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

Date: Sept. 6, 1984

From: Peter Ashkin

Subj: Front Desk Bus - An Alternative Proposal - REV 1.2

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 - 15. 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 fixed length (recommended to be 8 bits). All data transactions shall be fixed length (recommended to be 16 bits).* [This facilitates the decoding of commands by devices of limited intelligence.]

4. *The host shall be the undisputed bus master. [This removes any questions 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 a remote device. [When a new device is commanded to **TALK**, an old device that was addressed to **TALK** is "untalked".]

7. Bus 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 can be multiple active listeners on the bus. [**LISTEN** commands are additive, as needed, multiple devices can be addressed to listen. To remove a selected listener, a special "unlisten" command is sent to globally deselect all listeners.]

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 message 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.]

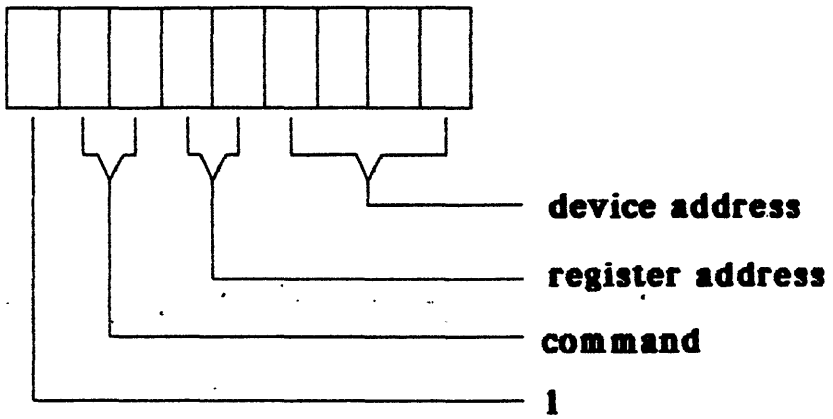
12. ***Hand-off of the bus from the master to a talker and back again must be made without bus contention.*** [Contentions hurt output drivers and are noisey. The pullup of the bus if it is actively driven must go tristate when inactive on the bus.]

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**.



Note that the MSB of all commands is set to "1". Only the bus master has the ability to drive the MSB of any transaction on the bus to a "1". Conversely, all data transactions have the MSB set to "0". The bus master as well as any other device on the bus has the ability to set the MSB to "0".

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 unselects itself and goes inactive on the bus. Thus **TALK** commands transfers *two bytes* at a time and a new **TALK** command must be issued to read the next *two* bytes.

When a device is addressed to **LISTEN**, it is enabled to accept data transaction from the host or other bus device. Multiple devices can be addressed to **LISTEN** simultaneously. This is done by sending multiple **LISTEN** commands to different listen addresses. Devices that are

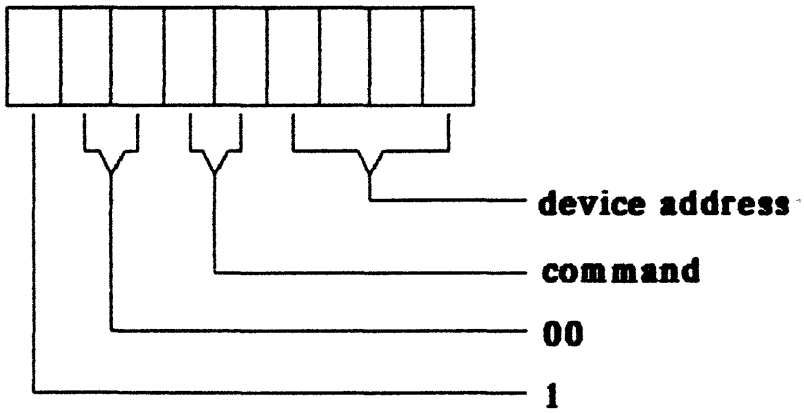
addressed to **LISTEN** remain listeners until either specifically commanded to "unlisten" by issuing a **LISTEN** command to reserved listen address 15 (which unlistens all devices that are listeners), or addressed to **TALK** (devices cannot be simultaneously a talker and a listener.) As an example:

LISTEN 7	:device 7 commanded to listen
LISTEN 14	:devices 7 and 14 both commanded to listen
TALK 2	:device 2 commanded to send a byte or timeout
(data byte)	:sent by device 2 received by devices 7, 14 and master
TALK 2	:send another byte
(data byte)	:another byte from device 2
LISTEN 15	:unlisten command
LISTEN 14	:turn on device 14 to listen again
(data byte)	:sent by bus master to device 14
(data byte)	:ditto
	: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. Address 15 is the unlisten address for **LISTEN** commands and the **TALK** alias address of the present bus master.

Advanced Command Group:

There are *three* commands in this group; **ENABLE (interrupt)**, **DISABLE (interrupt)** and **IDENTIFY**. There are also *five* reserved commands for future expansion.



Note that the MSB of these commands is again set to "1". The defined

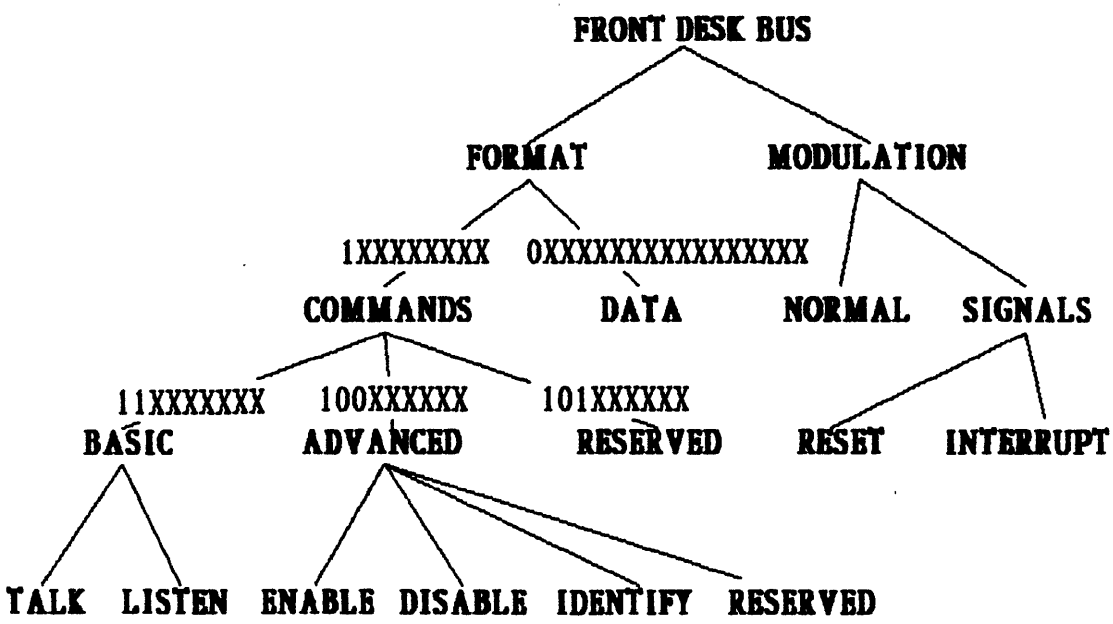
advanced commands have the next two bits set to "00".

These commands deal with the ability of devices on the bus to interrupt the bus master. 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 bus master may not know which device has signalled. This is because multiple devices may be enabled simultaneously. (Thus the command is additive, enabling one device does not disable any previously enabled device.) An **IDENTIFY** command may be issued to request that the interrupting device talk and send its address as a data transaction.

"00"	**ENABLE**
"01"	**DISABLE**
"10"	**IDENTIFY**
"11"	**RESERVED**

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. The **IDENTIFY** does not require an address to be specified, the address field in the instruction is a "don't care".

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:

TALK	1 11	R_1R_0	$A_3A_2A_1A_0$
LISTEN	1 10	R_1R_0	$A_3A_2A_1A_0$
ENABLE	1 00	00	$A_3A_2A_1A_0$
DISABLE	1 00	01	$A_3A_2A_1A_0$
IDENTIFY	1 00	10	XXXX
RESERVED	1 00	11	XXXX
RESERVED	1 01	XX	XXXX

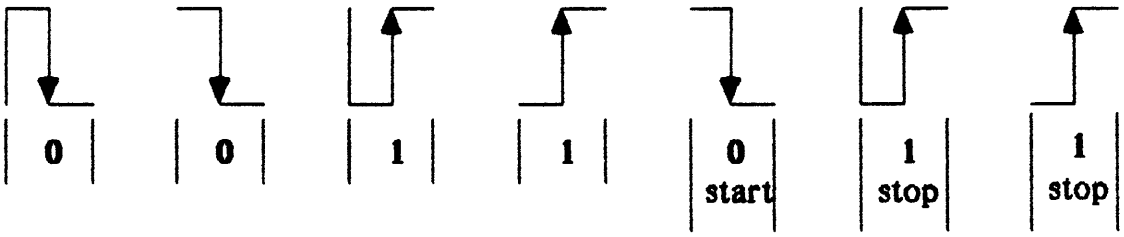
MODULATION:

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** and **INTERRUPT**.

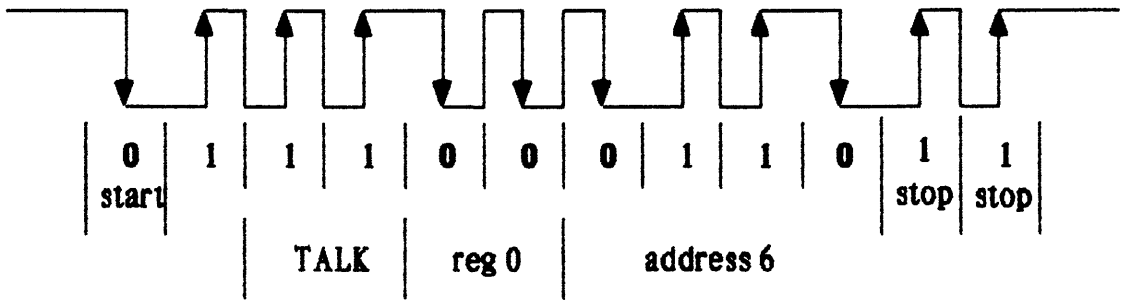
Normal transactions:

To achieve the goal of the bus being self clocked, a *Manchester* 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. *Manchester encoding uses the direction of transitions to convey the data. These transitions at the middle of each bit cell. If this transition is from a "0" to a "1", then the bit cell contains a "1". Conversely, if the transition is from a "1" to a "0", then the bit cell contains a "0". An additional transition may be made at the cell boundary to enable the proper direction transition to be made in the middle of the cell.*

To give initial timing of the bus, a "0" **start** bit is sent. This **start** bit cell is measured by the devices on the bus and is used to establish the transmission rate of this particular transaction. To synchronize the stopping of transactions, two "1" **stop** bits are sent. Following the imaginary bit cell boundary after the second **stop** bit, the transaction is complete and the bus master (or talker) releases its active drive of the bus.

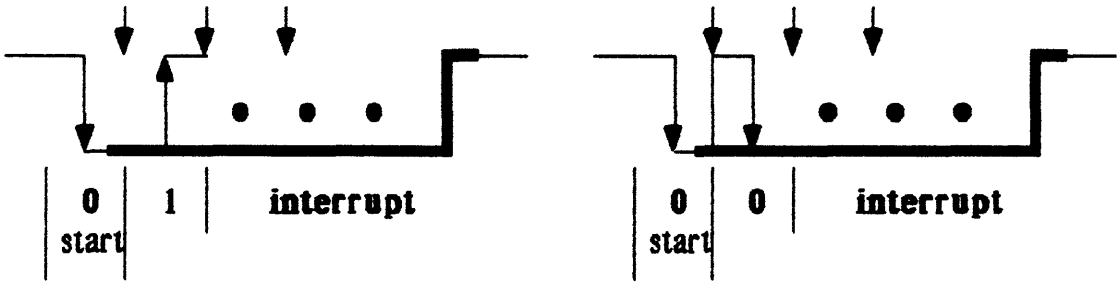


As a specific example, a **TALK** command to register 0 of device 6 would be encodes as "111000110". The bus would be modulated with the following:

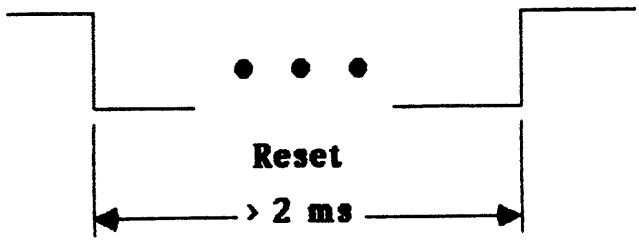


Signals:

Certain transactions fall under the category of neither command nor data transactions. These are special transactions which globally broadcast status to devices on the bus. There are two special transactions in this group. **Request Service** is a transaction that devices that have been enabled to interrupt can use to signal the master that they require service. *During the low portion of the start bit of any data or command transaction, an interrupting device can signal by holding the bus low.* The interrupting device holds the bus low beyond the *next* bit cell boundary to actually signal. The bus is held low at a minimum to just beyond the *second* bit cell boundary. Conforming advanced devices should hold the bus low a total of (1+(device address)*(bit cell time)/2) beyond the *first* bit cell boundary. *The signalling of an interrupt aborts the command or data transaction in progress.* If an interrupting device releases the bus but a higher addressed interrupting device holds the bus low for a longer time, then by protocol, the lower addressed device is inhibited from responding to an **IDENTIFY** command and must **Request Service** again. Only devices that **Request Service** and the bus returns high after their release of it can respond to the **IDENTIFY** command. Once a device has responded, it shall not respond again until the request/arbitrate procedure has occured again.



Reset has the effect that it "unlistens" all devices that have been previously addressed to **LISTEN**; it resets all pending interrupts; it turns the interrupt mode 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. (This in order not to be confused with a start bit suggests that the minimum speed of the bus be a bit cell period of < 3 ms.)


Reset
> 2 ms