ADVANTECH
Enabling an Intelligent Planet

Technical White Paper

October 2012

Industrial Bus and Network Standards

Author: Advantech

E-mail: eainfo@advantech.com

Over the last fifty years, improved external communication links from PC-based applications like Human Machine Interface (HMI) software to controllers, and from controllers to I/O, have greatly increased productivity in manufacturing and other industrial applications. More recently, internal hardware buses at the PC and the controller level have also improved, boosting performance and cutting costs.

In the past, external communication links were very difficult to establish and maintain, primarily due to a lack of standards. Most links were serial in nature, with custom software drivers written to allow communication between or among devices.

The majority of networks were very simple, usually only one device to another, or one device to multiple devices via a basic daisy-chain network. Baud rates tended to be very slow, limiting traffic to information that didn't have to be communicated in real time.

With the advent of standard communication protocols, both at the hardware and software levels, communications began to improve dramatically. Today, many different types of standard networks are available, with users able to pick the preferred network based on application requirements.

Dramatic increases in performance have made it possible to use networks for real-time control, making controller-to-I/O communications via external communication links a viable and often preferred alternative to internal and to local bus-based I/O.

In a similar fashion, internal hardware buses at the PC and the controller level have progressed from proprietary to standard, allowing manufacturers to mix and match components from different vendors to build a high performance system.

As soon as programmable controllers were designed and in common use, engineers looked for ways to get data out of the controller, or to download data or programs to the controller, without either taking the controller off-line or having to make a trip to the plant floor.

Depending on what the controller was expected to do, automation professionals developed several simple ways to communicate with them. These were commonly used for short range or short duration communication, such as downloading programs, but as they grew in complexity, they became real factory bus architectures.

COMMUNICATIONS FROM PC-BASED APPLICATIONS TO CONTROLLERS

Communications were very simple in the beginning. This was in part because early PC-based applications were run in MS-DOS, which used its command line extensively, and batch programs to download or upload data between controller and PC were simple to write and effective.

Unfortunately, the early communications protocols had significant drawbacks when automation engineers tried to use them as real-time communications networks. Since they were intended

originally to be data in/data out communication protocols, they were often hard to use, with implementations needing to be hard-coded. This, of course, meant that if the application using the protocol to communicate was changed, or the controller model or firmware was changed, the protocol implementation would be “brittle” and break.

One of the very first applications was to get data from PLCs and insert it into a spreadsheet. From early spreadsheet programs like VisiCalc to today’s Microsoft Excel, spreadsheets have been employed for data logging applications, using one or more of the simple external communications buses to communicate between the PLC and the PC.

The biggest drawback to all of the early communications buses was that they concentrated on the hardware specification to the extent that interoperability and interchangeability were forgotten, which could produce significant cost issues when trying to hook together devices and controllers from different manufacturers.

SERIAL COMMUNICATION VIA RS-232, RS-422 & RS-485

RS-232 predates controllers and microcomputers as it goes back to the days of teletypewriters and dumb terminals. It has been in use since 1962, and although it is not commonly used in PC interfaces (having been replaced almost completely by USB), automation professionals continue to see RS-232 and its multi-drop sister standards RS-422 and RS-485 fairly often in the instrumentation and laboratory device fields. This is likely to continue for several more years, even though most modern PCs no longer have an RS-232 port for communication to peripherals such as PLCs and modems, so professionals needing RS-232 connectivity must use USB to RS-232 converters.

The most important advance of the RS-232 standard was its reliance on serial binary data transmission. However, the limitations of the standard made it difficult to implement in a world where controllers and PCs used increasingly small power supplies. Moreover, the multi-drop specification, often called RS-485, was poorly implemented and required the use of workarounds.

RS-232 and RS-422 protocols are full duplex capable and easily capable of bidirectional data transfer over relatively short distances. Today, these protocols are often used for communication to laboratory or online analyzers, or to connect a stand-alone safety instrumented system to a distributed control system.

RS-422 uses twisted-pair differential signals, allowing it to traverse long distances like RS-485, but it is peer-to-peer. Although it can support up to 10 slaves, it only supports one master, and is a true four-wire system.

RS-485 is not full duplex, but it can be implemented as such by using a pair of RS-485 circuits. In this case, RS-485 must be used peer-to-peer, and not multi-drop.

Although RS-485 was the underpinning for many early proprietary PLC communication standards such as Rockwell Automation/Allen-Bradley's Data Highway and Siemens' Profibus, neither RS-232 nor RS-485 were designed with cyber security in mind because they were both designed before Internet use became prevalent, when intrusion from outside the network wasn't a consideration.

THE GPIB (IEEE-488) STANDARD EMERGES

Originally developed by Hewlett-Packard for its test and measurement products and laboratory equipment, IEEE-488 was originally known as HPIB for "Hewlett Packard Interface Bus." HPIB was a parallel bus and could even be implemented without a microprocessor, using TTL logic.

HPIB became a de-facto standard, because it could interconnect up to 15 devices, each with its own address. As other vendors started using the standard, it became known as GPIB (or "General Purpose Interface Bus") and then as the "recommended standard" IEEE-488. Eventually, the IEEE and the IEC formulated it into a dual standard, IEEE/IEC-488.1 and .2. Unfortunately, while the bus protocol defined the hardware, it didn't define common commands. Therefore, devices that were supposed to be linkable by IEEE-488 were often found to be non-interoperable.

The GPIB bus is an 8-bit, master/slave bus with a maximum data rate of between 1 Mbps and 8 Mbps, depending on the implementation. This is now considered a rather slow data transfer rate.

Because test, measurement and laboratory equipment have long lifecycles, GPIB is still in very common use.

MODBUS BECOMES A PLC COMMUNICATION STANDARD

Published by Modicon in the late 1970s, Modbus (for "Modicon Bus") was developed specifically for the industrial controller environment. At the time, Modicon (now part of Schneider Electric) had a very large market share in PLCs and industrial controllers, and needed a robust method for transmitting and receiving data to and from PLCs.

Modbus has become the single most successful industrial bus protocol in history. There are some basic reasons for this. First, it is simple and robust. Second, it is royalty free and administered by a third-party standards organization, the Modbus Organization. Third, it is easy to maintain and use, and it is also capable of moving data without placing too many restrictions on vendors.

Modbus now comes in several flavors. Modbus RTU is used to connect SCADA systems to RTUs (Supervisory Control and Data Acquisition, and Remote Terminal Units) in such fields as water and wastewater, electrical utilities, alternate energy and oil and gas pipelines. Modbus TCP/IP is used to connect over TCP/IP networks using port 502. It has become quite common as TCP/IP network protocols have proliferated. Modbus has even been used in mesh network applications and easily is transmitted over Ethernet networks.

ETHERNET TAKES ALL

The ubiquity of Ethernet in the home and enterprise computing space has made it the de facto networking technology for the 21st century. Ethernet is essentially transparent to the data traversing the network, and it can be used to move extremely high volumes of data at very high data transfer rates, up to Gigabit speeds. Ethernet switches and routers are supplied with management firmware, and with cyber-security provisions that other, older protocols lack.

Modern controllers and PCs are always furnished with Ethernet ports, and the implementation of the standard is designed to be plug-and-play.

For these reasons, it is likely that Ethernet will become the sole standard for data transmission in the next decade. Already, serial protocols such as Modbus, DeviceNet, Profibus and HART are designed to run over Ethernet. For very high-speed machine and robot control applications, specialized Ethernet-based networks such as EtherCAT and Powerlink are widely used.

With the advent of IPv6, and the vast increase in permitted IP addresses, it will be possible to run direct Ethernet to every device in the world several times over.

EXTERNAL I/O: WHERE DOES IT MAKE SENSE?

Modern external I/O and other devices such as field instruments are typically connected to controllers via some type of digital data network. By contrast, internal I/O is defined as I/O connected directly to a controller through an internal bus structure.

It is often not well understood where the cost of connecting a controller to I/O and other components actually lies. The majority of the cost is the actual wiring from the I/O to the controller and from the controller to the balance of the automation and information system.

For example, in a hazardous area, where special explosion proof wiring, conduit and junction boxes are required, installed wiring may cost as much as $3,000 per foot. In many instances, both in discrete part manufacturing and process plants, space is limited, and the use of cable concentrators and marshaling cabinets are required.

This means that the lifespan of the I/O is expected to be significantly longer than the controller to which the I/O connects, because replacing the I/O often requires the very expensive replacement of field wiring.

Many vendors make third-party I/O systems that connect to the major controllers because there are significant advantages in different configurations and sizes among I/O vendors. Using third-party I/O systems not only provides a wider array of features, but also drives down costs as controller vendors can't force users into proprietary and often expensive I/O architectures.

When I/O is external to the controller, it makes configuration and reconfiguration simpler and easier as compared to bus-based I/O. Most manufacturers of controllers—whether DCS, PLC or PAC type—can use external I/O mounted on DIN rails.

Retrofitting external I/O for new applications is also straightforward. For example, high-speed machine operation is facilitated by the ability to reconfigure the existing external I/O or replace it with compatible I/O, without having to replace all the wiring and the controller itself. This is often feasible because most external I/O uses the same communications network, namely Ethernet.

External I/O makes troubleshooting and maintenance easier, too, because the controller's case doesn't have to be breached in an environment where dirt, chemical vapor or other non-optimal contaminants could easily enter and damage the controller's circuitry. If a set of inputs fails, it is simpler to replace it when it is external to the controller, even if it is connected to the controller over a backplane bus.

Using external I/O also fosters better-designed software and device driver standardization. This is critical to avoid situations when a simple upgrade to a device driver may cause a failure in communications between the controller and the I/O.

Last, but often most important, it makes it very simple to mix and match I/O from different suppliers, and even from different generations of controllers.

HIGH-SPEED I/O PROLIFERATES

RS-232, IEEE-488, Modbus, Profibus, HART and the other industrial field buses have always been throughput limited. That is, their data transfer rates have been very low. When you consider RS-232 I/O, operating at 1 Mbps, compared to Gigabit or 10-Gigabit Ethernet, it is clear that Ethernet is the protocol for very high speed, high bandwidth applications like video and mesh networks.

The question is how fast is fast, and how fast does I/O need to go? The answer is somewhat dependent on the application. As Gigabit Ethernet has shown, fast can be very fast indeed. There is, however, a practical limit to the speed of I/O. That is the point at which the external I/O meets the internal data bus inside the PC or controller. Commonly, speed mismatches exist at this point, and the practical overall speed of the network is defined by the speed of the internal data bus. This has been an issue since the first microprocessor-based controllers were developed.

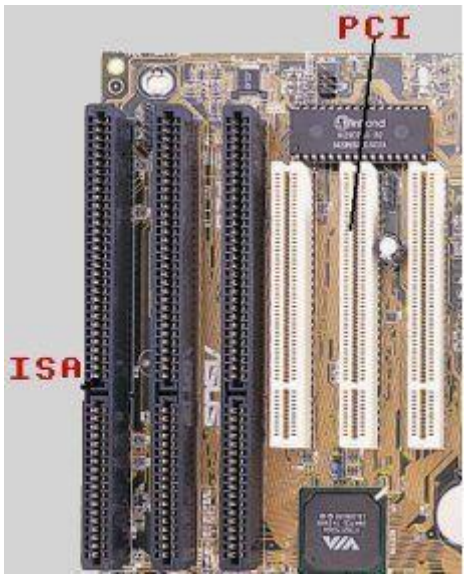
INTERNAL PC AND CONTROLLER INTERFACES

In computer architecture, the CPU and the main memory are closely interconnected, sometimes on the same chip, in a configuration known as cache memory. This is because the performance of CPUs has greatly outstripped the data transfer capability of most memory.

As soon as data moves out of the CPU toward peripheral devices, the speed of data transfer slows. Since the early 1980s there have been continuing improvements in these internal computer bus architectures, each improvement driving toward more speed for data transfer and more compatibility with peripherals.

The first microcomputers, such as the Altair 8800, required the ability to interconnect the CPU and main memory with peripheral devices including floppy disk drives and serial ports. The S100 bus structure, introduced in the mid-1970s, consisted simply of all the pins on the Intel 8080 chip run out to a backplane. S100 was the standard bus architecture for CP/M (Control Program for Microprocessors, the predecessor of PC-DOS and MS-DOS operating systems).

The first modern internal bus structure was ISA, standing for Industry Standard Architecture, which was introduced with the original IBM PC in 1981 and designed to operate with 8-bit Intel 8088 microprocessors. It was later modified to work with 16-bit data for “AT” computers running Intel 80286 microprocessors, and further extended as EISA (Extended Industry Standard Architecture) to 32-bit processors.



ISA and EISA were designed to work as “bus extenders” for the internal local bus architecture of the microprocessor. Expansion cards could be installed to allow communications with devices such as hard disk drives, printers and modems. The problem was that very few peripheral devices were “plug-and-play”, so jumpers and special device drivers were often required. Changes in system architecture and drivers often made peripheral devices not work well, or not work at all.

Because Apple did not use the Intel chipset, the Apple II, Apple III, Apple Lisa and the Apple Macintosh could not use S100 or ISA bus structures. They produced their own bus structure, NuBus, and later SCSI or “scuzzy” bus. These buses had the advantage of being significantly more plug and play than IBM PC or PC-clone computers, leading to the commonly held belief that Apple computers were easier to use than PCs.

In 1986, Western Digital, Control Data Corporation and Compaq Computer Corporation produced the first modern disk drive interface, IDE or “Integrated Drive Electronics” specifically to improve the performance of disk drives in personal computers. The advance was to embed the drive controller and the interface on the same card, connected to the motherboard. Later, this interface became known as ATA from the PC-AT. The SCSI bus, originally developed for Macintosh, was primarily used as a CD-ROM controller bus.

These buses all shared similar advantages and disadvantages. Advantages were standardization to some extent and increased performance as compared to predecessors.

Disadvantages were still insufficient performance for many industrial applications, and a lack of a true plug-and-play functionality.

PCI CHANGES EVERYTHING

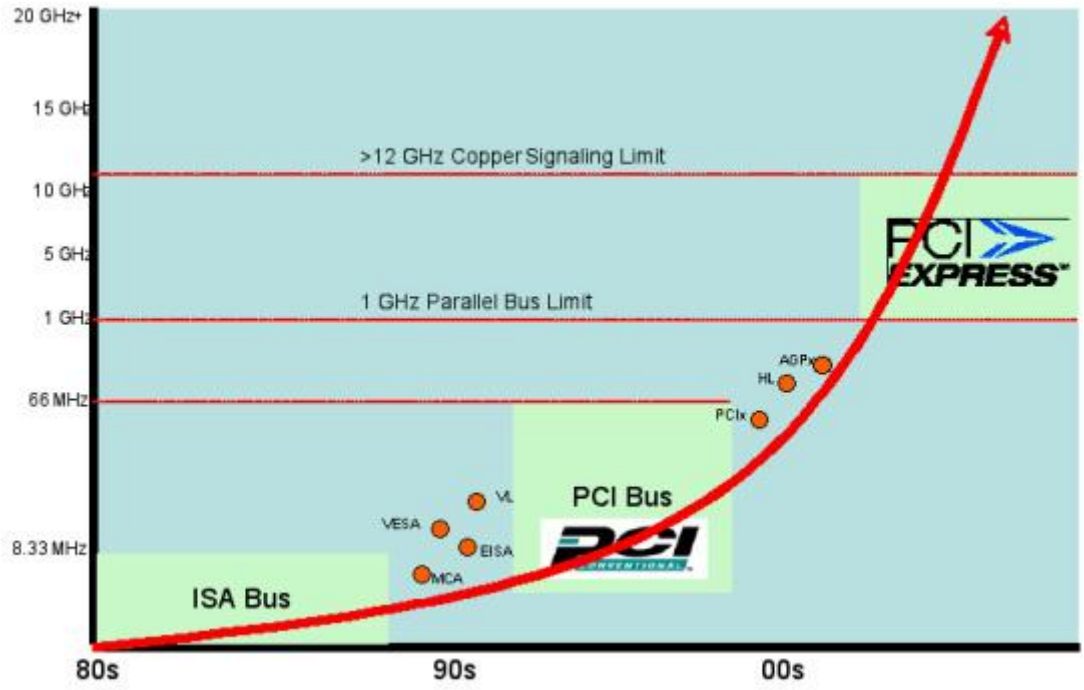
In the early 1990s, the Peripheral Component Interconnect (PCI) interface was introduced as a way to interconnect the chips on the motherboard and as a replacement for the problematic ISA bus. It operated at 33 MHz, a large speed advantage over the ISA and ATA buses commonly in use at the time, and the 33 MHz clock rate supported the speed of most existing peripherals.

As time passed, however, peripheral performance improved to the point that typical processor speed has increased from 33 MHz to 3GHz or more. In addition, the emergence of Gigabit Ethernet and other networking protocols often monopolized the PCI bus bandwidth for a single peripheral device.

When it was introduced, however, the PCI bus had significant advantages over the ISA and ATA buses including processor independence, buffered isolation, and plug-and-play operation. In short, the PCI bus fundamentally unified the internal structure of the PC.

PCI was not perfect, though. As time went on, its limited bandwidth became an issue, especially since the bandwidth of the bus was shared between all devices. PCI was designed without power management, which became an issue with newer, lower power devices, and it was not very scalable. It had very stringent data routing rules, and worst of all, there was no support at all for real-time data transfer.

To address these and other issues, the next evolution of bus technology was to improve the PCI bus into the PCI Express (PCIe) bus.



The PCIe bus is a serial interconnect bus that running at 2.5 Gbps, with packetization of the PCI data transaction, and then serializing for distribution on the bus. Bandwidth is available per slot and in both directions, and performance can be from 200 Mbps to 3.2 Gbps in either direction, input or output.

The evolution to PCIe made possible tremendous improvements to the limited bandwidth of the original PCI bus, while preserving backward compatibility.

The layered PCIe hardware can deliver thirty times the performance of the original PCI interface. This makes possible next generation I/O for use with high-bandwidth, high-performance applications like high-speed data acquisition and industrial multimedia applications. Recent evolutions of the PCIe interface even support virtualization of hardware I/O. Almost all modern graphics cards use the PCIe interface.

Bringing the PCIe interface outside the internal computer bus created the ePCIe or external PCIe bus, sometimes known as the cabled PCIe bus.

This development makes possible the extension of the internal bus speed of PCIe to external I/O. This, in turn, produces important increases in performance for very high speed I/O in machine, motion control and machine vision applications.

A new variant of PCIe, backed primarily by Apple, is Thunderbolt, which combines the PCIe and DisplayPort protocols for vastly improved graphic display and HMI performance. While Apple was the original implementer of Thunderbolt, other computer companies are beginning to also offer the protocol.

UBIQUITOUS USB

No discussion of PC bus architectures can be complete without a discussion of USB (Universal System Bus). USB has essentially replaced all serial and parallel interconnection schemes in PCs and laptops. USB has high throughput (USB 3.0 easily handles full motion video) and is plug-and-play with nearly every device currently available.

USB can be used as an internal bus structure, or it can be remote using a hub arrangement. USB is also capable of powering devices such as remote hard drives, CD or DVD drives and other peripherals. USB can also be provided as data and bus translators from GPIB, RS-232 and Modbus peripherals.

USB has the performance and reliability to support real-time industrial control applications, for example by providing connectivity between a PC-based controller and I/O.

THE ETHERNET BUS?

As ubiquitous as Ethernet is as a networking technology, its use as a computer bus has been extremely limited due to speed limitations, but primarily because there has been a dearth of IP addresses under the IPv4 Schema. In fact, by 2010, there were no available IP addresses left under the v4 Schema.

Some workarounds have been made, but in general the Ethernet world is migrating to IPv6, which provides a huge increase in IP addresses—enough to give every Ethernet-enabled device currently installed in the world more than one IP address. This makes internal IP addressing in devices possible, even practical. The rapid increases in Ethernet speed, for example Gigabit and Terabit Ethernet, approach or exceed the internal speed of the PCI and PCIe interfaces.

THE NETWORKS AND BUSES OF THE FUTURE

Currently, PCIe is the bus of choice, and Ethernet is the preferred network. With the migration of Ethernet TCP/IP to IPv6, Ethernet appears to be poised to become the dominant, and perhaps the only, networking technology. It's simple, manageable, has built-in security, and is clearly understood by a very large number of networking professionals both in industrial and commercial environments.

The question remains, however, whether there will be a new bus or network structure in the future. Based on the speed of advances in the technology, we can assume there will be, but because of the robustness and ubiquity of PCIe and Ethernet, any advance in computer or network bus technology will need to incorporate a significant capability for backward compatibility.

Which network or bus an automation professional selects depends, as always, on the application. For now, PCIe, ePCIe, USB and Ethernet seem to be able to solve most data communications issues for industrial automation.