Single-Chip-Microcontroller (controls FDB & Internal keyboard) ERS

Version Number: 3.0 Mar. 10, 1986

Peter Baum

Changes from Version 2.x

Appendix A, International stuff

Added Other Commands to uC

Communication Protocols

Reserved Memory Location $51 for power-on byte

(Allows RAM Disk to live between Cold & Warm Boots)

Single-chip-microcontroller Keyboard Interface (SKI)

This document describes the operation of the single-chip-microcontroller (uC) which controls Front Desk Bus (FDB) and the internal keyboard (built-in keyboard used by the retrofit). This document also describes the software protocols for communication between the system processor and the uC.

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. See Appendix C for exact specifications.

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 & after reset the uC will use the keyboard layout specified by a command from the system. The FDB keyboards have a key labelled '.' which is not very useful internationally, since some languages use the ',' instead. Each layout is preset to default to either the '.' or ',':
  '.'==> US, UK, DV, CN
  ','==> FR, DN, SP, IT, GR, SW
  It is possible to override the default by setting a specific bit when indicating the layout and character set to be used. This bit will swap the setting to the opposite of the preset default. (See SET CONFIGURATION command).

  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 is 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 93-126). 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. The software clears the interrupt by reading the status register followed by a read of the keyboard latch. If any software reads the keyboard after the status register indicates the keyboard interrupt has occurred then the interrupt condition will be cleared! The only other way to emulate keyboard interrupts is to use the VBL interrupt (the heartbeat chain) to read the keyboard and check if new keystroke has come in.

- 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 (which resets the system). If CONTROL has not been pressed then the uC will continue sampling the internal and FDB keyboard, but will only set RESETout if RESETin is still low while CONTROL is depressed.

  When RESETin or CONTROL is released, RESETout will be released. Even while the system is being RESET (RESETout low) the uC must continue to scan both the internal keyboard and the FDB keyboard, so that the release of either the RESET key or CONTROL key can be detected. After RESET (RESETout high) the system firmware will jump to the reset routine which sends the SYNCH command to notify the uC to do a software reset. In the middle of the SYNCH command the uC updates the OPEN-APPLE key by pulling the OAPLout line high if an OPEN-APPLE key is pressed (FDB or internal). At the completion of the SYNCH command the system firmware should immediately sample the OPEN-APPLE key to sense if it is pressed. The system can then determine if a warm or cold boot is required. (Note: The system firmware should sense the OPEN-APPLE key immediately because the OPEN-APPLE key status will be cleared within a few milliseconds by the uC due to a lag time in receiving the OPEN-APPLE key status from a FDB keyboard. In the SYNCH command the uC bypasses the normal method of reading FDB keystrokes and instead reads the instantaneous status of the keyboard to get the OPEN-APPLE key).

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 approximately every 8 ms.

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 Apple and Solid Apple keys, 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. 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 checks the status register to verify that a mouse interrupt has taken place and then reads the two data bytes, with mouse latch Y read first. The keygloo clears the interrupt whenever the second latch is read. To prevent an overrun the uC only writes 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	Clear if Button 0 Pressed
14-8	Y-Delta Movement (negative=up, positive=down)
7	'1' (Button 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 and 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 - See Appendix A

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. Typically used by the system as a placeholder command to synchronize the system with uC.

Reset uC Command

The uC performs a software reset by jumping to its power-on reset routine.

Flush Keyboard Command

This command is used by the system to flush the keyboard type-ahead buffer in the uC. The command also initiates a FDB flush command out to the FDB keyboard and the SRQ disable command to the FDB mouse.

Read & Write uC Memory Command

This command is used by testing programs to verify that the ROM code is accurate and to test other functions.

****

This command is used to read/write memory location $51 in the keyboard micro. Any future version of the keyboard micro must preserve this location as a free byte! (The byte is used to allow the RAM disk to live between applications, even though the user may have attempted to reboot, cold or warm, the system.

****

Read FDB Error Register Command

This command returns an error code generated by a FDB operation. The purpose of this byte is to assist developers of FDB devices to ascertain the cause of any problem encountered with their devices and during FDB operations.

Read Version Number Command

Returns w/ a version number of uC ROM code in the low nybble. The high nybble contains the result of reading the 'R' port of the uC (currently unused).

Read Layouts Command & Read Character Sets Command

These commands are used by the control panel to determine which keyboard layouts and language character sets are available in the system. It assumes that each uC is paired with a single MEGA chip in the system, since the MEGA chip actually contains the language character sets. The actual values passed by the uC to the system correspond to a table defined in the control panel. The order that the values are returned is important because the system must pass to the uC an indication of which layout and character set to use. If the system wants the uC to use character set #12, which was the third value passed back by the uC in response to the Read Character Set Command, then the system sends a 3 to the uC (using either the Send Configuration Command or the Synch Command). The system notifies the uC which value to use by passing its place in the response, not the value in the response (to the Read Layout/Character set Command).

Reset the System Command

This command pulls the System RESET line low for 4ms.

DATA RETURNED BY THE uC - see appendix A

Command/Status Byte

Response/Status Bit

This bit is used to indicate to the system whether a response to a FDB command is available or whether the uC is just notifying the system to a change in status. If it is a FDB response then bits 0-2 indicate how many bytes are ready.

Abort/Keystrobe Set on Flush Bit

If the Abort bit is the only one set in the command/status byte then the uC is indicating it has encountered some sort of catastrophic error and has reset itself. If this bit is set while the Flush Buffer Key Sequence Bit is set then the uC is indicating to the system that the CLRSTB should be cleared (because the key in the keylatch needs to be flushed, which only the system can do).

Desktop Manager Key Sequence Bit

CONTROL & OPEN-APPLE & ESCAPE pressed at same time.

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 a bit in 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. When the Desktop Manager bit is set all previous keystrokes have already been processed (this allows the Desktop Manager keystroke to work with a typeahead buffer).

Flush Buffer Key Sequence Bit

CONTROL & OPEN-APPLE & DELETE pressed at same time.

This bit is used to tell the system to clear any keyboard buffers it may be keeping internally. If the Abort/Flush bit is also set then the CLRSTB should be cleared to flush the keylatch of any pending keystroke. This is needed because the uC has no way of clearing the keylatch. The system shouldn't arbitrarily clear the latch because it may clear out a keystroke sent after the flush.

SRQ Valid Bit

This bit is used to indicate to the system that some device on FDB is requesting service.

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 2.4 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	= US
FDB Keyboard address	= $02
FDB Mouse Address	= $03

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

COMMUNICATION PROTOCOLS

Commands are sent to the uC thru the keygloo command register, while data is returned from the uC in the keygloo data register. The system sends a command by writing a byte to the command register when it is empty, which is indicated in the Command Register Full Bit of the keygloo status register. If the command entails sending more than one byte to the uC, then the succeeding bytes must be written to the command register within 10 ms. of the previous byte being read by the uC (which can be determined by watching for the command register to become empty). If the next byte of a command is not received within 10 ms. of the last byte then the uC will timeout and abort the command.

If the the command requires a response then the uC sends the response back to the system using the keygloo data register. If the command was a

FDB command then the first byte of the response indicates the number of data bytes to follow. Commands other than FDB commands do not require this header byte and just respond with resultant data.

Because the keygloo data register is also used by the uC to pass back status information, care must be taken to guarantee that data and status are not mixed together. To ensure this, the system must wait for the uC to read the command, then should immediately read the data register to clear any status which was sent by the uC earlier (or at the same time which is why clearing it out earlier won't always work). After the uC has read a command, only the response will be passed back to the system thru the data register. By reading the data register immediately after the uC has read the command byte, the data register is emptied and waiting for the data response. If the response is more than one data byte then succeeding bytes must be read within 10 ms. of the previous byte.

Responses to FDB commands (RECEIVE BYTES, POLL FDB) can be handled slightly differently since the first response byte will always contain the MSB set. Since a status byte always has the MSB clear, the system doesn't have to wait until the response is read, but instead just reads the data register until data w/ the MSB set is obtained. Normally the response to a FDB command generates an interrupt at which point the system reads the response.

Great care must be taken with some of the commands since they may clear or reset pending information. For example, if the RESET FDB command is sent at the same time a key is released then the key up code may be lost and the keyboard will auto-repeat until another key is pressed. If the keyboard SRQ is disabled (using DISABLE SRQ) then the keyboard will disappear and no more keystrokes will be read (and if it disappears while a key is down the key will auto-repeat indefinitely since in this case another key can't be read).

APPENDIX A - uC COMMANDS

COMMANDS TO uC:

```
BIT  76543210
-----------------------------------------------------------

     00000000   -
     00000001   ABORT COMMAND
     00000010   RESET KEYBOARD uC
     00000011   FLUSH KEYBOARD

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

                  Bit    Function
                  ------------------------
                   7     Reset on RESET key only (CONTROL not needed)
                   6     Set XOR LOCK-SHIFT mode
                   5     Change FDB Keyboard layout to //e layout
                   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 Nybble - FDB mouse    address
                    LO Nybble - FDB keyboard address

                 Byte 2:
                    HI Nibble - Char.set (needed for certain langs.)
                                MSB set if keypad '.' swapped with ','
                    LO Nybble - Set Keyboard Layout Language
                                LAYOUT/LANG.  =  CODE:
                                ----------------------
                                US       (US) =   0
                                UK       (UK) =   1
                                FRENCH   (FR) =   2
                                DANISH   (DN) =   3
                                SPANISH  (SP) =   4
                                ITALIAN  (IT) =   5
                                GERMAN   (GR) =   6
                                SWEDISH  (SW) =   7
                                DVORAK   (DV) =   8
                                CANADIAN (CN) =   9

                 Byte 3:
                    HI Nybble - Set Delay to repeat rate (3 bits)
                                0: 1/4 sec.
                                1: 1/2 sec.
```

2: 3/4 sec.
3: 1 sec.
4: NO REPEAT

LO Nybble - Set Auto-repeat rate (3 bits)

0: 40 keys/sec
1: 30 keys/sec
2: 24 keys/sec
3: 20 keys/sec
4: 15 keys/sec
5: 11 keys/sec
6: 08 keys/sec
7: 04 keys/sec

00000111 SYNCH COMMAND

Sets MODES byte (See Command 4 or 5 above) followed by Configuration bytes (Command 6). This command is issued by the system after reset to reset the keyboard. After receiving the command the uC will reset itself back to its internal power up state and then reset FDB devices.

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).
1st byte=low adrs byte & 2nd byte=hi adrs byte(=0 if RAM)

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

00001011 READ CONFIGURATION BYTES - Returned in Data latch:
See Set Configuration for values

Byte 1:

HI Nybble - FDB mouse address
LO Nybble - FDB keyboard address

Byte 2:

HI Nybble - Char.set (needed for certain langs.)
LO Nybble - Set Keyboard Layout Language

Byte 3:

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

00001100 READ THEN CLEAR FBD ERROR BYTE - Returned in Data latch

00001101 GET VERSION NUMBER - Returned in Data latch

(Also returns PORT R, which is undefined input port on uC, in HI nybble.)

00001110 READ CHARACTER SETS AVAILABLE - Returns # of bytes, then data
This command is used by control panel to determine which character sets are available in the system. This assumes that each uC is paired with a specific mega chip. (Though mega chips may be paired w/ more than one uC). The order that the character sets are returned is important. The first number returned corresponds to the character set 0 in the mega, while the next number is character set 1, etc.

00001111 READ LAYOUTS AVAILABLE - Returns # of bytes, then the data
This command is used by control panel to determine which keyboard layouts are available in the system. Again, like the character sets available command the order that the number are returned is important. The first number returned represents layout 0 in the uC. A predefined table defines which number corresponds to which layout language.

00010000 RESET THE SYSTEM - Pulls the reset line low for 4 ms.

00010001 SEND FDB KEYCODE - Pretend that 2nd byte is FDB keycode
This command can be used to emulate a FDB keyboard, by accepting keycodes from a device and then sending them to the uC to be processed as keystrokes. This command will not process either RESET up or RESET down codes, so they must be trapped out before using this command. This command can be used to watch for key up sequences.

0001---1 -
001----- -

01000000 RESET FDB - Pulls FDB low for 4 ms.
Care must be taken with this command because resetting a FDB keyboard will clear any pending commands including all key up events. This means that if a keystroke is used to launch this command while the key is released, then the key up code will be lost and the key will auto-repeat until another key is pressed. All keys should be up before this command is executed.

01001000 RECEIVE BYTES - Command, w/ address, is in 2nd byte
The system starts by sending a command byte on FDB and then waits for the uC to pass back any data that it receives. Returns bytes in opposite order (n->1).

01001num TRANSMIT num BYTES - Command, w/ address, is in 2nd byte
Note: If num=0 then command is RECEIVE BYTES described above
The system starts by sending a command followed by between 2 to 8 data bytes (num+1) to the uC, which are to be transmitted over FDB. The command sent will be transmitted

directly as the FDB command byte, which is the first byte received after the TRANSMIT num BYTES command.

0101abcd ENABLE SRQ ON FDB DEVICE AT ADDRESS abcd

[Send command = abcd Listen R3 (abcd1011)]
[data = 0010abcd 00000000]

0110abcd FLUSH BUFFER ON FDB DEVICE AT ADDRESS abcd
This command is dangerous - see RESET FDB description

[Send command = abcd0001]

0111abcd DISABLE SRQ ON FDB DEVICE AT ADDRESS abcd
This command may be dangerous. If data is pending when this command is executed then the pending data may be lost. For example if SRQ is disabled on the FDB keyboard then all key up codes may be lost. Also see RESET FDB description

[Send command = abcd Listen R3 (abcd1011)]
[data = 0000abcd 00000000]

10xyabcd TRANSMIT 2 BYTES:
Address - abcd
Register- xy

Assumes a two byte transfer of data using the FDB Listen command.

11xyabcd Poll FDB device:
Address - abcd
Register- xy

This command is used to get data from a specific device. It uses the FDB Talk command then waits for the device to either send back data or timeout. The uC waits until all data has been received then responds back to the system with a status byte which indicates the number of bytes received followed by the data. It returns the bytes in opposite order than received on FDB (n->1).

[Send command = abcd11xy]
[data = 1st byte - 2nd byte (If Listen command)]

* All commands which require more than a 1 byte transfer, will automatically timeout in 10 ms. if there is no response, except for the SYNCH cmd which may require 20 ms. to process the FDB address byte.

[] Actual Data sent on FDB

COMMANDS BYTES RETURNED BY THE uC:

BIT	Function
7	Response byte if set, otherwise status byte
6	Abort/Flush
5	Desktop Manager Key Sequence pressed
4	Flush Buffer Key Sequence pressed
3	SRQ valid
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	CTRL-X	KEYPAD	*(CLEAR)
24	=		72	DOWN ARROW(,)	KEYPAD	
25	9		73	,	KEYPAD	*NA
26	7		74	SPACE	KEYPAD	*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	ESCAPE	KEYPAD	
43	,		91	8	KEYPAD	
44	/		92	9	KEYPAD	
45	N		93		KEYPAD	
46	M		94		KEYPAD	
47	.		95		KEYPAD	

CODE FOR RESET UP (SFFFF) & RESET DOWN (S7F7F)

Other keypad codes (>93) are passed directly thru to keylatch, except for the international keycode 10 (ASCII = 00) and keycode 63 (ASCII = 01).

APPENDIX A TO SINGLE CHIP MICROCONTROLLER ERS

TO: Hardware Binder 5/20/86
From: Peter Baum

Power-up byte in Keyboard (ADB) Microcontroller

Because the traditional Apple // power-up detection method does not work reliably on the Cortland, the firmware has supplemented the old method with a new one to detect if power-up has occurred (as opposed to a RESET). This method depends on the fact that the 50740 keyboard (ADB) microcontroller chip has static RAM, which is reset almost immediately after power-down. Any hardware or firmware changes to the microcontroller must take this into account.

The old method of power-up detection assumed that when power was off that data in dynamic RAMs would be lost/changed. On Cortland the dynamic RAMs may retain data even if power is off for 30 seconds or more. When this happens the system would assume that a reset, instead of a power-up, had occurred, and certain power-on initialization routines are skipped.

To alleviate this problem another byte is now stored in a spare memory byte in the keyboard (ADB) microcontroller. This byte is checked during reset to determine if power-on occurred. While there is no guarantee from Mitsubishi that memory on the microcontroller will lose data quickly after power is off, so far it has performed reliably in all of the prototype systems.