HEWLETT-PACKARD CO.

NOTE: This page provides a running history of changes for a multi-page drawing which cannot conveniently be re-issued completely after each change. When making a change, list for each page all before-and-after numbers (within reason; use judgement, and use "extensive" revision note if loss of past history is tolerable, or retype complete page) and associate with each a symbol made up of the change letter and a serial subscript to appear here and on the page involved (there enclosed in a circle, triangle, or other attention-getting outline).

Ltr	REVISIONS	DATE	INITIALS
A	As issued per PC40-11541	11-13-81	B. Hunton

Model No. 9826A/9836A	Stock No.
Title THEORY OF OPERATION	
Description	Date 11-12-81
By Rob Horning	Sheet No. 1 of 47
Supersedes A-09826-66501-9	Drawing No. A-09826-90315-1

THEORY OF OPERATION

09826-66501 Mother Board

09826-66502 Mother Board

I. INTRODUCTION

This document is a theory of operation for the 9826 and 9836 mother boards. It also contains the IRS's for the motherboard. It contains the sections listed below.

1) Section II is the theory of operation for the keyboard/ real time clock electronics.

2) Section III is the theory of operation for the key board/real time clock 8041 firmware.

3) Section IV is the IRS for the keyboard and the real time clock.

4) Section V is the theoty of operation for the internal HPIB electonics.

5) Section VI is the IRS for the internal HPIB.

6) Section VII is the theory of operation for everything not covered in the other sections.

7) Section VIII is a list of all the test points on the mother board.

II. THE ELECTRONICS

A) THE CLOCK

The clock for the 8041 is provided by a 10 MHz canned crystal oscillator(U13). The clock is gated through U9 and U10 so that the automatic tester can over ride the 10 MHz clock with a lower frequency. This clock is the time base for the floppy control board, the CRT in the 9826, HPIB, and the keyboard/real time clock. The keyboard/real time clock and HPIB need 5 MHz and get it by divding the 10 MHz clock by 2 with part of a 74LS175 (U6).

The accuracy of the clock is 50 ppm which gives the real time clock an accuracy of about 130 seconds per month. The 8041 needs clock and clock inverted. The delayed signal should go to pin 3 of U11 and the other to pin 2. There are 1K pull up resistors on the clock lines to insure that the high level meets the spec of the 8041 of 3.8 volts.

B) RESET

Reset going away must be syncronized with the 8041 SYNC line (pin 11). Reset must be low for at least five cycles of SYNC. Both of these conditions are met by using a 74LS164 shift register (U14). The shift register is cleared by the system RESET signal and SYNC is used to shift in ones. The output of the shift register goes to the reset line on the 8041.

The 330 Ohm resistor is in the reset line so that the automatic tester can reset some of the circuit without resetting other parts of it.

C) CHIP SELECT

The chip selects for the keyboard, HPIB, the CRT, the floppy control board ,graphics, and the battery option are all generated by a 74S164 (U20). This is a schottky part because the floppy control board, HPIB and the CRT all need fast chip selects in order to have acceptable timing margins.

The keyboard/real time clock chip select is latched into a 74LS74 (U7) and is cleared by the systen IOR signal going away. The system timing does not spec ICR to go away before the chip select and the 8041 spec says that this must happen and so the

chip select must be latched.

D) SCANNING THE KEYS

The keyboard is scanned by using a 74159 (U24) open collector demultiplexer and a 74LS151 (U25) data selector. A seven bit key address is output from the 8041 (U11) on lines 27 through 33. The upper four bits of the address go to the four to sixteen line demultiplexer. This part is open collector because it is possible to short the outputs together by pressing three or more keys at the same time. Thirteen of the outputs are used as the column address lines of the keyboard matrix. The eight row address bits of the keyboard matrix go into the data selector. The lower three bits of the address output by the 8041 selects one of the eight rows to be scanned. The output of the data selector goes to a flag input on the 8041 (pin 1). For information on which address goes with which key, see HP drawing 3101-2447-6.

The shift and the control keys go directly into the 8041 (pins 21 and 22). This is because the keyboard can get ghost keys if more than two keys are pressed at once. Since shift and control are normally pressed with another key it is necessary to take them out of the matrix.

The three outputs of the demultiplexer that are not used to scan keys are used for other things. The C0 output is used to scan the language jumper. The C3 output is used to scan the configuration jumper. The C1 output is used to clear the RPG flag.

E) THE RPG

When the RPG is turned it will pulse the signal lines 120 times per revolution. The pulses on the two lines will be about 90 degrees out of phase. If the rotation is clockwise the RPG B line will lead the RPG A line and the other way around for counter clockwise.

The pulses are latched into a 74LS74 (U23) positive edge triggered duel flip-flop. The edges of RPG A latch a one into one of the flip-flops and the state of RPG B into the other flip-flop. The output of the first flip-flop goes to a flag input on the 8041 (pin 39) and is used to sense when the RPG has

been turned. The output of the second flip-flop goes to a port input on the 8041 (pin 24) and is used to tell what direction the RPG was turned. The RPG flip-flops are cleared by pulsing C1 of the keyboard scanning demultiplexer.

Both RPG lines are low pass filtered with a capacitor to ground and a pull up resistor because the length of the RPG lines makes then susceptible to electrical interference. When 1000 volt transients were applied to the power line false signals came from the RPG if the low pass filter was not in.

F) THE BEEP

The beep frequency is latched into a 74LS174 (U12) six bit latch when the 8041 pulses pin 34. The output of the latch goes to a 7497 (U17) rate multiplier. The clock input of the rate multiplier is the SYNC output of the 8041 (pin 11). This signal is a fixed 333.333 KHz. The output of the rate multplier is very asymmetrical and still too high of frequency for the speaker. A 74LS393 (U22) dual four bit counter is used to bring the frequency down to the desired range and also to make the signal symmetrical. The counter that goes to the speaker driver is cleared when the frequency is loaded so that the driver will be in the off state when the speaker is turned off. The speaker is turned off by loading a frequency of zero.

The speaker is driven by an open collector inverter with a transistor added to increase the current drive. The diode CR1 acts as a catch diode for the inductance of the speaker when the transistor turns off. A 47 Ohm resistor is used to limit the current through the speaker and the drive transistor. This must be a half watt resistor.

Both J1 and J2 must be in place for the speaker to work. When the speech option is added these jumpers are removed and replaced by a ten pin ribbon cable connector. It is up to the speach option to see to it that the beep from the keyboard gets to the speaker when the speech board is plugged in.

G) INTERRUPTS

The keyboard/real time clock will interrupt on either level 1 or level 7 depending on the type of interrupt. It is possible that both of these interrupts could be shared and so

they must be open collector.

The level 1 interrupt is generated automatically by the 8041 as a high level and thus it must be inverted.

The level 7 interrupt must be noninverted. (It is actually inverted twice). The reason it cannot be inverted by the 8041 firmware is because when the 8041 reset line is pulled the line automatically goes high. This would cause the 68000 to get an unmaskable interrupt whenever it did a reset command.

After being inverted once the level 7 interrupt goes to a 74LS257 (U16) data selector. This selector is used to look at two of the HPIB status registers. However, it had an unused bit and so this was used to add the capability of telling if the keyboard/real time clock is interrupting on level 7.

A bit in the 8041 status register tells if the keyboard/ real time clock is interrupting on level 1.

H) MISCELLANEOUS

Pin 33 of the 8041 is pulled low. The 8041 uses this signal to tell if it is in a 9826 or a REDWOOD controller. If this pin is high it will think that it is in a REDWOOD controller.

All the ports (not flags) that are used as inputs (pins 21, 22, 23 and 24) on the 8041 must have a 1K series resistor in the line to current limit the signal coming in. The 8041 will sometimes drive these lines even though they are inputs.

XII. THE 8041 FIRMWARE

A) INTRODUCTION

The following section is the theory of operation for the keyboard/real time clock 8041 firmware. It would be useful to have a copy of the flow chart for the code (1820-2564-3) and a copy of the block diagram (1820-2564).

B) THE MAIN LOOP

i) Introduction - Program control normally resides in a loop that does the following things.

1) Check the flags to see if the 8041 is supposed to give the 68000 some kind of interrupt.

2) If 10 msec. is up go to the timers update routine.

3) Check to see if the RPG has been turned

4) Scan one key.

5) Increament the key pointer.

ii) Interrupt check - The 8041 can interrupt the 68000 for the following reasons.

1) A key or RPG input. (level 1)

2) The reset key was hit. (level 7)

3) A user timer interrupt. (level 1)

4) A system 10 msec. interrupt. (level 1)

5) A fast hand snake time out. (level 7)

All the interrupts are maskable seperately. Before interrupting the mask for the type of interrupt is checked. The output buffer full flag is also checked (when the interrupt is on level 1). If it is full it means that the 68000 has not

responded to a previous interrupt yet and a new interrupt would over write the old one. The 8041 interrupts the 68000 by writing to the output buffer. This happens automatically. The interrupt that comes from the 68000 to the 8041 is disabled while the 8041 checks the interrupt mask so that the 8041 can not have the mask changed between the time the mask is checked and the time that the interrupt takes place.

when the 8041 interrupts the 68000 the upper four bits of the status register tells what type of interrupt it is. The data buffer will contain additional data needed. (Like the key code and what timer interrupted.)

The 8041 will automatically stop interrupting when the output buffer is read by the 68000.

It is possible to tell if the keyboard/real time clock is the device pulling on interrupt level 7 by checking a bit in the HPIB status regster. A bit in the 8041 status register will tell if it is a reset key interrupt or a fast hand shake interrupt.

iii) 10 msec. up - Ever 10 msec. the 8041 timer interrupt routine will set a flag saying that 10 msec. is up. The main loop detects this flag, clears it and calls a routine to increment all the timers. The interrupts from the 68000 are disabled during most of the timers update routine for two reasons

1) The 68000 may ask for a timer while it is in the middle of being up dated and get a wrong result.

2) The 68000 may change a bit in a status register while the 8041 is changing a bit in the same status register. This could cause one of the changes to be missed.

The timers update routine does the following things:

1) Increament the real time (10 msec. from midnight and days from january 1).

2) Increament and test all the real time clock timers. If one of the timers is up a flag is set telling the main loop to interrupt. The folling timers exist.

a) Match current real time. (user)

b) Interupt after a given delay. (user)

c) Interrupt periodically.(user)

d) Interrupt ever 10 msec. (system)

e) Interrupt on level 7 after delay (system FHS)

3) If there is currently a beep increment the beep timer and check to see if the time is up. Turn the beep off if the time is up or if it was already supposed to be off.

4) If the RPG has been turned increment the RPG timer. If the timer is up reset the timer and set a flag telling the main loop to give an RPG interrupt.

5) Check to see if the main loop has detected that a key is down. If it has check to see if this is the first time in the timer update routine since the key was detected. If it is the first time load the repeat couter with the repeat delay(as opposed to the repeat rate) and reset the debounce counter. If there is a key detected check to see if the key is still down. If the key is still down reset the debounce counter and increment the repeat counter. If the repeat counter overflows set the auto-repeat flag and reset the repeat counter. If the key is up increment the debounce counter. If the debounce time is up check to see if there is a waiting key and if there is make it the current key, set the auto-repeat flag (This causes the waiting key to be output immediately.), reset the repeat counter to the delay, and clear the waiting key buffer. If there is no waiting key just clear the current key buffer.

6) If the reset key has been depressed start the debounce. The main loop will clear the debounce counter if the key remains down.

iv) RPG turned - If the RPG has been turned a counter is incremented or decremented depending the the direction that it was turned. The counter is check to make sure that it does not overflow or under flow. The RPG flip-flops are cleared by outputing hex 08 to port 1.

iiv) Key down - If a key is detected to be dowwn it is first checked to see if it is the reset key (shift PAUSE). If it is the reset key and the reset key is enabled there is a 1.5 msec. wait before the 8041 pulls on interrupt level 7. This wait is to allow the 68000 to respond to any previous fast handshake interrupt before giving another level 7 interrupt. Only one level 7 interrupt will occure for each time the key is pushed down even if the keyboard is reset. The key does not auto repeat. After pulling on NMI control returns to the main loop. If reset is disabled nothing happens.

If the key that is down is not reset a flag is checked to see if a key is already down. If there was a key down it is checked to see if the key currently being checked is the same that was down. If it is the same key control returns to the main loop. If it is not the same key the saved key buffer is checked and if there is no saved key the key that is currently down is saved. If there is a saved key the key is disregarded and control returns to the main loop.

If there was no key down the key that has just been detected is saved as the key down. The output buffer full flag is checked to see if the key can be sent to the 68000 causing an interrupt on level 1. If the buffer is empty and the keyboard interrupt is enabled the key is output (causing a level 1 interrupt). If the key cannot be output because the keyboard is disabled or the ouput buffer is full the auto-repeat flag is set so that the key will be output as soon as possible.

vi) Scan done - When all the keys have been scanned the following things happen.

1) The key address is set back to hex 18.

2) Shift and control keys are input and flags set to there condition.

3) The auto-repeat flag is checked and it is set an attempt is made to output the current key.

C) RESET

When the 8041 is reset the following things happen.

1) Any current interrupt is cleared. (While the 8041 is

being reset it will be giving a false interrupt on level 1)

2) All types of interrupts are masked.

3) Any current key (except reset) and any roll over key is cleared.

4) The 8041 is put in the mode that causes it to automaticly interrupt the 68000 when the 8041 writes into the output buffer.

5) A check sum is done on the 8041 ROM.

6) The language jumper and the configuration jumper are read from the key switch board.

7) The reset key is assumed down and must be debounced. This prevents getting more than one level 7 interrupt for each time the reset key is hit.

If the check sum comes out wrong or the jumpers give an invalid code steps 5 and 6 are repeated. An invalid jumper code is when more than one jumper is detected in one of the groups. It is possible to detect more than one jumper is down even if there is only one jumper in if there are two or more keys depressed when the the jumper is checked.

When everything checks out the 8041 interupts the 68000 on interrupt level 1.

D) TIMER INTERRUPT

The 8041 timer is set up to interrupt the main loop every 10 msec. The timer is an eight bit counter that is counted up every 32 instruction cycles (96 micr-seconds). It interrupts when the timer overflows to zero and continues to count.

It is possible that the timer interrupt will not be serviced within 32 instuction cycles and thus it would count past zero. To not miss any counts the time is read and added to the constant that is loaded into the timer. It is possible that the counter would count between the time that it is read and the time that it is loaded. To prevent this from happening the counter is continuely read until it does change. Then within the 32

instruction cycles the new value is calculated and loaded into the timer.

The timer counts ever 96 micro-seconds and this does not go int 10 msec. evenly. In order to keep the the real time from drifting it is nesassary to decrement the time being counted every sixth time the timer is loaded. what is done, is for five times the counter is loaded with 104 (less any counts missed) and the sixth time the counter is loaded with 105.

(104*96*5)+(105*96*1*)= 60000 micro-seconds

Thus over a 60 msec. period the time is exact. The most that the clock will be off on the short term due to this is 96 micro-seconds and this is insigneficant compared to other things that can cause the time to be off in the short term. (For example the timer interrupt can not be serviced while control is in the 68000 interrupt service routine. It must wait until the sevice routine is left.)

G) THE 68000 ISR

When ever the 68000 writes to the 8041 (location 42800x, x= 1 or 3) an interrupt occurs.. If the 8041 is in the timer ISR the interrupt will occur when it returns to the main loop.

Address line A1 is latched as a flag by the 8041. If A1 was a zero when the 8041 was writen to the program will treat the byte as data. When data is sent, a pointer has already been set up as a result of a previous command. This pointer is used to store the data. After the data is stored the pointer is checked to see if anything besides just storeing the data has to be done. The following cases require that something else be done.

1) If the data is the last byte of data for one of the realtime clock timers (match, delay, cycle or FHS) a flag is set telling the real time update routine that the particular timer is activated. If the data is the last byte of the cycle timer the three bytes of the timer must be stored so that the timer can be reset to the original value when it overflows.

2) If the data sent is one of the bytes setting the real time clock the byte is added in rather than just saved. This is done so that if it takes a long time to send all the bytes of the real time it will still keep up.

3) If the data sent is a new RPG rate the current RPG counter is reset to the new value. This is done because at power up the RPG rate is 2.56 seconds and when it is changed to a lower value the new value should take effect immediately.

4) If the data is the second byte of beep data the beep is started. (The beep time is the first byte sent and this timer is started when it is sent.) If there is already a beep when a new beep comes the new beep wipes out the old beep.

After data is sent to the 8041 the data pointer is incremented. This is why one command can be followed by more than one data byte. If more data is sent then is needed for a command it will cause something t be writen over.

When address line A1 was a one when the 8041 was writen to the byte sent will be treated as a command. Commands cannot be considered carried out until the input buffer full flag is cleared indicating that the 8041 read the command.

If the command is one that is going to be followed by data the lower bits of the command are stored as the data pointer. When the some of the commands requiring tc be followed by data are sent some other action must aso be taken rather than just setting up the data pointer. These cases are listed below.

1) When the command to set up to input the real time is sent the real time is set to zero. This is so that the time can continue to be counted while the 68000 is sending the data bytes. The days are not cleared because it is possible to change the time without changing the days.

2) When a command comes to set up a real time match, a delay or a cycle interrupt the current real time function is canceled. This makes if possible to write over a real time function without canceling or getting the interrupt. Real time clock interrupts are canceled by sending the command to set up a new interrupt and not following it with any data.

3) If the command is to set up to input a fast hand shake interrupt not only is any current fast hand shake interrupt cleared but if the 8041 is pulling on interrupt level 7 it will stop pulling on it. This is the only way to make the 8041 stop interrupting on level 7 other than doing a reset command.

If the command sent is to change the interrupt mask the lower bits of the command are used as the new interrupt mask. This allows the interrupt mask to be changed whithout following with any data.

If the command is a command to load the timer output buffer with something the lower bits of the command are used as a pointer to were to start loading the buffer. This allows the delay, cycle, match or fast hand shake times to be read as well as the real time. The timer output buffer is needed because if as the time was being read it was updated and a carry propagated through the read result would be wrong. The timers cannot be updated while the buffer is being transfered because the 68000 interrupt routine cannot interrupt the timer routine and visa virsa.

If the command is to send a byte of data to the 68000 the lower bytes of the command are used as a pointer to the byte to be sent. This pointer is saved and a flag is set telling the main loop that the 68000 has asked that a byte be sent. The byte is not sent in the interrupt routine because it is possible that the main loop is or has sent data to the 68000 and the interrupt service routine would write over this data befor he 68000 got it. when ever data is sent to the 68000 an interrupt will occure.

Before leaving the 68000 interrupt service routine the 68000 interrupt is disabled. It is re-enabled at various places in the main loop and in the timer update routine. This keeps the 68000 from compleatly tieing up the 8041 and causing it mis updating the real time and scanning the keyboard.

H) KEY DEBOUNCE

The keyboard is debounced in the 8041. The edge of the key going down is only debounced for .1 msec. This is to prevent electical noise from causing a false key. The mechanical bounce of the keys is debounced on the key being released. A key is not considered gone until it has been gone for three interrupts from the 10 msec. timer.

IV. KEYBOARD/REAL TIME CLOCK IRS

A) THE IRS

The keyboard and real time clock are both controled by the same processor, an Intel 8041A. Real time is kept as msec. since midnight and days since January 1. There are three special purpose timers that the user can program to generate an interrupt. First, there is the on time match interrupt (this does not match on days). Next there is a delayed interrupt that is programmable between 10 msec and 1.94 days(3 bytes). Last there is a cycled interrupt (periodic) that has the same range as the delayed interrupt.

There are two system functions provided by the real time clock. A maskable 10 msec periodic interrupt and a programmable system time-out that generates an NMI. All functions of the real time clock have 10 msec resolution.

The CHIPMUNK keyboard has two key roll-over and has a programmable auto-repeat feature.Debounce is done on keys going away and so the keyboard is very fast. The keyboard is scanned every 10 to 15 msec, depending on how busy the 8041A is. The RPG is scanned about 100 times as fast as the keyboard. The beep can be programmed to frequencies between 81.38 and 5208 Hz with 81.38 Hz resolution. The duration can be from 10 to 2560 msec

The REDWOOD keyboard does not have either two key roll-over or an auto-repeat feature. The REDWOOD keyboard is similar to an RPN calculator; a detailed description is presented in the REDWOOD Keyboard/Display Definition.

The 8041A will normally interrupt on level 1. When an interrupt occurs, the status register should be read and then the data buffer should be read. The data buffer must be read even if the status register indicates that it contains no useful data. The 8041A status is read by reading HEX memory location 428003 for CHIPMUNK and 420003 for REDWOOD. The data buffer is read by reading memory location 428001 for CHIPMUNK and 420001 for REDWOOD. Commands are written to the 8041A by writing to memory location 428003 for CHIPMUNK and 420003 for REDWOOD. Data is written to the 8041A by writing to memory location 428001 for CHIPMUNK and 420001 for REDWOOD.

The status register may be read at any time. However, when the data register is read the keyboard interrupt is cleared and the keyboard may write in new data.

In the case of an interrupt request from the 8041A the most significant four bits of the status register are used to define the type of interrupt. In the case of an NMI bit 2 of the status register indicates the type of NMI. The following defines the status register contents:

0000XXXX - Not used

0001XXXX - The interrupt is a 10 msec periodic interrupt.

0010XXXX - The interrupt is from one of the special purpose timers.

0011XXXX - There is both a special purpose timer interrupt and a 10msec periodic interrupt.

0100XXXX - The data buffer contains a byte of data that the 68000 has requested.

0101XXXX - Not used

0110XXXX - Not used

0111XXXX - Power-up reset was completed successfully.

1000XXXX - The data buffer contains a key (both shift and control). This is unique to CHIPMUNK.

1001XXXX - The data buffer contains a key (only control). This is unique to CHIPMUNK.

1010XXXX - The data buffer contains a key (only shift).

1011XXXX - The data buffer contains a key (no shift or control).

1100XXXX - The data buffer contains an rpg count (both shift and control). This is unique to CHIPMUNK.

1101XXXX - The data buffer contains an rpg count (only control). This is unique to CHIPMUNK.

1110XXXX - The data buffer contains an rpg count.(only shift). This is unique to CHIPMUNK.

1111XXXX - The data buffer contains an rpg count.(no shift or control). This is unique to CHIPMUNK.

In the case of an NMI the 8041A status register must be read to determine the type of NMI; the following is the definition of the status register:

xxxxx1xx - The NMI is a fast-handshake time=out.
xxxxx0xx The NMI was generated to indicate that the Reset key on the front panel was pressed.

It is possable to tell if the keyboard/real time clock pulled on NMI by reading location 478005 and checking bit 2. If this bit is a one the keyboard is pulling on NMI.

For real time interrupts the data buffer is defined as:

bit 7- If set an on time match interrupt occurred.
bit 6- If set a delay interrupt occurred.
bit 5- If set a cycled interrupt occurred.
bits 4-0-If non-zero an error occurred.

This contains the binary number representing the number of cycle interrupts that were missed. (Any combination of bits 5, 6, or 7 can be set.)

For CHIPMUNK keyboard interrupts the data buffer contains a key matrix location. For RPG interrupts the data byte is a 2's complement number that will be negative for clockwise turns and positive for counterclockwise turns. The number is the number of pulses since the last RPG interrupt. This count will not overflow. It will stop at the maximum negative or positive count.

For REDWOOD keyboard interrupts the data buffer contains an 8-bit keycode;the following defines the keycodes:

UDK1 or A/ST - 00001000
UDK2 - 00000010
UDK3 or CONT - 00000100
PSE or RUN - 00010000

Before a command or data can be written to the 8041A the status register must be read to determine if the IBF flag, bit 1, is zero. If bit 1 is zero the 8041A can be addressed; however, if bit 1 is one the transfer cannot take place. A list and description of the commands will now be presented:

3E - Load the timer output buffer with the cycle interrupt time. The first three bytes of the timer output buffer will contain the current value of the cycle timer. The LSB will be in the first byte of the buffer and the MSB in the third byte.

3B - Load the timer output buffer with the delayed interrupt time. The first three bytes of the timer output buffer will contain the current value of the delay timer. The LSB will be in the first byte of the the buffer and the MSB in third byte.

38 - Load the timer output buffer with the on time match value. The first three bytes of the timer output buffer will contain the match value. The LSB will be in the first byte of the buffer and the MSB in the third byte.

36 - Load the timer output buffer with the fast handshake time. The first two bytes of the timer output buffer will contain the current value of the fast handshake timer. The LSB will be in the first byte of the buffer and the MSB in the second byte.

31 - Load the timer output buffer with the real time. The first three bytes will contain the number of 10 mseconds since midnight. The next two bytes contain the number of days since time was set. The LSB of the time will be in the first byte of the buffer and the MSB will be in the third byte. The LSB of the day will be in the fourth byte of the buffer and the MSB in the fifth byte. Both the days and the 10 msec are given as positive true binary numbers.

13 - Load the data buffer with the first byte of the timer output buffer. This will cause an interrupt.

14 - Load the data buffer with the second byte of the timer output buffer. This will cause an interrupt.

15 - Load the data buffer with the third byte of the timer output buffer. This will cause an interrupt.

16 - Load the data buffer with the fourth byte of the timer output buffer. This will cause an interrupt.

17 - Load the data buffer with the fifth byte of the timer output buffer. This will cause an interrupt.

11 - Load the data buffer with the configuration code. This will cause an interrupt.

12 - Load the data buffer with the language code. This will cause an interrupt.

04 - Load the data buffer with the interrupt mask. This will cause an interrupt.

A2 - Set up to input the repeat rate. This command is followed by sending the repeat rate to the 8041A. The number sent is the 2's complement of the number of 10 msec for the repeat rate. Sending zero means do do not repeat.

A0 - Set up to input the delay to start repeating. This command is followed by sending the delay. The number sent is the 2's complement of the required delay.This is unique to Chimunk.

A6 - Set up to input the rate at which the RPG can interrupt. This command is followed by sending the rate. The number sent is the number of 10 msec

A3 - Set up to input bell information. This command is followed by sending the duration (2's complement of the number of 10 msec) and then sending the frequency (f=81.38*number sent).This is unique to CHIPMUNK. The upper two bits of the frequency must be zero. If a beep command is sent while an old beep is still going the new beep will cancel the old beep. The old beep will continue until both data bytes are recieved for the new beep.

AD - Set up to input the 10 msec real time. This command is followed by three data bytes;which are sent LSB first and are the number of 10 ms since midnight. Time is counted from the time this command is recieved. The day count may be sent following the time data without sending the AF command.

AF - Set up to input the days real time. This command is followed by two data bytes that are sent LSB first.

B2 - Set up to input a fast handshake interrupt. This command is followed by two bytes, LSB first, that are the 2's complement of the number of 10 msec to wait until interrupting. This interrupt will be on the NMI line.If this command is not followed by 2 bytes of data the fast handshake interrupt will be cancelled. Sending this command will cancel any current NMI. If shift PAUSE (reset) is hit and reset is enabled any fast handshake interrupt will be canceled.

B4 - Set up to input a real time match interrupt. This command is followed by three bytes (LSB first) that are the number of 10 mseconds since midnight to match on. If this command is not follwoed by three bytes of data the real time match interrupt will be cancelled. Any real time match interrupt should be canceled when the real time is about to be changed. If half way through changing the real time it matches an interrupt will occure.

B7 - Set up to input a delayed interrupt. This command is followed by three bytes, LSB first, which are the 2's complement of the number of 10 msec to wait until interrupting. Time is counted after the last data byte is recieved.If this command is not followed by three bytes of data the interrupt will be cancelled.

BA - Set up to input a cycled interrupt. This command is followed by three bytes LSB first that are the 2's complement of the cycle time. Time is counted after the third byte is received.If this command is not followed by three bytes of data the interrupt will be cancelled.

010xxxxx - When the upper three bits of the command are 010 the lower five are the interrupt mask. The lower five bits are described below. (Masked means that the interrupt is disabled) A one means to disable the interrupt, and a zero means to enable the interrupt.

bit 0 - Mask keyboard and rpg
bit 1 - Mask the reset key
bit 2 - Mask the timer interrupts.
bit 3 - Mask the periodic system interrupt.
bit 4 - Mask the fast handshake interrupt.

When data is asked for an interrupt will occur. This interrupt cannot be masked.

When a command is sent to the keyboard/real time clock it cannot be considered carried out until the input buffer full flag (bit 1 of the status register) has been cleared. If the code following a command masking an interrupt or turning of one of the timers depends on not getting an interrupt from the masked or canceled function the IBF flag should be checked before going on.

Reseting the 8041A will not cause it to lose real time,the repeat rate,or the rate that the rpg can interrupt. It will be like a power up reset in all other ways.

At power up the 8041A comes up in the following state.

1) The auto repeat is random.
2) The rpg rate is random.
3) The real time is random.
4) All the 8041A interrups are masked

At power up the 8041A will do a check sum on it's ROM. It also checks the language code and the configuration code. If it gets the wrong check sum or an invalid language or configuration code it will keep trying until it gets it right. With REDWOOD there are no invalid configuration codes. When it does get it right it will interrupt on level 1. It will send 7 in the upper half of the status register and the data register will contain an 8E.

When the keyboard is reset it will pull on interrupt level 1 and will continue to pull on it until about 20 micro-seconds after reset is released. This interrupt should not be serviced. Level 1 interrupts should not be enabled during a reset command and for 20 micro-seconds after the reset command. The keyboard status register will not indicate that the keyboard is interrupting at this time.

The language or configuration code may be read at any time. For CHIPMUNK the languge and congiguration codes have the following meanings.

For the language jumper:
- 0-standard
- 1-French
- 2-German
- 3-Swedish/Finnish
- 4-Spanish
- 5-Katakana
- 6-J9
- 7-J10
- 8-J11

For the congiguration jumper:
- 0-no jumper
- 1-J1
- 2-J2
- 3-J3
- 4-J4
- 5-J5
- 6-J6
- 7-J7
- 8-J8

In order to be sure that the the 8041A does not lose real time when the 68000 does a reset command the following must be done.

1. Send command 31 to the 8041A. As usual do not send the the command until the IBF flag is clear.
2. Wait for the command to be taken by checking the IBF flag(bit 1 of the status register must be zero).
3. Do the reset command within 100 micro-seconds.

The keyboard/real time clock may be used as a polled device rather than an interrupting device by checking bit zero of the status register. When this bit is one it means that the 8041A is interrupting.

B) OTHER CAPABILITIES

The following section discusses in detail how information is read from the keyboard/real time clock. All of the commands that involve getting information from the keyboard/real time clock that were discussed in the previous section are a subset of just two commands. Using these two commands allows any byte in the 8041A memory location (64 bytes total) to be accessed.

The lower 32 bytes of the 8041A memory can be accessed by sending the command below.

000xxxxx

Were xxxxx is the address to be accessed. The upper 32 bytes of memory must be accessed inderectly by use of the timer output buffer. The command below will load the five bytes of the timer output buffer with five bytes of upper memory.

001xxxxx

Were 1xxxxx is the highest location to be loaded. (The lowest location is 1xxxxx-4.)

The use of every memory location is listed below.

0 - scratch

1 - scratch

2 - This is a set of flags listed below.
bits 0-2 - Used for debounce.
bit 3 - When one the fast hand shake timer is being used.
bit 4 - When one the cycle timer is being used.
bit 5 - When one the delay timer is being used.
bit 6 - When one the match is being used.
bit 7 - When one the bell is on.

3 - The key switch currently being scanned.

4 - This is a set of flags listed below.
- bit 0 - when one the keyboard/RPG is masked.
- bit 1 - When one the reset key is masked.
- bit 2 - When one the userer timers interrupts are masked.
- bit 3 - When one the PSI is masked.
- bit 4 - When one the fast hand shake interrupt is masked.
- bit 5 - When one it is time to auto-repeat.
- bit 6,7 - not used

5 - This is a set of flags listed below.
- bit 0 - When zero the shift key is down.
- bit 1 - When zero the control key is down.
- bit 2 - When one it is time to do a PSI.
- bit 3 - When one it is time to do a user timer interrupt.
- bit 4 - When one it is time to do an RPG interrupt.
- bit 5 - When one it is time to send the 68000 something it asked for.
- bit 6 - not used
- bit 7 - When it is time to o fast hand shake interrupt.

6 - roll over key save buffer.

7 - The current key that is down.

8-F - The 8041A stack.

10 - Reset debounce counter.

11 - Configuration jumper code.

12 - Language jumper code.

13 - First byte of timer output buffer.

14 - Second byte of timer cutput buffer.

15 - Third byte of the timer output buffer.

16 - Fourth byte of the timer output buffer.

17 - Fifth byte of the timer output buffer.

18 - scratch

19 - scratch

1A - scratch

1B - Timer status (The bits are listed below.)
bits 0-4 - The number of cycle interrupts missed.
bit 5 - When one a cycle is up
bit 6 - When one a delay is up.
bit 7 - When one there was a real time match.

1C - The current RPG count.

1D - The location to put data sent by the 68000.

1E - The location to send the 68000.

1F - Six counter for 8041A timer interupt.

20 - The time to wait to start auto-repeating.

21 - Auto-repeat timer.

22 - Auto-repeat rate.

23 - Beep frequency.

24 - Beep timer (counts up to zero).

25 - RPG timer.

26 - RPG interrupt rate.

27 - If bit 6 is a one a 8041A timer interrupt occured.

28-2C - not used

2D - LSB of time of day

2E - Second byte of time of day.

2F - Third byte of time of day.

30 - LSB of days.

31 - Second byte of days.

32 - LSB of fast hand shake timer.

33 - Second byte of fast hand shake timer.

34 - LSB of real time match.

35 - Second byte of real time match.

36 - Third byte of real time match.

37 - LSB of delay timer.

38 - Second byte of delay timer.

39 - Third byte of delay timer.

3A - LSB of cycle timer.

3B - Second byte of cycle timer.

3C - Third byte of cycle timer.

3D - LSB of cycle timer save.

3E - second byte of cycle timer save.

3F - Third byte of cycle timer save.

As an example of using these commands if it was nessasary to find out if shift and/or control were down the command 05 should be sent and the 8041A will send the byte containing the flags for these keys.

As an example of reading one of the bytes in upper memory if it were desired to find out how much time was left in a beep first send the command 24. This puts the memory locations 20 through 24 into the timer output buffer. The fourth location will have 23 which is the time left on the beep. To have this sent by the 8041A send the command 16.

As usual care must be taken if more than one resousce is talking to the keyboard/real time clock. For example a level 7 interrupt sevice routine might be trapped to while something else is reading the real time. If care is not taken it would be possible to have the level 7 ISR recieve real time data when it expected shift/control data.

C) BLACK BOX DESCRIPTION

This section was writen by Mark Allen.

The following section attempts to give more detail about the actual operation of the keyboard 8041, by defining each subsystem as a "black box".

REAL TIME CLOCK Functions:

Set time of day and date:

Begins counting time from the time the 8041 begins executing the command. 3 or 5 bytes, LS byte first. Invalid times (>= 24 hours) will be accepted, but within 10 minutes will be converted to valid times by a rollover check.

Maintain time of day:

Update 5 bytes; when 3 LS bytes = 23 hr, 59 min, 59.99 sec, increment top 2 bytes. Top bytes roll over to 0. 3 LS bytes may wake up with invalid time (>= 24 hrs), but will become valid within 10 min. At every RESET operation, may lose up to 10 mSec due to clearing the 8041's counter/timer or missing an interrupt while executing the Reset routine. (Hopefully, this will be prevented by only pulling on the Reset line while the 8041 is busy executing a dummy Sample Real Time command.)

Sample time of day:

Copies 5 bytes into timer output buffer.

Read timer output buffer:

5 separate commands; each causes one byte to be transmitted.

Setup real time match:

3 bytes (i.e. 24 hours), self-canceling. Sending an invalid time (i.e. >= 24 hours) will result in no match (unless clock is also set to an invalid number). The 8041 will begin checking for the match after it has received the 3rd byte. Warning: if the user sets the time while a realtime match is active, he may get spurious match interrupts.

Cancel real time match:

Can be sent at any time. Erases any pending (but masked) interrupt. Does not release NMI if it is already asserted as the result of a RT match. After this command is accepted (i.e. IBF goes false), no interrupt will be generated as a result of RT match, regardless of whether RT match is masked and/or logged at the time of accepting the command.

Generate RT match interrupt:

Rides along with the next PSI (if PSI is enabled), or is generated by itself.

Setup delayed interrupt:

3 bytes, LS byte first. Immediately cancels delayed interrupt; will begin counting time after receipt of 3rd byte. 0 = longest; all 1's = shortest (could be anywhere between 0 and 10 mSec actual delay).

Copy delay into output buffer:

Samples current value of timer.

Generate delayed interrupt:

Rides along with next PSI (if enabled), or happens by itself.

Cancel delayed interrupt:

Can be sent at any time. Erases any pending but masked interrupt (etc.--see Cancel R1 Match).

Setup cyclic interrupt, copy cyclic timer, cancel cyclic interrupt:

Analogous to Delayed interrupt commands.

Generate cyclic interrupt:

If none is pending, log a normal interrupt; if pending, increment the counter (saturating at 31).

BEEP functions:

Beep:

2 bytes: duration (0 = 2560 mSec, 11111111 = 10 mSec); frequency (00xxxxxx, where 00000000 is no beep (can be used to cancel a beep); 00111111 is highest frequency). Beep will start after receipt of 2nd byte. At power-up or reset, beep will be shut off regardless of selftest pass/fail. Actual beep frequency will be 81.38 times the number sent. (Hz).

FHS TIMEOUT functions:

Setup FHS timeout:

2 bytes, LSB first, 0 = 655360 mSec; 11111111 11111111 = 10 mSec. Sending the command cancels any pending FHS timeout immediately. Begins counting time after receipt of 2nd byte.

Cancel FHS timeout:

Analogous to Cancel delay. Also, releases NMI if it was asserted. (68000 must wait ____ micrcsec to be sure NMI is released.)

Generate FHS timeout interrupt:

Set flag bit in status reg, saying "not RESET key". (This flag bit won't be overwritten even if RESET is pressed before the 68000 reads the status reg, because the RESET key response is artificially delayed for 1.6 mSec from when the key goes down.) Assert NMI. NMI will remain asserted until either a Reset or a Cancel FHS timeout command.

KEYBOARD functions:

Keycodes:

102 keys. Each key is transmitted as a key matrix location byte, with bits for SHIFT and CONTROL in the status byte.

Scanning:

2-key rollover. 20 mSec trailing-edge debounce. Scan rate 10 mSec typical, 25 mSec worst-case.

Interrupts:

Key interrupts occur: when key is detected, if no other key is down; when other key goes up; when output buffer is empty. One key going down results in one interrupt, or none, but never two unless autorepeat is invoked. This includes RESET. Normal keys interrupt on level 1; RESET interrupts on level 7. Could miss a key if the key is held down for less than 20 mSec and, simultaneously, OBF lasts longer than 22 mSec (i.e. the debounce time plus a little). Keystrokes are masked by Key/RPG mask bit; they will be forgotten if they occur while the mask bit is set.

Auto-repeat:

All keys auto-repeat except RESET, SHIFT, CONTROL. Repeating starts after A Given Key has been down longer than the specified delay.

Setup auto-repeat rate:

One byte; 0 = no repeat; 1 = 2550 mSec; 11111111 = 10 mSec.

Setup auto-repeat delay:

One byte; 0 = 2560 mSec; 11111111 = 10 mSec.

RPG functions:

Scanning:

Scanned every 100 microsec normally; 200 microsec when a key is detected; 500 microsec when a 10 mSec interrupt occurs. When a pulse is detected, increment or decrement the counter, saturating at -128, +127. Clockwise rotation produces negative numbers, CCW positive. If Key/RPG interrupts are not masked and if the specified RPG period has elapsed, generate an interrupt (which will probably follow on the heels of a PSI, since the time period check is made within the 10 mSec service routine). If masked out, update the counter and log the interrupt, but don't generate one. Also update the counter even if interrupts are missed due to OBF.

Setup RPG interrupts:

One byte; 0 = 2560 mSec; 1 = 10 mSec; 11111111 = 2550 mSec.

Read RPG counter:

Output current value of counter. (Note that there is no way to clear the counter, and it will saturate at -128/+127.)

GENERAL functions:

Generate PSI (Periodic System Interrupt):

Interrupt every 10 mSec when not masked. Interrupts will occasionally be accompanied by RTC interrupts (i.e. bits in the status reg will be set). It is possable to get one more PSI after masking it. The jitter in the PSI interrupt could be as much as 2 msec. (This would require that all the timers the beep, the auto repeat etc. all over flow at the same time. Normally the jitter sould be about .1 msec.

Mask interrupts:

Cmd accompanied by one byte; there are bits for key/RPG, RESET key, match/delay/cycle, PSI, and FHS timeout. Sending a 1 disables the associated interrupt.

Establish defaults:

When the 8041's Reset line is pulled, the following defaults will be establisned:

* All RTC functions and bell OFF
* Cancel pending interrupts
* Mask all interrupts
* Cancel any pending RPG pulses
* Cancel any saved keystrokes
* Int 1 will be asserted for about 20 micro-seconds after reset goes away.

V. INTERNAL HP-IB THEORY OF OPERATION

A) 9914 TO HP-IB INTERFACE

THE HP-IB DATA BUFFER

The 9914(U3), 75162(U4) and the 75160(U8) form the heart of internal HP-IB and were specifically designed to work together by TI. The HP-IB buffers(U4,U8) are controlled directly by the 9914(U3) through the TE and CONT' lines(pins 21,30). The 75160(U8) is the HP-IB data buffer and has special bus drivers and terminators to match the HP-IB bus specifications. Control of the buffer is done through the TE(pin 1) input and is simply a direction control for the buffer. PULL-UP ENABLE(PE pin 11) controls the output buffers which can be either tri-state driver for normal operation or open-collector for parallel poll response. PE is generated by LS32(U1) as the logical AND of EOI and ATN, which is the condition for a parallel poll.

THE HP-IB CONTROL BUFFER

The 75162(U4) is the buffer for the HP-IB control lines and contains control logic since the 8 lines have a complex direction function. The IFC and REN lines are independent from the rest and are controlled by the SYS input. ATN and SRQ are directly controlled by the DC(pin 12) input since their direction is only a function of whether the 9914 is the controller in charge. DAV, NRFD, and NDAC are all controlled by the TE(pin 2) input since they are a function of the data direction. This only leaves the EOI line, and its controlled by both TE and DC and the level of ATN. Briefly, the 75162 drives EOI when: 1) the 9914 is the CIC and ATN is true, or 2) the 9914 is Talker Enabled and ATN is false. All the output buffers are tri-state with the exception of NRFD, NDAC and SRQ, as well as having bus terminating resistors. Also of note is that the buffers do not load the bus when they are powered-down.

75162(U4) CONTROL MAP

CONTROLS		DIRECTION OF BUFFER							9914 FUNCTION
		ATN							
TE	DC	LEVEL	DIRECTION	EOI	SRQ	NRFD	NDAC	DAV	
H	H	H	R	T	T	R	R	T	TALKER-NON CONTR
H	H	L	R	R	T	R	R	T	RESPOND P. POLL
H	L	X	T	T	R	R	R	T	TALKER-CIC
L	H	X	R	R	T	T	T	R	LISTEN-NON CONTR
L	L	H	T	R	R	T	T	R	LISTEN-CONTROLLER
L	L	L	T	T	R	T	T	R	CONDUCT P. POLL

SYS(PIN 1)	REN	IFC	
H	T	T	SYSTEM CONTROLLER
L	R	R	NOT SYSTEM CONTR.

T=TRANSMIT, R=RECEIVE, H=HIGH, L=LOW, X=DON'T CARE

B) 9914 TO 68000 BUS INTERFACE

THE TMS 9914 HP-IB INTERFACE ADAPTER

The most important chip in the chip-set is the 9914 which is a LSI MOS part that implements most of the IEEE-488 state diagrams, and makes a reasonably cheap high-speed interface possible. The chip is mainly an asynchronous part, but some timing is necessary for data settling delays, and command decoding so that a 5 MHZ clock is provided by LS175(U6). The processor side of the 9914 is totally asynchronous to the local 5 MHZ so that no synchronization is needed in the control lines. The reset line is provided by the LS164(U14) since it was already stretched for the 8041(U11). This line reset all functions of the 9914, and completely removes it from the HP-IB bus. The interrupt line from the 9914 is inverted by U19 and OC buffered by U18 before tying to interrupt level 3.

DATA BUFFERING

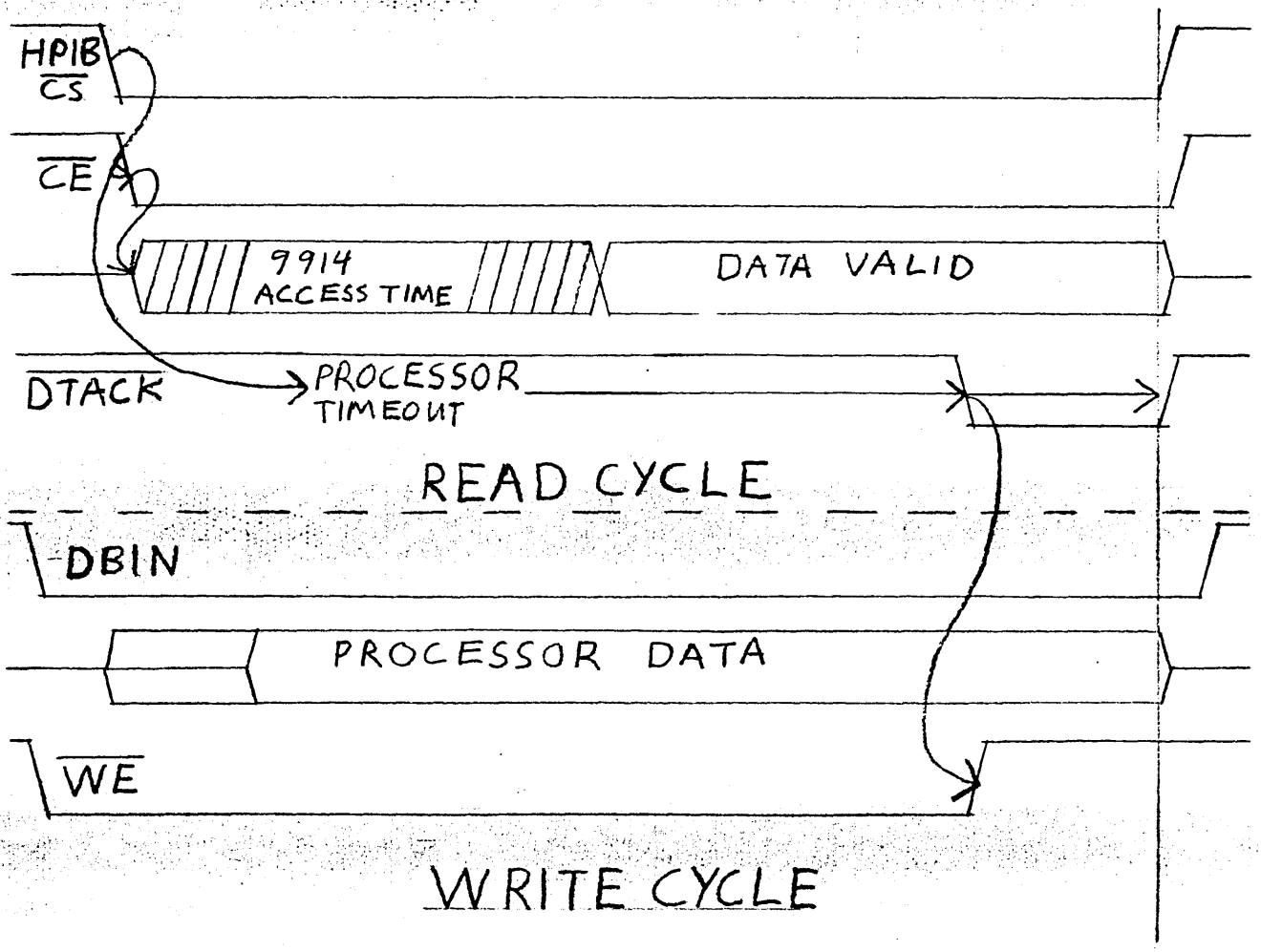
The data bus is buffered by U21 and tied directly to the 8 data lines on the 9914(pins 17-10). The data buffer is enabled by the inverted output of U15, which functions as a logical OR of KEYBOARD CS, HP-IB CS and HP-IB DMA acknowledge. The DIR(pin 1) input of U21 is controlled by the XOR gate U9, and has HP-IB DMACK and R/W as inputs. During normal program I/O the R/W line refers to the I/O since it is considered a memory device. During a DMA operation to HP-IB the sense of R/W is relative to the memory, and is inverted sense to the I/O card. The function of the XOR is to invert the R/W line during DMA to HP-IB. The 9914 has internal logic to invert the R/W line, so the XOR output is not used for 9914 R/W line.

CHIP-SELECT DECODING

The HP-IB chip select is generated by U20 and is decoded as hex address 47XXXXH. The 9914 chip enable is qualified by by BA4 high(U1 pins 1&2) and BLDS' low(U10 pins 12&13). The first qualifier selects 9914 over the external interface registers, and the second qualifier insures the R/W line is stable at the 9914 upon CE true. This also removes CE in the fastest possible time at the end of the cycle. If BA4 is low, then the external registers are selected by U15 and U2. The external read registers are decoded as BA4 low, CS true and R/W high(U15 pins 9,10&11). This enables the LS257(U16) selector and BA2 is decoded within U16 to select the I/O STATUS or the HP-IB EXTENTED STATUS read registers. If R/W is low, then a write to I/O CONTROL (U7) is decoded by CS and BA4 low(U2 pins 1&2) and WE(U1 pins 12&13). This flip-flop is cleared by reset for power-up initialization.

9914 PROGRAMMED READ/WRITE

The lines that control access to the 9914 are the register select lines(U3 pins 6,7&8), CE, Data Bus IN(DBIN pin 5), and WE. On read cycles DBIN is high and WE is inactive, the timing is controlled by CE pulse. The internal HP-IB is a synchronous I/O device mainly because this saved parts and marginally increased performance. The processor pulls on DTACK for the interface, and the interface uses the processor generated DTACK for timing information. This is done on write cycles to anticipate the termination of the cycle and is used in the generation of WE. On a write cycle DBIN goes low, then WE, and finally CE. WE is generated by U15 and is the logical AND of DTACK false, DMARDY false and R/W low. WE is terminated by DTACK going true when the processor times out the cycle, and this guaranties hold times on data and address. During programmed I/O the XOR gates (U9) are set up to invert R/W(pin 6) and not invert DTACK(pin 3).

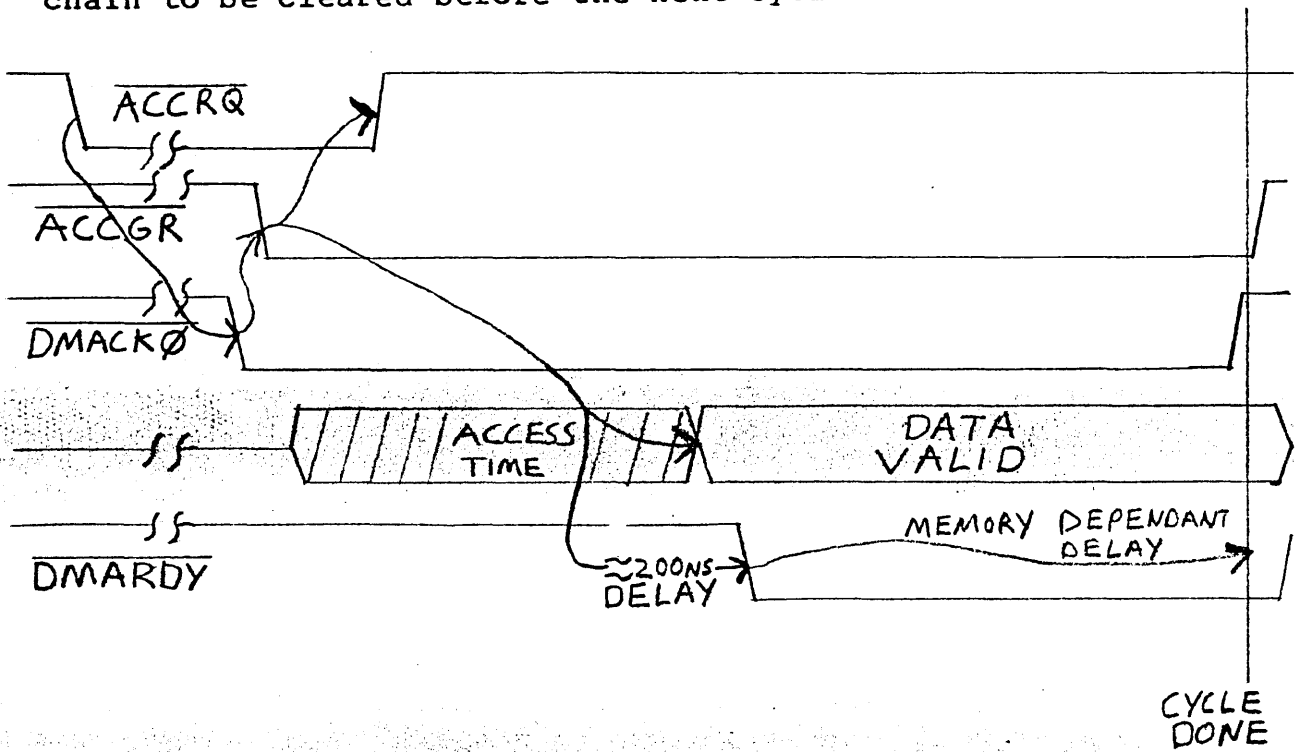


				MODEL 9826/9836,	STK NO
				Theory of Operation	
				BY John Zagnoli	DATE November 6, 1980
	See Pg.1 for Revision			APPD	SHEET NO 36 OF 47
LTR	PC NO	APPROVED	DATE	SUPERSEDES	DWG NO A-09826-90315-1
	REVISIONS				

9914 DMA INPUT AND OUTPUTS

Dma I/O is complicated by the overlap of the memory cycle with the I/O cycle, and this requires an additional handshake called DMARDY. Dma is enabled at two points, first the 9914 ACCRQ line is gated onto DMAR0 by U5(pins 9&8) and the OC inverter U18. Then DMACK0 is gated to the ACCGR(U3 pin 2) by U1(pins 4&5). Both of these gates are enabled by the Q' output of U7 which is set by a write to I/O control.

A dma input is considered a write cycle and the first event in the cycle is the HP-IB putting data on the bus for memory. This is done when DMACK goes true and is gated to ACCGR which starts the access to the 9914 DATA IN register. ACCGR also enables the data buffer and starts the timing delay through the 3 flip-flops of U6. The logic for starting the delay is the AND of DMACK(U2 pins 12&13) and the OR of WRITE and DTACK(U2 pins 9&10). Since input is considered a write memory, the timing starts as soon as DMACK is received.
When the timeout is finished DMARDY is asserted by U5(pins 5&6) and OC inverter U18. At this point the HP-IB is done, and the cycle terminates after memory captures data and DMACK goes false. The dma rate is guaranteed to be slow enough for the timing chain to be cleared before the next cycle starts.

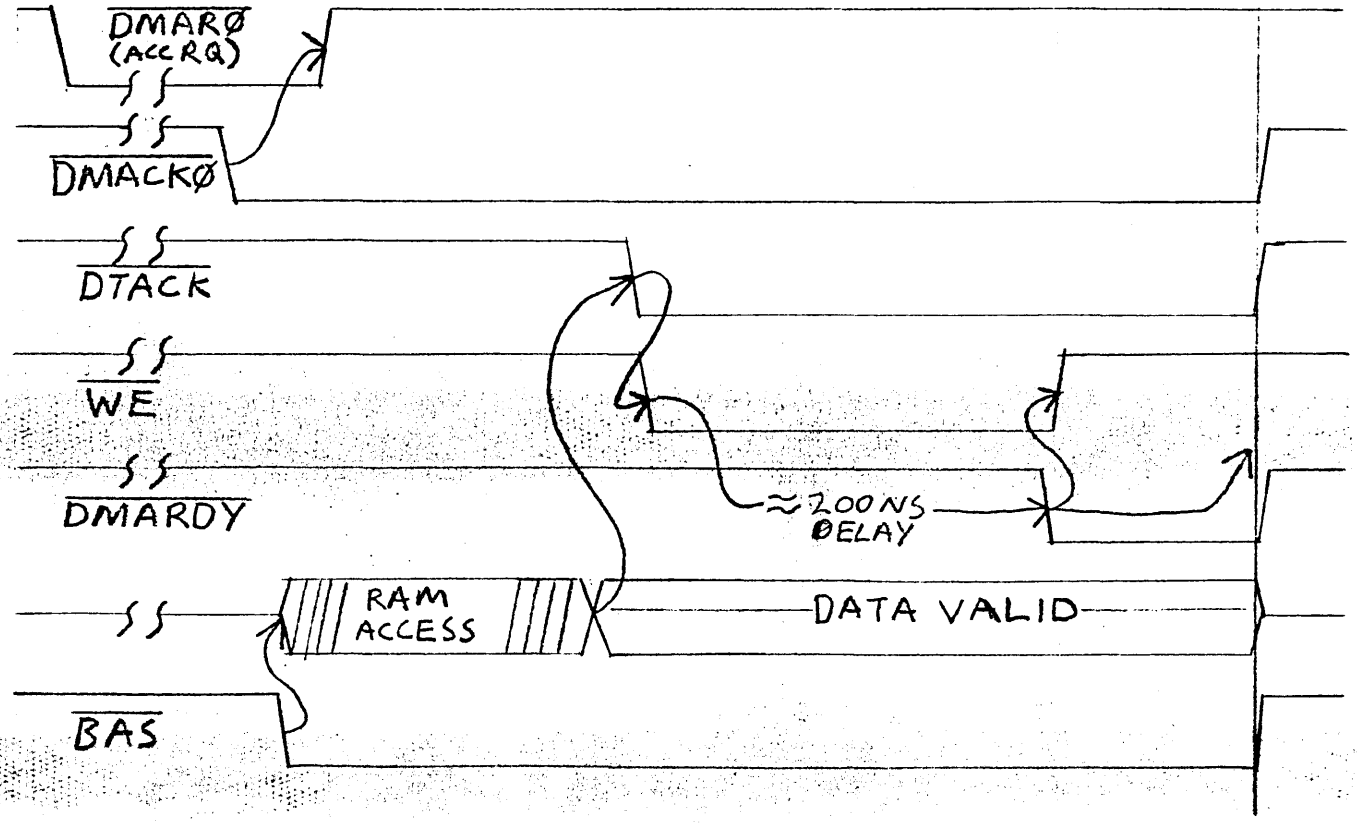


DMA INPUT CYCLE TIMNING

				MODEL 9826 /9836	STK NO
				Theory of Operation	
				BY John Zagnoli	DATE November 6, 1980
M	See Pg.1 for Revision			APPD	SHEET NO 37 OF 47
	LTR	PC NO	APPROVED	DATE	
	REVISIONS			SUPERSEDES	DWG NO A-09826-90315-1

DMA OUTPUT

A dma output is overlapped with a memory read cycle, and the 9914 doesn't receive data until the memory access is complete. The cycle starts with a normal access to memory, but DTACK doesn't terminate the cycle since the memory data must now be latched by the I/O device. The output cycle DMARDY hanshake is designed to allow the I/O device sufficient time to latch the data from memory. DMACK0 generates an ACCGR(if enabled), but the timing chain and WE aren't started until data is valid and DTACK is asserted by memory. DTACK true causes WE true by the XOR(U9 pins 1&2) and U15, which shows the function of the XOR because in programmed I/O, DTACK true caused WE false. DTACK true also starts the timing chain which allows the 9914 to latch the data after the 200ns delay and assert DMARDY. The assertion of DMARDY signals the dma control to terminate as soon a possible, so no assumption must be made about either I/O of memory speed. One of the inputs to U15 is DMARDY so WE goes false upon the assertion of DMARDY. As on a programmed I/O cycle, the anticipation of the end of the cycle guaranties the hold time on data.



DMA OUTPUT CYCLE TIMING

				MODEL 9826/9836	STK NO
				Theory of Operation	
				BY John Zagnoli	DATE November 6, 1980
	See Pg.1	for Revision		APPD	SHEET NO 38 OF 47
LTR	PC NO	APPROVED	DATE	SUPERSEDES	DWG NO A-09826-90315-1
	REVISIONS				

M

VI. INTERNAL HPIB-SOFTWARE DESCRIPTION

A) INTRODUCTION

This section describes the software interface to the HP-IB interface bus. Of primary concern is the operation of the TMS-9914 interface adapter, of which the HP-IB interface is built around. The TMS-9914(for short the '9914') is fully described in the Texas Instruments data sheet, and the programmer will have to refer to this document for detailed operation of the interface IC. The software interface consists of a set of memory-mapped registers for status, control and data. Some of these registers reside in the 9914 and others are implemented in external hardware. The registers will be treated separately, since the 9914 manual only describes the IC registers.

B) CONFIGURATION OF INTERFACE

The 9826 internal HP-IB is configured through the I/O backplane opening. Since this is somewhat inconvenient the only hardware configuration is the SYTEM CONTROLLER function. Upon power up, the interface must know if it is the system controller and does so by reading the position of the jumper located on the 9826 motherboard. To get to the jumper the back panel of 9826 must face you, and the bottom three cards in the backplane removed. The jumper is a blue rectangular plastic piece that slides onto two of three square posts. It is located one inch in from the back and about in the center of the opening. When the jumper is on the RIGHT the interface is SYSTEM CONTROLLER. It is important to note the differences between the built-in and the add-on HP-IB's, which are listed below:

1) Built in HP-IB is on select code 7, and cannot be set to any other select code.
2) Built in HP-IB has no HP-IB ADDRESS SWITCH, it is set to a software default on power up.
3) Built in HP-IB has no interrupt level select switch and it is always on I/O interrupt level 0 (this is equivalent to processor level 3).

C) REGISTER MAP

The following table is a register map of the INTERNAL HP-IB. It is important to note that the internal HP-IB is a subset of the optional plug-in HP-IB, and that all registers are mapped to the same offset addresses.

BASE ADDRESS=478000 (INTERNAL SYNC I/O #7, 3 WAIT STATE)

****** EXTERNAL REGISTERS ******

HEX OFF.	R/W	REG NAME	BIT# 7	6	5	4	3	2	1	ø
3	R	I/O STATUS	1	INT	X	X	X	X	X	DMAø
3	W	I/O CONTROL	X	X	X	X	X	X	X	DMAø
5	R	EXT. STATUS	SYS	AC'	X	X	X	X	X	X

****** 9914 REGISTERS ******

HEX OFF.	R/W	REG NAME	BIT# 7	6	5	4	3	2	1	0
11	R	INT STATUS 0	INT0	INT1	BI	BO	END	SPAS	RLC	MAC
11	W	INT MASK 0	X	X	BI	BO	END	SPAS	RLC	MAC
13	R	INT STATUS 1	GET	ERR	UCG	APT	DCAS	MA	SRQ	IFC
13	W	INT MASK 1	GET	ERR	UCG	APT	DCAS	MA	SRQ	IFC
15	R	ADD. STAT	REM	LLO	ATN	LPAS	TPAS	LADS	TADS	ulpa
17	R	BUS STATUS	ATN	DAV	NDAC	NRFD	EOI	SRQ	IFC	REN
17	W	AUX CMD	C/S	X	X	f4	f3	f2	f1	f0
19	W	ADDRESS REG	edpa	dal	dat	A5	A4	A3	A2	A1
1B	W	SERIAL POLL	S8	RSV	S6	S5	S4	S3	S2	S1
1D	R	CMD PASS	DIO8	DIO7	DIO6	DIO5	DIO4	DIO3	DIO2	DIO1
1D	W	PARALLEL POLL	PP8	PP7	PP6	PP5	PP4	PP3	PP2	PP1
1F	R	DATA IN	DIO8	DIO7	DIO6	DIO5	DIO4	DIO3	DIO2	DIO1
1F	W	DATA OUT	DIO8	DIO7	DIO6	DIO5	DIO4	DIO3	DIO2	DIO1

INTERNAL HP-IB REGISTER MAP
FIGURE VI.1

D) REGISTER DESCRIPTIONS

1) REGISTERS EXTERNAL TO THE 9914

The registers external to the 9914 are described in detail, since they are unique to the internal HP-IB interface. It should be noted that this interface has no ID register since it need not be found by the operating system, and is always known to exists at a specific address.

REGISTER 3: I/O STATUS

BIT # READ

7 ENI-The enable interrupt bit is always read as 1, since interrupts are always enabled.

6 INT-This bit indicates whether HP-IB is interrupting on level 3.

0 DMA0-This bit indicates HP-IB is enabled on dma channel 0. It should only be enabled while doing dma, since no other I/O card can use dma 0 while HP-IB is enabled.

* Notes-There are no int. level bits since internal HP-IB is always on processor level 3(I/O level 0). Also, dma is not possible on channel 1 due to hardware limitations.

REGISTER 3: I/O CONTROL

BIT # WRITE

0 DMA0-This bit controls HP-IB access to dma shared resource. It should always be disabled unless dma is in progress.

* Notes-There is no interrupt disable capability external to the 9914 on this interface, though interrupts can be fully controlled within the 9914.

REGISTER 5:HPIB EXTENDED STATUS

BIT # READ

7 SYS-This bit indicates if the interface is designated as the system controller. The 9914 does not use this bit since it simply controls the direction of the buffers on the HP-IB IFC and REN control lines.

6 AC'-This bit indicates if the interface is presently the contoller in charge(CIC). This status is not available from the 9914, though the 9914 controls the line from which this bit is derived.

* Notes-There is no HP-IB address bits in this register, as there are on the add-on hpib. This means the system must default in software the internal HP-IB address.

A note of interest is that the keyboard NMI status bit is loacated in bit 2 of this register, but does not concern the HP-IB in any way.

2) 9914 REGISTERS

The 9914 will be described briefly in this section, and the programmer is asked to refer to the 9914 data manual for further information.

REGISTER 11: INTERRUPT STATUS 0
READ

This register in conjunction with interrupt status 1 provide most of the status necessary to control the interface. The register contents are cleared upon reading, so that status is only seen once and then automatically cleared. The register is a combination status/interrupt register since each bit can be independently masked to interrupt. Even if disabled to interrupt a bit still provides status for programmed interaction. The top two bits INT0 and INT1 are special in that they are not really interrupt bits and have no masks. They simply provide a quick way to determine which status register has an interrupt. This is why INT1 is not cleared upon reading STATUS 0, since it really is tied to the STATUS 1 bits.

REGISTER 13: INTERRUPT STATUS 1
READ

This register operates exactly like STATUS 0, except there are no special bits like the INT0 AND INT1 bits.

REGISTER 11: INTERRUPT MASK 0
WRITE

This register controls interrupts by selecting which status bits will generate interrupts. This register is not cleared upon power up, so it must be written upon initialization to avoid spurious interrupts if the interface becomes active. If a status bit is set before the mask bit is enabled, and then the mask bit is set, an interrupt will be imediately generated. Also, the GET, UCG, APT, DCAS And MA bit cause an ACDS holdoff on receipt of the interface commands. This is important since these bits essentially change the state diagram of the 9914. There is one other function in the 9914 that interacts with the interrupts and that is the DAI auxiliary command. This command independently disables the interrupt line without affecting the 9914's response in any other way.

REGISTER 13: INTERRUPT MASK 1
WRITE

This register functions the same as MASK 0.

REGISTER 15: ADDRESS STATUS
READ

This register describes the address state of the 9914, and is not a true storage register as the data is obtained from the internal logic of the 9914 at the time of the access.

REGISTER 17: BUS STATUS
READ

The bus status register is simply a `snapshot' of the HP-IB interface control lines at the time of access. These line are not guaranteed to be stable at the time of access, so the data received must be carefully interpreted.

REGISTER 17: AUX COMMAND
WRITE

The auxiliary command provides most of the control over the HP-IB chip and bus. The specific command is encoded on the lower four bits of the data byte sent. Some of the commands are of the clear/set type(i.e. turn on ifc/turn off ifc) and use the msb of the data byte to indicate on/off. Other commands are pulse type commands(i.e. release rfd holdoff) and the clear/set bit is not applicable.

REGISTER 19: ADDRESS REGISTER
WRITE

This write only register is simply to set up the HP-IB address of the 9914 if it is not the controller in charge. If the 9914 is the CIC then it simply doesn`t care what its address is. When it is locally enabled to talk(through the aux. cmd. `ton') no talk address is sent. Therefore, simply sending it's talk address while it is controller doesn't address it to talk. The top three bits(edpa,dat, and dal) are not used in HP-IB implementation of devices.

REGISTER 1B: SERIAL POLL RESPONSE
WRITE

This register hold the serial poll response byte and is automatically sent during a serial poll to the 9914. The status register indicates when a serial poll has occurred and the 9914 was requesting. The RSV bit of the serial poll register must be cleared before the SRQ line can be set true again by the 9914.

REGISTER 1D: COMMAND PASS THROUGH
READ

This register is used when the 9914 receives a command that it doesn't handle automatically. It enables the CPU to read the command directly off the HP-IB data lines. This is only useful when the 9914 is in the ACDS so that data lines are held stable.

REGISTER 1D: PARALLEL POLL RESPONSE
WRITE

This register defines the parallel poll response and is written after the CPU has decoded a parallel poll configure message. The response is not decoded and the CPU is responsible for decoding the configure message and writing the proper bit of the poll response register. It is possible to enable the 9914 to respond on more than one data line in parallel poll, though this is not a legal response.

REGISTER 1F: DATA IN
READ

Data is latched in this register during the normal data handshake on the HP-IB and is indicated to be available by one of three methods. The first method to receive data is to continuously read the STATUS 0 register looking for a BI. The other status bits are also read and cleared, so care must be taken not to throw away wanted status. The second way is to enable interrupts on BI, so that the CPU is interrupted each time a byte is received. The BI is automatically cleared on reading the data register so that a read of the status register is not necessary if it is known that BI was the only possible interrupt. The third way is used during dma where the ACCRQ' line on the 9914 is active each time the data register is full. The handshake is automatically completed when the dma control services the dma request.

REGISTER 1F: DATA OUT
WRITE

The data out register is the data path for both commands when the 9914 is active controller, and data when the 9914 is talking. When the 9914 is in the controller active state, multi-line messages are sent using the BO status and writing the data register(no dma allowed). When data is being sent either BO or dma can be used. The source handshake is automatically controlled by the 9914, and the CPU only need check the status register to send bytes. The 9914 when talking always has a BO interrupt pending if there is no data on the bus, and the dma request line reflects the state of the BO and BI status bits. A read of the STATUS 0 register clears the BO and BI bits, but not the dma request line. The only operations that clear the dma request line are a read of the DATA IN or a write of the DATA OUT register. Since any output operation leaves the BO bit set(because the handshake is allowed to complete) it also leaves the dma request line set. If an input operation is attempted a dma cycle is generated immediately and a spurious data byte may be received. The only way to avoid this is to do a dummy read of the data register when it is certain to be empty(otherwise another byte may be lost) and this will clear the dma request line .

E) OTHER FUNCTIONS

HARDWARE RESET

Upon receiving a hardware reset(power-on or CPU reset instruction) the interface immediately enters all the idle states on the HP-IB state diagrams. In addition the serial and parallel poll and all the auxiliary commands are reset(with the exception 'SWRST' which has to be cleared before the interface can become active again). The only registers that are not cleared are the MASK registers.

SOFTWARE RESET

Software reset is actually an auxiliary command, but it really should be included here since the hardware reset is related. They both leave the chip in the `SWRST' state where the chip is ready to receive set-up commands, but doesn't interact with the bus until a clear `swrst' auxiliary is received.. Software reset causes all the HP-IB state functions to return to the idle state. What is not cleared by soft reset is listed below:

SERIAL POLL-not cleared
PARALLEL POLL-not cleared
AUX CMDS-not cleared(this includes SIC and SRE)
INT MASKS-not cleared(all status is cleared however)

VII. MISCELLANEOUS

A) POWER SUPPLY

The 09826-66501 motherboard has load resistors for the +12 volt power supplies. These are R23 and R24.

Inductor L1 is used to filter the floppy drive's +5 volt supplies.

B) THE CONNECTORS

The card cage connector and the expansion connector are exactly the same. There are four spare lines on these connectors. The spares are bussed together and up the back plain Some of the spare lines are also bussed to the CPU, the floppy control and the CRT boards. See drawing 09826-66501-4 (page 2) to see what lines are spares. The CPU has two spares. The floppy control board has three of the spares. On the -66501 board, the CRT board also has three of the spares.

Care was taken on the HPIB connector and J15, the analog connector in the -66502, to make sure that the component side of the board did not have any traces go under the metal part of the connector.

VIII. TEST POINTS

All the test points are listed below.

TP1 - The TMS9914 (U3) 'ACCRQ' line (pin 1).

TP2 - When high the 8041A (U11) is pulling on interrupt 7.

TP3 - +5

TP4 - GND

TP5 - The TMS9914 (U3) ACCRQ' line (pin 1).

TP6 - The speaker driver input.

TP7 - When high HPIB is pulling DMARDY'.

TP8 - The 8041A CS' input (pin 6, chip select).

TP9 - When low the 8041A (U11) is pulling on interrupt 7.

TP10 - 10 MHz

TP11 - When high the 8041A is pulling on interrupt 1.

TP12 - Will flash from low to high when the RPG is turned.

TP13 - The most significant bit of the key switch scan address.

TP14 - The buffered SYNC output (pin 11) of the 8041A (U11). This synchronizes reset and provides the base frequency for the beep.

TP15 - The least significant bit of the key switch scan address.