An exploration of why USB (Universal Serial Bus) would be a good choice for inside the slot machine communications.

Updated: July 25, 99 8:25 PM   Author: Jim Morrow
Printed: October 26, 99 1:27 PM   ***Bally Gaming, Inc. • 6601 South Bermuda Road • Las Vegas, NV 89119-3605***

Executive Summary:

The Universal Serial Bus (USB) is the best choice for an in-the-game serial bus. Solutions exist for handling the sorts of messages that would occur within a slot machine; asynchronous short real-time messages, code download, voice data transport and video transmission. There is a formal mechanism (but not required) for GAMMA to approach the USB board and request a message "page" specific to the Gaming industry. We manufacturers could concentrate on creating message content that is specific to the Gaming business as opposed to designing protocols.

A summary of critical issues:

- Cost: Processors that include a USB interface costing less than $1.00 are available. For $3.50 you can get a full speed USB processor with good sized RAM and ROM.

- The amount of processing power needed to implement: For a *device* like a button deck or handle controller you can use the $1.00 processor. For the *host* you need more ummph, like a PowerPC or Pentium however options exist for doing *host* functions from an 8031.

- Operating system constraints. USB has been implemented on a large number of real time operating systems (RTOSs). There are consultants specializing in bringing USB functionality your product.

- Polled versus CSMA/CD (CAN). USB is polled at rates up to every 1mS. USB guarantees message times. If you need to know about something faster than every 1mS, put that intelligence in the *device* instead of the *host*. In CAN each device has to contend for the bus with collision and re-try.

- Bus mastership. USB has one master – the *host*, through which all communications occur. In USB the *host* makes sure all devices are valid and behaving themselves. USB runs at 12Mbps – very fast. With USB the Gaming regulators are going to appreciate having one central control over the network. CAN allows all devices to talk to all other devices. Do you want the coin acceptor to tell the hopper to pay out?

- Off-line communications. USB is a computer standard and PCs have the connection ready today. For field testing, production, diagnostics and configuration USB is the coming PC industry standard.

Some issues and answers from the GAMMA meeting minutes:

- Multi-drop – USB is a tiered star configuration. Hubs provide additional connections and control power to devices. This means mis-behaving devices can be individually shut down to protect the overall system. This may be a potential advantage in ESD and power loss situations. It also helps with the security issue since unknown and unwelcome devices can be rejected.

- Locking connector – USB connectors are keyed, locking, polarized and inexpensive. They are also keyed for upstream and downstream connections so you cannot make a wiring error.

- Provide minimum power to operate interface for simple devices – USB allows for hub powered as well as self-powered devices. Hub powered devices can draw up to 500mA. The host learns who draws what current and decides if devices can be powered on or not.

- Hot swappable / plug and play – USB is always hot swapping and plug and play.

- One device malfunction does not bring down the bus – USB does well here with the hub being able to shut down any device that mis-behaves or the host does not want any longer.

- Each member to report power requirements to committee – You don't need the committee to know about power requirements, in USB the device reports it's power needs to the host. The host then figures out total power demands and makes decisions accordingly. The host can even make decisions in weird cases like power down, brown out, tilt, and static zap.

- EMI susceptibility, emission of physical link – USB is made for the PC industry so it has to pass the worlds most stringent standards.

- Physicals stay in cabinet with length less than 5 meters use hubs or repeaters if longer – With USB the peripheral connections can be up to five meters in length. Repeaters are available.

- Bandwidth / service rate / quality of service some discussion of response times – USB supports 1.5Mbps, 12Mbps and there is a 240Mbps V2.0 standard (completely compatible) coming. USB includes a quality of service concept in how it handles asynchronous messages versus isochronous versus bulk messaging. Asynchronous messages are polled every 1mS guaranteed.

- Downloads possible during live operation – In USB the point of bulk transfers is to handle large files while the asynchronous (button push, coin in) and isochronous (video and audio feed) data are being dealt with. In addition, USB has a 'class' pre-defined for how to handle code download.

- Maximum / minimum nodes - USB can accommodate up to 127 devices. Devices can be combinations of functions – so that you could have a button and light deck in one device. There is no minimum – well, at least one or all this does not make sense …

A short discourse on USB versus CAN:

Here is a decent web site that has good links: http://www.nrtt.demon.co.uk/can.html
This is one of the links I found helpful: http://www.omegas.co.uk/CAN/

To sum up, it looks like CAN does not go as far up the OSI protocol model as does USB. CAN runs at 1Mbps and is CSMA/CD like Ethernet. There are no addresses. Rather, a message is given an 'identifier' (I suspect this is like ECHELON) and everyone gets to see all the identifiers. So the problem is in deciding which piece of info you want. Apparently there are schemes for identifying messages you are interested in but there is mention of this consuming processor bandwidth with lots of interrupts.

There are multiple versions of CAN available - an 11bit and 29 bit version. There are also at least five competing higher level protocols for application messaging. It is not clear to me at this time that they have considered how to handle data, voice and video over the CAN bus. Message lengths are limited to 8bits so you'd have to send lots of packets. It is also not clear that CAN would like handling large data transfers like code download or audio/video data streams. There would also have to be a "right from the start" agreement on what a message means and its identifier for a vendor to make a CAN device. This is versus USB where you could produce a proprietary solution, market it and later add it to the class and usage page that everyone could agree to.

It does not have hubs, goes a reasonable distance over twisted pair wire and is simple to wire. It looks like it makes a lot of sense for a factory floor where sensors and actuators are handling real time processes. My major concern would be whether it could handle large bandwidth needs such as audio, video and code download. Also, it appears to leave much of the message and protocol design to the user. With USB I believe we can take advantage of the classes and usage tables to focus our efforts on Gaming specific application messaging.

The main body:

The Universal Serial Bus (USB) would be the best choice for an in-the-game communications network (B link) for the following reasons:

1. USB frees Engineering talent to focus on rapid creation of new Game content, feature and function by providing a hardware and software communications medium that can be applied to data, voice and video information.
2. USB is an industry standard where all documentation is freely available, managed in a professional, public way where third party verification of USB compatibility is available. Plus there is a possibility of integrating Gaming specific messaging content into the USB standard.
3. An array of powerful development tools that **ARE** those available for desktop computing can be used to speed and simplify new product design. Tool vendors like Transdimensions are developing a USB development environment where writing device drivers on the host side and writing USB related codes on the device side will become as simple as in filling forms. Once you fill in the desired configuration and what you want the device to do in the form, compile it and you get yourself a driver.
4. The details of how to handle such things as code download, voice traffic and video traffic have been solved and documented and are there for the Game developer to use versus each Game Engineer inventing transports and protocol and messaging to handle each new design challenge. Yet there is still flexibility to allow a designer to create custom, proprietary messaging.
5. Computer industry support which provides lower cost hardware and standard class device software through our piggybacking on consumer volumes, a large pool of trained developers and PC industry components that could be used in new Games such as cameras, scanners, D-to-A's, speakers, motion control, etc.

6. USB has a high bandwidth today: 12Mbps and has prospects of an optional 240Mbps capability in the future.

Moving into use of USB requires study on the part of the Engineer. The most challenging aspect is getting over all the new terms and language used to describe USB. There is a lot of documentation because so much of the background, basic communication service has been provided for you. If you want to use a feature, you need to read to understand how to access it. USB is an evolutionary two steps up: RS-232 was the start with basic physical, connector, voltage and 8N1 definition. I2C goes a little further and defines addressing rules and message format. USB takes the third step to provide a framework for dealing with multiple sorts of information over the wire. It documents it in a way that simplifies the job of the end application programmer. That means the embedded guy (like me) who wants to know how the bits fly around must wade through more standards in order to make it simpler for the end programmer.

Just to deal with two myths right up front:

The host does not have to be a big PC running Windows.

There are implementations of USB host code that run on processors other than a Pentium. What really matters is whether your RTOS vendor has implemented the host side USB interface. Some have. There are companies that will supply you a board or chip set that implement all USB host functionality. You see an 8 bit parallel interface to their chips – they take care of all the host software for you. See Appendix A for references.

The USB hardware is not going to be expensive.

If mice and keyboards are USB, you know it has to be cheap to be a device. The processor can be as low as $1.50. The trick here is that the really cheap stuff uses 1.5Mbps rate. Cypress makes micros that run the full 12Mbps and cost in the $3.50 range so a device like a note acceptor or hopper can be done pretty cost effectively. The hard part is learning the USB-speak enough to design a device. You can bootstrap that learning effort by hiring one of the USB consulting firms that are listed on the www.usb.org website.

Ok, now for the details.

A quote from reference 1 on USB:

"USB is a 12 Mbps serial channel that is shared by a wide variety of PC peripherals, up to 127 peripherals total can be mounted to a single PC. It has four wires: two wires for supplying the power, and two wires for data transfer. It is a token-based bus protocol, where the USB host is the master of the bus. The host broadcasts tokens on the bus and a device that detects a match in its address, as described in the token, responds to the host. The 12 Mbps bandwidth of the bus is framed into 1ms slots and shared by all the devices in the bus in a time division multiplexed (TDM) manner. The host has knowledge of all the bus access requirements of its devices, and schedules all the bus activities."

USB was developed to serve three main goals:

1. Ease of use achieved by providing hot plug and play capability. Hardware, drivers and protocol have been designed to allow devices to plug in, identify themselves to the host and be configured without or with a minimum of end user intervention. Naturally this means we as designers have more work to do initially to make sure all the drivers and devices know what to say to one another. The advantage of this particular feature would be smoother production and field upgrades. USB requires each device to exchange information with the host about its manufacturer, type, version number, and capabilities and in general a whole list of useful information. The general form of this information is called out in "Class" specifications. GAMMA could provide a great service by using the existing USB Class definitions and calling out the specifics of each piece of information such that Gaming equipment can all communicate and understand one another.
2. If a USB device is detached from the host, the host will recognize this and notify application software. A very useful feature for preventing scamming of the Game.
3. USB v1 provides the ability to support real time data (eg voice, audio and compressed video) and asynchronous messaging (ie button pushes, coin in, bill in, etc) USB can bring multiple data, audio and video paths into and out of the host. It does this by dividing the data into 'pipes' and 'endpoints' that flow over the single pair of copper wires. You can have data, audio and video paths all running at once to a single device or multiple devices. There is work being conducted on a USB v2 specification that would extend the data rate up to 200 Mbps thus allowing real time, streaming video.
4. USB provides a low-cost host port extension and allows up to 127 device attachments to a single host. Two categories of USB devices exist: a 1.5 Mbps and 12 Mbps. The 1.5 Mbps devices are limited to asynchronous, very short sort of messaging applications but processors that implement it are cheap – in the $1.50 range.

Bus Topology:

The USB specifies two main elements of a USB system: the USB host and the USB devices. There is only one host in a USB system. The host may be implemented with a combination of hardware, firmware and software.
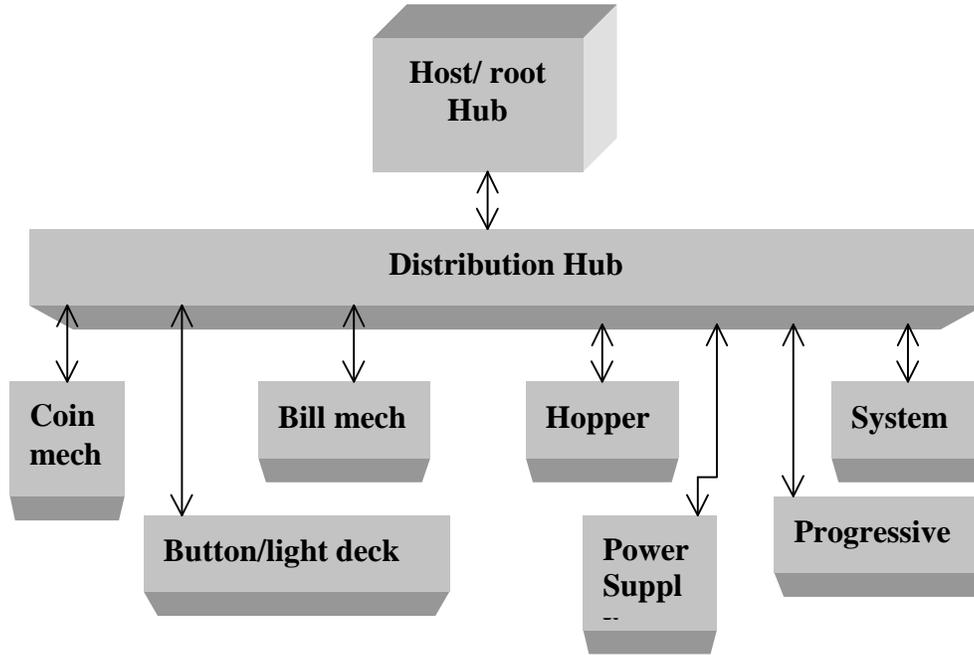
Fig 1. Physical connections for USB – tiered star topology.

While the physical connections of USB are through a tier, the logical data connections are direct. In other words, from a programmers point of view the connections in Figure 1 look like those in Figure 2.
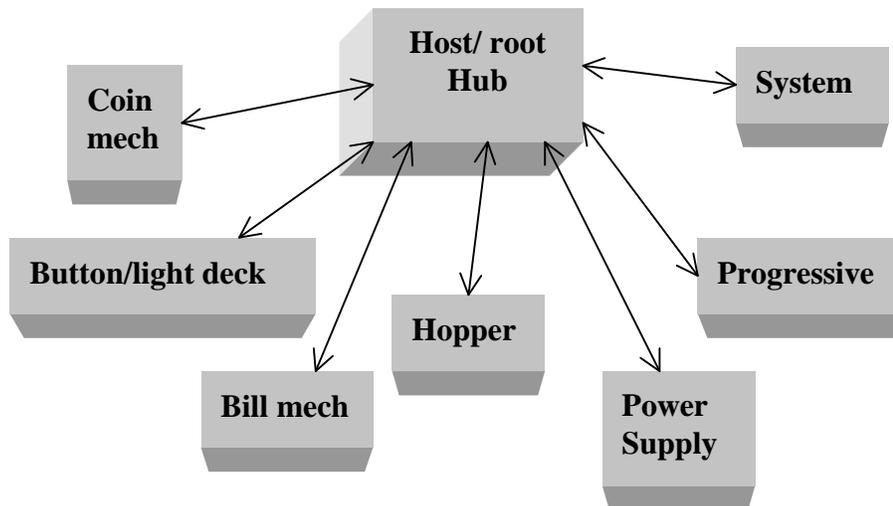
Figure 2. Logical connections for USB in a slot.

USB pipes and endpoints.

In order to transfer many different types of information (data, voice, video), USB employs the concept of pipes and endpoints as shown in Figure 3.
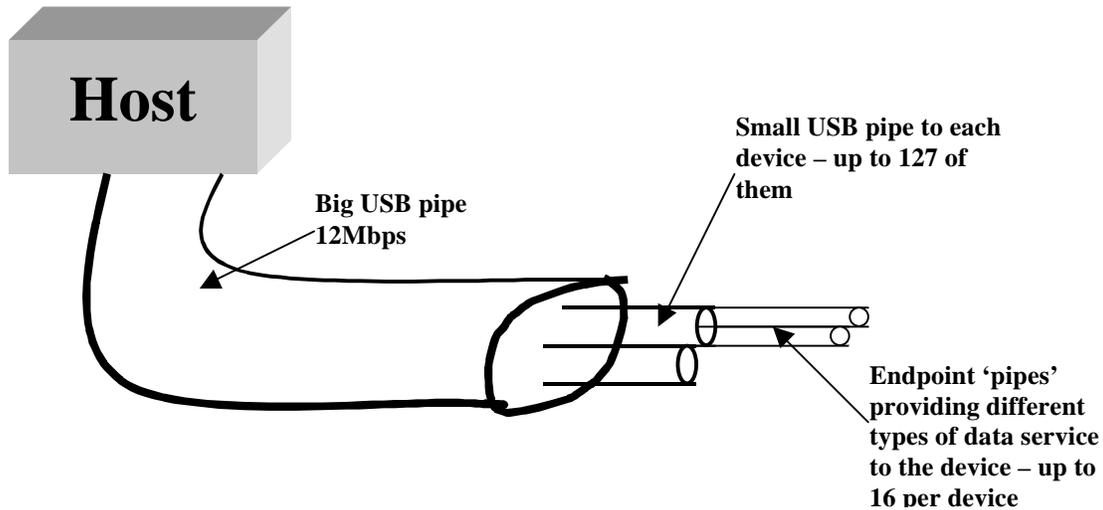

Fig. 3. USB pipes and endpoints from a conceptual point of view.

The copper wire supports up to 12Mbps total bandwidth of data flow. This 12Mbps is shared among all the devices connected to the USB. The host finds out (at connection time) what the bandwidth needs of all connected devices are then allocates the 12Mbps bandwidth to be sure every device receives the data bandwidth service they need. There can be 127 devices but in a typical slot machine there may be eighteen devices. Then each device can request several types of data service. For example, a button deck might have an interrupt endpoint 'pipe' for sending the asynchronous button push information. If the button deck had a speaker on it, it might also request an 'isochronous' endpoint pipe for transferring audio information.

Data service types for USB:

USB provides four forms of data service:
1. <u>Control.</u> Control data is bi-directional and supports configuration, common and status communication between the host and devices. All devices must support this form of data service. In addition to control information, it is possible to put some short, intelligent signaling data on the control endpoint. Some devices will only implement a control endpoint since they only need to send a small amount of data to and from the host.
2. <u>Isochronous.</u> Don't be intimidated by the name. This is simply data that MUST be delivered on time but can have errors in it. Typically it is pretty high bandwidth stuff like audio or video. The host will make sure that if you connect a camera up, given the 12Mbps bandwidth, plus all the other services (like interrupts for buttons, note acceptors, etc) there will be enough data bandwidth

to handle the camera. Of course, camera and audio data can tolerate some bad bits but it cannot tolerate late data. Thus this form of service puts a heavy demand on the 12Mbps total bandwidth.

3. <u>Interrupt.</u> This is the one Gaming devices would most likely use the most. It is uni-directional and inputs to the host from the device. Data transfers are small in size and happen relatively infrequently. This service is actually polled. The device tells the host (on power up) it wants to be checked for data every X milliseconds. X can be from 1mS to 255mS.

4. <u>Bulk transfer.</u> This is intended for sending large amounts of data accurately. This sort of service would be used if you wanted to download code to a note acceptor. Generally the delivery time of the transfer is not critical. It will keep re-trying until all the data gets there perfectly. (if there is error or the device times out for a packet, the retrying is "three strikes you are out".)

Cables, connectors and power:

USB employs connectors that are keyed, locking and polarized. An series A connector is used to connect devices downstream of the host. Series B connectors are used for the upstream connection to the host. There are four wires in the USB cable. Two for power and two for signal. At 12Mbps the two signal wires are a differential signal pair in a shield. In order to signal the speed of a device upon connection, a 1.5K resistor is tied from the D+ line to 3.0 to 3.6V. The two power lines are ground and 5V. Devices can be bus powered or self powered. A bus powered device can draw a maximum of 500mA from the bus. When a device starts up it can draw no more than 100mA from the bus until it gets permission to draw more.

The USB spec requires each cable to be maximum 5 meters. There can be up to 5 hubs in between for a total of 30 meters. There are companies make special cables that can allow distances up to 100 meters without using hubs.

A notification of the power a device draws has to be sent up to the host on startup. The host adds up all the power requirements and decides who can be powered up fully. It turns out all devices have to power up drawing under 100mA and stay that way until they get permission to pull up to 500mA. This is for bus powered devices. Self-powered devices are not so constrained. The power up and down control from the host may have great usefulness in the future.

USB has been designed to be concerned with RFI issues. Thus the differential signaling and shielding. USB should have not trouble passing IEC1000 and FCC part 15 RFI tests. It should also have excellent static zap immunity. The ability of the hub to shut down a malfunctioning device (like one that has gone wild after a static zap) should be of help in passing the regulators zap gun testing.

Packets, formats, etc:

I won't go into great detail here. It seems the USB line is quiet unless someone has something to say (except the Start of Frame SOF which are sent out from the host every millisecond). The host is sending out these token packets every 1mS to check on devices and do the interrupt thing. The host only sends device specific token packets when it has something to say to a device or when an interrupt

endpoint needs to be polled. This is normally not every mS. Only when an isochronous transfer is active, then the host needs to send one packet each frame to each active isochronous endpoint. There is a sync character so it would seem the USB receivers have a phase lock loop in them which recovers clock and data from the signal. This is basically a packet system with maximum data transmit times set by the need to re-sync every 1mS.

It does not seem that we as designers of Gaming related equipment need to be concerned with the electron level of signals on the wire. USB introduces the concept of "classes" in which we can overlay our application messages on top of a protocol and format that is an 'industry standard'. Why not use defined USB classes so we can focus on the Gaming related content that are the heart of our business?

Comments on Classes:

From: http://www.usb.org/developers/data/hid1_1.pdf the following comes …

"USB protocols can configure devices at startup or when they are plugged in at run time. These devices are broken into various device classes. Each device class defines the common behavior and protocols for devices that serve similar functions. Some examples of USB device classes are shown in the following table:

Device class	Example device
Display	Monitor
Communication	Modem
Audio	Speakers
Mass storage	Hard drive
Human interface	Data glove

"

"The HID class consists primarily of devices that are used by humans to control the operation of computer systems. Typical examples of HID class devices include:
- Keyboards and pointing devices – for example, standard mouse devices, trackballs, and joysticks.
- Front panel controls – for example, knobs switches buttons and sliders.
- Controls that might be found on such devices as telephones, VCR remote controls, games or simulations devices – for example, data gloves, throttles, steering wheels and rudder pedals.
- Device that may not require human interaction but provide data in a similar format to HID class devices – for example, bar code readers, thermometers, or voltmeters.

Many HID class devices include indicators, specialized displays, audio feedback and force or tactile feedback. Therefore the HID class definition includes support for various types of output directed to the end user."

" The primary and underlying goals of the HID class definition are to:

- Be as compact as possible to save device data space.
- Allow the software application to skip unknown information.
- Be extensible and robust.
- Support nesting and collections.
- Be self-describing to allow generic software applications.
"

The human interface device (HID) class looks like it meets the majority of the needs for an in-the-game communication link. The HID class only uses Control and Interrupt endpoints. See pg 10 of HID specification. This is not a limitation though because for a high speed device, the interrupt pipe can have a maximum packet size of 64 bytes. Note that the primary goal of the HID class was to make the devices as compact (inexpensive?) as possible. An important consideration for anyone designing devices such as coin acceptors, note acceptors, hoppers, reels, etc. The audio, communications and download class may also be useful for the B-link. It is possible to work with the USB technical team to define a set of application specific messages that are specific to an industry. For example, the power device industry has a set of messages that will be recognized for AC, DC and battery information. GAMMA could create a set of messages that are Gaming device related and fit them within the framework of USB and the HID class with the help of the USB technical advisors. More details on doing an industry specific set of messages can be found on the usb.org web site under developers and class definition.

Note above that the first, primary goal of the HID class was to keep the data space requirements on a device as compact as possible. It should be possible to create an effective USB HID class device for nearly the same price as the device costs today.

Appendix A:

An alternative to my explanation of USB can be found here. I found it very well written. More technical detail than I gave: http://www.cypress.com/design/techarticles/v3n1p4.html

Ref 1. Developing USB PC Peripherals by Wooi Ming Tan, Annabooks @1997
Ref 2. USB Explained by McDowell, Steven and Sayer, Martin, Prentice Hall @ 1999

USB web site http://www.usb.org/

Where the good stuff on USB is for developers http://www.usb.org/developers/index.html

Phillips at http://www-us.semiconductors.com/usb/products/interface/pdiusbd12/   makes a chip that interfaces USB to devices over a 8 bit bus or by DMA.

Cypress makes micros that have USB built in.

http://www.transdimension.com/embedded.htm makes chips sets and boards to perform the host functions of USB under the control of your micro.

Temic makes micros with USB on them.

Intel makes a big, expensive micro and an evaluation board for doing USB designs.

See www.usb.org web site for lots more manufacturers and equipment vendors.

End.