HOW THE VIC/64 SERIAL BUS WORKS

By Jim Butterfield, Associate Editor

The Serial bus connects VIC or Commodore 64 to its major peripherals, especially disk and tape. The workings of this interface have been a source of bafflement to most of us. We know that it's somehow related to the IEEE-488 bus which is used on PET and CBM computers. But it has fewer wires, and it's slower. For anyone interested in interfacing details, this article will clear up the mystery.

GROUND RULES

To understand the workings of this bus, you must work through a few concepts. Later, we'll get technical for this who want it.

The bus, like the IEEE, has two modes of operation: Select mode, in which the computer calls all devices and asks for a specific device to remain connected after the call (“Jones, would you stay in my office after the meeting?”); and Data mode, in which actual information is transmitted (“Jones, I've decided to give you a raise”). Select mode is invoked by the use of a special control line called “Attention,” or ATN.

By using Select mode, you can call in any device you choose, but you may need to do more before you transmit data. You might have several disk files in progress - writing some and reading others – and when you select the disk, device 8, you'll still need to specify which “part” of the disk you want to reach: subchannel 3, subchannel 15, or whatever. To do this, we use a “secondary address” which usually signals a subsystem within a specific device. That goes in as part of the command during Select mode. Finally, we may need to send other control information: the name of the file we wish to open, for example. That's not data; it's device setup information, so we also send it in Select mode.

But the main part is: you select a device, and then you send to it or receive from it. Finally, you shut it off. All devices are connected, but only the one you have selected will listen or talk.

SOME TECHNICAL GROUND RULES

If you're not into volts and signals and things, the rest of this article may not do much for you. I want to talk about technical aspects of the bus.

First, all the data flows over two wires; They are called the Clock line and the Data line. There are other wires used for control purposes, but the data uses only the two main ones.

All wires connect to all devices. The wires don't go “one way”; any device can put a ground on a signal line, and all other devices will see it. Indeed, that's the secret of how it works: each wire serves as a common signal bus.

When no device puts a ground on a signal line, the voltage rises to almost five volts. We call this the

“false” logic condition of the wire. If any device rounds the line, the voltage drops to zero; we call this the “true” condition of the line. Note that if two devices signal “true” on a line (by grounding it), the effect is exactly the same as if only one has done so: the voltage is zero and that's that. We can summarize this as an important set of logic rules:

- A line will become “true” (PULLED DOWN, or 0V) if one or more devices signal true;
- A line will become “false” (RELEASED, or 5V) only if all devices signal false.

Remember that we have several lines, but the important ones for information transmission are the Clock line and the Data line. Let's watch them work.

TRANSMISSION: STEP ZERO

Let's look at the sequence when a character is about to be transmitted. At this time, both the Clock line and the Data line are being held down to the true tate. With a test instrument, you can't tell who's doing it, but I'll tell you: the talker is holding the Clock line true, and the listener is holding the Data line true. There could be more than one listener, in which case all of the listeners are holding the Data line true. Each of the signals might be viewed as saying, “I'm here!”.

STEP 1: READY TO SEND

Sooner or later, the talker will want to talk, and send a character. When it's ready to go, it releases the Clock line to false. This signal change might be translated as “I'm ready to send a character. “The listener must detect this and respond, but it doesn't have to do so immediately.

The listener will respond to the talker's “ready to send” signal whenever it likes; it can wait a long time. If it's a printer chugging out a line of print, or a disk drive with a formatting job in progress, it might hold back for quite a while; there's no time limit.

STEP 2: READY FOR DATA

When the listener is ready to listen, it releases the Data line to false. Suppose there is more than one listener. The Data line will go false only when all listeners have released it - in other words, when all listeners are ready to accept data.

What happens next is variable. Either the talker will pull the Clock line back to true in less than 200 microseconds - usually within 60 microseconds - or it will do nothing. The listener should be watching, and if 200 microseconds pass without the Clock line going to true, it has a special task to perform: note EOI.

INTERMISSION: EOI

If the Ready for Data signal isn't acknowledged by the talker within 200 microseconds, the listener knows that the talker is trying to signal EOI. EOI, which formally stands for “End of Indicator,” means

“this character will be the last one.” If it's a sequential disk file, don't ask for more: there will be no more. If it's a relative record, that's the end of the record. The character itself will still be coming, but the listener should note: here comes the last character.

So if the listener sees the 200 microsecond time-out, it must signal “OK, I noticed the EOI” back to the talker, I does this by pulling the Data line true for at least 60 microseconds, and then releasing it.

The talker will then revert to transmitting the character in the usual way; within 60 microseconds it will pull the Clock line true, and transmission will continue.

At this point, the Clock line is true whether or not we have gone through the EOI sequence; we're back to a common transmission sequence.

STEP 3: SENDING THE BITS

The talker has eight bits to send. They will go out without handshake; in other words, the listener had better be there to catch them, since the talker won't wait to hear from the listener. At this point, the talker controls both lines, Clock and Data. At the beginning of the sequence, it is holding the Clock true, while the Data line is released to false. the Data line will change soon, since we'll send the data over it.

The eights bits will go out from the character one at a time, with the least significant bit going first. For example, if the character is the ASCII question mark, which is written in binary as 00011111, the ones will go out first, followed by the zeros.

Now, for each bit, we set the Data line true or false according to whether the bit is one or zero. As soon as that's set, the Clock line is released to false, signalling “data ready.” The talker will typically have a bit in place and be signalling ready in 70 microseconds or less.

Once the talker has signalled “data ready,” it will hold the two lines steady for at least 20 microseconds timing needs to be increased to 60 microseconds if the Commodore 64 is listening, since the 64's video chip may interrupt the processor for 42 microseconds at a time, and without the extra wait the 64 might completely miss a bit.

The listener plays a passive role here; it sends nothing, and just watches. As soon as it sees the Clock line false, it grabs the bit from the Data line and puts it away. It then waits for the clock line to go true, in order to prepare for the next bit.

When the talker figures the data has been held for a sufficient length of time, it pulls the Clock line true and releases the Data line to false. Then it starts to prepare the next bit.

STEP 4: FRAME HANDSHAKE

After the eighth bit has been sent, it's the listener's turn to acknowledge. At this moment, the Clock line is true and the Data line is false. The listener must acknowledge receiving the byte OK by pulling the Data line to true.

The talker is now watching the Data line. If the listener doesn't pull the Data line true within one millisecond - one thousand microseconds - it will know that something's wrong and may alarm

appropriately.

STEP 5: START OVER

We're finished, and back where we started. The talker is holding the Clock line true, and the listener is holding the Data line true. We're ready for step 1; we may send another character - unless EOI has happened. If EOI was sent or received in this last transmission, both talker and listener “let go.” After a suitable pause, the Clock and Data lines are released to false and transmission stops.

[DIAGRAM]

ATTENTION!

This is all very well for a transmission that's under way, but how do we set up talker and listener? We use an extra line that overrides everything else, called the ATN, or Attention line. Normally, the computer is the only device that will pull ATN true. When it does so, all other devices drop what they are doing and become listeners. Signals sent by the computer during an ATN period look like ordinary characters - eight bits with the usual handshake - but they are not data. They are “Talk,” “Listen,” “Untalk,” and “Unlisten” commands telling a specific device that it will become (or cease to be) a talker or listener. The commands go to all devices, and all devices acknowledge them, but only the ones with the suitable device numbers will switch into talk and listen mode. These commands are sometimes followed by a secondary address, and after ATN is released, perhaps by a file name. An example might help give an idea of the nature of the communications that take place. To open for writing a sequential disk file called “XX,” the following sequence would be sent with ATN on:DEVICE-8-LISTEN;SECONDARY-ADDRESS-2-OPEN. When ATN switches off, the computer will be waiting as a talker, holding the Clock line true; and the disk will be the listener, holding the Data line true. That's good, because the computer has more to send, and it will transmit: X;X;comma;s;comma;W - the W will be accompanied with an EOI signal. Shortly thereafter, the computer will switch ATN back on and send DEVICE-8- UNLISTEN. The file is now open; later, the computer will want to send data there. It will transmit, with ATN on, DEVICE-8-LISTEN;SECONDARY- ADDRESS-2-DATA. Then the computer releases the ATN line and sends its data; only the disk will receive the data, and the disk will know to put it onto the file called XX. The last character sent by the computer will also signal EOI. After the computer has sent enough data for the moment, it will pull ATN on again and send DEVICE-8-UNLISTEN. Many bursts of data may goto the file; eventually, the computer will close the file by sending (with ATN on, of course) DEVICE-8-LISTEN;SECONDARY-ADDRESS-2-CLOSE. ATN overrides everything in progress, A disk file might have lots of characters to give to the computer, but the computer wants only a little data. It accepts the characters it wants, then switches on ATN and commands the disk to Untalk. The disk has not sent EOI, but it will disconnect as commanded. Later, when it's asked to Talk again, it will send more characters.

ATN SEQUENCES

When ATN is pulled true, everybody stops what they are doing. The processor will quickly pull the Clock line true (it's going to send soon), so it may be hard to notice that all other devices release the Clock line. At the same time, the processor releases the Data line to false, but all other devices are

getting ready to listen and will each pull Data to true. They had better do this within one millisecond (1000 microseconds), since the processor is watching and may sound an alarm (“device not available”) if it doesn't see this take place. Under normal circumstances, transmission now takes place as previously described. The computer is sending commands rather than data, but the characters are exchanged with exactly the same timing and handshakes as before. All devices receive the commands, but only the specified device acts upon it. This results in a curious situation: you can send a command to a nonexistent device (try “OPEN 6,6”) - and the computer will not know that there is a problem, since it receives valid handshakes from the other devices. The computer will notice a problem when you try to send or receive data from the nonexistent device, since the unselected devices will have dropped off when ATN ceased, leaving you with nobody to talk to.

TURNAROUND

An unusual sequence takes place following ATN if the computer wishes the remote device to become a talker. This will usually take place only after a Talk command has been sent. Immediately after ATN is released, the selected device will be behaving like a listener. After all, it's been listening during the ATN cycle, and the computer has been a talker. At this instant, we have “wrong way” logic; the device is holding down the Data line, and the computer is holding the Clock line. We must turn this around. Here's the sequence: the computer quickly realizes what's going on, and pulls the Data line to true (it's already there), as well as releasing the Clock line to false. The device waits for this: when it sees the Clock line go true, it releases the Data line (which stays true anyway since the computer is now holding it down) and then pulls down the Clock line. We're now in our starting position, with the talker (that's the device) holding the Clock true, and the listener (the computer) holding the Data line true. The computer watches for this state; only when it has gone through the cycle correctly will it be ready to receive data. And data will be signalled, of course, with the usual sequence: the talker releases the Clock line to signal that it's ready to send. The logic sequences make sense. They are hard to watch with a voltmeter or oscilloscope since you can't tell which device is pulling the line down to true. The principles involved are very similar to those on the PET/CBM IEEE-488 bus - the same Talk and Listen commands go out, with secondary addresses and similar features. There are fewer “handshake” lines than on IEEE, and the speed is slower; but the principle is the same.

-End of document