Date: Oct. 24, 1985

Author: Peter Baum

Subject: Front Desk Bus ERS

Document Version Number: 02:50

6/17 Updated Appendix A - Commands to uC: MODES
6/21 Updated Appendix A,B; added boot sequence & filled in misc. ????
8/29 Updated Appendix A
10/24 misc.

Single-chip-microcomputer Keyboard Interface (SKI)

This document describes the software protocols for communication between the system processor and the single-chip processor which controls the internal keyboard and Front Desk Bus (FDB).

Cast of Devices:

uC - Single-chip micro with 3 basic functions:

1) Scans the built-in (internal) keyboard and periodically polls FDB for Keyboard and Keypad data.

2) Acts as FDB host for the mouse by periodically polling the FDB mouse.

3) This device also acts as a transceiver chip for other FDB devices. The system tells the chip to issue listen/talk commands on FDB.

The uC can be interrupted or polled by the system, but it may not respond for up to 4.5 ms. if it has started a FDB operation. FDB operations cannot be interrupted once they have begun or data will be lost.

KEYGLOO - Allows communication between the uC and the system processor. The chip acts as a holding register so that data written by the uC can be read by the system and data written by the system can be read by the uC. This chip is also used to generate interrupts to the system, and to aid in performing the internal keyboard scan.

THE KEYBOARD

The uC handles all keyboard operations by scanning the built-in keyboard and FDB for keypresses. All keystrokes are passed back to the system using the same method as the Apple //e. If a FDB keyboard or keypad is connected to the system, the uC will act as the FDB host and automatically read keystrokes from the devices. The keyboard matrix is the same as the one implemented on the //e (80 positions) so that the retrofit board can use the existing //e keyboard and keypad.

During normal keyboard operation the SKI will perform the following:

Check for keystokes

- Scan keyboard
  Scanning the built-in keyboard consists of checking for keypresses and converting the keypress into the proper ASCII code. Auto-repeat rate is selectable: no repeat or 40,30,24,20,15,11,8,4 keystrokes per second. The keyboard will only auto-repeat as fast as keys are being read. If the buffer (normally 1-key, unless buffer mode selected which employs 16-key buffer) is not empty than an auto-repeat key will be not be put in the buffer (This prevents the cursor, etc. from jumping immediately after long operations such as disk accesses, etc). The delay before auto-repeat is also selectable: 1/4, 1/2, 3/4, & 1 sec. The keyboard scan will attempt to implement the same idiosyncrasies as the current keyboard encoder (1-key buffer, pseudo N-key rollover including ghosting and phantom keys).

  International keyboard layouts are identified at power-up by reading a specific location in the battery backed RAM. A command can be executed to change the current layout. On power-up the uC will use the keyboard layout specified by a command from the system. On reset the uC will use the last layout specified by the software or system menu.

  A new mode, called dual speed mode, doubles the auto-repeat rate for the four arrow keys, when the control key is pressed. This mode is always enabled. An optional extension of this mode will allow the user to also double the repeat rate of the delete key and the spacebar when the control key is pressed. This mode extension must be enabled (using the setup menu / control panel). Another optional mode allows the user to repeat at 4 times the normal repeat rate, instead of just double.

- Poll FDB using Keyboard/Keypad address
  All FDB keyboards and keypads will automatically be handled by the uC. Keystrokes read from FDB keyboards/keypads will be incorporated into the normal stream of keystrokes detected on the built-in keyboard. A command that disables the automatic FDB keyboard/keypad poll will be implemented so that a multi-player game that requires many keypads can be used.

Return Keystroke data to system

- Key presses are returned by loading keylatch w/ data
  Normal keyboard operations are compatible with the Apple //e and //c keyboard. The Keylatch is read at address $C000, with the MSB indicating whether the key is valid (KYSTB). AKD is read on MSB of address $C010 and the KYSTB is cleared by reading $C010 or writing $C01X.

- Key modifiers are updated every keypress
  Key modifiers, such as Shift, Control, Capslock, Open Apple, Solid Apple, Auto-repeat, and Keypad are stored in the key-modifiers latch. The key-modifiers latch is updated when a key is pressed and the ASCII value is loaded into the keylatch. The values stored in the key-modifiers latch reflect the state of the key modifiers when the key was pressed and not the current state of the modifiers. This allows a program to read the keylatch and modifiers after a disk operation and detect if open-apple was pressed when the key was pressed. Currently if a key and open-apple are pressed during a disk operation, a program will detect that a key was pressed, but may miss the open-apple, since it was let up before the disk access was finished. (Keystrokes are not read during ProDOS operations).

Bit	Modifier
0	Shift
1	Control
2	Lock
3	Repeat
4	Keypad
5	Updated Modifier latch without keypress
6	Solid Apple
7	Open Apple

Bit 5 (Update bit) is used to signify that the modifier latch was changed without any other keypresses occuring. This will only occur when the KYSTB is clear. For example, if only the control or shift key is pressed and the KYSTB iss clear, then the uC would indicate this by setting the Update bit and changing the status of the control or shift bit in the modifiers latch. When a new key was pressed, such as 'x', then the modifier latch would be updated along with the Keylatch (and KYSTB is set) and the update bit would be cleared. The modifier latch will be updated in only two cases: Whenever a new key is written into the Keylatch (with the update bit cleared) and if the KYSTB is clear and a modifier condition changes (with the update bit cleared).

Appendix B shows the keycodes generated by the FDB keyboard. There are some extra undefined codes for FDB keyboards (codes 96-126). These codes are handled by just passing them directly through the keyboard latch with the keypad bit set. These codes can then be used as macros keys, software defined keys, or function keys. The code 127 ($7F) is reserved for reset (not to be confused with keypad key 64, the DELETE key, which will be translated into ASCII code $7F with the keypad bit set).

The Open and Solid Apple bits are needed so that the uC knows when to drive the Open or Solid Apple outputs. These outputs are driven high before the key is sent to simulate someone pressing the Apple keys. This allows the system to emulate the existing keyboard with the FDB keyboard.

Since the current keyboard is unbuffered and allows an overrun to occur, the key-modifiers latch is not always valid (unless the keyboard is in buffering mode). There is a small window of time where the keylatch has key1, while the key-modifier latch has modifiers to key2. (The key modifiers must be written out first, since keylatch sets the KYSTB, indicating both bytes are valid). To verify that the key-modifiers latch is accurate, both the keylatch and key-modifiers latches must be read until the data in each latch is the same for two successive times. The second pair of latch reads should be at least 30 usec. apart. (If the keyboard has been placed in Buffer Mode then the modifier byte is valid after the KYSTB is set). The different methods of reading the keyboard are described later in this document.

- Keyboard interrupt mode

  The keygloo chip can be setup to interrupt whenever the keylatch is written to by the uC. This interrupt can be cleared with two different operations. Typically, software will read a key then clear the interrupt by clearing the KYSTB. But occasionally certain tasks, such as a background routine, want to intercept a key before it gets to the keyboard. If the software installs an interrupt handler that looks for a specific keypress, the handler must always clear the interrupt, regardless of whether it reads the keyboard, or the system will hang in an infinite loop. If the handler chooses to ignore the current key and not clear the KYSTB, then it can clear the interrupt by reading the keyboard.

- Keyboard buffering mode

  Another command will enable a buffer mode where the uC buffers keystrokes & modifiers. The uC will send a new keystroke and modifiers whenever the KYSTB is cleared (FLUSH buffer command, also). In this mode the system polls the keylatch until the MSB is set, then reads the keymodifiers latch, and finally clears the KYSTB. Both latches will stay valid until the KYSTB is cleared.

- Reset and the Keyboard

The uC will periodically sample the RESETin line to determine whether it should RESET the system (using the RESETout signal). If it detects that RESETin is low then the uC will check if both the CONTROL key and RESET have been pressed on either the internal or the FDB keyboard before setting RESETout low. If CONTROL has not been pressed then the uC will continue sampling the internal keyboard and FDB, but will only set RESETout if RESETin is still low while CONTROL is depressed.

When RESETin is released, RESETout will also be released. If the FDB OPEN Apple key had been pressed then the uC will simulate this by pulling the OAPLout line high. The system firmware will detect this and institute a power-up reset sequence, which includes a software reset command to the uC, which will then reset itself and FDB. Even though RESET is pressed the keyboard must continue to scan both the internal keyboard and the FDB keyboard, so that the status of the Apple keys can be maintained.

The keyboard can be read by the system using one of the following modes:

APPLE // MODE

Compatible with existing Apple //. Unbuffered and asynchronous. (Read keyboard data @ $C000, Clear Keyboard Strobe @ $C010). The (Open & Solid) Apple keys are read just like on the Apple //, at softswitch locations $C061 & $C062.

APPLE // MODE with KEY MODIFIERS

Emulates existing Apple //, but extends the keyboard data by including keyboard modifiers. The bits in the modifiers latch which represent the status of the Apple keys may not reflect the same state as the Apple key locations $C061 & $C062. When the uC is running with the keyboard unbuffered the Apple key softswitch values always reflect the state of the Apple key inputs. The Apple bits in the modifiers latch may not be the same because the modifiers latch is updated as follows:

1) Whenever the keylatch is updated with a new ASCII code
2) If no keys are down and the KYSTB is clear, then the modifier latch is updated a[illegible]

In this unbuffered mode the uC updates the keylatch and modifier latch asynchronously to the system. To determine whether the data in the key modifiers latch is accurate, use the following method:

1) If bit 5 (Updated w/o keypress) is set then the latch contents are accurate.

2) Otherwise both the keyboard latch and Modifier register should be read (30 us. apart) until the data in each is the same for two successive times.

BUFFERED APPLE // MODE

Emulates Apple // mode with Key Modifiers, but only sends new keystrokes and modifiers after the KYSTB has been cleared. To use this mode properly both bytes, the keyboard latch and the modifiers register, should be read, while the Apple key locations ($C061, $C062) should be ignored. The program looks for keystrokes by waiting until the KYSTB bit (MSB of keyboard latch) is set, before reading the keyboard and modifier latch. After reading both bytes the Keyboard strobe is cleared to indicate that the program is ready for the next keystroke.

In this mode, a program can also detect if only a modifier key such as Control, Shift, or Lock has been pressed. Normally the modifiers latch reflects the state of the modifiers keys when another key was pressed. But if no keys are down (which can be detected by checking the AKD flag on the MSB of location $C010), the KYSTB is clear, and the Update bit (5) of the modifiers register is set, then the modifiers byte has been updated to reflect the current state of the modifiers. If another keypress occurs then the Update will be cleared, both the keylatch and modifiers latch will be updated, and the KYSTB is set. The KYSTB must be cleared before the modifiers latch will be updated.

While a user can switch in this mode with existing software to prevent the system from missing keystrokes when an overrun occurs, it has certain side effects. Some existing software will automatically clear the strobe at certain times, such as coming back from a disk access. In this mode the automatic clear will only clear the first key of a string of keystrokes. Also since the buffer has no control over the Open Ap[illegible], existing software, such as games, that reads the hardware locations ($C061, $C062) may not interpret the Apple keys properly. In this buffer mode the Apple key softswitch locations $C061 & $C061 will reflect the state of the Apple key bits in the modifier latch.

The user can flush the buffer by pressing certain keys at the same time. Currently this key sequence is CONTROL-OPEN APPLE-DELETE.

THE FRONT DESK BUS MOUSE

The FDB mouse is handled automatically by the uC. The uC will periodically poll the FDB mouse to check for mouse activity. If the mouse has moved or the button pushed, it will respond to the FDB mouse poll by returning two bytes of data. The uC will return this data to the system by writing both mouse data bytes to the keygloo chip (Mouse byte Y followed by byte X, which enables the interrupt). The system would check the status register to verify that a mouse interrupt had taken place and then read the two data bytes, with mouse latch Y read first. The keygloo would clear the interrupt whenever the second latch was read. To prevent an overrun the uC would only write out mouse data when the registers are empty (i.e. after mouse latch X has been read by the system)

The advantages of this protocol is that the system is only interrupted at VBL time, if the mouse has moved. This keeps the number of interrupts, and therefore the system overhead, to a bare minimum.

The uC won't do another FDB mouse poll until both bytes have been transferred to the system.

Part of the initialization protocol includes the system sending the FDB address of the device to be automatically polled. While this address will typically indicate the FDB mouse as the polled device, it is possible to specify some other FDB device (with one caveat: Whichever device is picked must transfer only 2 bytes. The uC will ignore all data sent by the mouse device if more than 2 bytes are sent, since there is no way to handle more automatically.)

The FDB mouse returns 16 bits of data as follows:

Bit	Function
15	Button 1 Pressed
14-8	Y-Delta Movement (negative=up, positive=down)
7	'1'
6-0	X-Delta Movement (negative=left, positive=right)

OTHER FRONT DESK BUS COMMANDS

All other FDB devices will only be polled when the system sends a FDB POLL token to the uC. In this mode the uC acts as a dumb transceiver for FDB activity. This command protocol requires that the system specify the FDB command byte to be transmitted. The uC will transmit the byte then wait for the FDB response. The uC returns data by storing a token in the data latch identifying the data that will follow, then sending a new data byte each time the system reads the previous one.

If the token stored by the system indicates a FDB listen command then the uC will read all the data bytes before initiating the FDB operation. The uC will suspend all other operations until it receives all data bytes.

The FDB response token stored in the data latch will indicate to the system that the uC is responding to a FDB command. This token will contain status bits which indicate if the FDB device responded (data valid), how many bytes are coming (typically 2), and whether a Service Request (SRQ) on FDB was detected. For example, only one byte will be sent back if there was no response to the FDB poll. This byte will be the response token indicating no response and SRQ status.

Whenever the token byte of a multi-byte response is stored in the data latch, the uC will wait around for 1 millisecond for the system to read the first data byte. This will allow the system to read the FDB data back quickly if the data latch interrupt and system interrupts are enabled.

If a FDB Service Request is detected and none of the auto-poll devices (keyboard, keypad or mouse) are causing the interrupt, then the SRQ token will be written into the data latch register. The SRQ token indicates, by setting the SRQ status bit, that some FDB device is currently requesting service. The system should start reading (polling) FDB devices, when this bit is set, by sending FDB poll commands to the uC. A device which is requesting service will respond with data when it is polled.

If the system receives a FDB response with the SRQ status clear, then it should assume that there are no FDB devices requiring service. The converse is not true and it may be possible for an SRQ set status, which disappears later, to be passed to the system. The system must be prepared to receive handle spurious SRQ's. For example, if data is returned from a poll of a device which is not auto-polled and SRQ is also set, then the SRQ may be from an auto-poll device and could disappear when the uC automatically does a poll of that device.

The system can send data to FDB devices (FDB listen mode) by sending the FDB LISTEN token to the uC, followed by the command byte, then two data bytes. This transaction uses the CMD/DATA latch to transfer the bytes from the system to the uC.

OTHER COMMANDS TO THE uC

Abort Command

If the system passes the abort command to the uC, then the uC will flush out any active commands. All commands which require that the uC transfer data to the system using the data latch will be abruptly terminated.

DATA RETURNED BY THE uC

The Special Key Sequence Bit

One bit is reserved to signify that a special key sequence has been pressed. This bit can be used by the desktop manager, or desktop accessories, or a switcher program. Using the data latch, rather than using the keyboard latch, allows the uC to interrupt the system and indicate this condition, without also sending a dummy keystroke.

BOOT SEQUENCE PROTOCOL

At boot time the uC will do some preliminary initialization and then attempt to synchronize itself with the system by waiting for the SYNCH command (7) from the system. All other commands will be ignored. If the uC doesn't receive the SYNCH command within 1.5 seconds then it will use its built in defaults:

Mode byte	= All modes clear (See Appendix A - Command 5)
Delay before auto-repeat	= 3/4 sec
Repeat rate	= 15/sec
Language & Layout	= U.S.
FDB Keyboard address	= $02
FDB Mouse Address	= $03

After this initialization the system can change the defaults by using the change configuration bytes command.

APPENDIX A - uC COMMANDS

COMMANDS TO uC:

********** Make init commands two byte commands **********

BIT 76543210

00000000 -
00000001 RESET KEYBOARD
00000010 ABORT COMMAND
00000011 FLUSH KEYBOARD

00000100 SET MODES using next byte as follows:
00000101 CLR MODES using next byte as follows:

Bit	Function
7	-
6	Set XOR LOCK-SHIFT mode
5	-
4	Buffer keyboard mode
3	4X repeat enabled, instead of Dual (2X) repeat
2	Include Spacebar, Delete key on Dual repeat
1	Disable Auto-poll of FDB mouse
0	Disable Auto-poll of FDB keyboard

00000110 * SET CONFIGURATION BYTES using next 3 bytes as follows:

Byte 1:
HI nibble - Set Auto-repeat rate (3 bits)
(No repeat or 40,30,24,20,15,11,8,4 Keystrokes/sec)
LO nibble - Set Delay to repeat rate (3 bits)
(1/4, 1/2, 3/4, & 1 sec)

Byte 2:
HI nibble - Set Keyboard Layout Language
LO Nibble - Char.set (needed for certain langs.)

Byte 3:
HI nibble - FDB mouse address
LO Nibble - FDB keyboard address

LAYOUT/LANG.	=	CODE:	
US	=	0	
DVORAK	=	1	
SPANISH	=	2	
CANADIAN	=	3	
FRENCH	=	4	
ITALIAN	=	5	
GERMAN	=	6	
SWEDISH	=	7	
UK	=	8	
DANISH	=	9	(CURRENTLY NOT IMPLEMENTED)

00000111 SYNCH COMMAND

Sets MODES byte (See Command 4 or 5 above) followed by Configuration bytes (Command 6).

00001000 WRITE uC MEMORY
Send 1 byte address (for RAM) followed by 1 byte of data

00001001 READ uC MEMORY
Send 2 bytes address of uC location (ROM or RAM)

00001010 READ MODES BYTE (See command 4 or 5 above)

00001011 * READ CONFIGURATION BYTES - Returned in Data latch:
(NOTE: returned in reverse order from command 6 above)

Byte 1:
HI nibble - FDB mouse address
LO Nibble - FDB keyboard address

Byte 2:
Hi nibble - Set Keyboard Layout Language
LO Nibble - Char.set (needed for certain langs.)

Byte 3:
HI nibble - Set Auto-repeat rate (3 bits)
LO nibble - Set Delay to repeat rate (3 bits)

000011-- READ FBD ERROR BYTE - Returned in Data latch

0001---- -

001----- -

01---abc Poll FDB w/Command in 2nd byte. Transaction involves abc bytes. (limit of 8 byte transfers)

1cxyabcd Poll FDB device:
Address - abcd
Register- xy
Command - 1c
c =1 talk
c =0 listen

This assumes that the FDB command is to either Talk or Listen. Other FDB commands are implemented using a 2 byte protocol (see above). If the command is Listen, then a 2-byte transfer is assumed.

* All commands which require more than a 1 byte transfer, will automatically timeout in 10 ms. if there is no response.

The following commands will also be added thought the exact protocol hasn't been determined yet:

RESET SYSTEM - Used by software to RESET the system
CHARSET AVAILABLE - Returns a list of character sets which are available in in the MEGA chip. This assumes that each Keyboard Micro is paired with a single MEGA chip. Protocol will probably require 1st byte returned to indicate the number of

character sets available, with each set number to follow.
LAYOUTS AVAILABLE - Returns a list of Layouts provided by MEGA chip. Will use same protocol as Charset above.
GETKEYS DOWN - Sends back a list of the current keys which are down, including modifiers & both the internal and FDB keyboard.

****
REORGANIZE THESE SO THAT LANGUAGES W/ MORE THAN
26 LETTERS ARE AT THE END
ALSO LEAVE ROOM FOR DANISH
****

****
Optimize the talk commands since they require only one byte transfers
****

COMMANDS BYTES RETURNED BY THE uC:

BIT	Function
7	Response byte if set, otherwise status byte
6	SRQ Valid if set
5	Desktop Manager Key Sequence pressed
4	Flush Buffer Key Sequence pressed
3	- (abort ???)
2-0	If all bits clear then no FDB data valid, otherwise the bits indicate the number of valid bytes received -1 (between 2-8 bytes total). (001 means two bytes ready, 011 = 4 bytes, etc.)

APPENDIX B - FDB KEYCODES

CODE	KEY	APPLE // MAC DIFFERENCES	CODE	KEY	APPLE // MAC DIFFERENCES
0	A		48	TAB	
1	S		49	SPACE	
2	D		50	`	
3	F		51	DELETE	
4	H		52	RETURN	*(ENTER)
5	G		53	ESCAPE	*NA
6	Z		54	CONTROL	*NA
7	X		55	OPEN APPLE	*(COMMAND)
8	C		56	SHIFT	
9	V		57	LOCK	
10	*INTERNATIONAL		58	SOLID APPLE	*(OPTION)
11	B		59	LEFT ARROW	
12	Q		60	RIGHT ARROW	
13	W		61	DOWN ARROW	
14	E		62	UP ARROW	
15	R		63		
16	Y		64	DELETE	KEYPAD *NA
17	T		65	.	KEYPAD
18	1		66	RIGHT ARROW(*)	KEYPAD
19	2		67	*	KEYPAD *NA
20	3		68	?	KEYPAD *NA
21	4		69	+	KEYPAD *NA
22	6		70	LEFT ARROW(+)	KEYPAD
23	5		71	ESCAPE	KEYPAD *(CLEAR)
24	=		72	DOWN ARROW(,)	KEYPAD
25	9		73	,	KEYPAD *NA
26	7		74	SPACE	[illegible] *NA
27	-		75	/	KEYPAD *NA
28	8		76	RETURN	KEYPAD *(ENTER)
29	0		77	UP ARROW(/)	KEYPAD
30	]		78	-	KEYPAD
31	O		79	(	KEYPAD *NA
32	U		80	)	KEYPAD *NA
33	[		81		KEYPAD
34	I		82	0	KEYPAD
35	P		83	1	KEYPAD
36	RETURN		84	2	KEYPAD
37	L		85	3	KEYPAD
38	J		86	4	KEYPAD
39	'		87	5	KEYPAD
40	K		88	6	KEYPAD
41	;		89	7	KEYPAD
42	\		90		KEYPAD
43	,		91	8	KEYPAD
44	/		92	9	KEYPAD
45	N		93		KEYPAD
46	M		94		KEYPAD
47	.		95		KEYPAD

CODE FOR RESET UP ($FFFF) & RESET DOWN ($7F7F)
Other keypad codes ()95) are passed directly thru to keylatch