes the length, angle, and ratio data types for computation of coordinates.

3.6 Start-up and Shut-down Sequences, describes how the communications hardware automatically adjusts to the host computer's baud rate when beginning a session.

3.7, Data Structure, describes in detail the arm_rec data structure used by the SDK.

3.8, Diagnostic Reports and Error Handlers, explains how the SDK handles errors and other conditions that may require output to the user.

3.9, Command Modes, explains how to use SDK commands in the foreground or background during other processing.

3.10, Library Reference, contains a full list of specific C functions and their uses.

3.11, HCI Data Structure, describes the low-level workings of the Software Development Kit.

3.12, HCI Low-Level Library Reference, contains a full list of specific C functions and their uses for the HCI low-level software.

3.1 The Immersion HCI Controller

The Immersion MicroScribe-3D is built around an electronic module called the Immersion HCI Controller (HCI stands for Human-Computer Interface). The Immersion HCI Controller, or HCI for short, is an embedded system that uses a microcontroller to read human input data (such as optical encoder signals) and report this data to the host computer. To communicate directly with the HCI electronics, one uses the HCI low-level software module; this module is a part of the Software Development Kit for any Immersion Corp. product that uses an HCI.

The Immersion MicroScribe-3D SDK is an abstraction built upon the HCI software module. The HCI module handles communication and access to directly-reported quantities, such as encoder count values or ON/OFF button status's, while the Arm module computes physical quantities and transformations, such as joint angles in degrees or radians and (X,Y,Z) coordinates. For most MicroScribe-3D programming needs, the programmer will not need to be familiar with the HCI module; the most commonly needed features are provided at a higher level. The HCI module provides access to the more advanced features which are less often used. The following section describes both the Immersion MicroScribe-3D SDK high-level functions and the underlying HCI functions.

3.2 Library Files

The Immersion MicroScribe-3D SDK exists in source code form for several different computing platforms. All platform-specfic code is isolated in a driver file that varies among different platforms, but all other source files are identical across all platforms. Compiled versions of the SDK are also available for some platforms.

Platform-independent files:

arm.c High-level functions for accessing the Arm.

arm.h Definitions for arm.c.

hci.c Low-level functions for direct access to the Immersion HCI Controller.

hci.h Definitions for hci.c.

drive.h Definitions for drivepc.c, drivemac.c, and driveunx.c

Platform-specific files:

drivepc.c Serial i/o and timing functions for PC-DOS

drivemac.c Serial i/o and timing functions for MAC

driveunx.c Serial i/o and timing functions for UNIX. Consult the files unxhelp and sgihelp on your SDK disk for additional information regarding serial i/o on a UNIX or SGI workstation.

drivewin.c Serial i/o and timing functions for PC-Windows. Consult the readme.win file on your SDK disk for information on Windows development.

Sample files:

Digitize.c - Sample program showing how to get points from the MicroScribe for digitizing applications. Prints MicroScribe coordinates to screen.

MSTestPC.c - Sample test and diagnostic program for the MicroScribe. Prints MicroScribe serial number, model number, firmware version, joint angles, xyz coordinates, and other information to screen.

armfgnd.c - Shows typical main loop using FOREGROUND update methods. Prints MicroScribe coordinates to screen.

armmotn.c - Shows typical main loop using MOTION SENSING update methods. Prints MicroScribe coordinates to screen.

armbgnd.c - Shows typical main loop using BACKGROUND update methods. Prints MicroScribe coordinates to screen.

Compiling and Linking:

To write your own Immersion MicroScribe-3D application,

¥ Write source code for use with the Immersion MicroScribe-3D SDK, using the library reference later in this chapter.

¥ Be sure to #include the files hci.h and arm.h in your source code. Hci.h must be included before arm.h.

¥ You will NOT need to #include any of the platform-specific files, unless you need direct access to your computer's timing functions or serial port. In most cases, this will not be necessary.

¥ Compile arm.c, hci.c, driveXXX.c and your own source code.

¥ Link the resulting object files together to produce an executable file.

Pre-compiled Libraries:

Depending on your computing platform, you may have received a pre-compiled library file. If so, its name will end with a ".lib" extension. This makes compiling and linking more convenient, because there is only one SDK file to link to.

¥ Write source code for use with the Immersion MicroScribe-3D SDK, using the library reference later in this chapter.

¥ Be sure to #include the files hci.h and arm.h in your source code. Hci.h must be included before arm.h.

¥ You will NOT need to #include any of the platform-specific files, unless you need direct access to your computer's timing functions or serial port. In most cases, this will not be necessary.

¥ Compile your source code, and link it with the pre-compiled SDK library. See your compiler's manual to learn its procedure for linking to pre-compiled libraries.

Source Code Example

The following is a simple example of source code that reads the x, y , z positions from the MicroScribe-3D and prints them to the screen. This example file and others demonstrating the background and motion-sensing modes can be found on the accompanying SDK disk.

```
/***********************************************
* - - - - - - - -   IMMERSION CORP.   - - - - - - - - *
 *                                                     *
 *       Platform-independent software series          *
 *                Copyright (c) 1993                   *
 ***********************************************
 * ARMFGND.C   |         SDK1-2 |   November 1995
 *
 * Immersion Corp. Software Developer's Kit 1-2
 *      Sample source code file prints stylus coordinates to the screen
 *      Requires HCI firmware version MSCR1-1C or later
 */

#include <stdio.h>
#include "hci.h"
#include "arm.h"

/* Declare an arm_rec to represent the 6DOF arm */
arm_rec arm;

void main(void)
{
        /* --- Declaration Stuff --- */
        /* Reasonable default values */
        int port = 1;
        long baud = 38400L;

        /* Flag to tell us when we're ready to exit */
        int done = 0;

        /* A result code that will let us monitor the success
         *  of each operation */
        arm_result result;

/* --- Initialization Stuff --- */

        /* Every arm_rec must be init'd one time at the very beginning */
        arm_init(&arm);

        /* (Optional) installs simple error handlers.  You can create
         *   your own error handlers and install them instead, but that
         *   is an 'advanced' feature.  If you leave this call out, things
         *   will still work, but any errors will cause the program to
         *   exit without giving you a chance to fix them and continue.
         *   THIS ERROR HANDLER USES PRINTF() TO WRITE
         *   TO THE SCREEN.  IT WILL ONLY WORK IN CONSOLE
        *    PROGRAMS!
         */
        arm_install_simple(&arm);

        /* Greet the user & get comm params */
        printf("\n\nImmersion 6DOF Arm sample program\n");
        printf("---------------------------------\n\n");
        printf("Which port is the interface connected to?  ");
        scanf("%d", &port);
        printf("What baud rate would you like to use?  ");
        scanf("%ld", &baud);
```

```
        printf("\nNote: if you chose an unavailable baud rate, we will\n");
        printf("use the closest available one.\n");


        /* Time to connect to the hardware */
        printf("Connecting on port %d ...\n", port);
result = arm_connect(&arm, port, baud);
        if (result == SUCCESS) printf("Connection established.\n");

        /* --- Real Stuff --- */
        /* Do this until there is an error or a button is pressed */
while ( (result == SUCCESS) && !done)
        {
                /* Update the stylus position */
                result = arm_stylus_3DOF_update(&arm);

                /* Print stylus coordinates */
                printf("X = %f, Y = %f, Z = %f\n",
                arm.stylus_tip.x, arm.stylus_tip.y,
                        arm.stylus_tip.z);

                /* If a button is being pressed, we want to exit */
                if (arm.hci.buttons) done = 1;
        }

        /* Time to end this session.  Another session can be begun with
         *   another call to arm_connect(), without manually resetting the
         *   hardware. */
        arm_disconnect(&arm);

        /* Print diagnostic info as we exit. */
        printf("Session ended on port %d at %ld baud\n",
                arm.hci.port_num, arm.hci.baud_rate);
        printf("Exit condition: %s\n", result);
}
```

3.3 Porting to Different Computing Platforms

The Immersion MicroScribe-3D SDK is very easy to adapt for customized hardware or to port to other computer systems. By using the appropriate machine-specific file, you should be able to use Immersion Corp. products with any computer that is capable of serial communication. As long as your new customized files adhere to the specifications set forth in the corresponding header files, the Immersion MicroScribe-3D SDK will function correctly when compiled. For your convenience, Immersion Corp. provides machine-specific files for use with some of today's most popular computers.

3.4 Strings vs. Enumerated Types

Any Immersion Corp. SDK contains some sets of constants that would traditionally be declared as enumerated types. For instance, error codes are often defined as arbitrary integers, each representing some error condition (0 = success, 1 = bad port number, etc.). Rather than assigning an arbitrary value to each error condition, an Immersion Corp. SDK assigns descriptive strings:

```
typedef char* arm_result;

char SUCCESS[8] = "Success";

char BAD_PORT_NUM[25] = "Port Number Out of Range";
```

In this example, the resulting labels SUCCESS and BAD_PORT_NUM are pointers to descriptive strings and can be compared to variables of type arm_result. They can be used in if() statements just as if they were arbitrary enumerated labels, but they can also be printed directly, since they are actually pointers to strings:

```
/* Example function: update arm_rec data and report any

 * errors that occur */



arm_result update_and_report(arm_rec *arm)

{

/* Update MicroScribe-3D data and save the result code */

arm_result result = arm_full_update(arm);



/* Print message, but don't bother unless there is an

 * error to report */

if (result != SUCCESS) printf("%s", result);



/* Return the result code for use by other functions

 */

return result;

}
```

3.5 Units and Computation

The quantities of interest to an Immersion MicroScribe-3D programmer are angles and lengths. However, the host computer must use trigonometric functions to calculate anything beyond the six Immersion MicroScribe-3D joint angles, and to calculate the three-dimensional orientation of the stylus, the host must evaluate inverse trigonometric functions along with a large number of multiply operations. Some computer systems cannot do the above operations very quickly using standard floating point variables. For this reason, the Immersion MicroScribe-3D SDK defines the data types length, angle, and ratio. The ratios are used for unitless quantities such as sines and cosines. The lengths are used for all spatial coordinates, and angles are used for representing the joint angles and the stylus' three-dimensional orientation. Currently, all of these types are simply defined as float variables. Later versions of the Immersion MicroScribe-3D MicroScribe-3D SDK will implement them as fixed-point data types and may contain additional modifications. All programmers of the Immersion MicroScribe-3D should take care to treat these quantities syntactically and functionally as the types length, ratio, and angle and not as their underlying data types. In this way, future releases of the Immersion MicroScribe-3D SDK will be compatible with previously-written software.

3.6 Start-Up and Shut-Down Sequences

The Immersion HCI hardware has an initialization procedure that adapts to the host computer's baud rate. Programmers need only call arm_connect() with the desired port number and baud rate; manual hardware settings are not required. If the HCI hardware is turned on and connected to that port, then it will respond and "sign on" to the host computer, even if it was previously being used at a different baud rate by a different program. When the function arm_connect() sees the HCI hardware "sign on," it re-

quests some basic information concerning the MicroScribe-3D, stores it, and returns. The system is then ready for angle and coordinate reporting.

The "basic information" that the host requests from the MicroScribe-3D includes everything from a serial number to the lengths of the Arm's linkages. These constants are immediately stored for future reference and computation.

When the host program is ready to terminate, it needs to call arm_disconnect(). This will close the HCI's serial port and inform it that it is on its own for a while. The HCI hardware will then simply wait until it sees some more activity on the serial port (via another arm_connect() call), and then the start-up procedure will happen all over again. In theory, the HCI hardware need never be switched off, as long as every arm_connect() call is balanced by an arm_disconnect(). In many cases, unbalanced calls will be forgiven. The arm_connect() function will actually run arm_disconnect() if it has trouble connecting. This will recover from situations in which a previous session was not ended properly, IF that session was at the same baud rate as the current new session.

If you need to know the exact binary communications sequence used to establish a connection to an Immersion MicroScribe-3D, see Chapter 4.

3.7 Data Structure

arm.h contains a definition for a struct called an arm_rec. An arm_rec contains all the information there is to know about a single Immersion MicroScribe-3D. No arm_rec variables are allocated by the SDK; the user should declare one variable of type arm_rec for each MicroScribe-3D in use. For example, if an Immersion MicroScribe-3D is connected to serial port #1 while another is connected to port #2, then the user would declare two variables of type arm_rec, perhaps scribe1 and scribe2. Almost all functions in this SDK operate on an arm_rec variable.

All fields in an arm_rec variable are for output only. That is, the programmer may freely examine them but should NEVER directly set their values. Function calls are provided for manipulation of all fields in the proper manner.

Underlying HCI data:

hci_rec hci;

The arm_rec structure has a sub-structure called an hci_rec which contains information read directly from the HCI hardware. This includes raw encoder counts, values of analog controller inputs, button states (ON or OFF), as well as communications parameters. The only common reason to access the hci_rec is to read the states of the buttons. A single byte representing all buttons can be accessed at arm.hci.buttons (assuming arm is an arm_rec). Each bit corresponds to one of seven remote digital buttons that may be connected to the Immersion MicroScribe-3D. A bit value of zero (OFF) indicates that the corresponding button is open (not pressed, or inactive). The current assignments are

Bit 0: Rear button jack: standard single pedal or right pedal of double pedal

Bit 1: Rear button jack: left pedal of double pedal

Bit 2: Future Expansion / customization

Bit 3: Future Expansion / customization

Bit 4: Future Expansion / customization

Bit 5: Future Expansion / customization

Bit 6: Future Expansion / customization

The buttons are also decoded separately in the array arm.hci.button[] for access to individual button states. Therefore, arm.hci.button[0] has the state of the standard single pedal or the right pedal of the standard double pedal and arm.hci.but-

ton[1] has the state of the left pedal of the double pedal.

Final calculated quantities:

```
length_3D       stylus_tip;

angle_3D        stylus_dir;

angle   joint_rad [ NUM_ENCODERS ];

angle   joint_deg [ NUM_ENCODERS ];

matrix_4        T;
```

The length_3D type is merely a struct with three fields called x, y, and z, all of type length. The angle_3D type is similar, but it is for angles rather than lengths. The matrix_4 type is a 4 by 4 array of float variables.

The stylus_tip field holds Cartesian coordinates for the position of the stylus' tip in inches or millimeters. The origin is at the base of the very first MicroScribe-3D linkage. See Figures 2-2, 2-3, and 2-4 for representation of the x, y, and z axis directions relative to the MicroScribe-3D.

The stylus_dir field represents the stylus' three-dimensional angles in radians or degrees. Before making any _update() function call, the programmer can specify what format should be used. The function arm_angle_format() takes an argument that specifies the format for all future stylus direction calculations. Legal values are: XYZ_FIXED, ZYX_FIXED, YXZ_FIXED, ZYX_EULER, XYZ_EULER, and ZXY_-
EULER. The x, y, and z sub-fields of this stylus_dir field always denote right-handed rotations about their respective axes, starting from the negative z direction. The arm_angle_format() call simply affects the order in which axial rotations are to be performed conceptually when interpreting the angles in the stylus_dir field.

The joint_rad[] and joint_deg[] fields hold the joint angles in radians and de-grees, respectively.

The matrix T is a 4 by 4 transformation matrix representing the position and orientation of the stylus. It contains no additional information to that provided by the fields sty-lus_tip and stylus_dir, but it is a more convenient form for software environ-ments that natively support matrix computation. The upper-left element of T is T[0][0], and the upper-right element is T[0][3].

Intermediate calculated quantities:

```
ratio   cs[NUM_JOINTS];

ratio   sn[NUM_JOINTS];
```

The cs[] and sn[] fields are cosines and sines of the six joint angles, respectively, They are pre-calculated and stored here whenever the host begins a calculation of coor-dinates or orientation.

Units and data format specifiers:

```
length_units    len_units;

angle_units     ang_units;

angle_format    ang_format;
```

The data type length_units is an enumerated string type which can equal INCHES or MM. The field len_units tells the programmer what units all coordinate fields are reported in. To change the reported units, call arm_length_units(). The an-g_units field is very similar; it indicates DEGREES or RADIANS, but it only applies to the stylus_dir field. All other arm_rec angles are stored as radians, because all internal math functions use radians. The values of ang_units and ang_format

should be set only by calling arm_ang_units() and arm_ang_format(). The possible values of ang_format are: XYZ_FIXED, ZYX_FIXED, YXZ_FIXED, ZYX_EULER, XYZ_EULER, and ZXY_EULER.

3.8 Diagnostic Reports and Error Handlers

People use Immersion Corp. products with a wide variety of user-software interfaces, and therefore an Immersion Corp. SDK does not attempt to directly output any diagnostic messages. Instead, when an error occurs or some progress is to be reported, the SDK executes a handler supplied by the programmer. If the programmer has not supplied any such handler, then this feature is simply skipped. New SDK programmers need not be familiar with this mechanism immediately; a set of simple call-back handlers is provided for those who are just getting started.

For each possible error condition or reporting situation, there is a function pointer which the user sets to point to a handler which is to be executed in response. For instance, in a window-based system, if the host computer is unable to open the serial port, the corresponding error handler might open a window displaying the text "Unable to open serial port" with an OK button for the user to acknowledge. Alternatively, this call-back function might first try to close the port and re-open it before alerting the user that there is a problem. Each arm_rec has its own private set of these function pointers.

To get started with simple error handlers, the file arm.c contains a generic set of handlers for simple command-line output using the standard printf()function. The function arm_install_simple() will install all of these simple handlers for any given arm_rec, and you can use this to obtain rudimentary output with minimal effort.

Here is how to declare an error handler:

```
arm_result my_handler(arm_rec *arm,             arm_result condition);
```

The parameter arm points to the data record for the particular MicroScribe-3D that is currently being processed. The parameter condition is the result code of the error or condition that has prompted the execution of this handler. For example, if the user attempts to open a non-existent serial port, then a handler will be executed with its condition parameter equal to BAD_PORT_NUM. Note that because the condition parameter is passed to all handlers, a single handler can handle several different error conditions.

To install a handler to execute on a particular error or condition, you need to identify the function pointer that corresponds to this condition and set it to point to your handler. The handler function pointers are fields of an hci_rec, which in turn is a low-level field of an arm_rec. The handler function pointers are:

```
arm_result (*NO_HCI_handler)(arm_rec *arm,

        arm_result condition);
```

NO_HCI_handler executes when arm_connect() times out while waiting for the HCI hardware to echo "IMMC" during the start-up sequence. This usually occurs when the hardware is turned off or is not connected to the proper port. See Chapter 4 for details on the start-up sequence.

```
arm_result (*CANT_BEGIN_handler)(arm_rec *arm,

        arm_result condition);
```

CANT_BEGIN_handler executes if arm_connect() does not receive a null-terminated product ID string from the HCI hardware after successfully receiving the "IMMC" signon string. See Chapter 4 for details on the start-up sequence.

```
arm_result (*CANT_OPEN_PORT_handler)(arm_rec *arm,
```

```
        arm_result condition);
```

CANT_OPEN_PORT_handler executes if the host tries unsuccessfully to open a serial port.

```
arm_result (*TIMED_OUT_handler)(arm_rec *arm,

        arm_result condition);
```

TIMED_OUT_handler executes when any function other than arm_connect() times out while trying to access the HCI hardware. This occurs if the baud rate is too high for the host system or if the hardware is turned off by mistake.

```
arm_result (*BAD_PORT_NUM_handler)(arm_rec *arm,

        arm_result condition);
```

BAD_PORT_NUM_handler executes when any function is asked to access an invalid port.

```
arm_result (*BAD_PACKET_handler)(arm_rec *arm,

        arm_result condition);
```

BAD_PACKET_handler executes when an incoming packet's first bit is not set. This indicates a communications error, possibly caused by a noisy environment or a bad connection. See Chapter 4 for more information on packets and the packet marker bit.

The following example shows how to install a handler function as the BAD_PORT_NUM_handler.

```
/* Define the handler function */

arm_result (*my_handler)(arm_rec *arm,

        arm_result condition)

{
```

CHAPTER 4 Hardware and Low-level Reference

The following sections contain detailed information on the hardware and software used in the MicroScribe-3D products.

Contained in this section:

- MICROSCRIBE-3D CONNECTION PIN ASSIGNMENTS
- MICROSCRIBE-3D SPECIFICATIONS
- SGI SERIAL ADDENDUM
- OPERATIONAL OVERVIEW
- RESPONSE SPECIFICATION
- INITIALIZATION SEQUENCE
- COMMAND SPECIFICATION
- CONFIGURATION COMMANDS

MICROSCRIBE-3D CONNECTION PIN ASSIGNMENTS

In this section, all the pin assignments for external connectors on the Immersion Micro-Scribe-3D are shown.

¥ WARNINGS:

Be sure to use only the appropriate connector with each port. Any incorrect device attached or improper use of external connections voids your warranty.

Never make connections while the MicroScribe-3D unit's power is on.

The following connectors are found on the rear panel of the MicroScribe-3D:

¥ Serial Port

¥ Power Port

¥ Digital Pedal or Accessory Port

SERIAL PORT

¥ Connects the Electronics Module to the Host Computer

¥ Mini-DIN 8 type connector

¥ Serial cables for PC-AT (DB-9) and Apple Macintosh (Mini-DIN 8) are available from Immersion Corp.

Pin Number	Signal Name	Signal Description
1	NC	Not connected
2	NC	Not Connected
3	Tx	Transmit Data
4	GND	Electrical Ground
5	Rx	Receive Data
6	NC	Not connected
7	NC	Not connected
8	NC	Not connected

POWER PORT

¥ Use only an Immersion Corp. supplied power supply or a +5V DC, 500 mA third party power supply with standard 2.1 mm DC-coax type plug, center tip positive.

Immersion Corporation has power supplies for international use. Please call for details. Immersion Corp. is not responsible for any damage resulting from the use of a non-Immersion Corp. authorized power supply.

Immersion Corporation Authorized Power Supplies include the following:

¥ CPS5-120 , Standard North American Power Supply

¥ CPS5-220-EURO, 220V, 50Hz Power Supply with European Plug

¥ CPS5-220-IEC, same as above but with IEC receptacle

¥ CPS5-U, Universal Input Power Supply (Requires IEC Cord)

DIGITAL PEDAL PORTS

¥ Mini-DIN 6 type connector

¥ Use only with Immersion Corp. approved pedal/digital input device, including but not limited to CPED-D-STD, CPED-D-STD2, CPED-D-HD, CPED-D-HD2

¥ Not compatible with Immersion Interface Boxª and Immersion Probeª and Immersion Personal Digitizerª digital peripherals. However, these peripherals can be purchased with the Mini-DIN 6 type connector from Immersion Corp. for an additional charge. Adaptors are available to let you use the Mini-DIN6 style peripherals with the Interface Box, Personal Digitizer, and Probe products.

¥ Digital bits 2 and 3 are reserved for MicroScribe-3D accessories like the rotary table. Digital bits 0 and 1 can be used for any digital input device. The standard Immersion pedal connects to bit 0 and is normally low (depressing the pedal changes bit 0 to hi). On the Immersion double pedal, the right pedal connects to bit 0, and the left pedal connects to bit 1. Both pedals are normally low.

Pin Number	Signal Name	Signal Description
1	GND	Electrical Ground
2	+5V	VCC
3	D3	Bit 3
4	D0	Bit 0
5	D2	Bit 2
6	D1	Bit 1

MICROSCRIBE-3D SPECIFICATIONS

Position Resolution: 0.005" (0.13 mm)

Position Accuracy: 0.015" (0.38 mm) MicroScribe-3D

0.009" (0.23 mm) MicroScribe-3DX

0.025" (0.64 mm) MicroScribe-3DL

Workspace Size: Models 3D and 3DX: 50" diameter sphere

Model 3DL: 66" diameter sphere

Footprint Size: 6" diameter circle

Interface: RS-232C

Baud Rate: Up to 115 Kbps

Latency: 1.0 ms

Processor: MC68HC11

Clock Frequency: 1.8 MHz

Software Version: ROM 1.0, 1.1

Button Options: Foot-pedal, desktop unit, and hand-held units available.

Power Requirements: External power supply

Uses +5V DC, 600 mA max.

All specifications subject to change by Immersion Corp.

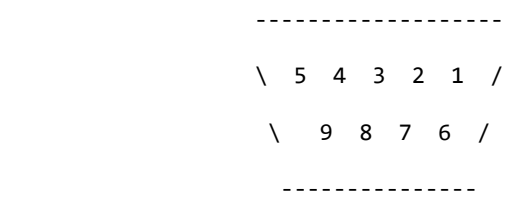
SGI Serial Addendum

The following information is provided as a service to SGI users who may need additional information to connect the Electronics Module to their computer via a serial cable.

All IRIS-4D Series machines have two or more general purpose serial ports. These ports may be used to connect terminals, printers, modems, other machines, or graphical input devices such as a tablet or dial and button box. Each line may be independently set to run at any of several speeds, as high as 19,200 or even 38,400 bps. Various character echoing and interpreting parameters can also be set. See stty(1) and termio(7) for details on the various modes.

Special files for the serial ports exist in the /dev directory. These files, tty[dfm][1-56] are created automatically by MAKEDEV(1M) when system software is installed. Each port is referred to by three different names, /dev/ttydnn, /dev/ttymnn, and /dev/ttyfnn, where nn represents the port number. Opening the ttyd, ttym, or ttyf versions of a port enables different signals and modes on the communication line. Typically, the ttyd version is used for directly connecting simple devices including most terminals; ttym is used for devices that use modem control signals; and ttyf is used for devices that understand hardware flow control signals.

There are two different types of connectors found on various 4D models. The DB-9 serial port connectors have the following pin assignments.



Pin	Name	Description
2	TD	Transmit Data

| 3 | RD | Receive Data |

| 4 | RTS | Request To Send |

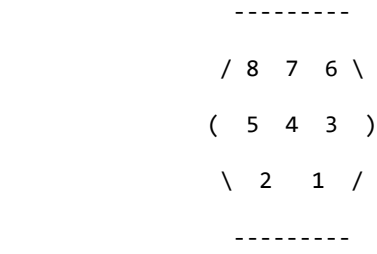
| 5 | CTS | Clear To Send |

| 7 | SG | Signal Ground |

| 8 | DCD | Data Carrier Detect |

| 9 | DTR | Data Terminal Ready |

The DIN-8 serial port connectors on the Personal Iris 4D/30, 4D/35, 4D/RPC, and 4D/RPC-50 have the following pin assignments.



4D_Compatible_Pin_Assignments_(RS-232)		
Pin	Name	Description
1	DTR	Data Terminal Ready
2	CTS	Clear To Send
3	TD	Transmit Data
4	SG	Signal Ground
5	RD	Receive Data

6	RTS	Request To Send
7	DCD	Data Carrier Detect
8	SG	Signal Ground

Macintosh SE Compatible Pin Assignments (RS-422)

Pin	Name	Description
1	HSKo	Output Handshake
2	HSKi	Input Handshake Or External Clock
3	TxD-	Transmit Data -
4	GND	Signal Ground
5	RxD-	Receive Data -
6	TxD+	Transmit Data +
7	GPi	General Purpose Input
8	RxD+	Receive Data +

The set of signals that are actually used depends upon which form of the device was opened. If the ttyd name was used, then only TD, RD, and SG signals are meaningful. These three signals are typically used with "dumb" devices that either do not need any sort of data flow control or use software flow control (see the description of the ixon, ixany, and ixoff options in stty(1) for more information on setting up software flow control). If the ttym device is used, then the DCD, and DTR signals are also used. These signals provide a two way handshake for establishing and breaking a communication link with another device and are normally used when connecting via a modem. When the port is initially opened, the host asserts the DTR line and waits for the DCD line to become active. If the port is opened with the O_NDELAY flag, then the open will succeed even if the DCD line is not active. A hangup condition will occur if the DCD line transitions from active to inactive. See open(2), and termio(7) for more information. If the ttyf device is used, then all of the signals are used. The additional signals provide for full hardware flow control between the host and the remote device. The RTS line is asserted by the host whenever it is capable of receiving more data. The CTS line is sampled before data is transmitted and if it is not active, the host will suspend output until it is.

The DIN-8 serial port connectors on the Personal Iris 4D/30, 4D/35, 4D/RPC, and 4D/RPC-50 can be used to communicate with serial devices using RS-422 protocol. User can use the stream ioctl commands, SIOC_EXTCLK and SIOC_RS422, defined in /usr/include/sys/z8530.h to switch between internal/external clock and RS-232/RS-422 protocols. Another command that may be useful is SIOC_ITIMER, it informs the driver how long it should buffer up input data before sending them upstream.

RESPONSE SPECIFICATION

The Immersion MicroScribe-3D responds to each command from the host computer by sending back a packet. For "Motion-generated" commands, packets are repeatedly sent until a new command is received from the host. Every packet begins with a byte that specifies what type(s) of information are included in the rest of the packet. The rest of the packet consists of several data fields, each of which may or may not be included in

any particular packet. Those which are included are sent in the order that they are listed here. All multi-byte values are sent MSB first. All diagrams of bit fields below show the MSB to the left.

In all packets, the MSB of the first byte is always set. No other packet bytes contain a high MSB, with the exception of some "special configuration commands" which are only rarely executed. This makes it easy to identify the beginning of a packet during normal operation and allows the host to recover from transmission errors.

Packet Header Byte:

1 bit

7 bits

Byte 0:

1

Command Bits

The packet's first byte is a copy of the host computer's Command Byte to which this packet is responding. Bits 0-5 specify which data fields are included in the packet. Bit 6 indicates whether or not this packet is a report of special configuration information. Bit 7 is always on, and is the ONLY occurrence of a high MSB in a "normal" packet. The Command Byte is discussed in Command Specification.

Button Status Byte:

1 bit

7 bits

Byte 1:

0

Buttons

Buttons contains one bit for each of seven remote digital buttons that may be connected to the Immersion MicroScribe-3D. A bit value of zero (OFF) indicates that the corresponding button is open (not pressed, or inactive). The current assignments are

Bit 0: Rear button jack: standard single pedal or right pedal of double pedal

Bit 1: Rear button jack: left pedal of double pedal

Bit 2: Future Expansion / customization

Bit 3: Future Expansion / customization

Bit 4: Future Expansion / customization

Bit 5: Future Expansion / customization

Bit 6: Future Expansion / customization

Packet Data Fields:

Of the data fields listed below, those that are present (as indicated by Command Bits) are sent in the order in which they are listed. All data is sent in 7-bit fields, so that no data byte has a high MSB. Angles are sent as unsigned 14-bit integers that are scaled from zero to some maximum value which depends on each angle's encoder resolution. If an angle "wraps around," its encoder count continues to increase. This provides the angle's "winding number" as well as its angular value. See Configuration Commands for details on determining each angle's range.

1 bit

7 bits

1 bit

7 bits

Timestamp

0

Timer highest 7 bits

0

Timer lowest 7 bits

The 14-bit Timestamp is included immediately after the Button Status Byte if Bit 5 of Command Bits is ON.

Next, analog controllers are provided for custom systems only. Bits 2 and 3 of Command Bits specify how many controllers (A/D conversions from pedals, knobs, or similar controls) are included in the packet. If your Immersion Microscribe-3D system is not customized to support analog controllers, then any reported controller values will be meaningless "stray" readings. Each controller has an 8-bit value but is reported in a 7-bit field so as to preserve the uniqueness of the packet marker bit. The eighth bit (least significant bit) of each controller is packed into an extra byte that follows the controller values in the packet. This extra byte also has only seven data bits; the eighth bit of the last controller (controller #7) is not included. The table below shows which controllers are included in the packet for each of the four possible values of bits 2 and 3.

Bit 3	Bit 2	Controllers included
0	0	none
0	1	Controllers 0 and 1
1	0	Controllers 0, 1, 2, and 3
1	1	Controllers 0, 1, 2, 3, 4, 5, 6, and 7

1 bit

7 bits

Each Controller

0

Highest 7 Controller bits

1 bit

7 bits

Extra Controller Bits

0

LSB's of contr's 0 - 6

The extra bits are ordered from left to right in the same fashion as the controller bytes they correspond to. That is, the LSB of controller #0 is the leftmost of the extra bits, and the LSB of controller #6 is the rightmost of the extra bits. The "extra controller bits" byte is reported if and only if at least one controller value is reported.

Next, joint angles are reported. Bits 0 and 1 of Command Bits specify how many joint angles are included in the packet. The table below shows which controllers are included in the packet for each of the four possible values of bits 2 and 3. Angle 5 corresponds to stylus roll on the Immersion MicroScribe-3D and is currently unused. Requests for Angle 5 will return a constant value. Angle 6 is for the rotary table, which will be available soon.

Bit 1	Bit 0	Controllers included
0	0	none
0	1	Angles 0, 1, 2, 3 and 4
1	0	Angles 0, 1, 2, 3, 4, 5, and 6
1	1	Angles 0, 1, 2, 3, 4, and 5

1 bit

7 bits

1 bit

7 bits

Each Angle

0

Angle highest 7 bits

0

Angle lowest 7 bits

Exceptions for Special Configuration Commands:

If bit 6 of Command Bits is ON, then the packet is a response to a special configuration command. Bits 0-5 then do not have the meanings assumed above. Instead, they designate a configuration command, and the data in the packet will describe such things as baud rate, physical parameters, and angle normalizations. This set of exceptions is described in Configuration Commands.

INITIALIZATION SEQUENCE

When the Immersion MicroScribe-3D is first turned on, it initializes its angle registers and determines the host computer's baud rate. The host is then free to execute any command. Generally, the host will begin by executing several special configuration commands in order to obtain the physical parameters, firmware version number, or other global settings. Then the host will enter "normal operation" in which it monitors the joint angles and/or button states versus time.

Initialize Angle Registers:

On power-up, the Immersion MicroScribe-3D is assumed to be in its "home" position, with the stylus resting vertically on the base. The angle registers are pre-loaded with values corresponding to this position. The MicroScribe-3D can be re-calibrated to this position later by executing the Home command as described in Configuration Commands.

Synchronize to Host Baud Rate:

After pre-loading its angle registers, the Immersion MicroScribe-3D identifies the host's baud rate through trial and error. The host must repeatedly send the ASCII string "IMMC". The Electronics Module will attempt to receive this string at various baud rates until it is successful. The baud rates it will attempt are: 9600, 14400, 19200, 28800, 38400, 57600, 115200. When it has succeeded, it will echo back "IMMC", and the host computer must cease the repeated message. Other Immersion Corp. products also use this start-up synchronization technique.

Begin Session:

When the baud rates have been synchronized, the host must transmit "BEGIN" one time. Then the Microscribe-3D will identify itself by sending "MSCR" as a null-terminated string, and normal operation will commence.Other Immersion Corp. products use different identifier strings, so the host should check this string to ensure that it has found the correct product. Programs which use several Immersion Corp. products on different serial ports need not be told which port is which; the host will find out upon beginning the session with each device. Note that the identifier string is null-terminated, while the other strings mentioned in this section are not. This enables the host to deal with different products having identifier strings of different lengths.

Recommended Initial Commands:

At this point, the system is in normal operation as described in Command Specification. It is recommended that the host perform the following Special Configuration Commands next. The Software Developer's Kit already includes these commands in its arm_connect() function. To read in detail about these commands and the packets they generate, see Configuration Commands.

Get Firmware Version. This should be done immediately to check compatibility between the host software and the Immersion Microscribe-3D firmware.

Get Parameter Format. Before asking the Immersion MicroScribe-3D to report is physical parameters, the host needs to know what format the parameters will be sent in. Since this format may change or be elaborated upon with future Immersion MicroScribe-3D releases, the host can obtain a format string specifying which format is used by this particular Immersion MicroScribe-3D system.

Get Physical Parameters. The host needs to know all linkage lengths, angle normalization factors, and other physical parameters in order to calculate the stylus coordinates.

Get Extended Physical Parameters. The host may need to know additional physical parameters in order to calculate the stylus coordinates. The comment string should be checked (Get Comment) to determine if Get Extended Physical Parameters is necessary. If the comment string is "Standard", this command is not needed. If the comment string is "Standard+Beta", this command is required. Refer to the function arm_get_constants() in arm.c for an example of how to implement this check.

Get Maximum Field Values. This will tell the host how each reported data field is scaled. For angle fields, this "maximum" is actually the number of counts in one physical revolution. Higher values can be encountered if the angle "wraps around."

Get Home Reference Offsets. This information helps the host to resolve ambiguity in the Immersion MicroScribe-3D "home" position.

Normal Operation:

When the host has finished the above steps, it will be able to properly interpret time-varying data from the Immersion Microscribe-3D system. Refer to Immersion Microscribe-3D Command Specification for details on the commands available for reading the Microscribe-3D's time-varying quantities.

End Session:

Corresponding to the "BEGIN" command, there is an "END" command that will put the Immersion Microscribe-3D back into the baud rate identification phase. At any time after the "BEGIN" string has been sent, the host can send the string "E", or "END", to end the current session. The Immersion Microscribe-3D will then return to its initial power-up state, awaiting the string "IMMC" to synchronize for the next session.

COMMAND SPECIFICATION

Each command from the host computer consists of a single command byte followed by zero or more argument bytes. Only Special Configuration Commands require arguments, and those commands are described in Configuration Commands. The command byte is composed of several independent bit fields, each of which specifies whether or not certain Microscribe-3D quantities are being requested.

Command Bit Fields:

¥ Bit 7: Unused; reserved for use as Packet Marker

This bit does not affect the meaning of the command. When the command byte is echoed by the Immersion MicroScribe-3D, its bit 7 will always be turned ON. See

Packet Specification for details on response packets.

¥ Bit 6: Special Configuration Flag

If bit 6 is ON, then the command is a Special Configuration Command, and the remaining bit fields discussed below do NOT take on the interpretations given here. Special Configuration Commands may require arguments. Refer to Configuration Commands for details on the available configuration commands. As long as bit 6 is OFF, bits 0 through 5 function as described below, and the command requires no arguments.

¥ Bit 5: Timestamp Flag

The Immersion Microscribe-3D contains a running 14-bit counter which can be referenced for accurate digitization of stylus position versus time. The counter has a resolution of 1.111 ms and wraps around approximately every 18 seconds. Bit 5 controls whether or not this timing data will be reported along with other data from the Immersion Microscribe-3D. Turning off the timestamp feature decreases the amount of data that the Immersion Microscribe-3D must send and therefore increases its reporting speed.

¥ Bit 4: Future Expansion

¥ Bits 3 and 2: Analog Controller Flags

The Immersion Microscribe-3D supports up to eight analog controller inputs for custom configurations. If your Immersion Microscribe-3D has not been customized to read analog controller inputs, then these bits of the command byte will not be useful. If your Immersion Microscribe-3D does read analog controller inputs, then these bits specify which controllers' values will be reported, as given in the following table.

Bit 3	Bit 2	Controllers Reported
0	0	none
0	1	Controllers 0 and 1
1	0	Controllers 0, 1, 2, and 3
1	1	Controllers 0, 1, 2, 3, 4, 5, 6, and 7

¥ Bits 1 and 0: Joint Angle Flags

Bits 0 and 1 of Command Bits specify how many joint angles are included in the packet. The table below shows which controllers are included in the packet for each of the four possible values of bits 2 and 3. Angle 5 corresponds to stylus roll on the Immersion MicroScribe-3D and is currently unused. Requests for Angle 5 will return a constant value. Angle 6 is for the rotary table, which will be available soon.

Bit 1	Bit 0	Controllers included

0	0	none
0	1	Angles 0, 1, 2, 3 and 4
1	0	Angles 0, 1, 2, 3, 4, 5, and 6
1	1	Angles 0, 1, 2, 3, 4, and 5

Example A:

Bit #: 7 0

Value: 1 0 1 0 0 0 1 0

(HEX A2)

Meaning: Give a single time-stamped report of Angles 0, 1, 2, 3, 4, 5, and 6. Do not report any controller values.

Example B:

Bit #: 7 0

Value: 0 0 0 0 1 1 1 1

(HEX 0F)

Meaning: Give a single report of Angles 0, 1, 2, 3, 4, and 5 and all 8 controllers, with no time stamp.

CONFIGURATION COMMANDS

When the Immersion MicroScribe-3D receives a Command Byte which has Bit 6 turned ON, it interprets Bits 0-5 as a special configuration code rather than making the usual bit-field interpretations. This makes some special functions accessible to the host computer. The Immersion MicroScribe-3D acknowledges all of these commands by echoing the Command Byte followed by any requested data. Data reported in response to a Special Configuration Command is sent as a series of standard 8-bit binary numbers, not in the 7-bit fields used by non-configuration packets. Each Special Configuration Command is described below, along with the hexadecimal Command Byte that represents it. Note that each command corresponds to two hexadecimal codes, since bit 7 does not affect the meaning of a command byte.

Get Physical Parameters: Hex 40 or C0

The Get Physical Parameters command causes the Microscribe-3D to report all physical parameters (link lengths, linear and angular offsets) necessary to calculate stylus coordinates from the five joint angles. These parameters are currently stored in firmware in a Denavit & Hartenberg (reported by firmware as "Format DH0.5"). See the "Get Parameter Format" Command later in this section for information on determining the parameter format in use. The response packet for a Get Physical Parameters command, in Format DH0.5, is as follows:

Byte #	Meaning
0	Command Byte C0
1	Number of bytes to follow in this parameter set (36)
2-3	ALPHA 0, 16-bit signed integer, -180 deg is -32768

4-5 ALPHA 1, 16-bit signed integer, -180 deg is -32768

6-7 ALPHA 2, 16-bit signed integer, -180 deg is -32768

8-9 ALPHA 3, 16-bit signed integer, -180 deg is -32768

10-11 ALPHA 4, 16-bit signed integer, -180 deg is -32768

12-13 ALPHA 5, 16-bit signed integer, -180 deg is -32768

14-15 A 0, 16-bit signed integer in thousandths of an inch

16-17 A 1, 16-bit signed integer in thousandths of an inch

18-19 A 2, 16-bit signed integer in thousandths of an inch

20-21 A 3, 16-bit signed integer in thousandths of an inch

22-23 A 4, 16-bit signed integer in thousandths of an inch

24-25 A 5, 16-bit signed integer in thousandths of an inch

26-27 D 0, 16-bit signed integer in thousandths of an inch

28-29 D 1, 16-bit signed integer in thousandths of an inch

30-31 D 2, 16-bit signed integer in thousandths of an inch

32-33 D 3, 16-bit signed integer in thousandths of an inch

34-35 D 4, 16-bit signed integer in thousandths of an inch

36-37 D 5, 16-bit signed integer in thousandths of an inch

Get Home Reference Offsets: Hex 41 or C1

The Get Home Reference Offsets command is not relevant to MicroScribe-3D users. The command is used by other Immersion 6-DOF devices.

Home: Hex 42 or C2

The Home command restores the Immersion MicroScribe-3D's angle registers to their power-up values. This command must only be given when it is known that the Immersion MicroScribe-3D is in its resting position, with the stylus on its peg. The response packet for a Home command contains only the echoed command byte (hex C2).

Set Baud Rate: Hex 44 or C4

The Set Baud Rate command sets the Immersion MicroScribe-3D's baud rate to one of 26 possible values. The value is determined by 5 bits of a single-byte argument that follows the Set Baud Rate command byte. The response packet for a Set Baud Rate command contains only the echoed command byte (hex C4), and it is echoed at the OLD baud rate. The following table illustrates how these five bits determine the new baud rate:

Argument

00**

01**

10**

11**

*****000

115,200

38,400

28,800

8862

*****001

57,600

19,200

14,400

4431

*****010

28,800

9600

7200

2215

*****011

14,400

4800

3600

1108

*****100

7200

2400

1800

554

*****101

3600

1200

900

277

*****110

1800

600

450

138

*****111

900

300

225

69

End Session: Hex 45 or C5 (note 45 is ASCII for ÔE')

The End command is issued by sending the ASCII string "E", or the string "END," and it ends the current Immersion MicroScribe-3D session. The Immersion MicroScribe-3D will echo the command byte (hex C5) and return to its power-up sequence for baud rate detection as described in Initialization Sequence. Normal operation will not resume until the host has followed the Initialization Sequence. The response packet for an End command contains only the echoed command byte (hex C5).

Get Max Field Values: Hex 46 or C6

The Get Max Field Values command causes the MicroScribe-3D to report the maximum possible values for each field that could occur in a response packet. It generates a response packet that includes all possible packet fields (timestamp, all controllers, all encoders), with each one set to its maximum possible value. If a field corresponds to some optional or custom equipment that is absent from this particular system, then that field will be reported as zero. The response packet to a Get Max Field Values command is as follows:

Byte #	Field rep'd	Meaning
0	Command Byte	C6
1	Buttons	each bit rep's a button: 0 = not supported, 1 = supported.
2-3	Timestamp	Maximum value timer will reach before wrapping around
4	Controller #0	0 = not supported, non-0 = controller's full-scale value
5	Controller #1	0 = not supported, non-0 = controller's full-scale value
...	...	...
11	Controller #7	0 = not supported, non-0 = controller's full-scale value

12	Extra A/D Bits	bit values corresponding to full-scale A/D readings
13-14	Angle #0	Max value reached by encoder before full revolution. (equals # pulses/rev minus one)
15-16	Angle #1	Max value reached by encoder before full revolution. (equals # pulses/rev minus one)
...	...	...
23-24	Angle #5	Max value reached by encoder before full revolution. (equals # pulses/rev minus one)

Get Product Name: Hex 48 or C8

The Get Product Name command requests a printable ASCII string with the name of this Immersion Corp. product. For an Immersion Microscribe-3D, this string is "Microscribe-3D". The response packet generated by a Get Product Name command contains the echoed command byte (hex C8), followed by a null-terminated string.

Get Immersion Product ID: Hex 49 or C9 (note that 49 is ASCII for ÔI')

The Get Immersion Product ID command requests a short ASCII ID string to identify this Immersion Corp. product. This command can be used to verify that the product is still functioning properly, in cases when some errors have occured or no data has been received for a while. A product's ID string is the same as the string that is sent in response to a "BEGIN" command during the Initialization Sequence. For the Immersion MicroScribe-3D, this string is "MSCR". The response packet generated by a Get Immersion Product ID command contains the echoed command byte (hex C9), followed by a null-terminated string.

Get Model Name: Hex 4A or CA

The Get Model Name command requests an ASCII string that identifies the particular model of this Immersion Corp. product. For example, an Immersion Microscribe-3DX would respond with the string "DX". The response packet generated by a Get Model Name command contains the echoed command byte (hex CA), followed by a null-terminated string.

Get Serial Number: Hex 4B or CB

The Get Serial Number command requests an ASCII string for this Immersion Corp. product's serial number. The response packet generated by a Product Identification command contains the echoed command byte (hex CB), followed by a null-terminated string.

Get Comment: Hex 4C or CC

The Get Comment command requests an ASCII string describing any custom information about this Immersion Corp. product. The response packet generated by a Get Comment command contains the echoed command byte (hex CC), followed by a null-terminated string.

Get Parameter Format: Hex 4D or CD

The Get Parameter Format command requests an ASCII string that identifies the format in which this Immersion Microscribe-3D's physical parameters are stored. An example of such a string is "Format 1.1". This is provided for compatibility checking. The only format in use as of May, 1994 is format 1.1. The response packet generated by a Get Parameter Format command contains the echoed command byte (hex CD), followed by a null-terminated string.

Get Firmware Version: Hex 4E or CE

The Get Firmware Version command requests an ASCII string that identifies this Immersion Corp. product's ROM firmware version. An example of such a string is "Format DH0.5". This is provided for compatibility checking. The response packet generated by a Get Firmware Version command contains the echoed command byte (hex CE), followed by a null-terminated string.

Respond to Motion: Hex 4F or CF

The Respond to Motion command causes the Immersion MicroScribe-3D to monitor its time-varying quantities and spontaneously send packets when any quantities have changed sufficiently. This command requires parameters specifying a minimum delay between spontaneous packets, a minimum change required for each time-varying quantity to generate a packet, which buttons will generate a packet if clicked, and what command byte should be used to generate the packet when a packet needs to be sent. If the minimum time delay is zero, then there is no restriction on the frequency of reported packets. If any field's minumum change is given as zero, then that quantity will not trigger a packet, no matter how much it changes. If ALL fields after the command byte (byte #3) are zero, then the Immersion MicroScribe-3D will generate packets at a fixed rate determined by the delay parameter, regardless of motion or button-clicking. The arguments required are listed in the table below. The response packet generated by a Respond to Motion command contains only the echoed command byte (hex CF).

Byte #	Meaning
0	Command Byte 4F or CF
1-2	Minimum delay between packets: 16-bit integer, in ticks of approximately 1 ms
3	Command Byte that is to be responded to when sufficient change is detected
4	One bit per button, 0=button click does not generate packet, 1=it does
5	Controller #0: min change required to generate packet
6	Controller #1: min change required to generate packet
...	...
12	Controller #7: min change required to generate packet
13-14	Angle #0: min change in encoder count required to generate packet
15-16	Angle #1: min change in encoder count required to generate packet
...	...
23-24	Angle #5: min change required in encoder count to generate packet

Insert Marker: Hex 52 or D2

The Insert Marker command gives the host process a way to insert place markers in the Immersion MicroScribe-3D's data stream. This can be useful for real-time situations in which the host needs to changes modes at a precise instant relevant to the incoming MicroScribe-3D data. If several packets are in the host's input buffer when an external event requires a mode change, the host will generally not be able to tell which packets arrived before the mode change and which arrived after, unless some additional timing mechanism is implemented. The Insert Marker command allows the host to command the MicroScribe-3D to send a place mark as a special type of packet, thereby marking

where the event occured in relation to unprocessed packet data. The Insert Marker command requires a one-byte argument to be used as an arbitrary place marker. The response packet generated by an Insert Marker command contains the echoed command byte (hex D2), followed by the echoed place marker byte.

Get Extended Physical Parameters: Hex 53 or D3

The Get Extended Physical Parameters command causes the Microscribe-3D to report all extended physical parameters not already sent by Get Physical Parameters. These parameters may be necessary to calculate stylus coordinates from the five joint angles. The comment string should be checked (Get Comment) to determine if Get Extended Physical Parameters is necessary. If the comment string is "Standard", this command is not needed. If the comment string is "Standard+Beta", this command is required. Calling this command if it not required will cause an hci timeout. Refer to the function arm_get_constants() in arm.c for an example of how to implement this check. The response packet for a Get Extended Physical Parameters command is as follows:

Byte #	Meaning
0	Command Byte C0
1	Number of bytes to follow in this parameter set (2)
2-3	BETA, 16-bit signed integer, -180 deg is -32768