Date: Sept 30, 1988

Subject: Apple Desktop Bus (ADB) Keyboard Microcontroller ERS

Document Version Number: 00.03k

Revision History

Ver	Date	Changes or Additions
Draft !!!	April 5, 88	
00.01	July 5, 1988	Adding some notes to explain the hardware and firmware implementations of keyuc in Discovery and future machines.
00.02	Sept 6, 1988	Add a new section for version number.
00.03	Sept 30, 1988	More descriptions for turning on Sticky key mode with 5 down/up SHIFT keys.

Introduction

This document describes the operation of the single-chip-microcontroller (uC), M50741, which resides between the system CPU and the Apple Desktop Bus (ADB) devices. Since It is an intelligent controller that overseas the ADB and thus always refers as ADB microcontroller or Keyboard micro. The M50741 microcontroller uses a superset of the 6502 instruction set and contains 96 bytes of RAM and 4K of ROM.

Basic Functions

1. Acts as ADB host for the keyboard and mouse by periodically polling those devices.
2. This microcontroller also acts as a transceiver chip for other ADB devices. It works together with ADB General Logic Unit (GLU) to form a communication path between the system and the devices.
3. It controls Sticky Keys and Keyboard Mouse and ADB Mouse scaling.

Differences between IIgs and Discovery keyboard micro

1. IIgs uses M50740 (3K ROM, 96 bytes RAM) but Discovery uses M50741 (4K ROM, 96 bytes RAM).
2. No internal keyboard scanning in Discovery.
3. New in Discovery:
   (a) Sticky Keys and Keyboard Mouse
   (b) ADB Mouse or Mouse type devices scaling
   (c) New commands to support new functions
   (d) Extend two more steps for keys repeat speed
   (e) Bugs fix and miscellaneous changes.....

Version Number

The new keyboard-controller is rated as version number 6. It can be read via the ADB tool call.

General and Functional Descriptions

Refer to Apple IIgs Firmware Reference Chapter 9: Apple Desktop Bus Microcontroller for details.

Commands and Data for Keyboard Micro

Old IIgs commands, see Apple IIgs Firmware Reference: Chapter 9, Apple Desktop Bus Microcontroller.

New commands and data for Discovery ADB microcontroller to interpret: these commands and data are sent from the system firmware directly or through ADB tool set routines. When the ADB microcontroller receives the specific command and the required data, then the microcontroller would react to some commands immediately or wait for the following data to be interpreted as certain specific functions in the microcontroller.

The following table lists the new commands being add into the Discovery ADB microcontroller.

Receives data from the microcontroller by specific command.

Command	dataLength	Name	Action
$20	2	readKBMouse	Read Keyboard Mouse parameters Byte1: high nibble - bit7: reserved but not modify bit6-4: Delay-to-Start, 0-4 low nibble - mouse tracking, 0-4 Byte2: high nibble - acceleration rate, 4-0 low nibble - maximum speed, 0-9

Sends data to the microcontroller by specific command.

Command	dataLength	Name	Action
$12	2	sendKBMouse	Send Keyboard Mouse parameters Byte1: high nibble - bit7: reserved but not modify bit6-4: Delay-to-Start, 0-4 low nibble - mouse tracking, 0-4 Byte2: high nibble - acceleration rate, 4-0 low nibble - maximum speed, 0-9
$16	1	setSticky	Set sticky keys on/off If data byte is zero then turns off and nonzero for on.
$17	1	setKBMouse	Set Keyboard Mouse on/off If data byte is zero then turns off and nonzero for on.

Special sections for Sticky Keys and Keyboard Mouse

This section describes Sticky Keys and Keyboard Mouse in details. Since it is designed to aid input events from keyboard and mouse and thus sometimes refer as **Easy Input Access**.

Easy Input Access, a built-in firmware in keyboard micro, which provides additional support for our handicapped users. It consists of two parts, **Sticky Keys**, which lets you type combination keystrokes one key at a time, and **Mouse Keys (keyboard equivalent for mouse input)**, which lets you manipulate the mouse pointer from the keyboard.

During power up, warm start and forced cold start, the **Sticky Keys** and **Mouse Keys** are always off.

Sticky Keys

Sticky Keys are activated by pressing the SHIFT key five times in succession without moving the mouse (this is to prevent accidentally starting Sticky Keys when shift-click-dragging).

If Sticky Keys are already activated, it will be turned off instead. The mouse events (button on/off or movements) in between of the five down/up SHIFT key do not affect the accumulation of this five SHIFT key events.

If users use the SHIFT key in combination with character keys and release the character keys before the SHIFT key and then release the SHIFT key some time later, then this is treated as one down/up SHIFT count. Therefore, four other successive down/up SHIFT counts are required to activate the Sticky keys. This is because the modifier keys are updated every 8 ms if there is no other character keys being pressed.

Also pressing any keys simultaneously with a modifier (not Caps Lock) will automatically turn off Sticky Keys. Normal user doesn't have to worry about getting locked into the Sticky Keys mode unless one intends to turn it on purposely.

Also a specific Apple DeskTop Bus (ADB) command is defined to turn it on/off directly. Applications can take the advantage of this command. Detailed in later session.

When this feature is active, pressing any of the four non-locking modifier keys (i.e. control, shift, command (option or open apple) and solid apple, from now on referred to simply as modifiers) will cause the appropriate key to "stick", that is, it will appear to remain down until any key other than modifier is pressed.

If the modifier key is pressed twice in succession, it will stay down until the same modifier key is pressed again.

A mouse click would never have any affect to this feature because it is not treated as a key press.

Goals of a good design

The goals of a good mouse keys design are 1) to provide the same information that the mouse provides to the programs, namely pointer position, motion, and button events and 2) to provide a good human-machine interface.

Which keys to use

The two considerations in determining which keys to use are 1) which keys provide the most intuitive interface to moving the pointer around the screen and 2) which keys are duplicated by others so that with mouse keys enabled, the user can still operate the software.

Enable/Disable Mouse Keys

The APPLE-SHIFT-CLEAR combination keys are selected to turn on the mouse keys. To turn off the mouse keys, just press CLEAR key. The reason for not using the same combinations to get in and out is that it can be very confusing. It is difficult to determine if the mouse keys are enabled or disabled unless there is some indication fo the state. During power up, it is off.

For some users, it may be necessary to get into the Sticky keys first and then activate with the mentioned keys.

Also a specific Apple DeskTop Bus (ADB) command is defined to turn it on/off directly. Applications can take the advantage of this command. Detailed in later session.

Buttons and Direction keys

The numeric keypad is a very good choice since the key layout provides an intuitive set of keys for moving the pointer in eight directions, and most of the numeric keypad is duplicated by other keys on the keyboard.

Besides the direction keys (the eight keys around keypad '5'), keypad '5' is defined as button0, keypad '0' is defined as button0 locked so that the dragging can be operated by pressing this key first and then the direction keys. Keypad '.' releases the locked button0. In some countries the ',' key is used instead of the '.' key. In this case the ',' key is used for releasing button0 from locked mode. Remember it is always the key next to '0'.
Kepad '-' is assigned as button1, keypad '=' is button1 locked and '/' for releasing from button1 locked as in the case of button0.

A two button mouse is supported as opposed to the one button in ADB mouse. An application can take the advantage of this second button if desired.

Besides the above selected keys, a new feature which operates off of the * key on the numeric keypad keys: "*" followed by a number (1,2,3...0) sets the number of pixels (1 to 10) that the mouse keys will jump when the user hits a mouse key and releases it. At startup this number is 1. This multiplier does not affect acceleration and maximum speed.

Hitting "*" twice in a row clears it back to 1 pixel (as would * 1).

Use of auto repeat feature

The key repeat delay is the time after the press of a key before the auto repeat starts.
The key repeat interval is the time between the auto repeat of the keys, i.e. the number of keys being repeated per second if the key is still held down.

The Mouse Keys are actually the keys on the keyboard. Therefore the repeat feature is directly related to the mouse pointer motion.

The repeat delay and repeat interval of the Mouse Keys is independent of the normal keys. It is designed so that the user can fully control the normal repeat feature and also the motion of the mouse pointer without affecting each other.

Mouse Panel (in control panel)

A new panel is created to control the activities of the mouse, these include the ADB (Apple Desktop Bus) mouse and the Mouse keys of the numeric keypad keys.

Displays of Mouse Panel:

```
Mouse Tracking: |*----------|
Double     Click: |-----*---------|

-Keyboard Mouse-
Delay-to-Start   : |-----*---------|
Acceleration     : |-----*---------|
Maximum Speed: |------*--------|
```

Mouse Tracking, 5 steps i.e. 0, 2 ,4, 6, 8
It is the scaling of the mouse cursor and only affect the Apple DeskTop Bus (ADB) mouse. The keyboard mouse can control its movements by setting its maximum speed. It is defined with a manipulation rates of 0, 2 ,4, 6, 8 with respect to each step. The mouse curosr can be manipulated only if the x-y coordinates send from the mouse are equal or greater than 3. Design in this way so that the user can fully control the mouse cursor in slow movements.

Double Click, 5 steps i.e. 50, 40, 30, 20, 10 ticks, 1 tick=1/60 sec
Setting the time interval for double click to activate an event. Imply to both ADB and Keyboard mouse.

Delay-to-Start, 5 steps i.e. 0.25, 0.50, 0.75, 1.00, 1.25 sec
It is same as key repeat delay but it only affects mouse keys if mouse keys are enabled.
It is the time after the press of a mouse key before it starts moving.

Acceleration, 5 steps i.e. 6, 4.5, 3, 1.5, 0 sec
It is the time to take to attain its maximum speed.

Maxm Speed, 10 steps i.e. 180, 120, 60, 40, 30, 24, 20, 15, 11, 8 pixels/sec
It is the maximum speed the mouse can move.

Parameters setting through Control Panel via ADB tool

To send the parameters being updated through the control panel from the battery back up RAM to keyboard micro-controller, a specific command is required. The communication protocol is what had been implemented in //gs. Refer to Apple //gs ADB toolbox for details.

Command : $12, 00010010 and the following 2 data bytes

byte1 : high nibble - bit7: reserved but not modify
bit6-4: Delay-to-Start, 0-4
low nibble - mouse tracking, 0-4

byte2 : high nibble - acceleration rate, 4-0
low nibble - maximum speed, 0-9

Reading the parameters from the system via ADB tool

To read the mouse keys' parameters from the system, an ADB command is defined:

Command : $20, 00100000 and waiting for 2 bytes to be returned

return byte1 : what had been set by the system previously

byte2 : what had been set by the system previously

Smooth versus jumpy pointer motion

The keyboard is a discrete event device, so the most natural way to implement pointer motion using the keyboard is to move the mouse every time there is a key, or a key repeat. This would produce a smooth motion if the pointer moved only a pixel per key. However, with this method it would take a long time to move across the screen for users with a slow key repeat rate.

Since the pointer motion is directly related to the Delay-to-start, acceleration and maximum speed and all those parameters can be set through the mouse panel, therefore the motion of the pointer will depend on those parameters to move smoothly or jumpy.

Due to the different modes of display on the Apple // system, it is best to have different parameters for different modes of movement.

For example, in //gs, there are four kinds of display:

1) Super high resolution, 320 x 200 or 640 x 200 pixels
2) Double high resolution, 560 x 192 pixels
3) High resolution, 280 x 192 pixels
4) Low resolution, 48 x 40 pixels

Since the Finder is designed to poll the activities of insertion and ejection of the disks from the disk drives, it is barely able to obtain a smooth motion.