To: Bob Bailey
Gary Butts - APG
Dave Christensen - APG
Mike Clark - APG
Burrell Smith

Date: Sept. 26, 1984

From: Peter Ashkin

Subj: Front Desk Bus - **Rev 2.1**

To make the "Front Desk Bus" a more flexible and powerful interface, I believe that it should have the following properties:

1. The bus shall be bidirectional. [An input only bus is too restrictive.]

2. Each device on the bus has a unique address. For practical purposed the address range should be 0 - 14. Some of these addresses may be reserved for broadcasting universal messages. [This seems like a sane number of devices, particularly since there exists today only three devices; keyboard, keypad and mouse.]

3. All command transactions shall be eight bits long. All data transactions shall be 16 bits long. [This facilitates the decoding of commands by devices of limited intelligence.]

4. The host shall be the undisputed bus master. [This removes any question of who's controlling the bus.]

5. There shall be a limited number of commands. Commands should be broken into two groups, basic commands (**TALK** and **LISTEN**) which all devices on the bus shall understand; and advanced commands which only intelligent devices (as appropriate) should understand. [This makes the command interpreter, be it hardware or software, simple. It also allows more complex devices to used some of the "fancier" features of the bus.]

6. There shall be only one active talker on the bus at any time, this may be the host or an addressed device. [A device addressed to **TALK** with data to send "untalks" itself after it sends its 16 bits of data or if it has no data to send "untalks" itself immediately and allows the bus to time-out.]

7. The bus protocol must accept devices that talk at different speeds. The host, at a minimum, must be able to listen at various speeds. [This implies that the data on the bus must be "self-clocked". By not rigidly fixing the speed of transmission, the bus does not need to be crystal (etc.) controlled.]

8. There shall be only one active listener on the bus at any time, this may be the host or an addressed device. [A device addressed to **LISTEN** "unlistens" itself after it receives 16 bits of data or if it receives a new command before receiving 16 bits of data.]

9. An interrupt mechanism must be available which circumvents the needs to poll devices that need service. [Since the bus is relatively slow, the interrupt latency time in a polled environment is long. The ability to interrupt the master for service is important.]

10. There shall exist a mechanism that sends a unique signal that puts all devices on the bus into the command (reset) mode. [This is important if for some reason the bus gets "hung".]

11. There should be a minimum number of "time-outs" needed on the bus. The only needed time out should be to time out a non-responsive talker. [Timers are ugly, but waiting for a dead device is uglier. The length of this time-out should be controlled by the host.]

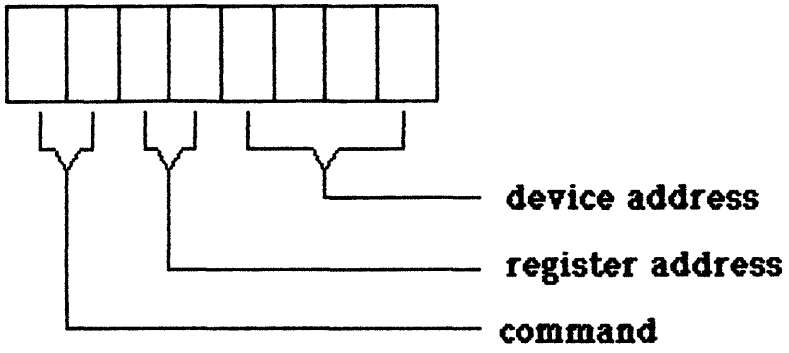
12. Hand-off of the bus from the host to a talker and back again must be made without bus contention. [Contentions hurt output drivers and are noisey. The pullup of the bus if it is actively driven must go tristate when inactive on the bus.]

Commands:

There are two major command groups; basic commands and advanced commands. All devices on the bus shall understand at least one command in the basic group and optionally understand commands in the advanced group.

BASIC Command Group:

There are two commands in this group; **TALK** and **LISTEN**.



Commands may only be sent by the host. Data may be sent by either the host (**LISTEN**) or the addressed device (**TALK**).

The next two bits form the command: "11" for **TALK** and "10" for **LISTEN**. All devices on the bus must obey at least one of these commands. Keyboards, numeric keypads and mice as a minimum must respond to the "00" **TALK** command. When a device is addressed to **TALK**, it must respond before being timed out by the host. This timeout as suggested by APG might be in the range of 200us. [This is reasonable for devices that are microcomputer based. The GI PIC series of processors that APG will use executes an instruction in approximately 2us.] The selected device then becomes active on the bus, performs its 16 bit transaction then "untalks" itself and goes inactive on the bus. Thus **TALK** commands transfer only 16 bits at a time and a new **TALK** command must be issued to transfer the next 16 bits.

When a device is addressed to **LISTEN**, it is enabled to accept a data transaction from the host. Only a single device at a time can be addressed to **LISTEN**. After the device is addressed, it is enabled to receive the next 16 data bits that are placed on the bus by the host. After the data bits are received, the transaction is complete and the device "unlistens" itself. If a device is addressed to **LISTEN** and it receives another command on the bus before it receives any data, then by definition the transaction is

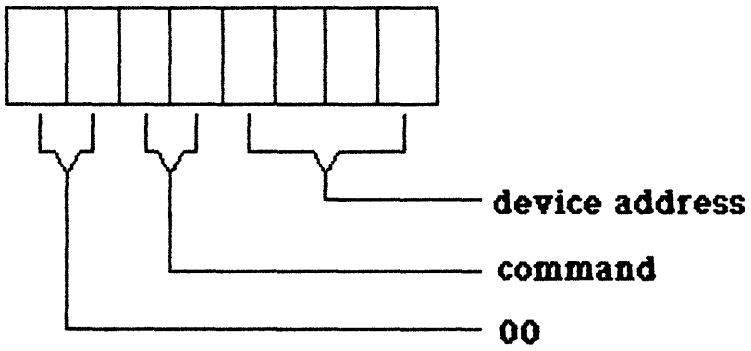
immediately complete and the device "unlistens" itself. As an example:

TALK 2	:device 2 commanded to send a 16 bits or timeout
(16 data bits)	:sent by device 2, received by host
TALK 2	:send another 16 bits
(16 data bits)	:another 16 bits from device 2
LISTEN 14	:enables device 14 to listen
(16 data bits)	:sent by host to device 14
LISTEN 6	:enables device 6 to listen
LISTEN 14	:enables device 14 to listen and "unlistens" device 6
(16 data bits)	:sent by host to device 14
	:etc

The next field is a two bit register address field. This field, which is optional, allows a specific register within an addressed device to be specified. An example of where this might be used is to differentiate a data register (in a keyboard, the specific keystroke) from a status/configuration register (in a keyboard, a response that signifies the model of the keyboard). Finally there is a four bit device address field which specifies the address of the selected device. These addresses range from 0 - 14.

Advanced Command Group:

There are two commands in this group; **ENABLE (INTERRUPT)** and **DISABLE (INTERRUPT)**. There are also five reserved commands for future expansion.



Note that the defined advanced commands have the two most significant bits set to "00".

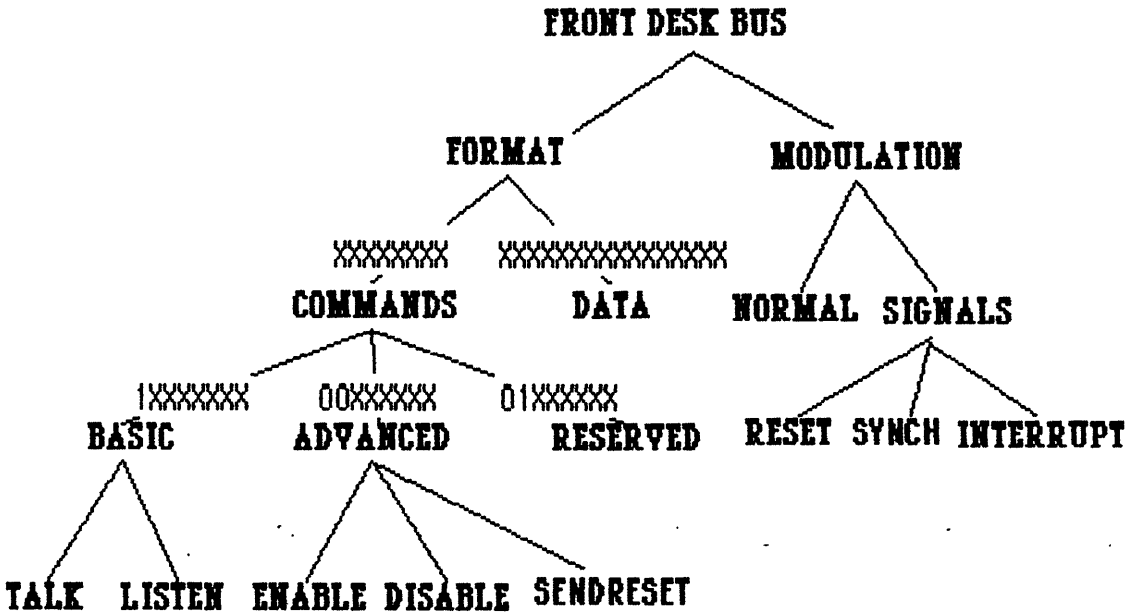
These commands deal with the ability of devices on the bus to interrupt the host. This is useful in systems where the interrupt response time in a polled system is longer than desired. **ENABLE** allows selected devices to signal an interrupt on the bus, or conversely **DISABLE** selectively inhibits

the signalling of an interrupt. When an enabled device signals an interrupt, the host may not know which device has signalled if multiple devices have been enabled.

"00"	**ENABLE**
"01"	**DISABLE**

ENABLE and **DISABLE** require that the address of the desired device be specified. The range is 0 - 14. Address 15 is a reserved address for the **DISABLE** command and serves as a global disable.

To allow for future expansion of the command structure, a group of "place holder" **RESERVED** instructions has been defined. These instructions shall be treated as no-ops.



Command syntax:

ENABLE	00	00	$A_3A_2A_1A_0$
DISABLE	00	01	$A_3A_2A_1A_0$
SENDRESET	00	10	XXXX (forces a **RESET** on bus)
RESERVED	00	11	XXXX
RESERVED	01	XX	XXXX
LISTEN	10	R_1R_0	$A_3A_2A_1A_0$
TALK	11	R_1R_0	$A_3A_2A_1A_0$

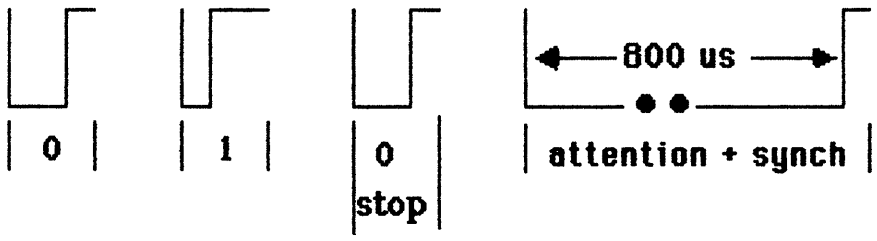
MODULATION:

There are two forms of modulation on the bus, **NORMAL** transactions which transmit commands and data, and **SIGNALS** which broadcast global messages such as **RESET, SYNCH** and **INTERRUPT.**

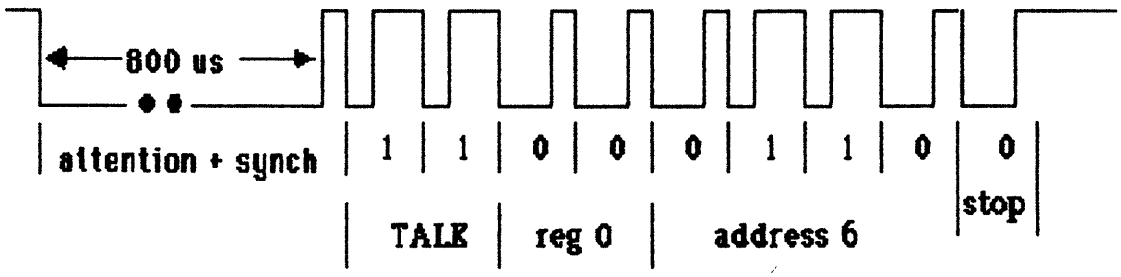
NORMAL transactions:

To achieve the goal of the bus being self clocked, a RZ code for modulation has been adopted. This code has several properties that are advantageous to the Front Desk Bus. Among these advantages are: ease of recovery of the clock and the data; always leaves the bus in a known state (without the use of dummy transactions); and has definite "openings" in the waveforms to signal special transactions. Each bit cell boundary is signified by a falling edge on the bus. The period of each bit cell is the time between two falling edges on the bus. The data is encoded as the ratio of low to high time of each bit cell. Thus a "0" is encoded as a bit cell inwhich the low time is greater than the high time. Conversely, a "1" is encoded as a bit cell inwhich the low time is less than the high time. Typically this ratio might range from 2:1 to 3:1 high to low (or low to high).

To signal the start of a command, an 800 usec long **attention** pulse is sent. This is followed by a **synch** pulse to give the initial bus timing. The falling edge of the **synch** pulse is used as a timing reference for the first bit of the command or data. To synchronize the stopping of transactions, one "0" **stop** bit is sent. Following the imaginary bit cell boundary after the **stop** bit, the transaction is complete and the host (or talker) releases their active drive of the bus.



As a specific example, a **TALK** command to register 0 of device 6 would be encodes as "11000110". The bus would be modulated with the following:

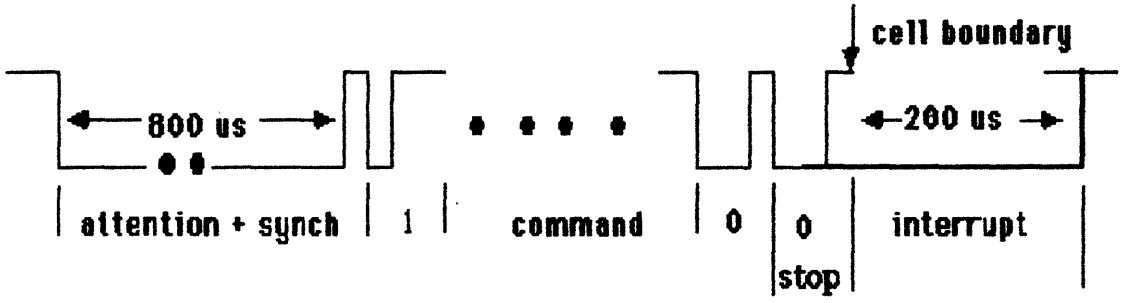


SIGNALS:

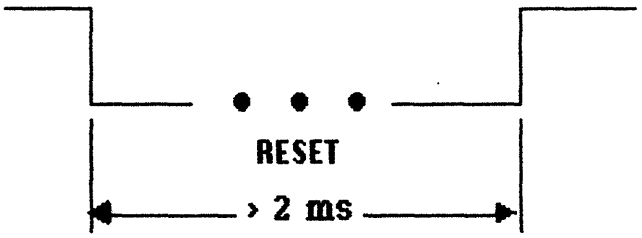
Certain transactions fall under the category of neither commands nor data transactions. These are special transactions which globally broadcast status to devices on the bus. There are three special transactions in this group. **INTERRUPT** is a transaction that devices that have been enabled to interrupt can use to signal the host that they require service. Following any command transaction, an interrupting device can signal by holding the bus low during the low portion of the stop bit of the **NORMAL** transaction. The interrupting device holds the bus low 200 usec beyond the bit cell boundary to signal. Once a device has interrupted, it shall **INTERRUPT** repeatedly until serviced. When the interrupting device is addressed to **TALK**, it shall be considered serviced and not **INTERRUPT** again until it needs to be serviced again.

wont this cause the kbd to lose keys.

eg, it won't signal interrupt on the command in which it is addressed to talk??



The **RESET** signal has the effect that it resets all pending interrupts; it turns the interrupt mode of all devices to **DISABLE**; and in general puts the devices in a mode inwhich they will accept commands. **RESET** issues a break on the bus by holding the bus low for a minimum of 2 ms.



Is there a no-op command that can be used simply to determine whether a device has some user input??